ABSTRACT

Title of dissertation: Fast optimization methods for machine learning, and game-theoretic models of cultural evolution

Soham De
Doctor of Philosophy, 2018

Dissertation directed by: Dr. Tom Goldstein and Dr. Dana Nau
Department of Computer Science

This thesis has two parts. In the first part, we explore fast stochastic optimization methods for machine learning.

Mathematical optimization is a backbone of modern machine learning. Most machine learning problems require optimizing some objective function that measures how well a model matches a data set, with the intention of drawing patterns and making decisions on new unseen data. The success of optimization algorithms in solving these problems is critical to the success of machine learning, and has enabled the research community to explore more complex machine learning problems that require bigger models and larger datasets.

Stochastic gradient descent (SGD) has become the standard optimization routine in machine learning, and in particular in deep neural networks, due to its impressive performance across a wide variety of tasks and models. SGD, however, can often be slow for neural networks with many layers and typically requires careful user oversight

for setting hyperparameters properly. While innovations such as batch normalization and skip connections have helped alleviate some of these issues, why such innovations are required eludes full understanding, and it is worthwhile to gain deeper theoretical insights into these problems and to consider more advanced optimization methods specifically tailored towards training large complex models.

In this part of the thesis, we review and analyze some of the recent progress made in this direction, and develop new optimization algorithms that are provably fast, significantly easier to train, and require less user oversight. Then, we will discuss the theory of quantized networks, which use low-precision weights to compress and accelerate neural networks, and when/why they are trainable. Finally, we discuss some recent results on how the convergence of SGD is affected by the architecture of neural nets, and we show using theoretical analysis that wide networks train faster than narrow nets, and deeper networks train slower than shallow nets – an effect often observed in practice.

In the second part of the thesis, we study the evolution of cultural norms in human societies using game-theoretic models, drawing from research in cross-cultural psychology. Understanding human behavior and modeling how cultural norms evolve in different human societies is vital for designing policies and avoiding conflicts around the world. In this part, we explore ways to use computational game-theoretic techniques, and in particular evolutionary game-theoretic (EGT) models, to gain insight into why different human societies have different norms and behaviors.

We first describe an evolutionary game-theoretic model to study how norms change in a society, based on the idea that different strength of norms in societies translate to

different game-theoretic interaction structures and incentives. We identify conditions that determine when societies change their existing norms, when they are resistant to such change, and how this depends on the strength of norms in a society.

Next, we extend this study to analyze the evolutionary relationships between the tendency to conform and how quickly a population reacts when conditions make a change in norm desirable. Our analysis identifies conditions when a tipping point is reached in a population, causing norms to change rapidly.

Next we study conditions that affect the existence of group-biased behavior among humans (i.e., favoring others from the same group, and being hostile towards others from different groups). Using an evolutionary game-theoretic model, we show that out-group hostility is dramatically reduced by mobility. Technological and societal advances over the past centuries have greatly increased the degree to which humans change physical locations, and our results show that in highly mobile societies, ones choice of action is more likely to depend on what individual one is interacting with, rather than the group to which the individual belongs.

Fast optimization methods for machine learning, and game-theoretic models of cultural evolution

by

Soham De

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Advisory Committee:
Dr. Tom Goldstein, Co-Chair/Advisor
Dr. Dana S. Nau, Co-Chair/Advisor
Dr. David W. Jacobs
Dr. Michele J. Gelfand
Dr. John P. Dickerson

Acknowledgments

I am grateful to my advisors Dana Nau and Tom Goldstein for their constant support, encouragement and guidance over the years, and for giving me the freedom to pursue my own varied interests. Their brilliance and dedication have been an ongoing source of inspiration. I have been fortunate to also have the opportunity to work closely with Michele Gelfand from the Psychology department. The numerous fascinating and lively discussions with her and Dana were among the highlights during my time here.

I was lucky to have been part of an excellent department filled with many wonderful people. I would like to thank my other thesis defense committee members, David Jacobs and John Dickerson, for their insightful comments and encouragement. I would also like to thank Jodie, Sharron, Jenny and Tom Hurst, for always ensuring that everything ran smoothly in the CS department, and for going out of their way to help me on a few occasions.

This thesis would not have been possible without the help of my close collaborators Hao, Karthik, James M., Anirbit, Sohil, Abhay, Zheng, Xinyue and Patrick, to whom I am truly indebted. I would also like to thank Bhiksha Raj and Karen Livescu, with whom I was fortunate to be able to work with during my undergraduate years, and who were instrumental in developing my interests in machine learning.

I would also like to acknowledge the many friends I made while pursuing my PhD, as well as some of my older friends, all of whom made the last few years immensely enjoyable. These include Siddharth, Soumyadip, Agniv, Jia, Upamanyu, Rishov, Souvik, Piyana, Udit, Dipankar, Debdipta, Biswadip, Shawon, Arijit, Wrick, Sunandita, Kartik,

Sudha, Manaswi, Bhaskar, Pallabi, Meethu, Sankha, Prashanth, Vicky, Karol, Emmy, Amit, Prarthana, Arunima, Amrita, Aritra, Anirban and others who I am surely forgetting. I would also like to thank my uncle and aunt in Maryland who have never made me feel too far from home.

Finally I would like to thank my parents for their continued love and support, and for always encouraging me in every endeavor of my life.

Table of Contents

List of Tables

List of Figures

Chapter 1: **Introduction and organization of the thesis**

This thesis has two parts. In the first part, we explore fast stochastic optimization methods for machine learning. In the second part of the thesis, we study the evolution of cultural norms in human societies using game-theoretic models, drawing from research in cross-cultural psychology. In this chapter, we provide a brief overview of each part of the thesis.

Fast and efficient training in machine learning

Mathematical optimization is a backbone of modern machine learning. Most machine learning problems require optimizing some objective function that measures how well a model matches a data set, with the intention of drawing patterns and making decisions on new unseen data. The success of optimization algorithms in solving these problems is critical to the success of machine learning, and has enabled the research community to explore more complex machine learning problems that require bigger models and larger datasets.

Stochastic gradient descent (SGD) has become the standard optimization routine in machine learning, and in particular in deep neural networks, due to its impressive performance across a wide variety of tasks and models. SGD, however, can often be

slow for neural networks with many layers and typically requires careful user oversight for setting hyperparameters properly. While innovations such as batch normalization and skip connections have helped alleviate some of these issues, why such innovations are required eludes full understanding, and it is worthwhile to gain deeper theoretical insights into these problems and to consider more advanced optimization methods specifically tailored towards training large complex models.

In this part of the thesis, we review and analyze some of the recent progress made in this area, develop new optimization algorithms of our own, and theoretically and empirically analyze the performance of existing well-known optimization techniques. In Chapter 2, we review existing work in this area, and present some of the open problems that we explore in the rest of the thesis. In Chapters 3 and 4, we develop new optimization algorithms that are provably fast, significantly easier to train, and require less user oversight. In Chapter 5, we discuss the theory of quantized networks, which use low-precision weights to compress and accelerate neural networks, and when/why they are trainable. Finally in Chapter 6, we discuss some recent results on how the convergence of SGD is affected by the architecture of neural nets, and we show using theoretical analysis that wide networks train faster than narrow nets, and deeper networks train slower than shallow nets – an effect often observed in practice.

Studying the evolution of cultural norms

Understanding human behavior and modeling how cultural norms evolve in different human societies is vital for designing policies and avoiding conflicts around the

world. In this part, we explore ways to use computational game-theoretic techniques, and in particular evolutionary game-theoretic (EGT) models, to gain insight into why different human societies have different norms and behaviors.

In Chapter 7, we introduce evolutionary game theory, and review how it has been previously used to study biological and cultural evolution.

In Chapter 8, we describe an evolutionary game-theoretic model to study how norms change in a society, based on the idea that different strength of norms in societies translate to different game-theoretic interaction structures and incentives. We identify conditions that determine when societies change their existing norms, when they are resistant to such change, and how this depends on the strength of norms in a society.

Next, in Chapter 9, we extend this study to analyze the evolutionary relationships between the tendency to conform and how quickly a population reacts when conditions make a change in norm desirable. Our analysis identifies conditions when a tipping point is reached in a population, causing norms to change rapidly.

Finally, in Chapter 10, we study conditions that affect the existence of group-biased behavior among humans (i.e., favoring others from the same group, and being hostile towards others from different groups). Using an evolutionary game-theoretic model, we show that out-group hostility is dramatically reduced by mobility. Technological and societal advances over the past centuries have greatly increased the degree to which humans change physical locations, and our results show that in highly mobile societies, ones choice of action is more likely to depend on what individual one is interacting with, rather than the group to which the individual belongs.

Part I

FAST & EFFICIENT TRAINING IN MACHINE LEARNING

Chapter 2: **Introduction, background and notation**

Interest in the field of machine learning has grown rapidly over the past decade, and is generally considered now to be one of the key components towards building intelligent systems. Millions of people today use applications that run on machine learning algorithms, in the form of recommendation systems on platforms such as Amazon or Netflix, search engines like Google, speech recognition softwares such as Apple's Siri or the Google Assistant, or image recognition softwares used on various social media websites. Machine learning algorithms draw inferences from massive amounts of data by building a mathematical model to capture patterns or make predictions. As computing resources become increasingly powerful and more easily accessible, machine learning has become increasingly prevalent, and will most likely continue to do so over the coming years.

Mathematical optimization is one of the backbones of modern machine learning. Most machine learning problems can be formulated as optimizing some objective based on a current available set of data (a process typically called *training*), with the intention of drawing patterns and making decisions on new unseen data (*testing*). The success of optimization algorithms in solving these problems is critical to the success of machine learning, and has led the research community to to explore more complex machine learning problems that require core complex mathematical models and larger datasets.

Due to the increasing size of datasets, complex machine learning models can take days to train even with high-performance computing hardware. Moreover, there is a need for efficient optimization algorithms specifically tailored towards training on huge datasets. Thus, there has been widespread interest recently, not only in more efficient optimization algorithms, but also in coming up with heuristics that enable existing optimization algorithms to work better. In this thesis, we review and analyze some of the recent progress made in this direction, and develop several optimization algorithms of our own that are provably fast. Using a principled approach, we also investigate and provide a theoretical justification for why certain optimization algorithms and certain heuristics have been successful in training complex models, while others have not.

For the rest of this chapter, we provide an introduction to optimization methods for solving large-scale machine learning problems and define the notation to be used in the rest of this part of the thesis. In the next section, we show how many popular machine learning models can be formulated as solving an optimization problem. In subsequent sections, we review some existing algorithms that have been successfully used to solve such large-scale problems, and investigate the open questions in this area. Finally, we summarize the main contributions of this part of the thesis.

2.1 Machine learning as an optimization problem

Consider the simple case of linear regression, which is used to model a linear relationship between independent variables $\mathbf{x}$ and a dependent variable y. Suppose we are given a dataset of n observations: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$. In linear regression,

the objective is to find parameters $\mathbf{w}$ such that: $\langle \mathbf{w}, \mathbf{x}_i \rangle = y_i, \forall i$, where $\langle \mathbf{w}, \mathbf{x} \rangle$ denotes the inner product between $\mathbf{w}$ and $\mathbf{x}$. This may not be a solvable problem due to a variety of reasons; for example, the underlying relationship between $\mathbf{x}$ and y may not be linear, or due to measurement noise when collecting the observations in the dataset. Thus, the typical approach is to solve the problem of finding parameters $\mathbf{w}$ such that $\langle \mathbf{w}, \mathbf{x}_i \rangle$ is close to y_i on average. This is typically formulated as the following optimization problem: $\min_{\mathbf{w}} f(\mathbf{w}) \coloneqq \frac{1}{n} \sum_{i=1}^{n} (\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2$.

Similar to the linear regression case, many popular machine learning problems can be formulated as optimization problems of the form:

$$\min_{\mathbf{w}} f(\mathbf{w}) \coloneqq \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{w}; \mathbf{x}_i), \tag{2.1}$$

where $\{\mathbf{x}_i\}$ is a collection of data drawn from some unknown probability distribution p. In typical machine learning applications, each term $f_i(\mathbf{w}; \mathbf{x}_i)$ measures how well a model with parameters $\mathbf{w}$ fits one particular data observation $\mathbf{x}_i$. Given a dataset $\mathcal{D}$ of n data samples $\{\mathbf{x}_i\}$, $f(\mathbf{w})$ measures how well the model fits the entire corpus of data on average. This is typically called an empirical risk minimization problem, and it is an estimate of the *true* problem we want to solve, i.e., the expected risk minimization problem: $\min_{\mathbf{w}} \mathbb{E}_{\mathbf{x}_i \sim p}[f_i(\mathbf{w}; \mathbf{x}_i)]$. Since we typically don't have enough information on the underlying data distribution p to solve the expected risk minimization problem, we typically solve (2.1) instead.

For supervised learning problems, where the objective is to predict a value/label based on some input, each data sample $\mathbf{x}$ in the dataset $\mathcal{D}$ has a corresponding label $\mathbf{y} = \mathcal{C}(\mathbf{x})$, for some unknown labeling function $\mathcal{C}$. In this case, a *training pair* refers

to the tuple $(\mathbf{x}, \mathbf{y})$. We consider that $\mathbf{x}$ is a d-dimensional vector with $\mathbf{x} \in \mathbb{R}^d$, unless specified otherwise. For clarity in presentation, from hereon, we denote $f_i(\mathbf{w}; \mathbf{x}_i)$ as $f_i(\mathbf{w})$. We sometimes also use $f_{\mathbf{x}}$ to denote the model's loss corresponding to a data sample $\mathbf{x}$, which will be clear from the context. Some of the notation used in the rest of the thesis is summarized in Section 2.5.

Logistic regression Many popular machine learning models use objective functions of the same form as in (2.1). For example, logistic regression, which is a linear model for doing binary classification (i.e., distinguishing between two classes of data), uses the following objective function: $f_i(\mathbf{w}) = \log(1 + \exp(-y_i \langle \mathbf{w}, \mathbf{x}_i \rangle))$, where y_i denotes the binary label, $+1$ or -1, which is averaged over n observations.

Neural networks Another powerful class of models that are formulated as (2.1) are deep neural networks. Neural networks use a series of non-linear transformations to build highly complex and flexible function approximators. The output of a typical deep neural network with $\beta + 1$ layers is given by:

$$\hat{\mathbf{y}}_i = \sigma(\mathbf{W}_\beta \sigma(\dots \sigma(\mathbf{W}_1 \sigma(\mathbf{W}_0 \mathbf{x}_i + \mathbf{b}_0) + \mathbf{b}_1) \dots) + \mathbf{b}_\beta).$$

Here $\mathbf{x}_i \in \mathbb{R}^d$ is the input data sample to the neural net, the $\mathbf{W}$'s denote the weight matrices, $\mathbf{b}$'s denote the bias vectors, and $\hat{\mathbf{y}}_i$ denotes the output of the neural network. The function $\sigma(.)$ is typically non-linear and applied point-wise to its arguments. Common choices for $\sigma(.)$ are the sigmoid function: $\sigma(x) = 1/(1+\exp(-x))$, or the ReLU: $\sigma(x) = \max(0, x)$. This sequence of non-linear transformations help the neural network express complex function classes. The shapes of the weight matrices and biases are such that

the output $\hat{\mathbf{y}}_i$ is the same size as the label $\mathbf{y}_i$. Thus, for this neural net, the parameters of the model are given by $\mathbf{w} = [\text{vec}(\mathbf{W}_0)^\top \ \text{vec}(\mathbf{W}_1)^\top \ \cdots \ \text{vec}(\mathbf{W}_\beta)^\top \ \mathbf{b}_0^\top \ \mathbf{b}_1^\top \ \cdots \ \mathbf{b}_\beta^\top]^\top$ (where we imagine all vectors to be column vectors by default and $\top$ denotes the transpose operator). Neural networks have been very successful at wide range of applications, and the loss function used depends on the specific application. For multi-class classification, a common loss function is the cross entropy, where each f_i would have the form: $f_i(\mathbf{w}) = -\sum_{j=1}^{c} (\mathbf{y}_i)_j \log(\hat{\mathbf{y}}_i)_j$, where c is the number of classes (thus the dimensions of $\mathbf{y}_i$ and $\hat{\mathbf{y}}_i$ are also c). One can also use the L2 loss function for regression problems where each f_i would be: $f_i(\mathbf{w}) = \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2$.

Other examples of machine learning models that follow a similar form as (2.1) are support vector machines, matrix completion and graph cuts, among others.

2.2 Stochastic gradient descent

Traditionally, optimization problems of the form (2.1) have been solved using iterative deterministic optimization methods. A popular example of such a method is the gradient descent algorithm, which uses iterative updates of the form:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \nabla f(\mathbf{w}_k),$$

where α denotes the step size, and ∇f denotes the gradient of f w.r.t. the parameters $\mathbf{w}$.

Deterministic optimization methods like gradient descent enjoy fast convergence rates and require less user oversight for setting the step size α, and thus is easy to use. However, when n is large (or even infinite) and the model is large, as is often the case

in modern machine learning, it becomes intractable to exactly evaluate $f(\mathbf{w})$ or its gradient $\nabla f(\mathbf{w})$, which makes classical gradient methods impossible. In such situations, the method of choice for minimizing (2.1) is the stochastic gradient descent (SGD) algorithm [RM51]. On iteration k, SGD uses an approximation $\tilde{f}$ of the true function f, and then computes

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla \tilde{f}_k(\mathbf{w}_k), \tag{2.2}$$

where α_k denotes the step size used on the k-th iteration. Typically, $\tilde{f}$ is an unbiased estimate of f, where a batch $\mathcal{B}_k \subseteq \mathcal{D}$ of data is selected uniformly at random on each iteration k. Thus, $\tilde{f}_k(\mathbf{w}) = \frac{1}{|\mathcal{B}_k|} \sum_{\mathbf{x}_i \in \mathcal{B}_k} f_i(\mathbf{w})$. Note that $\mathbb{E}_{\mathcal{B}_k}[\nabla \tilde{f}_k(\mathbf{w}_k)] = \nabla f(\mathbf{w}_k)$, and so the calculated gradient $\nabla \tilde{f}_k(\mathbf{w}_k)$ can be interpreted as a "noisy" approximation to the true gradient.

2.3 On the successes and drawbacks of SGD

Stochastic gradient descent (SGD) has become one of the most popular optimization algorithms for training deep neural networks, achieving impressive generalization performance across a wide variety of tasks and models. When SGD's hyper-parameters (learning rate, batch size) are set properly, it can usually good generalization performance compared to other optimization algorithms on a variety of benchmark neural network tasks [WRS+17, KS17, SMDH13]. There are, however, a number of open questions and well-known limitations of SGD.

SGD can often be slow for neural networks with many layers, or ones with recurrent connections. While innovations such as batch normalization and skip connections have

helped alleviate this issue to a certain extent, why such techniques are required eludes full understanding, and it is worthwhile to gain deeper theoretical insights into these problems.

A major drawback of SGD is that it requires careful user oversight for setting the step size schedule. Performance is very sensitive to this choice, and all state-of-the-art results were achieved on very careful choice of the learning rate schedule. While there have been some recent work on methods for automatically setting step sizes for stochastic algorithms [KB14,MH15,SZL13,TMDQ16], they are largely heuristic without any theoretical guarantees on convergence rates, and don't work well in practice either [WRLG18].

Moreover, as the datasets grow larger and models become more complex (such as increasing depth on neural networks), SGD typically takes a much longer time to train to high accuracies (i.e., convergence rates are slow). While innovations such as batch normalization and skip connections have helped alleviate this issue to a certain extent, why such techniques are required eludes full understanding. Further, SGD being an inherently sequential algorithm and because of the noise in the gradients, can't be efficiently distributed over computing clusters. This indicates the need for faster optimization methods for training these models.

2.4 Contributions

In this part of the thesis, we explore a few of the open questions mentioned in Section 2.3. We list the main contributions below.

In Chapter 3, we develop stochastic optimization algorithms that require no user oversight by automatically setting the hyperparameters of SGD. This is done by adap-

tively growing the batch size over time to control the amount of noise in the gradient estimate relative to the signal in the gradient estimate. Controlling the noise, in turn, makes the process of setting step sizes much easier, and we present various adaptive step size methods that have provable convergence rate guarantees, as well as good empirical performance on a wide range of machine learning models and datasets.

In Chapter 4, we explore a variant of SGD that has a provably faster convergence rate. We show that this variant can scale linearly over hundreds of computing cores and can speed up training of machine learning models on massive datasets without experiencing the slowdown that existing stochastic methods experience. This was done by leveraging a class of stochastic algorithms called variance reduction, that explicitly reduce the variance in the SGD gradient estimate by adding an error correction term.

In Chapter 5, we investigate quantized networks, which use low-precision weights to compress and accelerate neural networks. We discuss the theory of quantized networks, and when/why they are trainable. In particular, we show that quantized training algorithms that exploit high-precision representations have an important greedy search phase that purely quantized training methods lack, which explains the difficulty of training using low-precision arithmetic.

Finally in Chapter 6, we explore why SGD is efficient for neural nets when tuned properly, and how neural net design affects SGD. In particular, we investigate how over-parametrization – an increase in the number of parameters beyond the number of training data and typical setting in most neural network problems – affects the dynamics of SGD. We find that wide networks train faster than narrow nets, and deeper networks train slower than shallow nets – an effect often observed in practice.

2.5 Table of notation

d	data dimension
n	number of data points
$\mathbf{w}$	vector of parameters of the machine learning model (boldface denotes a vector)
$\mathbf{x}$	d-dimensional input data sample
$\mathbf{y}$	label of the input data sample; we assume $\mathbf{y} = \mathcal{C}(\mathbf{x})$ for a labeling function $\mathcal{C}$
$\mathcal{D}$	training data; $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ for supervised problems, $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$ otherwise
$\mathcal{B}$	set of data points chosen in the mini-batch, i.e., $\mathcal{B} \subseteq \mathcal{D}$
k	current iteration of the optimizer
$f_{\mathbf{x}_i}$ or f_i	scalar function denoting the model's loss corresponding to training pair $(\mathbf{x}_i, \mathbf{y}_i)$
f	scalar loss function to be minimized; typically $f = \frac{1}{n}\sum_{i=1}^n f_i$
$\tilde{f}_k$	approximation of f at iteration k used by stochastic optimization algorithms
$\tilde{f}_{\mathcal{B}}$	(overloading notation) approximation of f by using mini-batch $\mathcal{B} \subseteq \mathcal{D}$
$(\mathbf{v})_i$	i-th element of the vector $\mathbf{v}$
$\nabla\gamma(\mathbf{v})$	gradient of a scalar function γ, i.e., $(\nabla\gamma(\mathbf{w}))_i = \partial\gamma/\partial(\mathbf{v})_i$
$\langle \mathbf{v}_1, \mathbf{v}_2 \rangle$	inner product between two vectors, i.e., $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = \sum_{i=1}^d (\mathbf{v}_1)_i \cdot (\mathbf{v}_2)_i$
$\|\mathbf{v}\|$	L^2 norm of vector $\mathbf{v}$, unless otherwise specified; i.e., $\|\mathbf{v}\| = \sqrt{\sum_{i=1}^d (\mathbf{v})_i^2}$

Chapter 3: **Automated inference using adaptive batch sizes**

3.1 Introduction

SGD uses noisy gradient approximations to solve (2.1). Since the gradient approximations are noisy, the step size α_k must vanish as $k \to \infty$ to guarantee convergence of the method. Typical step size rules require the user to find the optimal decay rate schedule, which usually requires an expensive grid search over different possible parameter values. In this chapter, we propose a "big batch" strategy for SGD. Rather than letting the step size vanish over time as the iterates approach a minimizer, we let the mini-batch $\mathcal{B}$ *adaptively grow* in size to maintain a constant signal-to-noise ratio of the gradient approximation. This prevents the algorithm from getting overwhelmed with noise, and guarantees convergence with an appropriate constant step size. Recent results [KMN+16] have shown that large *fixed* batch sizes fail to find good minimizers for non-convex problems like deep neural networks. Adaptively increasing the batch size over time overcomes this limitation: intuitively, in the initial iterations, the increased stochasticity (corresponding to smaller batches) can help land the iterates near a good minimizer, and larger batches later on can increase the speed of convergence towards this minimizer.

Using this batching strategy, we show that we can keep the step size constant, or let it adapt using a simple Armijo backtracking line search, making the method completely

adaptive with no user-defined parameters. We also derive an adaptive step size method based on the [BB88] curvature estimate that fully automates the big batch method, while empirically enjoying a faster convergence rate than the Armijo backtracking line search.

Big batch methods that adaptively grow the batch size over time have several potential advantages over conventional small-batch SGD:

- Big batch methods don't require the user to choose step size decay parameters. Larger batch sizes with less noise enable easy estimation of the accuracy of the approximate gradient, making it straightforward to adaptively scale up the batch size and maintain fast convergence.

- Backtracking line search tends to work very well when combined with big batches, making the methods completely adaptive with no parameters. A nearly constant signal-to-noise ratio also enables us to define an adaptive step size method based on the Barzilai-Borwein curvature estimate, that performs better empirically on a range of convex problems than the backtracking line search.

- Higher order methods like stochastic L-BFGS typically require more work per iteration than simple SGD. When using big batches, the overhead of more complex methods like L-BFGS can be amortized over more costly gradient approximations. Furthermore, better Hessian approximations can be computed using less noisy gradient terms.

- For a restricted class of non-convex problems (functions satisfying the Polyak-Łojasiewicz Inequality), the per-iteration complexity of big batch SGD is linear and the approximate gradients vanish as the method approaches a solution, which

makes it easy to define automated stopping conditions. In contrast, small batch SGD exhibits sub-linear convergence, and the noisy gradients are not usable as a stopping criterion.

- Big batch methods are much more efficient than conventional SGD in massively parallel/distributed settings. Bigger batches perform more computation between parameter updates, and thus allow a much higher ratio of computation to communication.

For the reasons above, big batch SGD is potentially much easier to automate and requires much less user oversight than classical small batch SGD.

Related work

In this section, we focus on automating stochastic optimization methods by reducing the noise in SGD. We do this by adaptively growing the batch size to control the variance in the gradient estimates, maintaining an approximately constant signal-to-noise ratio, leading to automated methods that do not require vanishing step size parameters. While there has been some work on adaptive step size methods for stochastic optimization [MH15, SZL13, TMDQ16, KB14, Zei12], the methods are largely heuristic without any kind of theoretical guarantees or convergence rates. The work in [TMDQ16] was a first step towards provable automated stochastic methods, and we explore in this direction to show provable convergence rates for the automated big batch method.

While there has been relatively little work in provable automated stochastic methods, there has been recent interest in methods that control gradient noise. These methods

mitigate the effects of vanishing step sizes, though choosing the (constant) step size still requires tuning and oversight. There have been a few papers in this direction that use dynamically increasing batch sizes. In [FS12], the authors propose to increase the size of the batch by a constant factor on *every* iteration, and prove linear convergence in terms of the iterates of the algorithm. In [BCNW12], the authors propose an adaptive strategy for growing the batch size; however, the authors do not present a theoretical guarantee for this method, and instead prove linear convergence for a continuously growing batch, similar to [FS12].

Variance reduction (VR) SGD methods use an error correction term to reduce the noise in stochastic gradient estimates. The methods enjoy a provably faster convergence rate than SGD and have been shown to outperform SGD on convex problems [DBLJ14, JZ13, SRB13, DD+14], as well as in parallel [RHS+15] and distributed settings [DG16]. A caveat, however, is that these methods require either extra storage or full gradient computations, both limiting factors when the dataset is very large. In a recent paper [HAV+15], the authors propose a growing batch strategy for a VR method that enjoys the same convergence guarantees. However, as mentioned above, choosing the constant step size still requires tuning. Another conceptually related approach is importance sampling, i.e., choosing training points such that the variance in the gradient estimates is reduced [BTPG15, CR16, NWS14].

3.2 Big Batch SGD

3.2.1 Preliminaries and motivation

Classical stochastic gradient methods thrive when the current iterate is far from optimal. In this case, a small amount of data is necessary to find a descent direction, and optimization progresses efficiently. As $\mathbf{w}_k$ starts approaching the true solution $\mathbf{w}^\star$, however, noisy gradient estimates frequently fail to produce descent directions and do not reliably decrease the objective. By choosing larger batches with less noise, we may be able to maintain descent directions on each iteration and uphold fast convergence. This observation motivates the proposed "big batch" method. We now explore this idea more rigorously. We wish to show that a noisy gradient approximation $\nabla \tilde{f}$ produces a descent direction when the noise is comparable in magnitude to the true gradient ∇f.

Lemma 3.2.1. *A sufficient condition for $-\nabla \tilde{f}(\mathbf{w})$ to be a descent direction is*

$$\|\nabla \tilde{f}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2 < \|\nabla \tilde{f}(\mathbf{w})\|^2.$$

Proof. This is a standard result in stochastic optimization. We know that $-\nabla \tilde{f}(\mathbf{w})$ is a descent direction iff $\langle \nabla \tilde{f}(\mathbf{w}), \nabla f(\mathbf{w}) \rangle > 0$. Expanding $\|\nabla \tilde{f}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2$ we get: $\|\nabla \tilde{f}(\mathbf{w})\|^2 + \|\nabla f(\mathbf{w})\|^2 - 2\langle \nabla \tilde{f}(\mathbf{w}), \nabla f(\mathbf{w}) \rangle < \|\nabla \tilde{f}(\mathbf{w})\|^2$. We can re-write this as: $-2\langle \nabla \tilde{f}(\mathbf{w}), \nabla f(\mathbf{w}) \rangle < -\|\nabla f(\mathbf{w})\|^2 \leq 0$, which is true for a descent direction. ■

Thus, we see that: if the error $\|\nabla \tilde{f}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2$ is small relative to the gradient $\|\nabla \tilde{f}(\mathbf{w})\|^2$, the stochastic approximation is a descent direction. But how big is this error and how large does a batch need to be to guarantee this condition? Let $\tilde{f}_{\mathcal{B}}$ denote the unbiased estimate of f using a mini-batch $\mathcal{B}$ sampled uniformly at random from dataset

$\mathcal{D}$. Also, let $f_{\mathbf{x}}$ denote the loss corresponding to training pair $(\mathbf{x}, \mathcal{C}(\mathbf{x}))$. Then, by the weak law of large numbers[1]

$$\mathbb{E}[\|\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2] = \frac{1}{|\mathcal{B}|}\mathbb{E}_{\mathbf{x}}[\|\nabla f_{\mathbf{x}}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2] = \frac{1}{|\mathcal{B}|}\operatorname{Tr}\operatorname{Var}_{\mathbf{x}}\nabla f_{\mathbf{x}}(\mathbf{w}),$$

and so we can estimate the error of a stochastic gradient if we have some knowledge of the variance of $\nabla f_{\mathbf{x}}(\mathbf{w})$. In practice, this variance could be estimated using the sample variance of a batch $\{\nabla f_i(\mathbf{w})\}_{\mathbf{x}_i \in \mathcal{B}}$. However, we would like some bounds on the magnitude of this gradient to show that it is well-behaved, and also to analyze worst-case convergence behavior. To this end, we make the following assumption.

Assumption 3.2.1. *We assume that each f_i has L_x-Lipschitz dependence on data $\mathbf{x}$, i.e., given two data points $\mathbf{x}_1, \mathbf{x}_2 \sim p(\mathbf{x})$, we have: $\|\nabla f_1(\mathbf{w}) - \nabla f_2(\mathbf{w})\| \leq L_x\|\mathbf{x}_1 - \mathbf{x}_2\|$.*

Under this assumption, we can bound the error of the stochastic gradient. The bound is uniform with respect to $\mathbf{w}$, which makes it rather useful in analyzing the convergence rate for big batch methods.

Theorem 3.2.1. *Given the current iterate $\mathbf{w}$, suppose Assumption 3.2.1 holds and that the data distribution p has bounded second moment. Then the estimated gradient $\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w})$ has variance bounded by*

$$\mathbb{E}_{\mathcal{B}}\|\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2 := \operatorname{Tr}\operatorname{Var}_{\mathcal{B}}(\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w})) \leq \frac{4L_x^2 \operatorname{Tr}\operatorname{Var}_{\mathbf{x}}(\mathbf{x})}{|\mathcal{B}|},$$

where $\mathbf{x} \sim p(\mathbf{x})$. Note the bound is uniform in $\mathbf{w}$.

[1] We assume the random variable $\nabla f_{\mathbf{x}}$ is measurable and has bounded second moment. These conditions will be guaranteed by the hypothesis of Theorem 3.2.1.

Proof. Let $\bar{\mathbf{x}} = \mathbb{E}[\mathbf{x}]$ be the mean of $\mathbf{x}$. Given the current iterate $\mathbf{w}$, we assume that the batch $\mathcal{B}$ is sampled uniformly with replacement from p. We then have:

$$\begin{aligned}
\|\nabla f_{\mathbf{x}}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2 &\leq 2\|\nabla f_{\mathbf{x}}(\mathbf{w}) - \nabla f_{\bar{\mathbf{x}}}(\mathbf{w})\|^2 + 2\|\nabla f_{\bar{\mathbf{x}}}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2 \\
&\leq 2L_x^2\|\mathbf{x} - \bar{\mathbf{x}}\|^2 + 2\|\mathbb{E}_{\mathbf{x}}[\nabla f_{\bar{\mathbf{x}}}(\mathbf{w}) - \nabla f_{\mathbf{x}}(\mathbf{w})]\|^2 \\
&\leq 2L_x^2\|\mathbf{x} - \bar{\mathbf{x}}\|^2 + 2\mathbb{E}_{\mathbf{x}}\|\nabla f_{\bar{\mathbf{x}}}(\mathbf{w}) - \nabla f_{\mathbf{x}}(\mathbf{w})\|^2 \\
&\leq 2L_x^2\|\mathbf{x} - \bar{\mathbf{x}}\|^2 + 2L_x^2\mathbb{E}_{\mathbf{x}}\|\bar{\mathbf{x}} - \mathbf{x}\|^2 \\
&= 2L_x^2\|\mathbf{x} - \bar{\mathbf{x}}\|^2 + 2L_x^2 \operatorname{Tr}\operatorname{Var}_{\mathbf{x}}(\mathbf{x}),
\end{aligned}$$

where the first inequality uses the property $\|\mathbf{a} + \mathbf{b}\|^2 \leq 2\|\mathbf{a}\|^2 + 2\|\mathbf{b}\|^2$, the second and fourth inequalities use Assumption 3.2.1, and the third inequality uses Jensen's inequality. This bound is *uniform* in $\mathbf{w}$. We then have

$$\mathbb{E}_{\mathbf{x}}\|\nabla f_{\mathbf{x}}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2 \leq 2L_x^2\mathbb{E}_{\mathbf{x}}\|\mathbf{x} - \bar{\mathbf{x}}\|^2 + 2L_{\mathbf{x}}^2 \operatorname{Tr}\operatorname{Var}_{\mathbf{x}}(\mathbf{x}) = 4L_x^2 \operatorname{Tr}\operatorname{Var}_{\mathbf{x}}(\mathbf{x}),$$

uniformly for all $\mathbf{w}$. The result follows from the observation that

$$\mathbb{E}_{\mathcal{B}}\|\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2 = \frac{1}{|\mathcal{B}|} E_{\mathbf{x}}\|\nabla f_{\mathbf{x}}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2.$$

■

Note that, using a finite number of samples, one can approximate the quantity $\operatorname{Var}_{\mathbf{x}}(\mathbf{x})$.

3.2.2 A template for big batch SGD

Theorem 3.2.1 and Lemma 3.2.1 together suggest that we should expect $d = -\nabla \tilde{f}_{\mathcal{B}}$ to be a descent direction reasonably often provided

$$\theta^2\|\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w})\|^2 \geq \frac{1}{|\mathcal{B}|}[\operatorname{Tr}\operatorname{Var}_{\mathbf{x}}(\nabla f_{\mathbf{x}}(\mathbf{w}))], \tag{3.1}$$

$$\text{or} \quad \theta^2 \|\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w})\|^2 \geq \frac{4L_x^2 \operatorname{Tr} \operatorname{Var}_{\mathbf{x}}(\mathbf{x})}{|\mathcal{B}|},$$

for some $\theta < 1$. Big batch methods capitalize on this observation.

On each iteration k, starting from a point $\mathbf{w}_k$, the big batch method performs the following steps:

1. Estimate the variance $\operatorname{Tr} \operatorname{Var}_{\mathbf{x}}[\nabla f_{\mathbf{x}}(\mathbf{w}_k)]$, and a batch size K large enough that

$$\begin{aligned} \theta^2 \mathbb{E}\|\nabla \tilde{f}_{\mathcal{B}_k}(\mathbf{w}_k)\|^2 &\geq \mathbb{E}\|\nabla \tilde{f}_{\mathcal{B}_k}(\mathbf{w}_k) - \nabla f(\mathbf{w}_k)\|^2 \\ &= \frac{1}{K} \operatorname{Tr} \operatorname{Var}_{\mathbf{x}} \nabla f_{\mathbf{x}}(\mathbf{w}_k), \end{aligned} \tag{3.2}$$

   where $\theta \in (0, 1)$ and $\mathcal{B}_k$ is the selected batch on the k-th iteration with $|\mathcal{B}_k| = K$.

2. Choose a step size α_k.

3. Perform the update: $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla \tilde{f}_{\mathcal{B}_k}(\mathbf{w}_k)$.

One can implement these steps using different variance estimators and different step size strategies. In the next section, we show that, if condition (3.2) holds, then fast convergence can be achieved using an appropriate constant step size. In subsequent sections, we address the issue of how to build practical big batch implementations using automated variance and step size estimators that require no user oversight.

3.3 Convergence analysis

We now present convergence bounds for big batch SGD methods (3.3). We rewrite the SGD update as:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \nabla \tilde{f}_{\mathcal{B}_k}(\mathbf{w}_k) = \mathbf{w}_k - \alpha(\nabla f(\mathbf{w}_k) + \tilde{\mathbf{e}}_k), \tag{3.3}$$

where $\tilde{\mathbf{e}}_k = \nabla \tilde{f}_{\mathcal{B}_k}(\mathbf{w}_k) - \nabla f(\mathbf{w}_k)$, and $\mathbb{E}_{\mathcal{B}}[\tilde{\mathbf{e}}_k] = 0$. Let us also define $\tilde{\mathbf{g}}_k = \nabla f(\mathbf{w}_k) + \tilde{\mathbf{e}}_k$. Before we present our results, we first state two assumptions about the loss function $f(\mathbf{w})$.

Assumption 3.3.1. *We assume that the objective function f has L-Lipschitz gradients:* $f(\mathbf{w}) \leq f(\mathbf{w}') + \langle \nabla f(\mathbf{w}'), (\mathbf{w} - \mathbf{w}') \rangle + \frac{L}{2}\|\mathbf{w} - \mathbf{w}'\|^2$.

This is a standard smoothness assumption used widely in the optimization literature. Note that a consequence of Assumption 3.3.1 is: $\|\nabla f(\mathbf{w}) - \nabla f(\mathbf{w}')\| \leq L\|\mathbf{w} - \mathbf{w}'\|$.

Assumption 3.3.2. *We assume that the objective function f satisfies the Polyak-Łojasiewicz Inequality:* $\|\nabla f(\mathbf{w})\|^2 \geq 2\mu(f(\mathbf{w}) - f(\mathbf{w}^\star))$, *where $\mathbf{w}^\star$ is the optimal solution.*

Note that this inequality does not require f to be convex. It does, however, imply that every stationary point is a global minimizer [KNS16, Pol63]. We now present a result that establishes an upper bound on the objective value in terms of the error in the gradient of the sampled batch.

Lemma 3.3.1. *Suppose we apply an update of the form* (3.3) *where the batch $\mathcal{B}_k$ is uniformly sampled from the dataset $\mathcal{D}$ on each iteration k. If the objective f satisfies Assumptions 3.3.1 and 3.3.2, we have:*

$$\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}^\star)] \leq \Big(1 - 2\mu\big(\alpha - \frac{L\alpha^2}{2}\big)\Big)\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)] + \frac{L\alpha^2}{2}\mathbb{E}\|\tilde{\mathbf{e}}_k\|^2.$$

Proof. From (3.3) and Assumption 3.3.1 we get

$$f(\mathbf{w}_{k+1}) \leq f(\mathbf{w}_k) - \alpha\langle \tilde{\mathbf{g}}_k, \nabla f(\mathbf{w}_k)\rangle + \frac{L\alpha^2}{2}\|\tilde{\mathbf{g}}_k\|^2.$$

Taking expectation with respect to the batch $\mathcal{B}_t$ and conditioning on $\mathbf{w}_k$, we get

$$\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}^\star)] \leq f(\mathbf{w}_k) - f(\mathbf{w}^\star) - \alpha\langle \mathbb{E}[\tilde{\mathbf{g}}_k], \nabla f(\mathbf{w}_k)\rangle + \frac{L\alpha^2}{2}\mathbb{E}\|\tilde{\mathbf{g}}_k\|^2$$

$$
\begin{aligned}
&= f(\mathbf{w}_k) - f(\mathbf{w}^\star) - \big(\alpha - \frac{L\alpha^2}{2}\big)\|\nabla f(\mathbf{w}_k)\|^2 + \frac{L\alpha^2}{2}\mathbb{E}\|\tilde{\mathbf{e}}_k\|^2 \\
&\leq \Big(1 - 2\mu\big(\alpha - \frac{L\alpha^2}{2}\big)\Big)(f(\mathbf{w}_k) - f(\mathbf{w}^\star)) + \frac{L\alpha^2}{2}\mathbb{E}\|\tilde{\mathbf{e}}_k\|^2,
\end{aligned}
$$

where the second inequality follows from Assumption 3.3.2. Taking expectation, the result follows. ■

Using Lemma 3.3.1, we now provide convergence rates for big batch SGD.

Theorem 3.3.1. *Suppose f satisfies Assumptions 3.3.1 and 3.3.2. Suppose further that on each iteration the batch size is large enough to satisfy* (3.2) *for $\theta \in (0, 1)$. If $0 \leq \alpha < \frac{2}{L\beta}$, where $\beta = \frac{\theta^2 + (1-\theta)^2}{(1-\theta)^2}$, then we get the following linear convergence bound for big batch SGD using updates of the form 3.3:*

$$
\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}^\star)] \leq \gamma \cdot \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)],
$$

where $\gamma = \big(1 - 2\mu(\alpha - \frac{L\alpha^2\beta}{2})\big)$. Choosing the optimal step size of $\alpha = \frac{1}{\beta L}$, we get

$$
\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}^\star)] \leq \big(1 - \frac{\mu}{\beta L}\big) \cdot \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)].
$$

Proof. We begin by applying the reverse triangle inequality to (3.2) to get $(1-\theta)\mathbb{E}\|\nabla f_{\mathcal{B}}(x)\| \leq \mathbb{E}\|\nabla f(x)\|$, which applied to (3.2) yields:

$$
\frac{\theta^2}{(1-\theta)^2}\mathbb{E}\|\nabla f(\mathbf{w}_k)\|^2 \geq \mathbb{E}\|\nabla f_{\mathcal{B}}(\mathbf{w}_k) - \nabla f(\mathbf{w}_k)\|^2 = \mathbb{E}\|\tilde{\mathbf{e}}_k\|^2. \tag{3.4}
$$

Applying (3.4) to the result in Lemma 3.3.1, we get

$$
\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}^\star)] \leq \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)] - \big(\alpha - \frac{L\alpha^2\beta}{2}\big)\mathbb{E}\|\nabla f(\mathbf{w}_k)\|^2,
$$

where $\beta = \frac{\theta^2 + (1-\theta)^2}{(1-\theta)^2} \geq 1$. Assuming $\alpha - \frac{L\alpha^2\beta}{2} \geq 0$ and using Assumption 3.3.2, we get:

$$
\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}^\star)] \leq \Big(1 - 2\mu\big(\alpha - \frac{L\alpha^2\beta}{2}\big)\Big)\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)],
$$

which proves the theorem. Note that $\max_\alpha\{\alpha - \frac{L\alpha^2\beta}{2}\} = \frac{1}{2L\beta}$, and $\mu \leq L$. It follows that $0 \leq \Big(1 - 2\mu\big(\alpha - \frac{L\alpha^2\beta}{2}\big)\Big) < 1$. The second result follows immediately. ■

Note that the above linear convergence rate bound holds without requiring convexity. Comparing it with the convergence rate of deterministic gradient descent under similar assumptions, we see that big batch SGD suffers a slowdown by a factor β, due to the noise in the estimation of the gradients. We now present a result proving a $\mathcal{O}(1/k)$ convergence rate for general smooth convex functions.

Theorem 3.3.2. *Suppose f satisfies Assumptions 3.3.1, is convex, and condition* (3.2) *is satisfied on each iteration. Then we get sub-linear convergence of the form:*

$$\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)] \leq \frac{\|\mathbf{w}_0 - \mathbf{w}^\star\|^2}{(2\alpha - 2L\alpha^2\beta)(k+1)} = \mathcal{O}(1/k),$$

where $\beta = \frac{\theta^2 + (1-\theta)^2}{(1-\theta)^2}$ and $\alpha < \frac{1}{L\beta}$. Choosing the optimal step size of $\alpha = \frac{1}{2L\beta}$, we get

$$\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)] \leq \frac{2L\beta\|\mathbf{w}_0 - \mathbf{w}^\star\|^2}{k+1} = \mathcal{O}(1/k).$$

Proof. Applying the reverse triangle inequality to (3.2) and using Lemma 3.3.1 we get, as in Theorem 3.3.1:

$$\mathbb{E}[f(\mathbf{w}_{k+1})] \leq \mathbb{E}[f(\mathbf{w}_k)] - \big(\alpha - \frac{L\alpha^2\beta}{2}\big)\mathbb{E}\|\nabla f(\mathbf{w}_k)\|^2, \tag{3.5}$$

where $\beta = \frac{\theta^2 + (1-\theta)^2}{(1-\theta)^2} \geq 1$. Note that $\alpha - \frac{L\alpha^2\beta}{2} > 0$ if $\alpha < \frac{2}{L\beta}$. From (3.3), taking norm on both sides and taking expectation, conditioned on all $\mathbf{w}_t$, with $t = 0, 1, \cdots, k$, we get

$$\begin{aligned}
\mathbb{E}\|\mathbf{w}_{k+1} - \mathbf{w}^\star\|^2 &= \|\mathbf{w}_k - \mathbf{w}^\star\|^2 - 2\alpha\mathbb{E}\langle \mathbf{w}_k - \mathbf{w}^\star, \nabla f(\mathbf{w}_k) + \tilde{\mathbf{e}}_k\rangle + \alpha^2\mathbb{E}\|\nabla f(\mathbf{w}_k) + \tilde{\mathbf{e}}_k\|^2 \\
&\leq \|\mathbf{w}_k - \mathbf{w}^\star\|^2 - 2\alpha\langle \mathbf{w}_k - \mathbf{w}^\star, \nabla f(\mathbf{w}_k)\rangle + \alpha^2\beta\|\nabla f(\mathbf{w}_k)\|^2 \\
&\leq \|\mathbf{w}_k - \mathbf{w}^\star\|^2 - 2\alpha(f(\mathbf{w}_k) - f(\mathbf{w}^\star)) + \alpha^2\beta\|\nabla f(\mathbf{w}_k)\|^2
\end{aligned}$$

$$\leq \|\mathbf{w}_k - \mathbf{w}^\star\|^2 - 2\alpha(f(\mathbf{w}_k) - f(\mathbf{w}^\star)) + 2L\alpha^2\beta(f(\mathbf{w}_k) - f(\mathbf{w}^\star))$$

$$= \|\mathbf{w}_k - \mathbf{w}^\star\|^2 - (2\alpha - 2L\alpha^2\beta)(f(\mathbf{w}_k) - f(\mathbf{w}^\star)),$$

where we use the property that $\mathbb{E}[\tilde{\mathbf{e}}_k] = 0$, and the properties $f(\mathbf{w}) \leq f(\mathbf{w}^\star) + \langle \mathbf{w} - \mathbf{w}^\star, \nabla f(\mathbf{w}) \rangle$ (which follows from the convexity of f) and $\|\nabla f(\mathbf{w})\|^2 \leq 2L(f(\mathbf{w}) - f(\mathbf{w}^\star))$ (a proof for this identity can be found in [Nes13]). Note that $2\alpha - 2L\alpha^2\beta > 0$ when $\alpha < \frac{1}{L\beta}$. Taking expectation on all $\mathbf{w}_k$, we get

$$\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)] \leq \frac{1}{2\alpha(1 - L\alpha\beta)}(\mathbb{E}\|\mathbf{w}_k - \mathbf{w}^\star\|^2 - \mathbb{E}\|\mathbf{w}_{k+1} - \mathbf{w}^\star\|^2). \tag{3.6}$$

Summing (3.6) over all $k = 0, 1, \cdots, T$, and using the telescoping sum in $\|\mathbf{w}_k - \mathbf{w}^\star\|^2$:

$$\sum_{k=0}^{T} \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)] \leq \frac{1}{2\alpha(1 - L\alpha\beta)}(\mathbb{E}\|\mathbf{w}_0 - \mathbf{w}^\star\|^2 - \mathbb{E}\|\mathbf{w}_{k+1} - \mathbf{w}^\star\|^2)$$

$$\leq \frac{1}{2\alpha(1 - L\alpha\beta)}\|\mathbf{w}_0 - \mathbf{w}^\star\|^2. \tag{3.7}$$

From (3.5) we see that $\mathbb{E}[f(\mathbf{w}_{k+1})] \leq \mathbb{E}[f(\mathbf{w}_k)]$ when $\alpha < \frac{2}{L\beta}$. Thus, we rewrite (3.7) as:

$$\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)] \leq \frac{1}{T+1}\sum_{k=0}^{T} \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)] \leq \frac{\|\mathbf{w}_0 - \mathbf{w}^\star\|^2}{(2\alpha - 2L\alpha^2\beta)(T+1)}.$$

Choosing the optimal step size of $\alpha = \frac{1}{2L\beta}$, the second result follows. ■

3.3.1 Comparison to classical SGD

Conventional small batch SGD methods can attain only $O(1/k)$ convergence for strongly convex problems, thus requiring $O(1/\epsilon)$ gradient evaluations to achieve an optimality gap less than ϵ, and this has been shown to be *optimal* in the online setting (i.e., the infinite data setting) [RSS11]. In the previous section, however, we have shown that

big batch SGD methods converge linearly in the number of iterations, under a weaker assumption than strong convexity, in the online setting. Unfortunately, per-iteration convergence rates are not a fair comparison between these methods because the cost of a big batch iteration grows with the iteration count, unlike classical SGD. For this reason, it is interesting to study the convergence rate of big batch SGD as a function of *gradient evaluations*.

