Fall 12-1-2018

Towards Scalable Characterization of Noisy, Intermediate-Scale Quantum Information Processors

Travis Luke Scholten

Travis Luke Scholten
Candidate

Physics and Astronomy
Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Robin Blume-Kohout , Chairperson

Carlton M. Caves

Francisco Elohim Becerra

Gabriel Huerta

Towards Scalable Characterization of Noisy, Intermediate-Scale Quantum Information Processors

by

Travis Luke Scholten

B.S., Physics, California Institute of Technology, 2012
M.S., Physics, University of New Mexico, 2015

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Physics

The University of New Mexico

Albuquerque, New Mexico

December, 2018

Dedication

To those who seek Truth:

"For the gate is narrow and the way is hard, that leads to life, and those who find it are few." Matthew 7:14, RSV-CE

Acknowledgments

To my parents Stephen and Joyce, for supporting me throughout the years. From the plains of South Dakota, to the coasts of California, to the deserts of New Mexico, you have always encouraged me to pursue my calling.

To Dr. Carlton M. Caves, for acting as a sounding board for my ideas, both good and bad. Thanks for listening in a cheerful manner, offering useful feedback, and encouraging me to think carefully.

To my good friend Dr. Jonathan A. Gross, for a willingness to discuss all manner of topics, scientific and otherwise. Thank you for humoring me in our discussions, challenging my assumptions, and reminding me to seek the greater good.

To my advisor, Dr. Robin Blume-Kohout, for taking a chance on the graduate student who contacted him 5 years ago. Our journey together has experienced its fair share of twists and turns ... but the experience and views have been worth it.

Towards Scalable Characterization of Noisy, Intermediate-Scale Quantum Information Processors

by

Travis Luke Scholten

B.S., Physics, California Institute of Technology, 2012
M.S., Physics, University of New Mexico, 2015
Ph.D. Physics, University of New Mexico, 2018

Abstract

In recent years, quantum information processors (QIPs) have grown from one or two qubits to tens of qubits. As a result, characterizing QIPs – measuring how well they work, and how they fail – has become much more challenging. The obstacles to characterizing *today's* QIPs will grow even more difficult as QIPs grow from tens of qubits to hundreds, and enter what has been called the "noisy, intermediate-scale quantum" (NISQ) era. This thesis develops methods based on advanced statistics and machine learning algorithms to address the difficulties of "quantum characterization, validation, and verification" (QCVV) of NISQ processors. In the first part of this thesis, I use *statistical model selection* to develop techniques for choosing between several models for a QIPs behavior. In the second part, I deploy *machine learning algorithms* to develop a new QCVV technique and to do experiment design. These investigations help lay a foundation for extending QCVV to characterize the next generation of NISQ processors.

Contents

List of Figures

List of Tables

Publication information:

The entirety of Chapter 3, plus Section 4.2, was published as (267). Section 4.1 is an unpublished piece of research. The material in Chapter 5 is under review for submission to the arXiv, and the content of Chapter 6 will commence that process soon.

Chapter 1

Introduction to quantum computing

Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy. - Richard Feynman, 1982 (101)

1.1 Why study quantum computing?

Quantum mechanics is one of the most successful theories for describing the physical world. Since its development in the early 1900s, the theory has been used to predict a variety of physical phenomena, and its predictions closely match experimental results. *Applications* of quantum mechanics have been found in areas as diverse as semiconductor electronics (173), lasers (122; 279), molecular dynamics (217; 178; 315), and medical imaging (32; 243). Over the past thirty-some years, physicists and other researchers have realized that quantum mechanics enables an entirely new computing paradigm, dubbed *quantum computing*, and have explored its implications for computer science, information theory, and physics. Surprisingly, these implications

raise deep questions about the nature of the Universe, its structure, and the ultimate limits on computation.

At its core, quantum computing addresses the limits to and possibilities of a computer whose fundamental physical mode of operation is quantum-mechanical in nature[1]. Phrased another way, the operation of a classical computer is well-understood using a small amount of quantum theory, but it does not require other quantum phenomena such as *superposition* or *entanglement*. In contrast, quantum computers harness such phenomena, and put them to work for computing.

What's more, the underlying logic of the computation is quantum-mechanical in nature. Classical computation uses classical bits, which can only take the discrete values 0 and 1. In contrast, quantum computers store and process information in quantum bits (*qubits*), or their higher-dimensional relatives, called *qudits*[2]. Because the unit of information is different, the logic of quantum computers is necessarily distinct from that of classical computers. Quantum bits can be encoded in a wide variety of physical systems such as trapped ions, neutral atoms, superconducting circuits, or photons. In what follows in this chapter, I'll use the phrase "quantum processor" to denote some quantum system that is being used for quantum computation. Through precision control of a quantum processor, quantum information can be encoded, reliably stored, and manipulated. Given the unique nature of quantum theory, and how different it is from classical theory, quantum computing offers a radically different paradigm for computation.

With some reflection, the idea that physical law *should* affect what computers can do doesn't seem too far-fetched: after all, computers are physical devices in the world, subject to those laws. Several examples illustrate this fact:

[1]Note that here I am ignoring the fact that some quantum effects show up in semiconductor physics.

[2]See Chapter 2 for details.

- Special relativity places an absolute speed limit on the rate at which information can be transmitted between two points (namely, the speed of light), so no computer can send/transmit information between its processing units faster than that.
- Information transfer requires energy, limiting the rate at which information can be transmitted (24).
- A finite-volume region of spacetime can contain only so much information, placing limits on the information storage capacity of a computer (25).
- Information erasure always comes with an entropy cost (186) – a non-zero amount of energy is always required to erase a bit – giving a lower bound on the energy consumption of an irreversible computer.
- Quantum information can decohere ("leak out") from a quantum system, which leads to the emergence of classical behavior (34). Therefore, any quantum processor must address the problem of decoherence in order to harness quantum phenomena for computation.

The reasons just given show how physical law places *limits* on computation. However, different physical models of a computer can lead to new insights in computer science, and *expand* the conception of what a computer is, and what it can do. For example, reversible computing was introduced in the early 1970s as a way to mitigate the energy cost associated with information erasure (26). However, a good understanding of the relationship between reversible *computing* and reversible *processes* in physics took approximately another decade to develop. Fredkin and Toffoli considered the "billiard ball computer", and showed such a computer could operate according to a "conservative logic" necessary for reversible computation (111). (Of course, a reversible computer won't be built out of billiard balls any time soon, but the concept introduced by Fredkin and Toffoli helps build intuition.) Later, a

more useful model was constructed using Josephson junctions (193). Within the past decade, researchers have made strides in probing the physics of reversible computing (29). In this way, toy models of computers help us explore the connections between computation and physics. What's more, these toy models can lead to new insights about how to build computers.

Additional reflection suggests that computer science should have something to say about physical law. For instance, *computability theory* studies the (abstract) computational power of various models for ideal computers, such as classical Turing machines (300). Without making reference to any physical instantiation of a computer, computability theory tells us that some problems such as the HALTING problem are *undecidable*: no (abstract) computer can solve them. Under the assumption that Turing machines are a faithful model of what computation *is* and how it *physically takes place*, then computability theory would imply that no physical computer could ever be built to solve undecidable problems[3]. In turn, physical law had better prohibit such a machine from being built!

This interplay of physics and computer science is a non-trivial one, and raises deep questions about the nature of the Universe. For instance, is an appropriate underlying theory for describing reality fundamentally informational or computational in nature? If so, what kind of "program" is the Universe executing (196)? This line of questioning isn't unique in the history of science – consider the the "clockwork model" of the Universe developed during the Enlightenment (88) – but quantum mechanics provides arguably the most powerful physical theory developed to date that could be used for answering such questions.

In fact, quantum computing has some surprising implications for computational complexity theory. A recent result (1) shows that by taking the standard model for what

[3]These and related questions relate to the *Church-Turing thesis* and variants thereof (83).

a quantum computer is (i.e., the rules that describe its operation), and changing it in two seemingly banal ways, one ends up with a model for computation that can efficiently solve any decision problem whatsoever! Such a result clearly indicates that quantum computers are worth understanding. What's more, it suggests that (standard) quantum computing is right on an edge between trivial and non-trivial models of computation, which makes it an interesting topic of study. By exploring the implications of quantum mechanics on computability theory, we gain a better understanding of how physics and computer science relate to one another.

Computational complexity arguments aside, there are other reasons to study quantum computing. The opening quote of this chapter is taken from a talk Feynman gave in 1982. Entitled "Simulating Physics with Computers", Feynman considered the problem of using a classical computer (e.g. a deterministic or probabilistic Turing machine) to simulate a quantum-mechanical system. The main difficulty is that quantum systems exhibit correlations between subsystems that cannot be explained using just those subsystems alone. This property (quantum entanglement) makes simulating a quantum system using a classical computer difficult. The computer has to keep a record of the state of the system at each step in the simulation, and because the correlations are across the *global* state of the system and not within smaller, localized subsystems, the computer has to keep track of the entire state. This places storage requirements on the computer carrying out the simulation.

Why might a quantum computer help? Because its mode of operation is quantum-mechanical in nature, if one assumes entanglement can be generated by the system (a non-trivial task), then quantum computers seem to come with entanglement "for free" as part of their operation. They won't have to store a *record* of the state; instead, they *directly generate and manipulate it*. Hence, a reasonable conjecture would be that quantum computers could simulate quantum systems more efficiently or more accurately than classical computers. Understanding the complex behavior

of quantum systems could enable breakthrough developments in the technologies of the 21st century, including materials design (batteries, solar cells), medicine (pharmaceuticals, protein folding), and agriculture (nitrogen fixation).

Earlier, I alluded to the idea that quantum computing can shed some light on the fundamental way the Universe operates. The fact that quantum mechanics – at its core a theory about probabilities and information – is so good at describing the world suggests that a more advanced version of physical theory should be grounded in information theory, a notion that goes by the catchphrase "it from bit" (107; 320). Some research in this direction has explored how quantum computing interfaces with general relativity (196; 236; 142). The deeper connections between quantum mechanics and general relativity are still being developed.

1.2 A brief overview of quantum computing history c. 1980 - c. 2010

Feynman's talk in 1982 helped catalyze interest in quantum computing. Some pioneering results included the definition of a model of quantum computing based on a "quantum Turing machine" (83) that generalizes the classical Turing machine, the development of an equivalent "circuit model" that defines quantum computation as a generalization of classical digital circuits and logic (329), and the discovery of a simple problem ("Simon's problem") for which a quantum computer has a provable exponential speedup over a classical computer (280).

Arguably, the first "killer app" for a quantum computer was developed by Peter Shor, who showed in 1994 that quantum computers could solve the FACTORING problem – "Factor an integer N into products of primes." – in time $\mathcal{O}(\log^3(N))$ (278). This problem is generally believed to be hard for classical computers, as the best-known algorithm (the general number field sieve) runs in time approximately

Exp $\left[\mathcal{O}(\log^{1/3}(N)(\log\log N)^{2/3})\right]$. Given that commonly-used encryption algorithms (such as RSA) rely on the hardness of factoring for their security, Shor's result stimulated much interest in quantum algorithms for cryptanalysis, including the development of "post-quantum" cryptosystems that *are* resistant to attack by quantum computers (27).

The second is a result due to Lov Grover, who in 1996 showed that quantum computers, when given oracle access to an unstructured database of N items, could search over that database in time $\mathcal{O}(\sqrt{N})$ (127). Classically, the worst-case runtime is $\mathcal{O}(N)$, so while the speedup is less dramatic, it's nontrivial and can be applied to other oracle problems. Since Shor and Grover discovered the algorithms that bear their names, others have been created for problems including constraint satisfaction (7; 206), computing the gradient of a function (162), and solving linear systems of equations (139), among other problems (218; 161).

Although there are numerous algorithms that could be run on a quantum computer, the intervening years between Shor's discovery and the present day have highlighted the challenges to developing a quantum computer that is *universal* (capable of executing arbitrary computations) and *fault-tolerant* (can compute for an arbitrarily long time). Classically, these problems are well-understood and have been addressed. For example, the NAND gate is a universal logic gate (any logical operation can be expressed using it), and fault-tolerant encoding protocols are known for storing and transmitting classical bits. These problems are more subtle for quantum computers, however.

The issue of universality for quantum computers was addressed by results such as the Solovay-Kitaev theorem, (78) which showed that any computation could be decomposed (*compiled*) into a sequence of primitive operations (*gates*). The subject of "quantum compiling" remains an active area of research, particularly with respect to compiling algorithms on near-term hardware (143; 169; 138).

Making quantum computers fault-tolerant has proven to be quite difficult. By definition, a fault-tolerant computer is able to compute even in the presence of errors and corruption of information. Classically, fault-tolerance is relatively easy, because classical bits can be very stable. In contrast, quantum bits are *fragile*, and have to protected from the surrounding environment. The quantum information contained within those bits *decoheres* – "leaks out" into the environment (34) – so reducing decoherence is vital. At the same time, controlling quantum information requires the ability to strongly couple to a quantum processor and manipulate it. This duality – quantum bits need to be well-isolated from the environment, while also accessible for controlled manipulation of the quantum information they contain – has been called the "Tao of quantum computing" (84), and shows the fundamental difficulty in building a quantum computer.

Fault-tolerant quantum error correction (FTQEC) is crucial for achieving fault-tolerant quantum computation. A quantum computer using FTQEC can, in principle, compute for an arbitrarily long time, provided the error rates of the individual components of the computer are below a suitable threshold (238; 85; 4). (Note that a given quantum error correction technique may not be fault tolerant in the sense just described. For example, it may not be robust against faulty gates or measurement procedures.) Just as classical error correction encodes information to safeguard its integrity as it is transmitted over a noisy channel, quantum error correction encodes quantum information and protects it while a quantum computation is executed.

However, there are three ways in which QEC is unique and cannot be analogized to classical error correction. First, arbitrary quantum information cannot be cloned (326). Thus, a QEC technique cannot rely on the ability to blithely copy quantum information from one set of qubits to another. Second, information stored in qubits is disturbed when those qubits are measured. Therefore, a QEC technique cannot avoid the no-cloning constraint by a protocol of the form "measure the qubits and then copy

their information". Third, qubits experience a richer set of errors than their classical counterparts. There are only two kinds of errors that can affect classical bits: flips and erasure. Qubits on the other hand can be flipped, erased, dephased, and damped, among other errors; QEC has to protect quantum information against all of them (85). Numerous QEC techniques have been proposed for doing so (110; 277; 121).

Protecting quantum information comes with a steep cost. Even using sophisticated quantum error correcting codes (such as surface/color codes (110; 109; 185)), the number of physical qubits required to encode some number of fault-tolerant ("logical") qubits sufficient to address some near-term problems is currently estimated to be in the tens of millions to billions (250; 288). Given that up until approximately 5 years ago the "state-of-the-art" in quantum hardware was devices with 1 or 2 qubits, the road ahead to universal, fault-tolerant quantum computing looked bleak. Within the past 5 years, however, several developments have helped push quantum computing forward. While fault-tolerance remains a future-term goal, the field has begun to explore the usefulness of *near-term* devices with limited numbers of qubits that lack error correction.

1.3 Advances in quantum computing c. 2010 - present

Currently, a device with millions of high-quality physical qubits is out of reach. However, very small devices with a modest number of noisy qubits are being built: within the past 5 years, "state-of-the-art" in quantum computers has gone from 1 or 2 qubits to 5 to 10. Further, next-generation devices with 20 to approximately 100 qubits are being brought online or are under development (253; 312; 167). These small devices are typically called *quantum information processors* (QIPs). QIPs *are not* quantum computers - they won't have error-corrected qubits, and they won't be

able to run sophisticated algorithms.

Quantum computing is thus at an "in-between" or "adolescent" stage: experimentalists are building enough noisy qubits that tackling new problems seems feasible, but qubit error rates are sufficiently high that very long computations are impractical. Two phrases describe this watershed moment – "quantum supremacy" and "noisy, intermediate-scale quantum" (NISQ). Coined by John Preskill in late 2011 (239) and 2017 (240), respectively, these phrases highlight and temper the promise of what near-term QIPs can reasonably be expected to achieve.

Briefly, "quantum supremacy" will be demonstrated when a QIP does some task that cannot be done using then-available classical computing power. Earlier in this chapter, I noted that quantum systems can be entangled (correlations encoded in the global state of a quantum system, as opposed to subsystems), and suggested that a classical computer would have to keep track of this global state in order to faithfully simulate the system. A heuristic argument for "quantum supremacy" goes as follows: a quantum system over N qubits requires $\mathcal{O}(2^N)$ real amplitudes to describe a general (pure) state; if each of these amplitudes is stored using b bits of precision, then a brute-force simulation of a highly-entangled quantum system is to have at least $b * 2^N$ bits of classical storage available[4]. Given a computer with a fixed amount of storage, there's always a sufficiently large quantum system whose storage requirements exceed that. However, a highly-controlled quantum computer with N qubits *might* be able to do such a simulation.

The argument just given is not a "proof" of why quantum supremacy is achievable in any formal sense. Indeed, there are many techniques for simulating complex quantum systems beyond a brute-force approach[5]. More compelling (formalized)

[4]In practice, more storage will be required, as the computer will need to keep track of the gates being applied to the state.

[5]For example, Hartree-Fock, coupled cluster, and density functional theory are all well-established methods for simulating quantum systems.

arguments for quantum supremacy have been developed using facts about sampling from the output distribution of some quantum circuits. The canonical example is the BOSON SAMPLING problem, introduced by Aaronson and Arkhipov in 2010 (2). They showed that, under certain computational complexity assumptions, sampling from the output distribution of a particular linear optics experiment – N single photons traveling through a linear-optical network with M ports – is quite hard for classical computers. In contrast, that output distribution could be sampled using a QIP by "simply" building the optical network and then running it. Current experimental demonstrations of BOSON SAMPLING have used a modest number of photons ($\lesssim 10$) (22; 284; 57; 294).

Another problem proposed for demonstrating quantum supremacy is RANDOM CIRCUIT SAMPLING. As the name implies, it is a sampling problem that requires the computer to sample from the output distribution of a *random* quantum circuit. This sampling problem is expected to be hard for classical computers (40). Intuitively the reason for this hardness is that the evolution of an input state under a random circuit will display chaotic behavior. That is, given a fixed input state, small perturbations to the Hamiltonian that generates the evolution of that state could lead to radically different output states[6]. Therefore, accurate sampling from the output distribution using a classical computer would necessitate a high-accuracy simulation of the evolution, which leads back to the earlier issue of (classical) storage requirements. Again, this output distribution could be sampled by a QIP by "simply" running the random circuit.

Notice that problems such as FACTORING are not typically considered as candidates for demonstrating quantum supremacy. Shor's factoring algorithm requires FTQEC, which will not be available on near and intermediate-term QIPs. Therefore, while factoring RSA-2048 (or any other large number) would certainly be a notable achieve-

[6]See Section 2.1.1 for a few details on Hamiltonian dynamics in quantum systems.

ment in the development of quantum computers, devices being built today won't have the requisite fault-tolerance infrastructure to do so.

Quantum supremacy is an evolving milestone in the development of QIPs, as research groups develop new classical algorithms for simulating quantum systems with many qubits (137; 232; 63; 207). This "back-and-forth" between the quantum and classical computing communities certainly has its own benefits in the form of more powerful algorithms and a better understanding of quantum simulation. The current folk wisdom is that a QIP with a number of qubits between 50 and 100 that runs BOSON SAMPLING or RANDOM CIRCUIT SAMPLING could not be simulated by even the most powerful supercomputers of today. At that point, the field will have "attained" quantum supremacy.

If the phrase "quantum supremacy" highlights the promise of near-term QIPs, then the phrase "noisy, intermediate-scale quantum" (NISQ) helps re-focus attention on the realities of building them. NISQ processors have more qubits than what has typically been available historically, but noise rates will be too high, and qubits too few, to enable large-scale quantum error correction. (Here and in what follows, "processor" is used as a synonym for "QIP".) Consequently, they will not be able to run some of quantum computing's "killer apps" (Shor's factoring algorithm, or Grover's search). In the meantime, researchers have been developing *pre-fault-tolerant* algorithms. Hurdles include: (a) the number of qubits is modest, (b) qubit noise rates are high, and (c) the circuit depth (the number of primitive operations in the circuit) cannot be very large.

In spite of these hurdles, NISQ processors can run some non-trivial algorithms that could be used to solve problems of interest (255; 291; 46; 324; 141; 268). Such algorithms are said to demonstrate "quantum advantage", because unlike the toy problems used for demonstrating quantum supremacy, the problems they solve are useful and not contrived. For this reason, achieving quantum advantage is perhaps

a more compelling milestone in the development of NISQ processors and the field of quantum computing in general.

The past 5 years has seen a surge in *private-sector* interest in the field. Companies such as Google, IBM, Intel, and Microsoft are all working on building NISQ processors. As of the time of this writing, hardware startups include IonQ, Rigetti Quantum Computing, and Silicon Quantum Computing. A growing ecosystem of companies and firms are growing up around these and other companies. All told, a fledgling private-sector industry is coalescing around quantum computing.

A paradigm shift has also taken place on the practical question of how a quantum computer would be used, and how quantum computing fits into modern computing environments (e.g., data centers). The NISQ processors being built today are not being installed on-site at a user facility. Instead, they are being accessed through a cloud-based service such as IBM's Quantum Experience (312) or Rigetti's Forest API (252). NISQ processors can act as an "accelerator", analogous to a graphics processing unit (GPU) or tensor processing unit (TPU) in a modern data center, where users design their programs to offload *some*, but not all, of their computation onto special-purpose hardware. This paradigm makes clear that NISQ processors could be useful for speeding up the time-to-solution for *some* complex problems, but that the bulk of the computation will be classical.

1.4 Summary

This era in human civilization has sometimes been dubbed the "Information Age". It's an appropriate adage, as we generate and store an increasing amount of information/data each year. We have also sought to solve increasingly complex problems, and develop ever-more sophisticated computational tools for doing so. From the development of the CPU, to the GPU, and now the TPU, special-purpose hardware is becoming increasingly necessary to tackle certain problems. As time goes by, "QPU"

(*quantum* processing unit) may be added to that list[7].

Quantum computing has experienced ebbs and flows of interest over the past 30 years. It uses a fundamentally different paradigm for computing and that difference makes it an engaging topic to study[8]. As quantum computing enters the NISQ era, scientists and engineers will have their hands full. One of the major issues with a NISQ processor is the fact it is error-prone: processor initialization, running circuits, and performing measurements are all noisy and imperfect operations. This limits how long the processor can run before it effectively "crashes". (Here, "crashing" could mean that some of the quantum information stored in the processor has decohered into the surrounding environment, or that some of its qubits have been lost, among other things.) In the near term, improving the performance of NISQ processors is important. The motivation behind the research in this thesis is the observation that techniques for characterizing the one or two-qubit systems of today won't be sufficient for characterizing the devices coming online today or that will be built in the near-term future.

The major research question addressed in this thesis is "What are barriers to characterizing a NISQ processor, and what are some steps that can be taken to overcome them?". Chapters 3 and 4 contribute an answer by (a) discussing how *statistical model selection* will be useful in building simpler models of a QIP's behavior, (b) showing that blithely applying classical statistical model selection techniques to the problem of choosing between those doesn't work, and (c) proposing remedies. Chapter 5 leverages the power of *machine learning algorithms* to develop targeted characterization techniques for specific processor properties, and Chapter 6 uses machine learning algorithms to do experiment design.

[7]Whether that acronym will refer to a NISQ processor or a large-scale, fault-tolerant quantum computer remains to be seen.

[8]Note that other computing paradigms such as *neuromorphic* computing (212) have also been proposed for "post-classical" computing. Quantum computing is not unique in that regard.

In the future, large-scale, fault-tolerant quantum hardware may be available for arbitrary-length, universal quantum computing. In the meantime, as the NISQ era gets underway, we must face the challenges of understanding the utility of NISQ processors, and debugging/characterizing their performance.

Chapter 2

Quantum characterization, validation, and verification (QCVV)

The aim of natural science is not simply to accept the statements of others, but to investigate the causes that are at work in nature. - St. Albert the Great, patron saint of scientists (1491) (203)

To harness and increase the computational utility of a NISQ processor, characterization, validation, and verification of the device is necessary. This chapter traces the history of "quantum characterization, validation, and verfication" (QCVV) techniques, and discusses the need for new QCVV techniques in the NISQ era.

2.1 Introduction

Chapter 1 left off with the observation that quantum computing is moving into the "noisy, intermediate-scale quantum" (NISQ) era. As processors with 50 to 100 qubits

come online, characterizing their behavior and evaluating their performance will be necessary to (a) understand what computation(s) these processors can execute, (b) detecting and diagnosing their failure modes, and (c) proposing ways to fix them. This issue is not unique to quantum computers or information processors, however. *Classical* computer hardware can suffer from bugs introduced during the manufacturing process (67); the task of *chip verification* is to certify the chip performs as desired (246). The corresponding task in quantum computing goes by the name "QCVV", for "quantum characterization, verification, and validation. In typical use, "QCVV" is synonymous with "characterize", and that's the usage I'll retain throughout this thesis. In the remainder of this chapter, I introduce the relevant postulates of quantum mechanics and some mathematical preliminaries (Section 2.1.1), trace some of the history of QCVV techniques (Sections 2.2 and 2.3), and conclude by explaining why new techniques will be required to characterize next-generation hardware (Section 2.4).

2.1.1 Postulates of quantum mechanics and mathematical preliminaries

Quantum states

Ever since physicists formalized the notion of the state of a quantum system, a tension has existed regarding the reality of the "quantum state". Depending on one's philosophic bent, this state could be an objective truth about the world, or simply the best description of it given available knowledge. Either way, there's a natural question that arises – "Given some quantum system, is there a procedure or process that can be used to extract information about the system, and update (or estimate) its state?". That is, "Can the state of the system be inferred using data collected by measuring it?". This question (rather, a specific variant of it) is typically referred to as the *Pauli question* (70; 230). To discuss this problem more

precisely, some mathematics and notation will need to be introduced, along with some of the postulates of quantum mechanics. These preliminaries will be necessary for the discussion of quantum *state tomography* in Section 2.2.1.

Postulate 1 (quantum states): *The state of a quantum system can be described using a* density operator *that acts on some Hilbert space.*

If the system has d dimensions (number of distinguishable states), then the associated Hilbert space is d-dimensional, and ρ is an operator with the following properties:

1. $\rho \in \mathcal{B}(\mathcal{H}_d)$, where $\mathcal{B}(\mathcal{H}_d)$ is the set of bounded operators acting on the Hilbert space $\mathcal{H}_d$.
2. $\mathrm{Tr}(\rho) = 1$, where $\mathrm{Tr}(A)$ denotes the trace of A.
3. $\rho \geq 0$, meaning that $\langle\psi|\rho|\psi\rangle \geq 0 \ \forall \ |\psi\rangle \in \mathcal{H}_d$[1].

Each of these properties follows from a specific requirement that's necessary for quantum mechanics. Requiring ρ to be an element of $\mathcal{B}(\mathcal{H}_d)$ is necessary when describing *continuous-variable* quantum systems (where the dimension of the associated Hilbert space is uncountably infinite.) In this thesis, (except in Section 4.2) the focus will be solely on *finite-dimensional* quantum systems. For finite-dimensional systems, the condition $\rho \in \mathcal{B}(\mathcal{H}_d)$ could be relaxed to requiring ρ be in the set of *linear* operators acting on the Hilbert space. For finite-dimensional systems, $\rho \in \mathcal{B}(\mathcal{H}_d)$ implies that ρ can be represented as a $d \times d$ matrix. A general d-dimensional quantum system is called a "*qudit*"; if $d = 2$, a special name is used - "*qubit*". Generally, ρ is represented as a $d \times d$ matrix. However, some states can be written in the form $\rho = |\psi\rangle\langle\psi|$ with $\langle\psi|\psi\rangle = 1$; such a state is said to be *pure*. Pure states can be represented simply as a column vector with d rows. If ρ cannot be written in this way, it is said to be *mixed*.

The second and third properties follow from other postulates of quantum mechanics

[1]Later I will discuss the use of the symbol $|A\rangle$ to denote the vector A.

that relate to *measurement* of a quantum system. Explaining these two properties requires a bit of a digression on measuring a quantum system, and the introduction of more notation.

Postulate 2 (quantum measurements): *Any physical quantity that can be measured about the system (an* observable*) is described by a $d \times d$ Hermitian operator O.*

All Hermitian matrices are *normal* (can be diagonalized), meaning O has a *spectral decomposition* of the form

$$O = \sum_{j=1}^{d} o_j |o_j\rangle\langle o_j|, \tag{2.1}$$

where $O|o_j\rangle = o_j|o_j\rangle$. This equation also uses the *Pauli bra-ket notation* that was used earlier in Postulate 1. This notation is a succinct and convenient way of expressing vector operations. Usually, the *ket* $|A\rangle$ is thought of as a *column vector*; the bra (*dual vector*) $\langle A|$ is a linear map from kets to scalar variables, by the complex dot product:

$$\langle A|B\rangle = \sum_j A_j^\star B_j. \tag{2.2}$$

That is, $\langle A|$ is a row vector, but one where every element has been replaced by its complex conjugate. Thus, to represent an inner product in bra-ket notation, all one needs to write is $\langle A|B\rangle$. Now, Equation (2.1) doesn't use an *inner* product; instead, it uses an *outer* product, $|A\rangle\langle B|$. This object is a *matrix*, with matrix elements given by $(|A\rangle\langle B|)_{jk} = \langle j|A\rangle\langle B|k\rangle = \langle j|A\rangle\langle k|B\rangle^\star$. In contrast, the inner product is a map from vectors and dual vectors to scalars.

If observable O is measured, then the measurement outcome is one of the eigenvalues o_j. Quantum mechanics is a *probabilistic* theory about the measurement outcomes. The probability of observing outcome o_k is given by the Born rule:

$$p_k = \mathrm{Tr}(\rho|o_k\rangle\langle o_k|) = \langle o_k|\rho|o_k\rangle. \tag{2.3}$$

With these postulates in hand, we can now see why properties 2 and 3 above are necessary. Because probabilities are non-negative, ρ itself must be a non-negative (*positive-semidefinite*) matrix - otherwise some p_j could be less than zero. When measuring an observable, *some* measurement outcome has to occur, meaning that the total probability $\sum_j p_j$ must be 1. Since O is Hermitian, its eigenvectors are orthogonal. From the eigenvalue condition $O|o_j\rangle = o_j|o_j\rangle$, it follows that they have norm 1 ($\langle o_j|o_j\rangle = 1$). Because the d vectors $|o_j\rangle$ are orthonormal, $\sum_j \langle o_j|\rho|o_j\rangle = \mathrm{Tr}(\rho)$. Consequently, conservation of probability means $\mathrm{Tr}(\rho) = 1$.

The kind of measurements just discussed are called *projective*, because the measurement is described by an operator $P_j = |o_j\rangle\langle o_j|$ that satisfies $P_j^2 = P_j$. The formalism developed using projective measurements can be generalized by modeling a measurement on a d-dimensional quantum system using a *positive, operator-valued measure* (POVM):

Postulate 2' (quantum measurements): *Measurements on a quantum system are described by a POVM, which associates to each measurement outcome j a $(d \times d)$ Hermitian operator E_j satisfying $E_j \geq 0$ and $\sum_j E_j = \mathcal{I}_d$ where $\mathcal{I}_d$ is the d-dimensional identity matrix.*

Using the POVM formalism, the Born rule (Equation (2.3)) becomes $p_j = \mathrm{Tr}(\rho E_j)$. Another way of writing p_j as function of ρ and E_j – one that will be useful later – can be developed by observing that the trace is an inner product on Hermitian matrices. That is, if A and B are two Hermitian matrices, then $\mathrm{Tr}(A^\dagger B)$ is a valid inner product, as it satisfies the requirements of (a) conjugate symmetry ($\mathrm{Tr}(A^\dagger B) = \mathrm{Tr}(B^\dagger A)^\star$), (b) linearity ($\mathrm{Tr}(A^\dagger[B+C]) = \mathrm{Tr}(A^\dagger B) + \mathrm{Tr}(A^\dagger C)$, and (c) positive semi-definiteness ($\mathrm{Tr}(A^\dagger A) = 0 \iff A = 0$.) Consequently, $\mathrm{Tr}(A^\dagger B)$ is an inner product between two matrices, which I write as $(A|B)$. This "super/generalized-Pauli" notation makes calculations involving the trace of a product of matrices easier to think about[2]. In

[2]This inner product is sometimes written using two angle brackets: $\mathrm{Tr}(A^\dagger B) = \langle\langle A|B\rangle\rangle$.

this notation,

$$p_j = \mathrm{Tr}(\rho E_j) = \mathrm{Tr}(E_j^\dagger \rho) \to (E_j|\rho). \tag{2.4}$$

Notice the POVM formalism makes contact with the "measurements are projections onto eigenstates of an observable" formalism by taking $E_j = |o_j\rangle\langle o_j|$. The POVM formalism is more general. For instance, the POVM effects are not required to be projectors ($E_j E_k \neq \delta_{jk} E_j$). In addition, the POVM formalism more readily allows for "chaining together" measurements back-to-back.

Postulate 3 (state collapse)[3]: *If the state ρ is measured and outcome j observed, then after the measurement, the state of the system "collapses"*:

$$\rho \to \frac{M_j \rho M_j^\dagger}{\mathrm{Tr}(\rho E_j)}, \text{ where } E_j = M_j^\dagger M_j. \tag{2.5}$$

Note that because the POVM effects are positive operators, they always have a decomposition of the form $M_j^\dagger M_j$, just as any real number $x \geq 0$ has a root of the form $x = a^\star a$. However, this decomposition is not unique: taking $M_j \to U M_j$ leaves E_j invariant. Physically, this corresponds to a measurement model in which the state ρ is measured by coupling it to an ancilla, measuring the ancilla, and then applying a post-measurement unitary to the state[4]. This state collapse postulate implies that *repeated measurements on a quantum system do not measure the same (original) state as the system was prepared in.* Suppose the system is prepared in a state ρ, and consider two measurements E_1, E_2 that are performed one after another. After E_1, the state is

$$\rho \to \rho' = \frac{M_1 \rho M_1^\dagger}{\mathrm{Tr}(\rho E_1)}, \tag{2.6}$$

[3]Note: Commonly postulates 2 or 2' are lumped together with 3. I split them here to highlight the difference between the projective measurement and POVM formalisms.

[4]This detail will not be necessary for the chapters that follow, but I include it here for completeness.

and after E_2, it is

$$\rho' \to \rho'' = \frac{M_2 \rho' M_2^\dagger}{\mathrm{Tr}(\rho' E_2)}. \tag{2.7}$$

Because $\rho' \neq \rho$, generally, the outcome probability for E_2 depends on whether E_1 or E_2 is measured first. Therefore, in order to build up sufficient statistics about the *distribution* of outcomes (i.e., their probabilities), we will require many *identical* copies of ρ. The approach described here measures each copy *one at a time*; *joint* measurements are possible across many copies changes the sample complexity necessary for accurate recovery of the state (15; 136).

Finally, if ρ_A describes system A and ρ_B describes system B, the *joint/composite* state of the *combined* system is given by $\rho_{AB} = \rho_A \otimes \rho_B$. The state ρ_{AB} is said to be *separable* with respect to subsystems A and B. Note that when using the bra-ket notation, the $\otimes$ is sometimes suppressed; for example, $|0\rangle \otimes |0\rangle$ is often written as $|00\rangle$. Given a general state ρ_{AB}, it is said to be *separable* if it can be written in the form

$$\rho_{AB} = \sum_j p_j \rho_{A,j} \otimes \rho_{B,j} \quad \text{where} \quad \sum_j p_j = 1. \tag{2.8}$$

If this decomposition is not possible, ρ_{AB} is said to be *entangled*. When there is only one term in the sum (i.e. $\rho_{AB} = \rho_A \otimes \rho_B$), ρ_{AB} is in a *product state*. As an example of an entangled state, let $|\psi\rangle_{AB} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, and take ρ_{AB} to be the pure state $|\psi\rangle_{AB} \otimes \langle\psi|_{AB}$. Entanglement is a resource for quantum computation and communication (recall Chapter 1), but I do not discuss it in further detail here.

Quantum processes

QIPs run quantum algorithms (such as those discussed in Chapter 1) by performing operations on qubits. Analogous to how classical computers implement digital logic using binary operations such as AND, NOT, and OR, QIPs implement these operations using certain *primitive operations*, referred to as *gates*, *channels*, or *processes*.

These operations affect the quantum state of the QIP. As physical examples, consider a laser beam that's used to drive an electron from one energy level to another, or wave plates that change the polarization of a propagating photon. The mathematical preliminaries introduced here will be necessary for the discussion of quantum *process tomography* in Section 2.2.2.

A quantum channel will be denoted by $\mathcal{E}$; typically, the notation $\mathcal{E}[\rho]$ will mean "the state that results from applying $\mathcal{E}$ to the state ρ". (Recall classical logic, where a Boolean gate acts on bits.) If $\mathcal{E}_1$ and $\mathcal{E}_2$ are applied in succession, the resulting channel is their composition: $(\mathcal{E}_2 \circ \mathcal{E}_1)[\rho] = \mathcal{E}_2[\mathcal{E}_1[\rho]]$. Physically, this means "Start with ρ, apply $\mathcal{E}_1$, and then apply $\mathcal{E}_2$.".

The joint channel that results from applying channel $\mathcal{E}_A$ on subsystem A and channel $\mathcal{E}_B$ on subsystem B is $\mathcal{E}_{AB} = \mathcal{E}_A \otimes \mathcal{E}_B$. Some composite channels $\mathcal{E}_{AB}$ can generate entanglement; consider the CNOT gate, which acts on pure states as

$$\text{CNOT}[|j\rangle \otimes |k\rangle] = |j\rangle \otimes |j \oplus k\rangle. \tag{2.9}$$

The CNOT gate flips the second qubit if the first qubit is in the state $|1\rangle$. Applying the CNOT gate to the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle$ yields the entangled state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. (As we'll see, quantum channels are linear operations.) Quantum mechanics postulates certain properties for quantum channels, which I review here.

Postulate 4 (closed system dynamics): *For* closed *quantum systems (systems that do not interact with their environment), a quantum state evolves under the action of unitary operators*:

$$\rho(t) = \mathcal{U}(t, t')[\rho(t')] = U(t, t')\rho(0)U^\dagger(t, t'), \tag{2.10}$$

where $U(t, t')U^\dagger(t, t') = \mathcal{I}$. The operator $U(t, t')$ *propagates* the state from time t' to time t. This operator satisfies the differential equation

$$\dot{U}(t) = -\frac{i}{\hbar}H(t)U(t), \tag{2.11}$$

where the operator $H(t)$ is the underlying *Hamiltonian* generating the time dynamics of the system. Generally, the Hamiltonian $H(t)$ does not *commute*[5] with itself at different times, so the solution to the above equation is

$$U(t,t') = \mathcal{T}\left[\text{Exp}\left(-\frac{i}{\hbar}\int_{t'}^{t} H(s)\ ds\right)\right], \tag{2.12}$$

where $\mathcal{T}$ is the time ordering operator. If $H(t)$ does commute with itself at different times, the time ordering is trivial, and

$$U(t,t') = \text{Exp}\left(-\frac{i}{\hbar}\int_{t'}^{t} H(s)\ ds\right). \tag{2.13}$$

Finally, if $H(t)$ is time-independent, then the integral is trivial:

$$U(t,t') = \text{Exp}\left(-\frac{i}{\hbar}H(t-t')\right). \tag{2.14}$$

These solutions are very useful when solving for the evolution of a given system under some Hamiltonian. To develop a slightly more general framework for describing the transformations of quantum states, we examine two properties that are satisfied by closed system dynamics. First, unitary transformations preserve the trace of ρ:

$$\text{Tr}[\mathcal{U}(t,t')[\rho(t')]] = \text{Tr}(U(t,t')\rho(t')U^{\dagger}(t,t')) = \text{Tr}(\rho(t')). \tag{2.15}$$

Second, unitary transformations preserve the *positivity* of the output state, meaning that $\mathcal{U}(t,t')[\rho(t')] \geq 0$, provided the input state $\rho(t') \geq 0$.

Exactly how these properties are generalized to arbitrary closed-system dynamics is beyond the scope of this chapter. See (223) for details. Suffice it to say that we want $\mathcal{E}$ to have the following properties:

1. It is *trace-preserving*: $\text{Tr}[\mathcal{E}[\rho]] = \text{Tr}[\rho]$.
2. It satisfies *complete positivity*: $(\mathcal{E}_A \otimes \mathcal{I}_B)[\rho_{AB}] \geq 0\ \forall \rho_{AB} \geq 0$.

[5]The commutator of two operators A, B, denoted $[A,B]$, is $[A,B] = AB - BA$.

Any channel satisfying these two conditions is called "CPTP", and I will focus exclusively on such channels here. Because $\mathcal{E}$ is CP, the *Kraus decomposition theorem* guarantees $\mathcal{E}$ has a representation as

$$\mathcal{E}[\rho] = \sum_{j=1}^{k} K_j \rho K_j^\dagger, \tag{2.16}$$

where the operators $\{K_j\}$ are called the *Kraus operators* for the channel (223). This theorem also shows that the converse is true: - if a representation of the form given in Equation (2.16) exists, then $\mathcal{E}$ is completely positive. Because $\mathcal{E}$ is TP, the Kraus operators satisfy $\sum_j K_j^\dagger K_j = \mathcal{I}$. The number of terms in the sum, k, is upper-bounded by d^2. A channel is called *unitary* if, and only if, $k = 1$.

Given a process $\mathcal{E}$, there are many equivalent *representations* for describing its action. Appendix H discusses these representations. Readers unfamiliar with the Kraus, superoperator, and Pauli transfer matrix representations are encouraged to consult it.

With these mathematical preliminaries in hand, I now turn to an overview of QCVV techniques from roughly 1980 to the present day.

2.2 "Traditional" QCVV: c. 1980 - c. 2005

2.2.1 Quantum state tomography

Earlier in Section 2.1.1, I pointed out that the question of whether the state of a quantum system can be reliably inferred from measurements on it is a problem almost as old as quantum mechanics itself. Pauli's original question focused on the problem of characterizing *pure* states. The problem of characterizing general quantum states – pure or mixed – is much newer, and is known as *quantum state tomography* (14; 12; 229). The name "tomography" is used because solutions to this

task were originally developed in the context of characterizing continuous-variable quantum systems using techniques applicable to medical imaging (12; 247; 204; 12; 228; 310; 318). There are several notions of "characterizing" a quantum state. One of the most common – and the one discussed here – is "estimate the density matrix". Others include "estimating the amount of entanglement in the state" (39; 129), or "estimate its fidelity with a fiducial (target) state" (106; 129).

If ρ_0 is the state that (best) describes the device, then the problem of state tomography is 'Given data generated by measuring N_{samples} copies of ρ_0 using a POVM $\{E_j\}$, compute an estimate[6] $\hat{\rho}_0$." At least 2 questions have to be answered in solving this problem: "What POVM should be performed?" and "How will experimental data be processed to compute the estimate?". The first question is one of *experiment design*; the second *data processing*.

The question of experiment design naturally lends itself to questions about *optimal* POVMs, their properties, etc. A variety of POVMs have been developed that have different properties and utility, such as *symmetric, informationally complete* POVMs (251), and *mutually unbiased bases* (154). The optimal sample complexity[7] with respect to different loss metrics is known (136; 183). For example, if the loss metric is the trace distance $D(\rho, \sigma) = \frac{1}{2}||\rho - \sigma||_1$, then to achieve $D(\rho_0, \hat{\rho}) \leq \epsilon$, then $N_{\text{samples}} \geq \Omega(d^2/\epsilon^2)$ is necessary (136), and $N_{\text{samples}} \leq \mathcal{O}(d^2/\epsilon)\log(d/\epsilon)$ is sufficient. Note that the estimation strategy uses an experiment design with *joint* measurements of $\rho_0^{\otimes N_{\text{samples}}}$. For independent measurements – each copy is measured one-at-a-time – $N_{\text{samples}} \geq \Omega(d^3/\epsilon^2)$ is necessary.

My focus in this section is less on experiment design, and more on data processing. Depending on one's statistical inclination, there are two major methodologies for

[6]Throughout this thesis the caret symbol is used to denote estimates, so that $\hat{\theta}$ means "an estimate of θ". Some authors use the symbol to denote quantum-mechanical operators; I do not do so here.

[7]Number of copies of ρ_0 needed

processing the data: *frequentist* and *Bayesian*. Both use the fact that the probabilities of the measurement outcomes ("outcome probabilities") are given by the Born rule: $p_j = \mathrm{Tr}(\rho_0 E_j)$. However, outcome *probabilities* are generally not available, since $N_{\mathrm{samples}} \ll \infty$, and so the outcome probabilities need to be estimated in one fashion or another. A common estimator is to take the observed outcomes and divide by the total number of samples. If the outcome modeled by E_j was seen n_j times out of the N_{samples} trials, then a reasonable estimator for p_j is the observed frequency $f_j = n_j/N$: $\hat{p}_j = f_j$.

Frequentist estimation strategies

In a *frequentist paradigm*, there are at least two ways to compute $\hat{\rho}_0$: *linear inversion* and *maximum likelihood*.

Linear inversion tomography proceeds by observing that outcome probabilities can be represented as a *matrix multiplication* acting on ρ_0:

$$\begin{aligned} & p_1 = (E_1|\rho_0), p_2 = (E_2|\rho_0), \cdots \\ \implies & \mathbf{p} \equiv \begin{pmatrix} p_1 \\ p_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} (E_1|\rho_0) \\ (E_2|\rho_0) \\ \vdots \end{pmatrix} \equiv M|\rho), \end{aligned} \tag{2.17}$$

where M is the *measurement* matrix specified by the experiment design. By writing the relationship between outcome probabilities and the state ρ_0 in this way, a natural estimation approach suggests itself: replace $\mathbf{p}$ by the estimated outcome probabilities, and then invert Equation 2.17:

$$\mathbf{f} \approx M|\rho_0) \implies |\hat{\rho}_0) = (M^T M)^{-1} M^T \mathbf{f}. \tag{2.18}$$

Note that generally, M will not be invertible (especially if the experiment design is underdetermined, so that the number of rows of M is less than the number of columns), which is why the Moore-Penrose *pseudoinverse* is commonly used. The

pseudoinverse is guaranteed to exist, even when the inverse is not. Note that if M^{-1} exists, then the Moore-Penrose pseudoinverse is equal to it. This estimator is also the solution to the unweighted least-squares optimization problem (228)

$$|\hat{\rho}_0) = \underset{\sigma \in \mathcal{M}_d}{\operatorname{argmin}} \; ||\mathbf{f} - M|\sigma)||^2, \tag{2.19}$$

where $\mathcal{M}_d = \{\sigma \mid \sigma \in \mathcal{B}(\mathcal{H}_d),\ \mathrm{Tr}(\sigma) = 1\}$. Crucially, *the model $\mathcal{M}_d$ does not impose the positivity constraint $\hat{\rho}_0 \geq 0$, meaning it may return an estimate that, while solving the optimization problem, is not physical.* For this reason, linear inversion tomography, while conceptually and computationally simple, is typically eschewed in favor of other, more powerful computational techniques where the positivity constraint *can* be imposed. In certain contexts however, linear inversion is just as powerful (see (283; 131) and section 3.5.3).

One such technique is *maximum likelihood* (338; 149). Maximum likelihood (ML) estimation maximizes the *likelihood function*, given by

$$\mathcal{L}(\rho) = \Pr(\text{Observed data} \mid \rho). \tag{2.20}$$

The likelihood function tells us how probable the data observed was given a particular choice for ρ. An intuitive reason for why ML estimation is a good approach is that if $\mathcal{L}(\rho) = 0$ for some ρ, then the data that was actually observed could never have been generated by ρ. Therefore, it could not have been generated by measurements of ρ! For this reason, ML estimation posits that a good estimate for ρ_0 is the state that *maximizes* the likelihood function:

$$\hat{\rho}_{\mathrm{ML},\mathcal{M}} = \underset{\rho \in \mathcal{M}}{\operatorname{argmin}} \; \mathcal{L}(\rho), \tag{2.21}$$

where $\mathcal{M}$ is a set of density matrices. I'll discuss in Chapter 3 why a good choice for $\mathcal{M}$ is crucial for accurate inference of ρ_0, and how statistical model selection can be used to ensure a good choice is made. As an estimation paradigm, ML is well-studied, and many properties of the ML estimate are known. For these reasons, ML is an attractive approach for state tomography.

For the kind of state tomography problems we'll consider (N_{samples} copies of ρ_0 measured independently), the likelihood function is easy to compute:

$$\mathcal{L}(\rho) = \prod_j [\text{Tr}(\rho E_j)]^{n_j} \quad \text{where} \quad \sum_j n_j = N_{\text{samples}}. \tag{2.22}$$

The likelihood function can be maximized using optimization algorithms such as gradient ascent or conjugate gradient (45; 293). Chapter 3 discusses the implications of the positivity constraint $\hat{\rho}_{\text{ML},\mathcal{M}} \geq 0$ on the behavior of the *distribution* of ML estimates and the impact that has on statistical model selection. Chapter 4 includes a discussion of some of the practical issues that have to be considered when doing ML estimation.

While ML estimation is powerful and has a well-developed statistical theory behind it, it does suffer from the problem that $\hat{\rho}_{\text{ML},\mathcal{M}}$ may be *rank-deficient* (i.e., may have zero eigenvalues). Such an estimate predicts zero probability for a POVM that is the projection onto its eigenstates, so *hedging* techniques have been proposed to ensure the estimate is full-rank (35; 98), by replacing the zero eigenvalues with small, yet non-zero values.

Bayesian estimation strategies

Another strategy that avoids returning an estimate with zero eigenvalues leverages *Bayesian* estimation. A straightforward way to do so is by using a *Bayesian mean* estimate, which computes $\hat{\rho}_0$ as

$$\hat{\rho}_0 = \int \rho \; \mathcal{L}(\rho) \; \pi(\rho) \; d(\rho), \tag{2.23}$$

with $\mathcal{L}(\rho)$ is the likelihood function, $\pi(\rho)$ is the *prior*, and $d(\rho)$ is a *measure* over the state space (36; 123).

2.2.2 Process tomography

Being able to characterize the primitive operations of a QIP is important: the channel implemented by the QIP may not necessarily be the one required in the circuit compilation, and those kinds of errors need to be detected and diagnosed. The task of characterizing an unknown quantum channel $\mathcal{E}$ is called *quantum process tomography* (66; 223). Since the early 2000s, several techniques have been developed for doing so.

One simple process tomography technique is based on the observation that the Kraus operators can be expanded in terms of a fixed basis $\{B_j\}$: $K_j = \sum_{l=1}^{d^2} c_{jl} B_l$. From this, it follows that any CPTP channel $\mathcal{E}$ can be written as

$$\mathcal{E}[\rho] = \sum_{j,l,m} c_{jl} c_{jm}^{\star} B_l \rho B_m^{\dagger} = \sum_{l,m=1}^{d^2} \left(\sum_j c_{jl} c_{jm} \right) B_l \rho B_m^{\dagger} \equiv \sum_{l,m=1}^{d^2} \chi_{lm} B_l \rho B_m^{\dagger}, \quad (2.24)$$

where χ is the $d^2 \times d^2$ *Choi matrix* for the channel. Notice χ is Hermitian: $\chi = \chi^{\dagger}$. The requirement $\mathcal{E}$ be completely positive means that $\chi \geq 0$. By construction, an estimate of χ is sufficient to estimate $\mathcal{E}$. Because χ is a $d^2 \times d^2$ Hermitian matrix that maps complex matrices to complex matrices, it has d^4 parameters. However, if $\mathcal{E}$ is trace-preserving, then the condition $\sum_j K_j K_j^{\dagger} = \mathcal{I}$ implies

$$\sum_{l,m=1}^{d^2} \chi_{lm} B_l B_m^{\dagger} = \mathcal{I}. \quad (2.25)$$

This equation is a set of d^2 constraints ($\mathcal{I}$ is $d \times d$), and the constraint is *linear* in χ. Therefore, there are d^2 linear constraints on χ when $\mathcal{E}$ is trace-preserving, so the number of real parameters necessary to specify a CPTP channel is $d^4 - d^2$.

The outcome probability associated with sending a *known* input state ρ_j into the channel and measuring a *known* POVM effect E_k is

$$p_{jk} = \mathrm{Tr}[E_k \mathcal{E}[\rho_j]] = \sum_{l,m=1}^{d^2} \chi_{lm} \mathrm{Tr}[E_k B_l \rho_j B_m^{\dagger}]. \quad (2.26)$$

This equation can be written in vectorized form by defining a measurement design matrix M with matrix elements $M_{(jk),(lm)} = \mathrm{Tr}[E_k B_l \rho_j B_m^\dagger]$:

$$\mathbf{p} = M\boldsymbol{\chi}. \tag{2.27}$$

If the POVM effects and input state are known, then the design matrix M is also known. Therefore, just as in the case of state tomography, a simple way to estimate χ is to estimate the outcome probabilities $\mathbf{p}$, and then invert Equation 2.27 using linear inversion (223). Alternatively, Equation (2.26) can be used in conjunction with, e.g. maximum likelihood inference (336; 10), as another way of estimating χ.

Around 2005 though, the limitations of state and process tomography were becoming clear, and certain characterization tasks where identified where full knowledge of the state or process were not necessary. For these reasons, researchers began to develop new QCVV techniques.

2.3 "Contemporary" QCVV: c. 2005 - present day

Although state and process tomography are two powerful techniques for characterizing a quantum information processors, they have a few problems. First, state and process tomography estimate matrix elements, and the number of matrix elements describing an n-qubit system (or a gate acting on the state of such a system) scales exponentially with n. For an n-qubit system, the Hilbert space dimension d is 2^n. A general state of the system ρ has $\mathcal{O}(d^2)$ parameters, and a channel, $\mathcal{O}(d^4)$. Second, tomography requires that the Hilbert space dimension is known (a point we'll return to in Chapter 3). Third, under state preparation and measurement error ("SPAM error"/"SPAM"), both kinds of tomography perform poorly. All of these reasons pointed to the need for new QCVV techniques. Over the past decade or so, researchers have been developing various QCVV techniques that probe a QIP

at different levels of interest, from self-consistent inference of the QIP's *gate set*, to characterizing average error rates. These "contemporary"[8] QCVV techniques have been extremely useful for providing more diagnostic information about a QIP.

2.3.1 Randomized benchmarking

Arguably, a watershed moment in the history of QCVV was the development of *randomized benchmarking* (RB) between 2005 and 2008 (175; 91). RB characterizes a QIP without having to directly estimate the state(s) it prepares or the channel(s) it implements. Instead, RB measures an average "error rate" for the QIP known as the "RB number", denoted r (244; 241). To do so, RB prescribes certain experiments. Each experiment consists of the following steps: initialize the QIP to some state $\rho_0 \approx |\psi_0\rangle\langle\psi_0|$, perform a random[9] circuit c (usually drawn from the Clifford group) of depth m, and measure whether the state of the QIP "survived" (i.e., has returned to ρ_0). This measurement can be done by performing the POVM $\{|\psi_0\rangle\langle\psi_0|, \mathcal{I} - |\psi_0\rangle\langle\psi_0|\}$. By running each circuit many times, the survival probability $p_{c,m}$ can be estimated.

By averaging $p_{c,m}$ over many random circuits of depth m, the expected survival probability $\langle p_{c,m}\rangle$ can be estimated. After estimating $\langle p_{c,m}\rangle$ for many values of m, the data are usually fit to a model of the form

$$\langle p_{c,m}\rangle = A + (B + Cm)p^m, \tag{2.28}$$

where A, B, C, and p are the parameters of the model. The RB number is then

[8] Here, "contemporary" is used more to refer to the fact the technique isn't just state or process tomography, and has less to do with when the technique was actually developed.

[9] The circuit is not entirely random, as the final gate of the circuit is chosen so that the circuit would be equivalent to the identity channel if all the gates were perfect. This is why the circuit is typically drawn from the Clifford group, as computing the inverse of a Clifford group element can be done efficiently. RB can be done using other groups (49).

estimated as

$$\hat{r} = \frac{d-1}{d}(1-\hat{p}). \tag{2.29}$$

Note that $d = 2^n$ for an n-qubit benchmarking experiment. Importantly, SPAM error is accounted for in RB by the nuisance parameters. For this reason, plus the fact the RB technique is simple and efficient to implement, RB is one of the most commonly-used techniques for QCVV of a QIP. What's more, many variants beyond the basic idea presented here have been developed (202; 113; 273; 325; 69; 49). While there are issues in extending RB to multi-qubit systems, recent research has developed a method that overcomes some of them (242).

2.3.2 Gate set tomography

Gate set tomography (GST) (214; 37; 125) addresses the SPAM problem in state and process tomography in a different way. GST estimates the entire *gate set* describing the QIP – the state(s) it prepares, the channel(s) it performs, and the measurement(s) it does – in a self-consistent way. Demanding the gate set be self-consistent means that any miscalibrations or errors have to be accounted for and made explicit in the model of the QIP's behavior. Like other QCVV techniques, GST uses a Markovian (CPTP) model for QIP's gate set. Its experiment design consists of circuits that amplify all possible Markovian noise affecting the gate set. Section 5.3.2 gives details on GST.

2.3.3 Other contemporary QCVV techniques

Other contemporary QCVV techniques that go beyond state and process tomography address a variety of characterization problems, such as: direct fidelity estimation (106; 75), drift detection (304), entanglement witnesses (39; 305; 129), estimating Lindblad operators (43), Hamiltonian learning (124), leakage detection (325), randomized benchmarking tomography (RBT) (171), robust phase estimation (RPE)

(172; 264), and unitarity/purity benchmarking (313; 96). Researchers have also constructed sensible *error bars* for state and process tomography (187; 123; 92; 33; 289; 97).

There are two other advances in contemporary QCVV of note. First is the development of "quantum compressed sensing" (126; 105; 257; 164), which generalizes classical compressed sensing – the high-accuracy recovery of a sparse signal using few measurements – to the quantum realm. (In the context of state tomography, ρ_0 is sparse if it has *low rank*, since its spectral decomposition would contain few terms.) As I'll discuss in Chapter 4, some of the work developed in this thesis has implications for this QCVV technique. Briefly, most quantum compressed sensing techniques rely on a *low-rank* assumption about the state being estimated, or require an experimentalist to perform very particular POVMs. The work developed in this thesis provides another avenue for understanding these results, and suggests that *the geometry of quantum state space is sufficient to give compressed-sensing-like behavior for state tomography, without requiring particular POVMs or invoking a low-rank assumption.*

The second development lies at the intersection of state tomography and classical machine learning. As I'll discuss in Chapter 5, machine learning algorithms define highly expressive representations of functions. For this reason, some algorithms – such as neural networks (NNs) (210; 94; 146) or restricted Boltzmann machines (RBMs) (282; 145) – are particularly well-suited to represent quantum states, which we can view as functions from POVM effects to outcome probabilities (via the Born rule). State tomography has traditionally focused on estimating the parameters of the density matrix; in this paradigm, instead of having to *estimate* a large matrix, a NN or RBM has to be *learned*. This paradigm has led to "neural network state tomography" (56; 297), which has overlap with tomography by tensor networks such as MPS/MERA (73; 188). Crucially, a wide variety of physical states can be rep-

resented by these and similar algorithms (114; 298; 82; 81). The work presented in this thesis leverages machine learning algorithms in a different way, by using them to develop new QCVV techniques.

Contemporary QCVV techniques span a spectrum, from predictive modeling (e.g., GST) to regression analysis (e.g., RB). Each of them has utility for certain characterization tasks. As quantum computing enters the NISQ era, more techniques will be required.

2.4 QCVV of next-generation quantum hardware

There are several reasons why new QCVV techniques will be required to characterize intermediate-scale QIPs. First, many existing techniques require resources that grow rapidly with the number of qubits. These resources include the number of experiments, the repetitions of each experiment, and the amount of computation required to process experimental data. As an example, taking single-qubit GST and performing it on 10 separate qubits will require at least 10 times more resources than single-qubit GST performed on just 1 qubit. In practice, the requirements will be much greater because of the need to not only perform single-qubit GST on each of 10 qubits, but also many kinds of multi-qubit GST to detect multi-qubit noise.

Second, even scalable techniques may not be suitable to characterize novel kinds of noise that affect next-generation processors, such as very long-range correlations between qubits or crosstalk. Characterizing novel types of noise is beyond what those techniques can do, simply because they weren't designed with those noise types in mind. Detecting and diagnosing them will generally require new QCVV techniques.

Third, characterizing the "holistic" performance of QIPs on tasks that utilize the whole processor will be necessary. These kinds of device-wide metrics would be useful to, for example, compare different QIPs in a standard way. Of the existing

QCVV techniques, those that infer an error rate (e.g., RB) suffer from the problem that they are typically designed for use on small numbers of qubits, which makes deploying them for holistic characterization difficult. What's more, it is unclear how to characterize the total error rate of a QIP from inferences of the error rates of smaller subcomponents, particularly if the size of the subcomponents grows slowly (or is constant) relative to the size of the QIP.

For all these reasons, scalable QCVV of NISQ processors is likely to demand a wide array of new QCVV techniques. Developing them is a challenging task, and the research presented in the following 4 chapters takes some steps in the direction of overcoming them.

Chapter 3

Impact of state-space geometry on tomography

There are three kinds of lies: lies, damned lies, and statistics. - Mark Twain, 1906

To characterize NISQ devices, simpler models that describe their behavior are required. This chapter introduces the idea of statistical model selection, and shows how commonly-used model selection techniques in classical statistical inference problems cannot be blithely applied to models describing a quantum information processor. To remedy this, I define a new generalization of a powerful statistical framework for reasoning about the infinite-sample behavior of estimators, and discuss its implication for maximum likelihood (ML)[1] *quantum state tomography.*

3.1 Introduction and overview

Determining the quantum state ρ_0 produced by a specific preparation procedure for a quantum system is a problem almost as old as quantum mechanics itself (70;

[1]Please note that in Chapters 5 and 6, the acronym ML will be used for 'machine learning'.

230). This task, known as *quantum state tomography* (229), is not only useful in its own right (diagnosing and detecting errors in state preparation), but is also used in other characterization protocols including entanglement verification (286; 39; 305) and process tomography (10). A typical state tomography protocol proceeds as follows: many copies of ρ_0 are produced, they are measured in diverse ways, and finally the outcomes of those measurements (data) are collated and analyzed to produce an estimate $\hat{\rho}$. This is a straightforward statistical inference process (249; 317), where the data are used to fit the parameters of a statistical model. In state tomography, the parameter is ρ, and the model is the set of all possible density matrices on a Hilbert space $\mathcal{H}$ (equipped with the Born rule). However, we don't always know what model to use. It is not always *a priori* obvious what $\mathcal{H}$ or its dimension is; examples include optical modes (6; 28; 201; 47; 192) and leakage levels in AMO and superconducting (220; 95) qubits. In such situations, we seek to let the data itself determine which of many candidate Hilbert spaces is best suited for reconstructing ρ_0.

Choosing an appropriate Hilbert space on the fly is an instance of a general statistical problem called *model selection*. Although model selection has been thoroughly explored in classical statistics (50), its application to state tomography encounters some obstacles. They stem from the fact that quantum states – and therefore, estimates of them – must satisfy a *positivity constraint* $\rho \geq 0$. (See Figure 3.1.) A similar constraint, complete positivity, applies to process tomography. The impact of positivity constraints on state and process tomography is an active area of research (55; 105; 289; 58), and its implications for model selection have also been considered (271; 132; 304; 187; 330; 219; 177). In this chapter, I address a specific question at the heart of this matter: *How does the loglikelihood ratio statistic used in many model selection protocols, including (but not limited to) information criteria such as Akaike's AIC (5), behave in the presence of the positivity constraint* $\rho \geq 0$?

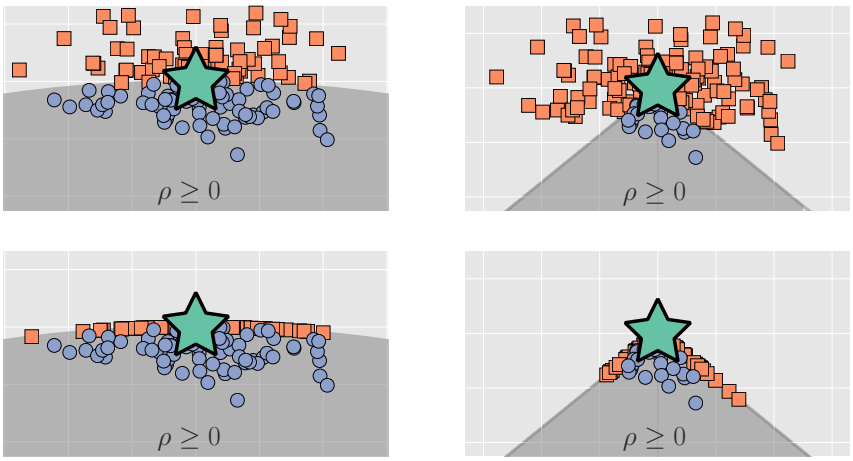

Figure 3.1: **Impact of the positivity constraint ($\rho \geq 0$) on tomographic estimates**. The boundary of quantum state space – which results from the constraint $\rho \geq 0$ – affects maximum likelihood (ML) tomography for a qutrit state ρ_0 (star). Two different 2-dimensional cross-sections of the state space are shown, which correspond to a qubit (left) and a classical 3-outcome distribution (right). **Top**: Without the positivity constraint, some ML estimates (orange squares) are not valid estimates of a quantum state, because they are not positive semidefinite. However, some ML estimates (blue circles) are. Further, the ML estimates are Gaussian distributed. **Bottom**: Imposing the positivity constraint forces the (previously negative) ML estimates to "pile up" on the boundary of state space; the distribution $\Pr(\hat{\rho}_{\mathrm{ML}})$ is not Gaussian, and local asymptotic normality is not satisfied. In turn, the assumptions necessary to invoke the Wilks theorem are not satisfied either.

Section 3.2 begins by introducing the loglikelihood ratio statistic λ, and outline how it can be used to choose a Hilbert space. In Section 3.3, we show how and why the classical null theory for its behavior, the Wilks theorem, falls apart in the presence of the positivity constraint, because quantum state space does not generally satisfy *local asymptotic normality* (LAN). We define a new generalization of LAN, *metric-projected local asymptotic normality* (MP-LAN), in Section 3.4; this generalization explicitly accounts for the positivity constraint, and is satisfied by quantum state space. This chapter provides a novel result that uses the MP-LAN formalism; namely, deriving a replacement for the classical Wilks theorem that *is* applicable in state

tomography (Section 3.5). (Two other results are derived in Chapter 4.) Finally, this chapter concludes in Section 3.6 with a discussion of some other ways MP-LAN could be used to improve statistical model selection for tomographic problems beyond state tomography.

3.2 Statistical model selection and the Wilks theorem

Discussing model selection for state tomography requires introducing some terminology/notation from statistics. A *model* $\mathcal{M}$ is a parameterized family of probability distributions over some data D, usually denoted as $\Pr_{\boldsymbol{\theta}}(D)$, where $\boldsymbol{\theta} \in \mathcal{M}$ are the *parameters* of the model. A fixed value for the parameters is called a *simple hypothesis*; a set of parameters is called a *composite hypothesis*. In state tomography, the parameters are a quantum state ρ, the data are the observed outcomes of the measurement of a positive operator-valued measure (POVM) $\{E_j\}$, and the probability of observing outcome "j"[2] is given by the Born rule: $p_j = \mathrm{Tr}(\rho E_j)$. Throughout this chapter, a model is a set of density matrices, and a state ρ is a particular choice of the model's parameters.

Suppose we have data D obtained from an unknown state ρ_0, and two candidate models $\mathcal{M}_1, \mathcal{M}_2$ that could be used to reconstruct it. Many of the known methods for choosing between them (i.e., model selection) involve quantifying how well each model fits the data by its *likelihood*. The likelihood of a simple hypothesis ρ is defined as $\mathcal{L}(\rho) = \Pr(D|\rho)$. Models, however, are *composite* hypotheses, comprising many possible values of ρ. A canonical way to define model $\mathcal{M}$'s likelihood is via the general method of *maximum likelihood* (ML), by maximizing $\mathcal{L}(\rho)$ over $\rho \in \mathcal{M}$. In practice, the maximization is usually done explicitly to find an ML estimate $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$

[2]The index j may be continuous or discrete.

(149; 155; 35) of $\mathcal{M}$'s parameters, and then $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\hat{\rho}_{\mathrm{ML},\mathcal{M}})$. (Although it is common to refer to $\hat{\rho}_{\mathrm{ML}}$ without specifying the model over which $\mathcal{L}$ was maximized, the model is listed explicitly because in this chapter, many different models are frequently used!)

Then, the models can be compared using the *loglikelihood ratio statistic* (221; 35; 219):

$$\begin{aligned}\lambda(\mathcal{M}_1, \mathcal{M}_2) &\equiv -2 \log\left(\frac{\mathcal{L}(\mathcal{M}_1)}{\mathcal{L}(\mathcal{M}_2)}\right) \\ &= -2 \log\left(\frac{\mathcal{L}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_1})}{\mathcal{L}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_2})}\right) \\ &= -2 \log\left(\frac{\max\limits_{\rho \in \mathcal{M}_1} \mathcal{L}(\rho)}{\max\limits_{\rho \in \mathcal{M}_2} \mathcal{L}(\rho)}\right).\end{aligned} \tag{3.1}$$

All else being equal, a positive λ favors $\mathcal{M}_2$ – i.e., the model with the higher likelihood is more plausible, because it fits the data better. However, all else is rarely equal. If both models are equally valid – e.g., they both contain ρ_0 – but $\mathcal{M}_2$ has more parameters, then $\mathcal{M}_2$ will very probably fit the data better. Models with more adjustable parameters do a better job of fitting *noise* (e.g., finite sample fluctuations) in the data. This becomes strictly true when the models are *nested*, so that $\mathcal{M}_1 \subset \mathcal{M}_2$. In this case, the likelihood of $\mathcal{M}_2$ is at least as high as that of $\mathcal{M}_1$; not only is $\lambda \geq 0$, but almost surely $\lambda > 0$.

Remarkably, the same effect also makes $\mathcal{M}_2$'s fit less accurate (almost surely), because the fit incorporates more of the noise in the data. These two effects constitute *overfitting*, which can be summed up as "Extra parameters make the fit look better, but perform worse.". An overfitted model would fit *current* data extremely well, but would fail to accurately predict *future* data. This provides strong motivation to correct for overfitting by penalizing or handicapping larger models, to prevent them from being chosen over smaller models that are no less valid, and may even yield better estimates in practice (5). As I'll discuss in Chapter 5, a similar problem occurs

when using *machine learning* algorithms to design new QCVV techniques. Just as complex statistical models can overfit the data and then generalize poorly to predict future data, machine learning algorithms can do the same, and require either explicit regularization or cross-validation to ensure the tools they learn generalize well.

For this reason, any model selection method/criterion that relies (explicitly or implicitly) on a statistic to quantify "how well model $\mathcal{M}$ fits the data" also relies on a *null theory* to predict how that statistic will behave if some *null hypothesis* is true. For the model selection problems we consider, the null hypothesis is that $\rho_0 \in \mathcal{M}$, and the null theory will tell us how statistics of interest behave when that null hypothesis is in fact true. A model selection criterion based on an invalid null theory (or a criterion used in a context where its null theory does not apply) will tend to perform sub-optimally (as compared to a method based on a correct null theory).

The null theory can be used to formulate a *decision rule* for choosing between models. If how the test statistic behaves when both models are equally valid is known, then calculating the observed value of the statistic under the null theory is possible. If the observed value is very improbable under the null theory, then that constitutes evidence against the smaller model, and justifies rejecting it. On the other hand, if the observed value is *consistent* with the null theory, there is no reason to reject the smaller model.

