Summer 2017

Motion-capture-based hand gesture recognition for computing and control

Andrew Gardner
Louisiana Tech University

MOTION-CAPTURE-BASED HAND GESTURE RECOGNITION FOR COMPUTING AND CONTROL

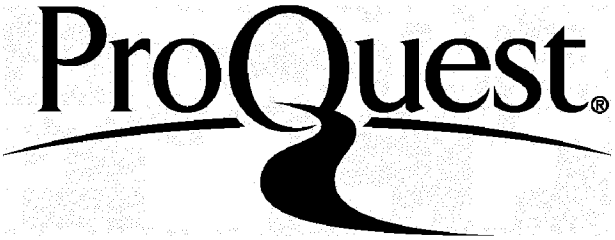
by

Andrew Gardner, B.S., M.S.

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

August 2017



LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

5/15/2017

Date

We hereby recommend that the dissertation prepared under our supervision by Andrew Bryan Gardner

entitled Motion-Capture-Based Hand Gesture Recognition for Computing and Control

be accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computational Analysis and Modeling

Rastko Selmic J.K.

Supervisor of Dissertation Research

Weizhong Dai

Head of Department

Computational Analysis and Modeling

Department

Recommendation concurred in: J.K.

Christian Duncan

Jorge Kamine

Weizhong Dai

Advisory Committee

Approved:

Director of Graduate Studies

Approved:

Dean of the Graduate School

Hisham Hegab/SD

Dean of the College

GS Form 13a
(6/07)

ABSTRACT

This dissertation focuses on the study and development of algorithms that enable the analysis and recognition of hand gestures in a motion capture environment. Central to this work is the study of unlabeled point sets in a more abstract sense. Evaluations of proposed methods focus on examining their generalization to users not encountered during system training.

In an initial exploratory study, we compare various classification algorithms based upon multiple interpretations and feature transformations of point sets, including those based upon aggregate features (e.g. mean) and a pseudo-rasterization of the capture space. We find aggregate feature classifiers to be balanced across multiple users but relatively limited in maximum achievable accuracy. Certain classifiers based upon the pseudo-rasterization performed best among tested classification algorithms. We follow this study with targeted examinations of certain subproblems.

For the first subproblem, we introduce the *a fortiori* expectation-maximization (AFEM) algorithm for computing the parameters of a distribution from which unlabeled, correlated point sets are presumed to be generated. Each unlabeled point is assumed to correspond to a target with independent probability of appearance but correlated positions. We propose replacing the expectation phase of the algorithm with a Kalman filter modified within a Bayesian framework to account for the unknown point labels which manifest as uncertain measurement matrices. We also

propose a mechanism to reorder the measurements in order to improve parameter estimates. In addition, we use a state-of-the-art Markov chain Monte Carlo sampler to efficiently sample measurement matrices. In the process, we indirectly propose a constrained k-means clustering algorithm. Simulations verify the utility of AFEM against a traditional expectation-maximization algorithm in a variety of scenarios.

In the second subproblem, we consider the application of positive definite kernels and the earth mover's distance (EMD) to our work. Positive definite kernels are an important tool in machine learning that enable efficient solutions to otherwise difficult or intractable problems by implicitly linearizing the problem geometry. We develop a set-theoretic interpretation of EMD and propose earth mover's intersection (EMI), a positive definite analog to EMD. We offer proof of EMD's negative definiteness and provide necessary and sufficient conditions for EMD to be conditionally negative definite, including approximations that guarantee negative definiteness. In particular, we show that EMD is related to various min-like kernels. We also present a positive definite preserving transformation that can be applied to any kernel and can be used to derive positive definite EMD-based kernels, and we show that the Jaccard index is simply the result of this transformation applied to set intersection. Finally, we evaluate kernels based on EMI and the proposed transformation versus EMD in various computer vision tasks and show that EMD is generally inferior even with indefinite kernel techniques.

Finally, we apply deep learning to our problem. We propose neural network architectures for hand posture and gesture recognition from unlabeled marker sets in a coordinate system local to the hand. As a means of ensuring data integrity, we also

propose an extended Kalman filter for tracking the rigid pattern of markers on which the local coordinate system is based. We consider fixed- and variable-size architectures including convolutional and recurrent neural networks that accept unlabeled marker input. We also consider a data-driven approach to labeling markers with a neural network and a collection of Kalman filters. Experimental evaluations with posture and gesture datasets show promising results for the proposed architectures with unlabeled markers, which outperform the alternative data-driven labeling method.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author 

Date 7/5/2017

GS Form 14
(5/03)

DEDICATION

This dissertation is dedicated to my loving and supportive parents, without whom it would likely have never been completed.

TABLE OF CONTENTS

ABSTRACT iii

DEDICATION vii

LIST OF TABLES xiii

LIST OF FIGURES xiv

ACKNOWLEDGMENTS xvii

CHAPTER 1 INTRODUCTION 1

1.1 An Overview of Hand Gesture Recognition 2

1.2 Problem Statement and Setting 6

1.2.1 Problem Setting 6

1.2.2 Key Objectives 10

1.3 Contribution 12

1.4 Limitations of the Study 13

1.5 Organization of Dissertation 14

CHAPTER 2 BACKGROUND 15

2.1 Fundamentals 15

2.1.1 Linear Algebra 16

2.1.2 Quaternions 17

2.1.3 Metrics 20

2.1.4 Kernels 21

2.1.5 Measures 24
2.1.6 Probability 25
2.1.7 Statistics 28
2.1.8 Classification 29
2.2 Special Topics in Statistics 30
2.2.1 Divergences 31
2.2.2 The Maximum-Likelihood Principle 33
2.2.3 Expectation-Maximization 35
2.2.4 Markov Chain Monte Carlo 36
2.3 Constrained Optimization 39
2.4 Optimal Transport 41
2.5 Filtering 45
2.5.1 The Kalman Filter 45
2.5.2 Bayes Filters 47
2.5.3 Extended Kalman Filter 48
2.5.4 Other Filters 50
2.6 Support Vector Machines 51
2.7 Neural Networks and Deep Learning 55
2.7.1 Multi-layer Perceptrons 55
2.7.2 Recurrent Neural Networks 58
2.7.3 Convolutional Neural Networks 60
2.7.4 Deep Learning 62

CHAPTER 3 3D HAND POSTURE RECOGNITION FROM SMALL, UNLABELED POINT SETS 67

3.1 Methodology 68

3.1.1 Data 69

3.1.2 Classifiers 71

3.1.3 Evaluation 77

3.2 Results 78

CHAPTER 4 ESTIMATING THE DISTRIBUTION OF UNLABELED, CORRELATED POINT SETS 82

4.1 Problem Definition and Related Work 83

4.2 The Proposed Algorithm 85

4.2.1 A Kalman Filter for Uncertain Measurement Matrices 86

4.2.2 *A Fortiori* Estimates 88

4.2.3 Defining the State 89

4.2.4 Defining the Measurements 89

4.2.5 Uncertain Measurement Matrices 91

4.2.6 The AFEM Algorithm 93

4.3 Experimental Evaluation 95

4.3.1 Simulation Description 96

4.3.2 Comparing Profiles 97

4.3.3 Results 100

CHAPTER 5 ON THE DEFINITENESS OF EARTH MOVER'S DISTANCE AND ITS RELATION TO SET INTERSECTION 104

5.1 Preliminaries 105

5.1.1 Multisets 106

5.1.2 Earth Mover's Distance 107

5.2 A Definite Preserving Transformation 108

5.3 EMD Is Conditionally Negative Definite For Certain Ground Distances 114

5.3.1 Earth Mover's Intersection: A Set Theoretic Interpretation of EMD 115

5.3.2 Transportation on the Real Line 119

5.3.3 Transportation on the Circle 120

5.3.4 Transportation on the L_2 Hypersphere 121

5.4 Experiments 124

5.4.1 Datasets 127

5.4.2 Design of Experiments 128

5.4.3 Results and Discussion 130

CHAPTER 6 NEURAL NETWORK ARCHITECTURES FOR GESTURE RECOGNITION 135

6.1 Tracking a Rigid Pattern 137

6.2 Feature Extraction from Unlabeled Marker Sets 141

6.2.1 Labeling Markers with Kalman Filters 141

6.2.2 Architectures for Unlabeled, Unordered Markers 143

6.3 Evaluation 146

6.3.1 Posture Recognition 147

6.3.2 Gesture Recognition 150

CHAPTER 7 CONCLUSIONS 154

7.1 Discussion 155

7.2 Future Work 157

7.2.1 Enhanced Data Collection 157

7.2.2 Constrained k-Means 158

7.2.3 Improved Online Marker Tracking and Labeling 159

7.2.4 MCMC Algorithms for Weighted Permutations 159

7.2.5 Posture and Gesture Recognition in Global Coordinates 160

7.2.6 Marker Filtering and Labeling with Neural Networks 161

7.2.7 Kernel Methods for Gesture Recognition 161

7.2.8 A Kernel Trick for Optimal Transport 162

APPENDIX A NOTATION 164

APPENDIX B DATASETS 167

B.1 General Remarks 168

B.2 Labeled Marker Dataset 169

B.2.1 Data Collection and Description 169

B.2.2 File Format 170

B.3 Posture Dataset 171

B.3.1 Data Collection and Description 172

B.3.2 File Format 173

B.4 Gesture Dataset 175

B.4.1 Data Collection and Description 175

B.4.2 File Format 175

BIBLIOGRAPHY 178

LIST OF TABLES

Table 3.1: The average BERs per user left out and corresponding standard deviations for tested transformed feature classifiers. Lower BER is better. 78

Table 3.2: The average BERs per user left out and corresponding standard deviations for tested aggregate feature classifiers tested. Lower BER is better. 79

Table 3.3: The average BERs per user left out and corresponding standard deviations for tested raw feature classifiers. Lower BER is better. 80

Table 5.1: Accuracies for texture recognition on normalized sets with KTH-TIPS. All kernels were found to be positive definite. Since sets are normalized, EMD is equal to $\widehat{\text{EMD}}$ 130

Table 5.2: Accuracies for object category classification on normalized sets with Caltech-101. All kernels were found to be positive definite. Since sets are normalized, EMD is equal to $\widehat{\text{EMD}}$ 130

Table 5.3: Accuracies for handwritten character recognition on unnormalized sets with the MNIST derived data 131

Table 5.4: Accuracies for posture recognition on unnormalized sets 131

Table 6.1: Accuracies for leave-one-user-out classification with the posture dataset. 149

Table 6.2: Accuracies for leave-one-user-out classification with the gesture dataset. 152

Table A.1: A list of symbols and notation used throughout the dissertation along with definitions and short descriptions. 165

Table A.2: A list of acronyms and their expansions used throughout the dissertation. 166

LIST OF FIGURES

Figure 1.1: The MAVSeN laboratory near the time data was gathered. A non-reflective padded covering was placed on the floor after the photo was taken. 7

Figure 1.2: The glove used as the data source for all experiments and datasets. The axes of the local coordinate system based upon the rigid pattern are shown. 8

Figure 2.1: An illustration of a separating hyperplane for a non-separable problem. Support vectors are circled. Note that the error ξ is with respect to the margin for the side of the plane on which the point should ideally be located 52

Figure 2.2: A detailed illustration of a multilayer perceptron with input $\mathbf{x}$, output $\mathbf{o}$, and two layers, one of which is hidden. The arrows show the flow of input from left to right. During back-propagation in training, the gradient of the error flows backwards from right to left. The bias is not shown. 56

Figure 2.3: A block diagram representing a hidden layer in an RNN. The layer's output $\mathbf{h}_t$ is provided as input to the next layer, which may or may not be recurrent. 59

Figure 2.4: A block diagram representing a GRU. The layer's output $\mathbf{h}_t$ is provided as input to the next layer, which may or may not be recurrent. 60

Figure 2.5: A 2×2 filter is convolved with a 3×3 input layer to produce a 2×2 output. The output is then given to a nonlinear activation function such as hyperbolic tangent. Note that without padding the input, the output will be smaller. 61

Figure 3.1: A 2D grid transformation with $m = 4$, $r_s = 2$, and $i = 2$. The opacity of each sphere is proportional to its activation by the top-left marker. 71

Figure 3.2: The algorithm used to train heuristic GMMs. Convergence depends upon O. If R is used, convergence occurs when the number of markers rematched to a different component drops below a threshold. Otherwise, convergence depends on the matching cost under C. A maximum number of iterations is allowed before convergence. 74

Figure 3.3: The sub-procedure used to initialize a heuristic Gaussian mixture model (GMM). 75

Figure 3.4: The sub-procedure used to refine a heuristic GMM 76

Figure 4.1: The improvement in $\mathcal{H}_1^2$ for each sampler when using AFEM versus normal EM as a function of spread for $d = 3$ and unconstrained true covariance. 100

Figure 4.2: The improvement in $\mathcal{H}_1^2$ for each sampler when using AFEM versus normal EM as a function of target count for $d = 2$ and unconstrained true covariance. 101

Figure 4.3: The improvement between each pair of samplers in $\mathcal{H}_1^2$ for AFEM as a function of the number of targets for $d = 3$ and unconstrained true covariance. 101

Figure 4.4: The improvement between each pair of samplers in $\log \mathcal{H}_2^2$ for AFEM as a function of the number of targets for $d = 3$ and unconstrained true covariance. 102

Figure 4.5: The improvement in the Rand index for each sampler when using AFEM versus normal EM as a function of spread for $d = 3$ and unconstrained true covariance 102

Figure 6.1: A high-level diagram of the overall architecture and flow of data from the lowest accessible level (Vicon Datastream SDK) to the desired result (probabilities for gesture classification). 135

Figure 6.2: An illustration of a CDAN architecture for sets of 3D marker positions, arbitrarily ordered. A function represented by an MLP is convolved with the positions to produce a dynamically learned embedding in some potentially high-dimensional space. 145

Figure 6.3: An illustration of a (unidirectional) RDAN architecture for sequences of 3D marker positions. Embeddings are no longer independent 146

Figure B.1: Pseudocode for calculating axes of the hand's local coordinate system using labeled markers 169

Figure B.2: The labeled marker dataset after processing (i.e. in local coordinates). Some outliers for certain classes are visible. 170

Figure B.3: The glove used to capture data along with a sample from each class of posture projected onto the local XY plane. The classes are fist (1), stop (2), point with one finger (3), point with two fingers (4), and grab (5). 172

ACKNOWLEDGMENTS

Many people have contributed in many ways to this dissertation. For those not explicitly mentioned in this small space, know that you are not forgotten.

I gratefully acknowledge Dr. Rastko Selmic, Dr. Jinko Kanno, and Dr. Christian Duncan for jointly advising me throughout my studies and discussing numerous tangents and diversions encountered along the way. Special thanks is extended to Dr. Duncan for originally approaching me as an undergraduate with talk of an upcoming gesture recognition project. I would also like to thank Dr. Jean Gourd and Dr. Weizhong Dai for serving on my committee. Thanks to my research group member Ademola for his insights regarding certain neural network architectures. Further thanks is given to Dr. Selmic for securing my financial support. On a related note, I am grateful for the funding provided by the Louisiana Space Consortium (LaSPACE) and the Louisiana Tech College of Engineering and Science.

My family and friends also offered tremendous support and motivation. To my immediate and extended family I offer my love and extreme gratitude for their unwavering support. To my friends I offer my appreciation for the many conversations and encounters that broke the monotony of an ordinary day in the office. Finally, to Josh I give my completely sincere and heartfelt gratitude for performing the critical task of eyeballing the dimensions of his office as a surrogate for the erstwhile MAVSeN laboratory.

CHAPTER 1

INTRODUCTION

The motivating subject of this dissertation is the development of an interactive hand posture and gesture recognition system for various computing and control environments. The primary objective of the contained research is to develop and explore methods and algorithms for robust and efficient hand gesture recognition for computing and control, with applications including but not limited to virtual reality, home automation, and robotics control. Examples include pointing to direct a robot to its destination, directly controlling a drone's pitch or yaw with a mimicked joystick gesture, or interpreting hand signals for commands or authentication. Accomplishing these tasks requires the accurate recognition of a user's posture, motion, and intent. Recognition of intent, however, is not within the scope of this project. The project is especially focused on characterizing the physical aspects of the gestures irrespective of context or semantics, which may change with the application. Effectively, the research focuses on developing an application independent software layer for gesture recognition. Vicon motion capture cameras act as a source of data, providing precise 3D coordinates of keypoints (infrared markers) on the user's hand.

Seeking alternative problems beyond gesture recognition to which developed algorithms or perceived insights can be applied is the chief secondary objective. The

development of algorithms and theory related to gesture recognition, computer vision, and various other and potentially unforeseen areas is emphasized. Multi-target tracking, optimal transport, and the design of neural network architectures are included in the list of related problems. As such, algorithms and methods proposed in the dissertation for gesture recognition are usually tailored for transference to other domains.

1.1 An Overview of Hand Gesture Recognition

Gesture recognition, as a means of human-computer interaction, provides an intuitive and effective interface for user control, offering the ability to perform complicated tasks with minimal effort. The success of smartphones and tablets with touchscreens supports this hypothesis. A significant amount of research involving gestures has been performed in the past two decades with many methods and solutions offered [97, 145]. Hand gesture recognition is an especially appealing branch of the gesture recognition field because it can offer a more tantalizing avenue for the average end-user, even if only for the visceral thrill of execution. However, there is no current camera-based system that can demonstrate robust and precise finger-based gesture recognition (or even tracking) in a sizable 3D space [152] (although significant strides in finger tracking have been made recently [2]).

We separate our recognition targets into two categories: postures and gestures. A posture, or static gesture, is one in which the hand makes a certain pose, such as holding a closed fist, whereas a (dynamic) gesture involves motion of the hand, arm, or fingers, such as pointing or waving. Examples of each abound in the literature. Ge *et al.* [42] project depth images of a hand onto orthogonal planes and use convolutional

neural networks (see Section 2.7.3) to estimate the hand's 3D pose. Bhuyan *et al.* [9] use a finite state machine with fuzzy logic to segment continuous gestures from video streams and present an integrated system for recognition of various postures and gestures. Hand gesture recognition is inherently interactive, providing a wide range of applications including virtual reality and games [78, 125], robot control [16, 92], and interactive sign language [75].

There are many different methods by which hand features can be measured. Gloves are sometimes used [32]. Ceruti *et al.* [14] use wireless magnetic sensors embedded in a glove to detect finger motion and interpret a Braille-like binary code for communication. Luzanin and Plancak [89] and Weissmann and Salomon [150] each use neural networks and data gloves to classify a variety of postures. Vision-based approaches [129] are of particular interest as they do not require any peripheral accessories other than the camera or equivalent sensing device. The Microsoft Kinect [53] and Leap Motion Controller [107] are both commercially available and affordable. The Kinect employs an RGB-D (color plus depth) camera for full body tracking, whereas the Leap Motion uses only a depth camera to track the hands. Both devices operate in a limited field of view, although of the two the Kinect is larger. However, the Kinect is generally focused on full-body gestures and lacks the precision to model individual fingers at a significant distance [53, 125]. In particular, the Kinect only differentiates between a closed and open hand using the commercial software. The Leap Motion Controller offers a peripheral-free interaction system in a limited 3D space, but its detection currently suffers from some notable limitations. The controller primarily detects extended fingers, and thus, like the Kinect, requires the hand to be

held at a certain angle with respect to the sensor. In fact, the Leap Motion Controller is incapable of recognizing a fist, and touching or crossing fingers can lead to spurious approximations of the hand's pose [107]. Developers are responsible for detecting certain gestures or postures, such as a fist, thus yielding inconsistent performance across applications and platforms. Wang *et al.* [149] offer an alternative vision-based approach that is capable of detecting a limited class of pinching gestures for 3D CAD applications.

Posture recognition is an integral component of gesture recognition. The level of detail with which the hand is probed affects the expressiveness and variety of recognizable gestures. A system should generally ensure that the user's hand is not relaxed and is making the correct shape before positively interpreting the motion, assuming that the gesture is not defined solely by the motion or trajectory (such as a figure-eight). A system such as Vicon enables the greatest range of dynamic expression in a gesture by tracking the articulation of individual fingers.

The usage of Vicon motion capture cameras is similar to but fundamentally distinct from both depth-based methods and vision-based approaches, which we define to be detection methods based on the visible spectrum of light. Motion capture cameras instead observe infrared (i.e. not visible) light reflected by markers placed at preselected locations on the subject of interest. A noteworthy advantage of motion capture is the low-volume and sparsity of the data. A significant amount of noise that can be introduced by the environment is automatically filtered. Only the coordinates of the markers, inferred by triangulation, are reported for each frame measured. Aside from the exceptionally high costs for the hardware and software involved and the

careful camera calibration required to make practical use of the system, motion capture also comes with another major disadvantage encountered repeatedly throughout this research: marker identity is not known except under very limited circumstances.

Marker identity is generally known (or equivalently, markers are labeled) only when part of a rigid pattern or predefined skeleton. A *rigid pattern* is a configuration of markers such that if each marker is connected by an inflexible rod, then the angle between each pair of rods is constant. Consequently, the rod lengths are also fixed. A *skeleton* differs from a rigid pattern in that certain rods and angles are explicitly defined whereas others are free to change. Marker identities are often determined by having the subject strike a pose (such as a "T"-pose for a full body skeleton) in order to label markers, after which joint angles and other parameters of the skeleton are determined via inverse kinematics [3] or some other, perhaps probabilistic, method [95]. Automatic skeleton learning [28] and tracking [118] are also possible under certain conditions. In many cases, though, marker trajectories need to be manually labeled in a post-processing step, and not all of these methods operate in real-time.

The term skeleton is not a misnomer; predefined skeletons often correspond to major bone and joint segments in the human body. Motion capture cameras are commonly used to model and record human motion for animation in movies and video games. The use of motion capture cameras for hand gesture recognition is also well-established. Chang *et al.* [15] use supervised feature selection techniques to discover a reduced marker set sufficient for classifying certain classes of grasp gestures and use a similar method to our own in determining a local reference frame for the hand. Liu and McMillan [83] propose a method to estimate missing marker positions

during motion from a Random Forest based on local linear kinematic models, which is related to previous work that focused on accurately estimating the motion itself with limited markers [84]. Martin *et al.* [91] use a similar camera system to our own in order to recognize specific user actions, such as lifting and tipping a carton of milk or writing, via a combination of vector quantization and dynamic time warping. Lee and Tsai [75] also use a Vicon camera system with neural networks that are trained to completion to recognize 20 Taiwanese sign language static gestures. Both of these works are distinguished from our own in that they do not deal with anonymous markers but with labeled entities; i.e. it was known prior to classification which marker corresponded to the thumb or other location. Martin *et al.* [91] employed Vicon Nexus software to define a skeletal model of the user's hand, although it is not clear how Lee and Tsai [75] accomplished the labeling.

1.2 Problem Statement and Setting

This section states the specific problems that we attempt to solve, provides a detailed description of the setting in which the problems lie, and gives brief sketches of possible solutions.

1.2.1 Problem Setting

This subsection describes the laboratory in which the research was conducted as well as the glove constructed to serve as the source of data for all algorithms analysis and development.

Laboratory

Ten Vicon MX T40 (4 megapixel) motion capture cameras available in the Micro-Aerial Vehicle and Sensor Networks (MAVSeN) Laboratory at Louisiana Tech University act as the source of data. The MAVSeN lab conducts research and development in small-scale vehicle design, cooperative intelligent sensing, and control algorithms for unmanned air and ground vehicles (see Figure 1.1). As the figure partially shows, the cameras are arranged roughly on the boundary of a rectangular area approximately 10×15 m^2. The cameras are capable of recording at multiple frame-rates, with 50 Hz and 100 Hz being the options used in the majority of situations including data capture and interactive tests.

Figure 1.1: The MAVSeN laboratory near the time data was gathered. A non-reflective padded covering was placed on the floor after the photo was taken.

Data Source

The collection of data is facilitated by the Vicon Tracker application, which provides a graphical user interface to configure camera settings and define rigid patterns. Vicon Tracker does not support skeletons. Vicon DataStream SDK [1] enables programmatic access to streaming data from Vicon Tracker via C++ and C# libraries. This data can then be written to a file or reacted to in a real-time or near real-time fashion.

A glove with 15 markers attached is used as the source of data for posture and gesture recognition, both for the generation of datasets and for the practical evaluation of developed algorithms. Figure 1.2 shows a picture of the glove with all markers visible.

Figure 1.2: The glove used as the data source for all experiments and datasets. The axes of the local coordinate system based upon the rigid pattern are shown.

Four of the markers form a rigid pattern on the back of the hand to serve as identification of the hand's position and orientation and to create a local coordinate system for the remaining 11 markers. The remaining 11 markers are unlabeled; they do not form part of a rigid pattern nor skeleton. A rigid pattern is infeasible because the markers are not related in any manner that could be described as rigid. All distances and angles between these markers are flexible. For a similar reason, a skeleton is also infeasible since in theory the skeleton needs fixed segment lengths between certain markers. In reality, even if some distortion is allowed in the segment lengths, a skeleton is still infeasible, or at the least impractical, due to the variance in the lengths but more so due to the inherently high rates of marker occlusion. Visibility of fingers can be blocked by other fingers or the hand itself depending upon the hand's pose and orientation. For example, the fingertips are occluded when making a fist and multiple markers may become occluded simply when the user's hand is relaxed at their side and pointing downwards. An effective skeletal model also requires a denser marker set than ours in order to capture the 20+ degrees of freedom of the hand [2, 32] and eliminate ambiguity between similar poses. A denser marker set is not very practical in our laboratory (but also in general for large capture spaces) due to limited resolution as the cameras have a hard time discerning individual markers that are too close together.

We therefore chose to develop algorithms that either extract the markers' identity and are robust to incorrect labels or avoid using this information altogether. In the course of this dissertation, we collected posture and gesture recognition datasets using unlabeled markers. The samples in each dataset were voluntarily provided by

12 users to provide a corpus of five postures and six gestures. In addition, a dataset of labeled markers was collected for one user (the author) and was meant to provide a representation of each marker's range of motion. These datasets are referred to multiple times throughout the dissertation and serve to evaluate proposed methods. For more detailed descriptions of each dataset including their capture, please refer to Appendix B.

1.2.2 Key Objectives

The key objectives of this research can ultimately be broken down into three related, but ultimately distinct, subproblems: marker tracking, marker labeling, and classification of postures and gestures. The subproblem of marker tracking deals with tracking the unlabeled markers through sequential frames in order to build complete tracks (i.e. trajectories) and fill in missing portions due to occlusion. Marker labeling is concerned with assigning parts of the hand either to individual markers in each frame or the complete tracks if available. These tasks are complementary in that solving one aids the solution of the other. Taken together, they serve as a way to cleanse input prior to posture or gesture recognition. Consistently labeled markers and trajectories would greatly simplify the application of different classification algorithms. However, labeling and tracking the markers are challenging problems that may introduce errors if not done in a adequately robust manner. Therefore, the design of classifiers that operate directly on the unlabeled markers would avoid any bias introduced by sub-par solutions to the other problems and may be considered the holy grail of this dissertation.

Marker Tracking

Since the markers that do not form part of the rigid pattern on the hand are unlabeled in each frame and inconsistently ordered when their positions are obtained from the Vicon DataStream SDK, motion information of individual markers is unavailable. One cannot select a marker and know its path for the duration of a gesture. This key objective is concerned with designing algorithms to address this fault. Natural contenders for the solution include the Kalman filter (Section 2.5.1) and its relatives. However, a Kalman filter will not work "out of the box" due to the unknown associations between markers from frame to frame. One class of solutions uses the assignment problem, which seeks the minimum-cost assignment between two sets of items given a cost for each pair of potentially assigned items (see Section 2.4 for more details). The assignment problem also plays a pivotal role in marker labeling.

Marker Labeling

The labeling of markers (or assignment of each marker to part of the hand) comprises the second major objective. A solution to this problem would make marker tracking trivial. However, the system must necessarily have some idea of what the "thumb" is or where it appears. A data-driven approach may be sufficient if not necessary to resolve this issue, where examples of labeled markers captured over a wide range of motion are collected. The labels for this data almost certainly need to be manually generated (see Section B.2). Similar to how a solution to marker labeling makes tracking easier, so too does the converse. Observing the entire or

partial trajectory of a tracked marker increases the confidence of assigning a label based upon position.

Classification

Classification comprises the ultimate goal of posture or gesture recognition wherein frames or sequences of frames are classified as a type of posture or gesture. Part of the study is concerned with identifying robust and efficient methods that can accomplish recognition to a reasonable degree of accuracy. A significant portion of the study is devoted to classification without labels or tracked markers. In fact, as stated above, the ideal results include methods that are effective using just the raw unlabeled data. While kernel methods (described in Sections 2.1.4 and 2.6) appear very promising in terms of accuracy (see Chapter 5) and theoretical support, they are relatively inefficient. Deep learning via neural networks (Section 2.7) provides a possible alternative. Deep learning avoids engineered features by instead providing a mechanism that implicitly learns important features during training [73]. As a result, one may expect better results using deep learning without labels, especially since there is no guarantee that a label is correct. In Chapter 6, we explore this idea and find it to hold true.

1.3 Contribution

Aside from the datasets described in Appendix B, the chief contributions of this dissertation are both algorithmic and theoretical. In Chapter 4, we propose a Kalman filter based algorithm for estimating the generating distribution of a collection of unlabeled, correlated point sets. This algorithm can also be considered a type of

constrained k-means clustering algorithm. We also provide proof of both positive and negative definite preserving kernel normalizations in Chapter 5, and we provide a principled generalization of the Wasserstein distance on sets of different sizes for kernel methods. Finally, in Chapter 6, we propose neural network architectures for posture and gesture recognition with labeled and unlabeled markers. Minor contributions are also contained within the text of each chapter.

1.4 Limitations of the Study

We are not especially concerned in this study with defining an extensive corpus of postures or gestures but rather on developing methods that could be applied on an arbitrary corpus with reasonable robustness and reliability. Rather than focusing on the anatomy of the hand and any particularly special qualities of it, we develop algorithms that apply to unlabeled point sets, which is a more general point of view.

We also note that the quality of our results are limited by the quality of our data. The markers used on the glove in Figure 1.2 were each 4 mm in diameter. The MAVSeN laboratory was not calibrated or setup to reliably detect these markers in the entire space. This inadequacy of the laboratory was due to multiple factors including camera count, layout, and bright ambient light that limited camera exposure. Data capture was generally confined to a small volume as a result, which may bias results and prohibits certain studies from taking place such as those involving both hands. We expect future extensions of this work to involve the collection of a much more extensive dataset under more favorable conditions.

1.5 Organization of Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 provides the knowledge necessary to understand concepts and tools fundamental to succeeding chapters. The following three chapters focus on posture recognition. Chapter 3 applies a variety of different classification algorithms and feature transforms in an exploratory study meant to guide future efforts. Chapter 4 provides a principled approach to an algorithm sketched in Chapter 3 for estimating the distribution of a posture defined by unlabeled marker sets. Chapter 5 applies the Wasserstein distance (see Section 2.4) to posture recognition and obtains the best accuracy of any method reported in this dissertation. The chapter also proposes a positive definite (PD)-preserving transformation for kernels and a principled adaptation of the Wasserstein distance to kernel methods with sets of different sizes. Chapter 6 introduces time to the discussion by directly examining the key objectives with deep learning. Finally, we conclude the dissertation with a discussion of the overall results and possible extensions and future work.

The dissertation is supplemented by Appendix A and Appendix B, which provide a table defining notation, a list of defined acronyms, and descriptions of the datasets gathered in support of the dissertation and used in various chapters.

CHAPTER 2

BACKGROUND

This chapter introduces concepts and tools used throughout the dissertation including notation, definitions, algorithms, and equations. First we review certain foundational elements that are used repeatedly throughout the dissertation. We follow this with introductions to various topics including Kalman filters, support vector machines, and neural networks. Note that this chapter is meant to provide the reader with merely a basic understanding of their fundamental theory and practical application. References for further information are provided.

2.1 Fundamentals

This section introduces notation, definitions, and basic fundamental topics such as linear algebra, probability, and kernels. Familiarity with certain topics not explicitly covered is assumed (e.g. set theory). Table A.1 provides a summary of the major notational elements used throughout the background and dissertation, some of which are given greater elaboration in the text. In addition, Table A.2 in the same appendix provides a reference for acronyms used in the text.

2.1.1 Linear Algebra

Readers are assumed to be familiar with topics in linear algebra such as matrices, vectors, and various matrix decompositions (e.g. eigen-, Cholesky). For a review of decompositions, a numerical analysis textbook such as Burden and Faires [12] suffices. Familiarity with calculus including differentiation and integration is also assumed.

Regarding notation conventions, a vector $\mathbf{v}$ can always be safely assumed to be a column vector. *Tensors* are also applied in certain contexts (see Section 2.7.3), where a tensor is a multi-dimensional array with an arbitrary number of dimensions. A tensor generalizes a matrix, which may be considered a two-dimensional array. We only use tensors to organize data, so no further knowledge of their theory is required. Just as elements of a matrix A are referenced by subscripts separated by commas (e.g. $A_{i,j}$ is the element in the i-th row and j-th column of the matrix A), so too are elements of a tensor.

Regarding multidimensional calculus, let us explicitly recall that the derivative of an m-dimensional vector-valued function $\mathbf{y}$ with respect to an n-dimensional vector $\mathbf{x}$ is

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}. \tag{2.1}$$

This matrix is known as the *Jacobian*. Note that the Jacobian is column-oriented in that the columns correspond to dimensions of $\mathbf{x}$, which differs from some representations of

the gradient. A similar matrix composed of second order partial derivatives is called the *Hessian.* If J is the Jacobian, then the Hessian H is given by $\frac{\partial \text{vec}(J)}{\partial \mathbf{x}}$.

Certain special matrix products are employed including the Hadamard and Kronecker tensor product. The *Hadamard* (or Schur) product between two matrices A and B of the same shape is defined to be the element-wise product denoted by $A \odot B$ and given by

$$A \odot B = [A_{i,j} B_{i,j}], \tag{2.2}$$

where i and j are valid indices. See Theorem 2.5 for a result concerning Hadamard products. The *Kronecker tensor* (or simply Kronecker) product [86] between an $n \times m$ matrix A and a $p \times q$ matrix B is denoted by $A \otimes B$ and defined to be the $np \times mq$ matrix given by

$$A \otimes B = \begin{bmatrix} A_{1,1}B & A_{1,2}B & \dots & A_{1,m}B \\ A_{2,1}B & A_{2,2}B & & \vdots \\ \vdots & & \ddots & \\ A_{n,1}B & \dots & & A_{n,m}B \end{bmatrix}. \tag{2.3}$$

2.1.2 Quaternions

Quaternions are an extension of the complex numbers that contain three imaginary components instead of one. We do not delve too deeply into the theory behind quaternion algebra. By definition [133], a quaternion is given by

$$\overline{q} = q_0 + q_1 \mathbf{i} + q_2 \mathbf{j} + q_3 \mathbf{k}, \tag{2.4}$$

where $q_i \in \mathbb{R}$, $i \in [0,3]$, and $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are an imaginary basis satisfying

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1, \tag{2.5}$$

$$\mathbf{ij} = -\mathbf{ji} = \mathbf{k}, \tag{2.6}$$

$$\mathbf{jk} = -\mathbf{kj} = \mathbf{i}, \tag{2.7}$$

$$\mathbf{ki} = -\mathbf{ik} = \mathbf{j}. \tag{2.8}$$

For our purposes, it is sufficient to note that a quaternion $\overline{q}$ can be represented as a four-dimensional vector or equivalently as a real scalar paired with an imaginary three-dimensional vector, i.e. $\overline{q} = (q_0, \mathbf{q})$ with $\mathbf{q} = \langle q_1, q_2, q_3 \rangle^{\mathsf{T}}$. The product of two quaternions $\overline{p} = (p_0, \mathbf{p})$ and $\overline{q} = (q_0, \mathbf{q})$ is non-commutative and is given by

$$\overline{pq} = (p_0 q_0 - \mathbf{p}^T \mathbf{q}, q_0 \mathbf{p} + p_0 \mathbf{q} + \mathbf{p} \times \mathbf{q}), \tag{2.9}$$

where

$$\mathbf{p} \times \mathbf{q} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ p_1 & p_2 & p_3 \\ q_1 & q_2 & q_3 \end{vmatrix} \tag{2.10}$$

is the cross product of $\mathbf{p}$ and $\mathbf{q}$. The *conjugate* of a quaternion $\overline{q} = (q_0, \mathbf{q})$ is given by $\overline{q}^\star = (q_0, -\mathbf{q})$.

Unit quaternions (i.e. those satisfying $\sqrt{\overline{qq}^\star} = 1$) are especially useful for modeling rotations. If θ and $\mathbf{v}$ (as a unit vector) are the angle and the axis of a right-handed rotation, then the rotation may be represented by the unit quaternion

$$\overline{q} = \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right) \mathbf{v} \right). \tag{2.11}$$

For axis-angle representations (where the magnitude of the vector $\mathbf{v}$ is the angle of rotation), we define the operation

$$\text{quat}\,(\mathbf{v}) = \left(\cos\left(\frac{\|\mathbf{v}\|}{2}\right), \sin\left(\frac{\|\mathbf{v}\|}{2}\right)\frac{\mathbf{v}}{\|\mathbf{v}\|}\right) \tag{2.12}$$

to facilitate conversions between the two representations. Note that $-\overline{q}$ represents the same rotation but in the opposite direction and that the unit quaternion with positive scalar represents the shorter of the two rotations. Let us assume that all unit quaternions henceforth denote rotations. Unit quaternions can be used to rotate arbitrary three-dimensional vectors by placing the vectors in the imaginary part of a quaternion and performing quaternion multiplication. In other words, if $\overline{q}$ is a unit quaternion, $\mathbf{v} \in \mathbb{R}^3$, and $\mathbf{p} = (0, \mathbf{v})$, then

$$\mathbf{qpq}^\star = \begin{bmatrix} 0 \\ q_0(q_0\mathbf{v} + 2\mathbf{q} \times \mathbf{v}) + 2\mathbf{q}\mathbf{q}^T\mathbf{v} - \mathbf{v}\mathbf{q}^T\mathbf{q} \end{bmatrix} \tag{2.13}$$

yields the rotated vector in the imaginary part of the result. Since $p_0 = 0$, the result is purely imaginary. A sequence of rotation quaternions $\overline{q}_1, \overline{q}_2 \ldots, \overline{q}_r$ can be composed with a single quaternion $\overline{q}_R = \overline{q}_r\overline{q}_{r-1}\cdots\overline{q}_1$.