From Lemma 3.3.1, we see that we should not expect to achieve an optimality gap less than ϵ until we have: $\frac{L\alpha^2}{2}\mathbb{E}_{\mathcal{B}_k}\|\tilde{\mathbf{e}}_k\|^2 < \epsilon$. In the worst case, by Theorem 3.2.1, this requires $\frac{L\alpha^2}{2}\frac{4L_x^2 \operatorname{Tr}\operatorname{Var}_{\mathbf{x}}(\mathbf{x})}{|\mathcal{B}|} < \epsilon$, or $|\mathcal{B}| \geq O(1/\epsilon)$ gradient evaluations. Note that in the online or infinite data case, this is an optimal bound, and matches that of other SGD methods.

All our results hold for the infinite sample case. Note that the finite sample case is fairly trivial with a growing batch size: asymptotically, the batch size becomes the whole dataset, at which point we get the same asymptotic behavior as deterministic gradient descent, achieving linear convergence rates.

3.4 Practical implementation with backtracking line search

While one could implement a big batch method using analytical bounds on the gradient and its variance (such as that provided by Theorem 3.2.1), the purpose of big batch methods is to enable automated adaptive estimation of algorithm parameters. Furthermore, the step size bounds provided by our convergence analysis, like the step size bounds for classical SGD, are fairly conservative and more aggressive step size choices

Algorithm 1 Big batch SGD: fixed step size

initialize $\mathbf{w}_0$, step size α, initial batch size $K > 1$, batch size increment δ_k

while not converged **do**

  Draw random batch with size $|\mathcal{B}| = K$; Calculate $V_{\mathcal{B}}$ and $\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}_k)$ using (3.8)

  while $\|\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}_k)\|^2 \leq V_{\mathcal{B}}/K$ **do**

    Increase batch size $K \leftarrow K + \delta_K$

    Sample more gradients and update $V_{\mathcal{B}}$ and $\nabla f_{\mathcal{B}}(\mathbf{w}_k)$

  end while

  $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}_k)$

end while

are likely to be more effective.

The framework outlined in Section 3.2.2 requires two ingredients: estimating the batch size and estimating the step size. To estimate the batch size needed to achieve (3.2), we start with an initial batch size K, and draw a random batch $\mathcal{B}$ with $|\mathcal{B}| = K$. We then compute the stochastic gradient estimate $\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}_k)$ and the sample variance

$$V_{\mathcal{B}} := \frac{1}{|\mathcal{B}| - 1} \sum_{\mathbf{x} \in \mathcal{B}} \|\nabla f_{\mathbf{x}}(\mathbf{w}_k) - \nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}_k)\|^2 \approx \operatorname{Tr} \operatorname{Var}_{\mathbf{x} \in \mathcal{B}}(\nabla f_{\mathbf{x}}(\mathbf{w}_k)). \tag{3.8}$$

We then test whether $\|\nabla f_{\mathcal{B}}(\mathbf{w}_k)\|^2 > V_{\mathcal{B}}/|\mathcal{B}|$ as a proxy for (3.2). If this condition holds, we proceed with a gradient step, else we increase the batch size $K \leftarrow K + \delta_K$, and check our condition again. We fix $\delta_K = 0.1K$ for all our experiments. Our implementation also simply chooses $\theta = 1$. The fixed step size big batch method is listed in Algorithm 1. We also consider a backtracking variant of SGD that adaptively tunes the step size. This method selects batch sizes using the same criterion (3.8) as in the constant step size

case. However, after a batch has been selected, a backtracking Armijo line search is used to select a step size. In the Armijo line search, we keep decreasing the step size by a constant factor (in our case, by a factor of 2) until the following condition is satisfied on each iteration:

$$\tilde{f}_{\mathcal{B}}(\mathbf{w}_{k+1}) \leq \tilde{f}_{\mathcal{B}}(\mathbf{w}_k) - c\alpha_k \|\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}_k)\|^2, \tag{3.9}$$

where c is a parameter of the line search usually set to $0 < c \leq 0.5$. We now present a convergence result of big batch SGD using the Armijo line search.

Theorem 3.4.1. *Suppose that f satisfies Assumptions 3.3.1 and 3.3.2 and on each iteration, and the batch size is large enough to satisfy* (3.2) *for $\theta \in (0, 1)$. If an Armijo line search, given by* (3.9)*, is used, and the step size is decreased by a factor of 2 failing* (3.9)*, then we get the following linear convergence bound for big batch SGD using updates of the form 3.3:*

$$\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}^\star)] \leq \gamma \cdot \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)],$$

where $\gamma = \left(1 - 2c\mu \min\left(\alpha_0, \frac{1}{2\beta L}\right)\right)$ and $0 < c \leq 0.5$. If the initial step size α_0 is set large enough such that $\alpha_0 \geq \frac{1}{2\beta L}$, then we get:

$$\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}^\star)] \leq \Big(1 - \frac{c\mu}{\beta L}\Big)\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)].$$

Proof. Applying the reverse triangle inequality to (3.2) and using Lemma 3.3.1 we get, as in Theorem 3.3.1:

$$\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}^\star)] \leq \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)] - \big(\alpha - \frac{L\alpha^2\beta}{2}\big)\mathbb{E}\|\nabla f(\mathbf{w}_k)\|^2, \tag{3.10}$$

where $\beta = \frac{\theta^2 + (1-\theta)^2}{(1-\theta)^2} \geq 1$.

We will show that the backtracking condition in (3.9) is satisfied whenever $0 < \alpha_t \leq \frac{1}{\beta L}$. Notice that: $0 < \alpha_t \leq \frac{1}{\beta L}$ implies $-\alpha_t + \frac{L\alpha_t^2\beta}{2} \leq -\frac{\alpha_t}{2}$. Thus, we can rewrite (3.10) as

$$\begin{aligned}\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}^\star)] &\leq \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)] - \frac{\alpha_t}{2}\mathbb{E}\|\nabla f(\mathbf{w}_k)\|^2 \\ &\leq \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)] - c\alpha_t\mathbb{E}\|\nabla f(\mathbf{w}_k)\|^2,\end{aligned}$$

where $0 < c \leq 0.5$. Thus, the backtracking line search condition (3.9) is satisfied whenever $0 < \alpha_t \leq \frac{1}{L\beta}$. Now we know that either $\alpha_t = \alpha_0$ (the initial step size), or $\alpha_t \geq \frac{1}{2\beta L}$, where the step size is decreased by a factor of 2 each time the backtracking condition fails. Thus, we can rewrite the above as

$$\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}^\star)] \leq \mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)] - c\min\left(\alpha_0, \frac{1}{2\beta L}\right)\mathbb{E}\|\nabla f(\mathbf{w}_k)\|^2.$$

Using Assumption 3.3.2 we get

$$\mathbb{E}[f(\mathbf{w}_{k+1}) - f(\mathbf{w}^\star)] \leq \left(1 - 2c\mu\min\left(\alpha_0, \frac{1}{2\beta L}\right)\right)\mathbb{E}[f(\mathbf{w}_k) - f(\mathbf{w}^\star)].$$

Assuming we start off the step size at a large value such that $\min(\alpha_0, \frac{1}{2\beta L}) = \frac{1}{2\beta L}$, we can rewrite the above to get the desired bound. ■

In practice, on iterations where the batch size increases, we double the step size before running line search to prevent the step sizes from decreasing monotonically. The complete details are listed in Algorithm 2.

3.5 Adaptive step sizes using the Barzilai-Borwein estimate

While the Armijo backtracking line search leads to an automated big batch method, the step size sequence is monotonic (neglecting the heuristic mentioned in the previous

Algorithm 2 Big batch SGD: backtracking line search

initialize $\mathbf{w}_0$, initial step size α, initial batch size $K > 1$, batch size increment δ_k, backtracking line search parameter c, flag $F = 0$

while not converged **do**

  Draw random batch with size $|\mathcal{B}| = K$; Calculate $V_{\mathcal{B}}$ and $\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}_k)$ using (3.8)

  while $\|\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}_k)\|^2 \leq V_{\mathcal{B}}/K$ **do**

    Increase batch size $K \leftarrow K + \delta_K$

    Sample more gradients and update $V_{\mathcal{B}}$ and $\nabla f_{\mathcal{B}}(\mathbf{w}_k)$

    Set flag $F = 1$

  end while

  if flag $F == 1$ **then**

    $\alpha \leftarrow \alpha * 2$; Reset flag $F = 0$

  end if

  while $\tilde{f}_{\mathcal{B}}(\mathbf{w}_k - \alpha \nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}_k)) > \tilde{f}_{\mathcal{B}}(\mathbf{w}_k) - c\alpha_t \|\nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}_k)\|^2$ **do**

    $\alpha \leftarrow \alpha/2$

  end while

  $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \nabla \tilde{f}_{\mathcal{B}}(\mathbf{w}_k)$

end while

section). In this section, we derive a non-monotonic step size scheme that uses curvature estimates to propose new step size choices.

Our derivation follows the classical adaptive [BB88] (BB) method. The BB methods fits a quadratic model to the objective on each iteration, and a step size is proposed that is optimal for the local quadratic model [GSB14]. To derive the analog of the

BB method for stochastic problems, we consider quadratic approximations of the form $f(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\phi}} f_{\boldsymbol{\phi}}(\mathbf{w})$, where $f_{\boldsymbol{\phi}}(\mathbf{w}) = \frac{\nu}{2}\|\mathbf{w} - \boldsymbol{\phi}\|^2$ and $\boldsymbol{\phi} \sim \mathcal{N}(\mathbf{w}^\star, \sigma^2 I)$. We derive the optimal step size for this. We can rewrite the quadratic approximation as:

$$f(\mathbf{w}) = \frac{\nu}{2}\mathbb{E}_{\boldsymbol{\phi}}\|\mathbf{w} - \boldsymbol{\phi}\|^2 = \frac{\nu}{2}[\langle \mathbf{w}, \mathbf{w}\rangle - 2\langle \mathbf{w}, \mathbf{w}^\star\rangle - \mathbb{E}\langle \boldsymbol{\phi}, \boldsymbol{\phi}\rangle] = \frac{\nu}{2}\big(\|\mathbf{w} - \mathbf{w}^\star\|^2 + d\sigma^2\big),$$

since we can write: $\mathbb{E}\langle \boldsymbol{\phi}, \boldsymbol{\phi}\rangle = \sum_{i=1}^d \mathbb{E}(\boldsymbol{\phi})_i^2 = \sum_{i=1}^d (\mathbf{w}^\star)_i^2 + \sigma^2 = \|\mathbf{w}^\star\|^2 + d\sigma^2$.

Further, notice that: $\mathbb{E}_{\boldsymbol{\phi}}[\nabla f(\mathbf{w})] = \nu(\mathbf{w} - \mathbf{w}^\star)$ and $\operatorname{Tr}\operatorname{Var}_{\boldsymbol{\phi}}[\nabla f(\mathbf{w})] = d\nu^2\sigma^2$. Using the quadratic approximation, we can rewrite the update for big batch SGD as:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_t \frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|} \nu(\mathbf{w}_k - \boldsymbol{\phi}_i) = (1 - \nu\alpha_t)\mathbf{w}_k + \nu\alpha_t \mathbf{w}^\star + \frac{\nu\sigma\alpha_t}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \boldsymbol{\xi}_i,$$

where we write $\boldsymbol{\phi}_i = \mathbf{w}^\star + \sigma\boldsymbol{\xi}_i$ with $\boldsymbol{\xi}_i \sim \mathcal{N}(\mathbf{0}, I)$. The expected value of f is:

$$\begin{aligned}\mathbb{E}[f(\mathbf{w}_{k+1})] &= \frac{\nu}{2}\mathbb{E}_{\boldsymbol{\xi}}\left[\left\|(1 - \nu\alpha_t)(\mathbf{w}_k - \mathbf{w}^\star) + \frac{\nu\sigma\alpha_t}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\boldsymbol{\xi}_i\right\|^2 + d\sigma^2\right] \\ &= \frac{\nu}{2}\left(\|(1 - \nu\alpha_t)(\mathbf{w}_k - \mathbf{w}^\star)\|^2 + (1 + \frac{\nu^2\alpha_t^2}{|\mathcal{B}|})d\sigma^2\right).\end{aligned}$$

Minimizing $\mathbb{E}[f(\mathbf{w}_{k+1})]$ w.r.t. α_k we get:

$$\alpha_k = \frac{1}{\nu}\cdot\left(1 - \frac{\frac{1}{|\mathcal{B}_k|}\operatorname{Tr}\operatorname{Var}_{\mathbf{x}}[\nabla f_{\mathbf{x}}(\mathbf{w}_k)]}{\mathbb{E}\big\|\nabla \tilde{f}_{\mathcal{B}_k}(\mathbf{w}_k)\big\|^2}\right). \tag{3.11}$$

Here ν denotes the curvature of the quadratic approximation. Note that, in the case of *deterministic* gradient descent, the optimal step size is simply $1/\nu$ [GSB14]. We estimate the curvature ν_t on each iteration using the BB least-squares rule [BB88]:

$$\nu_k = \frac{\langle \mathbf{w}_k - \mathbf{w}_{k-1}, \nabla \tilde{f}_{\mathcal{B}_k}(\mathbf{w}_k) - \nabla \tilde{f}_{\mathcal{B}_k}(\mathbf{w}_{k-1})\rangle}{\|\mathbf{w}_k - \mathbf{w}_{k-1}\|^2}. \tag{3.12}$$

Thus, each time we sample a batch $\mathcal{B}_k$ on the k-th iteration, we calculate the gradient on that batch in the previous iterate, i.e., we calculate $\nabla \tilde{f}_{\mathcal{B}_k}(\mathbf{w}_{k-1})$. This gives us an approximate curvature estimate, with which we derive the step size α_k using (3.11).

3.5.1 Convergence proof

Here we prove convergence for the adaptive step size method described above. For the convergence proof, we first state two assumptions:

Assumption 3.5.1. *Each f_i has L-Lipschitz gradients:*

$f_i(\mathbf{w}) \leq f_i(\mathbf{w}') + \langle \nabla f_i(\mathbf{w}'), \mathbf{w} - \mathbf{w}' \rangle + \frac{L}{2}\|\mathbf{w} - \mathbf{w}'\|^2, \forall i.$

Assumption 3.5.2. *Each f_i is μ-strongly convex:*

$\langle \nabla f_i(\mathbf{w}) - \nabla f_i(\mathbf{w}'), \mathbf{w} - \mathbf{w}' \rangle \geq \mu\|\mathbf{w} - \mathbf{w}'\|^2, \forall i.$

Note that both assumptions are stronger than Assumptions 3.3.1 and 3.3.2, i.e., Assumption 3.5.1 implies 3.3.1 and Assumption 3.5.2 implies 3.3.2 [KNS16]. Both are very standard assumptions frequently used in the convex optimization literature.

From (3.11), we see that we can lower bound the step size as: $\alpha_k \geq (1-\theta^2)/\nu$. Thus, the step size for big batch SGD is scaled down by *at most* $1-\theta^2$. For simplicity, we assume that the step size is set to this lower bound: $\alpha_k = (1-\theta^2)/\nu_k$. Thus, from Assumptions 3.5.1 and 3.5.2, we can bound ν_k, and also α_k, as follows:

$$\mu \leq \nu_t \leq L \quad \Longrightarrow \quad \frac{1-\theta^2}{L} \leq \alpha_t \leq \frac{1-\theta^2}{\mu}.$$

From Theorem 3.3.1, we see that we have linear convergence with the adaptive step size method when:

$$1 - 2\mu\big(\alpha - \frac{L\alpha^2\beta}{2}\big) \leq 1 - \frac{2(1-\theta^2)}{\kappa} + \beta(1-\theta^2)^2\kappa < 1 \quad \Longrightarrow \quad \kappa^2 \ < \frac{2}{\beta(1-\theta^2)},$$

where $\kappa = L/\mu$ is the condition number. We see that the adaptive step size method enjoys a linear convergence rate when the problem is well-conditioned. In the next section, we talk about ways to deal with poorly-conditioned problems.

3.5.2 Practical implementation

To achieve robustness of the algorithm for poorly conditioned problems, we include a backtracking line search after calculating (3.11), to ensure that the step sizes do not blow up. Further, instead of calculating two gradients on each iteration ($\nabla \tilde{f}_{\mathcal{B}_k}(\mathbf{w}_k)$ and $\nabla \tilde{f}_{\mathcal{B}_k}(\mathbf{w}_{k-1})$), our implementation uses the same batch (and step size) on two consecutive iterations. Thus, one parameter update takes place for each gradient calculation.

We found the step size calculated from (3.11) to be noisy when the batch is small. While this did not affect long-term performance, we perform a smoothing operation to even out the step sizes and make performance more predictable. Let $\tilde{\alpha}_k$ denote the step size calculated from (3.11). Then, the step size on each iteration is given by $\alpha_k = \left(1 - \frac{|\mathcal{B}|}{n}\right)\alpha_{k-1} + \frac{|\mathcal{B}|}{n}\tilde{\alpha}_k$. This ensures that the update is proportional to how accurate the estimate on each iteration is. This simple smoothing operation seemed to work very well in practice as shown in the experimental section. Note that when $|\mathcal{B}_k| = n$, we just use $\alpha_k = 1/\nu_k$. Since there is no noise in the algorithm in this case, we use the optimal step size for a deterministic algorithm. Algorithm 3 shows the complete details.

Algorithm 3 Big batch SGD: with BB step sizes

initialize $\mathbf{w}_0$, initial step size α, initial batch size $K > 1$, batch size increment δ_k, backtracking line search parameter c

while not converged **do**

  Draw random batch with $|\mathcal{B}| = K$; Calculate $V_{\mathcal{B}}$ and $G_{\mathcal{B}} = \nabla \tilde{f}_{\mathcal{B}}(x)$ using (3.8)

  while $\|G_{\mathcal{B}}\|^2 \leq V_{\mathcal{B}}/K$ **do**

    Increase batch size $K \leftarrow K + \delta_K$

    Sample more gradients and update $V_{\mathcal{B}}$ and $G_{\mathcal{B}}$

  end while

  while $\tilde{f}_{\mathcal{B}}(x - \alpha \nabla \tilde{f}_{\mathcal{B}}(x)) > \tilde{f}_{\mathcal{B}}(x) - c\alpha \|\nabla \tilde{f}_{\mathcal{B}}(x)\|^2$ **do**

    $\alpha \leftarrow \alpha/2$

  end while

  $x \leftarrow x - \alpha \nabla \tilde{f}_{\mathcal{B}}(x)$

  if $K < n$ **then**

    Calculate $\tilde{\alpha} = (1 - V_B/(K\|G_{\mathcal{B}}\|^2))/\nu$ using (3.11) and (3.12)

  else

    Calculate $\tilde{\alpha} = 1/\nu$ using (3.12)

  end if

  step size smoothing: $\alpha \leftarrow \alpha(1 - K/n) + \tilde{\alpha} K/n$

  while $\tilde{f}_{\mathcal{B}}(x - \alpha \nabla \tilde{f}_{\mathcal{B}}(x)) > \tilde{f}_{\mathcal{B}}(x) - c\alpha \|\nabla \tilde{f}_{\mathcal{B}}(x)\|^2$ **do**

    $\alpha \leftarrow \alpha/2$

  end while

  $x \leftarrow x - \alpha \nabla \tilde{f}_{\mathcal{B}}(x)$

end while

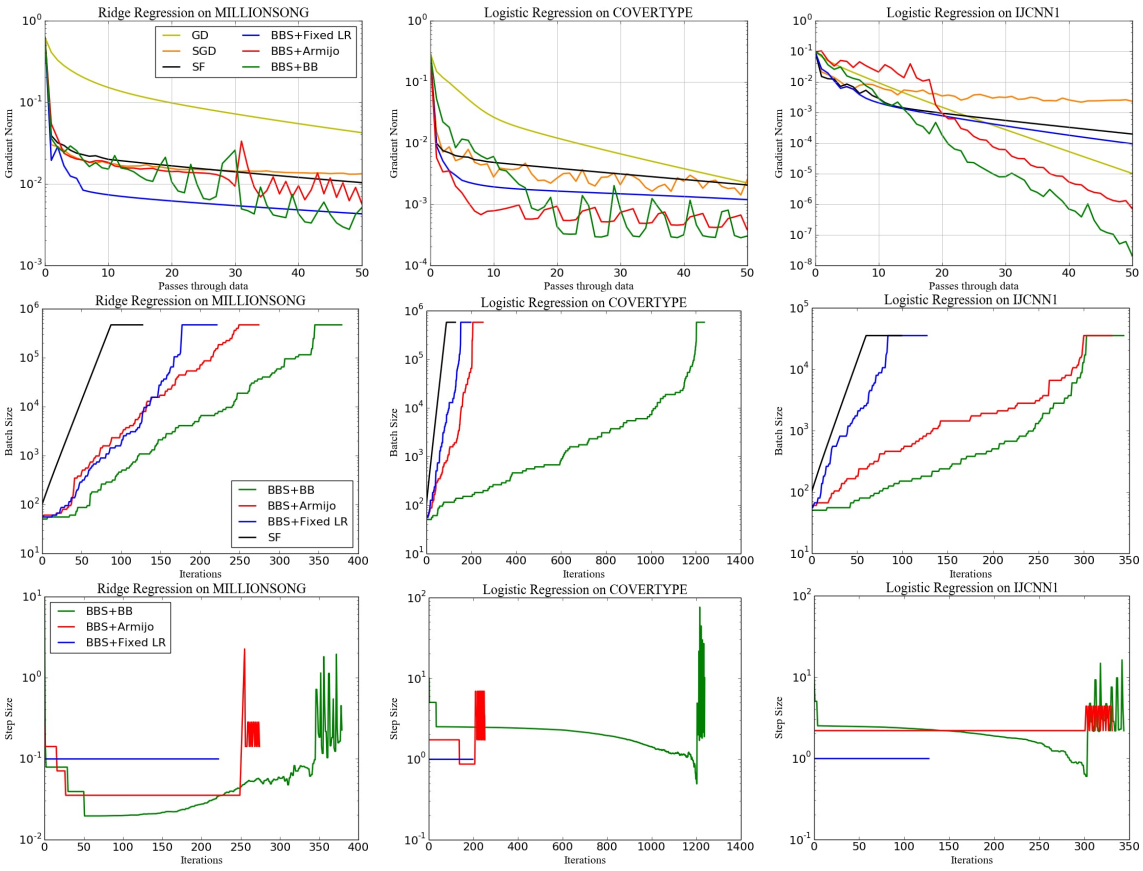


Figure 3.1: Convex experiments. Left to right: Ridge regression on MILLIONSONG; Logistic regression on COVERTYPE; Logistic regression on IJCNN1. The top row shows how the norm of the true gradient decreases with the number of epochs, the middle and bottom rows show the batch sizes and step sizes used on each iteration by the big batch methods. Here 'passes through the data' indicates number of epochs, while 'iterations' refers to the number of parameter updates used by the method (there may be multiple iterations during one epoch).

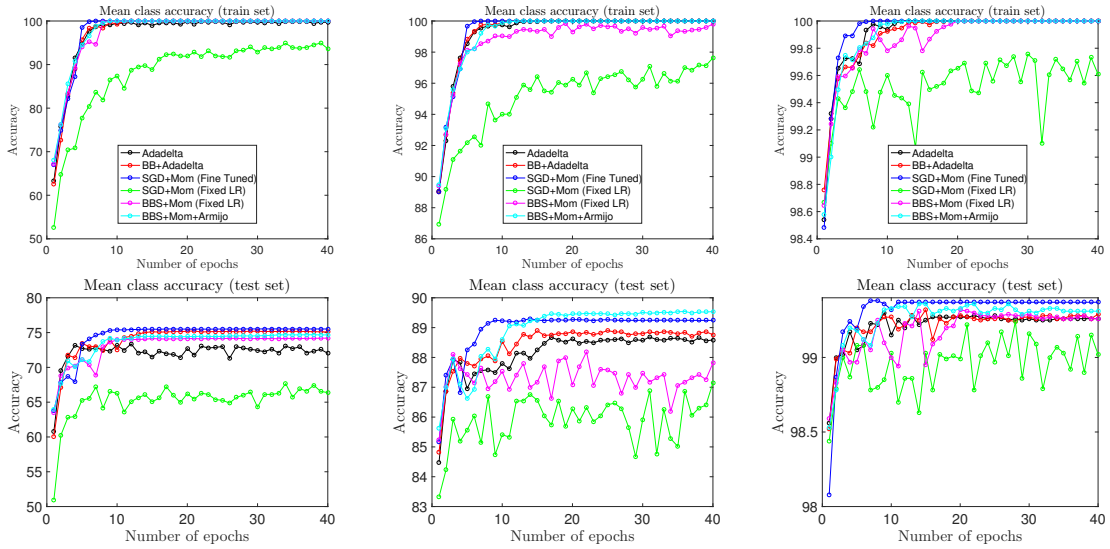


Figure 3.2: Neural Network Experiments. The three columns from left to right correspond to results for CIFAR-10, SVHN, and MNIST, respectively. The top row presents classification accuracies on the training set, while the bottom row presents classification accuracies on the test set.

3.6 Experiments

In this section, we present our experimental results. We explore big batch methods with both convex and non-convex (neural network) experiments on large and high-dimensional datasets.

3.6.1 Convex experiments

For the convex experiments, we test big batch SGD on a binary classification problem with logistic regression and a linear regression problem:

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i \langle \mathbf{x}_i, \mathbf{w} \rangle)),$$

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} (\langle \mathbf{x}_i, \mathbf{w} \rangle - y_i)^2.$$

Figure 3.1 presents the results of our convex experiments on three standard real world datasets: IJCNN1 [Pro01] and COVERTYPE [BD99] for logistic regression, and MILLIONSONG [BMEWL11] for linear regression. As a preprocessing step, we normalize the features for each dataset. We compare deterministic gradient descent (GD) and SGD with step size decay ($\alpha_k = a/(b+k)$) to big batch SGD using a fixed step size (BBS+Fixed LR), with backtracking line search (BBS+Armijo) and with the adaptive step size (3.11) (BBS+BB), as well as the growing batch method described in [FS12] (denoted as SF; while the authors propose a quasi-Newton method, we adapt their algorithm to a first-order method). We selected step size parameters using a comprehensive grid search for all algorithms, except BBS+Armijo and BBS+BB, which require no parameter tuning.

We see that across all three problems, the big batch methods outperform the other algorithms. We also see that both fully automated methods are always comparable to or better than fixed step size methods. The automated methods increase the batch size more slowly than BBS+Fixed LR and SF, and thus, these methods can take more steps with smaller batches, leveraging its advantages longer. Further, note that the step sizes derived by the automated methods are very close to the optimal fixed step size rate.

3.6.2 Neural network experiments

To demonstrate the versatility of the big batch SGD framework, we also present results on neural network experiments. We compare big batch SGD against SGD with finely tuned step size schedules and fixed step sizes. We also compare with Adadelta [Zei12],

and combine the big batch method with AdaDelta (BB+AdaDelta) to show that more complex SGD variants can benefit from growing batch sizes. In addition, we had also compared big batch methods with L-BFGS. However, we found L-BFGS to consistently yield poorer generalization error on neural networks, and thus we omitted these results.

We train a convolutional neural network [LBBH98] (ConvNet) to classify three benchmark image datasets: CIFAR-10 [KH09], SVHN [NWC+11], and MNIST [LBBH98]. Our ConvNet is composed of 4 layers. We use 32×32 pixel images as input. The first layer of the ConvNet contains $16 \times 3 \times 3$, and the second layer contains $256 \times 3 \times 3$ filters. The third and fourth layers are fully connected [LBBH98] with 256 and 10 outputs respectively. Each layer except the last one is followed by a ReLu non-linearity [KSH12] and a max pooling stage [RHBL07] of size 2×2. This ConvNet has over 4.3 million weights.

To compare against fine-tuned SGD, we used a comprehensive grid search on the step size schedule to identify optimal parameters (up to a factor of 2 accuracy). For CIFAR10, the step size starts from 0.5 and is divided by 2 every 5 epochs with 0 step size decay. For SVHN, the step size starts from 0.5 and is divided by 2 every 5 epochs with $1e{-}05$ learning rate decay. For MNIST, the learning rate starts from 1 and is divided by 2 every 3 epochs with 0 step size decay. All algorithms use a momentum parameter of 0.9, and SGD and AdaDelta use mini-batches of size 128.

Fixed step size methods use the default decay rule of the *Torch* library: $\alpha_k = \alpha_0/(1 + 10^{-7}k)$, where α_0 was chosen to be the step size used in the fine-tuned experiments. We also tune the hyper-parameter ρ in the Adadelta algorithm, and we found 0.9, 0.9 and 0.8 to be best-performing parameters for CIFAR10, SVHN and MNIST respec-

tively.

We plot the accuracy on the train and test set vs the number of epochs (full passes through the dataset) in Figure 3.2. We notice that the big batch SGD with backtracking performs better than both Adadelta and SGD (Fixed LR) in terms of both train and test error. Big batch SGD even performs comparably to fine tuned SGD but without the trouble of fine tuning. This is interesting because most state-of-the-art deep networks (like AlexNet [KSH12], VGG Net [SZ14], ResNets [HZRS16a]) were trained by their creators using standard SGD with momentum, and training parameters were tuned over long periods of time (sometimes months). Finally, we note that the big batch AdaDelta performs consistently better than plain AdaDelta on both large scale problems (SVHN and CIFAR-10), and performance is nearly identical on the small-scale MNIST problem.

3.7 Summary

We analyzed and studied the behavior of alternative SGD methods in which the batch size increases over time. Unlike classical SGD methods, in which stochastic gradients quickly become swamped with noise, these "big batch" methods maintain a nearly constant signal to noise ratio of the approximate gradient. As a result, big batch methods are able to adaptively adjust batch sizes without user oversight. The proposed automated methods are shown to be empirically comparable or better performing than other standard methods, but without requiring an expert user to choose learning rates and decay parameters.

Chapter 4: **Distributing SGD using variance reduction**

4.1 Introduction

For truly large datasets, parallel or distributed algorithms are vital, driving interest in SGD variants that parallelize over massive distributed datasets. While there has been quite a bit of recent work in the area of parallel asynchronous SGD algorithms [RRWN11, DCM+12, LHLL15, AD11, LASY14, SS14, ZLS09, BT89, ZWLS10, ZCL15], these methods typically experience substantially reduced marginal benefit as the number of worker nodes increase over a certain limit. Thus, while some of these algorithms scale linearly when the number of worker nodes is small, they are less effective when the data is distributed over hundreds or thousands of nodes.

Moreover, most research in parallel or distributed SGD methods has been focused on the parameter server model of computation [RRWN11, DCM+12, AD11, LASY14, ZLS09], where each update to the centrally stored parameter vector requires a communication phase between the local node and the central server. However, SGD methods tend to become unstable with infrequent communication, and there has been less work in the truly distributed setting where communication costs are high [ZWLS10, ZCL15, MR16]. In this section, we propose to boost the scalability of stochastic optimization algorithms using *variance reduction* techniques, yielding SGD methods that scale linearly over hun-

dreds or thousands of nodes and can train models on massive datasets without the slowdown that existing stochastic methods experience.

Notation For this chapter, let $f_{\tilde{k}}$ denote the stochastic function chosen on the k-th iteration, where $\tilde{k}$ is an index chosen uniformly at random from $\{1, 2, \ldots, n\}$. Thus, using this notation, the regular SGD update can be written as $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \nabla f_{\tilde{k}}(\mathbf{w}_k)$.

Background

Variance reduction (VR) methods [JZ13,DBLJ14,RHS+15,RSB12,DD+14,KLRT14, KR13, XZ14, WCSX13, HAV+15] have recently gained popularity as an alternative to classical SGD. These methods reduce the variance in the stochastic gradient estimates, and are able to maintain a large constant step size to achieve fast convergence to high accuracy.

VR methods exploit the fact that gradient errors are highly correlated between different uses of the same function $f_{\tilde{k}}$. This is done by subtracting an error correction term from $\nabla f_{\tilde{k}}(\mathbf{w}_k)$ that estimates the gradient error from the most recent use of $f_{\tilde{k}}$. Thus the stochastic gradients used by VR methods have the form

$$\tilde{\mathbf{g}}_k = \underbrace{\nabla f_{\tilde{k}}(\mathbf{w}_k)}_{\text{approximate gradient}} \underbrace{- \nabla f_{\tilde{k}}(\tilde{\mathbf{w}}) + \mathbf{g}_{\tilde{\mathbf{w}}}}_{\text{error correction term}}, \tag{4.1}$$

where $\tilde{\mathbf{w}}$ is an old iterate, and $\mathbf{g}_{\tilde{\mathbf{w}}}$ is an approximation of the true gradient $\nabla f(\tilde{\mathbf{w}})$. Typically, $\mathbf{g}_{\tilde{\mathbf{w}}}$ can be kept fixed over an epoch or can be updated cheaply on every iteration. As an example, the SVRG algorithm [JZ13] has an update rule of the form:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha\big(\nabla f_{\tilde{k}}(\mathbf{w}_k) - \nabla f_{\tilde{k}}(\tilde{\mathbf{w}}) + \nabla f(\tilde{\mathbf{w}})\big), \tag{4.2}$$

where $\tilde{\mathbf{w}}$ is chosen to be a recent iterate from the algorithm history and is fixed over 1 or 2 epochs, and $\mathbf{g}_{\tilde{\mathbf{w}}} = \nabla f(\tilde{\mathbf{w}})$ is the true gradient of f at $\tilde{\mathbf{w}}$, which needs to be computed once every 1 or 2 epochs. Another popular VR algorithm, SAGA [DBLJ14], uses the following corrected gradient approximation

$$\tilde{\mathbf{g}}_k = \nabla f_{\tilde{k}}(\mathbf{w}_k) - \nabla f_{\tilde{k}}(\tilde{\mathbf{w}}_{\tilde{k}}) + \frac{1}{n}\sum_{j=1}^{n} \nabla f_j(\tilde{\mathbf{w}}_j), \tag{4.3}$$

where each $\nabla f_j(\tilde{\mathbf{w}}_j)$ denotes the most recent value of ∇f_j and $\tilde{\mathbf{w}}_j$ denotes the iterate at which the most recent ∇f_j was evaluated. In this case $\mathbf{g}_{\tilde{\mathbf{w}}}$ is the average of the $\nabla f_j(\tilde{\mathbf{w}}_j)$ values for all $j \in \{1, 2, \ldots, n\}$. This error correction term reduces the variance in the stochastic gradients, and thus ensures fast convergence. Notice that for both the algorithms, SVRG and SAGA, if $\tilde{k}$ is chosen uniformly at random from $\{1, 2, \cdots, n\}$, we have $\mathbb{E}_{\tilde{k}}\left[\tilde{\mathbf{g}}_k\right] = \nabla f(\mathbf{w}_k)$. Thus, the error correction term has expected value 0 and the approximate gradient $\tilde{\mathbf{g}}_k$ is unbiased for both SVRG and SAGA.

Most work on VR methods has focused on studying their faster convergence rates and better stability properties when compared to classical SGD in the sequential setting. While there have been a few recent papers on parallelizing VR methods, these methods scale poorly in distributed settings and all prior work that we know of has focussed on small-scale parallel or shared memory settings, with the data distributed over 10 or 20 nodes [RHS+15, MPP+15, PLT+16]. These parallel algorithms use a parameter server model of computation, and are based on the assumption that communication costs are low, which may not be true in large-scale heterogenous distributed computing environments. The fact that the error correction term reduces the variance in the stochastic gradients, however, seems to indicate that distributed VR methods could be helpful in distributed

settings. In particular, the variance-reduced gradients would help in dealing with the problems of instability and slower convergence faced by regular stochastic methods when the frequency of communication between the server and the local nodes is increased.

Contributions

In this work, we use variance reduction to dramatically boost the performance of SGD in the distributed setting. We do this by exploiting the dependence of VR methods on the gradient correction term $\mathbf{g}_{\tilde{\mathbf{w}}}$. We allow many local worker nodes to run simultaneously, while communicating with the central server only through the exchange of this central error correction term and the locally stored iterates. The proposed schemes allow many asynchronous processes to work towards a central solution with minimal communication, while simultaneously benefitting from the fast convergence provided by VR.

This work has four main contributions:

- First, we present a new VR algorithm *CentralVR*, built on SAGA, that is robust to noise and variance in the dataset. We propose synchronous (*CentralVR-Sync*) and asynchronous (*CentralVR-Async*) variations of *CentralVR* which can linearly scale up over massive datasets using hundreds of cores.

- Second, we theoretically study the convergence of *CentralVR* and prove linear convergence of the method with constant step sizes.

- Third, we propose distributed versions of the existing popular VR algorithms, SVRG and SAGA, that are robust to high communication latency between the worker nodes and the central server, and can scale over large distributed settings ranging

over hundreds of nodes. Table 4.1 summarizes the distributed algorithms proposed in this section and their storage and computation requirements.

- Finally, we present empirical results over different models and datasets that show that these distributed algorithms can be trained on massive highly distributed datasets in far less time than existing state-of-the-art stochastic optimization methods. Performance of all these distributed methods scales linearly up to hundreds of workers with low communication frequency. We show empirically that the proposed methods converge much faster than competing options.

Table 4.1: Distributed Algorithms Proposed

Proposed Algorithm	Asynchronous?	Storage (No. of gradients)	Gradients/Iteration
CentralVR-Sync	No	n	1
CentralVR-Async	Yes	n	1
Distributed SVRG	No	2	2.5
Distributed SAGA	Yes	n	1

4.2 CentralVR algorithm: single-worker case

We begin by proposing our new VR scheme, CentralVR, in the single-worker case. As we will see later, the proposed method has a natural generalization to the distributed setting that has low communication requirements.

4.2.1 Algorithm overview

Our proposed VR scheme is divided into epochs, with n updates taking place in each epoch. Let the iterates generated in the m-th epoch be written as $\{\mathbf{w}_j^m\}_{j=1}^n$. Also let $\tilde{\mathbf{w}}_l^m$ denote the iterate at which the l-th data index was most recently used before the $m+1$-th epoch (i.e., on or before the m-th epoch). Then, the update for ***CentralVR*** is:

$$\mathbf{w}_{k+1}^{m+1} = \mathbf{w}_k^{m+1} - \alpha \mathbf{v}_k^{m+1}, \tag{4.4}$$

$$\mathbf{v}_k^{m+1} = \nabla f_{\tilde{k}}(\mathbf{w}_k^{m+1}) - \nabla f_{\tilde{k}}(\tilde{\mathbf{w}}_{\tilde{k}}^m) + \frac{1}{n}\sum_{j=1}^{n} \nabla f_j(\tilde{\mathbf{w}}_j^m). \tag{4.5}$$

Denote $\bar{\mathbf{g}}^m = \frac{1}{n}\sum_{j=1}^n \nabla f_j(\tilde{\mathbf{w}}_j^m)$. Thus, $\bar{\mathbf{g}}^m$ is the average of the gradients of all component functions $\{\nabla f_j\}_{j=1}^n$, each evaluated at the most recent iterate $\{\tilde{\mathbf{w}}_j^m\}_{j=1}^n$ at which the corresponding function was used on or before the m-th epoch. These gradients are stored in a table, and the average gradient $\bar{\mathbf{g}}^m$ is updated at the end of each epoch, i.e., after every n parameter updates. Note that if $\tilde{k}$ is chosen uniformly at random from the set $\{1, 2, \cdots, n\}$ on each iteration k, then we have $\mathbb{E}_{\tilde{k}}\big[\nabla f_{\tilde{k}}(\tilde{\mathbf{w}}_{\tilde{k}}^m)\big] = \bar{\mathbf{g}}^m$. Thus, the error correction term has expected value 0, and $\mathbb{E}\big[\mathbf{v}_k^{m+1}\big] = \nabla f(\mathbf{w}_k^{m+1})$, i.e., the approximate gradient $\mathbf{v}_k^{m+1}$ is unbiased.

4.2.2 Permutation sampling

In practical implementations, it is natural to consider a random permutation of the data indices on every epoch, rather than uniformly choosing a random index on every iteration. Thus, on each epoch, a random permutation of the data indices is chosen and a pass is made over the entire dataset, resulting in n updates, one per data sample. Permu-

tation sampling often outperforms uniform random sampling empirically [Bot09, Bot12], although theoretical justification for this is still limited (see [GOP15, Sha16] for some recent results).

As an alternative to uniform random sampling, *CentralVR* can leverage random permutations over the data indices. Let π^m denote a random permutation of the data indices $\{1, 2, \cdots, n\}$ for the m-th epoch, with π_j^m denoting the data index chosen in the j-th iteration in the m-th epoch. Thus, now $\tilde{\mathbf{w}}_m^l$ denotes the iterate corresponding to the point when the l-th data index was chosen in the m-th epoch. The update rule with the random permutation is given by (4.4) and (4.5), with $\tilde{k} = \pi_k^{m+1}$.

Summing (4.4) over all $k = 0, 1, \cdots, n-1$, we get $\sum_{k=0}^{n-1} \mathbf{v}_k^{m+1} = \sum_{k=0}^{n-1} \nabla f_k(\tilde{\mathbf{w}}_k^{m+1})$. Thus, summing (4.4) over all $k = 0, 1, \cdots, n-1$, using the telescoping sum in $\mathbf{w}_k^{m+1}$, and using the convention that $\mathbf{w}_0^{m+1} = x_m^n$, we get

$$\mathbf{w}_0^{m+2} = \mathbf{w}_0^{m+1} - \alpha \sum_{j=1}^{n} \nabla f_j\left(\tilde{\mathbf{w}}_j^{m+1}\right). \tag{4.6}$$

Equation (4.6) shows the update rule in terms of the iterates at the ends of the epochs. Thus, over an epoch, the average gradient accumulated by *CentralVR* is unbiased and thus is a good estimate of the true gradient. This average gradient term can be accumulated cheaply during an epoch, without any noticeable overhead.

4.2.3 Algorithm details for CentralVR

The detailed steps of *CentralVR* are listed in Algorithm 4. Note, the stored gradients and the average gradient term $\mathbf{g}_{\tilde{\mathbf{w}}}$ are initialized using a single epoch of "vanilla" SGD with no VR correction.

Algorithm 4 *CentralVR* Algorithm: single worker case

parameters learning rate α

initialize $\mathbf{w}$, $\{\nabla f_j(\tilde{\mathbf{w}}_j)\}_j$, and $\bar{\mathbf{g}}$ using plain SGD

while not converged **do**

  $\tilde{\mathbf{g}} \leftarrow 0$

  set π: random permutation of indices $1, 2, \cdots, n$

  for k in $\{1, \ldots, n\}$ **do**

    set: $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \alpha\big(\nabla f_{\pi_k}(\mathbf{w}_k) - \nabla f_{\pi_k}(\tilde{\mathbf{w}}_{\pi_k}) + \bar{\mathbf{g}}\big)$

    accumulate average: $\tilde{\mathbf{g}} \leftarrow \tilde{\mathbf{g}} + \nabla f_{\pi_k}(\mathbf{w}_k)/n$

    store gradient: $\nabla f_{\pi_k}(\tilde{\mathbf{w}}_{\pi_k}) \leftarrow \nabla f_{\pi_k}(\mathbf{w}_k)$

  end for

  set average gradient for next epoch: $\bar{\mathbf{g}} \leftarrow \tilde{\mathbf{g}}$

end while

CentralVR builds on the SAGA method. SAGA relies on the update rule (4.3), which requires an average over a large number of iterates ($\mathbf{g}_{\tilde{\mathbf{w}}} = \frac{1}{n}\sum_j \nabla f_j(\mathbf{w}_j)$) to be continuously updated on every iteration. In the distributed setting, where the vector $\mathbf{g}_{\tilde{\mathbf{w}}}$ must be shared across nodes, maintaining an up-to-date average requires large amounts of communication. This makes SAGA less stable in distributed implementations when the communication frequency is decreased. Updating $\mathbf{g}_{\tilde{\mathbf{w}}}$ only occasionally (as we do in the distributed variants of *CentralVR* below) translates into significant communication savings in the distributed setting.

CentralVR has the same time and space complexities as SAGA. Namely, on every iteration, 1 gradient computation is required, similar to SGD, and the n gradients

$\{\nabla f_j(\tilde{\mathbf{w}}_j^m)\}_{j=1}^n$ also need to be stored. Note that this is not always a significant storage requirement, since for models like logistic regression and ridge regression only a single number is required to be stored corresponding to each gradient.

4.3 Convergence analysis

We now present convergence bounds for Algorithm 4. We make the following standard assumptions about the function when studying convergence properties. First, each f_i is strongly convex with strong convexity constant μ:

$$f_i(\mathbf{w}) \geq f_i(\mathbf{w}') + \langle \nabla f_i(\mathbf{w}'), \mathbf{w} - \mathbf{w}' \rangle + \frac{\mu}{2}\|\mathbf{w} - \mathbf{w}'\|^2. \tag{4.7}$$

Second, each f_i has Lipschitz continuous gradients with Lipschitz constant L so that

$$f_i(\mathbf{w}) \leq f_i(\mathbf{w}') + \langle \nabla f_i(\mathbf{w}'), \mathbf{w} - \mathbf{w}' \rangle + \frac{L}{2}\|\mathbf{w} - \mathbf{w}'\|^2. \tag{4.8}$$

We now present our main result.

Theorem 4.3.1. *Consider* CentralVR *with data index $\tilde{k}$ drawn uniformly at random (with replacement) on each iteration k. Define $\rho := \max\left(1 - \alpha\mu, \frac{2L^2\alpha}{\mu(1-2L\alpha)}\right)$. If the step size α is small enough such that $0 < \rho < 1$, then we have the following bound:*

$$\left\|\mathbf{w}_0^{m+2} - \mathbf{w}^\star\right\|^2 + c\big(\overline{f(\mathbf{w}^{m+1})} - f(\mathbf{w}^\star)\big) \leq \rho\Big(\left\|\mathbf{w}_0^{m+1} - \mathbf{w}^\star\right\|^2 + c\big(\overline{f(\tilde{\mathbf{w}}^m)} - f(\mathbf{w}^\star)\big)\Big),$$

where $c = 2n\alpha(1 - 2L\alpha)$ and we define $\overline{f(\mathbf{w}^m)} := \frac{1}{n}\sum_{k=0}^{n-1} f(\mathbf{w}_k^m)$. In other words, the method converges linearly.

We first start with two lemmas that will be useful in the proof for Theorem 4.3.1.

Lemma 4.3.1. *For any f defined as $f := \frac{1}{n}\sum_{i=1}^{n} f_i$, where each f_i satisfies* (4.7) *and* (4.8)*, and on conditioning on any* $\mathbf{w}$*, we have*

$$\mathbb{E}\big\|\nabla f_j(\mathbf{w}) - \nabla f_j(\mathbf{w}^\star)\big\|^2 \leq 2L(f(\mathbf{w}) - f(\mathbf{w}^\star)),$$

where j is sampled uniformly at random from $\{1, 2, \ldots, n\}$ and $\mathbf{w}^\star$ is the minimizer of f.

Proof. A standard result used frequently in the convex optimization literature is:

$$\|\nabla f_j(\mathbf{w}) - \nabla f_j(\mathbf{w}^\star)\|^2 \leq 2L(f_j(\mathbf{w}) - f_j(\mathbf{w}^\star) - \langle \nabla f_j(\mathbf{w}^\star), \mathbf{w} - \mathbf{w}^\star\rangle),$$

where f_j is L-Lipschitz smooth. A proof for this inequality can be found in [Nes13] (Theorem 2.1.5 on page 56). Since j is sampled uniformly at random from $\{1, 2, \ldots, n\}$, we can write: $\mathbb{E}_j(f_j(\mathbf{w}) - f_j(\mathbf{w}^\star) - \langle \nabla f_j(\mathbf{w}^\star), \mathbf{w} - \mathbf{w}^\star\rangle) = f(\mathbf{w}) - f(\mathbf{w}^\star)$, using the property that $\nabla f(\mathbf{w}^\star) = 0$. The result follows. ■

Lemma 4.3.2. *For any f defined as $f := \frac{1}{n}\sum_{i=1}^{n} f_i$, where each f_i satisfies* (4.7) *and* (4.8)*, and for any* $\mathbf{w}$ *and* i *we have*

$$\big\|\nabla f_i(\mathbf{w}) - \nabla f_i(\mathbf{w}^\star)\big\|^2 \leq \frac{2L^2}{\mu}\big(f(\mathbf{w}) - f(\mathbf{w}^\star)\big),$$

where $\mathbf{w}^\star$ denotes the minimizer of f.

Proof. A standard result used frequently in the convex optimization literature is:

$$\|\nabla f_i(\mathbf{w}) - \nabla f_i(\mathbf{w}^\star)\|^2 \leq L^2\|x - \mathbf{w}^\star\|^2,$$

where f_i is L-Lipschitz smooth. A proof for this inequality can be found in [Nes13] (Theorem 2.1.5 on page 56). From (4.7), we get:

$$\|\mathbf{w} - \mathbf{w}^\star\|^2 \leq \frac{2}{\mu}\big(f(\mathbf{w}) - f(\mathbf{w}^\star) - \langle \mathbf{w} - \mathbf{w}^\star, \nabla f(\mathbf{w}^\star)\rangle\big) = \frac{2}{\mu}\big(f(\mathbf{w}) - f(\mathbf{w}^\star)\big),$$

using the property that $\nabla f(\mathbf{w}^\star) = 0$. The desired result follows immediately. ■

We now move on to the proof of Theorem 4.3.1.