The standard null theory for λ is the *Wilks theorem* (323). It relies on *local asymptotic normality* (LAN) (190; 189). LAN is a property of $\mathcal{M}$; if $\mathcal{M}$ satisfies LAN, then as $N_{\text{samples}} \to \infty$:

- The ML estimate $\hat{\rho}_{\text{ML},\mathcal{M}}$ is normally distributed around ρ_0 with covariance matrix $\mathcal{I}^{-1}$:

$$\Pr(\hat{\rho}_{\text{ML},\mathcal{M}}) \propto \exp\left[-\text{Tr}[(\rho_0 - \hat{\rho}_{\text{ML},\mathcal{M}})\mathcal{I}(\rho_0 - \hat{\rho}_{\text{ML},\mathcal{M}})]/2\right]. \tag{3.2}$$

- The likelihood function in a neighborhood of $\hat{\rho}_{\text{ML},\mathcal{M}}$ is locally Gaussian with

Hessian $\mathcal{I}$:

$$\mathcal{L}(\rho) \propto \exp\left[-\mathrm{Tr}[(\rho - \hat{\rho}_{\mathrm{ML},\mathcal{M}})\mathcal{I}(\rho - \hat{\rho}_{\mathrm{ML},\mathcal{M}})]/2\right]. \tag{3.3}$$

Here, $\mathcal{I}$ is the (classical) *Fisher information matrix* associated with the POVM. It quantifies how much information the data carry about a parameter in the model. Note that in expressions involving $\mathcal{I}$, states ρ are treated as vectors in state space, and $\mathcal{I}$ is a matrix or 2-index tensor acting on that state space.

Most statistical models satisfy LAN. When LAN is satisfied *and* N_{samples} is large enough to reach the "asymptotic" regime, we can invoke the Wilks theorem to determine the behavior of λ. This theorem says that under suitable regularity conditions, if $\rho_0 \in \mathcal{M}_1 \subset \mathcal{M}_2$, where $\mathcal{M}_2$ has K more parameters than $\mathcal{M}_1$, then λ is a χ^2_K random variable. This is a complete null theory for λ (under the specified conditions), and implies that $\langle\lambda\rangle = K$ and $(\Delta\lambda)^2 = 2K$.

Therefore, in the "Wilks regime", a simple criterion for model selection would be to compare the observed value of λ to $\lambda_{\mathrm{thresh}} = \langle\lambda\rangle + k\Delta\lambda$, for some $k \approx 1$, and reject the smaller model if $\lambda > \lambda_{\mathrm{thresh}}$. While model selection rules can be more subtle and complex than this (5; 270; 166; 285), they usually take the general form of a threshold in which $\langle\lambda\rangle$ plays a key role. Rather than attempting to define a specific rule, the purpose in this chapter is to understand the behavior of $\langle\lambda\rangle$ and derive an approximate expression for it in the context of state tomography.

The first step in doing so is to explain how and why the Wilks theorem breaks down in that context.

3.3 Quantum state tomography and the breakdown of the Wilks theorem

Quantum state tomography typically begins with N_{samples} independently and identically prepared quantum systems – i.e., N_{samples} copies of an unknown state ρ_0. Each copy is measured, and without loss of generality we can assume that each measurement is described by the same positive operator-valued measure (POVM). A POVM is a collection of positive operators $\{E_j\}$ summing to $\mathbb{1}$, and the probability of outcome "j" is given by $\text{Tr}(\rho_0 E_j)$. The results of all N_{samples} measurements constitute data, represented as a record of the frequencies of the possible outcomes $\{n_j\}$, where n_j is the number of times "j" was observed, and $\sum_j n_j = N_{\text{samples}}$. Finally, this data is processed through some *estimator* to yield an estimate of ρ_0, denoted $\hat{\rho}$.

Although a variety of estimators have been proposed (310; 149; 155; 36; 35; 335; 99), the exact estimator used is not our concern here. However, since we *are* concerned with computing the likelihood of a model $\mathcal{M}$, which is defined as the likelihood of the most likely $\rho \in \mathcal{M}$, we will make extensive use of the *maximum likelihood* (ML) estimator. This should not be taken as advocacy for the ML estimator; it is only a convenient way to find the maximum of $\mathcal{L}$ over $\mathcal{M}$, and once a model is chosen, a different estimator could be used. The likelihood $\mathcal{L}(\rho)$ is

$$\mathcal{L}(\rho) = \prod_j \text{Tr}(\rho E_j)^{n_j}, \tag{3.4}$$

and $\hat{\rho}_{\text{ML},\mathcal{M}}$ is the solution to the optimization problem

$$\hat{\rho}_{\text{ML},\mathcal{M}} = \underset{\rho \in \mathcal{M}}{\text{argmax}}\ \mathcal{L}(\rho). \tag{3.5}$$

In state tomography, $\mathcal{M}$ is almost always the set of all density matrices over a Hilbert space $\mathcal{H}$:

$$\mathcal{M}_{\mathcal{H}} = \{\rho \mid \rho \in \mathcal{B}(\mathcal{H}),\ \text{Tr}(\rho) = 1,\ \rho \geq 0\}, \tag{3.6}$$

where $\mathcal{B}(\mathcal{H})$ is the space of bounded linear operators on $\mathcal{H}$. To determine $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$, we can use the following facts: (a) $\mathcal{M}_{\mathcal{H}}$ is a convex set, and (b) $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ minimizes the value of the convex function $-\log[\mathcal{L}(\rho)]$. Because $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ is the solution to minimizing a convex function over a convex set, it can be found efficiently via any of several algorithms for convex optimization (45).

Usually, $\mathcal{H}$ is taken for granted or chosen by fiat. In this chapter, I consider a nested family of different Hilbert spaces, indexed by their dimension d: $\mathcal{H}_1 \subset \cdots \subset \mathcal{H}_d \subset \mathcal{H}_{d+1} \subset \cdots$. The models we consider are therefore given by:

$$\mathcal{M}_d \equiv \mathcal{M}_{\mathcal{H}_d} = \{\rho \mid \rho \in \mathcal{B}(\mathcal{H}_d),\ \mathrm{Tr}(\rho) = 1,\ \rho \geq 0\}. \tag{3.7}$$

For notational brevity, we will use $\hat{\rho}_{\mathrm{ML},d}$ to denote the ML estimate over $\mathcal{M}_d$. Selecting between these models means determining whether one model (say, $\mathcal{M}_{d+1}$) is "better" than another (say, $\mathcal{M}_d$). To evaluate which is better, the likelihood of each model is computed, and then $\lambda(\mathcal{M}_d, \mathcal{M}_{d+1})$ is used to choose between them. As mentioned in the previous section, this requires having a *null theory* for λ that describes its behavior when $\rho_0 \in \mathcal{M}_d \subset \mathcal{M}_{d+1}$.

The Wilks theorem, which is the classical null theory for λ, relies on local asymptotic normality (LAN). If the models under consideration satisfy LAN, then as mentioned in the previous section, the likelihood $\mathcal{L}(\rho)$ is Gaussian with a Hessian given by the Fisher information. In classical statistics, it is common to assume that boundaries are not relevant, either because the models of interest have none, or because the true parameter values ρ_0 lie far away from them. In the absence of boundaries, and in the asymptotic limit where the curvature of the Fisher information metric is also negligible, many calculations can be simplified by changing to *Fisher-adjusted* coordinates in which the Fisher information is isotropic (i.e., $\mathcal{I} \propto \mathbb{1}$). Under these assumptions and simplifications, the Wilks theorem can be derived.

In quantum state tomography, the Wilks theorem breaks down for two reasons. First, the quantum state space *does* have boundaries. Second, the Fisher information is

anisotropic, and the anisotropy can't easily be eliminated by a coordinate change because those boundaries define a preferred coordinate system. We discuss these obstacles – and our plan to address them – in detail in the remainder of this section.

Given a model $\mathcal{M}_d$, its boundary is the set of rank-deficient states within it. When $\rho_0 \in \mathcal{M}_d$ and is full rank, LAN will hold – which is to say that, asymptotically, the boundary can indeed be ignored. But when ρ_0 is rank-deficient, it lies *on* the boundary of the model. LAN is not satisfied, because positivity constrains $\hat{\rho}_{\mathrm{ML},d}$, and so $\Pr(\hat{\rho}_{\mathrm{ML},d})$ is not Gaussian (see Figure 3.1). The Wilks theorem does not apply in this case, and its predictions regarding $\langle\lambda\rangle$ aren't even close (see Figure 3.2). Moreover, this is the relevant situation for our analysis, because *even if* ρ_0 *is full-rank in* $\mathcal{M}_d$*, it must be rank-deficient in* $\mathcal{M}_{d+1}$. So we require a replacement for the Wilks theorem; that is, we need a null theory for λ when ρ_0 is rank-deficient.

One challenge in deriving this replacement is that the Fisher information generally depends strongly on ρ_0 and the POVM being measured (see Figure 3.5). In many standard derivations, such anisotropy has no impact and can be eliminated easily by changing to Fisher-adjusted coordinates. But the models we consider (quantum states) have boundaries that break scale-invariance, and define preferred coordinate systems. Changing to Fisher-adjusted coordinates does not eliminate the effect of anisotropy, because the boundary has a new shape in the new coordinates that serves as a record of the anisotropy. Moreover, the methods we derive here for calculating the impact of the boundary rely heavily on a particular coordinate system (Hilbert-Schmidt coordinates), and changing to Fisher-adjusted coordinates would break them. This makes it very difficult to derive a precise generalization of the Wilks theorem for *arbitrary* Fisher information, so to derive our results we make the key simplifying assumption that the Fisher information is isotropic with respect to Hilbert-Schmidt metric. (This metric defines a distance between density matrices ρ_1 and ρ_2 that is given by $d(\rho_1, \rho_2) = \mathrm{Tr}[(\rho_1 - \rho_2)^2]$.) This is almost never exactly

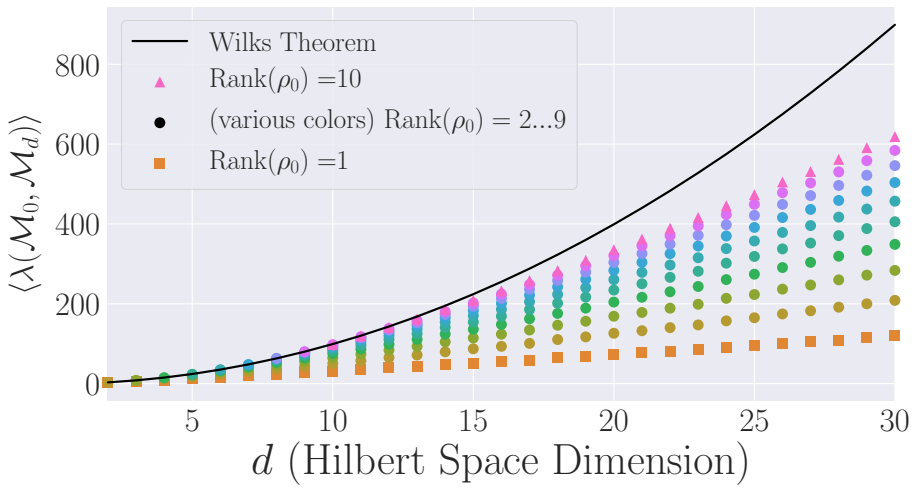

Figure 3.2: **Predictions of the Wilks theorem vs reality**. In the context of state tomography on a true state ρ_0 in a d-dimensional Hilbert space, the Wilks theorem can be used to predict that, when comparing the zero-parameter model $\mathcal{M}_0 = \{\rho_0\}$ and the (d^2-1)-parameter model $\mathcal{M}_d$ defined in Equation (3.7), the expected loglikelihood ratio $\langle\lambda(\mathcal{M}_0, \mathcal{M}_d)\rangle$ will be $d^2 - 1$. Here, we compare that prediction to numerical simulations of tomography on states ρ_0 in dimension $d = 2, \ldots, 30$, with ranks $r = 1, \ldots, \min(10, d)$. The Wilks theorem only predicts $\langle\lambda\rangle$ correctly for full-rank states; when $r \ll d$, the actual expected loglikelihood ratio is much smaller. Our main result (Equation 3.43) gives a replacement that works correctly (see Figure 3.10).

true in practice[3], but it is reasonable to presume that our results remain useful and approximately true when the Fisher information is *almost* isotropic. In Chapter 4 Section 4.2, I present numerical results showing that the null theory for λ developed in this chapter under the assumption of isotropic Fisher information appears to be surprisingly robust to significant anisotropy.

To derive our replacement for the Wilks theorem, we first need a new framework for reasoning about models with convex constraints. Such a framework is developed in the next section by defining a new generalization of LAN.

[3]Jonathan A Gross, private communication

3.4 MP-LAN: a generalization of LAN for models with convex constraints

In this section, I develop a framework that will allow us to derive a replacement for the Wilks theorem that holds for rank-deficient ρ_0. To do so, I define a generalization of LAN in the presence of boundaries, which called *metric-projected local asymptotic normality* (MP-LAN). (For other generalizations of LAN, see (262; 158).) Like LAN, MP-LAN is a property that a statistical model may satisfy. Unlike LAN, MP-LAN is satisfied by quantum state space. For any model that satisfies MP-LAN (quantum or classical), I compute an asymptotically exact expression for λ, a necessary building block in our replacement for the Wilks theorem.

In Section 3.5, I show that the models $\mathcal{M}_d$ satisfy MP-LAN, and derive an approximation for $\langle\lambda\rangle$ (Equation (3.43), on page 71). Section 3.5.5 compares the theory to numerical results.

The reader should note that to enhance readability, in this section (and only this section) N is used to denote the number of samples, previously denoted as N_{samples}.

3.4.1 Defining MP-LAN; overview of its implications

The main definitions and results required for the remainder of the chapter are presented in this subsection. Technical details and proofs are presented in the next subsection.

Definition 1 (Metric-projected local asymptotic normality, or MP-LAN)**.** *A model $\mathcal{M}$ satisfies MP-LAN if $\mathcal{M}$ is a convex subset of a model $\mathcal{M}'$ that satisfies LAN.*

The model $\mathcal{M}'$ will be used to define a set of *unconstrained ML estimates* $\hat{\rho}_{\text{ML},\mathcal{M}'}$, some of which may not satisfy the positivity constraint. While there are many possible choices for this "unconstrained model" $\mathcal{M}'$, we will find it useful to let $\mathcal{M}'$ be a model whose dimension is the same as $\mathcal{M}$, but where any of the constraints that define $\mathcal{M}$

are lifted. (For example, in Lemma 5, we will take $\mathcal{M}'$ to be Hermitian matrices of dimension d.) Other choices of $\mathcal{M}'$ are possible, but I do not explore those here.

Although the definition of MP-LAN is rather short, it implies some very useful properties. These properties follow from the fact that, as $N \to \infty$, the behavior of $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ and λ is entirely determined by their behavior in an arbitrarily small region of $\mathcal{M}$ around ρ_0, called the *local state space*.

Definition 2 (Local state space)**.** *For each natural number N, let $I_N = \mathcal{I}/N$ be the (scaled) Fisher information matrix of $\mathcal{M}$ at ρ_0, and let $\mathcal{M}_N$ be the set obtained by re-scaling each point in $\mathcal{M}$ by $I_N^{-1/2}$. For each N, let C_N be a convex subset of $\mathcal{M}_N$, chosen so that (a) C_{N+1} contains C_N, and (b) $\lim_{N\to\infty} \Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}} \in C_N) = 1$. Then the sequence $\{C_N : N = 1, 2...\}$ converges to the local state space around ρ_0.*

Models that satisfy MP-LAN have the following *asymptotic* properties:

- The local state space is the *solid tangent cone* of the model at ρ_0, denoted $T(\rho_0)$.
- The ML estimate $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ is given by the *metric projection* of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ onto $T(\rho_0)$:

$$\hat{\rho}_{\mathrm{ML},\mathcal{M}} = \operatorname*{argmin}_{\rho \in T(\rho_0)} (\rho - \hat{\rho}_{\mathrm{ML},\mathcal{M}'})\mathcal{I}(\rho - \hat{\rho}_{\mathrm{ML},\mathcal{M}'}). \tag{3.8}$$

  I first encountered the term "metric projection" in the convex optimization literature (209; 8), and inspires our choice for the acronym "MP-LAN". However, it should be noted that in the problem setting considered in those references, $\mathcal{I} = \mathbb{1}$. See discussion in Chapter 4, Section 4.2.7.

- The loglikelihood ratio $\lambda(\rho_0, \mathcal{M})$, defined as

$$\lambda(\rho_0, \mathcal{M}) = -2 \log \left(\frac{\mathcal{L}(\rho_0)}{\max_{\rho \in \mathcal{M}} \mathcal{L}(\rho)} \right), \tag{3.9}$$

  takes the following simple form:

$$\lambda(\rho_0, \mathcal{M}) = \mathrm{Tr}[(\rho_0 - \hat{\rho}_{\mathrm{ML},\mathcal{M}})\mathcal{I}(\rho_0 - \hat{\rho}_{\mathrm{ML},\mathcal{M}})]. \tag{3.10}$$

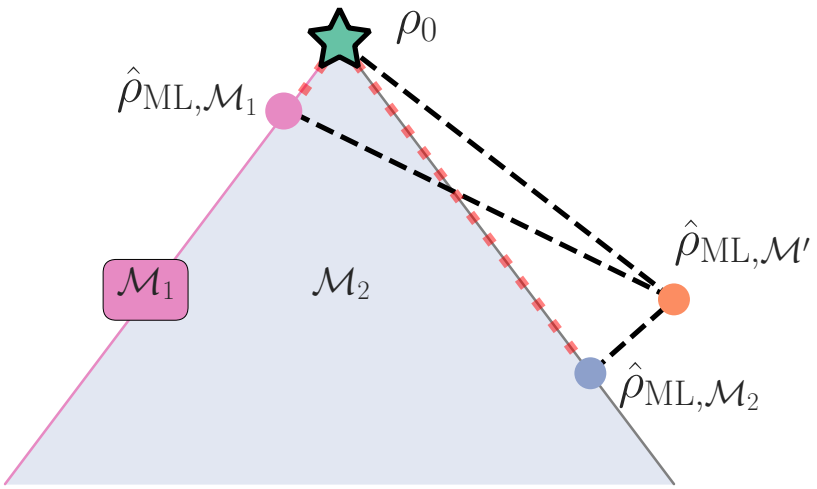

Figure 3.3: **Equivalence of λ and squared distance when MP-LAN is satisfied**. For any model $\mathcal{M}_k$, $\lambda(\rho_0, \mathcal{M}_k)$ is the difference between the squared distance from ρ_0 to $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ and that from $\hat{\rho}_{\mathrm{ML},\mathcal{M}_k}$ to $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ (black lines). If $\mathcal{M}_k$ satisfies MP-LAN, then (a) $\mathcal{M}_k \subset \mathcal{M}'$ for an *unconstrained model* $\mathcal{M}'$, and (b) λ is equal, in the asymptotic limit, to the squared distance from $\hat{\rho}_{\mathrm{ML},\mathcal{M}_k}$ to ρ_0 (red lines), because $\rho_0, \hat{\rho}_{\mathrm{ML},\mathcal{M}'}$, and $\hat{\rho}_{\mathrm{ML},\mathcal{M}_k}$ form a right triangle. This is also true for models with *curved* boundaries (such as quantum state space) because asymptotically, the local state space is the solid tangent cone, whose boundaries are always *flat*.

This property is non-trivial; see Figure 3.3.

Even when $\mathcal{M}$ satisfies MP-LAN, these properties may not be true when N is finite; they are guaranteed only in the asymptotic limit. When N is sufficiently large, we can (and will!) use the asymptotic properties above.

The following subsection presents the technical details and definitions necessary to show the above results. The reader may skip it without loss of continuity, and proceed to Section 3.5.

3.4.2 Technical details: implications of MP-LAN

Assume a statistical model $\mathcal{M}$ that satisfies MP-LAN. Below, I prove the properties of $\mathcal{M}$ asserted in Section 3.4.1.

Convergence of the local state space to the solid tangent cone

Because $\mathcal{M}$ satisfies MP-LAN, there exists a model $\mathcal{M}' \supset \mathcal{M}$ of dimension d' that satisfies LAN. This means that as $N \to \infty$, the distribution of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ converges to a Gaussian:

$$\Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}'}) \xrightarrow{\mathrm{d}} \mathcal{N}(\rho_0, \Sigma/N), \tag{3.11}$$

where $\xrightarrow{\mathrm{d}}$ means "converges in distribution to", and $\Sigma = \mathcal{I}^{-1}$. The shape of the distribution is entirely determined by $\mathcal{I}$. As $N \to \infty$, this Gaussian distribution becomes more and more tightly concentrated around ρ_0. Although there is always a non-zero probability that $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ will be arbitrarily far away from ρ_0, it is possible to define a sequence of balls B_N that shrink with N, yet contain every $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ with probability 1 as $N \to \infty$.

First, we switch coordinates by sending $\rho \to \rho - \rho_0$, establishing ρ_0 as the origin of the coordinate system. In these coordinates, $\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \sim \mathcal{N}(0, \Sigma/N)$, and the following lemma constructs B_N.

Lemma 1. *Let $\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \sim \mathcal{N}(0, \Sigma/N)$, and let $\lambda_{\max}(\Sigma)$ denote the largest eigenvalue of Σ. Define $B_N = \{\rho \in \mathcal{M}' \mid \mathrm{Tr}(\rho^2) \leq r^2\}$, where $r = \sqrt{\lambda_{\max}(\Sigma)}/N^{1/4}$. Then, $\lim_{N\to\infty} \Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N) = 1$.*

Proof. Let B_N^0 be an ellipsoidal ball defined by $\{\rho \in \mathcal{M}' \mid \mathrm{Tr}(\rho\Sigma^{-1}\rho) \leq 1/N^{1/2}\}$. Change coordinates by defining $\sigma = N^{1/2}\Sigma^{-1/2}\rho$. In these new coordinates $\hat{\sigma}_{\mathrm{ML},\mathcal{M}'} \sim$

$\mathcal{N}(0, \mathbb{1}_{d'})$, and $B_N^0 = \{\sigma \in \mathcal{M}' \mid \mathrm{Tr}(\sigma^2) \leq N^{1/2}\}$. Therefore,

$$\begin{aligned} \Pr(\hat{\sigma}_{\mathrm{ML},\mathcal{M}'} \in B_N^0) &= \Pr(\mathrm{Tr}(\hat{\sigma}^2_{\mathrm{ML},\mathcal{M}'}) \leq N^{1/2}) \\ &= \int_0^{N^{1/2}} \chi^2_{d'}(z)\ dz, \end{aligned} \tag{3.12}$$

because $\mathrm{Tr}(\hat{\sigma}^2_{\mathrm{ML},\mathcal{M}'})$ is a $\chi^2_{d'}$ random variable. It follows that

$$\lim_{N\to\infty} \Pr(\hat{\sigma}_{\mathrm{ML},\mathcal{M}'} \in B_N^0) = \int_0^{\infty} \chi^2_{d'}(z)\ dz = 1. \tag{3.13}$$

Switching back to the original coordinates, we have

$$B_N^0 = \{\rho \in \mathcal{M}' \mid \mathrm{Tr}(\rho\Sigma^{-1}\rho) \leq 1/N^{1/2}\}, \tag{3.14}$$

and $\lim_{N\to\infty} \Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N^0) = 1$.

Now that we know B_N^0 contains all $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ as $N \to \infty$, we can now show the same holds true for B_N. It suffices to show $B_N^0 \subset B_N$. To see that this is the case, write the equation for B_N^0 in the standard quadratic form for an ellipsoid:

$$B_N^0 = \{\rho \in \mathcal{M}' \mid \mathrm{Tr}(\rho(N^{1/2}\Sigma^{-1})\rho) \leq 1\}. \tag{3.15}$$

The standard ellipsoid $\{\mathbf{x} \mid \mathbf{x}^T A\mathbf{x} \leq 1\}$ has semi-major axes whose lengths s_j are related to the eigenvalues a_j of A: $s_j = 1/\sqrt{a_j}$. The matrix $A = N^{1/2}\Sigma^{-1}$ has eigenvalues $N^{1/2}/\lambda_j$, where λ_j are the eigenvalues of Σ. Thus, the lengths of the semi-major axes of B_N^0 are given by $s_j = 1/\sqrt{N^{1/2}/\lambda_j} = \sqrt{\lambda_j}/N^{1/4}$. Letting $\lambda_{\max}(\Sigma)$ denote the largest eigenvalue of Σ, the longest semi-major axis of B_N^0 has length $\sqrt{\lambda_{\max}(\Sigma)}/N^{1/4}$. Because B_N is a ball whose radius is equal to this length, B_N circumscribes B_N^0, and $B_N^0 \subset B_N$.

As $B_N^0 \subset B_N$, it follows from the monotonicity of probability that $\Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N^0) \leq \Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N)$. As the asymptotic limit of $\Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N^0)$ is 1, and $\Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N)$ itself is bounded above by 1, it follows from the squeeze theorem that $\lim_{N\to\infty} \Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N) = 1$. □

Informally speaking, Lemma 1 implies that as $N \to \infty$ "all the action" about $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ takes place inside B_N. Accordingly, to understand the behavior of quantities which depend on $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ (such as $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ and λ), it is sufficient to consider their behavior within B_N. In fact, asymptotically all the $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ are contained within the region $C_N \equiv B_N \cap \mathcal{M}$:

Lemma 2. $\lim_{N\to\infty} \Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}} \in C_N) = 1$.

Proof. Using the law of total probability, write $\Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}} \in C_N)$ as a sum of two terms, depending on whether $\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N$. Letting p denote $\Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N)$, we have

$$\begin{aligned}\Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}} \in C_N) &= p\Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}} \in C_N | \hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N) \\ &\quad + (1-p)\Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}} \in C_N | \hat{\rho}_{\mathrm{ML},\mathcal{M}'} \notin B_N) \\ &\geq p\Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}} \in C_N | \hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N).\end{aligned} \tag{3.16}$$

For any $\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N$, the corresponding $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ is somewhere in $\mathcal{M}$. To show $\hat{\rho}_{\mathrm{ML},\mathcal{M}} \in C_N$, we use a proof by contradiction. Suppose that $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ is the ML estimate in $\mathcal{M}$ for $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$, and that $\hat{\rho}_{\mathrm{ML},\mathcal{M}} \notin C_N$. Let ρ_C denote the closest point in C_N to $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$. Because $B_N \supset C_N$, it follows that ρ_C is closer to $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ than $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$, contradicting the assumption $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ was the ML estimate in $\mathcal{M}$ for $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$. Therefore, if $\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N$, it must be the case that $\hat{\rho}_{\mathrm{ML},\mathcal{M}} \in C_N$.

Consequently, $\Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}} \in C_N | \hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N) = 1$, implying $\Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}} \in C_N) \geq \Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in B_N)$. Applying the squeeze theorem, plus Lemma 1, we conclude $\lim_{N\to\infty} \Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}} \in C_N) = 1$. □

In the original coordinates, both B_N and the distribution of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ shrink with N, but B_N shrinks more slowly. Suppose, instead, that we switch to N-dependent coordinates that shrink with the distribution of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$. In these coordinates, $\mathcal{M}$ and $\mathcal{M}'$ grow with N, and B_N also grows (but more slowly). This *homothetic transformation*

of $\mathcal{M}$, $\mathcal{M}'$, and B_N scales all of them up. As $N \to \infty$, $B_N \to \mathbb{R}^{d'}$, and the local state space is the *solid tangent cone* of $\mathcal{M}$ at ρ_0.

Definition 3 (Homothetic Transformation)**.** *Given a convex set C, the homothetic transformation of C with respect to any point $X \in C$, with homothety coefficient h, is the set C_h defined by*

$$C_h = \{X + hY \mid \forall\ Y \in C, Y \neq X\}. \tag{3.17}$$

Definition 4 (Solid Tangent Cone)**.** *For each point X in a convex set C, let C_h be the homothetic transformation of C with respect to X, with homothety coefficient h. Then, the solid tangent cone $T(X)$ is defined as the following limit:*

$$T(X) = \lim_{h \to \infty} C_h. \tag{3.18}$$

Tangent cones are a general feature of convex sets; see (256), Chapter 6, Section A for more information about them and their properties.

Let $C_N = B_N \cap \mathcal{M}$ in Hilbert-Schmidt coordinates. I show that, in an N-dependent coordinate system, C_N converges to the solid tangent cone, and is the local state space.

Lemma 3. *Consider the set $C_N = B_N \cap \mathcal{M}$ in Hilbert-Schmidt coordinates, and define $C'_N = \{N^{1/2}\rho\ \forall\ \rho \in C_N\}$. Then:*

1) $\lim_{N\to\infty} C'_N$ *is the solid tangent cone at ρ_0.*

2) $\lim_{N\to\infty} C'_N$ *is the local state space.*

Proof.

1) By definition, C'_N is a homothetic transformation of C_N, with homothety coefficient $N^{1/2}$. (The homothetic center is ρ_0; in these coordinates, it is 0.) By definition, $\lim_{N\to\infty} C'_N$ is the solid tangent cone at ρ_0.

2) The original set C_N is a convex subset of $\mathcal{M}$, and from Lemma 2,

$$\lim_{N\to\infty} \Pr(\hat{\rho}_{\mathrm{ML},\mathcal{M}} \in C_N) = 1. \tag{3.19}$$

Further, the coordinate system defined by the mapping $\rho \to N^{1/2}\rho$ turns the (previously N-dependent) Fisher information $\mathcal{I}$ into a constant. Thus, $\lim_{N\to\infty} C'_N$ is the local state space.

□

Therefore, I have shown that, asymptotically, the local state space around ρ_0 *is* the solid tangent cone $T(\rho_0)$. The geometry of $T(\rho_0)$ depends strongly on ρ_0. If ρ_0 is rank-deficient within $\mathcal{M}$, then $T(\rho_0)$ is the cone whose faces touch $\mathcal{M}$ at ρ_0. (See Figure 3.4 for a rebit example.) However, if ρ_0 is full-rank, $T(\rho_0)$ is $\mathbb{R}^{d^2-1}$.

MLE as metric projection

As $N \to \infty$, all the $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ are contained within the ball B_N, and the local state space is the solid tangent cone. Because $\mathcal{M}'$ satisfies LAN, the likelihood function around each $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ is Gaussian, meaning the optimization problem defining $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ is given by

$$\hat{\rho}_{\mathrm{ML},\mathcal{M}} = \operatorname*{argmin}_{\rho\in T(\rho_0)} \mathrm{Tr}[(\rho - \hat{\rho}_{\mathrm{ML},\mathcal{M}'})\mathcal{I}(\rho - \hat{\rho}_{\mathrm{ML},\mathcal{M}'})]. \tag{3.20}$$

This equation shows that $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ is the *metric projection* of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ onto the tangent cone. See Figure 3.4 for a rebit example. (Notice that if $\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in T(\rho_0)$, then $\hat{\rho}_{\mathrm{ML},\mathcal{M}} = \hat{\rho}_{\mathrm{ML},\mathcal{M}'}$.) What makes this nontrivial is the replacement of the original state space $\mathcal{M}$, whose geometry can be arbitrarily complicated, with its tangent cone $T(\rho_0)$. As shown in the next section, cones can be much simpler and tractable.

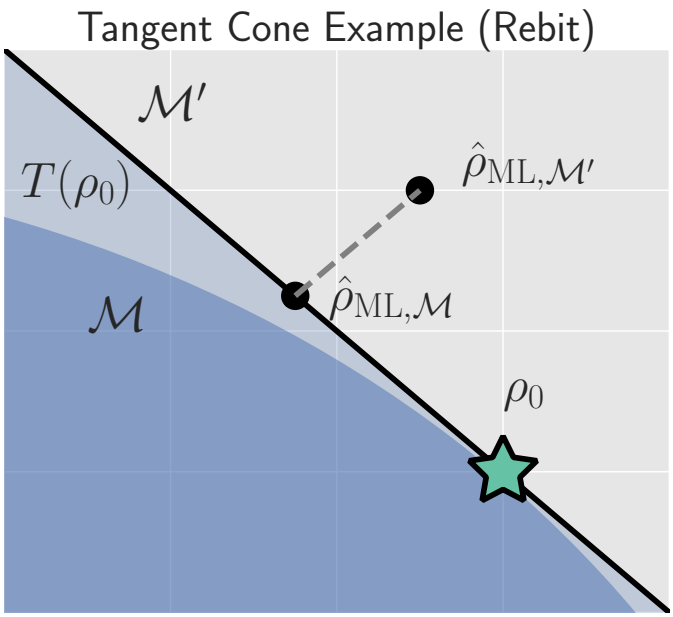

Figure 3.4: **Example of the solid tangent cone for a rebit**. As $N \to \infty$, the local state space around ρ_0 is $T(\rho_0)$. In Fisher-adjusted coordinates, it's easy to show that (a) $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ is the metric projection of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ onto $T(\rho_0)$, and (b) $\lambda(\rho_0, \mathcal{M}) = \mathrm{Tr}[(\hat{\rho}_{\mathrm{ML},\mathcal{M}} - \rho_0)^2]$.

Expression for $\lambda(\rho_0, \mathcal{M})$

The loglikelihood ratio statistic between any two models $\lambda(\mathcal{M}_1, \mathcal{M}_2)$ can be computed using a *reference model* $\mathcal{R}$:

$$\lambda(\mathcal{M}_1, \mathcal{M}_2) = \lambda(\mathcal{R}, \mathcal{M}_2) - \lambda(\mathcal{R}, \mathcal{M}_1), \tag{3.21}$$

where

$$\lambda(\mathcal{R}, \mathcal{M}) = -2 \log \left(\frac{\mathcal{L}(\mathcal{R})}{\mathcal{L}(\mathcal{M})} \right) = -2 \log \left(\frac{\max\limits_{\rho \in \mathcal{R}} \mathcal{L}(\rho)}{\max\limits_{\rho \in \mathcal{M}} \mathcal{L}(\rho)} \right). \tag{3.22}$$

Let us take $\mathcal{R} = \rho_0$. Because $\mathcal{M}'$ satisfies LAN, asymptotically $\mathcal{L}(\rho)$ is Gaussian, and λ relates to a difference in squared distances:

$$\begin{aligned}\lambda(\rho_0, \mathcal{M}) &= -2\log\left(\frac{\mathcal{L}(\rho_0)}{\max\limits_{\rho\in\mathcal{M}} \mathcal{L}(\rho)}\right) \\ &\xrightarrow[N\to\infty]{} \operatorname{Tr}[(\rho_0 - \hat{\rho}_{\mathrm{ML},\mathcal{M}'})\mathcal{I}(\rho_0 - \hat{\rho}_{\mathrm{ML},\mathcal{M}'})] \\ &\qquad - \operatorname{Tr}[(\hat{\rho}_{\mathrm{ML},\mathcal{M}} - \hat{\rho}_{\mathrm{ML},\mathcal{M}'})\mathcal{I}(\hat{\rho}_{\mathrm{ML},\mathcal{M}} - \hat{\rho}_{\mathrm{ML},\mathcal{M}'})].\end{aligned} \tag{3.23}$$

Using the fact $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ is a metric projection, $\lambda(\rho_0, \mathcal{M})$ has a simple form.

Lemma 4. $\lambda(\rho_0, \mathcal{M}) = \operatorname{Tr}[(\rho_0 - \hat{\rho}_{\mathrm{ML},\mathcal{M}})\mathcal{I}(\rho_0 - \hat{\rho}_{\mathrm{ML},\mathcal{M}})]$.

Proof. We switch to Fisher-adjusted coordinates ($\rho \to \mathcal{I}^{1/2}\rho$), and in these coordinates $\mathcal{I}$ becomes $\mathbb{1}$:

$$\lambda(\rho_0, \mathcal{M}) = \operatorname{Tr}[(\rho_0 - \hat{\rho}_{\mathrm{ML},\mathcal{M}'})^2] - \operatorname{Tr}[(\hat{\rho}_{\mathrm{ML},\mathcal{M}} - \hat{\rho}_{\mathrm{ML},\mathcal{M}'})^2]. \tag{3.24}$$

To prove the lemma, we must consider two cases:

Case 1: Assume $\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \notin T(\rho_0)$. Because $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ is the metric projection of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ onto $T(\rho_0)$ (Equation (3.20)), the line joining $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ and $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ is normal to $T(\rho_0)$ at $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$. Because $T(\rho_0)$ contains ρ_0 (as its origin), it follows that the lines joining ρ_0 to $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$, and $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ to $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$, are perpendicular (see Figure 3.4). By the Pythagorean theorem,

$$\operatorname{Tr}[(\rho_0 - \hat{\rho}_{\mathrm{ML},\mathcal{M}'})^2] = \operatorname{Tr}[(\rho_0 - \hat{\rho}_{\mathrm{ML},\mathcal{M}})^2] + \operatorname{Tr}[(\hat{\rho}_{\mathrm{ML},\mathcal{M}} - \hat{\rho}_{\mathrm{ML},\mathcal{M}'})^2]. \tag{3.25}$$

Subtracting $\operatorname{Tr}[(\hat{\rho}_{\mathrm{ML},\mathcal{M}} - \hat{\rho}_{\mathrm{ML},\mathcal{M}'})^2]$ from both sides, and comparing to Equation (3.24), yields the lemma statement in Fisher-adjusted coordinates.

Case 2: Assume $\hat{\rho}_{\mathrm{ML},\mathcal{M}'} \in T(\rho_0)$. Then, $\hat{\rho}_{\mathrm{ML},\mathcal{M}} = \hat{\rho}_{\mathrm{ML},\mathcal{M}'}$, and Equation (3.24) simplifies to the lemma statement in Fisher-adjusted coordinates.

Switching back from Fisher-adjusted coordinates yields

$$\lambda(\rho_0, \mathcal{M}) = \operatorname{Tr}[(\rho_0 - \hat{\rho}_{\mathrm{ML},\mathcal{M}})\mathcal{I}(\rho_0 - \hat{\rho}_{\mathrm{ML},\mathcal{M}})]. \tag{3.26}$$

□

So if $\mathcal{M}$ satisfies MP-LAN then as $N \to \infty$ the loglikelihood ratio statistic becomes related to *squared error/loss* (as measured by the Fisher information metric.) This result may be of independent interest in, for example, defining new information criteria, which attempt to balance goodness of fit (as measured by λ) against error/loss (generally, as measured by squared error).

Having defined a new generalization of LAN for models with convex constraints, I now turn to three specific *applications* of MP-LAN. The first derives a replacement for the Wilks theorem, the second derives an expression for the expected rank of ML estimates, and the third shows how our replacement for the Wilks theorem can be used to choose a Hilbert space dimension for an optical quantum system.

3.5 A Wilks theorem for quantum state space

To derive a replacement for the Wilks theorem, I start by showing the models $\mathcal{M}_d$ satisfy MP-LAN.

Lemma 5. *The models $\mathcal{M}_d$, defined in Equation* (3.7)*, satisfy MP-LAN.*

Proof. Let $\mathcal{M}'_d = \{\sigma \mid \dim(\sigma) = d, \sigma = \sigma^\dagger, \mathrm{Tr}(\sigma) = 1\}$. $\mathcal{M}'_d$ is the set of all trace-1, $d \times d$ Hermitian matrices, but they are not required to be non-negative. It is clear $\mathcal{M}_d \subset \mathcal{M}'_d$. Now, $\forall\ \sigma \in \mathcal{M}'_d$, the likelihood $\mathcal{L}(\sigma)$ is twice continuously differentiable, meaning $\mathcal{M}'_d$ satisfies LAN. Thus, $\mathcal{M}_d$ satisfies MP-LAN. □

The problem of computing $\lambda(\mathcal{M}_d, \mathcal{M}_{d+1})$ can be reduced to that of computing $\lambda(\rho_0, \mathcal{M}_k)$ for $k = d, d+1$ using the identity

$$\lambda(\mathcal{M}_d, \mathcal{M}_{d+1}) = \lambda(\rho_0, \mathcal{M}_{d+1}) - \lambda(\rho_0, \mathcal{M}_d), \tag{3.27}$$

where $\lambda(\rho_0, \mathcal{M}_k)$ is given in Equation (3.9). Because each model satisfies MP-LAN, asymptotically, $\lambda(\rho_0, \mathcal{M}_k)$ takes a very simple form, via Equation (3.10):

$$\lambda(\rho_0, \mathcal{M}_k) = \mathrm{Tr}[(\rho_0 - \hat{\rho}_{\mathrm{ML},\mathcal{M}_k})\mathcal{I}_k(\rho_0 - \hat{\rho}_{\mathrm{ML},\mathcal{M}_k})]. \tag{3.28}$$

The Fisher information $\mathcal{I}_k$ is generally anisotropic, depending on ρ_0, the POVM being measured, and the model $\mathcal{M}_k$ (see Figure 3.5). And while the $\rho \geq 0$ constraint that invalidated LAN in the first place is at least somewhat tractable in standard (Hilbert-Schmidt) coordinates, it becomes completely intractable in Fisher-adjusted coordinates. So, to obtain a semi-analytic null theory for λ, I will simplify to the case where $\mathcal{I}_k = \mathbb{1}_k/\epsilon^2$ for some ϵ that scales as $1/\sqrt{N_{\mathrm{samples}}}$. (That is, $\mathcal{I}_k$ is proportional to the Hilbert-Schmidt metric.) This simplification permits the derivation of analytic results that capture realistic tomographic scenarios surprisingly well (283).

With this simplification, $\lambda(\mathcal{M}_d, \mathcal{M}_{d+1})$ is given by

$$\lambda = \frac{1}{\epsilon^2}\left(\mathrm{Tr}[(\rho_0 - \hat{\rho}_{\mathrm{ML},d+1})^2] - \mathrm{Tr}[(\rho_0 - \hat{\rho}_{\mathrm{ML},d})^2]\right). \tag{3.29}$$

That is, λ is a *difference* in Hilbert-Schmidt distances. This expression makes it clear why a null theory for λ is necessary: if $\rho_0 \in \mathcal{M}_d, \mathcal{M}_{d+1}$, then $\hat{\rho}_{\mathrm{ML},d+1}$ will lie further from ρ_0 than $\hat{\rho}_{\mathrm{ML},d}$ (because there are more parameters that can fit noise in the data). The null theory for λ tells us how much extra error will be incurred in using $\mathcal{M}_{d+1}$ to reconstruct ρ_0 when $\mathcal{M}_d$ is just as good.

Describing $\Pr(\lambda)$ is difficult because the distributions of $\hat{\rho}_{\mathrm{ML},d}, \hat{\rho}_{\mathrm{ML},d+1}$ are complicated, highly non-Gaussian, and singular (estimates "pile up" on the various faces of the boundary as shown in Figure 3.1). For this reason, I will not attempt to compute $\Pr(\lambda)$ directly. Instead, I focus on deriving a good approximation for $\langle\lambda\rangle$.

I consider each of the terms in Equation (3.29) separately and focus on computing $\epsilon^2\langle\lambda(\rho_0, \mathcal{M}_d)\rangle = \langle\mathrm{Tr}[(\hat{\rho}_{\mathrm{ML},d} - \rho_0)^2]\rangle$ for arbitrary d. Doing so involves two main steps:

(1) Identify which degrees of freedom in $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$ are, and are not, affected by projection onto the tangent cone $T(\rho_0)$.

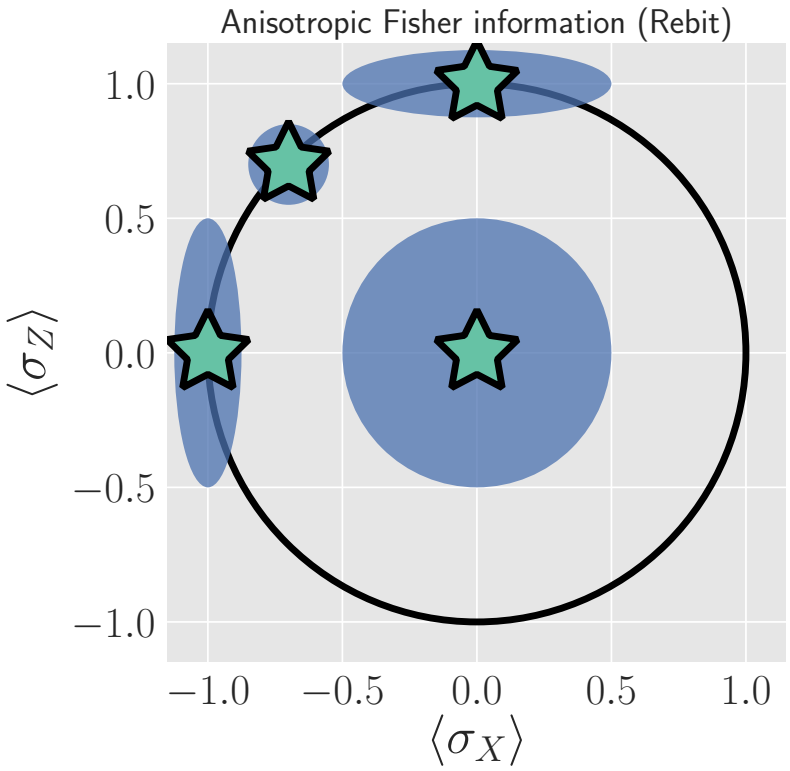

Figure 3.5: **Anisotropy of the Fisher information for a rebit** Suppose a rebit state ρ_0 (star) is measured using the POVM $\frac{1}{2}\{|0\rangle\langle 0|, |1\rangle\langle 1|, |+\rangle\langle +|, |-\rangle\langle -|\}$. Depending on ρ_0, the distribution of the unconstrained estimates $\hat{\rho}_{\mathrm{ML}}$ (ellipses) may be anisotropic. Imposing the positivity constraint $\rho \geq 0$ is difficult in Fisher-adjusted coordinates; in this chapter, these complexities are simplified to the case where $\mathcal{I} \propto \mathbb{1}$, *and is independent of* ρ_0.

(2) For each of those categories, evaluate its contribution to the value of $\langle\lambda\rangle$.

In Section 3.5.1, I identify two types of degrees of freedom in $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$, called the "L" and the "kite". Section 3.5.2 computes the contribution of degrees of freedom in the "L", and Section 3.5.3 computes the contribution from the "kite". The total expected value is given in Equation (3.43) in Section 3.5.4, on page 71.

3.5.1 Separating out degrees of freedom in $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$

We begin by observing that $\lambda(\rho_0, \mathcal{M}_d)$ can be written as a sum over matrix elements,

$$\begin{aligned}\lambda &= \epsilon^{-2}\mathrm{Tr}[(\hat{\rho}_{\mathrm{ML},d} - \rho_0)^2] = \epsilon^{-2}\sum_{jk}|(\hat{\rho}_{\mathrm{ML},d} - \rho_0)_{jk}|^2 \\ &= \sum_{jk}\lambda_{jk} \quad \text{where} \quad \lambda_{jk} = \epsilon^{-2}|(\hat{\rho}_{\mathrm{ML},d} - \rho_0)_{jk}|^2,\end{aligned} \tag{3.30}$$

and therefore $\langle\lambda\rangle = \sum_{jk}\langle\lambda_{jk}\rangle$. Each term $\langle\lambda_{jk}\rangle$ quantifies the mean-squared error of a single matrix element of $\hat{\rho}_{\mathrm{ML},d}$, and while the Wilks theorem predicts $\langle\lambda_{jk}\rangle = 1$ for all j, k, due to positivity constraints, this no longer holds. In particular, the matrix elements of $\hat{\rho}_{\mathrm{ML},d}$ now fall into two parts:

1. Those for which the positivity constraint *does affect* their behavior.
2. Those for which the positivity constraint *does not affect* their behavior, as they correspond to directions on the surface of the tangent cone $T(\rho_0)$. (Recall Figure 3.4 - as a component of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'}$ along $T(\rho_0)$ changes, the component of $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$ changes by the same amount. These elements are unconstrained.)

The latter, which lie in what I call the "L", comprise all off-diagonal elements on the support of ρ_0 *and* between the support and the kernel, while the former, which lie in what I call the "kite", are all diagonal elements *and* all elements on the kernel (null space) of ρ_0.

Performing this division is also supported by numerical simulations (see Figure 3.6). Matrix elements in the "L" appear to contribute $\langle\lambda_{jk}\rangle = 1$, consistent with the Wilks theorem, while those in the "kite" contribute more (if they are within the support of ρ_0) or less (if they are in the kernel). Having performed the division of the matrix elements of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$, observe that $\langle\lambda\rangle = \langle\lambda_{\mathrm{L}}\rangle + \langle\lambda_{\mathrm{kite}}\rangle$. Because each $\langle\lambda_{jk}\rangle$ is not necessarily equal to one (as in the Wilks theorem), and because many of them are less than 1, it is clear that their total $\langle\lambda\rangle$ is dramatically lower than the prediction of the Wilks theorem. (Recall Figure 3.2.)

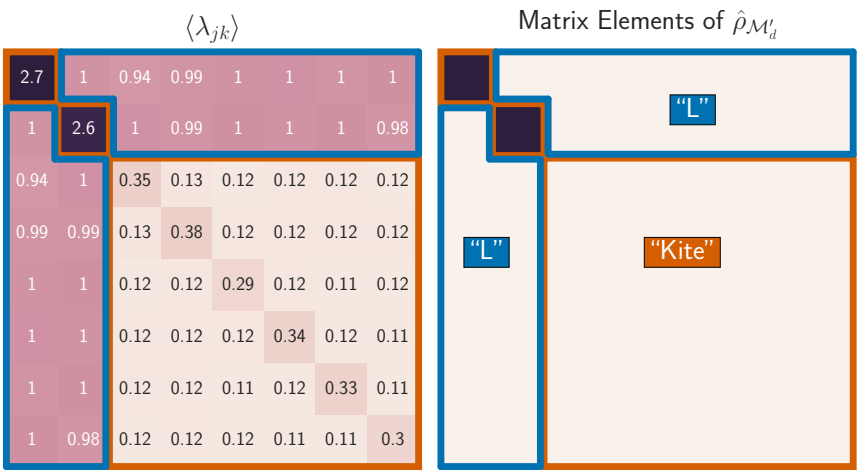

Figure 3.6: **Division of the matrix elements of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$.** When a rank-2 state is reconstructed in $d = 8$ dimensions, the total loglikelihood ratio $\lambda(\rho_0, \mathcal{M}_8)$ is the sum of terms λ_{jk} from errors in each matrix element $(\hat{\rho}_{\mathrm{ML},d})_{jk}$. **Left**: Numerics show a clear division; some matrix elements have $\langle\lambda_{jk}\rangle \sim 1$ as predicted by the Wilks theorem, while others are either more or less. **Right**: The numerical results support our theoretical reasoning for dividing the matrix elements of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$ into two parts: the "kite" and the "L".

In the following subsections, I develop a theory to explain the behavior of $\langle\lambda_{\mathrm{L}}\rangle$ and $\langle\lambda_{\mathrm{kite}}\rangle$. In doing so, it is helpful to think about the matrix $\delta \equiv \hat{\rho}_{\mathrm{ML},\mathcal{M}'_d} - \rho_0$, a normally-distributed *traceless* matrix. To simplify the analysis, I explicitly drop the $\mathrm{Tr}(\delta) = 0$ constraint and let δ be $\mathcal{N}(0, \epsilon^2 \mathbb{1})$ distributed over the d^2-dimensional space of Hermitian matrices (a good approximation when $d \gg 2$), which makes δ proportional to an element of the Gaussian Unitary Ensemble (GUE) (112).

3.5.2 Computing contributions from the "L"

The value of each δ_{jk} in the "L" is invariant under projection onto the boundary (the surface of the tangent cone $T(\rho_0)$), meaning that it is also equal to the error $(\hat{\rho}_{\mathrm{ML},d} - \rho_0)_{jk}$. Therefore, $\langle\lambda_{jk}\rangle = \langle\delta_{jk}^2\rangle/\epsilon^2$. Because $\mathcal{M}'$ satisfies LAN, it follows that each δ_{jk} is an i.i.d. Gaussian random variable with mean zero and variance ϵ^2. Thus, $\langle\lambda_{jk}\rangle = 1 \ \forall \ (j,k)$ in the "L". The dimension of the surface of the tangent cone is

equal to the dimension of the manifold of rank-r states in a d-dimensional space. A direct calculation of that quantity yields $2rd - r(r+1)$, so $\langle \lambda_{\mathrm{L}} \rangle = 2rd - r(r+1)$.

Another way of obtaining this result is to view the δ_{jk} in the "L" as errors arising due to small unitary perturbations of ρ_0. Writing $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d} = U^\dagger \rho_0 U$, where $U = e^{i\epsilon H}$, we have

$$\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d} \approx \rho_0 + i\epsilon[\rho_0, H] + \mathcal{O}(\epsilon^2), \tag{3.31}$$

and $\delta \approx i\epsilon[\rho_0, H]$. If $j = k$, then $\delta_{jj} = 0$. Thus, small unitaries cannot create errors in the diagonal matrix elements, at $\mathcal{O}(\epsilon)$. If $j \neq k$, then $\delta_{jk} \neq 0$, in general. Small unitaries *can* introduce errors in off-diagonal elements.

However, if either j or k (or both) lie within the *kernel* of ρ_0 (i.e., $\langle k|\rho_0|k\rangle$ or $\langle j|\rho_0|j\rangle$ is 0), then the corresponding δ_{jk} are zero. The only off-diagonal elements where small unitaries can introduce errors are those which are coherent between the kernel of ρ_0 and its support. These off-diagonal elements are precisely the "L", and are the set $\{\delta_{jk} \mid \langle j|\rho_0|j\rangle \neq 0, j \neq k,\ 0 \leq j,k \leq d-1\}$. This set contains $2rd - r(r+1)$ elements, each of which has $\langle \lambda_{jk} \rangle = 1$, so we again arrive at $\langle \lambda_{\mathrm{L}} \rangle = 2rd - r(r+1)$.

3.5.3 Computing contributions from the "kite"

Computing $\langle \lambda_{\mathrm{L}} \rangle$ was made easy by the fact that the matrix elements of δ in the "L" are invariant under the projection of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$ onto $T(\rho_0)$. Computing $\langle \lambda_{\mathrm{kite}} \rangle$ is a bit harder, because the boundary *does* constrain δ. To understand how the behavior of $\langle \lambda_{\mathrm{kite}} \rangle$ is affected, I analyze an algorithm presented in (283) for explicitly solving the optimization problem in Equation (3.8).

This algorithm, a (very fast) numerical method for computing $\hat{\rho}_{\mathrm{ML},d}$ given $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$, utilizes two steps:

1. Subtract $q\mathbb{1}$ from $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$, for a particular $q \in \mathbb{R}$.
2. "Truncate" $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d} - q\mathbb{1}$, by replacing each of its negative eigenvalues with zero.

Here, q is defined implicitly such that $\mathrm{Tr}\left[\mathrm{Trunc}(\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d} - q\mathbb{1})\right] = 1$, and must be determined numerically. However, we can analyze how this algorithm affects the eigenvalues of $\hat{\rho}_{\mathrm{ML},d}$, which turn out to be the key quantity necessary for computing $\langle\lambda_{\mathrm{kite}}\rangle$.

The truncation algorithm above is most naturally performed in the eigenbasis of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$. Exact diagonalization of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$ is not feasible analytically, but only its *small* eigenvalues are critical in truncation. Further, only knowledge of the *typical* eigenvalues of $\hat{\rho}_{\mathrm{ML},d}$ is necessary for computing $\langle\lambda_{\mathrm{kite}}\rangle$. Therefore, we do not need to determine $\hat{\rho}_{\mathrm{ML},d}$ exactly, which would require explicitly solving Equation (3.8) using the algorithm presented in (283); instead, a procedure for determining its typical eigenvalues is all that is required.

I assume that N_{samples} is sufficiently large so that all the nonzero eigenvalues of ρ_0 are much larger than ϵ. This means the eigenbasis of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$ is accurately approximated by: (1) the eigenvectors of ρ_0 on its support; and (2) the eigenvectors of $\delta_{\mathrm{ker}} = \Pi_{\mathrm{ker}}\delta\Pi_{\mathrm{ker}} = \Pi_{\mathrm{ker}}\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}\Pi_{\mathrm{ker}}$, where Π_{ker} is the projector onto the kernel of ρ_0.

Changing to this basis diagonalizes the "kite" portion of δ, and leaves all elements of the "L" unchanged (at $\mathcal{O}(\epsilon)$). The diagonal elements fall into two categories:

1. r elements corresponding to the eigenvalues of ρ_0, which are given by $p_j = \rho_{jj} + \delta_{jj}$ where ρ_{jj} is the j^{th} eigenvalue of ρ_0, and $\delta_{jj} \sim \mathcal{N}(0, \epsilon^2)$.
2. $n \equiv d - r$ elements that are eigenvalues of δ_{ker}, which I denote by $\boldsymbol{\kappa} = \{\kappa_j : j = 1, \ldots, n\}$.

In turn, q is the solution to

$$\sum_{j=1}^{r}(p_j - q)^+ + \sum_{j=1}^{n}(\kappa_j - q)^+ = 1, \tag{3.32}$$

where $(x)^+ = \max(x, 0)$, and λ_{kite} is

$$\epsilon^2\lambda_{\mathrm{kite}} = \sum_{j=1}^{r}[\rho_{jj} - (p_j - q)^+]^2 + \sum_{j=1}^{n}\left[(\kappa_j - q)^+\right]^2. \tag{3.33}$$

To solve Equation (3.32), and derive an approximation for (3.33), I use the fact that I am interested in computing the *average value* of λ_{kite}, which justifies approximating the random variable q by a closed-form, deterministic value. To do so, we need to understand the behavior of $\boldsymbol{\kappa}$. Developing such an understanding, and a theory of its *typical value*, is the subject of the next section.

Approximating the eigenvalues of a GUE(n) matrix

Observe that while the κ_j are random variables, they are *not* normally distributed. Instead, because δ_{ker} is proportional to a GUE(n) matrix, for $n \gg 1$, the distribution of any eigenvalue κ_j converges to a Wigner semicircle distribution (322), given by $\Pr(\kappa) = \frac{2}{\pi R^2}\sqrt{R^2 - \kappa^2}$ for $|\kappa| \leq R$, with $R = 2\epsilon\sqrt{n}$. The eigenvalues are not independent; they tend to avoid collisions ("level avoidance" (292)), and typically form a surprisingly regular array over the support of the Wigner semicircle. Since the goal is to compute $\langle\lambda_{\text{kite}}\rangle$, we can capitalize on this behavior by replacing each random sample of $\boldsymbol{\kappa}$ with a *typical sample* given by its order statistics $\bar{\boldsymbol{\kappa}}$. These are the average values of the *sorted* $\boldsymbol{\kappa}$, so $\overline{\kappa}_j$ is the average value of the j^{th} largest value of $\boldsymbol{\kappa}$. Large random samples are usually well approximated (for many purposes) by their order statistics even when the elements of the sample are independent, and level avoidance makes the approximation even better.

Suppose that $\boldsymbol{\kappa}$ are the eigenvalues of a GUE(n) matrix, sorted from highest to lowest. Figure 3.7 illustrates such a sample for $n = 100$. It also shows the *average* values of 100 such samples (all sorted). These are the *order statistics* $\overline{\boldsymbol{\kappa}}$ of the distribution (more precisely, what is shown is a good *estimate* of the order statistics; the actual order statistics would be given by the average over infinitely many samples). As the figure shows, while the order statistics *are* slightly more smoothly and predictably distributed than a single (sorted) sample, the two are remarkably similar. A single sample $\boldsymbol{\kappa}$ will fluctuate around the order statistics, but these fluctuations are relatively small, partly because the sample is large, and partly because the GUE

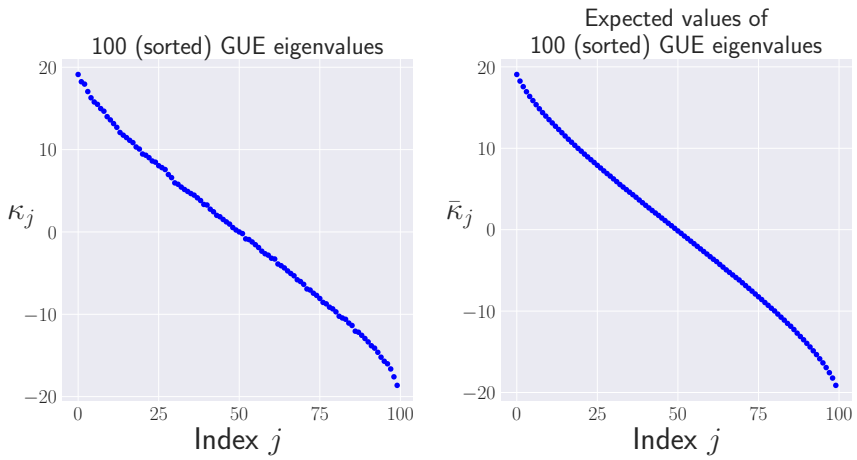

Figure 3.7: **Approximating typical samples of GUE(n) eigenvalues by order statistics**. I approximate a typical sample of GUE(n) eigenvalues by their order statistics (average values of a sorted sample). **Left**: The sorted eigenvalues (i.e., order statistics κ_j) of one randomly chosen GUE(100) matrix. **Right**: Approximate expected values of the order statistics, $\bar{\kappa}_j$, of the GUE(100) distribution, computed as the average of the sorted eigenvalues of 100 randomly chosen GUE(100) matrices.

eigenvalues experience level repulsion. Thus, the "typical" behavior of a sample – by which I mean the mean value of a statistic of the sample – is well captured by the order statistics (which have no fluctuations at all).

I now turn to the problem of modeling $\boldsymbol{\kappa}$ quantitatively. I note up front that later, we are only going to be interested in certain properties of $\boldsymbol{\kappa}$: specifically, partial sums of all κ_j greater or less than the threshold q, or partial sums of functions of the κ_j (e.g., $(\kappa_j - q)^2$). I require only that an ansatz be accurate for such quantities. I do not use this fact explicitly, but it motivates the approach – and I do not claim that the ansatz is accurate for *all* conceivable functions.

In general, if a sample $\boldsymbol{\kappa}$ of size n is drawn so that each κ has the same probability density function $\Pr(\kappa)$, then a good approximation for the j^{th} order statistic is given

by the inverse *cumulative* distribution function (CDF):

$$\overline{\kappa}_j \approx \mathrm{CDF}^{-1}\left(\frac{j-1/2}{n}\right). \tag{3.34}$$

This is closely related to the observation that the histogram of a sample tends to look similar to the underlying probability density function. More precisely, it is equivalent to the observation that the empirical distribution function (the CDF of the histogram) tends to be (even more) similar to the underlying CDF. For i.i.d. samples, this is the content of the Glivenko-Cantelli theorem (302). Figure 3.8 compares the order statistics of GUE(100) and GUE(10) eigenvalues (computed as numerical averages over 100 random samples) to the inverse CDF for the Wigner semicircle distribution. Even though the Wigner semicircle model of GUE eigenvalues is only exact as $n \to \infty$, it provides a nearly-perfect model for $\overline{\boldsymbol{\kappa}}$ even at $n = 10$ (and remains surprisingly good all the way down to $n = 2$).

I make one further approximation, by assuming that $n \gg 1$, so the distribution of the $\overline{\kappa}_j$ is effectively continuous and identical to $\Pr(\kappa)$. For the quantities computed, this is equivalent to replacing the empirical distribution function (which is a step function) by the CDF of the Wigner semicircle distribution. So, whereas for any given sample the partial sum of all $\kappa_j > q$ jumps discontinuously when $q = \kappa_j$ for any j, in this approximation it changes smoothly. This accurately models the *average* behavior of partial sums.

Deriving an approximation for q

The approximations of the previous section allow us to use $\{p_j\} \cup \{\overline{\kappa}_j\}$ as the ansatz for the eigenvalues of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$, where the p_j are $\mathcal{N}(\rho_{jj}, \epsilon^2)$ random variables, and the $\overline{\kappa}_j$ are the (fixed, smoothed) order statistics of a Wigner semicircle distribution. In turn, the defining equation for q (Equation (3.32)) is well approximated as

$$\sum_{j=1}^{r}(p_j - q)^+ + \sum_{j=1}^{n}(\overline{\kappa}_j - q)^+ = 1. \tag{3.35}$$

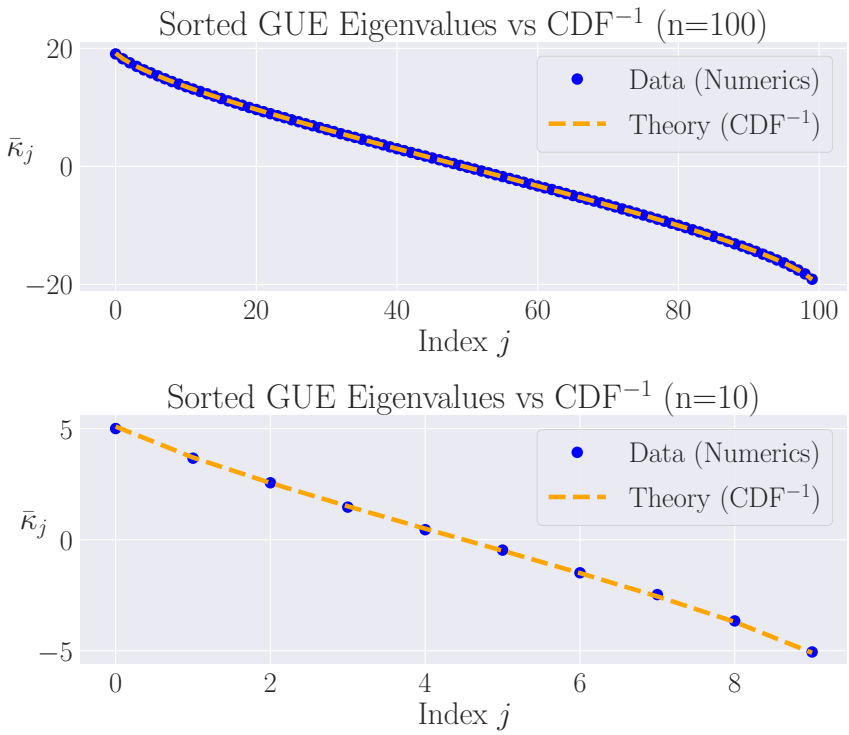

Figure 3.8: **Approximating order statistics by the inverse CDF** Order statistics of the GUE(n) eigenvalue distribution are very well approximated by the inverse CDF of the Wigner semicircle distribution. In both figures, I compare the order statistics of a GUE(n) distribution to the inverse CDF of the Wigner semicircle distribution. **Top**: $n = 100$. **Bottom**: $n = 10$. Agreement in both cases is essentially perfect.

Because the $\overline{\kappa}_j$ are symmetrically distributed around $\kappa = 0$, half of them are negative. Therefore, with high probability, $\mathrm{Tr}\left[\mathrm{Trunc}(\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d})\right] > 1$, and so $q\mathbb{1}$ will need to be subtracted from $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$ before truncating.

Because I have assumed N_{samples} is sufficiently large ($N_{\mathrm{samples}} \gg \min_j 1/\rho_{jj}^2$), the eigenvalues of ρ_0 are large compared to the perturbations δ_{jj} and q. This implies

$(p_j - q)^+ = p_j - q$. Under this assumption, q is the solution to

$$\sum_{j=1}^{r}(p_j - q) + \sum_{j=1}^{n}(\overline{\kappa}_j - q)^+ = 1 \tag{3.36}$$

$$\implies -rq + \Delta + n\int_{\kappa=q}^{2\epsilon\sqrt{n}}(\kappa - q)\Pr(\kappa)\mathrm{d}\kappa = 0 \tag{3.37}$$

$$\implies -rq + \frac{\epsilon}{12\pi}\left[(q^2+8n)\sqrt{-q^2+4n} - 12qn\left(\frac{\pi}{2} - \sin^{-1}\left(\frac{q}{2\sqrt{n}}\right)\right)\right] = 0. \tag{3.38}$$

There are several simplifications I use in the derivation of Equation (3.38) from Equation (3.36). First, I choose to replace a discrete sum in Equation (3.36) with an integral. This approximation is valid when $n \gg 1$, as I can accurately approximate a discrete collection of closely spaced real numbers by a smooth density or distribution over the real numbers that has approximately the same CDF. It is also remarkably accurate in practice. The quantity $\Delta = \sum_{j=1}^{r}\delta_{jj}$ in Equation (3.37) is a $\mathcal{N}(0, r\epsilon^2)$ random variable. In yet another approximation, I replace Δ with its average value, which is zero. An even more accurate expression could be obtained by treating Δ more carefully, but this crude approximation turns out to be quite accurate already.

To solve Equation (3.38), it is necessary to further simplify the complicated expression resulting from the integral in Equation (3.37) (the bracketed term in Equation (3.38)). To do so, I assume ρ_0 is relatively low-rank, so $r \ll d/2$. In this case, the sum of the positive $\overline{\kappa}_j$ is large compared with r, almost all of them need to be subtracted away, and therefore q is close to $2\epsilon\sqrt{n}$. I therefore replace the complicated expression with its leading order Taylor expansion around $q = 2\epsilon\sqrt{n}$, substitute into Equation (3.38), and obtain the equation

$$\frac{rq}{\epsilon} = \frac{4}{15\pi}n^{1/4}\left(2\sqrt{n} - \frac{q}{\epsilon}\right)^{5/2}. \tag{3.39}$$

This equation is a quintic polynomial in q/ϵ, so by the Abel-Ruffini theorem, it has no algebraic solution. However, as $n \to \infty$, its roots have a well-defined algebraic

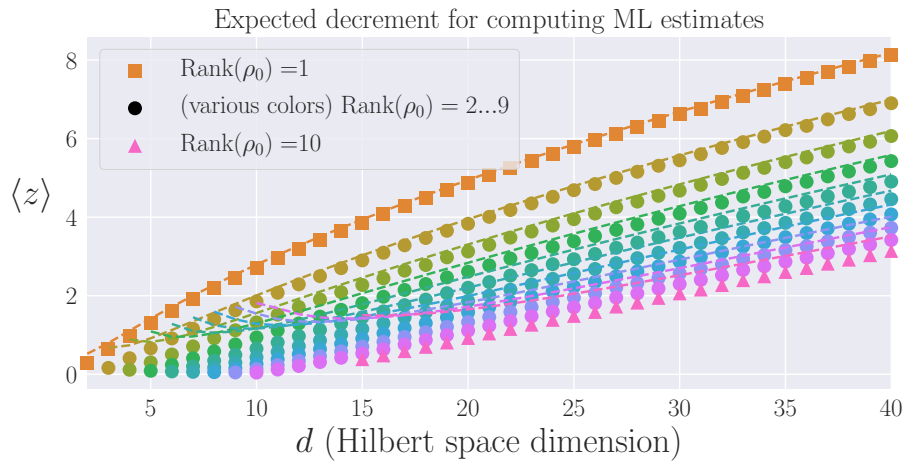

Figure 3.9: **Comparing theory for z to numerical results**. The formula for z given in Equation (3.40) agrees well with numerics, provided $r \ll d$. (Simulations use $\epsilon = 10^{-4}$).

approximation[4] that becomes accurate quite rapidly (e.g., for $n > 4$):

$$z \equiv q/\epsilon \approx 2\sqrt{n}\left(1 - \frac{1}{2}x + \frac{1}{10}x^2 - \frac{1}{200}x^3\right), \tag{3.40}$$

where

$$x = \left(\frac{15\pi r}{2n}\right)^{2/5}. \tag{3.41}$$

Figure 3.9 compares this expression for z to numerical results. Once $r \ll d$, the agreement is very good, although it suffers quite dramatically as $r \to d$.

[4]See Appendix B for a derivation of this solution.

Expression for $\langle\lambda_{\text{kite}}\rangle$

Now that we know how much to subtract off in the truncation process, we can approximate $\langle\lambda_{\text{kite}}\rangle$, originally given in Equation (3.33):

$$\begin{aligned}
\langle\lambda_{\text{kite}}\rangle &\approx \frac{1}{\epsilon^2}\left\langle \sum_{j=1}^{r}[\rho_{jj} - (p_j - q)^+]^2 + \sum_{j=1}^{n}\left[(\bar{\kappa}_j - q)^+\right]^2 \right\rangle \\
&\approx \frac{1}{\epsilon^2}\left\langle \sum_{j=1}^{r}[-\delta_{jj} + q]^2 + \sum_{j=1}^{n}\left[(\bar{\kappa}_j - q)^+\right]^2 \right\rangle \\
&\approx r + rz^2 + \frac{n}{\epsilon^2}\int_{\kappa=q}^{2\epsilon\sqrt{n}} \Pr(\kappa)(\kappa - q)^2 d\kappa \\
&= r + rz^2 + \frac{n(n+z^2)}{\pi}\left(\frac{\pi}{2} - \sin^{-1}\left(\frac{z}{2\sqrt{n}}\right)\right) \\
&\quad - \frac{z(z^2+26n)}{24\pi}\sqrt{4n - z^2}.
\end{aligned} \tag{3.42}$$

3.5.4 Complete expression for $\langle\lambda\rangle$

The total expected value, $\langle\lambda\rangle = \langle\lambda_{\text{L}}\rangle + \langle\lambda_{\text{kite}}\rangle$, is thus

$$\begin{aligned}
\langle\lambda(\rho_0, \mathcal{M}_d)\rangle \approx\ & 2rd - r^2 + rz^2 \\
& + \frac{n(n+z^2)}{\pi}\left(\frac{\pi}{2} - \sin^{-1}\left(\frac{z}{2\sqrt{n}}\right)\right) \\
& - \frac{z(z^2+26n)}{24\pi}\sqrt{4n - z^2}.
\end{aligned} \tag{3.43}$$

where z is given in Equation (3.40), $n = d - r$, and $r = \text{Rank}(\rho_0)$.