We use quaternions in Chapter 6 to design Kalman filters (see Section 2.5) for estimating the orientation of rigid patterns of markers. Certain derivatives are useful and are listed here. The partial derivative of a quaternion product with respect to the right-hand quaternion is

$$\frac{\partial\overline{pq}}{\partial\overline{q}} = \begin{bmatrix} p_0 & -\mathbf{p}^\intercal \\ \mathbf{p} & p_0\mathbf{I}_3 + \lfloor\mathbf{p}\times\rfloor \end{bmatrix}, \tag{2.14}$$

where $\lfloor \mathbf{p}\times \rfloor$ is the skew-symmetric cross product matrix

$$\lfloor \mathbf{p}\times \rfloor = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}. \tag{2.15}$$

Differentiating with respect to the left-hand quaternion yields

$$\frac{\partial \overline{pq}}{\partial \overline{p}} = \begin{bmatrix} q_0 & -\mathbf{q}^\intercal \\ \mathbf{q} & q_0\mathbf{I}_3 - \lfloor \mathbf{q}\times \rfloor \end{bmatrix}. \tag{2.16}$$

In addition,

$$\frac{\partial \text{quat}\,(\mathbf{v})}{\partial \mathbf{v}} = \begin{bmatrix} -\frac{1}{2}\sin\left(\frac{\|\mathbf{v}\|}{2}\right)\frac{\mathbf{v}^\intercal}{\|\mathbf{v}\|} \\ \frac{1}{\|\mathbf{v}\|}\sin\left(\frac{\|\mathbf{v}\|}{2}\right)\mathbf{I}_3 + \mathbf{v}\mathbf{v}^\intercal\frac{\|\mathbf{v}\|/2\cos(\|\mathbf{v}\|/2)-\sin(\|\mathbf{v}\|/2)}{\|\mathbf{v}\|\mathbf{v}^\intercal\mathbf{v}} \end{bmatrix}. \tag{2.17}$$

In order to account for small $\|v\|$, we note via L'Hospital's rule that

$$\lim_{\|v\|\to 0}\frac{\partial \text{quat}\,(\mathbf{v})}{\partial \mathbf{v}} = \begin{bmatrix} -\frac{1}{4}\mathbf{v}^\intercal \\ \frac{1}{2}\mathbf{I}_3 - \frac{1}{24}\mathbf{v}\mathbf{v}^\intercal \end{bmatrix}. \tag{2.18}$$

2.1.3 Metrics

A *metric* is a function that satisfies certain axioms (outlined in Definition 2.1) and can be used to represent a distance between two items.

Definition 2.1. *A function $\delta : X \times X \to \mathbb{R}$ is a metric on some set X if and only if the following properties are satisfied for every $x, y, z \in X$:*

- *Non-negativity:* $\delta(x, y) \geq 0$.
- *Symmetry:* $\delta(x, y) = \delta(y, x)$.
- *Identity of indiscernibles:* $\delta(x, y) = 0$ *if and only if* $x = y$.
- *Triangle inequality:* $\delta(x, y) \leq \delta(x, z) + \delta(y, z)$.

We will use the term *discrete metric* to refer to the 0-1 distance defined by $\delta_{0\text{-}1}(x, y) = 0$ if $x = y$ and 1 otherwise. As can be inferred from its name, the discrete metric is a metric. We also define the term *semimetric* to indicate satisfaction of all of the preceding properties except for the triangle inequality. The Euclidean distance is a metric, and the squared Euclidean distance is a semimetric. A simple example of the squared Euclidean distance failing the triangle inequality may be noted with the points $x = (0, 0)$, $y = (0, 2)$, and $z = (0, 1)$ as elements of $\mathbb{R}^2$ since the resulting distances are $\delta(x, y) = 4$, $\delta(x, z) = 1$, and $\delta(y, z) = 1$.

2.1.4 Kernels

A *kernel* on a set X is, in general, a function $K : X \times X \to \mathbb{R}$. Kernels can be used to represent the similarity or distance between objects, and therefore generalize the notion of a metric. Kernels that satisfy Definition 2.2 below are especially useful. Be aware that our notation condenses the double summation when each index i and j shares the same range as in (2.19).

Definition 2.2. *A kernel K is PD if and only if it is symmetric and for any choice of n distinct elements $x_1, \ldots, x_n$ and real numbers $c_1, \ldots, c_n$,*

$$\sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) \geq 0. \tag{2.19}$$

If the constraint $\sum_{i=1}^{n} c_i = 0$ is added, then K is conditionally positive definite (CPD).

The condition (2.19) is equivalent to testing whether the kernel matrix for the chosen elements $G_K = [K(x_i, x_j)]$ is positive semi-definite via a quadratic form, i.e. $\mathbf{c}^{\mathsf{T}} G_K \mathbf{c} \geq 0$ where $\mathbf{c} = [c_i]$. If the kernel is PD, then G_K is called the Gram matrix. A (conditionally) strictly PD kernel is one in which the preceding inequalities

are strict with equality holding only if each $c_i = 0$. One may note that PD implies CPD, but the converse does not hold. Simply reversing the inequality of (2.19) yields *negative definite (ND)* kernels of each respective type. Consequently, if K is PD, then $-K$ is ND. PD kernels are useful for a variety of machine learning tasks including classification, regression, and principal component analysis and are sometimes known as Mercer kernels [13].

PD-ness is an attractive property because it implies the existence of a mapping $\phi : X \to H$ from X to some Hilbert space H in which the kernel gives the value of the inner product and certain nonlinear problems in X become linear [123], i.e.

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle . \tag{2.20}$$

This property is the key component of the so-called "kernel trick," wherein a separating hyperplane is implicitly found without ever working directly in H (see Section 2.6). A conditionally negative definite (CND) kernel is also related to some Hilbert space H through a mapping ϕ by

$$K(x_i, x_j) = \|\phi(x_i) - \phi(x_j)\|^2. \tag{2.21}$$

Note that the existence of ϕ implies the respective type of definiteness and vice versa. CND kernels are sometimes referred to as metrics of negative type, and as indicated by (2.21), correspond to functions that isometrically embed into squared Euclidean space. Note that we follow traditional nomenclature for kernels in that PD and strictly PD kernels correspond to positive semi-definite and PD matrices, respectively.

The following three results are adapted from Berg *et al.* [7] and form a basis for several later propositions. Theorem 2.3 and Lemma 2.4 propose relationships between

CND and PD kernels. Kernels of the form $\exp(uK)$ with arbitrary K are sometimes called generalized radial basis function (RBF) kernels. Theorem 2.5, originally proved by Schur [120] and known as the Schur product theorem, demonstrates that PD kernels are closed under multiplication. Note that Theorem 2.5 does not apply to CPD kernels.

Theorem 2.3 ([7]). *Let X be a nonempty set and let $K : X \times X \to \mathbb{R}$ be a symmetric kernel. Then K is CND (CPD) if and only if $\exp(uK)$ is PD for each $u < 0$ $(0 < u)$.*

Lemma 2.4 ([7]). *Let X be a nonempty set, $x_0 \in X$, and let $D : X \times X \to \mathbb{R}$ be a symmetric kernel. Let $K(x, y) = D(x, x_0) + D(y, x_0) - D(x, y) - D(x_0, x_0)$. Then K is PD if and only if D is CND. If $D(x_0, x_0) \geq 0$, then $K_0(x, y) = K(x, y) + D(x_0, x_0)$ is also PD.*

Theorem 2.5 ([7]). *If $K_1 : X \times X \to \mathbb{R}$ and $K_2 : X \times X \to \mathbb{R}$ are both PD, then their Schur product $(K_1 \cdot K_2)(x, y) = K_1(x, y)K_2(x, y)$ is also PD.*

The next two propositions are adapted from Boughorbel *et al.* [11] and were involved in the derivation of the generalized histogram intersection kernel. As a preview of upcoming proofs and an example of working with kernels, a proof of Proposition 2.6 is given since the statement appears counterintuitive at first.

Proposition 2.6 ([11]).

$$K_f(x, y) = f(x) + f(y) \tag{2.22}$$

is both a CPD and CND kernel for any function f.

Proof. Let $c_1, \ldots, c_n$ and $x_1, \ldots, x_n$ be defined as in Definition 2.2 with $\sum_{i=1}^{n} c_i = 0$. Then the following holds.

$$\begin{aligned}
\sum_{i,j=1}^{n} c_i c_j K_f(x_i, x_j) &= \sum_{i,j=1}^{n} c_i c_j \left[f(x_i) + f(x_j) \right] \\
&= \sum_{i,j=1}^{n} c_i c_j f(x_i) + \sum_{i.j=1}^{n} c_i c_j f(x_j) \\
&= 2 \sum_{i,j=1}^{n} c_j c_i f(x_i) \qquad (2.23) \\
&= 2 \left(\sum_{j=1}^{n} c_j \right) \left(\sum_{i=1}^{n} c_i f(x_i) \right) \\
&= 0.
\end{aligned}$$

□

Proposition 2.7 ([11]). *If K is positive valued and a CND kernel, then $K^{-\gamma}$ is PD for each $\gamma \geq 0$.*

2.1.5 Measures

A *measure* is a function that generalizes the notion of cardinality, area, volume, or length. To be precise, a measure $\mu : \Sigma_X \to \mathbb{R}$ assigns a number to subsets contained in a *σ-algebra* Σ_X of some set X, where a σ-algebra is some collection of subsets of X that contains the empty set and is closed under complement, countable unions, and countable intersections. The measure of a subset must be less than or equal to that of its superset, i.e. $\mu(A) \leq \mu(B)$ if $A \subseteq B$. Measures also possess countable additivity, i.e. the measure of the union of disjoint sets is the sum of their measures. The elements of X on which μ has non-zero measure constitute its support, denoted $\text{supp}\,(\mu)$. We use measures in a somewhat informal sense; we do not particularly care whether the measure is Lebesgue, Radon, Haar, etc. For a deeper understanding of

measures, please refer to any introductory textbook on analysis or measure theory, e.g. Tau [131].

2.1.6 Probability

Readers are assumed to be familiar with the concept of a random variable and probability density or mass functions. Random variables are often denoted with capital letters, but we will not maintain this convention as we more often deal with random vectors and matrices. A very brief review of some key topics is given the following subsections.

Joints and Marginals

Recall that a *joint* distribution $f_{X,Y}$ between two random variables X and Y assigns probability mass or density to every possible combination of X and Y. A *marginal* distribution assigns a density f_X to X (or f_Y for Y) and is related to the joint distribution according to

$$f_X(x) = \int_y f_{X,Y}(x, y)\mathrm{d}y. \tag{2.24}$$

In this situation, Y is said to have been marginalized out. A *conditional* distribution on X given Y (denoted $X \mid Y$) is also related to the marginal and joint distributions according to

$$f_{X|Y}(x, y) = \frac{f_{X,Y}(x, y)}{f_Y(y)}. \tag{2.25}$$

One may note that (2.25) is simply a restatement of the ubiquitous Bayes' theoerem relating the conditional probabilities of two events A and B

$$p(A \mid B) = \frac{p(B \mid A)p(A)}{p(B)}, \tag{2.26}$$

where $p(A)$ denotes the probability of the event A (and similarly for the other terms).

Probability Measures

Define $\mathcal{F}(X)$ to be the family of measures $\mu : \Sigma_X \to \mathbb{R}$ yielding finite, non-negative measure $\mu(X)$. In addition, let $\mathcal{P}(X) \subset \mathcal{F}(X)$ be the family of all *probability measures* on X, where a probability measure is a measure that assigns a total mass of 1 to X, i.e. $\mu(X) = 1$. A probability measure μ is associated with a mass or density function that can be obtained by restricting μ to just individual elements of X. Let $\mu^{(r)} : X \to \mathbb{R}$ be this restricted version of μ. The set X is sometimes referred to as the state space or domain of the random variable associated with μ. One should note that μ is uniquely defined by $\mu^{(r)}$ and vice versa. This fact should be evident since for a given set $Y \in \Sigma_X$,

$$\mu(Y) = \int_Y \mu^{(r)}(x)\mathrm{d}x. \tag{2.27}$$

Due to this relation, we will often abuse notation and simply refer to $\mu^{(r)}$ directly as a measure or μ as a distribution or even interchange the terms distribution and random variable. We will also tend to use $p(Y)$ to denote the probability of the event Y when no probability measure is explicitly given as in (2.26).

Common Distributions

Let us review some common probability distributions—namely the normal, binomial, and multinomial distributions. We also discuss mixture models.

Let $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ denote a *multivariate normal* or *Gaussian distribution* with mean $\boldsymbol{\mu}$ and covariance Σ, and let $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma)$ denote the probability density function of

$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ evaluated at $\mathbf{x}$, or

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = |2\pi\Sigma|^{-1/2} e^{(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})}. \tag{2.28}$$

In the event that $\boldsymbol{\mu}$ and Σ are scalars, then we simply have the normal distribution. A (multivariate) normal distribution is parameterized by its mean and variance, which are respectively $\boldsymbol{\mu}$ and Σ.

Let $\mathcal{B}(p, n)$ represent a *binomial distribution* with n trials and success parameter p. A binomial distributions may represent the outcome of n coin flips. The probability mass function is given by

$$\mathcal{B}(x; p, n) = \binom{n}{x} p^x (1-p)^{n-x}. \tag{2.29}$$

Note that $\mathcal{B}(p, 1)$ denotes a *Bernoulli distribution.*

A *multinomial distribution* is a generalization of the binomial distribution to k outcomes instead of two. For example, where a binomial distribution can represent the outcome of independent coin flips, a multinomial distribution represents the outcome of independent k sided dice rolls. The probability mass function is given by

$$\mathcal{C}(\mathbf{x}; \mathbf{p}, n) = \frac{n!}{\prod_{i=1}^{k} x_i!} \prod_{i=1}^{k} p_i^{x_i}, \tag{2.30}$$

where n is the number of trials, p_i is the probability of the i-th outcome on an independent trial, and x_i is the number of times the i-th outcome occurs in n trials. The vectors $\mathbf{x}$ and $\mathbf{p}$ are just notationally convenient. By necessity, $\sum p_i = 1$ and $\sum x_i = n$. Note that $\mathcal{C}(\mathbf{p}, 1)$ denotes a *categorical distribution.*

A *mixture* is a distribution defined as a weighted combination of two or more distributions with the same domain. For example, the probability density η of a

mixture of two distributions μ and ν given respective weights α and β ($\alpha + \beta = 1$) would be given by

$$\eta(x) = \alpha\mu(x) + \beta\nu(x). \tag{2.31}$$

The *GMM* is possibly the most common example of a mixture, which is defined to be a mixture of normal or multivariate normal distributions. The probability density function of a GMM with k components $\mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$, $i \in [1, k]$, and mixture weights $\boldsymbol{\pi} = [\pi_i]$, $\sum_{i=1}^{k} \pi_i = 1$, is given by

$$\mathcal{M}(\mathbf{x}; \boldsymbol{\mu}_1, \Sigma_1, \ldots, \boldsymbol{\mu}_k, \Sigma_k, \boldsymbol{\pi}) = \sum_{i=1}^{k} \pi_i \mathcal{N}(\mathbf{x}; \mu_i, \Sigma_i). \tag{2.32}$$

Despite their prevalence, many problems involving GMMs, such as estimation of their parameters [38], do not have closed form solutions.

2.1.7 Statistics

Readers are assumed to be familiar with basic statistics terminology including but not limited to moments (e.g. mean, variance, etc.) For completeness, certain topics will be reviewed here. The reader is referred to the textbook by Hogg *et al.* [59] for more complete coverage of the subject.

The *expected value* (commonly referred to as the mean or average) of a function $f(X)$ of a random variable X is a sum over the variable's entire domain weighted by the associated probability measure μ and is defined to be

$$\mathbb{E}_X\left[f(X)\right] = \int_{\text{supp}(\mu)} f(x)\mu(x)\mathrm{d}x. \tag{2.33}$$

Note that μ is implicit to the random variable X here. In fact, given the context, we could have simply written $\mathbb{E}\left[f(X)\right]$. In the event that we are given a probabiliy

measure/density without a random variable, then an implicit random variable can be denoted by the notation $\mathbb{E}_{X\sim\mu}[f(X)]$. Note that the value reported by the expectation does not necessarily lie within the domain of the random variable. A related quantity, the variance, is defined as $\mathbb{E}[(X - \mathbb{E}[X])^2]$. The generalization of the variance to jointly distributed variables X and Y is called the covariance and is defined as

$$\mathrm{Cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])], \tag{2.34}$$

where the expectation is understood to be taken with respect to their joint distribution.

2.1.8 Classification

Classification is a problem within the field of machine learning that involves assigning (i.e. classifying) an object or *instance* to one of several categories or *classes*. A *classifier* is an algorithm that performs this assignment. Equivalently, we may consider the classifier a parametric function that maps instances to classes. Generally, we wish to maximize the expected accuracy (or minimize the expected error) of a classifier when presented with an arbitrary instance. We *train* the classifier by selecting (or trying to select) the optimal parameters with respect to this (or some proxy) criterion. The process of training typically assumes that a set of independent and identically distributed data $\mathbb{X}$ is available and is accompanied by a set of known class labels $\mathbb{Y}$. In order to fairly evaluate the efficacy of the classifier, one must usually partition $\mathbb{X}$ and $\mathbb{Y}$ into a training set on which we train the classifier and a disjoint test set on which we assess its performance. This treatment is necessary since the error on the training set is not necessarily indicative of the classifier's generalization error when given new data.

Many different classification algorithms exist of varying complexity. For example, k-nearest neighbor (k-NN) and naive Bayes sit at the simpler end of the spectrum. The k-NN classifier is non-parametric (and thus requires no training) and consists of classifying an instance by a majority vote based on the classes of the k most similar instances in the training set (assuming a kernel to compute similarity between instances is defined). Naive Bayes is the name of a probabilistic classifier that assigns a class $\hat{y}$ to an instance represented by a *feature* vector $\mathbf{x}$ according to

$$\hat{y} = \arg\max_{\kappa \in \mathcal{K}} p(\kappa) \prod_i p(x_i \mid \kappa), \tag{2.35}$$

where $\mathcal{K}$ is the set of classes. The classifier is naive because it assumes that each feature x_i is conditionally independent given a class κ. Features, though not always explicitly required, play an important role in classification, and the selection or computation of useful features is a commonly pursued research topic [15, 51, 88, 114, 156]. In this work, we focus especially on kernel-based classifiers (Section 2.6) and neural networks (Section 2.7). For a more thorough review of classification and machine learning in general, please refer to introductory textbooks on the subject, e.g. Smola and Vishwanathan [126].

2.2 Special Topics in Statistics

In this section we cover some more advanced topics in statistics of which a casual acquaintance with the subject may not be entirely knowledgeable.

2.2.1 Divergences

An *f-divergence* is a function that measures the difference between two probability distributions [79]. We describe three well-known f-divergences that are used or referred to in following chapters.

The *Kullback-Leibler (KL) divergence* D_{KL} between two probability distributions $\mu : X \to \mathbb{R}$ and $\nu : X \to \mathbb{R}$ is given by

$$D_{KL}(\mu \| \nu) = \mathbb{E}_{x \sim \mu} \left[\ln \frac{\mu(x)}{\nu(x)} \right] \tag{2.36}$$

and measures the information gain when transitioning from ν to μ. Although it can be quite difficult to compute in general (for example, if both distributions are Gaussian mixture models [61]), a special case for which a closed form solution exists is the KL divergence between two d-dimensional multivariate Gaussian distributions $\mu \sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$ and $\nu \sim \mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2)$, which is given by

$$D_{KL}(\mu \| \nu) = \frac{1}{2} \left[\ln \frac{|\Sigma_2|}{|\Sigma_1|} + \operatorname{tr} \left[\Sigma_2^{-1} \Sigma_1 \right] - d + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\intercal \Sigma_2 (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \right]. \tag{2.37}$$

Another special case is of that between two Bernoulli distributions $\mu \sim \mathcal{B}(p, 1)$ and $\nu \sim \mathcal{B}(q, 1)$, which is given by

$$D_{KL}(\mu \| \nu) = p \ln \frac{p}{q} + (1 - p) \ln \frac{1 - p}{1 - q}. \tag{2.38}$$

Finally, KL divergence is additive. If μ and ν can each be decomposed into independent distributions $\mu_1 : X \to \mathbb{R}$, $\mu_2 : Y \to \mathbb{R}$, $\nu_1 : X \to \mathbb{R}$, and $\nu_2 : Y \to \mathbb{R}$ such that $\mu(x, y) = \mu_1(x)\mu_2(y)$ and $\nu(x, y) = \nu_1(x)\nu_2(y)$, then

$$D_{KL}(\mu \| \nu) = D_{KL}(\mu_1 \| \nu_1) + D_{KL}(\mu_2 \| \nu_2). \tag{2.39}$$

Drawbacks to KL divergence include the facts that it is asymmetric and unbounded.

The *Jensen-Shannon (JS) divergence* is a symmetric divergence based on KL divergence that compares μ and ν to the midpoint distribution $\eta = (\mu + \nu)/2$,

$$D_{JS}(\mu, \nu) = D_{KL}(\mu \| \eta) + D_{KL}(\nu \| \eta). \tag{2.40}$$

Defined in terms of *Shannon entropy*,

$$H(\mu) = -\mathbb{E}_{x \sim \mu} [\ln \mu(x)], \tag{2.41}$$

one finds that

$$D_{JS}(\mu, \nu) = H(\eta) - \frac{1}{2}[H(\mu) + H(\nu)]. \tag{2.42}$$

Aside from being symmetric, JS divergence offers some other advantages. First, it ranges from 0 to $\ln 2$. Perhaps more importantly, $\sqrt{D_{JS}}$ is a metric, which is a consequence [7] of the fact that JS divergence is CND [40].

The squared *Hellinger distance* between two distributions μ and ν defined on the same σ-algebra is defined as

$$H^2(\mu, \nu) = \frac{1}{2} \int \left(\sqrt{\mu(x)} - \sqrt{\nu(x)} \right)^2 \mathrm{d}x. \tag{2.43}$$

The squared Hellinger distance is symmetric and ranges from 0 (if and only if $\mu = \nu$) to 1. In fact, the squared Hellinger distance is CND [57], which can rather trivially be seen by definition of H^2 as a sum of squared differences or as a constant minus PD kernel through its alternative form

$$H^2(\mu, \nu) = 1 - \int \sqrt{\mu(x)\nu(x)} \mathrm{d}x. \tag{2.44}$$

Consequently, H is a metric [7]. The Hellinger distance is also multiplicative for joint independent distributions. If μ and ν can each be decomposed into independent

distributions $\mu_1 : X \to \mathbb{R}$, $\mu_2 : Y \to \mathbb{R}$, $\nu_1 : X \to \mathbb{R}$, and $\nu_2 : Y \to \mathbb{R}$ such that $\mu(x, y) = \mu_1(x)\mu_2(y)$ and $\nu(x, y) = \nu_1(x)\nu_2(y)$, then

$$\begin{aligned} H^2(\mu, \nu) &= 1 - \iint \sqrt{\mu_1(x)\mu_2(y)\nu_1(x)\nu_2(y)}\mathrm{d}x\mathrm{d}y \\ &= 1 - \int \sqrt{\mu_1(x)\nu_1(x)}\mathrm{d}x \int \sqrt{\mu_2(y)\nu_2(y)}\mathrm{d}y. \end{aligned} \tag{2.45}$$

We may also express H^2 as

$$H^2(\mu, \nu) = 1 - \int \mu(x)\sqrt{\frac{\nu(x)}{\mu(x)}}\mathrm{d}x = 1 - \mathbb{E}_{x\sim\mu}\left[\sqrt{\frac{\nu(x)}{\mu(x)}}\right], \tag{2.46}$$

which enables Monte Carlo estimation of H^2 by sampling from μ (or ν by a similar construction). The squared Hellinger distance between two multivariate Gaussian distributions $\mu \sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$ and $\nu \sim \mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2)$ is given by

$$H^2(\mu, \nu) = 1 - \frac{|\Sigma_1|^{1/4}\,|\Sigma_2|^{1/4}}{\left|\frac{\Sigma_1+\Sigma_2}{2}\right|^{1/2}} \exp\left[-\frac{1}{8}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\intercal \left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right]. \tag{2.47}$$

2.2.2 The Maximum-Likelihood Principle

Let $p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})$ be the probability density or mass function of a family of probability distributions parameterized by $\boldsymbol{\theta}$, and let $\mathbb{X} = \mathbf{x}_1, \ldots, \mathbf{x}_l$ be a set of samples drawn independently from an unknown distribution p_{data}. The *maximum likelihood principle* states that the value of $\boldsymbol{\theta}$ that maximizes the *likelihood*

$$L(\boldsymbol{\theta}; \mathbb{X}) = \prod_{i=1}^{l} p_{\text{model}}(\mathbf{x}_i; \boldsymbol{\theta}) \tag{2.48}$$

of generating $\mathbb{X}$ from the family p_{model},

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \mathbb{X}), \tag{2.49}$$

is an asymptotically minimum variance unbiased estimator (subject to some regularity conditions). Therefore, maximizing the likelihood is a prudent objective for many

problems. Often, as a matter of practicality, the log-likelihood is maximized instead since the logarithm is monotonic, i.e.

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{l} \log p_{\text{model}}(\mathbf{x}_i; \boldsymbol{\theta}). \tag{2.50}$$

We show here that maximum likelihood estimation is closely related to KL divergence (Section 2.2.1) (note that the argument is paraphrased from Goodfellow *et al.* [44]).

The ultimate purpose of p_{model} is to estimate the true probability p_{data}. Let $\hat{p}_{\text{data}}$ be the empirical distribution of the data defined by

$$\hat{p}_{\text{data}}(\mathbf{x}) = \frac{1}{l}\mathbb{I}_{\mathbb{X}}(\mathbf{x}). \tag{2.51}$$

Since (2.50) is invariant to changes in scale, we can divide the right-hand side by l to obtain

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\text{data}}} \left[\log p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})\right]. \tag{2.52}$$

Now observe the KL divergence between $\hat{p}_{\text{data}}$ and p_{model}:

$$D_{\text{KL}}(\hat{p}_{\text{data}} \| p_{\text{model}}) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\text{data}}} \left[\log \hat{p}_{\text{data}}(\mathbf{x}) - \log p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})\right]. \tag{2.53}$$

Note that since the term on the left does not depend upon $\boldsymbol{\theta}$ but only on the data, minimizing (2.53) with respect to $\boldsymbol{\theta}$ is the same as maximizing (2.52).

The rightmost term of (2.53) actually has a special name: *cross entropy.* The cross entropy H between two probability distributions μ and ν generalizes the Shannon entropy (2.41) and is given by

$$H(\mu, \nu) = -\mathbb{E}_{\mathbf{x}\sim\mu} \left[\log \nu(\mathbf{x})\right] = H(\mu) + D_{\text{KL}}(\mu \| \nu), \tag{2.54}$$

where we can see that $H(\mu) = H(\mu, \mu)$. Therefore, minimizing the negative log-likelihood is equivalent to minimizing the cross entropy between the empirical distribution and the model. Many objectives can be placed into this framework. For example, mean squared error (or sum of squared errors) is the cross entropy between a Gaussian model and the empirical distribution.

2.2.3 Expectation-Maximization

An *expectation-maximization (EM) algorithm* [30] is an iterative method to obtain maximum likelihood estimates of a statistical model's parameters $\boldsymbol{\theta}$ given a set of data $\mathbb{X}$ generated from the model and *latent* (unobserved) data $\mathbf{Z}$. The method consists of repeating a two-step procedure until convergence. The first step consists of calculating the expectation of the log-likelihood $L(\boldsymbol{\theta}; \mathbb{X}, \mathbf{Z})$ with respect to the latent variables given the data and current parameter estimates $\boldsymbol{\theta}^{(t)}$. The second step consists of calculating $\boldsymbol{\theta}^{(t+1)}$ by maximizing the expected log-likelihood. Both steps can be concisely represented by the following equation

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{Z}|\mathbb{X},\boldsymbol{\theta}^{(t)}} \left[L(\boldsymbol{\theta}; \mathbb{X}, \mathbf{Z}) \right], \tag{2.55}$$

where $\mathbb{E}_X [Y]$ denotes the expected value of Y with respect to X. Practical implementations sometimes consist of calculating the mode $\mathbf{Z}'$ of $\mathbf{Z}|\mathbb{X}, \boldsymbol{\theta}^{(t)}$ and maximizing with respect to $L(\boldsymbol{\theta}; \mathbb{X}, \mathbf{Z}')$, which is sometimes called a "hard" EM algorithm. Computing the full expectation with respect to all possible values of $\mathbf{Z}$ on the other hand is sometimes known as a "soft" EM algorithm. Both variants are guaranteed to converge to at least locally optimal values of the parameters, although the soft variant is likely to be better. The standard k-means clustering algorithm [70] is an example of hard EM

in which $\boldsymbol{\theta}$ represents the cluster means, $\mathbb{X}$ is a collection of points, and $\mathbf{Z}$ determines the cluster from which each point was drawn.

2.2.4 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a family of techniques used to sample from difficult or intractible probability distributions. MCMC works by constructing a Markov chain with equilibrium distribution equal to some target density function (see below for more details). Many MCMC methods work even if the user only knows a function proportional to the true density. Some of the challenge comes from constructing a suitable Markov chain, but the majority of the challenge is ensuring that the chain reaches equilibrium in a reasonable amount of time. We review here the definition and qualities of a Markov chain that are required for MCMC and summarize the Metropolis-Hasting algorithm, which is a relatively simple, widely applicable MCMC algorithm for constructing a suitable Markov chain. See the technical report by Neal [101] for more information.

Recall that a *Markov chain* is a series of random variables X_0, X_1, X_2, ... with the same state space or domain where the t-th variable is only dependent upon the immediately preceding one, i.e.

$$p(X_t \mid X_{t-1}, \ldots, X_2, X_1, X_0) = p(X_t \mid X_{t-1}). \tag{2.56}$$

A Markov chain is completely defined by the initial marginal distribution $p_0(x)$ of X_0 and the conditional probability $T_t(x, x')$ of transitioning from x to x' at time t. A chain is homogenous if T_t is the same for all t. Assume that the state space is discrete and finite (not necessary, but it simplifies the following equations and definitions). The

marginal probability of X_t may then be represented by a vector $\mathbf{p}_t$, and the transition probabilities may be represented by a stochastic (each element is non-negative and each row sums to 1) matrix T_t where each row is the conditional distribution for a specific element. We then have

$$\mathbf{p}_{t+1} = T_t^{\intercal}\mathbf{p}_t = \left(\prod_{i=1}^{t} T_i^{\intercal}\right)\mathbf{p}_0, \tag{2.57}$$

where the product is understood to left-multiply as i increases. A distribution π (represented by vector $\boldsymbol{\pi}$) is invariant with respect to the chain if for all t

$$\boldsymbol{\pi} = T_t^{\intercal}\boldsymbol{\pi}. \tag{2.58}$$

If the chain satisfies the detailed balance condition for any choice of x and x', i.e.

$$\pi(x)T_t(x, x') = \pi(x')T_t(x', x), \tag{2.59}$$

then π is an invariant distribution. We also need the Markov chain to be ergodic, i.e. p_t needs to converge to an invariant distribution—called the equilibrium distribution—as t grows regardless of the initial choice of p_0. For homogenous chains, one finds that the chain is ergodic with respect to an invariant distribution π if the probability of transitioning from any state x to any $x' \in \text{supp}\,(\pi)$ is strictly greater than zero, or

$$\min_{x} \min_{x' \in \text{supp}(\pi)} T(x, x') > 0. \tag{2.60}$$

One therefore just needs to satisfy (2.59) and (2.60) to construct a valid Markov chain for MCMC.

The *Metropolis-Hastings algorithm* provides a generic framework for constructing valid Markov chains. The algorithm requires a function μ proportional to the desired distribution as well as specification of a proposal distribution $Q(x|y)$ that can

be used to suggest a candidate sample x_t^- at each iteration t. Note that a means to sample from Q is practically a corequisite. Given an initial sample x_0, the algorithm proceeds by repreating the following steps at each iteration t:

1. Sample $x_t^- \sim Q(x_t^- \mid x_{t-1})$.
2. Calculate the acceptance ratio

$$\alpha = \min\left\{1, \frac{\mu(x_t^-)}{\mu(x_t)} \frac{Q(x_{t-1} \mid x_t^-)}{Q(x_t^- \mid x_{t-1})}\right\}. \tag{2.61}$$

3. Accept $x^t = x_t^-$ with probability α. Otherwise, $x_t = x_{t-1}$.

If Q is symmetric, i.e. $Q(x|y) = Q(y|x)$, then α simplifies somewhat and the algorithm is usually just referred to as the Metropolis algorithm. The intuition of the algorithm is that it attempts to randomly move about the sample space with a low probability of falling "downhill" to a low density area and a guarantee to move "uphill" when the option presents itself. As a result, we spend little time in low density areas and a more time in high density areas, with α ensuring that the relative amount of time remains proportional to μ. Choosing an appropriate proposal density is the primary challenge.

A common, though not entirely justified in theory, post-processing operation known as *burn-in* is to ignore the first m samples based on the assumption that the Markov chain has not converged in the first m steps and therefore these samples do not represent the target distribution. An appropriate burn-in time must be determined from experience or various heuristics if burn-in is used at all. For example, starting the Markov chain from a mode or otherwise high-density region may render burn-in moot. One should always desire that the Markov chain exhibits the rapid mixing property,

which basically states that it reaches equilibrium quickly with high probability. Proving that a given chain is rapidly mixing, however, is a challenging problem.

2.3 Constrained Optimization

In this section, we review the basics of constrained optimization. Please refer to the text by Griva *et al.* [50] for a more comprehensive introduction. Solving a constrained optimization problem usually necessitates the introduction of *Lagrange* or *Karush-Kuhn-Tucker (KKT) multipliers*, which are additional variables that represent activation of the constraints. To simplify further discussion, we will assume that all constraints are linear, i.e. we wish to solve

$$\begin{aligned} \underset{(\mathbf{x})}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b}, \end{aligned} \tag{2.62}$$

which is known as the *primal* problem. For each inequality constraint $\mathbf{a}_i{}^{\mathsf{T}}\mathbf{x} \leq b_i$, where $\mathbf{a}_i{}^{\mathsf{T}}$ is the i-th row of A, we introduce a non-negative KKT multiplier λ_i. For equality constraints, an unconstrained multiplier is introduced. The objective function $f(\mathbf{x})$ is then replaced with the *Lagrangian*

$$L_P(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathsf{T}}(A\mathbf{x} - \mathbf{b}). \tag{2.63}$$

This problem is directly related to what is known as the Lagrangian *dual*

$$\begin{aligned} \underset{(\boldsymbol{\lambda})}{\text{maximize}} \quad & \inf_{(\mathbf{x})} \left[f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathsf{T}}(A\mathbf{x} - \mathbf{b}) \right] \\ \text{subject to} \quad & \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned} \tag{2.64}$$

A concept known as strong duality states that if f is convex and there exists at least one point that satisfies the constraints, then the value of the objective functions of (2.62)

and (2.64) are equal at their optimal solutions $\mathbf{x}^*$ and $\boldsymbol{\lambda}^*$. A direct consequence of this fact is the condition of *complementary slackness*, which states that

$$\lambda_i(\mathbf{a}_i^\intercal \mathbf{x} - b_i) = 0 \tag{2.65}$$

or more concisely $\boldsymbol{\lambda} \odot (A\mathbf{x} - \mathbf{b}) = \mathbf{0}$. Convexity is not necessary for strong duality to hold, but it is an otherwise useful property in that it guarantees global optimality of a locally optimal solution. Note that weak duality, which states that the primal objective is always greater than or equal to the dual objective, always holds between a primal minimization and dual maximization problem regardless of convexity.

A couple of examples of convex optimization follow. A linear program is a convex problem that can be written in the canonical form

$$\begin{aligned} \underset{(\mathbf{x})}{\text{minimize}} \quad & \mathbf{c}^\intercal \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2.66}$$

In other words, a linear program is a constrained optimization problem in which the objective and all of the constraints are linear. A quadratic program differs from a linear program by adding a quadratic form to the objective and has canonical form

$$\begin{aligned} \underset{(\mathbf{x})}{\text{minimize}} \quad & \frac{1}{2}\mathbf{x}^\intercal H \mathbf{x} + \mathbf{c}^\intercal \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b}. \end{aligned} \tag{2.67}$$

The matrix H is the Hessian, and the problem is convex if and only if H is positive semi-definite.

2.4 Optimal Transport

Optimal transport is the name given to the study of the optimal transportation or allocation of resources. One of the most important topics in optimal transport is the *Wasserstein distance*. Consider two probability distributions on a metric space (M, d) with finite p-th moments, $p \in [1, \infty)$, and probability density functions given by $\mu : M \to \mathbb{R}$ and $\nu : M \to \mathbb{R}$. The p-th Wasserstein distance between μ and ν is given by

$$\mathcal{W}_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{M \times M} d^p(x, y) \mathrm{d}\gamma(x, y) \right)^{(1/p)}, \tag{2.68}$$

where $\Gamma(\mu, \nu)$ is the collection of all joint distributions on $M \times M$ with marginals μ and ν [137]. Note that the p-th Wasserstein distance can also be expressed in terms of the joint distribution that minimizes the expectation

$$\mathcal{W}_p^p(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu,\nu)} \mathbb{E}_{(X,Y)\sim\gamma} \left[d^p(X, Y) \right]. \tag{2.69}$$

The Wasserstein distance can be interpreted as the minimum cost required to transform μ into ν or vice versa. If we consider μ and ν to represent piles of dirt, then we see the intuition behind one of the Wasserstein distance's commonly known other names: the *earth mover's distance (EMD)*. Under the moniker EMD—first used in print by Rubner *et al.* [116]—the metric has been applied in computer vision for comparing color distribution or texture histograms of images for content based image retrieval [20, 116, 82, 105, 106].

The Wasserstein distance, however, has a much longer history than its use in computer vision would imply. Gaspard Monge [98] originally laid the groundwork for EMD, and the problem was reformulated in the mid-20th century by Leonid

Kantorovich [68, 69]. Thus does $\mathcal{W}_p$ receive another name, the Monge-Kantorovich mass transportation distance, under which it is applied in economics, fluid mechanics, meteorology, and partial differential equations (PDEs) [36, 41]. In statistics, the metric may also be known as the Mallows distance [76]. The Wasserstein distance is also used as a means of evaluating the performance of multiple-object trackers and filters [58, 113, 119]. Other names not listed still exist, and for further information and a more comprehensive survey of the Wasserstein distance's history and optimal transport in general, the reader is referred to Vershik's article [138] and Villani's texts [139, 140].

We now turn our focus towards EMD, which often takes a discrete (i.e. countable) form. While the choice of the metric space (M, d) can have significant implications on the existence and feasibility of computing $\mathcal{W}_p$, the choice has less severe implications when M is discrete as the solution depends on only one algorithm regardless of the choice of d. Let the term "ground distance" refer to the metric d. Application of EMD requires specification of a ground distance and computation of the *flow* $f(a, b)$ of mass from $x \in \text{supp}(\mu)$ to $y \in \text{supp}(\nu)$. EMD is then calculated as the cost of the minimum-cost maximum flow and is defined to be the solution of the linear program

$$EMD(\mu, \nu) = \mathcal{W}_p^p(\mu, \nu) = \min_{(f)} \sum_{x \in \text{supp}(\mu)} \sum_{y \in \text{supp}(\nu)} f(x, y) d^p(x, y) \tag{2.70}$$

subject to the constraints

$$\sum_{y \in \text{supp}(\nu)} f(x, y) \leq \mu(x), \tag{2.71}$$

$$\sum_{x \in \text{supp}(\mu)} f(x,y) \leq \nu(y), \tag{2.72}$$

$$\sum_{x \in \text{supp}(\mu)} \sum_{y \in \text{supp}(\nu)} f(x,y) = \min \left\{ \sum_{x \in M} \mu(x), \sum_{y \in M} \nu(y) \right\}. \tag{2.73}$$

The solution to this problem can be found in $O(n^3 \log n)$ time, where n is the larger cardinality of each measure's support. Several observations can be made from this linear program. First, the value of p is irrelevant in computing its solution; one could just as well use $D = d^p$ (hence the lack of a p subscript or superscript in $EMD(\mu, \nu)$). In fact, d does not need to be a metric for the solution to exist and be computable in polynomial time. Furthermore, μ and ν do not actually need to be probability measures and may have different total masses as hinted by (2.73). The downside of allowing arbitrary masses is that certain properties of EMD no longer hold. For example, EMD is a metric on $\mathcal{P}(X)$ if d^p is a metric on X [116], but EMD is not a metric on $\mathcal{F}(X)$. In the special case that $\mu(x) = 1$ for each $x \in \text{supp}(\mu)$ and $\nu(y) = 1$ for each $y \in \text{supp}(\nu)$, then EMD is also known as the assignment problem and can be solved in $O(n^3)$ time [34]. As is often the case with the assignment problem, the flow f is sometimes the variable of interest rather than the actual minimum cost.