Proof. Let the update rule for *CentralVR* be denoted as

$$\mathbf{w}_{k+1}^{m+1} = \mathbf{w}_k^{m+1} - \alpha \mathbf{v}_k^{m+1},$$
$$\mathbf{v}_k^{m+1} = \Big[\nabla f_{\tilde{k}}(\mathbf{w}_k^{m+1}) - \nabla f_{\tilde{k}}(\tilde{\mathbf{w}}_{\tilde{k}}^m) + \frac{1}{n}\sum_j \nabla f_j(\tilde{\mathbf{w}}_j^m)\Big].$$

In this proof, we assume that the data indices are accessed randomly with replacement. Thus, $\tilde{\mathbf{w}}_m^{\tilde{k}}$ denotes the last iterate when the $\tilde{k}$-th data index was chosen in or before the m-th epoch. Thus, conditioning on all $\mathbf{w}$, $\mathbf{v}_k^{m+1}$ is an unbiased estimator of the true gradient at $\mathbf{w}_k^{m+1}$, i.e., $\mathbb{E}\Big[\mathbf{v}_k^{m+1}\Big] = \nabla f(\mathbf{w}_k^{m+1})$. Conditioned on all history (all $\mathbf{w}$), we first begin with the standard identity:

$$\begin{aligned}\mathbb{E}\big[\|\mathbf{w}_{k+1}^{m+1} - \mathbf{w}^\star\|^2\big] &= \mathbb{E}\Big[\|\mathbf{w}_k^{m+1} - \alpha\mathbf{v}_k^{m+1} - \mathbf{w}^\star\|^2\Big] \\ &= \|\mathbf{w}_k^{m+1} - \mathbf{w}^\star\|^2 - 2\alpha\langle \mathbf{w}_k^{m+1} - \mathbf{w}^\star, \nabla f(\mathbf{w}_k^{m+1})\rangle + \alpha^2\mathbb{E}\|\mathbf{v}_{m+1}^k\|^2.\end{aligned} \tag{4.9}$$

We now bound (4.9). Using the definition of strong convexity in (4.7), we can simplify the inner product term in (4.9) as

$$\langle \mathbf{w}^\star - \mathbf{w}_k^{m+1}, \nabla f(\mathbf{w}_k^{m+1})\rangle \leq -(f(\mathbf{w}_k^{m+1}) - f(\mathbf{w}^\star)) - \frac{\mu}{2}\|\mathbf{w}^\star - \mathbf{w}_k^{m+1}\|^2. \tag{4.10}$$

We now bound the magnitude of the gradient term in (4.9):

$$\begin{aligned}&\mathbb{E}\|\mathbf{v}_k^{m+1}\|^2 \\ =&\mathbb{E}\Big\|\nabla f_{\tilde{k}}(\mathbf{w}_k^{m+1}) - \nabla f_{\tilde{k}}(\tilde{\mathbf{w}}_{\tilde{k}}^m) + \frac{1}{n}\sum_j \nabla f_j(\tilde{\mathbf{w}}_j^m)\Big\|^2 \\ =&\mathbb{E}\Big\|\nabla f_{\tilde{k}}(\mathbf{w}_k^{m+1}) - \nabla f_{\tilde{k}}(\mathbf{w}^\star) + \nabla f_{\tilde{k}}(\mathbf{w}^\star) - \nabla f_{\tilde{k}}(\tilde{\mathbf{w}}_{\tilde{k}}^m) + \frac{1}{n}\sum_j \nabla f_j(\tilde{\mathbf{w}}_j^m)\Big\|^2\end{aligned}$$

$$
\begin{aligned}
&\leq 2\mathbb{E}\left\|\nabla f_{\tilde{k}}(\mathbf{w}_k^{m+1}) - \nabla f_{\tilde{k}}(\mathbf{w}^\star)\right\|^2 + 2\mathbb{E}\Big\|\nabla f_{\tilde{k}}(\tilde{\mathbf{w}}_{\tilde{k}}^m) - \nabla f_{\tilde{k}}(\mathbf{w}^\star) \\
&\quad - \Big(\frac{1}{n}\sum_j \nabla f_j(\tilde{\mathbf{w}}_j^m) - \frac{1}{n}\sum_j \nabla f_j(\mathbf{w}^\star)\Big)\Big\|^2 \\
&= 2\mathbb{E}\left\|\nabla f_{\tilde{k}}(\mathbf{w}_k^{m+1}) - \nabla f_{\tilde{k}}(\mathbf{w}^\star)\right\|^2 + 2\mathbb{E}\Big\|\nabla f_{\tilde{k}}(\tilde{\mathbf{w}}_{\tilde{k}}^m) - \nabla f_{\tilde{k}}(\mathbf{w}^\star) - \mathbb{E}\Big[\nabla f_{\tilde{k}}(\tilde{\mathbf{w}}_{\tilde{k}}^m) - \nabla f_{\tilde{k}}(\mathbf{w}^\star)\Big]\Big\|^2 \\
&\leq 2\mathbb{E}\left\|\nabla f_{\tilde{k}}(\mathbf{w}_k^{m+1}) - \nabla f_{\tilde{k}}(\mathbf{w}^\star)\right\|^2 + 2\mathbb{E}\left\|\nabla f_{\tilde{k}}(\tilde{\mathbf{w}}_{\tilde{k}}^m) - \nabla f_{\tilde{k}}(\mathbf{w}^\star)\right\|^2 \\
&\leq 4L\big(f(\mathbf{w}_k^{m+1}) - f(\mathbf{w}^\star)\big) + \frac{4L^2}{\mu}\mathbb{E}\big(f(\tilde{\mathbf{w}}_{\tilde{k}}^m) - f(\mathbf{w}^\star)\big). \qquad (4.11)
\end{aligned}
$$

The second equality uses the property that $\nabla f(\mathbf{w}^\star) = 0$. The first inequality uses the property that $\|\mathbf{a}+\mathbf{b}\|^2 \leq 2\|\mathbf{a}\|^2 + 2\|\mathbf{b}\|^2$. The second inequality uses $\mathbb{E}\|\boldsymbol{\phi} - \mathbb{E}\boldsymbol{\phi}\|^2 = \mathbb{E}\|\boldsymbol{\phi}\|^2 - \|\mathbb{E}\boldsymbol{\phi}\|^2 \leq \mathbb{E}\|\boldsymbol{\phi}\|^2$, for any random vector $\boldsymbol{\phi}$. The third inequality follows from Lemma 4.3.1 and Lemma 4.3.2.

We now plug (4.10) and (4.11) into (4.9) and rearrange:

$$
\begin{aligned}
&\mathbb{E}\Big[\|\mathbf{w}_{k+1}^{m+1} - \mathbf{w}^\star\|^2\Big] + 2\alpha(1-2L\alpha)\big(f(\mathbf{w}_k^{m+1}) - f(\mathbf{w}^\star)\big) \\
&\leq \left\|\mathbf{w}_k^{m+1} - \mathbf{w}^\star\right\|^2 - \alpha\mu\left\|\mathbf{w}_k^{m+1} - \mathbf{w}^\star\right\|^2 + \frac{4L^2\alpha^2}{\mu}\mathbb{E}\big(f(\tilde{\mathbf{w}}_{\tilde{k}}^m) - f(\mathbf{w}^\star)\big). \qquad (4.12)
\end{aligned}
$$

Taking expectation on all $\mathbf{w}$ and summing (4.12) over all $k = 0, 1, \ldots, n-1$, we get a telescoping sum in $\left\|\mathbf{w}_k^{m+1} - \mathbf{w}^\star\right\|^2$ that yields:

$$
\begin{aligned}
&\mathbb{E}\left\|\mathbf{w}_0^{m+2} - \mathbf{w}^\star\right\|^2 + 2n\alpha(1-2L\alpha)\mathbb{E}\big(\overline{f(\mathbf{w}^{m+1})} - f(\mathbf{w}^\star)\big) \\
&\leq \mathbb{E}\left\|\mathbf{w}_0^{m+1} - \mathbf{w}^\star\right\|^2 - \alpha\mu\sum_{k=0}^{n-1}\mathbb{E}\left\|\mathbf{w}_k^{m+1} - \mathbf{w}^\star\right\|^2 + \frac{4nL^2\alpha^2}{\mu}\mathbb{E}\big(\overline{f(\tilde{\mathbf{w}}^m)} - f(\mathbf{w}^\star)\big), \qquad (4.13)
\end{aligned}
$$

where we use the convention $\mathbf{w}_n^m = \mathbf{w}_0^{m+1}$, and define $\overline{f(\mathbf{w}^m)}$ as $\overline{f(\mathbf{w}^m)} := \frac{1}{n}\sum_{k=0}^{n-1} f(\mathbf{w}_k^m)$.

We now observe that

$$
\mathbb{E}\left\|\mathbf{w}_0^{m+1} - \mathbf{w}^\star\right\|^2 \leq \sum_{k=0}^{n-1}\mathbb{E}\left\|\mathbf{w}_k^{m+1} - \mathbf{w}^\star\right\|^2.
$$

Thus we can rewrite

$$-\alpha\mu\sum_{k=0}^{n-1}\mathbb{E}\big\|\mathbf{w}_k^{m+1}-\mathbf{w}^\star\big\|^2 \leq -\alpha\mu\mathbb{E}\big\|\mathbf{w}_0^{m+1}-\mathbf{w}^\star\big\|^2.$$

Substituting this in (4.13), we get:

$$\begin{aligned}&\mathbb{E}\big\|\mathbf{w}_0^{m+2}-\mathbf{w}^\star\big\|^2+2n\alpha(1-2L\alpha)\mathbb{E}\big(\overline{f(\mathbf{w}^{m+1})}-f(\mathbf{w}^\star)\big)\\&\leq(1-\alpha\mu)\mathbb{E}\big\|\mathbf{w}_0^{m+1}-\mathbf{w}^\star\big\|^2+\frac{4nL^2\alpha^2}{\mu}\mathbb{E}\big(\overline{f(\tilde{\mathbf{w}}^m)}-f(\mathbf{w}^\star)\big).\end{aligned}$$

We can rewrite this to get:

$$\begin{aligned}&\mathbb{E}\big\|\mathbf{w}_0^{m+2}-\mathbf{w}^\star\big\|^2+2n\alpha(1-2L\alpha)\mathbb{E}\big(\overline{f(\mathbf{w}^{m+1})}-f(\mathbf{w}^\star)\big)\\&\leq\rho\Big(\mathbb{E}\big\|\mathbf{w}_0^{m+1}-\mathbf{w}^\star\big\|^2+2n\alpha(1-2L\alpha)\mathbb{E}\big(\overline{f(\tilde{\mathbf{w}}^m)}-f(\mathbf{w}^\star)\big)\Big),\end{aligned}$$

where $\rho=\max\Big(1-\alpha\mu,\frac{4nL^2\alpha^2}{2n\mu\alpha(1-2L\alpha)}\Big)$. The result immediately follows. ■

Remark on step size restrictions From Theorem 4.3.1, notice that *CentralVR* converges linearly when the step size α is small enough such that

$$\alpha<\min\Big(\frac{1}{\mu},\frac{1}{2L},\frac{\mu}{2L(L+\mu)}\Big).$$

Since $L\geq\mu$, we see that this condition is satisfied whenever $\alpha<\frac{\mu}{2L(L+\mu)}$.

4.4 Distributed algorithms

We now consider the distributed setting, with a single central server and p local clients, each of which contains a portion of the data set. In this setting, the data is decomposed into disjoint subsets $\{\Omega_s\}$, where s denotes a particular local client, and

$\sum_s |\Omega_s| = n$. We denote the i-th function stored on client s as f_i^s. Our goal is to minimize the global objective function of the form

$$f(\mathbf{w}) = \frac{1}{n} \sum_{s=1}^{p} \sum_{j=1}^{|\Omega_s|} f_j^s(\mathbf{w}).$$

We consider a centralized setting, where the clients can only communicate with the central server, and our goal is to derive stochastic algorithms that scale linearly to high p, while remaining stable even under low communication frequencies between local and central nodes.

4.4.1 Synchronous version

CentralVR naturally extends to the distributed synchronous setting, and is presented in Algorithm 5. To distinguish the algorithm from the single worker case, we call it *CentralVR-Sync*. On each epoch, the local nodes first retrieve a copy of the central iterate $\mathbf{w}$, and also $\mathbf{g}_{\tilde{\mathbf{w}}}$, which represents the averaged gradient over all data. The *CentralVR* method is then performed on each node, and the most recent gradient for each data point $\nabla f_{\tilde{k}}^s(\tilde{\mathbf{w}}_{\tilde{k}})$ is stored. By sharing $\mathbf{g}_{\tilde{\mathbf{w}}}$ across nodes, we ensure that the local gradient updates utilize global gradient information from remote nodes. This prevents the local node from drifting far away from the global solution, even if each local node runs for one whole epoch before communicating back with the central server.

In *CentralVR-Sync*, each local node performs local updates for one epoch, or $|\Omega_s|$ iterations, before communicating with the server. This is a rather low communication frequency compared to a parameter server model of computation in which updates are continuously streamed to the central node. This makes a significant difference in runtimes

when the number of local nodes is large, as shown in later sections.

4.4.2 Asynchronous version

The synchronous algorithm can be extended very easily to the asynchronous case, *CentralVR-Async*, as shown in Algorithm 6. In *CentralVR-Async*, the central server keeps a copy of the current iterate $\mathbf{w}$ and average gradient $\bar{\mathbf{g}}$. The key idea for *CentralVR-Async* is that, once a local node completes an epoch, it sends the *change* in the local averages, given by $\Delta\mathbf{w}^s$ and $\Delta\bar{\mathbf{g}}^s$, over the last epoch to the central server. This change is added to the global $\mathbf{w}$ and $\bar{\mathbf{g}}$ to update the parameters stored on the central server. Thus, when the central server receives parameters from a local node s, it performs the updates:

$$\mathbf{w} = \mathbf{w} + \frac{1}{p}\Delta\mathbf{w}^s \quad \text{and} \quad \bar{\mathbf{g}} = \bar{\mathbf{g}} + \frac{1}{p}\Delta\bar{\mathbf{g}}^s,$$

where $\Delta\mathbf{w}^s$ and $\Delta\bar{\mathbf{g}}^s$ are given by

$$\Delta\mathbf{w}^s = \{\mathbf{w}_n^{m+1} - \mathbf{w}_n^m\}_s \quad \text{and}$$
$$\Delta\bar{\mathbf{g}}^s = \left\{ \frac{1}{|\Omega_s|}\sum_{j\in\Omega_s} \nabla f_j^s(\tilde{\mathbf{w}}_j^{m+1}) - \frac{1}{|\Omega_s|}\sum_{j\in\Omega_s} \nabla f_j^s(\tilde{\mathbf{w}}_j^m) \right\}_s.$$

Sending the change in the local parameter values rather than the local parameters themselves ensures that, when updating the central parameter, the previous contribution to the average from that local worker is just replaced by the new contribution. Thus, a fast working local node does not bias the global average solution toward its local solution with an excessive number of updates. This makes the algorithm more robust to heterogenous computing environments where nodes work at disparate speeds.

The proposed *CentralVR* scheme has several advantages. It does not require a full gradient computation as in SVRG, and thus can be made fully asynchronous. Moreover,

since the average gradient $\mathbf{g}_{\tilde{\mathbf{w}}}$ in the error correction term is updated only at the end of an epoch, communication periods can be increased between the central server and the local nodes, while still maintaining fast and stable convergence.

4.5 Distributed variants of SVRG and SAGA

In this section, we propose distributed variants of popular variance reduction methods: SVRG and SAGA. The properties of these variants are overviewed in Table 4.1.

4.5.1 Distributed SVRG

In this section, we present a distributed version of SVRG appropriate for distributed scenarios with high communication delays. Recently, in [RHS+15], the authors presented an asynchronous distributed version of SVRG using a parameter server model of computation. In SVRG, the average gradient term is $\mathbf{g}_{\tilde{\mathbf{w}}} = \nabla f(\tilde{\mathbf{w}})$ as shown in (4.2). This correction term is very accurate because it uses the entire dataset. This would indicate that the algorithm would be robust to high communication periods between the local nodes and the server.

However, a truly asynchronous method is not possible with SVRG since a synchronization step is unavoidable when computing the full gradient. Thus, in this section, we present a synchronous variant of SVRG in Algorithm 7. We define an additional parameter τ to denote the communication period, i.e., the number of updates to run on each local node before communicating with the central server.

The true gradient $\bar{\mathbf{g}}$ is maintained across all nodes throughout the whole commu-

nication period τ, thus ensuring that the local workers stay close to the desired solution, even when τ is large. After τ updates, the current iterate $\mathbf{w}^s$ on each local node s is averaged on the central server to get $\mathbf{w}$. The true gradient is evaluated at $\mathbf{w}$, i.e., $\bar{\mathbf{g}} = \nabla f(\mathbf{w})$, and $\bar{\mathbf{w}} = \mathbf{w}$ is used on each local node during the next epoch.

4.5.2 Distributed SAGA

The update rule for SAGA is given in (4.3). Since there is no synchronization step required as in SVRG, there is a very natural asynchronous version of the algorithm under the parameter server model of computation. A linear convergence proof has been presented for the parameter server model of SAGA (see Theorem 3 in [RHS+15]). However, this work does not contain any empirical studies of the method. The parameter server framework is a very natural generalization of SAGA, however it has very high bandwidth requirements for large numbers of nodes.

Algorithm 8 presents an asynchronous version of SAGA with lower communication frequency. Like SVRG, we define a communication period parameter τ which determines the number of iterations to run on each machine before central communication.

In the SAGA algorithm, the average gradient term $\bar{\mathbf{g}}$ is updated on each iteration. Thus, as local iterations progress, the average gradient evolves differently on each local node. This makes the algorithm less robust to higher communication periods τ. As the communication period increases, the local nodes drift farther apart from each other and the global solution. Thus, the learning rate needs to shrink as τ increases over a certain limit. This in turn slows down convergence. For this reason, distributed SAGA is less

tolerant to long communication periods than the Algorithms in Sections 4.4 and 4.5.1. However, it still has fast convergence for much higher communication periods than existing stochastic schemes.

The asynchronous SAGA method (Algorithm 8) is built on the same idea as the proposed asynchronous algorithm: running averages are kept on each local node, and at the end of an epoch the *change* in the parameter values are sent to the central server. This makes the algorithm more robust when local nodes work at heterogenous speeds.

In our distributed SAGA algorithm, care has to be taken while updating the average gradient $\bar{\mathbf{g}}$. Note that $\bar{\mathbf{g}}$ is averaged over the whole dataset. Thus, when replacing the gradient value at the current index $\tilde{k}$, the update is scaled down by a factor of n (the total number of global samples, as opposed to $|\Omega_s|$, the number of local samples). At the end of a local epoch, the average of the stored gradients on each local node is sent back to the central server, along with the current estimate $\mathbf{w}$. This ensures that the average gradient term on the central server $\bar{\mathbf{g}}$ is built from the most recent gradient computations at each index.

4.6 Empirical results

In this section, we present the empirical performance of the proposed methods, both in sequential and distributed settings. We benchmark the methods for two test problems: first, a binary classification problem with ℓ_2-regularized logistic regression where each f_i is of the form: $f_i(\mathbf{w}) = \log\left(1 + \exp(-y_i\langle \mathbf{x}_i, \mathbf{w}\rangle)\right) + \lambda\|\mathbf{w}\|^2$, where feature vector $\mathbf{x}_i \in \mathbb{R}^d$ has label $y_i \in \mathbb{R}$. We also consider a ridge regression problem of the form

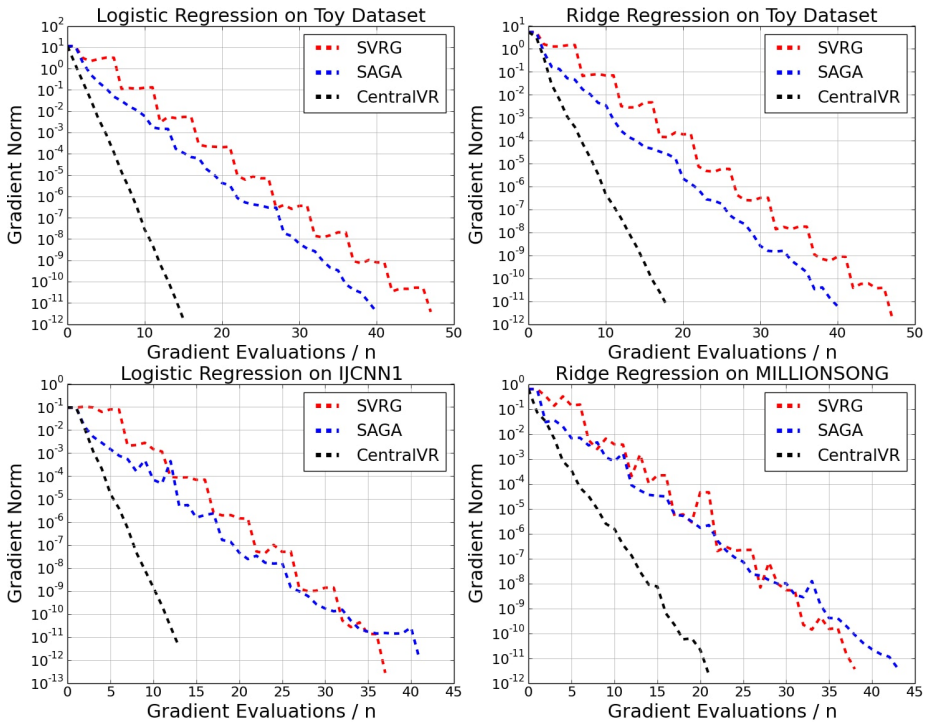


Figure 4.1: Single Worker Results. Logistic regression on toy dataset; Ridge regression on toy data; Logistic regression on IJCNN1 dataset; Ridge regression on MILLIONSONG dataset; In each case *CentralVR* converges much faster than SVRG and SAGA.

$f_i(\mathbf{w}) = (\langle \mathbf{x}_i, \mathbf{w} \rangle - y_i)^2 + \lambda \|\mathbf{w}\|^2$. We present all our results with the ℓ_2 regularization parameter set at $\lambda = 10^{-4}$, though we found that our results were not sensitive to this choice of parameter.

4.6.1 Single worker results

We first test our algorithms in the sequential, non-distributed setting. It is well known that VR beats vanilla SGD by a wide margin in many applications. However, the different VR methods vary widely in their empirical behavior. We compare the single worker *CentralVR* algorithm to the two most popular VR methods, SVRG [JZ13] and

SAGA [DBLJ14].

We test the methods on two synthetic "toy" datasets, in addition to two real-world datasets. Synthetic classification data was generated by sampling two normal distributions with unit variance and means separated by one unit. For the least-squares prediction problem, we generate a random normal matrix $\mathbf{X}$ and random labels of the form $\mathbf{y} = \mathbf{A}\mathbf{w} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is standard Gaussian noise. For each case, we kept the size of the dataset at $n = 5000$ with $d = 20$ features. For the binary classification problem, we kept equal numbers of data samples for each class. We also tested performance of our algorithms on two standard real world datasets: IJCNN1 [Pro01] for binary classification and the MILLIONSONG [BMEWL11] dataset for least squares prediction. IJCNN1 contains 35,000 training data samples of 22 dimensions, while MILLIONSONG contains 463,715 training samples of 90 dimensions. For all our experiments, we maintain a constant learning rate, and choose the learning rate that yields fastest convergence.

Results appear in Figure 4.1. We compare convergence rates of the algorithms in terms of number of gradient computations for each method. This provides a level playing field since different VR methods require different numbers of gradient computations per iteration, and gradient computations dominate the computing time. The proposed *CentralVR* algorithm widely out-performs SAGA and SVRG in all cases, requiring less than one-third of the gradient computations of the other methods.

4.6.2 Distributed results

We now present results of our algorithms in highly distributed settings. We implement the algorithms using a Python binding to MPI, and all experiments were run on an Intel Xeon E5 cluster with 24 cores per node. All our asynchronous implementations are "locked", where at a given time only one local node can update the parameters on the central server. However, all proposed asynchronous algorithms can be easily implemented in a lock-free setting, leading to further speedups.

We compare the distributed versions of CentralVR, *CentralVR-Async* [CVR-Async in Figures 4.2 and 4.3] and *CentralVR-Sync* [CVR-Sync in Figures 4.2 and 4.3], proposed in Section 4.4 with the following algorithms:

1. Distributed SVRG (Section 4.5.1) [D-SVRG in Figures 4.2 and 4.3]. We set the communication period $\tau = 2n$ as recommended in [JZ13]. We found the performance of the algorithm to be very robust to τ.

2. Distributed SAGA (Section 4.5.2) [D-SAGA in Figures 4.2 and 4.3]. We vary the communication period $\tau = \{10, 100, 1000, 10000\}$ and present results for the τ yielding best results. The algorithm remains relatively stable for $\tau = \{10, 100, 1000\}$ but convergence speeds start slowing down significantly at $\tau = 10000$.

3. Elastic Averaging SGD (EASGD): This is a recently proposed asynchronous SGD method [ZCL15] that has been shown to efficiently accelerate training times of deep neural networks. As in [ZCL15], we tested the algorithm for communication periods $\tau = \{4, 16, 64\}$, and found results to be nearly insensitive to τ (τ updates occur

before communication). We also found the regular EASGD algorithm to outperform the momentum version (M-EASGD). We test performance both for a constant step size as well as a decaying step size (using a local clock on each machine) as given by $\alpha_0/(1+\gamma k)^{0.5}$ (as in [ZCL15]), where α_0 is the initial step size, k is the local iteration number, and γ is the decay parameter. EASGD has been shown to outperform the related popular asynchronous SGD method Downpour [DCM+12], on both convex and non-convex settings.

4. Asynchronous "Parameter Server" SVRG [PS-SVRG in Figures 4.2 and 4.3]: an asynchronous version of SVRG on a parameter server model of computation [RHS+15]. This method outperforms a popular asynchronous SGD method, Hogwild [RRWN11], which also uses a parameter server model. We set the epoch size to $2n$, as recommended in [RHS+15].

For the variance reduction methods, we performed experiments using a constant step size, as well as the simple learning rate decay rule $\alpha_l = \alpha_0\gamma^l$ (here, l is the number of *epochs*, instead of iterations). Decaying the step size does not yield consistent performance gains, and constant step sizes work very well in practice.

We compared the algorithms on a binary classification problem and a least-squares prediction problem using both toy datasets and real world datasets. The toy datasets were created on each local worker exactly the same way as for the sequential experiments. The toy datasets had $d = 1000$ features and $|\Omega_s| = 5000$ samples for *each* core s, i.e., the total size of the dataset was $p \times 5000$, where p denotes the number of local nodes. We also used the real world datasets MILLIONSONG [BMEWL11] (containing close to 500,000 data

samples) for ridge regression and SUSY [BSW14] (5,000,000 data samples) for logistic regression.

Figure 4.2 shows results of our distributed experiments on toy datasets. The left two plots compare the rates of convergence of our algorithms scaled over 192 cores for logistic regression and ridge regression. The x-axis displays wall clock time in seconds and the y-axis displays the relative norm of the gradient, i.e., the ratio between the current gradient norm and the initial gradient norm. In almost all cases the proposed algorithms, in particular CentralVR, have substantially superior rates of convergence over established schemes. The right two plots in Figure 4.2 demonstrate the scalability of our algorithms. On the y-axis, we plot the wall clock time (in seconds) required for convergence, and on the x-axis, we vary the number of nodes as 96, 192, 480 and 960. Each local worker has $|\Omega_s| = 5000$ data points in each case, i.e., the amount of data scales linearly with the number of nodes. Notice that *CentralVR-Sync* and *CentralVR-Async* exhibit nearly perfect linear scaling, even when the number of workers is almost 1000. The dataset size in this regime is close to 5 million data points, and the proposed *CentralVR* methods train both our logistic and ridge regression models to five digits of precision in less than 15 seconds.

Figure 4.3 shows results of our distributed experiments on the large datasets SUSY and MILLIONSONG. The left two plots show convergence results for our algorithms over 500 nodes for SUSY and 240 nodes for MILLIONSONG. In both cases, we see that our proposed algorithms outperform or remain competitive with previously proposed schemes. The right two plots show the scaling of our algorithms as we increase the number of local workers for training SUSY and MILLIONSONG. We see that for MILLION-

SONG, increasing the number of local workers initially decreases convergence time, but speed levels out for large numbers of workers, likely due to the smaller size of the local dataset fragments. On the larger SUSY problem, we find a consistent decrease in the convergence times as we increase the number of workers. We train on this 5,000,000 sample dataset in less than 5 seconds using 750 local workers.

4.7 Summary

This section introduced a new variance reduction scheme, CentralVR, that has lower communication requirements than conventional schemes, allowing it to perform better in highly parallel cloud or cluster computing platforms. In addition, distributed versions of well-known variance reduction stochastic gradient descent (SGD) methods are presented that also perform well in highly distributed settings. We show that by leveraging variance reduction, we can combat the diminishing returns that plague classical SGD methods when scaled across many workers, achieving linear performance scaling to over 1000 cores. This represents a significant increase in scalability over previous stochastic gradient methods.

Algorithm 5 CentralVR-Sync Algorithm

parameters learning rate α

initialize $\mathbf{w}$, $\{\nabla f_j(\tilde{\mathbf{w}}_j)\}_j$, $\bar{\mathbf{g}}$

while not converged **do**

  for each local node s **do**

    $\tilde{\mathbf{g}} \leftarrow 0$

    set π: random permutation of indices $1, 2, \cdots, |\Omega_s|$

    for k in $\{1, \ldots, |\Omega_s|\}$ **do**

      $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \alpha\big(\nabla f^s_{\pi_k}(\mathbf{w}_k) - \nabla f^s_{\pi_k}(\tilde{\mathbf{w}}_{\pi_k}) + \bar{\mathbf{g}}\big)$

      accumulate average: $\tilde{\mathbf{g}} \leftarrow \tilde{\mathbf{g}} + \nabla f^s_{\pi_k}(\mathbf{w}_k)/|\Omega_s|$

      store gradient: $\nabla f^s_{\pi_k}(\tilde{\mathbf{w}}_{\pi_k}) \leftarrow \nabla f^s_{\pi_k}(\mathbf{w}_k)$

    end for

    set average gradient to send to server: $\bar{\mathbf{g}} \leftarrow \tilde{\mathbf{g}}$

    send $\mathbf{w}$, $\bar{\mathbf{g}}$ to central node

    receive updated $\mathbf{w}$, $\bar{\mathbf{g}}$ from central node

  end for

  central node:

    average $\mathbf{w}$, $\bar{\mathbf{g}}$ received from workers

    broadcast averaged $\mathbf{w}$, $\bar{\mathbf{g}}$ to local workers

end while

Algorithm 6 CentralVR-Async Algorithm

parameters learning rate α

initialize $\mathbf{w}$, $\{\nabla f_j(\tilde{\mathbf{w}}_j)\}_j, \bar{\mathbf{g}}, \rho = 1/p, \mathbf{w}_{\text{old}} = \bar{\mathbf{g}}_{\text{old}} = 0$

while not converged **do**

  for each local node **do**

    $\tilde{\mathbf{g}} \leftarrow 0$

    set π: random permutation of indices $1, 2, \cdots, |\Omega_s|$

    for k in $\{1, \ldots, |\Omega_s|\}$ **do**

      $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \alpha\big(\nabla f^s_{\pi_k}(\mathbf{w}_k) - \nabla f^s_{\pi_k}(\tilde{\mathbf{w}}_{\pi_k}) + \bar{\mathbf{g}}\big)$

      accumulate average: $\tilde{\mathbf{g}} \leftarrow \tilde{\mathbf{g}} + \nabla f^s_{\pi_k}(\mathbf{w}_k)/|\Omega_s|$

      store gradient: $\nabla f^s_{\pi_k}(\tilde{\mathbf{w}}_{\pi_k}) \leftarrow \nabla f^s_{\pi_k}(\mathbf{w}_k)$

    end for

    set average gradient: $\bar{\mathbf{g}} \leftarrow \tilde{\mathbf{g}}$

    compute change: $\Delta\mathbf{w} \leftarrow \mathbf{w} - \mathbf{w}_{\text{old}}$, $\Delta\bar{\mathbf{g}} \leftarrow \bar{\mathbf{g}} - \bar{\mathbf{g}}_{\text{old}}$

    set: $\mathbf{w}_{\text{old}} \leftarrow \mathbf{w}$, $\bar{\mathbf{g}}_{\text{old}} \leftarrow \bar{\mathbf{g}}$

    send $\Delta\mathbf{w}$, $\Delta\bar{\mathbf{g}}$ to central node

    receive updated $\mathbf{w}$, $\bar{\mathbf{g}}$ from central node

  end for

  central node:

    receive $\Delta\mathbf{w}$, $\Delta\bar{\mathbf{g}}$ from a local worker

    update: $\mathbf{w} \leftarrow \mathbf{w} + \rho\Delta x$, $\bar{\mathbf{g}} \leftarrow \bar{\mathbf{g}} + \rho\Delta\bar{\mathbf{g}}$

    send new $\mathbf{w}$, $\bar{\mathbf{g}}$ back to local worker

end while

Algorithm 7 Synchronous SVRG

parameters step size α, communication period τ

initialize $\mathbf{w}$

while not converged **do**

  set: $\bar{\mathbf{w}} \leftarrow \mathbf{w}$

  set: $\bar{\mathbf{g}} \leftarrow \nabla f(\bar{\mathbf{w}})$ via synchronization step

  for each local node s **do**

    for k in $\{1, \ldots, \tau\}$ **do**

      sample $\tilde{k} \in \{1, \ldots, |\Omega_s|\}$ with replacement

      $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \alpha\big(\nabla f^s_{\tilde{k}}(\mathbf{w}_k) - \nabla f^s_{\tilde{k}}(\bar{\mathbf{w}}) + \bar{\mathbf{g}}\big)$

    end for

    send $\mathbf{w}$ to central node

    receive updated $\mathbf{w}$ from central node

  end for

  central node:

    average $\mathbf{w}$ received from workers

    broadcast averaged $\mathbf{w}$ to local workers

end while

Algorithm 8 Asynchronous SAGA

parameters step size α, communication period τ

initialize $\mathbf{w}$, $\{\nabla f_j(\tilde{\mathbf{w}}_j)\}_j, \rho = 1/p, \mathbf{w}_{\text{old}} = \bar{\mathbf{g}}_{\text{old}} = 0$

set average gradient: $\bar{\mathbf{g}} \leftarrow \frac{1}{n}\sum_j \nabla f_j(\tilde{\mathbf{w}}_j)$

while not converged **do**

  for each local node **do**

    for k in $\{1, \ldots, \tau\}$ **do**

      sample $\tilde{k} \in \{1, \ldots, n\}$ with replacement

      $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \alpha\big(\nabla f^s_{\tilde{k}}(\mathbf{w}_k) - \nabla f^s_{\tilde{k}}(\tilde{\mathbf{w}}_{\tilde{k}}) + \bar{\mathbf{g}}\big)$

      update: $\bar{\mathbf{g}} \leftarrow \bar{\mathbf{g}} + \frac{1}{n}\big(\nabla f^s_{\tilde{k}}(\mathbf{w}_k) - \nabla f^s_{\tilde{k}}(\tilde{\mathbf{w}}_{\tilde{k}})\big)$

      store gradient: $\nabla f^s_{\tilde{k}}(\tilde{\mathbf{w}}_{\tilde{k}}) \leftarrow \nabla f^s_{\tilde{k}}(\mathbf{w}_k)$

    end for

    compute change: $\Delta\mathbf{w} \leftarrow \mathbf{w} - \mathbf{w}_{\text{old}}$, $\Delta\bar{\mathbf{g}} \leftarrow \bar{\mathbf{g}} - \bar{\mathbf{g}}_{\text{old}}$

    set: $\mathbf{w}_{\text{old}} \leftarrow x$, $\bar{\mathbf{g}}_{\text{old}} \leftarrow \bar{\mathbf{g}}$

    send $\Delta\mathbf{w}$, $\Delta\bar{\mathbf{g}}$ to central node

    receive updated $\mathbf{w}$, $\bar{\mathbf{g}}$ from central node

  end for

  central node:

    receive $\Delta\mathbf{w}$, $\Delta\bar{\mathbf{g}}$ from a local worker

    update: $\mathbf{w} \leftarrow \mathbf{w} + \rho\Delta\mathbf{w}$, $\bar{\mathbf{g}} \leftarrow \bar{\mathbf{g}} + \rho\Delta\bar{\mathbf{g}}$

    send new $\mathbf{w}$, $\bar{\mathbf{g}}$ back to local worker

end while

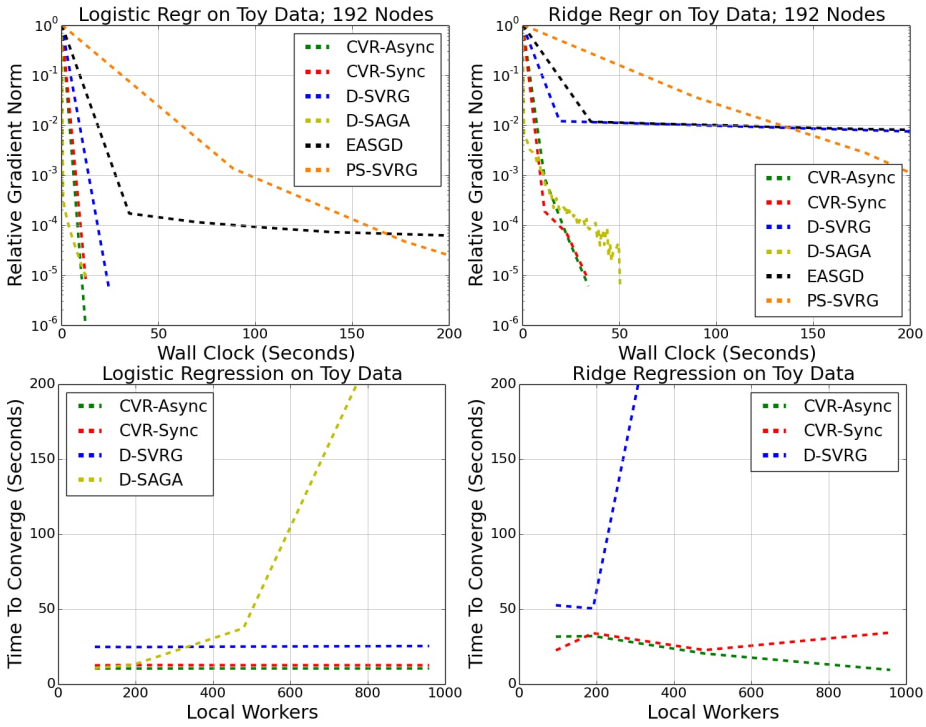


Figure 4.2: Distributed Results on toy datasets for *CentralVR-Sync* and *CentralVR-Async*, compared to Distributed SVRG (Section 4.5.1), Distributed SAGA (Section 4.5.2), Parameter Server SVRG and EASGD. Left two plots: Convergence curve for Logistic and ridge regression on synthetic data over 192 nodes. Right two plots: Time required for convergence as number of local workers is increased (data on each local worker is *constant* – i.e., total data scales *linearly* with the number of local workers) for logistic and ridge regression.

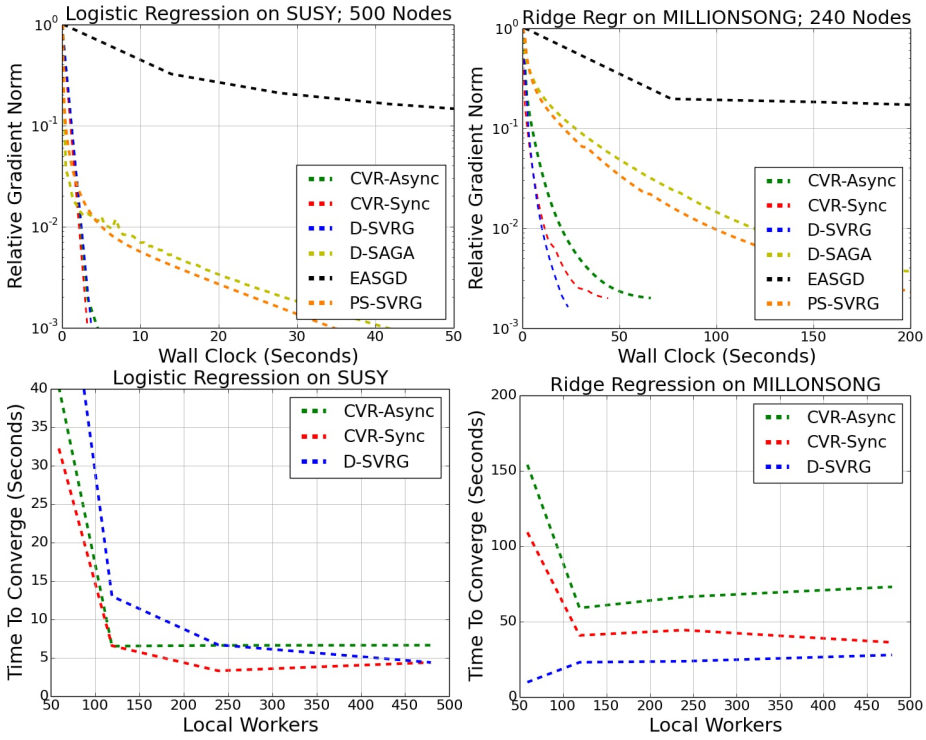


Figure 4.3: Distributed Results on SUSY and MILLIONSONG for *CentralVR-Sync* and *CentralVR-Async*, compared to Distributed SVRG (Section 4.5.1), Distributed SAGA (Section 4.5.2), Parameter Server SVRG (Param Server SVRG) and EASGD. (Left two plots) Convergence curve for Logistic regression and ridge regression on SUSY over 500 nodes and on MILLIONSONG over 240 nodes. (Right two plots) Time required for convergence as number of local workers is increased.

Chapter 5: **Investigating training methods for quantized neural nets**

5.1 Introduction

Deep neural networks are an integral part of state-of-the-art computer vision and natural language processing systems. Because of their high memory requirements and computational complexity, networks are usually trained using powerful hardware. There is an increasing interest in training and deploying neural networks directly on battery-powered devices, such as cell phones or other platforms. Such low-power embedded systems are memory and power limited, and in some cases lack basic support for floating-point arithmetic.

To make neural nets practical on embedded systems, many researchers have focused on training nets with *coarsely quantized* weights. For example, weights may be constrained to take on integer/binary values, or may be represented using low-precision (8 bits or less) fixed-point numbers. Quantized nets offer the potential of superior memory and computation efficiency, while achieving performance that is competitive with state-of-the-art high-precision nets. Quantized weights can dramatically reduce memory size and access bandwidth, increase power efficiency, exploit hardware-friendly bitwise operations, and accelerate inference throughput [CHS+16, MOPU93, RORF16].

Handling low-precision weights is difficult and motivates interest in new training

methods. When learning rates are small, stochastic gradient methods make small updates to weight parameters. Binarization/discretization of weights after each training iteration "rounds off" these small updates and causes training to stagnate [CHS+16]. Thus, the naïve approach of quantizing weights using a rounding procedure yields poor results when weights are represented using a small number of bits. Other approaches include classical stochastic rounding methods [GAGN15], as well as schemes that combine full-precision floating-point weights with discrete rounding procedures [CBD15]. While some of these schemes seem to work in practice, results in this area are largely experimental, and little work has been devoted to explaining the excellent performance of some methods, the poor performance of others, and the important differences in behavior between these methods.

Contributions

In this chapter, we study quantized training methods from a theoretical perspective, with the goal of understanding the differences in behavior, and reasons for success or failure, of various methods. In particular, we present a convergence analysis showing that classical stochastic rounding (SR) methods [GAGN15] as well as newer and more powerful methods like BinaryConnect (BC) [CBD15] are capable of solving convex discrete problems up to a level of accuracy that depends on the quantization level. We then address the issue of why algorithms that maintain floating-point representations, like BC, work so well, while fully quantized training methods like SR stall before training is complete. We show that the long-term behavior of BC has an important annealing property that is needed for non-convex optimization, while classical rounding methods lack this property.

5.2 Background and related work

The arithmetic operations of deep networks can be truncated down to 8-bit fixed-point without significant deterioration in inference performance [GAGN15,LTA16,HS14, LCMB16, LZL16]. The most extreme scenario of quantization is binarization, in which only 1-bit (two states) is used for weight representation [KS15,CBD15,CHS+16,RORF16, HCS+16, BIL+15].

Previous work on obtaining a quantized neural network can be divided into two categories: quantizing pre-trained models with or without retraining [HS14, AHS15, LTA16, ZHMD17, ZYG+17], and training a quantized model from scratch [GAGN15, CBD15, RORF16, CHS+16, ZWN+16]. We focus on approaches that belong to the second category, as they can be used for both training and inference under constrained resources.

For training quantized NNs from scratch, many authors suggest maintaining a high-precision floating point copy of the weights while feeding quantized weights into back-prop [CBD15, HCS+16, RORF16, ZWN+16], which results in good empirical performance. There are limitations in using such methods on low-power devices, however, where floating-point arithmetic is not always available or not desirable. Another widely used solution using only low-precision weights is *stochastic rounding* [HF92, GAGN15]. Experiments show that networks using 16-bit fixed-point representations with stochastic rounding can deliver results nearly identical to 32-bit floating-point computations [GAGN15], while lowering the precision down to 3-bit fixed-point often results in a significant performance degradation [MLM16]. Bayesian learning has also been applied to train binary networks [SHM14, CSML15]. A more comprehensive review can be found in [RORF16].

5.3 Algorithms for training quantized neural nets

Neural networks have objective functions of the same form as (2.1) where each f_i is a non-convex loss function. When floating-point representations are available, the standard method for training neural networks is SGD (2.2). In this chapter, we consider the problem of training convolutional neural networks (CNNs) with low precision weights. Convolutions are computationally expensive; low precision weights can be used to accelerate them by replacing expensive multiplications with efficient addition and subtraction operations [RORF16, LZL16] or bitwise operations [HCS+16, ZWN+16].

To train networks using a low-precision representation of the weights, a quantization function $Q(\cdot)$ is needed to convert a real-valued number w into a quantized/rounded version $\hat{w} = Q(w)$. We use the same notation for quantizing vectors, where we assume Q acts on each dimension of the vector. Different quantized optimization routines can be defined by selecting different quantizers, and also by selecting when quantization happens during optimization. The common options are:

Deterministic Rounding (R) A basic *uniform* or deterministic quantization function snaps a floating point value to the closest quantized value as:

$$Q_d(w) = \text{sign}(w) \cdot \Delta \cdot \left\lfloor \frac{|w|}{\Delta} + \frac{1}{2} \right\rfloor, \tag{5.1}$$

where Δ denotes the quantization step or resolution, i.e., the smallest positive number that is representable. One exception to this definition is when we consider binary weights, where all weights are constrained to have two values $w \in \{-1, 1\}$ and uniform rounding becomes $Q_d(w) = \text{sign}(w)$.

The deterministic rounding SGD maintains quantized weights with updates of the form:

$$\text{Deterministic Rounding:} \quad \mathbf{w}_{k+1}^b = Q_d\big(\mathbf{w}_k^b - \alpha_k \nabla \tilde{f}_k(\mathbf{w}_k^b)\big), \tag{5.2}$$

where $\mathbf{w}^b$ denotes the low-precision weights, which are quantized using Q_d immediately after applying the gradient descent update. If gradient updates are significantly smaller than the quantization step, this method loses gradient information and weights may never be modified from their starting values.

Stochastic Rounding (SR) The *stochastic rounding* quantization function is defined as:

$$Q_s(w) = \Delta \cdot \begin{cases} \lfloor \frac{w}{\Delta} \rfloor + 1 & \text{for } p \leq \frac{w}{\Delta} - \lfloor \frac{w}{\Delta} \rfloor, \\ \lfloor \frac{w}{\Delta} \rfloor & \text{otherwise,} \end{cases} \tag{5.3}$$

where $p \in [0, 1]$ is produced by a uniform random number generator. This operator is non-deterministic, and rounds its argument up with probability $w/\Delta - \lfloor w/\Delta \rfloor$, and down otherwise. This quantizer satisfies the important property $\mathbb{E}[Q_s(\mathbf{w})] = \mathbf{w}$. Similar to the deterministic rounding method, the SR optimization method also maintains quantized weights with updates of the form:

$$\text{Stochastic Rounding:} \quad \mathbf{w}_{k+1}^b = Q_s\big(\mathbf{w}_k^b - \alpha_k \nabla \tilde{f}_k(\mathbf{w}_k^b)\big). \tag{5.4}$$

BinaryConnect (BC) The BinaryConnect algorithm [CBD15] accumulates gradient updates using a full-precision buffer $\mathbf{w}^r$, and quantizes weights just before gradient computations as follows.

$$\text{BinaryConnect:} \quad \mathbf{w}_{k+1}^r = \mathbf{w}_k^r - \alpha_k \nabla \tilde{f}_k\big(Q(\mathbf{w}_k^r)\big). \tag{5.5}$$

Either stochastic rounding Q_s or deterministic rounding Q_d can be used for quantizing the weights $\mathbf{w}^r$, but in practice, Q_d is the common choice. The original BinaryConnect paper constrains the low-precision weights to be $\{-1, 1\}$, which can be generalized to $\{-\Delta, \Delta\}$. A more recent method, Binary-Weights-Net (BWN) [RORF16], allows different filters to have different scales for quantization, which often results in better performance on large datasets.

Notation For the rest of the chapter, we use Q to denote both Q_s and Q_d unless the situation requires this to be distinguished. We also drop the superscripts on $\mathbf{w}^r$ and $\mathbf{w}^b$, and simply write $\mathbf{w}$.