This null theory is much more complicated than the Wilks theorem, but as Figure 3.10 shows, it is very accurate when $2r \ll d$. (In contrast, the prediction of the Wilks theorem is wildly incorrect for $r \ll d$.) Although our null theory does break down as $r \to d$, it does so fairly gracefully. I conclude that our analysis (and Equation (3.43)) correctly models tomography *if* the Fisher information is isotropic ($\mathcal{I} \propto \mathbb{1}$), a point I turn to in the next subsection and in Section 4.2.

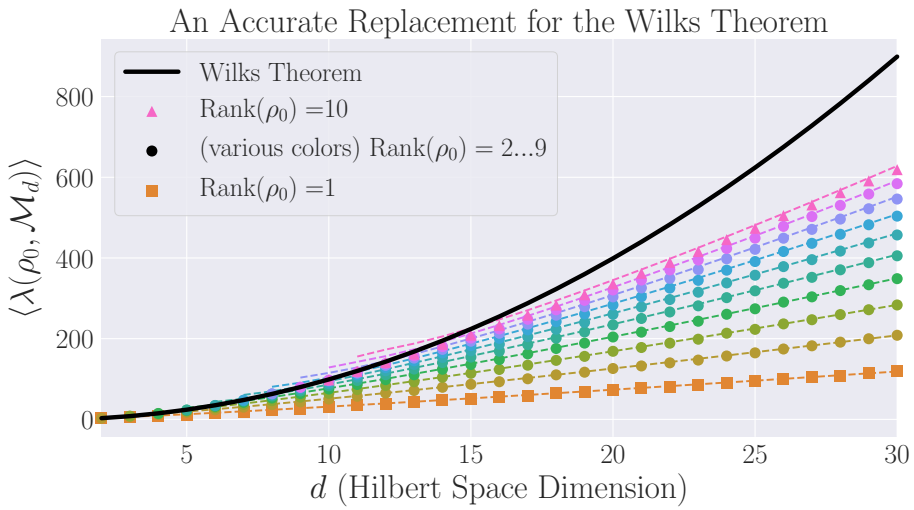

Figure 3.10: **Improved prediction for $\langle\lambda(\rho_0, \mathcal{M}_d)\rangle$, as compared to the Wilks theorem**. Numerical results for $\langle\lambda(\rho_0, \mathcal{M}_d)\rangle$ compared to the prediction of the Wilks theorem (solid line) and our replacement theory as given in Equation (3.43) (dashed lines). Our formula depends on the rank r of ρ_0 (unlike the Wilks prediction), and is nearly perfect for $r \ll d/2$. It becomes less accurate as r approaches $d/2$, and is invalid when $r \approx d$.

In the asymptotic limit $d \to \infty$, while keeping r fixed, $\langle\lambda\rangle$ takes the following form:

$$\langle\lambda\rangle \underset{d\to\infty}{\longrightarrow} rd\left[6 - \frac{20}{7}\left(\frac{15\pi r}{2d}\right)^{2/5} + \frac{20}{21}\left(\frac{15\pi r}{2d}\right)^{4/5}\right] - 5r^2. \tag{3.44}$$

That $\langle\lambda\rangle$ scales as $\mathcal{O}(rd)$ in this regime is to be expected, as a rank-r density matrix has $\mathcal{O}(rd)$ free parameters. Curiously though, this asymptotic result is not equal to $\langle\lambda_{\mathrm{L}}\rangle$, meaning that the "kite" elements continue to contribute to the behavior of the statistic, even though most of them will be set to zero in the projection step when computing $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$.

3.5.5 Comparison to idealized tomography

To evaluate this null theory for $\langle\lambda\rangle$, I compare it to numerical experiments, described below. The derivation of Equation (3.43) assumed both that the Fisher information

is isotropic and that the number of samples is asymptotically infinite. To simulate the expected value of λ under both these assumptions, I chose a variety of true states ρ_0 with dimension $d = 2, \ldots, 30$ and rank $r = 1, \ldots, 10$ and: (a) generated $N = 500$ i.i.d. $\mathcal{N}(\rho_0, \epsilon^2 \mathcal{I})$ unconstrained ML estimates $\{\hat{\rho}_{\mathrm{ML}, \mathcal{M}'_{d,j}}\}_{j=1}^{N}$, thereby simulating the unconstrained ML estimates *at* the $N_{\mathrm{samples}} = \infty$ limit, (b) numerically solved Equation (3.20) for each $\hat{\rho}_{\mathrm{ML}, \mathcal{M}'_{d,j}}$ to obtain the constrained ML estimate $\hat{\rho}_{\mathrm{ML}, \mathcal{M}_{d,j}}$, and (c) estimated $\langle \lambda \rangle$ as $\frac{1}{N} \sum_{j=1}^{N} \mathrm{Tr}[(\rho_0 - \hat{\rho}_{\mathrm{ML}, \mathcal{M}_{d,j}})^2]/\epsilon^2$. I took $\epsilon = 10^{-4}$, to ensure that all of the unconstrained ML estimates are close to ρ_0, and that I was not erroneously generating estimates which are too far away. Recall that the derivation used the fact that, asymptotically, we can "zoom in" on ρ_0 to understand the behavior of λ. Consequently, if ϵ is too large, then some of the unconstrained ML estimates may almost be orthogonal to ρ_0, which clearly violates the conditions used in the derivation.

Figure 3.10 compares the theory (dashed lines) to these numerical results (solid dots). It is clear Equation (3.43) is almost perfectly accurate when $r \ll d/2$, but it does begin to break down as r becomes comparable to d. Therefore, we can have confidence that the theory – although derived in a highly-idealized scenario – is more-or-less correct within that scenario.

3.6 Conclusion and discussion

Quantum state space violates local asymptotic normality, a key property satisfied by classical statistical models. Through the introduction of metric-projected local asymptotic normality (MP-LAN), I have provided a new framework for reasoning about results in classical statistical model selection for models that don't satisfy LAN because of convex constraints.

This chapter explicitly investigated one such result, the Wilks theorem, found it is not generally reliable in quantum state tomography, and provided a much more

broadly applicable replacement that can be used in model selection methods (Equation (3.43)). While Equation (3.43) is an *approximate* expression for $\langle\lambda\rangle$, further refinements to the approximation (such as considering the Δ-fluctuations in the ML estimate (Equation (3.36)), or using higher-order polynomial expansions of Equations (3.36) and (3.39)) would yield an even better theory.

Note that the Wilks theorem is rigorously proven under certain assumptions. Proving a quantum version of the Wilks theorem would be difficult due to the fact that λ depends non-trivially on the distribution of $\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}$. However, one way to go about proving (instead of simply approximating) facts about λ would be to recognize that λ is a "chi-bar-squared" random variable (21; 20; 19), and understand how the structure of that distribution depends on the distribution of $\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}$. (A chi-bar-squared random variable is distributed according to a *mixture* of χ^2 random variables with different degrees of freedom.)

MP-LAN is also applicable to information criteria such as the AIC and BIC (5; 270; 166; 50) that do not explicitly use the Wilks theorem, but rely on the same assumptions (local asymptotic normality, equivalence between loglikelihood and squared error, etc.). Information criteria balance how well a model fits the data against how much the model *overfits* the data; usually, the fact that λ and squared loss are equivalent is taken for granted. One of the implications of MP-LAN is that this fact is *also* true for models with convex boundaries. Hence, the derivation of a "quantum information criterion" is now within reach.

Null theories of loglikelihood ratios have many other applications, including hypothesis testing (35; 219) and confidence regions (117), and our result is directly applicable to them. Refs. (219; 117) both point out explicitly that their methods are unreliable near boundaries and therefore cannot be applied to rank-deficient states; our result fixes this outstanding problem.

Because MP-LAN is a generalization applicable to *all* models with convex bound-

aries, it can also be used to reason about the properties of other kinds of statistical estimation problems involving such models. In particular, MP-LAN is also applicable to the models describing the *gates* (channels) implemented by a quantum information processor. As alluded to in Section 3.1, models for channels must satisfy *complete positivity*, which places a constraint on the parameter(s) of the model. MP-LAN could be used to compute the expected value of λ for different models of the channel. (For example, one model might be that a classical memory exists somewhere in the environment that affects which channel is actually applied, which would be useful for modeling (seemingly) non-Markovian noise.) By defining MP-LAN and examining its implications for statistical model selection (more are discussed in the next chapter), more advanced statistical modeling techniques can be developed for characterizing QIPs.

The next chapter discusses some more applications of MP-LAN, including its implications for quantum compressed sensing.

Chapter 4

Other applications of MP-LAN

I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail. - Abraham Maslow, 1966 (208)

The generalization of LAN given in Chapter 3 (MP-LAN) was extremely useful for understanding the behavior of the maximum likelihood (ML)[1] *estimator in state tomography. This chapter explores other applications of MP-LAN. The first connects MP-LAN to quantum compressed sensing, where I show that the expected rank of ML estimates can be accurately approximated, and depends only on the Hilbert space dimension d and the rank of the true state ρ_0. The second application is tomography of optical quantum states, where I show that the expected value of the loglikelihood ratio statistic in ideal (Fisher-isotropic) tomography generally tracks its expected value for heterodyne tomography.*

[1]Please note that in Chapters 5 and 6, the acronym ML will be used for 'machine learning'.

4.1 Quantum compressed sensing: expected rank of ML estimates

The fact that quantum state space has boundaries impacts many properties of ML estimates. Chapter 3 showed that the boundary affects the squared distance from the ML estimate to the true state (i.e., the loglikelihood ratio statistic λ). However, other properties are affected as well. In this section, I show how the positivity constraint in state tomography affects the expected *rank* of ML estimates. This is an active area of research within the field of "quantum compressed sensing" (126; 257; 164; 174; 105; 287; 58; 163; 51; 195).

4.1.1 Overview of quantum compressed sensing

Historically, quantum compressed sensing has focused on two main research questions – first, whether compressed sensing of quantum states is possible, and second, what measurements enable quantum compressed sensing. Before discussing quantum compressed sensing, a brief digression on *classical* compressed sensing is in order.

Classical compressed sensing concerns itself with the following problem: "Given some *sparse signal* $\mathbf{x}_0 \in \mathbb{R}^d$, and some noisy data $\mathbf{y}$ acquired by measuring that signal, under what conditions is recovery (inference) of $\mathbf{x}_0$ possible?". Usually, classical compressed sensing proceeds by assuming

$$\mathbf{y} = A\mathbf{x} + \boldsymbol{\epsilon}, \tag{4.1}$$

where A is the *sensing* (design) matrix, and $\boldsymbol{\epsilon}$ is a noise term. The sensing is "compressive" in the sense that A is an $m \times d$ matrix, where ideally, $m \ll d$: if the measurements are sufficiently informative, then a small number of them is sufficient for inferring x_0.

The notion of the *sparsity* of the signal depends on what kind of signal is being

inferred. If $\mathbf{x}_0$ is a vector, then the canonical notion of sparsity is the L_0 "norm"[2]:

$$L_0(\mathbf{x}) = \sum_j I[\mathbf{x}_j] \quad I[x] = 1 \iff x \neq 0. \tag{4.2}$$

One other other hand, if $\mathbf{x}_0$ represents a matrix (as it does in, e.g., matrix completion problems), then the canonical notion of sparsity is that $\mathbf{x}_0$ is low *rank*.

Recovery of $\mathbf{x}_0$ is usually done by solving some convex optimization problem:

$$\hat{\mathbf{x}}_0 = \underset{\mathbf{x}\in\mathbb{R}^d}{\operatorname{argmin}} \; f(\mathbf{x}) \quad s.t. \quad ||A\mathbf{x} - \mathbf{y}|| \leq \tau, \tag{4.3}$$

where the data $\mathbf{y}$ is gathered according to Equation (4.1). The convex function f encodes the notion of sparsity (e.g., for convex recovery of sparse vectors, it is the L_1 norm $L_1(\mathbf{x}) = \sum_j |\mathbf{x}_j|$). The condition $||A\mathbf{x} - \mathbf{y}|| \leq \tau$ enforces the constraint that the recovered signal be *consistent* with the observed data. The regularization parameter τ could be a bound on $||\boldsymbol{\epsilon}||$, for example.

A full discussion of classical compressed sensing is beyond the scope of this thesis. However, what is relevant for a discussion of quantum compressed sensing is that historically, classical compressed sensing has focused on understanding what measurement maps A enable compressed sensing (i.e., the properties A must have so that m does not have to scale very rapidly with d in order to ensure $\hat{\mathbf{x}}_0$ is close to $\mathbf{x}_0$). Historically, A is required to satisfy the *Restricted Isometry Property* (RIP). Briefly, a measurement map satisfies s-restricted RIP if its action on s-sparse vectors doesn't distort their 2-norm very much. This is a very powerful property, and much of literature in classical compressed sensing involves analyzing measurement maps that satisfy it (52; 54).

Quantum compressed sensing has likewise focused on convex recovery of low-rank quantum states, also by considering measurement designs that satisfy RIP (164; 328;

[2]Because L_0 does not satisfy the homogeneity condition $L(a\mathbf{x}) = aL_0(\mathbf{x})$, it is not a norm.

257). Analogous to the classical case, quantum compressed sensing assumes that N_{samples} copies of ρ_0 are available, and they are measured using some POVM $\{E_j\}$. The experiment design (recall Equation (2.17)) is a matrix M of the form

$$M = \begin{pmatrix} (E_1| \\ (E_2| \\ \vdots \\ (E_m| \end{pmatrix}. \tag{4.4}$$

One of the key questions in quantum compressed sensing is "If ρ_0 is rank r, d-dimensional density matrix, then what are the optimal values of N_{samples} and m such that convex recovery of ρ_0 is possible?". Previous research has illuminated some aspects of this problem:

1. For a POVM consisting of Pauli measurements (either a randomly chosen set (126) or a fixed set (195)), $m = \mathcal{O}(rd\text{poly}\log d)$ such measurements suffice to recover ρ_0 with high probability.

2. If the POVM is *strictly rank-r informationally complete* (164), convex recovery of ρ_0 will yield $\hat{\rho}_0 = \rho_0$.

These results were derived in the asymptotic $N_{\text{samples}} \to \infty$ limit (i.e., *exact sampling*). In the remainder of this section, I show how MP-LAN (itself a property that is only valid asymptotically) can be applied to the problem of computing the expected rank of ML estimates. Importantly, MP-LAN is a property of the *model* (quantum state space), and doesn't explicitly depend on any particular *measurement* of ρ_0. For this reason, MP-LAN provides a new way for thinking about quantum compressed sensing (a point I return to in Chapter 7).

4.1.2 Using MP-LAN to compute $\langle \mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}) \rangle$

This section reprises some of the terminology and language used in Chapter 3. Let $\mathcal{M}_d$ denote the set of $d \times d$ density matrices, and assume $\rho_0 \in \mathcal{M}_d$ has rank r. Let $\mathcal{M}'_d$ denote the set of $d \times d$, trace-1, Hermitian matrices. Clearly $\mathcal{M}_d \subset \mathcal{M}'_d$. The ML estimate in a model $\mathcal{M}$ is denoted $\hat{\rho}_{\mathrm{ML},\mathcal{M}}$. Because $\mathcal{M}_d$ satisfies MP-LAN, there is a relationship between ML estimates in $\mathcal{M}_d$ and ML estimates in $\mathcal{M}'_d$:

$$\hat{\rho}_{\mathrm{ML},\mathcal{M}_d} = \operatorname*{argmin}_{\rho \in \mathcal{M}_d} \mathrm{Tr}[(\rho - \hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}) \mathcal{I} (\rho - \hat{\rho}_{\mathrm{ML},\mathcal{M}'_d})], \tag{4.5}$$

where $\mathcal{I}$ is the Fisher information. As $N_{\mathrm{samples}} \to \infty$, $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$ are Gaussian-distributed around ρ_0, with a covariance matrix given by $\mathcal{I}^{-1}/N_{\mathrm{samples}}$. As in Chapter 3, I will make the assumption that $\mathcal{I}$ is the identity, so that the metric induced on state-space is the Hilbert-Schmidt metric.

Recall that solving Equation (4.5) when $\mathcal{I}$ is proportional to the identity is straightforward:

- Take $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$ and diagonalize it (so that $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d} = V^\dagger D V$) yielding eigenvalues of two types:

  1. r eigenvalues that relate to the support of ρ_0, given by $p_j = \rho_{jj} + \delta_{jj}$ where ρ_{jj} is the j^{th} eigenvalue of ρ_0, and $\delta_{jj} \sim \mathcal{N}(0, \epsilon^2)$, where $\epsilon = 1/\sqrt{N_{\mathrm{samples}}}$.
  2. $n \equiv d - r$ elements that are eigenvalues of δ_{ker} – a matrix drawn from the Gaussian Unitary Ensemble – denoted by $\boldsymbol{\kappa} = \{\kappa_j : \ j = 1, \ldots, n\}$.

  (Recall the discussion in Section 3.5.3.)

- Determine a *decremement* q, which is the solution to

$$\sum_{j=1}^{r} (p_j - q)^+ + \sum_{j=1}^{n} (\kappa_j - q)^+ = 1. \tag{4.6}$$

- Compute the eigenvalues of the constrained ML estimate $\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}$ by shrinking

the eigenvalues of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$ according to

$$p_j \to (p_j - q)^+ \ , \ \kappa_j \to (\kappa_j - q)^+, \tag{4.7}$$

where $(x)^+ = \max(x, 0)$. Form the matrix $D' = \mathrm{diag}((p_1 - q)^+, \cdots, (p_r - q)^+, (\kappa_1 - q)^+, \cdots, (\kappa_n - q)^+)$.

- Compute $\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}$ as $V^\dagger D' V$.

The decrement q is the amount that has to be subtract from the eigenvalues of $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$ to make the constrained ML estimate both positive-semidefinite and trace-1. In general, q is a random variable – it depends on the random perturbations of the eigenvalues of ρ_0 – but in the same spirit that pervaded the derivation of an approximation to the Wilks theorem throughout Section 3.5, we will interest ourselves only in the *expected value* of $\mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})$, so q can be treated as the deterministic variable solved for back in Equation (3.40).

Because the rank is simply the number of non-zero eigenvalues, $\langle \mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}) \rangle$ is easy to write down:

$$\langle \mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}) \rangle = \sum_{j=0}^{r-1} \langle I[(p_j - q)^+] \rangle + \sum_{j=r}^{d-1} \langle I[(\kappa_j - q)^+] \rangle, \tag{4.8}$$

with $I[x]$ as the indicator function: $I[x] = 1$ if $x > 0$, and $I[x] = 0$ otherwise. In terms of the diagonal elements of ρ_0, Equation (4.8) is

$$\langle \mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}) \rangle = \sum_{j=0}^{r-1} \langle I[(\rho_{jj} + \delta_{jj} - q)^+] \rangle + \sum_{j=r}^{d-1} \langle I[(\kappa_j - q)^+] \rangle. \tag{4.9}$$

In the asymptotic ($N_{\mathrm{samples}} \to \infty$) limit, $\rho_{jj} \gg \delta_{jj}, q$. In turn, $(\rho_{jj} + \delta_{jj} - q)^+ = \rho_{jj}$, and so

$$\sum_{j=0}^{r-1} \langle I[(\rho_{jj} + \delta_{jj} - q)^+] \rangle = \sum_{j=0}^{r-1} I[\rho_{jj}] = r. \tag{4.10}$$

This assumption implies *the expected rank of the constrained ML estimate is not lower than the rank of the state being tomographed*:

$$\langle \mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}) \rangle = r + \sum_{j=r}^{d-1} \langle I[(\kappa_j - q)^+] \rangle \geq r. \tag{4.11}$$

This result implies that asymptotically, maximum likelihood state tomography is *self-certifying*: all non-zero eigenvalues of ρ_0 will be estimated to non-zero values by the ML estimator[3]. In turn, this implies that *the lower the expected rank of the ML estimate, the more pure the underlying true state has to be*[4].

The derivation of a replacement for the Wilks theorem relied on two key assumptions: first, that κ_j could be replaced by their typical values, and that $N \equiv d - r \gg 1$. The first assumption was justified due to our interest in computing $\langle \lambda \rangle$; here, I make that same assumption because I am computing the expected rank. The second assumption enabled discrete sums involving the κ_j to be replaced by continuous integrals over their probability distribution. Taking that same limit here allows me to replace the sum over the indicator function by an integral.

The sum $\sum_{j=r}^{d-1} \langle I[(\kappa_j - q)^+] \rangle$ simply counts the number of κ_j that are larger than q. Hence, this sum divided by the total range over which j varies is the probability that any element of the set $\{\kappa_j\}_{j=r}^{d-1}$ is greater than or equal to q. Therefore, by the two assumptions made in the previous paragraph, it follows that

$$\frac{1}{d-r} \sum_{j=r}^{d-1} \langle I[(\kappa_j - q)^+] \rangle \to \frac{1}{d-r} \sum_{j=r}^{d-1} I[(\bar{\kappa}_j - q)^+] \to \int_{\kappa=q}^{\kappa_{\max}} \Pr(\kappa)\, d\kappa. \tag{4.12}$$

Hence,

$$\langle \mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}) \rangle = r + (d-r) \int_{\kappa=q}^{\kappa_{\max}} \Pr(\kappa)\, d\kappa. \tag{4.13}$$

[3]Though other compressed sensing protocols, such as an L_1-regularized convex estimator, can have this property: it is not unique to ML state tomography.

[4]Numerical evidence indicates the distribution $\Pr(\mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}))$ is tightly-concentrated about its expected value, implying that rank of *each* ML estimate could provide the same kind of self-certifying guarantee.

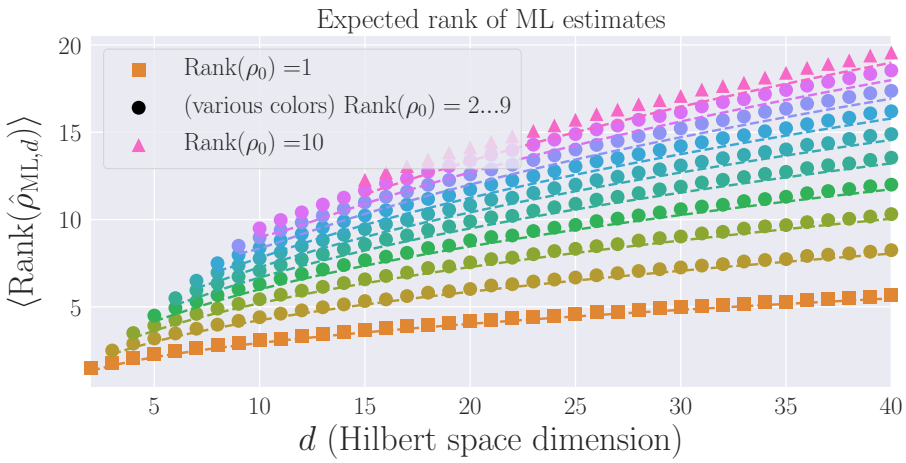

Figure 4.1: **Comparing theory for the expected rank to numerical results**. The formula for $\langle \mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}) \rangle$ in Equation (4.15) agrees well with numerics. As in Figure 3.10, discrepancies begin to appear as r becomes comparable to d.

Recall from Section 3.5.3 that asymptotically, the eigenvalues of the kernel of ρ_0 are distributed according to the Wigner semicircle distribution:

$$\Pr(\kappa) = \frac{2}{\pi R^2}\sqrt{R^2 - \kappa^2} \text{ where } R = 2\epsilon\sqrt{d-r}. \tag{4.14}$$

4.1.3 Results and comparison to numerical experiments

Given the distribution of κ, computing $\langle \mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}) \rangle$ is straightforward:

$$\begin{aligned}\frac{\langle \mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}) \rangle - r}{d-r} &= \int_{\kappa=q}^{\kappa_{\max}} \Pr(\kappa)\, d\kappa = \frac{2}{\pi R^2}\int_{\kappa=q}^{R} \sqrt{R^2-\kappa^2}\, d\kappa \\ &= \frac{1}{2\pi}\left(-2\sin^{-1}(q/R) - 2(q/R)\sqrt{1-(q/R)^2} + \pi\right).\end{aligned} \tag{4.15}$$

By Equation (3.40), $q/R \approx 1 - \frac{1}{2}x + \frac{1}{10}x^2 - \frac{1}{200}x^3$, where x is given in Equation (3.41). Figure 4.1 compares this approximation to numerical results.

4.1.4 Classical case

The result derived in Equation (4.15) is straightforward, but required a great deal of preliminaries to get to. By considering the problem of estimating outcome probabilities over a d-dimensional simplex, we can derive a similar result that suggests a surprising connection between the classical and quantum versions of this problem.

Suppose we want to estimate a probability vector $\mathbf{p}_0 = (p_0, p_1, \cdots p_{r-1}, 0, \cdots, 0) \in \Delta^d$, where there are $d-r$ zeroes, and Δ^d is the d-dimensional simplex. (For example, we may be rolling a d-sided die, and want to estimate, for each of its faces, the probability that face will come up on any given roll.)

Consider the model $\mathcal{M}_d = \{\mathbf{p} \in \Delta^d\}$. This model satisfies MP-LAN, because it is a convex subset of $\mathcal{M}'_d = \{\mathbf{f} \in \mathbb{R}^d \mid \sum_j \mathbf{f}_j = 1\}$, which itself is a model that satisfies LAN.

Even though $\mathcal{M}_d$ is a model over a *classical* state-space, the results derived in the MP-LAN framework are also applicable to it, because MP-LAN is a framework that applies to *any* model with convex boundaries. In particular, because $\mathcal{M}_d$ satisfies MP-LAN, every ML estimate $\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}$ can be computed by truncating an unconstrained ML estimate $\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}'_d}$

$$\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d} = \operatorname*{argmin}_{\mathbf{p}\in\Delta^d} ||\mathbf{p} - \hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}'_d}||^2. \tag{4.16}$$

Notice that the sum of the components of $\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}'_d}$ is 1, and that some of those components will be negative. (Within the $d-r$ elements that are in the kernel of $\mathbf{p}_0$, typically half of the δ_j will be negative.) Therefore, it's necessary to *subtract* from the components before truncating. (Otherwise, the components of the resulting estimate will sum to a number greater than 1.) Computing the constrained estimate is possible using the truncation algorithm presented in Section 3.5.3; namely,

$$(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d})_j = \left(\left(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}'_d}\right)_j - q\right)^+, \tag{4.17}$$

where q is the solution to

$$\sum_j \left(\left(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}'_d} \right)_j - q \right)^+ = 1. \tag{4.18}$$

The components of $\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}'_d}$ are given by

$$\left(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}'_d} \right)_j = \begin{cases} p_j + \delta_j & 0 \leq j \leq r-1 \\ \delta_j & r < j < d-r \end{cases}, \tag{4.19}$$

where $\delta_j \sim \mathcal{N}(0, \epsilon^2)$. In what follows, let $\Phi_\epsilon(x)$ denote the cumulative distribution function of their distribution:

$$\Phi_\epsilon(x) = \int_{-\infty}^{x} \Pr(\delta)\, d\delta = \frac{1}{\epsilon\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\delta^2/2\epsilon^2}\, d\delta. \tag{4.20}$$

From Equations (4.17) and (4.19), it follows that the expected rank of the ML estimate is

$$\langle \mathrm{Rank}(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}) \rangle = \sum_{j=1}^{r} I[(p_j + \delta_j - q)^+] + \sum_{j=r}^{d-r} I[(\delta_j - q)^+]. \tag{4.21}$$

As we did when solving for q in the quantum case, we invoke the asymptotic assumption $N_{\mathrm{samples}} \to \infty$, so that $\epsilon \to 0$, and $p_j \gg \delta_j, q$. From this, it follows that

$$\langle \mathrm{Rank}(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}) \rangle = r + \sum_{j=r}^{d-r} I[(\delta_j - q)^+]. \tag{4.22}$$

Again, we consider the large-d limit – so that the sum over the indicator function can be turned into an integral – and replace the δ_j by a typical sample:

$$\begin{aligned} \langle \mathrm{Rank}(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}) \rangle &= r + (d-r) \int_{\delta=q}^{\infty} \Pr(\delta)\, d\delta \\ &= r + (d-r)\left(1 - \Phi_\epsilon(q)\right). \end{aligned} \tag{4.23}$$

Solving for q

Under the asymptotic assumption, Equation (4.18) simplifies to

$$\begin{aligned} &\sum_{j=0}^{r-1} p_j + \sum_{j=0}^{r-1} \delta_j - rq + \sum_{j=r}^{d-r} (\delta_j - q)^+ = 1 \\ \implies &-rq + \sum_{j=0}^{r-1} \delta_j + \sum_{j=r}^{d-r} (\delta_j - q)^+ = 0. \end{aligned} \tag{4.24}$$

Next, we approximate the sum over δ_j by the expected value of any δ_j, replace the random sample of δ_j by their typical values, and write the sum as an integral over the distribution of δ:

$$-rq + \frac{d-r}{\sqrt{2\pi}\epsilon} \int_{\delta=q}^{\infty} (\delta - q) e^{-\delta^2/2\epsilon^2} \, d\delta = 0. \tag{4.25}$$

A little bit of algebra yields a nicer-looking equation for q in terms of $z \equiv (q/\epsilon)$:

$$\begin{aligned} &-rq + \frac{d-r}{\sqrt{2\pi}\epsilon} \int_{\delta=q}^{\infty} (\delta - q) e^{-\delta^2/2\epsilon^2} \, d\delta = 0 \\ \implies &-rz + \frac{d-r}{\sqrt{2\pi}} \int_{u=z}^{\infty} (u - z) e^{-u^2/2} \, du = 0. \end{aligned} \tag{4.26}$$

Note that z is ϵ-*invariant*, and so is a useful variable to use in these calculations. The Gaussian integral may be computed as

$$\int_{u=z}^{\infty} (u - z) e^{-u^2/2} \, du = e^{-z^2/2} - z\sqrt{\pi/2}\,\mathrm{Erfc}(z/\sqrt{2}). \tag{4.27}$$

where $\mathrm{Erfc}(x)$ is the *complementary error function for the standard normal distribution*, which may be related to the cumulative distribution function of a standard normal random variable by

$$\mathrm{Erfc}(x) = 2(1 - \Phi_1(x\sqrt{2})). \tag{4.28}$$

Therefore, the equation for z becomes

$$\begin{aligned} &-rz + \frac{d-r}{\sqrt{2\pi}} \left[e^{-z^2/2} - z\sqrt{2\pi}\,(1 - \Phi_1(z)) \right] = 0 \\ &\implies -z + A\left(e^{-z^2/2} + z\Phi_1(z)\sqrt{2\pi} \right) = 0, \end{aligned} \tag{4.29}$$

where $A = (1 - (r/d))/\sqrt{2\pi}$. Notice that as $r \to d$, $A \to 0$, and the solution $z \to 0$ as well. (When the probability vector being estimated is close to full-rank, then asymptotically, every unconstrained ML estimate lies inside the model $\mathcal{M}_d$, so no truncation is necessary.) Conversely, if $r \ll d$, then most of the unconstrained ML estimates lie outside $\mathcal{M}_d$, so truncation is necessary.

Solving this equation analytically would be difficult, as it involves Φ_1, which is a function with no known expression in terms of elementary functions. This difficulty is compounded by the fact that the fluctuations δ_j are in principle unbounded (unlike the quantum case, where the finite support of the Wigner semicircle distribution constrains how large the eigenvalue perturbations can be). Notice that in Equation (4.24), q is upper-bounded by the *largest* value of the $d-r$ values of δ_j coming from the null space of $\mathbf{p}_0$. To get a handle on how large δ_j will be, it's useful to turn to *extreme value theory*. As shown in Appendix J , Section J.1, if m is the random variable given by

$$m = \max_{r \leq j \leq d-r} \delta_j, \tag{4.30}$$

then

$$\langle m \rangle \leq \epsilon \sqrt{2 \log(d-r)}. \tag{4.31}$$

When $d = r$, asymptotically every $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$ is contained within $\mathcal{M}_d$, so no truncation is necessary. Hence, the odd behavior of this bound when $d = r$ is not an issue. While $\langle m \rangle$ does depend on d, this dependence is modest. What's more, the asymptotic assumption $\epsilon \to 0$ ensures that $\langle m \rangle$ will not be too big. Consequently, q is typically $\mathcal{O}(\epsilon)$, so z is $\mathcal{O}(1)$. Therefore, to solve Equation (4.29), I will simply expand everything about $z = c$ for a suitable choice of c, and then solve the resulting equation. I take $c = 1$ primarily for simplicity, which also allows for a reasonably accurate solution to Equation (4.29) (when compared to numerics). Expanding (4.29) using

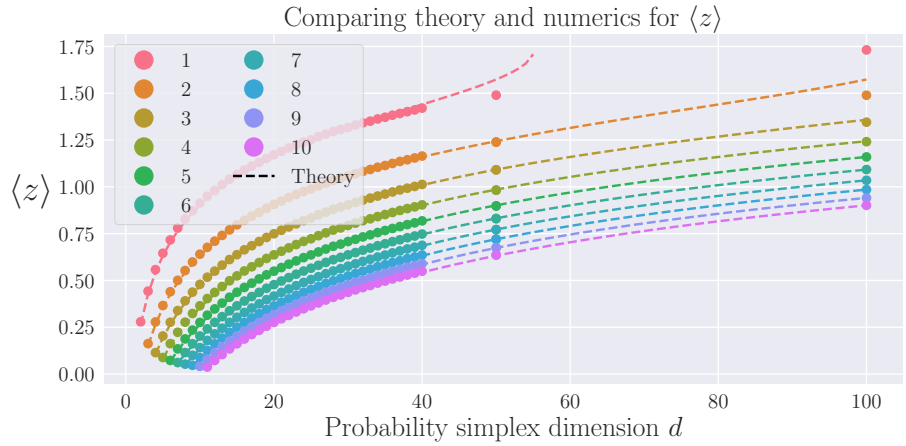

Figure 4.2: **Comparing theory (Equation** (J.17)**) to numerical results**. Once $r \ll d$, the theory begins to break down, as expected from a consideration of the solution. As the rank of ρ_0 decreases, the scaled decrement $z = q/\epsilon$ increases.

Taylor series to 3rd order, and solving the resulting equation, gives

$$z = \frac{\sqrt{2\pi e}}{B}\left(1 - B\Phi_1(1) + B\frac{e^{-1/2}}{\sqrt{2\pi}} - \sqrt{D}\right), \tag{4.32}$$

where $B = 1 - r/d$ and

$$\begin{aligned} D = B^2 &\left((\Phi_1(1))^2 - \sqrt{\frac{2}{\pi}}\Phi_1(1)e^{-1/2} - \frac{1}{\pi}e^{-1}\right) \\ &+ B\left(-2\Phi_1(1) + \sqrt{\frac{2}{\pi}}e^{-1/2}\right) + 1. \end{aligned} \tag{4.33}$$

Section J.2 provides the details for this calculation. Notice that if $D < 0$, then the solution for z is ill-defined. This happens when $r/d \lesssim 0.012$. Thus, for a fixed value of d, the expression for z in Equation (4.32) is ill-defined when $r \lesssim .012d$. Figure 4.2 compares Equation (4.32) to numerical simulations. Considering $r = 1$, once $d \gtrsim 50$, the solution no longer tracks the numerical results. The fact that it is higher than the numerical results has implications for the expected value of the rank, as we shall soon see. Another reason for discrepancies is that z is not exactly $\mathcal{O}(1)$, as it depends on d, r. A more precise calculation would do the Taylor series about $\sqrt{2\log(d-r)}$.

Computing $\langle \mathrm{Rank}(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}) \rangle$

In terms of q, the expected rank of the constrained ML estimate is

$$\langle \mathrm{Rank}(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}) \rangle = r + (d-r)\left(1 - \Phi_\epsilon(q)\right), \tag{4.34}$$

where

$$\Phi_\epsilon(q) = \frac{1}{\epsilon\sqrt{2\pi}} \int_{-\infty}^{q} e^{-x^2/2\epsilon^2}\, dx. \tag{4.35}$$

Notice that $\Phi_\epsilon(q) = \Phi_1(z)$, so

$$\langle \mathrm{Rank}(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}) \rangle = r + (d-r)\left(1 - \Phi_1(z)\right). \tag{4.36}$$

This expression, coupled with the result in Equation (J.17) completely suffices for computing the expected rank. Because Φ_1 is a special function, with no known expansion in terms of elementary functions, I leave this expression as-is. Given that Φ_1 is readily computed numerically for arbitrary values of z, I see no harm in doing so. Figure 4.3 compares this theory to numerics. In numerical simulations, I take $\epsilon = 10^{-4}$, and estimate $\langle \mathrm{Rank}(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}) \rangle$ using $10^4 - 10^5$ i.i.d. realizations of the unconstrained ML estimates $\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}'_d}$.

Because the theory for z in Equation (J.17) tends to *over-predict* the amount subtracted to do the truncation, the *rank* of the truncated ML estimate is going to be lower in theory than in practice. This is readily seen in the discrepancies for $r = 1$ around $d = 25$, and at other values of d for other values of r. Again, a more accurate theory for z would remedy these issues.

4.1.5 Comparing quantum and classical cases

The thrust of this section has been to derive results for the expected rank of ML estimates for the task of quantum state tomography and that of estimating a probability

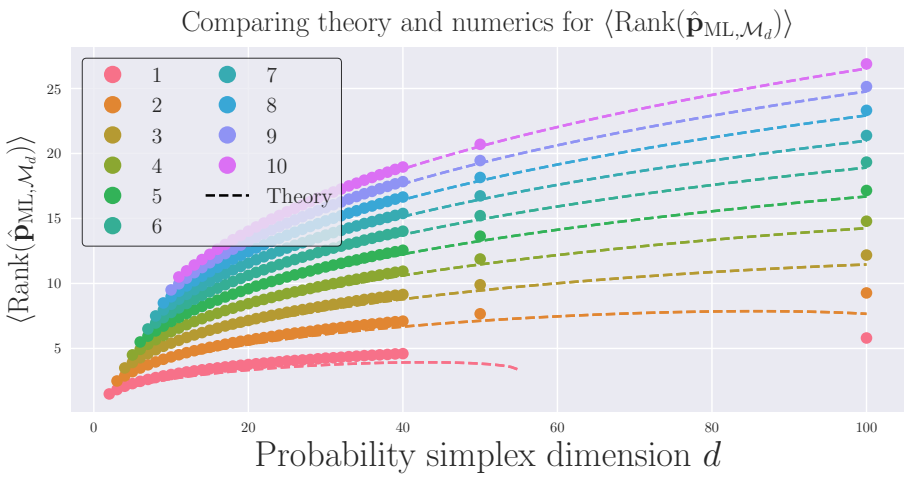

Figure 4.3: **Comparing theory and numerics for** $\langle \mathrm{Rank}(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}) \rangle$. From Equation (J.17), we know that once $r \ll d$, the theory for z starts to break down; this is reflected in the discrepancies between the theory (dashed lines) and numerical results (solid dots) at higher values of d. The growth in the expected rank is modest with respect to d for a fixed value of r.

vector. A natural question is how these two quantities relate to one another. Figure 4.4 compares $\langle \mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}) \rangle$ and $\langle \mathrm{Rank}(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}) \rangle$ using both numerics and the theories given in Equations (4.15) and (4.36) respectively. A surprising feature of this comparison is that $\langle \mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}) \rangle$ and $\langle \mathrm{Rank}(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}) \rangle$ tend to track one another, though numerically, $\langle \mathrm{Rank}(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}) \rangle$ appears to be greater than $\langle \mathrm{Rank}(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}) \rangle$ usually. Whether that is a genuine fact about these models, or an artifact of my numerical simulations, is difficult to say. Comparing the theories is also fraught with complications, as both are derived under various approximations. As noted previously, the fact that the classical theory *underpredicts* $\langle \mathrm{Rank}(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}) \rangle$ means that the comparison in the rightmost panel of Figure 4.4 is off a bit.

4.1.6 Conclusion and discussion

The research presented in this section shows that low-rank estimates are a ubiquitous feature of maximum-likelihood state tomography. This suggests that the success of

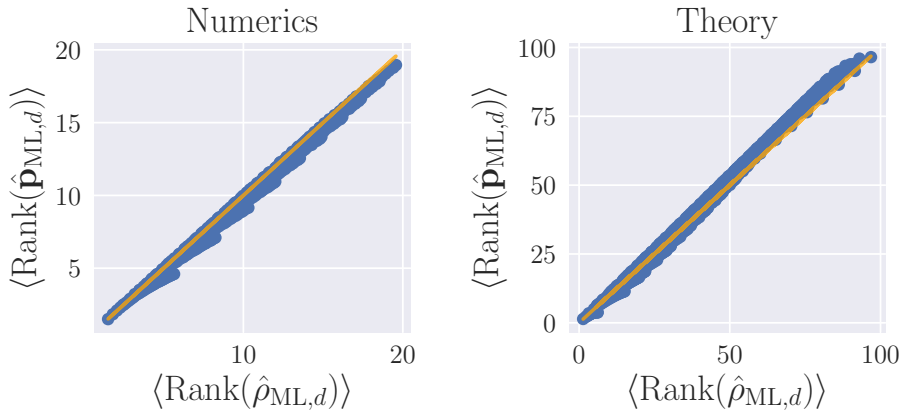

Figure 4.4: **Comparing quantum and classical cases**. The orange line is what would result if the two cases agreed exactly. For the most part, the two cases do mirror one another. However, in the rightmost panel, the classical theory for $\langle \mathrm{Rank}(\hat{\mathbf{p}}_{\mathrm{ML},\mathcal{M}_d}) \rangle$ tends to *underpredict*; hence the excess of blue points above the orange line.

quantum compressed sensing – while certainly dependent on the POVM being measured – also depends on the geometry of quantum state space itself. This observation was made in (164) in the definition of a "rank-r strictly complete POVM". Understanding the relationship between such POVMs and their implications for the asymptotic distribution of ML estimates would be a fruitful research direction.

In addition, there appears to be a close connection between compressed sensing of quantum states and an analogous problem for compressed sensing over classical probability simplices (Figure 4.4). This suggests connections between the geometry of classical simplices and quantum state space, as has been noted in (11). Understanding how these two cases are genuinely different would be a useful research direction to pursue.

Finally, I suspect there are some fairly deep and general connections between an analysis of state tomography using MP-LAN, and quantum compressed sensing. This suspicion is based on several threads of reasoning. First, classical compressed sensing

research has moved beyond measurement maps that satisfy RIP (53; 181). Modern compressed sensing protocols have been developed based on the observation that if the measurement map A doesn't provide sufficient restrictions on the set of possible solutions, then Equation (4.3) is *underdetermined*, and no unique solution exists. On the other hand, if A provides "just enough" constraints on the solution, then it's possible for convex recovery to succeed with high probability. (If A has too many constraints, the problem is overdetermined, and no feasible point may exist.) Phrased another way, as the number of rows of A are varied, there is a "phase transition" from infeasibility to feasibility of the convex recovery problem.

Understanding how that phase transition depends on m – number of rows of A – is an active area of research (299; 55). Crucially, this phase transition relates to geometric properties of Equation (4.3), by looking at the *descent cone* of $f(\mathbf{x})$ at the signal $\mathbf{x}_0$. (See (299) for details.) That is, given a fixed measurement map A, of an appropriate sort, there may exist some convex functions such that recovery by solving Equation (4.3) is feasible, while for others, it is not. For this reason, modern compressed sensing research has come to focus on what could be called the "geometry of convex optimization"[5] (299; 332).

Second, it turns out that a *geometric* quantity of this convex function, the *statistical dimension of its descent cone*, is crucial in the modern compressed sensing framework (9). This quantity governs the transition region from an infeasible problem to a feasible one. That is, if the number of rows of A is less than the statistical dimension, recovery is infeasible.

Third, a careful reading of (9) indicates that *the expected value of the loglikelihood ratio statistic given in Equation* (3.43) *is in fact the statistical dimension of the tangent cone at* ρ_0. From a discussion with Joel Tropp, I learned this is the descent

[5] I am not aware if those within the classical compressed sensing community would use that phrase, however.

cone of the indicator function over positive-semidefinite matrices: $I[M] = 1 \iff M \geq 0$. This function would show up in, e.g., a regularized convex optimization with a penalty of the form $\log(I[M])$. That is, if the estimate is negative, the penalty is infinite.

These lines of reasoning converge to the observation that if the descent cone for a convex recovery problem of the form (convex function + regularization term) can be expressed in terms of the descent cone of each, then we can apply both MP-LAN and the framework developed in (9) to port quantum state tomography into a modern compressed sensing framework. This may help us understand existing results in quantum compressed sensing regarding a sufficient number of POVM elements for convex recovery of ρ_0. To determine whether this is in fact the case, studying how the descent cone of a convex function changes under regularization would be required, and most likely would also require deriving results about the descent cone of the likelihood function.

4.2 Choosing a Hilbert space dimension for continuous-variable quantum systems

In Chapter 2, I alluded to how the word "tomography" came to be associated with characterizing quantum devices because some of the first state tomography protocols were developed for continuous-variable (CV) quantum systems (192; 191; 247; 248; 201**?**). These systems also provide a natural testing ground for the statistical model selection problem discussed in Chapter 3. Within this context, a relevant statistical model selection problem is: choose a good Hilbert space dimension d for modeling the CV system. In this Section, I show how use the approximate replacement for the Wilks theorem (Equation (3.43)) to do so.

4.2.1 Continuous-variable (CV) quantum systems

In the context of this section, "continuous" means the state of the system is described by a continuous parameter; for example, the wavefunction $|\psi\rangle$ of a one-dimensional quantum system can be expanded in the *position* basis

$$|\psi\rangle = \int dx\ \psi(x)\ |x\rangle\,, \quad \int |\psi(x)|^2\, dx = 1,\ \langle x|x'\rangle = \delta(x-x'),\ X|x\rangle = x|x\rangle. \tag{4.37}$$

An analogous expansion of $|\psi\rangle$ is possible in the *momentum* basis, $|p\rangle$. Formally, states of one-dimensional CV systems are described by density operators on the infinite-dimensional Hilbert space $L^2(\mathbb{R})$. Characterizing these states (i.e., estimating them) clearly requires advanced statistical methods, as fitting infinitely many parameters to a finite amount of data is not tenable.

Here, I'll focus on optical quantum systems (quantum states of light). These systems are important in metrology, sensing, communications, and computation (227; 160; 153; 184; 89; 176; 87; 245); hence, being able to adequately characterize them is important. For several books on the subject of quantum optics, see (128; 272; 64; 205). A full discussion of quantum optics and continuous-variable systems is beyond the scope of this thesis. However, a small digression on some of the basic concepts is in order.

Classically, the behavior of light propagating in some medium (free space or otherwise) is described by the Maxwell equations, plus any requisite boundary conditions. Generally, those differential equations have families of solutions, indexed by some set of integers. For a particular choice of these indices, the corresponding solution is said to be a *mode* of the field. For instance, consider a rectangular cavity whose side lengths are l_x, l_y, and l_z. Then, the modes are labeled by $\mathbf{k} = (k_x, k_y, k_z, p)$, where $k_j = \pi n_j / l_j$, and p denotes the polarization state of the mode. Further, mode $\mathbf{k}$ propagates with an angular frequency given by $\omega_{\mathbf{k}}^2 = c^2(k_x^2 + k_y^2 + k_z)^2$. Thus, $\mathbf{k}$ indexes the mode. For each mode, there is an associated electric and magnetic field.

The modes can be *quantized* using a simple harmonic oscillator with an angular frequency $\omega_{\mathbf{k}}$. In what follows, I'll focus on characterizing a single mode of light, and so will suppress the index $\mathbf{k}$ unless it is needed. Classically, the Hamiltonian describing a simple harmonic oscillator of mass m and angular frequency ω is

$$H = \frac{p^2}{2m} + \frac{m\omega^2}{2}x^2. \tag{4.38}$$

Quantizing this Hamiltonian means replacing x, p by their quantum-mechanical counterparts X, P, which are operators that satisfy the canonical commutation relation $[X, P] = i\hbar$:

$$H \to \frac{P^2}{2m} + \frac{m\omega^2}{2}X^2. \tag{4.39}$$

For a single mode, the operators X and P are directly related to the electric field and the magnetic vector potential; see (64, Section 2.1.2) for details. For this reason, X and P are said to be the *quadrature operators* for the mode. The eigenstates of this Hamiltonian form a useful basis in which any mode can be expanded. There are various ways to calculate them; here, I use an approach due to Dirac. By introducing the (non-Hermitian) operators

$$\begin{aligned} a &= \left(\frac{m\omega}{2\hbar}\right)^{1/2} X + i\left(\frac{1}{2m\omega\hbar}\right)^{1/2} P \\ a^\dagger &= \left(\frac{m\omega}{2\hbar}\right)^{1/2} X - i\left(\frac{1}{2m\omega\hbar}\right)^{1/2} P, \end{aligned} \tag{4.40}$$

whose commutation relations are $[a, a^\dagger] = 1$, the quantized Hamiltonian takes the form

$$H = \hbar\omega(a^\dagger a + 1/2). \tag{4.41}$$

If $|E\rangle$ is an energy eigenstate of H, so that $H|E\rangle = E|E\rangle$, then

$$\begin{aligned} Ha|E\rangle &= (E-1)a|E\rangle \\ Ha^\dagger|E\rangle &= (E+1)a^\dagger|E\rangle. \end{aligned} \tag{4.42}$$

Therefore, $a|E\rangle$ is an eigenstate of H with eigenvalue $E-1$, and $a^\dagger|E\rangle$ is an eigenstate of H with eigenvalue $E+1$. For this reason, a is called the *lowering* operator, and $a^\dagger$, the *raising* operator.

Because H is a positive operator ($\langle\psi|H|\psi\rangle \geq 0$), it follows that the energy eigenstates cannot be lowered indefinitely: there exists some state $|E_0\rangle$ such that $a|E_0\rangle = 0$. This observation, plus some algebra, yields the fact that the energy eigenstates of H are states with a definite number of excitations. These are known as the *Fock states*, denoted $|n\rangle$. Under a and $a^\dagger$, they transform as

$$a\,|n\rangle = \sqrt{n}\,|n-1\rangle \quad , \quad a^\dagger\,|n\rangle = \sqrt{n+1}\,|n+1\rangle\,. \tag{4.43}$$

These states are eigenstates of H: $H\,|n\rangle = \hbar\omega(n+1/2)\,|n\rangle$, and form a complete orthonormal basis:

$$\sum_{n=0}^{\infty} |n\rangle\langle n| = \mathcal{I},\ \langle m|n\rangle = \delta_{mn}. \tag{4.44}$$

Within the context of quantum optics, the Fock states are also called "photon-number states", as they are eigenstates of the number operator $N = a^\dagger a$. A state of the mode can be expanded in terms of the Fock states:

$$\rho = \sum_{j,k=1}^{\infty} \rho_{jk}|j\rangle\langle k|, \tag{4.45}$$

and this representation naturally lends itself to representing ρ as a matrix. There are a couple of problems with this. First, the sum is formally infinite, so a finite-dimensional approximation to the matrix will always be necessary. Second, the dynamics of a simple harmonic oscillator (i.e., the mode) is more naturally described using a *phase space representation* in terms of the phase-space variables x, p that were promoted to quantum-mechanical operators in quantizing the Hamiltonian. One such representation is known as the *Wigner function* (321):

$$W(x,p) = \frac{1}{\hbar\pi}\int_{-\infty}^{\infty} \langle x+y|\rho|x-y\rangle e^{-2ipy/\hbar}\,dy, \tag{4.46}$$

where $\langle x|\psi\rangle = \psi(x)$. Formally, the Wigner function is the *Wigner-Weyl transformation* of ρ. This transformation can also be represent other quantum-mechanical operators (such as observables) in terms of functions over phase space (60), and is known as the *Weyl representation*. For any operator A, the Wigner-Weyl transform maps it to a function $f_A(x,p)$. These functions have the property that calculating traces in operator space (e.g., $\mathrm{Tr}(AB)$) relates to integrals over their phase-space representations. In particular, this means that for any observable O of the system, its expectation value $\langle O\rangle = \mathrm{Tr}(\rho O)$ can be computed as

$$\langle O\rangle = \int W(x,p) f_O(x,p)\; dx\; dp. \tag{4.47}$$

A cursory glance at the equivalence tempts one to think that W is a probability distribution over phase space. That line of reasoning would be incorrect: the Wigner function defines a *quasiprobability distribution* in phase space, meaning there exist states ρ such that $W(x,p) < 0$ for some (x,p). For example, let ρ be the single-photon state $|1\rangle\langle 1|$. While the negativity of Wigner function has spurred intense interest in the differences between quantum mechanical and classical systems (168; 309; 100), it is not my intent to add or subtract to that discussion here.

4.2.2 The need for statistical model selection in CV tomography

Historically, tomography of optical modes has focused on using the Wigner function representation of the state (Equation (4.46)), in part because of the work of Vogel and Risken (310), which showed that by measuring the *generalized quadrature* $X_\theta = \cos\theta X + \sin\theta P$ along various angles θ using *homodyne detection*, and processing homodyne measurement data results using an inverse Radon transform, an estimate of the Wigner function $\hat{W}(x,p)$ could be computed (201).

In using this tomographic technique, experimentalists often use smoothing protocols

(47) or binning procedures (200) which are implemented to either remove unwanted "high frequency" noise or to provide sufficient counts for statistical inference (201). This noise typically is an artifact of inverting the Radon transform, or is caused by imperfections in the homodyne detector. These approaches use a type of statistical model selection, by *regularizing* the estimated Wigner function. This can be seen expanding ρ in the Fock basis:

$$W(x,p) = \frac{1}{\hbar\pi}\sum_{n=0}^{D}\rho_{jk}\int_{-\infty}^{\infty}\langle x+y|j\rangle\langle k|x-y\rangle e^{-2ipy/\hbar}\,dy, \tag{4.48}$$

with D being the dimension of ρ in the Fock basis. The integral above simply integrates over the position-basis representations of the Fock states $|m\rangle, |n\rangle$, which are functions of the form

$$\langle x|n\rangle = \left(\frac{m\omega}{\pi\hbar 2^{2n}(n!)^2}\right)^{1/4}\mathrm{Exp}\left[-\frac{m\omega x^2}{2\hbar}\right]H_n\left[\left(\frac{m\omega}{\hbar}\right)^{1/2}x\right]. \tag{4.49}$$

The integrand of Equation (4.48) relates to integrals of *Hermite polynomials* H_n. The structure of the Hermite polynomials depends very strongly on the index n: as it increases, the number of oscillations of H_n increases as well. As such, a specific choice for D corresponds to a specific choice for the amount of *structure* in the Wigner function, and vice-versa (337). From that perspective, smoothing out high-frequency wiggles is to making the assumption the state does not have support on higher dimensions.

Consequently, a *plot* of an estimated Wigner function must come with some particular choice of Hilbert space dimension D. Therefore, *smoothing or otherwise regularizing an estimated Wigner function corresponds to selecting a model.* When the representation of choice is a density matrix ρ, it is clear that many of its parameters are underdetermined by experimental data – the number of parameters is infinite, and the amount of data, finite – and regularization (such as restricting the support of $\hat{\rho}$ to some d-dimensional subspace) is typically used to avoid overfitting. Reconstructing the Wigner function using the inverse Radon transform also implies

an implicit choice for d, and can lead to overfitting (see (47) as a possible example of overfitting).

In the remainder of this section, I show how the result derived in Section 3.5 regarding $\langle\lambda\rangle$ can be applied to the problem of choosing a good Hilbert space dimension for an continuous-variable quantum system.

4.2.3 How to choose a Hilbert space dimension for CV quantum system

Suppose a single mode of light is prepared in the state ρ_0. To characterize this state using tomography, some model needs to be constructed. The largest possible model is $L^2(\mathbb{R})$ – all square-integrable wavefunctions – but this model has infinitely many parameters, and only a finite amount of data will be collected. Clearly a simpler model is necessary!

The number of photons in the mode relates to the total energy. Recall the simple harmonic oscillator Hamiltonian in Equation (4.41): if there are N photons in the mode, the energy is $\mathcal{O}(N\omega)$. A system with an infinite amount of energy is nonsensical, which means ρ_0 can be well-described by a model with a *finite* number of photons. Therefore, considering few-photon models is sensible. Let $\mathcal{M}_d$ denote the set of d-dimensional density matrices when expanded in the Fock basis:

$$\mathcal{M}_d = \{\rho \mid \rho \in \mathcal{B}(\mathcal{H}_d),\ \mathrm{Tr}(\rho) = 1,\ \rho \geq 0\}, \tag{4.50}$$

where

$$\mathcal{H}_d = \mathrm{Span}\left\{|0\rangle, |1\rangle, \cdots, |d-1\rangle\right\}. \tag{4.51}$$

Any state $\rho \in \mathcal{M}_d$ can be expanded as

$$\rho \in \mathcal{M}_d \implies \rho = \sum_{m,n=0}^{d-1} \rho_{mn} |m\rangle\langle n|. \tag{4.52}$$

Choosing a good Hilbert space dimension for reconstructing ρ_0 is equivalent to determining which of the two models $\mathcal{M}_d$ and $\mathcal{M}_{d'}$ is better. However, there are many choices for d, d'. A simplification is to take $d' = d+1$, meaning the two models differ by at most one quanta of energy. Motivated by the observation that physical states do not have arbitrarily high energy, a simple algorithm for choosing d goes as follows: beginning with $d = 2$, compute $\lambda(\mathcal{M}_d, \mathcal{M}_{d+1})$. Compare to a threshold value $\lambda_{\text{thresh}}(d, d+1)$. If $\lambda > \lambda_{\text{thresh}}$, increment d by 1 and repeat. However, if $\lambda < \lambda_{\text{thresh}}$, report d as the choice for the Hilbert space dimension.

This algorithm uses the loglikelihood ratio statistic to determine if a particular low-energy model is sufficient; if it is not, an additional Fock state is added to the basis set, and the comparison performed again. Phrased another way, this algorithm expands the support of the estimate in phase space out from the origin (the vacuum state $|0\rangle$) until that support is of sufficiently high energy to adequately model ρ_0. In terms of a matrix representation, each iteration compares a density matrix with d rows and columns to one with $d+1$ rows and columns, and determines whether the additional parameters introduced help fit the data better.

Section 3.2 noted that commonly, λ_{thresh} is take to be $\langle\lambda\rangle + \Delta\lambda$. The algorithm above uses $\lambda(\mathcal{M}_d, \mathcal{M}_{d+1})$ as the test statistic, so to set a threshold, knowing its expected value and variance would be necessary. Here, I won't focus on computing that expected value. Instead, I use the fact that

$$\lambda(\mathcal{M}_d, \mathcal{M}_{d'}) = \lambda(\rho_0, \mathcal{M}_{d'}) - \lambda(\rho_0, \mathcal{M}_d), \tag{4.53}$$

which reduces the problem of computing $\langle\lambda(\mathcal{M}_d, \mathcal{M}_{d+1})\rangle$ to that of computing $\langle\lambda(\rho_0, \mathcal{M}_d)\rangle$[6]. As a simple test of the theory for $\langle\lambda(\rho_0, \mathcal{M}_d)\rangle$ given in Equation (3.43) – which was derived under highly ideal circumstances – I'll compare it to numerical results for *heterodyne tomography*, which is a commonly-used tomographic technique for CV systems.

[6]Recall Section 3.5.

4.2.4 Application: heterodyne tomography

In quantum optics, two measurements are commonly used for tomography: *homodyne* (201; 17; 191; 318; 281; 199), alluded to in Section 4.2.2, and *heterodyne* (200; 28; 192; 201; 12). Below I briefly sketch out how these two measurements are performed.

Balanced homodyne measurement uses a local oscillator (LO) operating at the same frequency as the input signal being characterized. The LO is phase-shifted relative to the input by θ. To do the homodyne measurement, the input and LO signals are combined on a 50:50 beamsplitter. The output arms of the beamsplitter are then detected using photodetection, and the two photocurrent signals are subtracted from one another. With ideal photodetectors, this measurement is described by a POVM that projects the input signal onto the eigenstate of the generalized quadrature $X_\theta = \cos\theta X + \sin\theta P$ (12).

The heterodyne measurement can be performed in two different ways. The simplest way is to use the homodyne measurement twice, measuring the input signal along the orthogonal quadratures X_θ and $X_{\theta+\pi/2}$. This is done by first passing the input signal through a 50:50 beamsplitter, which splits the signal. Then the homodyne measurement is performed on each output arm using two local oscillators. These local oscillators are phase-shifted relative to one another by $\pi/2$, which measures orthogonal quadratures.

An equivalent way to perform the heterodyne measurement uses just one beam splitter. Instead of the LO being held at the frequency of the input signal, it is shifted slightly below the input frequency. Then, a photodetector is used to measure the output current. Focusing on the photocurrent that comes out at a frequency given by the difference of the input and LO frequencies, the quadratures of that output are the quadratures of the input signal.

Within the POVM formalism, the heterodyne measurement is modeled by projections

of the signal ρ_0 onto *coherent states*, often labeled as $|\alpha\rangle$. In general, α is complex number; often it is written as $\alpha = x + ip$ to demonstrate the formal correspondence with the canonical classical phase space variables x, p (192). Writing $\alpha = re^{i\theta}$, the evolution of the coherent states under the harmonic oscillator Hamiltonian $H \propto \omega a^\dagger a$ is simply a rotation in phase space:

$$|\alpha(t)\rangle = U(t)|\alpha(0) = re^{i\theta}\rangle = |re^{i(\theta-\omega t)}\rangle. \tag{4.54}$$

The coherent states saturate the Heisenberg uncertainty principle along any quadrature: $\Delta(X_\theta)\Delta(P_\theta) = 1/2$. In this sense they are the most classical states in quantum phase space.

One of the surprising properties of coherent states is that they are eigenstates of the lowering operator a: $a\,|\alpha\rangle = \alpha\,|\alpha\rangle$[7]. From this fact, it follows that they can be expanded in the Fock basis as

$$|\alpha\rangle = e^{-|\alpha|^2/2}\sum_{n=0}^{\infty}\frac{\alpha^n}{\sqrt{n!}}\,|n\rangle = e^{-|\alpha|^2/2}\sum_{n=0}^{\infty}\frac{\alpha^n}{n!}\,|0\rangle = e^{\alpha a^\dagger - \alpha^\star a}\,|0\rangle\,. \tag{4.55}$$

This final form shows that coherent states are simply *displacements* of the vacuum state $|0\rangle$.

The heterodyne measurement corresponds to measuring the *coherent-state POVM*, given by

$$\Pi_\alpha = \frac{|\alpha\rangle\langle\alpha|}{\pi}\ ,\ \int_\alpha \Pi_\alpha = 1\ ,\ \mathrm{Tr}(\Pi_\alpha\Pi_\beta) \propto e^{-|\alpha-\beta|^2}. \tag{4.56}$$

The POVM elements Π_α are *overcomplete*, as they provide a resolution of the identity operator, but are not orthogonal. When performing the heterodyne measurement on a state ρ_0, the probability of observing an outcome α is given by

$$\Pr(\alpha) = \mathrm{Tr}(\Pi_\alpha\rho_0) = \frac{\langle\alpha|\rho_0|\alpha\rangle}{\pi} \equiv Q_{\rho_0}(\alpha). \tag{4.57}$$

[7]This can be taken as the *definition* of a coherent state.

This distribution gives yet another representation of ρ_0, known as the Husimi Q-function (152).

To evaluating how well Equation (3.43) performs in comparison to $\langle\lambda(\rho_0, \mathcal{M}_d)\rangle$ under heterodyne tomography, I numerically simulated (a) heterodyne tomography experiments (i.e., data), and (b) numerically computed the ML estimates $\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}$. The next subsection provides details on this simulation, and the results are presented in Section 4.2.6.

4.2.5 Simulating heterodyne tomography

The following subsections detail numerical experiments I performed to simulate heterodyne tomography and to compare the performance of Equation (3.43) in predicting $\langle\lambda\rangle$[8].

Generating synthetic heterodyne data

I generated synthetic experimental data for the heterodyne POVM by generating an *a priori* known quantum state ρ_0. Heterodyne tomography samples from the Husimi Q-function for this state (Equation (4.57)). Given ρ_0, I used *rejection sampling* (61) to sample from $Q_{\rho_0}(\alpha)$. Rejection sampling is a technique for sampling from some desired probability distribution $p(x)$ that might be hard to sample from directly using the ability to efficiently sample from some other distribution $q(x)$. Rejection sampling works as follows: given $p(x)$ (the one we want to sample from) and $q(x)$ (the one we *can* sample from), define $r = \max_x q/p$. Draw a candidate point y from q. Then, generate a random number $u \sim \mathrm{Unif}[0, 1]$. If $p(y)/(rq(y)) \geq u$, then y is *accepted* as a sample; otherwise, it is *rejected*. Each time a fresh sample is required, a new value for u is chosen.

[8]For roughly the first year of my PhD research, I worked on developing a code base for doing these numerical simulations and processing the results. Hence the detailed description.

The Husimi Q-function is a function over $\mathbb{R}^2$, so in my simulations, I took q to be a 2-dimensional, isotropic Gaussian distribution with mean zero and standard deviation 3: $q = \mathcal{N}(0, 9 * \mathcal{I})$. This choice of q is motivated by the fact that (a) it's very easy to sample from, and (b) its tails roll off exponentially, just as the Husimi Q-distribution's do. For the coherent state POVM, there is generally no closed-form expression for r, which means that r has to be *estimated*. To do so, I started by randomly sampling 100 points $\mathbf{x}_j$ from q, estimate r as the $\hat{r} = \max_{\mathbf{x}_j} q(\mathbf{x})/p(\mathbf{x})$. As the rejection sampling algorithm proceeds, if there is a point generated $\mathbf{x}'$ such that $q(\mathbf{x}')/p(\mathbf{x}') > \hat{r}$, then I set $\hat{r} = q(\mathbf{x}')/p(\mathbf{x}')$, and remove some of the points that had been previously generated. Note that simply *dumping* the points generated up to that iteration, and starting afresh, is not tenable, unless some kind of *certificate* on $|\hat{r} - r|$ could be computed: otherwise, there may exist an even higher value of $\hat{r}$!

Computing ML estimates

The N_{samples} of heterodyne outcomes (points in phase space $\{\alpha_j\}$) constitute the *experimental data*. To compute $\lambda(\rho_0, \mathcal{M}_d)$ on this data, computing the ML estimates $\hat{\rho}_{\text{ML},\mathcal{M}_d}$ is necessary. Unlike the idealized problem considered in Chapter 3, the only known numerical algorithm for computing the ML estimate is a brute-force optimization of the likelihood function.

Recall the likelihood function $\mathcal{L}(\rho) = \Pr(\text{Data}|\rho)$; for heterodyne data, this likelihood function is

$$\mathcal{L}(\rho) = \frac{1}{\pi^{N_{\text{samples}}}} \prod_{j=1}^{N_{\text{samples}}} \langle \alpha_j | \rho | \alpha_j \rangle. \tag{4.58}$$

Because the maximum of $\mathcal{L}$ and $a\mathcal{L}$ are the same for any $a > 0$, I ignore the $1/\pi^{N_{\text{samples}}}$ prefactor in what follows. Since the models $\mathcal{M}_d$ are convex, and the likelihood function is convex, standard numerical optimizers *could* be used to compute $\hat{\rho}_{\text{ML},\mathcal{M}_d}$. Instead, I chose to write my own optimization routine, based on gradient ascent and conjugate gradient algorithms.

Gradient ascent algorithms are well-suited for maximizing functions. Given a guess for an initial starting point $(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_0$ (which I take to be the d-dimensional maximally mixed state $\mathcal{I}_d/d$), both of these algorithms ascend the likelihood function by iteratively updating the estimate for the state according to some rule until a termination condition is reached. Instead of maximizing the likelihood, I chose to instead maximize its logarithm. Note that maximizing the *logarithm* of the gradient will give the same ML estimate as maximizing the likelihood directly, as taking the logarithm preserves inequalities ($x \leq y \iff \log(x) \leq \log(y)$).

Gradient ascent algorithm

A straightforward gradient ascent algorithm updates the estimated state via the rule

$$(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_{j+1} = (\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j + \eta \nabla \log(\mathcal{L}((\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j)). \tag{4.59}$$

Here, η is a parameter that controls the *rate* at which the estimate is updated, and the gradient is taken with respect to the parameters of ρ. The logarithm of the likelihood is

$$\log(\mathcal{L}(\rho)) = \sum_{j=1}^{N_{\mathrm{samples}}} \log(\langle \alpha_j | \rho | \alpha_j \rangle), \tag{4.60}$$

and its gradient can be straightforwardly computed by writing the matrix element $\langle \alpha_j | \rho | \alpha_j \rangle$ as a trace:

$$\nabla \log \mathcal{L} = \sum_{j=1}^{N_{\mathrm{samples}}} \frac{\nabla \mathrm{Tr}(\rho |\alpha_j\rangle\langle\alpha_j|)}{\mathrm{Tr}(\rho |\alpha_j\rangle\langle\alpha_j|)}. \tag{4.61}$$

Recall that the trace is an inner product on complex matrices, so when differentiating $\mathrm{Tr}(\rho |\alpha_j\rangle\langle\alpha_j|)$ with respect to ρ, the gradient is $|\alpha_j\rangle\langle\alpha_j|$, just as $\nabla_{\mathbf{x}}(\mathbf{x}\cdot\mathbf{y}) = \mathbf{y}$. Hence,

$$\nabla \log \mathcal{L} = \sum_{j=1}^{N_{\mathrm{samples}}} \frac{|\alpha_j\rangle\langle\alpha_j|}{\mathrm{Tr}(\rho |\alpha_j\rangle\langle\alpha_j|)}. \tag{4.62}$$

Looking at the update rule (4.59), it's clear that unless $\nabla \log \mathcal{L}$ is modified in some way, the *trace* of the estimated state could change from one iteration to the next:

$$\mathrm{Tr}[(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_{j+1}] = \mathrm{Tr}[(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j] + \eta \mathrm{Tr}[\nabla \log(\mathcal{L}((\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j))]. \tag{4.63}$$

Examining Equation (4.62), it's clear $\nabla \log \mathcal{L}$ is not trace-zero. Thus, it's necessary to *subtract out* the traceful component of $\nabla \log \mathcal{L}$ prior to using it in any update step. Note that other schemes are possible for ensuring trace preservation; a common one is to divide the estimate at each iteration by its trace (293).

Importantly, this operation doesn't change the gradient in the other directions of interest in state space. Because $\nabla \log \mathcal{L}$ is a Hermitian operator, it can be expanded in any basis for Hermitian matrices (e.g., generalized Pauli matrices):

$$\nabla \log \mathcal{L} = c_0 \tau_0 + \sum_j c_j \tau_j, \quad \text{where} \quad \mathrm{Tr}(\tau_j) = 0 \text{ for } j > 0, \ \mathrm{Tr}(\tau_j \tau_k) = \delta_{jk}, \ \mathrm{Tr}(\tau_0) = d. \tag{4.64}$$

The gradient of the loglikelihood may be non-zero in the direction of τ_0 (i.e., the identity direction), and if the component c_0 was retained, then the trace of $(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_{j+1}$ would change from one iteration to the next. For this reason, I define $g \equiv \nabla \log \mathcal{L} - \mathrm{Tr}(\nabla \log \mathcal{L})\tau_0/d$, and the update rule is

$$(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_{j+1} = (\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j + \eta g_j. \tag{4.65}$$

Conjugate gradient algorithm

While gradient ascent is a simple and reliable algorithm for maximizing functions, it suffers from the problem that if the function's maximum lies on a "ridgeline", then gradient ascent can "ping-pong" back-and-forth across the ridge for many iterations. To remedy this problem, I also coded up an optimization algorithm that uses a *nonlinear conjugate gradient* update (276; 293). Unlike gradient ascent, in the conjugate gradient algorithm, the direction along which the estimate is updated depends on

the local gradient *as well as past directions*. The update rule is similar to that of gradient ascent (Equation (4.59)):

$$(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_{j+1} = (\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j + \eta s_j, \tag{4.66}$$

but s_j is *not* simply the gradient at $(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j$. Instead, it is updated as

$$s_{j+1} = s_j + \beta(G_j, G_{j-1})G_j, \tag{4.67}$$

where G_j is the gradient of the loglikelihood evaluated at $(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j$, and

$$\beta(G_j, G_{j-1}) = \frac{\mathrm{Tr}[G_j^\dagger(G_j - G_{j-1})]}{\mathrm{Tr}[G_{j-1}^\dagger G_{j-1}]}. \tag{4.68}$$

This choice for β is known as the *Polak-Ribière* choice (235). Just as I did for the gradient, I also subtract out the τ_0 component of s_j before doing the update.