EMD is usually assumed to possess a Euclidean ground distance, but examples of other ground distances exist in the literature. Igbida *et al.* [62] study EMD in the context of PDEs with a discretized version of the Euclidean ground distance rounded up to the nearest whole number. Ling and Okada [82] proposed an efficient tree-based algorithm for computing EMD with a Manhattan ground distance, and Pele and Werman [106] explored the effect of applying a threshold to various ground distances

and its impact on computation time and accuracy. In the realm of image retrieval, EMD is often applied as a metric for nearest neighbor searches.

The computational complexity of EMD is often a hindrance to applying it in large problems. The fastest known algorithm to compute EMD exactly (up to precision) for a general cost function is $O(n^3 \log n)$ [104]. However, approximate algorithms have been introduced in recent years that are linear in complexity. The first such algorithm, the computation of Sinkhorn distances introduced by Cuturi [24], adds an entropic term that regularizes the objective and makes it solvable numerically via a simple iterated procedure. To be precise, a constraint is placed on the entropy of the flow so that it yields the dual

$$EMD(\mu, \nu, \lambda) = \min_{f} \sum_{x \in \text{supp}(\mu)} \sum_{y \in \text{supp}(\nu)} f(x, y) \left[d^p(x, y) - \lambda \log f(x, y) \right]. \tag{2.74}$$

Sinkhorn distances parallelize easily, which is a significant advantage over other algorithms. Convolutional Wasserstein distances, introduced by Solomon *et al.* [127], improve on Cuturi's Sinkhorn distance by removing the need to compute pairwise distances $d^p(x, y)$. Instead, the convolutional Wasserstein distance algorithm exploits the relationship between the heat kernel and the geodesic distance $g : M \times M \to \mathbb{R}$ on a manifold M, where the geodesic distance is the shortest path possible between two points on the manifold and the heat kernel $\mathcal{H}_t : M \times M \to \mathbb{R}$ solves the heat equation $\partial_t f_t = \Delta f_t$ with initial condition $f_0 : M \to \mathbb{R}$ via

$$f_t(x) = \int_M f_0(y) \mathcal{H}_t(x, y) \mathrm{d}y. \tag{2.75}$$

The exploited relationship is Varadhan's formula [134], which states that as the time goes to zero, the geodesic distance may be recovered from the heat kernel

$$g^2(x,y) = \lim_{t\to 0} -2t \log \mathcal{H}_t(x,y). \tag{2.76}$$

The intuition behind the formula arises from the fact that the heat equation models the diffusion of many particles taking random walks and that as time approaches zero, the particles have had less and less opportunity to deviate from the geodesic [22]. Of course, the heat kernel must exist and be known as a necessary precondition, and therefore the algorithm is primarily useful for only geometric domains such as shape interpolation or color manipulation in image processing.

2.5 Filtering

This section introduces the concept of filtering. Filtering solves the problem of estimating the state of a dynamic process observed through a noisy signal. The exact algorithm used to filter the signal depends on the application, and certain trade-offs between efficiency and optimality may be necessary. We focus on the Kalman filter and some of its relatives, outlining the theory behind their derivation and use.

2.5.1 The Kalman Filter

The *Kalman filter* [67] is a recursive two-step procedure for obtaining state estimates of a Markov process that is observed through intermittent and possibly incomplete measurements. Generally, the model for the state's evolution over time must be known disregarding noise. In fact, for the standard Kalman filter (numerous extensions exist including but not limited to the extended [151] and unscented Kalman

filters [147]), the process must be linear and described by an equation of the form

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \mathbf{u}_k + \mathbf{w}_k, \tag{2.77}$$

where $\mathbf{x}_k$ is the state vector at time k, A_k is the transition matrix that describes the state dynamics, B_k is a matrix that relates some external control input $\mathbf{u}_k$ to the state, and $\mathbf{w}_k$ is zero-mean Gaussian noise with covariance Q_k, known as the process noise covariance. The state is observed through measurements $\mathbf{y}_k$, which are related to the state according to

$$\mathbf{y}_k = H_k \mathbf{x}_\mathbf{k} + \mathbf{v}_k, \tag{2.78}$$

where H_k is the measurement matrix and $\mathbf{v}_k$ is zero mean Gaussian noise with covariance R_k, known as the measurement noise covariance. Neither the process nor measurement noises at different times are correlated, i.e. $\mathrm{Cov}(\mathbf{w}_k, \mathbf{w}_l) = \mathrm{Cov}(\mathbf{v}_k, \mathbf{v}_l) = \mathbf{0}$ for $k \neq l$.

The objective of the Kalman filter is to minimize the error between the state estimate $\hat{\mathbf{x}}$ and the true state $\mathbf{x}$, and it does so by minimizing the estimated error covariance

$$\hat{P}_k = \mathbb{E}\left[(\hat{\mathbf{x}}_k - \mathbf{x}_k)(\hat{\mathbf{x}}_k - \mathbf{x}_k)^\intercal\right]. \tag{2.79}$$

The filter alternates between calculating *a priori* estimates (predictions of the state and its error) and *a posteriori* estimates (corrections made upon observing the measurement). The prediction is based upon the transition function and is given by

$$\hat{\mathbf{x}}_k^- = A_k \hat{\mathbf{x}}_{k-1}, \tag{2.80}$$

$$\hat{P}_k^- = A_k \hat{P}_{k-1} A_k^\intercal + Q_k, \tag{2.81}$$

where $\hat{\mathbf{x}}_k^-$ and $\hat{P}_k^-$ are the *a priori* estimates at time k and $\hat{\mathbf{x}}_{k-1}$ and $\hat{P}_{k-1}$ are the *a posteriori* estimates of the previous timestep. A correction based upon the measurement $\mathbf{y}_k$ is applied to obtain the *a posteriori* estimates

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + K_k(\mathbf{y}_k - H_k\hat{\mathbf{x}}_k^-), \tag{2.82}$$

$$\hat{P}_k = (\mathbf{I} - K_k H_k)\hat{P}_k^-, \tag{2.83}$$

where K_k is the *Kalman gain* used to weight the prediction and measurement and is calculated as

$$K_k = \hat{P}_k^- H_k^\intercal (H_k \hat{P}_k^- H_k^\intercal + R_k)^{-1}. \tag{2.84}$$

The Kalman gain in (2.84) is optimal in the sense that it minimizes the trace of $\hat{P}_k$. Note that (2.83) is only valid for the optimal Kalman gain. A more numerically stable version that is valid for any value of K_k is given by

$$\hat{P}_k = (\mathbf{I} - K_k H_k)\hat{P}_k^- (\mathbf{I} - K_k H_k)^\intercal + K_k R K_k^\intercal. \tag{2.85}$$

In fact, minimizing the trace of the right-hand side of (2.85) with respect to K_k yields the optimal Kalman gain. For a linear transition and measurement function with Gaussian additive noise as defined above, the Kalman filter is optimal in the mean squared error sense. In particular, the Kalman filter provides the minimum variance unbiased estimator of the state $\mathbf{x}$ assuming an unbiased initial estimate $\hat{\mathbf{x}}_0$.

2.5.2 Bayes Filters

The Kalman filter may also be considered a special case of a *recursive Bayesian* (or *Bayes*) *filter* [4]. A recursive Bayesian filter performs similar prediction and correction steps, but instead of maintaining an estimate of the state mean and error

covariance, such a filter maintains the probability distribution of the state when considered as a random variable. Let $\mathbf{y}_{1:k}$ be shorthand for the first k measurements. The prediction equation is given by

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}) \mathrm{d}\mathbf{x}_{k-1}, \tag{2.86}$$

which can be derived using Bayes' rule on the condition that $\mathbf{x}$ is a Markov process. The correction or update equation is given by

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})}{\int p(\mathbf{y}_k \mid \mathbf{x}) p(\mathbf{x} \mid \mathbf{y}_{1:k-1}) \mathrm{d}\mathbf{x}}, \tag{2.87}$$

which also largely follows from Bayes' rule. Although theoretically optimal, Bayes filters are often intractable. The Kalman filter is an exception. One can check that a Kalman filter is a Bayes filter by using the relations

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) \sim \mathcal{N}(\hat{\mathbf{x}}_k^-, \hat{P}_k^-), \tag{2.88}$$

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \sim \mathcal{N}(\hat{\mathbf{x}}_k, \hat{P}_k). \tag{2.89}$$

2.5.3 Extended Kalman Filter

The *extended Kalman filter (EKF)* [151] is a heuristic applied when the transition or measurement equations do not satisfy normal linear assumptions. In other words, (2.77) and/or (2.78) are replaced with

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k), \tag{2.90}$$

$$\mathbf{y}_k = h(\mathbf{x}_k, \mathbf{v}_k), \tag{2.91}$$

where f and h are presumed to be differentiable functions. The extended Kalman filter operates by replacing (2.80) and (2.81) with

$$\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, \mathbf{0}), \tag{2.92}$$

$$\hat{P}_k^- = A_k \hat{P}_{k-1} A_k^\intercal + W_k Q_k W_k^\intercal, \tag{2.93}$$

and replacing (2.82) and (2.84) with

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + K_k(\mathbf{y}_k - h(\hat{\mathbf{x}}_k^-, \mathbf{0})), \tag{2.94}$$

$$K_k = \hat{P}_k^- H_k^\intercal (H_k \hat{P}_k^- H_k^\intercal + V_k R_k V_k^\intercal)^{-1}, \tag{2.95}$$

where A_k, H_k, W_k, and V_k are linearizations of f and h such that

$$A_k = \frac{\partial f}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, \mathbf{0}), \tag{2.96}$$

$$H_k = \frac{\partial h}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k^-, \mathbf{0}), \tag{2.97}$$

$$W_k = \frac{\partial f}{\partial \mathbf{w}_k}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, \mathbf{0}), \tag{2.98}$$

$$V_k = \frac{\partial h}{\partial \mathbf{v}_k}(\hat{\mathbf{x}}_k^-, \mathbf{0}). \tag{2.99}$$

One can see that the EKF is only a first order approximation. Consequently, the EKF is not an optimal estimator in any sense unless the transition and measurement functions are linear and it reduces to the standard filter. Higher order versions based upon successive terms of the Taylor series expansions of f and h are possible, but are not typical and are not guaranteed to provide a significant benefit despite the increased computational burden.

2.5.4 Other Filters

Numerous filters have been proposed over the years for a diverse array of applications. The unscented Kalman filter [147] is a direct alternative to the EKF that applies a deterministic sampling called the unscented transform to the transition function that is able to preserve the mean and covariance of the process regardless of the nature of the transition. As a result, the unscented Kalman filter does not require one to compute partial derivatives for linearization nor even a differentiable transition. Nondeterministic Monte Carlo sampling leads particle filters. For the purpose of multi-target tracking (e.g. radar systems), the problem is complicated by uncertain associations between measurement and state variables. The joint probabilistic data association (JPDA) [39] filter, probability hypothesis density (PHD) filters [142], and relatives such as that given by Vo and Vo [143] can be used as heuristics to solve the problem. A JPDA filter aggregates measurements based upon possible assignments between observed and predicted targets. Since the number of assignments is combinatorial, the JPDA filter suffers some practical drawbacks. The PHD filter avoids sampling and combinatorics by estimating the intensity of the target locations instead of the actual locations, where the intensity is a function giving the expected number of targets within a given volume. The intensity is normally modeled with a Gaussian mixture model, and heuristics regularly prune components with low weight in order to avoid an unbounded growth in their number. In Chapter 4, we use a modified Kalman filter similar to the JPDA in order to estimate the positions and correlations of stationary targets.

2.6 Support Vector Machines

A *support vector machine (SVM)* is a binary *classifier* that assigns positive or negative labels to a set of instances by calculating an optimal separating hyperplane that separates the positive and negative instances. In the event that the instances are not separable, then penalties for each misclassification are incorporated into the calculation of the hyperplane. SVMs are an example of a kernel method (see Section 2.1.4). The primary advantage of SVMs is that they are theoretically well-founded. Provided with a PD kernel and non-negative misclassification cost C, an SVM finds the globally optimal solution to a quadratic programming problem.

Let us derive the quadratic program with a linear kernel (see Burges [13] for a more detailed derivation) as it gives some insight into how SVMs operate. Consider a set of data $\mathbb{X}$ with l elements a set of labels (represented by a vector) $\mathbf{y} \in \{-1, 1\}^l$ for each element of $\mathbb{X}$. Without loss of generality, assume that all of the elements of $\mathbb{X}$ are vectors $\mathbf{x}_i \in \mathbb{R}^n$ for some $n > 0$ and $i \in [1, l]$. A label y_i indicates whether a data point belongs to the positive or negative group. Our goal is to determine a hyperplane that separates the positive and negative groups of points with the widest margin possible, where the margin is defined to be the shortest perpendicular distance from the hyperplane to any $\mathbf{x}_i \in \mathbb{X}$. If the groups cannot be separated by a hyperplane, then we want to minimize the number of violations. A hyperplane can be defined by a vector $\mathbf{w}$ normal to its surface and a scalar b that offsets the plane from the origin. The (signed) distance of any point $\mathbf{x}$ from the hyperplane is given by $(\mathbf{w}^\intercal \mathbf{x} + b)/\|\mathbf{w}\|$. We require that members of the positive or negative group lie on the positive or negative

side of the plane. This constraint can be represented for all i by

$$y_i(\mathbf{w}^\intercal \mathbf{x} + b) - 1 + \xi_i \geq 0, \tag{2.100}$$

where $\xi_i \geq 0$ is the error incurred by any point $\mathbf{x}_i$ on the wrong side of the margin. Any point with $\xi_i > 1$ is on the wrong side of the hyperplane. Note that points satisfying equality in (2.100) with $\xi_i = 0$ lie on the margin, which means that the margin has a magnitude of $1/\|\mathbf{w}\|$. Points that lie on (or within) the margin are called support vectors. See Figure 2.1 for a visualization. Maximizing the margin therefore corresponds to minimizing $\|\mathbf{w}\|^2$. We also wish to minimize the cost of misclassifications given by $C\sum_i \xi_i$, where C is chosen beforehand.

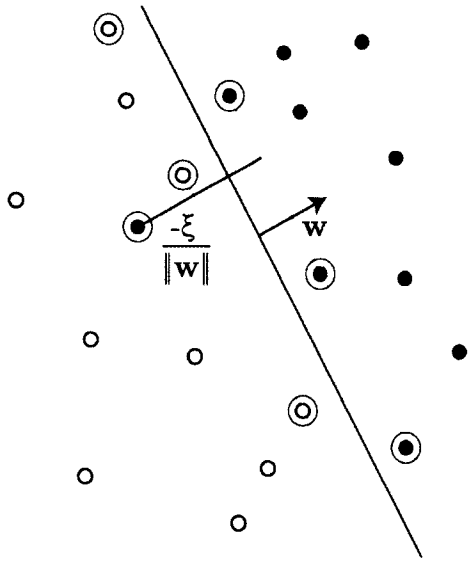


Figure 2.1: An illustration of a separating hyperplane for a non-separable problem. Support vectors are circled. Note that the error ξ is with respect to the margin for the side of the plane on which the point should ideally be located.

Combining both objectives and introducing Lagrange multipliers for each constraint yields the desired Lagrangian

$$L_P = \frac{1}{2}\mathbf{w}^\intercal \mathbf{w} + C\sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i[y_i(\mathbf{w}^\intercal \mathbf{x} + b) - 1 + \xi_i] - \sum_{i=1}^{l} \mu_i \xi_i, \tag{2.101}$$

where $\alpha_i \geq 0$ and $\mu_i \geq 0$ are the Lagrange multipliers for the inequality constraints given by (2.100) and $\xi_i \geq 0$. Recall the definition of the dual (2.64) as a maximization with respect to the Lagrange multipliers with the constraint that the partial derivatives of the Lagrangian with respect to the primal variables must be zero. Let us therefore note that

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \implies \mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i, \tag{2.102}$$

$$\frac{\partial L_P}{\partial b} = 0 \implies \sum_{i=1}^{l} \alpha_i y_i = 0, \tag{2.103}$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 \implies C = \alpha_i + \mu_i. \tag{2.104}$$

Applying these equality constraints to the Lagrangian yields the dual objective

$$L_D = \sum_{i=1}^{l} \alpha_i - \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x}_i{}^{\intercal} \mathbf{x}_j \tag{2.105}$$

subject to $0 \leq \alpha_i \leq C$. The dual is remarkably easier to solve as it has fewer variables and simpler constraints, and general or special purpose methods may be used to find the solution.

A more significant observation is the fact that the data only appears in the dual in the form of dot products, which means that one can replace these dot products with kernel evaluations, i.e.

$$L_D = \sum_{i=1}^{l} \alpha_i - \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{2.106}$$

where K is a PD kernel and $\mathbf{x}_i$ no longer needs to be an element of $\mathbb{R}^n$. A new data point $\mathbf{z}$ is classified (assigned a label y_z) according to the side of the hyperplane on

which it falls, which can be determined by

$$y_z = \text{sign}\left(\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{z}) + b\right). \tag{2.107}$$

One can see then that SVMs can be expensive to operate depending on the number of support vectors and the complexity of the kernel. Note that if the kernel K is not PD, then the solution may only be locally optimal. Many interesting problems are characterized by indefinite kernels, and learning SVMs with indefinite kernels is an active research area. See Chapter 5 for an example of two indefinite kernel techniques.

We state several facts about the solution. First, due to complementary slackness,

$$\alpha_i[y_i(\mathbf{w}^\intercal \mathbf{x} + b) - 1 + \xi_i] = 0, \tag{2.108}$$

$$\mu_i \xi_i = 0. \tag{2.109}$$

Consequently, $\alpha_i > 0$ only for support vectors, and $\alpha_i < C$ if and only if $\xi_i = 0$ (and $\alpha_i = C$ if and only if $\xi_i \neq 0$). This fact yields another interpretation for the term support vector since support vectors are simply those that have nonzero or non-null support over α considered as a domain. Any vector that is not a support vector has no effect on the solution and can be removed from further calculations. Given the solution of the dual $\boldsymbol{\alpha}^*$, we can also calculate $\mathbf{w}$ using (2.102) and

$$b = y_i - \sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j{}^\intercal \mathbf{x}_i \tag{2.110}$$

for any i such that $0 < \alpha_i < C$. If the data is separable, then C and ξ_i can be ignored and the primary difference is that each α_i is no longer bounded above by C.

2.7 Neural Networks and Deep Learning

An *artificial neural network (ANN)* or simply neural network is a computational paradigm designed to emulate the biological neural network found in organisms. In a neural network, nonlinear functions emulate the behavior (activity) of neurons, and scalar weights emulate the connectivity strength of synapses between the neurons. Despite this layer of abstraction, neural networks can often be concisely represented as functions of matrices and vectors. Many types of neural networks exist, including *multi-layer perceptrons (MLPs)*, *recurrent neural networks (RNNs)*, and *convolutional neural networks (CNNs)*, which we will introduce in the following subsections, and they have been successfully used in a wide variety of applications including speech recognition [49], computer vision [72], and nonlinear control [77]. More advanced neural networks including restricted Boltzmann machines and deep belief networks (see Bengio [6]) are not covered. In the final subsection, we review the basics of *deep learning*, which may be considered a collection of best practices for ANNs. Unless otherwise stated, the main reference for this section is Goodfellow *et al.* [44], which gives the background for an understanding of the current state of the art in neural networks and deep learning. The reader is referred to Jain *et al.* [64] for a simpler and much smaller in scope reference.

2.7.1 Multi-layer Perceptrons

The MLP or *feed-forward* neural network is perhaps the simplest and most widely used form of ANN. In an MLP, neurons (or *nodes*) are organized into n distinct layers with directed connections only from neurons in layer i to neurons in layer $i+1$.

See Figure 2.2 for an example of a network with one hidden layer. The symbols in the figure are defined in the next two paragraphs.

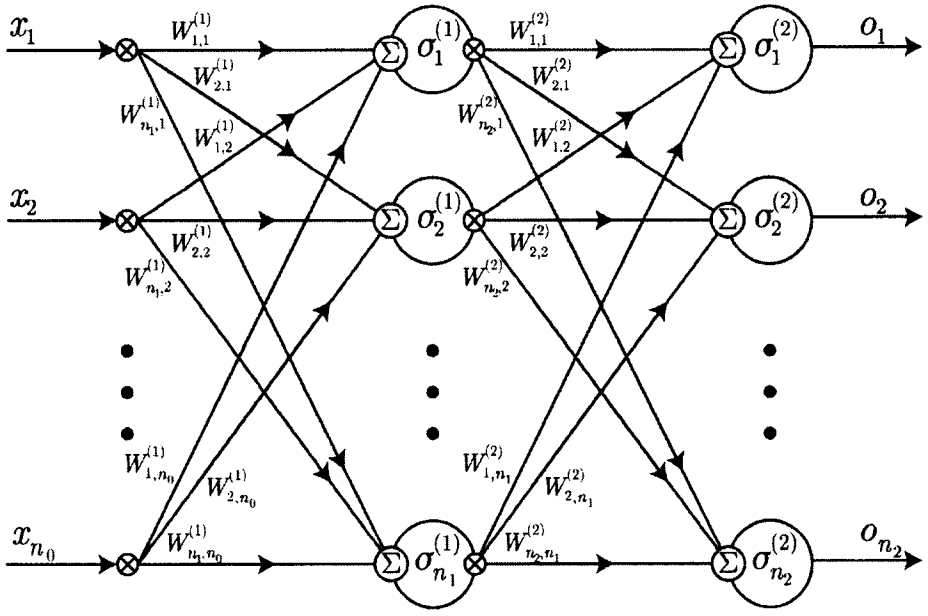


Figure 2.2: A detailed illustration of a multilayer perceptron with input **x**, output **o**, and two layers, one of which is hidden. The arrows show the flow of input from left to right. During back-propagation in training, the gradient of the error flows backwards from right to left. The bias is not shown.

Let n_i be the number of neurons in layer i with the input considered layer 0. Let $W^{(i)} = \left[w_{jk}^{(i)}\right]$ represent the connections between layers i and $i+1$, where $w_{jk}^{(i)}$ is the weight of the connection between the j-th neuron of layer i and the k-th neuron of layer $i+1$. In addition, let the n_i-dimensional vector-valued function $\sigma^{(i)} = \left[\sigma_j^{(i)}\right]$ be the activation function of layer i, where $\sigma_j^{(i)}$ represents the scalar activation function of neuron j in layer i, i.e. for a given n_i-dimensional vector $\mathbf{v}$

$$\sigma^{(i)}(\mathbf{v}) = \left[\sigma_1^{(i)}(v_1), \ldots \sigma_j^{(i)}(v_j), \ldots \sigma_{n_i}^{(i)}(v_{n_i})\right]^{\mathsf{T}}. \tag{2.111}$$

The input of $\sigma^{(i)}$ is found from the output of the previous layer multiplied by the appropriate connection weights. Each layer may also have an optional bias $\mathbf{b}^{(i)}$ that is

added to its input. The output $\mathbf{z}_i$ of layer i is then given by

$$\mathbf{z}_i = \sigma^{(i)}(W^{(i)}\mathbf{z}_{i-1} + \mathbf{b}^{(i)}), \tag{2.112}$$

$$= \sigma^{(i)}(W^{(i)}\sigma^{(i-1)}(W^{(i-1)}\mathbf{z}_{i-2} + \mathbf{b}^{(i-1)})), \tag{2.113}$$

$$= \sigma^{(i)}(W^{(i)}\sigma^{(i-1)}(W^{(i-1)}\ldots(W^{(2)}\sigma^{(1)}(W^{(1)}\mathbf{x} + \mathbf{b}^{(1)})))), \tag{2.114}$$

where $\mathbf{x}$ is an input vector provided to the first (input) layer of neurons.

The weights and biases of a network may be simultaneously learned using the back-propagation algorithm, which is effectively just an efficiently organized application of the chain rule from calculus followed by gradient descent, i.e. the updated value of a weight ${}^*w_{jk}^{(i)}$ is given by

$${}^*w_{jk}^{(i)} = w_{jk}^{(i)} - \alpha\frac{\partial\Omega(\mathbf{o})}{\partial w_{jk}^{(i)}}, \tag{2.115}$$

where α is a learning rate and Ω measures the error of the network's output $\mathbf{o}$. For example, Ω may be the squared error of the output with respect to some target values $\mathbf{o}^*$, which results in

$$\Omega(\mathbf{o}) = \frac{1}{2}\|\mathbf{o} - \mathbf{o}^*\|^2. \tag{2.116}$$

Layers in between the input and output layer of neurons are said to be hidden. Activation functions are usually assumed to be sigmoidal functions that approximate the Heaviside step function, e.g. the logistic sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.117}$$

A powerful result [26] states that one hidden layer (with enough neurons) is sufficient to model arbitrarily complex functions. Since the input and output layers are always

presumed to exist, from this point forward let the term n-layer network refer to an ANN (not necessarily MLP) with n hidden layers.

2.7.2 Recurrent Neural Networks

An RNN allows the existence of feedback loops or connections from higher to lower layers (although it is not necessarily organized into layers). RNNs implicitly possess a sort of memory, which allows them to model time-dependent phenomena or sequences of variable length. We consider two types of RNNs. The first type is considered only for illustration and is a single layer RNN with connections from the hidden layer to itself. The second type is more advanced and employs what is known as a *gated recurrent unit (GRU).*

Consider a single layer MLP parameterized by hidden weight matrix W. An RNN extends the hidden layer with the introduction of an additional weight matrix U that is multiplied by the hidden layers previous output. More formally, let $\mathbf{x}_t$, $t \in [1, T]$ be a sequence of inputs to the network, and let $\mathbf{h}_t$ be the output of the hidden layer at time t. The hidden layer's output is computed according to

$$\mathbf{h}_t = \sigma(W\mathbf{x}_t + U\mathbf{h}_{t-1}), \tag{2.118}$$

where $\mathbf{h}_0 = \mathbf{0}$ and σ is the logistic sigmoid function (see Figure 2.3). The RNN outputs a vector at every timestep, but for certain applications only certain timesteps (e.g. the last) are of interest. The training and operation of an RNN is more difficult than that of an MLP due to either a vanishing or exploding gradient as the number of timesteps grows.

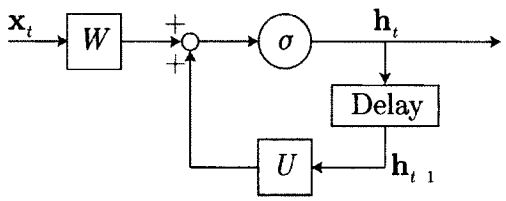


Figure 2.3: A block diagram representing a hidden layer in an RNN. The layer's output $\mathbf{h}_t$ is provided as input to the next layer, which may or may not be recurrent.

GRUs provide a solution to the gradient problem encountered by basic RNNs by constructing a more elaborate activation function. A GRUs consists of so called update and reset gate vectors $\mathbf{z}_t$ and $\mathbf{r}_t$ that control to what degree a layer's previous output $\mathbf{h}_{t-1}$ and current input $\mathbf{x}_t$ contribute to its next output $\mathbf{h}_t$. If an element of the reset gate is zero, then the corresponding element of $\mathbf{h}_{t-1}$ is ignored and that part of the network behaves as though the sequence has just started. The update gate, on the other hand, controls whether elements of $\mathbf{h}_{t-1}$ or the candidate output $\tilde{\mathbf{h}}_t$ are propagated forward in time. The update and reset gates are computed in a similar manner to the output of a traditional hidden recurrent layer, although they are parameterized by their own respective weight matrices W_z, U_z and W_r, U_r, i.e.

$$\mathbf{z}_t = \sigma(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1}), \tag{2.119}$$

$$\mathbf{r}_t = \sigma(W_r\mathbf{x}_t + U_r\mathbf{h}_{t-1}). \tag{2.120}$$

The candidate output is the traditional output modulated by the reset gate and is given by

$$\tilde{\mathbf{h}}_t = \tanh\left(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1})\right). \tag{2.121}$$

Interpolating $\mathbf{h}_{t-1}$ and $\tilde{\mathbf{h}}_t$ using the update gate provides the next output

$$\mathbf{h}_t = (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t. \tag{2.122}$$

Figure 2.4 provides a block diagram of a GRU. The reader is referred to Chung *et al.* [19] for extra details and an introduction to another type of gated RNN, the Long Short Term Memory network, which predates GRUs.

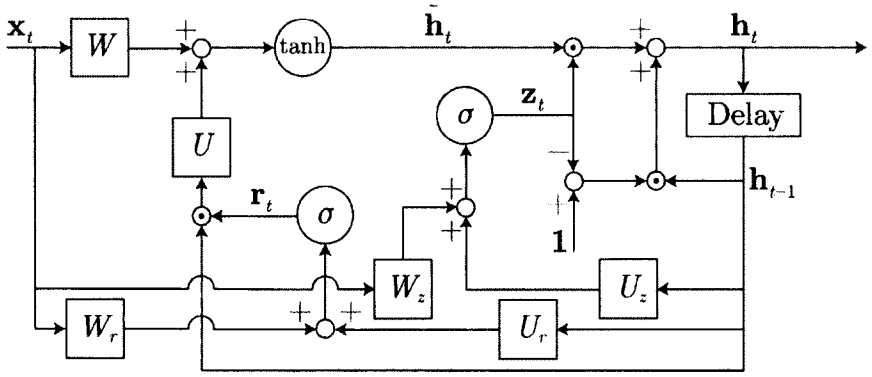


Figure 2.4: A block diagram representing a GRU. The layer's output $\mathbf{h}_t$ is provided as input to the next layer, which may or may not be recurrent.

2.7.3 Convolutional Neural Networks

A CNN is a special type of feed-forward neural network designed especially for processing images or other regular grids. Inspired by mammalian vision systems (especially cats), a CNN consists of tiled or replicated weights that connect to only local regions of the input. Weights that are tiled together as a single unit are usually referred to as a kernel, but to avoid confusion we will use the term filter. The name of this type of network comes from its similarity to the convolution operation between two functions $f, g : \mathbb{R} \to \mathbb{R}$:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(x)g(t-x)\mathrm{d}x. \tag{2.123}$$

In a sense, the input is one function and the weights of the filter are the other. If the input has more than one dimension (e.g. a two-dimensional image), then the filter is convolved in each dimension separately (see Figure 2.5). A convolutional layer is

comprised of one or more filters that are convolved with the image to produce an output layer of similar shape. Thinking of each entity as a tensor aids the interpretation and definition of equations.

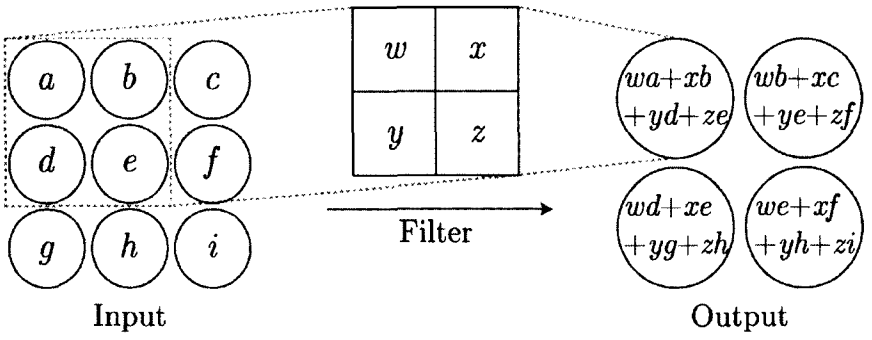


Figure 2.5: A 2 × 2 filter is convolved with a 3 × 3 input layer to produce a 2 × 2 output. The output is then given to a nonlinear activation function such as hyperbolic tangent. Note that without padding the input, the output will be smaller.

To be more precise, suppose the input is a color image represented by the tensor X where $X_{i,j,k}$ is the intensity of the red, green, or blue *channel* ($i = 1$, 2, or 3) in the j-th row and k-th column. In addition, suppose that we are using an l-channel $M \times N$ filter, where M and N are the width and height in pixels of a patch of the image on which the filter operates, and let $F_{i,j,k,l}$ represent the weight between the i-th output channel and j-th input channel at the k-th row and l-th column of the filter's input patch. Assuming the output is also organized as a two-dimensional grid, then the i-th output channel $Z_{i,j,k}$ in the j-th row and k-th column can be given by

$$Z_{i,j,k} = \sum_{l=1}^{3} \sum_{m=1}^{M} \sum_{n=1}^{N} X_{l,j+m-1,k+n-1} F_{i,l,m,n}. \tag{2.124}$$

Multiple filters may be used, and the input may consist of multiple channels at each position (e.g. red, green, and blue color channels). Note that near the borders of the input, zero-padding may be necessary in order to have the filter cover all possible

positions. Alternatively, the number of output neurons is simply shrunk based upon the width and height of the filter. Filters may also be defined that skip input in regular intervals. The *stride* of a filter is one plus the number of pixels that it skips. Note that the size of the output is reduced relative to the input by a factor equal to the stride, which can be useful in reducing the computational load of the network.

A convolutional layer is typically composed of three stages: filtering, detecting via nonlinear activations, and an important operation known as *pooling*. Filters serve as local, translation invariant feature extractors that are automatically learned during training. Increased computational efficiency resulting from the replicated weights is a beneficial side-effect. Pooling replaces the output of a layer at a certain location with an aggregate function of nearby outputs. The aggregate function may be a max function or an average, for example. Aside from reducing the size of the output, pooling makes the network largely invariant to small translations of the input. Instead of knowing precisely where a feature was located in the input, pooling informs the network that the feature exists. Pooling is also an important ingredient in making a CNN capable of handling variable-sized input. For example, regardless of the size of the image, we may choose to pool each quadrant before passing the output to a fixed-size layer.

2.7.4 Deep Learning

Deep learning is the latest name given to the branch of machine learning dominated by ANNs, which has garnered renewed interest since massively parallel architectures and large datasets have made certain challenging problems feasible.

CNNs in particular have been very successful in recent years and have played a major role in the resurgence of interest in neural networks and the development of deep learning. From the author's point of view, deep learning is best thought of as a collection of best practices for using neural networks, and we will list some of them here.

One of the most significant best practices is to treat a neural network as a probabilistic model. A neural network is parameterized by a set of weights and biases that can be collectively represented by the vector $\boldsymbol{\theta}$. Let $f(\mathbf{x};\boldsymbol{\theta})$ be the function representing the output of a neural network parameterized by $\boldsymbol{\theta}$. Since the network is a probabilistic model, we choose to train our networks using the maximum likelihood principle in order to find the maximum likelihood estimate $\hat{\boldsymbol{\theta}}$. A common objective or loss function when training the network is to minimize squared error as in (2.116). When the network is interpreted as a probabilistic model, however, this loss function is likely to be inappropriate as it places a Gaussian prior on the output (see Section 2.2.2). For example, networks used for classification are often multinomial models with a single output per class constrained to be between 0 and 1 by a *softmax* output layer, where the j-th output of a softmax function is defined to be

$$\operatorname{softmax}[\mathbf{x}]_j = \frac{e^{x_j}}{\sum_{i=1}^{K} e^{x_i}} \tag{2.125}$$

for $\mathbf{x} \in \mathbb{R}^K$. The outputs of the network are then constrained to be probabilities. A Gaussian prior is entirely inappropriate for this situation. Instead, the categorical cross entropy should be used, which is given by

$$H(p_{\text{model}}, \hat{p}_{\text{data}}) = -\frac{1}{l}\sum_{i=1}^{l} \log f(\mathbf{x}_i;\boldsymbol{\theta})_{y_i}, \tag{2.126}$$

where p_{model}, $\hat{p}_{\text{data}}$, and $\mathbf{x}_i$ are defined as in Section 2.2.2 and y_i is the index of the output corresponding to the correct classification of $\mathbf{x}_i$.

Alternative nonlinear activation functions comprise the second best practice. Sigmoidal activations such as the logistic function (2.117) or hyperbolic tangent have practical deficiencies that render them unsuitable, especially for networks with many layers. The main deficiency is that the gradient tends to vanish outside of a very small window, which slows training considerably. Instead of such activations, the rectified linear unit (ReLU) should be used. An ReLU is calculated as the element-wise maximum of a given vector and 0, i.e. the j-th output is

$$\text{ReLU}[\mathbf{x}]_j = \max\{0, x_j\}. \tag{2.127}$$

Networks employing ReLUs are more efficiently computed than those with sigmoids, do not suffer from vanishing gradients, and retain the universal approximation property.

Deep learning also places great emphasis on *regularization.* Regularization is any method that is used with the intention of reducing test or generalization error, possibly at the expense of training error. In other words, the goal of regularization is to avoid overfitting to the training data. Some types of regularization are listed here.

- *Weight decay*, also known as l_2 regularization, adds the norm of the model's parameter vector to the objective function scaled by a coefficient λ. For example, if we have a model with parameter vector $\boldsymbol{\theta}$ and loss function $\mathcal{L}(\mathbf{x}; \boldsymbol{\theta})$, then the training objective Ω becomes

$$\Omega(\boldsymbol{\theta}) = \mathcal{L}(\mathbf{x}; \boldsymbol{\theta}) + \lambda \boldsymbol{\theta}^T \boldsymbol{\theta}, \tag{2.128}$$

where $\lambda > 0$ controls the influence of the regularization. Note that weight decay does not have to be applied to the entire parameter vector. Other l_p norms may be used for weight decay that have subtle effects on the regularization.

- In *dropout* [130], units in the neural network are randomly dropped during training by setting their outputs to zero. Dropout makes the network more robust to noise and is theoretically a means to simulate an ensemble of 2^N networks, where N is the number of neurons in the target architecture. In addition, dropout reduces codependencies and correlations between weights by forcing them to train separately from one another, thus allowing the network to learn multiple partial representations of the data. A downside is that the network's size and training time must generally be increased to accommodate dropout.

  Two similarly themed forms of dropout exist, namely dropconnect [148] and word dropout [63]. In dropconnect, individual weights are randomly set to zero rather than entire units. Word dropout is a somewhat constrained version of dropout that drops out certain subsets of units together. For example, word dropout may consist of randomly dropping timesteps in a recurrent neural network.

- Gaussian noise may be added to the input layer during training as a form of regularization.

Regularization is not guaranteed to improve generalization error, and applying too much or too strong of a regularizer can make the classifier perform worse than without.