5.4 Convergence analysis

We now present convergence guarantees for the Stochastic Rounding (SR) and BinaryConnect (BC) algorithms, with updates of the form (5.4) and (5.5), respectively. For the purposes of deriving theoretical guarantees, we assume each f_i in (2.1) is differentiable and μ-strongly convex: $\langle \nabla f(\mathbf{w}'), \mathbf{w} - \mathbf{w}' \rangle \leq f(\mathbf{w}) - f(\mathbf{w}') - \frac{\mu}{2}\|\mathbf{w} - \mathbf{w}'\|^2$. We assume the (stochastic) gradients are bounded: $\mathbb{E}\|\nabla \tilde{f}_k(\mathbf{w}_k)\|^2 \leq G^2$. Some results below also assume the domain of the problem is finite. In this case, the rounding algorithm clips values that leave the domain. For example, in the binary case, rounding returns bounded values in $\{-1, 1\}$.

5.4.1 Convergence of Stochastic Rounding (SR)

We can rewrite the update rule (5.4) as:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla \tilde{f}_k(\mathbf{w}_k) + \mathbf{r}_k, \tag{5.6}$$

where $\mathbf{r}_k = Q_s(\mathbf{w}_k - \alpha_k \nabla \tilde{f}_k(\mathbf{w}_k)) - \mathbf{w}_k + \alpha_k \nabla \tilde{f}_k(\mathbf{w}_k)$ denotes the quantization error on the k-th iteration. We want to bound this error in expectation. To this end, we present the following lemma.

Lemma 5.4.1. *The stochastic rounding error $\mathbf{r}_k$ on each iteration can be bounded, in expectation, as:*

$$\mathbb{E}\left\|\mathbf{r}_k\right\|^2 \leq \sqrt{d}\Delta\alpha_k G,$$

where d denotes the dimension of $\mathbf{w}$.

Proof. We want to bound the quantization error $\mathbf{r}_k$. Consider the i-th entry in $\mathbf{r}_k$ denoted by $(\mathbf{r}_k)_i$. Similarly, we define $(\mathbf{w}_k)_i$ and $(\nabla \tilde{f}_k(\mathbf{w}_k))_i$. Choose some random number $p \in [0, 1]$. The stochastic rounding operation produces a value of $\mathbf{r}_k$ given by

$$\begin{aligned}(\mathbf{r}_k)_i &= Q_s((\mathbf{w}_k)_i - \alpha_k(\nabla \tilde{f}_k(\mathbf{w}_k))_i) - (\mathbf{w}_k)_i + \alpha_k(\nabla \tilde{f}_k(\mathbf{w}_k))_i \\ &= \Delta \cdot \begin{cases} -q+1, & \text{for } p \leq q, \\ -q, & \text{otherwise,} \end{cases}\end{aligned}$$

where $q = -\dfrac{\alpha_k(\nabla \tilde{f}_k(\mathbf{w}_k))_i}{\Delta} - \left\lfloor \dfrac{-\alpha_k(\nabla \tilde{f}_k(\mathbf{w}_k))_i}{\Delta} \right\rfloor$ and $q \in [0, 1]$. Now we have

$$\mathbb{E}_p\left[((\mathbf{r}_k)_i)^2\right] \leq \Delta^2((-q+1)^2 q + (-q)^2(1-q)) = \Delta^2 q(1-q) \leq \Delta^2 \min\{q, 1-q\}.$$

Because $\min\{q, 1-q\} \leq \left|\frac{\alpha_k(\nabla \tilde{f}_k(\mathbf{w}_k))_i}{\Delta}\right|$, it follows that:

$$\mathbb{E}_p\left[((\mathbf{r}_k)_i)^2\right] \leq \Delta^2 \left|\frac{\alpha_k(\nabla \tilde{f}_k(\mathbf{w}_k))_i}{\Delta}\right| \leq \Delta \left|\alpha_k(\nabla \tilde{f}_k(\mathbf{w}_k))_i\right|.$$

Summing over the index i yields:

$$\mathbb{E}_p\left\|\mathbf{r}_k\right\|_2^2 \leq \Delta\alpha_k\left\|\nabla \tilde{f}_k(\mathbf{w}_k)\right\|_1 \leq \sqrt{d}\alpha_k\Delta\left\|\nabla \tilde{f}_k(\mathbf{w}_k)\right\|_2. \tag{5.7}$$

The result follows from: $\left(\mathbb{E}\left\|\nabla \tilde{f}_k(\mathbf{w}_k)\right\|_2\right)^2 \leq \mathbb{E}\left\|\nabla \tilde{f}_k(\mathbf{w}_k)\right\|_2^2 \leq G^2$. ■

From Lemma 5.4.1, we see that the rounding error per step decreases as the learning rate α_k decreases. This is intuitive since the probability of an entry in $\mathbf{w}_{k+1}$ differing from $\mathbf{w}_k$ is small when the gradient update is small relative to Δ. Using the above lemma, we now present convergence rate results for Stochastic Rounding (SR) in the strongly-convex case. Our error estimates are ergodic, i.e., they are in terms of $\bar{\mathbf{w}}_k = \frac{1}{k}\sum_{t=1}^{k}\mathbf{w}_t$, the average of the iterates.

Theorem 5.4.1. *Assume that f is μ-strongly convex and the learning rates are given by $\alpha_k = \frac{1}{\mu(k+1)}$. Consider the SR algorithm with updates of the form* (5.4)*. Then, we have:*

$$\mathbb{E}[f(\bar{\mathbf{w}}_k) - f(\mathbf{w}^\star)] \leq \frac{(1+\log(k+1))G^2}{2\mu k} + \frac{\sqrt{d}\Delta G}{2},$$

where $\mathbf{w}^\star = \arg\min_w f(\mathbf{w})$.

Proof. Subtracting $\mathbf{w}^\star$ from (5.6), taking norm, and expectation conditioned on $\mathbf{w}_k$:

$$\begin{aligned}
\mathbb{E}\|\mathbf{w}_{k+1} - \mathbf{w}^\star\|^2 &= \|\mathbf{w}_k - \mathbf{w}^\star\|^2 - 2\mathbb{E}\langle \mathbf{w}_k - \mathbf{w}^\star, \alpha_k\nabla \tilde{f}_k(\mathbf{w}_k) - \mathbf{r}_k\rangle + \mathbb{E}\|\alpha_k\nabla \tilde{f}_k(\mathbf{w}_k) - \mathbf{r}_k\|^2 \\
&= \|\mathbf{w}_k - \mathbf{w}^\star\|^2 - 2\alpha_k\langle \mathbf{w}_k - \mathbf{w}^\star, \nabla f(\mathbf{w}_k)\rangle + \alpha_k^2\mathbb{E}\|\nabla \tilde{f}_k(\mathbf{w}_k)\|^2 + \mathbb{E}\|\mathbf{r}_k\|^2 \\
&\leq \|\mathbf{w}_k - \mathbf{w}^\star\|^2 - 2\alpha_k\langle \mathbf{w}_k - \mathbf{w}^\star, \nabla f(\mathbf{w}_k)\rangle + \alpha_k^2 G^2 + \sqrt{d}\Delta\alpha_k G,
\end{aligned}$$

where we use the bounded variance assumption, $\mathbb{E}[\mathbf{r}_k] = 0$, and Lemma 5.4.1. Using the assumption that f is μ-strongly convex, we can simplify this to:

$$\mathbb{E}\|\mathbf{w}_{k+1} - \mathbf{w}^\star\|^2 \leq (1 - \alpha_k\mu)\|\mathbf{w}_k - \mathbf{w}^\star\|^2 - 2\alpha_k(f(\mathbf{w}_k) - f(\mathbf{w}^\star)) + \alpha_k^2 G^2 + \sqrt{d}\Delta\alpha_k G.$$

Re-arranging the terms, and taking expectation we get:

$$2\alpha_k\mathbb{E}(f(\mathbf{w}_k) - f(\mathbf{w}^\star)) \leq (1 - \alpha_k\mu)\mathbb{E}\|\mathbf{w}_k - \mathbf{w}^\star\|^2 - \mathbb{E}\|\mathbf{w}_{k+1} - \mathbf{w}^\star\|^2 + \alpha_k^2 G^2 + \sqrt{d}\Delta\alpha_k G.$$

Assume that the step size decreases with the rate $\alpha_k = 1/\mu(k+1)$. Then we have:

$$\mathbb{E}(f(\mathbf{w}_k) - f(\mathbf{w}^\star)) \leq \frac{\mu k}{2}\mathbb{E}\|\mathbf{w}_k - \mathbf{w}^\star\|^2 - \frac{\mu(k+1)}{2}\mathbb{E}\|\mathbf{w}_{k+1} - \mathbf{w}^\star\|^2 + \frac{1}{2\mu(k+1)}G^2 + \frac{\sqrt{d}\Delta G}{2}.$$

Averaging over $k = 0$ to T, we get a telescoping sum on the right hand side, which yields:

$$\begin{aligned}\frac{1}{T}\sum_{k=0}^{T}\mathbb{E}(f(\mathbf{w}_k) - f(\mathbf{w}^\star)) &\leq \frac{G^2}{2\mu T}\sum_{k=0}^{T}\frac{1}{k+1} + \frac{\sqrt{d}\Delta G}{2} - \frac{\mu(T+1)}{2}\mathbb{E}\|\mathbf{w}_{k+1} - \mathbf{w}^\star\|^2 \\ &\leq \frac{(1 + \log(T+1))G^2}{2\mu T} + \frac{\sqrt{d}\Delta G}{2}.\end{aligned}$$

Using Jensen's inequality, we have: $\mathbb{E}(f(\bar{\mathbf{w}}_T) - f(\mathbf{w}^\star)) \leq \frac{1}{T}\sum_{k=0}^{T}\mathbb{E}(f(\mathbf{w}_k) - f(\mathbf{w}^\star))$, where $\bar{\mathbf{w}}_T = \frac{1}{T}\sum_{k=0}^{T}\mathbf{w}_k$, the average of the iterates. The desired bound follows. ■

We see that SR converges until it reaches an "accuracy floor." As the quantization becomes more fine grained, our theory predicts that the accuracy of SR approaches that of high-precision floating point at a rate linear in Δ. This extra term caused by the discretization is unavoidable since this method maintains quantized weights.

5.4.2 Convergence of Binary Connect (BC)

When analyzing the BC algorithm, we assume that the Hessian satisfies the Lipschitz bound: $\|\nabla^2 f_i(\mathbf{w}) - \nabla^2 f_i(\mathbf{w}')\| \leq L_2\|\mathbf{w} - \mathbf{w}'\|$ for some $L_2 \geq 0$. While this is a

slightly non-standard assumption, we will see that it enables us to gain better insights into the behavior of the algorithm.

We assume that the quantization function in BC uses stochastic rounding. In the case of BC, we see that the quantization error $\mathbf{r}$ does not approach 0 as in SR-SGD. Nonetheless, the effect of this rounding error diminishes with shrinking α_k because α_k multiplies the gradient update, and thus implicitly the rounding error as well.

Theorem 5.4.2. *Assume that f is μ-strongly convex, the domain has finite diameter D, and the learning rates are given by $\alpha_k = \frac{1}{\mu(k+1)}$. Consider the BC algorithm with updates of the form* (5.5)*. Then we have:*

$$\mathbb{E}[f(\bar{\mathbf{w}}_k) - f(\mathbf{w}^\star)] \leq \frac{(1+\log(k+1))G^2}{2\mu k} + \frac{DL_2\sqrt{d}\Delta}{2}.$$

Proof. We can rewrite the update rule (5.5), as

$$\begin{aligned}\mathbf{w}_{k+1} &= \mathbf{w}_k - \alpha_k \nabla \tilde{f}\big(\mathbf{w}_k + \mathbf{r}_k\big) \\ &= \mathbf{w}_k - \alpha_k[\nabla \tilde{f}\big(\mathbf{w}_k\big) + \nabla^2 \tilde{f}\big(\mathbf{w}_k\big)\mathbf{r}_k + \hat{\mathbf{r}}_k]\end{aligned}$$

where $\|\hat{\mathbf{r}}_k\| \leq \frac{L_2}{2}\|\mathbf{r}_k\|^2$ from our assumption on the Hessian. Note that in general $\mathbf{r}_k$ has mean zero while $\hat{\mathbf{r}}_k$ does not. Using the same steps as in the Theorem 5.4.1, we get

$$\begin{aligned}\mathbb{E}\|\mathbf{w}_{k+1} - \mathbf{w}^\star\|^2 &= \|\mathbf{w}_k - \mathbf{w}^\star\|^2 - 2\alpha_k \mathbb{E}\langle \mathbf{w}_k - \mathbf{w}^\star, \nabla \tilde{f}_k(\mathbf{w}_k + \mathbf{r}_k)\rangle + \alpha_k^2 \mathbb{E}\|\nabla \tilde{f}_k(\mathbf{w}_k + \mathbf{r}_k)\|^2. \\ &\leq \|\mathbf{w}_k - \mathbf{w}^\star\|^2 - 2\alpha_k \mathbb{E}\langle \mathbf{w}_k - \mathbf{w}^\star, \nabla f(\mathbf{w}_k) + \hat{\mathbf{r}}_k\rangle + \alpha_k^2 G^2 \\ &= \|\mathbf{w}_k - \mathbf{w}^\star\|^2 - 2\alpha_k \mathbb{E}\langle \mathbf{w}_k - \mathbf{w}^\star, \nabla f(\mathbf{w}_k)\rangle + \alpha_k^2 G^2 - 2\alpha_k \mathbb{E}\langle \mathbf{w}_k - \mathbf{w}^\star, \hat{\mathbf{r}}_k\rangle\end{aligned}$$

Assuming the domain has finite diameter D, and observing that the quantization error for BC-SGD can always be upper-bounded as $\|\mathbf{r}_k\| \leq \sqrt{d}\Delta$, we get:

$$-2\alpha_k \mathbb{E}\langle \mathbf{w}_k - \mathbf{w}^\star, \hat{\mathbf{r}}_k\rangle \leq 2\alpha_k D \mathbb{E}\|\hat{\mathbf{r}}_k\| \leq 2\alpha_k D \frac{L_2}{2}\|\mathbf{r}_k\| \leq \alpha_k D L_2 \sqrt{d}\Delta.$$

Following the same steps as in Theorem 5.4.1, the desired bound follows. ■

Now, the error floor is determined by both Δ and L_2. For a quadratic least-squares problem, the gradient of f is linear and the Hessian is constant. Thus, $L_2 = 0$ and we get the following corollary.

Corollary 5.4.1. *Assume that f is quadratic and the learning rates are given by $\alpha_k = \frac{1}{\mu(k+1)}$. The BC algorithm with updates of the form* (5.5) *yields*

$$\mathbb{E}[f(\bar{\mathbf{w}}_k) - f(\mathbf{w}^\star)] \leq \frac{(1 + \log(k+1))G^2}{2\mu k}.$$

We see that the real-valued weights accumulated in BC can converge to the *true minimizer* of quadratic losses. Furthermore, this suggests that, when the function behaves like a quadratic on the distance scale Δ, one would expect BC to perform fundamentally better than SR. While this may seem like a restrictive condition, there is evidence that even non-convex neural networks become well approximated as a quadratic in the later stages of optimization within a neighborhood of a local minimum [MG15].

Note, our convergence results on BC are for $\mathbf{w}^r$ instead of $\mathbf{w}^b$, and these measures of convergence are not directly comparable. It is not possible to bound $\mathbf{w}^b$ when BC is used, as the values of $\mathbf{w}^b$ may not converge in the usual sense (e.g., in the +/-1 binary case $\mathbf{w}^r$ might converge to 0, in which case arbitrarily small perturbations to $\mathbf{w}^r$ might send $\mathbf{w}^b$ to +1 or -1).

5.5 What about non-convex problems?

The global convergence results presented above for convex problems show that, in general, both the SR and BC algorithms converge to within $\mathcal{O}(\Delta)$ accuracy of the mini-

mizer (in expected value). While we observe some differences between the two methods when the iterates are close to a minimizer (where the objective function behaves like a quadratic), these results do not explain the large differences generally observed when applied to non-convex neural nets. We now study how the long-term behavior of SR differs from BC. Note that this section makes no convexity assumptions, and the proposed theoretical results are directly applicable to neural networks.

Typical (continuous-valued) SGD methods have an important exploration-exploitation tradeoff. When the learning rate is large, the algorithm explores by moving quickly between states. Exploitation happens when the learning rate is small. In this case, noise averaging causes the algorithm more greedily pursues local minimizers with lower loss values. Thus, the distribution of iterates produced by the algorithm becomes increasingly concentrated near minimizers as the learning rate vanishes (see, e.g., the large-deviation estimates in [LNS12]). BC maintains this property as well—indeed, we saw in Corollary 5.4.1 a class of problems for which the iterates concentrate on the minimizer for small α_k.

In this section, we show that the SR method lacks this important tradeoff: as the step size gets small and the algorithm slows down, the quality of the iterates produced by the algorithm does *not* improve, and the algorithm does *not* become progressively more likely to produce low-loss iterates. This behavior is illustrated in Figures 5.1 and 5.2.

To understand this problem conceptually, consider the simple case of a one-variable optimization problem starting at $w_0 = 0$ with $\Delta = 1$ (Figure 5.1). On each iteration, the algorithm computes a stochastic approximation $\nabla \tilde{f}$ of the gradient by sampling from a distribution, which we call p. This gradient is then multiplied by the step size to get $\alpha \nabla \tilde{f}$. The probability of moving to the right (or left) is then roughly proportional to the

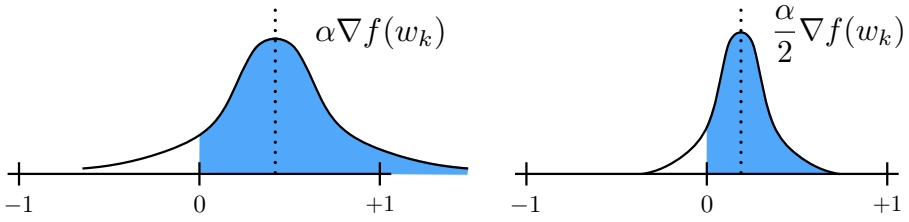


Figure 5.1: The SR method starts at some location w (in this case 0), adds a perturbation to w, and then rounds. As the learning rate α gets smaller, the distribution of the perturbation gets "squished" near the origin, making the algorithm less likely to move. The "squishing" effect is the same for the part of the distribution lying to the left and to the right of w, and so it does not effect the *relative* probability of moving left or right.

magnitude of $\alpha\nabla\tilde{f}$. Note the random variable $\alpha\nabla\tilde{f}$ has distribution $p_\alpha(z) = \alpha^{-1}p(z/\alpha)$.

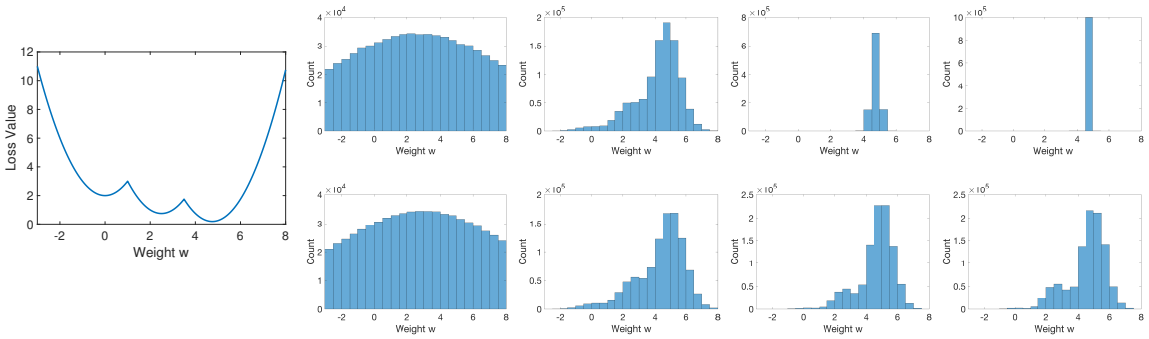
Now, suppose that α is small enough that we can neglect the tails of $p_\alpha(z)$ that lie outside the interval $[-1, 1]$. The probability of transitioning from $w_0 = 0$ to $w_1 = 1$ using stochastic rounding, denoted by $T_\alpha(0, 1)$, is then

$$T_\alpha(0,1) \approx \int_0^1 z p_\alpha(z) dz = \frac{1}{\alpha}\int_0^1 z p(z/\alpha)\, dz = \alpha \int_0^{1/\alpha} p(w) w\, dw \approx \alpha \int_0^\infty p(w) w\, dw,$$

where the first approximation is because we neglected the unlikely case that $\alpha\nabla\tilde{f} > 1$, and the second approximation appears because we added a small tail probability to the estimate. These approximations get more accurate for small α. We see that, assuming the tails of p are "light" enough, we have $T_\alpha(0,1) \sim \alpha\int_0^\infty p(w)w\,dw$ as $\alpha \to 0$. Similarly, $T_\alpha(0,-1) \sim \alpha\int_{-\infty}^0 p(w)w\,dw$ as $\alpha \to 0$.

What does this observation mean for the behavior of SR? First of all, the probability of leaving w_0 on an iteration is

$$T_\alpha(0,-1) + T_\alpha(0,1) \approx \alpha\left[\int_0^\infty p(w)w\,dw + \int_{-\infty}^0 p(w)w\,dw\right],$$



(a) $\alpha = 1.0$ (b) $\alpha = 0.1$ (c) $\alpha = 0.01$ (d) $\alpha = 0.001$

Figure 5.2: Effect of shrinking the learning rate in SR vs BC on a toy problem. The left figure plots the objective function (5.8). Histograms plot the distribution of the quantized weights over 10^6 iterations. The top row of plots correspond to BC, while the bottom row is SR, for different learning rates α. As the learning rate α shrinks, the BC distribution concentrates on a minimizer, while the SR distribution stagnates.

which vanishes for small α. This means the algorithm slows down as the learning rate drops off, which is not surprising. However, the *conditional* probability of ending up at $w_1 = 1$ given that the algorithm *did* leave w_0 is

$$T_\alpha(0, 1 | w_1 \neq w_0) \approx \frac{T_\alpha(0,1)}{T_\alpha(0,-1) + T_\alpha(0,1)} = \frac{\int_0^\infty p(w) w \, dw}{\int_{-\infty}^0 p(w) w \, dw + \int_0^\infty p(w) w \, dw},$$

which does not depend on α. In other words, provided α is small, SR, on average, makes the same decisions/transitions with learning rate α as it does with learning rate $\alpha/10$; it just takes 10 times longer to make those decisions when $\alpha/10$ is used. In this situation, there is no exploitation benefit in decreasing α.

5.5.1 Toy problem

To gain more intuition about the effect of shrinking the learning rate in SR vs BC, consider the following simple 1-dimensional non-convex problem:

$$\min_w f(w) := \begin{cases} w^2 + 2, & \text{if } w < 1, \\ (w - 2.5)^2 + 0.75, & \text{if } 1 \leq w < 3.5, \\ (w - 4.75)^2 + 0.19, & \text{if } w \geq 3.5. \end{cases} \tag{5.8}$$

Figure 5.2 shows a plot of this loss function. To visualize the distribution of iterates, we initialize at $w = 4.0$, and run SR and BC for 10^6 iterations using a quantization resolution of 0.5.

Figure 5.2 shows the distribution of the quantized weight parameters w over the iterations when optimized with SR and BC for different learning rates α. As we shift from $\alpha = 1$ to $\alpha = 0.001$, the distribution of BC iterates transitions from a wide/explorative distribution to a narrow distribution in which iterates aggressively concentrate on the minimizer. In contrast, the distribution produced by SR concentrates only slightly and then stagnates; the iterates are spread widely even when the learning rate is small.

5.5.2 Asymptotic analysis of Stochastic Rounding

The above argument is intuitive, but also informal. To make these statements rigorous, we interpret the SR method as a Markov chain. On each iteration, SR starts at some state (iterate) $\mathbf{w}$, and moves to a new state $\mathbf{w}'$ with some transition probability $T_\alpha(\mathbf{w}, \mathbf{w}')$ that depends only on $\mathbf{w}$ and the learning rate α. For fixed α, this is clearly a Markov

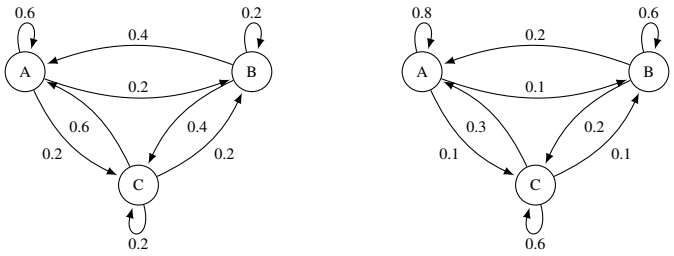


Figure 5.3: Markov chain example with 3 states. In the right figure, we halved each transition probability for moving between states, with the remaining probability put on the self-loop. Notice that halving all the transition probabilities would not change the equilibrium distribution, and instead would only increase the mixing time of the Markov chain.

process with transition matrix[1] $T_\alpha(\mathbf{w}, \mathbf{w}')$.

The long-term behavior of this Markov process is determined by the *stationary distribution* of $T_\alpha(\mathbf{w}, \mathbf{w}')$. We show below that for small α, the stationary distribution of $T_\alpha(\mathbf{w}, \mathbf{w}')$ is nearly invariant to α, and thus decreasing α below some threshold has virtually no effect on the long term behavior of the method. This happens because, as α shrinks, the relative transition probabilities remain the same (conditioned on the fact that the parameters change), even though the absolute probabilities decrease (see Figure 5.3). In this case, there is no exploitation benefit to decreasing α.

Theorem 5.5.1. *Let $p_{\mathbf{w},i}$ denote the probability distribution of the i-th entry in $\nabla \tilde{f}(\mathbf{w})$, the stochastic gradient estimate at $\mathbf{w}$. Assume there is a constant C_1 such that for all $\mathbf{w}$, i, and ν we have $\int_\nu^\infty p_{\mathbf{w},i}(z)\,dz \leq \frac{C_1}{\nu^2}$, and some C_2 such that both $\int_0^{C_2} p_{\mathbf{w},i}(z)\,dz > 0$*

[1] Our analysis below does not require the state space to be finite, so $T_\alpha(\mathbf{w}, \mathbf{w}')$ may be a linear operator rather than a matrix. Nonetheless, we use the term "matrix" as it is standard.

and $\int_{-C_2}^{0} p_{\mathbf{w},i}(z)\,dz > 0$. *Define the matrix*

$$\tilde{U}(\mathbf{w},\mathbf{w}') = \begin{cases} \int_0^\infty p_{\mathbf{w},i}(z)\frac{z}{\Delta}\,dz, \textit{ if } \mathbf{w} \textit{ and } \mathbf{w}' \textit{ differ only at coordinate } i, \textit{ and } (\mathbf{w}')_i = (\mathbf{w})_i + \Delta \\ \int_{-\infty}^0 p_{\mathbf{w},i}(z)\frac{z}{\Delta}\,dz, \textit{ if } \mathbf{w} \textit{ and } \mathbf{w}' \textit{ differ only at coordinate } i, \textit{ and } (\mathbf{w}')_i = (\mathbf{w})_i - \Delta \\ 0, \textit{otherwise,} \end{cases}$$

and the associated Markov chain transition matrix

$$\tilde{T}_{\alpha_0} = I - \alpha_0 \cdot \mathrm{diag}(\mathbf{1}^T\tilde{U}) + \alpha_0\tilde{U}, \tag{5.9}$$

where α_0 *is the largest constant that makes* $\tilde{T}_{\alpha_0}$ *non-negative. Suppose* $\tilde{T}_\alpha$ *has a stationary distribution, denoted* $\tilde{\pi}$. *Then, for sufficiently small* α, T_α *has a stationary distribution* π_α, *and* $\lim_{\alpha\to 0}\pi_\alpha = \tilde{\pi}$. *Furthermore, this limiting distribution satisfies* $\tilde{\pi}(\mathbf{w}) > 0$ *for any state* $\mathbf{w}$, *and is thus not concentrated on local minimizers of* f.

Proof. Let the matrix U_α be a partial transition matrix defined by $U_\alpha(\mathbf{w},\mathbf{w}) = 0$, and $U_\alpha(\mathbf{w},\mathbf{w}') = T_\alpha(\mathbf{w},\mathbf{w}')$ for $\mathbf{w} \neq \mathbf{w}'$. From U_α, we can get back the full transition matrix T_α using the formula

$$T_\alpha = I - \mathrm{diag}(\mathbf{1}^T U_\alpha) + U_\alpha.$$

Note that this formula is essentially "filling in" the diagonal entries of T_α so that every column sums to 1, thus making T_α a valid stochastic matrix.

Let's bound the entries in U_α. Suppose that we begin an iteration of the stochastic rounding algorithm at some point $\mathbf{w}$. Consider an adjacent point $\mathbf{w}'$ that differs from $\mathbf{w}$ at only 1 coordinate, i, with $(\mathbf{w}')_i = (\mathbf{w})_i + \Delta$. Then we have

$$U_\alpha(\mathbf{w},\mathbf{w}') = \frac{1}{\alpha}\int_0^\Delta p_{\mathbf{w},i}((\mathbf{w})_i/\alpha)\frac{(\mathbf{w})_i}{\Delta}\,d(\mathbf{w})_i + \frac{1}{\alpha}\int_\Delta^{2\Delta} p_{\mathbf{w},i}((\mathbf{w})_i/\alpha)\frac{2\Delta - (\mathbf{w})_i}{\Delta}\,d(\mathbf{w})_i$$

$$
\begin{aligned}
&= \frac{1}{\alpha}\int_0^{\Delta/\alpha} p_{\mathbf{w},i}(z)\frac{\alpha z}{\Delta}\alpha\, dz + \frac{1}{\alpha}\int_{\Delta/\alpha}^{2\Delta/\alpha} p_{\mathbf{w},i}(z)\frac{2\Delta-\alpha z}{\Delta}\alpha\, dz \\
&\leq \alpha\int_0^{\Delta/\alpha} p_{\mathbf{w},i}(z)\frac{z}{\Delta}\, dz + \int_{\Delta/\alpha}^{\infty} p_{\mathbf{w},i}(z)\, dz \\
&= \alpha\int_0^{\infty} p_{\mathbf{w},i}(z)\frac{z}{\Delta}\, dz + O(\alpha^2). \qquad (5.10)
\end{aligned}
$$

Note we have used the decay assumption: $\int_\nu^\infty p_{\mathbf{w},i}(z) \leq \frac{C}{\nu^2}$. If $(\mathbf{w}')_i = (\mathbf{w})_i - \Delta$, then similarly the transition probability is $U_\alpha(\mathbf{w},\mathbf{w}') = \alpha\int_{-\infty}^0 p_{\mathbf{w},i}(z)\frac{z}{\Delta}\, dz + O(\alpha^2)$, and if $(\mathbf{w}')_i = (\mathbf{w})_i \pm m\Delta$ for an integer $m > 1$, $U_\alpha(\mathbf{w},\mathbf{w}') = O(\alpha^2)$. We can approximate the behavior of U_α using the matrix

$$
\tilde{U}(\mathbf{w},\mathbf{w}') = \begin{cases} \int_0^\infty p_{\mathbf{w},i}(z)\frac{z}{\Delta}\, dz, \text{if } \mathbf{w} \text{ and } \mathbf{w}' \text{ differ only at coordinate } i, \text{ and } (\mathbf{w}')_i = (\mathbf{w})_i + \Delta \\ \int_{-\infty}^0 p_{\mathbf{w},i}(z)\frac{z}{\Delta}\, dz, \text{if } \mathbf{w} \text{ and } \mathbf{w}' \text{ differ only at coordinate } i, \text{ and } (\mathbf{w}')_i = (\mathbf{w})_i - \Delta \\ 0, \text{otherwise.} \end{cases}
$$

Define the associated Markov chain transition matrix

$$
\tilde{T}_{\alpha_0} = I - \alpha_0 \cdot \mathrm{diag}(\mathbf{1}^T\tilde{U}) + \alpha_0\tilde{U}, \qquad (5.11)
$$

where α_0 is the largest scalar such that the stochastic linear operator $\tilde{T}_{\alpha_0}$ has non-negative entries. For $\alpha < \alpha_0$, $\tilde{T}_\alpha$ has non-negative entries and column sums equal to 1; it thus defines the transition operator of a Markov chain. Let $\tilde{\pi}$ denote the stationary distribution of the Markov chain with transition matrix $\tilde{T}_{\alpha_0}$.

We now claim that $\tilde{\pi}$ is also the stationary distribution of $\tilde{T}_\alpha$ for all $\alpha < \alpha_0$. We verify this by noting that

$$
\begin{aligned}
\tilde{T}_\alpha &= (I - \alpha \cdot \mathrm{diag}(\mathbf{1}^T\tilde{U})) + \alpha\tilde{U} \\
&= (1 - \frac{\alpha}{\alpha_0})I + \frac{\alpha}{\alpha_0}[I - \alpha_0 \cdot \mathrm{diag}(\mathbf{1}^T\tilde{U}) + \alpha_0\tilde{U}]
\end{aligned}
$$

$$= (1 - \frac{\alpha}{\alpha_0})I + \frac{\alpha}{\alpha_0}\tilde{T}_{\alpha_0}, \tag{5.12}$$

and so $\tilde{T}_\alpha \tilde{\pi} = (1 - \frac{\alpha}{\alpha_0})\tilde{\pi} + \frac{\alpha}{\alpha_0}\tilde{\pi} = \tilde{\pi}$.

Recall that T_α is the transition matrix for the Markov chain generated by the stochastic rounding algorithm with learning rate α. We wish to show that this Markov chain is well approximated by $\tilde{T}_\alpha$. Note that

$$T_\alpha(\mathbf{w}, \mathbf{w}') = \prod_{i,(\mathbf{w})_i \neq (\mathbf{w}')_i} T_\alpha(\mathbf{w}, \mathbf{w} + ((\mathbf{w}')_i - (\mathbf{w})_i)\Delta \mathbf{e}_i) \leq O(\alpha^2)$$

when $\mathbf{w}, \mathbf{w}'$ differ at more than 1 coordinate, and $\mathbf{e}_i$ denotes a vector that is 1 in the i-th coordinate, and 0 everywhere else. In other words, transitions between multiple coordinates simultaneously become vanishingly unlikely for small α. When $\mathbf{w}$ and $\mathbf{w}'$ differ by exactly 1 coordinate, we know from (5.10): $T_\alpha(\mathbf{w}, \mathbf{w}') = \alpha \tilde{U}(\mathbf{w}, \mathbf{w}') + O(\alpha^2)$. These observations show that the off-diagonal elements of T_α are well approximated (up to uniform $O(\alpha^2)$ error) by the corresponding elements in $\alpha\tilde{U}$. Since the columns of T_α sum to one, the diagonal elements are well approximated as well, and we have

$$T_\alpha = (I - \alpha \cdot \mathrm{diag}(\mathbf{1}^T \tilde{U})) + \alpha\tilde{U} + O(\alpha^2) = \tilde{T}_\alpha + O(\alpha^2).$$

To be precise, the notation above means that

$$|T_\alpha(\mathbf{w}, \mathbf{w}') - \tilde{T}_\alpha(\mathbf{w}, \mathbf{w}')| < C\alpha^2, \tag{5.13}$$

for some C that is uniform over $(\mathbf{w}, \mathbf{w}')$.

We are now ready to show that the stationary distribution of T_α exists and approaches $\tilde{\pi}$. Re-arranging (5.12) gives us: $\alpha_0 \tilde{T}_\alpha + (\alpha - \alpha_0)I = \alpha \tilde{T}_{\alpha_0}$. Combining this with (5.13), we get

$$\left\| \alpha_0 T_\alpha + (\alpha - \alpha_0)I - \alpha \tilde{T}_{\alpha_0} \right\|_\infty < O(\alpha^2),$$

and so

$$\left\| \frac{\alpha_0}{\alpha} T_\alpha + (1 - \frac{\alpha_0}{\alpha}) I - \tilde{T}_{\alpha_0} \right\|_\infty < O(\alpha). \tag{5.14}$$

From (5.14), we see that the matrix $\frac{\alpha_0}{\alpha} T_\alpha + (1 - \frac{\alpha_0}{\alpha}) I$ approaches $\tilde{T}_{\alpha_0}$. Note that $\tilde{\pi}$ is the Perron-Frobenius eigenvalue of $\tilde{T}_{\alpha_0}$, and thus has multiplicity 1. Multiplicity 1 eigenvalues/vectors of a matrix vary continuously with small perturbations to that matrix (Theorem 8, p130 of [Lax07]). It follows that, for small α, $\frac{\alpha_0}{\alpha} T_\alpha + (1 - \frac{\alpha_0}{\alpha}) I$ has a stationary distribution, and this distribution approaches $\tilde{\pi}$. The leading eigenvector of $\frac{\alpha_0}{\alpha} T_\alpha + (1 - \frac{\alpha_0}{\alpha}) I$ is the same as the leading eigenvector of T_α, and it follows that T_α has a stationary distribution that approaches $\tilde{\pi}$.

Finally, note that we have assumed $\int_0^{C_2} p_{\mathbf{w},i}(z)\, dz > 0$ and $\int_{-C_2}^0 p_{\mathbf{w},i}(z)\, dz > 0$. Under this assumption, for $\alpha < \frac{1}{C_2}$, $\tilde{T}_{\alpha_0}(\mathbf{w}, \mathbf{w}') > 0$ whenever $\mathbf{w}, \mathbf{w}'$ are neighbors the differ at a single coordinate. It follows that every state in the Markov chain $\tilde{T}_{\alpha_0}$ is accessible from every other state by traversing a path of non-zero transition probabilities, and so $\tilde{\pi}(\mathbf{w}) > 0$ for every state $\mathbf{w}$. ■

While the long term stationary behavior of SR is relatively insensitive to α, the convergence speed of the algorithm is not. To measure this, we consider the *mixing time* of the Markov chain. Let π_α denote the stationary distribution of a Markov chain. We say that the ϵ-mixing time of the chain is M_ϵ if M_ϵ is the smallest integer such that [LPW09]

$$|\mathbb{P}(\mathbf{w}_{M_\epsilon} \in A | \mathbf{w}_0) - \pi(A)| \leq \epsilon, \quad \text{for all } \mathbf{w}_0 \text{ and all subsets of states } A \subseteq W. \tag{5.15}$$

We show below that the mixing time of the Markov chain gets large for small α, which means exploration slows down, even though no exploitation gain is being realized.

Theorem 5.5.2. *Let $p_{\mathbf{w},i}$ satisfy the assumptions of Theorem 5.5.1. Choose some ϵ sufficiently small that there exists a* proper *subset of states $A \subset W$ with stationary probability $\pi_\alpha(A)$ greater than ϵ. Let $M_\epsilon(\alpha)$ denote the ϵ-mixing time of the chain with learning rate α. Then,*

$$\lim_{\alpha \to 0} M_\epsilon(\alpha) = \infty.$$

Proof. Given some distribution π over the states of the Markov chain, and some set A of states, let $[\pi]_A = \sum_{a \in A} \pi(a)$ denote the measure of A with respect to π.

Suppose for contradiction that the mixing time of the chain remains bounded as α vanishes. Then we can find an integer M_ϵ that upper bounds the ϵ-mixing time for all α. By the assumption of the theorem, we can select some set of states A with $[\tilde{\pi}]_A > \epsilon$, and some starting state $a \notin A$. Let e be a distribution (a vector in the finite-state case) with $e_a = 1$, $e_b = 0$ for $b \neq a$. Note that $[e]_A = 0$ because $a \notin A$. Then

$$\left| [e]_A - [\tilde{\pi}]_A \right| > \epsilon.$$

Note that, as $\alpha \to 0$, we have $\|T_\alpha - \tilde{T}_\alpha\| \to 0$ and thus $\|T_\alpha^{M_\epsilon} - \tilde{T}_\alpha^{M_\epsilon}\| \to 0$. We also see from the definition of $\tilde{T}_\alpha$ in (5.11), $\lim_{\alpha \to 0} \tilde{T}_\alpha = I$. It follows that

$$\lim_{\alpha \to 0} \left| [T_\alpha^{M_\epsilon} e]_A - [\tilde{\pi}]_A \right| = \left| [e]_A - [\tilde{\pi}]_A \right| > \epsilon,$$

and so for some α the inequality (5.15) is violated. This is a contradiction because it was assumed M_ϵ is an upper bound on the mixing time. ■

5.6 Experiments

To explore the implications of the theory, we train both VGG-like networks [SZ15] and Residual networks [HZRS16b] with binarized weights on image classification problems. On CIFAR-10, we train ResNet-56, wide ResNet-56 (WRN-56-2, with two times more filters than ResNet-56), VGG-9, and the high capacity VGG-BC network used for the original BC model [CBD15]. We also train ResNet-56 on CIFAR-100, and ResNet-18 on ImageNet [RDS+15]. VGG-9 on CIFAR-10 consists of 7 convolutional layers and 2 fully connected layers. The convolutional layers contain 64, 64, 128, 128, 256, 256 and 256 of 3×3 filters respectively. There is a Batch Normalization and ReLU after each convolutional layer and the first fully connected layer. The details of the architecture are presented in Table 5.1. VGG-BC is a high-capacity network used for the original BC method [CBD15], which contains 6 convolutional layers and 3 linear layers. We use the same architecture as in [CBD15] except using softmax and cross-entropy loss instead of SVM and squared hinge loss, respectively. The details of the architecture are presented in Table 5.2. ResNets-56 has 55 convolutional layers and one linear layer, and contains three stages of residual blocks where each stage has the same number of residual blocks. WRN-56-2 doubles the number of filters in each residual block as in [ZK16]. ResNets-18 for ImageNet has the same description as in [HZRS16b].

We implement all models in Torch7 [CKF11] and train the quantized models with NVIDIA GPUs. The default minibatch size is 128. Following [CBD15], we do not use weight decay during training. We use Adam [KB15] as our baseline optimizer as we found it to frequently give better results than well-tuned SGD (an observation that

is consistent with previous papers on quantized models [CHS+16, MOPU93, RORF16, GAGN15, CBD15]), and we train with the three quantized algorithms mentioned in Section 5.3, i.e., R-ADAM, SR-ADAM and BC-ADAM. The image pre-processing and data augmentation procedures are the same as [HZRS16b]. Similar to [RORF16], we only quantize the weights in the convolutional layers, but not linear layers, during training. Binarizing linear layers causes some performance drop without much computational speedup. This is because fully connected layers have very little computation overhead compared to Conv layers. Also, for state-of-the-art CNNs, the number of FC parameters is quite small. The number of params of Conv/FC layers for CNNs in Table 1 are (in millions): VGG-9: 1.7/1.1, VGG-BC: 4.6/9.4, ResNet-56: 0.84/0.0006, WRN-56-2: 3.4/0.001, ResNet-18: 11.2/0.5. While the VGG-like nets have many FC parameters, the more efficient and higher performing ResNets are almost entirely convolutional.

The weights of convolutional layers are intitialized with random Rademacher (± 1) variables. We set the initial learning rate to 0.01 and decrease the learning rate by a factor of 10 at epochs 82 and 122 for CIFAR-10 and CIFAR-100 [HZRS16b]. For ImageNet experiments, we train the model for 90 epochs and decrease the learning rate at epochs 30 and 60. The authors of BC [CBD15] adopt a small initial learning rate (0.003) and it takes 500 epochs to converge. It is observed that large binary weights ($\Delta = 1$) will generate small gradients when batch normalization is used [IS15a], hence a large learning rate is necessary for faster convergence. We experiment with a larger learning rate (0.01) and find it converges to the same performance within 160 epochs, comparing with 500 epochs in the original paper [CBD15].

Table 5.1: VGG-9 on CIFAR-10.

layer type	kernel size	input size	output size
Conv_1	3×3	$3 \times 32 \times 32$	$64 \times 32 \times 32$
Conv_2	3×3	$64 \times 32 \times 32$	$64 \times 32 \times 32$
Max Pooling	2×2	$64 \times 32 \times 32$	$64 \times 16 \times 16$
Conv_3	3×3	$64 \times 16 \times 16$	$128 \times 16 \times 16$
Conv_4	3×3	$128 \times 16 \times 16$	$128 \times 16 \times 16$
Max Pooling	2×2	$128 \times 16 \times 16$	$128 \times 8 \times 8$
Conv_5	3×3	$128 \times 8 \times 8$	$256 \times 8 \times 8$
Conv_6	3×3	$256 \times 8 \times 8$	$256 \times 8 \times 8$
Conv_7	3×3	$256 \times 8 \times 8$	$256 \times 8 \times 8$
Max Pooling	2×2	$256 \times 8 \times 8$	$256 \times 4 \times 4$
Linear	1×1	1×4096	1×256
Linear	1×1	1×256	1×10

Results The overall results are summarized in Table 5.3. The binary model trained by BC-ADAM has comparable performance to the full-precision model trained by ADAM. SR-ADAM outperforms R-ADAM, which verifies the effectiveness of Stochastic Rounding. There is a performance gap between SR-ADAM and BC-ADAM across all models and datasets. This is consistent with our theoretical results in Sections 5.4 and 5.5, which predict that keeping track of the real-valued weights as in BC-ADAM should produce

Table 5.2: VGG-BC for CIFAR-10.

layer type	kernel size	input size	output size
Conv_1	3×3	$3 \times 32 \times 32$	$128 \times 32 \times 32$
Conv_2	3×3	$128 \times 32 \times 32$	$128 \times 32 \times 32$
Max Pooling	2×2	$128 \times 32 \times 32$	$128 \times 16 \times 16$
Conv_3	3×3	$128 \times 16 \times 16$	$256 \times 16 \times 16$
Conv_4	3×3	$256 \times 16 \times 16$	$256 \times 16 \times 16$
Max Pooling	2×2	$256 \times 16 \times 16$	$256 \times 8 \times 8$
Conv_5	3×3	$256 \times 8 \times 8$	$512 \times 8 \times 8$
Conv_6	3×3	$512 \times 8 \times 8$	$512 \times 8 \times 8$
Max Pooling	2×2	$512 \times 8 \times 8$	$512 \times 4 \times 4$
Linear	1×1	1×8192	1×1024
Linear	1×1	1×1024	1×1024
Linear	1×1	1×1024	1×10

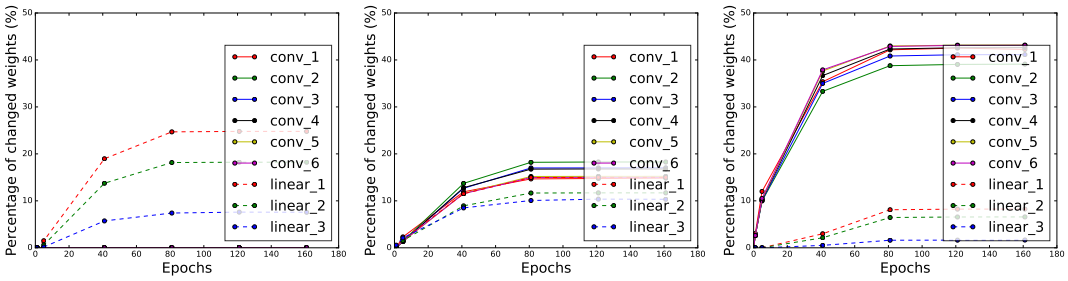
better minimizers.

Exploration vs exploitation tradeoffs Section 5.5 discusses the exploration/exploitation tradeoff of continuous-valued SGD methods and predicts that fully discrete methods like SR are unable to enter a greedy phase. To test this effect, we plot the percentage of changed weights (signs different from the initialization) as a function of the training epochs (Figures 5.4 and 5.5). SR-ADAM explores aggressively; it changes more weights

Table 5.3: Top-1 test error after training with full-precision (ADAM), binarized weights (R-ADAM, SR-ADAM, BC-ADAM), and binarized weights with big batch size (Big SR-ADAM).

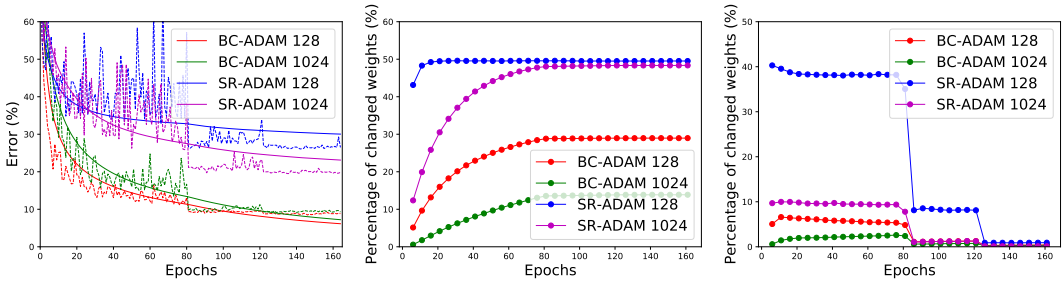
	CIFAR-10				CIFAR-100	ImageNet
	VGG-9	VGG-BC	ResNet-56	WRN-56-2	ResNet-56	ResNet-18
ADAM	7.97	7.12	8.10	6.62	33.98	36.04
BC-ADAM	10.36	8.21	8.83	7.17	35.34	52.11
Big SR-ADAM	16.95	16.77	19.84	16.04	50.79	77.68
SR-ADAM	23.33	20.56	26.49	21.58	58.06	88.86
R-ADAM	23.99	21.88	33.56	27.90	68.39	91.07

in the conv layers than both R-ADAM and BC-ADAM, and keeps changing weights until nearly 40% of the weights differ from their starting values (in a binary model, randomly re-assigning weights would result in 50% change). The BC method never changes more than 20% of the weights (Fig 5.4b), indicating that it stays near a local minimizer and explores less. Interestingly, we see that the weights of the conv layers were not changed at all by R-ADAM; when the tails of the stochastic gradient distribution are light, this method is ineffective.



(a) R-ADAM (b) BC-ADAM (c) SR-ADAM

Figure 5.4: Percentage of weight changes during training of VGG-BC on CIFAR-10.