A "smart" update rule: golden section search

Both the gradient ascent and conjugate gradient algorithms use update rules of the form

$$(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_{j+1} = (\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j + \eta\Delta_j, \tag{4.69}$$

where Δ_j is the gradient or conjugate gradient at iteration j. The *step size/rate* η modifies the change to the estimate Δ_j, and consequently, affects the behavior of the algorithm. Assuming η is fixed, then there's the problem that if it's set too low, the algorithm will take a long time to converge, while if it's set too high, the algorithm might needlessly go through many iterations bouncing back and forth around the maximum. For this reason, I chose to dynamically update η as the algorithm was running, by using *golden section search* (170) to choose an optimal value for η at each iteration.

To compute the maximum or minimum of a strictly unimodal function $f : \mathbb{R} \to \mathbb{R}$ (e.g., the loglikelihood function), golden section search progressively narrows down a

search interval $[a, b]$ to the extremal value $x^\star$ that extremizes f by considering subintervals $[a, x_1], [x_1, x_2], [x_2, b]$ that have the property that their ratios form a golden ratio.

In order to use this algorithm, it's necessary to identify a and b. Because the search parameter η is lower-bounded by 0 (if it's negative, the algorithm is *descending*), $a = 0$. To determine b, it suffices to consider $\log(\mathcal{L}((\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j) + \kappa\Delta_j$ for increasing values of κ. Once a value of κ is identified such that $\log(\mathcal{L}((\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j + \kappa^\star\Delta_j) < \log(\mathcal{L}((\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j)$, then because the loglikelihood is a strictly unimodal function, the extremal value of $\mathcal{L}$ lies in the interval $[0, \kappa^\star]$, and the golden section search algorithm can be used.

Dealing with the boundary of state-space

Regardless of which algorithm is used, any gradient-based update rule will run into problems when the current estimate gets close to the boundary. In particular, suppose the unconstrained ML estimate $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$ lies outside $\mathcal{M}_d$. Then the optimization rules above will fairly rapidly update the estimate to one that lies on the boundary of the model, but then subsequent updates will cause the estimate to slowly "crawl" along the boundary: at each iteration, the gradient $\nabla \log \mathcal{L}$ will point toward $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$, and its projection onto the local tangent plane of the boundary will be small. At $\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}$, the gradient will point directly at $\hat{\rho}_{\mathrm{ML},\mathcal{M}'_d}$, and its projection onto the local tangent plane will be 0. Thus, the amount of "horizontal" update that the gradient gives goes to zero as the optimizer converges on $\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}$. This poses challenges for gradient-based optimization of the likelihood near a boundary.

To remedy this, I use a simple test: given the gradient g_j, compute $(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j + \epsilon g_j$ with $\epsilon = 10^{-5}$. If this matrix is negative, then the current estimate is probably on the boundary. Thus, the gradient should be replaced by its local approximation

$$g_j \to \frac{\Pi[(\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j + \epsilon g_j] - (\hat{\rho}_{\mathrm{ML},\mathcal{M}_d})_j}{\epsilon}, \tag{4.70}$$

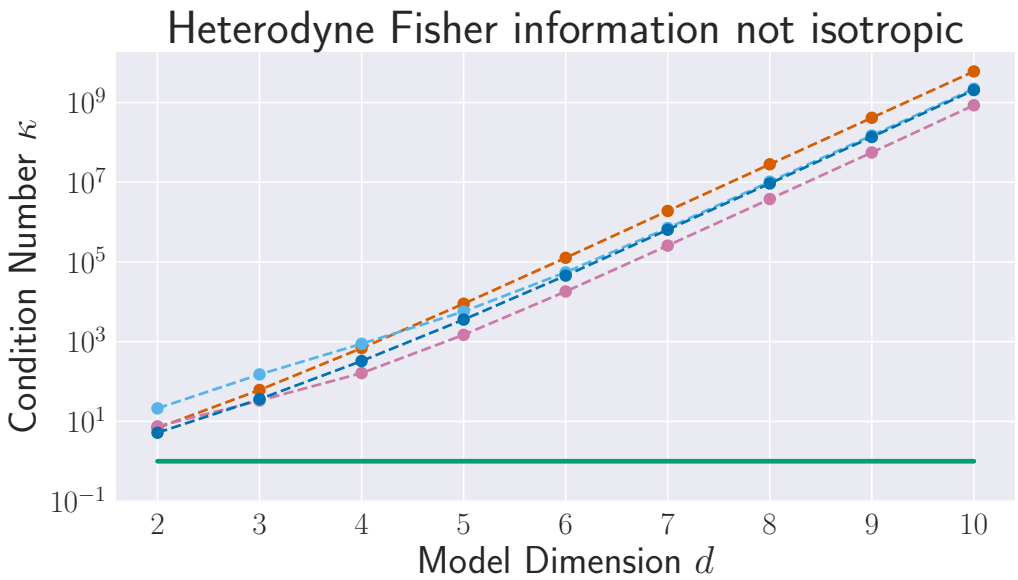

Figure 4.5: **Anisotropy of the heterodyne POVM Fisher information**. The condition number κ – the ratio of the largest eigenvalue to the smallest – of the estimated heterodyne Fisher information. (Estimates are the average over 100 Hessians of the loglikelihood function.) κ grows with model dimension, meaning anisotropy is increasing. The dashed lines indicate different states ρ_0, and the solid line is $\kappa = 1$ (i.e., $\mathcal{I} \propto \mathbb{1}$.).

where $\Pi[A]$ performs the truncation algorithm described in Section 3.5.3. This approach uses a finite-difference method to estimate the gradient along the boundary, which is the direction we want the optimizer to go anyway.

4.2.6 Results

The theory derived in Section 3.5 models ideal tomography, where the Fisher information is isotropic. In practice, this is rarely the case, particularly for the heterodyne POVM. Figure 4.5 plots the *condition number* – the ratio of the largest eigenvalue to the smallest – of the estimated Fisher information. It is clear $\mathcal{I} \not\propto \mathbb{1}$. Although the Fisher information is highly anisotropic, this turns out to be less of a problem than Figure 4.5 indicates.

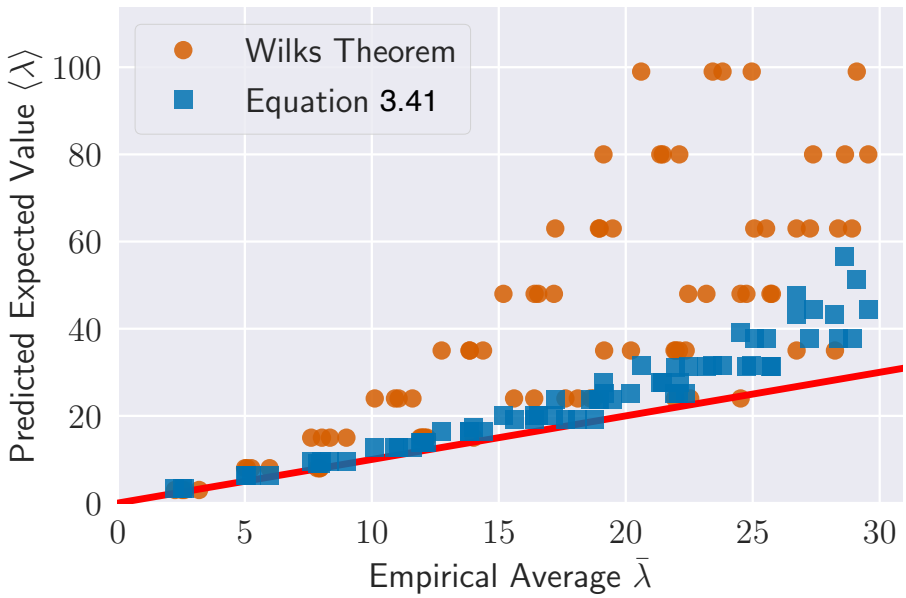

Figure 4.6: **Applying isotropic formula to heterodyne tomography**. The Wilks theorem (orange dots) dramatically over-estimates $\langle\lambda(\rho_0, \mathcal{M}_d)\rangle$ in optical heterodyne tomography. Our formula, Equation (3.43) (blue squares), is far more accurate. Residual discrepancies occur in large part because N_{samples} is not yet "asymptotically large". The solid red line corresponds to perfect correlation between theory ($\langle\lambda\rangle$) and practice ($\bar{\lambda}$).

I examined the behavior of λ for 13 distinct states ρ_0, both pure and mixed, supported on $\mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4$, and $\mathcal{H}_5$. As described earlier, I used rejection sampling to simulate 100 heterodyne datasets with up to $N_{\text{samples}} = 10^5$, and numerical optimization to compute the ML estimates $\hat{\rho}_{\text{ML},\mathcal{M}_d}$ over each of the 9 models $\mathcal{M}_2, \ldots, \mathcal{M}_{10}$. (The model $\mathcal{M}_1$ is trivial, as $\mathcal{M}_1 = \{|0\rangle\langle 0|\}$. This model will be generally be a poor choice.) For each ρ_0 and each d, I averaged $\lambda(\rho_0, \mathcal{M}_d)$ over all 100 datasets to obtain an empirical average loglikelihood ratio $\bar{\lambda}(\rho_0, d)$.

Results of this test are shown in Figure 4.6, which plots the predictions for $\langle\lambda\rangle$ given by the Wilks theorem and Equation (3.43), against the empirical average $\bar{\lambda}$, for a variety of ρ_0 and d. Equation (3.43) correlates very well with the empirical average, while the Wilks theorem (unsurprisingly) overestimates λ dramatically for low-rank

states. Whereas a model selection procedure based on the Wilks theorem would tend to falsely reject larger Hilbert spaces (by setting the threshold for acceptance too high), Equation (3.43) provides a reliable null theory.

Interestingly, as d grows, Equation (3.43) also begins to overpredict. As Figure 4.7 indicates, a more accurate description is that the numerical experiments are *underachieving*, because $\bar{\lambda}$ is still growing with N_{samples}. Both the Wilks theorem and the theory developed in Section 3.5 are derived in the limit $N_{\text{samples}} \to \infty$; for finite but large N_{samples}, both may be invalid. Figure 4.7 shows that, even at $N_{\text{samples}} \sim 10^5$, the behavior of $\bar{\lambda}$ has failed to become asymptotic. This is surprising, and suggests heterodyne tomography is a particularly exceptional and challenging case to model statistically.

However, the analysis of Section 3.5 *does* get some of the qualitative features correct. Figure 4.8 presents simulated values of $\langle \lambda_{jk} \rangle$ for an isotropic Fisher information and for heterodyne tomography. While the values of $\langle \lambda_{jk} \rangle$ do not agree exactly, they still decompose into two types, the "L" and the "kite". (See Figure 4.9 for an analysis of the discrepancies.)

The loglikelihood ratio statistic $\lambda(\rho_0, \mathcal{M}_d)$ is also the squared error between ρ_0 and $\hat{\rho}_{\text{ML},\mathcal{M}_d}$. If $\mathcal{M}_d$ is sufficiently complex (i.e., $d > \dim(\rho_0)$, then we usually expect that some parameters in $\hat{\rho}_{\text{ML},\mathcal{M}_d}$ will be estimated with non-zero values, even when the corresponding matrix element(s) in ρ_0 are 0. In turn, the squared error would be non-zero. However, if the data itself didn't provide any evidence that the matrix elements in $\hat{\rho}_{\text{ML},\mathcal{M}_d}$ *should* be estimated with non-zero values, then the squared loss would be zero.

Figure 4.8 indicates that such an effect might be at play when considering heterodyne tomography. In particular, if ρ_0 is a state with n photons, but the dimension of the model $d > n$, then it's very unlikely the tomographic data set will contain many values of α with $|\alpha|^2 > n$. In turn, $\hat{\rho}_{\text{ML},\mathcal{M}_d}$ shouldn't have much support on the

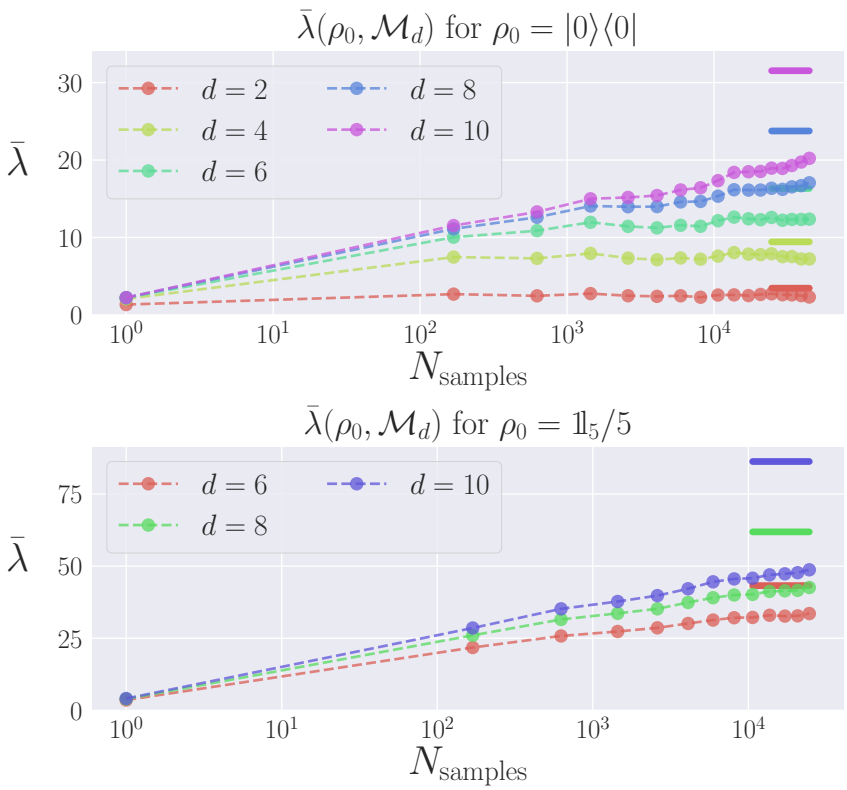

Figure 4.7: **"Underachievement" of $\bar{\lambda}$ in heterodyne tomography**. The empirical average $\bar{\lambda}$ may have achieved its asymptotic value, or is still growing, depending on ρ_0 and d. Solid lines indicate the value of Equation (3.43).

higher-energy number states, so $(\hat{\rho}_{\text{ML},\mathcal{M}_d})_{jj} \sim 0$ for $j \in [n, n+1, \cdots d]$. This is in fact the behavior observed in Figure 4.8. Because $\hat{\rho}_{\text{ML},\mathcal{M}_d} \geq 0$, if it has small diagonal elements, then the off-diagonal elements that are coherent with them must also be small. Both of these effects imply the high-photon-number matrix elements of $\hat{\rho}_{\text{ML},\mathcal{M}_d}$ are small, *which means they are close to zero, thereby driving down the squared loss* (i.e., $\langle \lambda_{jk} \rangle$).

With very few heterodyne "counts" out in the high-photon-number region of phase space, the data provide very little reason/evidence to fit the extra parameters in the model to any value besides zero. Because the estimates are essentially "pinned", they

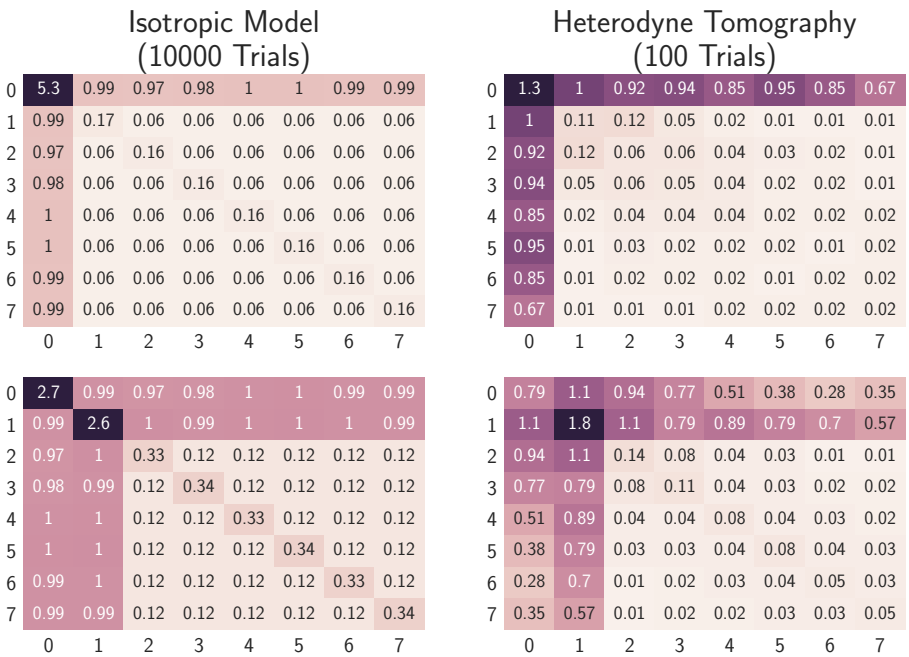

Figure 4.8: **Detailed comparison of isotropic model and heterodyne tomography**. The values of $\langle \lambda_{jk} \rangle$ for an isotropic Fisher information (left), and for heterodyne tomography (right). **Top**: $\rho_0 = |0\rangle\langle 0|$. **Bottom**: $\rho_0 = \mathcal{I}_2/2$. **Discussion**: Qualitatively, the behavior is the same, though there are quantitative differences, particularly within the kite.

don't fluctuate too much, so their contribution to the loglikelihood ratio statistic is small.

This behavior is not unique to heterodyne tomography. In fact, it also occurs for the loglikelihood ratio statistic for classical Poisson distributions (see Appendix G): when the Poisson rate parameter – which determines the expected number of counts for the process – is small, the expected value of λ is small as well. Therefore, I conjecture that the matrix elements of $\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}$ with support on higher-energy photon states $|n\rangle$ do not reach their asymptotic contribution level until *extremely* large sample sizes N_{samples}. If this conjecture is true, then at least some of Equation (3.43)'s failure to match the observed values occurs simply because the empirical datasets are not yet

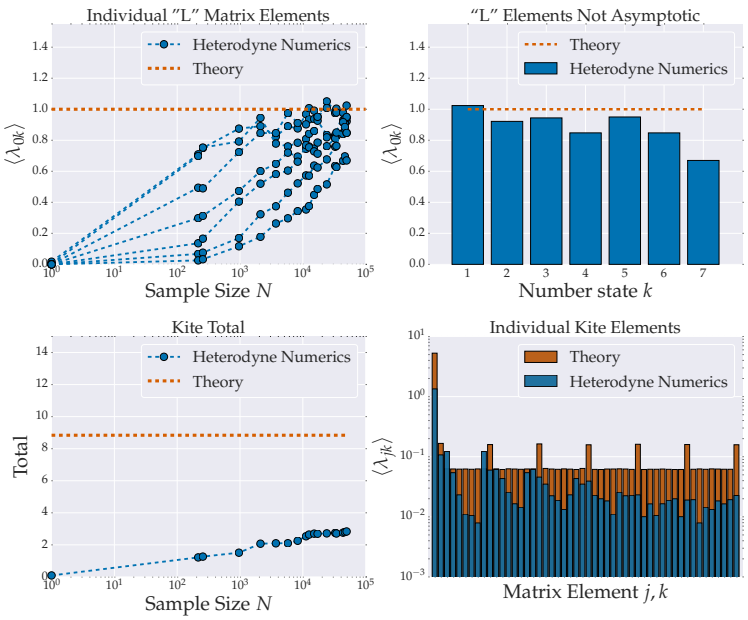

Figure 4.9: **Discrepancies between isotropic model and heterodyne tomography**. Examining how the analysis of Section 3.5 for $\langle\lambda_{jk}\rangle$ disagrees with simulated heterodyne experiments. We take $\rho_0 = |0\rangle\langle 0|$ and $d = 8$. **Top Left**: The values of $\langle\lambda_{0k}\rangle$ in the "L" as a function of N_{samples}. **Top Right**: At the largest N_{samples} studied, $\langle\lambda_{0k}\rangle$ is less than 1, especially for the higher number states. **Bottom Left**: The total from the "kite" versus N_{samples}. It is clear the total is still growing. **Bottom Right**: The individual "kite" elements $\langle\lambda_{jk}\rangle$ at the largest N_{samples} studied; most are small compared to their values in the isotropic case.

"asymptotically" large, meaning individual $\langle\lambda_{jk}\rangle$ are not yet asymptotic, causing $\langle\lambda\rangle$ to not have achieved its asymptotic value either.

4.2.7 Conclusion and discussion

Tomography of continuous-variable systems presents a challenging statistical inference problem: formally, the model is infinite-dimensional, but a finite amount of experimental data will be collected for characterizing the state. Thus, model selection techniques are essential. This section showed that the result for $\langle\lambda(\rho_0, \mathcal{M}_d)\rangle$ derived in a highly-idealized model of tomography (an isotropic Fisher information)

does capture *some* effects about the loglikelihood ratio statistic's behavior, particularly the division of the matrix elements of $\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}$ into the "L" and "kite" show up. Based on correspondence with Jonathan A. Gross, the reason for this may be that the directions along which the Fisher information is most highly anisotropic are those corresponding to the "kite" matrix elements.

The numerical experiments with heterodyne tomography I presented show unexpected behavior, indicating that quantum tomography can still surprise, and may violate *all* asymptotic statistics results. In particular, the idea that some matrix elements in $\hat{\rho}_{\mathrm{ML},\mathcal{M}_d}$ don't "turn on" until extremely large sample sizes N_{samples} has some legs, given the analysis of Poisson-distributed random variables in Appendix G. In such cases, bootstrapping (90; 144) may be the only reliable way to construct null theories for λ.

Chapter 5

Machine-learned QCVV techniques

Machines take me by surprise with great frequency. - Alan Turing (1950) (301)

Designing a QCVV technique to probe a new property of interest takes time, effort, and expertise. Machine learning (ML[1]*) can help automate the development of new QCVV techniques. In this chapter, I investigate the geometry of QCVV data sets to determine what kind(s) of surfaces can separate different types of noise. As a testing ground, I use the simple and canonical problem of determining whether a processor's dominant noise source is coherent or stochastic. I find that noise signatures in data from long circuits can reliably be separated by linear surfaces. When only short circuits, like the ones used for linear gate-set tomography, are available, feature engineering allows linear surfaces to reliably separate noise signatures.*

[1]Here and in Chapter 6, "ML" is used for 'machine learning', and *not* 'maximum likelihood' as it was in Chapters 3 and 4.

5.1 Introduction - characterizing next-generation quantum information processors

5.1.1 The need for new characterization techniques

Quantum information processors (QIPs) perform quantum circuits. Ideally, QIPs would perform specific circuits designed for a particular task (e.g., encoding classical data into a quantum state). In practice, QIPs are imperfect, idiosyncratic, and subject to noise and errors, so the circuits they run deviate from that ideal. To describe these deviations from ideal behavior, we use noise models that have many properties. Determining these properties – or more precisely, inferring them from data – is nontrivial. But doing so is necessary to diagnose problems and predict performance. The field of quantum characterization, validation, and verification (QCVV) provides a toolbox for measuring and inferring various properties (e.g. fidelity) of noisy processors. But as QIPs grow in size and quantum computing enters the "noisy, intermediate-scale quantum" (NISQ) era (237) (heralded by releases of multi-qubit processors by Google, IBM, Rigetti Quantum Computing, and others (167; 312; 333)), relying on existing tools in the QCVV toolbox becomes increasingly hard. The QCVV techniques that exist today may not work for characterizing next-generation processors (or those coming online today). (Recall Section 2.4.)

Inventing a new QCVV technique is generally a hard task, and typically corresponds to a fairly high-impact scientific paper. It requires *creativity*, *effort*, and months or years of *time*. New QCVV techniques are informed by significant domain-specific expertise, and distilling the complex behavior of a processor into a meaningful set of characterizable properties requires thoughtful effort. Is there a more efficient way to leverage such domain-specific expertise to speed up this process and help QCVV practitioners develop new QCVV techniques more rapidly? I think the answer is "yes". In this chapter, I show how to use machine learning (ML) to automate one of

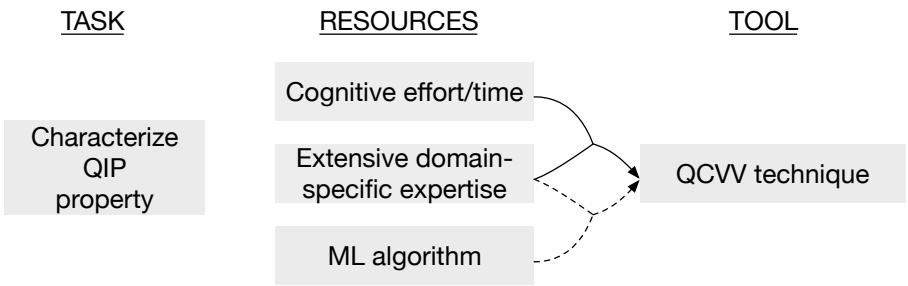

Figure 5.1: **Using machine learning (ML) to characterize quantum information processors (QIPs)**. Developing a quantum characterization, validation, and verification (QCVV) technique for a QIP characterization task currently requires time, effort, and domain-specific expertise. In this chapter, I show how ML can make this task easier. ML algorithms are not, themselves, QCVV techniques. Instead, algorithms *create* QCVV techniques if they are given the right ingredients. The most important of these is a specification of the task (a property of the QIP to be inferred); in *supervised learning*, this property is presented to the ML algorithm as a label on training data.

the more challenging parts in creating a QCVV technique (see Figure 5.1).

I start by explaining what ML can (and can't!) reasonably do in this area. Then, I outline a set of steps for formulating some QCVV tasks as *supervised learning* problems, and identify the key ingredients that need to be specified before an ML algorithm can be deployed (Section 5.2). I then demonstrate this process by going through the steps and using several ML algorithms for supervised learning to create a QCVV technique for determining whether a single-qubit QIP suffers from (1) coherent errors, or (2) stochastic errors (Sections 5.3 and 5.4). I examine the performance of several different ML *pipelines* – an ML algorithm together with pre- and post-processing of the data – and show that the *geometry* of QCVV datasets governs which algorithms can achieve high accuracy. *Feature engineering* (a rich form of pre-processing) can help linear classification algorithms learn how to distinguish coherent from stochastic noise using QCVV data. Section 5.5 concludes this chapter with an outlook on the viability of machine-learned QCVV for NISQ processors.

5.1.2 Advantages and disadvantages of using machine learning (ML) for QCVV tasks

Broadly speaking, machine learning (ML) is a field within computer science that focuses on creating and using algorithms for finding patterns in large data sets (266; 140; 119; 156). ML has already been used with great success in a variety of scientific disciplines (16; 68; 80; 327; 260; 102; 213); readers interested in the intersection of ML and quantum computation/information may appreciate the growing list of papers available at (118).

An operational definition for "learning", given in reference (216), is:

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."

Most QCVV tasks can be summarized as "Given some data, infer some property about the QIP that produced the data." A "property" is anything that describes the processor's behavior. Examples include T_1 and T_2 times, process matrices that describe gates, average fidelities, and the presence (or absence) of leakage. Interesting, relevant properties are quite diverse – they can be binary, real-valued, matrix-valued, or qualitative.

QCVV techniques that address tasks like this require answering two questions: "What kind of experimental data will be collected?" (the answer constitutes an *experiment design*) and "How will the data be processed to infer the property of interest?" (the answer constitutes a *data analysis* pipeline).

Once the property of interest is specified, ML algorithms can generate (or at least help to generate) answers for both of these questions. The focus in this chapter is on data processing, and on using ML algorithms to automatically generate good maps from data to inferred properties. Chapter 6 discusses how ML could also be used

to construct experiment designs. In this chapter, I will assume that an expert has already specified the experiment design – *and* that the data from those experiments ($\mathcal{D}$) is in fact sufficient to infer the property of interest ($\mathcal{P}$). With the experiment design specified, solving the characterization problem "Infer $\mathcal{P}$ from $\mathcal{D}$" requires finding a map $f : \mathcal{D} \to \mathcal{P}$ that yields the right answer with high probability.

For most existing QCVV techniques, this map is constructed by a theorist using statistical theory. There are several steps. First, the theorist posits a parameterized statistical model $\mathcal{M}(\boldsymbol{\theta})$ for the QIP. For each value of the parameters $\boldsymbol{\theta}$, this model predicts the probability distribution of the observed data, or a coarse-graining of it (e.g., the model in randomized benchmarking predicts certain averages, rather than individual circuit probabilities). Then, the theorist chooses an estimation procedure (e.g. maximum likelihood estimation, or Bayesian inference) that will map the data $\mathcal{D}$ to an estimate $\hat{\boldsymbol{\theta}}_{\mathcal{M}}(\mathcal{D})$ of the model parameters that describe the QIP that generated the data. Finally, the value of the property $\mathcal{P}$ can be inferred as its value for the estimated parameters: $\hat{\mathcal{P}} = P(\hat{\boldsymbol{\theta}}_{\mathcal{M}}(\mathcal{D}))$. For example, if $\boldsymbol{\theta}$ is a process matrix describing a gate, and the property of interest $\mathcal{P}$ is "Is the gate's error coherent?", then the inferred answer is "yes" if $\hat{\boldsymbol{\theta}}$ corresponds to an undesired unitary rotation. This approach is illustrated in Figure 5.2.

"Statistics-based QCVV techniques" have several drawbacks, all of which center around the statistical model. The entire approach relies critically on a wise, informed choice for $\mathcal{M}(\boldsymbol{\theta})$. There is no unique, obvious choice. George Box famously observed "All models are wrong; some are useful." (44). If the model is too complex, and has too many parameters, then statistical inference will be inaccurate, and prone to false detection of effects. But overly simple models are worse; if the model is not rich enough to capture the QIP's behavior and fit the data, then all conclusions drawn from it are suspect – the final estimate of the property will be biased. Selecting between models is something of a dark art. Finally, a model that captures the

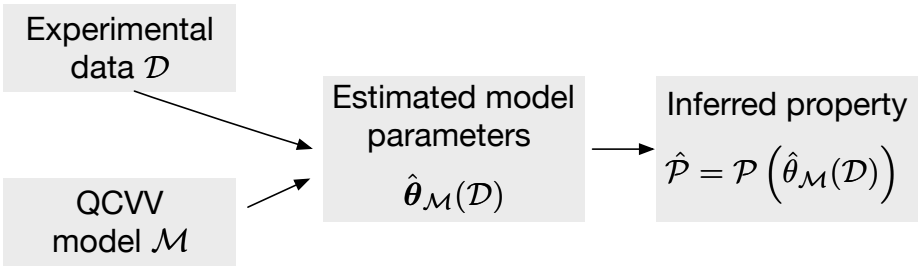

Figure 5.2: **Structure of statistical QCVV techniques**. Most existing QCVV protocols rely on statistical inference to approximate the relationship between a QIP's property $\mathcal{P}$ and experimental data $\mathcal{D}$. A theorist constructs a statistical model $\mathcal{M}$ and uses $\mathcal{D}$ to estimate its parameters, yielding $\hat{\boldsymbol{\theta}}_{\mathcal{M}}(\mathcal{D})$. Then the inferred value of $\mathcal{P}$ is simply $\mathcal{P}\left(\hat{\boldsymbol{\theta}}_{\mathcal{M}}(\mathcal{D})\right)$.

desired property has to exist or be invented by a creative theorist before statistical machinery can be deployed.

"Machine-learned QCVV techniques" are quite different, and require a distinct approach to their development. Essentially (explored in much greater detail below), an ML algorithm creates a data analysis function $f : \mathcal{D} \to \mathcal{P}$ by using a large collection of *training examples* (data sets generated by QIPs with known values for the property) which effectively describe the property of interest, to search or optimize over a large *hypothesis class* of candidate functions, and find one that (1) works reliably on the training examples, and (2) satisfies some robustness criterion that ensures this reliability will extend to future, as-yet-unseen data generated by a QIP whose value for the property needs to be inferred.

Why might this be expected to work? ML algorithms are known to be particularly good at learning approximations to functions. So *if* a good f exists – e.g., one that could be derived via statistics – then some ML approach should be able to find a good approximation to it. Some ML algorithms are in fact *universal function approximators* (79), and can approximate arbitrarily well any given function (subject to some mild regularity conditions). Furthermore, the ML toolbox contains well-

established methods for ensuring (and verifying) that the approximation to f learned by the ML algorithm will *generalize* to make sensible inferences on new data. Like statistical model selection, these techniques mitigate the risk of overfitting.

The ML approach has two major advantages. First, ML algorithms do not rely at all on (statistical) models. Instead, they infer *inductively*, by generalizing from a large array of sample cases (training data) by searching over a hypothesis class. So there is no obligation to come up with a good model, and no risk of choosing a bad one. Second, because ML algorithms only care about the relationship between $\mathcal{D}$ and $\mathcal{P}$, the *same* algorithm can be applied across a wide variety of characterization tasks. A single ML algorithm can learn how to solve many different QCVV problems, as long as it is supplied with a representative training set for each. This shifts the burden of effort from statistics (which requires fine reasoning about hypothetical data that might be observed) to simulation (generating reams of training data).

This ML approach isn't a panacea; ML algorithms come with several costs. First, the technique learned by the algorithm may not make any sense (to humans), and therefore not lend itself to insight or generalization. This is particularly true for powerful algorithms such as neural networks (210; 94; 146). Second, ML algorithms require access to data ("experience E"). Generating data for ML algorithms to learn on can require lots of experiments and/or compute cycles. Computer-generated ("synthetic") data might become extremely costly once NISQ QIPs achieve quantum supremacy/advantage and become effectively unsimulatable on classical computers. Third, while ML algorithms do not rely on a model, they do represent a hypothesis class (a set of functions), which can have the same of issues that affect the statistical models (e.g., the hypothesis class has to be rich enough to solve the problem, but simple enough to be learned efficiently). The hope in using ML instead of statistics is that finding a good hypothesis within the hypothesis class (i.e., learning a high-accuracy classifier) may be easier than finding the right model.

5.1.3 Related work

This work focuses on how *classical* ML algorithms can process *classical* data that comes from to performing experiments on a *quantum* information processor. Within the domain of "quantum machine learning" research (269; 30), this work falls in the "quantum system/classical algorithm" case, in contrast to other work that focuses on using quantum algorithms to process classical or quantum data (198; 197; 139).

There has been much research in recent years addressing problems in this "Q/C" case (150; 306; 59; 65; 319; 334; 124; 331; 130). The line of work developed here is unique in that I use classical ML algorithms to develop a new characterization technique, as opposed to deploying ML algorithms for known characterization tasks.

5.1.4 Problem statement and key results

In this chapter, I consider the problem of using supervised classification algorithms to learn a high-accuracy classifier for determining whether the noise affecting a single qubit is coherent or stochastic. (This problem is a special case of the more general problem of *estimating the coherence of the noise*, for which there are known QCVV techniques (313; 96).) I will show that this problem can be solved using "off-the-shelf" classifiers that learn from the estimated outcome probabilities of the circuits prescribed by gate set tomography (GST). I also find that *linear* classification algorithms – ones that learn separating hyperplanes – can have comparable performance to nonlinear algorithms, but only if *feature engineering* is used to represent the data in a nonlinear way. Finally, I show that the linear support vector machine (SVM) algorithm, which learns a hyperplane that is robust under small perturbations of the data, can be used to classify noise even in the presence of finite-sample errors. These results show that developing a machine-learned QCVV technique can be easier than the traditional, statistics-based approach to QCVV.

5.2 How to use ML for developing new QCVV techniques

A QCVV technique is a way to infer a property of a QIP from the experimental data it generates. Machine learning is a diverse field with many applications, so there are surely many ways that ML could contribute to QCVV. This section lays out one particular rubric – which is somewhat general, but not universal – for using supervised learning to build a QCVV technique.

The focus here is on supervised learning because, in general, algorithms for supervised learning infer a function from data to QIP property using data sets with known properties, whereas those for unsupervised learning *discover* structure within data. The canonical QCVV task is to measure, estimate, or infer a specific property (e.g., fidelity, coherence, leakage rate, etc). That the subject of interest is the property itself suggests that supervised learning is more relevant.

There are 7 general components necessary to develop a "machine-learned QCVV technique":

1. $\mathcal{P}$, the property of interest,
2. An experiment design that determines the kind and format of the data $\mathcal{D}$,
3. An embedding of $\mathcal{D}$ into a *feature space* $\mathbb{R}^n$.
4. A data processing pipeline for learning a map $f : \mathcal{D} \to \mathcal{P}$, centered around an ML algorithm $\mathcal{A}$.
5. A metric of success, used to evaluate candidate solutions against the training data.
6. Values (or a search protocol) for the *hyperparameters* that control how the algorithm behaves.

7. A collection of labeled training data from which the algorithm can learn.

The property $\mathcal{P}$ is the answer to the question "What do I want to know about the QIP?". can be many things – QIP properties include T_1 time, logical error rates, two-qubit gate fidelities, etc. In supervised learning, the property is used to label the training data: for each value of $\mathcal{P}$, some training data will be generated, and these effectively define the property, by allowing the ML algorithm to recognize what data are "typically" generated by a QIP with that value of $\mathcal{P}$. Unsupervised learning is not always focused on properties, but it can be – we might be interested in a general question about the structure of the data, such as "Is data collected today consistent with historical data collected from this device?".

In this chapter, the experiment design is taken as a given. It constitutes a description of what experiments will be run, in what order and arrangement, at whatever level of granularity is necessary. The result of running these experiments is a single dataset $\mathcal{D}$. QCVV experiments are described by the ideal quantum circuits to be performed by the QIP; the QIP performs these circuits (usually with noise and errors), and generates outcomes. So "data" means "the outcomes of various quantum circuits". If the QIP's behavior is assumed to be stationary, then it is sufficient for $\mathcal{D}$ to consist of the aggregated outcome frequencies of the circuits. Except for Section 5.4.5, I consider the "infinite-sample" limit, so that the outcome frequencies are the outcome probabilities. If not, then $\mathcal{D}$ would need to consist of the time-stamped outcomes for each repetition of a each circuit.

Third, to make experimental data usable by ML algorithms, it needs to be embedded in a *feature space* $\mathcal{F}$. This is typically isomorphic to $\mathbb{R}^n$ for some n. The embedding is described by a feature map $\phi : \mathcal{D} \to \mathbf{f} \in \mathcal{F}$. ϕ takes a QCVV data set and maps it into a *feature vector* $\mathbf{f}$. Different feature maps can yield very different feature spaces, which impacts the performance of ML algorithms. Varying the feature space and the feature map intentionally is called *feature engineering*, and its effects are investigated

in some detail in Section 5.4.3.

Fourth is the choice of ML algorithm $\mathcal{A}$. The choice for $\mathcal{A}$ depends strongly on $\mathcal{P}$ (and also the learning paradigm). If $\mathcal{P}$ is a binary property ("Is the noise entirely coherent or entirely stochastic?"), then $\mathcal{A}$ should be a binary classification algorithm. For continuous properties such as error rates, a regression algorithm would be used. ML algorithms can be complex, but at heart they consist of parameters $\boldsymbol{\theta}$ that index a hypothesis for the relationship between $\mathcal{D}$ and $\mathcal{P}$, and are updated in response to data (training) to find an element of the hypothesis class that can successfully predict the property.

Fifth, a metric of success is necessary. Since ML algorithms learn inference tools, that metric should measure the quality of the inference. For supervised learning tasks, a *loss function* is typically used to quantify the penalty for incorrect inference of $\mathcal{P}$. For binary or discrete classification, the loss function is usually "0/1" (a penalty of 1 is applied for an incorrect classification; no penalty is applied for a correct one), while for regressing a continuous parameter the loss function would typically quantify the distance between the inferred value of $\mathcal{P}$ and its true value. During training, ML algorithms update their parameters $\boldsymbol{\theta}$ to minimize the loss.

Sixth, most ML algorithms are configurable via user-controllable hyperparameters, and an algorithm's performance (i.e., how good a QCVV technique it learns) will depend on their values. In simple cases the hyperparameters can be specified *a priori*. In others specifying a procedure for varying hyperparameters to find good values is necessary. For the example problem discussed in Chapter 5.3, the solution quality turns out to be strongly dependent on hyperparameters.

Finally, one of the most important ingredients is a collection of *training data* $\mathcal{C} = \{(\mathcal{D}_1, \mathcal{P}_1), (\mathcal{D}_2, \mathcal{P}_2), \cdots\}$ that the ML algorithm can learn from. Each data set in the collection $\mathcal{D}_j$ carries with it a label defined by the property of interest $\mathcal{P}_j$. The collection is used by the algorithm to identify a good QCVV technique (i.e., a good

hypothesis out of the hypothesis class available to it.) Some of the training data is also withheld (temporarily) to be used in testing the technique that is identified. This *cross-validation* is essential for validating that the technique generalizes well, confirming that the training data were large enough, and setting expectations for how well the tool might work in the future. For QCVV purposes, I envision this training data being generated via simulation, not from actual experiments. However, in the situation where we wanted to detect an unknown effect observed previously in another experiment, the training data might constitute data from that previous experiment.

Figure 5.3 gives a pictorial description showing how these components relate to one another. Each needs to be specified to develop an ML-learned QCVV technique. Note that the first two (property and experiment design) need to be specified for *any* QCVV technique. Several of these ingredients could, at least in principle, be discovered themselves by machine learning: the experiment design (at least partly, via reinforcement learning), the feature map ϕ (via automated feature engineering), and the choice of the algorithm's hyperparameters (via automated hyperparameter tuning).

The next section demonstrates exactly how to specify these components and then deploy ML to generate a simple QCVV technique for a simple but interesting problem. Section 5.4 will show how to evaluate ML algorithms using cross-validation, demonstrate the importance of hyperparameter tuning, and introduce *feature engineering* as a way to enhance ML with domain-specific expertise. Finally, Section 5.5 concludes by discussing the viability of "machine-learned QCVV" for near-term processors.

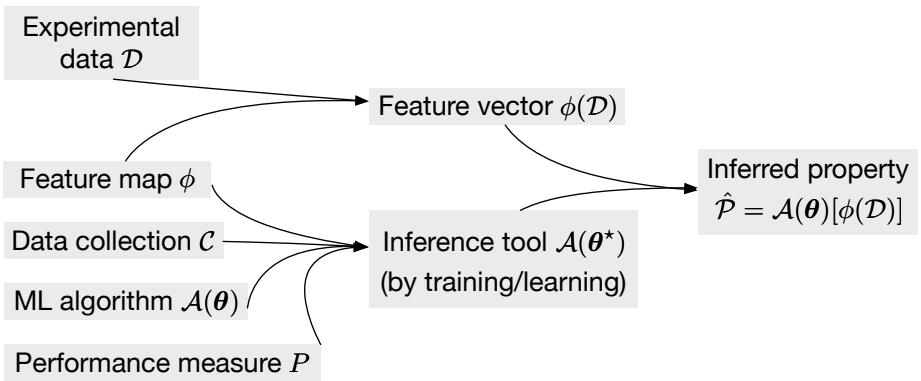

Figure 5.3: **Using ML to develop new QCVV techniques**. Using ML to build a QCVV protocol that can infer a QIP property $\mathcal{P}$ from data requires different inputs from (e.g.) traditional statistical QCVV techniques (compare with Figure 5.2). In particular, the statistical model is replaced by an ML algorithm $\mathcal{A}$ that defines a *hypothesis class* of candidate functions $f : \mathcal{D} \to \mathcal{P}$. The inference method is fundamentally different - instead of defining f implicitly via estimation of a model's parameters, the result of the ML design process is an atomic analysis map $f : \mathcal{D} \to \mathcal{P}$ that was learned directly from example data $\mathcal{C}$.

5.3 A machine-learned QCVV technique for distinguishing single-qubit coherent and stochastic noise

Although a reasonable conjecture is that supervised ML could be applied successfully to arbitrarily complex QCVV problems – e.g., estimating a full gate set, as gate set tomography (37; 125; 38) does – the work presented here focuses on a much simpler yet still interesting task: *estimating the coherence of noise* (313; 96). Gate errors are often implicitly assumed to be *stochastic*, but it is known that they can also be coherent (general noise is a mixture of stochastic, coherent, and other types of error). The coherence of the noise impacts the relationship between average and worst-case error rates (182), and potentially the performance of quantum error-correcting codes (18; 133; 77; 314; 273). Thus, determining whether errors are coherent or stochastic

is an important step in assessing how well a QIP could perform error correction (and probably other tasks). Recently, QCVV protocols have been introduced that can accurately detect and measure coherence. Some are based on randomized benchmarking (313; 96). Gate set tomography can also efficiently measure the degree of coherence.

One way to estimate the coherence of a QIP's noise is to apply circuits that randomly sample from a unitary 2-design (e.g. Clifford circuits), and then estimate expectation values of particular observables (313). This "unitarity benchmarking" protocol is reasonably straightforward – but its derivation very clearly demanded extensive subject matter expertise! Here, ML is used to solve a simpler version of the same problem: instead of estimating *how* coherent the noise is, the task is to classify whether it is entirely coherent, or entirely stochastic (incoherent). This is a *classification* problem, and estimating the coherence quantitatively is a closely related *regression* problem. I demonstrate that ML algorithms can learn how to classify coherent and stochastic noise. Although this does not guarantee that ML algorithms can perform regression too, it does suggest that doing so could be possible (although different algorithms may be required). I leave that topic for future work. As noted in Section 5.1, characterizing NISQ QIPs is challenging for many reasons. The goal here is to prove the principle, so I simplify by considering a single-qubit processor.

The next subsections specify each of the components identified in the previous section.

5.3.1 Property $\mathcal{P}$: "Are the gate errors coherent or stochastic?"

A gate-based QIP implements quantum circuits by compiling the circuit into primitive operations (gates), and executing those gates in the order specified by the compiler. The property we want to extract is a property of these primitive operations.

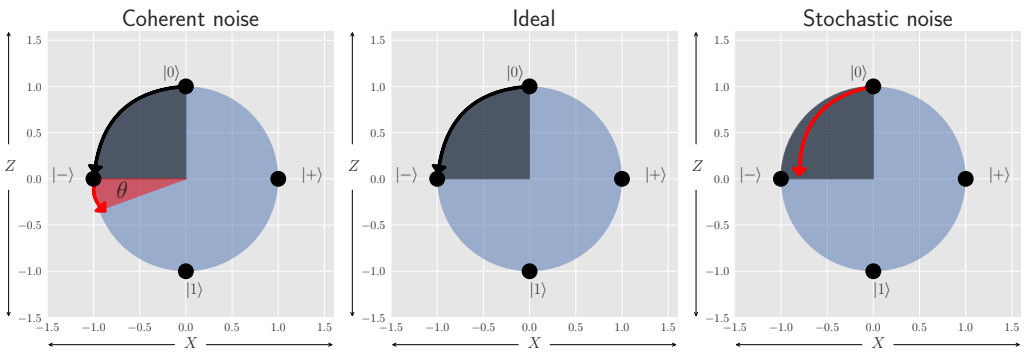

Figure 5.4: **Example of how different types of gate noise affect circuit outcome probabilities**. Suppose that $\rho_0 = |0\rangle\langle 0|$, $G_0 = R_Y(\phi)$ is a rotation about the Y-axis by an angle $\phi = \pi/2$, and the POVM is $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$. Under G_0 (center), $\Pr(0) = \Pr(1) = .5$. If instead the applied gate is a coherent over-rotation $r_Y(\theta) = R_Y(\pi/2 + \theta)$ (left), then $\Pr(0) = (1 - \sin\theta)/2 \neq .5$, in general. This same outcome probability of $1/2$ can arise from stochastic noise, $\mathcal{G}_0 \to \mathcal{E}[\rho] = (r_Y(\theta)\rho r_Y^\dagger(\theta) + r_Y(-\theta)\rho r_Y^\dagger(-\theta))/2$ (right), although the effect of this noisy channel on the initial state is different – it decreases the purity of ρ_0, provided $\theta \neq \pm\pi/2$.

Ideal gates on a qubit are described by 2×2 unitary matrices U. For a more general model that encompasses Markovian errors, gates are described by 4×4 *completely positive trace-preserving maps* (CPTP maps) that act linearly on density matrices. A model that assigns a CPTP map to each primitive operation of a QIP is a *gate set*. The single-qubit processor considered here has five operations: initialization (ρ), measurement (M), and three logic gates corresponding to idling and $\pi/2$ rotations around the X and Y Bloch axes (G_I, G_X, and G_Y, respectively).

An ideal unitary gate G_0 is described by a CPTP map that acts as $G_0[\rho] = U\rho U^\dagger$. The real gate $\mathcal{E}$ will deviate from G_0. This deviation is the *error* in the gate, and a variety of errors are possible. This chapter focus on two specific classes of errors, which are described by their *generators*, as follows.

The ideal unitary U can be described by the 2×2 Hamiltonian H_0 that generates it as $U = e^{-iH_0}$. The ideal CPTP map G_0 is a different representation of the same

unitary action. It can also be described by a generator as $G_0 = e^{\mathcal{H}_0}$, where now $\mathcal{H}_0$ is a 4×4 matrix that acts on density matrices as $\mathcal{H}_0[\rho] = -i[H_0, \rho]$.

If $\mathcal{E}$'s error (deviation from G_0) is purely *coherent*, then it still acts unitarily, but with an additional Hamiltonian term:

$$G_0[\rho] \to \mathcal{E}[\rho] \equiv e^{-i(H_0+H_e)}\rho e^{i(H_0+H_e)}. \tag{5.1}$$

It is easy and useful to write this using the *generator* of the error, which is of the same kind as the generator of ideal dynamics:

$$\mathcal{E} = e^{\mathcal{H}_0+\mathcal{H}_e}. \tag{5.2}$$

An error is purely coherent if and only if it is of this form. Coherent errors often result from imperfect calibration of classical control fields.

"Stochastic" noise is widely understood and discussed, but does not seem to be precisely defined anywhere. A particular definition is given for purely stochastic noise that is generally consistent with usage in the quantum computing community (see below). The concept to be captured is this: stochastic errors are what occur when control fields are *fluctuating* around the desired value in a random way, but the *expected value* of those fluctuations is zero.

This concept is most straightforwardly seen when considering an imperfect idle operation (where the target unitary is $\mathbb{1}$). This operation has purely stochastic errors if its noisy version can be written as a convex sum of unitary operations *and* it is invariant under time reversal:

$$\mathcal{E}[\rho] = \sum_{j=1}^{n} w_j U_j \rho U_j^\dagger, \ \mathcal{E} = \mathcal{E}^\dagger. \tag{5.3}$$

This definition can be generalized to nontrivial unitary gates $G_0 = e^{\mathcal{H}_0}$ via the generators for stochastic noise that appear in the canonical Lindblad equation. Letting $\mathcal{S}$ represent a parameterized generator, then a noisy gate is computed as

$$\mathcal{E} = e^{\mathcal{H}_0+\mathcal{S}}. \tag{5.4}$$

Full details are given in Appendix C.2. Stochastic errors can have several causes, including fluctuations of the classical control fields and weak coupling to a rapidly mixing quantum bath.

Both of these noise models are *Markovian* - for a fixed realization of the noise, when G_0 is applied more than once in a circuit, the *same* CPTP affects each application. Figure 5.4 shows examples of how purely coherent or purely stochastic noise affect the outcome probabilities of a simple circuit for a single qubit. For details on how random realizations of stochastic and/or coherent error are generated, see Appendix C.2. In generating realizations of these two noises (e.g., for training data or for testing purposes), the *strength* of the errors is controlled with a parameter η. The coefficient or "rate" r that multiplies each generator of stochastic or coherent noise is randomly distributed, and expected value of its magnitude is $\langle |r| \rangle = O(\eta)$. For small η, this means that $|\mathcal{E} - G_0| = O(\eta)$ as well.

5.3.2 Experiment design: gate set tomography (GST) circuits

I do not try to produce a highly optimized set of circuits adapted specifically to the task of distinguishing coherent from stochastic noise. Instead (and in keeping with the general spirit of ML), I want a general purpose experiment design that should at least in principle provide information about almost *any* property. (That way, the property of interest is guaranteed to be inferrable from experimental data.) For this reason, the circuits used for gate set tomography (GST) (37; 125; 38) are used. They are intended to capture all aspects of Markovian noise, because GST seeks to completely reconstruct the gates' process matrices, and so its experiment design must be sensitive to everything about that noise. This experiment design is thus sensitive to the binary "coherent vs. stochastic" property of the noise. Therefore, the data I consider are the outcome probabilities (or, in the case of real finite-sample

data, the observed *frequencies*) of the circuits prescribed for GST.

Each GST circuit c_j is of the form $F_M(g_m)^l F_S$ where F_S, F_M, and g_m are short subsequences comprised of the elementary gates from the QIP's gate set. GST doesn't actually prescribe a single, specific set of circuits, because the protocol has some configurable parameters. One is the *maximum depth* of the experiment design, L. If L is increased, then new circuits are added to the experiment design, by adding new values of the l parameter in the circuit description above. These circuits are designed so that the GST experiment design is not just sensitive to every observable parameter of the gate set, but *increasingly* sensitive as L increases – i.e., every parameter gets amplified. For a fixed value of L, the circuits in the GST experiment design have depth at most $L + O(1)$.

For a given gate set (i.e., QIP), each circuit c_j has an outcome probability p_j that describes the random result of the circuit's terminating measurement M. For single-qubit GST, M is a 2-outcome POVM, with outcomes "up" and "down". In experiments, one of these outcome probabilities (say, for the "up" outcome) has to be *estimated* by repeating c_j a total of N_{samples} times and counting the number of times the "up" outcome is observed

$$\hat{p}_j = f_j \equiv \frac{\text{Number of times "up" seen when } c_j \text{ was run}}{N_{\text{samples}}}. \tag{5.5}$$

For a given circuit family, a GST data set $\mathcal{D}$ is a list $[(c_1, f_1, N_{\text{samples}}), (c_2, f_2, N_{\text{samples}}), \cdots, (c_d, f_d, N_{\text{samples}})]$. Here, d is the total number of circuits (*not* the Hilbert space dimension, which is 2 throughout this chapter!), and depends on the family parameter L.

5.3.3 Feature space $\mathcal{F}$: the unit hypercube

A GST data set ($\mathcal{D}$) is usually presented as a list of count statistics, one for each circuit. But ML algorithms (e.g., classifiers) represent data as *feature vectors*. So

GST data sets need to be mapped into a *feature vector* $\mathbf{f}$ in a *feature space* $\mathcal{F}$. Any way of doing so is called a *feature map*, $\phi : \mathcal{D} \to \mathbf{f} \in \mathcal{F}$.

The simplest and most obvious feature map is the one that takes the estimated outcome probabilities for each circuit and arranges them into a vector:

$$\phi(\mathcal{D}) = \mathbf{f} \equiv [f_1, f_2, \cdots, f_d]. \tag{5.6}$$

ϕ defines a feature space that is the unit hypercube in $\mathbb{R}^d$:

$$\mathcal{F} = [0, 1]^d. \tag{5.7}$$

The dimension of $\mathcal{F}$ is d, the total number of circuits in the GST experiment. It is not fixed, and varies with the GST experiment design. In particular, d varies with L, the parameter that sets the maximum length of circuits used for GST. If $\mathcal{D}$ only contains circuits from the $L = l$ family, I call the feature vector $\phi(\mathcal{D})$ an "$L = l$ GST feature vector". Increasing L (typically by factors of 2) adds new, longer circuits that amplify the noise. It also increases d, albeit relatively slowly (see Figure 5.5 and Table 5.1). Even the largest GST experiments considered here are quite manageable for ML algorithms, which can learn well on feature spaces that have dimension up to $d = 10^5$. Figure 5.6 illustrates GST feature vectors, showing how their components (outcome probabilities of a given circuit) respond to particular realizations of coherent and stochastic noise.

The "base" feature space defined above is just a starting point. In principle, it contains all the necessary information for classifying the noise (since the GST circuits are designed to capture and amplify every property of Markovian noise). But that information may not be *easily accessible* to ML algorithms – especially ones like linear classifiers that rely on a certain structure – and it may be highly redundant. There are two canonical techniques for pre-processing data to enhance ML performance. *Feature engineering* means augmenting the feature space with new, (possibly nonlinear) functions of existing features. This adds no new information,

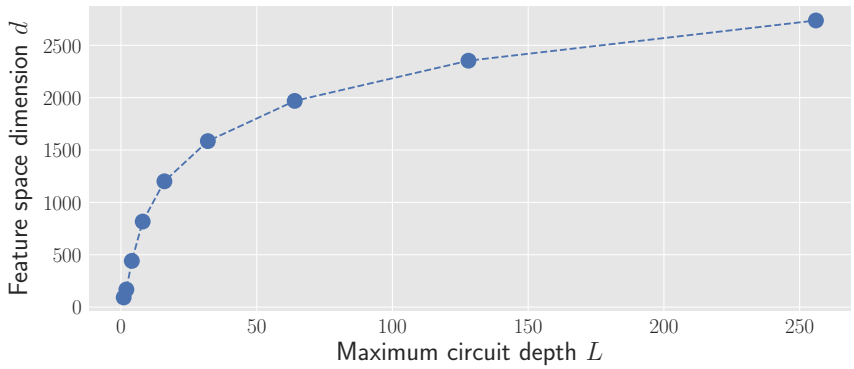

Figure 5.5: **The feature space dimension d grows with the GST circuit family parameter L.** As L is increased, the number of circuits in the GST data set $\mathcal{D}$ grows. The dimension of the feature vector $\phi(\mathcal{D})$ also grows. However, this growth is roughly logarithmic in L, and even at the largest value of L considered here, the feature space dimension is not too large for the ML algorithms used.

but it changes the geometric structure of the problem, and can make properties of the base feature space accessible to linear classifiers. *Feature selection* means throwing out redundancies and unnecessary features, by determining which features are useful or necessary for a given task and only keeping them. Feature engineering is used later in this chapter, to allow linear classifiers to capture properties that are provably not accessible to them in the base feature space. Feature selection is not used here. (See Chapter 6 for research showing that feature selection is generally possible for a wide variety of Markovian noise models.)

5.3.4 Algorithm $\mathcal{A}$: supervised binary classifiers

Distinguishing between coherent and stochastic noise is naturally posed as a binary classification problem. This chapter considers and explores several supervised learning algorithms for solving this problem. The general task of supervised learning is: "Given a collection of feature vectors $\mathcal{C} = \{(\mathbf{f}_j, y_j)\}_{j=1}^N$, with $y_j \in \{\pm 1\}$ indicating which *class* $\mathbf{f}_j$ belongs to, learn (find) a *classifier* $c : \mathbf{f} \to \{\pm 1\}$." All such algorithms

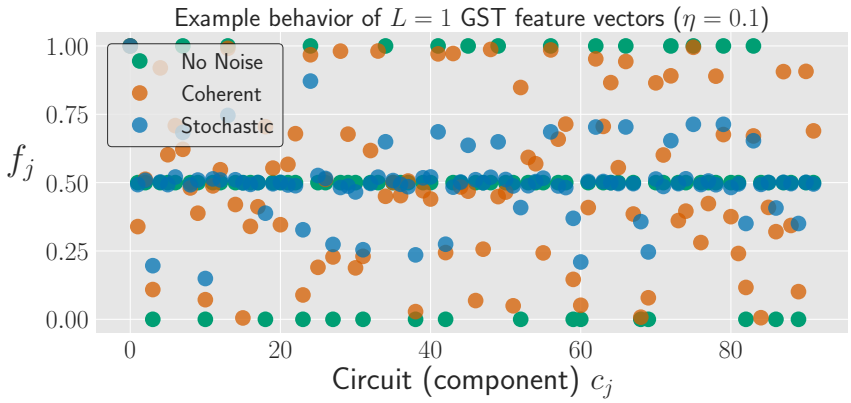

Figure 5.6: **GST data sets as feature vectors**. The $L = 1$ feature vectors for a noiseless gate set (green), a gate set where each gate has been affected by an independent coherent error (orange), and a gate set where each gate has been affected by an independent stochastic error (blue) is plotted. Both realizations of the noise had $\eta = 0.1$.

seek a classifier that performs well on the training data $\mathcal{C}$. They differ in (1) how they search for good decision rules, and (2) how they try to avoid overfitting.

A good measure of a supervised learning algorithm's performance is the accuracy A with which they infer the class label. There are many measures of accuracy for supervised learning problems; here, I use

$$A = \begin{cases} 1 & \text{if } \; c(\mathbf{f}_j) = y_j \\ 0 & \text{otherwise} \end{cases} , \tag{5.8}$$

for which the *average* accuracy is between 0 and 1 and equals the probability of correct classification. The quantity $1 - A$ is the "0/1 loss" alluded to in Section 5.2.

Every binary classification algorithm learns a classifier that divides the feature space into two parts. *Linear* classifiers are particularly simple; the boundary that separates the feature space is an affine hyperplane described by a normal vector and a scalar offset. Here, the focus is primarily on linear classifiers. However, when they are deployed on engineered feature spaces, they are effectively nonlinear classifiers on

the base feature space. Two nonlinear classifiers are also considered. But because linear classifiers are so important to the analysis that follows, I now briefly review the geometry of affine hyperplanes.

Affine hyperplane geometry

An affine hyperplane $H = (\boldsymbol{\beta}, \beta_0)$ is the set of vectors $\mathbf{x}_j \in \mathbb{R}^d$ satisfying $\boldsymbol{\beta} \cdot \mathbf{x}_j + \beta_0 = 0$. The geometric distance from any point $\mathbf{x} \in \mathbb{R}^d$ to any point $\mathbf{x}_0 \in H$ can be computed by noting that $\boldsymbol{\beta}$ is normal to H, and that $\forall\ \mathbf{x}_0 \in H,\ \boldsymbol{\beta} \cdot \mathbf{x}_0 = -\beta_0$. Therefore,

$$d(\mathbf{x}, \mathbf{x}_0) = |\boldsymbol{\beta} \cdot \mathbf{x} + \beta_0| / ||\boldsymbol{\beta}||. \tag{5.9}$$

Since the relation above is independent of $\mathbf{x}_0$, we may refer to the "distance from $\mathbf{x}$ to H" without ambiguity, denoted as $d(\mathbf{x}, H)$.

The classification rule c learned by a linear classification algorithm is often formulated in terms of a separating affine hyperplane $H = (\boldsymbol{\beta}, \beta_0)$:

$$c(\mathbf{f}) = \text{sign}\left[\boldsymbol{\beta} \cdot \mathbf{f} + \beta_0\right]. \tag{5.10}$$

Suppose $H = (\boldsymbol{\beta}, \beta_0)$ separates $\mathcal{C}$, so that $c(\mathbf{f}_j) = y_j\ \forall\ j$. The *geometric margin* M_H of H is the minimum distance from any feature vector to H:

$$M_H = \min_{\mathbf{f}_j \in \mathcal{C}} d(\mathbf{f}_j, H) = \frac{1}{||\boldsymbol{\beta}||} \min_{\mathbf{f}_j \in \mathcal{C}} \left|\boldsymbol{\beta} \cdot \mathbf{f}_j + \beta_0\right|. \tag{5.11}$$

Figure 5.7 gives a pictorial description of the margin. Suppose H is a fixed separating hyperplane for $\mathcal{C}$. Its margin M_H has a nice geometric interpretation: the classifier would only misclassify a feature vector in the training set if were perturbed by at least M_H along $\hat{\boldsymbol{\beta}}$. So a large margin is desirable. When there are multiple hyperplanes that separate the training data, the *optimal* hyperplane is the one that maximizes M_H. If H has the largest geometric margin of all hyperplanes that could separate $\mathcal{C}$, then it follows that H would generalize well to the task of classifying feature vectors

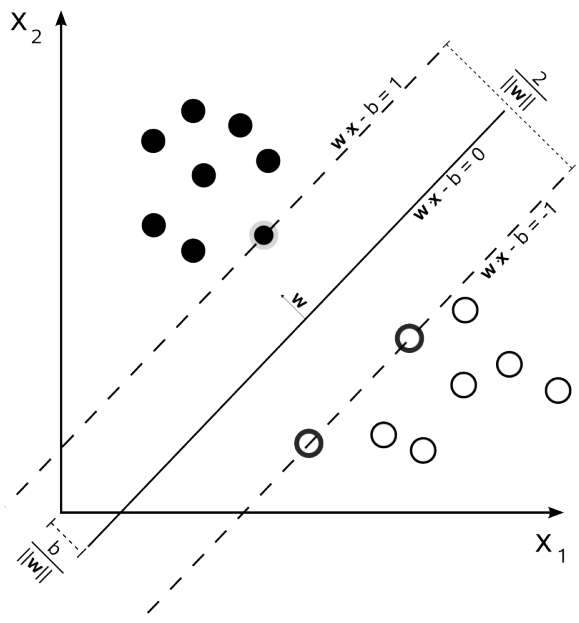

Figure 5.7: **Example of margin for a separating hyperplane**. In this toy example, a hyperplane $H = (\mathbf{w}, b)$ separates the data (solid black line). The dashed line intersects the feature vectors closest to the separating hyperplane. The distance from the separating hyperplane to the nearest feature vectors is called the *margin*; here, the margin is $1/||\mathbf{w}||$. Credit: Wikimedia commons.

that are merely small perturbations on the feature vectors in $\mathcal{C}$. The margin becomes especially important when the training data are limited, or when the QCVV data to be classified has finite sample noise (see Section 5.4.5). *Support vector machines* (Appendix D.3) are particularly good at finding robust, large-margin hyperplanes.

Algorithms for supervised binary classification

This chapter examines and compares the performance of several algorithms for supervised binary classification. One class that is *not* considered is neural networks. Neural networks have a complex internal structure that makes them very powerful, but makes their behavior difficult to understand and explain. (Recall the "interpretability issue" raised in Section 5.1.2.) Instead, five simpler, widely-used algorithms are

considered. Three are linear classifiers: linear discriminant analysis (LDA), perceptrons, and linear support vector machines (SVMs). Two are intrinsically nonlinear: quadratic discriminant analysis (QDA), and radial basis function (RBF) SVMs. Each of these algorithms is explained below. Further discussion can be found in Hastie *et al.* (140), and in Appendix D.

The behavior of most ML algorithms can be configured by setting user-specified *hyperparameters* (so named to distinguish them from the parameters that index the hypothesis, which the algorithm is trying to learn). Tuning an algorithm's hyperparameters so that it can learn from known data *and* generalize to perform well on unseen, future data is something of a subtle art. In the examples considered here, the relevant hyperparameters are typically *swept* across a wide range to find values that work. (See Appendix F for details, particularly Table F.1, which lists the hyperparameter values examined.)

The first two algorithms examined, *LDA* and *QDA*, approach the binary classification problem from a statistical perspective. They are derived from a Gaussian ansatz, in which the feature vectors for each class are normally-distributed with means $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ and covariances Σ_1 and Σ_2. Under this assumption, there is an optimal decision boundary for the feature vector $\mathbf{f}$ that can be derived from a likelihood ratio test. The "learning" in LDA and QDA corresponds to assuming that the means and covariances are unknown, and must be estimated from the training data. Denote the estimated means and covariances (respectively) by $\hat{\boldsymbol{\mu}}_j$ and $\hat{\Sigma}_j$.

LDA assumes that the two covariance matrices are identical, in which case the optimal decision boundary is a hyperplane. The LDA decision rule is

$$\begin{aligned} c_{\text{LDA}}(\mathbf{f}) = \text{sign}\Big[&\mathbf{f}\cdot\hat{\Sigma}^{-1}(\hat{\boldsymbol{\mu}}_1-\hat{\boldsymbol{\mu}}_2) \\ &+(\hat{\boldsymbol{\mu}}_1\cdot\hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_1-\hat{\boldsymbol{\mu}}_2\cdot\hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_2)/2\Big], \end{aligned} \tag{5.12}$$

which is of the form $\text{sign}[\mathbf{f}\cdot\boldsymbol{\beta}+\beta_0]$. QDA allows the two covariance matrices to be

different, which produces a nonlinear optimal decision rule that takes the form

$$c_{\text{QDA}}(\mathbf{f}) = \text{sign}\left[\frac{1}{2}\log\left(\frac{|\hat{\Sigma}_2|}{|\hat{\Sigma}_1|}\right) + \frac{1}{2}(\mathbf{f}-\hat{\boldsymbol{\mu}}_2)^T\hat{\Sigma}_2^{-1}(\mathbf{f}-\hat{\boldsymbol{\mu}}_2) - \frac{1}{2}(\mathbf{f}-\hat{\boldsymbol{\mu}}_1)^T\hat{\Sigma}_1^{-1}(\mathbf{f}-\hat{\boldsymbol{\mu}}_1)\right]. \tag{5.13}$$

Derivations of these classification rules can be found in Appendix D.1.

Estimating the means $\hat{\boldsymbol{\mu}}_1$ and $\hat{\boldsymbol{\mu}}_2$ is relatively straightforward as long as there are significantly more training examples than dimensions in feature space. But unbiased estimation of the covariance matrices would require much larger training sets (more than d^2 examples), so both LDA and QDA *regularize* the estimated covariance matrices. LDA does so through a hyperparameter τ that affects the dimension of β (by controlling the *rank* of Σ^{-1}). QDA introduces a hyperparameter s, where $0 \leq s \leq 1$, and replaces $\hat{\Sigma}_k$ by $s\hat{\Sigma}_k + (1-s)\mathcal{I}$.

The *perceptron* (259; 258), one of the oldest supervised binary classification algorithms, makes no assumptions about the distribution of the feature vectors. It learns a separating hyperplane using a simple, iterative training algorithm whose only hyperparameter, N_{epochs}, determines the number of iterations allowed before the algorithm terminates. It generates a linear decision rule of the form

$$c_{\text{Perceptron}}(\mathbf{f}) = \text{sign}\left(\boldsymbol{\beta}\cdot\mathbf{f} + \beta_0\right). \tag{5.14}$$

Appendix D.2 provides details on how the perceptron is trained.

Support vector machines (SVMs) (308) address a notable flaw of the perceptron: the perceptron will find *some* separating hyperplane (if one exists), but not necessarily an *optimal* one. The *soft-margin* SVM defines the optimal hyperplane to be the one that maximizes the geometric margin, subject to a regularization penalty C. (See Appendix D.3 for details.) The decision rule learned by the linear SVM is

$$c_{\text{Linear SVM}}(\mathbf{f}) = \text{sign}\left[\sum_{j=1}^{N} y_j c_j(\mathbf{f}_j\cdot\mathbf{f}) + \beta_0\right], \tag{5.15}$$

where $0 \leq c_j \leq C$. C is a hyperparameter for this algorithm, and controls a trade-off between maximizing M_H and minimizing the number of mis-classified points. (See Equation (D.30).) The decision rule explicitly depends on the feature vectors in $\mathcal{C}$. If $c_j \neq 0$, $\mathbf{f}_j$ is called a *support vector*.

The *radial basis function (RBF) SVM* is intended for situations where the data can only be separated by curved (nonlinear) hypersurfaces. The RBF SVM uses a kernel K (147; 134) to implicitly map the data to a high-dimensional feature space where a linear decision boundary (which is nonlinear in the original feature space) can separate the data. So the RBF SVM is implicitly performing a particular kind of feature engineering, akin to what is explicitly done later in this chapter. The RBF SVM decision rule is

$$c_{\text{RBF SVM}}(\mathbf{f}) = \text{sign}\left[\sum_{j=1}^{N} y_j c_j K(\mathbf{f}_j, f) + \beta_0\right], \tag{5.16}$$

where $K(\mathbf{x}, \mathbf{y}) = \text{Exp}\left[-\gamma||\mathbf{x} - \mathbf{y}||^2\right]$, and again $0 \leq c_j \leq C$. The RBF SVM has two hyperparameters: C, which behaves exactly as with the linear SVM, while γ controls the *width* of the kernel.

5.3.5 Data collection $\mathcal{C}$

The main goals of this chapter are to determine whether any ML algorithm *could* learn a technique to distinguish stochastic and coherent noise, and to compare the performance of different algorithms. To test them, a large collection of labeled data – simulated GST datasets for many realizations of stochastic and coherent noise – was generated, and was used to train each algorithm.