These tools should therefore be used with careful intent. Used correctly, however, one can expect a more reliable model in untested situations.

CHAPTER 3

3D HAND POSTURE RECOGNITION FROM SMALL, UNLABELED POINT SETS

In this chapter we explore several classification algorithms for identifying hand postures using the 3D marker positions reported by Vicon. A natural complication arises when choosing features with which to perform classification, and choosing an appropriate feature extraction or transformation is challenging. Note that there is no context provided for the points other than each other; each frame or set of markers is a standalone entity with no external context. For the duration of this chapter, we will refer to the positions as raw features. Our comparison will include methods that work directly with the raw features as well as those that do not.

Some might think that image or point set registration [8], which aims to align several images or point sets via rigid or non-rigid [66] transformations, would yield a solution or be a practical step towards one. This intuition would be incorrect. For all intents and purposes, we consider our point sets to already be aligned via a rigid transformation based upon the distinctive rigid pattern of markers organized on the back of the glove in Figure 1.2. Since we are dealing with a small number of points, some of which may be missing or occluded, registration under normal assumptions

could lead to spurious alignments, such as the knuckles of one posture being aligned with the fingertips of another.

Our main contribution is an analysis and comparison of various methods for the classification of relatively sparse, aligned, unlabeled point sets of variable size. As stated above, we assume that there is no further context for each point set beyond the information contained in the positions themselves. At no point in time do we know which marker is the "thumb" or something similar. Note that in this framework, terms such as "thumbs up" or "thumbs down" (used by Song *et al.* [129]) are considered synonymous as they correspond to the same posture. By including details such as orientation after posture recognition, one can make more refined distinctions (e.g. thumbs up at θ degrees). Any system capable of generating positions corresponding to landmarks (e.g. fingertips), especially with respect to some standard reference frame, may benefit from our analysis.

3.1 Methodology

In this section, we describe our dataset and the classification algorithms evaluated with it. To our knowledge, there is no public dataset directly related to our purpose, i.e. a dataset comprised of instances of small unordered point sets, especially for 3D hand gesture or posture recognition. Therefore, we have produced our own. Several classification algorithms were evaluated on both the raw data (unordered positions) as well as on certain feature transformations.

3.1.1 Data

We base our analysis on the posture recognition data set described in Section B.3. The five postures are fist, pointing with one finger, pointing with two fingers, stop (hand flat), and grab (fingers curled) (see Figure B.3). Since each instance is a variable-size (due to occlusion) unordered set of 3D points, multiple derivative datasets were created to address the lack of structure.

Raw

This dataset is comprised of simply the instances with the minimal preprocessing described in the appendix.

Aggregate

We extract aggregate features that do not depend on the points' order. In particular, the following aggregate features were considered: number of markers, mean, eigenvalues and vectors of the points' covariance matrix, and the dimensions of the axis-aligned minimum bounding box centered on the mean. The expectation is that aggregate features will suffice as long as marker occlusion is not too severe, at which stage more locally sensitive features may be beneficial.

Grid Transformation

Although one could rasterize the space, the resolution of the rasterization would likely be prohibitive in terms of associated time and space constraints. As an alternative, we used a low-resolution pseudo-rasterization based upon a limited 3D grid of overlapping spheres. Cubes or diamonds could have alternatively been used by changing the spheres' associated l_p metric, but these were not ultimately considered.

Each sphere contributes one feature to a transformed instance, recording the presence of markers within its boundary according to some function of their Euclidean distance to the sphere's center. In this manner, we impose a spatially relevant order on the raw features. Step, linear, and Gaussian functions $f_i(x)$ of a marker's distance x from the center of a sphere with radius r were considered and are defined as

$$f_1(x) = \begin{cases} 1 & \text{if } x < r, \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

$$f_2(x) = \max\left\{1 - \frac{x}{r}, 0\right\}, \tag{3.2}$$

$$f_3(x) = \exp\left[-\frac{\sigma^2}{2}\left(\frac{x}{r}\right)^2\right], \tag{3.3}$$

where σ is the number of standard deviations (compared to standard normal) within the sphere. Each function is scaled so that it has a value of 1 for $x = 0$ and a value of 0 (or near 0 in the case of the Gaussian with $\sigma = 3$) for $x \geq r$. In a sense, the spheres are like neurons in a neural network whose activations are triggered by the markers. The activations caused by multiple markers in the same sphere are simply aggregated in a summation. One may note that this grid transformation is reminiscent of a convolutional layer of a neural network, albeit more hand-crafted.

We first determined a box in which the user's hand was expected to lie based upon the mean position plus or minus two standard deviations. Supposing there are m spheres per dimension, the spheres are scaled and arranged such that they form a regular, densely packed grid spanning the internal volume of the box. Their radii are then uniformly scaled by an integer multiple r_s such that the entire internal volume is covered. The advantage of letting the spheres overlap lies in the implicit

creation of extra detection regions according to their intersections. We considered 36 transformations based upon the following options: $m \in \{3, 4, 5, 6\}$, $r_s \in \{2, 3, 4\}$, and $f_i(x)$ with $i \in \{1, 2, 3\}$. See Figure 3.1 for a visualization.

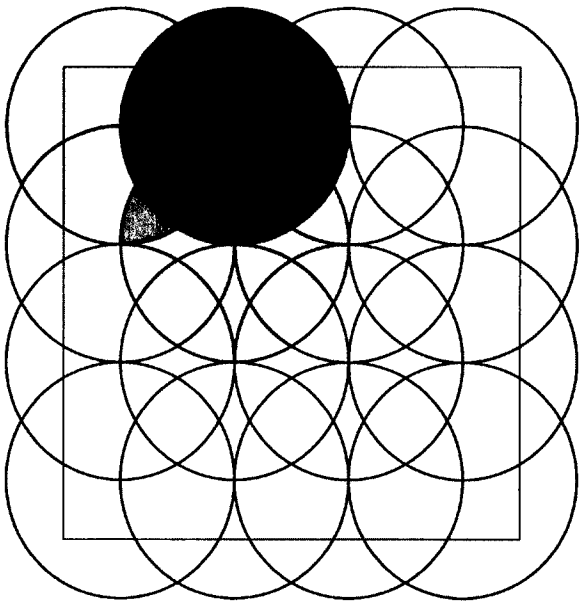


Figure 3.1: A 2D grid transformation with $m = 4$, $r_s = 2$, and $i = 2$. The opacity of each sphere is proportional to its activation by the top-left marker.

3.1.2 Classifiers

Our classifiers are split into multiple categories based upon the associated type of dataset. We will list raw data classifiers first, which require extra explanation, before providing the traditional classifiers, e.g. SVM, considered for use with the aggregate and transformed data. First, we briefly describe two tools used thoroughly in the raw classifiers, GMMs and minimum-cost matchings.

Recall the definition of a GMM given in Section 2.1.6. A GMM is a collection of n multivariate Gaussian distributions, or components, used to estimate an arbitrary probability density distribution.

A minimum-cost matching is a solution to the assignment problem (see Section 2.4). In our case, we wish to assign the points of one instance to the points of another (or to the components of a GMM) such that the summation of costs over all matched pairs is minimized. In this scenario, a GMM component's distribution describes the region in which a marker (such as the tip of the thumb) is expected to lie. Each component represents the expected location of a certain marker. The mixture gains represent the probability of each component being present at all in a given instance. A given GMM approximates the shape of a certain posture, and therefore we construct one per class.

Five cost functions for matching component c_k to the j-th position $\mathbf{x_j}$ of an instance were considered. The first, $C_1(c_k, \mathbf{x_j})$, is simply the Euclidean distance between the point and the mean, i.e.

$$C_1(c_k, \mathbf{x_j}) = \|\mathbf{x_j} - \boldsymbol{\mu}_k\|_2. \tag{3.4}$$

The other cost functions measure the probability of observing $\mathbf{x_j}$ independently of other components and factors. The first of these is a normalized version of the component's probability density function and is calculated according to

$$C_2(c_k, \mathbf{x_j}) = -\ln\left(f_k(\mathbf{x_j})\sqrt{(2\pi)^3|\Sigma_k|}\right), \tag{3.5}$$

where the negative logarithm is taken so that the minimum-cost matching will maximize the product of independent probabilities. Similarly,

$$C_3(c_k, \mathbf{x_j}) = -\ln\left(1 - \chi^2\left((\mathbf{x_j} - \boldsymbol{\mu}_k)^T\Sigma_k^{-1}(\mathbf{x_j} - \boldsymbol{\mu}_k)\right)\right), \tag{3.6}$$

where $\chi^2(z)$ is the cumulative distribution function of a chi-squared variable of degree 3. Note that z in $C_3(c_k, \mathbf{x_j})$ is the square of the Mahalanobis distance, which has a chi-squared distribution with 3 degrees of freedom [122]. We augment C_2 and C_3 to produce the last two cost functions

$$C_4(c_k, \mathbf{x_j}) = C_2(c_k, \mathbf{x_j}) - \ln \pi_k, \tag{3.7}$$

$$C_5(c_k, \mathbf{x_j}) = C_3(c_k, \mathbf{x_j}) - \ln \pi_k, \tag{3.8}$$

which account for components that are more or less likely to appear.

For classification based upon a matched GMM, we choose the class corresponding to the GMM with the minimum average-cost (per component) matching. An unmatched component c_k is given a cost of $-\ln(1 - \pi_k)$, the probability of it being absent in a given instance.

Greedy GMM

The standard algorithm for computing a GMM with a fixed number of components is the EM algorithm. Since we do not necessarily know how many components to expect, we use the greedy algorithm of Verbeek *et al.* [136]. The GMMs are constructed using the greedy algorithm on the union of all training sets I. For unmatched classification, we treat each instance I as a small standalone dataset and classify it as the GMM with the highest log-likelihood

$$\mathcal{L}(I) = \sum_{\mathbf{x} \in I} \ln \mathcal{M}(\mathbf{x}). \tag{3.9}$$

The classification corresponds to which model's parameters are more likely given the data in I. A matched version uses the same greedy algorithm for construction with C_4 or C_5 for classification.

Heuristic GMM

We also explore alternative heuristics employing minimum-cost matchings for constructing the mixtures. The underlying motive for the heuristic GMM is to produce a pseudo-naive Bayes classifier where each GMM component contributes an independent observation. The algorithm in Figure 3.2 produces pseudo-GMMs that respect the constraint where two markers of the same instance cannot belong to the same component.

---

procedure train(I, $C \in \{C_1, C_2, C_3\}$, $O \in \{R, E\}$)

Given: Set of instances I, cost function C, option O
Output: GMM G that approximates I

```
Initialize G with 0 components
matchInstances(G, I, C)
while G is not converged do
    Randomly permute I
    if O equals R then
        rematchInstances(G, I, C)
    else
        matchInstances(G, I, C)
    end if
end while
Set mixture gains to percentage of I containing matches
```

Figure 3.2: The algorithm used to train heuristic GMMs. Convergence depends upon O. If R is used, convergence occurs when the number of markers rematched to a different component drops below a threshold. Otherwise, convergence depends on the matching cost under C. A maximum number of iterations is allowed before convergence.

The algorithm relies on two sub-procedures: "matchInstances" and "rematchInstances." The first procedure merges each instance into an empty (i.e. 0 components) or pre-existing GMM according to sequential bipartite matchings and is shown in Figure 3.3.

```
procedure matchInstances(G, I, C)
Given: GMM G, set of instances I, cost function C
Output: Refined GMM G
for all instances i in I do
    Match points of i to components of G according to C
    for all points p of i do
        if p is matched to component c then
            Merge p into c; update mean and covariance
        else
            Add component to G with mean p
        end if
    end for
end for
```

Figure 3.3: The sub-procedure used to initialize a heuristic GMM.

The second sub-procedure (Figure 3.4) adjusts a GMM presumably constructed by the first sub-procedure by iteratively removing and re-merging each instance based upon the assumption that the assignment from points to components used to re-merge is more accurate than the assignment previously used in the removal. One of C_1, C_4, or C_5 is used for classification. If C_1 is chosen, then we ignore the cost of an unmatched component as we are not performing a probabilistic classification. Let $O(C_i, C_j)$ denote a heuristic GMM built with option O and cost function C_i that classifies according to C_j. Note that we do not produce a GMM in the strict sense of the definition (the mixture gains do not add up to 1). However, GMMs do provide

a useful vocabulary with which to discuss the classifier. See Chapter 4 for a more developed version of this idea.

---

procedure rematchInstances(G, I, C)

Given: GMM G, set of previously matched instances I, cost function C
Output: Refined GMM G

---

for all instances i in I **do**
    Remove points of i from prior matched components
    Rematch and merge points of i into components of G
end for

Figure 3.4: The sub-procedure used to refine a heuristic GMM.

Raw Nearest Neighbor

We use the normalized matching distance of Ramon and Bruynooghe [111]. This distance is similar to EMD (Section 2.4) except it is normalized and remains a metric on sets of unequal size. As such, it measures the cost of transforming one set into the other via a minimum-cost deformation. When applied between two measures μ and ν on a domain X with bounded metric d, it takes the form

$$M(\mu,\nu) = \frac{2EMD(\mu,\nu) + |\mu(X) - \nu(X)| \max_{x,y}\{d(x,y)\}}{\max\{\mu(X),\nu(X)\} + EMD(\mu,\nu)}. \tag{3.10}$$

A majority vote among the 6 nearest neighbors is used to classify a given query instance, with ties broken by the query's minimum average distance per class of neighbor. Chapter 5 shifts this approach to a more theoretically sound kernel setting.

Traditional

Six traditional classifiers are considered for the aggregate and grid transformed datasets: naive Bayes, Bayesian networks, MLPs, SVMs, random forests, and k-NN (with $k = 6$). The implementation and testing of these classifiers is provided by

WEKA [52]. Grid transformed classifiers employ feature selection (FS) to both reduce processing time and improve accuracy since many of the spheres in the grid hardly ever contain a marker. Aggregate classifiers did not necessarily use FS. Generally, with the exception of k in k-NN, we let WEKA choose default classification parameters.

3.1.3 Evaluation

Due to the streaming nature of the data capture, it is likely that for an instance of a given user there will be a duplicate or near duplicate within the user's dataset. Therefore, we adopted a leave-one-user-out evaluation strategy. In addition, this strategy allows us to measure the ability of a given classifier to generalize to users it has not seen before, just as it would need to do in practice.

We found it prohibitive in terms of time to consider every possible combination of grid transformed dataset, traditional classifier, and left out user. Thus, we opted to first select the "best" on-average classifier and dataset combination via a reduced user set of 4 randomly selected users and 12 randomly selected transformations. The selected classifier would then be compared against the raw and aggregate classifiers on the remaining 8 users. The "best" on-average grid transformed classifier was determined to be the MLP, and it attained its best performance on a transformation with 6 spheres per dimension, each of radius 4, and the linear function $f_2(x)$ (Equation 3.2).

We used *balanced error rate (BER)* as our primary metric for evaluating the performance of a classifier on c classes with confusion matrix A, which is defined as

$$\text{BER} = 1 - \frac{1}{c}\sum_{i=1}^{c}\frac{A_{ii}}{\sum_{j=1}^{c} A_{ij}}. \tag{3.11}$$

BER weights classes equally regardless of their representation in the dataset. If all classes are equally weighted, then it is equivalent to 1 minus the accuracy. Thus, the lower the value of this metric, the better our perceived evaluation of the classifier.

3.2 Results

For reference, we report a BER of 0.540 ± 0.123 for a "Simple" naive Bayes classifier based upon a single feature, the number of visible markers. This reference classifier gives us an idea of the discriminative power that comes just from counting markers.

Results for transformed feature classifiers are given in Table 3.1. The other traditional classifiers (aside from MLP) were also evaluated on the "best" on-average transformation. The transformed classifiers had a wide range of performance. Even though the MLP achieved the best on-average performance, the k-NN classifier performed better on the chosen final transformation. Note that increasing the number of spheres or sphere radii may yield improved performance. However, this increased transformation complexity automatically translates to increased model complexity, which potentially complicates training and increases the risk of overfitting.

Table 3.1: The average BERs per user left out and corresponding standard deviations for tested transformed feature classifiers. Lower BER is better.

Classifier	BER
k-NN	0.158 ± 0.152
MLP	0.183 ± 0.168
SVM	0.204 ± 0.155
Random Forest	0.241 ± 0.151
Bayes-Net	0.353 ± 0.154
Naive Bayes	0.375 ± 0.181

Results for aggregate feature classifiers are given in Table 3.2. Aggregate classifiers performed fairly similarly to one another, exhibiting fairly balanced performance across the different users. On average, though, they were generally on the higher end in terms of BER. FS did not generally yield improvement in average BER, which is not particularly surprising given the relatively small number of aggregate features. Deviation in BER across users, on the other hand, was generally reduced by FS. Only the MLP and Bayesian network noticeably benefited from attribute selection. Many of the aggregate classifiers had relatively low deviation, reflective of the smooth nature of their features (i.e. they are not prone to overfitting).

Table 3.2: The average BERs per user left out and corresponding standard deviations for tested aggregate feature classifiers tested. Lower BER is better.

Classifier	**BER**
SVM	0.216 ± 0.136
Random Forest	0.221 ± 0.156
SVM (FS)	0.232 ± 0.098
MLP (FS)	0.248 ± 0.098
Naive Bayes	0.273 ± 0.117
MLP	0.289 ± 0.128
Random Forest (FS)	0.292 ± 0.148
k-NN	0.300 ± 0.165
k-NN (FS)	0.301 ± 0.142
Naive Bayes (FS)	0.303 ± 0.202
Bayes-Net (FS)	0.352 ± 0.101
Bayes-Net	0.421 ± 0.187

Table 3.3 provides results for raw feature classifiers. The matched pseudo-GMMs built using the provided algorithm performed best on average among all raw classifiers, in particular the variants that trained with the C_1 cost function and classified with C_4 or C_5. The reasoning for this is not completely clear; perhaps using

the incomplete covariance as in C_2 or C_3 leads to a feedback loop that augments the initial error. On the other hand, ignoring covariance entirely by building and classifying with C_1 yielded very poor results. Note, however, that the raw nearest neighbor has significantly less deviation among the users despite possessing slightly worse on average performance. This lower deviation indicates that it generalizes more readily. The GMMs are prone to overfitting. The greedy GMMs performed quite poorly, likely due to the implicit assumption that a given marker would appear in roughly the same place for each user. Since marker constraints were ignored, overlapping distributions from each user caused the resulting components to poorly reflect the true distributions of individual markers. This problem was magnified in the matched greedy GMMs, whose components clearly did not represent the expected locations of the markers.

Table 3.3: The average BERs per user left out and corresponding standard deviations for tested raw feature classifiers. Lower BER is better.

Classifier	BER
Unmatched Greedy	0.416 ± 0.183
Matched Greedy (C_5)	0.681 ± 0.155
Matched Greedy (C_4)	0.719 ± 0.134
$R(C_1, C_5)$	0.192 ± 0.180
$R(C_1, C_4)$	0.194 ± 0.178
$R(C_2, C_4)$	0.197 ± 0.192
$E(C_2, C_4)$	0.199 ± 0.203
$E(C_1, C_5)$	0.203 ± 0.179
$E(C_3, C_5)$	0.203 ± 0.190
$R(C_3, C_5)$	0.216 ± 0.227
$E(C_1, C_4)$	0.217 ± 0.169
$R(C_1, C_1)$	0.375 ± 0.211
$E(C_1, C_1)$	0.383 ± 0.211
k-NN	0.214 ± 0.089

We also note that some users are inherently harder to classify than others, regardless of the chosen algorithm. One user consistently had the highest error when left out, even though their rates of marker occlusion were not particularly abnormal or even above average. We think that this supports the idea that an online learning scheme will inevitably be required for any system that expects users to perform natural gestures rather than those precisely dictated by the system. Regardless of the initial training set, there will likely be outlying users that require the system to adapt automatically or through some form of positive and/or negative feedback.

CHAPTER 4

ESTIMATING THE DISTRIBUTION OF UNLABELED, CORRELATED POINT SETS

The specific problem we address in this chapter involves the estimation of positions and correlations of multiple unlabeled, presumed stationary targets. The goal is to obtain a distribution that describes the expected position of each target, the dependencies (captured via a covariance matrix) between the targets, as well as the probability that each target will appear or not, assuming independence. A practical motivation for this problem is that of the estimation of hand postures from noisy, incomplete, and unlabeled point sets that represent positions of certain landmarks on the hand as recorded by a motion capture environment. As will be shown, this problem bears some relation to k-means clustering [70] (with constraints), multi-target tracking, and certain optimal transport problems such as the computation of Wasserstein barycenters.

We solve the problem through use of the EM algorithm (Section 2.2.3) and propose using the Kalman filter (Section 2.5.1) in a manner similar to Einicke *et al.* [35] to provide partial, incremental EM steps with the intention of obtaining better solutions. A fundamental problem encountered is the uncertainty in assigning observed targets to estimated targets since the targets are unlabeled. One could compute the single

(or k) best assignment(s) based upon current state estimates, yielding a so-called hard EM algorithm. Alternatively, one could compute the expectation over all possible assignments and obtain a soft EM algorithm. However, sampling such assignments effectively can be remarkably difficult due to the constraints and multi-modality of the distribution, and literature is rather sparse. We apply a state-of-the-art MCMC sampling algorithm [144] and evaluate it against the hard EM approach.

We also propose two modifications of the Kalman filter in order to address the uncertainty in observed target identity. In the first modification, we propose treating the measurement matrix as a random variable and propagating a Gaussian approximation of the true state density. In the second modification, we present a way to iteratively reorder the measurements such that the state estimate improves.

A series of simulations with varying numbers of targets and spreads are performed in which a true distribution is known and sampled from to generate simulated data. The quality of a computed distribution is assessed through the use of the Rand index [112] and a modified Hellinger distance with respect to this true distribution. We find that the Kalman-filter-based EM algorithm yields improvements in accuracy in the majority of scenarios (for example, see the figures in Section 4.3). Similarly, the MCMC sampler is found to usually be better even without careful tuning.

4.1 Problem Definition and Related Work

Consider as motivation unlabeled point sets that represent positions on a user's hand of infrared reflective markers (*targets*) used in motion capture systems. Our

goal is to obtain a statistical description of a posture that includes expected positions, deviations, and probabilities of appearance for each marker/target. The estimated postures may then be used to classify new point sets, identify users, or simply visualize the posture. Given that certain markers are attached to the same finger, we reason that the inter-target covariance is significant and worth estimating. Let m be the number of targets, $M_i \in \mathbb{R}^d$ be the position of the i-th target, π_i the probability of the i-th target appearing or being detected, $\boldsymbol{\mu} \in \mathbb{R}^{md}$ be a concatenation of these positions into a single vector, and Σ be the covariance of $\boldsymbol{\mu}$. We define a *profile*

$$P \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma) \times \prod_{i=1}^{m} \mathcal{B}(\pi_i, 1). \tag{4.1}$$

We will simplify this by introducing the notation $\mathcal{P}(\boldsymbol{\mu}, \Sigma, \boldsymbol{\pi})$ to represent such a distribution. Our goal is to find the maximum likelihood estimates of a profile given some data. Note that this parameterization assumes that the probability of each target being occluded is independent from other targets, which is not likely to be true in practice but avoids the problem of modeling the potentially intractable combinatorial relationships between targets.

This work generalizes and formalizes the heuristic GMMs proposed in Chapter 3. Otherwise, to our knowledge, this precise parameterization has not been considered before. The main ways in which our problem and proposed solution differ from existing work is in the explicit consideration of the target covariance and different appearance or detection probabilities for each target. For example, if we assumed that the targets were independent, then numerous filtering algorithms exist for estimating target positions (see Section 2.5.4). On the other hand, if we ignore Σ and $\boldsymbol{\pi}$, then

what we effectively propose is a constrained k-means clustering algorithm [21] where constraints force points to have the same or different clusters. One algorithm for constrained k-means uses Wasserstein barycenters [25], which are conceptually similar to a profile. Unlike Wasserstein barycenters in this context, profiles are not equivalent to Gaussian mixtures, however, as the targets are not individually weighted. We note, though, that our algorithm can benefit from the entropically regularized algorithm of Cuturi [24] for computing the Wasserstein distance and Wasserstein barycenters, although this would again ignore the covariance between targets and only be an approximation of our goal. Adapting Cuturi's algorithm to handle the quadratic program induced by the covariance may be possible, although it is beyond the scope of this dissertation and we did not use the algorithm in our experiments.

4.2 The Proposed Algorithm

This section is devoted to defining *a fortiori* expectation-maximization (AFEM), a Kalman-filter-based EM approach to obtaining maximum likelihood estimates of a profile. The interpretation of each point set as a *performance* or sample of a posture guides us as we define a Kalman filter that estimates the posture's profile. Since the point sets are inherently unlabeled, there is no *a priori* association between targets measured at different times. Consequently, H_k is unknown. Therefore, we must first introduce a modified Kalman filter capable of handling uncertain measurement matrices.

4.2.1 A Kalman Filter for Uncertain Measurement Matrices

Note that as a Bayes filter (Section 2.5.2), the measurement matrix H_k is implicitly given in that the Kalman filter actually maintains $p(\mathbf{x}_k|\mathbf{y}_{1:k}, H_{1:k})$. We assume that H_l is independent of H_k for $l \neq k$, so we simplify this expression to $p(\mathbf{x}_k|\mathbf{y}_{1:k}, H_k)$. Consequently, we must marginalize out H_k in order to obtain $p(\mathbf{x}_k|\mathbf{y}_{1:k})$. To do so, note the joint distribution

$$p(\mathbf{x}_k, H_k|\mathbf{y}_{1:k}) = p(\mathbf{x}_k|\mathbf{y}_{1:k}, H_k)p(H_k|\mathbf{y}_{1:k}) \tag{4.2}$$

and the marginal distribution

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \int p(\mathbf{x}_k, H_k|\mathbf{y}_{1:k})\mathrm{d}H_k = \mathbb{E}_{H_k|\mathbf{y}_{1:k}}\left[p(\mathbf{x}_k|\mathbf{y}_{1:k}, H_k)\right], \tag{4.3}$$

where

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}, H_k) = \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_k, \hat{P}_k) \tag{4.4}$$

$$p(H_k|\mathbf{y}_{1:k}) = p(H_k|\mathbf{y}_k, \hat{\mathbf{x}}_k^-, \hat{P}_k^-), \tag{4.5}$$

and $\mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_k, \hat{P}_k)$ is the probability density function of $\mathcal{N}(\hat{\mathbf{x}}_k, \hat{P}_k)$ evaluated at $\mathbf{x}_k$. A similar derivation for unknown filter parameters including A_k, Q_k, or R_k is given by Mehra [93].

In practice, $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ is likely to be multimodal and difficult to compute depending on the complexity of $p(H_k|\mathbf{y}_{1:k})$. We therefore propose to propagate the mean and covariance of $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ in an adapted Kalman filter as a Gaussian approximation to the true distribution. In order to accomplish this, we observe that

for an arbitrary function $f(\mathbf{x}_k)$,

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}_k|\mathbf{y}_{1:k}}\left[f(\mathbf{x}_k)\right] &= \int f(\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k})\mathrm{d}\mathbf{x}_k \\
&= \int f(\mathbf{x}_k)\mathbb{E}_{H_k|\mathbf{y}_{1:k}}\left[p(\mathbf{x}_k|\mathbf{y}_{1:k}, H_k)\right]\mathrm{d}\mathbf{x}_k \\
&= \int f(\mathbf{x}_k)\left[\int p(\mathbf{x}_k|\mathbf{y}_{1:k}, H_k)p(H_k|\mathbf{y}_{1:k})\mathrm{d}H_k\right]\mathrm{d}\mathbf{x}_k \\
&= \int \left[\int f(\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k}, H_k)\mathrm{d}\mathbf{x}_k\right] p(H_k|\mathbf{y}_{1:k})\mathrm{d}H_k \\
&= \mathbb{E}_{H_k|\mathbf{y}_{1:k}}\left[\mathbb{E}_{\mathbf{x}_k|\mathbf{y}_{1:k},H_k}\left[f(\mathbf{x}_k)\right]\right].
\end{aligned} \tag{4.6}$$

We can therefore infer how to adapt our Kalman filter to an uncertain measurement matrix by converting (2.82) and (2.83) into expectations, i.e.

$$\begin{aligned}
\hat{\mathbf{x}}_k &= \mathbb{E}_{\mathbf{x}_k|\mathbf{y}_{1:k}}\left[\mathbf{x}_k\right] \\
&= \mathbb{E}_{H_k|\mathbf{y}_{1:k}}\left[\hat{\mathbf{x}}_k^- + K_k(\mathbf{y}_k - H_k\hat{\mathbf{x}}_k^-)\right],
\end{aligned} \tag{4.7}$$

$$\begin{aligned}
\hat{P}_k &= \mathbb{E}_{\mathbf{x}_k|\mathbf{y}_{1:k}}\left[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^\intercal\right] \\
&= \mathbb{E}_{H_k|\mathbf{y}_{1:k}}\left[(\mathbf{I} - K_kH_k)\,\hat{P}_k^-\right],
\end{aligned} \tag{4.8}$$

where the Kalman gain is defined normally as a function of H_k. At each step we assume that the previous steps of the Kalman filter satisfied the standard assumptions for optimality, compute the expected filter output, and then treat the result as a typical state estimate. A normal Kalman filter can be considered a special case where H_k is distributed as a discrete variable with only one point of nonzero probability mass. The proposed Kalman filter is similar to the JPDA filter [39] when H_k is constrained to be a permutation matrix. Instead of aggregating measurements, we aggregate *a posteriori* estimates. Depending on the distribution of H_k, specialized algorithms or Monte Carlo sampling may be necessary for the sake of efficiency.

4.2.2 *A Fortiori* Estimates

The fact that measurements are unordered allows us to employ one additional heuristic to enhance our filter's accuracy. By assuming that $\hat{\mathbf{x}}_k$ is a better estimate than $\hat{\mathbf{x}}_l$ for $k \gg l$, we reason that a more accurate distribution for H_l would result if $\mathbf{y}_l$ had appeared last. We can simulate this scenario by removing $\mathbf{y}_l$ from the calculation of $\hat{\mathbf{x}}_k$, i.e. reversing a Kalman filter timestep as though $\mathbf{y}_l$ was the most recent measurement, and then repeating the timestep with $\mathbf{y}_l$ as the observation to obtain $\hat{\mathbf{x}}'_k$ and $\hat{P}'_k$. We use the term *a fortiori* to denote the presumably improved estimates that result from this process since *a fortiori* indicates a conclusion with stronger evidence than a previously accepted one.

In order to undo a timestep, it suffices to calculate simulated *a priori* estimates $\hat{\mathbf{x}}_k^{(-)}$ and $\hat{P}_k^{(-)}$ given the *a posteriori* estimates and a measurement. Suppose H_l is known. The *a priori* error covariance can then be calculated using the Sherman-Morrison-Woodbury identity [124, 153], which states that for some selection of conformable matrices A, U, C, and V,

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}. \tag{4.9}$$

Letting $A^{-1} = \hat{P}_k^{(-)}$, $V = U^\intercal = H_l$, and $C^{-1} = R_l$ and examining (2.83), we find

$$\hat{P}_k^{(-)} = [(\hat{P}_k)^{-1} - H_l{}^\intercal R_l^{-1} H_l]^{-1}. \tag{4.10}$$

Solving for $\hat{\mathbf{x}}_k^{(-)}$, we obtain

$$\hat{\mathbf{x}}_k^{(-)} = (C_k{}^\intercal C_k)^{-1} C_k{}^\intercal (\hat{\mathbf{x}}_k - K'_k \mathbf{y}_l), \tag{4.11}$$

where K'_k is the Kalman gain computed with $\hat{P}_k^{(-)}$, H_l, and R_l and

$$C_k = \mathbf{I} - K'_k H_l. \tag{4.12}$$

The *a fortiori* estimates $\hat{\mathbf{x}}'_k$ and $\hat{P}'_k$ follow after a normal correction step with $\mathbf{y}_l$ as the measurement and $\hat{\mathbf{x}}_k^{(-)}$ and $\hat{P}_k^{(-)}$ as the *a priori* estimates. Of course, if H_l is known, then $\hat{\mathbf{x}}'_k = \hat{\mathbf{x}}_k$ and $\hat{P}'_k = \hat{P}_k$. If H_l is not known, then a procedure similar to that of Section 4.2.1 can be used, i.e. calculating the expected values of $\hat{\mathbf{x}}_k^{(-)}$ and $\hat{P}_k^{(-)}$ with respect to $H_l|\mathbf{y}_{1:l}$. In practice, one re-uses samples of the measurement matrix computed at the l-th timestep. A normal timestep with new samples follows.

4.2.3 Defining the State

A posture is defined to be a translation and rotation invariant static configuration of points or targets. We assume that the posture is described within a local coordinate system such that translation and rotation are not an issue. Since a posture is by definition constant, we resolve the question of process noise by interpreting each performance as a measurement of a noiseless state that describes the ideal posture. We define the state $\mathbf{x}$ of our filter to be the parameter $\boldsymbol{\mu}$ of the target profile. The discrete-time transition function is then $\mathbf{x}_{k+1} = \mathbf{x}_k$. Note that as a consequence of zero process noise and a constant state, the *a priori* estimates are equal to the previous time-step's *a posteriori* estimates. Since the state is constant and noiseless at all times, we drop the k subscript on $\mathbf{x}$.

4.2.4 Defining the Measurements

Let $\mathbb{Y}_k$ be a point set, n_k its cardinality, and let $\mathbf{y}_k \in \mathbb{R}^{n_k d}$ be a concatenation of each point's position vector in an arbitrary order. The measurement equation is

given by

$$\mathbf{y}_k = H_k\mathbf{x} + H_k\mathbf{w}_k + \mathbf{v}_k, \tag{4.13}$$

where $\mathbf{w}_k$ and $\mathbf{v}_k$ are zero-mean white noise with covariances Σ and R_k, respectively. The vector $\mathbf{w}_k$ represents the deviation of the k-th performance from the ideal posture and is assumed to have constant covariance, i.e. each performance has the same expected deviation. The vector $\mathbf{v}_k$ denotes the error due to measurement and is not necessarily of constant variance. By hypothesis, we do not know H_k. We can, however, constrain H_k with $H_k = B_k \otimes \mathbf{I}_d$, where B_k is a binary matrix, which ensures that the components of a measurement cannot be assigned to different components of one target, e.g. x-axis to z-axis.

Note that (4.13) assumes that $n_k \leq m$ and that each observed point corresponds to a target, which is not always the case with extraneous targets. In the event that extraneous targets are present, one can consider $\mathbf{y}_k = \tilde{H}_k\tilde{\mathbf{y}}_k$, where $\tilde{H}_k$ is similar in structure to H_k in that it selects and possibly permutes elements of the full measurement vector $\tilde{\mathbf{y}}_k$. Note that H_k and $\tilde{H}_k$ depend upon one another.

The sum $H_k\Sigma H_k^\intercal + R_k$ represents the traditional measurement noise covariance given in the Kalman filter's introduction. We remark here that Σ plays an almost identical role to that of the process noise in a standard Kalman filter except for the prediction stage. The main effect is that the estimated error covariance $\hat{P}$ does not increase in the prediction stage or when a target is not observed, which more accurately reflects the notion of a constant state not subject to random walks.

4.2.5 Uncertain Measurement Matrices

A distribution must be specified for $p(H_k|\mathbf{y}_{1:k})$. The choice of the distribution is significant as it reflects the assumptions and constraints on the possible assignments. The distribution presented here corresponds to the conditions of our simulations later in the chapter. In particular, we keep the constraint that no two observed targets may be assigned to the same states. We also discount the possibility of extraneous targets, requiring that each target must be assigned an observation (if any are available) without violating constraints.

Regardless of the choice of the conditional distribution $p(\mathbf{y}_k|H_k, \hat{\mathbf{x}}_k^-, \hat{P}_k^-)$, we assume that each permutation of targets is equally likely to be observed. Since by (4.7) and (4.8) we define the state to be Gaussian,

$$\mathbf{y}_k|H_k, \hat{\mathbf{x}}_k^-, \hat{P}_k^- \sim \mathcal{N}(H_k\hat{\mathbf{x}}_k^-, S_k), \tag{4.14}$$

where

$$S_k = H_k(\Sigma + \hat{P}_k^-)H_k{}^\intercal + R_k. \tag{4.15}$$

The conditional distribution of H_k given $\mathbf{y}_k$ and the marginal distribution of $\mathbf{y}_k$ are thus

$$\begin{aligned} p(H_k|\mathbf{y}_k, \hat{\mathbf{x}}_k^-, \hat{P}_k^-) &\propto \mathcal{N}(\mathbf{y}_k; H_k\hat{\mathbf{x}}_k^-, S_k) \\ &= \frac{\mathcal{N}(\mathbf{y}_k; H_k\hat{\mathbf{x}}_k^-, S_k)}{\sum_H \mathcal{N}(\mathbf{y}_k; H\hat{\mathbf{x}}_k^-, S_k)}, \end{aligned} \tag{4.16}$$

$$p(\mathbf{y}_k|\hat{\mathbf{x}}_k^-, \hat{P}_k^-) = \frac{\sum_H p(\mathbf{y}_k|H, \hat{\mathbf{x}}_k^-, \hat{P}_k^-)}{\min\{n_k, m\}!\binom{n_k}{\min\{n_k,m\}}}. \tag{4.17}$$

If the probability of each target being occluded is not the same, then (4.16) should be augmented with Bernoulli random variables. Let $\hat{\boldsymbol{\pi}}$ be an estimate of the

profile's target occlusion probability vector. Then

$$p(H_k|\mathbf{y}_k, \hat{\mathbf{x}}_k^-, \hat{P}_k^-, \hat{\boldsymbol{\pi}}) \propto \mathcal{N}(\mathbf{y}_k; H_k\hat{\mathbf{x}}_k^-, S_k) \times \hat{\boldsymbol{\pi}}^{B_k{}^\intercal \mathbf{1}_{n_k}} (\mathbf{1}_m - \hat{\boldsymbol{\pi}})^{(\mathbf{1}_m - B_k{}^\intercal \mathbf{1}_{n_k})}, \tag{4.18}$$

where $\mathbf{v}^{\mathbf{b}}$ represents element-wise exponentiation such that

$$\mathbf{v}^{\mathbf{b}} = \prod_i v_i^{b_i}. \tag{4.19}$$

We see from (4.16) that computation of $p(H_k|y_{1:k})$ (and hence (4.7) and (4.8)) involves an intractable sum in the denominator, which rules out exact computations for moderately sized problems. The mode or most likely assignment is also difficult to compute as it can be shown that maximizing the logarithm of (4.16) yields an NP-Hard quadratic program in most situations. We are forced to rely upon approximations regardless of whether we wish to perform hard or soft EM.