(a) BC-ADAM vs SR-ADAM (b) Weight changes since beginning (c) Weight changes every 5 epochs

Figure 5.5: Effect of batch size on SR-ADAM when tested with ResNet-56 on CIFAR-10. (a) Test error vs epoch. Test error is reported with dashed lines, train error with solid lines. (b) Percentage of weight changes since initialization. (c) Percentage of weight changes per every 5 epochs.

5.6.1 A way forward: big batch training

We saw in Section 5.5 that SR is unable to exploit local minima because, for small learning rates, shrinking the learning rate does not produce additional bias towards moving downhill. This was illustrated in Figure 5.1. If this is truly the cause of the problem,

then our theory predicts that we can improve the performance of SR for low-precision training by increasing the batch size. This shrinks the variance of the gradient distribution in Figure 5.1 without changing the mean and concentrates more of the gradient distribution towards downhill directions, making the algorithm more greedy.

To verify this, we tried different batch sizes for SR including 128, 256, 512 and 1024, and found that the larger the batch size, the better the performance of SR. Figure 5.5a illustrates the effect of a batch size of 1024 for BC and SR methods. We find that the BC method, like classical SGD, performs best with a small batch size. However, a large batch size is essential for the SR method to perform well. Figure 5.5b shows the percentage of weights changed by SR and BC during training. We see that the large batch methods change the weights less aggressively than the small batch methods, indicating less exploration. Figure 5.5c shows the percentage of weights changed during each 5 epochs of training. It is clear that small-batch SR changes weights much more frequently than using a big batch. This property of big batch training clearly benefits SR; we see in Figure 5.5a and Table 5.3 that big batch training improved performance over SR-ADAM consistently.

In addition to providing a means of improving fixed-point training, this suggests that recently proposed methods using big batches [DYJG16, GDG+17] may be able to exploit lower levels of precision to further accelerate training.

5.7 Conclusion

The training of quantized neural networks is essential for deploying machine learning models on portable and ubiquitous devices. We provide a theoretical analysis to better understand the BinaryConnect (BC) and Stochastic Rounding (SR) methods for training quantized networks. We proved convergence results for BC and SR methods that predict an accuracy bound that depends on the coarseness of discretization. For general non-convex problems, we proved that SR differs from conventional stochastic methods in that it is unable to exploit greedy local search. Experiments confirm these findings, and show that the mathematical properties of SR are indeed observable in practice.

Chapter 6: **Why is SGD so fast for neural nets?**

6.1 Introduction

Stochastic gradient descent [RM51] (and its momentum variants [Nes83]) has become the standard optimization routine for deep learning due to its fast convergence and good generalization properties [WRS+17, KS17, SMDH13], but the performance of these methods defies explanation.

Classical convex optimization theory predicts that the learning rate of SGD needs to decrease over time for convergence to be guaranteed [SZ13, Ber11]. With constant learning rates, it has been shown that SGD converges fast to a neighborhood of the minimizer, but then reaches a noise floor that depends on the variance of the gradients at the minimizer [MB11, NWS14]. When models contain the same number of parameters as training data, it is possible for a model to over-fit the data while still being strongly convex. In this case, convergence without a noise floor is possible without decaying the learning rate [MB11, NWS14].

But the behavior of SGD on high-dimensional neural models still evades explanation. Neural networks operate in a regime where the number of parameters is much larger than the number of training data. In this regime, SGD seems to converge very quickly. So quickly, in fact, that practitioners often use exponentially decaying learning rate schedules

without seeing the method stall. Furthermore, network architecture seems to affect SGD a lot. It is common knowledge among practitioners that wider networks train faster [ZK16], and deeper networks train slower [BSF94, GB10].

The goal of this paper is to study why SGD is efficient for neural nets, and how neural net design affects SGD. In particular, we investigate how *over-parametrization* – an increase in the number of parameters beyond the number of training data – affects the dynamics of SGD.

To explain the fast convergence of SGD on over-parameterized problems, we introduce a simple concept called *gradient confusion*. When confusion is high, stochastic gradients produced by different data samples may be negatively correlated. When this happens, data samples contradict one another, causing slow convergence. When confusion is low, the gradients produced by different samples are similar, and we show via theoretical and empirical results that convergence is much faster than predicted by classical theory. For randomized training data, we show that the gradients of over-parameterized neural networks are likely to have low confusion. Finally, we present experimental results showing that low gradient confusion leads to efficient convergence of SGD.

Problem formulation & conceptual overview

Notation. In this chapter, we use upper-case bold fonts to represent matrices. We use $(\mathbf{W})_{i,j}$ to indicate the (i, j) cell in matrix $\mathbf{W}$ and $(\mathbf{W})_i$ for the i-th row of matrix $\mathbf{W}$.

SGD works by iteratively selecting a random function $\tilde{f}_k$, and modifying the parameters to decrease the value of the objective term $\tilde{f}_k$. It may happen that the selected

gradient $\nabla \tilde{f}_k$ is negatively correlated with the gradient of another term ∇f_j. In this case, the gains we make by decreasing $\tilde{f}_k$ are partially cancelled out by an increase in f_j, and convergence becomes slow. When the gradients of different mini-batches are negatively correlated, the objective terms disagree on which direction the parameters should move, and we say that there is *gradient confusion.*

Definition 6.1.1. *A set of objective functions $\{f_i\}$ has gradient confusion η if the pairwise inner products between gradients satisfy*

$$\langle \nabla f_i(\mathbf{w}), \nabla f_j(\mathbf{w}) \rangle \geq -\eta, \ \forall i, j. \tag{6.1}$$

SGD converges fast when gradient confusion is low. To see why, consider the case of training a logistic regression model on a dataset with orthogonal vectors. We have $f_i(\mathbf{w}) = \ell(y_i \cdot \langle \mathbf{x}_i, \mathbf{w} \rangle)$, where $\ell : \mathbb{R} \to \mathbb{R}$ is the logistic loss, $\{\mathbf{x}_i\}$ is a set of orthogonal training vectors, and $y_i \in \{-1, 1\}$ is a label. We then have $\nabla f_i(\mathbf{w}) = y_i \ell'(y_i \cdot \langle \mathbf{x}_i, \mathbf{w} \rangle)\mathbf{x}_i$ and $\langle \nabla f_i(\mathbf{w}), \nabla f_j(\mathbf{w}) \rangle = y_i y_j \ell'(\langle \mathbf{x}_i, \mathbf{w} \rangle)\ell'(\langle \mathbf{x}_j, \mathbf{w} \rangle)\langle \mathbf{x}_i, \mathbf{x}_j \rangle = 0$, and so there is no gradient confusion ($\eta = 0$). Because of gradient orthogonality, an update in the gradient direction f_i has *no* effect on the loss value of f_j for $i \neq j$. In this case, SGD decouples into a (deterministic) gradient decent on each objective term separately, and we can expect to see the fast rates of convergence attained by deterministic gradient descent, rather than the slow rates of SGD.

Can we expect a problem to have low gradient confusion in practice? It is known that randomly chosen vectors in high dimensions are nearly orthogonal with high probability [GS16]. For this reason, we would expect an average-case (i.e., random) problem to have nearly orthogonal gradients, provided that we don't train on too many training

vectors (in which case it becomes likely that we will see two training vectors with large negative correlation). In other words, we should expect a random optimization problem to have low gradient confusion when the number of parameters is "large" and the number of training data is "small" – i.e., when the model is over-parameterized.

The above argument is rather informal, and ignores issues like random sampling order and non-convexity. Furthermore, it is unclear whether we can expect low levels of gradient confusion in practice, and what effect non-zero confusion has on convergence rates. Below, we present a rigorous argument that low confusion levels accelerate SGD for both convex and non-convex problems. Then, we turn to the issue of over-parameterization, and show that gradient confusion is low for over-parameterized classifier problems with random data. Finally, we use computational experiments to show that gradient confusion is low for real-world neural nets, and that this explains the superior optimization performance of SGD.

Related work

The authors of [ACH18] study the behavior of SGD on over-parameterized problems, and show that SGD on over-parameterized linear neural nets is similar to applying a certain preconditioner while optimizing. Our work differs from [ACH18] in that it studies a completely different mechanism of acceleration, and that we establish a more direct relationship between width, depth, problem dimensionality, and the error floor of SGD convergence. The behavior of SGD on over-parameterized problems was also studied in [MBB17] with the purpose of exploring how SGD hyper-parameters (learning rates,

batch size, etc...) affect convergence. In contrast, this study focuses on how and why network architecture choices affect convergence.

Several other authors have studied the impact of structured gradients on SGD. [BFL+17] study the effects of "shattered gradients," which is when (non-stochastic) gradients at different (but close) locations in parameter space become negatively correlated. This is different from gradient confusion, which refers to negative correlations between stochastic mini-batch gradients at the same location in parameter space. Another related issue is that stochastic noise during training leads to improved generalization performance because of implicit regularization. This is not our main focus – we addresses the question of why SGD is good for optimization, rather than why it's good for generalization.

6.2 SGD is fast when gradient confusion is low

We now present a rigorous analysis of gradient confusion and its effect on SGD. We begin by looking at the case where the objective satisfies the PL inequality (a condition related to, but weaker than, strong convexity), where we can provide tight bounds on the rate of convergence in terms of the optimality gap. Then we look at a broader class of non-convex functions, and prove fast convergence to a stationary point.

We begin by making two standard assumptions about the objective function.

Assumption 6.2.1. *The individual gradients ∇f_i are L-Lipschitz continuous:*

$$f_i(\mathbf{w}') \leq f_i(\mathbf{w}) + \langle \nabla f_i(\mathbf{w}),\, \mathbf{w}' - \mathbf{w} \rangle + \tfrac{L}{2}\|\mathbf{w}' - \mathbf{w}\|^2,\ \forall i.$$

Assumption 6.2.2. *The individual functions f_i satisfy the PL inequality:*

$$\tfrac{1}{2}\|\nabla f_i(\mathbf{w})\|^2 \geq \mu(f_i(\mathbf{w}) - f_i^\star),\ \forall i,$$

where $f_i^\star = \min_{\mathbf{w}} f_i(\mathbf{w})$.

Using these assumptions, we now state the following convergence result.

Theorem 6.2.1 (Linear convergence under bounded gradient confusion)**.** *If the objective function satisfies Assumptions 6.2.1 and 6.2.2, and has gradient confusion bounded by* η*, SGD with updates of the form* (2.2) *converges linearly to a neighborhood of the minima on* (2.1) *as*

$$f(\mathbf{w}_k) - f^\star \leq \rho^k (f(\mathbf{w}_0) - f^\star) + \tfrac{\alpha\hat{\eta}}{1-\rho},$$

where $\hat{\eta} = \max\{\eta, 0\}$*, the learning rate* $\alpha \leq 2/nL$ *and* $\rho = 1 - \frac{2\mu}{n}\big(\alpha - \frac{nL\alpha^2}{2}\big)$.

Proof. From Assumption 6.2.1, we have

$$\begin{aligned}
f(\mathbf{w}_{k+1}) &\leq f(\mathbf{w}_k) + \langle \nabla f(\mathbf{w}_k),\, \mathbf{w}_{k+1} - \mathbf{w}_k \rangle + \tfrac{L}{2}\|\mathbf{w}_{k+1} - \mathbf{w}_k\|^2 \\
&= f(\mathbf{w}_k) - \alpha \langle \nabla f(\mathbf{w}_k),\, \nabla \tilde{f}_k(\mathbf{w}_k) \rangle + \tfrac{L\alpha^2}{2}\|\nabla \tilde{f}_k(\mathbf{w}_k)\|^2 \\
&= f(\mathbf{w}_k) - \big(\tfrac{\alpha}{n} - \tfrac{L\alpha^2}{2}\big)\|\nabla \tilde{f}_k(\mathbf{w}_k)\|^2 - \tfrac{\alpha}{n}\textstyle\sum_{\forall i: f_i \neq \tilde{f}_k} \langle \nabla f_i(\mathbf{w}_k),\, \nabla \tilde{f}_k(\mathbf{w}_k) \rangle \\
&\leq f(\mathbf{w}_k) - \big(\tfrac{\alpha}{n} - \tfrac{L\alpha^2}{2}\big)\|\nabla \tilde{f}_k(\mathbf{w}_k)\|^2 + \tfrac{\alpha(n-1)\hat{\eta}}{n}, \\
&\leq f(\mathbf{w}_k) - \big(\tfrac{\alpha}{n} - \tfrac{L\alpha^2}{2}\big)\|\nabla \tilde{f}_k(\mathbf{w}_k)\|^2 + \alpha\hat{\eta},
\end{aligned}$$

where the second-last inequality follows from Definition 6.1.1. Let $\alpha < 2/nL$. Then, using Assumption 6.2.2 and subtracting by $f^\star = \min_{\mathbf{w}} f(\mathbf{w})$ on both sides, we get

$$f(\mathbf{w}_{k+1}) - f^\star \leq f(\mathbf{w}_k) - f^\star - 2\mu\Big(\tfrac{\alpha}{n} - \tfrac{L\alpha^2}{2}\Big)(\tilde{f}_k(\mathbf{w}_k) - \tilde{f}_k^\star) + \alpha\hat{\eta},$$

where $\tilde{f}_k^\star = \min_{\mathbf{w}} \tilde{f}_k(\mathbf{w})$. Taking expectation and using the fact that $\mathbb{E}_i[f_i^\star] \leq f^\star$, we get:

$$f(\mathbf{w}_{k+1}) - f^\star \leq \Big(1 - \tfrac{2\mu\alpha}{n} + \mu L\alpha^2\Big)(f(\mathbf{w}_k) - f^\star) + \alpha\hat{\eta}.$$

Writing $\rho = 1 - \frac{2\mu\alpha}{n} + \mu L\alpha^2$, and unrolling the iterations, we get

$$\begin{aligned} f(\mathbf{w}_{k+1}) - f^\star &\leq \rho^{k+1}(f(\mathbf{w}_0) - f^\star) + \textstyle\sum_{i=0}^{k} \rho^i \alpha\hat{\eta} \\ &\leq \rho^{k+1}(f(\mathbf{w}_0) - f^\star) + \textstyle\sum_{i=0}^{\infty} \rho^i \alpha\hat{\eta} \\ &= \rho^{k+1}(f(\mathbf{w}_0) - f^\star) + \tfrac{\alpha\hat{\eta}}{1-\rho}, \end{aligned}$$

which completes the proof. ■

This result shows that SGD converges *linearly* to a neighborhood of a minimizer, and the size of this neighborhood depends on the level of gradient confusion. When $\eta \leq 0$, there is no confusion, and SGD converges directly to a minimizer without using a vanishing learning rate schedule.

In the case of non-convex functions, we can still prove convergence to a neighborhood of a stationary point under the following standard assumption.

Assumption 6.2.3. *Assume that the variance of the gradients is bounded as:*

$$\mathbb{E}\Big[\|\nabla\tilde{f}(\mathbf{w}) - \nabla f(\mathbf{w})\|^2\Big] \leq \sigma^2.$$

The following theorem shows fast convergence in the case of a smooth non-convex function when gradient confusion is low.

Theorem 6.2.2. *If the objective function satisfies Assumptions 6.2.1, and 6.2.3, and the confusion bound* (6.1)*, then SGD converges to a stationary point with*

$$\min_{k=1,\dots,T} \mathbb{E}\|\nabla f(\mathbf{w}_k)\|^2 \leq \frac{2n}{2\alpha - nL\alpha^2}\frac{f(\mathbf{w}_1) - f^\star}{T} + \frac{2n\eta}{2 - nL\alpha}$$

for learning rate $\alpha < 2/(nL)$.

Proof. From Theorem 6.2.1, we have:

$$f(\mathbf{w}_{k+1}) \leq f(\mathbf{w}_k) - \Big(\frac{\alpha}{n} - \frac{L\alpha^2}{2}\Big)\|\nabla \tilde{f}_k(\mathbf{w}_k)\|^2 + \alpha\eta. \tag{6.2}$$

Using Assumption 6.2.3, we can write:

$$\mathbb{E}\|\nabla \tilde{f}_k(\mathbf{w}_k)\|^2 = \mathbb{E}\|\nabla \tilde{f}_k(\mathbf{w}_k) - \nabla f(\mathbf{w}_k)\|^2 + \mathbb{E}\|\nabla f(\mathbf{w}_k)\|^2 = \sigma^2 + \mathbb{E}\|\nabla f(\mathbf{w}_k)\|^2.$$

Thus, taking expectation and assuming the step size $\alpha < 2/L$, we can rewrite equation (6.2) as:

$$\begin{aligned}\mathbb{E}\|\nabla f(\mathbf{w}_k)\|^2 &\leq \frac{2n}{2\alpha - nL\alpha^2}\mathbb{E}\big(f(\mathbf{w}_k) - f(\mathbf{w}_{k+1})\big) - \sigma^2 + \frac{2n\eta}{2 - nL\alpha} \\ &\leq \frac{2n}{2\alpha - nL\alpha^2}\mathbb{E}\big(f(\mathbf{w}_k) - f(\mathbf{w}_{k+1})\big) + \frac{2n\eta}{2 - nL\alpha}.\end{aligned}$$

Taking an average over T iterations, and using $f^\star = \min_{\mathbf{w}} f(\mathbf{w})$, we get:

$$\min_{k=1,\ldots,T} \mathbb{E}\|\nabla f(\mathbf{w}_k)\|^2 \leq \frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\|\nabla f(\mathbf{w}_k)\|^2 \leq \frac{2n}{2\alpha - nL\alpha^2}\frac{f(\mathbf{w}_1) - f^\star}{T} + \frac{2n\eta}{2 - nL\alpha}.$$

■

The presence of a noise floor is not always observed for over-parameterized problems, and the assumption that $\eta \leq 0$ is unrealistically strong to guarantee such convergence. In the next section, we present a few additional assumptions under which faster convergence can be observed.

6.2.1 Conditions for even faster convergence

Faster convergence can be guaranteed if we re-define gradient confusion using the correlation between gradients (rather than the dot product). If these correlations are bounded below, then linear convergence occurs with no noise floor. The following results

prove this convergence in the case of *over-fitting*. Over-fitting occurs when the minimal objective value of the composite objective F in (2.1) is the average of all the minimal values the terms $\{f_i\}$. In other words, parameter $\mathbf{w}^\star$ that minimizes F also simultaneously minimizes every objective term f_i.

Theorem 6.2.3. *Suppose F satisfies Assumptions 6.2.1, 6.2.2, and the correlation-based confusion condition*

$$\frac{\langle \nabla f_i(\mathbf{w}), \nabla f_j(\mathbf{w}) \rangle}{\|\nabla f_i(\mathbf{w})\| \|\nabla f_j(\mathbf{w})\|} \geq -\nu, \quad \forall i, j. \tag{6.3}$$

Suppose further the loss satisfies the over-fitting condition $\min_{\mathbf{w}} \frac{1}{n} \sum_i f_i(\mathbf{w}) = \frac{1}{n} \sum_i \min_{\mathbf{w}} f_i(\mathbf{w})$.

If the objective has confusion $\nu < \frac{\mu}{nL}$ and learning rate $\alpha < \frac{2}{nL} - \frac{2\nu}{\mu}$, then SGD converges with

$$f(\mathbf{w}_k) - f^\star \leq \rho^k (f(\mathbf{w}_0) - f^\star),$$

where $\rho = 1 - 2\mu\alpha/n + \mu L\alpha^2 + 2\alpha\nu L$.

Proof. From (6.3) and the identity $2ab \leq a^2 + b^2$ we get

$$\begin{aligned}
\langle \nabla f_i(\mathbf{w}), \nabla f_j(\mathbf{w}) \rangle &\geq -\nu \|\nabla f_i(\mathbf{w})\| \|\nabla f_j(\mathbf{w})\| \geq \frac{-\nu}{2} (\|\nabla f_i(\mathbf{w})\|^2 + \|\nabla f_j(\mathbf{w})\|^2) \\
&\geq -\nu L (f_i(\mathbf{w}) - f_i^\star + f_j(\mathbf{w}) - f_j^\star).
\end{aligned}$$

Following the proof of Theorem 6.2.1, we have

$$\begin{aligned}
f(\mathbf{w}_{k+1}) &\leq f(\mathbf{w}_k) + \langle \nabla f(\mathbf{w}_k),\, \mathbf{w}_{k+1} - \mathbf{w}_k \rangle + \frac{L}{2} \|\mathbf{w}_{k+1} - \mathbf{w}_k\|^2 \\
&= f(\mathbf{w}_k) - \alpha \langle \nabla f(\mathbf{w}_k),\, \nabla \tilde{f}_k(\mathbf{w}_k) \rangle + \frac{L\alpha^2}{2} \|\nabla \tilde{f}_k(\mathbf{w}_k)\|^2 \\
&= f(\mathbf{w}_k) - \Big(\frac{\alpha}{n} - \frac{L\alpha^2}{2} \Big) \|\nabla \tilde{f}_k(\mathbf{w}_k)\|^2 - \frac{\alpha}{n} \sum_{\forall i : f_i \neq \tilde{f}_k} \langle \nabla f_i(\mathbf{w}_k),\, \nabla \tilde{f}_k(\mathbf{w}_k) \rangle
\end{aligned}$$

$$\leq f(\mathbf{w}_k) - 2\mu\Big(\frac{\alpha}{n} - \frac{L\alpha^2}{2}\Big)(\tilde{f}_k(\mathbf{w}_k) - \tilde{f}_k^\star) + \frac{\alpha\nu L}{n}\sum_{\forall i: f_i \neq \tilde{f}_k}(f_i(\mathbf{w}_k) - f_i^\star + \tilde{f}_k(\mathbf{w}_k) - \tilde{f}_k^\star)$$

where the second-last inequality follows from Definition 6.1.1. Let the learning rate $\alpha < 2/nL$. Then, using Assumption 6.2.2 and subtracting by $f^\star = \min_{\mathbf{w}} f(\mathbf{w})$ on both sides

$$f(\mathbf{w}_{k+1}) - f^\star \leq f(\mathbf{w}_k) - f^\star - 2\mu\Big(\frac{\alpha}{n} - \frac{L\alpha^2}{2}\Big)(\tilde{f}_k(\mathbf{w}_k) - \tilde{f}_k^\star) + \frac{\alpha\nu L}{n}\sum_{\forall i: f_i \neq \tilde{f}_k}(f_i(\mathbf{w}_k) - f_i^\star + \tilde{f}_k(\mathbf{w}_k) - \tilde{f}_k^\star),$$

where $\tilde{f}_k^\star = \min_{\mathbf{w}} \tilde{f}_k(\mathbf{w})$. Taking expectation and using the fact that $\mathbb{E}_i[f_i^\star] = f^\star$, we get

$$f(\mathbf{w}_{k+1}) - f^\star \leq \Big(1 - \frac{2\mu\alpha}{n} + \mu L\alpha^2 + 2\alpha\nu L\Big)(f(\mathbf{w}_k) - f^\star).$$

■

Finally, we can strengthen the definition of confusion by examining the correlation between $\nabla f_i(\mathbf{w})$ and $\nabla f_j(\mathbf{w}')$ for all $\mathbf{w}$ and $\mathbf{w}'$. Compared to Theorem 6.2.1, convergence is guaranteed with a larger learning rate that is independent of the training set size n, and faster geometric decay.

Theorem 6.2.4. *If the objective function satisfies Assumptions 6.2.1 and 6.2.2, and satisfies the strengthened gradient confusion bound*

$$\langle \nabla f_i(\mathbf{w}), \nabla f_j(\mathbf{w}') \rangle \geq -\eta, \ \forall i, j, \mathbf{w}, \mathbf{w}',$$

then SGD converges with

$$f(\mathbf{w}_k) - f^\star \leq \rho^k(f(\mathbf{w}_0) - f^\star) + \frac{\alpha\hat{\eta}}{1-\rho},$$

where $\hat{\eta} = \max\{\eta, 0\}$, *the learning rate* $\alpha \leq 2/L$ *and* $\rho = 1 - 2\mu\alpha/n + \mu L\alpha^2/n$.

Proof. We start by noting that, for $i \neq j$

$$f_i(\mathbf{w} - \alpha \nabla f_j(\mathbf{w})) = f_i(\mathbf{w}) + \int_{t=0}^{\alpha} \frac{\partial}{\partial t} f_i(\mathbf{w} - t\nabla f_j(\mathbf{w}))\, dt \tag{6.4}$$

$$= f_i(\mathbf{w}) - \int_{t=0}^{\alpha} \nabla f_j(\mathbf{w})^T f_i(\mathbf{w} - t\nabla f_j(\mathbf{w}))\, dt \tag{6.5}$$

$$\leq f_i(\mathbf{w}) + \int_{t=0}^{\alpha} \hat{\eta}\, dt \leq f_i(\mathbf{w}) + \alpha\hat{\eta}. \tag{6.6}$$

We then have

$$\begin{aligned} nf(\mathbf{w}_{k+1}) &= \tilde{f}_k(\mathbf{w}^k - \alpha\nabla\tilde{f}_k(\mathbf{w}^k)) + \sum_{f_i \neq \tilde{f}_k} f_i(\mathbf{w}^k - \alpha\nabla\tilde{f}_k(\mathbf{w}^k)) \\ &\leq \tilde{f}_k(\mathbf{w}_k) - \alpha\langle\nabla\tilde{f}_k(\mathbf{w}_k),\, \nabla\tilde{f}_k(\mathbf{w}_k)\rangle + \frac{L\alpha^2}{2}\|\nabla\tilde{f}_k(\mathbf{w}_k)\|^2 + \sum_{f_i \neq \tilde{f}_k} f_i(\mathbf{w}) + \alpha\hat{\eta} \\ &\leq nf(\mathbf{w}^k) - (\alpha - \frac{L\alpha^2}{2})\|\nabla\tilde{f}_k(\mathbf{w}_k)\|^2 + n\alpha\hat{\eta}. \end{aligned}$$

Re-arranging and applying Assumption 6.2.2 we get

$$f(\mathbf{w}_{k+1}) \leq f(\mathbf{w}^k) - \frac{2\mu}{n}(\alpha - \frac{L\alpha^2}{2})(\tilde{f}_k(\mathbf{w}_k) - \tilde{f}_k^\star) + \alpha\hat{\eta}.$$

Taking expectations and subtracting $f^\star$ from both sides we get

$$f(\mathbf{w}_{k+1}) - f^\star \leq \Big(1 - \frac{2\mu\alpha}{n} + \frac{\mu L\alpha^2}{n})\Big)(\tilde{f}_k(\mathbf{w}_k) - \tilde{f}_k^\star) + \alpha\hat{\eta}.$$

Unrolling this expression gives us

$$f(\mathbf{w}_{k+1}) - f^\star \leq \rho^{k+1}(f(\mathbf{w}_0) - f^\star) + \frac{\alpha\hat{\eta}}{1-\rho},$$

where $\rho = 1 - 2\mu\alpha/n + \mu L\alpha^2/n$. which completes the proof. ■

6.3 Over-parameterized problems have low gradient confusion

In the previous section, we showed that low *gradient confusion* can result in much faster convergence of SGD to the neighborhood of a critical point for general smooth

non-convex functions. The question still remains, however, when such a condition might arise, and how it might explain the effectiveness of SGD on neural net problems.

In practice the level of gradient confusion will depend on the structure of the training data. However, we can analyze gradient confusion for generic (i.e., random) model problems using methods from high-dimensional probability. We rigorously analyze the case where training data is randomly sampled from a unit sphere, and identify specific cases where gradient confusion (Definition 6.1.1) is low with high-probability.

We consider a synthetic datasets of the form $\mathcal{D} = \{(\mathbf{x}_i, \mathcal{C}(\mathbf{x}_i))\}_{i=1}^n$, for some labeling function $\mathcal{C}$. The data points $\{\mathbf{x}_i\}$ are drawn uniformly and independently from the surface of a d-dimensional unit sphere. The function $f_i(\mathbf{w})$ we consider is the least-squares function.

We show below that, given a confusion parameter η, a range of models (including neural networks) with randomized training data naturally attain confusion less than η provided the dimension of the problem is sufficiently large. The method we use to prove this result is very similar for a range of different problem classes, and so we begin by illustrating the result on the simple class of linear regression problems.

6.3.1 A simple case: linear regression

We begin by examining gradient confusion in the case of a simple linear least-squares regression. We assume that the hidden concept that the algorithm is trying to learn is given by $\mathcal{C}(\mathbf{x}) = \langle \tilde{\mathbf{w}}, \mathbf{x} \rangle$, for some "true" weight vector $\tilde{\mathbf{w}}$[1]. Throughout we will

[1]The arguments in this paper actually hold for the case of a noisy model $\mathcal{C}(\mathbf{x}) = \langle \tilde{\mathbf{w}}, \mathbf{x} \rangle + \zeta$ where ζ is a Gaussian random variable, however we will omit the noise term for notational simplicity.

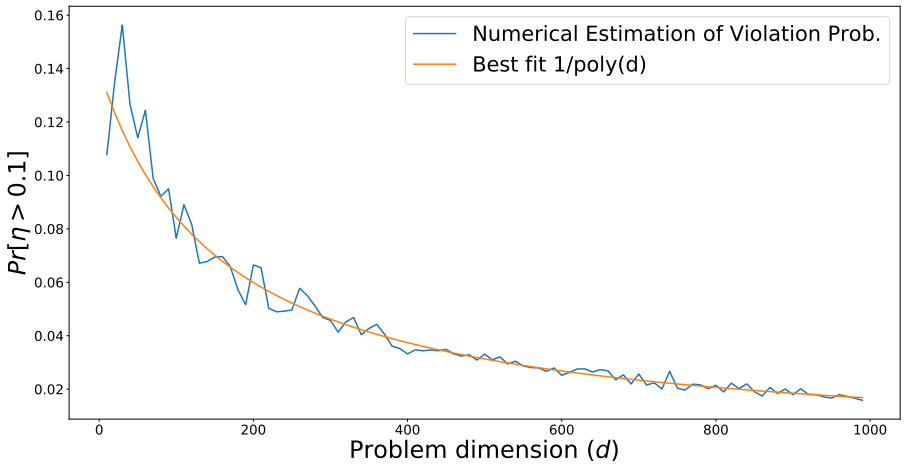


Figure 6.1: Simulation proof for Theorem 6.3.1. As the dimensionality of a random linear regression problem increases, the probability of violating the gradient confusion condition $\eta > 0.1$ vanishes.

denote by $g_{\mathbf{w}} : \mathbb{R}^d \to \mathbb{R}$ the function we fit to the training data. In this section we simply have

$$g_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle, \qquad \text{and} \qquad f_i(\mathbf{w}) = \frac{1}{2}(g_{\mathbf{w}}(\mathbf{x}_i) - \mathcal{C}(\mathbf{x}_i))^2,$$

but we will consider more complex situations below. For this problem, we have

$$\nabla f_i(\mathbf{w}) = \alpha_i \mathbf{x}_i, \qquad \text{where} \qquad \alpha_i := g_{\mathbf{w}}(\mathbf{x}_i) - \mathcal{C}(\mathbf{x}_i).$$

With this definition of the gradient, we can prove the following theorem.

Theorem 6.3.1 (Concentration for linear regression)**.** *Let* $\mathbf{w}, \tilde{\mathbf{w}} \in [-1, 1]^d$ *be the approximate and true weight vectors and let* $\eta > 0$ *be a given constant. For* $d > \Omega(\log n)$*, we have that with probability at least* $1 - \Omega\left(\frac{1}{\text{poly}(d)}\right)$*, equation* (6.7) *holds at* $\mathbf{w}$.

Figure 6.1 shows a numerical demonstration of this theorem.

Note that one limitation of Theorem 6.3.1, and of all other results presented in the rest of this section, is that the bound is *non-uniform* in the weights $\mathbf{w}$.

Technical approach & proof sketch

To prove that any bound on the gradient confusion $\eta > 0$ is attained for sufficiently large d, examine the values of the function $h(\mathbf{x}_i, \mathbf{x}_j) := \langle \nabla f_i, \nabla f_j \rangle$ when $\mathbf{x}_i$ and $\mathbf{x}_j$ are selected at random from the unit sphere. Our goal will be to show that this function has positive expected value, and that h concentrates around its expected value for large d, thus making it extremely unlikely that a large negative value of h is observed. Once we have shown that $h(\mathbf{x}_i, \mathbf{x}_j) < \eta$ with extremely low probability for a random pair of points $(\mathbf{x}_i, \mathbf{x}_j)$, we can use a union bound to show that this occurs with low probability for all pairwise comparisons between data points.

For a given constant $\eta > 0$, our goal is to find an appropriate concentration bound for the event

$$\text{for some } i \neq j, \qquad h(\mathbf{x}_i, \mathbf{x}_j) \geq -\eta. \tag{6.7}$$

In other words, we want to find a sharp bound $\tau(n, d, \eta)$ such that for fixed i, j

$$\Pr[h(\mathbf{x}_i, \mathbf{x}_j) \leq -\eta] \leq \tau(n, d, \eta). \tag{6.8}$$

By using the union bound and the fact that $h(\mathbf{x}_i, \mathbf{x}_j)$ is identically distributed for all $i, j \in [n]$, we have that

$$\Pr[\exists i \neq j, h(\mathbf{x}_i, \mathbf{x}_j) \leq -\eta] \leq n^2 \tau(n, d, \eta). \tag{6.9}$$

We use tools from high-dimensional probability to find an appropriate function τ for a large class of predictors g, and show that the quantity $n^2\tau(n, d, \eta)$ vanishes for large d.

Proof of Theorem 6.3.1

We will briefly describe some technical lemmas we require in our analysis. The following Chernoff-style concentration bound is proved in Chapter 5 of [Ver].

Lemma 6.3.1 (Concentration of Lipshitz function over a sphere)**.** *Let $\mathbf{x} \in \mathbb{R}^d$ be sampled uniformly from the surface of a d-dimensional sphere. Consider a Lipshitz function $\ell : \mathbb{R}^d \to \mathbb{R}$ which is differentiable everywhere. Let $||\nabla \ell||_\infty$ denote $\sup_{\mathbf{x}\in\mathbb{R}^d} \|\nabla \ell(\mathbf{x})\|_\infty$. Then for any $t \geq 0$ and some fixed constant $c \geq 0$, we have the following.*

$$\Pr\left[\left|\ell(\mathbf{x}) - \mathbb{E}[\ell(\mathbf{x})]\right| \geq t\right] \leq 2\exp\left(-\tfrac{cdt^2}{\rho^2}\right), \tag{6.10}$$

where $\rho \geq \|\nabla \ell\|_\infty$ is a entry-wise bound on $\nabla \ell$.

We will rely on the following generalization of Lemma 6.3.1.

Corollary 6.3.1. *Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ be two mutually independent vectors sampled uniformly from the surface of a d-dimensional sphere. Consider a Lipshitz function $\ell : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ which is differentiable everywhere. Let $\|\nabla \ell\|_\infty$ denote $\sup_{(\mathbf{x},\mathbf{y})\in\mathbb{R}^d\times\mathbb{R}^d} \|\nabla \ell(\mathbf{x},\mathbf{y})\|_\infty$. Then for any $t \geq 0$ and some fixed constant $c \geq 0$, we have the following.*

$$\Pr\left[\left|\ell(\mathbf{x},\mathbf{y}) - \mathbb{E}[\ell(\mathbf{x},\mathbf{y})]\right| \geq t\right] \leq 2\exp\left(-\tfrac{cdt^2}{\rho^2}\right), \tag{6.11}$$

where $\rho \geq \|\nabla \ell\|_\infty$ is a entry-wise bound on $\nabla \ell$.

Proof. This corollary can be derived from Lemma 6.3.1 as follows. Note that for every fixed $\tilde{\mathbf{y}} \in \mathbb{R}^d$, equation (6.10) holds. Additionally, we have that the vectors $\mathbf{x}$ and $\mathbf{y}$ are mutually independent. Hence we can write the LHS of equation (6.11) as the following.

$$\int_{(y)_1=-\infty}^{(y)_1=\infty} \cdots \int_{(y)_d=-\infty}^{(y)_d=\infty} \Pr\left[\left|\ell(\mathbf{x},\mathbf{y}) - \mathbb{E}[\ell(\mathbf{x},\mathbf{y})]\right| \geq t \;\middle|\; \mathbf{y} = \tilde{\mathbf{y}}\right] \phi(\tilde{\mathbf{y}}) d(\mathbf{y})_1 \ldots d(\mathbf{y})_d.$$

Here $\phi(\tilde{\mathbf{y}})$ refers to the pdf of the distribution of $\mathbf{y}$. From independence, the inner term in the integral evaluates to $\Pr\left[\left|\ell(\mathbf{x},\tilde{\mathbf{y}}) - \mathbb{E}[\ell(\mathbf{x},\tilde{\mathbf{y}})]\right| \geq t\right]$. We know this is less than or equal to $2\exp\left(-\frac{cdt^2}{||\nabla\ell||_\infty^2}\right)$. Therefore, the integral can be upper bounded by the following.

$$\int_{(y)_1=-\infty}^{(y)_1=\infty} \dots \int_{(y)_d=-\infty}^{(y)_d=\infty} 2\exp\left(-\frac{cdt^2}{||\nabla\ell||_\infty^2}\right)\phi(\tilde{\mathbf{y}})d(\mathbf{y})_1\dots d(\mathbf{y})_d.$$

Since $\phi(\tilde{\mathbf{y}})$ is a valid pdf, we get the required equation (6.11). ■

Additionally, we will use the following facts about a normalized Gaussian random variable.

Lemma 6.3.2. *For a normalized Gaussian* $\mathbf{x}$ *(*i.e., *an* $\mathbf{x}$ *sampled uniformly from the surface of a unit* d*-dimensional sphere) the following statements are true.*

1. $\forall i \in [d]$ *we have that* $\mathbb{E}[(\mathbf{x})_i] = 0$.
2. $\forall i \in [d]$ *we have that* $\mathbb{E}[(\mathbf{x})_i^2] = 1/d$.

Proof. Part (1) can be proved by observing that the normalized Gaussian random variable is spherically symmetric about the origin. In other words, for every $i \in [d]$ the vectors $(x_1, x_2, \dots, x_i, \dots, x_d)$ and $(x_1, x_2, \dots, -x_i, \dots, x_d)$ are identically distributed. Hence $\mathbb{E}[x_i] = \mathbb{E}[-x_i]$ which implies that $\mathbb{E}[x_i] = 0$.

Part (2) can be proved by observing that for any $i, j \in [d]$, x_i and x_j are identically distributed. Fix any $i \in [d]$. We have that $\sum_{j=1}^{d}\mathbb{E}[x_j^2] = d \times \mathbb{E}[x_i^2]$. Note that we have

$$\sum_{j=1}^{d}\mathbb{E}[x_j^2] = \int_{(x)_1=-\infty}^{(x)_1=\infty} \dots \int_{(x)_d=-\infty}^{(x)_d=\infty} \frac{\sum_{j=1}^{d} x_j^2}{\sum_{j'=1}^{d} x_{j'}^2}\phi(\mathbf{x})d(x)_1\dots d(x)_d = 1.$$

Therefore $\mathbb{E}[x_i^2] = 1/d$. ■

We are now ready to prove Theorem 6.3.1.

Proof. The proof illustrates a general strategy we will use for other general models. We will prove that $h(\cdot,\cdot)$ has two properties, namely that $\nabla h(\cdot,\cdot)$ is bounded, and the entries in $\nabla h(\cdot,\cdot)$ have non-negative expectation. These properties enable us to use Corollary 6.3.1 to show that $h(\cdot,\cdot)$ concentrates on non-negative values.

Bounded Gradient. Fix an arbitrary value $\mathbf{x}_i = \tilde{\mathbf{x}}_i$ and $\mathbf{x}_j = \tilde{\mathbf{x}}_j$. Consider an arbitrary coordinate corresponding to the variable $(\mathbf{x}_i)_p$ (by symmetry a corresponding argument holds for $(\mathbf{x}_j)_p$) in the vector $\nabla h(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)$. The term evaluates to $\alpha_i\alpha_j(\tilde{\mathbf{x}}_j)_p + (\boldsymbol{\Delta})_p\alpha_j\langle\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j\rangle$. Note that we have $|(\tilde{\mathbf{x}}_i)_p| \leq 1$ for every $i \in [n], p \in [d]$. Additionally, from our assumption on weights $\mathbf{w}$ and $\tilde{\mathbf{w}}$ we have $-2 \leq \sum_{p=1}^{d}(\boldsymbol{\Delta})_p \leq 2$. Therefore we have that for every $i \in [n]$, $-2 \leq \alpha_i \leq 2$. And hence $\alpha_i\alpha_j(\tilde{\mathbf{x}}_j)_p + \alpha_j\sum_{p=1}^{d}(\boldsymbol{\Delta})_p(\tilde{\mathbf{x}}_i)_p(\tilde{\mathbf{x}}_j)_p \leq 8$. Hence we have that $||\nabla h||_\infty \leq \rho = 8$. In particular, the upper bound ρ is a constant.

Non-negative Expectation. To compute bounds on $\mathbb{E}[h(\mathbf{x}_i, \mathbf{x}_j)]$, we want to evaluate $\mathbb{E}[\alpha_i\alpha_j\langle\mathbf{x}_i, \mathbf{x}_j\rangle]$. On expanding the product and removing all summands where either $(\mathbf{x}_i)_p$ or $(\mathbf{x}_j)_p$ appear as an odd-term, we have $\mathbb{E}[h(\mathbf{x}_i, \mathbf{x}_j)] = \|\boldsymbol{\Delta}\|^2/d^2$. Therefore we have $0 \leq \mathbb{E}[h(\mathbf{x}_i, \mathbf{x}_j)]$. Alternatively, we can obtain this lower-bound as follows. Note that since $-2 \leq \alpha_i, \alpha_j \leq 2$, we have that $\mathbb{E}[\alpha_i\alpha_j\langle\mathbf{x}_i, \mathbf{x}_j\rangle] \geq -4\mathbb{E}[\langle\mathbf{x}_i, \mathbf{x}_j\rangle] = 0$. The last equality is because of the following: $\mathbb{E}[\langle\mathbf{x}_i, \mathbf{x}_j\rangle] = \mathbb{E}[\sum_{p=1}^{d}(\mathbf{x}_i)_p(\mathbf{x}_j)_p] = \sum_{p=1}^{d}\mathbb{E}[(\mathbf{x}_i)_p(\mathbf{x}_j)_p] = \sum_{p=1}^{d}\mathbb{E}[(\mathbf{x}_i)_p]\mathbb{E}[(\mathbf{x}_i)_p] = 0$. The second equality follows from Linearity of Expectation, the third from independence of $\mathbf{x}_i$ and $\mathbf{x}_j$ and the last follows from Lemma 6.3.2.

We combine the two properties as follows. From **Non-negative Expectation** property and equation (6.11), we have that

$$\Pr[h(\mathbf{x}_i, \mathbf{x}_j) \leq -\eta] \leq \Pr[h(\mathbf{x}_i, \mathbf{x}_j) \leq \mathbb{E}[h(\mathbf{x}_i, \mathbf{x}_j)] - \eta] \leq 2\exp\left(-\tfrac{cd\eta^2}{\rho^2}\right).$$

The probability that *some* value of $h(\nabla f_i, \nabla f_j)$ lies below $-\eta$ is then bounded by $2n^2 \exp\left(-\frac{cd\eta^2}{\rho^2}\right)$. For any choice of c_1, we can solve the inequality $2n^2 \exp\left(-\frac{cd\eta^2}{\rho^2}\right) \leq d^{-c_1}$, to get $d > \Omega(\rho^2 \log n)$ and $\eta > \Omega(1/\sqrt{d})$. In particular, the bound holds for any constant $\eta > 0$. ■

6.3.2 Linear neural networks

We now study the behavior of gradient confusion for neural networks. We begin with the simplified case of linear neural networks (i.e., with no non-linearities) with one output feature, one hidden layer, and a quadratic loss. We'll then examine the case of more general networks.

Let $\mathbf{W}_0 \in [-1,1]^{d \times \ell_1}$, $\mathbf{W}_1 \in [-1,1]^{\ell_1 \times 1}$ denote the weight matrices connecting the input layer to the hidden layer, and the hidden layer to the output, respectively. Thus, the output of the neural net is given by $\mathbf{W}_1\mathbf{W}_0\mathbf{x}$. Then we have,

$$g(\mathbf{x}) = \sum_{p=1}^{d} \sum_{p'=1}^{\ell_1} (\mathbf{W}_1)_{p'} (\mathbf{W}_0)_{p,p'} (\mathbf{x})_p, \text{ and } \alpha_i := g(\mathbf{x}_i) - \mathcal{C}(\mathbf{x}_i).$$

Later, we consider the case of more than one hidden-layer, and use ℓ_i to denote the width of layer i and β to be the total number of hidden layers. Further we define $\ell := \max_{i \in [\beta]} \ell_i$. Throughout this sub-section we make the following assumption.

Assumption 6.3.1 (Small Weights)**.** *We will assume* $\mathbf{W}$ *is such that the outputs at each neuron of the hidden layer,* $\mathcal{C}(\mathbf{x})$ *and* $g(\mathbf{x})$ *all lie between* $[-1,1]$ *for every value of* $\mathbf{x}$ *in*

the unit ball[2]*. Additionally, we will assume that the entries in the weight matrices* $\mathbf{W}_i$ *(*e.g., $\mathbf{W}_0$, $\mathbf{W}_1$, …, $\mathbf{W}_\beta$*) lie in the range* $[-1/\ell, 1/\ell]$.

Is the small-weights assumption reasonable? If we relax the assumption and let every entry in each of the weight matrices lie in the range $[-1, 1]$, then a product of matrices $\mathbf{W}_\beta \cdot \mathbf{W}_{\beta-1} \cdot \ldots (\mathbf{W}_{\beta'})_{k_{\beta'}}$ may lead to an exponential blow up in values. If we have vectors $\mathbf{v}_1, \mathbf{v}_2 \in [-1, 1]^\ell$ then $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle$ can be as large as ℓ. Hence in a sequence of β products of matrices, the final value can become as large as ℓ^β. Note this would imply that $||\nabla h||_\infty \leq \ell^\beta$ and hence we need $d \geq \Omega(\ell^{2\beta} \log n)$ for the required concentration. The small weights assumption is not just a theoretical concern, but also usually mandated in practice. Without small weights, the gradients ∇f_i blows up in magnitude and a phenomenon known as *gradient-explosion* (which is particularly problematic for recurrent neural networks) is observed. This may indicate why weights are typically initialized as $\mathcal{N}(0, 1/\ell_i)$ in practice [Ben12].

Similar to the case for linear regression, we prove the following theorem. A full discussion of this result, and a rigorous proof, are in the Appendix.

Theorem 6.3.2 (Bounded gradient confusion for single layer linear neural network)**.** *Consider a single hidden layer neural network with fixed weight vectors* $\mathbf{W}_1 \in \mathbb{R}^{1 \times \ell_1}, \mathbf{W}_0 \in \mathbb{R}^{\ell_1 \times d}$ *satisfying Assumption 6.3.1, and let* $\eta > 0$ *be a given constant. We have that with probability at least* $1 - \Omega\left(n \exp(-c_1 \ell^2 d \eta^2)\right)$*, equation* (6.7) *holds.*

Proof. We will prove the two properties.

[2]Note one could use an activation function (sigmoid, tanh, or softmax) to enforce this assumption. For now we will assume that we can bound the outputs appropriately.

Bounded Gradient. Fix an arbitrary value $\mathbf{x}_i = \tilde{\mathbf{x}}_i$ and $\mathbf{x}_j = \tilde{\mathbf{x}}_j$. Consider an arbitrary coordinate corresponding to the variable $(\mathbf{x}_i)_p$ (by symmetry a corresponding argument holds for $(\mathbf{x}_j)_p$) in the vector $\nabla \mathbf{h}(\mathbf{x}, \mathbf{y})$. The term evaluates to the following.

$$\alpha_i \alpha_j \left(\sum_{p''=1}^{\ell_1} \sum_{p'=1}^{d} (\mathbf{W}_0)_{p,p''} (\mathbf{W}_0)_{p',p''} (\mathbf{x}_j)_{p'} + \sum_{p'=1}^{\ell_1} (\mathbf{W}_1)_{p'}^2 (\mathbf{x}_j)_p \right)$$
$$+ \alpha_j \kappa_{i,j} \left(\sum_{p'=1}^{\ell_1} (\mathbf{W}_1)_{p'} (\mathbf{W}_0)_{p,p'} - (\tilde{\mathbf{W}})_p \right).$$

By the assumption that both $\mathcal{C}(\mathbf{x})$ and $g(\mathbf{x})$ lie between $[-1, 1]$, we have that $-2 \leq \alpha_i \leq 2$ for all $i \in [n]$. From Assumption 6.3.1 assumption and the sampling procedure of $\mathbf{x}_i$'s we have the following.

$$\sum_{p=1}^{d} (\mathbf{W}_0)_{p,p''} (\mathbf{x}_i)_p \leq \frac{1}{\ell_1} \sum_{p=1}^{d} (\mathbf{x}_i)_p \leq \frac{1}{\ell_1}. \tag{6.12}$$

Hence, combining this with Assumption 6.3.1 assumption, the first term in the sum above can be upper-bounded by $8/\ell_1$. Note that for every pair $i, j \in [n]$ we have that $-1 \leq \sum_{p=1}^{d} (\mathbf{x}_i)_p (\mathbf{x}_j)_p \leq 1$ since this quantity represents the cosine of the angle between two vectors. Again using the observation in Equation (6.12), we have that $\kappa_{i,j} \leq 2/\ell_1$. Hence, the second term in the summand above can be upper-bounded by $8/\ell_1$ and hence $||\nabla(\mathbf{h}(\mathbf{x_i}, \mathbf{x_j}))||_\infty \leq 16/\ell_1$. In other words, we have that ρ is $O(1/\ell_1)$.