To produce this data, I numerically simulated GST experiments – for several different values of L – using noisy gate sets. In all cases, the noiseless "target" gate set was $\mathcal{G} = \{\rho_0 = |0\rangle\langle 0|, \{G_j\} = \{\mathcal{I}, X_{\pi/2}, Y_{\pi/2}\}, E = \{|0\rangle\langle 0|, |1\rangle\langle 1|\}\}$. I chose 19 values for the noise strength η (see Table 5.1). For each noise type (stochastic / coherent)

Target gate set $\mathcal{G}$	$\rho_0 = \|0\rangle\langle 0\|$, $\{G_j\} = \{G_I, G_X, G_Y\}$, $E = \{\|0\rangle\langle 0\|, \|1\rangle\langle 1\|\}$
Noise type	Coherent and stochastic (see Appendix C.1)
GST data set family parameter L	$[1, 2, 4, 8, 16, 32, 64, 128, 256]$
Noise strength η	$[10^{-4}, 2.15 \times 10^{-4}, 4.64 \times 10^{-4}, 10^{-3}, 2.15 \times 10^{-3}, 4.64 \times 10^{-3}, 10^{-2}, 2.15 \times 10^{-2}, 4.64 \times 10^{-2}, 10^{-1}, 1.19 \times 10^{-1}, 1.43 \times 10^{-1}, 1.71 \times 10^{-1}, 2.04 \times 10^{-1}, 2.44 \times 10^{-1}, 2.92 \times 10^{-1}, 3.49 \times 10^{-1}, 4.18 \times 10^{-1}, 5 \times 10^{-1}]$
Noise realizations for fixed η and noise type	900 (1 for each gate in $\mathcal{G}$, yielding 300 noisy gate sets, generated as specified in Appendix C.2)

Table 5.1: **Data set description.** For each value of L, a collection of data $\mathcal{C}_L$ was generated using noisy versions of the gate set $\mathcal{G}$. The noise strength η quantifies the discrepancy between the ideal gate set and its noisy version; as $\eta \to 0$, the noisy gate set converges to the ideal one.

and noise strength η, I generated 300 random noisy gatesets, each one obtained by postpending a randomly chosen noise channel of that noise type to each gate. (Note that an independently selected noise channel was applied to each of the three gates in the gate set). pyGSTi (222) was used to generate the GST data sets that would result from running the GST experiment design with a given noisy gate set. Although I do consider finite-sample effects (see Section 5.4.5), the main focus is on the $N \to \infty$ exact-sampling limit, where the estimated frequencies equal the exact outcome probabilities.

The collection of labeled feature vectors generated this way, for a specific value of L, is denoted $\mathcal{C}_L$. So $\mathcal{C}_L = \{(\mathbf{f}_j, y_j)\}_{j=1}^N$, where $\mathbf{f}_j = \phi(\mathcal{D}_j) \in \mathbb{R}^d$, d depends on L, and y_j is the binary label indicating "stochastic" or "coherent". For each value of L, there are $N = 2 \times 19 \times 300 = 11400$ labeled feature vectors that can be used for training or testing.

Because most ML algorithms are designed to perform best on data with zero mean

and unit variance, the feature vectors in $\mathcal{C}_L$ are *standardized*. Standardization expresses the feature vector in coordinates that are (conceptually) the same across components. A common standardization approach (and the one used here) is to replace the components of each feature vector by their Z-score. Let $\hat{\boldsymbol{\mu}}$ denote the estimated mean of the feature vectors, and $\hat{\Sigma}$ denote the estimated covariance matrix. For each $\mathbf{f}_j$, the standardized $\mathbf{z}_j$ is

$$\mathbf{z}_j = \mathrm{diag}(\hat{\Sigma})^{-1}(\mathbf{f}_j - \hat{\boldsymbol{\mu}}), \tag{5.17}$$

so that $\langle \mathbf{z}_j \rangle = 0$ and $\mathrm{Cov}(\mathbf{z}_j, \mathbf{z}_k) = \mathcal{I}_{d^2}$. Standardization adjusts each component of $\mathbf{f}_j$ independently, so no correlation is introduced between the features.

5.4 Results

5.4.1 Classifying GST feature vectors

Testing whether linear classification is feasible

Three of the classification algorithms presented in Section 5.3.4 learn a linear decision boundary (i.e., an affine hyperplane H). Of course, this is impossible if $\mathcal{C}_L$ is not linearly separable! So in each case, I began by determining whether the training data were linearly separable. This is usually checked by running the perceptron algorithm with N_{epochs} set to some very large number. If the algorithm converges to a separating hyperplane, then clearly $\mathcal{C}_L$ is linearly separable. However, a failure to converge doesn't guarantee that $\mathcal{C}_L$ is linearly *inseparable*. Instead, a linear program is used to test for separability that either finds a separating hyperplane *or* (if the data are inseparable) constructs a provable witness to non-separability (see Appendix E). Using this technique, I found each of the $\mathcal{C}_L$ to be linearly separable *except* for $L = 1$ ("linear GST" data). So no linear classification algorithm can attain 100% accuracy in classifying coherent and stochastic noise using $L = 1$ GST feature vectors, even in

Value(s) of η	Separability witness?
Fixed	Yes
$[10^{-4}, .5]$	**No**
$[10^{-4}, .34]$	Yes
$[10^{-2}, 10^{-1}]$	Yes
$[10^{-4}, 10^{-2}]$	Yes
$[10^{-4}, 10^{-3}]$	Yes

Table 5.2: **Testing for linear separability of subsets of $\mathcal{C}_1$.** Linear separability for *some* subsets of $\mathcal{C}_1$ is established by checking a separability witness (see Appendix E). A subset with noise strength η spanning a slightly restricted range (relative to the original range considered) is separable, as are subsets with a *fixed* value for η.

principle. This result indicates that there is something nontrivial about the *geometry* of the data (see Section 5.4.2).

However, this only holds true when the range of values for the noise strength η is quite large, extending over 3 orders of magnitude from 10^{-4} to .5. Restricting the range of η produced smaller subsets of $\mathcal{C}_1$, which (see Table 5.2) were almost always linearly separable. A simple conclusion from this result is that this QCVV problem gets harder (at least somewhat) when the gate errors are allowed to be very large. This is unsurprising; similar challenges afflict randomized benchmarking and GST, and most QCVV methods focus on the regime where gate errors are perturbative.

For $L > 1$, the training data were always linearly separable. However, I considered the possibility that this might have been an artifact of finite training data (undersampling), rather than an indication that the two noise classes can always be separated using linear classifiers. To check for this, a linear soft-margin SVM ($C = 10^4$) was trained using the original 300 realizations of each noise type, and then evaluated the accuracy of the hyperplane it learned on 20,000 previously unseen realizations. On each of $L = 4$ and $L = 8$, the hyperplane learned had $\sim 96\%$ accuracy on the new noise realizations. This suggests that $\mathcal{C}_4, \mathcal{C}_8$ are in fact linearly separable, and that I was not undersampling the noise realizations.

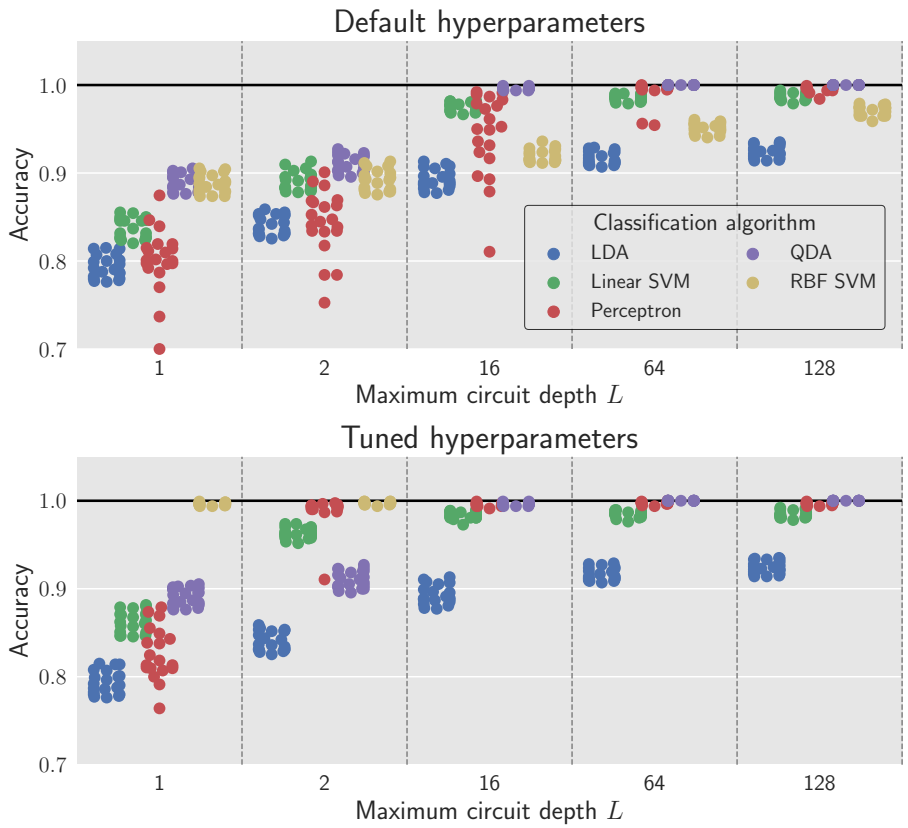

Figure 5.8: **Swarmplot of $K = 20$ cross-validated classification accuracies as a function of L.** As L increases, the accuracy increases in a classifier-dependent way. **Top**: Under the default value of its hyperparameters, the QDA algorithm typically performs best. **Bottom**: Hyperparameter tuning boosts the accuracy of the linear SVM, perceptron, and RBF SVM algorithms.

Classification accuracy depends on L

If $\mathcal{C}_L$ is linearly separable, then in principle a linear classification algorithm could successfully learn a separating hyperplane. As noted in Section 5.2, the *hyperparameters* of an algorithm influence the inference tool it learns, and consequently, the accuracy of its inferences.

Evaluating the accuracy of an inference tool is straightforward, especially in supervised learning: ask the tool to classify a feature vector, and compare the label it

assigns to the true label. To evaluate the accuracy of an *algorithm*, however, is a slightly harder task. A common way to do so is to use *cross-validation*. The data collection $\mathcal{C}$ is split into two parts, $\mathcal{C}_{\text{train}}$ and $\mathcal{C}_{\text{test}}$; the algorithm is trained using data in $\mathcal{C}_{\text{train}}$, and the accuracy of the tool it learns is computed on $\mathcal{C}_{\text{test}}$. In this work, I typically use a "shuffle-split" cross-validation approach, where $\mathcal{C}$ is split K times into training and testing data; I usually take the number of feature vectors in $\mathcal{C}_{\text{test}}$ to be 10% of the feature vectors in $\mathcal{C}$.

Figure 5.8 plots the cross-validated accuracies of the ML algorithms as a function of the GST family parameter L. As the top portion of the figure shows, the accuracy of the inference tools increases with L. However, under the *default* value of the hyperparameters for the linear classification algorithms, the accuracy is not close to 1 for $L = 2$ or $L = 16$. In turn, this implies that hyperparameter tuning is necessary to boost accuracy.

The bottom portion of Figure 5.8 shows the accuracies under the best hyperparameter values. For both the linear SVM and perceptron algorithms, hyperparameter tuning improved accuracy from $\sim .9$ to $\geq .95$. In contrast, the performance of the LDA and QDA algorithms is best on the default value of their hyperparameters. (See Figure F.1 in Appendix F for plots of the accuracy as a function of classification algorithm hyperparameters.)

The performance of the LDA algorithm is noteworthy because it highlights an important fact – *QIP properties are not generally learnable by* ***arbitrary*** *ML algorithms.* That is, even though the data for $L > 2$ *is* linearly separable, and LDA is a *linear* classification algorithm, it does not follow that the LDA algorithm will *always* succeed in learning a separating hyperplane, even with hyperparmeter tuning.

Unsurprisingly, the *linear* classification algorithms continue to perform poorly on $\mathcal{C}_1$, even with hyperparameter tuning. Given that $\mathcal{C}_1$ *isn't* linearly separable, this result is to be expected.

These results highlight the importance of checking the hyperparameters of a given algorithm. Depending on the data and the algorithm, hyperparameter tuning could help. However, hyperparameter tuning takes time, and there is the risk that even with cross-validation, the hyperparameter settings that yield maximal accuracy on data we currently have may cause the algorithm to learn a classifier that performs poorly on future data.

For this reason, I also investigated the use of *feature engineering* – creating new features out of existing ones – to boost accuracy. Of the data collections considered, there is one where it is most glaringly apparent that increases in accuracy are possible; namely, $\mathcal{C}_1$. It is not linearly separable, and the reason is fairly straightforward to understand. In the next subsection, dimensionality reduction techniques are used to probe the structure of $\mathcal{C}_1$. The insights about this structure will inform the design of new feature maps in Section 5.4.3 that take $L = 1$ GST data sets and map them into a feature space where they *are* linearly separable.

5.4.2 Probing the structure of $\mathcal{C}_1$ by dimensionality reduction

Dimensionality reduction techniques embed data from a $d \gg 1$-dimensional feature space into a $k \ll d$-dimensional space to visualize its structure and related properties. To explore the structure of $\mathcal{C}_1$, two techniques were used: principal component analysis and multidimensional scaling.

Principal component analysis (PCA) (159; 148; 231) is a technique that projects N feature vectors onto the directions along which they vary maximally vary These directions – the "principal components" – are the eigenvectors $\{\mathbf{e}_j\}_{j=1}^{E}$ of the estimated covariance matrix $\hat{\Sigma}$:

$$\hat{\Sigma} = \sum_{j=1}^{E} \sigma_j \mathbf{e}_j \mathbf{e}_j^T, \tag{5.18}$$

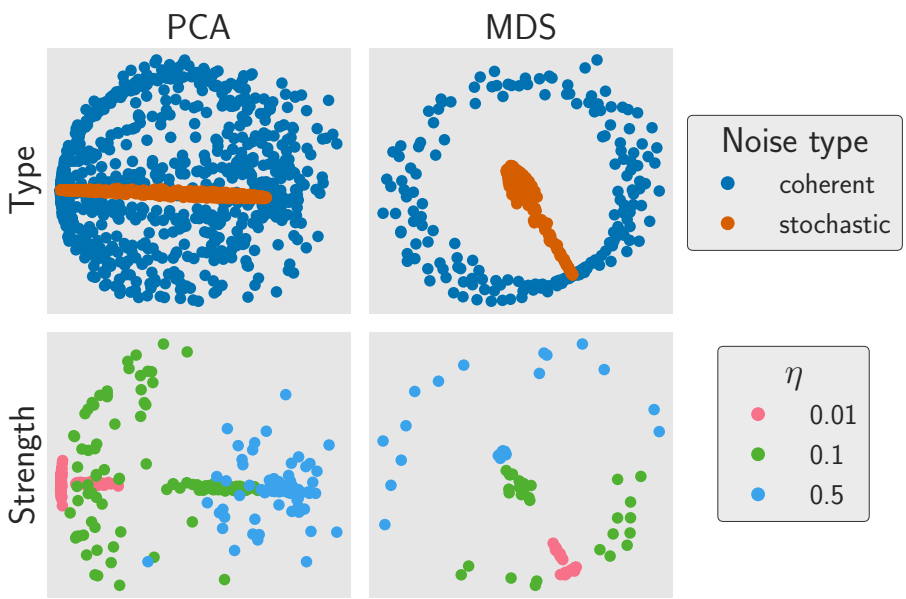

Figure 5.9: **2-dimensional embeddings of $\mathcal{C}_1$. Top**: The embedded feature vectors for stochastic noise appear to be "surrounded" by those for coherent noise. **Bottom**: As the noise strength $\eta \to 0$, the feature vectors for coherent and stochastic noise approach one another, which is to be expected: at $\eta = 0$, the underlying "noisy" gate set is actually the ideal one.

where the number of eigenvectors $E \leq \min(d, N)$ and σ_j^2 is the variance of the data along $\mathbf{e}_j$. The principal components can be used to define a projector Π_k from $\mathbb{R}^d$ to $\mathbb{R}^k$:

$$\mathbf{f}_j \to \mathbf{y}_j \equiv \Pi_k[\mathbf{f}_j] = \sum_{j=1}^{k} \mathbf{e}_j(\mathbf{e}_j \cdot \mathbf{f}_j). \tag{5.19}$$

When using PCA for dimensionality reduction (i.e., defining Π_k) k is usually taken to be less than E. Instead, only the principal components that have *large* eigenvalues are kept, since if $\sigma_j \sim 0$, the principal component $\mathbf{e}_j$ is an "uninformative" direction – the data does not vary much along it – and there is no reason to include it in the projector.

Multdimensional scaling (MDS) (41; 296; 180; 274; 275) provides a different approach for dimensionality reduction by defining the k-dimensional representation of the data

as a solution to an optimization problem in the k-dimensional space that preserves as much as possible all pairwise distances.

Given a data set with pairwise distances (or dissimilarities) between the feature vectors $\{d_{mn}\}_{m,n=1}^{N}$, the MDS embedding is a set $\{\mathbf{y}_j\} \in \mathbb{R}^k$ satisfying

$$\{\mathbf{y}_j\}_{j=1}^{N} = \underset{\mathbb{R}^k}{\operatorname{argmin}} \sum_{m,n=1}^{N} \left(||\mathbf{y}_m - \mathbf{y}_n|| - d_{mn}\right)^2 . \tag{5.20}$$

Because the feature space for the $L = 1$ feature vectors is a subset of $\mathbb{R}^{92}$, a natural distance measure between $\mathbf{f}_m$ and $\mathbf{f}_m$ is their Euclidean distance: $d_{mn} = ||\mathbf{f}_m - \mathbf{f}_n||_2$.

Figure 5.9 plots the $k = 2$-dimensional embeddings of $\mathcal{C}_1$ using PCA or MDS. The top row colors the embedded points are colored by noise *type*, and the bottom row colors them by noise *strength*. Both plots indicate that $\mathcal{C}_1$ bears some resemblance to a high-dimensional radio dish.

These embeddings are low-dimensional approximations to high-dimensional feature spaces, leading to the question "Are these 'radio dishes' real?" A simple argument suggests the answer is "mostly yes". The circuits used for $L = 1$ GST give rise to a feature vector that depends on the gate set in an almost linear fashion (in process tomography, the feature vector would be exactly linear in the process). Because the deviations from linearity are small, the structure of $\mathcal{C}_1$ is similar to the structure of the underlying gate sets generating the feature vectors.

This structure can be understood by mapping the gate set to a quantum state using the Choi-Jamiołkowski isometry (157). A gate set is the direct sum of the constituent gates (a linear operation), and the Choi-Jamiołkowski isometry is a linear map from gates (channels) to quantum states. Hence, a linear map exists taking gate sets into quantum state space. A gate affected by purely coherent noise maps to a pure state, and a gate affected by purely stochastic noise maps to a mixed state. Therefore, gate sets affected by purely coherent noise "envelop" those affected by purely stochastic

noise. This behavior is analogous to that observed with $\mathcal{C}_1$, although the exact structure may not be comparable. (Again, there are small deviations from a linear relationship between feature vectors and gate sets for $L = 1$ GST.) Therefore, the radio dish structure present in $\mathcal{C}_1$ appears to be genuinely real, and not an artifact of our simulations.

This explains why algorithms using a *nonlinear decision rule* (QDA or RBF SVM) achieve higher accuracy than linear algorithms (recall Figure 5.8): the surfaces that most naturally separate $\mathcal{C}_1$ have curvature, and the nonlinear algorithms can successfully learn that curvature.

Given this domain-specific knowledge, we can use *feature engineering* to change the feature map that takes GST data sets and maps them to feature vectors. In particular, the radio dish structure in $\mathcal{C}_1$ can be "unrolled" in such a way that *linear* classification algorithms can achieve high accuracy. Two new feature maps are discussed in the next subsection, and Section 5.4.4 shows that the perceptron and linear SVM algorithms can achieve high accuracy in these new feature spaces.

5.4.3 Overcoming linear inseparability of $\mathcal{C}_1$ using feature engineering

A linear classification algorithm can learn a quadratic function of the features if the algorithm has access to quadratic functions of the features. There are two feature engineering maps that naturally "unroll" the quadratic radio dish structure of $\mathcal{C}_1$. Both add new components to the feature vectors on top of the "base" features defined by ϕ, thereby enlarging the feature space. Let $\mathbf{f}$ be a feature vector in $\mathcal{C}_1$. The two feature engineering maps considered are

$$\begin{aligned} \phi_{\text{SQ}} &: \mathcal{F} \to [0,1]^{2d} \\ \phi_{\text{SQ}}(\mathbf{f}) &= \mathbf{f} \underset{j}{\oplus} f_j^2, \end{aligned} \tag{5.21}$$

and

$$\begin{aligned} \phi_{\mathrm{PP}} &: \mathcal{F} \to [0,1]^{d(d+3)/2} \\ \phi_{\mathrm{PP}}(\mathbf{f}) &= \mathbf{f} \underset{j<k}{\oplus} f_j f_k. \end{aligned} \tag{5.22}$$

Note that for ϕ_{PP}, *unique* pairwise products are used, so that $f_j f_k$ and $f_k f_j$ are not both added as extra components. As an example of how these maps act, consider $\mathbf{f} = (x, y) \in \mathbb{R}^2$. Then $\phi_{\mathrm{SQ}}(\mathbf{f}) = (x, y, x^2, y^2) \in \mathbb{R}^4$, and $\phi_{\mathrm{PP}}(\mathbf{f}) = (x, y, xy, x^2, y^2) \in \mathbb{R}^5$.

ϕ_{SQ} adds a quadratic nonlinearity while preserving the coordinate axes, whereas ϕ_{PP} allows quadratic nonlinearity together with rotation of the coordinate axes. If the coordinate axes are significant for this problem, ϕ_{SQ} is preferable, while ϕ_{PP} is preferable if correlations between different variables are important. Also, note that ϕ_{PP} has quadratically more parameters than ϕ_{SQ} and hence, a greater danger of overfitting. From an ML perspective, ϕ_{SQ} is the simplest way to enable linear classification algorithms to separate coherent and stochastic noise using $L = 1$ GST feature vectors, but the pipeline it defines is not as rich or complex as the pipeline defined by ϕ_{PP}.

5.4.4 Feature engineering of $\mathcal{C}_1$ enables linear separability

Checking the separability witness (Appendix E) confirms that ***under the action of*** ϕ_{SQ} ***and*** ϕ_{PP}***, the previously-inseparable*** $L = 1$ ***GST feature vectors become linearly separable in the new feature spaces***.

Because the feature vectors are linearly separable in these new feature spaces, linear classification algorithms should perform better than they did on $\mathcal{C}_1$. Figure 5.10 shows that feature engineering does boost the performance of the linear classification algorithms, and that this boost can be further enhanced using hyperparameter tuning. (See Figure F.2 in Appendix F for results of the hyperparameter sweep.)

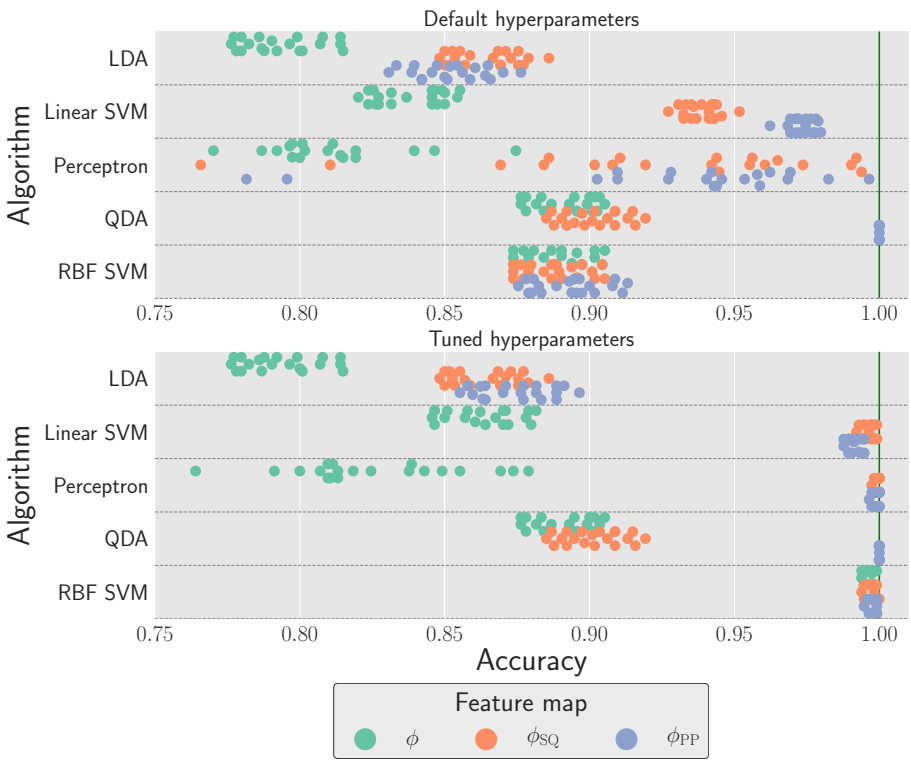

Figure 5.10: **Feature engineering boosts accuracy of some classification algorithms**. Cross-validated accuracies of the algorithms under their default hyperparameters (top) and with tuned hyperparameters (bottom). The performance of the RBF SVM algorithm indicates *nonlinear* separating surfaces are best. The QDA algorithm's performance on ϕ_{PP} feature vectors suggests a "quartic surface" – a 4^{th} order polynomial in the feature vector coefficients – is a sufficient amount of nonlinearity. The performance of the linear SVM and perceptron algorithms implies quadratic separating surfaces work well as approximations to the nonlinear surfaces learned by the RBF SVM and/or QDA algorithms.

The performance of the RBF SVM algorithm indicates *nonlinear* separating surfaces are generally best. However, the amount of nonlinearity required is modest: the performance of the QDA algorithm on ϕ_{PP} feature vectors suggests a "quartic surface" – a 4^{th} order polynomial in the feature vector coefficients – is a sufficient amount of nonlinearity. QDA learns a quadratic decision rule, and ϕ_{PP} uses all pairwise products. Therefore, the decision rule is *quartic* in the feature vector components.

The fact that linear SVM and perceptron algorithms also work well implies that quadratic separating surfaces are sufficiently good approximations to the nonlinear surfaces learned by the RBF SVM and/or QDA algorithms. Therefore, while the curvature of surface that would best separate the QCVV data sets mapped under ϕ_{SQ} or ϕ_{PP} deviates fairly strongly from a flat curvature, those deviations can be captured by quadratic surfaces.

As noted in Section 5.4.1, there is always the risk that the noise realizations were under-sampled. In this context, undersampling would mean that the geometry of the infinite-sample data set would not be faithfully represented by the geometry of the finite-sample data set. To check for this, I trained each classification algorithm on all the engineered feature vectors for each feature map and then cross-validated how the separating surface learned by the algorithm would perform on $\sim 20,000$ *previously unseen* feature vectors. The accuracies remain stable, indicating that the noise types are not being undersampled (see Table F.2 in Appendix F).

In sum, ***feature engineering separates $\mathcal{C}_1$, and linear classification algorithms such as perceptrons or linear SVMs can successfully learn a high-accuracy classifier whose performance is comparable to classifiers learned by nonlinear algorithms.***

5.4.5 Robustness to finite-sample effects

The tests and analysis done up to this point have been performed in the *exact-sampling* limit. However, real GST data sets have finite-sample fluctuations because $N_{\mathrm{samples}} \ll \infty$. This section examines the robustness of our conclusions regarding separability of the engineered feature vectors under finite-sample effects. Finite sampling means the outcome frequencies of the circuits will *flucuate* around the outcome probabilities by an amount that goes as $1/\sqrt{N_{\mathrm{samples}}}$. If the amount of fluctuation is modest, then classifiers trained on fluctuation-free data sets should still be able

to reliably classify a feature vector with finite-sample effects. The reason is simple: suppose a hyperplane H had been learned by training a classification algorithm on fluctuation-free data. H would also do well in classifying finite-sample data, provided the statistical fluctuations are less than the geometric margin M_H.

M_H is a property of both the hyperplane H and the data it separates (recall Equation (5.11)). As larger-margin hyperplanes are more robust under larger finite-sample effects, a hyperplane that *maximizes* the margin is preferable. Consequently, the linear SVM is the ideal choice for an ML algorithm to learn a separating hyperplane that would later be used to classify finite-sample data.

As the feature spaces generated by the feature engineering maps ϕ_{SQ} and ϕ_{PP} are linearly separable, and the linear SVM algorithm can learn a high-accuracy hyperplane, I investigated how the accuracy of the hyperplane learned by the algorithm changes when classifying feature vectors with finite-sample effects.

To generate finite-sample versions of the feature vectors with N_{samples} finite-sampling effects, I first add finite-sample fluctuations to the feature vectors in $\mathcal{C}_1$. Then I apply either ϕ_{SQ} or ϕ_{PP}. The resulting data collection is then mean-standardized (seeSsection 5.3.5) using the estimated mean of the fluctuation-free data. The reason for mean-standardizing the data is because the margin of a hyperplane is invariant under translating the data – which is what mean-standardization does – but not under variance-standardizing it. Due to the technical challenges of using a hard-margin SVM to learn a hyperplane in the engineered feature spaces, a soft-margin SVM with $C = 10^5$ was trained on the fluctuation-free data to learn a separating hyperplane. The accuracy of this hyperplane is then evaluated on the finite-sample feature vectors.

Results of this test are shown in Figure 5.11. (Cross-validation was done using 50 independent realizations of finite-sampling effects for each value of N_{samples}.) The vertical grey line shows the margin of the hyperplane learned by the SVM algorithm.

(It turns out that the geometric margin of the hyperplanes learned on feature vectors mapped using ϕ_{SQ} and ϕ_{PP} were comparable.) Once the statistical noise $1/\sqrt{N_{\mathrm{samples}}}$ is less than the margin of the hyperplane, classification accuracy increases to 1.

The two feature engineering maps considered here add terms of the form $\hat{p}_j^2$ (ϕ_{SQ}) or $\hat{p}_j\hat{p}_k$ (ϕ_{PP}) to the original feature vector. Simple algebra shows that the fluctuations in $\hat{p}_j^2$ and $\hat{p}_j\hat{p}_k$ both also go as $1/\sqrt{N_{\mathrm{samples}}}$. So adding these components doesn't make the engineered feature vector *more* sensitive to finite-sample effects.

This test confirms the intuition that maximum-margin hyperplanes are more robust for classifying noise in the presence of finite-sample effects. The margin is not generally invariant under different feature maps, although this happened to be the case here.

5.5 Conclusion and discussion

This work showed that supervised learning algorithms can successfully learn a QCVV technique for distinguishing between coherent and stochastic noise on a single qubit. The success of these algorithms depends strongly on the experiment design, the algorithm's own hyperparameters, and the feature map used to embed experimental data into a feature space. QCVV practitioners developing their own machine-learned QCVV technique will need to be mindful of how the quality of the technique learned by the ML algorithm is impacted by the choices they make for the components of the technique (recall Section 5.2).

As quantum computing enters the NISQ era, opportunities are opening up for demonstrating how NISQ processors can help solve near-term problems of interest. The central aim of QCVV is to improve performance of QIPs; for NISQ processors, improved performance generally means it is capable of running longer-depth circuits.

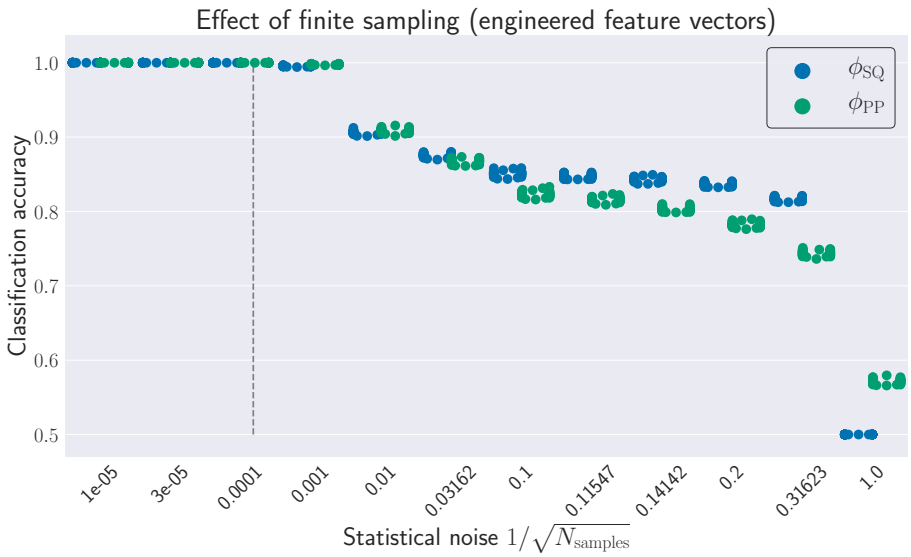

Figure 5.11: **Classification accuracy under finite-sampling effects using feature engineering**. A soft-margin linear SVM ($C = 10^5$) was trained using *noiseless* feature-engineered feature vectors, and then its accuracy evaluated noisy versions of that data. The geometric margin of the hyperplane learned by the SVM algorithm was comparable between the two feature engineering maps, and is indicated by the dashed vertical line. Once the statistical noise is less than the geometric margin, the accuracy goes to 1.

Improving the performance requires lowering the error rate(s)[2]. (If the error rate is ϵ, a circuit whose depth exceeds $\mathcal{O}(1/\epsilon)$ will most likely output a state with low fidelity to the correct answer.) Characterizing what's going wrong with a processor is a necessary first step in doing so. However, characterizing NISQ processors comes with its own set of unique challenges, so new QCVV techniques are needed.

Machine learning (ML) algorithms can help develop new QCVV techniques, as discussed in Section 5.1.2. ML algorithms don't model the underlying complexity of a QIP in any significant detail, and they don't rely on statistical theory. Instead, they operate from the premise that all a QCVV technique needs to do is approxi-

[2]For some metrics of computational utility – such as the *quantum volume* (31) – lowering the error rate past some effective threshold doesn't improve the metric.

mate the functional relationship between data and a QIP property. ML algorithms excel at learning approximations to functions, and so, can help automate the task of developing new QCVV techniques.

This doesn't mean QCVV practitioners are unnecessary. They are still needed to propose relevant QIP properties to be characterized, or to think about an appropriate experiment design. They also need to choose between algorithms, evaluate their performance, write code, interpret results, etc. ML algorithms *augment* the expertise of QCVV practitioners, not *replace* it.

This kind of domain-specific expertise helps. For instance, it can guide wise choices for the feature map, or suggest good feature engineering techniques. (As seen in Section 5.4.3, knowing that coherent and stochastic noise affects gate sets in particular ways helped us realize that ϕ_{SQ} or ϕ_{PP} might be the good feature engineering techniques.) Conversely, ML algorithms for dimensionality reduction help QCVV practitioners understand the geometric structure of the data they are working with, and could provide some physical insight about the underlying noise models.

The nature of the QIP characterization task determines the ML paradigm (supervised/unsupervised/transductive); within each, different algorithms will need to be evaluated for their ability to learn a good QCVV technique for the task at hand. Again, QCVV practitioners cannot expect ML to provide "on-demand" solutions to QIP characterization tasks. This work showed that those interested in using ML will need to become more conversant in the language, methodologies, and vagaries of ML algorithms!

Because there are a plethora of ML algorithms for a wide variety of tasks, QCVV practitioners will need to make informed and prudent judgements about which algorithms to deploy. Here, supervised learning was prudent because the characterization task involved inferring a particular property of the QIP, and because generating *synthetic* (artificial) example data was easy. Given the characterization task consid-

ered in this Chapter, supervised binary classification algorithms were an appropriate choice of ML algorithm.

As NISQ processors increase in size and sophistication, supervised learning using synthetic data will become more difficult, as these kinds of forward simulations will become harder once quantum supremacy has been demonstrated, or quantum advantage attained. At that point, well-calibrated devices can be used to generate example data by, e.g., deliberately injecting specific kinds of noise into the QIP when running various circuits.

Other ML paradigms, not examined here, include unsupervised and transductive learning. Unsupervised learning would excel in discovering structure in a large number of data sets generated by a QIP. An example of a QCVV task where unsupervised learning would be useful is one where algorithms could evaluate whether data from experiments generated today (e.g., calibrations) are consistent with the historical operation of the device, or whether they are wildly out-of-spec, a problem known as *outlier detection*.

The question of machine-learned experiment design was alluded to in Section 5.1.2. Experiment design is a non-trivial problem, and has spurred much research in statistics. For this reason, this chapter borrows the experiment design used for GST (Section 5.3.2), as it is well-studied and sufficient for characterizing coherent vs. stochastic noise. There are two ways that "ML-learned experiment design" could be pursued. The first is to use ML algorithms for *feature selection* by selecting, out of a candidate set of circuits, a smaller subset that is useful for characterizing a property. (This approach is explored in Chapter 6.) Another, potentially more powerful, approach is to use *reinforcement learning* (RL) (290) to construct QCVV experiment designs from scratch. (See the conclusions of the next chapter.)

As QIPs advance and the rate of their advancement increases, QCVV theorists are

faced with the challenge of developing increasingly-powerful characterization techniques on ever-shorter timeframes. Leveraging ML algorithms can help solve some of these problems.

Chapter 6

Machine-learned experiment design for QCVV

The design of experiments is, however, too large a subject, and of too great importance to the general body of scientific workers, for any incidental treatment to be adequate. - Ronald A. Fisher, 1935 (103)

Chapter 5 focused on how ML[1] *algorithms can help with* data processing*, by learning inference tools for a targeted characterization problem on a qubit. This chapter investigates how ML can improve the other component of QCVV techniques, the experiment design. I show that ML algorithms for "feature selection" can pare down the circuits used for gate set tomography (GST), and construct small subsets – smaller experiment designs – that remain sufficient for characterizing particular qubit properties. The size of the subset constructed by these algorithms depends strongly on the characterization task. But for each of the characterization tasks considered, ML algorithms identify a set of circuits that is smaller than the original set used for GST. This suggests that feature selection of GST circuits by these and other algorithms is*

[1] As in Chapter 5, "ML" is used for 'machine learning', and *not* 'maximum likelihood' as it was in Chapters 3 and 4.

a useful jumping off point for developing targeted QCVV experiment designs.

6.1 Introduction

Characterizing a QIP requires running experiments (circuits) on it and collecting data about the outcomes of those experiments. That data is processed to yield an inference about a particular property of interest. In this chapter, I'll use the phrase *experiment design* to denote a collection of circuits. An experiment design specifies the experiments to be performed on the QIP; as noted in Chapter 5, one of the questions that has to be answered in developing a QCVV technique is "What is the experiment design?". Different QCVV techniques require different experiment designs, because different techniques target different properties, and different properties are best captured by different circuits. In this chapter, I take up the question of using machine learning (ML) to come up with experiment designs. Unlike Chapter 5, I won't focus on how ML algorithms could be used to *analyze* experimental data.

The *complexity* of an experiment design – the number of circuits it contains[2] – depends on the property being characterized. Generally speaking, less complex experiment designs are preferable to more complex ones. Less complex experiment designs take less time to run in the lab, thereby saving time and other experimental resources. Further, more complex experiment designs may have a high amount of redundancy, the elimination of which would streamline data processing. These points are extremely salient for characterizing NISQ processors. As noted in Section 2.4, experimental resources will need to be managed, and developing new experiment designs for characterizing novel noise types will be necessary. For this reason, studying how machine learning (ML) algorithms can help with experiment design is

[2]As far as I am aware, a formal definition for the complexity of an experiment design in the context of QCVV has not been given. I use a fairly loose definition here that jives with intuition, and acknowledge there may be difficulties in translating that intuition into a formal definition.

a worthwhile inquiry.

This chapter focuses on experiment design for QCVV techniques that characterize targeted properties. The QCVV tasks considered are of the form "Distinguish noise type A from noise type B". These tasks are similar in spirit to the problem addressed in Chapter 5. All the noise types considered here are Markovian. This ensures that some subset of the circuits used for gate set tomography (GST) is sufficient for successfully performing the task. GST can fully reconstruct the process matrices in the presence of Markovian noise, so there has to exist a subset of the GST experiment design that's capable of distinguishing any two *fixed* Markovian noise models.

The trick, of course, is figuring out which circuits are sufficient. Collectively, the circuits in a GST experiment design are sensitive to *arbitrary* Markovian errors, but this does not mean that *each* circuit is sensitive to *every* Markovian error. Each GST circuit amplifies a few linear combinations of the parameters of the gate set. A given Markovian noise may not affect some linear combinations, but will affect others. If the outcome probability of a circuit in the experiment design is *not* affected by noise type A *and* noise type B, it follows that the the outcome probability of that circuit provides no information on distinguishing the two noise types. Clearly then, that circuit could be removed from the experiment design.

For instance, robust phase estimation (RPE) can be used to efficiently estimate the angles of rotation of the gates in a single-qubit gate set (172) using a subset of the circuits necessary for single-qubit GST. Phrased another way, for the particular QCVV task that RPE performs, there are circuits necessary for GST that can be removed from the experiment design; what's more, removing those circuits *does not lessen the quality of the inferences derived from RPE*. Quantifying the relationship between the complexity of the experiment design and the nature of the QCVV task and assumptions about the QIP's behavior is an useful problem to tackle, but I do not consider it here.

Paring down the experiment design for GST requires selecting a sufficient number of circuits out of the experiment design that are useful for solving a given task. Physical intuition could help guide a search over the circuits, but could rapidly become bogged down, since the same noise affects different circuits in different and possibly hard-to-understand ways, and an exhaustive search over the circuits in a GST experiment design can rapidly become prohibitive (see Section 6.3.1). For this reason, ML algorithms offer a promising alternative.

In Chapter 5, I introduced the idea of a *feature space* into which the outcome probabilities of GST experiments are embedded for processing by ML algorithms. That feature space is a subset of $\mathbb{R}^d$, where an identification is made between each canonical unit vector of $\mathbb{R}^d$ and one particular circuit in the experiment design. The ML task corresponding to the task "select relevant GST circuits" is called *feature selection*, and the ML community has developed some algorithms for doing so (62). Feature selection is something of a "dark art" in ML, and only fairly recently (c. 2015) is end-to-end automation of feature selection becoming possible (165; 13; 295). For this reason, the feature selection approaches used in this chapter will require a great deal of "manual hand-holding".

The remainder of this chapter shows that ML algorithms can successfully identify a reduced set of GST circuits that are useful for characterizing coherent and stochastic noise, and also demonstrates that ML algorithms can also do this "circuit reduction" for a wide variety of QCVV tasks of a similar spirit (Section 6.6). Other ML algorithms for experiment design certainly could be investigated; in the conclusions (Section 6.7) I'll discuss how reinforcement learning could provide a powerful approach to experiment design for complex Markovian noise, or possibly even non-Markovian noise.

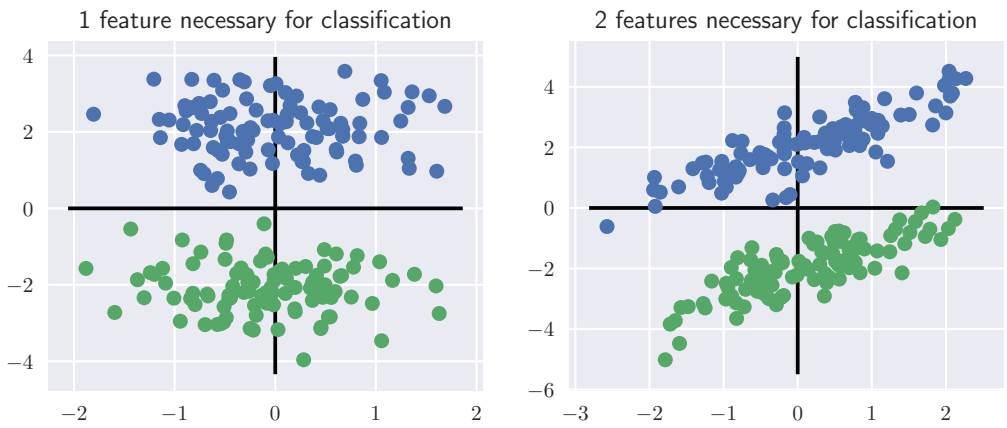

Figure 6.1: **The task of feature selection is to determine which features are essential**. Given a feature vector $\mathbf{f} = (x, y)$, task is to determine which cloud it belongs to. In the left panel, the x-component of $\mathbf{f}$ has little relationship to which cloud a feature vector belongs to, so the feature "the x-component of $\mathbf{f}$" can be removed from the feature space without lessening our ability to tell which cloud $\mathbf{f}$ belongs to. In the right panel, both features are necessary. Feature selection algorithms process collections of data and determine which features are most necessary (i.e., which features are informative). Note that the goal here is to select *subsets* of the features, not *subspaces* of the feature space.

6.2 Experiment design by feature selection

Consider a set of features $\{f_j\}$. The task of feature selection is to identify a good subset of $\{f_j\}$ that is sufficient to accomplish some task. (For example, see Figure 6.1.) Just as there are different *paradigms* for developing machine-learned QCVV tasks (supervised/unsupervised/transductive), there are also different ML paradigms for feature selection. The three major paradigms for feature selection (135; 265) are

- The *filter* paradigm, in which an ML algorithm evaluates features one-by-one. For each feature f_j, the algorithm evaluates how *important/good* the feature is, using some measure of importance/goodness. Features that are important are kept, and those which are not are dropped. A *forward selection* algorithm *adds* features one-by-one, while a *backwards selection* algorithm starts with the

entire set of features $\{f_j\}$ and *removes* them one-by-one. (A hybrid algorithm that combines adding and removing features is also common.) I will investigate forward selection algorithms (Section 6.5).

- The *wrapper* paradigm, in which there are two ML algorithms: one that proposes a subset of features, and another which evaluates how well the task can be performed using those features (the "predictor" algorithm). Usually, this evaluation is done by training the predictor algorithm on data that uses only the proposed features and evaluating the accuracy of the inference tool it learns. The wrapper paradigm is useful when the data has a high degree of correlation over the features. However, searching over subsets of the features is non-trivial, and requires re-training the predictor algorithm once a new subset is proposed. I do not investigate wrapper algorithms here.

- The *embedded* paradigm, which does feature selection in the process of training. Two common ways of doing so are to *regularize* the behavior of the algorithm (e.g., using a penalty for overly-complex decision rules), or by estimating how the objective function of the algorithm would change if features were added/subtracted. I consider one algorithm in this paradigm, an L_1-regularized support vector machine (Section 6.5)

Each of these paradigms is appropriate under different circumstances. The filter paradigm is most useful if each of the features is more-or-less independently related to the property of interest. (Though as we'll see, it can also be applied with some success when there is correlation between the features.) Filtering is quick and computationally cheap, because the measure of importance has to be computed only once for each feature. The wrapper paradigm is most useful as an "off-the-shelf" method for doing feature selection. Finally, algorithms in the embedded paradigm may be more efficient in terms of time to solution, because they are trained only once.

6.3 Feature selection for QCVV tasks

The central question of this chapter is "Can ML algorithms successfully select, from the circuits in a GST experiment design, a *reduced set* of circuits such that distinct noise types A and B can be reliably distinguished?". I call this problem "circuit reduction", and will use the phrases "feature selection" and "circuit reduction" almost interchangeably (see next paragraph). The phrase "can be reliably distinguished" should be taken to mean "can be separated using a linear decision surface". A GST experiment design with index L[3] is said to be "circuit reducible for noise types (A, B)" if there exists a subset of the circuits in the experiment design with the property that using only those circuits, A and B can be reliably distinguished.

The distinction between "circuit reduction" and "feature selection" is necessary because of subtleties involved when doing feature selection on *feature engineered* feature vectors. As an example, suppose $\mathbf{f}$ is a base GST feature vector of dimension d (i.e., is a list of outcome probabilities of a GST experiment design), and feature engineering is done to extend $\mathbf{f}$ to include all squares of its components: $\mathbf{f} \to \phi_{\mathrm{SQ}}(\mathbf{f})$. If a feature selection algorithm determines that none of the quadratic components of $\phi_{\mathrm{SQ}}(\mathbf{f})$ are necessary for distinguishing the the two noises, then feature selection has *succeeded*. However, the *circuits* selected are still the d circuits originally used to construct $\mathbf{f}$ in the first place, so circuit reduction has *failed*. ***Thus, feature selection is useful insofar as it leads to circuit reduction.***

6.3.1 Why feature selection is generally hard

Formally, feature selection is simply the construction of a new feature space out of a set of features. If $\mathcal{F}$ is the "base" feature space for a problem, then feature selection

[3]Recall that each GST experiment design can be indexed by a family parameter L introduced in 5.3.2 that controls the length of the longest-depth circuit in the design. In what follows, I will use L as an index for GST experiment designs.

is the task of deciding which basis vectors (features) in $\mathcal{F}$ are actually necessary to solve the problem at hand, thereby defining a new, "reduced" feature space $\mathcal{F}'$, where $\dim(\mathcal{F}') < \dim(\mathcal{F})$.

To see why feature selection is generally a challenging problem, consider a collection of labeled feature vectors $(\mathbf{f}_j, y_j), y_j \in \pm 1$ in a generic feature space $\mathcal{F} \subset \mathbb{R}^d$. Suppose further that in $\mathcal{F}$ the collection is separable. The task of feature selection is to choose a minimal subset of the basis vectors of $\mathcal{F}$ such that when the feature vectors are expanded in that basis, the collection remains separable. (If it becomes inseparable, then the two classes cannot be distinguished using ML classification algorithms!) The set of of all possible combinations of the basis vectors is quite large – it's the *power set* of $\mathcal{F}$, denoted $\mathcal{P}(\mathcal{F})$, and has size 2^d. Any element $s \in \mathcal{P}(\mathcal{F})$ defines a feature space $\mathcal{F}_s$, by the map

$$\mathbf{f} \in \mathcal{F} \to \sum_{j \in s} f_j \mathbf{e}_j \in \mathcal{F}_s. \tag{6.1}$$

The feature space $\mathcal{F}_s$ is simply $\mathcal{F}$, but where every component of $\mathbf{f}$ where every index $j \notin s$ has been removed. The feature selection task we're interested in is defined by the following optimization problem:

$$\begin{aligned} s^\star = & \min_{s \in \mathcal{P}(\mathcal{F})} |s| \\ & \text{s.t. } \mathcal{F}_s \text{ is separable.} \end{aligned} \tag{6.2}$$

Here, $|s|$ means "the cardinality of s". Notice this problem may have degenerate solutions (i.e., there could be many minimal elements of $\mathcal{P}(\mathcal{F})$ that yield separable feature spaces). The introduction of other constraints would be necessary in order to choose among them (see Section 6.7).

Equation (6.2) is a discrete combinatorial optimization problem over 2^d possible solutions, so in general, brute-force optimization is infeasible. For this reason, the ML community has introduced several heuristic algorithms, mentioned in the discussion of feature selection paradigms in Section 6.2.

6.3.2 Problem statement

The problem considered in this chapter can now be formally stated: "Given a GST experiment design L, two *gapped* noise models A and B, and a feature map ϕ or ϕ_{SQ}, then (a) determine whether the experiment design is circuit reducible, and (b) identify a less complex experiment design (a subset of circuits in the GST experiment design) that is sufficient for distinguishing A from B using linear ML classification algorithms".

A solution to this problem statement is an element $s \in \mathcal{P}(\mathcal{F})$ with the following properties: (a) $|s| < d$, where d is the dimension of the native feature space $\mathcal{F}_L$ defined by the action of ϕ, and (b) in the feature space $\mathcal{F}_s$, the two noise models are linearly separable. The existence of *any* such element provides a sufficient affirmative answer to the question given in the problem statement.

Note that the problem statement doesn't require finding the *smallest* experiment design. That would be ideal, but certifying that a given experiment design is the smallest possible would require a certificate showing all less-complex designs (i.e., designs with a smaller number of circuits) fail to be sufficient (i.e., showing the reduced feature spaces are not linearly separable).

One subtle point in the problem statement is that A and B need to be *gapped*. Ideally, "gapped" in this context would mean "the gate sets generated by arbitrary realizations of noise type A or B would be non-intersecting". However, the kinds of noise I'll consider here can be described as *cones*, parameterized by some strength η. Therefore, as $\eta \to 0$, two "gapped" noise models would limit to the same noise model[4], thereby rendering them un-gapped. To remedy this, I typically impose a lower bound on the noise strength η to ensure that even at its smallest value, the two noise models are gapped. The next section discusses the different noise models, and explains how they are gapped.

[4]Namely; the *noiseless* one!

As an overview of the remainder of this chapter, Section 6.4 discusses the noise models I'll be using. Section 6.5 introduces the feature selection algorithms that were investigated. Section 6.6 presents the results, and demonstrates that circuit reduction is possible for every pair of noise models considered. Finally, Section 5.5 concludes with a discussion on other experiment design problems for which feature selection algorithms would be useful, and touches on the idea of using reinforcement learning to create QCVV experiment designs from scratch.

6.4 Discussion of noise models

To demonstrate that feature selection/circuit reduction is possible in general using GST circuits as the primitive set to optimize over, I'll consider circuit reduction for 4 QCVV tasks, all of which are classification problems. They are of the form "distinguish between...

- ...arbitrary coherent and arbitrary stochastic noise".
- ...amplitude damping noise and arbitrary stochastic noise".
- ...isotropic Pauli stochastic noise and anisotropic Pauli stochastic noise".
- ...arbitrary Pauli stochastic noise and significantly non-Pauli stochastic noise".

These noise models are constructed in such a way that for a given lower bound on the noise strength η, the two models are gapped. For ease of exposition on how these noise types are simulated, the noise models are presented in terms of the time dynamics they generate under the Lindblad master equation. The general Lindblad equation describing noisy dynamics is

$$\dot{\rho} = -i[\rho, H_0]/\hbar - i[\rho, H_e]/\hbar + \sum_{j,k=1}^{3} h_{jk} \left(\sigma_j \rho \sigma_k - \frac{1}{2}\{\sigma_k \sigma_j, \rho\} \right), \tag{6.3}$$

where the ideal gate is $G_0 = e^{iH_0}$. The Hamiltonian error term H_e generates unitary

errors, and the coefficient matrix h generates non-unitary, Markovian noise and satisfies $h \geq 0$. In the following subsections, any Lindbladians written in matrix form are written in the basis of matrix units, *not* the Pauli basis.

6.4.1 Coherent noise

Recall from Section 5.3.1 that coherent noise generates time dynamics that are purely unitary, so that $h = 0$. To sample realizations of coherent noise, H_e is taken to be a traceless matrix drawn from the Gaussian unitary ensemble:

$$H_e = \sum_{j=1}^{3} c_j \sigma_j \quad c_j \sim \mathcal{N}(0, \eta^2). \tag{6.4}$$

To ensure the coherent noise model is gapped with respect to arbitrary stochastic noise, I take $\eta \geq 10^{-4}$. See Appendix I for a complete description of the values of η chosen.

6.4.2 Arbitrary stochastic noise

Recall from Section 5.3.1 that arbitrary stochastic noise has no unitary dynamics except for those generated by H_0, so that $H_e = 0$. Realizations of arbitrary stochastic noise are generated by sampling h according to

$$h = S^{-1} D S \ , \ D = \mathrm{diag}(|a|, |b|, |c|) \ , \ a, b, c \sim \mathcal{N}(0, \eta^2), \tag{6.5}$$

where S is a random SO(3) matrix. To ensure this noise model is gapped with respect to coherent noise, I take $\eta \geq 10^{-4}$. See Appendix I for a complete description of the values of η chosen.

6.4.3 Isotropic/anisotropic Pauli stochastic noise

Pauli stochastic noise is an subclass of arbitrary stochastic noise where h is constrained to be a diagonal matrix. Consequently, Equation (6.3) can be written as

$$\dot{\rho} = -i[\rho, H_0]/\hbar + \sum_j^3 h_j S_j[\rho], \tag{6.6}$$

where

$$S_j[\rho] = \sigma_j \rho \sigma_j - \rho. \tag{6.7}$$

The coefficients h_j give the *rate* of dephasing about the σ_j axis.

Under isotropic Pauli stochastic noise, $h_1 = h_2 = h_3 \equiv c$. That is, isotropic Pauli stochastic noise generates the *depolarizing* channel. Defining anisotropic Pauli stochastic noise is a bit more subtle, because we need to ensure the two models are gapped. Here, "anisotropic Pauli stochastic" is taken to mean *one* of the h_j is zero, while the other two are equal, e.g., $h_1 = 0, h_2 = h_3 = c$. This introduces a gap between the two moels, as there will always be some h_j that is 0 under anisotropic noise, but is equal to c under isotropic noise. Unless $c = 0$, the two noise models are different. If anisotropic meant $h_1 \neq h_2 \neq h_3$, the models are not really gapped, since as one could take $h_j = c(1 + \epsilon_j)$, with $\epsilon_j \ll 1$, and the noise would be essentially isotropic.

To generate realizations of isotropic or anisotropic Pauli stochastic noise, I take $c \sim |\mathcal{N}(0, \eta^2)|$. To ensure the two noise models are gapped, I take $\eta \geq 10^{-2}$. (See Appendix I for a complete description of the values of η chosen.)

6.4.4 Significantly non-Pauli stochastic noise

Non-Pauli stochastic noise is another subclass of arbitrary stochastic noise. Because Pauli stochastic noise corresponds to h being diagonal, non-Pauli stochastic noise corresponds to non-diagonal h. Ideally, "significantly" non-Pauli stochastic noise would mean that h has no on-diagonal elements. However, this cannot be the case: because $h \geq 0$, if h has zeros along the diagonal, then $h = 0$. Therefore, "significantly

non-Pauli stochastic noise" is taken to mean that h is *weakly, off-diagonally dominant* in its rows:

$$|h_{jj}| \leq \sum_{k \neq j} |h_{jk}| \ \forall \ j. \tag{6.8}$$

Because h is 3×3 and satisfies $h \geq 0$, these conditions are easily written as

$$\begin{aligned} 0 &\leq h_{11} \leq |h_{12}| + |h_{13}| \\ 0 &\leq h_{22} \leq |h_{21}| + |h_{23}| \\ 0 &\leq h_{33} \leq |h_{13}| + |h_{23}|. \end{aligned} \tag{6.9}$$

To generate realizations of significantly non-Pauli stochastic noise, I generate realizations of arbitrary stochastic noise, and then *post-select* on those realizations that satisfy the weakly, off-diagonally dominant condition above. To ensure this noise model is gapped with respect to Pauli stochastic noise, I take $\eta \geq 10^{-3}$. See Appendix I for a complete description of the values of η chosen.

6.4.5 Amplitude damping noise

Amplitude damping noise is a process that transfers population to some particular state of the qubit. The canonical example is amplitude damping to the ground state (e.g., energy dissipation by spontaneous decay). Unlike the other noise models discussed, this model is more naturally described directly in terms of its Lindbladian superoperator, as opposed to the time dynamics it generates given by Equation (6.3). Appendix C.3 gives a derivation of the noise model.

The canonical model of amplitude damping noise is one that damps to the $|0\rangle\langle 0|$ state of the qubit at a rate γ. This noise model is generated by the following Lindbladian[5]:

[5] $\mathcal{L}$ is written here in the basis of matrix units.

$$\mathcal{L} = \begin{pmatrix} 0 & 0 & 0 & \gamma \\ 0 & -\gamma/2 & 0 & 0 \\ 0 & 0 & -\gamma/2 & 0 \\ 0 & 0 & 0 & -\gamma \end{pmatrix}. \tag{6.10}$$

Appendix C.3 provides details on amplitude damping noise. In contrast to the canonical model – amplitude damping to the $|0\rangle\langle 0|$ state – the noise model considered here is more general, and allows for amplitude damping to *any given pure state*. There exists a yet more general model ("generalized amplitude damping"), which allows for damping to *mixed* states. However, the noise models "amplitude damping to mixed states" and "arbitrary stochastic noise" are not gapped. For this reason, I use the slightly more general model of amplitude damping, but not its fully general version.. To simulate amplitude damping to arbitrary pure states, it suffices to simulate amplitude damping to the $|0\rangle\langle 0|$ state, and then rotate the Bloch sphere, so that

$$\mathcal{E} = \mathrm{Exp}\left[\mathcal{H}_0 + \mathcal{U}^\dagger \mathcal{L} \mathcal{U}\right], \tag{6.11}$$

where $\mathcal{U}[\rho] = U\rho U^\dagger$, and U is a randomly-chosen element of $SU(2)$, and $\mathcal{H}_0$ is the superoperator representation of H_0. In simulations, I take $\gamma = \eta$, which fixes the rate of amplitude damping, and choose U at random by choosing a random polar and azimuthal angle. To ensure this noise models is gapped with respect to arbitrary stochastic noise, I take $\eta \geq 10^{-3}$. See Appendix I for a complete description of the values of η chosen.

Using each of these noise models, I generated a collection of GST data for different experiment designs (i.e., values of L). Appendix I provides details about the data sets generated. These data sets are the input to the ML algorithms for feature selection, discussed in the next section.

6.5 Description of the algorithms

Section 6.2 introduced different paradigms for feature selection using ML algorithms. I investigated five ML algorithms for feature selection. Four use the *filtering* paradigm: randomized optimization (Section 6.5.1), principal component analysis (Section 6.5.2), mutual information (Section 6.5.3), and perceptrons (Section 6.5.4). The last – an L_1 regularized support vector machine (Section 6.5.5) – uses the *embedded paradigm*.

Recall that in the filtering paradigm, features are added or removed one by one from a set of candidate features until a good set s has been selected. Here, a set is "good" if the reduced feature space $\mathcal{F}_s$ is linearly separable. For the filtering algorithms considered, *forward selection* was used: given a measure of "importance" for a feature f_j, denoted $g(f_j)$, forward selection starts with $s = \{\}$ and adds features in ranked order of their importance until $\mathcal{F}_s$ is linearly separable:

$$\begin{aligned} s = \min_{M} & \bigcup_{j=1}^{M} f_j, \\ s.t. \;\; & g(f_1) \geq g(f_2) \geq \cdots g(f_M) \\ & \mathcal{F}_s \text{ is linearly separable.} \end{aligned} \tag{6.12}$$

This algorithm works well if (a) each feature is independently important, and (b) there are large disparities in the goodness of each feature. As an example of how this algorithm would perform poorly, consider a measure of goodness that is *uniform* over the features. Then, forward feature selection will have to choose all of them!

I defer a discussion of the embedded paradigm until Section 6.5.5.

6.5.1 Randomized optimization: a baseline heuristic

One of the most naive heuristics to solving Equation (6.2) would be to randomly pick a subset of the features of some given size and see if the resulting feature space is

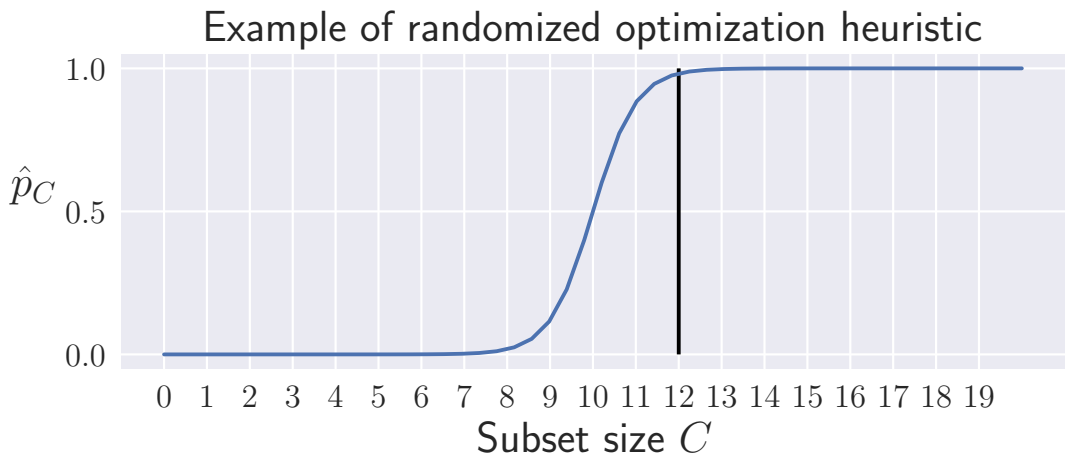

Figure 6.2: **Pictorial description of randomized optimization heuristic**. By generating random subsets of a given size C and estimating the probability that subset of size C is linearly separable, we can estimate an upper bound on how many features are necessary for circuit reduction. (In the figure, it's 12, because if $C \geq 12$, $\hat{p}_C \sim 1$.) In this example, if an ML algorithm cannot find fewer than 12 circuits, that algorithm is not really providing any advantage over randomized optimization.

separable. Admittedly, this isn't a particularly efficient approach, but it does provide a baseline. In particular, let $p_C = \Pr(\text{randomly-selected } \mathcal{F}_s \text{ is separable given } |s| = C)$. This probability can be estimated by Monte Carlo sampling N_{samples} elements $s \in \mathcal{P}(\mathcal{F})$ of size $|s| = C$, and counting the fraction of times the resulting feature space is linearly separable:

$$\hat{p}_C = \frac{\#\text{ times } \mathcal{F}_s \text{ was linearly separable given } |s| = C}{N_{\text{samples}}}. \tag{6.13}$$

Notice that the number of elements s of size C is $\binom{d}{C}$,which means that any fixed choice of N_{samples} will necessarily undersample the power set. In general then, we expect $\hat{p}_C \sim 0$ for many values of C. However, suppose almost every element of size C that is selected gives a linearly separable $\mathcal{F}_s$. This provides strong evidence that *any* element whose size $|s| \geq C$ will give a linearly separable $\mathcal{F}_s$. In this way, randomized sampling estimates an *upper bound* on the number of features that should need to be selected (see Figure 6.2).

Phrased another way, if $\hat{p}_C \sim 1$, then with high probability, any randomly-selected subset of that size will yield a separable feature space. Thus, if an ML algorithm doesn't select a subset whose size is less than C, that algorithms is performing quite poorly, because we could just randomly sample one! On the other hand, if $\hat{p}_C \sim 0$, then *random* selection of subsets may not yield a linearly separable feature space. This doesn't mean *intelligent* selection won't find a subset. To evaluate the performance of the randomized optimization heuristic, for each value of C, I generated 50 random subsets of that size, and computed the fraction of those subsets that were linearly separable. If $\hat{p}_C \geq .95$, C is accepted as the number of circuits selected by the algorithm; otherwise, $C \to C + 1$.

In this sense, randomized optimization provides a baseline measure of performance to compare the ML algorithms to. It won't provide a lower bound for C – what's the best that could be done, even in principle – but it does provide an estimate of an upper bound.

Note that randomized optimization doesn't use any notion of importance of the features. For this reason, we expect methods that *do* use such information to tend to outperform randomized optimization.

6.5.2 Principal component analysis (PCA): a geometric measure of feature importance

Recall that the PCA of a given data set yields a list of eigenvectors of the covariance matrix of the data, $\{\mathbf{e}_j\}_{j=1}^{K}$. In general, the eigenvectors will not line up with the basis for the feature space $\{\mathbf{f}_k\}$, and so they usually have a dense representation in terms of them:

$$\mathbf{e}_j = \sum_{k=1}^{d} f_{jk} \mathbf{f}_k. \tag{6.14}$$

There's a major caveat to using PCA for feature selection; namely, PCA focuses on the directions of maximal variance within the data, and those directions may actually not be particularly useful for classifying the noise. For example, consider a simple problem in $\mathbb{R}^3$ with 2 "pancakes" in the $x-y$ plane, separated by a vertical offset in the z-direction. Clearly, the feature to use in distinguishing which pancake you have is to look at the z-component of the feature vector. However, if the vertical separation is small compared to the variation in x and y (i.e., $\sigma_x, \sigma_y \gg \sigma_z$), then PCA identifies z as an "uninformative" direction, because the data don't vary much along it.

That issue aside, there are several ways of taking a PCA analysis of the data and doing feature selection. I explored several. My first attempts used methods that quantified how much a given feature "contributed" to the PCA components. There are several notions of "how much a given circuit contributes to the PCA components" (i.e., how *important* that circuit is). I looked at:

- Absolute value of components: let

$$g(f_k) = \sum_{j=1}^{K} |f_{jk}|. \tag{6.15}$$

  This quantity is closely related to $\sum_{j=1}^{K} I[f_{jk}]$, with $I[x]$ as the indicator function, which *counts* the number of times the feature f_k shows up in the PCA components. By considering an absolute value, $g(f_k)$ reduces the significance of features that show up in many PCA components, but where the coefficient is small.

- Variance-weighted sum of components: let

$$g(f_k) = \sum_{j=1}^{K} \sigma_j |f_{jk}|. \tag{6.16}$$

  This quantity weights the contributions by the variance, and assigns higher

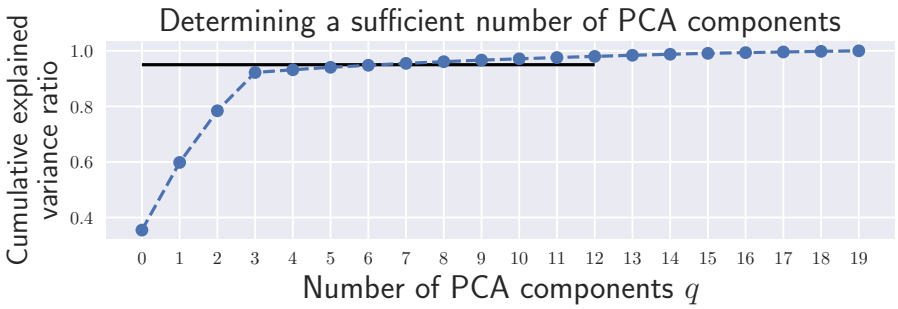

Figure 6.3: **Determining the value of q in Equation** (6.17). q is specified by thresholding the *cumulative explained variance ratio* $v(q) = \sum_{j=1}^{q} \sigma_j / \sum_j \sigma_j$. In particular, q is chosen as the smallest value satisfying $v(q) = .95$. For this simple example (5 i.i.d isotropic Gaussian blobs in $\mathbb{R}^{20}$, with variance 4), $q = 7$.

importance to features that show up in PCA components that explain larger amounts of the variance in the data.

- Projector component: let $\Pi_q = \sum_{j=1}^{q} \mathbf{e}_j \mathbf{e}_j^T$, and define

$$g(f_k) = (\Pi_q)_{kk} = \sum_{j=1}^{q} (f_{jk})^2. \tag{6.17}$$

  This measure places higher importance on features that would be most useful for dimensionality reduction. Recall Section 5.4.2, where the PCA of $\mathcal{C}_1$ was used to embed a 92-dimensional feature space into a 2-dimensional representation, by projecting each feature vector onto the 1st and 2nd eigenvectors of the covariance matrix.

Note that in defining $g(f_k)$ in Equation (6.17), the value of q first needs to be specified. In most ML applications, q is chosen by looking at how much variance the first q components explain about the data. (That is, if PCA component $\mathbf{e}_j$ has variance σ_j, it explains $\sigma_j / \sum_k \sigma_k$ of the variance in the data.) Here, q is set so that the first q components explain 95% of the variance in the data (see Figure 6.3). Note that setting $q = d$ means $\Pi_q = \mathcal{I}$, so setting q too high is undesirable.

Preliminary results indicated that the "absolute value of components" and "variance-weighted sum of components" measures of goodness tended to perform poorly (i.e., selected more features) than the "projector component" measure of importance. For this reason, I excluded them from the analysis that follows.

I also considered one other heuristic based on the observation that the diagonal elements of covariance matrix of the data indicates how much the data varies along each feature. If $\hat{\Sigma}$ is the estimated covariance matrix for the data, a natural measure of importance is

$$g(f_k) = (\hat{\Sigma})_{kk} = \sigma_k^2, \tag{6.18}$$

which is the variance of the data along f_k.