An approximation of the most likely measurement matrix can be obtained by linearizing $\log \mathcal{N}(\mathbf{y}_k; H_k\hat{\mathbf{x}}_k^-, S_k)$. The linearization consists of treating each target as though it were independent, i.e. as though S_k were block-diagonal and consequently

$$\mathcal{N}(\mathbf{y}_k; H_k\hat{\mathbf{x}}_k^-, S_k) = \prod_{i=1}^{n_k} \mathcal{N}(\mathbf{y}_k(i); [H_k\hat{\mathbf{x}}_k^-](i), S_k(i)), \tag{4.20}$$

where $\mathbf{y}_k(i)$ and $[H_k\hat{\mathbf{x}}_k^-](i)$ represent the coordinates of the i-th observation or target and $S_k(i)$ represents the i-th $d \times d$ block on the diagonal of the matrix S_k. An approximate mode can then be found in $O(m^3)$ time by using algorithms for the assignment problem [34]. In fact, approximations for the t best assignments can be found using Murty's ranked assignment algorithm [99].

For estimating expectations according to (4.16) (and indeed any expression proportional to $p(H_k|\mathbf{y}_{1:k})$), we turn to MCMC algorithms. MCMC methods for

sampling from a probability distribution are advantageous in that they only require an expression proportional to the true probability density function. The primary issue is one of practicality in that convergence to the true distribution is not guaranteed to be fast. Due to the combinatorial constraints on H_k, however, an effective sampler is difficult to obtain. Samplers have been proposed with special emphasis on calculating matrix permanents and related sub-problems [65], but few algorithms for sampling of arbitrarily weighted permutations exist. General purpose combinatorial samplers exist [80], but a special purpose sampler is likely to be more efficient. We propose using the sequential match sampling algorithm of Volkovs and Zemel [144], which is capable of sampling from arbitrary densities on permutations and appears to be the state of the art in this regard. The sequential match sampler is a Metropolis-Hastings algorithm [17] with a proposal distribution constructed by sampling partial assignments item by item. A temperature hyperparameter of the sampler controls the proposal distribution in a manner similar to temperature as used in simulated annealing in that higher temperatures promote jumps to more dissimilar permutations.

4.2.6 The AFEM Algorithm

The EM procedure to estimate a profile $\mathcal{P}(\boldsymbol{\mu}, \Sigma, \boldsymbol{\pi})$ from a collection of N point sets $\mathbb{Y}_{1:N}$ is outlined by the steps listed below. One may note that the Kalman filter fills the role of the expectation step along with the maximization with respect to $\boldsymbol{\mu}$. Setting the Kalman gain to zero yields a more traditional EM algorithm. Maximum likelihood estimates at iteration i can be obtained with

$$\hat{\boldsymbol{\mu}}^{(i)} = \frac{1}{N} \sum_{k=1}^{N} \mathbb{E}_{H_k | \mathbf{y}_{1:k}} \left[H_k{}^{\mathsf{T}} \mathbf{y}_k \right], \tag{4.21}$$

$$\hat{\Sigma}^{(i)} = \frac{1}{N}\sum_{k=1}^{N} \mathbb{E}_{H_k|\mathbf{y}_{1:k}}\left[(\mathbf{y}_k - H_k\hat{\boldsymbol{\mu}}^{(i)})(\mathbf{y}_k - H_k\hat{\boldsymbol{\mu}}^{(i)})^{\intercal}\right], \tag{4.22}$$

$$\hat{\boldsymbol{\pi}}^{(i)} = \frac{1}{N}\sum_{k=1}^{N} \mathbb{E}_{H_k|\mathbf{y}_{1:k}}\left[H_k{}^{\intercal}\mathbf{1}_{n_k}\right], \tag{4.23}$$

although each equation comes with some caveats. For example, (4.21) is optional as one could just use the final estimate given by the Kalman filter.

Both (4.21) and (4.22) assume that each measurement is complete, which is by hypothesis unlikely. If some performances are incomplete, then obtaining a maximum likelihood estimate is significantly more complicated. If the positions are missing at random [115], then algorithms exist [94] that can obtain maximum likelihood estimates of the covariance. Unfortunately, the positions are for the most part definitely not missing at random. In motion capture, the probability of a marker becoming occluded is mostly dependent upon its position. For example, the finger tips become occluded whenever a fist is made. Therefore, since no obvious solution exists to obtain biased or unbiased maximum likelihood estimates, we choose to perform random imputation by sampling $\mathcal{N}(\hat{\boldsymbol{\mu}}^{(i)}, \hat{\Sigma}^{(i-1)})$. Random imputation is deemed preferable to mean imputation since the latter will certainly underestimate the variance. Regardless of the method, note that this estimated covariance $\hat{\Sigma}^{(i)}$ is technically an estimate of both the performance and measurement covariance.

Regarding (4.23), Laplace smoothing should be performed since we cannot be absolutely certain that a target will or will not appear, which amounts to adding fictitious point sets $\tilde{\mathbb{Y}}_i$, $i \in [1, m]$, such that $\tilde{\mathbb{Y}}_i$ contains a single point assigned with absolute certainty to the i-th target. The steps of the algorithm follow.

1. Initialize $\hat{\mathbf{x}}_0^{(0,0)}$ to be a random measurement vector augmented with extra random values if needed and set $\hat{\Sigma}^{(0)}$ diagonal.
2. For $i = 1, 2, 3, \ldots$ until convergence,
   (a) Set $\hat{P}_0^{(i,0)} = \hat{\Sigma}^{(i-1)}$.
   (b) Randomly permute $\mathbb{Y}_{1:N}$ to obtain $\mathbb{Y}_{1:N}^{(i)}$.
   (c) For each measurement in $\mathbb{Y}_{1:N}^{(i)}$, perform a Kalman filter update using (4.7) and (4.8) to obtain $\hat{\mathbf{x}}_N^{(i,0)}$ and $\hat{P}_N^{(i,0)}$.
   (d) For $j = 1, 2, 3, \ldots$ until convergence,
      i. Set $\hat{\mathbf{x}}_0^{(i,j)} = \hat{\mathbf{x}}_N^{(i,j-1)}$ and $\hat{P}_0^{(i,j)} = \hat{P}_N^{(i,j-1)}$.
      ii. For each measurement in $\mathbb{Y}_{1:N}^{(i)}$, calculate *a fortiori* estimates using the procedure outlined in Section 4.2.2 to obtain $\hat{\mathbf{x}}_N^{(i,j)}$ and $\hat{P}_N^{(i,j)}$.
   (e) Calculate $\hat{\boldsymbol{\mu}}^{(i)}$, $\hat{\Sigma}^{(i)}$, and $\hat{\boldsymbol{\pi}}^{(i)}$ according to (4.21), (4.22), and (4.23).
   (f) Set $\hat{\mathbf{x}}_0^{(i+1,0)} = \hat{\boldsymbol{\mu}}^{(i)}$.

We make some final remarks to clarify the algorithm. Convergence is guaranteed since this is an EM algorithm and is indicated by small changes in the log-likelihood or parameter values. Step 2a presents a natural choice for the initial state error covariance $\hat{P}_0$ in that if each measurement set is complete, then $K_k = \mathbf{I}/(k+1)$. Furthermore, if $\hat{P}_0 = \alpha\Sigma$, then $\lim_{\alpha\to\infty} K_k = \mathbf{I}/k$.

4.3 Experimental Evaluation

This section describes simulations and measures of evaluation for AFEM. In particular, we compare our algorithm versus a more traditional (sans Kalman filter) EM algorithm. In addition, we compare the MCMC sampler versus linearized optimal

assignments, which respectively yield approximate soft and hard EM algorithms. The expectation is that the MCMC sampler will yield more accurate profile estimates, although we need experimental data to verify. To our knowledge, we are the first to apply the sampler to this type of problem.

4.3.1 Simulation Description

Since available real-world data does not come with labeled targets, we resorted to simulated data to evaluate the algorithm under a variety of conditions. Simulations involved independently varying the number and spread of targets sampled from some randomly generated true distribution. The parameters of each simulation included the number of targets m, the dimensionality of each target d, the spread of the targets σ, and whether the true distribution's targets are constrained to be independent or not (i.e. whether Σ is constrained to be block-diagonal or not). Target dimensionalities of $d = 2$ and $d = 3$ were considered.

For each considered combination of parameters, five random true distributions were generated. The m components for each distribution were drawn uniformly at random from within the volume of a d-dimensional hypercube and ranged from $m = 2$ to $m = 18$ in steps of 4. If varying the number of targets, the hypercube was scaled such that the intensity (the expected number of targets as a function of position) was kept approximately constant. To be precise, the length of a side was equal to $\lfloor 100m^{1/d} \rfloor$. The same side lengths were used to vary the spread when keeping the number of components fixed at $m = 10$ (that is, 10 targets were placed in volumes sized for 2, 6, 10, 14, and 18 targets). The true covariance matrix was sampled from

a Wishart distribution [103] with md degrees of freedom and a scale parameter of $\mathbf{I}_{md}$ before being multiplied by a factor of 100. The parameters for this Wishart distribution were chosen to ensure with high probability that the correlations between targets were significant, and the factor of 100 increases the likelihood of target overlap. The elements of the true appearance probability vector were independently sampled from the uniform distribution on $[0, 1]$.

For each distribution thus generated, 400 sets were sampled and their elements randomly permuted. Empty sets were kept as they contain information about $\boldsymbol{\pi}$. Three generators or samplers of measurement matrices were considered: the optimal (linearized) assignment, the 10 best linearized assignments, and a sequential match MCMC sampler with a default temperature of 1. These samplers are denoted in figures, respectively, as "Best-1," "Best-10,", and "MCMC-1." Burn-in of 10% was used for 1000 samples initialized from a random permutation. All of the remaining 900 samples were used. Thoroughly tuning the sequential match sampler was not the objective of this dissertation.

4.3.2 Comparing Profiles

Even if we know the true distribution, we are faced with a conundrum: which target is which? Even if we have perfectly estimated the true parameters, their order may be scrambled. We must therefore consider assignments between estimated and true targets, and these assignments need to respect the target positions, covariance, and appearance probabilities.

Let us first suppose that the correspondence between targets is known. In such a case, we propose using the Hellinger distance (Section 2.2.1) to compare the profiles. Since H^2 be computed in closed form for Gaussian densities, it can also be computed in closed form for profiles with known correspondence.

There are some issues, however, that make H^2 and similar metrics such as KL or JS divergence less than ideal for comparing profiles. For example, if just one target's position is wrong, H^2 tends towards 1. Similarly, differences in the structure of the covariance matrices can push H^2 towards 1 even if each position is correct. We therefore propose modifying H^2 in such a way that the comparison of individual targets and the covariance is separated. Define the modified squared Hellinger distance between profiles $P \sim \mathcal{P}(\mathbf{x}, \Sigma, \boldsymbol{\pi})$ and $Q \sim \mathcal{P}(\boldsymbol{\chi}, \Gamma, \boldsymbol{\tau})$ as

$$\mathcal{H}^2(P, Q) = 1 - \mathcal{H}_1^2(P, Q)\mathcal{H}_2^2(P, Q), \tag{4.24}$$

where

$$\mathcal{H}_1^2(P, Q) = 1 - \frac{1}{m}\sum_{i=1}^{m} H^2(\mathcal{P}(\mathbf{x}_i, \Sigma_i, \pi_i), \mathcal{P}(\boldsymbol{\chi}_i, \Gamma_i, \tau_i)), \tag{4.25}$$

$$\mathcal{H}_2^2(P, Q) = 1 - H^2(\mathcal{N}(\mathbf{0}, \Sigma), \mathcal{N}(\mathbf{0}, \Gamma)), \tag{4.26}$$

$\mathbf{x}_i$, $\boldsymbol{\chi}_i$ are the coordinates of the i-th targets of each profile, and Σ_i, Γ_i are the $d \times d$ diagonal blocks corresponding to the internal covariance of the i-th targets. One may note that $\mathcal{H}_1^2$ calculates the average similarity between individual targets whereas $\mathcal{H}_2^2(P, Q)$ calculates the similarity between the overall covariances ignoring positions. Note that $\mathcal{H}^2$ is still negative definite.

Negative definiteness is a potentially useful property as it implies that H^2 is isometric to L^2 and can be interpreted in a manner similar to the Euclidean distance as the logarithm of some (perhaps Gaussian) probability density function. Hence, one could set e^{-H^2} as the target density for a sequential match sampler that is used to approximate the expected value of the Hellinger distance when the correspondence between targets is not known. However, H^2 comes with an implicit variance and should be scaled depending on the application. We chose to compare two profiles using the correspondence that maximized $\mathcal{H}_1^2(P,Q)$, which can be computed in polynomial time. We did not attempt to directly minimize $\mathcal{H}^2(P,Q)$ since the objective ends up being concave and NP-Hard to solve.

To balance the results that use our custom metric, we also present results based upon the Rand index [112], a standard measure of cluster similarity that is less affected by target correspondence. The Rand index provides a means to assess the accuracy of a clustering without knowing cluster identity. Instead of operating on individual clustered points, the Rand index operates on pairs. Let $C = \bigcup_{i=1}^{c} C_i$ and $D = \bigcup_{j=1}^{d} D_j$ be partitions of a dataset $\mathbb{X}$ into disjoint subsets C_i and D_j. Let a and b be the number of pairs of points in $\mathbb{X}$ that belong respectively to the same or different set in C and the same or different set in D. The Rand index is then defined to be

$$R(C,D) = \frac{a+b}{\binom{n}{2}}. \tag{4.27}$$

For our purposes, $\mathbb{X} = \bigcup_k \mathbb{Y}_k$, the clusters are the targets, and points are assigned to clusters based upon the linearized mode of H_k (i.e. constraints for the data are known).

4.3.3 Results

Results focus on $\mathcal{H}_1^2$ and $\mathcal{H}_2^2$, which respectively measure the difference between individual targets or total covariances, and the Rand index, which measures the accuracy of the estimated profile when used as a clustering algorithm. In general, it was observed that AFEM tended to lead to better values of each measure regardless of the sampler or simulation parameters with few exceptions (for example, see Figure 4.1). Differences were more dramatic for three dimensions than for two when comparing the EM algorithms and comparing samplers. The primary situations in which the algorithm did not usually confer improved measures of accuracy were when the number of targets was relatively low (e.g. Figure 4.2) and when the MCMC sampler was used for covariance estimation. The MCMC sampler's occasional poor performance can likely be attributed to the fact that it was not carefully tuned for the problem.

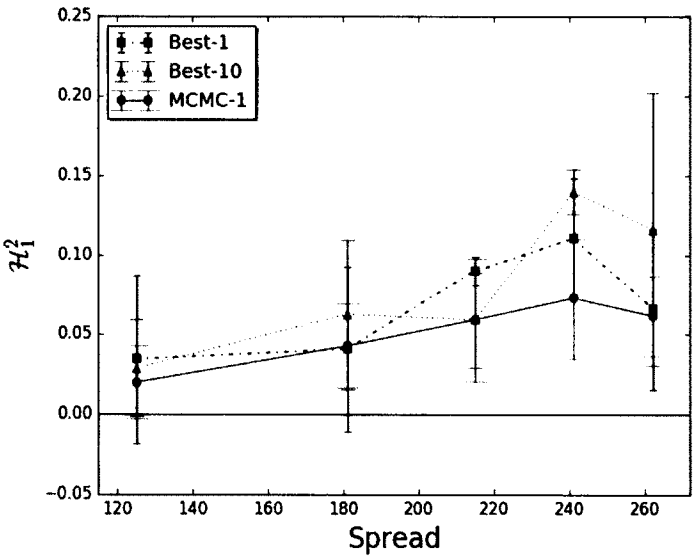


Figure 4.1: The improvement in $\mathcal{H}_1^2$ for each sampler when using AFEM versus normal EM as a function of spread for $d = 3$ and unconstrained true covariance.

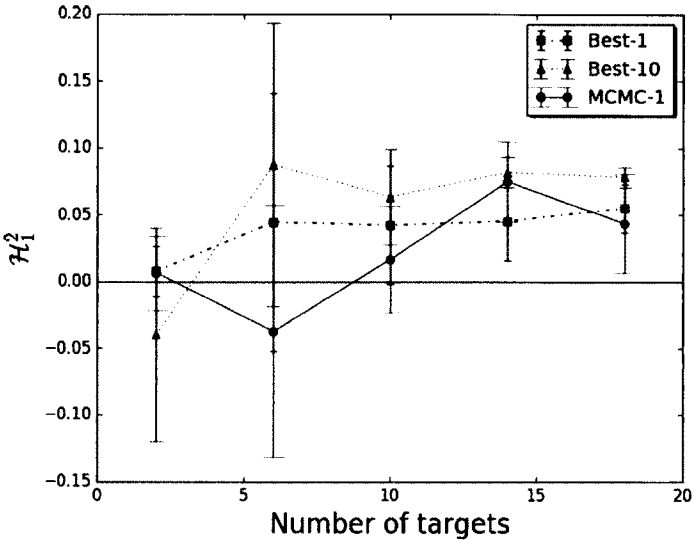


Figure 4.2: The improvement in $\mathcal{H}_1^2$ for each sampler when using AFEM versus normal EM as a function of target count for $d = 2$ and unconstrained true covariance.

Due to the large number of simulations, only a small subset are shown via Figures 4.1, 4.2, 4.3, 4.4, and 4.5. Descriptions of the rest are conveyed through general comments and observations. More detailed comments for each measure can be found in the following subsections.

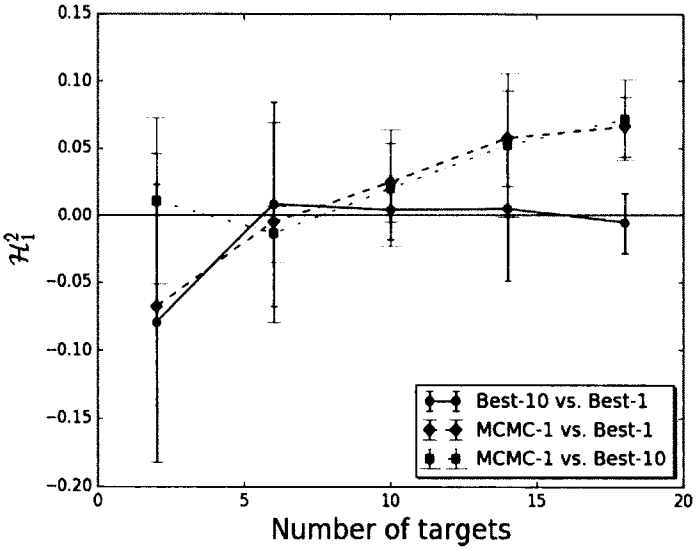


Figure 4.3: The improvement between each pair of samplers in $\mathcal{H}_1^2$ for AFEM as a function of the number of targets for $d = 3$ and unconstrained true covariance.

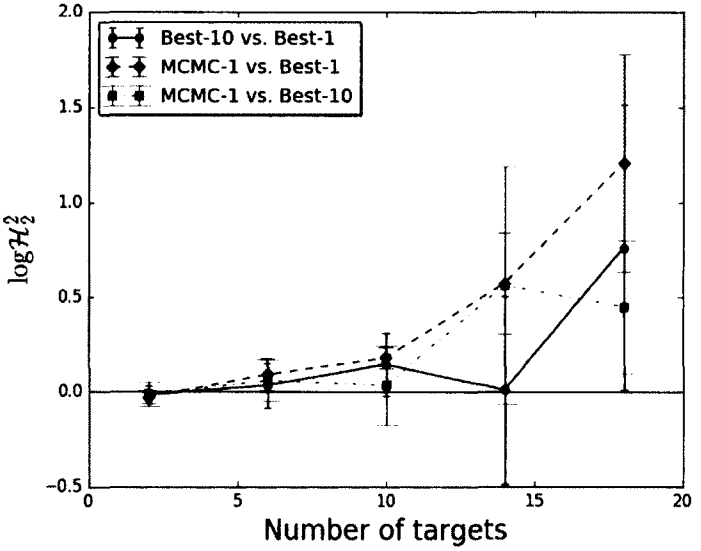


Figure 4.4: The improvement between each pair of samplers in $\log \mathcal{H}_2^2$ for AFEM as a function of the number of targets for $d = 3$ and unconstrained true covariance.

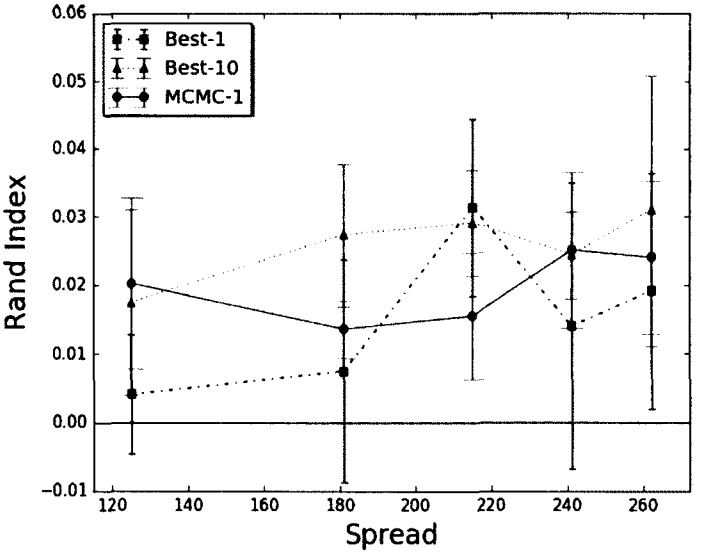


Figure 4.5: The improvement in the Rand index for each sampler when using AFEM versus normal EM as a function of spread for $d = 3$ and unconstrained true covariance.

Target Estimation

Target accuracy exhibited a fairly clear dependency on the spread in that increasing the spread seemed to yield further improvements (see Figure 4.1). The MCMC sampler tended to receive the least absolute benefit from the Kalman filter, but

also tended to yield better estimates than either of the optimal assignment samplers for larger target counts (see Figure 4.3). A slight bias in favor of the MCMC sampler was also observed for smaller spreads. The optimal assignment sampler performed worst overall as should be expected. The MCMC sampler also benefited from the extra dimension in $d = 3$.

Covariance Estimation

Since H_2^2 tended to be close to zero, plots show $\log H_2^2$ instead. AFEM was generally better than traditional EM except when the covariance was constrained, in which case the MCMC sampler tended to perform worse. Even though AFEM conferred no advantage to the MCMC sampler, MCMC still performed better than its peers for larger target counts ($m > 10$) in nearly all simulations (see Figure 4.4). No advantage in the MCMC sampler was observed for varying spread, although this may have been due to the target count being too low.

Clustering Accuracy

Results with the Rand index agreed with $\mathcal{H}_1^2$ and $\mathcal{H}_2^2$ in that AFEM generally conferred an advantage for each sampler. Since simultaneously observed targets are guaranteed to be placed into different clusters, each method is guaranteed to agree on a large fraction of b. Differences tended to be relatively small as a result (see Figure 4.5).

CHAPTER 5

ON THE DEFINITENESS OF EARTH MOVER'S DISTANCE AND ITS RELATION TO SET INTERSECTION

Recall the definition of the Wasserstein distance and EMD given in Section 2.4. The foundations of EMD's definiteness as a kernel (Section 2.1.4) are the primary topic of this chapter. EMD has been applied in kernel methods for texture and object category classification with SVMs [156]. However, it is not known whether kernels derived from EMD are actually PD. In fact, there is evidence to the contrary for a Euclidean ground distance [100]. Regardless, EMD continues to be used successfully for various purposes such as facial expression analysis [117] and EEG classification [27]. Methods to ensure PD-ness have been explored [155]. Cuturi [23] suggested using the permanent of the transportation polytope, which is guaranteed to be PD although difficult to compute. Grauman and Darrell [47] on the other hand proposed a PD approximation of a maximum-cost version of EMD that also has the advantage of being easier to compute.

In Section 5.2, we propose the PD-preserving transformation (5.4) that can be applied to any kernel, and we provide a new proof of the Jaccard index's PD-ness, which has already been the subject of at least two papers [45, 10]. Under certain conditions, the transformation may even induce PD-ness. As a corollary, we deduce

that the biotope transform [31] preserves CND-ness in addition to metric properties. In Section 5.3, we show that given certain ground distances, EMD is CND and may thus be used to construct PD kernels using standard relations. In particular, in Section 5.3.1 we use a set theoretic interpretation of EMD to show how EMD generalizes the intersection kernel. With a special emphasis on unnormalized sets, we generalize $\widehat{\text{EMD}}$ [105] for use as a kernel. In addition, we show in Section 5.3.2 that convex, non-negative symmetric ground distances of the form $h(x-y)$ for $x, y \in \mathbb{R}$ and some h yield CND EMD on the real line. On the circle in Section 5.3.3, we find that EMD is not in general CND, although a CND approximation can be found by substituting the mean for the median in a calculation. In Section 5.3.4, we apply (5.4) to transform ground distances to the form $\beta - K$ such that CND-ness in EMD is induced. Finally, we evaluate $\widehat{\text{EMD}}$ and the transformation on a variety of experiments, showing that both yield kernels superior to EMD, especially on unnormalized sets. Throughout the chapter we find that EMD is related to min-like kernels including intersection, Brownian bridge, and the Jaccard index.

The next section presents information including definitions and theorems that may be used as a reference for the rest of the chapter.

5.1 Preliminaries

This section provides definitions that are useful for following the rest of the chapter. A review of kernels (Section 2.1.4) and measures (Section 2.1.5) is advised. In addition, recall the definitions of $\mathcal{F}(X)$ and $\mathcal{P}(X)$ given in Section 2.1.6.

5.1.1 Multisets

A *multiset* generalizes a set by allowing duplicate elements. We use the terms multiset and set interchangeably with context indicating which is meant in the strict sense. By definition, the multiplicity of an element x is a non-negative integer indicating how many copies of x are contained in a given multiset A. We generalize this definition by allowing a non-negative real number of "copies." With this definition, we may also include probability distributions and other continuous functions with real output.

Let X be the set of all possible elements under consideration, i.e. the domain. Let $\chi_A : X \to \mathbb{R}$ be the mass density (or multiplicity) function of the multiset A that indicates the multiplicity of each $x \in X$ contained in A. For any element x not contained in A, $\chi_A(x) = 0$. The density function completely defines a multiset and is similar to a probability density without the restriction that it must sum to 1. When we refer to a multiset or density, the other is implied. Note that for a standard set A (i.e. not multiset), $\chi_A = \mathbb{I}_A$. The mass density function of A gives rise to a measure

$$\mu_A(Y) = \int_Y \chi_A(x)\mathrm{d}x. \tag{5.1}$$

For discrete sets, (5.1) simplifies to series summation. The support of a multiset is the same as the support of its measure. We use the term singleton to denote a multiset A with support satisfying $\mathrm{supp}\,(A) = \mathrm{supp}\,(\mu_A) = \{x_0\}$ for some fixed element $x_0 \in A$.

We generalize the definition of a subset $A \subseteq B$ in X to be such that $\chi_A(x) \leq \chi_B(x)$ for each $x \in X$. The density function for the intersection of two multisets A

and B is defined as

$$\chi_{A\cap B}(x) = \min\{\chi_A(x), \chi_B(x)\}, \tag{5.2}$$

and the union is defined in a similar manner with max instead of min.

Henceforth, we will abuse notation by defining $\mu(A) = \mu_A(\text{supp}(A))$, which we will refer to interchangeably as the size, mass, or measure of A. Note that unlike histograms, multisets do not imply a finite, countable base set X from which every set draws its support. This distinction allows somewhat more flexible definitions of EMD.

5.1.2 Earth Mover's Distance

We consider EMD to be a metric on $\mathcal{F}(X)$ for some set X. Note that here we mean metric as in dissimilarity measure. Based on the equivalency set forth between measures and multisets in the previous section, we henceforth consider EMD to act directly on the sets as in (5.3) instead of their associated measures. Recall that EMD is not a true metric in the sense of Definition 2.1 on $\mathcal{F}(X)$ but rather on $\mathcal{P}(X)$ for metric ground distance [116]. Violations of identity and triangle inequality are easily found when considering subsets and supersets.

Recall that EMD involves calculating the minimum-cost maximum flow via the linear program given by (2.70), (2.71), (2.72), and (2.73). Note that our definition of EMD differs slightly from that of Rubner *et al.* [116], which scales (2.70) by the inverse of the total flow given in (2.73). For sets of the same size, Rubner's definition is just (2.70) scaled by a constant factor. We say that a collection of multisets is normalized if each set is the same size or mass. The sets are unnormalized if any two sets are not the same size. Pele and Werman [105] introduced a means to calculate

EMD between unnormalized sets for use in nearest neighbor calculations and image retrieval and defined it as

$$\widehat{EMD}_\alpha(A,B) = EMD(A,B) + \alpha|\mu(A) - \mu(B)|\max_{a,b\in X}\{d(a,b)\}, \tag{5.3}$$

where A and B are sets, $\alpha \geq 0$ and d is presumed to be bounded. $\widehat{\text{EMD}}_\alpha$ is a metric on $\mathcal{F}(X)$ if EMD is a metric on $\mathcal{P}(X)$ and $\alpha \geq 0.5$ [105]. Schuhmacher *et al.* [119] independently proposed an almost identical version of $\widehat{\text{EMD}}$ under the acronym OSPA (Optimal Subpattern Assignment). Normalized forms of EMD such as (3.10) have also been proposed, although a connection to EMD was not explicitly acknowledged by Ramon and Bruynooghe [111]. The transformation of the following section was inspired by the search for and study of a normalized form.

5.2 A Definite Preserving Transformation

In this section we propose the PD-preserving transformation

$$K_T(x,y) = \frac{K(x,y)}{K(x,x) + K(y,y) - K(x,y)} \tag{5.4}$$

that normalizes any given PD kernel K. If $K(x,x) = K(y,y) = 0$, we define $K_T(x,y) = 1$. As opposed to the traditional normalization,

$$K_N(x,y) = \frac{K(x,y)}{\sqrt{K(x,x)K(y,y)}}, \tag{5.5}$$

which can be interpreted as a surjective mapping of images $\phi(x)$ in Hilbert space onto the unit hypersphere via projection, K_T can be interpreted as an injective mapping onto a unit hypersphere of unspecified dimension. Image vectors in Hilbert space

of different magnitude that share the same direction remain distinguishable post-transformation. Both transformations are nearest neighbor preserving for points on the same hypersphere.

Technically, this kernel (or one algebraically equivalent to it) has been proposed before as the Tanimoto kernel by Ralaivola *et al.* [110]. We stress the differences in our proposed transformation and how our contributions differ from existing work. First, the Tanimoto kernel is equivalent to the Jaccard index and has only been proved PD when X consists solely of binary vectors and K is the dot product (see the proof given by Ralaivola *et al.*, which hinges on the proof of semi-PD-ness of the Jaccard index given by Gower [45]). We prove (see Theorem 5.1) that (5.4) is strictly PD for any K if K is strictly PD (and similarly for semi-definiteness), which is stronger than the proof of Ralaivola *et al.* and more general than both it and the proof of strict PD-ness of the Jaccard index given by Bouchard *et al.* [10]. Since we are not limited to binary vectors, the range of (5.4) is not even constrained to be positive. This more general view of the transformation also allows us to examine its properties in new situations, such as when it is applied to itself or *nested.*

In fact, the transformation can be nested indefinitely as in

$$K_T^{(n)}(x,y) = \frac{K_T^{(n-1)}(x,y)}{K_T^{(n-1)}(x,x) + K_T^{(n-1)}(y,y) - K_T^{(n-1)}(x,y)}, \tag{5.6}$$

such that

$$\lim_{n\to\infty} K_T^{(n)}(x,y) \in \{0,1\}, \tag{5.7}$$

where $K_T^{(n)}$ is the n-th nested transformation with $K_T^{(0)} = K$. A closed form expression for the n-th nested transformation can be derived and is given by

$$K_T^{(n)}(x,y) = \frac{K(x,y)}{2^{n-1}[K(x,x)+K(y,y)] - (2^n-1)K(x,y)}. \tag{5.8}$$

From Lemma 2.4, the denominator converges to $2^{n-1}D(x,y)$, where $D(x,y) = K(x,x)+K(y,y)-2K(x,y)$. Geometrically, we may then loosely interpret the transformation as division of the inner product by the distance squared. Note that although we focus on $n = 1$, n could be considered a continuous hyperparameter within the range $(-\infty,\infty)$ for which (5.8) is PD on the subinterval $[0,\infty)$. In fact, if $n = 0$, then we obtain a generalization of the F measure (as interpreted as a kernel by Ralaivola *et al.* [110])

$$K_T^{(0)}(x,y) = \frac{2K(x,y)}{K(x,x)+K(y,y)}. \tag{5.9}$$

In the next section, we will use this transformation to define new EMD-based kernels and to define ground distances for which EMD is CND. First, however, we must show that the transformation preserves definiteness as claimed.

Theorem 5.1. *If $K : X \times X \to \mathbb{R}$ is PD, then the function K_T as defined by (5.4) is also PD.*

Proof. Without loss of generality, assume $K(x,x) = K(y,y) = 0 \implies x = y = p$ for some $p \in X$ and let us restrict K in the following discussion to $X \setminus \{p\}$. The denominator in (5.4) is positive valued due to a well-known property of PD kernels and matrices,

$$|K(x,y)| \leq \frac{K(x,x)+K(y,y)}{2} < K(x,x)+K(y,y). \tag{5.10}$$

The denominator is also CND as it is the sum of two CND kernels: $K(x,x)+K(y,y)$ (by Proposition 2.6) and $-K(x,y)$ (by hypothesis). Thus by Proposition 2.7 with $\gamma = 1$,

$$K_1(x,y) = [K(x,x) + K(y,y) - K(x,y)]^{-1} \tag{5.11}$$

is PD. We therefore have the product of two PD kernels

$$K_T(x,y) = K(x,y)K_1(x,y), \tag{5.12}$$

which is itself PD by Theorem 2.5.

In order to include the case $x = y = p$, we note that if $\phi : X \to H$ is the kernel's feature mapping into the Hilbert space H, then $K(p,p) = \langle \phi(p), \phi(p) \rangle = 0 \implies \phi(p) = \mathbf{0}$, which further implies

$$K(p,x_i) = \langle \phi(p), \phi(x_i) \rangle = \langle \mathbf{0}, \phi(x_i) \rangle = 0 \tag{5.13}$$

for $x_i \neq p$. Therefore, $K_T(x_i,p) = 0$ if $x_i \neq p$. Let $x_0 = p$ and $c_0 \in \mathbb{R}$. Then K_T is PD because

$$\sum_{i,j=0}^{n} c_i c_j K_T(x_i,x_j) = c_0^2 + \sum_{i,j=1}^{n} c_i c_j K_T(x_i,x_j) \geq 0. \tag{5.14}$$

□

Corollary 5.2. *Let $D : X \times X \to \mathbb{R}$ be a CND kernel, and let $p \in X$. Then,*

$$D_{T,p}(x,y) = \frac{2D(x,y) - D(x,x) - D(y,y)}{D(x,p) + D(y,p) + D(x,y) - \sum\limits_{z \in \{x,y,p\}} D(z,z)}, \tag{5.15}$$

is also CND.

Proof. We can define a PD kernel K according to the relation given by Lemma 2.4, i.e.

$$K(x,y) = D(x,p) + D(y,p) - D(x,y) - D(p,p). \tag{5.16}$$

Using (5.16), note that $K(x,x) = 2D(x,p) - D(x,x) - D(p,p)$. Furthermore, note that

$$\begin{aligned} K(x,x) + K(y,y) - K(x,y) = D(x,p) + D(y,p) + D(x,y) \\ - D(x,x) - D(y,y) - D(p,p). \end{aligned} \tag{5.17}$$

We see that the denominator of D_T is the same as that of K_T. Note then that

$$K_T(x,y) + D_{T,p}(x,y) = 1. \tag{5.18}$$

If $x_1, \ldots, x_n \in X$, $c_1, \ldots, c_n \in \mathbb{R}$, and $\sum_{i=1}^n c_i = 0$, then

$$\sum_{i,j=1}^{n} c_i c_j D_{T,p}(x_i, x_j) = \sum_{i,j=1}^{n} c_i c_j \left(-K_T(x_i, x_j)\right) \leq 0. \tag{5.19}$$

We have thus shown that $D_{T,p}$ is CND. □

If $K(x,y) \geq 0$, then $K_T(x,y) \in [0,1]$. Otherwise, $K_T(x,y) \in [-1/3, 1]$. Consequently, $D_{T,p}(x,y) \in [0, 4/3]$ and $D_{T,p}(x,y) > 1$ if and only if $D(x,y) + D(p,p) > D(x,p) + D(y,p)$. In addition, Theorem 5.1 also holds for strictly PD K. Using Theorem 5.1 with K as the intersection kernel therefore provides an easy proof for the PD-ness of the Jaccard index,

$$J(A,B) = \frac{\mu(A \cap B)}{\mu(A \cup B)}. \tag{5.20}$$

Note that $D_{T,p}$ generalizes the well-known biotope transform [31], showing that it preserves CND-ness in addition to metric properties. As an example, suppose A and B are sets and $D(A,B) = |\mu(A) - \mu(B)|$. This kernel is CND. By Corollary 5.2 with

$p = \emptyset$ followed by some simplification, we can derive the CND kernel

$$D_{T,\emptyset}(A, B) = \frac{|\mu(A) - \mu(B)|}{\max\{\mu(A), \mu(B)\}}. \tag{5.21}$$

The transformation K_T possesses another interesting property in that it can induce PD-ness in addition to preserving it. The following proposition gives a sufficient condition under which this phenomenon occurs for nested transformations. We hypothesize that the proposition holds simply if X is finite and K satisfies the equivalence relation

$$x \equiv y \iff 2K(x, y) = K(x, x) + K(y, y). \tag{5.22}$$

Proposition 5.3. *For any symmetric kernel $K : X \times X \to \mathbb{R}$ where X is finite and $x \neq y \implies 2K(x, y) \neq K(x, x) + K(y, y)$ for $x, y \in X$, there exists a number n_0 such that $K_T^{(n)} : X \times X \to \mathbb{R}$ is PD for all $n \geq n_0$.*

Proof. Consider the kernel matrix $G_K^{(n)} = [K_T^{(n)}(x_i, x_j)]$ for some selection of elements $x_1, \ldots, x_n \in X$ with $1 \leq i, j \leq n$. Since the definition of a PD kernel requires only distinct elements for (2.19), we may without loss of generality assume that each element is distinct, i.e. $i \neq j \implies x_i \neq x_j$. We now show that $G_K^{(n)}$ must eventually become diagonally dominant and thus PD [12] as n increases, where diagonal dominance for a symmetric matrix M is defined for each row index i by

$$|M_{ii}| \geq \sum_{j \neq i} |M_{ij}|. \tag{5.23}$$

We show this by noting that each transformation is effectively a step in a fixed point iteration wherein $G_K^{(n)}$ converges to identity. We allow infinite values in (5.4) due to division by zero as these can be removed by further transformations described shortly.