Non-negative Expectation. Note that $\alpha_i, \alpha_j \geq -2$. Therefore we have $\mathbb{E}[h(\mathbf{x}_i, \mathbf{x}_j)] \geq -4\mathbb{E}[\kappa_{i,j}]$. By using Linearity of Expectation and the fact that normalized Gaussian random variables have mean 0 at every co-ordinate, we have that $\mathbb{E}[\kappa_{i,j}] = 0$. Therefore we have $\mathbb{E}[h(\mathbf{x}_i, \mathbf{x}_j)] \geq 0$.

Armed with the bounded gradient and non-negative expectation properties, the rest of the proof follows as in the proof of Theorem 6.3.1. ■

Note that from theorem 6.3.2 we have that the concentration gets sharper as the

width of the hidden layer increases. In particular, SGD converges faster (on the simulated dataset) with increasing width.

6.3.3 Extension to arbitrary depth linear networks

We now extend the above analysis to an arbitrary depth linear neural network. Interestingly, we see in this case that the gradient confusion depends critically on the network architecture – in particular the width and depth.

In this case we have the model $g(\mathbf{x}) := \mathbf{W}_\beta \mathbf{W}_{\beta-1} \dots \mathbf{W}_1 \mathbf{W}_0 \mathbf{x}$ where $\{\mathbf{W}_i\}_{i=1}^{\beta}$ are weight matrices of appropriate dimensions. We let β denote the number of layers in our hidden network, and we let ℓ denote the width of our network – i.e., the maximal number of features in any layer.

Theorem 6.3.3 (Extension to arbitrary depth linear neural networks)**.** *Consider an arbitrary depth linear neural network, and assume the weights satisfy Assumption 6.3.1. Let* $\eta > 0$ *be a given constant. With probability at least* $1 - \Omega\left(n \exp(\frac{-c\ell^2 d\eta^2}{\beta^2})\right)$ *we have that equation* (6.7) *holds.*

Proof. This statement can be equivalently written as follows. Note that for constants $c', c'' > 0$, we want $c' n^2 \exp(\frac{-c\ell^2 d\eta^2}{\beta^2}) \leq c'' \frac{1}{\mathsf{poly}(d)}$,, since we we assume that d is the asymptotic parameter and hence this makes the concentration explicit. Rearranging and solving for d, we get the condition that $d \geq \Omega((\beta/\ell)^2 \log n)$.

We now show the two properties.

Bounded gradient. We will first compute $\nabla f_i(\mathbf{W}_\beta, \dots, \mathbf{W}_0)$. Note that the gradient can be visualized as follows. The first ℓ_β entries correspond to the entries in $\frac{\partial f_i}{\partial \mathbf{W}_\beta}$. The

next $\ell_\beta * \ell_{\beta-1}$ entries correspond to $\frac{\partial f_i}{\partial \mathbf{W}_{\beta-1}}$ and so on. These entries can be computed as follows.

$$\left(\frac{\partial f_i}{\partial \mathbf{W}_\beta}\right)_{p_\beta} = \alpha_i \left(\sum_{p_{\beta-1}=1}^{\ell_{\beta-1}} \cdots \sum_{p_1=1}^{\ell_1} \sum_{p=1}^{d} (\mathbf{W}_{\beta-1})_{p_\beta, p_{\beta-1}} \cdots (\mathbf{W}_0)_{p_1,p} (\mathbf{x}_i)_p \right).$$

$$\left(\frac{\partial f_i}{\partial \mathbf{W}_{\beta'}}\right)_{p_{\beta'+1}, p_{\beta'}} = \alpha_i \Bigg(\sum_{p_\beta=1}^{\ell_\beta} \cdots \sum_{p_{\beta'+2}=1}^{\ell_{\beta'+2}} \sum_{p_{\beta'-1}=1}^{\ell_{\beta'-1}} \cdots \sum_{p_1=1}^{\ell_1} \sum_{p=1}^{d}$$

$$(\mathbf{W}_\beta)_{p_\beta} (\mathbf{W}_{\beta-1})_{p_\beta, p_{\beta-1}} \cdots (\mathbf{W}_{\beta'+1})_{p_{\beta'+2}, p_{\beta'+1}} (\mathbf{W}_{\beta'-1})_{p_{\beta'}, p_{\beta'-1}} \cdots (\mathbf{W}_0)_{p_1,p} (\mathbf{x}_i)_p \Bigg)$$

$$\forall \beta' \in \{1, 2, \ldots, \beta-1\}.$$

$$\left(\frac{\partial f_i}{\partial \mathbf{W}_0}\right)_{p_1,p} = \alpha_i \left(\sum_{p_\beta=1}^{\ell_\beta} \cdots \sum_{p_2=1}^{\ell_2} (\mathbf{W}_\beta)_{p_\beta} \cdots (\mathbf{W}_1)_{p_2,p_1} (\mathbf{x}_i)_p \right).$$

Hence $h(\mathbf{x}_i, \mathbf{x}_j)$ can be written as follows.

$$h(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p_\beta=1}^{\ell_\beta} \left(\frac{\partial f_j}{\partial \mathbf{W}_\beta}\right)_{p_\beta} \left(\frac{\partial f_i}{\partial \mathbf{W}_\beta}\right)_{p_\beta} + \sum_{\beta'=1}^{\beta-1} \sum_{p_{\beta'+1}=1}^{\ell_{\beta'+1}} \sum_{p_{\beta'}=1}^{\ell_{\beta'}} \left(\frac{\partial f_j}{\partial \mathbf{W}_{\beta'}}\right)_{p_{\beta'+1}, p_{\beta'}} \left(\frac{\partial f_i}{\partial \mathbf{W}_{\beta'}}\right)_{p_{\beta'+1}, p_{\beta'}}$$
$$+ \sum_{p_1=1}^{\ell_1} \sum_{p=1}^{d} \left(\frac{\partial f_j}{\partial \mathbf{W}_0}\right)_{p_1,p} \left(\frac{\partial f_i}{\partial \mathbf{W}_0}\right)_{p_1,p}.$$

To bound $||\nabla \mathbf{h}||_\infty$, consider any fixed $\mathbf{x}_i = \tilde{\mathbf{x}}_i$ and $\mathbf{x}_j = \tilde{\mathbf{x}}_j$. Consider the entry corresponding to $(\mathbf{x}_i)_p$. The following claim can be obtained as a consequence of **small-weights** Assumption.

Lemma 6.3.3. *Let $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_\beta$ be weight matrices the satisfying Assumption 6.3.1. Then for any given $k_{\beta'}$, we have that any product consisting of a sequence of matrices $\mathbf{W}_\beta \cdot \mathbf{W}_{\beta-1} \cdot \ldots (\mathbf{W}_{\beta'})_{p_{\beta'}}$ lies in the interval $[-1/\ell, 1/\ell]$.*

Proof. For notational convenience, denote the column vector $(\mathbf{W}_{\beta'})_{p_{\beta'}}$ as $\overline{\mathbf{W}}_0$. Define $\nu := \beta - \beta'$. We also rewrite the chain $\mathbf{W}_\beta \cdot \mathbf{W}_{\beta-1} \cdot \ldots (\mathbf{W}_{\beta'})_{p_{\beta'}}$ as $\overline{\mathbf{W}}_\nu \cdot \overline{\mathbf{W}}_{\nu-1} \cdot \ldots \cdot \overline{\mathbf{W}}_0$. We

will now prove that the value of this matrix-vector product lies in the interval $[-1/\ell, 1/\ell]$. By induction on ν, we will prove that the product $\overline{\mathbf{v}}_\nu := \overline{\mathbf{W}}_{\nu-1} \cdot \ldots \cdot \overline{\mathbf{W}}_0$ will always yield a column vector where every entry lies in the interval $[-1/\ell, 1/\ell]$. Given the proof of this inductive hypothesis, note that we have the following inner-product $\langle \overline{\mathbf{W}}_\nu, \mathbf{v}_\nu \rangle$ where every entry in both these vectors lies in the interval $[-1/\ell, 1/\ell]$. Since the dot-product is a sum over at most ℓ terms with each term in the sum bounded in the interval $[-1/\ell^2, 1/\ell^2]$, we have that the dot-product lies in the interval $[-1/\ell, 1/\ell]$.

We will now prove the inductive statement. The base-case is when $\nu = 1$. In this case we have a single vector $\overline{\mathbf{W}}_0$. By assumption we have that each entry in this vector lies in the interval $[-1/\ell, 1/\ell]$ and therefore the hypothesis is true. Consider the case when $\nu > 1$. Consider the chain $\overline{\mathbf{v}}_\nu := \overline{\mathbf{W}}_{\nu-1} \cdot \overline{\mathbf{W}}_{\nu-2} \cdot \ldots \cdot \overline{\mathbf{W}}_0$. From the inductive hypothesis, we have that $\overline{\mathbf{v}}_{\nu-1} := \overline{\mathbf{W}}_{\nu-2} \cdot \ldots \cdot \overline{\mathbf{W}}_0$ gives a column vector where every entry is in the interval $[-1/\ell, 1/\ell]$. Consider the i-th entry in the column vector $\overline{\mathbf{v}}_\nu$. This is obtained by the inner product of the i-th row of matrix $\overline{\mathbf{W}}_{\nu-1}$ and the column vector $\overline{\mathbf{v}}_{\nu-1}$. Since this is a sum of at most ℓ term with each term in the interval $[-1/\ell^2, 1/\ell^2]$, we have that this i-th entry lies in the interval $[-1/\ell, 1/\ell]$. ■

In the analysis for general neural networks, we use the following corollary.

Corollary 6.3.2. *Let $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_\beta$ be weight matrices the satisfying Assumption 6.3.1. Then for any given $\beta \leq \beta' \leq 1$, we have that any product consisting of a sequence of matrices $\mathbf{W}'_\beta \cdot \mathbf{W}_{\beta'-1} \cdot \ldots \mathbf{W}_0 \cdot \mathbf{x}$ lies in the interval $[-1/\ell, 1/\ell]$, where $\mathbf{x}$ is a normalized Gaussian.*

Proof. The proof of this follows directly by combining Equation 6.12 and the above

theorem. ■

By the small weights assumption and lemma 6.3.3, we have that each entry in $\nabla f_i(\mathbf{W}_\beta, \ldots, \mathbf{W}_0)$ is at most $3/\ell$ (*i.e.,* $\alpha_i \leq 2$ and the terms involving the sum over weight matrices is at most $1/\ell$). By using the small-weights assumption repeatedly and noting that $\alpha_i, \alpha_j \leq 2$, we get that each entry in the gradient is at most $18/\ell^2 + 18(\beta - 1)/\ell + 18/\ell^2 \leq 54\beta/\ell$.

Hence we have ρ as $O(\beta/\ell)$.

Non-negative expectation. As before, using the fact that $\alpha_i \geq -2$ and $\alpha_j \geq -2$ and normalized gaussian random variable has mean 0, we have that $\mathbb{E}[h(\mathbf{x}_i, \mathbf{x}_j)] \geq 0$.

Note that the bound in this case depends both on the maximum "width" and the "depth". In particular, as depth increases or width decreases, we need the dimension d to be larger to get the required concentration. The other way to interpret this is when either the width increases or the depth decreases, the probability that Equation (6.7) holds increases. ■

6.3.4 More general neural networks

This result can be extended to models with certain non-linearities. We consider the model $g(\mathbf{x}) := \sigma(\mathbf{W}_\beta \sigma(\mathbf{W}_{\beta-1} \ldots \sigma(\mathbf{W}_1 \sigma(\mathbf{W}_0 \mathbf{x})) \ldots)$. Here, the function $\sigma(.)$ is applied point-wise to its arguments. We will assume that the non-linear activation is given by a function $\sigma(x)$ with the following properties.

- **(P1) Boundedness:** $-1 \leq \sigma(x) \leq 1$ for every value of $x \in \mathbb{R}$.
- **(P2) Twice Differentiability:** σ is twice differentiable at every point in $\mathbb{R}$.

- **(P3) Bounded Differentials:** $-1 \leq \sigma'(x) \leq 1$ and $-1 \leq \sigma''(x) \leq 1$ for all $x \in \mathbb{R}$.

Most classical activation functions such as *sigmoid*, *tanh*, and *softmax* satisfy these requirements (although *relu* does not).

Provided the small-weights assumption holds[3], we have the following theorem which is analogous to the case of linear neural networks, with a mildly stronger requirement on the constant η and weaker dependence on depth β.

Theorem 6.3.4 (Concentration bounds for arbitrary depth neural networks)**.** *Let $\eta > 4$ be a given constant. Consider a neural network with weights satisfying Assumption 6.3.1, non-linearity satisfying properties P1-P3, and quadratic loss. With probability at least $1 - \Omega\left(n \exp(\frac{-c\ell^2 d(\eta-4)^2}{\beta^4}\right)$ we have that equation* (6.7) *holds.*

The proof of this theorem follows in a similar manner as for linear neural networks. We critically use the property that terms involving $\sigma(...)$, $\sigma'(..)$ and $\sigma''(...)$ lie in the range $[-1, 1]$. In particular, we have the following expression for $\nabla f_i(\mathbf{W}_\beta, \ldots, \mathbf{W}_0)$. Here $A_1, A_2, \ldots, A_\beta$ are some fixed expressions involving the weight matrices and the random vector $\mathbf{x}_i$.

$$\left(\frac{\partial f_i}{\partial \mathbf{W}_\beta}\right)_{p_\beta} = \alpha_i \sigma'(...)(\sigma(\mathbf{W}_{\beta-1} \cdot \sigma(\ldots \sigma(\mathbf{W}_0 \cdot \mathbf{x}_i))))_{p_\beta}.$$

$$\left(\frac{\partial f_i}{\partial \mathbf{W}_{\beta'}}\right)_{p_{\beta'+1}, p_{\beta'}} = \alpha_i \sigma'(A_1)\sigma'(A_2)\ldots\sigma'(A_{\beta-\beta'})\left(\sum_{p_\beta=1}^{\ell_\beta} \ldots \sum_{p_{\beta'+2}=1}^{\ell_{\beta'+2}} \right.$$

$$\left.(\mathbf{W}_\beta)_{p_\beta}(\mathbf{W}_{\beta-1})_{p_\beta, p_{\beta-1}} \ldots (\mathbf{W}_{\beta'+1})_{p_{\beta'+2}, p_{\beta'+1}} \sigma(\mathbf{W}_{\beta'-1} \cdot \sigma(\mathbf{W}_{\beta'-2} \ldots \sigma(\mathbf{W}_0 \cdot \mathbf{x}_i)) \ldots)_{p_{\beta'}}\right)$$

[3]The activation function guarantees that the output at each layer is at most 1. However we still need the assumption that each entry in the weight matrix is not too large. Otherwise, the gradient can blow up on the backward pass, even if the forward pass is stable.

$$\forall \beta' \in \{1, 2, \ldots, \beta - 1\}.$$

$$\left(\frac{\partial f_i}{\partial \mathbf{W}_0}\right)_{p_1,p} = \alpha_i \sigma'(A_1)\sigma'(A_2)\ldots\sigma'(A_\beta)\left(\sum_{p_\beta=1}^{\ell_\beta}\ldots\sum_{p_2=1}^{\ell_2}(\mathbf{W}_\beta)_{p_\beta}\ldots(\mathbf{W}_1)_{p_2,p_1}(\mathbf{x}_i)_p\right).$$

Consider the expression for $h(\mathbf{x}_i, \mathbf{x}_j)$ as follows.

$$h(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p_\beta=1}^{\ell_\beta}\left(\frac{\partial f_j}{\partial \mathbf{W}_\beta}\right)_{p_\beta}\left(\frac{\partial f_i}{\partial \mathbf{W}_\beta}\right)_{p_\beta} + \sum_{\beta'=1}^{\beta-1}\sum_{p_{\beta'+1}=1}^{\ell_{\beta'+1}}\sum_{p_{\beta'}=1}^{\ell_{\beta'}}\left(\frac{\partial f_j}{\partial \mathbf{W}_{\beta'}}\right)_{p_{\beta'+1},p_{\beta'}}\left(\frac{\partial f_i}{\partial \mathbf{W}_{\beta'}}\right)_{p_{\beta'+1},p_{\beta'}} + \sum_{p_1=1}^{\ell_1}\sum_{p=1}^{d}\left(\frac{\partial f_j}{\partial \mathbf{W}_0}\right)_{p_1,p}\left(\frac{\partial f_i}{\partial \mathbf{W}_0}\right)_{p_1,p}.$$

To bound $||\nabla h||_\infty$, consider any fixed $\mathbf{x}_i = \tilde{\mathbf{x}}_i$ and $\mathbf{x}_j = \tilde{\mathbf{x}}_j$. Consider the entry corresponding to $(\mathbf{x}_i)_p$. Since $-1 \leq \sigma'(..) \leq 1$ we can use the same results from linear neural network to upper bound the value of partial differentials involving the function f_j. To compute the partial differential with respect to $(\mathbf{x}_i)_p$ we make the following observation. Differential with respect to $(\mathbf{x}_i)_p$ in $\left(\frac{\partial f_i}{\partial \mathbf{W}_{\beta'}}\right)_{p_{\beta'+1},p_{\beta'}}$ involves two terms both of which can be upper bounded by $2/\ell$ using Corollary 6.3.2 and Lemma 6.3.3. The differential in $\left(\frac{\partial f_i}{\partial \mathbf{W}_{\beta'}}\right)_{p_{\beta'+1},p_{\beta'}}$ involves β' terms each of them upper-bounded by $2/\ell$ by using the fact that $-1 \leq \sigma'' \leq 1$, Corollary 6.3.2 and Lemma 6.3.3. Finally the differential in $\left(\frac{\partial f_i}{\partial \mathbf{W}_0}\right)_{p_1,p}$ can be upper-bounded by $2\beta/\ell$.

Therefore, we get that $\rho \leq O(\ell/\beta^2)$.

Computing a lower-bound on the expectation is tricky because of the function σ. Although we cannot show the non-negative expectation property in this case, we can show a slightly weaker lower-bound that still suffices for our purposes. We show that $\mathbb{E}[h(\mathbf{x}_i, \mathbf{x}_j)] \geq -4$. The proof of this follows from the assumption that $\alpha_i \geq -2$ and $\alpha_j \geq -2$ and the fact that the remaining terms in $h(\mathbf{x}_i, \mathbf{x}_j)$ involve products of terms

σ, σ' and the product of matrices. From the boundedness assumptions on σ and its first-derivative and from lemma 6.3.3 we have that these terms are all at least -1. Hence we have that $\mathbb{E}[h(\mathbf{x}_i, \mathbf{x}_j)] \geq -4$.

To complete the proof, we now observe the following.

$$\Pr[h(\mathbf{x}_i, \mathbf{x}_j) \leq -\eta] \leq \Pr[h(\mathbf{x}_i, \mathbf{x}_j) \leq \mathbb{E}[h(\mathbf{x}_i, \mathbf{x}_j)] - \eta + 4] \leq 2 \exp\left(-\frac{cd(\eta - 4)^2}{\rho^2}\right)$$

Making the substitution of $\eta' := \eta - 4$ we obtain the theorem. Note that to have $\eta' > 0$ we want $\eta > 4$.

6.3.5 Beyond linearly generated data

In this sub-section we will briefly show when the theorem (and the proofs) in the previous sub-sections extend to the case beyond linearly generated data. Note that the only fact we used about the function $\mathcal{C}(\mathbf{x})$ was that it always lies in the interval $[-1, 1]$. This implies that $-2 \leq \alpha_i \leq 2$ and $-2 \leq \alpha_j \leq 2$ holds. When considering $||\nabla h||_\infty$ we used the first derivative of the concept function $\mathcal{C}$. In linearly generated data, we get that this first-derivative exists and it lies in the interval $[-1, 1]$. Hence the bounds on $||\nabla h||_\infty$ and $\mathbb{E}[h(\mathbf{x}_i, \mathbf{x}_j)]$ follow directly from these observations.

Therefore for any concept function $\mathcal{C}$ with the following properties, the above concentration theorems extend as is.

1. For every $\mathbf{x} \in [-1, 1]^d$ we have $-1 \leq \mathcal{C}(\mathbf{x}) \leq 1$.

2. The function $\mathcal{C}$ is differentiable everywhere in the interval $[-1, 1]^d$.

3. For every $\mathbf{x} \in [-1, 1]^d$ and every $i \in [d]$ we have that $-1 \leq \frac{\partial \mathcal{C}(\mathbf{x})}{\partial (\mathbf{x})_i} \leq 1$.

6.4 Experiments

We present experimental results to see the effect of depth and width on convergence rates and gradient confusion. It is worth noting that Theorem 6.3.4 implies that SGD becomes more effective when width increases or depth decreases. Also, Theorem 6.2.1 indicates that gradient confusion affects the heights of the final "noise floor" of constant step size SGD, and so we expect the effect of gradient confusion to be most prominent near the end of training, particularly when the convergence curve has flattened out near this floor.

We perform experiments on wide residual networks (WRN) [ZK16] for an image classification task on CIFAR-10. WRN is an extension of ResNet [HZRS16a], which is one of the state-of-the-art architectures for image classification. WRN is a stack of residual blocks, and we denote the architecture as WRN-β-ℓ following [ZK16], where β represents the depth and ℓ represents the width factor of the network[4].

The WRN architecture for CIFAR datasets is a stack of three groups of residual blocks. There is a downsampling layer between two blocks, and the number of channels (width of a convolutional layer) is doubled after downsampling. In the three groups, the width of convolutional layers is $\{16\ell, 32\ell, 64\ell\}$, respectively. Each group contains β_r residual blocks, and each residual block contains two 3×3 convolutional layers equipped with ReLU activation, batch normalization and dropout. There is a 3×3 convolutional layer with 16 channels before the three groups of residual blocks. And there is a global

[4]The width factor is the number of filters relative to the original ResNet, e.g., a factor of 1 corresponds to the original ResNet, and 2 means the network is twice as wide.

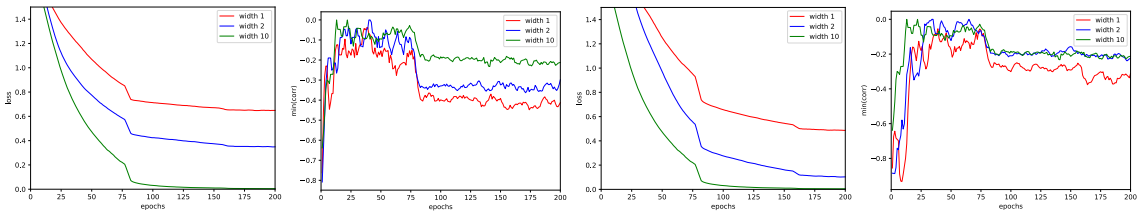
average pooling, a fully-connected layer and a softmax layer after the three groups. The depth of WRN is $\beta = 6\beta_r + 4$.

We turn off dropout for all our experiments. Our first round of experimental networks have no skip connections or batch normalization [IS15b] so as to stay as close to the assumptions of our theorems as possible. Later on, we study the effects that skip connections and batch normalization have on convergence rate and gradient confusion. We use SGD as the optimizer with no momentum. We train all experiments for 200 epochs and use a standard learning rate decay schedule, where the initial learning rate is reduced by a factor of 10 at epochs 80 and 160. We use a mini-batch of 128 for all our experiments.

To measure gradient confusion, at the end of every training epoch, we sample 100 mini-batches each of size 128. We calculate gradients on each of these mini-batches, and then calculate pairwise cosine similarities. To measure the worse-case gradient confusion, we calculate the lowest gradient cosine similarity among all pairs.

Effect of width. To test our theoretical results, and in particular Theorem 6.3.4, we consider a WRN with no batch normalization and no skip connections. This makes the network behave like a typical deep convolutional neural network. We now test the effect of increasing width in this network, while keeping the depth fixed. In particular, we consider the following networks: WRN-28-1, WRN-28-2, WRN-28-10. Figure 6.2 shows how the training loss and the minimum gradient cosine similarity is affected by a change in width. We present results with both a fixed initial learning rate across all networks, as well as where we tune the optimal initial learning rate to optimize the performance of each network. Quite clearly, width helps in faster convergence, as well as lower gradient

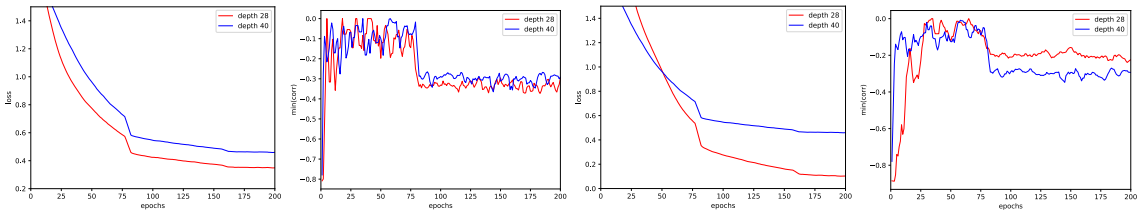
confusion.



(a) Loss; Fixed LR (b) Confusion; Fixed LR (c) Loss, Best LR (d) Confusion, Best LR

Figure 6.2: How width affects convergence curves and gradient inner products.

Effect of depth. Using the same experimental setup as above, we now keep the width fixed, and change the depth over the networks WRN-28-2 and WRN-40-2. Figure 6.3 shows the results. We again see that our theoretical results seem to be backed by the experiments, where we find faster convergence and lower gradient confusion with smaller depth.

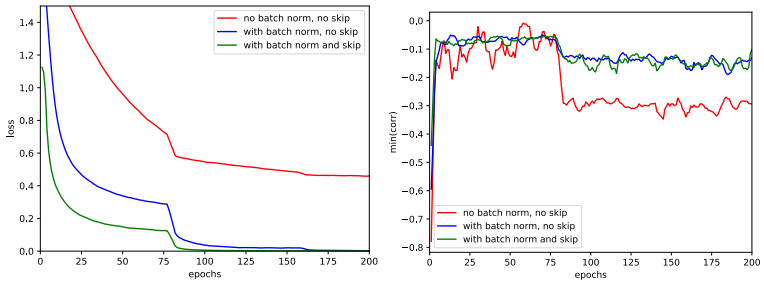


(a) Loss; Fixed LR (b) Confusion; Fixed LR (c) Loss, Best LR (d) Confusion, Best LR

Figure 6.3: How depth affects convergence curves and gradient inner products.

Effect of batch normalization and skip connections. Finally, we test the effect that techniques such as batch normalization and adding skip connections has on convergence

speed. Figure 6.4 shows results for WRN-40-2, where we start with a network with no batch normalization and no skip connections, and then progressively add them to the network. For these runs, we present results only with the best tuned initial learning rate since the optimal learning rates are usually very different on batch normalized vs. non-batch normalized networks. We see that adding batch normalization makes a big difference in the convergence speed as well as in lowering gradient confusion. Adding skip connections on top of this further accelerates training, although it seems to have minimal effect on gradient confusion (when used on top of batch normalization).



(a) Loss; Best LR (b) Confusion; Best LR

Figure 6.4: Effect of batch normalization and skip connections on a Wide ResNet

6.5 Conclusion

We study the effect of high dimensionality and over-parameterization on the convergence of SGD, and show that low gradient confusion in high dimensional problems can lead to accelerated convergence. This addresses the issue of why SGD is an effective optimizer for over-parameterized problems. An interesting question for future work is whether there is a connection between gradient confusion and generalization for SGD.

Part II

STUDYING THE EVOLUTION OF CULTURAL NORMS

Chapter 7: **Using game theory to study the evolution of cultural norms**

Understanding human behavior and modeling how cultural norms evolve in different human societies is vital for designing policies and avoiding conflicts around the world. This part of the thesis describes ways to use computational game-theoretic techniques, and in particular evolutionary game theoretic (EGT) models, to gain insight into why different human societies have different norms and behaviors.

Conventional (non-evolutionary) game theory is good for analyzing situations where we know an individuals' preferences, and want to predict what they will do based on those preferences. However in our work, we want to know how these preferences arose. We are interested in the following kinds of questions:

- What kinds of structural and external factors might have led to the emergence of behaviors we see among individuals in a society?
- What evolutionary pressures might have led to variations in those behaviors?
- Can they be validated by observed phenomena?

Conventional game theory can't properly answer these questions. To lay out an individual's preferences in a conventional game-theoretic model would, in essence, be building into the model the very traits whose emergence we want to study. We instead

need to lay out the structural/environmental factors that might be responsible for the evolution of those traits, to see whether those traits would evolve, and evolutionary game theory provides an efficient framework to do just that.

7.1 Evolutionary game theory in biology

Evolutionary game theory (EGT) was first developed as an application of game theory to evolving populations composed of multiple animal species, as a way to model how each species' evolutionary fitness causes its proportion of the population to grow or shrink [SP73]. The idea is to represent an interaction among animals as a normal-form game. The game's payoffs are intended to represent the effect that the interaction will have on the individuals' evolutionary fitness. For example, if two animals fight over a piece of food, one might expect that each individual's fitness would be affected by how the fight affects the animal's health, and whether the animal gets the piece of food.

Rather than developing a detailed model of each specific individual, EGT models typically are at a much more abstract level that does not distinguish among the individuals within each species, but instead looks at the average behavior of all individuals of that species. More specifically:

- If the population is composed of n different species, then for each species i ($i = 1, \ldots n$), all individuals of species i have the same strategy s_i, namely the strategy of being a member of species i. This strategy is intended to encompass—in an abstract way, of course—everything that might affect this species' average evolutionary fitness: size, aggressiveness, sensory abilities, intelligence, etc.

- Each species i constitutes some proportion x_i of the entire population, with $\sum_{i=1}^{n} x_i = 1$. If we choose an individual at random, then for $i = 1, \ldots, n$, the probability that this individual uses strategy s_i is x_i.

Now, consider an interaction (e.g., a conflict over a source of food) between two individuals: one from species i and one from species j. For simplicity of presentation, let's restrict this to just two individuals, but it can easily be generalized to interactions among k individuals for arbitrary k.

To formulate this interaction as a normal-form game, let's say that the individuals' expected payoffs are $u(s_i, s_j)$ and $u(s_j, s_i)$, where "payoff" means the effect that the interaction will have on the individual's evolutionary fitness.

The normal-form game is *symmetric*, i.e., if the two individuals are named a and b, then it doesn't matter whether the one with strategy s_i is individual a or individual b. In either case, this individual's expected payoff is $u(s_i, s_j)$, and the other individual's expected payoff is $u(s_j, s_i)$.

Suppose an individual with strategy s_i meets another individual chosen at random. Then for $j = 1, \ldots, n$, the other individual's strategy is s_j with probability x_j. Hence the expected payoff for the individual with strategy s_i is $\sum_{j=1}^{n} x_j u(s_i, s_j)$. Earlier, we said the expected payoff is intended to represent the interaction's effects on evolutionary fitness. The idea is that species i's expected payoff is higher than that of the entire population, then species i will reproduce at a higher rate, hence its proportion x_i will increase. If species j's expected payoff is lower than that of the entire population, then species j will reproduce at a lower rate, hence x_j will decrease.

The best-known way to model this is the *replicator dynamic* [TJ78]. The original version is a differential equation that assumes an infinite population and continuous time. Let $\pi_i(x) \geq 0$ be the average payoff obtained by individuals of species i when the proportions of each species are $x = (x_1, \ldots, x_n)$. Then the average payoff for the entire population is $\theta(x) = \sum_{i=1}^{n} x_i \pi_i(x)$. According to the replicator dynamic, the rate of change in each x_i is given by the following differential equation:

$$dx_i/dt = x_i(\pi_i(x) - \theta(x)). \tag{7.1}$$

The replicator dynamic is consistent with the Lotka-Volterra equations for the dynamics of biological systems. Indeed, the replicator dynamic is mathematically equivalent to a generalization of those equations [PN02].

The replicator dynamic can be translated into a difference equation in which the population is finite, and time proceeds as a sequence of discrete iterations [HS84]. This formulation can be used to run a discrete-event computer simulations and look at their outcomes—which is useful if the differential equations are too complicated to solve mathematically.

The above approach assumes that the species are *well-mixed*, i.e., that they are uniformly distributed geographically. Such an assumption is often inaccurate; there are many settings in which an individuals' location can make a huge difference in what interactions they have, and how those interactions affect their evolutionary fitness. To model such situations, it often is useful to locate the individuals in a network in which they are restricted to interact with their neighbors. This is further discussed later in this chapter.

7.2 Modeling cultural evolution

EGT can be used to model aspects of the evolution of human cultures. Here, strategies correspond not to collections of individuals, but instead to possible behaviors. A successful strategy—i.e., a behavior that produces good results—is likely to be adopted by others, hence become more prevalent in the population. Conversely, the prevalence of an unsuccessful strategy is likely to decrease. The propagation of these strategies corresponds not to biological reproduction, but instead to *cultural transmission*, in which humans imitate others and learn from others. Rather than the replicator dynamic, here the evolutionary model is a comparison process, e.g., a modified version of the Fermi rule from statistical mechanics [Blu93]. At each iteration t, each individual a uses some strategy in a game-theoretic interaction and receives a payoff u_a. Then, before the beginning of iteration $t+1$, a compares u_a to the payoff u_n received by a randomly chosen neighbor n, and decides whether to keep using the same strategy that it used before, or switch to the neighbor's strategy. The probability of switching is given by a version of the well-known sigmoid function (see Figure 7.1):

$$\Pr[a \text{ switches to } n\text{'s strategy}] = 1/(1 + e^{s(u_a - u_n)}),$$

where u_a and u_n are a's and n's payoffs in the current iteration, and $s \geq 0$ is an arbitrary constant called the *selection strength*. The Fermi rule can easily be adapted to situations in which the population isn't well-mixed. For example, one can locate the individuals at the nodes of a network, restrict each individual a to interact only with its neighbors, and restrict a to compare its payoff with the payoffs of its neighbors.

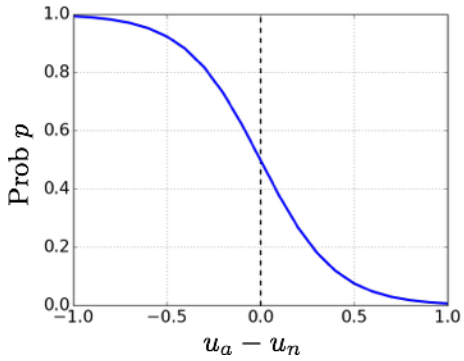


Figure 7.1: Graph of $\frac{1}{1+e^{s(u_a-u_n)}}$, for $s = 5$ and $-1 \leq u_a - u_n \leq 1$.

Usually the Fermi rule is further modified by introducing an *exploration dynamic* that is somewhat analogous to biological mutation. In biological evolution, mutation occurs so rarely that game-theoretic biological models often omit it. In cultural evolution, an analogous phenomenon happens more frequently: individuals to try out new behaviors [THDS+09]. The exploration dynamic models this as follows: when each agent a chooses what strategy to use at the next iteration, there is a small probability μ that a will choose a strategy s at random from the set of all possible strategies, regardless of whether s was a successful strategy for the agents who used it in the current iteration, or whether any agent even used it at all.

One of the limitations of EGT models is that they deliberately omit large amounts of detail. In EGT models of biological evolution, they ignore most of the factors that might influence whether a particular individual will reproduce successfully, and instead consider all individuals of a species to be equivalent. Similarly, EGT models of cultural evolution ignore most of the complexities of human interactions. For example, rather than reasoning about the physical outcomes of an interaction among several individuals, these outcomes are represented by payoff values. Because the models are highly simplified,

they don't give exact numeric predictions of what would happen in real life.

On the other hand, a good EGT model can provide explanations of the underlying dynamics of an evolving system, and establish support for causal relationships. Consequently, such models can provide a useful complement to empirical studies, in which there may be questions whether or not a correlation among various factors indicates a causal relationship [Ald95].

7.3 Contributions

We list the main contributions in this part of the thesis below.

In Chapter 8, we study how norms change in a society. To do this, we build an evolutionary game-theoretic model based on the idea that different strength of norms in societies translate to different game-theoretic interaction structures and incentives. This model is used to study the evolutionary relationships of the need for coordination in a society (which is related to its norm strength) with two key aspects of norm change: cultural inertia (whether or how quickly the population responds when faced with conditions that make a norm change desirable), and exploration rate (the willingness of agents to try out new strategies). Our results show that a high need for coordination leads to both high cultural inertia and a low exploration rate, while a low need for coordination leads to low cultural inertia and high exploration rate.

In Chapter 9, we extend this to study the *rate* at which a norm changes in different cultures. We analyze the evolutionary relationships between the tendency to conform and how quickly a population reacts when conditions make a change in norm desirable. Our

analysis identifies conditions when a tipping point is reached in a population, causing norms to change rapidly. We find that tighter cultures are more likely to be initially resistant to norm changes, but once it reaches a tipping point, they change faster than looser cultures.

In Chapter 10, we study conditions that affect the existence of group-biased behavior among humans (i.e., favoring others from the same group, and being hostile towards others from different groups). Using an evolutionary game-theoretic model, we show that out-group hostility is dramatically reduced by mobility. Technological and societal advances over the past centuries have greatly increased the degree to which humans change physical locations, and our results show that in highly mobile societies, ones choice of action is more likely to depend on what individual one is interacting with, rather than the group to which the individual belongs.

Chapter 8: **Understanding norm change in human societies**

8.1 Introduction

Human societies around the world are unique in their ability to develop, maintain, and enforce social norms. Social norms enable individuals in a society to coordinate actions, and are critical in accomplishing different tasks. Neuroscience, field, and experimental research have all established that there are marked differences in the strength of social norms around the globe [BVL13, EH14, GRN+11, HG14, HEM+10, HMB+06, HTG08, RGNL15]. Some cultures (e.g., some middle-eastern countries, India, South Korea, etc.) are *tight*, in the sense that they tend to have strong social norms, with a high degree of norm-adherence and higher punishment directed towards norm-violators. Other cultures (e.g., Netherlands, New Zealand, Australia, etc.) are *loose*, i.e., individuals tend to develop weaker norms with more tolerance for deviance [GRN+11, HG14, RGNL15]. This indicates that the nature of human interaction and influence is vastly different across different cultures around the world.

To date, there has been little research on the evolutionary processes of norm maintenance and the processes that lead to norm change, and how these processes are substantially different in societies around the world. However, recent world events (e.g., recent social uprisings and turmoil) show that it is critically important to develop such an un-

derstanding. In this section, we draw ideas from recent social science research to build culture-sensitive models that provide insights into the substantial societal differences that exist in how individuals interact and influence each other.

Here, we use EGT to examine the relationships of the amount of *need for coordination* (which psychological and sociological studies show is related to norm strength [RGNL15]), with two key aspects of norm change in societies:

1. the amount of *cultural inertia*, i.e., the amount of resistance to changing a cultural norm, and

2. the *exploration rate*, i.e., the extent to which agents are willing to try out new behaviors.

More specifically, our primary contributions in this work are as follows:

- We provide a novel way to

    1. model a society's strength of norms by using an agent's need for coordination in the society, and

    2. model the desirable/undesirable norms in a society,

    by characterizing how they affect the payoffs in a game-theoretic payoff matrix, leading to different interaction structures and incentives in a society.

- We investigate cultural evolution of norm change in this model using two well-known models of change in evolutionary game theory (the replicator dynamic [TJ78] and the Fermi rule [Blu93]). Using mathematical analyses and extensive agent-based simulations, we establish that: *the higher the need for coordination is, the*

higher the cultural inertia will be, and vice versa. When a population faces conditions that make a norm change desirable, a high need for coordination will make them slower to change to the new norm compared to a society with a lower need for coordination. Further, if the need for coordination is high enough, the existing norm will not change at all.

- In order to understand how norms change in different cultures, we also examine whether the need for coordination in a society has a causal evolutionary relationship to an agent's tendency to learn socially (i.e., adopt a behavior that is being used by other agents in the population) versus innovate/explore new random behaviors. In order to be able to do so, we propose a novel way to model this, where we let the exploration rate, i.e., the probability that an agent tries out a new action at random, *evolve* over time as part of the agent's strategy, rather than stay fixed as in previous work [THDS+09].

- The cultural differences in the distribution of agent strategies favoring social learning versus innovation or exploration can have a critical impact on how attitudes, beliefs and behaviors spread throughout the population, and thus, is vital to understanding norm change. At a societal level, such differences can affect the rate at which new technologies, languages, moral traditions, and political institutions are adopted, while at local levels, they can affect the processes of influence at the individual level. Using the above model of evolving exploration rates, we verify this by establishing, via extensive agent-based simulations, that: *the higher the need for coordination is, the lower the exploration rate will be, and vice versa*.

$M_c =$

	A	B
A	a_c, a_c	$0, 0$
B	$0, 0$	b_c, b_c

$M_f =$

	A	B
A	a_f, a_f	a_f, b_f
B	b_f, a_f	b_f, b_f

Figure 8.1: Individual payoff matrices. M_c denotes the coordination game and M_f denotes the fixed-payoff game used in our model.

These results provide insight into the reasons why tight societies are less open to change, and why cultural inertia and high levels of social learning develop in such societies. To our knowledge, this is the first work to provide a culturally-sensitive model of norm change and to show how the processes of norm propagation differ across societies.

The rest of the chapter is organized as follows. Section 8.2 provides our model of the need for coordination, and mathematical analyses and agent-based simulations showing how it affects cultural inertia. Section 8.3 describes our model of evolving exploration rates, and shows how the degree of need for coordination affects the evolution of exploration rates. In Section 8.4 we discuss the significance of our results.

8.2 Proposed model

Past field and experimental research have shown that tight societies have stronger norms, where individuals adhere to norms much more than loose societies, and face higher punishment when deviating. On the other hand, individuals in loose societies typically have more tolerance for deviant behavior [GRN+11, HG14, RGNL15]. Past EGT studies

$M =$

	A	B
A	$ca_c + (1-c)a_f, ca_c + (1-c)a_f$	$(1-c)a_f, (1-c)b_f$
B	$(1-c)b_f, (1-c)a_f$	$cb_c + (1-c)b_f, cb_c + (1-c)b_f$

Figure 8.2: Weighted payoff matrix M defined as $M = cM_c + (1-c)M_f$.

have shown that a society's exposure to *societal threat* is a key mediating factor in its strength of norms [RGNL15], where threats can be either ecological like natural disasters and scarcity of resources, or manmade such as threats of invasions and conflict. In high-threat situations, societies tend to develop strong norms for coordinating social interaction, (i.e., to become tighter), since coordination is vital for the society's survival. In low-threat situations, there is less need for coordination, which affords weaker norms and looser societies.

Using this intuition, we hypothesize that the interactions between individuals in different societies are governed by *different* payoff structures and incentives. Tight societies tend to have a high need for coordination, and we can model the extreme case as a *coordination game* M_c, where one only gets a payoff if playing the same action as the agent one is interacting with. In loose societies, on the other hand, individuals' payoffs are less affected by others' actions, and we can model the extreme case as a *fixed-payoff* game M_f, in which an agent's payoff depends only on the action played by that agent, and not on the actions of the other agent. For cases in between the two extremes, we use a game in which the payoff matrix is a weighted combination of a coordination game and a fixed-payoff

$M' =$

	A	B
A	a, a	$(1-c)a, (1-c)b$
B	$(1-c)b, (1-c)a$	b, b

Figure 8.3: Updated payoff matrix after assuming $a_c - b_c = a_f - b_c$ and adding a suitable constant to the payoffs in M in Figure 8.2.

game, with the weighting factor $0 \leq c \leq 1$ denoting the need for coordination.

As is done in many EGT studies, we consider games in which individuals have two possible actions to choose from. In our case, the two actions A and B correspond to possible norms that the society could settle on. As shown in Figure 8.1, the coordination game has a payoff matrix M_c in which a_c and b_c are the payoff parameters; and the fixed-payoff game has a payoff matrix M_f in which a_f and b_f are the payoff parameters. The weighted combination of the two games, shown in Figure 8.2, is $M = cM_c + (1-c)M_f$, where $0 \leq c \leq 1$ is the need for coordination.

We first present a lemma that shows that under a mild assumption, the payoff matrix M can be much simplified on adding a constant to all payoffs in the matrix.

Lemma 8.2.1. *Consider the game matrix M defined in Figure 8.2, and assume that $a_c - b_c = a_f - b_f$. Then, under a suitable addition of a constant to the payoffs, and using $a_c = a$ and $b_c = b$, the game matrix M reduces to the matrix M' shown in Figure 8.3.*

Proof. On adding the constant value of $(1-c) * (a_c - a_f) = (1-c) * (b_c - b_f)$ (where equality holds under the assumption) to all payoffs in M, the payoff matrix M reduces to

M', shown in Figure 8.3, where we denote $a_c = a$ and $b_c = b$. ■

The assumption $a_c - b_c = a_f - b_f$ is very reasonable, since this just ensures that switching from one norm to the other always results in the same change in payoffs, regardless of the weight c on the coordination game. Otherwise, there would be an added causal factor for the dynamics of norm change. Also note that, from Lemma 8.2.1, under additions with a constant, this assumption reduces to just setting $a_c = a_f$ and $b_c = b_f$. For the rest of the section, we will work with payoff matrix M' where we set $a_c = a_f = a$ and $b_c = b_f = b$. In subsequent sections, we will show why simplifying the payoff matrix by adding a constant value to all payoffs (as shown in Lemma 8.2.1) is a perfectly reasonable step to take.

From payoff matrix M', we see that whenever $b < a$, the better action for the society to settle on (in terms of payoff) is A, while if $a < b$ then it is B. Let M'_{AB} be the payoff that an agent receives when they play action A and their opponent plays action B. Let M'_{AA}, M'_{BA} and M'_{BB} be defined similarly. Studying the Nash equilibrium of the game M', we get the following lemma.

Lemma 8.2.2. *Consider the game matrix M' defined in Figure 8.3, where all payoff values are positive, i.e., $a, b > 0$. Then we have:*

(i) If $b > a$, the strategy profile (B, B) is a Nash Equilibrium. Further, if $c \geq \frac{b-a}{b}$, then (A, A) is also a Nash equilibrium. Further, the strategy profile $((q, 1-q), (q, 1-q))$ is a Nash Equilibrium only when $c \geq \frac{b-a}{b}$, where $q = \frac{b-(1-c)a}{c(a+b)}$. Note that the mixed strategy $(q, 1-q)$ denotes playing action A with probability q and action B with probability $1-q$.

(ii) Similarly, if $a > b$, the strategy profile (A, A) is a Nash Equilibrium. Further, if $c \geq \frac{a-b}{a}$, then the strategy profile (B, B), as well as $((q, 1-q), (q, 1-q))$ are also Nash Equilibria, with $q = \frac{b-(1-c)a}{c(a+b)}$.

Proof. For (A, A) to be a Nash equilibrium of the game M', as defined in Figure 8.3, the following condition has to hold:

$$M'_{AA} \geq M'_{BA} \quad \Rightarrow \quad c \geq \frac{b-a}{b}. \tag{8.1}$$

Similarly, for (B, B) to be a Nash equilibrium, the required condition is:

$$M'_{BB} \geq M'_{AB} \quad \Rightarrow \quad c \geq \frac{a-b}{a}. \tag{8.2}$$

Consider the following two cases:

1. $b > a$: In this case, (8.2) is always satisfied. Thus, (B, B) is a NE. If c is large enough such that (8.1) is satisfied, then (A, A) is also a NE.

2. $a > b$: In this case, (8.1) is always satisfied. Thus, (A, A) is a NE. If c is large enough such that (8.2) is satisfied, then (B, B) is also a NE.

Note that $((q, 1-q), (q, 1-q))$ is a mixed-strategy Nash Equilibrium when the strategy $(q, 1-q)$ makes the agent indifferent to the opponent's strategy, i.e., when:

$$qM'_{AA} + (1-q)M'_{BA} = qM'_{AB} + (1-q)M'_{BB}. \tag{8.3}$$

Simplifying this, we get:

$$q = \frac{b-(1-c)a}{c(a+b)}.$$

We know that $0 \leq q \leq 1$. Thus, this reduces to the following two conditions for $((q, 1-q), (q, 1-q))$ to be a Nash Equilibrium:

$$c \geq \frac{b-a}{b} \quad \text{and} \quad c \geq \frac{a-b}{a}.$$

When $b > a$, $c \geq \frac{a-b}{a}$ is always satisfied. Thus, when c is large enough such that $c \geq \frac{b-a}{b}$, $((q, 1-q), (q, 1-q))$ is a mixed-strategy Nash Equilibrium. Similarly, when $a > b$, $c \geq \frac{b-a}{b}$ is always satisfied, and when c is large enough such that $c \geq \frac{a-b}{a}$, $((q, 1-q), (q, 1-q))$ is a mixed-strategy Nash Equilibrium. ■

From Lemma 8.2.2, we see that only when c is high enough, the *sub-optimal* action pair becomes a Nash Equilibrium, where *sub-optimal* action pair refers to the situation where both agents get a lower payoff than otherwise possible using, for example, the optimal action pair. This means that when $b > a$, (A, A) is the sub-optimal action pair. Thus, from Lemma 8.2.2, we that see if the need for coordination c is high, then the population may converge to either of two different equilibria, one of which is sub-optimal in terms of overall payoff. When c is low, on the other hand, the society will converge to a single globally-optimal equilibrium.

In the next two sub-sections we introduce two models for studying norm change, using two well-known models of evolutionary change (the replicator dynamic [TJ78] and the Fermi rule [Blu93]). We show that both models of evolutionary change are invariant to additions to the payoffs by a constant, and thus the results from this section carry forward. We derive results for how different societies respond to a need for norm change using both mathematical analysis on infinite *well-mixed* populations (where well-mixed denotes that any agent can interact with any other agent in the population), and extensive agent-based

simulations on finite structured populations (where agents are placed on a network and can interact with only their neighbors).