6.5.3 Mutual information: an information-theoretic measure of feature importance

If knowing the value of a given feature is sufficient for inferring the noise label, then clearly that feature is important. Another way of phrasing that statement is "feature f_k is important if it contains a lot of *information* about the noise label". The *mutual information* (72) (defined below) is a principled way to compute how much information one random variable contains about the another. To see why the mutual information is a good measure of feature importance, consider the following scenario. Let f be a feature that's going to be used to classify whether the noise is type 1 or 2. Suppose f has the following distribution

$$f \in \begin{cases} \text{Unif}(0, a) & \text{when noise is type 1} \\ \text{Unif}(a + \epsilon, 1) & \text{when noise is type 2.} \end{cases} \tag{6.19}$$

Knowing whether $f \in [0, a]$ or $[a + \epsilon, 1]$ immediately allows for an exact inference of the noise type. A feature with this distribution would have high mutual information

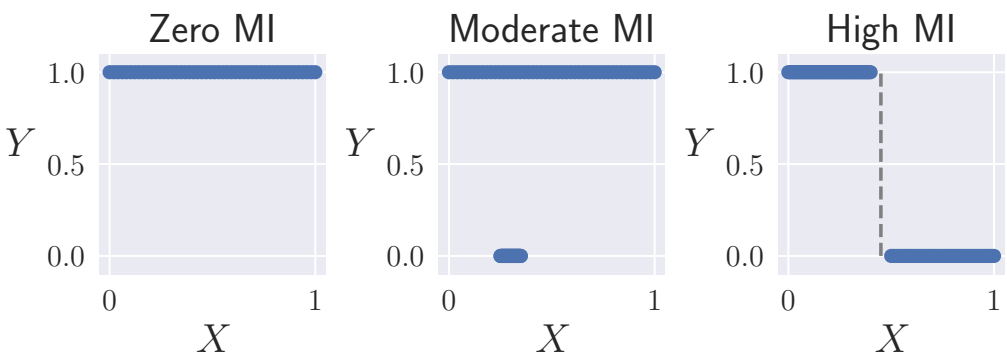

Figure 6.4: **Example: Mutual information between a continuous and discrete variable**. The mutual information $I(X;Y)$ indicates how much information X contains about Y. Left: When knowledge of X provides no information about Y (i.e., they are independent), the mutual information is 0. Middle: When knowledge of X provides some information about Y, the mutual information is non-zero (in this example, $\hat{I}(X;Y) = .44$). Right: When knowledge of X provides complete information about Y, the mutual information is maximal ($\hat{I}(X;Y) = .698$). NOTE: Here $\hat{I}(X;Y)$ is computed in *nats* (base-e), so the maximum amount of information contained in a Boolean random variable is $\sim .693$ nats.

with respect to the noise label. Figure 6.4 gives toy example showing how the distribution of the feature (the random variable X and the noise label Y) affect the mutual information. At one extreme, if the noise label is *independent* of the feature, then the mutual information is 0 (left-most panel). If the noise label depends strongly on the feature (right-most panel), the mutual information is high.

If X and Y are two random variables with joint probability density function $p(x, y)$ and marginals $p(x) \equiv \int p(x, y)\ dy$ and $p(y) \equiv \int p(x, y)\ dx$, the mutual information $I(X;Y)$ is

$$I(X;Y) = \int \int p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right)\ dx\ dy. \tag{6.20}$$

Equivalently, $I(X;Y)$ is the Kullback-Leibler divergence between the joint distribution $p(x, y)$ and the product of its marginals $p(x)p(y)$:

$$I(X;Y) = D_{\mathrm{KL}}(p(x, y)\ ||\ p(x)p(y)). \tag{6.21}$$

Notice that $I(X;Y) = 0$ if, and only, if $p(x,y) = p(x)p(y)$, meaning X and Y are *independent* from one another. If X and Y are independent, there's no correlation between them ($\mathrm{Cov}(X,Y) = 0$), so knowing one cannot yield any information about the other.

Each feature f_j is a continuous random variable (because it is the outcome probability of a noisy circuit), while the noise label is *discrete*. This is easily accommodated in the definition of mutual information by replacing one of the integrals by a sum:

$$I(f_j;Y) = \sum_{y\in\pm 1} \int p(f_j,y) \log\left(\frac{p(f_j,y)}{p(f_j)p(y)}\right)\, d(f_j). \tag{6.22}$$

The features derived from a GST experiment design are not independent from one another, since the same noisy gate set is used to estimate the outcome probabilities of the various circuits. For this reason, using the *conditional mutual information* $I(f_j;Y|f_k,f_l,\cdots)$ would be the proper measure of mutual information to use. However, estimating this quantity could be hard, especially given that the features are continuous, but the noise label is discrete. Here, I make the simplifying assumption that the features can be treated as independent random variables, and use known algorithms for estimating $I(f_j;Y)$ (179; 261). That is, the measure of importance is the estimated mutual information: $g(f_k) = \hat{I}(f_k;Y)$.

There's an important caveat to be mindful of when using this feature selection algorithm on engineered feature vectors. The mutual information is invariant under homeomorphisms (continuous functions that are bijections, and whose inverses are continuous), which means that *reparameterizing* the features doesn't change their information content. In particular, this means that $I(f_j;y) = I(f_j^2;y)$, and that if the feature map ϕ_{SQ} is used, then this algorithm will select *both* f_j and f_j^2.

6.5.4 Perceptron-based feature selection

Because a linear classifier learns a hyperplane, the coefficients of that hyperplane itself indicate which circuits are useful for classification. Recall that a linear classification algorithm learns a decision rule of the form

$$c(\mathbf{f}) = \text{sign}[\boldsymbol{\beta} \cdot \mathbf{f} + \beta_0]. \tag{6.23}$$

If any component $\boldsymbol{\beta}_j$ of the hyperplane is zero, it follows that the feature $\mathbf{f}_j$ is not necessary for predicting the noise label. Thus, the coefficients themselves can be used to guide a feature selection process. This can also be seen by computing the gradient of the argument of the sign function with respect to f_j:

$$\frac{\partial(\boldsymbol{\beta} \cdot \mathbf{f} + \beta_0)}{\partial f_j} = \boldsymbol{\beta}_j. \tag{6.24}$$

Therefore, $\boldsymbol{\beta}_j$ controls how much *fluctuations* in f_j affect the classification rule. A sensible measure of importance derived from a separating hyperplane is absolute value of the coefficients:

$$g(f_j) = |\boldsymbol{\beta}_j|. \tag{6.25}$$

Figure 6.5 gives an example of this idea.

A crucial assumption here is that the hyperplane does in fact separate the data. A hyperplane that doesn't separate the data doesn't provide useful information about which features are important. I use separating hyperplanes learned by the perceptron algorithm. Recall that the perceptron algorithm will converge to *some* hyperplane (provided one exists), but not necessarily an *optimal* one (Appendix D.2). For this reason, this approach is perhaps best used in the following way: run the perceptron algorithm with many different initializations, which leads to a *distirbution* of importance values $g(f_j)$, which can then be post-processed to select, e.g., the features which tend to have the most weight in that distribution.

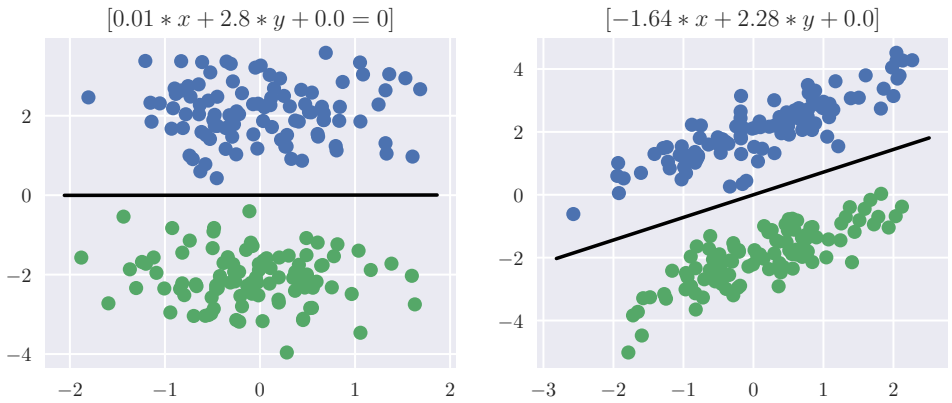

Figure 6.5: **Example: Using a perceptron for feature selection**. The coefficients of the hyperplane learned by the perceptron algorithm can be used for feature selection. In the left panel, the hyperplane has coefficients $\boldsymbol{\beta} = (0.01, 2.8)$, indicating that perturbations in x-component of the feature vector will not dramatically affect the classification output, so it can be removed from the feature vector. On the right panel, the hyperplane has coefficients $\boldsymbol{\beta} = (-1.64, 2.28)$ indicating that perturbations in both features will impact the classification output, so they both need to be kept.

6.5.5 The embedded paradigm: L_1-regularized SVM

The four algorithms just discussed use the filtering paradigm for feature selection. This paradigm has a notable drawback – the algorithms don't have access to information about how well a (different) ML algorithm can do in solving the given task using the features selected! Knowing this information – "If feature f_j is added, classification accuracy goes up 10%" – would be extremely useful for feature selection. This observation gives rise to the *embedded paradigm* for feature selection wherein feature selection and actually solving the ML task are done concurrently.

A paradigmatic algorithm that falls into this paradigm is an L_1-regularized support vector machine (SVM). Note this algorithm is different than soft-margin SVM algorithm discussed in Chapter 5. The following remainder of this section goes through a derivation showing that by starting with the soft-margin SVM, reinterpreting the

slack variables as loss functions, and thinking of the margin as an L_2-regularization of the hyperplane learned by the SVM, there is a straightforward generalization of the soft-margin SVM algorithm to one that imposes and L_1-regularization penalty. Importantly, *the resulting algorithm will no longer strive to maximize the margin of the hyperplane it learns.*

Consider the optimization problem defining the soft-margin SVM:

$$\begin{aligned} &\min_{\boldsymbol{\beta},\beta_0} \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_j \xi_j \\ &\text{s.t. } y_j(\boldsymbol{\beta}\cdot\mathbf{f}_j + \beta_0) \geq 1 - \xi_j \;\forall\; j \\ &\qquad\qquad \xi_j \geq 0 \;\forall\; j. \end{aligned} \tag{6.26}$$

The slack variables $\{\xi_j\}$ give the algorithm the flexibility to bring the hyperplane closer to some points. Another way to think about them – one that naturally leads to a useful generalization of the SVM algorithm – is to observe that the slack variables relate to the hyperplane by the requirement

$$\xi_j \geq 1 - y_j(\boldsymbol{\beta}\cdot\mathbf{f}_j + \beta_0), \tag{6.27}$$

as well as the requirement $\xi_j \geq 0$. This leads to the idea of the slack variables as quantifying the *loss* incurred by a given hyperplane. Recall that if the slack variables are all zero, then the hyperplane learned by the algorithm is one where every point either lies on the ± 1 decision boundary, or is further away from it.

Viewed this way then, the loss for a given hyperplane can be defined as

$$L(\boldsymbol{\beta}, \beta_0, j) = \max(0, 1 - y_j(\boldsymbol{\beta}\cdot\mathbf{f}_j + \beta_0)). \tag{6.28}$$

This loss function is called the *hinge loss*, because it looks like a "hinge" about 1 (see Figure 6.6).

By viewing the slack variables as loss functions, the SVM algorithm can be reinterpeted as one that *minimizes the hinge loss, while also imposing an L_2-regularization*

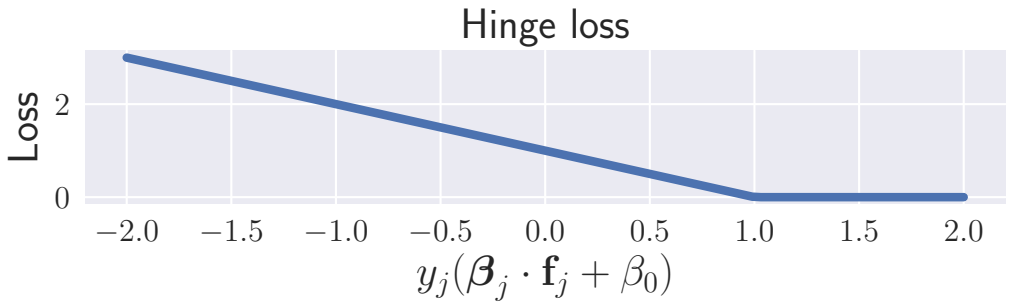

Figure 6.6: **Behavior of the hinge loss**. The hinge loss (Equation (6.28)) is a common loss function for ML algorithms. It penalizes misclassification with a severity that's proportional to $(\boldsymbol{\beta} \cdot \mathbf{f}_j + \beta_0)$. If a hyperplane successfully classifies a point, the loss is zero.

penalty on the hyperplane:

$$\min_{\boldsymbol{\beta},\beta_0} \left[\frac{1}{2}||\boldsymbol{\beta}||^2 + C \sum_j L(\boldsymbol{\beta}, \beta_0, j) \right] \tag{6.29}$$

The hyperparameter C specifies the relative importance of minimizing the loss versus having a sufficiently L_2-regularized hyperplane. By viewing the SVM algorithm in this way, other generalizations of the SVM algorithm are possible, by considering different regularization penalties. For the purposes of feature selection, we want the SVM algorithm to learn a hyperplane whose normal vector $\boldsymbol{\beta}$ is *sparse*. The canonical measure of sparseness of a vector is the L_0 "norm":

$$L_0(\mathbf{f}) = \sum_k I[\mathbf{f}_k] \qquad I[x] = \begin{cases} 1 & x \neq 0 \\ 0 & x = 0 \end{cases} . \tag{6.30}$$

The function L_0 simply counts the number of non-zero elements in $\mathbf{f}$. However, it's not a true norm, because $L_0(a\mathbf{f}) \neq |a|L_0(\mathbf{f})$ (i.e., is not *homogenous*), as can be seen directly from the definition. L_0 also has the issue that it's a discrete function of $\mathbf{f}$, which means optimizing it is a combinatorial optimization problem. For this reason, "relaxing" it in such a way that its relaxation can be optimized efficiently is

necessary. The convex relaxation of L_0 is the L_1 or *nuclear norm*:

$$L_1(\mathbf{f}) = \sum_k |\mathbf{f}_k|. \tag{6.31}$$

To see why, consider any set of vectors satisfying $L_0(\mathbf{x}) = 1$. Construct the convex hull of that set. For any vector on this hull, $L_1(\mathbf{x}) = 1$. Because the hull contains the points used to construct it, and it is convex, L_1 is said to be the convex relaxation of L_0. Notice $L_1(\mathbf{0}) = L_0(\mathbf{0})$, which ensures that the optimal point is included in the feasible set of the L_1 norm. Further, L_1 norm penalizes small components just as much as it does larger ones, thereby driving all components to zero uniformly. So, to use an SVM-like algorithm to determine a sparse set of features, we can change the regularization term from L_2 to L_1. The L_1-regularized optimization problem used to define the SVM algorithm is

$$\min_{\boldsymbol{\beta},\beta_0} \left[||\boldsymbol{\beta}||_1 + C \sum_j L^2(\boldsymbol{\beta}, \beta_0, j) \right], \tag{6.32}$$

where L is given in Equation (6.28) (93). Note though that the objective function defining this algorithm cares only about *sparse solutions* and *minimizing the loss* - we've discarded the idea of *maximizing the margin*. In particular, *this change also means that the hyperplane learned by the SVM* ***is not*** *a maximal-margin one!* Because $||\mathbf{x}||_2^2 \leq ||\mathbf{x}_1^2|| \leq d||\mathbf{x}||_2^2 \; \forall \; x \in \mathbb{R}^d$, it follows that $1/||\boldsymbol{\beta}||_2^2 \geq 1/||\boldsymbol{\beta}||_1^2$. Consequently, the margin of the hyperplane learned by an L_2-regularized SVM is greater than that of a L_1-regularized SVM, in general.

This algorithm has a hyperparameter (C) that needs to be tuned appropriately, and which will affect the number of features chosen. (If C is small, then the L_1 penalty dominates the objective function.) Because we are interested in a subset of circuits in a GST experiment design that give rise to a linearly separable feature space, C should be set just high enough so that the classification accuracy is 1. Recall the discussion about the perceptron feature selection algorithm – if the hyperplane is

extremely inaccurate, it won't provide any useful information about which features to keep.

6.6 Results

6.6.1 Separability of the feature spaces

Feature selection for noise classification only makes sense in a feature space where the two noise models can be separated. (If they cannot be separated in the full feature space, they won't be separable in any subset of it.) Chapter 5 showed the feature engineering enabled linear separability for distinguishing coherent and stochastic noise under the $L = 1$ GST experiment design. Here, I also use feature engineering to enable linear separability, though I only use one feature engineering map: ϕ_{SQ}, defined in Equation (5.21). Table 6.1 shows which feature spaces are separable for the noise model comparisons discussed in Section 6.4.

For most of the noise comparisons, the feature spaces under that result from ϕ or ϕ_{SQ} are linearly separable, particularly at higher-L experiment designs. This makes sense, as those experiment designs add higher-depth circuits to the experiment design, and therefore amplify the noises more. Having established that the feature spaces are linearly separable, the next question is whether the corresponding experiment designs are circuit reducible for the noise models.

6.6.2 Circuit reduction possible for all noise models

I used the ML algorithms discussed in Section 6.5 to find heuristic solutions to the circuit reduction problem introduced in Section 6.3.2. Table 6.2 shows how many circuits were selected by each ML algorithm for a given value of L, feature map, and noise model. It also shows what fraction of circuits in the experiment design were selected. Crucially, ***every fraction is less than one – some substantially***

L	Feature map	Noise	Separable
1	ϕ	I	No
		II	No
		III	Yes
		IV	No
	ϕ_{SQ}	I	Yes
		II	Yes
		III	Yes
		IV	Yes
2	ϕ	I	No
		II	Yes
		III	Yes
		IV	??
	ϕ_{SQ}	I	Yes
		II	Yes
		III	Yes
		IV	Yes
4	ϕ	I	Yes
		II	Yes
		III	Yes
		IV	Yes
	ϕ_{SQ}	I	Yes
		II	Yes
		III	Yes
		IV	Yes

Table 6.1: **Separability of feature spaces under different noise models**. For the "Noise" column, I = amplitude damping/stochastic, II = coherent/stochastic, III = isotropic/anisotropic Pauli stochastic, and IV = Pauli/non-Pauli stochastic. Using a higher-L experiment design or using feature engineering generally enables linear separability of all the noise models. NOTE: the "??" for IV at $L = 2$ indicates that the separability test was inconclusive (because the solvers used all failed). If the entry is "No", there is a certificate for the inseparability; see Appendix E.

so – indicating that circuit reduction is possible for all the noise models considered. Again, I emphasize that the experiment designs selected are not known to be minimal, meaning further reductions could be possible.

The *amount* of circuit reduction depends on many factors, including the noise models

	L	Feature map	Noise	L1 SVM	MI	Naive PCA	PCA	Perceptron	Random
Circuits	1	ϕ	III	49	87	74	74	70	73
		ϕ_{SQ}	I	55	6	31	30	54	41
			II	62	12	31	5	27	34
			III	55	79	31	31	36	55
			IV	19	19	35	36	23	62
	2	ϕ	II	91	114	111	128	83	141
			III	21	126	32	49	12	55
		ϕ_{SQ}	I	66	7	41	27	47	44
			II	111	24	37	9	31	44
			III	18	125	43	33	8	58
			IV	38	33	57	57	44	81
	4	ϕ	I	178	212	216	191	183	206
			II	83	163	146	181	86	184
			III	74	300	52	83	13	59
			IV	75	160	158	157	101	163
		ϕ_{SQ}	I	95	10	77	14	74	56
			II	289	55	78	17	48	66
			III	83	299	87	67	21	71
			IV	–	58	118	102	–	112
Reduction	1	ϕ	III	0.53	0.95	0.80	0.80	0.76	0.79
		ϕ_{SQ}	I	0.6	0.07	0.34	0.33	0.59	0.45
			II	0.67	0.13	0.34	0.05	0.29	0.37
			III	0.6	0.86	0.34	0.34	0.39	0.60
			IV	0.21	0.21	0.38	0.39	0.25	0.67
	2	ϕ	II	0.54	0.68	0.66	0.76	0.49	0.84
			III	0.12	0.75	0.19	0.29	0.07	0.33
		ϕ_{SQ}	I	0.39	0.04	0.24	0.16	0.28	0.26
			II	0.66	0.14	0.22	0.05	0.18	0.26
			III	0.11	0.74	0.26	0.20	0.05	0.35
			IV	0.23	0.20	0.34	0.34	0.26	0.48
	4	ϕ	I	0.4	0.48	0.49	0.43	0.41	0.47
			II	0.19	0.37	0.33	0.41	0.2	0.42
			III	0.17	0.68	0.12	0.19	0.03	0.13
			IV	0.17	0.36	0.36	0.36	0.23	0.37
		ϕ_{SQ}	I	0.22	0.02	0.17	0.03	0.17	0.13
			II	0.66	0.12	0.18	0.04	0.11	0.15
			III	0.19	0.68	0.20	0.15	0.05	0.16
			IV	–	0.13	0.27	0.23	–	0.25

Table 6.2: **Circuit reduction results**. "Noise" labels the same as those used in Table 6.1. The top half ("Circuits") counts are the absolute number of circuits, while the bottom half ("Reduction") gives the fraction of circuits selected relative to the dimension of the feature space defined by the action of ϕ. NOTE: "Naive PCA" uses the importance measure defined in Equation (6.18), while "PCA" uses the importance measure defined in Equation (6.17).

being compared, the algorithm, and the feature space. An examination of the table indicates that the algorithms based on mutual information, PCA (Equation (6.17)), and running a perceptron tend to do much better than random circuit selection

or "Naive PCA" (Equation (6.18)). Also of note is that the embedded algorithm (L_1-regularized SVM) doesn't do as well in general.

The role of feature engineering in enabling circuit reduction is complex. Feature engineering increases the dimension of the feature space (because it adds new features to the feature vector). Note that the fractions presented in Table 6.2 are computed with respect to the dimension of the *non-feature-engineered* feature space. As the table makes clear, feature engineering sometimes helps with circuit reduction, and sometimes not.

Circuit reduction using *mutual information* either does really well, or really poorly. This is particularly true for the "isotropic vs. anisotropic Pauli stochastic" comparison (III). Figure 6.7 gives some hint about why this might be the case. The outcome probabilities under the two noise models tend to fall into two camps. In the first, the GST circuit is highly insensitive to the noise, so the outcome probability of the circuit tells us essentially nothing about the noise (rightmost plot). Circuits that are sensitive to the noise tend to have the same mutual information (leftmost plot), meaning that an optimization heuristic based on choosing features with highest mutual information will tend to have to choose many features. This problem is exacerbated under feature engineering, since $\mathbf{f}_j$ and $\mathbf{f}_j^2$ tell us the same amount of information about the noise type.

The "randomized optimization" heuristic (Section 6.5.1) was introduced as a *baseline* to compare the other ML algorithms to. Examining Table 6.2, this heuristic is generally outperformed by all the ML algorithms *except* the L_1-regularized SVM. See, e.g., $L = 4$, ϕ_{SQ}, noise type II (coherent/stochastic), where the randomized heuristic selects 66 circuits, while the L_1-regularized SVM selects 289!

Looking at the bottom half of Table 6.2, different ML algorithms also achieve substantially different levels of circuit reduction. Generally speaking, it appears that filtering algorithms based on mutual information, PCA, or running a perceptron can

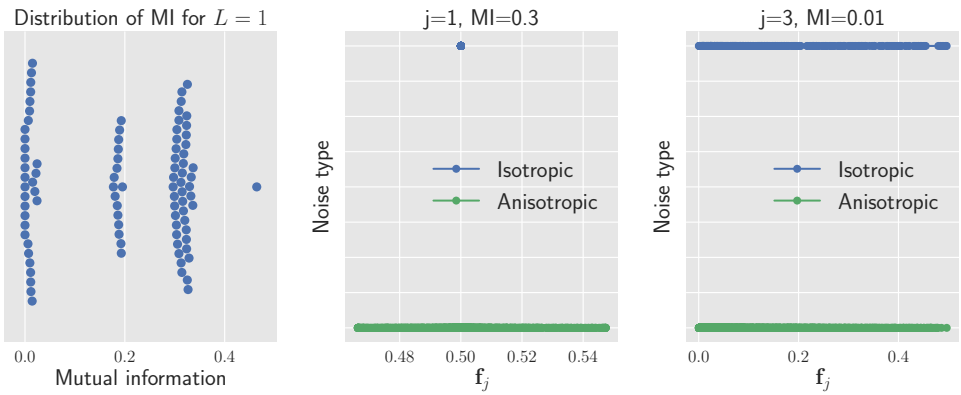

Figure 6.7: **Example: estimated mutual information (MI) for the $L = 1, \phi$ feature space for isotropic vs. anisotropic Pauli stochastic noise**. For noise type III in Table 6.2, most of the features end up either having very low mutual information for the noise type (rightmost plot), or the mutual information is almost the same across every feature (leftmost plot). The middle plot shows a typical distribution of the estimated outcome probability when the mutual information is high.

provide a level of reduction below 10% *for some noise models*. That is, for those noise models, they select fewer than 10% of the total circuits used in the GST experiment design.

Finally, Table 6.2 shows that all the noise models *except Pauli vs. non-Pauli stochastic* (IV) are substantially circuit-reducible, particularly with higher-L experiment designs and feature engineering. Compare the "Reduction" portion of the table for $L = 4, \phi_{\text{SQ}}$ to $L = 1, \phi_{\text{SQ}}$.

6.6.3 Mean depth of selected circuits

Table 6.2 indicates *how many* circuits were selected by the ML algorithms, but doesn't indicate *what* they were or what *properties* they have. Some circuits are more useful for characterizing certain kinds of noise, and part of the reason for using ML algorithms to identify such circuits is in the hope that they might yield some

L	Mean	Max	Feature map	Noise	L1 SVM	MI	Naive PCA	PCA	Perceptron
1	4.45	7	ϕ	III	4.4	4.6	4.4	4.4	4.5
			ϕ_{SQ}	I	4.3	5.3	4.5	4.5	4.6
				II	4.5	6.4	4.7	4.8	5.3
				III	4.7	4.5	4.4	4.4	4.5
				IV	4.7	4.6	4.1	4.3	5
2	5.12	8	ϕ	II	5.2	6.0	5.3	5.5	5.3
				III	6.6	5.2	6.4	5.5	5.8
			ϕ_{SQ}	I	5.4	6.6	6.0	5.7	5.7
				II	5.2	6.8	5.8	5.7	6.1
				III	6.8	5.2	5.8	5.8	5.8
				IV	5.2	5.6	5.2	5.2	5.6
4	6.40	10	ϕ	I	6.7	6.8	6.6	7.0	6.6
				II	6.2	8.1	6.5	7.4	6.4
				III	7.7	6.4	7.2	6.5	7.5
				IV	7	6.3	6.3	6.3	7.3
			ϕ_{SQ}	I	7.2	7.6	6.7	6.5	7.6
				II	6.4	8.3	6.5	8.0	8
				III	7.8	6.4	6.5	6.6	7
				IV	–	6.6	7.0	6.7	–

Table 6.3: **Mean circuit depth of circuits chosen by ML algorithms**. For comparison, the mean and maximum depth of all circuits in the experiment design are given (columns "Max" and "Mean", respectively). The algorithms choose a set of circuits whose average circuit depth is close to the average for the circuits in the experiment design.

intuition about the kind(s) of experiments that help us characterize the noise.

One intuition about GST circuits is that longer-depth circuits are more useful for amplifying noise. However, longer-depth circuits can also make noise *less distinguishable*: consider depolarization noise at rates r_1 and r_2. In the limit the circuit depth is very large, the outcome probability of that circuit is .5, *regardless of the rate*. Table 6.3 shows the *mean* circuit depth for the circuits selected by each algorithm when learning to select features in the feature space given by ϕ_{SQ}. Comparing this to the mean and maximum depth over all circuits for a given value of L, we see that the algorithms aren't necessarily selecting long circuits. Further, no algorithm in particular is selecting circuits that have wildly different average depth.

The properties of the feature space depend strongly on the experiment design. Section 5.4.5 discussed how the margin of hyperplane learned by the support vector

machine (SVM) algorithm relates to robustness of classification accuracy under finite-sample noise. Different experiment designs may have different levels of robustness to such finite-sample effects. Robustness to finite-sample effects is investigated in the next section, by considering the margin of the data in the reduced feature spaces. The next section investigates this property, shows that training an SVM was difficult, and presents simple *bounds* on the margin of the optimal-margin hyperplane.

6.6.4 Margins of the reduced feature spaces

A subset of the circuits for a given experiment design defines a new experiment design and a reduced feature space. Smaller subsets (less complex experiment designs) are more desirable if all else is equal – but usually all else is not equal! Therefore, other considerations may need to be taken into account when evaluating the experiment designs selected by feature selection algorithms.

For example, suppose algorithm A chooses 5 circuits, and algorithm B, 20. Algorithm A clearly found a smaller subset with lower experimental complexity. On the basis of "number of circuits to do", the experiment design selected by A might be preferable – it requires one quarter the number of experiments as that selected by B. But this might come at a price. For instance, if the *margin* separating the feature vectors is 10 times smaller in the reduced feature space selected by A, then 100 times as many samples will need to be taken to have the requisite statistical precision to ensure a reliable inference of the noise. Other properties of the reduced feature space may also be relevant, but its dimension (complexity of the experiment design) and margin are clearly critical.

The margin of the reduced feature spaces defined by the reduced experiment designs of Section 6.6.2 can be computed by training an SVM and then computing the margin of the hyperplane it learns. Because the data is linearly separable, a maximal-margin hyperplane exists, and an SVM can learn it. However, I encountered some

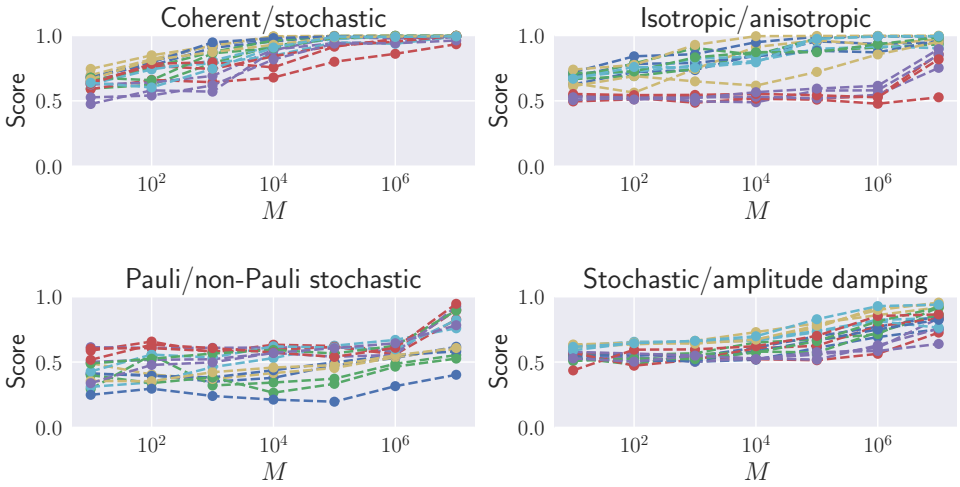

Figure 6.8: **Cross-validated accuracy of hyperplane learned by the SVM algorithm as a function of the maximum number of algorithm iterations** M. Each colored line is specified by a feature selection approach, GST experiment design, and feature map. For each value of M, a soft-margin SVM ($C = 10^{10}$) algorithm is run for M iterations, terminated, and then the accuracy of the hyperplane learned is computed. Cross-validation was done using a 10-fold shuffle-split approach, with 10% of the data held back for testing.

computational challenges in evaluating the margin of the reduced feature space; the SVM algorithm took an extremely long time before converging for any one reduced feature space.

The SVM algorithm solves the optimization problem given in Equation (D.25), (equivalently, Equation (D.26)). This optimization problem can be solved in an iterative fashion, say by gradient ascent. To get a handle on why the SVM algorithm was taking so long, I ran the following test: run the SVM for M iterations, terminate it, and evaluate the accuracy of the hyperplane it learns. As $M \to \infty$, the accuracy should increase to 1, as the data sets are linearly separable. Figure 6.8 show that the rate at which the algorithm's accuracy increases depends strongly on the noise model.

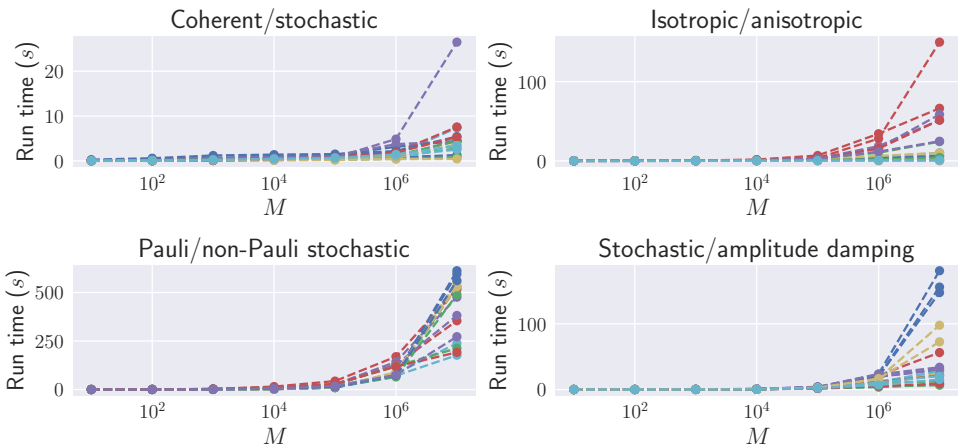

Figure 6.9: **Cross-validated runtime of the SVM algorithm as a function of the maximum number of algorithm iterations** M. The runtime of the algorithm appears to blow up around $M = 10^6$. Note the different y-axis scales for the different panels; in particular, for Pauli vs. non-Pauli stochastic noise, a single run takes upwards of 10 minutes on average.

A deeper look at the behavior of the algorithm also indicates that the *runtime* is highly dependent on the noise models, as indicated in Figure 6.9. For every noise comparison except "coherent vs. stochastic", the fact that the learned hyperplane has not yet achieved an accuracy of 1 suggests that runtimes in Figure 6.9 will generally keep increasing. Once the accuracy reaches 1, the algorithm terminates, so the fact that accuracies are not yet 1 means the algorithm will need to keep running, thereby increasing the runtime.

This behavior suggests there a difficulty in using the SVM algorithm to learn a hyperplane separating the noises. The general theory of SVMs indicates the runtime for separating N data points in a feature space with d dimensions goes as $\mathcal{O}(d^* N^2)$ or $\mathcal{O}(d * N^3)$. For this reason, the SVM algorithm is usually not recommended when the number of data points is on the order of a few tens of thousands. However, *the data set with the largest number of points – coherent vs. stochastic – has the lowest runtimes!* I am not entirely sure why this is the case.

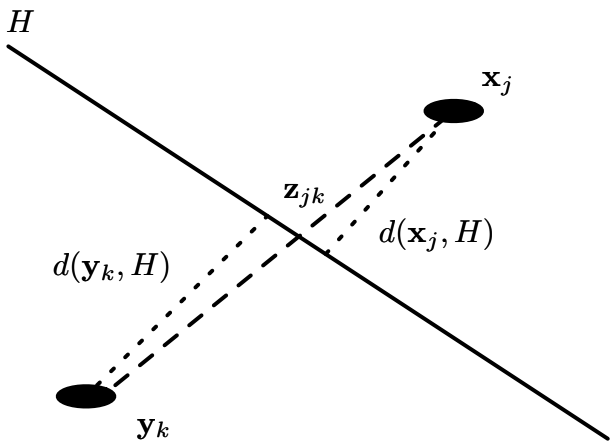

Figure 6.10: **Geometry to determine an upper bound on the margin of any separating hyperplane**. Given any separating hyperplane H, its margin is upper-bounded by the minimum of $d(\mathbf{x}_j, \mathbf{y}_k)/2$, where $\mathbf{x}_j, \mathbf{y}_k$ are feature vectors with different labels. To prove this, it suffices to observe that H bisects the line joining $\mathbf{x}_j$ and $\mathbf{y}_k$.

These complications do pose a problem to determining the exact margin of the reduced feature spaces. However, the margin can be *bounded* using two simple arguments. First, the margin of any separating hyperplane is upper bounded by the half of the minimum distance between any two feature vectors with different noise labels.

This can be proven as follows (see Figure 6.10): consider two sets of feature vectors $\{\mathbf{x}_j\}_{j=1}^{M}, \{\mathbf{y}_j\}_{j=1}^{M'}$, and assume the two sets are linearly separable. Take any hyperplane H that separates them. Because H is a separating hyperplane, it bisects the line joining any $\mathbf{x}_j$ to any $\mathbf{y}_k$. Let $\mathbf{z}_{jk}$ denote the point where that intersection occurs. Notice

$$d(\mathbf{x}_j, \mathbf{y}_k) = d(\mathbf{x}_j, \mathbf{z}_{jk}) + d(\mathbf{y}_k, \mathbf{z}_{jk}). \tag{6.33}$$

Now,

$$d(\mathbf{x}_j, \mathbf{z}_{jk}) \geq d(\mathbf{x}_j, H) \text{ and } d(\mathbf{y}_k, \mathbf{z}_{jk}) \geq d(\mathbf{y}_k, H). \tag{6.34}$$

Therefore,

$$d(\mathbf{x}_j, \mathbf{y}_k) \geq d(\mathbf{x}_j, H) + d(\mathbf{y}_k, H). \tag{6.35}$$

Finally, M_H is the minimum distance from H to any feature vector, meaning

$$d(\mathbf{x}_j, H) \geq M_H \text{ and } d(\mathbf{y}_k, H) \geq M_H. \tag{6.36}$$

From this, it follows that

$$M_H \leq \frac{d(\mathbf{x}_j, \mathbf{y}_k)}{2} \; \forall \; j, k \implies M_H \leq \min_{j,k} \frac{d(\mathbf{x}_j, \mathbf{y}_k)}{2}. \tag{6.37}$$

The conditions under which the equality saturates are easily understood. The inequalities in Equation (6.34) saturate when H bisects the line joining $\mathbf{x}_j$ and $\mathbf{y}_k$ at a right angle. The inequalities in Equation (6.36) saturate when both of $\mathbf{x}_j$ and $\mathbf{y}_k$ correspond to feature vectors that are closest to H. Putting these two facts together, it follows that Equation (6.37) saturates when $\mathbf{x}_j$ and $\mathbf{y}_k$ are *support vectors*.

Computing all pairwise distances has a complexity that is $\mathcal{O}(MM')$, so this upper bound can be computed efficiently. Demonstrating a lower bound is simple, as the margin of *any* separating hyperplane gives a lower bound on the margin of the maximal-margin hyperplane. What's more, this lower bound is easy to compute *in practice*, as we can use the margin of the hyperplane used to certify linear separability in Section 6.6.1. Let M_{LP} denote the margin of this hyperplane. Note that the lower bound could be easily improved by constructing a different separating hyperplane with a higher margin. Such hyperplanes could be found by, e.g., running the perceptron algorithm with many different initializations and post-selecting on the hyperplane with the largest margin.

Therefore, the margin of the optimal-margin separating hyperplane in the reduced feature space is bounded as $M_{\text{LP}} \leq M_{\text{optimal}} \leq \min_{jk} \; d(\mathbf{x}_j, \mathbf{y}_k)/2$. These bounds are presented in Figure 6.11 as a function of L, the feature map, and the noise models being compared. Focusing on coherent and stochastic noise (blue-hued lines), the

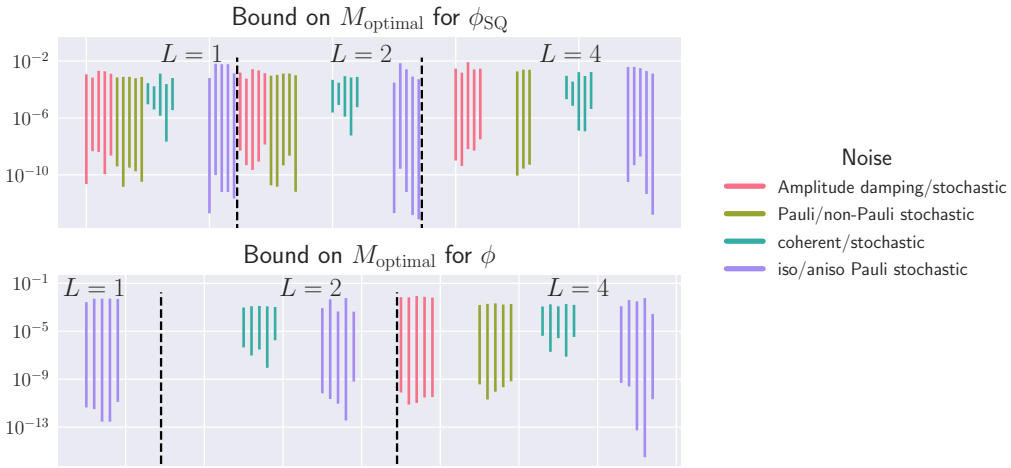

Figure 6.11: **Bound on the optimal margin M_{optimal} for the reduced feature spaces**. The vertical range of each solid vertical line denotes an interval containing M_{optimal}. The two panels split out the feature map. Data are grouped by value of L indexing the GST experiment design; each color indicates the noise models being compared.

interval containing the optimal margin are rather narrow, extending only over 3 orders of magnitude. This is in contrast with the other noise models, where the range is anywhere from 4 to 6 orders of magnitude. This may relate to the fact the SVM algorithm succeeded in learning a separating hyperplane (Figure 6.8).

Each reduced feature space given in Figure 6.11 is specified by a choice for L, the ML algorithm for doing feature selection, the feature map, and the noise models being compared. Figure 6.11 aggregates the data in terms of L and the noise model. Figure 6.12 shows another view on this data, by plotting the bound as a function of the number of features selected, with different choices of the aggregating variable. The panels in this figure indicate that *the bound doesn't depend strongly on the number of features selected*. Taking Figures 6.11 and 6.12 together, the bounds on the margin depend more strongly on the *noise models being compared* than anything else.

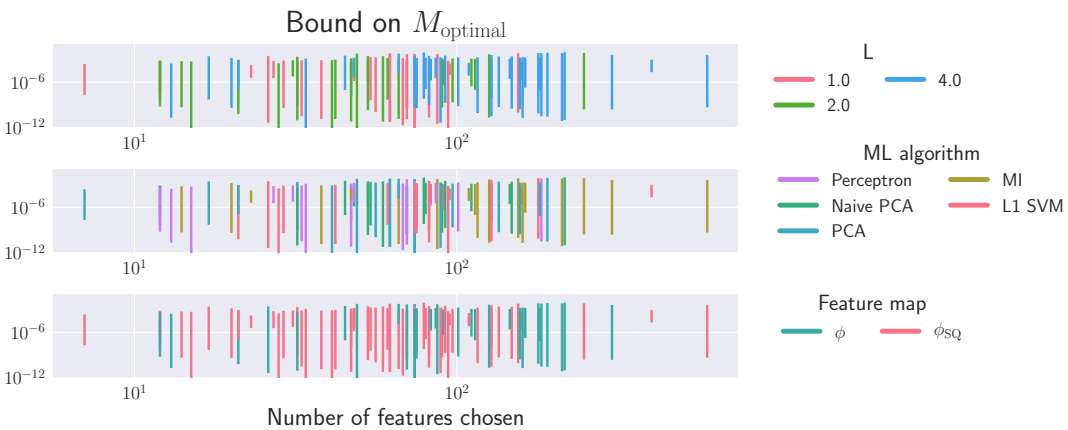

Figure 6.12: **Relation between number of features chosen and bound on M_{optimal}.** Each panel splits out the data presented in Figure 6.11 in different ways, either by the experiment design L, the ML algorithm, or the feature map. The bound appears not to depend strongly on these properties.

6.7 Conclusion and discussion

To characterize a property of a QIP, experimental data is required. Creating an experiment design – a specification of experiments (circuits) to be performed – sufficient for characterizing a given property is a non-trivial task. This chapter showed that ML algorithms can generate experiment designs. In particular, focusing on experiment designs for characterizing Markovian noise affecting a single qubit, ML algorithms for feature selection can generate suitable experiment designs by pruning the experiment design used for GST.

As mentioned in Section 6.2, feature selection is something of a dark art. A brute-force search over all possible experimental designs is generally impractical, since the number of experiment designs over d features is 2^d. For this reason, various heuristics are typically necessary. While state-of-the-art ML algorithms (e.g., deep feature synthesis) can automate much of the feature selection process, the work presented here is probably more typical, in the sense that the algorithms are used with a lot of "hand-holding". A fruitful research direction would be to examine whether state-

of-the-art algorithms can be deployed for more-or-less automated QCVV experiment design.

Imposing different requirements on the experiment design learned by ML algorithms leads to different experiment designs being selected. Here, the focus was solely on the criteria "the experimental design has to contain fewer circuits than the original GST experimental design" and "the resulting feature space has to be linearly separable". For the noise models and ML algorithms considered, meeting these criteria was feasible (Table 6.2). A reasonable conjecture would be that loosening the second requirement from "linearly separable" to "separable by some decision surface" will lead to further reductions in the number of circuits in the experimental design.

Another desideratum that would be useful to consider is maximizing the margin of the reduced feature spaces. This work made some progress in understanding this property for some of the reduced feature spaces (Section 6.6.4), by showing that the margin of the optimal-margin separating hyperplane is (a) upper-bounded by a geometric property of the data set that can be computed efficiently, and (b) lower-bounded by the margin of any separating hyperplane, such as a hyperplane used to establish linear separability of the data. The lower bound follows trivially from the definition of the optimal margin.

This analysis may be of independent interest for quickly determining an upper bound on the amount of statistical noise that would be tolerable, and establishing a lower bound on the number of samples required. In addition, since *any* separating hyperplane provides a lower bound on the margin, the margin of that hyperplane immediately yields an upper bound on the number of samples required. Note that the data has to be linearly separable for this analysis to be applicable.

One notable result from this investigation is that *maximizing the margin of the reduced feature space* and *minimizing its dimension* may be constraints that are at odds with one another. In particular, the derivation of the L_1-regularized SVM (Section

6.5.5) showed that "maximizing the margin" was equivalent to "L_2-regularizing the hyperplane". Using a different regularization principle (such as L_1) inevitably leads to a hyperplane that need not have the maximal margin. One way to balance both properties would be to define an SVM with both L_1 and L_2 penalties, in addition to a loss penalty.

For the problem of characterizing Markovian noise, ***the results presented in this chapter suggest that circuit reduction should be generally feasible***. In all 4 of the noise comparisons considered, every ML algorithm successfully identified a reduced experiment design. The experiment design selected by the algorithms depended strongly on the noise, the original GST experiment design, and the algorithm itself. Algorithms that seemed to do well include filtering algorithms based on mutual information or PCA. One result of note is the behavior of the L_1-regularized SVM. Based on its performance for the 4 noise comparisons considered here, whether this algorithm would do well for other QCVV tasks is not clear.

The results shown in Table 6.2 indicate that higher-L GST experiment designs, and using feature engineering, can lead to big impacts in the amount of reduction achievable by the ML algorithms. Of course, the *absolute* number of circuits selected tends to be a bit higher, simply because the original GST experiment design contains more circuits.

This chapter considered circuit reduction based on GST experimental designs. The algorithms presented here could also be applied to other experimental designs, such as those used randomized or unitarity benchmarking. The QCVV problem considered in Chapter 5 was motivated in part by the fact that the amount of coherence of the noise impacts the feasibility of quantum error correction (18; 133; 77; 314; 273). The amount of coherent noise can be estimated by randomized/unitarity/purity benchmarking. Identifying a simpler experiment design that could provably demonstrate the noise is purely stochastic – or would provide a high-accuracy estimate of the

unitarity itself – would be useful. This could be done by using circuit reduction on the circuits necessary for those benchmarking protocols.

Another way ML algorithms could create QCVV experiment designs is by *reinforcement learning* (RL) (290). RL algorithms learn a *policy* that dictates their action in response to external variables (the "environment"), with the goal of maximizing the reward associated with their actions. RL has already been used to create new quantum error correction techniques (108), and has been used to construct experimental designs for simple classical problems (115; 116).

As QCVV practitioners and theorists work to flesh out the spectrum of QCVV techniques, a variety of experiment designs will be required. ML algorithms for feature selection or RL can help develop them.

Chapter 7

Conclusions and outlook: scalable QCVV of NISQ processors

7.1 Conclusions

Characterizing NISQ processors is a challenging problem. The QCVV techniques that worked well for few (1 or 2) qubit QIPs are beginning to show the limits of their applicability on many (5-10) qubit QIPs, and suggests they will need to be modified or replaced to characterize 20+ qubit QIPs. This is not to say that the techniques themselves are flawed; on the contrary, they have been quite successful and useful! Instead, what should be modified is the *methodology* of QCVV. The methodology that underlies most existing QCVV techniques today is *"use statistical models to describe a QIP's behavior"*. There are two reasons why that methodology is inadequate for characterizing NISQ processors. First, the number of parameters in commonly-used models grows too rapidly with the number of qubits (the "curse of dimensionality"). Second, models with a small number of parameters can fail to capture some of the complexities of noise affecting a NISQ processor (see the discussion in Section 2.4). Both of these inadequacies imply that the next generation of QCVV

techniques will need to use *modest-complexity, yet expressive*, models. This thesis presents two methodologies for developing such techniques; namely, statistical model selection and machine learning. These two methodologies are largely complementary to one another, and address different problems to scalability.

Taming the curse of dimensionality for any model is a well-studied statistics problem. One solution is to use statistical model selection to choose between several competing models in a principled way. However, tomography (of all kinds) subtly differs from canonical statistical inference problems. This difference arises because the models used in tomography have *boundaries*, which imposes *constraints* on the parameters of the model. Examples of constraints in tomography include positivity of quantum states and complete positivity of quantum channels.

Constraints couple the parameters together, so they cannot be varied independently. This is unlike the setting of most statistical inference problems where model selection is used, where the models typically do not have boundaries. Most classical model selection results were not derived with the presence of boundaries in mind. This means blithely porting over statistical model selection to tomography in particular (and QCVV in general) has complications.

Chapter 3 considered this issue in great detail, and focused on the impact of the positivity constraint $\rho \geq 0$ on the behavior of the maximum likelihood estimator. I showed that quantum state space, viewed as a statistical model, does not satisfy a powerful property called *local asymptotic normality* (LAN). This follows immediately from the fact that maximum likelihood estimates in quantum state space may not be Gaussian distributed. (For details, see Section 3.3.)

Since quantum state space doesn't satisfy LAN, it follows that most classical statistical model selection techniques will give erroneous results. Chapter 3 examined the behavior of the loglikelihood ratio statistic λ, and presented numerical evidence showing that its behavior under the classical *Wilks theorem* is nothing like its ob-

served behavior in practice. The Wilks theorem generally over-predicts $\langle\lambda\rangle$, which itself is used to set thresholds for rejecting smaller (fewer-parameter) models in favor of larger (more complex) ones. Because the Wilks theorem over-predicts, thresholds set based on its prediction for $\langle\lambda\rangle$ would be too high, and would fail to reject smaller models when there is in fact enough evidence to do so. I remedied this situation in two ways.

First, I defined a generalization of LAN that *is* applicable to *any* model with convex constraints, classical or quantum. This generalization, *metric-projected local asymptotic normality* (MP-LAN) embeds a constrained model into one that does satisfy LAN (usually, by lifting the constraints). For models satisfying MP-LAN, many properties of the maximum likelihood estimator follow from considering properties of the maximum likelihood estimator in the unconstrained model, and then determining how those properties change when the constraint(s) is (are) imposed.

A subtle point is that while the positivity constraint clearly imposes the *same* constraint everywhere in state space, the way it manifests itself in statistical inference (i.e., state tomography) is *not the same*: as Chapter 3 shows, the behavior of the maximum likelihood estimator depends strongly on the rank of ρ_0.

MP-LAN is a natural generalization of LAN, in the sense that ***models satisfying MP-LAN have similar properties to models that satisfy LAN***. As shown in Section 3.4.1, if a model satisfies MP-LAN, then: (a) the loglikelihood ratio statistic λ is equivalent to squared loss as measured by the Fisher information, (b) asymptotically, the behavior of the maximum likelihood estimator is entirely determined by its behavior in a shrinking region around ρ_0, and (c) this region is equivalent to the *tangent cone* at ρ_0. All three of these properties also hold for models that satisfy LAN.

Second, I derived an approximation for $\langle\lambda\rangle$ for d-dimensional quantum state spaces. Classically, $\langle\lambda\rangle$ is equal to the number of parameters in the model. The result

given in Equation (3.43) indicates that ***in the presence of constraints, the expected value of the loglikelihood ratio statistic is usually less than the number of parameters in the model, and depends strongly on properties of the underlying true parameter***. In particular, the result I derived depends on the rank of ρ_0. This result suggests that thinking about an "effective" state-space dimension is a good way to get a handle on statistical model selection, by treating different regions in state space as different-dimensional models within the larger d-dimensional state space.

One of the consequences of this result is that the number of parameters of a model may be less important than the "effective" number of parameters when doing model selection. Consider two state spaces with dimension D and $D + d$. Classically $\langle \lambda(\mathcal{M}_D, \mathcal{M}_{D+d}) \rangle = 2Dd + d^2$, while Equation (3.43) gives $\langle \lambda(\mathcal{M}_D, \mathcal{M}_{D+d}) \rangle \sim 6rd$, where $r = \text{Rank}(\rho_0)$. In the presence of constraints, the expected value of the loglikelihood ratio statistic is *not* the difference in the number of parameters of the two models. Instead, it's a difference of the *effective* number of parameters.

Chapter 3 introduced MP-LAN, proved properties of models that satisfy it, and used those properties in one particular application (computing $\langle \lambda \rangle$). Chapter 4 developed further applications of the MP-LAN formalism, and identified some connections between maximum likelihood and quantum compressed sensing. The first was that the expected rank of the maximum likelihood estimator provides a certificate of the rank of ρ_0. Most quantum compressed sensing protocols in use today that do not use this estimator are also self-certifying, though usually in a different sense than the certificate presented in Section 4.1. An interesting question is whether other techniques – which may use estimators other than maximum likelihood – are self-certifying in the sense described in Section 4.1.

I found a second connection by computing the expected rank of the maximum likeli-

hood estimator in idealized tomography[1] and showing that it is usually much smaller than the dimension of the state space. ***This implies that the positivity constraint yields low-rank estimates for free when doing state tomography using maximum likelihood estimation.***

Chapter 4 also examined whether the results derived in Chapter 3 hold up in a non-idealized tomographic setting. Section 4.2 discussed an application of statistical model selection in a situation where the Fisher information was not close to isotropic: choosing a Hilbert space dimension for a continuous-variable (CV) system. In that problem, the model is a d-dimensional quantum state space, and model selection is necessary because formally, the dimension of a CV system is infinite. For the particular measurement considered – optical heterodyne tomography – I presented numerical evidence showing that the Fisher information is not isotropic (evidenced by large condition numbers). Still, the expression derived in Equation (3.43) for $\langle\lambda\rangle$ held up reasonably well against numerical results for heterodyne tomography (Figure 4.6).

This result is surprising because it means that some results derived for idealized tomography also hold up in the non-ideal case. One reason why (at least in heterodyne tomography) is the fact that the contributions to λ from each matrix element of the estimate depend very strongly on the number of heterodyne counts (Figure 4.9). Analogizing heterodyne tomography to estimating the rate of a Poisson process, I showed that unless the number of counts in optical phase space is sufficiently high, then the contribution of the corresponding matrix element is very low (Appendix G).

Taking a holistic view of the first half of this thesis, my research on the challenges of model-based QCVV for NISQ processors indicates that the geometry of the model is extremely important when fitting model parameters and deploying statistical model

[1]Recall that the idealized setting is where the Fisher information of the measurement is isotropic, so that the state-space metric is Hilbert-Schmidt.

selection techniques. Although the results given in Chapters 3 and 4 were derived in the context of state tomography, they can also be applied in the context of process or gate set tomography. The notion of "an effective number of parameters for a model" is an important one, and highlights the importance of constraints in tomography[2].

The first half of this thesis addressed some of the pitfalls of model-based QCVV, by identifying and overcoming obstacles to using statistical model selection to tame the complexity curse of tomography. The second half picks up on the other reason mentioned at the start of these conclusions for why model-based QCVV isn't scalable: models with small numbers of parameters may not be adequate for characterizing new kinds of noise. Using *machine learning algorithms*, I showed that tailored, targeted QCVV techniques can be developed. These algorithms don't rely on statistical models[3]; instead, they search over a space of hypotheses to learn an analysis map that takes experimental data and returns an inference of a property of interest. The key insight that enabled the successful deployment of machine learning algorithms is ***QCVV data sets can be treated as feature vectors (input to machine learning algorithms) and by doing so, ML algorithms can learn using QCVV data***.

Chapter 5 showed that machine learning algorithms can learn new QCVV techniques. I presented a formalism for using supervised learning to develop targeted characterization techniques, and applied it with great success to the problem of distinguishing coherent and stochastic noise on a single qubit. However, there are several factors that contribute to the success or failure of machine learning algorithms for such tasks. They include the native geometry of the QCVV data sets (thought of as feature vec-

[2]An open research question is identifying which parameters of the model have to be inferred from data and which can be inferred from the constraint. An intriguing possibility is that that any random subset of the parameters whose size is close to the effective number of parameters is sufficient.

[3]Here, I am ignoring *generative* machine learning algorithms, which are used to generate samples from a probability distribution. Those algorithms do learn a statistical model.

tors), hyperparameter settings, and whether or not feature engineering is used. I investigated both hyperparameter tuning and feature engineering.

Hyperparameter turning – changing an algorithm's hyperparameters – boosted classification accuracy for many algorithms. That this should happen is intuitively obvious: by turning more knobs specifying the behavior of an algorithm, its performance can improve. However, as Figure 5.8 shows, if the geometry of the data set isn't amenable to being learned by a given algorithm, then hyperparameter tuning isn't necessarily going to help. For this reason, other techniques may be necessary to improve accuracy.

Changing the geometry of the data is a less straightforward but possibly more powerful way to do so. An algorithm makes certain assumptions about the geometry of the data, and when those assumptions don't hold, the algorithm performs poorly. This issue manifested itself in Section 5.4.1, which showed how different algorithms can behave quite differently on the same data. ***Feature engineering changes the geometry of the data, and can make the data more amenable to learning by different algorithms*** (recall Section 5.4.3). Knowledge of the geometry can developed either by considering *a priori* first principles ("domain-specific knowledge"), or by using dimensionality reduction (Section 5.4.2). For the problem considered in Chapter 5, feature engineering does help (Figure 5.10). The reason why is because the natural geometry of QCVV data sets generated by coherent and stochastic noise resembles a radio dish. That kind of "physics-informed" knowledge is extremely useful in evaluating which algorithms should be used.

Model-based QCVV techniques typically come with *error bars* to quantify the accuracy of the estimated parameters. At first glance, the fact that machine learning algorithms don't use statistical models would seem to imply that defining error bars is not possible. However, there are several ways to quantify the certainty of the QCVV

technique learned by the algorithm, such as the *margin* of a separating hyperplane[4]. As shown in Section 5.4.5, the margin quantifies how robust the classification accuracy is to perturbations of the data. Some algorithms, such as the hard-margin support vector machine (SVM) learn a decision rule (separating surface) that has the highest possible margin. As a consequence, ***if the inference tool learned by the algorithm needs to robust against finite-sample (statistical) noise, then the SVM is the best one to use***.

The work in Chapter 5 relied on the experiment design (set of circuits) that's used for gate set tomography. Targeted QCVV techniques don't necessarily require many circuits, and Chapter 6 took up the question of whether machine learning algorithms can come up with lean and simple experiment designs for targeted QCVV. Because the outcome probability of a given circuit in an experiment design is treated as one component of a feature vector, the problem of experiment design can be framed as the problem of *feature selection*: determining the right features (circuits) for solving the QCVV task. If the noise is assumed to be Markovian, then the circuits that comprise the experiment design for gate set tomography provide a natural "candidate set" of circuits to select from, since they are sensitive to arbitrary Markovian noise.

The features selected by the feature selection algorithms define a *reduced* feature space whose dimension is less than the dimension of the "native" feature space for the experiment design. One important consideration when deploying feature selection algorithms is "What property(ies) should the reduced feature space possess?". Chapter 6 required the reduced feature space identified by the feature selection algorithms to satisfy two properties: the number of circuits in the experiment design selected by the algorithm needed to be less than the number of circuits in the original experiment design, and the QCVV data should be linearly separable in the reduced feature space.

[4]Other approaches are possible, such as using *Platt scaling* to transform a classification output into a probability distribution over the noise types.

One important caveat concerns the interplay between feature selection and feature engineering, because feature engineering adds new features by combining existing features. Thus, a situation could arise wherein, relative to the dimension of the feature-engineered feature space, the dimension of the reduced feature space is smaller, but relative to the original feature space, it's the same. Therefore, feature engineering is useful in this context only insofar as it leads to a reduced feature space that's smaller than the original one.

By considering several different characterization problems of the form "Determine if the noise is type A or B", I showed that machine learning algorithms could identify a small subset of the circuits used for gate set tomography that met the desiderata given above. As each of the 4 characterization problems I considered was circuit reducible, I think ***pruning gate set tomography experiment designs circuits using feature selection algorithms is a viable approach to creating experiment designs for targeted QCVV tasks***.

Different feature selection algorithms approach the task of feature selection with different assumptions. For instance, algorithms that use a *filtering* paradigm rate each feature on the basis of some measure of importance, and then iteratively add features in decreasing order of importance. These algorithms (such as PCA or mutual information estimation) work particularly well when the features are more-or-less independent of one another. Other algorithms, such as the L_1-regularized SVM, approach the task from an *embedded* paradigm, in which the algorithm is trying to select few features while also taking into consideration the feasibility of classification in the reduced feature space. The embedded paradigm makes sense when the features are correlated (as is the case for gate set tomography feature vectors). Because outcome probabilities of the circuits are usually correlated in some way – after all, they are generated by the same underlying noisy gate set – the embedded paradigm might be preferable when developing new QCVV experiment designs.

Finally, different requirements on the reduced feature spaces may be necessary depending on what properties are of importance. For instance, by focusing solely on the properties "fewer circuits than the native feature space" and "linearly separable", the optimal separating hyperplane for the reduced feature space may have very small margin. The margin would be an important property to keep in mind because, as noted earlier, a high-margin hyperplane is more robust to finite-sample effects, and other kinds of noise.

Taken together, chapters Chapter 5 and Chapter 6 indicate that the subject matter expertise of QCVV practitioners can be leveraged in new ways through the use of machine learning algorithms. These algorithms do not entirely replace the need for human knowledge and insight; instead, the algorithms can be put in service to it. In this way, machine learning algorithms help QCVV practitioners extend the reach of their expertise.

7.2 Outlook

There are many challenges to realizing the near-term potential of quantum computers. At all levels of the "stack" – from device hardware, to control software, to programming libraries – more work will need to be done to strengthen and integrate the components of a QIP. By focusing on device characterization, my research addresses problems "close to the metal", as it were. As with the other levels in the stack, there is a lot of work to do in the QCVV "layer", especially with respect to developing new, scalable QCVV techniques.

Most traditional QCVV techniques rely on statistical models. This thesis has pointed out some of the ways statistics-based QCVV goes awry. However, this doesn't mean modeling is unnecessary. An emphasis on device modeling at all levels, from physics-informed modeling (e.g., electromagnetic fields on a chip) to abstract, statistical models that incorporate some physics knowledge (e.g., reduced GST), remains nec-

essary. There are numerous scientific questions about how to model the noise affecting NISQ processors. For example, the *locality* of the noise (the number of qubits it concurrently acts on) will significantly influence the kind(s) of models necessary for characterizing a QIP. If errors in a QIP can occur across arbitrarily many qubits concurrently (i.e., the error generators are *high-weight*) then increasingly-complex models are going to need to be developed to capture those kind(s) of noise. In turn, there is a commensurate increase in the difficulty in estimating model parameters. On the other hand, if errors occur in *bounded regions* of the QIP, and in non-overlapping patches, then simple, locality-respecting models can be used to describe those noises, and there may not be as great an increase in the resources necessary to estimate model parameters.

Machine-learned QCVV techniques offer one way to go beyond these parametric QCVV techniques. This thesis showed that machine learning algorithms can develop new QCVV techniques using data necessary for existing QCVV techniques (namely, GST). Those same algorithms could be used to, e.g., develop more efficient versions of randomized benchmarking (RB), either by learning a new technique for processing RB data, or by determining a small subset of RB circuits that's sufficient for inferring the RB number.

In this thesis, I trained machine learning algorithms using *synthetic* (computer-generated) data. As NISQ processors scale in size, generating such data will become increasingly difficult. Therefore, a potential path to scalable characterization using machine learning could involve using a highly-calibrated QIP to generate the training data by, e.g., injecting known noise into it. By bootstrapping from one generation of QIPs to the next, the need to do highly-involved computer simulations would be bypassed.

—

As quantum computing enters the NISQ era, there are numerous challenges to the development of next-generation quantum information processors. This thesis has contributed several ideas on methodologies for developing scalable QCVV techniques for NISQ processors, and gives me optimism that despite the hard and difficult path ahead, advances are achievable.

Appendices

Appendix A

Software acknowledgements

Much of this research relied on and leveraged software packages written by others. I'd like to thank those who write and maintain code for: cvxpy (86; 3), Jupyter (234), matplotlib (151), mpi4py (76), NumPy (303), pandas (211), pyGSTi (222), Python 2.7 (307), scikit-learn (233), SciPy (226), seaborn (316), and SymPy (215).

Appendix B

Solving Equation (3.39)

In Chapter 3, I state without a derivation the solution to Equation (3.39):

$$\frac{rq}{\epsilon} = \frac{4n^{1/4}}{15\pi}\left(2\sqrt{n} - \frac{q}{\epsilon}\right)^{5/2} \tag{B.1}$$

in terms of the variable $z \equiv q/\epsilon$, where $n = d - r$. In this section, I present a simple derivation of the solution. For details and comparisons between analytic approximations and numerical results, see this Jupyter notebook.

As previously mentioned, the above equation is a quintic polynomial in q/ϵ, meaning that an algebraic solution is generally impossible. Consequently, we'll end up having to make some approximations in order to obtain a tractable solution. Let's first start by defining $z \equiv q/\epsilon$, and then squaring both sides:

$$r^2 z^2 = \left(\frac{4}{15\pi}\right)^2 \sqrt{n}\left(2\sqrt{n} - z\right)^5 . \tag{B.2}$$

Then, define $y = z/2\sqrt{n}$, and factor the equation to give

$$y^2 = 32\left(\frac{2n}{15\pi r}\right)^2 (1-y)^5 . \tag{B.3}$$

For convenience, define $A = 32\left(\frac{2n}{15\pi r}\right)^2$:

$$y^2 = A\,(1-y)^5 . \tag{B.4}$$

Our task is to now solve this equation for y. Observe that we may write the defining equation for y as

$$y^{2/5} + A^{1/5}(y-1) = 0. \tag{B.5}$$

As mentioned in Section 3.5.3, with high probability $z \sim 2\sqrt{n}$, meaning $y \sim 1$. Using this fact, we can sipmlify the above equation by computing a Taylor series of $y^{2/5}$ about $y_0 = 1$. Using a computer algebra system gives the following Taylor series

$$y^{2/5} \approx 1 + .4(y-1) - .12(y-1)^2 + \mathcal{O}((y-1)^3). \tag{B.6}$$

As $r \to d$, $n \to 0$, and the solution to Equation (B.1) is $q = 0$. In turn, this means there will be discrepancies between the exact value of $y^{2/5}$ and its Taylor series. Replacing $y^{2/5}$ by its Taylor series gives a quadratic equation in y:

$$1 + (.4 + A^{1/5})(y-1) - .12(y-1)^2 = 0. \tag{B.7}$$

This equation is easily solved, and yields two roots:

$$y = \frac{25\sqrt[5]{A}}{6} \pm \frac{5}{6}\sqrt{25A^{\frac{2}{5}} + 20\sqrt[5]{A} + 16} + \frac{8}{3}. \tag{B.8}$$

By comparing the predicted values for y with numerical results, it turns out the negative root is the right one to pick. In turn, z is given by

$$z = 2\sqrt{n}\left(\frac{25A^{1/5}}{6} - \frac{5}{6}\sqrt{25A^{2/5} + 20A^{1/5} + 16} + \frac{8}{3}\right). \tag{B.9}$$

with $A = 32\left(\frac{2n}{15\pi r}\right)^2$. This expression can be greatly simplified by taking the $n \to \infty$ limit (again, using a computer algebra system). Doing so yields a series expansion for z:

$$\begin{aligned} z = 2\sqrt{n} &- \left(\frac{15\pi r}{2}\right)^{2/5} n^{1/10} + \frac{1}{5}\left(\frac{15\pi r}{2}\right)^{4/5} n^{-3/10} \\ &- \frac{1}{100}\left(\frac{15\pi r}{2}\right)^{6/5} n^{-7/10} - \frac{1}{100}\left(\frac{15\pi r}{2}\right)^{8/5} n^{-11/10}. \end{aligned} \tag{B.10}$$

Keeping the $n^{1/10}, n^{-3/10}$, and $n^{-7/10}$ terms gives the best agreement to numerical results. Therefore the approximate expression for z is

$$z \approx 2\sqrt{n} - \left(\frac{15\pi r}{2}\right)^{2/5} n^{1/10} + \frac{1}{5}\left(\frac{15\pi r}{2}\right)^{4/5} n^{-3/10} - \frac{1}{100}\left(\frac{15\pi r}{2}\right)^{6/5} n^{-7/10}. \quad \text{(B.11)}$$

Finally, a little bit of work is necessary to re-write this expression in a convenient way. Observe that it's possible to factor out a $2\sqrt{n}$ from the entire expression:

$$z = 2\sqrt{n}\left(1 - \frac{1}{2}\left(\frac{15\pi r}{2}\right)^{2/5} n^{-2/5} + \frac{1}{10}\left(\frac{15\pi r}{2}\right)^{4/5} n^{-4/5} - \frac{1}{200}\left(\frac{15\pi r}{2}\right)^{6/5} n^{-6/5}\right). \quad \text{(B.12)}$$

Define $x \equiv \left(\frac{15\pi r}{2n}\right)^{2/5}$, and the expression for z takes the simple form given in Equation (3.40):

$$\boxed{z \equiv q/\epsilon = 2\sqrt{n}\left(1 - \frac{1}{2}x + \frac{1}{10}x^2 - \frac{1}{200}x^3\right).} \quad \text{(B.13)}$$

Appendix C

Quantum noise affecting single qubits

C.1 Purely coherent and stochastic noise on single qubits

Consider a single-qubit unitary $U = e^{i\theta \mathbf{n}\cdot\boldsymbol{\sigma}}$, where $\boldsymbol{\sigma}$ is a vector of single-qubit Pauli matrices and $\mathbf{n}$ is a unit vector $|\mathbf{n}| = 1$. Under the action of purely coherent noise, U goes to some other unitary V, and the purity of the input state is preserved:

$$\mathrm{Tr}[(V\rho V^\dagger)^2] = \mathrm{Tr}[V\rho V^\dagger V\rho V^\dagger] = \mathrm{Tr}[\rho^2]. \tag{C.1}$$

On the other hand, purely stochastic noise will generally change the purity. As an example, consider

$$\mathcal{E}[\rho] = \frac{1}{2}\left(U\rho U^\dagger + U^\dagger\rho U\right). \tag{C.2}$$

This channel can be interpreted operationally as "With probability 1/2 rotate ρ about the axis $\mathbf{n}$ through angle θ, and with probability 1/2 rotate ρ about the axis

$\mathbf{n}$ through an angle $-\theta$." Utilizing the fact that $U = \mathcal{I}\cos\theta + i(\mathbf{n}\cdot\boldsymbol{\sigma})\sin\theta$, $\mathcal{E}[\rho]$ can be written as

$$\begin{aligned}\mathcal{E}[\rho] &= \frac{1}{2}\left[(\mathcal{I}\cos\theta + i\mathbf{n}\cdot\boldsymbol{\sigma}\sin\theta)\rho(\mathcal{I}\cos\theta - i\mathbf{n}\cdot\boldsymbol{\sigma}\sin\theta)\right.\\ &\quad\left.+(\mathcal{I}\cos\theta - i\mathbf{n}\cdot\boldsymbol{\sigma}\sin\theta)\rho(\mathcal{I}\cos\theta + i\mathbf{n}\cdot\boldsymbol{\sigma}\sin\theta)\right]\\ &= \cos^2\theta\rho + \sin^2\theta(\mathbf{n}\cdot\boldsymbol{\sigma})\rho(\mathbf{n}\cdot\boldsymbol{\sigma})\\ &= (1-\sin^2\theta)\rho + \sin^2\theta(\mathbf{n}\cdot\boldsymbol{\sigma})\rho(\mathbf{n}\cdot\boldsymbol{\sigma}).\end{aligned} \tag{C.3}$$

This form of $\mathcal{E}$ makes it clear the action of this channel is to dephase ρ about the axis $\mathbf{n}\cdot\boldsymbol{\sigma}$, an operation that generally decreases its purity. However, if ρ is a state whose Bloch vector lies along $\mathbf{n}$, the purity is *unchanged*:

$$\rho = \frac{1}{2}(\mathcal{I} + k\mathbf{n}\cdot\boldsymbol{\sigma}),\ k\in[-1,1] \implies \mathcal{E}[\rho] = \rho. \tag{C.4}$$

Another way to see this is to directly calculate $\mathrm{Tr}[\mathcal{E}^2]$:

$$\mathrm{Tr}[\mathcal{E}^2] = \frac{1}{2}\left(\mathrm{Tr}[\rho^2] + \mathrm{Tr}\left[U^2\rho(U^\dagger)^2\rho\right]\right). \tag{C.5}$$

The second term can be expressed as the Hilbert-Schmidt inner product between $U^2\rho(U^\dagger)^2$ and ρ; applying the Cauchy-Schwarz inequality, it follows that

$$\mathrm{Tr}\left[U^2\rho(U^\dagger)^2\rho\right] \le \mathrm{Tr}[\rho^2], \tag{C.6}$$

which implies

$$\mathrm{Tr}[\mathcal{E}^2] \le \mathrm{Tr}[\rho^2]. \tag{C.7}$$

$\mathcal{E}$ is purity-preserving if and only if the Cauchy-Schwarz inequality saturates, which happens if and only if

$$U^2\rho(U^\dagger)^2 = k\rho. \tag{C.8}$$

That is, the two matrices $U^2\rho(U^\dagger)^2$ and ρ are parallel when thought of as vectors in Hilbert-Schmidt space. Taking the trace of both sides, it follows that k must be

equal to 1. Further, pre-multiplying by $(U^\dagger)^2$, we find that the inequality saturates if and only if $[\rho, (U^\dagger)^2] = 0$. That is, if ρ is an eigenstate of U^2.