Note that for $n \geq 1$, $K_T^{(n)}(x, y) = 1$ if and only if $x = y$, and since $K_T^{(n)}(x, x) = 1$ for all x and $n \geq 1$,

$$K_T^{(n+1)}(x_i, x_j) = \frac{K_T^{(n)}(x_i, x_j)}{2 - K_T^{(n)}(x_i, x_j)}. \tag{5.24}$$

Consequently, repeated transformations with fixed i, j are effectively identical to fixed point iteration with fixed points of 0 and 1, and any sequence $K_T^{(1)}, K_T^{(2)}, \ldots$ starting at $K_T^{(1)}(x_i, x_j) \neq 1$ will converge to 0 [12]. An infinite value in any sequence is followed by -1, obtained by observing the limits at infinity of (5.24). We can then deduce that there must exist an m such that for any $n \geq m$ and $i \neq j$, $|K_T^{(n)}(x_i, x_j)| < |K_T^{(n-1)}(x_i, x_j)|$. Therefore, as the number of nested transformations increases beyond the m-th, the diagonal of $G_K^{(n)}$ stays constant at a value of 1 and the absolute value of each off-diagonal element decreases. Eventually, $G_K^{(n)}$ must become diagonally dominant and hence PD for all n greater than or equal to some finite $n_0 \geq m$. □

5.3 EMD Is Conditionally Negative Definite For Certain Ground Distances

In this section, we introduce new results on ground distances and conditions under which EMD can be proved to be CND. In some cases, we offer CND approximations.

Since any ground distance is just a special case of EMD between singletons of unit mass, EMD is CND only if the ground distance is CND. Unless otherwise noted, we will assume that ground distances discussed henceforth are CND.

5.3.1 Earth Mover's Intersection: A Set Theoretic Interpretation of EMD

In this section we introduce earth mover's intersection (EMI), a useful concept and PD analog to EMD that computes the similarity between two sets rather than their difference for a given ground distance. The name comes from the following motivating scenario.

Suppose there are two sets of two-dimensional points where one is a slightly perturbed version of the other. According to the strict definition of set intersection given by (5.2), their intersection is empty despite the fact that they are clearly related by their elements. The inability of set intersection to account for the sets' inherent similarity is a problem. EMD provides a natural solution to this problem, although it is proportional to their difference rather than similarity. EMD also reflects the qualities of whatever norm is chosen to compare the individual points. We now show that EMD and subsequent related functions define smooth (in the sense of strictness of equality) generalizations or approximations of classic set operations.

Sets are usually normalized prior to application of EMD by dividing their density function by their total mass, an operation analagous to normalizing a vector to unit norm. The disadvantage of this method is that sets with differently scaled but otherwise identical density functions become indistinguishable post-normalization. As a side-effect, one removes an entire dimension of the data (for the most extreme case, consider singleton point sets with non-negative mass on the real line). An application where this distinction is important is that of multi-object tracking and filtering [119, 113]; normalizing set mass can cause one to ignore the fact that the incorrect number of objects are being tracked. For our set theoretic interpretation

of EMD, we prefer to retain the sets' original mass and transport excess mass to a predetermined point $p \in X$, referred to as the sink. One could also consider this a form of additive normalization by supplementing mass at the point p. EMD then more accurately represents the relative magnitudes of set differences as well as distinguishes differently scaled sets.

Define the term $\overline{EMD}_p$ to represent the transportation of excess mass from the larger of two sets A and B to some sink $p \in X$, i.e.

$$\overline{EMD}_p(A,B) = \sum_{b \in B} \left(\chi_B(b) - \sum_{a \in A} f^*(a,b) \right) \left[D(b,p) - \frac{D(p,p)}{2} \right], \tag{5.25}$$

where D is the ground distance, f^* is the optimal flow, and we assume without loss of generality that $\mu(A) \leq \mu(B)$. The total cost of transforming one set into another is then given by

$$\widehat{EMD}_p(A,B) = EMD(A,B) + \overline{EMD}_p(A,B), \tag{5.26}$$

where we have adopted the notation for Pele and Werman's $\widehat{\text{EMD}}_\alpha$. Note that the sink does not necessarily have to be in X (in which case we must replace D with an appropriate function in (5.25)). Ideally, though, p is a reserved point that does not naturally appear in the sets under consideration. Otherwise, there is a different type of potential identity loss.

We define EMI as the kernel resulting from Lemma 2.4 with $x_0 = \emptyset$ and $D = \widehat{EMD}_p$, which is

$$EMI_p(A,B) = \widehat{EMD}_p(A,\emptyset) + \widehat{EMD}_p(B,\emptyset) - \widehat{EMD}_p(A,B). \tag{5.27}$$

Note that EMI is PD whenever $\widehat{\text{EMD}}$ is CND for some collection of sets (and vice versa). By assuming $p \in X$, we can define a PD kernel K_p according to Lemma 2.4 with $x_0 = p$ and D as the ground distance, which we can then use with (5.25) to simplify EMI to

$$EMI_p(A, B) = \sum_{a \in A} \sum_{b \in B} f^*(a, b) K_p(a, b). \tag{5.28}$$

Observe that the minimum-cost maximum flow with respect to D is the same as the maximum-cost maximum flow with respect to K_p, regardless of the choice of p. As a result, EMI can hypothetically be specified in terms of just a PD ground distance without explicitly specifying the sink. The definition of EMI also provides some insight into the pyramid match kernel [47], which can be viewed as an approximation of $EMI_{\mathbf{0}}$ on $\mathcal{F}(\mathbb{R}^n)$. We may also consider an alternative definition $EMI'_p(A, B) = EMI_p(A, B) + \sum \sum f^*(a, b) D(p, p)$ that is also PD if $D(p, p) \geq 0$ and $\widehat{\text{EMD}}$ is CND; this is equivalent to discarding $D(p, p)$ in (5.25).

As our first example of a situation in which EMI is PD on $\mathcal{F}(X)$ (and hence $\widehat{\text{EMD}}$ and EMD are respectively CND on $\mathcal{F}(X)$ and $\mathcal{P}(X)$), consider the discrete metric, which can trivially be verified to be CND. Define the *discrete kernel* corresponding to this ground distance to be $K_{0\text{-}1}(x, y) = 1 - \delta_{0\text{-}1}(x, y)$, which is PD. We can show that EMI in this case is equivalent to the intersection kernel.

Proposition 5.4. *Let $EMI_{0\text{-}1}(A, B)$ be EMI equipped with the discrete kernel as the ground distance on an arbitrary set X. Then $EMI_{0\text{-}1}$ is equivalent to the intersection kernel.*

Proof. The goal is to find the maximum-cost maximum flow subject to constraints, and the only way to increase the cost with the discrete kernel is to send available mass from a point in one set up to the capacity allowed by the other set at the same location. Therefore, $f^*(a,a)$ will be saturated up to the available capacity at a in each set, i.e.

$$f^*(a,a) = \min\left\{\chi_A(a), \chi_B(a)\right\}. \tag{5.29}$$

The cost to transport this mass is simply the amount of mass transported. The exact mapping of the remaining mass is irrelevant as it costs nothing to move. As a result,

$$EMI_{0\text{-}1}(A,B) = \sum_{a \in A \cup B} f^*(a,a) = \mu(A \cap B). \tag{5.30}$$

□

Since the intersection kernel is PD [11], we conclude that $EMI_{0\text{-}1}$ is as well. One can then deduce that $EMD_{0\text{-}1}$ and $\widehat{EMD}_{0\text{-}1}$ give measures of the set difference between A and B. Specifically, $EMD_{0\text{-}1}$ gives the set difference of the larger set from the smaller, and $\widehat{EMD}_{0\text{-}1}$ gives the set difference of the smaller set from the larger. The sum of both yields the symmetric difference. One may also apply (5.4) with $K = EMI_{0\text{-}1}$ or (5.15) with $D = \widehat{EMD}_{0\text{-}1}$ and $p = \emptyset$ to obtain the Jaccard index and distance.

Switching to a ground distance other than the discrete metric is like allowing a degree of uncertainty in element identity. The sharper or more concave the comparison function, the closer EMD and its derivatives are to their respective binary set operations. The point p is used to determine the cost of an unmatched element, which could potentially vary if some point is considered more important than another. Practically,

thresholding a ground distance by some upper bound can be used to artificially induce concavity and make comparisons more strict.

Another result that can be derived as a special case of EMI follows.

Proposition 5.5. *If there exists a function $g : X \to \mathbb{R}$ such that the ground distance $D(x, y) = g(x) + g(y)$, then $EMI_p = 0$ and is trivially PD on $\mathcal{F}(X)$ for any choice of p.*

Proof. Let $f(a, b)$ be the maximum-cost maximum flow between sets A and B with respect to K_p defined using Lemma 2.4 with $x_0 = p$. Note that in this case, $K_p(x, y) = 0$. As a result, $EMI_p(A, B) = 0$, which is trivially PD. □

If $g(p) \geq 0$ and we opt to use EMI′ by discarding $D(p, p)$ in (5.25), then

$$EMI'_p(A, B) = 2g(p) \min \{\mu(A), \mu(B)\}, \tag{5.31}$$

which is simply a scaled version of the min-kernel, which is known to be PD. We now explore more complex scenarios.

5.3.2 Transportation on the Real Line

Consider the space of probability distributions on the real line $\mathbb{R}$. Let $D : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_0^+$ be a convex, non-negative symmetric function that takes the form $D(a, b) = h(a - b)$, where $h : \mathbb{R} \to \mathbb{R}_0^+$. If D is CND, then one can show that EMD equipped with D is CND as well. A well known result [108] states that EMD between two probability distributions $u, v \in \mathcal{P}(\mathbb{R})$ with a ground distance such as D can be written

$$EMD(u, v) = \int_0^1 D(U^{-1}(s), V^{-1}(s))\mathrm{d}s, \tag{5.32}$$

where U^{-1} and V^{-1} are the inverse cumulative distribution functions of u and v. In essence, the i-th point in ascending order of one distribution maps to the i-th point of the other. Since EMD in this form is clearly just the summation of CND functions, then EMD must also be CND.

5.3.3 Transportation on the Circle

Transportation on the circle is similar to transportation on the real line. In fact, one simply has to find an optimal point at which to cut the circle prior to treating it like the real line. In this case, the geodesic distance (i.e. length of arc or angle) is used to compare points. If the points x, y are linearly indexed on S^1, the circle with radius 1, then

$$D(x,y) = \min\{|x-y|, 2\pi - |x-y|\} \tag{5.33}$$

or equivalently

$$D(x,y) = \arccos\left(\begin{bmatrix}\cos(x) & \sin(x)\end{bmatrix}\begin{bmatrix}\cos(y) & \sin(y)\end{bmatrix}^{\intercal}\right), \tag{5.34}$$

which is provably CND by an infinite series expansion [60]. With the given ground distance and probability distributions $u, v \in \mathcal{P}(S^1)$, it can be shown that

$$EMD(u,v) = \|U - V - \alpha\|_1 = \int_0^{2\pi} |U(s) - V(s) - \alpha| \mathrm{d}s, \tag{5.35}$$

where U and V are cumulative distribution functions and α is the weighted median of $U - V$ [29, 108]. Surprisingly, one can empirically show that for arbitrary u and v, EMD is not CND on the circle despite its similarity to the line.

The reason that EMD is not CND on the circle is due to the use of the median in (5.35). If we approximate the median with the mean (guaranteed by Jensen's inequality

to be within 1 standard deviation [90]), then we obtain a CND approximation of EMD. Note that substituting the mean in (5.35) yields

$$EMD(u,v) \approx \int_0^{2\pi} \left| U(s) - \int_0^{2\pi} U(t)\mathrm{d}t - \left(V(s) - \int_0^{2\pi} V(t)\mathrm{d}t \right) \right| \mathrm{d}s. \tag{5.36}$$

which is a sum of CND kernels. If the median can be expressed by a function h as $\alpha = h(u) - h(v)$ (perhaps only for certain families of distributions), then EMD is CND.

5.3.4 Transportation on the L_2 Hypersphere

Consider the class of ground distances of the form $\beta - K$, where β is a positive constant and K is PD. This class of ground distances coincides with those implied by CND $\widehat{\mathrm{EMD}}_\alpha$ since we may note that Pele and Werman's $\widehat{\mathrm{EMD}}_\alpha$ is a special case formulation of (5.26) that uses $D(a,p) = \alpha \max\{D(a,b)\}$ for every $a, b \in X$. If a point p can be found or created such that $D(a,p) = \beta$ for each $a \in X \setminus \{p\}$ and D is CND, then by Lemma 2.4 we can conclude that D is of the form $\beta - K$ (in this case, $\beta = 2\alpha \max D(a,b) - D(p,p)$). A characterization of kernels of this form is given by Berg *et al.* [7]. However, if we add the condition that D satisfies identity of indiscernibles, then a geometric interpretation of D is readily forthcoming. In particular, the image $\phi(X)$ from K's feature mapping lies on the hypersphere of radius $\sqrt{\beta}$ in a Hilbert space centered on the point $\phi(p) = \mathbf{0}$. This follows from the fact that $K(a,a) = \beta$ as a consequence of $D(a,a) = 0$. In other words, this subclass is comprised of normalized kernels and embeds into squared L_2 on the hypersphere.

Ground distances of this form have already appeared in the literature. Rabin *et al.* [108] considered geodesic distances on the circle and used them for color image

retrieval and color transfer between images. Zhang *et al.* [156] used a Euclidean ground distance in a high-dimensional space to compare SIFT descriptors for object and texture recognition in images. However, they normalized the vectors comprising each set's support, effectively restricting their computations to distance between points on the hypersphere. This study provided empirical evidence that EMD tends to be CND for this restricted case since no violations were found.

However, the result of Naor and Schechtman [100] states that EMD is indefinite on the $\{0,1\}^2 \subset \mathbb{R}^2$ grid with a Euclidean ground distance. We can thus conclude that EMD is actually not CND for ground distances of the form $\beta - K$ in general since one can find a subspace of the hypersphere isometric to $\{0,1\}^2$. Consequently, any ground distance must necessarily not include subspaces isometric to $\{0,1\}^2$ if there is any hope for EMD to be CND. We do have one example, though, of a ground distance of this form—the discrete metric—where EMD is CND, and we hypothesize that ground distances close to discrete in form are also sufficient. More formally, we hypothesize that there exists $\epsilon > 0$ such that if $K(x,x) = 1$ for all $x \in X$ and $K(x,y) < \epsilon$ for all $x \neq y$, then EMI equipped with K is PD. We will now illustrate this notion with a method that transforms a ground distance into a nearly discrete form in order to yield CND EMD.

Under the following assumptions about the distribution of the sets under consideration for use with EMD, we may use Proposition 5.3 to show that there exists a transformed ground distance of the form $\beta - K$ that yields CND EMD. The assumptions that we make are that the sets are discrete, the collection of sets is finite, and that each pair of sets is disjoint. Note that these assumptions form sufficient but

not necessary conditions for the strategy that follows. We also assume that K strictly satisfies (5.10) for different x, y but is not necessarily PD. One may then infer that there exists a number n_0 for which $K_T^{(n)}$ is PD for $n \geq n_0$.

Let $X_1, X_2, \ldots, X_n \in \mathcal{F}(X)$ be subsets of X discretely supported with support cardinalities s_i, $i \in [1, n]$. Let K_i^j be the $s_i \times s_j$ kernel matrix computed between the elements of X_i and X_j, and let

$$F_i^j = \arg\max_f \operatorname{vec}\left(K_i^j\right)^\intercal \operatorname{vec}(f) \tag{5.37}$$

be the $s_i \times s_j$ maximum-cost maximum-flow matrix computed between X_i and X_j, where $\operatorname{vec}(M)$ is the vectorization of the matrix M made by concatenating columns. Note that

$$EMI(X_i, X_j) = \operatorname{vec}\left(K_i^j\right)^\intercal \operatorname{vec}\left(F_i^j\right). \tag{5.38}$$

Let H_i^j be the Schur product of F_i^j and K_i^j. Note that H_i^i is diagonal for each i as a consequence of (5.10). Additionally, $H_j^i = H_i^{j\intercal}$, and

$$EMI(X_i, X_j) = \sum_{h=1}^{s_i} \sum_{k=1}^{s_j} H_{i\,h,k}^j. \tag{5.39}$$

By an application of the derived subsets kernel [123], we may deduce that EMI is PD if the kernel matrix G_H, where the (i,j)-th block $G_H(i,j) = \left[H_i^j\right]$, is PD, i.e. if $H : X^* \times X^* \to \mathbb{R}$ is a PD kernel, where $X^* = \bigcup_{i=1}^n X_i$.

There are several ways one may proceed to obtain PD EMI. One may transform K and either keep or recompute the flow. One may also transform H or EMI itself. Since the sets are disjoint and K satisfies the conditions of Proposition 5.3, then H and EMI satisfy the same conditions. By Proposition 5.3, repeated transformation of H or

EMI will eventually become PD. Transforming only K is slightly more complicated to analyze, but one may note by similar reasoning used in the proof of Proposition 5.3 that G_H must eventually become PD since the off-diagonals converge to zero and the diagonal will be constant after the first transformation. Note that we do not endorse this transformation scheme for use with any ground distance, and we hypothesize that it is most appropriate for ground distances that are already normalized, i.e. of the form $\beta - K$. We do not test this idea in our experiments since the kernels that would have been candidates for the approach turned out to be PD. An exploration of this idea is beyond the scope of this dissertation.

5.4 Experiments

In this section we describe experiments with classification using SVMs (see Section 2.6) designed to demonstrate the utility of $\widehat{\text{EMD}}$ as well as the utility of the definite preserving transformation of Section 5.2 with respect to EMD. To our knowledge, $\widehat{\text{EMD}}$ has not been applied in a kernel setting and we therefore perform the first such experiments. In particular, we evaluate the effect of choosing some different values of p (the sink to which excess mass is transported in our generalization of Pele and Werman's $\widehat{\text{EMD}}$). For each of the EMD variants, we make use of Theorem 2.3 to construct generalized RBF kernels of the form $\exp(-uD_{EMD})$, where D_{EMD} is an EMD-based distance between sets. In order to avoid the overhead of tuning u via cross-validation, we assign u to be the inverse of the average value of D_{EMD} on the training set as suggested by Zhang *et al.* [156].

We also show that when using unnormalized sets, especially when the magnitude of the mass has semantic significance relevant to classification, that $\widehat{\text{EMD}}$ is superior to EMD. Since we are dealing with indefinite kernels, we evaluate the results in the context of two techniques designed to address the nonconvex optimization encountered in training SVMs with such kernels. The techniques mentioned are eigenvalue shifting of the kernel matrix and the Krein support vector machine (KSVM) recently proposed by [87]. Both methods were chosen for their relative simplicity of implementation as well as the fact that test points (or associated kernel evaluations) do not need to be modified. Where appropriate, these methods are balanced against SVMs trained directly with the indefinite kernels.

Shift is a heuristic that involves shifting the eigenvalues of the kernel matrix to be non-negative (e.g. by adding $s\mathbf{I}$ to the kernel matrix, where s is the amount to shift each eigenvalue and $\mathbf{I}$ is the identity matrix). Shifting causes the SVM training problem to become convex, assuring a globally optimal solution. Wu *et al.* [154] show that shifting adds a regularization term that penalizes the norm of the support vector coefficients. Thus, simply choosing a very large s that guarantees PD-ness is not necessarily beneficial as it may constrain possible solutions. The smallest possible s (i.e. the magnitude of the least negative eigenvalue) is generally a good default choice. Approximations for s that assure PD-ness without requiring an eigendecomposition of the kernel matrix can be used. We did not make use of these approximations, however.

On the other hand, KSVM is formulated in the theory of *Krein spaces* (generalizations of Hilbert spaces with indefinite inner products) and may be considered a state of the art indefinite kernel technique. Our results certainly reflect its ability to

compensate for deficiencies in an indefinite kernel. However, KSVM is computationally expensive, requiring an eigendecomposition of the entire precomputed kernel matrix used for training. Therefore, Loosli *et al.* [87] also proposed KSVM-L, a more practical alternative that uses partial decompositions.

For completeness, we briefly describe the KSVM algorithm. Given a kernel matrix G_K and label vector $\mathbf{y}$ containing ± 1 for each respective positive or negative instance, one must compute an eigendecomposition of YG_KY, where $Y = \text{diag}(\mathbf{y})$ is an otherwise zero matrix with $\mathbf{y}$ on the diagonal. If U and D are the resulting eigenvector and eigenvalue matrices satisfying $UDU^\intercal = YG_KY$, then one trains the SVM using a standard solver with the PD kernel matrix $\overline{G}_K = USDU^\intercal$, where $S = \text{sign}(D)$ and $\text{sign}(D)$ is the element-wise sign function of the matrix D that yields 1 for each positive element, -1 for each negative, and 0 otherwise. Finally, one transforms the resulting support vector coefficients $\overline{\boldsymbol{\alpha}}$ (not to be confused with α in $\widehat{\text{EMD}}_\alpha$) to obtain support vector coefficients $\boldsymbol{\alpha} = USU^\intercal\overline{\boldsymbol{\alpha}}$ in the original indefinite space. The solution is not sparse. One may note that KSVM is equivalent to flipping each negative eigenvalue of the kernel matrix to be positive prior to transforming the result. We also note that a one-versus-all scheme for multiclass SVMs can have a distinct computational advantage over one-versus-one schemes since if $\mathbf{y}_i$ is the label vector treating the i-th class as positive and the remainder negative, $Y_i = \text{diag}(\mathbf{y}_i)$, and V contains the eigenvectors of G_k, then $U_i = Y_iV$ provides the eigenvectors of $Y_iG_kY_i$. Consequently, only one eigendecomposition is required regardless of the number of classes. We take advantage of this fact in our experiments; i.e. all results are computed using one-versus-all binary SVMs.

5.4.1 Datasets

Each considered kernel—EMD with Rubner's scaling, $\widehat{\text{EMD}}$, and its biotope transformation $\widehat{\text{EMD}}_{T,p}$ (hereafter referred to as earth mover's Jaccard distance (EMJD))—was evaluated on four datasets: the texture database KTH-TIPS [55], the object category database Caltech-101 [37], the handwritten character database MNIST [74], and the motion capture hand posture dataset described in Section B.3. The Euclidean distance served as the ground distance for each dataset except for Caltech-101, for which it was squared.

The KTH-TIPS database consists of 10 texture classes under varying scale, pose, and illumination with 81 instances per class. Images are standardized by resizing to a horizontal resolution of 480 pixels while preserving aspect ratio. We adopted much of the experimental design of Zhang *et al.* [156], constructing image *signatures* from SIFT descriptors. The SIFT descriptor [88] computes an N-bin histogram of image gradient orientations for an $M \times M$ grid of samples in the region of interest, resulting in an $M \times M \times N$ dimensional vector. We used the implementation of the SIFT descriptor provided by Vedaldi and Fulkerson [135] with $M = 4$ and $N = 8$. The resulting 128-dimensional vectors were scaled to have a Euclidean norm of 1 to reduce the influence of illumination changes. The descriptors were then clustered using a k-means algorithm (with $k = 40$). Each mean was weighted with the percentage of descriptors assigned to it, and the means paired with these weights constituted the so-called signature for a single image.

A very similar feature extraction procedure was conducted for the Caltech-101 dataset composed of color images of 101 categories (e.g. face, car, etc.) with varied

presentation. Instead of SIFT descriptors, the PHOW descriptor implemented by Vedaldi and Fulkerson [135] was used to represent images. At a high level, the PHOW descriptor is a dense SIFT extractor (the regions of interest are densely sampled in a grid) that can operate on multiple color channels instead of just grayscale. However, we simply used grayscale. Sets were normalized for both KTH-TIPS and Caltech-101.

The MNIST dataset comprises 28×28 grayscale images of handwritten digits ranging from zero to nine. Noble's version [102] of the Harris corner detector [54] was used to identify keypoints in the image (implemention again provided by Vedaldi and Fulkerson [135]). Images were smoothed with a Gaussian window with a variance of 1 prior to application of the Harris response function, which also used a Gaussian window with a variance of one. Local maxima in the response were interpreted as corners. The set of coordinates (scaled to lie between zero and one) of these detected corners then constitute the features of the image with the expected number of corners and their locations depending upon the digit. The number of detected corners typically ranged from 5 to 15.

5.4.2 Design of Experiments

Each experiment on each dataset involves the choice of a different sink p to which excess mass is sent. If the ground distance is thresholded and p lies beyond the threshold for every point in the training and test sets, then one can use a flat rate equal to the threshold as the cost of transporting excess mass. Therefore, we simply use the threshold to identify different experiments. The thresholds that were used are reported in the next subsection's tables. One will note that the bottom row of

each table has no threshold (denoted by a dash), and in this case p was generally chosen to be the origin with the exception of MNIST, where it was chosen to be the center of an image, $[0.5, 0.5]^T$. In the case of KTH-TIPS and Caltech-101, choosing the origin is not much different than choosing a threshold of 1 since every point lies on the surface of a unit hypersphere. The advantage of flat thresholds lies in their simplicity of implementation (the precise value of the optimal flow is irrelevant) as well as the ability to use faster algorithms [106].

The following data selection schemes were repeated for each experiment (threshold) with the exception that the selection of data for experiments with no threshold matched that of the highest threshold in order to enable a direct comparison. For KTH-TIPS (and Caltech-101), 40 (15) images from each class were randomly drawn to be the training set with an equivalently drawn disjoint test set. This random selection was repeated five times in order to obtain five training/test set pairs, the results of which were averaged. For MNIST, 200 examples from each class were randomly chosen and five-fold cross validation was computed for each experiment. For the posture recognition dataset, special consideration was required due to the fact that there is signficant correlation and even near duplication for samples corresponding to a single user. Therefore, a leave-one-user-out approach was employed where each of the 12 users served in turn as the test set. As a result, experiments measured the generalization of the classifier to new users. The size of the dataset was reduced and classes balanced by randomly selecting 75 examples per class per user.

5.4.3 Results and Discussion

For normalized sets contained in KTH-TIPS and Caltech-101 (Tables 5.1 and 5.2), there is no significant difference between the three kernels. In fact, EMD and $\widehat{\text{EMD}}$ are the exact same for any two normalized sets since the difference in mass is zero.

Table 5.1: Accuracies for texture recognition on normalized sets with KTH-TIPS. All kernels were found to be positive definite. Since sets are normalized, EMD is equal to $\widehat{\text{EMD}}$.

Threshold	$EMD/\widehat{EMD}$	$EMJD$
0.5	71.45 ±6.19	70.95 ±6.16
1	74.75 ±1.00	74.55 ±0.65
$\sqrt{2}$	70.70 ±7.96	70.85 ±8.06
-	70.70 ±7.96	70.80 ±8.07

Table 5.2: Accuracies for object category classification on normalized sets with Caltech-101. All kernels were found to be positive definite. Since sets are normalized, EMD is equal to $\widehat{\text{EMD}}$.

Threshold	$EMD/\widehat{EMD}$	$EMJD$
0.5	49.97 ±0.90	49.65 ±0.80
1	48.77 ±0.75	48.84 ±0.81
2	48.57 ±1.39	48.71 ±1.19
-	48.57 ±1.39	48.55 ±1.26

However, for unnormalized sets (Tables 5.3 and 5.4), $\widehat{\text{EMD}}$ and EMJD are noticeably better than EMD despite the indefinite kernel techniques. KSVM actually improved EMD's accuracy far beyond what was expected, nearly matching $\widehat{\text{EMD}}$'s performance (and surpassing it on the highest thresholds for MNIST). However, this state of the art indefinite kernel technique was still unable to bridge the difference in

all cases, and the results should be balanced by the more computationally practical Shift, which was completely unable to compensate for EMD's indefiniteness.

Table 5.3: Accuracies for handwritten character recognition on unnormalized sets with the MNIST derived data.

	Indefinite			Shift			KSVM		
Threshold	*EMD*	$\widehat{EMD}$	*EMJD*	*EMD*	$\widehat{EMD}$	*EMJD*	*EMD*	$\widehat{EMD}$	*EMJD*
0.25	34.30 ±5.45	67.80 ±1.36	78.20 ±2.22	32.25 ±2.36	79.90 ±1.76	80.65 ±1.97	75.30 ±1.78	78.05 ±1.56	79.50 ±1.99
0.5	28.10 ±4.03	60.30 ±2.77	73.90 ±3.22	28.70 ±1.22	78.80 ±1.90	78.85 ±1.74	75.30 ±1.34	76.00 ±0.98	76.90 ±0.84
1	32.70 ±3.43	58.65 ±0.38	67.10 ±1.11	29.10 ±2.37	77.70 ±2.19	77.45 ±1.93	72.15 ±1.81	73.65 ±1.62	74.85 ±1.61
$\sqrt{2}$	32.75 ±2.71	59.90 ±0.72	65.45 ±1.81	27.85 ±1.46	77.70 ±2.03	77.75 ±1.85	76.05 ±2.00	74.70 ±1.23	74.65 ±1.72
-	32.75 ±2.71	49.60 ±1.56	52.00 ±2.05	27.85 ±1.46	75.30 ±1.67	76.85 ±1.93	76.05 ±2.00	73.85 ±1.11	74.35 ±1.01

Table 5.4: Accuracies for posture recognition on unnormalized sets.

	Indefinite			Shift			KSVM		
Threshold	*EMD*	$\widehat{EMD}$	*EMJD*	*EMD*	$\widehat{EMD}$	*EMJD*	*EMD*	$\widehat{EMD}$	*EMJD*
25	37.20 ±16.56	80.87 ±11.11	80.53 ±10.53	53.31 ±15.42	80.64 ±11.15	80.53 ±10.53	73.00 ±13.76	80.67 ±10.99	80.53 ±10.53
50	38.96 ±18.65	90.91 ±12.03	90.96 ±12.00	42.20 ±17.87	91.13 ±11.76	90.96 ±12.00	87.98 ±13.36	90.96 ±12.06	90.96 ±12.00
100	32.80 ±20.22	95.02 ±6.37	94.44 ±6.63	34.07 ±16.94	95.00 ±6.40	94.44 ±6.63	92.93 ±10.06	95.00 ±6.12	94.44 ±6.63
150	28.96 ±22.31	95.47 ±6.40	95.02 ±6.60	30.69 ±16.30	95.00 ±6.77	95.02 ±6.60	91.82 ±11.92	95.42 ±6.54	95.02 ±6.60
200	29.73 ±18.65	95.09 ±6.73	94.31 ±7.17	30.89 ±16.82	94.44 ±7.20	94.24 ±7.22	92.22 ±8.43	94.60 ±7.22	94.27 ±7.23
-	29.73 ±18.65	95.20 ±5.97	95.24 ±6.07	30.89 ±16.82	95.27 ±5.69	95.09 ±6.15	92.22 ±8.43	95.60 ±5.77	95.58 ±5.92

Our experiments on KTH-TIPS and Caltech-101 confirmed the report of Zhang *et al.* [156] that the RBF kernel for EMD is PD with this data. However, computation of EMI revealed an indefinite kernel matrix, which indicates that only a subset of $u < 0$ from Theorem 2.3 is satisfied and that Zhang *et al.*'s selection strategy for u just happens to fall within this subset. The same behavior was observed for EMJD on these two datasets. The ground distance's support for posture recognition and MNIST,

on the other hand, does not consist of normalized vectors. For posture recognition, we noticed that EMJD was more likely to yield a PD RBF using the aforementioned selection strategy. For example, observe that the Shift and KSVM results are the same as the indefinite results for certain thresholds, with lower threshelds apparently increasing the likelihood of generating a PD kernel. Exploration on normalized sets (not shown) with both MNIST and posture recognition made this effect more pronounced.

Of special note is the fact that $\widehat{\text{EMD}}$ and EMJD yield significant improvements in accuracy even without applying any indefinite kernel technique. On the posture recognition dataset in particular, the effective results are nearly indistinguishable from Shift and KSVM. For the MNIST dataset, indefinite EMJD consistently outperformed the other two kernels and rivaled Shift and KSVM at the lowest threshold. These results indicate that $\widehat{\text{EMD}}$, EMJD, and perhaps the definite preserving transformation in general have value on their own without additional indefinite kernel methods.

In general, one can observe that the threshold has a significant effect on the quality of the classifier. The highest threshold for each dataset, which matches or exceeds the diameter of the ground distance's support, did not yield the best observed results for any dataset. Lower thresholds tended to yield better results (up to a point). As the threshold lowers, EMD becomes a closer approximation to the set symmetric difference and thus more similar to the intersection kernel. As stated in Section 5.3.1, thresholding can be interpreted as a means to induce concavity in the ground distance and make it more similar to the discrete metric. This explains why the accuracy drops off after a certain minimum threshold (as it becomes too similar to classical

intersection to associate slightly different elements) as well as its tendency to improve prior to the drop off.

Our work raised some open questions. We do not know whether thresholding a distance preserves CND properties as it does metric properties [106]. Our experiments did not contradict the hypothesis. The choice of the optimal threshold is also open. One could always tune the threshold via cross-validation, but we suspect that a decent approximation to the optimal threshold would be to use the average or median distance between all points. Using no threshold or choosing p to be closer than the threshold is also an option to consider as the posture recognition experiments demonstrate.

One unexpected result was KSVM's poor performance on MNIST relative to Shift for $\widehat{\text{EMD}}$ and EMJD. This result is at odds with the expectation that KSVM should be at least as good as other indefinite kernel techniques, which is fairly well justified in its introductory article [87]. We noted that the eigenspectrum of an MNIST kernel matrix was much less concentrated than those for the other datasets. Whereas performing a partial decomposition with the 50 highest magnitude eigenvalues was typically sufficient to retain approximately 95% of the spectrum's total magnitude on the other datasets, as many as 1200 eigenvalues were required to achieve the same preservation of the spectrum on MNIST. In fact, the results reported in Table 5.3 are from a complete eigendecomposition. Additional research may be required to determine if this is a peculiarity unique to our treatment of MNIST or some weakness of KSVM.

As a final addendum on EMD's definiteness, we expect there to be many other instances of PD kernels based either directly or indirectly of EMD. For example,

recalling Section 5.3.2, note that Kolouri *et al.* [71] use $D(a,b) = (a-b)^2$ to show that the *sliced Wasserstein kernel*, which is calculated between distributions in $\mathbb{R}^d$ via one-dimensional projections, is PD. Cuturi [24] on the other hand proposed a regularized version of EMD via an additional entropic term that yields the PD *independence kernel* when the entropic term's effect is maximized. One may also consider the following special case to reveal similarities to another min-like kernel, the Brownian bridge product kernel [128],

$$K_B(x,y) = \min\{x,y\} - xy. \tag{5.40}$$

Suppose the ground distance D is supported by two points $p_1, p_2 \in \mathbb{R}$, and without loss of generality assume $p_1 = -p_2 = 1$. Assuming $u, v \in \mathcal{P}(\{p_1,p_2\})$, let $\chi_u(p_1) = x$ and $\chi_v(p_1) = y$ so that $\chi_u(p_2) = 1-x$ and $\chi_v(p_2) = 1-y$. Then for $i \in \{1,2\}$, the optimal flow f^* satisfies

$$f^*(p_i,p_i) = \min\{\chi_u(p_i), \chi_v(p_i)\}, \tag{5.41}$$

$$f^*(p_1,p_2) + f^*(p_2,p_1) = 1 - f^*(p_1,p_1) - f^*(p_2,p_2). \tag{5.42}$$

Choosing the sink $p = 0$ in (5.28), we can determine that

$$\begin{aligned} EMI_0(u,v) = \sum_{i,j=1}^{2} f^*(p_i,p_j)p_ip_j &= 2(\min\{x,y\} + \min\{1-x,1-y\}) - 1 \\ &= K_B(x,y) + 2K_B(1-x,1-y) \\ &\quad + \min\{x,y\} - x(1-y) \\ &\quad - y(1-x) + (1-x)(1-y), \end{aligned} \tag{5.43}$$

which is clearly the sum of two Brownian bridge product kernels and a similarly structured term.

CHAPTER 6

NEURAL NETWORK ARCHITECTURES FOR GESTURE RECOGNITION

In this chapter we propose ANN architectures for posture and gesture recognition and evaluate them on the posture and gesture recognition datasets described in Appendix B. As a prerequisite, we describe the steps taken to prepare the data for processing by neural networks. In order to establish context, consider the diagram in Figure 6.1.

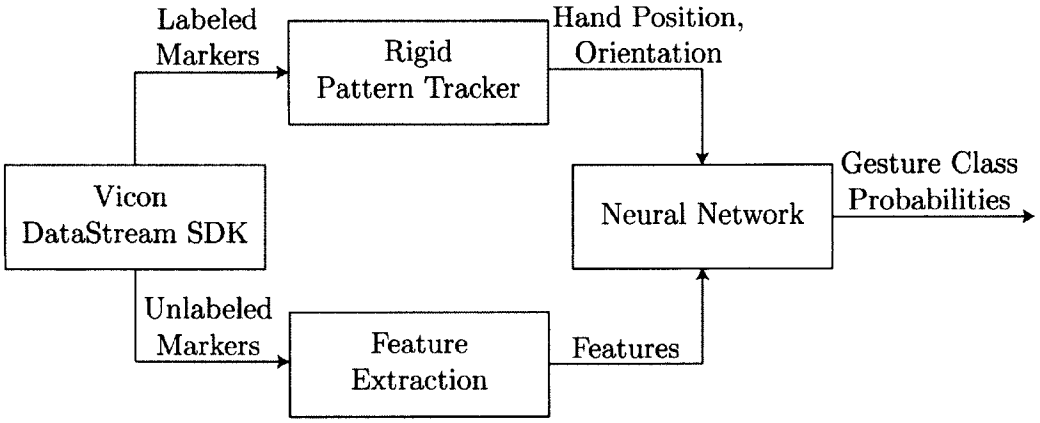


Figure 6.1: A high-level diagram of the overall architecture and flow of data from the lowest accessible level (Vicon Datastream SDK) to the desired result (probabilities for gesture classification).

Some effort should be made to ensure that the data provided to the neural networks is consistent and relatively error free. Thus, a preprocessing layer is interjected prior to the neural network that filters or otherwise transforms information provided

by the Vicon DataStream SDK. This figure shows that we separate the processing of unlabeled markers and the four labeled markers that constitute the pattern on the back of the hand (see Figure 1.2). The pattern's markers, which may be noisy, incorrect, or partially occluded, are filtered to produce more reliable estimates of the hand's position and orientation than what the Vicon DataStream SDK provides. As in previous chapters, we use the pattern to establish a local coordinate system for the hand. We also use the pattern to estimate the position and orientation of the hand, which are expected to be important features for gesture recognition. The extraction of features from unlabeled markers is not quite as straightforward and is intentionally vague in the figure. However, we avail ourselves a resource denied in prior chapters by exploiting the context of temporally adjacent frames. We consider two general approaches to extracting features from unlabeled markers. We either use the positions directly by extracting marker identities, or we transform the unlabeled marker sets with certain neural network architectures. After features are extracted from both labeled and unlabeled markers, the application of an RNN is rather straightforward for the considered data.

The chapter is organized as follows. In Section 6.1, we describe an EKF (Section 2.5) for tracking an arbitrary rigid pattern. We follow this with a discussion on feature extraction from unlabeled markers in Section 6.2 before describing experiments to evaluate and compare the proposed architectures in Section 6.3 and their results. For a review of neural networks, please refer to Section 2.7. A review of quaternions (Section 2.1.2) and EKFs (Section 2.5.3) is also advised for the next section.