8.2.1 Replicator dynamic on infinite well-mixed populations

Consider a well-mixed infinite population of agents. This is a standard setting used in evolutionary game theory, since a well-mixed infinite population is usually analytically tractable. Let the agents be interacting with each other using game matrix M' defined in Figure 8.3, and the proportion of agents playing each strategy be denoted by $x = (x_A, x_B)$, i.e., x_A proportion of agents with strategy A, and proportion $x_B = 1 - x_A$ with strategy B. Also, let $u_A(x)$ and $u_B(x)$ denote the payoffs received by an agent playing actions A and B respectively, given the strategy proportion x. The expected payoff for an agent is given by interacting with a randomly chosen agent in the population. Thus, we get the following:

$$\mathbb{E}[u_A(x)] = x_A M'_{AA} + x_B M'_{AB},$$

$$\mathbb{E}[u_B(x)] = x_A M'_{BA} + x_B M'_{BB}.$$

On analyzing the Nash Equilibria of this system, we observe the following lemma.

Lemma 8.2.3. *Consider a well-mixed infinite population where agents interact using the game M' in Figure 8.3. Assuming all payoff values are positive, i.e., $a, b > 0$, and using Lemma 8.2.2, we have:*

(i) When $b > a$, $x_A = 0$ is a Nash Equilibrium. If $c \geq \frac{b-a}{b}$, then $x_A = 1$ and $x_A = \frac{b-(1-c)a}{c(a+b)}$ (which corresponds to the mixed-strategy Nash Equilibrium in Lemma 8.2.2) are also Nash Equilibria.

(ii) Similarly, when $a > b$, $x_A = 1$ is a Nash Equilibrium, while if $c \geq \frac{a-b}{a}$, then $x_A = 0$ and $x_A = \frac{b-(1-c)a}{c(a+b)}$ also are Nash Equilibria.

Proof. Consider the cases: $x_A = 0$ (the strategy set where all of the population plays B) and $x_A = 1$ (the strategy set where all of the population plays A). From Lemma 8.2.2, we get the following two cases:

1. $b > a$: In this case, (8.2) is always satisfied. Thus, $x_A = 0$ is a NE. If c is large enough such that (8.1) is satisfied, then $x_A = 1$ is also a NE.

2. $a > b$: In this case, (8.1) is always satisfied. Thus, $x_A = 0$ is a NE. If c is large enough such that (8.2) is satisfied, then $x_A = 1$ is also a NE.

Now consider the intermediate case where $x_A = p$ with $0 < p < 1$. For $x_A = p$ to be a NE, no A agent should have a strictly better payoff if switching to B, and vice versa. Thus, the following two conditions need to be simultaneously satisfied:

$$pM'_{AA} + (1-p)M'_{AB} \geq pM'_{BA} + (1-p)M'_{BB},$$

$$\text{and} \quad pM'_{BA} + (1-p)M'_{BB} \geq pM'_{AA} + (1-p)M'_{AB}.$$

Both of these conditions are satisfied only when:

$$pM'_{AA} + (1-p)M'_{AB} = pM'_{BA} + (1-p)M'_{BB}.$$

This simplifies to:

$$p = \frac{b-(1-c)a}{c(a+b)},$$

and similar to Lemma 8.2.2, the results follow. ■

We assume that on each iteration, agents interact with other randomly chosen agents, and the population evolves according to the replicator dynamic. The replicator dynamic is based on the idea that the proportion of agents of a type (or strategy) increases when it achieves expected payoff higher than the average payoff, and decreases when achieving lower payoff than the average payoff. Thus, over time, the proportion of agents of a type that achieves payoff higher than the average payoff starts increasing in the population, and eventually take over. More formally, the replicator dynamic is given by the differential equation

$$\dot{x}_A = \frac{dx_A}{dt} = x_A \cdot (\mathbb{E}[u_A(x)] - \theta(x)), \tag{8.4}$$

where $\theta(x) = x_A\mathbb{E}[u_A(x)] + x_B\mathbb{E}[u_B(x)]$ is the average payoff received by all agents in the population. From (8.4), it is clear that the rate of change remains the same on adding a constant to the payoff matrix, since the added constants would just cancel each other out. Thus, Lemma 8.2.1 follows through to this section as well.

Using the game matrix M', the rate of change in the proportion x_A is given by:

$$\dot{x}_A = x_A(1 - x_A)(c(a + b)x_A - (b - (1 - c)a)). \tag{8.5}$$

The fixed points of this rate of change are given by:

$$x_A = 0, \quad x_A = 1, \quad \text{and} \quad x_A = \frac{b - (1 - c)a}{c(a + b)}. \tag{8.6}$$

These correspond to the Nash Equilibria derived earlier. Next, we study the stability of the Nash equilibria derived above, where we define a stable Nash equilibrium under the replicator dynamic to be one where: if an infinitesimal proportion of agents change their strategy, the replicator dynamic always forces the population back to the original Nash

equilibrium. More precisely, let the Nash equilibrium be $x_A = p$. If x_A increases an infinitesimal amount to $p + \epsilon$, the Nash equilibrium is stable only if $\dot{x}_A < 0$, which drives the population back to the Nash equilibrium $x_A = p$. Similarly, if x_A decreases by ϵ, $x_A = p$ is stable only if $\dot{x}_A > 0$. Thus, we state the following corollary.

Corollary 8.2.1. *From Lemma 8.2.3 and Eq.* (8.5) *and Eq.* (8.6)*, we see that the Nash Equilibria* $x_A = 0$ *and* $x_A = 1$ *are stable, while the Nash Equilibrium* $x_A = \frac{b-(1-c)a}{c(a+b)}$ *is unstable.*

Proof. Let $\phi = \frac{b-(1-c)a}{c(a+b)}$. From Eq. (8.5), we notice that, if $x_A = \phi + \epsilon$, then $\dot{x}_A > 0$, while if $x_A = \phi - \epsilon$, then $\dot{x}_A < 0$, for any small $\epsilon > 0$. Thus, $x_A = \frac{b-(1-c)a}{c(a+b)}$ represents an unstable fixed point. Similarly notice that if $x_A = \epsilon$, $\dot{x}_A < 0$, while if $x_A = 1 - \epsilon$, $\dot{x}_A > 0$. Thus, $x_A = 0$ and $x_A = 1$ represent stable fixed points. ■

There is a further notion of equilibrium used in EGT called evolutionarily stable strategies (ESS) [Smi82]. A strategy S is an ESS if there is a small proportion p_y such that, when any other strategy T has a proportion $p_x < p_y$ (where the rest of the population has strategy S), the payoff of an S agent is always strictly greater than a T agent. Using this definition, we state the following theorem.

Theorem 8.2.1. *From Lemma 8.2.3 and Corollary 8.2.1, we see:*

(i) When $b > a$*,* B *is an ESS. If* $c \geq \frac{b-a}{b}$*, then* A *is also an ESS.*

(ii) When $a > b$*,* A *is an ESS. If* $c \geq \frac{a-b}{a}$*, then* B *is also an ESS.*

Proof. Let C denote the mixed strategy $(q, 1-q)$. From (8.3), we see that for C to be a

mixed strategy NE, the following condition needs to hold:

$$q = \frac{M'_{BB} - M'_{AB}}{M'_{AA} - M'_{BA} + M'_{BB} - M'_{AB}}, \tag{8.7}$$

Let M'_{AC} denote the payoff received by the row player when an A player (row player) interacts with a C player (column player). Similarly, we define M'_{CC}, M'_{CA}, M'_{BC} and M'_{CB}. Thus we get:

$$M'_{CC} := q^2 M'_{AA} + q(1-q)M'_{AB} + q(1-q)M'_{BA} + (1-q)^2 M'_{BB},$$

$$M'_{CA} := qM'_{AA} + (1-q)M'_{BA},$$

$$M'_{AC} := qM'_{AA} + (1-q)M'_{AB},$$

$$M'_{CB} := qM'_{AB} + (1-q)M'_{BB},$$

$$M'_{BC} := qM'_{BA} + (1-q)M'_{BB}.$$

Let us derive conditions for which A is an Evolutionarily Stable Strategy (ESS). Thus, we consider the proportion of agents playing A to be close to 1, i.e., $x_A = 1 - \epsilon$, where $0 < \epsilon \ll 1$.

Let S denote the set of strategies other than A that agents can play, i.e., $S \in \{B, C\}$. For A to be an ESS, one of the following conditions need to hold: either

1. $M'_{AA} > M'_{SA}$, or
2. $M'_{AA} = M'_{SA}$ and $M'_{AS} > M'_{SS}$.

From Lemma 8.2.2, we see that $M'_{AA} > M'_{BA}$ simplifies to the condition $c > \frac{b-a}{b}$. Further, $M'_{AA} = M'_{BA}$ simplifies to $c = \frac{b-a}{b}$.

We also notice that $M'_{AA} > M'_{CA}$ simplifies to:

$$M'_{AA} > qM'_{AA} + (1-q)M'_{BA} \quad \Rightarrow \quad M'_{AA} > M'_{BA}.$$

Now consider the three cases:

1. $b > a$: In this case, if c is large enough such that $c > \frac{b-a}{b}$ is satisfied, then A is an ESS.

2. $a > b$: In this case, $c > \frac{b-a}{b}$ is always satisfied. Thus, A is an ESS.

3. $b > a$ and $c = \frac{b-a}{b}$: In this case, for A to be an ESS, both the conditions $M'_{AB} > M'_{BB}$ and $M'_{AC} > M'_{CC}$ have to be satisfied. $M'_{AB} > M'_{BB}$ simplifies to $(1-c)a > b$, which is never satisfied. $M'_{AC} > M'_{CC}$ simplifies to:

$$qM'_{AA} + (1-q)M'_{AB} > q^2M'_{AA} + q(1-q)M'_{AB} + q(1-q)M'_{BA} + (1-q)^2M'_{BB},$$

$$\Rightarrow \quad q(1-q)(M'_{AA} - M'_{BA}) > (1-q)^2(M'_{BB} - M'_{AB}), \tag{8.8}$$

which is also never satisfied (follows from (8.7)). Thus, for this case, A is not an ESS.

We can similarly derive conditions where B is an Evolutionarily Stable Strategy (ESS).

Now we examine whether C is an ESS. Using (8.7), we can show that the following conditions are satisfied: $M'_{CC} = M'_{AC}$ and $M'_{CC} = M'_{BC}$. Thus, for C to be an ESS, both $M'_{CA} > M'_{AA}$ and $M'_{CB} > M'_{BB}$ need to be satisfied. These two conditions simplify to the following conditions: $M'_{BA} > M'_{AA}$ and $M'_{AB} > M'_{BB}$, which in turn simplify to the conditions $c < \frac{b-a}{b}$ and $c < \frac{a-b}{a}$. Both of these conditions cannot be simultaneously satisfied. Thus, C is not an ESS. ■

We observe that the strategies A and B are Evolutionary Stable Strategies (ESS), when adopted by everyone in the population (corresponding to the stable Nash equilibria

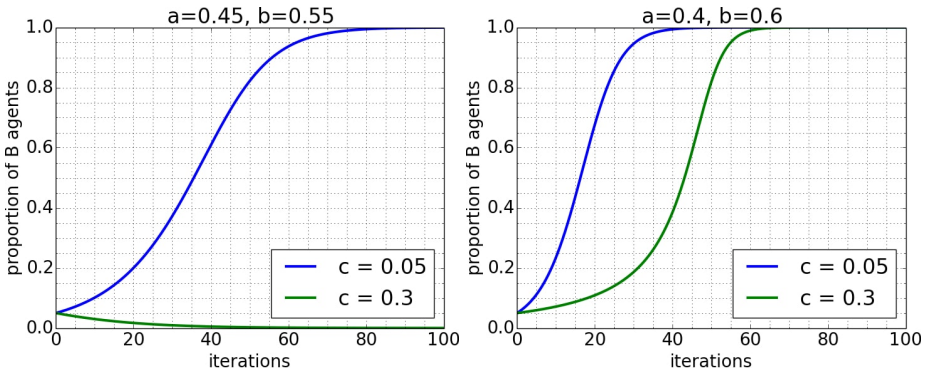


Figure 8.4: Figures show the change in the proportion of B agents with time with a well-mixed infinite population where reproduction is determined by the replicator dynamic with $b > a$.

$x_A = 1$ and $x_A = 0$). The unstable Nash Equilibrium, on the other hand, does not correspond to an ESS, since even a small group with a different strategy is able to force the population to a different equilibrium. Thus, only stable Nash Equilibria correspond to evolutionarily stable strategies.

Theorem 8.2.1 indicates that a society under our model is bound to end up at one of the evolutionarily stable strategies: with every individual on action A or everyone on action B, since even a small perturbation moves the society away from the unstable Nash equilibrium. When c is low, there exists only a single ESS, and thus the society adapts itself and settles on the ESS. When c is high, there are two ESSs, and thus the society might settle on either one, depending on the starting point of the society.

Let us consider two societies: one with a lower need for coordination c_1, and one with a high need for coordination $c_2 > c_1$. To avoid some awkward phrasing, we'll call these the "looser" and "tighter" societies, respectively. Suppose a majority of both societies are playing norm A, and suppose they evolve according to the replicator dynamic

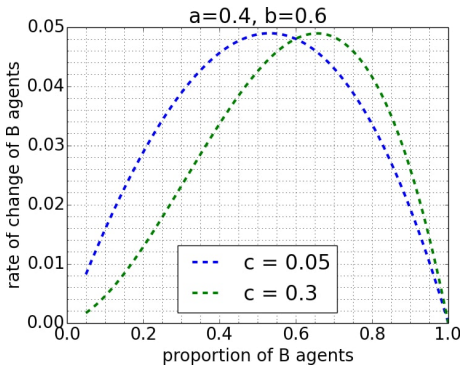


Figure 8.5: Figure shows the rate of change of B agents versus the proportion of B agents, with a well-mixed infinite population where reproduction is determined by the replicator dynamic with $b > a$.

given in Eq. (7.1). We are interested in how these two societies would respond to the action B, when the payoff of action B is higher than A, i.e., when $b > a$, or equivalently, $M'_{BB} > M'_{AA}$. First notice that if $c_2 > (b-a)/b$, and $c_1 < (b-a)/b$, it follows from Theorem 8.2.1 that the tighter society remains on norm A while the looser one switches to the globally optimal norm B.

Now suppose the difference in norm payoffs is large enough such that $c_2 < (b-a)/b$ (and thus also, $c_1 < (b-a)/b$). This ensures that there is only a single equilibrium for both societies at $x_A = 0$. Thus, both societies would eventually switch to norm B, and we are interested in the rate at which this change occurs. Let $\dot{x}_{B1}$ and $\dot{x}_{B2}$ denote the rate of change when the need to conform is c_1 or c_2, respectively. Then we can show that:

$$\dot{x}_{B2} - \dot{x}_{B1} = x_B(1-x_B)(c_2-c_1)((a+b)x_B - b).$$

This simplifies to

$$\dot{x}_{B2} - \dot{x}_{B1} \begin{cases} \leq 0, & \text{when } x_B \leq \frac{b}{a+b}; \\ > 0, & \text{when } x_B > \frac{b}{a+b}. \end{cases} \tag{8.9}$$

Thus, $\dot{x}_{B2} < \dot{x}_{B1}$ in the initial stages when $x_B < b/(a+b)$. However, once the proportion of B agents become big enough such that $x_B > b/(a+b)$, then the higher the value of c, the higher the rate of change will be. Thus, when c is high, the switch from A to B takes time to speed up, with more cultural inertia than when c is low, even when the payoff of the new norm is arbitrarily large compared to the previous norm. The initial cultural inertia results in the society with a higher c value to take longer overall to switch to the new norm.

Figure 8.4 illustrates these properties of well-mixed populations using the replicator dynamic. We start off the society at the proportion $x_A = 0.95$. In the first of the three graphs, the tighter society (again using "tighter" as shorthand for "higher need for coordination") has $c > (b-a)/b$. Thus, while the less-tight society switches to the more beneficial norm B, the tighter society is resistant to the change (since the difference in payoffs is small) and stays with norm A. The second and third graphs show situations where both societies switch to norm B. We observe that the tighter society switches more slowly towards changing to norm B, but the difference in speed decreases as the difference in payoffs between B and A increases.

As derived in Eq. (8.9), the rate of change for a society with higher c grows larger than with lower c only after $x_B > \frac{b}{a+b}$. This is shown in Figure 8.5. This also indicates the initial inertia that societies with a higher need for coordination experience towards

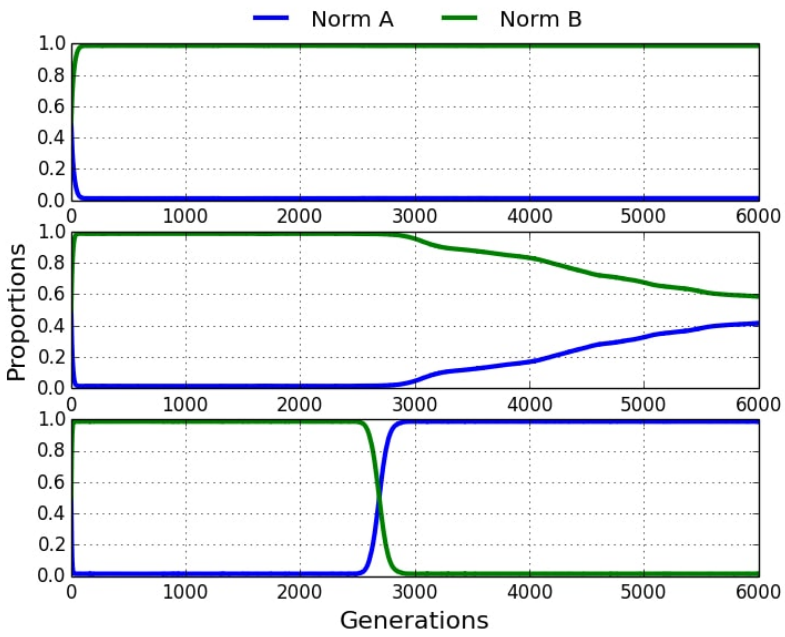


Figure 8.6: Simulations with the Fermi rule on a toroidal grid of size 2500. From top to bottom: $c = 1.0$, $c = 0.75$, $c = 0.5$. Initially: $a = 1.0, b = 1.15$. We use a structural shock at 2500 iterations, after which the payoffs become: $a = 1.15, b = 1.0$.

changing norms. The need for coordination in these societies lead to individuals being reluctant to try out new norms, which in turn leads to inertia.

8.2.2 Agent simulations on finite networks

A limitation of the model introduced in the previous section, is that it assumes that the population is *infinite* and *well-mixed*. While the assumption that a population is infinite is not a bad approximation for very large populations (which is the scale that we are interested in), the assumption that agents are well-mixed, i.e., where any agent can interact with any other agent, is often inaccurate. In this section, we show that the results derived in the previous section, also extend to a model in which agents are structured on

the nodes of a graph/network, where agents can only interact with another agent if they are connected by an edge in the graph.

More specifically, we now consider a structured population where agents are arranged on the nodes of a toroidal (wrap-around) grid, such that each agent can interact only with the 4 other agents they are connected to. We consider toroidal grids as a convenient example, however, the results we describe below also extend to other network structures like small-world networks [WS98], and preferential attachment [BA99] models. Mathematical analysis of evolutionary games on structured populations is not yet a well-developed field, and thus we perform simulations of our model as follows.

Initially, we arrange agents with random strategies (A or B) on each node of the grid. In each iteration, each pair of agents connected by an edge interact in a two-player game defined by the payoff matrix M. The total payoff of each agent is computed by summing over the payoffs received by an agent for each game that they play. Since the population is finite, we use dynamics defined on finite populations. After each interaction phase, agents use the Fermi rule to update it's strategy for the next iteration. Under the Fermi rule, an agent ψ_a picks a random neighbor ψ_n and observes its payoff, and the agent then decides to switch to the neighbor's strategy with probability $p = (1 + \exp(-s(u_a - u_n)))^{-1}$, where u_a and u_n are the payoffs of the agent and the neighbor, and s is a user-defined parameter (in all our experiments, we set $s = 5$). With probability $1 - p$, the agent retains its old strategy. With a small probability μ, called the exploration rate, an agent then tries out an action completely randomly. This repeats for every iteration of the simulation. Note that the Fermi rule also only depends on a difference between payoff values, and thus, like the replicator dynamic, is also invariant to addition of a constant

to the payoffs. Thus, for all our experiments in this section, we use the simplified game matrix M' from Figure 8.3.

To study cultural inertia (i.e., resistance to changing a cultural norm) or rapid cultural change in different societies, we use a game-theoretic model of a *structural shock*. A structural shock represents a catastrophic incident in a society, where suddenly there is abrupt change in the payoffs for actions A and B. We are interested in studying how societies with different needs for coordination react to such an abrupt and drastic shift in the payoffs of the possible actions. In our EGT model, we implement a structural shock by simply interchanging the payoffs of actions A and B, thus, denoting a sudden change in the globally optimal action in a society. This is equivalent to interchanging the payoff values a and b. Thus, if initially, we have $b > a$, after a structural shock, we get $a > b$. Note that in the simulations presented in Figure 8.4 with well-mixed populations, we use a structural shock implicitly by assuming the norm is A and the payoff of action B is higher than action A.

Consider that, initially, the action with a higher utility (and the current norm) in a society is B, i.e., $b > a$ with $x_A = 0$. Suppose, the society experiences a structural shock, where now action A becomes more desirable with $a > b$. On introducing a small proportion of agents playing norm A (say $x_A = 0.01$), if the need for coordination is low then the population will switch to the new norm with $x_A = 1$. This is because, after the structural shock, the Nash Equilibrium (and ESS) is $x_A = 1$, as shown above. However, if the need for coordination is high (i.e., $c \geq \frac{a-b}{a}$), then $x_A = 0$ is still a Nash Equilibrium (and ESS) and the population will remain on the sub-optimal norm B even after the structural shock.

All experiments were run on a grid with 2500 nodes, and the simulation goes on for 6000 iterations, with a structural shock implemented at 2500 iterations. 100 independent simulations are run for each setting and the results are averaged over the 100 runs. Figure 8.6 shows the results of our simulations. The plots show the proportion of agents playing norm A vs norm B. As before, the parameter c denotes the need for coordination. When c is low, very little cultural inertia develops and agents are more willing to innovate by exploring behaviors other than the current societal norms. In this case, the population will change more quickly to a different norm if the new norm will be beneficial. By contrast, when c is high, we see the evolutionary emergence of higher levels of cultural inertia, with agents less willing to innovate or to violate established cultural norms. In this case, the population is slower to change to the new norm, and if c is high enough it may not change at all. Thus, qualitatively, the results with a structured populations match those from the infinite well-mixed populations in Section 8.2.1, and the mechanics that lead to the above results can be explained using the same equilibrium results derived above.

8.3 Evolving exploration rates

In addition to the amount of cultural inertia, another key aspect to understanding how norms change in different cultures is to study whether an agent is evolutionarily more likely to learn socially (i.e., adopt a behavior that is being used by other agents in the population) or to innovate and explore new random behaviors. Such tendencies are critical in understanding the rate at which new technologies, languages, or moral traditions are adopted in a population, and help us understand the processes of influence

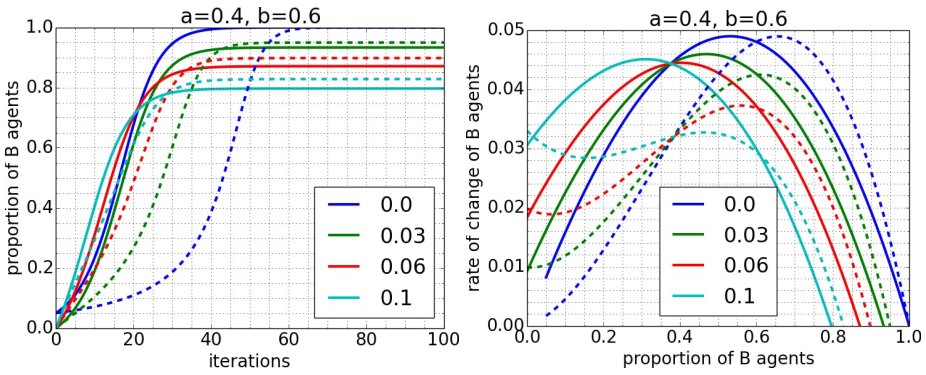
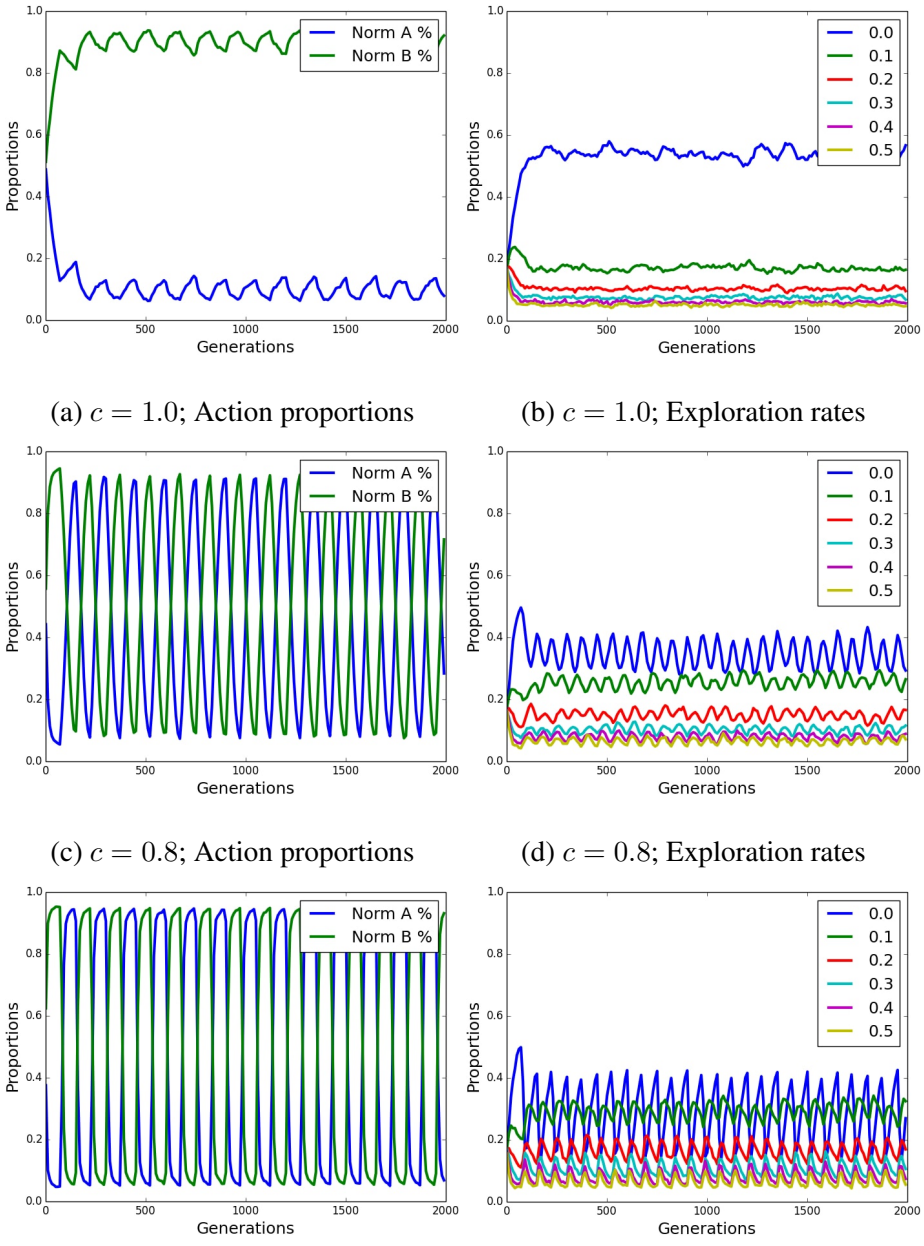


Figure 8.7: Replicator-mutator dynamic on an infinite well-mixed population with $a = 0.4$ and $b = 0.6$. The solid and dotted lines denote $c = 0.05$ and $c = 0.3$, respectively. The colors denote the exploration rates.

and persuasion at the individual level.

In the model presented in Section 8.2.2 for finite structured populations, the exploration rate (i.e., the small probability with which an agent tries out a new strategy at random) was kept at a constant low value. This exploration rate denotes how much an agent is open to change and trying out new actions at random. Thus, it seems that the need for coordination in a society might affect how likely an individual is to try out different actions, instead of conforming to their neighbors. Particularly, it seems natural to assume that individuals in tight societies are much less likely to try out random actions than individuals in loose societies [GRN+11, HG14]. In this section, we test this hypothesis by presenting a model to study the evolution of exploration rates in different societies.

To get some intuition about the hypothesis, we go back to our setting of a well-mixed infinite population. Note that the replicator dynamic does not have a provision for exploration rates. Thus, we use a variant of the replicator dynamic called the replicator-mutator equation [Lev03]. Using this variant, one can include exploration rates into



(a) $c = 1.0$; Action proportions

(b) $c = 1.0$; Exploration rates

(c) $c = 0.8$; Action proportions

(d) $c = 0.8$; Exploration rates

(e) $c = 0.5$; Action proportions

(f) $c = 0.5$; Exploration rates

Figure 8.8: Simulations with the Fermi rule on a toroidal grid of size 2500, with structural shocks at intervals of 75 iterations. From left to right: $c = 1.0$, $c = 0.8$, $c = 0.5$. Initially: $a = 1.0, b = 1.15$. The left column shows proportions of norms A and B. The right column shows proportions of the population that use each different exploration rate.

the replicator dynamic. Thus, if we fix μ to be the exploration rate, we can write the replicator-mutator equation as:

$$\begin{aligned}\dot{x}_A &= (1-\mu)x_A\mathbb{E}[u_A(x)] + \mu x_A\mathbb{E}[u_B(x)] - x_A\theta(x),\\ &= x_A(\mathbb{E}[u_A(x)] - \theta(x)) + \mu x_A(\mathbb{E}[u_B(x)] - \mathbb{E}[u_A(x)]).\end{aligned}$$

Thus, like the replicator dynamic, we can write the rate of change in terms of payoff differences, which makes the dynamic invariant to additions to the payoffs by a constant. Thus, we again use the simplified game matrix M' from Figure 8.3. Simplifying the equation for $\dot{x}_A$, we get:

$$\begin{aligned}\dot{x}_A =& x_A(1-x_A)(c(a+b)x_A - (b-(1-c)a))\\ &+ \mu(x_A x_B(1-c)(b-a) + (x_B^2 b - x_A^2 a)).\end{aligned} \tag{8.10}$$

Figure 8.7 plots the replicator-mutator equation (Eq. (8.10)) with a well-mixed infinite population. The solid lines are for a low need for coordination ($c = 0.05$), while the dotted lines are for a high need for coordination ($c = 0.3$), and we plot the proportion of B agents, as well as the rate of change, for various exploration rate μ values. From the figure, we see that for all exploration rates μ, when the need for coordination is high then there is higher cultural inertia.

To study how an agent's tendency to learn socially or explore develops in a culture, we let the exploration rate (referred to as the mutation rate in biological models) *evolve*. The exploration rate is the probability μ with which an agent chooses a random new strategy at each iteration ($0 \leq \mu \ll 1$). In biological evolution, mutation occurs so rarely that game-theoretic biological models often omit it. In cultural evolution, however, exploration is an important step since individuals try out new behaviors much more

frequently [THDS+09]. Studying the evolution of exploration rates helps us get insights about a society's openness to change. Low exploration rates suggest that individuals are less likely to try out new strategies and are more likely to coordinate with their neighbors. On the other hand, high exploration rates mean that individuals are more open to change and innovation.

To model the evolution of exploration rates, we first create a set L of possible exploration rates. These can be a finite discrete set of exploration rates. For all our experiments, we use the set of exploration rates: $L = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. The exploration rate is added as part of the strategy of an agent, and each individual now chooses an exploration rate in addition to the game action (A or B). Thus, an agent now copies the exploration rate of a neighbor, along with the game action, when updating its strategy using the Fermi rule.

Note that a regularly changing environment is essential for studying the evolution of exploration rates since, if the environment is not changing frequently enough, an exploration rate of 0 would always be evolutionarily stable. To model the changing environment, we will use the same switch in dominant norms (structural shock) that we used in our earlier experiments, except now we apply the structural shock multiple times at much shorter and regular intervals. We use a fixed interval of 75 iterations to apply the structural shock. We run the simulation for a total of 2000 iterations. For these experiments, an agent's strategy set now becomes a size of 10: 5 possible exploration rates in $L \times 2$ possible game actions (norm A or norm B). We use the same toroidal grid as described before. Figure 8.8 shows the experimental results. Each column in Figure 8.8 shows, for a specific c, the proportion of agents playing norm A vs norm B (top plot), and the

proportion of agents with each exploration rate (bottom plot).

We see that when the need for coordination is high, low exploration rates are adopted by the majority of the society. Individuals in such a society are more likely to adopt the strategies of their neighbors, and this leads to high cultural inertia. In loose societies, however, higher proportions of exploration rates $\mu > 0$ evolve, and individuals are more open towards change, leading to lower cultural inertia. This fits well with our results in Section 8.2, and provides insights into why cultural inertia develops in societies with a higher need for coordination.

8.4 Significance of the work

In this chapter, we examined the processes underlying cultural inertia and norm change. We build evolutionary game-theoretic models that show that societies that have a higher need for coordination – those that are tight – have higher cultural inertia, with individuals being less likely to switch to the new norm even when it might have a larger payoff. Societies with a lower need for coordination – those that are loose – on the other hand, have low cultural inertia, with individuals more willing to innovate and open to change.

By letting the exploration rate evolve, we used it to study an agent's tendency to either learn using social interaction or innovate and explore new random behaviors, and we show that exploration rates evolve differently in different cultures. When the need for coordination is high, the majority of the population has very low exploration rates, and individuals are more likely to adopt the strategies of their neighbors. When the need

for coordination is low, higher exploration rates evolve, leading to lower cultural inertia, and more openness to change. This explains why tight cultures tend to have less deviant behavior among individuals with more norm adherence.

To our knowledge, this is the first work that predicts the effects of the need for coordination on norm change and cultural inertia, and how it affects an agent's decision of whether to learn from others or to innovate and explore random behaviors. We found our main qualitative findings to be robust to a wide range of parameter values, in both our simulation and theoretical results.

By studying how socio-structural factors such as the need for coordination affect cultural inertia, this work aims to establish a culturally-sensitive model of norm change. With this model, we identify the conditions that lead to stability or instability in established population norms in different cultural contexts. Such knowledge is critical in providing us the ability to identify early markers of impending drastic shifts in populations' norms and thus enable tools providing alerts to potential social uprisings and turmoil.

Chapter 9: **Tipping points for norm change in human cultures**

9.1 Introduction

Tightness-looseness is a dynamic construct, yet to date, there has been little research on the evolutionary processes that lead to *changes* in societal norms, the *rate* at which such changes occurs, and how these processes *vary* across different cultures. In this chapter, we aim to study how cultural differences in the way humans interact and influence each other heavily influence how societal norms are established and the rate at which they change across the world. We examine the causal relationship between an individual's tendency to conform with those around them and the rate at which norms are changed in different cultures. More specifically, our primary contributions in this chapter are as follows:

- Drawing on recent research in cultural psychology, we propose a game-theoretic model of a culture based on the tendency of an individual to conform with others, vs. being more individualistic in their behavior.

- Using this model, we provide conditions under which a population is open to changing the current norm in a society.

- Finally, we analyze the *rate* at which such norm changes occur and compare the rates in different cultures. We find that tighter cultures are more likely to be initially

resistant to norm changes, but once it reaches a *tipping point*, they change faster than looser cultures.

9.2 Background and related work

There has been widespread interest in studying the *emergence* of social norms in a population both from an evolutionary perspective [You01,HO01,Mer38,Bic05,HYOR14], as well as in empirical research [JKV10, KJTW09, CB15]. There has been, however, much less work done on understanding the processes that lead to *change* in an already established norm in a population. A related concept, the propagation of information in social networks, has been well-studied (see [CLC13, Jac10, EK10] for an overview), but these works typically do not account for the differences in how individuals interact and influence each other in different cultures. Data science approaches have also explored this question, however, it is very challenging to separate out the various confounding factors (such as institutional influence) to establish clear causal relationships [ZZX15, LGRC12, KYC+12, LMK+13].

This chapter extends the results of Chapter 8. While we studied the processes of norm change in Chapter 8, there were limitations in the model that made it difficult to analyze the speed of norm change. In the next section, we describe our new model, which is more amenable to mathematical analysis, and provides a clearer picture into the factors that affect the speed of norm change in different cultures.

9.3 Proposed evolutionary game-theoretic model

Consider an infinite, well-mixed population (i.e., each individual can interact with any other individual in the population) that evolves according to the well-known replicator dynamic [HS03]. For simplicity of presentation, suppose each agent may choose one of two possible actions: A and B (see Section 9.4 for a discussion on the assumptions used in our model). The two actions A and B correspond to possible norms that the society could settle on. Let x_A and x_B denote the proportions of the population using actions A and B respectively, with $0 \leq x_A, x_B \leq 1$ and $x_A + x_B = 1$, and let $x = (x_A, x_B)$. According to the replicator dynamic, the rate of change in the proportions of agents using each action is given by (7.1): $\dot{x}_i = x_i[f_i(x) - \phi(x)]$, where $i \in \{A, B\}$, $\dot{x}_i = dx_i/dt$ (i.e., rate of change of x_i), $f_i(x)$ is the fitness of action i, and $\phi(x)$ denotes the average fitness of the population, i.e.: $\phi(x) = x_A f_A(x) + x_B f_B(x)$. The replicator dynamic is based on the idea that the proportion of agents with a particular strategy increases when it achieves expected fitness higher than the average fitness, and vice versa.

Let u_A and u_B denote the payoffs associated with actions A and B, where $0 < u_A, u_B < 1$ and $u_A + u_B = 1$. To define the fitness function f_i, we use the key insight that in loose cultures, individuals tend to choose the action that is most beneficial to them; but in tight cultures, individuals tend to conform to the same action that others use, even if a different action might be more beneficial to each individual. To model this mathematically, we let f_i be a weighted combination of the payoff u_i and an additional *conformism fitness* measure θ_i that depends on whether the individual is conforming to others in the population. Let m denote the parameter controlling the weighting between

these two fitness measures, i.e., the amount of conformist transmission in a population. Thus, we define f_i as:

$$f_i(x, m) = (1 - m)u_i + m\theta_i(x, k), \tag{9.1}$$

where $0 \leq m \leq 1$, and we define the conformism fitness measure θ_i as:

$$\theta_i(x, k) = \Big[1 + \exp\big(-k(x - 0.5)\big)\Big]^{-1}, \tag{9.2}$$

where $k > 0$. Note that we can vary the behavior of the conformism fitness measure θ_i using the parameter k (see Figure 9.1). For example, when k is large, θ_i is close to a step function where an agent has a non-zero conformism fitness only if they conform with the majority action:

$$\theta_i^{\infty}(x) = \lim_{k\to\infty} \theta_i(x, k) = \begin{cases} 0, & \text{if } x_i < 0.5; \\ 0.5, & \text{if } x_i = 0.5; \\ 1, & \text{if } x_i > 0.5. \end{cases}$$

Note that with no conformism whatsoever ($m = 0$), each action's fitness depends solely on its payoff, i.e., typical of a very loose culture. On the other hand, with 100% conformist transmission ($m = 1$), i's fitness depends solely on the conformism fitness measure (for the case of θ_i^{∞}, this means that i's fitness depends solely on whether i is in the majority or the minority of the population). This is more indicative of a very tight culture. For simplicity, for the rest of the chapter, we denote $\theta_i(x, k)$ as θ_i and $\theta_i^{\infty}(x)$ as θ_i^{∞}.

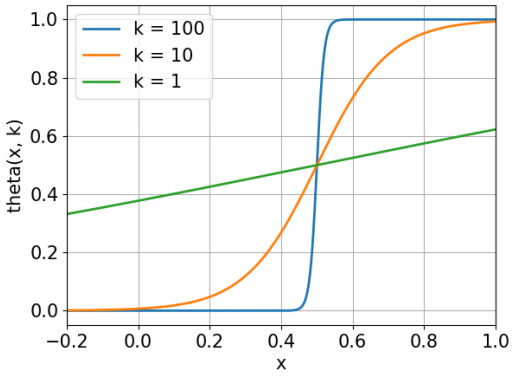


Figure 9.1: Plot of (9.2) for different values of k.

9.3.1 When does norm change occur?

Suppose norm B has a higher utility compared to A, i.e., $u_B > u_A$. We are interested in analyzing the conditions for which a population shifts from norm A to B (norm change). We can re-write the average fitness to be:

$$\phi(x) = (1-m)(x_A u_A + x_B u_B) + m(x_A \theta_A + x_B \theta_B). \tag{9.3}$$

We are interested in anlayzing the rate of change in the proportion of B individuals. From (7.1), (9.1), and (9.3), we get:

$$\dot{x}_B = x_B(1-x_B)\big[(1-m)(u_B - u_A) + m(\theta_B - \theta_A)\big]. \tag{9.4}$$

Note that $\theta_B \geq \theta_A$ when $x_B \geq 0.5$. Since $u_B > u_A$, we see that $\dot{x}_B > 0$, i.e., x_B will converge to 1 ($\lim_{t\to\infty} x_B = 1$) when $x_B \geq 0.5$. If $x_A > x_B$, i.e., if the current norm in the population is A, norm change takes place only if:

$$m < \frac{u_B - u_A}{(u_B - u_A) + (\theta_A - \theta_B)}. \tag{9.5}$$

Thus, norm change takes place only if the population is loose enough, while tighter cultures are more resistant to change. Further, note that $\theta_A^\infty - \theta_B^\infty = 1$ when $x_A > x_B$.

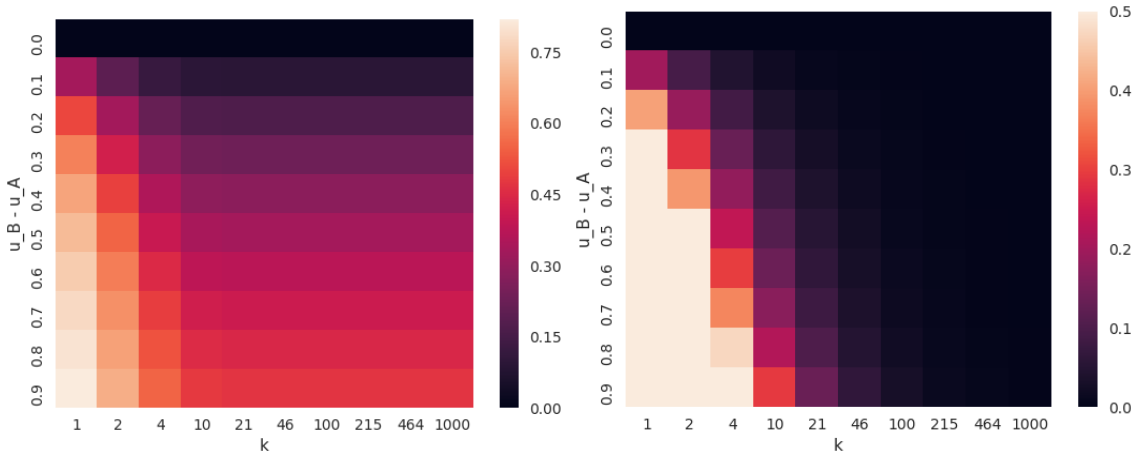


Figure 9.2: *Left:* Heatmap of the right-hand side in (9.5) when $x_B = 0.1$, for various $u_B - u_A$ and k values. *Right:* Heatmap of the right-hand side in (9.7), for various $u_B - u_A$ and k values. Best viewed in color.

Thus, when the conformist fitness measure is a step-function θ_i^∞, (9.5) becomes: $m < (u_B - u_A)/(u_B - u_A + 1) < 0.5$. Thus, for θ_i^∞, norm change occurs only in loose cultures where individuals weigh their individual payoff more than whether they conform to others in the population. Figure 9.2 (left) shows a heatmap of how condition (9.5) varies with $u_B - u_A$ and k when $x_B = 0.1$. We see that the bound on m increases as $u_B - u_A$ increases, i.e., a population becomes more likely to switch the norm. On increasing k, we see that the bound on m decreases. This makes intuitive sense, since a higher k makes the difference in fitness between A and B clearer. Thus, in tight cultures, where people tend to agree more on what behaviors are appropriate vs. inappropriate in different situations [GRN+11], a higher k would lead to more resistance to norm change.

9.3.2 Rate of norm change in tight vs. loose cultures

We are now interested in studying the speed with which norms change in different populations. Consider two possible values of m, namely m_1 and m_2, with $m_2 > m_1$ (i.e., m_2 is a more conformist culture than m_1). Let the corresponding values of $\dot{x}_B$ be denoted by $\dot{x}_B^1$ and $\dot{x}_B^2$, respectively. Assume further that both m_1 and m_2 satisfy (9.5), i.e., norm change takes place in both cultures. Analyzing the difference in the rates of change, from (9.4) we get:

$$\dot{x}_B^2 - \dot{x}_B^1 = x_B(1 - x_B)(m_2 - m_1)\big[(\theta_B - \theta_A) - (u_B - u_A)\big]. \tag{9.6}$$

Note that when $x_B \leq 0.5$, $\theta_B - \theta_A \leq 0$, which would mean: $\dot{x}_B^2 - \dot{x}_B^1 \leq 0$, i.e., the more conformist culture would be slow to change initially. To analyze the case when $x_B > 0.5$, let's assume $x_B = 0.5 + \epsilon$, for $\epsilon > 0$. Thus, $x_A = 0.5 - \epsilon$. From (9.6), we see that for $\dot{x}_B^2 - \dot{x}_B^1 > 0$, the following condition needs to hold:

$$\epsilon > \big[\ln(1 + u_B - u_A) - \ln(1 - (u_B - u_A))\big]/k. \tag{9.7}$$

Figure 9.2 (right) plots a heatmap of how this bound varies with $u_B - u_A$ and k. We see that as k increases, the point at which the more conformist culture starts changing faster moves closer to the point $x_B = 0.5$. Note that when $k \to \infty$, (9.7) reduces to $\epsilon > 0$, i.e., as soon as x_B becomes a majority, greater conformism would produce a larger rate of change.

We are also interested in studying how the rate of change $\dot{x}_B$ varies with x_B as a population switches to norm B, and how this relates to different levels of conformism. We first look at the *maximum* rate of change $\dot{x}_B^{\max} = \max_{x_B} \dot{x}_B$ (we numerically calculate

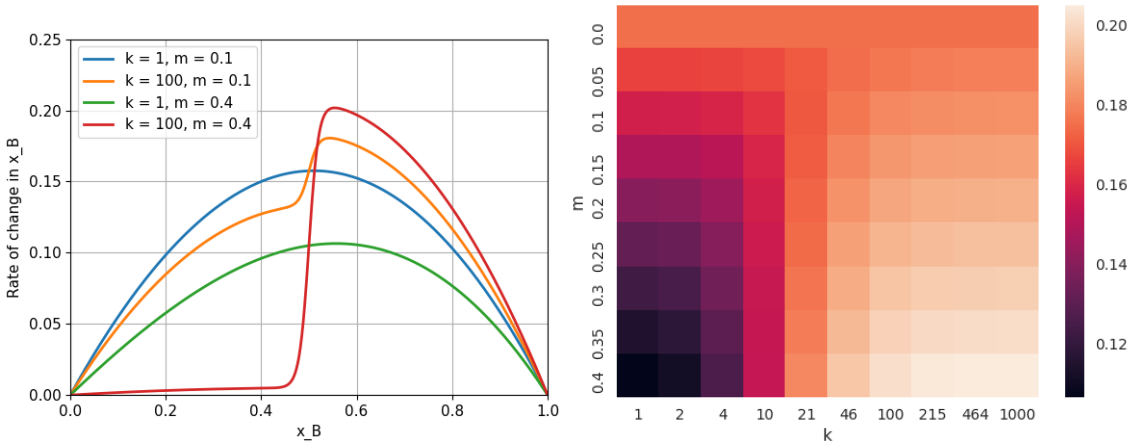


Figure 9.3: *Left:* Plot of (9.4) at $u_B - u_A = 0.7$. *Right:* Heatmap of $\max_{x_B} \dot{x}_B$ for various k and m values, with $u_B - u_A = 0.7$. Best viewed in color.

this, given values for k, m and $u_B - u_A$). Figure 9.3 (right) plots $\dot{x}_B^{\max}$ for different m and k values, where we set $u_B - u_A = 0.7$. These values were chosen such that the norm changes from A to B for all the considered combinations (using the bounds from Section 9.3.1). We see that when k is low, lower conformism leads to higher $\dot{x}_B^{\max}$. However, as k increases (i.e., as θ_B approaches θ_B^{∞}), there is a clear transition, where more conformist cultures end up having a higher maximum rate of change.

This effect is clearer in the left plot of Figure 9.3, where we show how $\dot{x}_B$ varies with x_B. We see that when k is low, i.e., when there is no clear difference between A and B in it's conformist fitness measure θ, $\dot{x}_B$ changes slowly for both tight and loose cultures, with the loose culture having a higher rate of change. With high k, however, we see that the tighter culture faces a "tipping point", which results in a sudden increase in $\dot{x}_B$ with the tighter culture adopting a higher rate of change $\dot{x}_B$ than the loose culture. Thus, in a more conformist culture with high k, initially peer pressure impedes the switch to the more beneficial norm B. But once enough of the population has switched, a tipping

point is reached where peer pressure causes the rest of them to switch very rapidly.

9.4 Discussion

In this chapter, we show that tight cultures sometimes experience a tipping point for norm change while loose cultures typically face a more gradual change. The results presented assume an infinite well-mixed population with two possible actions. We believe it would be relatively straightforward to extend our results for multiple actions. Assuming an infinite well-mixed population made our model mathematically tractable and to provide exact conditions for norm change. As future work, it would be interesting to extend this model to the finite population case, where interactions between individuals are dictated by a social network.