Thus, if the input state *is not* an eigenstate of U^2, then the channel $\mathcal{E}$ in Equation (C.2) does not preserve its purity.

C.2 Numerically simulating purely coherent or purely stochastic noise on single qubits

There are at least two ways to simulate the evolution of a quantum system. The first is to simulate the time dynamics of the system, and the second, the quantum channels generated by those dynamics. This is most clearly illustrated in the case of simulating unitary evolution of a pure quantum state. The Schrödinger equation gives the evolution of a state vector $|\psi\rangle$

$$|\dot{\psi}\rangle = -(i/\hbar)H(t)\,|\psi\rangle\,. \tag{C.9}$$

An entirely equivalent description of the evolution is $|\psi(t)\rangle = U(t,t')\,|\psi(t'\rangle$, where $U(t,t')$ is the unitary operator generated by time-order-integrating the Hamiltonian:

$$U(t,t') = \mathcal{T}\left[\mathrm{Exp}\left(-\frac{i}{\hbar}\left(\int_{t'}^{t} H(t)\;dt\right)\right)\right]. \tag{C.10}$$

The time dynamics description focuses on the instantaneous evolution of the state, while the channel description focuses on the operator that results when the state evolves for a time $t - t'$.

Describing how to simulate arbitrary realizations of coherent and stochastic noise, is easier using a time dynamics description. Both coherent and stochastic noise (as defined in Section 5.3.1) are *Markovian*. The most general Markovian, continuous-time dynamics of a d-dimensional quantum system is described by the Lindblad

master equation (48; 194; 120):

$$\dot{\rho} = -\frac{i}{\hbar}[H(t), \rho] + \sum_{j,k=1}^{d^2-1} h_{jk}(t) \left[A_j(t) \rho A_k^\dagger(t) - \frac{1}{2}\{A_k^\dagger(t) A_j(t), \rho\} \right], \tag{C.11}$$

where $H(t)$ is the Hamiltonian, and $\{A_j(t)\}$ are the Lindblad jump operators. The first term in the Lindblad equation generates unitary dynamics, while the second generates noisy dynamics such as dephasing, amplitude damping, or bit-flip noise.

In simulations, I make a simplification to Equation (C.11) by assuming the operators $H, \{A_j\}$ are all *time-independent*. In this simplification, we imagine that *regardless* of when the Hamiltonian that generates an ideal gate is (instantaneously) turned on in the course of running a circit, *the exact same set of noise operators are turned on at the same time*. Under this simplification, Equation (C.11) is *time-homogenous*:

$$\dot{\rho} = -\frac{i}{\hbar}[H, \rho] + \sum_{j,k=1}^{d^2-1} h_{jk} \left[A_j \rho A_k^\dagger - \frac{1}{2}\{A_k^\dagger A_j, \rho\} \right], \tag{C.12}$$

Just as evolution by unitary operators is an equivalent representation of Schrödinger evolution, *Lindbladians* provide an equivalent operator-based representation to time evolution under the Lindblad equation. For the time-homogenous Lindblad equation in Equation (C.12), the evolution of ρ can be written as $\rho(t) = e^{\mathcal{L}t}[\rho(0)]$. This follows from the fact that the quantum channels generated by the Lindblad equation form a quantum dynamical semigroup. In the circuit model, the updates are discrete (from one timestep of the circuit to the next), so we can take $t = 1$, and the corresponding quantum channel is $\mathcal{E}[\rho] = e^{\mathcal{L}}[\rho]$.

For purely coherent noise, the dynamics generated by the Lindblad equation must be purely unitary, so the jump coefficients h_{jk} are all 0:

$$\dot{\rho} = -\frac{i}{\hbar}[H, \rho]. \tag{C.13}$$

To simulate coherent noise, we write $H = H_0 + H_e$, where H_0 is the Hamiltonian of the ideal gate G_0, and H_e is the *error Hamiltonian*. In terms of the 4×4 generators of the dynamics the noisy channel is

$$\mathcal{E}[\rho] = e^{\mathcal{H}_0 + \mathcal{H}_e}[\rho], \tag{C.14}$$

where $\mathcal{H}[\rho] = -i[H, \rho]/\hbar$.

Each parameter in H_0 is usually associated with some external classical control field:

$$H_0 = \sum_{j=X,Y,Z} c_j \sigma_j, \tag{C.15}$$

where the coefficient c_j controls the evolution of the qubit about the σ_j axis. Thus, a natural choice for H_e is to draw it from the Gaussian unitary ensemble:

$$H_e = a\sigma_X + b\sigma_Y + c\sigma_Z \qquad a, b, c \sim \mathcal{N}(0, \eta^2). \tag{C.16}$$

For this model of purely coherent noise, each control c_j is subject to i.i.d. noise with mean zero, variance η^2, and *is constant in time*. (Recall that I use a time-homogenous version of the Lindblad equation.) Of course, it's possible that the controls will experience noise with the same functional form (Gaussian), but with different variance. All that would be necessary in that case is to then keep use a different value of η for the coefficient of each Pauli matrix.

For purely stochastic noise, there should be no unitary dynamics except the action of H_0:

$$\dot{\rho} = -\frac{i}{\hbar}[H_0, \rho] + \sum_{j,k=1}^{3} h_{jk} \left[A_j \rho A_k^\dagger + \frac{1}{2}\{A_k^\dagger A_j, \rho\} \right]. \tag{C.17}$$

Defining a superoperator $\mathcal{S}$ as

$$\mathcal{S}[\rho] = \sum_{j,k=1}^{3} h_{jk} \left[A_j \rho A_k^\dagger + \frac{1}{2}\{A_k^\dagger A_j, \rho\} \right], \tag{C.18}$$

the noisy channel can be written as

$$\mathcal{E} = e^{\mathcal{H}_0+\mathcal{S}}[\rho]. \tag{C.19}$$

The jump operators are taken to be the Pauli matrices: $A_j = \sigma_j$. To simulate this noise, I generate the coefficient matrix h as

$$\begin{aligned} h_{jk} &= (S^{-1}DS)_{jk} \\ D &= \text{diag}(a, b, c) \\ a, b, c &\sim |\mathcal{N}(0, \eta^2)|, \end{aligned} \tag{C.20}$$

where the columns of S form a randomly-chosen basis for $\mathbb{R}^3$. The variables a, b, c are drawn from a *folded normal* distribution.

A particular randomly-generated value for H_e or the coefficient matrix h is called a *realization* of the corresponding noise type. Again, because we are using a time-homogenous master equation, once a realization has been generated, then any time G_0 shows up in a given circuit, it is replaced by the *same* noisy version. Commonly when simulating these noise types, different realizations of the noise are generated *each time* the ideal gate G_0 occurs in a given quantum circuit. I do not use this approach.

For both coherent and stochastic noise, as $\eta \to 0$ both H_e and h go to zero. (That is, at $\eta = 0$, the channel is noiseless.) The role of η can also be formalized as follows. For purely coherent noise, notice that the average over all realizations of the noise of the Hamiltonian H is H_0:

$$\langle H \rangle = H_0 + \langle a \rangle \sigma_X + \langle b \rangle \sigma_Y + \langle c \rangle \sigma_Z = H_0. \tag{C.21}$$

On average, the Hamiltonian generates correct evolution. Next, consider $\langle H^2 \rangle$:

$$\begin{aligned} \langle H^2 \rangle &= H_0^2 + H_0 \langle H_e \rangle + \langle H_e \rangle H_0 + \langle H_e^2 \rangle \\ &= H_0^2 + 3\eta^2 \mathcal{I}, \end{aligned} \tag{C.22}$$

because the noise realizations a, b, c are i.i.d. random variables. Therefore, the variance $\Delta H = \langle H^2 \rangle - \langle H \rangle^2$ is $3\eta^2\mathcal{I}$. As $\eta \to 0$, H more tightly concentrates around its expected value, H_0.

For purely stochastic noise, consider the expected value of the coefficients h_{jk} in Equation (C.17). To compute $\langle h_{jk} \rangle$, it's important to note that the randomly-chosen basis for $\mathbb{R}^3$ is independent of the random variables a, b, c:

$$\begin{aligned} \langle h_{jk} \rangle &= \langle (S^{-1}DS)_{jk} \rangle = \sum_{qr} \langle (S^{-1})_{jq} D_{qr} S_{rk} \rangle \\ &= \sum_{r} \langle (S^{-1})_{jq} S_{rk} \rangle \langle D_{rr} \rangle \propto \eta. \end{aligned} \tag{C.23}$$

The expected value of h_{jk}^2 is

$$\begin{aligned} \langle h_{jk}^2 \rangle &= \left\langle \sum_{qr} (S^{-1})_{jq} D_{qr} S_{rk} \sum_{q'r'} (S^{-1})_{jq'} D_{q'r'} S_{r'k} \right\rangle \\ &= \left\langle \sum_{qq'rr'} (S^{-1})_{jq} S_{rk} (S^{-1})_{jq'} S_{r'k} D_{qr} D_{q'r'} \right\rangle \\ &= \sum_{qq'} \langle (S^{-1})_{jq} S_{qk} (S^{-1})_{jq'} S_{q'k} \rangle \langle D_{qq} D_{q'q'} \rangle \\ &= \sum_{q \neq q'} \langle (S^{-1})_{jq} S_{qk} (S^{-1})_{jq'} S_{q'k} \rangle \langle D_{qq} \rangle \langle D_{q'q'} \rangle \\ &+ \sum_{q=q'} \langle ([S^{-1}]_{jq} S_{qk}]^2 \rangle \langle D_{qq}^2 \rangle \propto \eta^2. \end{aligned} \tag{C.24}$$

Similar to the case of purely coherent noise, as $\eta \to 0$, the noise terms more and more tightly concentrate around 0.

Because η^2 controls how the noisy channel deviates from the ideal one, I say η relates to the "strength" of the noise.

C.3 Amplitude damping noise

This section derives the Lindbladian for amplitude damping noise, introduced in Section 6.4.5. Recall that the channel generated by Lindbladian $\mathcal{L}$ is $\mathcal{E}[\rho] = e^{\mathcal{L}}[\rho]$. If the Lindbladian is small, the exponential can be expanded to first order, giving

$$\mathcal{E}[\rho] \approx \mathcal{I}[\rho] + \mathcal{L}[\rho], \tag{C.25}$$

which implies the Lindbladian can be computed by taking the noisy channel, considering the small-noise limit, and then subtracting off the superoperator identify.

I'll start by considering the simplest model of amplitude damping, which is amplitude damping to the $|0\rangle$ state of the qubit at rate γ. This noise is described by the following Kraus operators (223, Section 8.3.5)

$$K_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix} \qquad K_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}, \tag{C.26}$$

and the action of the channel is

$$\mathcal{E}[\rho] = K_0 \rho K_0^\dagger + K_1 \rho K_1^\dagger. \tag{C.27}$$

To represent $\mathcal{E}$ as a 4×4 superoperator whose action is matrix multiplication on the vectorized form of ρ, apply the vec operation (see Equation (H.6)):

$$\begin{aligned} \text{vec}[\mathcal{E}] &= (K_0^\star \otimes K_0 + K_1^\star \otimes K_1) \\ &= \left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{1-\gamma} & 0 & 0 \\ 0 & 0 & \sqrt{1-\gamma} & 0 \\ 0 & 0 & 0 & 1-\gamma \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & \gamma \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \right) \\ &= \begin{pmatrix} 1 & 0 & 0 & \gamma \\ 0 & \sqrt{1-\gamma} & 0 & 0 \\ 0 & 0 & \sqrt{1-\gamma} & 0 \\ 0 & 0 & 0 & 1-\gamma \end{pmatrix}. \end{aligned} \tag{C.28}$$

To compute the Lindbladian, consider amplitude damping in the small-rate limit, so $\gamma << 1$. The noisy channel is then

$$\mathrm{vec}[\mathcal{E}] = \begin{pmatrix} 1 & 0 & 0 & \gamma \\ 0 & 1-\gamma/2 & 0 & 0 \\ 0 & 0 & 1-\gamma/2 & 0 \\ 0 & 0 & 0 & 1-\gamma \end{pmatrix}. \tag{C.29}$$

Therefore, the Lindbladian $\mathcal{L} = \mathcal{E} - \mathcal{I}$ is

$$\mathcal{L} = \begin{pmatrix} 0 & 0 & 0 & \gamma \\ 0 & -\gamma/2 & 0 & 0 \\ 0 & 0 & -\gamma/2 & 0 \\ 0 & 0 & 0 & -\gamma \end{pmatrix}. \tag{C.30}$$

A related representation can also be derived by considering how the channel acts on the operators $|j\rangle\langle k|$. In particular, this action can be used to define a representation according to

$$\mathcal{E}_{(jk),(lm)} = \mathrm{Tr}(|l\rangle\langle m|\mathcal{E}[|j\rangle\langle k|]) = \langle m|\mathcal{E}[|j\rangle\langle k|]|l\rangle. \tag{C.31}$$

The calculation of these matrix elements is expedited by the observation

$$K_0 = |0\rangle\langle 0| + \sqrt{1-\gamma}|1\rangle\langle 1| \quad K_1 = \sqrt{\gamma}|0\rangle\langle 1|, \tag{C.32}$$

so that

$$\begin{aligned} \mathcal{E}[|j\rangle\langle k|] &= \left(|0\rangle\langle 0| + \sqrt{1-\gamma}|1\rangle\langle 1|\right)|j\rangle\langle k|\left(|0\rangle\langle 0| + \sqrt{1-\gamma}|1\rangle\langle 1|\right) \\ &\quad + \gamma\delta_{j1}\delta_{k1}|0\rangle\langle 0| \\ &= \delta_{j0}\delta_{k0}|0\rangle\langle 0| + (1-\gamma)\delta_{j1}\delta_{k1}|1\rangle\langle 1| + \sqrt{1-\gamma}\delta_{j1}\delta_{k0}|1\rangle\langle 0| \\ &\quad + \sqrt{1-\gamma}\delta_{j0}\delta_{k1}|0\rangle\langle 1| + \gamma\delta_{j1}\delta_{k1}|0\rangle\langle 0|. \end{aligned} \tag{C.33}$$

Hence,

$$\begin{aligned} \mathcal{E}_{(jk),(lm)} &= \delta_{j0}\delta_{k0}\delta_{m0}\delta_{l0} + (1-\gamma)\delta_{j1}\delta_{k1}\delta_{l1}\delta_{m1} + \sqrt{1-\gamma}\delta_{j1}\delta_{k0}\delta_{m1}\delta_{l0} \\ &\quad + \sqrt{1-\gamma}\delta_{j0}\delta_{k1}\delta_{m0}\delta_{l1} + \gamma\delta_{j1}\delta_{k1}\delta_{l0}\delta_{m0}, \end{aligned} \tag{C.34}$$

or in matrix form

$$\mathcal{E} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{1-\gamma} & 0 & 0 \\ 0 & 0 & \sqrt{1-\gamma} & 0 \\ \gamma & 0 & 0 & 1-\gamma \end{pmatrix} \tag{C.35}$$

Compare this with Equation (C.28). The reason for the difference is that the vec() operator stacks ρ by its columns, while the representation used here goes across the *rows* of ρ. Therefore, the two representations are related by a transposition operation. Again, by considering the small-rate limit and subtracting off the superoperator identity $\mathcal{I}$, we arrive at the transposed form of Equation (C.30), as expected.

The *Pauli transfer matrix* representation is a matrix whose elements are given by $\mathcal{E}_{jk} = \mathrm{Tr}[b_j \mathcal{E}(b_k)]$, where $b_j = \sigma_j/\sqrt{2}$ and $\mathrm{Tr}[b_j b_k] = \delta_{jk}$. (See H.3 for details on this representation.) Explicitly computing these matrix elements gives

$$\begin{aligned} \mathcal{E} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{1-\gamma} & 0 & 0 \\ 0 & 0 & \sqrt{1-\gamma} & 0 \\ 0 & 0 & 0 & 1-\gamma \end{pmatrix} \\ &\underset{\gamma<<1}{\to} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-\gamma/2 & 0 & 0 \\ 0 & 0 & 1-\gamma/2 & 0 \\ 0 & 0 & 0 & 1-\gamma \end{pmatrix}. \end{aligned} \tag{C.36}$$

In this representation, it's clear that the action of $\mathcal{E}$ is to reduce the Bloch vector components r_X, r_Y, r_Z at two different rates: the rate for r_X, r_Y is $\gamma/2$, while the

rate for r_Z is γ. The Lindbladian $\mathcal{L}$ is given by

$$\mathcal{L} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -\gamma/2 & 0 & 0 \\ 0 & 0 & -\gamma/2 & 0 \\ 0 & 0 & 0 & -\gamma \end{pmatrix}. \tag{C.37}$$

Appendix D

Machine learning classifiers

This appendix describes in detail the classification algorithms discussed in the main text. In what follows, let $\mathcal{C} = \{\mathbf{f}_j, y_j\}$ be a collection of d-dimensional feature vectors $\mathbf{f}_j$ with associated class label $y_j \in \{\pm 1\}$.

D.1 Linear and quadratic discriminant analysis

Linear and quadratic discriminant analysis (LDA and QDA, respectively) are well-motivated from a statistics viewpoint, as they use Bayes factors to determine a decision rule. The basic ideas of LDA and QDA originated with the work of Fisher in the mid-1930's (104). Both techniques assume that the feature vectors belonging to class k are $\mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)$ random variables.

For the case of binary classification, there are two hypotheses to consider when classifying an unlabeled feature vector $\mathbf{f}$:

$$\begin{aligned} H_1 &: \mathbf{f} \text{ belongs in class 1} \\ H_2 &: \mathbf{f} \text{ belongs in class 2.} \end{aligned} \tag{D.1}$$

The Bayes factor K comparing these two hypotheses is

$$K = \frac{\Pr(\mathbf{f}|\text{class 1})}{\Pr(\mathbf{f}|\text{class 2})}. \tag{D.2}$$

Notice that $K > 1$ implies that $\mathbf{f}$ is more probable under the assumption it is drawn from class 1, and if $K < 1$, then it is more probable under the assumption it is drawn from class 2. Therefore, a reasonable classification rule is to compare the value of K to 1. Or equivalently, $\log(K)$ can be compared to 0. Under the assumption the feature vectors are Gaussian-distributed,

$$\begin{aligned} K &= \left(\frac{|\Sigma_2|}{|\Sigma_1|}\right)^{1/2} \frac{\text{Exp}\left[-(\mathbf{f}-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\mathbf{f}-\boldsymbol{\mu}_1)/2\right]}{\text{Exp}\left[-(\mathbf{f}-\boldsymbol{\mu}_2)^T\Sigma_2^{-1}(\mathbf{f}-\boldsymbol{\mu}_2)/2\right]} \\ \implies \log(K) &= \frac{1}{2}\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) + \frac{1}{2}(\mathbf{f}-\boldsymbol{\mu}_2)^T\Sigma_2^{-1}(\mathbf{f}-\boldsymbol{\mu}_2) \\ &\quad - \frac{1}{2}(\mathbf{f}-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\mathbf{f}-\boldsymbol{\mu}_1). \end{aligned} \tag{D.3}$$

Generally, the means and covariances of the distributions are not *a priori* known. Therefore, it is necessary to estimate them from $\mathcal{C}$.

LDA estimates the mean for each class separately, but assumes the covariance is the same for each:

$$\begin{aligned} &\Sigma_1, \Sigma_2 \to \hat{\Sigma} \\ &\quad \boldsymbol{\mu}_1 \to \hat{\boldsymbol{\mu}}_1 \\ &\quad \boldsymbol{\mu}_2 \to \hat{\boldsymbol{\mu}}_2. \end{aligned} \tag{D.4}$$

Plugging these values into Equation (D.3) yields the following decision rule:

$$\begin{aligned} c_{\text{LDA}}(\mathbf{f}) &= \text{sign}\Big[\mathbf{f}^T\hat{\Sigma}^{-1}(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_2) \\ &\qquad + (\hat{\boldsymbol{\mu}}_1^T\hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_2^T\hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_2)/2\Big] \\ &= \text{sign}\left[\boldsymbol{\beta}\cdot\mathbf{f} + \beta_0\right], \end{aligned} \tag{D.5}$$

where $\boldsymbol{\beta} = \hat{\Sigma}^{-1}(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_2)$ and $\beta_0 = (\hat{\boldsymbol{\mu}}_1^T\hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_2^T\hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_2)/2$. The decision rule is *linear* in $\mathbf{f}$, which gives rise to the name "linear" discriminant analysis.

QDA estimates the means and covariances for each class separately:

$$\begin{aligned}&\Sigma_1 \to \hat{\Sigma}_1 \ , \ \boldsymbol{\mu}_1 \to \hat{\boldsymbol{\mu}}_1\\ &\Sigma_2 \to, \hat{\Sigma}_2 \ , \ \boldsymbol{\mu}_2 \to \hat{\boldsymbol{\mu}}_2,\end{aligned} \tag{D.6}$$

although in practice usually each estimate $\hat{\Sigma}_k$ is regularized: $\hat{\Sigma}_k \to (1-s)\hat{\Sigma}_k + s\mathcal{I}$, with $\mathcal{I}$ as the $d \times d$ identity matrix. The resulting classification rule is a *quadratic* function of $\mathbf{f}$:

$$\begin{aligned}c_{\mathrm{QDA}}(\mathbf{f}) = \mathrm{sign}\Bigg[&\frac{1}{2}\log\left(\frac{|\hat{\Sigma}_2|}{|\hat{\Sigma}_1|}\right) + \frac{1}{2}(\mathbf{f}-\hat{\boldsymbol{\mu}}_2)^T\hat{\Sigma}_2^{-1}(\mathbf{f}-\hat{\boldsymbol{\mu}}_2)\\ &- \frac{1}{2}(\mathbf{f}-\hat{\boldsymbol{\mu}}_1)^T\hat{\Sigma}_1^{-1}(\mathbf{f}-\hat{\boldsymbol{\mu}}_1)\Bigg].\end{aligned} \tag{D.7}$$

While QDA and LDA are well-motivated from a statistical standpoint, they require strong assumptions on the feature vectors (such as being Gaussian-distributed) in order to be useful in practice. For this reason, algorithms which relax those assumptions are also useful for binary classification, such as the perceptron.

D.2 Perceptrons

Similar to LDA, the *perceptron* learns a separating hyperplane $H = (\boldsymbol{\beta}, \beta_0)$, with an associated decision rule

$$c_{\mathrm{Perceptron}}(\mathbf{f}) = \mathrm{sign}\left(\boldsymbol{\beta}\cdot\mathbf{f} + \beta_0\right). \tag{D.8}$$

Conceptually, the perceptron is simple – it *is* the hyperplane, and the main challenge in using perceptrons is determining $\boldsymbol{\beta}$ and β_0. To do so, the perceptron is *trained* using $\mathcal{C}$. Training the perceptron involves minimizing a *loss function*, L. This loss function depends on the hyperplane (i.e., the values of $\boldsymbol{\beta}$ and β_0). A natural choice for L is

$$L(\boldsymbol{\beta}, \beta_0) = -\sum_{\mathbf{f}_j} y_j(\boldsymbol{\beta}\cdot\mathbf{f}_j + \beta_0). \tag{D.9}$$

If y_j and $(\boldsymbol{\beta} \cdot \mathbf{f}_j + \beta_0)$ have the opposite sign (i.e., $\mathbf{f}$ is misclassified by H), then the corresponding term adds to the loss. On the other hand, if they have the same sign (so $\mathbf{f}$ is correctly classified by H) the corresponding term subtracts from the loss. Therefore, all of terms which increase the loss are those which correspond to feature vectors that are misclassified. Without loss of generality then, to minimize the loss it suffices to minimize its value on the set of misclassified points. Let $\mathcal{M}$ denote the set of misclassified points given particular values of $\boldsymbol{\beta}$ and β_0. Given $\mathbf{f}_j \in \mathcal{M}$ with label y_j, the gradients of the loss with respect to the parameters of the perceptron are

$$\frac{\partial L}{\partial \beta_0} = -y_j \tag{D.10}$$

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = -y_j \mathbf{f}_j. \tag{D.11}$$

These observations suggest the following training algorithm:

1. Choose initial values for $\boldsymbol{\beta}$ and β_0.
2. Compute the set of misclassified points $\mathcal{M}$.
3. For each point in $\mathcal{M}$, update the parameters of the hyperplane:

$$\boldsymbol{\beta} \to \boldsymbol{\beta} + \rho y_j \mathbf{f}_j \tag{D.12}$$

$$\beta_0 \to \beta_0 + \rho y_j, \tag{D.13}$$

   where ρ is the *learning rate* of the algorithm. (To descend down the loss function, the steps need to take in a direction *opposite* the direction of the gradient.)
4. Re-compute the set of misclassified points $\mathcal{M}$.

This training algorithm is a stochastic gradient descent with a batch size of 1 (i.e., an "online" training algorithm). It has the property that if a separating hyperplane exists, the algorithm is guaranteed to converge to it in a finite number of steps.

Commonly though, the training is stopped after steps 2 through 4 have been done N_{epochs} times, or when the error $\sum_{j=1}(c(\mathbf{f}_j) - y_j) \leq t$ for some tolerance t. It is known that if $\mathcal{C}$ is linearly separable, then provided N_{epochs} is sufficiently large, the perceptron will learn a separating hyperplane (254). However, determining N_{epochs} *a priori* is difficult; Novikoff showed that the number of epochs necessary to achieve convergence *under the assumption* $\mathcal{C}$ *is linearly separable by a hyperplane* H *with margin* M_H goes as $(R/M_H)^2$, where $||\mathbf{f}_j|| < R \; \forall \; j$ (225). If M_H is small, there's very little space separating the two classes – a separating hyperplane with that margin has to "thread the needle" between the two very carefully, resulting in a large number of misclassified points during training, and requiring many epochs before the perceptron learns it.

While the perceptron is appealing from the perspective that it can learn a separating hyperplane should one exist, its main drawback is that the number of iterations necessary for convergence may be prohibitive. What's more, the perceptron may find different hyperplanes depending on its initial parameters or the first feature vector used for updating them. For this reason, the *support vector machine* is useful, as it remedies some of the deficiencies of perceptrons by imposing the constraint that the hyperplane should maximize the geometric margin, thereby singling out a unique separating hyperplane.

D.3 Support vector machines (SVMs)

Like perceptrons, an SVM learns a separating hyperplane for the data. Unlike perceptrons, SVMs learn a hyperplane which maximizes the margin. That is, of all the hyperplanes that could separate the data, the one learned by the SVM is the farthest away from the points closest to it. For this reason, SVMs were also known as "perceptrons of optimal stability", because small perturbations of the data set do not generally result in dramatically different hyperplanes learned by the SVM, while

this is not the case for perceptrons. (For a detailed overview of SVMs, see (74).)

D.3.1 Hard-margin SVM

Under the assumption $\mathcal{C}$ is linearly separable, it is straightforward to derive the conditions that an optimal hyperplane should satisfy. Recall the signed, functional distance from a point $\mathbf{f}$ to a hyperplane $H = (\hat{\boldsymbol{\beta}}, \beta_0)$ is given by $\hat{\boldsymbol{\beta}} \cdot \mathbf{f} + \beta_0$. The *functional* margin of this hyperplane, M, is given by

$$M = \min_{\mathbf{f}_j} y_j(\hat{\boldsymbol{\beta}} \cdot \mathbf{f}_j + \beta_0). \tag{D.14}$$

Therefore, the hyperplane that maximizes the functional margin is the solution to the following optimization problem:

$$\begin{aligned} &\max_{\hat{\boldsymbol{\beta}}, \beta_0} M \\ &\text{s.t. } y_j(\hat{\boldsymbol{\beta}} \cdot \mathbf{f}_j + \beta_0) \geq M \ \forall \ j. \end{aligned} \tag{D.15}$$

Re-writing $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}/||\boldsymbol{\beta}||$ and re-defining β_0, the optimization problem can be re-written as

$$\begin{aligned} &\max_{\boldsymbol{\beta}, \beta_0} M \\ &\text{s.t. } y_j(\boldsymbol{\beta} \cdot \mathbf{f}_j + \beta_0) \geq M||\boldsymbol{\beta}|| \ \forall \ j. \end{aligned} \tag{D.16}$$

In this form, it is clear that if $(\boldsymbol{\beta}, \beta_0)$ is the solution to (D.16), then so is $(\alpha\boldsymbol{\beta}, \alpha\beta_0)$ for any $\alpha > 0$. It then follows that the *norm* of $\boldsymbol{\beta}$ is arbitrary. Thus, we can arbitrarily set $||\boldsymbol{\beta}|| = 1/M$. Maximizing the functional margin is then equivalent to minimizing the norm of $\boldsymbol{\beta}$. Because the minimizer of $||\boldsymbol{\beta}||$ also minimizes $||\boldsymbol{\beta}||^2$, we can re-formulate (D.16) as a simple convex optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{\beta}, \beta_0} \frac{1}{2}||\boldsymbol{\beta}||^2 \\ &\text{s.t. } y_j(\boldsymbol{\beta} \cdot \mathbf{f}_j + \beta_0) \geq 1 \ \forall \ j. \end{aligned} \tag{D.17}$$

In this formulation, the two classes are separated by a "slab" of thickness $2/||\boldsymbol{\beta}||$.

The Lagrangian dual of this problem is given by (See (140))

$$\max_{\boldsymbol{\alpha}} \sum_j \boldsymbol{\alpha}_j - \frac{1}{2}\sum_{j,k} \boldsymbol{\alpha}_j \boldsymbol{\alpha}_k y_j y_k (\mathbf{f}_j \cdot \mathbf{f}_k) \quad \text{s.t. } \boldsymbol{\alpha}_j \geq 0 \; \forall \; j. \tag{D.18}$$

By solving the problem in its dual form, the Krush-Kuhn-Tucker (KKT) conditions imply that the optimal solution $\boldsymbol{\alpha}$ satisfies

$$\boldsymbol{\alpha}_j [y_j(\boldsymbol{\beta} \cdot \mathbf{f}_j + \beta_0) - 1] = 0 \; \forall \; j. \tag{D.19}$$

Importantly, the KKT conditions imply that if $\boldsymbol{\alpha}_j \neq 0$, then the signed, functional distance $y_j(\boldsymbol{\beta} \cdot \mathbf{f}_j + \beta_0)$ must be equal to 1. Examining Equation (D.17), this implies $\mathbf{f}_j$ lies exactly on the boundary of the slab. For this reason, $\mathbf{f}_j$ is said to "support" the hyperplane, and is called a "support vector". On the other hand, if $y_j(\boldsymbol{\beta} \cdot \mathbf{f}_j + \beta_0) > 1$ (i.e. the point is located beyond the boundary of the slab), then $\boldsymbol{\alpha}_j = 0$.

In the dual form, it can also be shown that

$$\boldsymbol{\beta} = \sum_j \boldsymbol{\alpha}_j y_j \mathbf{f}_j, \tag{D.20}$$

(i.e., $\boldsymbol{\beta}$ depends on the feature vectors in $\mathcal{C}$), and that

$$\beta_0 = 1/y_j - \boldsymbol{\beta} \cdot \mathbf{f}_j, \tag{D.21}$$

where $\mathbf{f}_j$ is any support vector.

The geometric margin for the hard-margin SVM is given by

$$M_H = \frac{1}{||\boldsymbol{\beta}||} \min_{\mathbf{f}_j} |(\boldsymbol{\beta} \cdot \mathbf{f}_j + \beta_0)| = \frac{1}{||\boldsymbol{\beta}||} = M, \tag{D.22}$$

because the minimum is 1, and is attained by any support vector. ***Therefore, for a hard-margin SVM, the geometric margin and the functional margin are the same, and given by*** $1/||\boldsymbol{\beta}||$**.** (As we will see later, this is *not* the case for soft-margin SVMs.)

D.3.2 Soft-margin SVM

In the event $\mathcal{C}$ is not linearly separable, it still makes sense to speak of identifying an optimal hyperplane. However, this hyperplane may not separate the data exactly – some points may be misclassified. To accommodate this possibility, the *soft-margin* SVM (71) allows for some data points to be on the wrong side of the $y_j(\boldsymbol{\beta} \cdot \mathbf{f}_j + \beta_0) = 1$ boundary. This is done by returning to Equation (D.16) and introducing *slack variables* ξ_j, so that the feasibility condition is

$$y_j(\hat{\boldsymbol{\beta}} \cdot \mathbf{f}_j + \beta_0) \geq M(1 - \xi_j) \; \forall \; j. \tag{D.23}$$

Notice that ξ_j should be greater than or equal to zero – if it were negative, this would impose a more stringent condition than that for the hard-margin SVM! Further, if $\xi_j > 1$, then the right-hand side is negative, and the feasibility condition implies that y_j and $(\hat{\boldsymbol{\beta}} \cdot \mathbf{f}_j + \beta_0)$ have the opposite sign, implying that $\mathbf{f}_j$ is mis-classified. To minimize the number of mis-classified points, the total sum of the slack variables should be bounded: $\sum_j \xi_j \leq m$.

Any scalar multiple of $\boldsymbol{\beta}$ and β_0 will satisfy the feasibility conditions, so M again can be set to $1/||\boldsymbol{\beta}||$, yielding

$$\begin{aligned} &\min_{\boldsymbol{\beta}, \beta_0} \frac{1}{2}||\boldsymbol{\beta}||^2 \\ &\text{s.t. } y_j(\boldsymbol{\beta} \cdot \mathbf{f}_j + \beta_0) \geq 1 - \xi_j \; \forall \; j \\ &\qquad \xi_j \geq 0 \; \forall \; j \text{ and } \sum_j \xi_j \leq m. \end{aligned} \tag{D.24}$$

Finally, the constraint $\sum_j \xi_j \leq m$ can be embedded into the objective function itself through the use of a Lagrange multiplier C. This observation gives rise to the

standard form of the soft-margin SVM:

$$\begin{aligned} &\min_{\boldsymbol{\beta},\beta_0} \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_j \xi_j \\ &\text{s.t. } y_j(\boldsymbol{\beta}\cdot\mathbf{f}_j + \beta_0) \geq 1 - \xi_j \;\forall\; j \\ &\xi_j \geq 0 \;\forall\; j. \end{aligned} \tag{D.25}$$

The Lagrangian dual of Equation (D.25) is

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \sum_j \boldsymbol{\alpha}_j - \frac{1}{2}\sum_{j,k} \boldsymbol{\alpha}_j\boldsymbol{\alpha}_k y_j y_k (\mathbf{f}_j\cdot\mathbf{f}_k) \\ &\text{s.t. } 0 \leq \boldsymbol{\alpha}_j \leq C \;\forall\; j. \end{aligned} \tag{D.26}$$

In typical use, a soft-margin SVM may be used in place of a hard-margin SVM even if $\mathcal{C}$ is linearly separable. If $C \neq \infty$, then the geometric and functional margins need not be the same:

$$M_H = \frac{1}{||\boldsymbol{\beta}||}\min_{\mathbf{f}_j} |\boldsymbol{\beta}\cdot\mathbf{f}_j + \beta_0| = \frac{1}{||\boldsymbol{\beta}||}\min_j |1 - \xi_j|. \tag{D.27}$$

Because $\mathcal{C}$ is separable, no point should be misclassified, so $0 \leq \xi_j \leq 1$. Assuming H doesn't mis-classify a point, it follows that $M_H = 1/||\boldsymbol{\beta}||$ if and only if $\xi_j = 0 \;\forall\; j$ (i.e., the slack variables are all 0, and the SVM has actually identified a hard-margin hyperplane). In general then, $M_H \leq 1/||\boldsymbol{\beta}||$ for a soft-margin SVM that was trained using a linearly separable data set.

D.3.3 Kernel methods

The hard and soft-margin SVMs presented in the preceding sections use linear decision boundaries to separate $\mathcal{C}$. However, there are data sets that are separable, but not with a linear decision boundary (e.g., two concentric circles or spheres). A common way to do so is to map the data into a new feature space so that in this new feature space a linear decision boundary can be found. Let ϕ denote a (nonlinear)

map acting on $\mathcal{C}$:

$$\phi : \mathcal{C} \to \mathcal{C}' = \{(\phi(\mathbf{f}_j), y_j)\}. \tag{D.28}$$

Ideally, ϕ maps $\mathcal{C}$ in such a way that a linear decision boundary can be learned. That is, $\exists H = (\boldsymbol{\beta}, \beta_0)$ s.t. $\text{sign}(\boldsymbol{\beta} \cdot \phi(\mathbf{f}_j) + \beta_0) = y_j \ \forall \ j$.

Examining the feasibility conditions of the optimization problems defining the hard and soft-margin SVM, it is clear that the feature vectors $\mathbf{f}_j$ are "carried along" in defining the problem, so to speak. Without loss of generality, the optimization problem defining the optimal separating hyperplane in the new feature space has a Lagrangian dual given by

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \ & \sum_j \boldsymbol{\alpha}_j - \frac{1}{2} \sum_{j,k} \boldsymbol{\alpha}_j \boldsymbol{\alpha}_k y_j y_k (\phi(\mathbf{f}_j) \cdot \phi(\mathbf{f}_k)) \\ & \text{s.t. } 0 \leq \boldsymbol{\alpha}_j \leq C \ \forall \ j. \end{aligned} \tag{D.29}$$

In practice, coming up with the map ϕ is difficult. For this reason, "kernel methods" (42) are used to completely bypass the need to explicitly construct ϕ. In Equation (D.29), notice that the way in which ϕ appears is through the dot products $\phi(\mathbf{f}_j) \cdot \phi(\mathbf{f}_k)$. Thus, it's natural to associate a *kernel* with ϕ, denoted K_ϕ, as $K_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$.

It's clear that to every map ϕ there is an associated kernel K_ϕ. But what of the converse – given a map $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, is there a map ϕ such that $K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$? As long as K satisfies the *Mercer condition* $K \geq 0$, then the answer is yes.

Thus for every K satisfying the Mercer condition, an "SVM with kernel K" is defined

by the optimization problem (71)

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \sum_j \boldsymbol{\alpha}_j - \frac{1}{2}\sum_{j,k} \boldsymbol{\alpha}_j \boldsymbol{\alpha}_k y_j y_k K(\mathbf{x}_j, \mathbf{x}_k) \\ \text{s.t. } 0 \leq \boldsymbol{\alpha}_j \leq C \;\forall\; j, \\ \sum_j y_j \boldsymbol{\alpha}_j = 0. \end{aligned} \tag{D.30}$$

Equation (D.30) defines a quadratic program with linear equality and inequality constraints, and efficient algorithms exist for finding a solution (224).

Letting $\boldsymbol{\alpha}$ denote the solution, the decision rule for the SVM on any data point $\mathbf{f}$ is simple:

$$c_{\text{SVM}}(\mathbf{f}) = \text{sign}\left[\sum_{j=1}^{N} y_j \boldsymbol{\alpha}_j K(\mathbf{f}_j, \mathbf{f}) + \beta_0\right]. \tag{D.31}$$

As seen in the previous subsections, if $\boldsymbol{\alpha}_j \neq 0$, the corresponding feature vector $\mathbf{f}_j$ is said to be a support vector. Using any support vector $\mathbf{f}_k$, the bias β_0 can be computed as

$$\begin{aligned} &\sum_{j=1}^{N} y_j \mathbf{c}_j K(\mathbf{f}_j, \mathbf{f}_k) + \beta_0 = y_k \\ \implies &\beta_0 = y_k - \sum_{j=1}^{N} y_j \mathbf{c}_j K(\mathbf{f}_j, \mathbf{f}_k). \end{aligned} \tag{D.32}$$

For the *radial basis function (RBF) SVM*, $K(\mathbf{x}, \mathbf{y}) = \text{Exp}\left[-\gamma||\mathbf{x} - \mathbf{y}||^2\right]$. We can recover the linear SVM by taking $K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$.

Appendix E

Testing for linear separability as a linear programming problem

There is a straightforward test for determining whether a given data set is linearly separable using *linear programming*. Various formulations of this problem have been put forth (263; 23). Our formulation is simple – if the data is not linearly separable, the corresponding linear programming problem *should not* be feasible. The derivation below closely follows the line of reasoning presented in (311).

Consider two data sets $A = \{\mathbf{a}_1, \cdots \mathbf{a}_n\}$ and $B = \{\mathbf{b}_1, \cdots \mathbf{b}_m\}$ where $\mathbf{a}_j, \mathbf{b}_k \in \mathbb{R}^d$. If A and B are linearly separable, there exists a hyperplane $H = (\boldsymbol{\beta}, \beta_0)$ such that

$$\boldsymbol{\beta} \cdot \mathbf{a}_j > \beta_0, \;\; \boldsymbol{\beta} \cdot \mathbf{b}_k < \beta_0 \;\; \forall \; j, k. \tag{E.1}$$

This decision rule implies that the label for A is 1, while the label for B is -1, as the decision rule implied by this hyperplane is $\mathbf{f} \rightarrow \text{sign}\,[\boldsymbol{\beta} \cdot \mathbf{f} - \beta_0]$.

Multiplying the first inequality by -1 flips the sign of the inequality, resulting in the

following conditions:

$$\begin{aligned} -\boldsymbol{\beta} \cdot \mathbf{a}_j + \beta_0 &< 0 \\ \boldsymbol{\beta} \cdot \mathbf{b}_k - \beta_0 &< 0 \quad \forall\ j, k. \end{aligned} \tag{E.2}$$

These two inequalities can be formulated as a single matrix inequality by defining a new vector $\tilde{\boldsymbol{\beta}} = (\boldsymbol{\beta}, \beta_0)$ and new feature vectors $\mathbf{a}_j \to (-\mathbf{a}_j, 1)$ and $\mathbf{b}_k \to (\mathbf{b}_k, -1)$. In terms of these variables, the inequality constraints become

$$\begin{pmatrix} -\mathbf{a}_1^T, 1 \\ -\mathbf{a}_2^T, 1 \\ \vdots \\ -\mathbf{a}_n^T, 1 \\ \mathbf{b}_1^T, -1 \\ \vdots \\ \mathbf{b}_m^T, -1 \end{pmatrix} \tilde{\boldsymbol{\beta}} < \mathbf{0}. \tag{E.3}$$

Defining D to be the $(n+m) \times (d+1)$ matrix above, it follows that **if A and B are linearly separable, there exists a $\tilde{\boldsymbol{\beta}}$ such that $D\tilde{\boldsymbol{\beta}} < \mathbf{0}$**. Determining if any such $\tilde{\boldsymbol{\beta}}$ exists can be done by recasting the problem as a *linear programming* problem:

$$\begin{aligned} \tilde{\boldsymbol{\beta}} = \underset{\mathbf{x} \in \mathbb{R}^{N+1}}{\operatorname{argmin}} \ & \mathbf{0} \cdot \mathbf{x} \\ \text{s.t. } & D\mathbf{x} < \mathbf{0}. \end{aligned} \tag{E.4}$$

If A and B are linearly separable, there exists at least one feasible solution to this problem. If Equation (E.4) is feasible, then a convex optimization problem should be able to solve it. Suppose, though, that a solver *didn't* find a solution – how could we tell that this happened because the problem was in fact *infeasible*, and not because the solver terminated or crashed prematurely?

Thankfully, solving Equation (E.4) is simply solving a *strict* system of inequalities.

For any such problem, a theorem of *alternatives* (45, Example 2.21) gives the conditions under which it is feasible. Given a system of strict inequalities

$$A\mathbf{x} < \mathbf{b}, \tag{E.5}$$

the system is *infeasible* if, and only if, there exists $\mathbf{y}$ satisfying

$$\mathbf{y} \neq 0,\ \mathbf{y} > 0,\ A^T\mathbf{y} = 0,\ \mathbf{y} \cdot \mathbf{b} \leq 0. \tag{E.6}$$

Thus, suppose that in the course of solving Equation (E.4), a solver doesn't find a solution. We can easily check whether the problem is infeasible by demonstrating a solution to the following problem:

$$\mathbf{y} \neq 0,\ \mathbf{y} > 0,\ D^T\mathbf{y} = 0. \tag{E.7}$$

This system is nothing more than the *dual problem* of the original LP (Equation (E.4)). Thus, determining whether a given data set is linearly separable gives rise to a set of alternatives: *if the primal problem is infeasible, then the dual problem is feasible*. This allows us to certify that a given data set is inseparable by forming the dual problem and finding a solution.

In sum, A and B are separable if, and only if, the optimization problem* (E.4) *is feasible.

Appendix F

Hyperparameter sweeps for Chapter 5

As noted in Sections 5.4.1 and 5.4.4, the performance of ML algorithms can depend quite strongly on their hyperparameters. Table F.1 gives the values of the hyperparameters examined for the ML classifiers of Chapter 5.

Algorithm	**Hyperparameter**	**Default value**	**Values used in this work**
LDA	τ	10^{-4}	$10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}$, .1, .25, .5, .75, 1
QDA	s	0	0, .25, .5, .75, 1
Linear SVM	C	1	1,2,5,10,20,50,75,100,150,200,250
RBF SVM	C	1	1,2,5,10,20,50,75,100
	γ	$1/d$	.01, .1, 1, 10, 100
Perceptron	N_{epochs}	5	5, 50, 100 250, 300, 500, 750, 1000

Table F.1: **Algorithm hyperparameters**. The hyperparameters of each classification algorithm affect how it learns from the data. By sweeping the hyperparameters, its possible to determine good settings such that the classifier learned by the algorithm has high accuracy. (Default values are those used in scikit-learn 0.19.1.)

Algorithm	ϕ	Hyperparameter value	Accuracy I	Accuracy II
LDA	PP	10^{-1}	0.87	
LDA	SQ	10^{-5}	0.86	0.87
Linear SVM	PP	75	0.991	
Linear SVM	SQ	250	0.997	0.994
Perceptron	PP	100	0.999	
Perceptron	SQ	100	**0.9996**	**0.9994**
QDA	PP	0	**1.0**	
QDA	SQ	0	0.90	0.90
RBF SVM	PP	$C = 20, \gamma = .01$	0.998	
RBF SVM	SQ	$C = 10, \gamma = 1$	0.997	0.97

Table F.2: **Best classifier hyperparameters, as measured by mean classification accuracy**. Column "Accuracy I" gives the average accuracy of the classifier under a $K = 20$-fold shuffle-split cross-validation using the feature vectors in $\mathcal{C}_1$. Column "Accuracy II" shows the average accuracy when the classifier is trained using all the feature vectors, and then is used to predict the noise type for $\sim 20,000$ *previously unseen* feature vectors.

To determine which hyperparameter choices were best, we cross-validated the accuracy of the ML algorithms using 20 independently sampled training and testing sets, with 10% of the total data held back for testing. Figure F.1 show the results of sweeping the hyperparameters and evaluating classification accuracy on $\mathcal{C}_L$. Figure F.2 shows results for the feature engineered feature vectors. Finally, Table F.2 gives best hyperparameter values for learning using the engineered feature vectors, as determined by mean cross-validation accuracy.

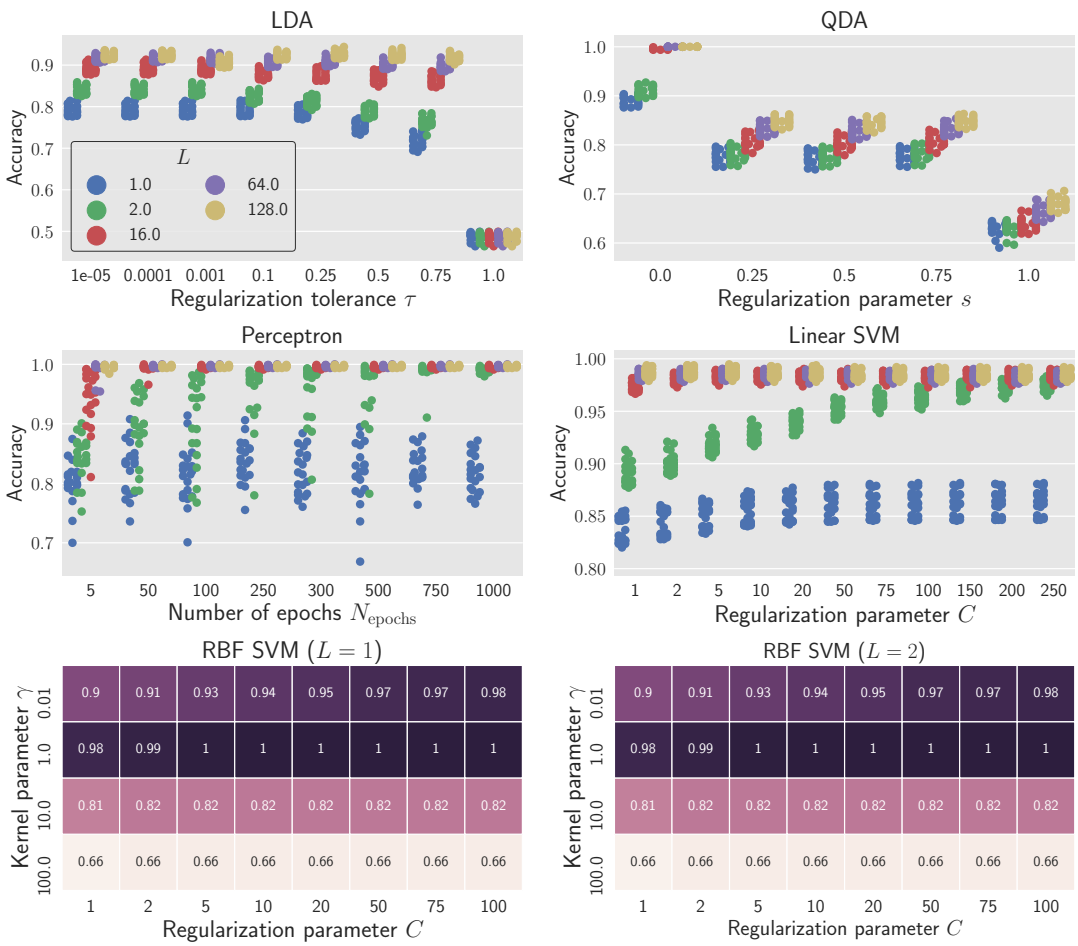

Figure F.1: **Classification accuracy on $\mathcal{C}_L$ depends on the hyperparameter of the classification algorithm**. For the top 4 plots, the hue indicates the value of L. For the bottom 2, the *average* classification accuracy is indicated in the heatmap.

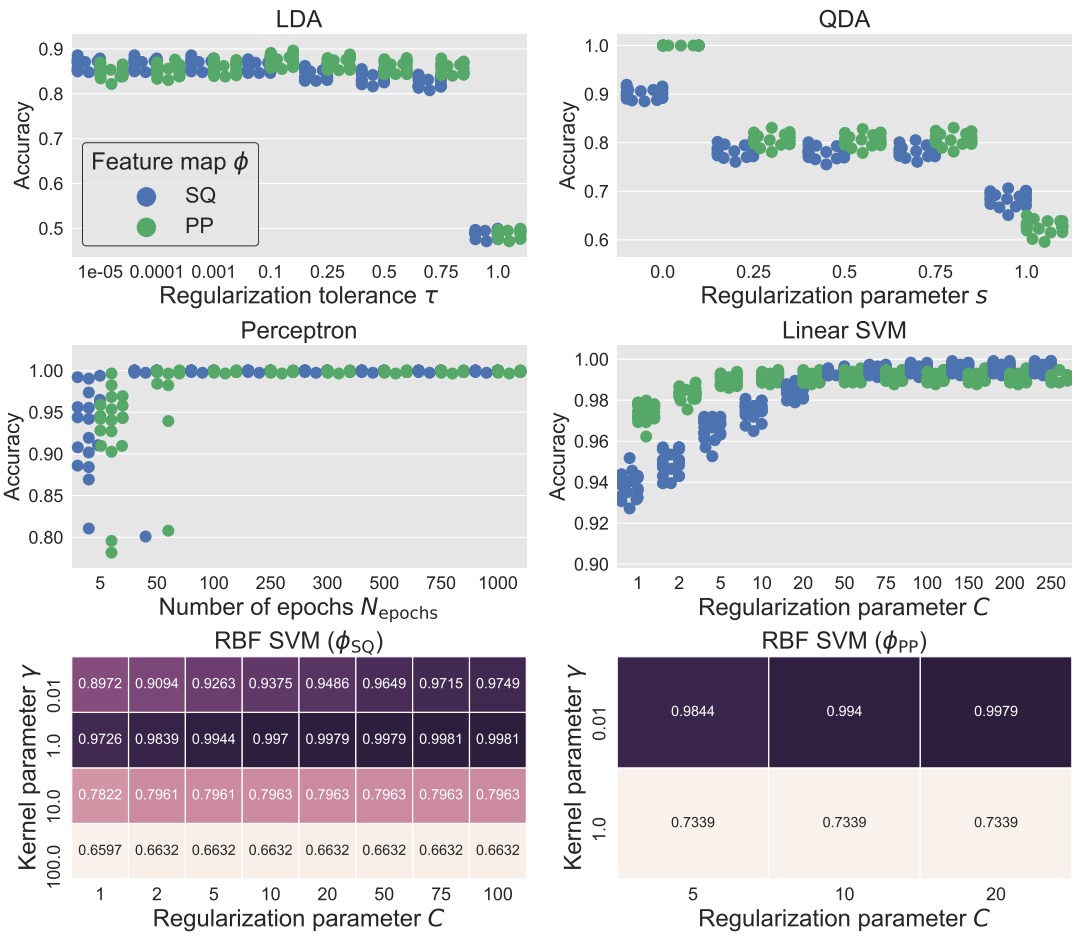

Figure F.2: **Hyperparameter sweep on engineered feature vectors**. For the top 4 plots, hue indicates which feature map was used. For the bottom 2, *average* classification accuracy is indicated in the heatmap.

Appendix G

The "Poisson bump/dip" phenomenon

Below, I show how, for Poisson-distributed data, the expected value of the loglikelihood ratio statistic depends strongly on the expected number of counts. In particular, the expected value is 1 only when the expected number of counts is large; otherwise, it very rapidly goes to zero.

If $K \sim \text{Poisson}(\theta_0)$, then

$$\Pr(K) = \frac{e^{-\theta_0}\theta_0^K}{K!}, \tag{G.1}$$

where θ_0 is the rate parameter, and the expected number of counts $\langle K \rangle = \theta_0$.

Consider two models: $\mathcal{M}_0 = \{\theta_0\}$ and $\mathcal{M}_1 = \{\theta \in [0, \infty)\}$. The loglikelihood ratio statistic comparing these two models is

$$\lambda(\mathcal{M}_0, \mathcal{M}_1) = -2 \log \left(\frac{\mathcal{L}(\theta_0)}{\mathcal{L}(\hat{\theta}_{\text{ML},1})} \right). \tag{G.2}$$

For $\mathcal{M}_1$, if $K = k$ counts were actually observed, $\hat{\theta}_{\text{ML},1} = k$. Plugging in the ML

estimate and writing out the likelihood function:

$$\begin{aligned}\lambda(\mathcal{M}_0, \mathcal{M}_1) &= -2\log\left(e^{k-\theta_0}\left(\frac{\theta_0}{k}\right)^k\right) \\ &= -2\left(k - \theta_0 + k\log(\theta_0) - k\log(k)\right).\end{aligned} \tag{G.3}$$

The expected value of the statistic is

$$\langle\lambda\rangle = -2(\langle k\rangle - \theta_0 + \langle k\rangle\log(\theta_0) - \langle k\log(k)\rangle). \tag{G.4}$$

Using the fact $\langle k\rangle = \theta_0$, the above equation simplifies some:

$$\langle\lambda\rangle = -2(\theta_0\log(\theta_0) - \langle k\log(k)\rangle). \tag{G.5}$$

It is important to note $\langle k\log(k)\rangle \neq \theta_0\log(\theta_0)$, as $\langle f(X)\rangle \neq f(\langle X\rangle)$, in general. The expected value of $k\log(k)$ can be computed as

$$\begin{aligned}\langle k\log(k)\rangle &= \sum_{k=0}^{\infty} k\log(k)\Pr(k) = \sum_{k=0}^{\infty} k\log(k)\frac{e^{-\theta_0}\theta_0^k}{k!} \\ &= e^{-\theta_0}\sum_{k=1}^{\infty}\frac{\log(k)\theta_0^k}{(k-1)!} \\ &= \theta_0 e^{-\theta_0}\sum_{k=0}^{\infty}\frac{\log(k+1)\theta_0^k}{k!}.\end{aligned} \tag{G.6}$$

Notice that the ratio of successive terms in the sum goes to zero as $k \to \infty$:

$$a_k = \frac{\log(k+1)\theta_0^k}{k!} \;;\; \lim_{k\to\infty}\left|\frac{a_{k+1}}{a_k}\right| = 0 \tag{G.7}$$

so that the sum is convergent. Here, I do not concern myself with computing the sum exactly, though I will make several approximations to get a handle on its behavior. Putting this expression into equation (G.5) gives

$$\langle\lambda\rangle = -2\left(\theta_0\log(\theta_0) - \theta_0 e^{-\theta_0}\sum_{k=0}^{\infty}\frac{\log(k+1)\theta_0^k}{k!}.\right) \tag{G.8}$$

A simple reindexing ($k = u - 1$) allows us to pull out a factor of θ_0:

$$\begin{aligned}\langle\lambda\rangle &= -2\left(\theta_0 \log(\theta_0) - \theta_0 e^{-\theta_0} \sum_{u=1}^{\infty} \frac{\log(u)\theta_0^{u-1}}{(u-1)!}\right) \\ &= -2\left(\theta_0 \log(\theta_0) - e^{-\theta_0} \sum_{u=1}^{\infty} \frac{\log(u)\theta_0^{u}}{(u-1)!}\right).\end{aligned} \tag{G.9}$$

Notice $\lim_{\theta_0 \to 0^+} \langle\lambda\rangle = 0$. This implies that when the expected number of counts is small, the ML estimate doesn't fluctuate too much, and so its contribution to the expected value of the loglikelihood ratio statistic is small.

Another interesting observation is that $\langle\lambda\rangle$ at $\theta_0 = 1$ is *greater* than 1:

$$\langle\lambda(\theta_0 = 1)\rangle = 2e^{-1} \sum_{u=1}^{\infty} \frac{\log(u)}{(u-1)!} \approx 1.15. \tag{G.10}$$

This implies that if the expected number of counts is about 1, then fluctuations in the ML estimate contribute *more* than 1 unit to the expected loglikelihood.

These two results show that $\langle\lambda\rangle$ depends on θ_0. Figure G.1 plots $\langle\lambda\rangle$ as a function of θ_0, where the sum is truncated to 150 terms. This plot has two features of note: $\langle\lambda\rangle \sim 1$ as $\theta_0 \to \infty$ (the expected number of counts is high), and that $\langle\lambda\rangle$ plunges to zero quickly if $\theta_0 < .5$.

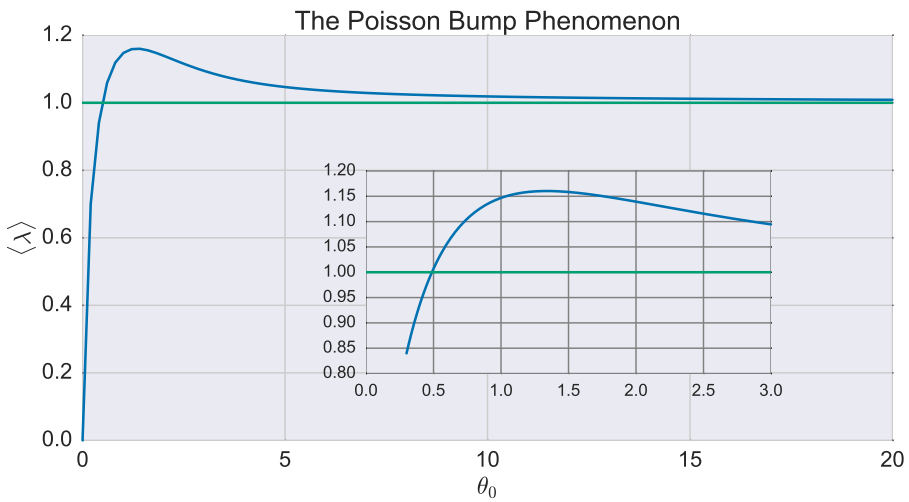

Figure G.1: **Behavior of loglikelihood ratio statistic for Poisson-distributed data**. For Poisson-distributed data, the behavior of λ depends strongly on the expected number of counts θ_0. The single parameter in the model can be fully contributing – i.e., the expected value of the statistic is 1 – only when the expected number of counts is large. Inset: Zooming in at $\theta_0 \approx 1$. The statistic grows for a bit after the expected number of counts is 1, and then begins to turn over and approach its asymptotic value.

Appendix H

Representations of Quantum Channels

The action of general quantum channel $\mathcal{E}$ on a state ρ is often denoted $\mathcal{E}[\rho]$. However, suppose we actually wanted to calculate the output of $\mathcal{E}$ when it acts on ρ. How exactly would we do that? That depends on the *representation* used for both $\mathcal{E}$ and ρ. This appendix discusses 3 commonly-used representations of ρ, and the corresponding representations for $\mathcal{E}$.

H.1 Kraus representation

In Chapter 2, we came across several representations of a quantum channel. The simplest is the *Kraus representation*, which generalizes the idea of unitary dynamics. (Recall that the unitary evolution of ρ is described by a channel $\rho \to U\rho U^\dagger$.) If ρ is a $d \times d$ density matrix, then the Kraus representation of $\mathcal{E}$ is

$$\mathcal{E}[\rho] = \sum_{j=1}^{k} K_j \rho K_j^\dagger, \;\; \sum_j K_j^\dagger K_j = \mathcal{I}_d, \tag{H.1}$$

where $k \leq d^2$ is the *Kraus rank* of the channel. The Kraus representation is nice in the sense that it has a straightforward connection to unitary evolution: the channel is unitary if, an only if, its Kraus rank is 1.

However, the *action* given by the Kraus representation is a bit weird: we have to sandwich ρ between two operators, and do two matrix multiplications. For the purposes of calculating an output state, this representation isn't the best. A representation where the action is just one matrix multiplication would be useful. This representation is called the *superoperator* representation.

H.2 Superoperator representation

A simple way to get to a superoperator representation is to use the *vectorization* operation, denoted vec(). This is a linear operation takes linear operators (matrices) and maps them to vectors. That is, if A is a $d \times d$ matrix, vec(A) is a vector of length d^2.)We say A is *row-vectorized* if vec(A) takes the rows of A and stacks them on top of one another, and *column-vectorized* if vec(A) stacks the columns. In what follows, we use column-vectorization.

Consider the single-qubit density matrix

$$\rho = \begin{pmatrix} a & b \\ b^\star & 1-a \end{pmatrix}, \ \rho \geq 0. \tag{H.2}$$

The column-vectorized form of ρ is given by

$$\text{vec}(\rho) \equiv |\rho) = \begin{pmatrix} a \\ b^\star \\ b \\ 1-a \end{pmatrix}. \tag{H.3}$$

This operation is linear as required, so that $\text{vec}(a\rho + b\sigma) = a\text{vec}(\rho) + b\text{vec}(\sigma)$. For

clarity, I'll use the notation $|\rho)$ to more indicate that ρ is now being treated as a column vector.

Applying vec() to do column-vectorization of a product of matrices yields:

$$\text{vec}(ABC) = (C^T \otimes A)\text{vec}(B). \tag{H.4}$$

Finally, because vec() is linear, its application on a sum of terms of the form in Equation (H.4) is simply the sum of their vectorized form:

$$\text{vec}\left(\sum_j A_j B C_j\right) = \sum_j \left(\left(C_j^T \otimes A_j\right)\text{vec}(B)\right) = \left(\sum_j C_j^T \otimes A_j\right)\text{vec}(B). \tag{H.5}$$

The right-most expression makes it clear that the vectorized form of the left-most side gives rise to an action that's just matrix multiplication on the vectorized form of B. Hence, a superoperator representation of $\mathcal{E}[\rho]$ is

$$\mathcal{E} = \sum_j (K_j^\star \otimes K_j), \tag{H.6}$$

and its action is

$$\mathcal{E}[\rho] = \mathcal{E}|\rho). \tag{H.7}$$

If we used row-vectorization instead, the superoperator representation would be $\mathcal{E} = \sum_j (K_j \otimes K_j^\star)$. This makes sense, as the difference between row and column-vectorization amounts to taking a complex conjugate, and that operation is invariant under multiplication and addition.

The representation of a linear operator depends on the representation of the vector space it acts on. In Equation (H.2), ρ is expressed in the basis of *matrix units* $|0\rangle\langle 0|, |0\rangle\langle 1|, |1\rangle\langle 0|$, and $|1\rangle\langle 1|$. Therefore, another way of representing $\mathcal{E}[\rho]$ is to consider its action on each basis vector. Because each basis vector is orthonormal, it's straightforward to define a 4×4 superoperator representation

$$\mathcal{E}_{(jk),(lm)} = \text{Tr}\left(|j\rangle\langle k|\mathcal{E}[|l\rangle\langle m|]\right), \tag{H.8}$$

where I've used a multi-index notation for simplicity. The matrix elements $\mathcal{E}_{(jk),(lm)}$ tell us "If the basis vector $|l\rangle\langle m|$ is acted on by $\mathcal{E}$, what is the overlap of the output with $|j\rangle\langle k|$?".

Of course, we could use any basis we want to expand ρ. This observation leads to the idea of the Pauli transfer matrix representation, discussed in the next subsection.

H.3 Pauli transfer matrix representation

Another common way to express a single-qubit state ρ is by using the Pauli matrices:

$$\rho = \frac{1}{2}\left(\mathcal{I} + \mathbf{r}\cdot\boldsymbol{\sigma}\right). \tag{H.9}$$

However, the Pauli matrices aren't orthonormal, but the matrices $b_j = \sigma_j/\sqrt{2}$ are. In terms of these matrices,

$$\rho = \sqrt{2}\left(b_{\mathcal{I}} + \mathbf{r}\cdot\boldsymbol{b}\right). \tag{H.10}$$

ρ can then be expressed in a vectorized form as

$$\rho \to \sqrt{2}\begin{pmatrix} 1 \\ r_X \\ r_Y \\ r_Z \end{pmatrix}. \tag{H.11}$$

The corresponding representation of a channel $\mathcal{E}$, known as the *Pauli transfer matrix representation* or *Liouvillian*, is given by

$$\mathcal{E}_{j,k} = \mathrm{Tr}(b_j\mathcal{E}[b_k]) = \frac{1}{2}\mathrm{Tr}(\sigma_j\mathcal{E}[\sigma_k]). \tag{H.12}$$

Recall that channels are linear, which allows us to pull out the $1/\sqrt{2}$ factor from b_k. More generally, given any basis of Hermitian matrices for the state (e.g., the Gell-Mann matrices or their generalization), then we can define this representation.

H.4 Deriving a superoperator representation for the Lindblad equation

Recall that the quantum channels generated by our noise models are a subset of those generated by general Lindbladian dynamics, described by the differential equation

$$\dot{\rho} = -i[H(t), \rho] + \sum_{jk} h_{jk} \left[A_j(t) \rho A_k^\dagger(t) - \frac{1}{2}\{A_k^\dagger(t) A_j(t), \rho\} \right]. \tag{H.13}$$

In this section, I show how to use the results of Section H to write the above equation using the vectorized representation of ρ, which will end up simplifying the differential equation to the form

$$\frac{d|\rho)}{dt} = \mathcal{L}(t)|\rho). \tag{H.14}$$

Recall that the vec() operation is linear:

$$\begin{aligned} \frac{d|\rho)}{dt} &= -i(\text{vec}(H\rho) - \text{vec}(\rho H)) \\ &\quad + \sum_{jk} h_{jk} \text{vec}\left(A_j \rho A_k^\dagger\right) - \frac{1}{2} \sum_{jk} h_{jk} \left(\text{vec}\left(A_k^\dagger A_j \rho\right) + \text{vec}\left(\rho A_k^\dagger A_j\right)\right). \end{aligned} \tag{H.15}$$

With the identities given in Equations (H.4) and (H.5), the above equation becomes

$$\begin{aligned} \frac{d|\rho)}{dt} &= -i\left(\mathcal{I} \otimes H - H^T \otimes \mathcal{I}\right)|\rho) + \sum_{jk} h_{jk} \left((A^\dagger)_k^T \otimes A_j(t)\right)|\rho) \\ &\quad - \frac{1}{2} \sum_{jk} h_{jk} \left[\left(\mathcal{I} \otimes \left(A_k^\dagger A_j\right)\right)|\rho) + \left(\left(A_k^\dagger A_j\right)^T \otimes \mathcal{I}\right)|\rho) \right]. \end{aligned} \tag{H.16}$$

This equation can be simplified using the fact that $(AB)^T = B^T A^T$, and that $(A^\dagger)^T = A^\star$, where * denotes the *conjugation* operation. Putting these pieces together yields

$$\begin{aligned} \frac{d|\rho)}{dt} &= -i\left(\mathcal{I} \otimes H - H^T \otimes \mathcal{I}\right)|\rho) \\ &\quad + \sum_{jk} h_{jk} \left(A_k^\star \otimes A_j\right)|\rho) - \frac{1}{2} \sum_{jk} h_{jk} \left[\mathcal{I} \otimes A_k^\dagger A_j + A_j^T A_k^\star \otimes \mathcal{I}\right]|\rho). \end{aligned} \tag{H.17}$$

This equation can be nicely re-written in the form given in Equation (H.14), where the (generally time-dependent) Lindbladian $\mathcal{L}$ is

$$\mathcal{L} = -i\left(\mathcal{I}\otimes H - H^T\otimes\mathcal{I}\right) + \sum_{jk} h_{jk}\left[A_k^\star\otimes A_j - \frac{1}{2}\left(\mathcal{I}\otimes A_k^\dagger A_j + A_j^T A_k^\star\otimes\mathcal{I}\right)\right]. \tag{H.18}$$

Because the units of $\mathcal{L}$ are "per unit time", the coefficients h_{jk} are interpreted as *rates*.