6.1 Tracking a Rigid Pattern

Recall the definition of a rigid pattern given in Section 1.1. In this subsection, we define a filter to estimate the position and orientation of an arbitrary pattern. Let us formally define a pattern to be a set of m functions $M_i : \mathbb{R} \to \mathbb{R}^3$, $i \in [1, m]$, representing marker positions whose pairwise Euclidean distances are constant over time, i.e.

$$\|M_i(t) - M_j(t)\| = e_{ij}, \tag{6.1}$$

where each e_{ij} is constant and t is time. In reality, some flexibility in the pattern is expected but assumed to be negligible. We use a rigid pattern composed of $m = 4$ markers to determine the location and orientation of the hand. Sometimes the pattern becomes partially or completely occluded, corrupted by noise, or misrepresented by a completely incorrect measurement reported by Vicon. We therefore need a filter to fill in these missing values as well as smooth the measurements. We choose to use an EKF (Section 2.5.3). For the remainder of the section, assume that we are sampling marker positions at a rate of τ^{-1} Hz in order to obtain measurement vectors

$$\mathbf{y}_k = \begin{bmatrix} M_1(k\tau)^\intercal I_1(k) & M_2(k\tau)^\intercal I_2(k) & \dots & M_m(k\tau)^\intercal I_m(k) \end{bmatrix}^\intercal, \tag{6.2}$$

where $I_i(k) = \mathbf{I}_3$ if the i-th pattern marker is visible at time $k\tau$ and $\mathbf{0}$ otherwise.

The state used to represent the pattern should be comprised of a minimal set of variables that represent the entire pattern's dynamics (position, velocity, acceleration, etc.). Let us select a marker M_j and assume that the vector-valued functions $E_{ij}(t) = M_i(t) - M_j(t)$, $i \neq j$, are known for $t = 0$. Without loss of generality, assume $j = 1$, let $\mathbf{O}_k = M_1(k\tau)$ and $\bar{e}_k^{(i)} = (0, E_{i1}(k\tau))$, and note via (2.13) that for each $k \geq 0$ there

exists a unit quaternion $\overline{\lambda}_k$ such that

$$\overline{e}_k^{(i)} = \overline{\lambda}_k \overline{e}_0^{(i)} \overline{\lambda}_k^{\star}. \tag{6.3}$$

In fact, $\overline{\lambda}_0 = (1, \mathbf{0})$. We can therefore conclude that a second order Taylor approximation of the pattern's dynamics can be represented by the state

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{O}_k{}^{\mathsf{T}} & \dot{\mathbf{O}}_k{}^{\mathsf{T}} & \ddot{\mathbf{O}}_k{}^{\mathsf{T}} & \overline{\lambda}_k{}^{\mathsf{T}} & \boldsymbol{\omega}_k{}^{\mathsf{T}} & \boldsymbol{\alpha}_k{}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}, \tag{6.4}$$

where $\boldsymbol{\omega}_k$ and $\boldsymbol{\alpha}_k$ are the angular velocity and acceleration, respectively, each represented as a vector whose direction is the axis of rotation and whose magnitude is the angle in radians.

The associated transition function is nonlinear due to the rotation and is given by

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) = \begin{bmatrix} \begin{bmatrix} \mathbf{O}_k + \tau \dot{\mathbf{O}}_k + \frac{1}{2}\tau^2 \ddot{\mathbf{O}}_k \\ \dot{\mathbf{O}}_k + \tau \ddot{\mathbf{O}}_k \\ \ddot{\mathbf{O}}_k \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \overline{\gamma}_k \overline{\lambda}_k \\ \boldsymbol{\omega}_k + \tau \boldsymbol{\alpha}_k \\ \boldsymbol{\alpha}_k \end{bmatrix}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} + \mathbf{w}_k, \tag{6.5}$$

where $\overline{\gamma}_k = \text{quat}\,(\boldsymbol{\gamma}_k)$ and $\boldsymbol{\gamma}_k = \tau \boldsymbol{\omega}_k + \frac{1}{2}\tau^2 \boldsymbol{\alpha}_k$. The measurement function is also nonlinear and is given by

$$\mathbf{y}_k = h(\mathbf{x}_k, \mathbf{v}_k) = \begin{bmatrix} I_1(k)\mathbf{O}_k \\ I_2(k)\left[\mathbf{O}_k + \text{Im}\left(\overline{\lambda}_k \overline{e}_k^{(2)} \overline{\lambda}_k^{\star}\right)\right] \\ \vdots \\ I_m(k)\left[\mathbf{O}_k + \text{Im}\left(\overline{\lambda}_k \overline{e}_k^{(m)} \overline{\lambda}_k^{\star}\right)\right] \end{bmatrix} + \mathbf{v}_k. \tag{6.6}$$

The linearized transition and measurement matrices are thus given by

$$A_k = \begin{bmatrix} \mathbf{I}_3 & \tau\mathbf{I}_3 & \frac{1}{2}\tau^2\mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 & \tau\mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial\bar{\gamma}_k\bar{\lambda}_k}{\partial\bar{\lambda}_k} & \tau\frac{\partial\bar{\gamma}_k\bar{\lambda}_k}{\partial\bar{\gamma}_k}\frac{\partial\bar{\gamma}_k}{\partial\gamma_k} & \frac{1}{2}\tau^2\frac{\partial\bar{\gamma}_k\bar{\lambda}_k}{\partial\bar{\gamma}_k}\frac{\partial\bar{\gamma}_k}{\partial\gamma_k} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \tau\mathbf{I}_3 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_3 \end{bmatrix}, \tag{6.7}$$

$$H_k = \begin{bmatrix} I_1(k) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ I_2(k) & \mathbf{0} & \mathbf{0} & I_2(k)\frac{\partial\bar{\lambda}_k\bar{e}_k^{(2)}\bar{\lambda}_k^{\star}}{\partial\bar{\lambda}_k} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ I_m(k) & \mathbf{0} & \mathbf{0} & I_m(k)\frac{\partial\bar{\lambda}_k\bar{e}_k^{(m)}\bar{\lambda}_k^{\star}}{\partial\bar{\lambda}_k} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{6.8}$$

The partial derivatives can be determined via substitution into the equations given in Section 2.1.2. The measurement noise covariance R_k is simply the identity matrix scaled by a factor σ_r^2, whereas the process noise covariance is based on discretized white noise models [5] with the assumption of independence between translation and rotation, which yields

$$Q_k = \begin{bmatrix} Q_{11} & \mathbf{0} \\ \mathbf{0} & Q_{22}(k) \end{bmatrix}, \tag{6.9}$$

where

$$Q_{11} = \sigma_a^2 \begin{bmatrix} t^6/36 & t^5/12 & t^4/6 \\ t^5/12 & t^4/4 & t^3/2 \\ t^4/6 & t^3/2 & t^2 \end{bmatrix} \otimes \mathbf{I}_3, \tag{6.10}$$

$$Q_{22}(k) = \begin{bmatrix} \frac{t^3}{6}\frac{\partial \bar{\lambda}_k}{\partial \boldsymbol{\lambda}_k}\mathbf{I}_3 \\ \frac{t^2}{2}\mathbf{I}_3 \\ t\mathbf{I}_3 \end{bmatrix} \sigma_\alpha^2 \begin{bmatrix} \frac{t^3}{6}\frac{\partial \bar{\lambda}_k}{\partial \boldsymbol{\lambda}_k}\mathbf{I}_3 \\ \frac{t^2}{2}\mathbf{I}_3 \\ t\mathbf{I}_3 \end{bmatrix}^\mathsf{T}, \tag{6.11}$$

$\boldsymbol{\lambda}_k$ is a vector with $\|\boldsymbol{\lambda}_k\| \in [0, 2\pi)$ such that $\bar{\lambda}_k = \text{quat}(\boldsymbol{\lambda}_k)$,and σ_a^2 and σ_α^2 are the freely chosen respective magnitudes of the translational and rotational covariance.

The measurement noise is not truly normal, although its exact form is unknown. Depending on visibility, the quality of marker reconstruction, and the closed source algorithm Vicon uses to label markers that belong to a pattern, occasionally a completely incorrect measurement is reported. For example, four of the unlabeled markers on the fingers may be erroneously labeled as part of or as the entire pattern. Aside from the fact that these markers do not represent the pattern, Vicon also overrides their positions to force them into the pattern's shape. The second issue is unavoidable if one uses Vicon for labeling. The first issue is addressed by rejecting measurements that exceed some threshold distance from the EKF's prediction. For specific details of the threshold used in experiments, please refer to Section 6.3.2.

6.2 Feature Extraction from Unlabeled Marker Sets

If markers were labeled, then subsequent classification would be trivial. However, if a sufficient percentage of labels are not correct, then we run the risk of introducing errors that lower the quality of the classifier, which is especially true if we interpolate or extrapolate data based on these labels. Therefore, establishing a method that effectively uses the raw unlabeled marker positions could be superior. Based on these guidelines, we consider two approaches to extracting a consistent set of ordered features from unlabeled markers: extracting marker identities via tracking with Kalman filters and unsupervised feature extraction via neural networks. Note that this is not an exhaustive list of possible feature extractors, although we think that these are among the most promising for practical purposes.

6.2.1 Labeling Markers with Kalman Filters

Labeled markers allow one to consistently order features for a neural network or other classifier. We use 11 Kalman filters to track each unlabeled marker separately, although there is no *a priori* label for each filter. Each Kalman filter is similar to the rigid pattern tracker described in the previous section albeit with all orientation-related variables removed. Global coordinates are used for each filter in order to avoid propagation of transient or persistent errors in the pattern tracker. Measured unlabeled marker positions are assigned to predicted positions based on a Euclidean ground distance (see Section 2.4). Adaptive spherical *gates* centered on each prediction are used to reject infeasible assignments. The radius of each gate varies between a

minimum of 40 mm and a maximum of 1000 mm, increasing or decreasing by a factor of 1.2 whenever a measurement is not or is available.

The purpose of the Kalman filters is to consistently provide 11 markers per frame regardless of the amount reported by Vicon. We must then decide which *a posteriori* estimate corresponds to which part of the hand. We determine this correspondence on a per-frame basis; labels are not assigned to the filters. We use a feed-forward neural network with softmax output to approximate the probability $p(\lambda \mid \mathbf{x})$ that a given position $\mathbf{x}$ generates a label λ. Let Λ be the set of labels (e.g. thumb tip, knuckle, etc.) and $\hat{X}_k$ the set of *a posteriori* estimates produced at time k (expressed in local coordinates). Labels are assigned to estimates according to the bijection $\psi_k : \Lambda \to \hat{X}_k$ possessing the maximum likelihood, i.e.

$$\psi_k = \arg\max_{\psi} \sum_{\lambda \in \Lambda} \log p(\lambda \mid \psi(\lambda)). \tag{6.12}$$

The neural network is trained beforehand using data captured in a controlled setting and described in Section B.2. Obviously, the quality of the labeling depends on the quality of the labeled marker set.

The described procedure for finger tracking and marker labeling is quite similar to that proposed by Alexanderson *et al.* [2]. Whereas Alexanderson *et al.* use a more elaborate configuration of Kalman filters based on multiple potential assignments of labels, we use only a single assignment. We also use a neural network instead of GMMs to provide label probabilities.

6.2.2 Architectures for Unlabeled, Unordered Markers

In this section we propose several architectures for handling unlabeled marker sets and extracting a fixed-size output from them. Each architecture has different advantages and disadvantages and may also be used directly for posture recognition. We separate these architectures into two groups—fixed-size and variable-size—that indicate the expected format of the input.

Fixed-Size Architectures

Using a fixed-size architecture enables the use of MLPs and certain CNNs. Determination of an MLP is straightforward from Section 2.7.1 and thus does not bear repeating. When considering CNNs, each coordinate x, y, z is treated as a channel of a $1 \times n$ "image," where n depends on the dataset and is the maximum number of markers observed at one time in a sample frame. Frames with fewer than n markers are padded with zeros.

In general, fixed-size architectures are easier to implement and train in the sense of requiring fewer epochs and regularization (especially CNNs). However, they also possess serious disadvantages. Note that the second dimension on the input is not 11 because the dataset may contain extraneous markers, which highlights one of the primary issues with a fixed-size architecture. Namely, fixed-size architectures cannot handle extraneous markers in a principled manner. If in practice more markers appear in an example than the network can accept, then there is no way to classify the example without additional heuristics. Since the entire purpose of these architectures is to generally minimize the processing of raw data through external means, this

problem could be significant. A potentially troublesome related issue is the fact that the network implicitly uses the number of missing markers as a feature since this information is encoded in the number of padded zeros provided as input. The number of missing markers may not be a reliable feature as it depends on the quality of the camera calibration and physical configuration in addition to the hand's posture. Extraneous markers also inflate the number of markers visible.

Variable-Size Architectures

Variable-size architectures offer a principled manner to address both occluded and extraneous markers. These architectures are designed to exploit only the information explicitly contained within the markers that are visible.

The first and arguably simplest variable-size architecture we discuss is based on deep averaging networks [63] for text classification. A deep averaging network takes an arbitrary number h of word embeddings as input (i.e. words converted to vectors through some mapping), averages the embeddings, and gives the average to a feed-forward network. Whereas the embedding function is typically predetermined, we propose to dynamically learn the embedding by representing it as a MLP (see Figure 6.2). We refer to our version of this architecture as a convolutional deep averaging network (CDAN) since it is equivalent to consider convolving a 3×1 filter with a $3 \times h$ image followed by a pooling operation over the entire horizontal axis. Even though "averaging" is part of the name, we allow other pooling operations such as max. In fact, max-pooling may be preferable as it introduces an additional nonlinearity. The

primary advantage of a CDAN is that it is invariant to permutations of the input markers.

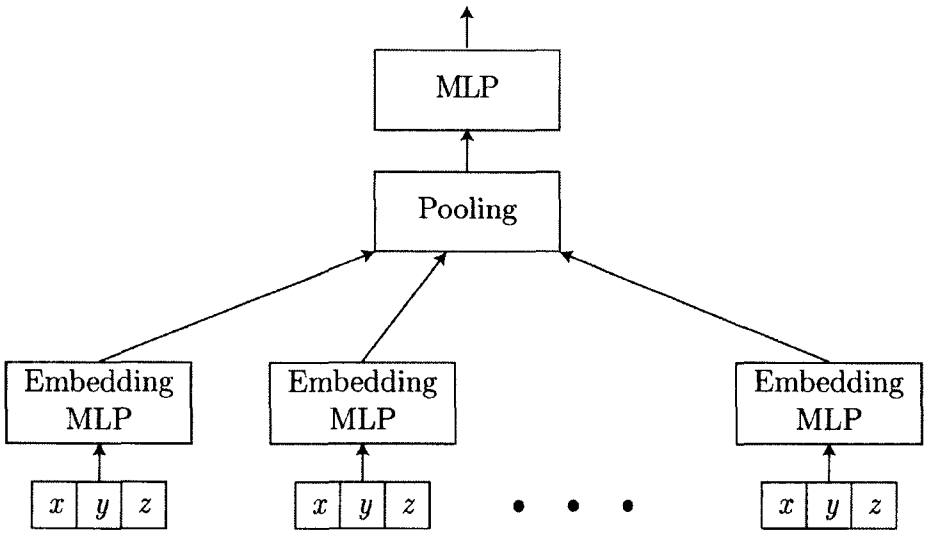


Figure 6.2: An illustration of a CDAN architecture for sets of 3D marker positions, arbitrarily ordered. A function represented by an MLP is convolved with the positions to produce a dynamically learned embedding in some potentially high-dimensional space.

However, the lack of connectivity before the pooling layer also constrains the ability of the network to learn. At the cost of giving up permutation invariance, we consider each marker set as a time series wherein each marker represents an observation at a certain time. RNNs provide a principled solution in this case. We consider two types of *bidirectional* RNN [121], where bidirectional denotes that we have two RNNs iterating over the input in opposite directions whose outputs are concatenated at each timestep. In the first type, we consider a GRU whose outputs are pooled over the entire time duration of the sequence prior to being given to an MLP. In essence, this architecture is equivalent to a CDAN if we allowed recurrent connections between the filters at each location (see Figure 6.3). For this reason, we refer to this architecture as

a recurrent deep averaging network (RDAN). The primary advantage of an RDAN over a traditional RNN is that the pooling allows earlier timesteps to override later ones. For example, an RDAN theoretically allows the detection of a relevant subsequence followed by meaningless noise that may otherwise lead the network astray. The second type of RNN is simply a GRU network, potentially multi-layer, that provides the output at its final time-step to an MLP. Of course, successfully training these networks to exploit their theoretical advantages is another matter.

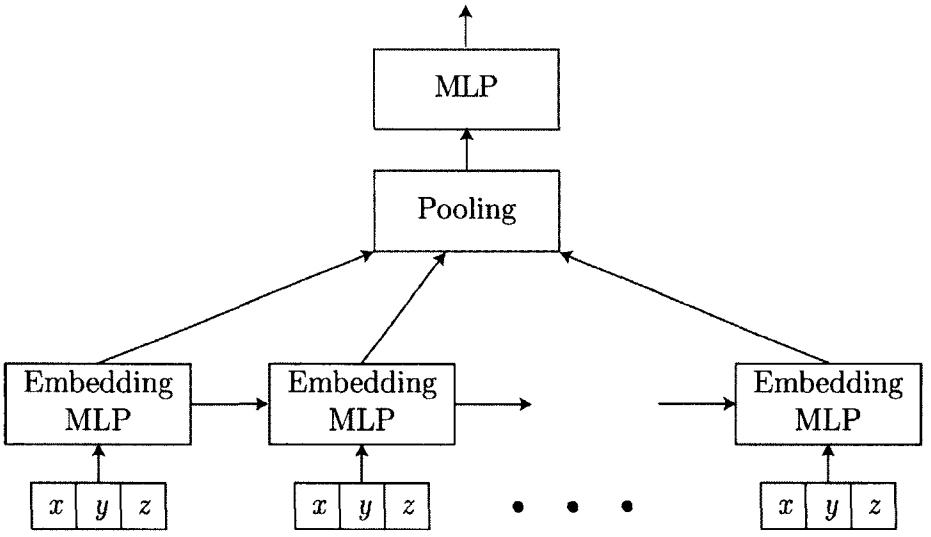


Figure 6.3: An illustration of a (unidirectional) RDAN architecture for sequences of 3D marker positions. Embeddings are no longer independent.

6.3 Evaluation

In this section we describe experiments used to evaluate neural networks for both posture and gesture recognition. In the case of gesture recognition, we compare feature extraction through labeling markers versus the features implicitly represented by the output of one of the architectures described in the previous section. We use

Keras [18] with the Theano [132] backend to implement and test our neural network architectures.

When using one of the proposed neural network architectures for unlabeled marker sets, the following additional preprocessing is performed to normalize the data. Unlabeled markers are lexicographically sorted according to perpendicular distance to three hyperplanes defined by linearly independent normal vectors (in local coordinates). In experiments, we simply use the basis vectors $\langle 1, 0, 0 \rangle$, $\langle 0, 1, 0 \rangle$, and $\langle 0, 0, 1 \rangle$. Consequently, in practice we effectively just sort by the x-coordinate from left to right since the probability of a tie is extremely low. Sorting minimizes the impact of the originally unordered nature of the markers, although it is not guaranteed to sort the markers in any consistent manner with respect to their latent labels. In addition, we center the markers of each frame on their mean as a form of normalization and optionally prepend the mean to the beginning of the sorted sequence. Prepending the mean ensures that the input is not invariant to translation of the original marker set, although it could potentially provide the networks a greater challenge during training. Theoretically, centering the markers is unnecessary as a sufficiently sized neural network should be able to learn the classification function without centering. Indeed, given enough resources, the proposed architectures are theoretically all equally capable. However, we found that centering yielded significant practical benefits.

6.3.1 Posture Recognition

For posture recognition, we considered the dataset described in Section B.3 and adopted the leave-one-user-out approach as taken in previous chapters with 75

samples per class per user. We tried to make the different architectures comparable by using similar amounts of regularization (see Section 2.7.4). In particular, we used weight decay with $\lambda = 0.001$, applied dropout to weights (not biases) at a 10% rate when indicated, and added Gaussian noise with a standard deviation of 20 mm to the input. All non-GRU layers used ReLU activations with the exception of a softmax layer as the output of each network, which is appropriate for classification. For the results reported in this section, the mean was not prepended to the centered marker positions.

The fixed-size MLP contained two hidden layers with 36 and 128 nodes, respectively. Dropout was applied to each. The fixed-size CNN used a 32 channel network-in-network [81] layer (i.e. 1×1 filters) followed by a 32 channel 1×3 filter. No pooling was applied. A dense hidden layer of 128 nodes followed convolution prior to the softmax output layer.

For each of the recurrent variable-size architectures, we used two bidirectional recurrent layers with 11 neurons each (and in each direction). The CDAN's embedding MLP possessed two layers with 11 neurons each. The MLPs to which these special layers feed their output are also two layers with 11 nodes in the first layer and five (the number of classes) in the second. Dropout was applied to all hidden layers, and max instead of average pooling was used in the CDAN and RDAN.

Results for each user left out as well as the overall average accuracy are listed in Table 6.1. The fixed size CNN achieves the best average performance, although the RDAN and RNN are not significantly worse. We attribute the slightly inferior performance of the recurrent architectures to the fact that they are in general harder

to train. The CDAN is significantly worse, however, which is due to the fact that it possesses less representational capacity for a given number of neurons as well as the fact that it is arguably harder to train. A CDAN is handicapped by the fact that each marker is considered in isolation prior to pooling. However, CDANs still possess potential as indicated by user 10. The MLP performs worse than the CNN, which is expected given that a CNN is an MLP with built-in regularization.

Table 6.1: Accuracies for leave-one-user-out classification with the posture dataset.

User	MLP	CNN	CDAN	RDAN	RNN
1	80.80	91.73	74.40	80.80	94.67
2	85.33	96.00	74.67	94.67	95.47
3	83.73	91.20	70.13	80.00	80.00
4	90.93	93.07	72.27	99.73	99.73
5	98.68	100.00	99.73	99.73	98.93
6	84.80	88.00	67.73	81.07	83.47
7	95.47	99.47	62.13	88.27	86.67
8	88.80	89.60	56.27	86.40	86.93
9	93.60	88.53	68.27	97.33	96.80
10	59.47	82.67	84.80	76.80	74.67
11	78.67	91.47	48.80	92.53	91.47
12	72.53	90.40	82.40	95.20	94.67
Average	84.40	91.84	71.80	89.38	90.29

We also experimented with training the variable-size architectures with fixed-size input (shorter sequences padded with zeros), which as stated previously implicitly encodes the number of missing markers as a feature. We found that there was no significant difference between the resulting accuracies, and so we can conclude that the majority of the information about a posture is contained in the markers that are present. Given the general disadvantages of a fixed-size architecture and the fact that there is no significant difference between the recurrent architectures and the fixed-size

CNN, we therefore recommend RDAN or RNN for classification tasks with unlabeled markers.

6.3.2 Gesture Recognition

For gesture recognition, we considered the dataset described in Section B.4 and adopted the same leave-one-user-out approach as that taken with the posture dataset. Every sample in the dataset was considered, and samples for each class underrepresented for a particular user were randomly sampled with replacement from within the user's data so that classes were equally represented. Sequences (i.e. samples) that were more than two standard deviations longer than the average sequence length were pruned as outliers prior to balancing classes. We compare the supervised and unsupervised feature extraction methods given in Section 6.2 to one another and find the unsupervised extraction with neural networks to yield superior accuracy.

Let us first describe the remaining characteristics of the rigid pattern tracker (Section 6.1), which affect the values of the features provided to the neural networks. Values of $\sigma_r^2 = 0.001$, $\sigma_a^2 = \sigma_\alpha^2 = 1000$ were chosen for the process and measurement noise magnitudes. The threshold used for rejecting measurements was based on differences between the local coordinate system that would be established using the algorithm in Figure B.1 for the *a priori* estimate and the measurement. The maximum distance between origins was set to be 40 mm, and the maximum angle between each respective axis was set to be 2.5 radians. If 30 consecutive frames were rejected, then the EKF was reinitialized with the latest measurement. In order to stop the EKF from diverging in scenarios with extended lapses of visibility or rejections, missing and

rejected measurements were substituted with the prior *a posteriori* estimate. This treatment stalled the estimated movement based upon the expectation that the user's hand was likely to reappear nearby.

Features extracted from the pattern tracker for each sample and timestep of a gesture included the global positions of each of the labeled markers, which implicitly encode both the position and orientation of the hand, as well as the global angular velocity and acceleration vectors. With the exception of the quaternion encoding the orientation, these features effectively are just the state of the EKF. We also extracted the position of each marker, angular velocity, and acceleration relative to the local coordinate system of the previous timestep. These relative features give a rotation and translation invariant representation of the hand with respect to itself. Since the dataset is relatively small and confined to a small space, we did not use the global features in these experiments.

Features extracted from the unlabeled marker set at each timestep were concatenated with the features from the pattern tracker and provided to an RNN composed of two 100-neuron GRU layers followed by a non-recurrent six node softmax layer replicated at each timestep. Weight decay with $\lambda = 0.001$ was applied to each layer. Only the RNN architecture for unlabeled feature extraction was considered as an alternative to labeling with Kalman filters, and it shared the same structure as the one used for postures except that the softmax function was stripped from the network and it possessed 200 neurons per recurrent layer and 100 neurons for both dense layers. Eleven neurons per layer was found to be insufficient given the much more diverse range of postures implicit to the gesture data. Weight decay applied to

the internal RNN was increased to $\lambda = 0.01$, but no Gaussian noise was added to the input. Dropout of 10% was applied to all hidden layers. The mean was prepended after centering the unlabeled markers. All markers, whether labeled via Kalman filters or not, were transformed each timestep to local coordinates using the pattern tracker's state. The gesture recognition results for each left-out user can be found in Table 6.2, where accuracy is determined by the classification of the final frame of each sequence.

Table 6.2: Accuracies for leave-one-user-out classification with the gesture dataset.

User	Label Extraction	RNN
1	61.51	68.25
2	75.80	90.87
3	67.06	78.97
4	76.98	79.76
5	69.04	72.62
6	60.71	88.00
7	81.35	98.81
8	83.33	79.76
9	57.14	59.13
10	90.08	73.02
11	72.62	74.60
12	69.84	87.70
Average	72.12	78.54

We can clearly see that the unlabeled feature extraction yielded a superior gesture classifier, which may seem counterintuitive at first. However, the results illustrate the problem with using supervised features that are not expertly crafted. Our supervised features (the labeled marker positions) depend on the quality of the labels, which are limited by both our method and our data. Ultimately, we conclude that our labeling algorithm introduced errors or inconsistencies in the marker positions and labels that were entirely avoided by the ANN-based extraction. We did not

employ the most sophisticated ensemble of Kalman filters possible for tracking the markers. More importantly, however, labels were based on data collected from only a single user. A need to collect this data for each of the original 12 users was not identified until long after the possibility had vanished. This limitation also highlights a weakness with a data-driven approach for extracting marker identities in that one must collect data for each user, which may not be practical or desirable.

CHAPTER 7

CONCLUSIONS

In this dissertation, we studied means to achieve gesture recognition in a motion capture environment. The Wasserstein distance and its derivatives (such as EMD and the assignment problem) pervaded nearly every aspect of the dissertation, underscoring the challenge induced by unlabeled motion capture markers. Various methods to address the fundamental uncertainty in marker identity were proposed. In Chapter 3, we explored different classifiers and feature transforms as effective ways to represent and characterize the data. We expanded these results in Chapters 4 and 5. In Chapter 4, we proposed the AFEM algorithm as a means to estimate the generative distribution of unlabeled, correlated point sets representing hand postures and showed that it was superior to a traditional EM algorithm. In Chapter 5, we proposed a generalization of EMD for kernels and explored scenarios under which EMD could be guaranteed to yield PD kernels. We also proved that a certain normalizing transformation was PD-preserving, described a family of transformed, normalized kernels, and implied that the biotope transform preserved CND-ness. Most importantly for our primary focus, we found that EMD-based SVMs yielded very accurate posture classifiers. Finally, in Chapter 6, we shifted focus to deep learning with neural networks

where we proposed an EKF for tracking rigid patterns along with several architectures for posture and gesture recognition with unlabeled markers.

7.1 Discussion

In Chapter 3, we demonstrated the performance of several classification algorithms on a variety of data transformations of small unlabeled point sets for 3D hand posture recognition. We found each data transformation to have inherent advantages and disadvantages. Aggregate features led to classification with reduced deviation but limited peak performance. Raw feature classifiers tended to the extremes in both overall error rate and deviation, likely due to their propensity for overfitting. On the other hand, the training objectives also significantly affected performance, as indicated by the results of the greedy GMMs. Grid transformed classifiers also possessed the potential for overfitting, but were capable of achieving maximum accuracy among the algorithms tested. In designing a classifier, one should strike a balance between global (e.g. aggregate) and local (e.g. individual point coordinates) features.

We presented an EM algorithm in Chapter 4 for estimating the parameters of a static distribution from which unlabeled point sets are presumed to be drawn. The algorithm consists of using a Kalman filter in the expectation phase of an EM algorithm. Modifications to the Kalman filter were proposed to handle intractable distributions resulting from unknown point labels and improve the likelihood of the algorithm's output. The algorithm is versatile in that arbitrary probability distributions may be assigned to the labels. Simulations found that AFEM had significant advantages versus an EM algorithm without the Kalman filter.

In Chapter 5, we presented proof that PD kernels can be derived from EMD and are dependent on the ground distance and the space in which it operates. We set our discussions in the context of set theory, providing motivation for our derivations and an intuitive interpretation of EMD's value, namely as a generalization of otherwise binary set operations. In doing so, we generalized $\widehat{\text{EMD}}$ for kernels. We also proposed a PD preserving transformation that normalizes a kernel's values and showed that the Jaccard index is simply the result of this transformation applied to the intersection kernel. As a corollary, the biotope transform was shown to preserve CND as well as metric properties. Finally, we provided the first assessment of $\widehat{\text{EMD}}$ in a kernel setting and showed that it and its biotope transform EMJD achieve superior accuracy over EMD on experiments with unnormalized sets and a state of the art indefinite kernel technique. Indeed, we showed that an indefinite kernel technique may not even be necessary. EMJD was found to have more favorable numerical properties than $\widehat{\text{EMD}}$.

In Chapter 6, we proposed neural networks capable of recognizing gestures represented by variable-length sequences of motion capture frames. We relied upon the rigid pattern on the back of the glove (Figure 1.2) to establish a local coordinate system in which our classifiers operated. Our proposed neural network architectures were not quite as effective at posture recognition as the SVM considered in Chapter 5, although they were much more computationally efficient. We also found that extracting features from the raw unlabeled markers using appropriately structured ANNs was more effective than a data-driven algorithm for sorting the markers by estimated labels. Further refinement of the proposed architectures along with layer-by-layer training

as stacked denoising autoencoders [141] or with residual learning [56] may yield even better results with raw unlabeled markers, especially for the CDAN architecture.

Ultimately, we found evidence that practical hand posture and gesture recognition is possible with motion capture cameras and unlabeled markers. The primary factors that control the feasibility and performance of a recognition system are the number of cameras, their configuration, and the amount of training data. If there are too few cameras or they are poorly placed, then the data used to train gesture classifiers will be of poor quality and unlikely to be representative of data encountered at a later time. Similarly, if not enough training data exists, then the generalization error encountered in practice is likely to be large. Deep learning appears especially promising as a potential solution, but these two factors will continue to play a major role in any future work.

7.2 Future Work

A significant amount of potential future work exists. The following subsections highlight areas of particular importance.

7.2.1 Enhanced Data Collection

Note that despite the variance in the accuracies reported in Tables 6.1 and 6.2, we found that the generalization error was consistently low when no users were left out. These results indicate that the challenge is not necessarily in the data but in generalizing to new users. The simplest way to address this problem is with the collection of more data, which is a characteristic common to machine learning algorithms.

In general, the quality of results in this work was highly dependent on the quality and amount of data that supported it. We collected data for a limited corpus of postures and gestures from 12 users. The number of users and the size of the corpus should be increased to assess the scalability of different algorithms and obtain more positive results. A camera configuration that allows a wide range of motion during capture should also be emphasized. A collection of (manually) labeled data would also be very beneficial for developing and evaluating future marker labeling algorithms.

Some needs were not foreseen at the time of collection. For example, gestures were captured as isolated segments of a stream of frames, and sequences of motion not corresponding to any gesture were ignored. This type of data capture, though useful for assessing whether a given classifier is capable of distinguishing gestures, is not entirely appropriate for online processing of a stream that may contain multiple gestures. Future data collection should aim to be compatible with so-called connectionist temporal classification [48], wherein the streams need only be labeled with the sequence of gestures they contain without any segmentation.

7.2.2 Constrained k-Means

In Chapter 4, we noted that the AFEM algorithm implicitly defined a constrained k-means algorithm. Future work could focus on clarifying this statement by rigorously expressing a variant of AFEM as such and evaluating the resulting algorithm against other constrained k-means algorithms. AFEM is also constrained by the requirement to know the number of targets beforehand. We believe this requirement could be relaxed by adapting the method of Figueiredo and Jain [38]

for GMM estimation, which uses the minimum message length criterion [146] to simultaneously estimate the number of components and their parameters. On a related note, the outer loop of the AFEM algorithm can probably be removed by shifting the covariance Σ and occlusion probability vector $\boldsymbol{\pi}$ directly into the state of a Bayes filter or smoother.

7.2.3 Improved Online Marker Tracking and Labeling

The marker tracking method proposed in Section 6.2.1 is relatively simple and largely dependent on heuristics. The method consists of a filtering phase followed by a labeling phase. The AFEM algorithm of Chapter 4 suggests a more principled method for filtering based on multiple assignments. Alternatively, a PHD filter [142] could be used that avoids assignments in the filtering phase. The labeling phase could be improved by employing one or more grid-based filters [4] that maintain the discrete label probability distribution for each filter. Grid-based filters form the class of closed-form Bayes filters for discrete processes just as Kalman filters form the class of closed-form Bayes filters for linear Gaussian processes. The resulting method could be compared against Alexanderson *et al.* [2], which could in fact probably benefit from a similar application of a grid-based filter.

7.2.4 MCMC Algorithms for Weighted Permutations

The sequential match sampling algorithm used in Chapter 4 by Volkovs and Zemel [144] has a temperature hyperparameter with no clear guidelines for its optimal value. We also found that the algorithm could get stuck in low density areas for highly skewed distributions and certain temperature ranges. Future work could involve

designing an improved sampler that avoids getting stuck or investigating efficient rules or heuristics for choosing the temperature.

7.2.5 Posture and Gesture Recognition in Global Coordinates

Working in global coordinates tends to add a great deal of complexity as classifiers may need to be translation or rotation invariant depending on the posture or gesture. Establishing a local coordinate system for the hand based on a rigid pattern was an expedient way to resolve this issue. However, this solution also imposes one of the greatest limitations on our work as the failure to observe or correctly identify the pattern renders many of the proposed methods inapplicable. We believe that spatially sparse CNNs [46] or similar networks offer an elegant solution that allows us to completely forgo any use of a rigid pattern or local coordinate system. In essence, one may consider a high-resolution grid over the entire capture space represented by a 3D tensor whose elements record the presence or absence of markers at grid locations. Exploiting sparse matrix representations [43] allows us to efficiently represent this tensor in a manner that scales with the number of visible markers rather than the grid's resolution. Convolutional layers can be implemented to inherit this sparsity, and pooling layers can eventually reduce the size of the grid to a fixed, manageable size while feeding important features to deeper layers of the network. Convolutions are also translation invariant and can be made rotation invariant given enough data and the correct structure. A spatial representation via a grid also avoids the combinatorial issues of ordering the markers in a list. In some ways, a spatially sparse grid can be

seen as a scalable, high-resolution descendant of the grid transformation proposed in Chapter 3.

7.2.6 Marker Filtering and Labeling with Neural Networks

The role of neural networks can be expanded beyond just classification to include marker filtering and labeling, especially with the spatially sparse representation described in the previous subsection. A denoising autoencoder [141] can be used to reconstruct missing markers or remove extraneous ones wherein a neural network is trained to output an observed sequence of markers (or spatially sparse grid representation of them) given a noisy version of the sequence. The denoising autoencoder (or a different ANN) could also be trained to simultaneously output the label or label probabilities of each marker assuming a dataset of labeled marker sets is available.

7.2.7 Kernel Methods for Gesture Recognition

Despite the shift to neural networks in Chapter 6, SVMs with EMD-based kernels are still a promising route to accomplish gesture recognition. The fact that labels (or the lack thereof) have no effect on their computation is a tremendous advantage. Despite the high accuracy reported in Chapter 5, the practicality of EMD-based kernels are limited by their computational complexity. However, with the advent of computationally efficient approximations of EMD such as the sinkhorn distance [24] and the convolutional Wasserstein distance [127], kernel methods may still be competitive with deep learning. Aside from SVMs, these methods also include kernel Kalman filters [109] and kernel principal component analysis (PCA) [96]. The former

can possibly be used as a label-free method to filter unlabeled markers whereas the latter could be used to calculate a permutation invariant feature vector representation of an unlabeled marker set. The kernel PCA representation could be used in myriad ways including but not limited to input to a neural network. Further work more directly related to the content of Chapter 5 could explore generalizations to set operations involving more than two sets, analyzing connections to rough or fuzzy set theory, a more in depth exploration of the proposed kernel transformation, further study on the definiteness of EMD, and various applications including the use of EMJD in the performance evaluation of multi-object filters.

7.2.8 A Kernel Trick for Optimal Transport

This final subsection on future work is more speculative than the previous and focuses on answering the question of whether a heat kernel can be defined and computed for CND ground distances. The motivation is to extend the convolutional Wasserstein distance of Solomon *et al.* [127] to CND ground distances, thereby enabling its application to a wider class of problems. We base our hypothesis on the facts that CND kernels with finite dimensional feature maps are equivalent to high-dimensional squared Euclidean distances and that the heat kernel for a flat finite-dimensional Euclidean manifold is a function of the distance. These facts form a reasonable basis to suggest that the convolutional Wasserstein distance can in fact be applied to certain nonlinear, non-geometric domains. The confounding issue is the fact that many CND kernels of interest correspond to infinite dimensional Hilbert spaces for which the heat kernel may not exist [33]. Furthermore, actual computation of the convolutional

Wasserstein distance may not translate even if the kernel exists. The point cloud Laplacian described by Crane [22] and Liu *et al.* [85] may or may not be sufficient. Consequently, based on the author's current knowledge, the difficulty in resolving this problem ranges anywhere from trivial to impossible.

APPENDIX A

NOTATION

This appendix serves as a reference for the rest of the dissertation regarding notation. Table A.1 provides a list of defined mathematical notation, and Table A.2 provides a list of acronyms used in various chapters.

Table A.1: A list of symbols and notation used throughout the dissertation along with definitions and short descriptions.