Chapter 10: **On the evolution of ethnocentrism in human cultures**

10.1 Introduction

Nearly all major conflicts across the globe, both current and historical, are characterized by individuals defining themselves and others in terms of their group membership. Substantial empirical evidence supports people's tendency to favor in-group members and show hostility towards out-group individuals [Taj82, BK85, HRW02, BFF06, CL09]. From an evolutionary perspective, numerous studies have shown how in populations comprised of various groups, group-biased behavior that discriminates or is hostile against out-groups evolves or emerges readily and dominantly [HA06, CB07, AOW+09, GvdB11, FTC+12, MNVV12, HKS13]. Since humans are social beings who establish and define groups constantly, the development of out-group hostility and resulting group conflict might thus seem inevitable.

In contrast, however, statistics have shown that violence and outgroup conflict have actually declined dramatically over the past few centuries of human civilization, suggesting out-group hostility is not inevitable after all [Pin11a, Pin11b]. What factors might lead to such a decrease in conflict? Evolutionary game-theoretic models can shed light on this question by exploring how various factors affect the emergence and maintenance of individuals' behaviors relating to group conflict.

	Cooperate	Defect
Cooperate	$b-c, b-c$	$-c, b$
Defect	$b, -c$	$0, 0$

Figure 10.1: Prisoner's Dilemma payoff matrix used in our model.

Our evolutionary game model builds on a prior model developed in Hammond and Axelrod's pioneering work [HA06] on the evolution of ethnocentrism, and used in Hartshorn, et al. [HKS13]. In their model, agents had perceivable group tags, played one-shot Prisoner's Dilemma games with their neighbors, and could behave differently toward in-group members than out-group members. Each agent's inherited traits included a group tag, an action (cooperate or defect) to use with in-group members, and a similar action to use with out-group members. Thus there were four possible strategies: Cooperate with both in-group and out-group members; Defect against both in-group and out-group members; Ethnocentric (cooperate with in-group members, defect against out-group members); Traitorous (defect against in-group members, cooperate with out-group members).

Using their model with four different groups (or group tags), we have replicated their result showing that after a period in which Cooperative agents are briefly abundant, evolutionary pressure leads to a predominance of Ethnocentric agents. Defectors and Traitors never establish themselves.

Since the agents in that model conditioned their actions only on the group tags, they

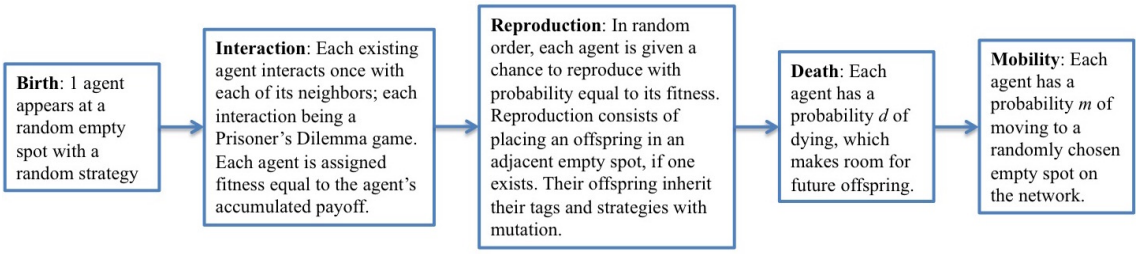


Figure 10.2: Sequence of events at each time step in our evolutionary game-theoretic model. The sequence of steps are the same as in Hammond and Axelrod's paper [HA06] except for the Mobility stage, which is new. For additional details, see the Methods section.

were in effect group-entitative. That leaves open the question whether there are conditions under which individual-entitative agents – agents that base their actions on knowledge of individuals *per se* rather than group tags – may be able to exist and perhaps even be favored by evolutionary pressures.

Moreover, that model does not incorporate mobility. Research in cultural psychology has demonstrated large empirical differences in residential mobility around the globe with important psychological consequences [Lon91, Ang00]. Researchers have shown that in high-mobility contexts, individuals change relationships often; they form new relationships and sever unwanted relationships with great ease [OKM+13, OSYA15]. In such contexts, having a broad network of weak ties and being open toward strangers (with whom it might be valuable to form relationships) is highly adaptive. Indeed, Oishi, et al. [OSYA15] observe that in highly mobile contexts, "since it is hard to keep track of behaviors of many strangers whom one meets, one needs to carefully avoid being associated with defectors or free-riders in order to exploit the greatest possible relational

benefit" (p. 228). Thus, individuals are more likely to adopt strategies that try to evaluate the "trustworthiness and worth" [OSYA15] of others in highly mobile contexts, i.e., adopt individual-entitative strategies. On the other hand, in low-mobility contexts, individuals have far fewer opportunities to form new relationships, and severing existing relationships can have extreme adverse effects such as being ostracized from one's only social circle [OSYA15], causing "the existential, social, and psychological death of the individual" (p. 755) [Lan92]. Based on these theories we would predict that group-entitative behavior and associative ethnocentrism is adaptive in low mobility societies, yet it is maladaptive in high-mobility contexts, where individual-entitative strategies would be evolutionarily favored.

We have run extensive new evolutionary simulations, augmenting the prior model to include individual-entitative strategies and mobility; and our results show that the evolution of ethnocentrism is driven by low mobility. Indeed, our subsequent empirical analysis of archival data verifies that contexts with high residential mobility have less out-group hostility than those with low mobility.

In our evolutionary game model, agents are arranged on a toroidal (wrap-around) grid, so that every node on the grid is connected to 4 neighboring nodes). Initially the grid is empty. The sequence of events at each time step is shown in Figure 10.2; these are the same as in Hammond and Axelrod's paper [HA06] except for the Mobility stage, which is new. For additional details, see the Methods section.

The agents' strategies are similar to those in Hammond and Axelrod's model [HA06], where agents can distinguish between in-group and out-group members by observing the group tags. Hence agents' strategies can be conditioned on whether they are interacting

with in-group or out-group members. In addition, in our model, each agent's strategies can be conditioned on the past history of other agents. Each agent can either be group-entitative or individual-entitative, and this is an inherited trait.

A group-entitative agent i ignores individual identities. Its actions toward an agent j depend only on its last encounter with anyone in j's group. It has two possibly different strategies: one for in-groups and another for out-groups. Each of those strategies is one of the following: AllC (always cooperate), AllD (always defect), TFT (Tit-for-Tat: play whatever action the opponent played in i's last interaction with anyone from j's group), or OTFT (play the opposite of what TFT would play). Note that this is unlike the models in previous chapters where agents had no memory of prior interactions.

An individual-entitative agent i ignores other agents' group tags; i's action toward j depends only on its last encounter specifically with j. Thus i has one of the above four strategies, except that TFT and OTFT depend on i's last interaction with j specifically, rather than someone in j's group.

To model mobility, there is a probability m with which, at the beginning of each iteration, an agent moves to a randomly chosen empty spot in the network. Thus a high value of m represents a highly mobile population, while a low value of m represents a population with low mobility. We vary m from 0 to 0.08 in our experiments. It is important to note that a mobility probability of 0.08 is quite high: it means that on average, 8% of the population move to different locations on each iteration – a substantial amount of movement even for small values of m. At higher levels of mobility ($m > 0.1$), cooperation breaks down in a society, and the majority of the population starts defecting – and thus is not representative of any stable society around the world.

10.2 Results

Figure 10.3 shows our results after letting the populations evolve for 30,000 iterations. Without mobility (i.e., $m = 0$), group-entitative agents comprise 75% of the population. These agents' strategies are predominantly out-group hostile (AllD) and in-group cooperative (AllC). This is reasonably consistent with Hammond and Axelrod's model [HA06], but notice that even when $m = 0$, individual-entitative agents comprise about 25% of the population.

As mobility increases, the evolutionary pressures shift to favor individual-entitative agents. For $m > 0.02$ they comprise about 80% of the population, and about 70% of them play TFT. Thus, the evolutionary dominance of group-entitative and ethnocentric strategies is thwarted by mobility.

The reason why low mobility favors group-entitative strategies while higher mobility favors individual-entitative strategies is related to the clustering of group members (Figure 10.3). With low mobility, groups tend to cluster together heavily; hence agents interact primarily with in-group members. Thus the ethnocentric strategy (i.e., group-entitativity with in-group cooperation and out-group-hostility) is effective and profitable in terms of payoff. Under higher mobility, however, agents are less clustered by group membership, hence more likely to interact with out-group members, hence cannot rely on high payoffs from in-group interactions. Furthermore, group-entitative strategies are less effective because different group members are much less likely to have the same strategy. This favors the individual-entitative Tit-for-Tat (TFT) strategy (see the Methods section for more information on the clustering coefficient).

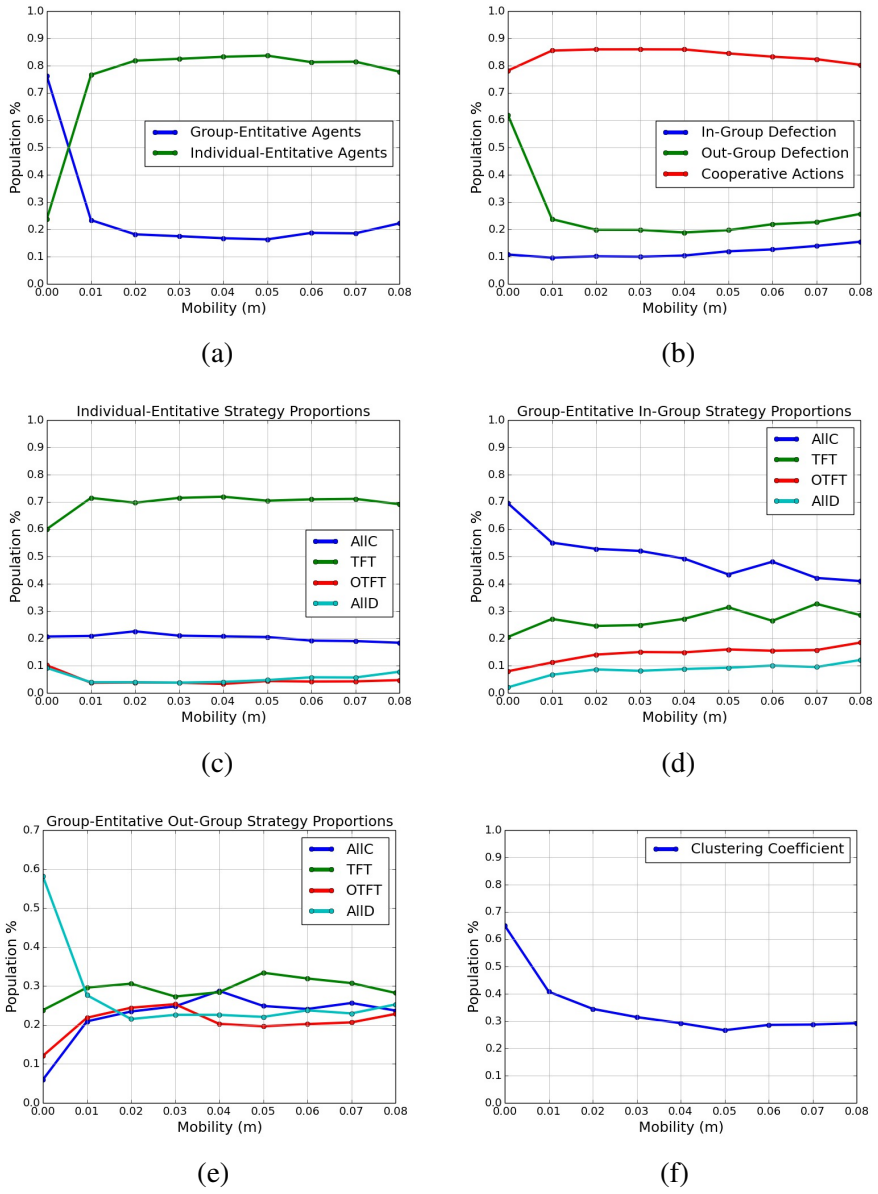


Figure 10.3: Proportions of actions and strategies as a function of mobility, after 30,000 iterations, averaged over 100 simulation runs. The plots show the proportions of (a) the group-entitative and individual-entitative agents, (b) the actions played by the agents, (c) the strategies of the individual-entitative agents, (d) the in-group and (e) out-group strategies of the group-entitative agents, (f) the degree of clustering on the grid.

To illustrate the evolutionary trajectories that led to the reported results, Figures 10.4 and 10.5 show representative evolutionary trajectories for single simulation runs. In Figure 10.4, there is no mobility. Group-entitative agents quickly become a majority, and most of them are ethnocentric (in-group cooperative and out-group hostile). In Figure 10.5, the mobility probability is $m = 0.05$. Individual-entitative agents evolve to become a majority, with most of those agents playing Tit-for-Tat (TFT).

To illustrate robustness of the results with our model, we also performed a series of experiments where we initialized the population on the grid to have high clustering of group-entitative or individual-entitative agents (instead of starting out with an empty grid). In each case, we notice that the results are the same as when we start out with an empty grid, i.e., group-entitative agents dominate under no mobility and individual-entitative agents dominate under higher values of mobility. This is due to the exploration dynamics (or mutation phase) in our model. The exploration dynamic has been shown to be a key aspect of evolutionary game-theoretic models for cultural evolution [TSS+10, ATTN12], and this ensures that our model remains robust to the initial conditions of the grid.

10.2.1 Empirical analysis

In order to complement these modeling efforts, we also gathered data to test the notion that mobility relates to lower ethnocentrism. We analyzed data from the U.S. Census Bureau [Ren11, Wor98] that provides measures of mobility in the U.S. 50 states (defined as the percentage of people born in the state of residence; reverse scored, with higher

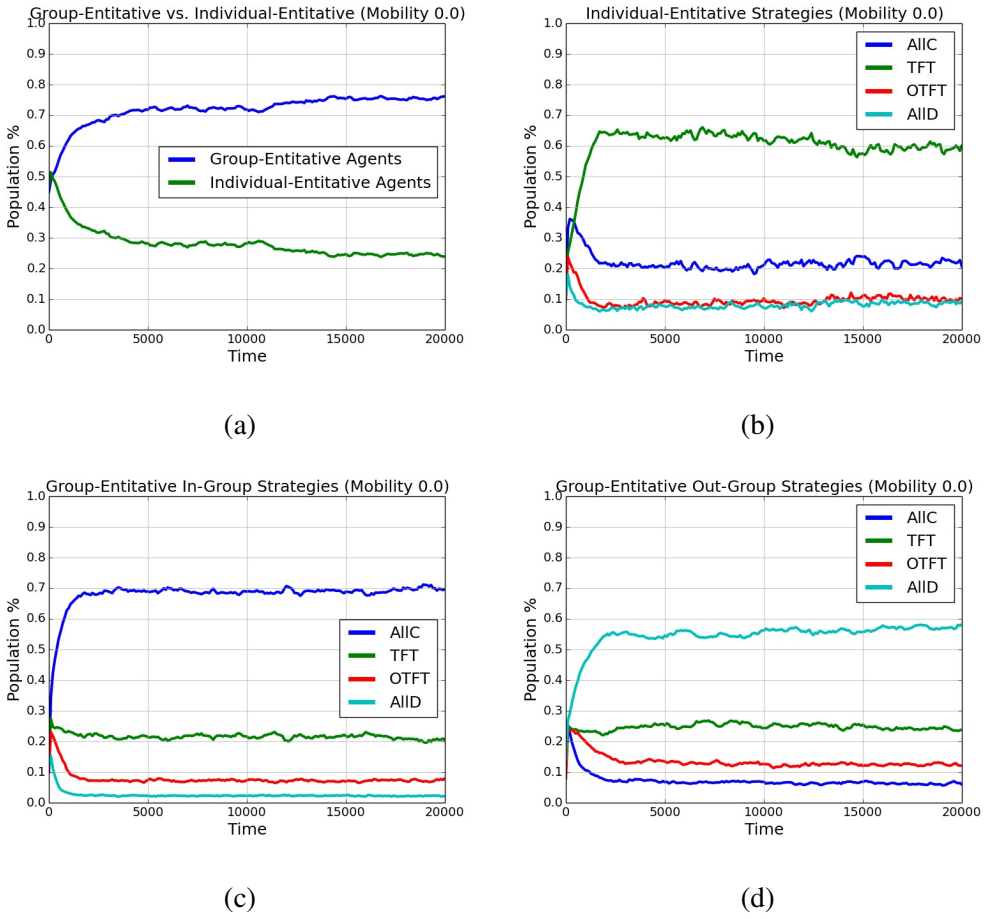


Figure 10.4: Single simulation run for 20000 generations with no mobility ($m = 0$). (a) Proportions of group-entitative and individual-entitative agents. (b) Relative proportions of the individual-entitative agents' strategies; Relative proportions of the group-entitative agents' (c) in-group and (d) out-group strategies.

scores being reflective of higher mobility) and data from the DDB Needham Life Style Survey [Wor98]. We found that mobility was positively correlated with responses to the question "I am interested in the cultures of other countries" ($r = 0.614$, $p < .001$), and negatively correlated with responses to questions regarding ethnocentrism (e.g., Americans should always buy American products, $r = -0.654$, $p < .001$; The government

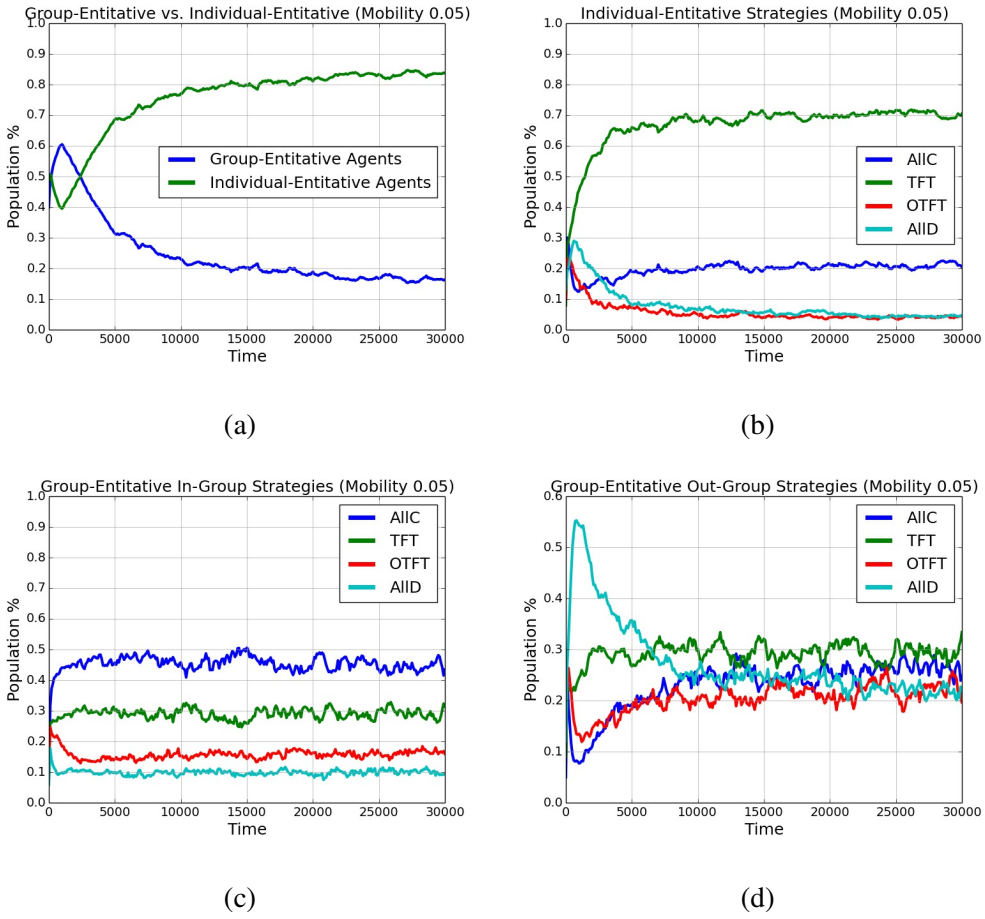


Figure 10.5: Single simulation run for 30000 generations with no mobility ($m = 0.05$). (a) Proportions of group-entitative and individual-entitative agents. (b) Relative proportions of the individual-entitative agents' strategies; Relative proportions of the group-entitative agents' (c) in-group and (d) out-group strategies.

should restrict imported projects, $r = -0.578$, $p < .001$).

In addition, states that have higher mobility also have higher openness, one of the big five personality dimensions, which is associated with breadth of experience and interest and interest in new ideas and other cultures ($r = 0.321$, $p = .023$) [JNS08].

10.3 Significance of the work

The evolution of cooperation has been of great scientific interest in many disciplines; and to date, many evolutionary and empirical studies have found that ingroup-favoring and outgroup hostile behaviors are common. This has caused much concern that group conflict and ethnocentrism is an inevitable threat on our planet. We integrate research on group conflict with human mobility [Now06, OLS07, SYHT09, YSS09, Ois10, SYM10, WEMG11], and show for the first time that the evolution of ethnocentrism and group entitative behavior is thwarted by high mobility. As mobility is rapidly changing around the globe [OSYA15], this work predicts that group conflict will continue to decrease, in line with Pinker's historical analysis [Pin11a, Pin11b].

Mobility is an important and well-studied topic in cultural psychology [OLS07, SYHT09, YSS09, Ois10, SYM10]. Low mobility leads to conditions where interacting individuals are likely to be reproductively related and this has been shown to be important in the evolution of cooperation [Now06, WEMG11]. In our model, we find mobility plays a crucial role in the evolution of ethnocentrism in a society. More specifically, we establish that low mobility leads to in-group cooperation and out-group hostility. High mobility, on the other hand, leads to more individual-entitative behavior, where agents take actions based on the specific individuals they interact with, and not based on the group that those individuals belong to.

Another unique aspect of our model is that we allow for agents to have memory of previous actions played by other agents and the possibility of individual-entitative agents, where agents take actions based on the individual they are playing against rather than

their tag. In a society with high mobility, agents would be moving to different parts of the grid, which leads to low clustering of agents belonging to the same group. Thus, agents with group-entitative strategies suffer, which leads to the evolution of individual-entitative strategies, with strategies like Tit-for-Tat gaining prominence. Under low mobility, on the other hand, agents of the same group cluster together much more, and simple group-entitative strategies like in-group cooperative and out-group hostility gain prominence.

It would be fruitful to incorporate mobility into other evolutionary game-theoretic models of conflict in future research. Moreover, since mobility in a society could be motivated by multiple factors that could have divergent effects, it would be good for future models of mobility and ethnocentrism to incorporate models of these motivations. Mobility might reduce ethnocentrism when agents move for economic reasons, but mobility might not reduce ethnocentrism if agents move primarily to be among other in-group members. In all, our work shows for the first time that mobility is a critical factor that affects the dynamics of conflict with important implications for theory and policy.

10.4 Methods

10.4.1 Evolutionary dynamics of our model

Here is a more detailed version of the sequence of steps in our evolutionary game-theoretic model:

1. Birth: One agent with a random strategy appears at a random empty site, if such a site exists.

2. Base Payoff: Each existing agent receives the base payoff from the environment; we use a base payoff of 0.12 throughout our experiments.

3. Interaction Payoff: Each agent plays a game with each of its neighboring agents on the grid, receiving payoffs according to the game definition. The game played by the agents is the canonical 2-player cooperation/defection dilemma (Figure 6), with the benefit of cooperation $b = 0.03$ and the cost of cooperation $c = 0.01$. In this phase, the action chosen by an agent in each game depends on the type of agents playing that game (i.e., group-entitative or individual-entitative agent) as well as the type of strategy being used by that agent.

4. Fitness: Each agent is assigned fitness equal to the agent's accumulated payoff.

5. Reproduction: In random order, each agent is given a chance to reproduce with probability equal to its fitness. If an agent gets a chance to reproduce, it places an offspring in a randomly chosen empty site in its neighborhood, if such a site exists. The offspring has the same traits as its parent, with a mutation rate of $\mu = 0.005$ per trait.

6. Death: Each agent has a probability $d = 0.1$ of dying. If an agent dies, it is removed from the grid.

7. Mobility: Each agent has a probability of m of moving to a randomly chosen empty spot on the grid.

10.4.2 Clustering coefficient

A clustering coefficient is a metric for measuring the amount of clustering of nodes in a graph. We can measure the clustering of group tags by comparing the group tags of the agents at neighboring locations. For each location (x, y) on the torus grid we consider four triples, each consisting of (x, y) and a pair of adjacent neighboring locations:

1. location (x, y), the neighbor above it, and the neighbor to its left;
2. location (x, y), the neighbor above it, and the neighbor to its right;
3. location (x, y), the neighbor below it, and the neighbor to its left;
4. location (x, y), the neighbor below it, and the neighbor to its right.

Our clustering coefficient is the total number of triples that contain three agents with the same group tag, divided by the total number of triples in the grid. For a torus grid of size $N \times M$, the denominator in our metric, i.e., the number of total triplets, is simply $4NM$. The clustering coefficient lies in the range from 0 to 1, and is higher when agents of the same tag cluster together on the grid, while being small when there is less clustering of agents of the same tag.

10.4.3 Strategy set

When an individual-entitative agent i interacts with an agent j that i has never encountered before, or when a group-entitative agent i interacts with an agent from a group that i has never encountered before, i must choose whether to cooperate or defect.

That choice is part of i's strategy; and in our experiments we allowed both possibilities. This doubled the total number of strategies in our simulations but made no meaningful difference in the results – so in favor of simplicity and clarity, we did not discuss this detail earlier.

10.4.4 Mutation rate

In Hammond and Axelrod's model [HA06], during reproduction, an offspring will have the same strategy s as its parent, except that for each trait in s, there is a small probability $\mu = 0.05$ that this trait will be changed to a randomly chosen one. Notice that μ is not the probability that the offspring's strategy will differ from s. Instead, for each trait in s, it is the probability that this trait will be changed; and this happens independently for each of the traits in s. Consequently, the probability that an offspring retains the exact same strategy as its parent is inversely proportional to the number of possible traits.

Our model has a higher number of different possible traits than Hammond and Axelrod's model. Thus, in order to maintain roughly the same probability that an offspring will retain the same strategy as its parent, we needed to use a smaller value for μ.

In Hammond and Axelrod [HA06], each agent has 3 traits: the group tag and the actions to take when playing against an in-group and an out-group agent. However, in our model, the number of traits is significantly higher. Each group-entitative agent has 7 traits: the group tag, and traits specifying what action to take in each of the following six situations:

1. when an in-group agent cooperated on the last meeting,

2. when an in-group agent defected on the last meeting,

3. the first time one meets an in-group agent,

4. when an out-group agent cooperated on the last meeting,

5. when an out-group agent defected on the last meeting,

6. the first time one meets an out-group agent.

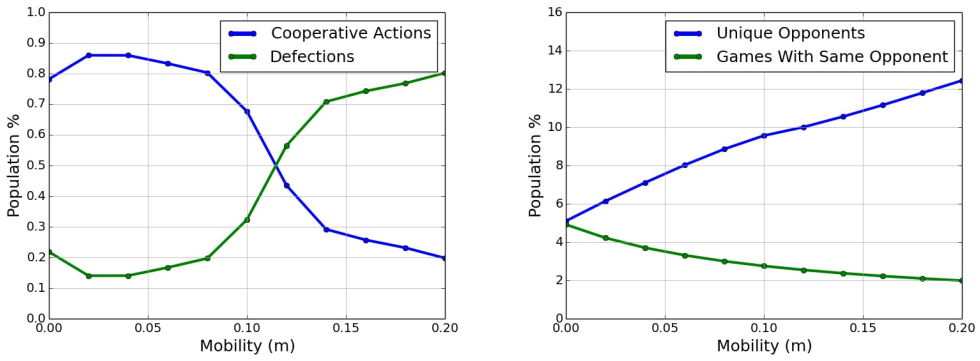
Similarly, each individual-entitative agent in our model has 4 traits. Thus, in order to ensure that the probability of an offspring retaining the same strategy as its parent is similar to that in Hammond an Axelrod's model, we needed to use a lower value for μ than what they used. We used $\mu = 0.005$.

10.4.5 Range of mobility

In our experiments, the reason we limited the mobility probability to $0 \leq m \leq 0.8$ is that cooperation breaks down at higher levels of mobility, as shown in Figure 10.6. For example, at $m = 0.2$, about 80% of all actions are defections. Figure 10.6 also shows the reason for this breakdown. As m increases, the average number of games that an agent plays with the same opponent decreases monotonically. For example, at $m = 0.2$, each agent plays only 2 games with each opponent on average – which favors AllD rather than Tit-for-Tat (TFT).

As a societal analogy, agents are more likely to defect against each other if no agent interacts with any other agent long enough to create any kind of interpersonal ties. For example, consider the limiting case where mobility probability $m = 1.0$, i.e., every

agent moves to a different location in the grid at every iteration. This condition is similar to a well-mixed population, in which every agent can interact with every other agent in the population, with no network structure defining the set of possible interactions. Under well-mixed populations, it is well established that in a cooperation game like the Prisoner's Dilemma, the society devolves into universal Defection.



(a) (b)

Figure 10.6: Cooperation breaking down at higher mobility values. Each data point is an average of 100 individual simulation runs. The plots show (a) the proportion of agents cooperating and defecting; and (b) over an agent's lifetime, the average number of unique opponents it encounters, and the average number of games played against each of them.

Bibliography

[ACH18] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. *arXiv preprint arXiv:1802.06509*, 2018.

[AD11] Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881, 2011.

[AHS15] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Fixed point optimization of deep convolutional neural networks for object recognition. In *ICASSP*. IEEE, 2015.

[Ald95] John Aldrich. Correlations genuine and spurious in Pearson and Yule. *Statistical Science*, pages 364–376, 1995.

[Ang00] Shlomo Angel. *Housing policy matters: A global analysis*. Oxford University Press, 2000.

[AOW+09] Tibor Antal, Hisashi Ohtsuki, John Wakeley, Peter D Taylor, and Martin A Nowak. Evolution of cooperation by phenotypic similarity. *Proceedings of the National Academy of Sciences*, 106(21):8597–8600, 2009.

[ATTN12] Benjamin Allen, Arne Traulsen, Corina E Tarnita, and Martin A Nowak. How mutation affects evolutionary games on graphs. *Journal of theoretical biology*, 299:97–105, 2012.

[BA99] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[BB88] Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[BCNW12] Richard H Byrd, Gillian M Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical programming*, 134(1):127–155, 2012.

[BD99] Jock A Blackard and Denis J Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999.

[Ben12] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[Ber11] Dimitri P Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning*, 2010(1-38):3, 2011.

[BFF06] Helen Bernhard, Urs Fischbacher, and Ernst Fehr. Parochial altruism in humans. *Nature*, 442(7105):912–915, 2006.

[BFL+17] David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? *arXiv preprint arXiv:1702.08591*, 2017.

[Bic05] Cristina Bicchieri. *The grammar of society: The nature and dynamics of social norms*. Cambridge University Press, 2005.

[BIL+15] Carlo Baldassi, Alessandro Ingrosso, Carlo Lucibello, Luca Saglietti, and Riccardo Zecchina. Subdominant dense clusters allow for simple learning and high computational performance in neural networks with discrete synapses. *Physical review letters*, 115(12):128101, 2015.

[BK85] Marilynn B Brewer and Roderick M Kramer. The psychology of intergroup attitudes and behavior. *Annual review of psychology*, 36(1):219–243, 1985.

[Blu93] Lawrence E Blume. The statistical mechanics of strategic interaction. *Games and Economic Behavior*, 5(3):387–424, 1993.

[BMEWL11] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[Bot09] Léon Bottou. Curiously fast convergence of some stochastic gradient descent algorithms. In *Proceedings of the symposium on learning and data science, Paris*, 2009.

[Bot12] L Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

[BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[BSW14] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5, 2014.

[BT89] Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.

[BTPG15] Guillaume Bouchard, Théo Trouillon, Julien Perez, and Adrien Gaidon. Accelerating stochastic gradient descent via online learning to sample. *arXiv preprint arXiv:1506.09016*, 2015.

[BVL13] Daniel Balliet and Paul AM Van Lange. Trust, punishment, and cooperation across 18 societies a meta-analysis. *Perspectives on Psychological Science*, 8(4):363–379, 2013.

[CB07] Jung-Kyoo Choi and Samuel Bowles. The coevolution of parochial altruism and war. *science*, 318(5850):636–640, 2007.

[CB15] Damon Centola and Andrea Baronchelli. The spontaneous emergence of conventions: An experimental study of cultural evolution. *Proceedings of the National Academy of Sciences*, 112(7):1989–1994, 2015.

[CBD15] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, 2015.

[CHS+16] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016.

[CKF11] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[CL09] Yan Chen and Sherry Xin Li. Group identity and social preferences. *The American Economic Review*, 99(1):431–457, 2009.

[CLC13] Wei Chen, Laks VS Lakshmanan, and Carlos Castillo. Information and influence propagation in social networks. *Synthesis Lectures on Data Management*, 5(4):1–177, 2013.

[CR16] Dominik Csiba and Peter Richtárik. Importance sampling for minibatches. *arXiv preprint arXiv:1602.02283*, 2016.

[CSML15] Zhiyong Cheng, Daniel Soudry, Zexi Mao, and Zhenzhong Lan. Training binary multilayer neural networks for image classification using expectation backpropagation. *arXiv preprint arXiv:1503.03562*, 2015.

[DBLJ14] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

[DCM+12] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012.

[DD+14] Aaron Defazio, Justin Domke, et al. Finito: A faster, permutable incremental gradient method for big data problems. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1125–1133, 2014.

[DG16] Soham De and Tom Goldstein. Efficient distributed SGD with variance reduction. In *2016 IEEE International Conference on Data Mining*. IEEE, 2016.

[DYJG16] Soham De, Abhay Yadav, David Jacobs, and Tom Goldstein. Big Batch SGD: Automated inference using adaptive batch sizes. *arXiv preprint arXiv:1610.05792*, 2016.

[EH14] Jean Ensminger and Joseph Henrich. *Experimenting with social norms: Fairness and punishment in cross-cultural perspective*. Russell Sage Foundation, 2014.

[EK10] David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

[FS12] Michael P Friedlander and Mark Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.

[FTC+12] Feng Fu, Corina E Tarnita, Nicholas A Christakis, Long Wang, David G Rand, and Martin A Nowak. Evolution of in-group favoritism. *Scientific reports*, 2:460, 2012.

[GAGN15] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *ICML*, 2015.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[GDG+17] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[GOP15] Mert Gürbüzbalaban, Asu Ozdaglar, and Pablo Parrilo. Why random reshuffling beats stochastic gradient descent. *arXiv preprint arXiv:1510.08560*, 2015.

[GRN+11] Michele J Gelfand, Jana L Raver, Lisa Nishii, Lisa M Leslie, Janetta Lun, Beng Chong Lim, Lili Duan, Assaf Almaliach, Soon Ang, Jakobina Arnadottir, et al. Differences between tight and loose cultures: A 33-nation study. *Science*, 332(6033):1100–1104, 2011.

[GS16] Tom Goldstein and Christoph Studer. Phasemax: Convex phase retrieval via basis pursuit. *arXiv preprint arXiv:1610.07531*, 2016.

[GSB14] Tom Goldstein, Christoph Studer, and Richard Baraniuk. A field guide to forward-backward splitting with a FASTA implementation. *arXiv eprint*, abs/1411.3406, 2014.

[GvdB11] Julián García and Jeroen CJM van den Bergh. Evolution of parochial altruism by multilevel selection. *Evolution and Human Behavior*, 32(4):277–287, 2011.

[HA06] Ross A Hammond and Robert Axelrod. The evolution of ethnocentrism. *Journal of Conflict Resolution*, 50(6):926–936, 2006.

[HAV+15] Reza Harikandeh, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, Jakub Konečnỳ, and Scott Sallinen. Stop wasting my gradients: Practical svrg. In *Advances in Neural Information Processing Systems*, pages 2242–2250, 2015.

[HCS+16] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*, 2016.

[HEM+10] Joseph Henrich, Jean Ensminger, Richard McElreath, Abigail Barr, Clark Barrett, Alexander Bolyanatz, Juan Camilo Cardenas, Michael Gurven, Edwins Gwako, Natalie Henrich, et al. Markets, religion, community size, and the evolution of fairness and punishment. *Science*, 327(5972):1480–1484, 2010.

[HF92] Markus Höhfeld and Scott E Fahlman. Probabilistic rounding in neural network learning with limited precision. *Neurocomputing*, 4(6):291–299, 1992.

[HG14] Jesse R Harrington and Michele J Gelfand. Tightness–looseness across the 50 united states. *Proceedings of the National Academy of Sciences*, 111(22):7990–7995, 2014.

[HKS13] Max Hartshorn, Artem Kaznatcheev, and Thomas Shultz. The evolutionary dominance of ethnocentric cooperation. *Journal of Artificial Societies and Social Simulation*, 16(3):7, 2013.

[HMB+06] Joseph Henrich, Richard McElreath, Abigail Barr, Jean Ensminger, Clark Barrett, Alexander Bolyanatz, Juan Camilo Cardenas, Michael Gurven, Edwins Gwako, Natalie Henrich, et al. Costly punishment across human societies. *Science*, 312(5781):1767–1770, 2006.

[HO01] Michael Hechter and Karl-Dieter Opp. *Social norms*. Russell Sage Foundation, 2001.

[HRW02] Miles Hewstone, Mark Rubin, and Hazel Willis. Intergroup bias. *Annual review of psychology*, 53(1):575–604, 2002.

[HS84] Josef Hofbauer and Karl Sigmund. *Evolutionstheorie und dynamische Systeme*. Parey, 1984.

[HS03] Josef Hofbauer and Karl Sigmund. Evolutionary game dynamics. *Bulletin of the American Mathematical Society*, 40(4):479–519, 2003.

[HS14] Kyuyeon Hwang and Wonyong Sung. Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1. In *IEEE Workshop on Signal Processing Systems (SiPS)*, 2014.

[HTG08] Benedikt Herrmann, Christian Thöni, and Simon Gächter. Antisocial punishment across societies. *Science*, 319(5868):1362–1367, 2008.

[HYOR14] Dirk Helbing, Wenjian Yu, Karl-Dieter Opp, and Heiko Rauhut. Conditions for the emergence of shared norms in populations with incompatible preferences. *PloS one*, 9(8):e104207, 2014.

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[IS15a] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2015.

[IS15b] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[Jac10] Matthew O Jackson. *Social and economic networks*. Princeton university press, 2010.

[JKV10] Stephen Judd, Michael Kearns, and Yevgeniy Vorobeychik. Behavioral dynamics and influence in networked coloring and consensus. *Proceedings of the National Academy of Sciences*, 107(34):14978–14982, 2010.

[JNS08] Oliver P John, Laura P Naumann, and Christopher J Soto. Paradigm shift to the integrative big five trait taxonomy. *Handbook of personality: Theory and research*, 3:114–158, 2008.

[JZ13] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[KB14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KB15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[KH09] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

[KJTW09] Michael Kearns, Stephen Judd, Jinsong Tan, and Jennifer Wortman. Behavioral experiments on biased voting in networks. *Proceedings of the National Academy of Sciences*, 106(5):1347–1352, 2009.

[KLRT14] Jakub Konečnỳ, Jie Liu, Peter Richtárik, and Martin Takáč. ms2gd: Mini-batch semi-stochastic gradient descent in the proximal setting. *arXiv preprint arXiv:1410.4744*, 2014.

[KMN+16] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[KNS16] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

[KR13] Jakub Konečnỳ and Peter Richtárik. Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*, 2013.

[KS15] Minje Kim and Paris Smaragdis. Bitwise neural networks. In *ICML Workshop on Resource-Efficient Machine Learning*, 2015.

[KS17] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[KYC+12] Farshad Kooti, Haeryun Yang, Meeyoung Cha, P Krishna Gummadi, and Winter A Mason. The emergence of conventions in online social networks. In *ICWSM*, 2012.

[Lan92] Hope Landrine. Clinical implications of cultural differences: The referential versus the indexical self. *Clinical Psychology Review*, 12(4):401–415, 1992.

[LASY14] Mu Li, David G Andersen, Alex J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pages 19–27, 2014.

[Lax07] P.D. Lax. *Linear Algebra and Its Applications*. Number v. 10 in Linear algebra and its applications. Wiley, 2007.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LCMB16] Zhouhan Lin, Matthieu Courbariaux, Roland Memisevic, and Yoshua Bengio. Neural networks with few multiplications. *ICLR*, 2016.

[Lev03] Simon Levin. Complex adaptive systems: exploring the known, the unknown and the unknowable. *Bulletin of the American Mathematical Society*, 40(1):3–19, 2003.

[LGRC12] Janette Lehmann, Bruno Gonçalves, José J Ramasco, and Ciro Cattuto. Dynamical classes of collective attention in twitter. In *Proceedings of the 21st international conference on World Wide Web*, pages 251–260. ACM, 2012.

[LHLL15] Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 2719–2727, 2015.

[LMK+13] Yu-Ru Lin, Drew Margolin, Brian Keegan, Andrea Baronchelli, and David Lazer. # bigbirds never die: Understanding social dynamics of emergent hashtag. *arXiv preprint arXiv:1303.7144*, 2013.

[LNS12] Guanghui Lan, Arkadi Nemirovski, and Alexander Shapiro. Validation analysis of mirror descent stochastic approximation method. *Mathematical programming*, 134(2):425–458, 2012.

[Lon91] Larry Long. Residential mobility differences among developed countries. *International Regional Science Review*, 14(2):133–147, 1991.

[LPW09] David Asher Levin, Yuval Peres, and Elizabeth Lee Wilmer. *Markov chains and mixing times*. American Mathematical Soc., 2009.

[LTA16] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *ICML*, 2016.

[LZL16] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[MB11] Eric Moulines and Francis R Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459, 2011.

[MBB17] Siyuan Ma, Raef Bassily, and Mikhail Belkin. The power of interpolation: Understanding the effectiveness of sgd in modern over-parametrized learning. *arXiv preprint arXiv:1712.06559*, 2017.

[Mer38] Robert K Merton. Science and the social order. *Philosophy of Science*, 5(3):321–337, 1938.

[MG15] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pages 2408–2417, 2015.

[MH15] Maren Mahsereci and Philipp Hennig. Probabilistic line searches for stochastic optimization. In *Advances In Neural Information Processing Systems*, pages 181–189, 2015.

[MLM16] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

[MNVV12] Melissa M McDonald, Carlos David Navarrete, and Mark Van Vugt. Evolution and the psychology of intergroup conflict: The male warrior hypothesis. *Phil. Trans. R. Soc. B*, 367(1589):670–679, 2012.

[MOPU93] Michele Marchesi, Gianni Orlandi, Francesco Piazza, and Aurelio Uncini. Fast neural networks without multipliers. *IEEE Transactions on Neural Networks*, 4(1):53–62, 1993.

[MPP+15] Horia Mania, Xinghao Pan, Dimitris Papailiopoulos, Benjamin Recht, Kannan Ramchandran, and Michael I Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *arXiv preprint arXiv:1507.06970*, 2015.

[MR16] Aryan Mokhtari and Alejandro Ribeiro. Dsa: Decentralized double stochastic averaging gradient algorithm. *Journal of Machine Learning Research*, 17(61):1–35, 2016.

[Nes83] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). 1983.

[Nes13] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[Now06] Martin A Nowak. Five rules for the evolution of cooperation. *science*, 314(5805):1560–1563, 2006.

[NWC+11] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 4. Granada, Spain, 2011.

[NWS14] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in Neural Information Processing Systems*, pages 1017–1025, 2014.

[Ois10] Shigehiro Oishi. The psychology of residential mobility implications for the self, social relationships, and well-being. *Perspectives on Psychological Science*, 5(1):5–21, 2010.

[OKM+13] Shigehiro Oishi, Selin Kesebir, Felicity F Miao, Thomas Talhelm, Yumi Endo, Yukiko Uchida, Yasufumi Shibanai, and Vinai Norasakkunkit. Residential mobility increases motivation to expand social network: But why? *Journal of Experimental Social Psychology*, 49(2):217–223, 2013.

[OLS07] Shigehiro Oishi, Janetta Lun, and Gary D Sherman. Residential mobility, self-concept, and positive affect in social interactions. *Journal of personality and social psychology*, 93(1):131, 2007.

[OSYA15] SHIGEHIRO Oishi, JOANNA Schug, MASAKI Yuki, and JORDAN Axt. The psychology of residential and relational mobilities. *Handbook of advances in culture and psychology*, 5:221–272, 2015.

[Pin11a] Steven Pinker. *The better angels of our nature: Why violence has declined*, volume 75. Viking New York, 2011.

[Pin11b] Steven Pinker. Decline of violence: Taming the devil within us. *Nature*, 478(7369):309–311, 2011.

[PLT+16] Xinghao Pan, Maximilian Lam, Stephen Tu, Dimitris Papailiopoulos, Ce Zhang, Michael I Jordan, Kannan Ramchandran, Chris Re, and Benjamin Recht. Cyclades: Conflict-free asynchronous machine learning. *arXiv preprint arXiv:1605.09721*, 2016.

[PN02] Karen M Page and Martin A Nowak. Unifying evolutionary dynamics. *Journal of Theoretical Biology*, 219(1):93–98, 2002.

[Pol63] Boris Teodorovich Polyak. Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 3(4):643–653, 1963.

[Pro01] Danil Prokhorov. Ijcnn 2001 neural network competition. *Slide presentation in IJCNN*, 1, 2001.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[Ren11] Ping Ren. *Lifetime mobility in the United States: 2010*. US Department of Commerce, Economics and Statistics Administration, US Census Bureau, 2011.

[RGNL15] Patrick Roos, Michele Gelfand, Dana Nau, and Janetta Lun. Societal threat and cultural variation in the strength of social norms: An evolutionary basis. *Organizational Behavior and Human Decision Processes*, 129:14–23, 2015.

[RHBL07] Marc Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[RHS+15] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alex J Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in Neural Information Processing Systems*, pages 2629–2637, 2015.

[RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[RORF16] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. *ECCV*, 2016.

[RRWN11] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[RSB12] Nicolas L Roux, Mark Schmidt, and Francis R Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671, 2012.

[RSS11] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

[Sha16] Ohad Shamir. Without-replacement sampling for stochastic gradient methods: Convergence results and application to distributed optimization. *arXiv preprint arXiv:1603.00570*, 2016.

[SHM14] Daniel Soudry, Itay Hubara, and Ron Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *NIPS*, 2014.

[SMDH13] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[Smi82] John Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[SP73] J. M. Smith and G. R. Price. The logic of animal conflict. *Nature*, 246:15–18, 1973.

[SRB13] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013.

[SS14] Ohad Shamir and Nathan Srebro. Distributed stochastic optimization and learning. In *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*, pages 850–857. IEEE, 2014.

[SYHT09] Joanna Schug, Masaki Yuki, Hiroki Horikawa, and Kosuke Takemura. Similarity attraction and actually selecting similar others: How cross-societal differences in relational mobility affect interpersonal similarity in japan and the usa. *Asian Journal of Social Psychology*, 12(2):95–103, 2009.

[SYM10] Joanna Schug, Masaki Yuki, and William Maddux. Relational mobility explains between-and within-culture differences in self-disclosure to close friends. *Psychological Science*, 2010.

[SZ13] Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning*, pages 71–79, 2013.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[SZ15] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015.

[SZL13] Tom Schaul, Sixin Zhang, and Yann LeCun. No more pesky learning rates. In *Proceedings of The 30th International Conference on Machine Learning*, pages 343–351, 2013.

[Taj82] Henri Tajfel. Social psychology of intergroup relations. *Annual review of psychology*, 33(1):1–39, 1982.

[THDS+09] Arne Traulsen, Christoph Hauert, Hannelore De Silva, Martin A Nowak, and Karl Sigmund. Exploration dynamics in evolutionary games. *Proceedings of the National Academy of Sciences*, 106(3):709–712, 2009.

[TJ78] Peter D Taylor and Leo B Jonker. Evolutionary stable strategies and game dynamics. *Mathematical Biosciences*, 40(1-2):145–156, 1978.

[TMDQ16] Conghui Tan, Shiqian Ma, Yu-Hong Dai, and Yuqiu Qian. Barzilai-borwein step size for stochastic gradient descent. *arXiv preprint arXiv:1605.04131*, 2016.

[TSS+10] Arne Traulsen, Dirk Semmann, Ralf D Sommerfeld, Hans-Jürgen Krambeck, and Manfred Milinski. Human strategy updating in evolutionary games. *Proceedings of the National Academy of Sciences*, 107(7):2962–2966, 2010.

[Ver] Roman Vershynin. High-dimensional probability.

[WCSX13] Chong Wang, Xi Chen, Alex J Smola, and Eric P Xing. Variance reduction for stochastic gradient optimization. In *Advances in Neural Information Processing Systems*, pages 181–189, 2013.

[WEMG11] Stuart A West, Claire El Mouden, and Andy Gardner. Sixteen common misconceptions about the evolution of cooperation in humans. *Evolution and Human Behavior*, 32(4):231–262, 2011.

[Wor98] DDB Worldwide. The DDB Life Style Survey Data. `http://bowlingalone.com/?page_id=7`, 1998. [Online; accessed 20-October-2015].

[WRLG18] Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. *arXiv preprint arXiv:1803.02021*, 2018.

[WRS+17] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4151–4161, 2017.

[WS98] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[XZ14] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[You01] H Peyton Young. *Individual strategy and social structure: An evolutionary theory of institutions*. Princeton University Press, 2001.

[YSS09] Toshio Yamagishi, NAOTO Suzuki, and M Schaller. An institutional approach to culture. *Evolution, culture, and the human mind*, pages 185–203, 2009.

[ZCL15] Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd. In *Advances in Neural Information Processing Systems*, pages 685–693, 2015.

[Zei12] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[ZHMD17] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *ICLR*, 2017.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[ZLS09] Martin Zinkevich, John Langford, and Alex J Smola. Slow learners are fast. In *Advances in Neural Information Processing Systems*, pages 2331–2339, 2009.

[ZWLS10] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.

[ZWN+16] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[ZYG+17] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless CNNs with low-precision weights. *ICLR*, 2017.

[ZZX15] Leihan Zhang, Jichang Zhao, and Ke Xu. Who creates trends in online social media: The crowd or opinion leaders? *Journal of Computer-Mediated Communication*, 21(1):1–16, 2015.