In the noise models I consider, $\mathcal{L}$ is time-invariant, so the formal solution to Equation (H.14) is

$$|\rho(t)) = e^{\mathcal{L}t}|\rho(0)). \tag{H.19}$$

Taking t to be one fundamental timestep of the QIP, $\mathcal{L}$ is the generator of a quantum channel:

$$|\rho) \to \mathcal{E}|\rho) \quad \text{where} \quad \mathcal{E} = e^{\mathcal{L}}. \tag{H.20}$$

Appendix I

Description of data for Chapter 6

	η	# realizations
Amplitude damping	0.001000	300
	0.002000	300
	0.005000	300
	0.008000	300
	0.010000	300
	0.020000	300
	0.050000	300
	0.080000	300
	0.100000	300
Stochastic	0.001000	300
	0.002154	300
	0.004642	300
	0.010000	300
	0.021544	300
	0.046416	300
	0.100000	300

Table I.1: Data set description for amplitude damping and stochastic noise

	η	# realizations
Non-Pauli stochastic	0.001	300
	0.002	300
	0.005	300
	0.008	300
	0.010	1562
	0.020	300
	0.050	300
	0.080	300
	0.100	1562
	0.250	1562
Pauli stochastic	0.001	800
	0.002	600
	0.005	600
	0.008	600
	0.010	4200
	0.020	600
	0.050	600
	0.080	600
	0.100	4200
	0.250	4200

Table I.2: Data set description for non-Pauli stochastic and Pauli-stochastic noise

	η	# realizations
Anisotropic	0.01	300
	0.02	300
	0.05	300
	0.08	300
	0.10	300
	0.25	300
Isotropic	0.01	300
	0.02	300
	0.05	300
	0.08	300
	0.10	300
	0.25	300

Table I.3: Data set description for isotropic and anisotropic Pauli-stochastic noise

	η	# realizations
Coherent	0.000100	300
	0.000215	300
	0.000464	300
	0.001000	300
	0.002154	300
	0.004642	300
	0.010000	300
	0.021544	300
	0.046416	300
	0.100000	300
	0.119581	300
	0.142997	300
	0.170998	300
	0.204481	300
	0.244521	300
	0.292402	300
	0.349658	300
	0.418126	300
	0.500000	300
Stochastic	0.000100	300
	0.000215	300
	0.000464	300
	0.001000	300
	0.002154	300
	0.004642	300
	0.010000	300
	0.021544	300
	0.046416	300
	0.100000	300
	0.119581	300
	0.142997	300
	0.170998	300
	0.204481	300
	0.244521	300
	0.292402	300
	0.349658	300
	0.418126	300
	0.500000	300

Table I.4: Data set description for coherent and stochastic noise

Appendix J

Calculations for Chapter 4

J.1 Expected maximum of Gaussian random variables

Consider N i.i.d. $\mathcal{N}(0, \epsilon^2)$ random variables $\delta_1, \cdots, \delta_N$, and let m be the maximum value of the sample:

$$m = \max_j \delta_j. \tag{J.1}$$

In this section, I compute an upper bound on $\langle m \rangle$.

Consider the quantity $\mathrm{Exp}[t\langle m \rangle]$. Applying Jensen's inequality gives $\mathrm{Exp}[t\langle m \rangle] \leq \langle \mathrm{Exp}[tm] \rangle$. Because the exponential is a monotonically increasing function, $\mathrm{Exp}[tm]$ is the same as the maximum, over δ_j, of $\mathrm{Exp}[t\delta_j]$:

$$\langle \mathrm{Exp}[tm] \rangle = \left\langle \max_j \mathrm{Exp}[t\delta_j] \right\rangle. \tag{J.2}$$

Now, $\max_j \mathrm{Exp}[t\delta_j]$ is itself upper-bounded by the *sum* of $\mathrm{Exp}[t\delta_j]$, which allows us to pull the expectation through and gives

$$\left\langle \max_j \mathrm{Exp}[t\delta_j] \right\rangle \leq \sum_{j=1}^{N} \langle \mathrm{Exp}[t\delta_j] \rangle. \tag{J.3}$$

Finally, $\mathrm{Exp}[t\delta_j]$ is the *moment generating function* of these Gaussian random variables, from which it follows that $\langle \mathrm{Exp}[t\delta_j] \rangle = \mathrm{Exp}\left[t^2\epsilon^2/2\right]$. Therefore,

$$\sum_{j=1}^{N} \langle \mathrm{Exp}[t\delta_j] \rangle = N\mathrm{Exp}\left[t^2\epsilon^2/2\right]. \tag{J.4}$$

Putting all these equalities and inequalities together gives

$$\mathrm{Exp}[t\langle m \rangle] \leq N\mathrm{Exp}\left[t^2\epsilon^2/2\right] \tag{J.5}$$

for any t. Taking the logarithm of both sides gives

$$\langle m \rangle \leq \frac{\log(N)}{t} + \frac{t\epsilon^2}{2}, \tag{J.6}$$

valid for any $t > 0$. To find the lowest upper bound, minimize over t by differentiating the above expression with respect to t and setting it to zero:

$$\frac{d\langle m \rangle}{dt} = -\frac{\log(d)^2}{t^2} + \epsilon^2/2 = 0 \implies t = \sqrt{2\log(d)}/\epsilon. \tag{J.7}$$

Plugging this expression for t back into Equation (J.6) gives

$$\langle m \rangle \leq \epsilon\sqrt{2\log(N)}. \tag{J.8}$$

J.2 Solving Equation (4.29)

In this section, I derive an approximate solution to Equation (4.29):

$$-z + A\left(e^{-z^2/2} + z\Phi_1(z)\sqrt{2\pi}\right) = 0, \tag{J.9}$$

where $A = (1 - (r/d))/\sqrt{2\pi}$. As noted in the main text, $z \sim \mathcal{O}(1)$, meaning that the above equation can be solved approximately by (a) expanding every term about $z = c$ for some choice of c, and (b) solving the resulting equation. For simplicity, I take $c = 1$.

The Taylor series for $e^{-z^2/2}$ about $z = 1$ is easily computed as

$$e^{-z^2/2} = e^{-1/2}(2 - z) + \mathcal{O}((z-1)^3) \tag{J.10}$$

Computing Taylor series of $z\Phi_1(z)$ about $z = 1$ is also straightforward, though we need to recall the fundamental theorem of calculus when taking the derivatives:

$$\begin{aligned} z\Phi_1(z) &= \Phi_1(1) + \left(\Phi_1(z) + \frac{z}{\sqrt{2\pi}}e^{-z^2/2}\right)\Big|_{z=1}(z-1) \\ &\quad + \frac{1}{2}\left(\frac{2e^{-z^2/2}}{\sqrt{2\pi}} - \frac{e^{-z^2/2}}{\sqrt{2\pi}}z^2\right)\Big|_{z=1}(z-1)^2 + \mathcal{O}((z-1)^3) \\ &= \Phi_1(1) + \left(\Phi_1(1) + \frac{e^{-1/2}}{\sqrt{2\pi}}\right)(z-1) + \frac{e^{-1/2}}{2\sqrt{2\pi}}(z-2)^2 + \mathcal{O}((z-1)^3) \end{aligned} \tag{J.11}$$

With these approximations, the defining equation for z becomes

$$-\frac{z}{A} + e^{-1/2}(2-z) + \left[\Phi_1(1) + \left(\Phi_1(1) + \frac{e^{-1/2}}{\sqrt{2\pi}}\right)(z-1) + \frac{e^{-1/2}}{2\sqrt{2\pi}}(z-2)^2\right]\sqrt{2\pi} = 0, \tag{J.12}$$

which simplifies to a nice quadratic equation

$$\frac{Ae^{-1/2}}{2}z^2 + (\sqrt{2\pi}A\Phi_1(1) - Ae^{-1/2} - 1)z + \frac{3A}{2}e^{-1/2} = 0. \tag{J.13}$$

The roots to this equation are

$$z_\pm = \frac{e^{1/2}}{A}\left(1 - \sqrt{2\pi}A\Phi_1(1) + Ae^{-1/2} \pm \sqrt{D}\right), \tag{J.14}$$

where

$$D = 2\pi A^2(\Phi_1(1))^2 - 2\sqrt{2\pi}A^2\Phi_1(1)e^{-1/2} - 2A^2e^{-1} - 2\sqrt{2\pi}A\Phi_1(1) + 2Ae^{-1/2} + 1. \tag{J.15}$$

Because there are a lot of factors of $\sqrt{2\pi}$ floating around, take $A = B/\sqrt{2\pi}$ (where

$B = 1 - r/d$), and simplify:

$$\begin{aligned} z_{\pm} &= \frac{\sqrt{2\pi e}}{B}\left(1 - B\Phi_1(1) + B\frac{e^{-1/2}}{\sqrt{2\pi}} \pm \sqrt{D}\right), \\ D &= B^2\left(\left(\Phi_1(1)\right)^2 - \sqrt{\frac{2}{\pi}}\Phi_1(1)e^{-1/2} - \frac{1}{\pi}e^{-1}\right) \\ &\quad + B\left(-2\Phi_1(1) + \sqrt{\frac{2}{\pi}}e^{-1/2}\right) + 1. \end{aligned} \tag{J.16}$$

Comparison to numerical results indicates we need to take the negative root, giving

$$\boxed{z = \frac{\sqrt{2\pi e}}{B}\left(1 - B\Phi_1(1) + B\frac{e^{-1/2}}{\sqrt{2\pi}} - \sqrt{D}\right).} \tag{J.17}$$

References

[1] S. Aaronson. PDQP/qpoly = ALL. *arXiv*, 1805.08577.

[2] S. Aaronson and A. Arkhipov. The Computational Complexity of Linear Optics. *Theory of Computing*, 9(4):143–252, 2013. doi:10.4086/toc.2013.v009a004.

[3] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd. A Rewriting System for Convex Optimization Problems. *Journal of Control and Decision*, 5(1):42–60, 2018. doi:10.1080/23307706.2017.1397554.

[4] D. Aharonov and M. Ben-or. Fault-tolerant quantum computation with constant error rate. *arXiv*. URL https://arxiv.org/abs/quant-ph/9906129.

[5] H. Akaike. A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, dec 1974. doi:10.1109/TAC.1974.1100705.

[6] J. B. Altepeter, E. R. Jefferey, and P. G. Kwiat. Photonic State Tomography. In *Advances in Atomic, Molecular, and Optical Physics*, volume 52, chapter 2, pages 1–54. Elsevier B.V., 2005. doi:10.1016/S1049-250X(05)52003-2.

[7] A. Ambainis. Quantum Search Algorithms. *arXiv*. URL https://arxiv.org/abs/quant-ph/0504012.

[8] D. Amelunxen and M. Lotz. Living on the edge: Phase transitions in convex

programs with random data. *Information and Inference*, 3(3):224–294, 2014, 1303.6672. doi:10.1093/imaiai/iau005.

[9] D. Amelunxen, M. Lotz, M. B. McCoy, and J. A. Tropp. Living on the edge: Phase transitions in convex programs with random data. *Information and Inference: A Journal of the IMA*, 3:224–294, 2014, 1303.6672. doi:10.1093/imaiai/iau005.

[10] A. Anis and A. I. Lvovsky. Maximum-likelihood coherent-state quantum process tomography. *New Journal of Physics*, 14, 2012, 1204.5936. doi:10.1088/1367-2630/14/10/105021.

[11] M. Appleby, C. A. Fuchs, B. C. Stacey, and H. Zhu. Introducing the Qplex: a novel arena for quantum theory. *European Physical Journal D*, 71(7), 2017, 1612.03234. doi:10.1140/epjd/e2017-80024-y.

[12] G. M. D. Ariano, M. G. A. Paris, and M. F. Sacchi. Quantum Tomography. *Advances in Imaging and Electron Physics*, 128:205–308, 2003. URL `https://books.google.com/books?id=wmD8LFmAsgoC`.

[13] I. Arnaldo, U.-M. O. Reilly, and K. Veeramachaneni. Building Predictive Models via Feature Synthesis. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 983–990, Madrid, Spain, 2015. doi:10.1145/2739480.2754693.

[14] L. M. Artiles, R. D. Gill, and M. I. Guta. An invitation to quantum tomography. *Journal of the Royal Statistical Society*, 67:109–134, 2005. doi:10.1111/j.1467-9868.2005.00491.x.

[15] K. M. R. Audenaert, J. Calsamiglia, R. Munoz-Tapia, E. Bagan, L. Masanes, A. Acín, and F. Verstraete. Discriminating States : The Quan-

tum Chernoff Bound. *Physical Review Letters*, 98(16):160501, 2007. doi:10.1103/PhysRevLett.98.160501.

[16] P. Baldi, K. Bauer, C. Eng, P. Sadowski, and D. Whiteson. Jet substructure classification in high-energy physics with deep neural networks. *Physical Review D*, 93(094034), 2016, 1603.09349. doi:10.1103/PhysRevD.93.094034.

[17] K. Banaszek. Quantum homodyne tomography with a priori constraints. *Physical Review A*, 59(6):4797–4800, jan 1999, 9901064. doi:10.1103/PhysRevA.59.4797.

[18] J. P. Barnes, C. J. Trout, D. Lucarelli, and B. D. Clader. Quantum error-correction failure distributions : Comparison of coherent and stochastic error models. *Physical Review A*, 95(062338), 2017. doi:10.1103/PhysRevA.95.062338.

[19] D. Bartholomew. A Test Of Homogeneity for Ordered Alternatives. *Biometrika*, 46(1):36–48, 1959. doi:10.2307/2332806.

[20] D. Bartholomew. A Test of Homogeneity for Ordered Alternatives. II. *Biometrika*, 46(3):328–335, 1959.

[21] D. Bartholomew. A Test of Homogeneity of Means Under Restricted Alternatives. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 23(2):239–281, 1961. URL `https://www.jstor.org/stable/2984019`.

[22] S. Barz, J. F. Fitzsimons, E. Kashefi, and P. Walther. Experimental verification of quantum computation. *Nature Physics*, 9:727–731, 2013. doi:10.1038/nphys2763.

[23] M. Basu and T. Kam Ho. Linear Separability in Descent Procedures for Linear Classifiers. In M. Basu and T. Kam Ho, editors, *Data Complexity*

in Pattern Recognition, pages 69–90. Springer-Verlag London Limited, 2006. doi:10.1007/978-1-84628-172-3_4.

[24] J. D. Bekenstein. Energy Cost of Information Transfer. *Physical Review Letters*, 46(10):623–626, 1981. doi:10.1103/PhysRevLett.46.623.

[25] J. D. Bekenstein. Universal upper bound on the entropy-to-energy ratio for bounded systems. *Physical Review D*, 23(2):287–298, 1981. doi:10.1103/PhysRevD.23.287.

[26] C. H. Bennett. Logical Reversibility of Computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973. doi:10.1147/rd.176.0525.

[27] D. J. Bernstein and T. Lange. Post-quantum cryptography. *Nature*, 549:188–194, sep 2017. doi:10.1038/nature23461.

[28] J. I. Bertrand and P. Bertrand. A Tomographic Approach to Wigner's function. *Foundations of Physics*, 17(4):397–405, 1987. doi:10.1007/BF00733376.

[29] A. Berut, A. Arakelyan, A. Petrosyan, S. Ciliberto, R. Dillenschneider, and E. Lutz. Experimental verification of Landauer's principle linking information and thermodynamics. *Nature*, 483:187–189, 2012. doi:10.1038/nature10872.

[30] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017, 1611.09347. doi:10.1038/nature23474.

[31] L. S. Bishop, S. Bravyi, A. W. Cross, J. M. Gambetta, and J. A. Smolin. Quantum Volume, 2017. URL `https://pdfs.semanticscholar.org/650c/3fa2a231cd77cf3d882e1659ee14175c01d5.pdf`.

[32] F. Bloch. Nuclear Induction. *Physical Review*, 70(7-8):460–474, 1946. doi:10.1103/PhysRev.70.460.

[33] R. Blume-Kohout. Robust Error Bars for Quantum Tomography. *arXiv*, 1202.5270.

[34] R. Blume-Kohout. *Decoherence and Beyond*. PhD thesis, University of California, Berkeley, 2005. URL http://adsabs.harvard.edu/abs/2005PhDT.....128B.

[35] R. Blume-Kohout. Hedged Maximum Likelihood Quantum State Estimation. *Physical Review Letters*, 105(20):200504, 2010. doi:10.1103/PhysRevLett.105.200504.

[36] R. Blume-Kohout. Optimal, reliable estimation of quantum states. *New Journal of Physics*, 12:043034, 2010, 0611080. doi:10.1088/1367-2630/12/4/043034.

[37] R. Blume-Kohout, J. K. Gamble, E. Nielsen, J. Mizrahi, J. D. Sterk, and P. Maunz. Robust, self-consistent, closed-form tomography of quantum logic gates on a trapped ion qubit. oct, 1310.4492.

[38] R. Blume-Kohout, J. K. Gamble, E. Nielsen, K. Rudinger, J. Mizrahi, K. Fortier, and P. Maunz. Demonstration of qubit operations below a rigorous fault tolerance threshold with gate set tomography. *Nature Communications*, 8:1–13, 2017. doi:10.1038/ncomms14485.

[39] R. Blume-Kohout, J. O. S. Yin, and S. J. van Enk. Entanglement verification with finite data. *Physical Review Letters*, 105(17):170501, 2010, 1005.0003. doi:10.1103/PhysRevLett.105.170501.

[40] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven. Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14:595–600, 2018. doi:10.1038/s41567-018-0124-x.

[41] I. Borg and P. Groenen. *Modern Multidimensional Scaling - Theory and Applications*. Springer-Verlag New York, 2nd edition, 2005. doi:10.1007/0-387-28981-X.

[42] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, Pennsylvania, USA, 1992. ACM. doi:10.1145/130385.130401.

[43] N. Boulant, T. F. Havel, M. A. Pravia, and D. G. Cory. Robust method for estimating the Lindblad operators of a dissipative quantum process from measurements of the density operator at multiple time points. *Physical Review A*, 67(4):042322, 2003, 0211046. doi:10.1103/PhysRevA.67.042322.

[44] G. E. P. Box. Robustness in the strategy of scientific model building. In R. L. Launer and G. N. Wilkinson, editors, *Workshop on Robustness in Statistics*, pages 201–236, Research Triangle Park, North Carolina, USA, 1978. Academic Press. doi:10.1016/C2013-0-11050-1.

[45] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, New York, USA, 7th edition, 2009.

[46] S. Bravyi, D. Gosset, and R. Koenig. Quantum advantage with shallow circuits. *Science*, 362(6412):308–311, 2018, 1704.00690. URL `https://dx.doi.org/10.1126/science.aar3106`.

[47] G. Breitenbach, S. Schiller, and J. Mlynek. Measurement of the quantum states of squeezed light. *Nature*, 387:471–475, 1997. doi:10.1038/387471a0.

[48] H.-P. Breuer and F. Petruccione. *The Theory of Open Quantum Systems*. Oxford University Press, 2007.

[49] W. G. Brown and B. Eastin. Randomized benchmarking with restricted gate sets. *Physical Review*, 97:062323, 2018. doi:10.1103/PhysRevA.97.062323.

[50] K. P. Burnham. Multimodel Inference: Understanding AIC and BIC in Model Selection. *Sociological Methods & Research*, 33(2):261–304, 2004. doi:10.1177/0049124104268644.

[51] T. Cai, D. Kim, Y. Wang, M. Yuan, and H. H. Zhou. Optimal large-scale quantum state tomography with Pauli measurements. *Annals of Statistics*, 44(2):682–712, 2016, 1603.07559. doi:10.1214/15-AOS1382.

[52] E. Candès and T. Tao. Decoding by Linear Programming. *IEEE Trans. Inf. Theory*, 51(12):4203–4215, 2005. doi:10.1109/TIT.2005.858979.

[53] E. J. Candes and Y. Plan. A Probabilistic and RIPless Theory of Compressed Sensing. *IEEE Transactions on Circuit Theory*, 57(11):7235–7254, 2010.

[54] E. J. Candes, J. K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006, 0503066. doi:10.1002/cpa.20124.

[55] E. J. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006, 0410542. doi:10.1109/TIT.2006.885507.

[56] G. Carleo and M. Troyer. Solving the Quantum Many-Body Problem with Artificial Neural Networks. *Science*, 355(6325):602–606, 2017, 1606.02318. doi:10.1126/science.aag2302.

[57] J. Carolan, J. D. A. Meinecke, P. J. Shadbolt, N. J. Russell, N. Ismail, K. Wörhoff, T. Rudolph, M. G. Thompson, J. L. O'Brien, J. C. F. Matthews, and A. Laing. On the experimental verification of quantum complexity in linear optics. *Nature Photonics*, 8:621–626, 2014. doi:10.1038/nphoton.2014.152.

[58] A. Carpentier, J. Eisert, D. Gross, and R. Nickl. Uncertainty Quantification for Matrix Compressed Sensing and Quantum Tomography Problems. *arXiv*, 1504.03234.

[59] J. Carrasquilla and R. G. Melko. Machine learning phases of matter. *Nature Physics*, 13(5):431–434, 2017, 1605.01735. doi:10.1038/nphys4035.

[60] W. B. Case. Wigner functions and Weyl transforms for pedestrians. *American Journal of Physics*, 76(10):937–946, 2008, 0512060. doi:10.1119/1.2957889.

[61] G. Casella, C. P. Robert, and M. T. Wells. Generalized Accept – Reject sampling schemes. *Lecture Notes–Monograph Series*, 45:342–347, 2004. doi:10.1214/lnms/1196285403.

[62] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers and Electrical Engineering*, 40(1):16–28, 2014, 1606.03476. doi:10.1016/j.compeleceng.2013.11.024.

[63] Z. Chen, Q. Zhou, C. Xue, X. Yang, G. Guo, and G. Guo. 64-Qubit Quantum Circuit Simulation. *Science Bulletin*, 63(15):964–971, 2018, 1802.06952. doi:10.1016/j.scib.2018.06.007.

[64] R. Chiao and J. Garrison. *Quantum Optics*. Oxford University Press, 2014.

[65] K. Ch'Ng, J. Carrasquilla, R. G. Melko, and E. Khatami. Machine learning phases of strongly correlated fermions. *Physical Review X*, 7(3):1–9, 2017, 1609.02552. doi:10.1103/PhysRevX.7.031038.

[66] I. L. Chuang and M. A. Nielsen. Prescription for experimental determination of the dynamics of a quantum black box. *Journal of Modern Optics*, 44(11-12):2455–2467, 1997. doi:10.1080/09500349708231894.

[67] B. Cipra. How Number Theory Got the Best of the Pentium Chip. *Science*, 267(5195):175, 1995. doi:10.1126/science.267.5195.175.

[68] J. Cogan, M. Kagan, E. Strauss, and A. Schwarztman. Jet-images: computer vision inspired techniques for jet tagging. *Journal of High Energy Physics*, 2015, 1407.5675. doi:10.1007/JHEP02(2015)118.

[69] J. Combes, C. Granade, C. Ferrie, and S. T. Flammia. Logical Randomized Benchmarking. *arXiv*, 1702.03688.

[70] J. V. Corbett. The Pauli Problem, State Reconstruction, and Quantum-Real Numbers. *Reports on Mathematical Physics*, 57(I):53–68, 2006. doi:10.1016/S0034-4877(06)80008-X.

[71] C. Cortes and V. N. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995. doi:10.1007/BF00994018.

[72] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

[73] M. Cramer, M. B. Plenio, S. T. Flammia, R. Somma, D. Gross, S. D. Bartlett, O. Landon-Cardinal, D. Poulin, and Y.-K. Liu. Efficient quantum state tomography. *Nature Communications*, 1(9):147–149, 2010, 1101.4366. doi:10.1038/ncomms1147.

[74] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 1st edition, 2000.

[75] M. P. da Silva, O. Landon-Cardinal, and D. Poulin. Practical characterization of quantum devices without tomography. *Physical Review Letters*, 107(21):210404, 2011, 1104.3835. doi:10.1103/PhysRevLett.107.210404.

[76] L. Dalcin, R. Paz, and M. Storti. MPI for Python. *Journal of Parallel and Distributed Computing*, 65(9):1108–1115, 2005. doi:10.1016/j.advwatres.2011.04.013.

[77] A. S. Darmawan and D. Poulin. Tensor-Network Simulations of the Surface Code under Realistic Noise. *Physical Review Letters*, 119(040502), 2017. doi:10.1103/PhysRevLett.119.040502.

[78] C. M. Dawson and M. A. Nielsen. The Solovay-Kitaev Algorithm. *Quantum Information and Computation*, 6(1):81–95, 2006, 0505030. URL https://arxiv.org/abs/quant-ph/0505030.

[79] C. Debao. Degree of approximation by superpositions of a sigmoidal function. *Approximation Theory and its Applications*, 9(3):17–28, 1993. doi:10.1007/BF02836480.

[80] S. Decherchi, A. Berteotti, G. Bottegoni, W. Rocchia, and A. Cavalli. The ligand binding mechanism to purine nucleoside phosphorylase elucidated via molecular dynamics and machine learning. *Nature Communications*, 6(6155), 2015, 1011.1669. doi:10.1038/ncomms7155.

[81] D.-L. Deng, X. Li, and S. Das Sarma. Machine learning topological states. *Physical Review B*, 96(19):195145, 2017. doi:10.1103/PhysRevB.96.195145.

[82] D.-L. Deng, X. Li, and S. Das Sarma. Quantum entanglement in neural network states. *Physical Review X*, 7(2):1–17, 2017, 1701.04844. doi:10.1103/PhysRevX.7.021021.

[83] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A*, 400(1818):97–117, 1985. doi:10.1098/rspa.1985.0070.

[84] I. H. Deutsch, G. K. Brennen, and P. S. Jessen. Quantum Computing with Neutral Atoms in an Optical Lattice. *Fortschritte der Physik*, 48(9-11):925–943, 2000.

[85] S. J. Devitt, W. J. Munro, and K. Nemoto. Quantum Error Correction for Beginners. jun, 0905.2794. doi:10.1088/0034-4885/76/7/076001.

[86] S. Diamond and S. Boyd. CVXPY: A Python-Embedded Modeling Language for Convex Optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016. doi:10.1186/s12958-015-0070-8.

[87] M. T. Dimario and F. E. Becerra. Robust Measurement for the Discrimination of Binary Coherent States. *Physical Review Letters*, 121(2):23603, 2018. doi:10.1103/PhysRevLett.121.023603.

[88] E. Dolnick. *The Clockwork Universe: Isaac Newton, the Royal Society, and the Birth of the Modern World*. HarperCollins, 2011. URL `https://books.google.com/books?id=IJ-GgyfvKL8C`.

[89] L. M. Duan, M. D. Lukin, J. I. Cirac, and P. Zoller. Long-distance quantum communication with atomic ensembles and linear optics. *Nature*, 414:413–418, 2001, 0105105. doi:10.1038/35106500.

[90] B. Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. doi:10.1214/aos/1176344552.

[91] J. Emerson, R. Alicki, and K. Zyczkowski. Scalable noise estimation with random unitary operators. *Journal of Optics B: Quantum and Semiclassical Optics*, 7:S347–S352, 2005. doi:10.1088/1464-4266/7/10/021.

[92] P. Faist and R. Renner. Practical, Reliable Error Bars in Quantum Tomography. *Physical Review L*, 117(1):010404, 2016, 1509.06763. doi:10.1103/PhysRevLett.117.010404.

[93] R.-E. Fan, K.-W. Change, C.-J. Hsie, X.-R. Wang, and C.-J. Lin. LIBLINEAR : A Library for Large Linear Classification. *Journal of Machine Learning Re-*

search, 9:1871–1874, 2008. URL http://www.jmlr.org/papers/v9/fan08a.html.

[94] B. Farley and W. Clark. Simulation of self-organizing systems by digital computer. *Transactions of the IRE Professional Group on Information Theory*, 4(4):76–84, 1954. doi:10.1109/TIT.1954.1057468.

[95] R. Fazio, G. M. Palma, and J. Siewert. Fidelity and Leakage of Josephson Qubits. *Physical Review Letters*, 83(25):5385–5388, 1999. doi:10.1103/PhysRevLett.83.5385.

[96] G. Feng, J. J. Wallman, B. Buonacorsi, F. H. Cho, D. K. Park, T. Xin, D. Lu, J. Baugh, and R. Laflamme. Estimating the Coherence of Noise in Quantum Control of a Solid-State Qubit. *Physical Review Letters*, 117(26):260501, 2016, 1603.03761. doi:10.1103/PhysRevLett.117.260501.

[97] C. Ferrie. High posterior density ellipsoids of quantum states. *New Journal of Physics*, 16(023006), 2014. doi:10.1088/1367-2630/16/2/023006.

[98] C. Ferrie and R. Blume-Kohout. Maximum likelihood quantum state tomography is inadmissible. *arXiv*, 1808.01072.

[99] C. Ferrie and R. Blume-Kohout. Minimax Quantum Tomography : Estimators and Relative Entropy Bounds. *Physical Review Letters*, 116(9):090407, 2016. doi:10.1103/PhysRevLett.116.090407.

[100] C. Ferrie and J. Emerson. Framed Hilbert space: Hanging the quasi-probability pictures of quantum theory. *New Journal of Physics*, 11, 2009, 0903.4843. doi:10.1088/1367-2630/11/6/063040.

[101] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982. doi:10.1007/BF02650179.

[102] C. C. Fischer, K. J. Tibbetts, D. Morgan, and G. Ceder. Predicting crystal structure by merging data mining with quantum mechanics. *Nature Materials*, 5(8):641–646, 2006. doi:10.1038/nmat1691.

[103] R. A. Fisher. *The Design of Experiments*. Oliver and Boyd, 1st edition, 1935.

[104] R. A. Fisher. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Human Eugenics*, 7(2):179–188, 1936. doi:10.1111/j.1469-1809.1936.tb02137.x.

[105] S. T. Flammia, D. Gross, Y.-K. Liu, and J. Eisert. Quantum tomography via compressed sensing: error bounds, sample complexity and efficient estimators. *New Journal of Physics*, 14(095022), 2012, 1205.2300. doi:10.1088/1367-2630/14/9/095022.

[106] S. T. Flammia and Y.-K. Liu. Direct fidelity estimation from few Pauli measurements. *Physical Review Letters*, 106(23):2–5, 2011, 1104.4695. doi:10.1103/PhysRevLett.106.230501.

[107] L. Foschini. Where the "it from bit" come from? *arXiv*, 1306.0545.

[108] T. Fösel, P. Tighineanu, T. Weiss, and F. Marquardt. Reinforcement Learning with Neural Networks for Quantum Feedback. *arXiv*, 1802.05267.

[109] A. G. Fowler. Two-dimensional color-code quantum computation. *Physical Review A*, 83(4):42310, 2011. doi:10.1103/PhysRevA.83.042310.

[110] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A - Atomic, Molecular, and Optical Physics*, 86(3):32324, 2012, 1208.0928. doi:10.1103/PhysRevA.86.032324.

[111] E. Fredkin and T. Toffoli. Conservative Logic. *International Journal of Theoretical Physics*, 21(3-4):219–253, 1982. doi:10.1007/BF01857727.

[112] Y. V. Fyodorov. Introduction to the Random Matrix Theory: Gaussian Unitary Ensemble and Beyond. *arXiv*. URL https://arxiv.org/abs/math-ph/0412017.

[113] J. M. Gambetta, A. D. Corcoles, S. T. Merkel, B. R. Johnson, J. A. Smolin, J. M. Chow, C. A. Ryan, C. Rigetti, S. Poletto, T. A. Ohki, M. B. Ketchen, and M. Steffen. Characterization of addressability by simultaneous randomized benchmarking. *Physical Review Letters*, 109(24):240504, 2012, 1204.6308. doi:10.1103/PhysRevLett.109.240504.

[114] X. Gao and L.-M. Duan. Efficient Representation of Quantum Many-body States with Deep Neural Networks. *Nature Communications*, 8(662), 2017, 1701.05039. doi:10.1038/s41467-017-00705-2.

[115] C. Gatti. *Design of Experiments for Reinforcement Learning*. Springer International Publishing, 1 edition, 2015. doi:10.1007/978-3-319-12197-0.

[116] C. J. Gatti, M. J. Embrechts, and J. D. Linton. An empirical analysis of reinforcement learning. In *Proceedings of the 21st European Symposium on Artifcial Neural Networks, Computational Intelligence and Machine Learning*, number April, pages 221–226, Bruges, Belgium, 2013.

[117] S. Glancy, E. Knill, and M. Girard. Gradient-based stopping rules for maximum-likelihood quantum-state tomography. *New Journal of Physics*, 14:095017, 2012, 1205.4043. doi:10.1088/1367-2630/14/9/095017.

[118] A. Go. Papers - Physics & Machine Learning, 2018. URL https://physicsml.github.io/pages/papers.html.

[119] I. Goodfellow, Y. Bengio, A. Courville, and F. Bach. *Deep Learning*. MIT Press, 2016.

[120] V. Gorini, A. Kossakowski, and E. C. G. Sudarshan. Completely positive dynamical semigroups of N-level systems. *Journal of Mathematical Physics*, 17(5):821–825, 1976. doi:10.1063/1.522979.

[121] D. Gottesman. *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology, 1997. URL http://arxiv.org/abs/quant-ph/9705052.

[122] R. G. Gould. The LASER, light amplification by stimulated emission of radiation. In *The Ann Arbor conference on optical pumping, the University of Michigan*, page 128, 1959.

[123] C. Granade, J. Combes, and D. G. Cory. Practical Bayesian tomography. *New Journal of Physics*, 18:033024, 2016, 1509.03770. doi:10.1088/1367-2630/18/3/033024.

[124] C. E. Granade, C. Ferrie, N. Wiebe, and D. G. Cory. Robust online Hamiltonian learning. *New Journal of Physics*, 14:103013, 2012, 1207.1655. doi:10.1088/1367-2630/14/10/103013.

[125] D. Greenbaum. Introduction to Quantum Gate Set Tomography. *arXiv*, 1509.02921.

[126] D. Gross, Y. K. Liu, S. T. Flammia, S. Becker, and J. Eisert. Quantum state tomography via compressed sensing. *Physical Review Letters*, 105(15):150401, 2010, 0909.3304. doi:10.1103/PhysRevLett.105.150401.

[127] L. K. Grover. A fast quantum mechanical algorithm for database search. In *STOC '96 Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, Philadelphia, Pennsylvania, USA, 1996. doi:10.1145/237814.237866.

[128] G. Grynberg, A. Aspect, and C. Fabre. *Introduction to Quantum Optics*. Cambridge University Press, 2010.

[129] O. Gühne, C. Y. Lu, W. B. Gao, and J. W. Pan. Toolbox for entanglement detection and fidelity estimation. *Physical Review A*, 76(3):1–4, 2007, 0706.2432. doi:10.1103/PhysRevA.76.030305.

[130] R. S. Gupta and M. J. Biercuk. Machine learning for predictive estimation of qubit dynamics subject to dephasing. *Physical Review Applied*, 9(6):64042, 2017, 1712.01291. doi:10.1103/PhysRevApplied.9.064042.

[131] M. Guta, J. Kahn, R. Kueng, and J. A. Tropp. Fast state tomography with optimal error bounds. *arXiv*, 1809.11162.

[132] M. Guta, T. Kypraios, and I. Dryden. Rank-based model selection for multiple ions quantum tomography. *New Journal of Physics*, 14, 2012, 1206.4032. doi:10.1088/1367-2630/14/10/105002.

[133] M. Guti, C. Smith, L. Lulushi, S. Janardan, and K. R. Brown. Errors and pseudothresholds for incoherent and coherent noise. *Physical Review A*, 94(4):042338, 2016. doi:10.1103/PhysRevA.94.042338.

[134] I. Guyon, B. Boser, and V. Vapnik. Automatic Capacity Tuning of Very Large VC-Dimension Classifiers. *Advances in Neural Information Processing Systems*, 5:147–155, 1993. doi:10.1186/s12864-015-2353-z.

[135] I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research (JMLR)*, 3(3):1157–1182, 2003, 1111.6189v1. doi:10.1016/j.aca.2011.07.027.

[136] J. Haah, A. W. Harrow, Z. Ji, X. Wu, and N. Yu. Sample-Optimal Tomography of Quantum States. *IEEE Transactions on Information Theory*, 63(9):5628–5641, 2017. doi:10.1109/TIT.2017.2719044.

[137] T. Häner and D. S. Steiger. 0.5 Petabyte Simulation of a 45-Qubit Quantum Circuit. In *SC '17 Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Denver, Colorado, USA, 2017. doi:10.1145/3126908.3126947.

[138] T. Häner, D. S. Steiger, K. Svore, and M. Troyer. A software methodology for compiling quantum programs. *Quantum Science and Techonlogy*, 3(020501), 2018. doi:10.1088/2058-9565/aaa5cc.

[139] A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):1–4, 2009, 0811.3171. doi:10.1103/PhysRevLett.103.150502.

[140] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2016.

[141] V. Havlicek, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta. Supervised learning with quantum enhanced feature spaces. *arXiv*, 1804.11326.

[142] S. W. Hawking. Particle Creation by Black Holes. *Communications in Mathematical Physics*, 43(3):199–220, 1975. doi:10.1007/BF02345020.

[143] L. E. Heyfron and E. T. Campbell. An Efficient Quantum Compiler that Reduces T Count. *arXiv*, 1712.01557.

[144] J. Higgins. *An Introduction to Modern Nonparametric Statistics*. Brooks/Cole, 2004.

[145] G. E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, 2002, 1207.0580. doi:10.1162/089976602760128018.

[146] G. E. Hinton, S. Osindero, and Y.-W. Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554, 2006. doi:10.1162/neco.2006.18.7.1527.

[147] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171–1220, 2008. doi:10.1214/009053607000000677.

[148] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933. doi:10.1037/h0071325.

[149] Z. Hradil. Quantum State Estimation. *Physical Review A*, 55(3):R1561–R1564, 1997, 9609012v1. doi:10.1103/PhysRevA.55.R1561.

[150] P. Huembeli, A. Dauphin, P. Wittek, and C. Gogolin. Automated discovery of characteristic features of phase transitions in many-body localization. *arXiv*, 1806.00419.

[151] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science and Engineering*, 9(3):90–95, 2007. doi:10.1109/MCSE.2007.55.

[152] K. Husimi. Some Formal Properties of the Density Matrix. In *Proceedings of the Physico-Mathematical Society of Japan*, volume 22, pages 264–314, 1940. doi:10.11429/ppmsj1919.22.4_264.

[153] S. D. Huver, C. F. Wildfeuer, and J. P. Dowling. Entangled Fock states for robust quantum optical metrology, imaging, and sensing. *Physical Review A*, 78(6):1–5, 2008, 0805.0296. doi:10.1103/PhysRevA.78.063828.

[154] I. D. Ivonovic. Geometrical description of quantal state determination. *Journal of Physics A*, 14:3241–3245, 1981. doi:10.1088/0305-4470/14/12/019.

[155] D. F. V. James, P. G. Kwiat, W. J. Munro, and A. G. White. Measurement of qubits. *Physical Review A*, 64(5):052312, oct 2001. doi:10.1103/PhysRevA.64.052312.

[156] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, 7 edition, 2017.

[157] A. Jamiołkowski. Linear transformations which preserve trace and positive semidefiniteness of operators. *Reports on Mathematical Physics*, 3(4):275–278, 1972. doi:10.1016/0034-4877(72)90011-0.

[158] P. Jeganathan. On the Asymptotic Theory of Estimation When the Limit of the Log-Likelihood Ratios Is Mixed Normal. *The Indian Journal of Statistics, Series A*, 44(2):173–212, 1982. URL https://www.jstor.org/stable/25050307.

[159] I. Jolliffe. *Principal Component Analysis*. Springer-Verlag New York, 2 edition, 2002. doi:10.1007/b98835.

[160] J. Joo, W. J. Munro, and T. P. Spiller. Quantum metrology with entangled coherent states. *Physical Review Letters*, 107(8):1–4, 2011, 1101.5044. doi:10.1103/PhysRevLett.107.083601.

[161] S. Jordan. The Quantum Algorithm Zoo. URL https://math.nist.gov/quantum/zoo/.

[162] S. P. Jordan. Fast Quantum Algorithm for Numerical Gradient Estimation. *Physical Review Letters*, 95, 2005. doi:10.1103/PhysRevLett.95.050501.

[163] A. Kalev, C. H. Baldwin, and I. H. Deutsch. The power of being positive: Robust state estimation made possible by quantum mechanics. *Physical Review A*, 93(5):052105, 2016, 1511.01433. doi:10.1103/PhysRevA.93.052105.

[164] A. Kalev, R. L. Kosut, and I. H. Deutsch. Quantum tomography protocols with positivity are compressed sensing protocols. *npj Quantum Information*, 1:15018, 2015, 1502.00536. doi:10.1038/npjqi.2015.18.

[165] J. M. Kanter and K. Veeramachaneni. Deep Feature Synthesis : Towards Automating Data Science Endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Paris, France, 2015. IEEE. doi:10.1109/DSAA.2015.7344858.

[166] R. E. Kass and A. E. Raftery. Bayes Factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995. doi:10.1080/01621459.1995.10476572.

[167] J. Kelly. A Preview of Bristlecone, Google's New Quantum Processor, 2018. URL `https://research.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html`.

[168] A. Kenfack and K. Yczkowski. Negativity of the Wigner function as an indicator of non-classicality. *Journal of Optics B: Quantum and Semiclassical Optics*, 6(10):396–404, oct 2004. doi:10.1088/1464-4266/6/10/003.

[169] S. Khatri, R. LaRose, A. Poremba, L. Cincio, A. T. Sornborger, and P. J. Coles. Quantum-assisted quantum compiling. *arXiv*, 1807.00800.

[170] J. Kiefer. Sequential Minimax Search for a Maximum. *Proceedings of the American Mathematical Society*, 4:502–506, 1952. doi:10.1090/S0002-9939-1953-0055639-3.

[171] S. Kimmel, M. P. da Silva, C. A. Ryan, B. R. Johnson, and T. Ohki. Robust Extraction of Tomographic Information via Randomized Benchmarking. *Physical Review X*, 4(1):011050, 2014. doi:10.1103/PhysRevX.4.011050.

[172] S. Kimmel, G. H. Low, and T. J. Yoder. Robust Single-Qubit Process Calibration via Robust Phase Estimation. *Physical Review A*, 92(6):062315, 2015, 1502.02677. doi:10.1103/PhysRevA.92.062315.

[173] C. Kittel. *Introduction to Solid State Physics*. Wiley, 8th edition, 2004.

[174] M. Kliesch, R. Kueng, J. Eisert, and D. Gross. Guaranteed recovery of quantum processes from few measurements. *arXiv*, 1701.03135.

[175] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland. Randomized benchmarking of quantum gates. *Physical Review A*, 77(1):012307, 2008, 0707.0963. doi:10.1103/PhysRevA.77.012307.

[176] E. Knill, G. J. Milburn, and R. Laflamme. A scheme for efficient quantum computation with linear optics. *Nature*, 409:46–52, 2001, 1208.4575. doi:10.1038/35051009.

[177] L. Knips, C. Schwemmer, N. Klein, J. Reuter, G. Tóth, and H. Weinfurter. How long does it take to obtain a physical density matrix? *arXiv*, 1512.06866.

[178] W. Kohn and L. J. Sham. Self-Consistent Equations Including Exchange and Correlation Effects. *Physical Review*, 140(4A):1133–1138, 1965. doi:10.1103/PhysRev.140.A1133.

[179] L. F. Kozachenko and N. N. Leonenko. Sample Estimate of the Entropy of a Random Vector. *Problems of Information Transmission*, 23(2):95–101, 1987.

[180] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964. doi:10.1007/BF02289565.

[181] R. Kueng and D. Gross. RIPless compressed sensing from anisotropic measurements. *Linear Algebra and Its Applications*, 441:110–123, 2014, 1205.1423. doi:10.1016/j.laa.2013.04.018.

[182] R. Kueng, D. M. Long, A. C. Doherty, and S. T. Flammia. Comparing Experiments to the Fault-Tolerance Threshold. *Physical Review Letters*, 117(17):170502, 2016, 1510.05653. doi:10.1103/PhysRevLett.117.170502.

[183] R. Kueng, H. Rauhut, and U. Terstiege. Low rank matrix recovery from rank one measurements. *Applied and Computational Harmonic Analysis*, 42(1):88–116, 2017, 1410.6913. doi:10.1016/j.acha.2015.07.007.

[184] P. G. Kwiat, A. G. White, J. R. Mitchell, O. Nairz, G. Weihs, H. Weinfurter, and A. Zeilinger. High-Efficiency Quantum Interrogation Measurements via the Quantum Zeno Effect. *Physical Review Letters*, 83(23):4725–4728, dec 1999. doi:10.1103/PhysRevLett.83.4725.

[185] A. J. Landahl, J. T. Anderson, and P. R. Rice. Fault-tolerant quantum computing with color codes. *arXiv*, 1108.5738.

[186] R. Landauer. Irreversibility and Heat Generation in the Computing Process. *IBM Journal of Research and Development*, 5(3):183–191, 1961. doi:10.1147/rd.53.0183.

[187] N. K. Langford. Errors in quantum tomography: Diagnosing systematic versus statistical errors. *New Journal of Physics*, 15(035003), 2013, 1212.2982. doi:10.1088/1367-2630/15/3/035003.

[188] B. P. Lanyon, C. Maier, M. Holzäpfel, T. Baumgratz, C. Hempel, P. Jurcevic, I. Dhand, A. S. Buyskikh, A. J. Daley, M. Cramer, M. B. Plenio, R. Blatt, and C. F. Roos. Efficient tomography of a quantum many-body system. *Nature Physics*, 13(12):1158–1162, 2017, 1612.08000. doi:10.1038/nphys4244.

[189] L. Le Cam. On the asymptotic theory of estimation and testing hypotheses. In J. Neyman, editor, *Proceedings of the Third Berkeley Symposium on*

Mathematical Statistics and Probability, pages 129–156, 1956. URL https://projecteuclid.org/euclid.bsmsp/1200501652.

[190] L. Le Cam. On the Assumptions Used to Prove Asymptotic Normality of Maximum Likelihood Estimates. *Annals of Mathematical Statistics*, 41(3):802–828, 1970. doi:10.1214/aoms/1177696960.

[191] U. Leonhardt, M. Munroe, T. Kiss, T. Richter, and M. G. Raymer. Sampling of photon statistics and density matrix using homodyne detection. *Optics Communications*, 127(1-3):144–160, 1996. doi:10.1016/0030-4018(96)00061-2.

[192] U. Leonhardt and H. Paul. Measuring the Quantum State of Light. *Progress in Quantum Electronics*, 19(2):89–130, 1995. doi:10.1016/0079-6727(94)00007-L.

[193] K. K. Likharev. Classical and Quantum Limitations on Energy Consumption in Computation. *International Journal of Theoretical Physics*, 21(3-4):311–326, 1982. doi:10.1007/BF01857733.

[194] G. Lindblad. On the generators of quantum dynamical semigroup. *Communications in Mathematical Physics*, 48(2):119–130, 1976. URL https://projecteuclid.org/euclid.cmp/1103899849.

[195] Y.-K. Liu. Universal low-rank matrix recovery from Pauli measurements. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1638–1646. Curran Associates, Inc., 2011, 1103.2816.

[196] S. Lloyd. A theory of quantum gravity based on quantum computation. 0501135v9. URL https://arxiv.org/abs/quant-ph/0501135.

[197] S. Lloyd, S. Garnerone, and P. Zanardi. Quantum algorithms for topological and geometric analysis of data. *Nature Communications*, 7:1–7, 2016, 1408.3106. doi:10.1038/ncomms10138.

[198] S. Lloyd, M. Mohseni, and P. Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv*, 1307.0411.

[199] A. I. Lvovsky. Iterative Maximum-Likelihood Reconstruction in Quantum Homodyne Tomography. *Journal of Optics B*, 6(6):556–559, jun 2004. doi:10.1088/1464-4266/6/6/014.

[200] A. I. Lvovsky, H. Hansen, T. Aichele, O. Benson, J. Mlynek, and S. Schiller. Quantum State Reconstruction of the Single-Photon Fock State. *Physical Review Letters*, 87(5):050402, 2001. doi:10.1103/PhysRevLett.87.050402.

[201] A. I. Lvovsky and M. G. Raymer. Continuous-variable optical quantum-state tomography. *Reviews of Modern Physics*, 81(1):299–332, 2009, 0511044. doi:10.1103/RevModPhys.81.299.

[202] E. Magesan, J. M. Gambetta, and J. Emerson. Characterizing quantum gates via randomized benchmarking. *Physical Review Letters*, 85(4):042311, 2012. doi:10.1103/PhysRevA.85.042311.

[203] A. Magnus. *De Mineralibus*. Christophorus de Canibus, Pavia, Italy, 1491.

[204] F. Mallet, M. a. Castellanos-Beltran, H. S. Ku, S. Glancy, E. Knill, K. D. Irwin, G. C. Hilton, L. R. Vale, and K. W. Lehnert. Quantum State Tomography of an Itinerant Squeezed Microwave Field. *Physical Review Letters*, 106(22):220502, jun 2011, 1012.0007. doi:10.1103/PhysRevLett.106.220502.

[205] L. Mandel and E. Wolf. *Optical Coherence and Quantum Optics*. Cambridge University Press, 1st edition, 1995.

[206] S. Mandrà, G. G. Guerreschi, and A. Aspuru-Guzik. Faster than classical quantum algorithm for dense formulas of exact satisfiability and occupation problems. *New Journal of Physics*, 18, 2016. doi:10.1088/1367-2630/18/7/073003.

[207] I. L. Markov, A. Fatima, S. Isakov, and S. Boixo. Quantum Supremacy Is Both Closer and Farther than It Appears. *arXiv*, 1807.10749.

[208] A. H. Maslow. *Psychology of Science: A Reconnaissance*. Gateway Editions, 1st edition, 1969.

[209] M. B. McCoy and J. A. Tropp. From Steiner Formulas for Cones to Concentration of Intrinsic Volumes. *Discrete and Computational Geometry*, 51(4):926–963, 2014, 1308.5265. doi:10.1007/s00454-014-9595-4.

[210] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. doi:10.1007/BF02478259.

[211] W. McKinney. Data Structures for Statistical Computing in Python. In S. van der Walt and J. Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51–56, 2010. URL `http://conference.scipy.org/proceedings/scipy2010/mckinney.html`.

[212] C. Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, 1990. doi:10.1109/5.58356.

[213] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab. A high-bias, low-variance introduction to Machine Learning for physicists. *arXiv*, 1803.08823.

[214] S. T. Merkel, J. M. Gambetta, J. A. Smolin, S. Poletto, A. D. Corcoles, B. R. Johnson, C. A. Ryan, and M. Steffen. Self-consistent quantum process tomography. *Physical Review A*, 87(6):062119, 2013, 1211.0322. doi:10.1103/PhysRevA.87.062119.

[215] A. Meurer. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103, 2017. doi:10.7717/peerj-cs.103.

[216] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[217] C. Moller and M. S. Plesset. Note on an Approximation Treatment for Many-Electron Systems. *Physical Review*, 46(7):618–622, 1934. doi:10.1103/PhysRev.46.618.

[218] A. Montanaro. Quantum algorithms: an overview. *npj Quantum Information*, 2:1–8, 2015, 1511.04206. doi:10.1038/npjqi.2015.23.

[219] T. Moroder, M. Kleinmann, P. Schindler, T. Monz, O. Guhne, and R. Blatt. Certifying systematic errors in quantum experiments. *Physical Review Letters*, 110(18):180401, 2013. doi:10.1103/PhysRevLett.110.180401.

[220] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm. Simple Pulses for Elimination of Leakage in Weakly Nonlinear Qubits. *Physical Review Letters*, 103(11):110501, 2009, 0901.0534. doi:10.1103/PhysRevLett.103.110501.

[221] J. Neyman and E. S. Pearson. On the Problem of the Most Efficient Tests of Statistical Hypotheses. *Philosophical Transactions of the Royal Society of London*, 231:289–337, 1933. doi:10.1098/rsta.1933.0009.

[222] E. Nielsen, J. Gross, T. L. Scholten, J. Gross, K. Rudinger, and T. Proctor. pyGSTio/pyGSTi, 2017. doi:10.5281/zenodo.594712.

[223] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 10th edition, 2011.

[224] J. Nocdeal and S. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[225] A. B. Novikoff. On Convergence Proofs on Perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, pages 615–622, New York, New York, USA, 1962. Polytechnic Institute of Brooklyn.

[226] T. E. Oliphant. Python for Scientific Computing. *Computing in Science and Engineering*, 9(3):10–20, 2007. doi:10.1109/MCSE.2007.58.

[227] J. P. Olson, K. R. Motes, P. M. Birchall, N. M. Studer, M. Laborde, T. Moulder, P. P. Rohde, and J. P. Dowling. Linear optical quantum metrology with single photons: Experimental errors, resource counting, and quantum Cramér-Rao bounds. *Physical Review A*, 96(1):013810, 2017, 1610.07128. doi:10.1103/PhysRevA.96.013810.

[228] T. Opatrny, D. G. Welsch, and V. W. Least-squares inversion for density-matrix reconstruction. *Physical Review A*, 56(3):1788–1799, 1997, 9703026. doi:10.1103/PhysRevA.56.1788.

[229] M. G. A. Paris and J. Rehacek, editors. *Quantum State Estimation*. Springer, Berlin-Heidelberg, 2004. doi:10.1007/b98673.

[230] W. Pauli. Die allgemeinen Prinzipien der Wellenmechanik. In H. Bethe, F. Hund, N. F. Mott, W. Pauli, A. Rubinowicz, G. Wentzel, and A. Smekal, editors, *Quantentheorie*, pages 83–272. Springer-Verlag Berlin Heidelberg, 1933. doi:10.1007/978-3-642-52619-0_2.

[231] K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. doi:10.1080/14786440109462720.

[232] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, T. Magerlein, E. Solomonik, and R. Wisnieff. Breaking the 49-Qubit Barrier in the Simulation of Quantum Circuits. *arXiv*, 1710.05867.

[233] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn:

Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. doi:10.5281/zenodo.49911.

[234] F. Pérez and B. E. Granger. IPython: A System for Interactive Scientific Computing. *Computing in Science and Engineering*, 9(3):21–29, 2007. doi:10.1109/MCSE.2007.53.

[235] E. Polak and G. Ribiere. Note sur la convergence de methodes de directions conjugees. *Rev. Francaise Informat Recherche Operationelle*, 3(R1):35–43, 1969.

[236] J. Preskill. Do Black Holes Destroy Information? *arXiv*. URL http://arxiv.org/abs/hep-th/9209058.

[237] J. Preskill. Quantum Computing in the NISQ era and beyond. *arXiv*, 1801.00862. URL http://arxiv.org/abs/1801.00862.

[238] J. Preskill. Reliable quantum computers. *Proceedings of the Royal Society of London A*, 454:385–410, 1998. URL http://www.theory.caltech.edu/{~}preskill/pubs/preskill-1998-reliable.pdf.

[239] J. Preskill. Quantum Computing and the Entanglement Frontier. In *25th Solvay Conference on Physics*, Brussels, 2012. 1203.5813.

[240] J. Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2, 2018, 1801.00862. doi:10.22331/q-2018-08-06-79.

[241] T. Proctor, K. Rudinger, K. Young, M. Sarovar, and R. Blume-Kohout. What randomized benchmarking actually measures. *Physical Review Letters*, 119(13):130502, 2017, 1702.01853. doi:10.1103/PhysRevLett.119.130502.

[242] T. J. Proctor, A. Carignan-Dugas, K. Rudinger, E. Nielsen, R. Blume-Kohout, and K. Young. Direct randomized benchmarking for multi-qubit devices. *arXiv*, 1807.07975.

[243] E. M. Purcell, H. C. Torrey, and R. V. Pound. Resonance Absorption by Nuclear Magnetic Moments in a Solid. *Physical Review*, 69(1-2):37–38, 1946. doi:10.1103/PhysRev.69.37.

[244] J. Qi and H. K. Ng. Randomized benchmarking does not measure average infidelity of gates. *arXiv*, 1805.10622.

[245] T. C. Ralph, A. Gilchrist, G. J. Milburn, W. J. Munro, and S. Glancy. Quantum computation with optical coherent states. *Physical Review A*, 68(4):042319, 2003, 0306004v1. doi:10.1103/PhysRevA.68.042319.

[246] P. Rashinkar, P. Paterson, and L. Singh. *System-on-a-Chip Verification: Methodology and Techniques*. Springer Science & Business Media, 2007.

[247] M. G. Raymer. Measuring the quantum mechanical wave function. *Contemporary Physics*, 38(5):343–355, 1997. doi:10.1080/001075197182315.

[248] M. G. Raymer, M. Beck, and D. F. McAlister. Complex Wave-Field Reconstruction Using Phase-Space Tomography. *Physical Review Letters*, 72(8):1137–1140, 1994. doi:10.1103/PhysRevLett.72.1137.

[249] N. Reid and D. R. Cox. On Some Principles of Statistical Inference. *International Statistical Review*, 83(2):293–308, 2015. doi:10.1111/insr.12067.

[250] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer. Elucidating reaction mechanisms on quantum computers. *PNAS*, 114(29):7555–7560, 2017. doi:10.1073/pnas.1619152114.

[251] J. M. Renes, R. Blume-Kohout, A. J. Scott, and C. M. Caves. Symmetric informationally complete quantum measurements. *Journal of Mathematical Physics*, 45(6):2171–2180, 2011, 0310075. doi:10.1063/1.1737053.

[252] C. Rigetti. Introducing Forest 1.0. URL `https://medium.com/rigetti/introducing-forest-f2c806537c6d`.

[253] C. Rigetti. The Rigetti 128-qubit chip and what it means for quantum, 2018. URL https://medium.com/rigetti/the-rigetti-128-qubit-chip-and-what-it-means-for-quantum-df757d1b71ea.

[254] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[255] D. Ristè, M. P. Silva, C. A. Ryan, A. W. Cross, A. D. Córcoles, J. A. Smolin, J. M. Gambetta, J. M. Chow, and B. R. Johnson. Demonstration of quantum advantage in machine learning. *npj Quantum Information*, 3, 2017. doi:10.1038/s41534-017-0017-3.

[256] R. T. Rockafellar and R. J.-B. Wets. *Variational Analysis*. Springer-Verlag Berlin Heidelberg, 1st edition, 1998. doi:10.1007/978-3-642-02431-3.

[257] A. V. Rodionov, A. Veitia, R. Barends, J. Kelly, D. Sank, J. Wenner, J. M. Martinis, R. L. Kosut, and A. N. Korotkov. Compressed sensing quantum process tomography for superconducting quantum gates. *Physical Review B*, 90(14):144504, 2014, 1407.0761. doi:10.1103/PhysRevB.90.144504.

[258] F. Rosenblatt. The Perceptron - A Perceiving and Recognizing Automaton. Technical report, Cornell Aeronautical Laboratory, Buffalo, New York, 1957.

[259] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. doi:10.1037/h0042519.

[260] C. W. Rosenbrock, E. R. Homer, G. Csányi, and G. L. W. Hart. Discovering the Building Blocks of Atomic Systems using Machine Learning. *npj Computational Materials*, 3, 2017, 1703.06236. doi:10.1038/s41524-017-0027-x.

[261] B. C. Ross. Mutual Information between Discrete and Continuous Data Sets. *PLOS One*, 9(2), 2014. doi:10.1371/journal.pone.0087357.

[262] G. G. Roussas and D. Bhattacharya. Revisiting Local Asymptotic Normality (LAN) and Passing on to Local Asymptotic Mixed Normality (LAMN) and Local Asymptotic Quadratic (LAQ) Experiments. In *Advances in Directional and Linear Statistics*, pages 253–280. Springer Physica-Verlag HD, 2010. doi:10.1007/978-3-7908-2628-9_17.

[263] V. P. Roychowdhury, K.-Y. Siu, and T. Kailath. Classification of Linearly Nonseparable Patterns by Linear Threshold Elements. *IEEE Transactions on Neural Networks*, 6(2):318–331, 1995. doi:10.1109/72.363468.

[264] K. Rudinger, S. Kimmel, D. Lobser, and P. Maunz. Experimental Demonstration of a Cheap and Accurate Phase Estimation. *Physical Review Letters*, 118(19):190502, 2017, 1702.01763. doi:10.1103/PhysRevLett.118.190502.

[265] Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007. doi:10.1093/bioinformatics/btm344.

[266] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959. doi:10.1147/rd.33.0210.

[267] T. L. Scholten and R. Blume-Kohout. Behavior of the maximum likelihood in quantum state tomography. *New Journal of Physics*, 20(023050), 2018, 1609.04385. doi:10.1088/1367-2630/aaa7e2.

[268] M. Schuld, M. Fingerhuth, and F. Petruccione. Implementing a distance-based classifier with a quantum interference circuit. *Europhysics Letters*, 119(60002), 2017, 1703.10793. doi:10.1209/0295-5075/119/60002.

[269] M. Schuld, I. Sinayskiy, and F. Petruccione. An introduction to quantum

machine learning. *Contemporary Physics*, 56(2):172–185, 2015, 1409.3097. doi:10.1080/00107514.2014.964942.

[270] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. doi:10.1214/aos/1176344136.

[271] L. Schwarz and S. J. van Enk. Error models in quantum computation: an application of model selection. *Physical Review A*, 88(3):032318, 2013, 1307.0858. doi:10.1103/PhysRevA.88.032318.

[272] M. O. Scully and M. S. Zubairy. *Quantum Optics*. Cambridge University Press, Cambridge, 1997. doi:10.1017/CBO9780511813993.

[273] S. Sheldon, L. S. Bishop, E. Magesan, S. Filipp, J. M. Chow, and J. M. Gambetta. Characterizing errors on qubit operations via iterative randomized benchmarking. *Physical Review A*, 93(1):012301, 2016, 1504.06597. doi:10.1103/PhysRevA.93.012301.

[274] R. N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. I. *Psychometrika*, 27(2):125–140, 1962. doi:10.1007/BF02289630.

[275] R. N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. II. *Psychometrika*, 27(3):219–246, 1962. doi:10.1007/BF02289621.

[276] J. R. Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 1994. URL `https://www.cs.cmu.edu/{~}quake-papers/painless-conjugate-gradient.pdf`.

[277] P. W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52(4):R2493–R2496, 1995. doi:10.1103/PhysRevA.52.R2493.

[278] P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. doi:10.1137/S0097539795293172.

[279] A. E. Siegman. *Lasers*. University Science Books, 1986.

[280] D. R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997. doi:10.1137/S0097539796298637.

[281] D. T. Smithey, M. Beck, M. G. Raymer, and A. Faridani. Measurement of the Wigner Distribution and the Density Matrix of a Light Mode Using Optical Homodyne Tomography: Application to Squeezed States and the Vacuum. *Physical Review Letters*, 70(9):1244–1247, 1993. doi:10.1103/PhysRevLett.70.1244.

[282] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McLelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 194–281. MIT Press, 1986. URL https://dl.acm.org/citation.cfm?id=104279.104290.

[283] J. A. Smolin, J. M. Gambetta, and G. Smith. Efficient method for computing the maximum-likelihood quantum state from measurements with additive gaussian noise. *Physical Review Letters*, 108(7):070502, 2012, 1106.5458. doi:10.1103/PhysRevLett.108.070502.

[284] N. Spagnolo, C. Vitelli, M. Bentivegna, D. J. Brod, A. Crespi, F. Flamini, S. Giacomini, G. Milani, R. Ramponi, P. Mataloni, R. Osellame, E. F. Galvão,

and F. Sciarrino. Experimental validation of photonic boson sampling. *Nature Photonics*, 8:615–620, 2014. doi:10.1038/nphoton.2014.135.

[285] D. J. Spiegelhalter, N. G. Best, B. P. Carlin, and A. Van Der Linde. Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(4), 2002. doi:10.1111/1467-9868.00353.

[286] M. Steffen, M. Ansmann, R. C. Bialczak, N. Katz, E. Lucero, R. McDermott, M. Neeley, E. M. Weig, A. N. Cleland, and J. M. Martinis. Measurement of the Entanglement of Two Superconducting Qubits via State Tomography. *Science*, 313(5792):1423–1425, 2006. doi:10.1126/science.1130886.

[287] A. Steffens, C. Riofrio, W. McCutcheon, I. Roth, B. A. Bell, A. McMillan, M. S. Tame, J. G. Rarity, and J. Eisert. Experimentally exploring compressed sensing quantum tomography. *arXiv*, 2016, 1611.01189.

[288] M. Suchara, J. Kubiatowicz, A. Faruque, F. T. Chong, C.-y. Lai, and G. Paz. QuRE : The Quantum Resource Estimator Toolbox. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pages 419–426, 2013. doi:10.1109/ICCD.2013.6657074.

[289] D. Suess, Ł. Rudnicki, T. O maciel, and D. Gross. Error regions in quantum state tomography : computational complexity caused by geometry of quantum states. *New Journal of Physics*, 19(093013), 2017, 1608.00374. doi:10.1088/1367-2630/aa7ce9.

[290] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, 2nd edition, 1998.

[291] R. Suttor. Quantum Ready now, Quantum Advantage tomorrow. URL https://medium.com/@rssutor/quantum-ready-now-quantum-advantage-tomorrow-14739a28c6f5.

[292] T. Tao and V. Vu. Random Matrices: Sharp Concentration of Eigenvalues. *Random Matrices: Theory and Applications*, 2(3), 2013, 1201.4789. doi:10.1142/S201032631350007X.

[293] Y. S. Teo. *Numerical Estimation Schemes for Quantum Tomography*. PhD thesis, National University of Singapore, 2013, 1302.3399.

[294] M. Tillmann, B. Dakić, R. Heilmann, S. Nolte, A. Szameit, and P. Walther. Experimental boson sampling. *Nature Photonics*, 7:540–544, may 2013. doi:10.1038/nphoton.2013.102.

[295] P.-S. Ting, C.-C. Tu, P.-Y. Chen, Y.-Y. Lo, and S.-M. Cheng. FEAST: An Automated Feature Selection Framework for Compilation Tasks. *arXiv*, 1610.09543.

[296] W. S. Torgerson. *Theory and Methods of Scaling*. Wiley, New York, New York, USA, 1958.

[297] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo. Neural-network quantum state tomography. *Nature Physics*, 14:447–450, 2018. doi:10.1038/s41567-018-0048-5.

[298] G. Torlai and R. G. Melko. Latent Space Purification via Neural Density Operators. *Physical Review Letters*, 120(24):240503, 2018, 1801.09684. doi:10.1103/PhysRevLett.120.240503.

[299] J. A. Tropp. Convex recovery of a structured signal from independent random linear measurements. In G. Pfander, editor, *Sampling Theory, A Renaissance*. Springer International Publishing Switzerland, 2015, 1405.1102. doi:10.1007/978-3-319-19749-4_2.

[300] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42:230–265, 1937. doi:10.1112/plms/s2-42.1.230.

[301] A. M. Turing. Computing Machinery and Intelligence. *Mind: A Quarterly Review of Psychology and Philosophy*, 59(236):433–460, 1950.

[302] A. W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 2000.

[303] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, 13(2):22–30, 2011, 1102.1523. doi:10.1109/MCSE.2011.37.

[304] S. J. van Enk and R. Blume-Kohout. When quantum tomography goes wrong: Drift of quantum sources and other errors. *New Journal of Physics*, 15:025024, 2013, 1302.0932. doi:10.1088/1367-2630/15/2/025024.

[305] S. J. van Enk, N. Lutkenhaus, and H. J. Kimble. Experimental procedures for entanglement verification. *Physical Review A*, 75(5):052318, 2007, 0611219. doi:10.1103/PhysRevA.75.052318.

[306] E. P. Van Nieuwenburg, Y. H. Liu, and S. D. Huber. Learning phase transitions by confusion. *Nature Physics*, 13(5):435–439, 2017, 1610.02048. doi:10.1038/nphys4037.

[307] G. van Rossum. Python Language Reference, 1995. URL `http://www.python.org`.

[308] V. N. Vapnik and A. Chervonenkis. A note on one class of perceptrons. *Automation and Remote Control*, 25, 1964.

[309] V. Veitch, C. Ferrie, D. Gross, and J. Emerson. Negative quasi-probability as a resource for quantum computation. *New Journal of Physics*, 14(113011), 2012, 1201.1256. doi:10.1088/1367-2630/14/11/113011.

[310] K. Vogel and H. Risken. Determination of quasiprobability distributions in terms of probability distributions for the rotated quadrature phase. *Physical Review A*, 40(5):2847–2849, sep 1989. doi:10.1103/PhysRevA.40.2847.

[311] R. Vogler. Testing for Linear Separability with Linear Programming in R, 2014. URL https://www.joyofdata.de/blog/testing-linear-separability-linear-programming-r-glpk/.

[312] C. Vu. IBM Announces Advances to IBM Quantum Systems & Ecosystem, 2017. URL https://www-03.ibm.com/press/us/en/pressrelease/53374.wss.

[313] J. Wallman, C. Granade, R. Harper, and S. T. Flammia. Estimating the coherence of noise. *New Journal of Physics*, 17:113020, 2015, 1503.07865. doi:10.1088/1367-2630/17/11/113020.

[314] J. J. Wallman and J. Emerson. Noise tailoring for scalable quantum computation via randomized compiling. *Physical Review A*, 94(052325), 2016. doi:10.1103/PhysRevA.94.052325.

[315] A. Warshel and M. Levitt. Theoretical Studies of Enzymic Reactions: Dielectric, Electrostatic, and Steric Stabilization of the Carbonium Ion in the Reaction of Lysozyme. *Journal of Molecular Biology*, 103:227–249, 1976. doi:10.1016/0022-2836(76)90311-9.

[316] M. Waskom. seaborn, 2016. doi:10.5281/zenodo.592845.

[317] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer New York, 2004. doi:10.1007/978-0-387-21736-9.

[318] D.-G. Welsch, W. Vogel, and T. Opatrny. Homodyne Detection and Quantum State Reconstruction. *Progress in Optics*, 39:63–211, jul 1999, 0907.1353. doi:10.1016/S0079-6638(08)70389-5.

[319] S. J. Wetzel. Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders. *Physical Review E*, 96(2):1–11, 2017, 1703.02435. doi:10.1103/PhysRevE.96.022140.

[320] J. A. Wheeler. Information, physics, quantum: the search for links. In *Proceedings III International Symposium on Foundations of Quantum Mechanics*, pages 354–368, Tokyo, Japan, 1989.

[321] E. P. Wigner. On the Quantum Correction for Thermodynamic Equilibrium. *Physical Review*, 40(5):749–759, 1932. doi:10.1103/PhysRev.40.749.

[322] E. P. Wigner. On the Distribution of The Roots of Certain Symmetric Matrices. *Annals of Mathematics*, 67(2):325–327, 1958. doi:10.2307/1970008.

[323] S. S. Wilks. The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses. *The Annals of Mathematical Statistics*, 9(1):60–62, 1938. doi:10.1214/aoms/1177732360.

[324] C. M. Wilson, J. S. Otterbach, N. Tezak, R. S. Smith, and G. E. Crooks. Quantum Kitchen Sinks. *arXiv*, 1806.08321.

[325] C. J. Wood and J. M. Gambetta. Quantification and characterization of leakage errors. *Physical Review A*, 97(3):032306, 2018, 1704.03081. doi:10.1103/PhysRevA.97.032306.

[326] W. K. Wootters and W. H. Zurek. A Single Quantum Cannot Be Cloned. *Nature*, 299:802–803, 1982. doi:10.1038/299802a0.

[327] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande. MoleculeNet: A Benchmark for Molecular Machine Learning. *Chemical Science*, 9(513), 2018, 1703.00564. doi:10.1039/C7SC02664A.

[328] D. Xia. Estimation of low rank density matrices by Pauli measurements. *arXiv*, 1610.04811.

[329] A. C.-C. Yao. Quantum Circuit Complexity. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 352–361, Palo Alto, CA, 1993. doi:10.1109/SFCS.1993.366852.

[330] J. O. S. Yin and S. J. van Enk. Information criteria for efficient quantum state estimation. *Physical Review A*, 83(6):062110, 2011, 1103.3251. doi:10.1103/PhysRevA.83.062110.

[331] A. Youssry, C. Ferrie, and M. Tomamichel. Efficient online quantum state estimation using a matrix-exponentiated gradient method. *arXiv*, 1807.01852.

[332] A. Yurtsever, M. Udell, J. A. Tropp, and V. Cevher. Sketchy Decisions : Convex Low-Rank Matrix Optimization. *arXiv*, 1702.06838.

[333] W. Zeng. Unsupervised Machine Learning on Rigetti 19Q with Forest 1.2, 2017. URL https://medium.com/rigetti/unsupervised-machine-learning-on-rigetti-19q-with-forest-1-2-39021339699.

[334] P. Zhang, H. Shen, and H. Zhai. Machine Learning Topological Invariants with Neural Networks. *Physical Review Letters*, 120(6):66401, 2018, 1708.09401. doi:10.1103/PhysRevLett.120.066401.

[335] H. Zhu. Quantum state estimation with informationally overcomplete measurements. *Physical Review A*, 90(1):012115, 2014. doi:10.1103/PhysRevA.90.012115.

[336] M. Ziman, M. Plesch, V. Bužek, and P. Štelmachovič. Process Reconstruction: From Unphysical to Physical Maps via Maximum Likelihood. *Physical Review A*, 72(2):1–5, aug 2005. doi:10.1103/PhysRevA.72.022106.

[337] W. H. Zurek. Sub-Planck structure in phase space and its relevance for quantum decoherence. *Nature*, 412:712–717, 2001. doi:10.1038/35089017.

[338] J. Řeháček, Z. Hradil, E. Knill, and A. I. Lvovsky. Diluted maximum-likelihood algorithm for quantum tomography. *Physical Review A*, 75(4):042108, apr 2007. doi:10.1103/PhysRevA.75.042108.