Notation	Description
$\mathbb{R}$, $\mathbb{Z}$	Real numbers, integers. $\mathbb{R}^+$ (resp. $\mathbb{Z}^+$) indicates positive numbers.
$[a, b]$	The closed interval from a to b in $\mathbb{R}$ or $\mathbb{Z}$ as context indicates.
$\mathbf{v}$	A column vector.
$V^\intercal$	The transpose of the matrix V.
v_i	The i-th coordinate of the vector $\mathbf{v}$.
$V_{i,j,\ldots,k}$	The $ij\ldots k$-th element (i-th row, j-th column, etc.) of the tensor V.
$\|\mathbf{v}\|$	The magnitude (2-norm) of the vector $\mathbf{v}$. Equal to $\sqrt{\mathbf{v}^\intercal\mathbf{v}}$.
$\|V\|$	The determinant of the matrix V.
$\operatorname{tr}[V]$	The trace of the matrix V.
$\mathbf{s}_n$	The $n \times n$ matrix of the scalar s, e.g. $\mathbf{0}_n$.
$\mathbf{s}_{n\times m}$	The $n \times m$ matrix of the scalar s, e.g. $\mathbf{0}_{n\times m}$.
$\mathbf{s}$	A tensor of scalars equal to s whose dimensions match the context.
$\mathbf{I}_n$	The $n \times n$ identity matrix.
$\mathbf{I}_{n\times m}$	An otherwise zero matrix containing $\mathbf{I}_{\min[n,m]}$ in the upper left corner.
$\mathbf{I}$	An identity matrix whose dimensions match the context.
$\mathbf{v} \times \mathbf{u}$	The cross product of the 3D vectors $\mathbf{v}$ and $\mathbf{u}$.
$\lfloor \mathbf{v}\times \rfloor$	The matrix $V = \lfloor \mathbf{v}\times \rfloor$ that satisfies $V\mathbf{u} = \mathbf{v} \times \mathbf{u}$ for any vector $\mathbf{u}$.
$\overline{q}$	A quaternion.
$\operatorname{quat}(\mathbf{v})$	The quaternion constructed from $\mathbf{v}$ according to (2.12).
$\operatorname{Im}(\overline{q})$	The imaginary vector component of the quaternion $\overline{q}$.
$\operatorname{vec}(V)$	The vectorization of the matrix V formed by stacking its columns.
$\operatorname{diag}(\mathbf{v})$	A diagonal matrix with the elements of $\mathbf{v}$ on the diagonal.
$V \otimes W$	The Kronecker tensor product of V and W.
$V \odot W$	The Hadamard product of V and W, i.e. element-wise multiplication.
$\mathbb{E}_X[Y]$	The expectation of Y taken with respect to the distribution of the random variable X.
$X \sim Y$	The random variable X has distribution Y.
$\mathbb{I}_X(x)$	Indicator function of the set X. 1 if $x \in X$, 0 otherwise.
$\operatorname{supp}(\mu)$	The support of the measure $\mu : X \to \mathbb{R}$, i.e. $\{x \mid \mu(x) > 0, x \in X\}$.

Table A.2: A list of acronyms and their expansions used throughout the dissertation.

Acronym	Expansion
AFEM	*a fortiori* expectation-maximization
ANN	artificial neural network
BER	balanced error rate
CDAN	convolutional deep averaging network
CNN	convolutional neural network
CND	conditionally negative definite
CPD	conditionally positive definite
CSV	comma separated value
EKF	extended Kalman filter
EM	expectation-maximization
EMD	earth mover's distance
EMI	earth mover's intersection
EMJD	earth mover's Jaccard distance
FS	feature selection
GMM	Gaussian mixture model
GRU	gated recurrent unit
JPDA	joint probabilistic data association
JS	Jensen-Shannon
KL	Kullback-Leibler
KKT	Karush-Kuhn-Tucker
k-NN	k-nearest neighbor
KSVM	Krein support vector machine
MCMC	Markov chain Monte Carlo
MLP	multi-layer perceptron
ND	negative definite
PCA	principal component analysis
PD	positive definite
PDE	partial differential equation
PHD	probability hypothesis density
RBF	radial basis function
RDAN	recurrent deep averaging network
ReLU	rectified linear unit
RNN	recurrent neural network
SVM	support vector machine

APPENDIX B

DATASETS

This appendix describes the collection, features, and organization of datasets gathered for this dissertation. In each case, the Vicon motion capture system described in Section 1.2.1 was used to collected the data. Each dataset can be downloaded separately as a zip archive of its described file format at `http://www2.latech.edu/~jkanno/collaborative.htm`.

B.1 General Remarks

A rigid pattern of markers on the back of the glove is used to establish a local coordinate system for the hand, and 11 other markers are attached to the thumb and fingers of the glove. Three markers are attached to the thumb with 1 above the thumbnail and the other 2 on the interphalangeal and metacarpophalangeal joints (i.e. the knuckles). Two markers are attached to each finger with 1 above the fingernail and the other on the proximal interphalangeal joints (see Figure 1.2 for a detailed view).

The pattern of markers visible in Figure 1.2 on the back of the glove plays an important role in establishing a local coordinate system for posture and gesture recognition. Four markers comprise the pattern and are given the labels "Origin," "XMarker," "YMarker," and "Extra." Four is the minimum number of markers required to define a pattern in Vicon Tracker, although only 3 must be visible in order for the pattern to be detected. The axes of the local coordinate system are determined according to the pseudocode in Figure B.1, which assumes that the origin is not occluded and tries to recover if any of the other markers are not visible.

procedure getLocalCoordinateAxes

Given: Origin $\mathbf{o}$, XMarker $\mathbf{x}$, YMarker $\mathbf{y}$, Extra $\mathbf{e}$
Output: local x-axis $\mathbf{x}^\star$, y-axis $\mathbf{y}^\star$, z-axis $\mathbf{z}^\star$

if XMarker is not occluded & YMarker is not occluded **then**
  $\mathbf{x}^\star = \mathbf{x} - \mathbf{o}$
  $\mathbf{y}^\star = \mathbf{y} - \mathbf{o}$
  $\mathbf{z}^\star = \mathbf{x}^\star \times \mathbf{y}^\star$
else if YMarker is not occluded **then**
  $\mathbf{y}^\star = \mathbf{y} - \mathbf{o}$
  $\mathbf{z}^\star = (\mathbf{e} - \mathbf{o}) \times \mathbf{y}^\star$
  $\mathbf{x}^\star = \mathbf{y}^\star \times \mathbf{z}^\star$
else if XMarker is not occluded **then**
  $\mathbf{x}^\star = \mathbf{x} - \mathbf{o}$
  $\mathbf{z}^\star = \mathbf{x}^\star \times (\mathbf{e} - \mathbf{o})$
  $\mathbf{y}^\star = \mathbf{z}^\star \times \mathbf{x}^\star$
end if
$\mathbf{x}^\star = \mathbf{x}^\star / \|\mathbf{x}^\star\|$
$\mathbf{y}^\star = \mathbf{y}^\star / \|\mathbf{y}^\star\|$
$\mathbf{z}^\star = \mathbf{z}^\star / \|\mathbf{z}^\star\|$

Figure B.1: Pseudocode for calculating axes of the hand's local coordinate system using labeled markers.

B.2 Labeled Marker Dataset

This section describes the dataset of labeled markers and its associated file format.

B.2.1 Data Collection and Description

In contrast to the posture and gesture datasets, a single user donated this data. The purpose of this dataset is to provide the range of motion for each part of the hand/glove to which a marker is attached. This dataset is naturally limited in that it cannot apply to all potential users, but it may still serve as a basis for future algorithm development.

In order to be absolutely certain that no confusion between markers was possible, only a single unlabeled marker was attached to the glove at a time during capture. The user performed a full range of motion with each marker.

The data described here is already preprocessed. First, all markers were transformed to the local coordinate system of the record containing them using the axes given by the algorithm in Figure B.1. Any record that could not be transformed was dropped. Second, each transformed marker with a norm greater than 200 millimeters was pruned. Finally, any record that contained more than one marker was dropped. Figure B.2 provides a plot the processed data.

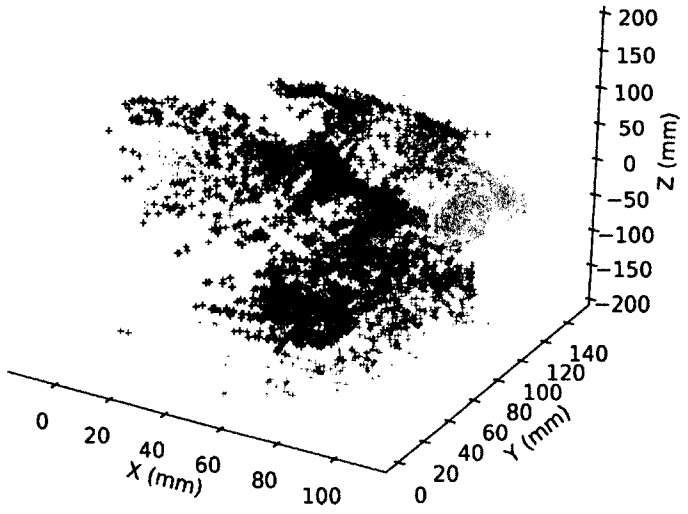


Figure B.2: The labeled marker dataset after processing (i.e. in local coordinates). Some outliers for certain classes are visible.

B.2.2 File Format

Data is provided as a comma separated value (CSV) file. A header row provides the name of each attribute. There are no missing values. Each record corresponds to

the position of a single labeled marker. The attributes are defined in the following list and are enumerated by their names:

- 'Class': Integer. The class ID of the given record. Ranges from 1 to 11 with

  $1 \mapsto$ Pinky Finger (Joint),

  $2 \mapsto$ Pinky Finger (Nail),

  $3 \mapsto$ Ring Finger (Joint),

  $4 \mapsto$ Ring Finger (Nail),

  $5 \mapsto$ Middle Finger (Joint),

  $6 \mapsto$ Middle Finger (Nail),

  $7 \mapsto$ Pointer Finger (Joint),

  $8 \mapsto$ Pointer Finger (Nail),

  $9 \mapsto$ Thumb (Metacarpophalangeal Joint),

  $10 \mapsto$ Thumb (Interphalangeal Joint),

  $11 \mapsto$ Thumb (Nail).

- 'X': Float. The x-coordinate of the marker.
- 'Y': Float. The y-coordinate of the marker.
- 'Z': Float. The z-coordinate of the marker.

B.3 Posture Dataset

This section describes the posture dataset used throughout the dissertation and its associated file format. Figure B.3 provides illustrations of instances within the dataset.

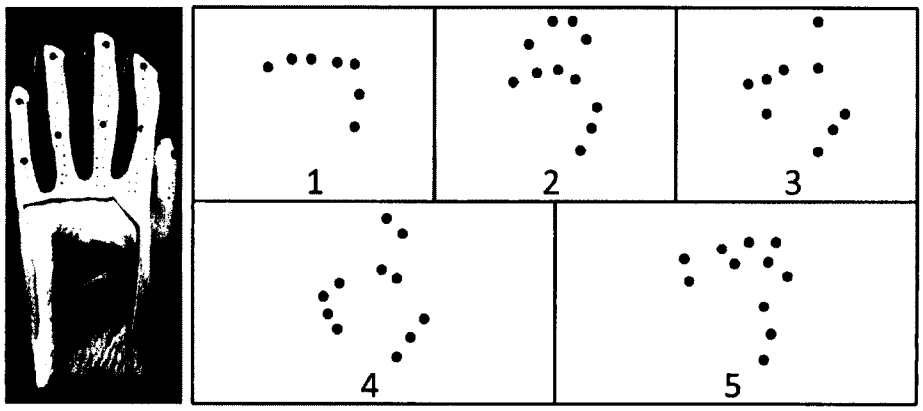


Figure B.3: The glove used to capture data along with a sample from each class of posture projected onto the local XY plane. The classes are fist (1), stop (2), point with one finger (3), point with two fingers (4), and grab (5).

B.3.1 Data Collection and Description

We recorded 12 users performing five hand postures with markers attached to a left-handed glove (Figure B.3).

The 11 markers not part of the rigid pattern were unlabeled; their positions were not explicitly tracked. Consequently, there is no *a priori* correspondence between the markers of two given records. In addition, due to the resolution of the capture volume and self-occlusion due to the orientation and configuration of the hand and fingers, many records have missing markers. Extraneous markers were also possible due to artifacts in the Vicon software's marker reconstruction/recording process and other objects in the capture volume. As a result, the number of visible markers in a record varies considerably.

The data described here is already partially preprocessed in the following manner. The data was transformed and pruned in the same manner as the Labeled Marker Dataset. Any record that could not be transformed or contained fewer than three markers was removed. The processed data has at most 12 markers per record and

at least three, which implies that at least one record has an extraneous marker. See the next subsection for more information on the attributes and file format. Unprocessed data in global coordinate is also available, but is not used anywhere in the dissertation and therefore an associated file format is not described.

Due to the manner in which data was captured, it is likely that for a given record and user there exists a near duplicate record originating from the same user. We recommend therefore to evaluate classification algorithms on a leave-one-user-out basis wherein each user is iteratively left out from training and used as a test set. One then tests the generalization of the algorithm to new users. The 'User' attribute is provided to accommodate this strategy.

This dataset may be used for a variety of tasks, the most obvious of which is posture recognition via classification. One may also attempt user identification. Alternatively, one may perform clustering (constrained or unconstrained) to discover marker distributions either as an attempt to predict marker identities or obtain statistical descriptions/visualizations of the postures (for example, the content of Chapter 4).

B.3.2 File Format

Data is provided as a CSV file. A header row provides the name of each attribute. An initial dummy record composed entirely of zeros should be ignored (this record was included for compatibility with WEKA [52]). A question mark '?' is used to indicate a missing value. A record corresponds to a single instant or frame as recorded by the camera system. Descriptions of each attribute are provided in the following list organized by attribute name:

- 'Class': Integer. The class ID of the given record. Ranges from 1 to 5 with

  $1 \mapsto$ Fist (with thumb out),

  $2 \mapsto$ Stop (hand flat),

  $3 \mapsto$ Point1 (point with pointer finger),

  $4 \mapsto$ Point2 (point with pointer and middle fingers),

  $5 \mapsto$ Grab (fingers curled as if to grab).

- 'User': Integer. The ID of the user that contributed the record. No meaning other than as an identifier.
- 'Xi': Float. The x-coordinate of the i-th unlabeled marker position. 'i' ranges from 0 to 11.
- 'Yi': Float. The y-coordinate of the i-th unlabeled marker position. 'i' ranges from 0 to 11.
- 'Zi': Float. The z-coordinate of the i-th unlabeled marker position. 'i' ranges from 0 to 11.

Each record is a set. The i-th marker of a given record does not necessarily correspond to the i-th marker of a different record. One may randomly permute the visible (i.e. not missing) markers of a given record without changing the set that the record represents. For the sake of convenience, all visible markers of a given record are given a lower index than any missing marker. A class is not guaranteed to have even a single record with all markers visible.

B.4 Gesture Dataset

This section describes the gesture dataset used in Chapter 6 and its associated file format.

B.4.1 Data Collection and Description

The same 12 users of the posture dataset reprised their roles for this dataset. Each user repeated each of six gestures for approximately 30 times.

Since the pattern is not always visible and has noisy or even incorrect observations, a filter should be used to smooth the measurements of the labeled markers. Since there are many ways one could define a filter for this purpose, no processing has been performed on the data as it could bias subsequent results. As a result of no pruning or local transformations, the number of unlabeled markers (i.e. not including the pattern) can be as high as 16 due to artifacts of the capture. See Chapter 6 for an example of an extended Kalman filter (Section 2.5.3) that simultaneously estimates the position and orientation of the pattern.

There is less of an issue with duplicated gestures than with postures, but we still advise evaluating the dataset with a leave-one-user-out approach. Once again, a 'User' attribute is provided to accommodate this strategy.

B.4.2 File Format

Data is provided as a CSV file. Two header rows provide the name of each attribute. The first header row indicates the attributes for an entire sequence of frames that together constitute a single gesture. The beginning of a gesture is heralded by the word "Start" at the beginning of the first header. The second header indicates

the attribute names for individual frames. An initial dummy sequence composed entirely of zeros is provided immediately after the headers as an example and should be ignored. Question marks are used to indicate missing values. Descriptions of each attribute are provided in the following list organized by attribute name:

- 'Class': Integer. The class ID of the given record. Ranges from 1 to 6 with

  $1 \mapsto$ Click (or poke with pointer finger),

  $2 \mapsto$ SwipeLeft (casual backhand as if swiping away),

  $3 \mapsto$ SwipeRight (opposite motion of SwipeLeft),

  $4 \mapsto$ TurnGrab (same as grab, but with left-handed rotation about forearm axis),

  $5 \mapsto$ Grab (hand closes with fingers outstretched),

  $6 \mapsto$ Release (opposite motion of grab).

- 'User': Integer. The ID of the user that contributed the record. No meaning other than as an identifier.
- 'Origin-X': Float. The x-coordinate of the origin marker of the rigid pattern.
- 'Origin-Y': Float. The y-coordinate of the origin marker of the rigid pattern.
- 'Origin-Z': Float. The z-coordinate of the origin marker of the rigid pattern.
- 'XMarker-X': Float. The x-coordinate of the X-axis marker of the rigid pattern.
- 'XMarker-Y': Float. The y-coordinate of the X-axis marker of the rigid pattern.
- 'XMarker-Z': Float. The z-coordinate of the X-axis marker of the rigid pattern.
- 'YMarker-X': Float. The x-coordinate of the Y-axis marker of the rigid pattern.
- 'YMarker-Y': Float. The y-coordinate of the Y-axis marker of the rigid pattern.

- 'YMarker-Z': Float. The z-coordinate of the Y-axis marker of the rigid pattern.
- 'Extra-X': Float. The x-coordinate of the extra marker of the rigid pattern.
- 'Extra-Y': Float. The y-coordinate of the extra marker of the rigid pattern.
- 'Extra-Z': Float. The z-coordinate of the extra marker of the rigid pattern.
- 'Xi': Float. The x-coordinate of the i-th unlabeled marker position. 'i' ranges from 0 to 15.
- 'Yi': Float. The y-coordinate of the i-th unlabeled marker position. 'i' ranges from 0 to 15.
- 'Zi': Float. The z-coordinate of the i-th unlabeled marker position. 'i' ranges from 0 to 15.

Each record is a set in a sequence of sets. The i-th marker of a given record does not necessarily correspond to the i-th marker of a different record. One may randomly permute the visible (i.e. not missing) markers of a given record without changing the set that the record represents. For the sake of convenience, all visible markers of a given record are given a lower index than any missing marker. A class is not guaranteed to have even a single record with all markers visible.

BIBLIOGRAPHY

[1] Vicon Datastream SDK developer's manual, Jan. 2013.

[2] S. Alexanderson, C. O'Sullivan, and J. Beskow. Robust online motion capture labeling of finger markers. In *Proceedings of the 9th International Conference on Motion in Games*, MIG '16, pages 7–13, New York, NY, USA, 2016. ACM.

[3] A. Aristidou and J. Lasenby. Motion capture with constrained inverse kinematics for real-time hand tracking. In *Proceedings of the 4th International Symposium on Communications, Control and Signal Processing*, ISCCSP '10, pages 1–5, Mar. 2010.

[4] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, Feb. 2002.

[5] Y. Bar-Shalom, T. Kirubarajan, and X.-R. Li. *Estimation with Applications to Tracking and Navigation.* John Wiley & Sons, Inc., New York, NY, USA, 2002.

[6] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, Jan. 2009.

[7] C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*, volume 100 of *Graduate Texts in Mathematics.* Springer, 1st edition, Jun. 1984.

[8] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb. 1992.

[9] M. K. Bhuyan, D. Ghosh, and P. K. Bora. Hand motion tracking and trajectory matching for dynamic hand gesture recognition. *Journal of Experimental and Theoretical Artificial Intelligence*, 18(4):435–447, 2006.

[10] M. Bouchard, A.-L. Jousselme, and P.-E. Dor. A proof for the positive definiteness of the Jaccard index matrix. *International Journal of Approximate Reasoning*, 54(5):615–626, 2013.

[11] S. Boughorbel, J.-P. Tarel, and N. Boujemaa. Generalized histogram intersection kernel for image recognition. In *Proceedings of the IEEE International Conference on Image Processing*, volume 3 of *ICIP '05*, pages 161–164, Sept. 2005.

[12] R. L. Burden and J. D. Faires. *Numerical Analysis*. Brooks/Cole, 9th edition, 2011.

[13] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[14] M. G. Ceruti, V. V. Dinh, N. X. Tran, H. Van Phan, L. T. Duffy, T. Ton, G. Leonard, E. Medina, O. Amezcua, S. Fugate, G. J. Rogers, R. Luna, and J. Ellen. Wireless communication glove apparatus for motion tracking, gesture recognition, data transmission, and reception in extreme environments. In *Proceedings of the ACM Symposium on Applied Computing*, SAC '09, pages 172–176. ACM, 2009.

[15] L. Chang, N. Pollard, T. Mitchell, and E. Xing. Feature selection for grasp recognition from optical markers. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS '07, pages 2944–2950. IEEE, Nov. 2007.

[16] M.-S. Chang and J.-H. Chou. A robust and friendly human-robot interface system based on natural human gestures. *International Journal of Pattern Recognition and Artificial Intelligence*, 24(6):847–866, 2010.

[17] S. Chib and E. Greenberg. Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

[18] F. Chollet. Keras. `https://github.com/fchollet/keras`, Nov. 2016. Version 1.2.2.

[19] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*, 2014.

[20] S. Cohen and L. Guibas. The earth mover's distance under transformation sets. In *Proceedings of the 7th International Conference on Computer Vision*, volume 2 of *ICCV '99*, pages 1076–1083, Washington, DC, USA, 1999. IEEE Computer Society.

[21] T. F. Covões, E. R. Hruschka, and J. Ghosh. A study of k-means-based algorithms for constrained clustering. *Intelligent Data Analysis*, 17(3):485–505, May 2013.

[22] K. Crane, C. Weischedel, and M. Wardetzky. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics*, 32(5):152:1–152:11, Oct. 2013.

[23] M. Cuturi. Permanents, transport polytopes and positive definite kernels on histograms. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, IJCAI '07, pages 732–737, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[24] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems 26*, NIPS '13, pages 2292–2300, 2013.

[25] M. Cuturi and A. Doucet. Fast computation of Wasserstein barycenters. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *ICML '14*, pages 685–693. PMLR, Jun. 2014.

[26] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[27] M. R. Daliri. Kernel earth mover's distance for EEG classification. *Clinical EEG and Neuroscience*, 44(3):182–187, 2013.

[28] E. de Aguiar, C. Theobalt, and H.-P. Seidel. *Automatic Learning of Articulated Skeletons from 3D Marker Trajectories*, pages 485–494. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[29] J. Delon, J. Salomon, and A. Sobolevski. Fast transport optimization for Monge costs on the circle. *SIAM Journal of Applied Mathematics*, 70(7):2239–2258, 2010.

[30] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[31] M. M. Deza and E. Deza. *Encyclopedia of Distances*. Springer, 1st edition, Aug. 2009.

[32] L. Dipietro, A. M. Sabatini, and P. Dario. A survey of glove-based systems and their applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(4):461–482, Jul. 2008.

[33] B. K. Driver. Heat kernels measures and infinite dimensional analysis. *Contemporary Mathematics*, 338:101–142, 2003.

[34] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the Association for Computing Machinery*, 19(2):248–264, 1972.

[35] G. A. Einicke, J. T. Malos, D. C. Reid, and D. W. Hainsworth. Riccati equation and EM algorithm convergence for inertial navigation alignment. *IEEE Transactions on Signal Processing*, 57(1):370–375, Jan. 2009.

[36] L. C. Evans. Partial differential equations and Monge-Kantorovich mass transfer. In *Current Developments in Mathematics*, CDM 1997, pages 65–126, Boston, MA, 1999. International Press.

[37] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, Apr. 2006.

[38] M. A. T. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, Mar. 2002.

[39] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8(3):173–184, Jul. 1983.

[40] B. Fuglede and F. Topsoe. Jensen-Shannon divergence and Hilbert space embedding. In *Proceedings of the International Symposium on Information Theory*, ISIT '04, pages 31–36, 2004.

[41] W. Gangbo. An introduction to the mass transportation theory and its applications. *Gangbo's notes from lectures given at the 2004 Summer Institute at Carnegie Mellon University and at IMA in March 2005*, Mar. 2005.

[42] L. Ge, H. Liang, J. Yuan, and D. Thalmann. Robust 3D hand pose estimation in single depth images: From single-view CNN to multi-view CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 3593–3601. IEEE Computer Society, Jun. 2016.

[43] J. R. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in MATLAB: Design and implementation. *SIAM Journal on Matrix Analysis and Applications*, 13(1):333–356, 1992.

[44] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[45] J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27(4):857–871, 1971.

[46] B. Graham. Spatially-sparse convolutional neural networks. *arXiv:1409.6070*, Sept. 2014.

[47] K. Grauman and T. Darrell. The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research*, 8:725–760, May 2007.

[48] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labeling unsegmented sequence data with recurrent neural networks. In 200, editor, *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 369–376, New York, NY, USA, Jun. 2006. ACM.

[49] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, ICASSP '13, pages 6645–6649. IEEE, May 2013.

[50] I. Griva, S. Nash, and A. Sofer. *Linear and Nonlinear Optimization.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA 19104, 2nd edition, 2009.

[51] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, Mar. 2003.

[52] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorer Newsletter*, 11(1):10–18, Nov. 2009.

[53] J. Han, L. Shao, D. Xu, and J. Shotton. Enhanced computer vision with Microsoft Kinect sensor: A review. *Cybernetics, IEEE Transactions on*, 43(5):1318–1334, 2013.

[54] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[55] E. Hayman, B. Caputo, M. Fritz, and J.-O. Eklundh. On the significance of real-world conditions for material classification. In *Proceedings of the European Conference on Computer Vision*, ECCV '04, May 2004.

[56] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778. IEEE Computer Society, Jun. 2016.

[57] M. Hein and O. Bousquet. Hilbertian metrics and positive definite kernels on probability measures. In *Proceedings of the 10th International Conference on Artificial Intelligence and Statistics*, AISTATS '05, pages 136–143, 2005.

[58] J. R. Hoffman and R. P. S. Mahler. Multitarget miss distance via optimal assignment. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 34(3):327–336, May 2004.

[59] R. V. Hogg, J. W. McKean, and A. T. Craig. *Introduction to Mathematical Statistics.* Pearson Education, Inc., 7th edition, 2013.

[60] P. Honeine and C. Richard. The angular kernel in machine learning for hyperspectral data classification. In *Proceedings of the 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, WHISPERS '10, pages 1–4, Jun. 2010.

[61] M. F. Huber, T. Bailey, H. Durrant-Whyte, and U. D. Hanebeck. On entropy approximation for Gaussian mixture random vectors. In *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, MFI '08, pages 181–188, Aug. 2008.

[62] N. Igbida, J. Mazn, J. Rossi, and J. Toledo. A Monge-Kantorovich mass transport problem for a discrete distance. *Journal of Functional Analysis*, 260(12):3494–3534, 2011.

[63] M. Iyyer, V. Manjunatha, J. L. Boyd-Graber, and H. Daum III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, ACL '15, pages 1681–1691. The Association for Computer Linguistics, 2015.

[64] A. K. Jain, J. Mao, and K. Mohiuddin. Artificial neural networks: A tutorial. *IEEE Computer*, 29:31–44, 1996.

[65] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51(4):671–697, Jul. 2004.

[66] B. Jian and B. Vemuri. Robust point set registration using Gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1633–1645, Aug. 2011.

[67] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[68] L. V. Kantorovich. On the translocation of masses. *C. R. (Doklady) Acad. Sci. USSR*, 321:199–201, 1942.

[69] L. V. Kantorovich. On a problem of Monge. *Uspekhi Matematicheskikh Nauk*, 3(2):225–226, 1948.

[70] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

[71] S. Kolouri, Y. Zou, and G. K. Rohde. Sliced Wasserstein kernels for probability distributions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16. IEEE Computer Society, Jun. 2016.

[72] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, NIPS '12, pages 1097–1105. Curran Associates, Inc., 2012.

[73] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[74] Y. Lecun and C. Cortes. The MNIST database of handwritten digits, 1998.

[75] Y.-H. Lee and C.-Y. Tsai. Taiwan sign language (TSL) recognition based on 3D data and neural networks. *Expert Systems with Applications*, 36(2):1123–1128, 2009.

[76] E. Levina and P. Bickel. The earth mover's distance is the Mallows distance: some insights from statistics. In *Proceedings of the 8th IEEE International Conference on Computer Vision*, volume 2 of *ICCV '01*, pages 251–256, 2001.

[77] F. L. Lewis, A. Yesildirak, and S. Jagannathan. *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Taylor & Francis, Inc., Bristol, PA, USA, 1998.

[78] H. Liang, J. Wang, Q. Sun, Y.-J. Liu, J. Yuan, J. Luo, and Y. He. Barehanded music: Real-time hand interaction for virtual piano. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '16, pages 87–94, New York, NY, USA, 2016. ACM.

[79] F. Liese and I. Vajda. On divergences and informations in statistics and information theory. *IEEE Transactions on Information Theory*, 52(10):4394–4412, Oct. 2006.

[80] D. Lin and J. W. Fisher III. Efficient sampling from combinatorial space via bridging. In N. D. Lawrence and M. A. Girolami, editors, *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, volume 22 of *AISTATS '12*, pages 694–702, La Palma, Canary Islands, 2012.

[81] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv:1312.4400*, 2013.

[82] H. Ling and K. Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, 29(5):840–853, May 2007.

[83] G. Liu and L. McMillan. Estimation of missing markers in human motion capture. *Vis. Comput.*, 22(9):721–728, Sept. 2006.

[84] G. Liu, J. Zhang, W. Wang, and L. McMillan. Human motion estimation from a reduced marker set. In *Proceedings of the 10th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '06, pages 35–42, New York, NY, USA, 2006. ACM.

[85] Y. Liu, B. Prabhakaran, and X. Guo. Point-based manifold harmonics. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1693–1703, Oct. 2012.

[86] C. F. Loan. The ubiquitous kronecker product. *Journal of Computational and Applied Mathematics*, 123(12):85–100, 2000.

[87] G. Loosli, S. Canu, and C. S. Ong. Learning SVM in Krein spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(6):1204–1216, Jun. 2016.

[88] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

[89] O. Luzanin and M. Plancak. Hand gesture recognition using low-budget data glove and cluster-trained probabilistic neural network. *Assembly Automation*, 34(1):94–105, 2014.

[90] C. Mallows. Another comment on O'Cinneide. *The American Statistician*, 45(3):257, 1991.

[91] M. Martin, J. Maycock, F. P. Schmidt, and O. Kramer. Recognition of manual actions using vector quantization and dynamic time warping. In *Proceedings of the 5th International Conference on Hybrid Artificial Intelligence Systems*, HAIS '10, pages 221–228, Berlin, Heidelberg, 2010. Springer-Verlag.

[92] J. Maycock, J. Steffen, R. Haschke, and H. Ritter. Robust tracking of human hand postures for robot teaching. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS '11, pages 2947–2952. IEEE, 2011.

[93] R. K. Mehra. Approaches to adaptive filtering. In *Proceedings of the 9th IEEE Symposium on Adaptive Processes: Decision and Control*, pages 141–141, Dec. 1970.

[94] X.-L. Meng and D. B. Rubin. Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80(2):267–278, 1993.

[95] J. Meyer, M. Kuderer, J. Müller, and W. Burgard. Online marker labeling for fully automatic skeleton tracking in optical motion capture. In *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA '14, pages 5652–5657. IEEE, May 2014.

[96] S. Mika, P. B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, NIPS '99, pages 536–542. MIT Press, 1999.

[97] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 37(3):311–324, 2007.

[98] G. Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences*, pages 666–704, 1781.

[99] K. G. Murty. Letter to the editor–an algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16(3):682–687, 1968.

[100] A. Naor and G. Schechtman. Planar earthmover is not in $L1$. *SIAM Journal of Computing*, 37(3):804–826, 2007.

[101] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical report, University of Toronto, Sept. 2003. Technical Report CRG-TR-93-1.

[102] J. A. Noble. Finding corners. *Image and Vision Computing Journal*, pages 2–121, 1988.

[103] S. W. Nydick. The Wishart and inverse Wishart distributions. *Electronic Journal of Statistics*, 6:1–19, 2012.

[104] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2):338–350, 1993.

[105] O. Pele and M. Werman. A linear time histogram metric for improved SIFT matching. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, pages 495–508, Berlin, Heidelberg, 2008. Springer-Verlag.

[106] O. Pele and M. Werman. Fast and robust earth mover's distances. In *Proceedings of the 12th IEEE International Conference on Computer Vision*, ICCV '09, pages 460–467, 2009.

[107] L. E. Potter, J. Araullo, and L. Carter. The Leap Motion Controller: A view on sign language. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, OzCHI '13, pages 175–178, New York, NY, USA, 2013. ACM.

[108] J. Rabin, J. Delon, and Y. Gousseau. Transportation distances on the circle. *Journal of Mathematical Imaging and Vision*, 41(1):147–167, 2011.

[109] L. Ralaivola and F. d'Alche Buc. Time series filtering, smoothing and learning using the kernel Kalman filter. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 3 of *IJCNN '05*, pages 1449–1454. IEEE, Jul. 2005.

[110] L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi. Graph kernels for chemical informatics. *Neural Networks*, 18(8):1093–1110, 2005. Neural Networks and Kernel Methods for Structured Domains.

[111] J. Ramon and M. Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica*, 37(10):765–780, 2001.

[112] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[113] B. Ristic, B.-N. Vo, D. Clark, and B.-T. Vo. A metric for performance evaluation of multi-target tracking algorithms. *IEEE Transactions on Signal Processing*, 59(7):3452–3457, 2011.

[114] G. Roffo, S. Melzi, and M. Cristani. Infinite feature selection. In *Proceedings of the 15th IEEE International Conference on Computer Vision*, ICCV '15, pages 4202–4210. IEEE, Dec. 2015.

[115] D. B. Rubin. Inference and missing data. *Biometrika*, 63:581–590, 1976.

[116] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40:2000, 2000.

[117] E. Sangineto, G. Zen, E. Ricci, and N. Sebe. We are not all equal: Personalizing models for facial expression analysis with transductive parameter transfer. In *Proceedings of the ACM International Conference on Multimedia*, MM '14, pages 357–366, New York, NY, USA, 2014. ACM.

[118] T. Schubert, A. Gkogkidis, T. Ball, and W. Burgard. Automatic initialization for skeleton tracking in optical motion capture. In *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA '15, pages 734–739. IEEE, May 2015.

[119] D. Schuhmacher, B. T. Vo, and B. N. Vo. A consistent metric for performance evaluation of multi-object filters. *IEEE Transactions on Signal Processing*, 56(8):3447–3457, Aug. 2008.

[120] J. Schur. Bemerkungen zur theorie der beschrnkten bilinearformen mit unendlich vielen vernderlichen. *Journal fr die reine und angewandte Mathematik*, 140:1–28, 1911.

[121] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, Nov. 1997.

[122] G. A. F. Seber. *Multivariate observations*. Wiley series in probability and mathematical statistics. Wiley, New York, NY, 1984.

[123] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, New York, NY, USA, 2004.

[124] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

[125] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1297–1304. IEEE Computer Society, 2011.

[126] A. J. Smola and S. Vishwanathan. *Introduction to Machine Learning.* Cambridge University Press, 2008.

[127] J. Solomon, F. de Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas. Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics*, 34(4):66:1–66:11, Jul. 2015.

[128] G. Song, H. Zhang, and F. J. Hickernell. Reproducing kernel Banach spaces with the $l1$ norm. *Applied and Computational Harmonic Analysis*, 34(1):96–116, 2013.

[129] Y. Song, D. Demirdjian, and R. Davis. Continuous body and hand gesture recognition for natural human-computer interaction. *ACM Transactions on Interactive Intelligent Systems*, 2(1):5:1–5:28, Mar. 2012.

[130] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[131] T. Tao. *An Introduction to Measure Theory.* Graduate studies in mathematics. American Mathematical Society, 2011.

[132] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv:1605.02688*, abs/1605.02688, May 2016. Version 0.8.2.

[133] N. Trawny and S. I. Roumeliotis. Indirect Kalman filter for 3D attitude estimation. Technical report, University of Minnesota, Department of Computer Science & Engineering, Mar. 2005. Technical Report-2005-002.

[134] S. R. S. Varadhan. On the behavior of the fundamental solution of the heat equation with variable coefficients. *Communications on Pure and Applied Mathematics*, 20(2):431–455, 1967.

[135] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[136] J. J. Verbeek, N. Vlassis, and B. Krse. Efficient greedy learning of Gaussian mixture models. *Neural Computation*, 15:469–485, 2003.

[137] R. Verde, A. Irpino, and A. Balzanella. Dimension reduction techniques for distributional symbolic data. *IEEE Transactions on Cybernetics*, 46(2):344–355, Feb. 2016.

[138] A. Vershik. Long history of the Monge-Kantorovich transportation problem. *The Mathematical Intelligencer*, 35(4):1–9, 2013.

[139] C. Villani. *Topics in optimal transportation.* Graduate studies in mathematics. American Mathematical Society, cop., Providence, RI, USA, 2003.

[140] C. Villani. *Optimal transport: old and new.* Grundlehren der mathematischen Wissenschaften. Springer, Berlin, 2009.

[141] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, Dec. 2010.

[142] B. N. Vo and W. K. Ma. The Gaussian mixture probability hypothesis density filter. *IEEE Transactions on Signal Processing*, 54(11):4091–4104, Nov. 2006.

[143] B. T. Vo and B. N. Vo. Labeled random finite sets and multi-object conjugate priors. *IEEE Transactions on Signal Processing*, 61(13):3460–3475, Jul. 2013.

[144] M. N. Volkovs and R. S. Zemel. Efficient sampling for bipartite matching problems. In *Advances in Neural Information Processing Systems 25*, NIPS '12, pages 1313–1321. Curran Associates Inc., 2012.

[145] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan. Vision-based hand-gesture applications. *Communications of the ACM*, 54(2):60–71, 2011.

[146] C. S. Wallace and D. L. Dowe. Minimum message length and Kolmogorov complexity. *The Computer Journal*, 42(4):270, 1999.

[147] E. A. Wan and R. V. D. Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158, 2000.

[148] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In S. Dasgupta and D. Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *ICML '13*, pages 1058–1066. PMLR, May 2013.

[149] R. Wang, S. Paris, and J. Popović. 6D hands: Markerless hand-tracking for computer aided design. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 549–558, New York, NY, USA, 2011. ACM.

[150] J. Weissmann and R. Salomon. Gesture recognition for virtual reality applications using data gloves and neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3 of *IJCNN '99*, pages 2043–2046, Jul. 1999.

[151] G. Welch and G. Bishop. An introduction to the Kalman filter. Technical report, University of North Carolina, Chapel Hill, NC, USA, 1995. Technical Report 95-041.

[152] N. Wheatland, Y. Wang, H. Song, M. Neff, V. Zordan, and S. Jörg. State of the art in hand and finger modeling and animation. *Computer Graphics Forum*, 34(2):735–760, May 2015.

[153] M. A. Woodbury. *Inverting Modified Matrices.* Number 42 in Statistical Research Group Memorandum Reports. Princeton University, Princeton, NJ, 1950.

[154] G. Wu, E. Y. Chang, and Z. Zhang. An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. In *Proceedings of the 22nd International Conference on Machine Learning*, volume 8 of *ICML '05*. ACM, Aug. 2005.

[155] A. Zamolotskikh and P. Cunningham. An assessment of alternative strategies for constructing EMD-based kernel functions for use in an SVM for image classification. In *Proceedings of the 5th International Workshop on Content-Based Multimedia Indexing*, CBMI '07, pages 11–17, Jun. 2007.

[156] J. Zhang, M. Marszaek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision*, 73(2):213–238, 2007.