Separability and Systematic Design of Gestural Human-Robot Interaction

Rose Hendrix

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2020

Reading Committee:

Santosh Devasia, Chair

Joseph Garbini, Chair

James Buttrick

Program Authorized to Offer Degree:
Department of Mechanical Engineering

University of Washington

Abstract

Separability and Systematic Design
of Gestural Human-Robot Interaction

Rose Hendrix

Co-Chairs of the Supervisory Committee:
Professor Santosh Devasia
Department of Mechanical Engineering

Professor Joseph Garbini
Department of Mechanical Engineering

This work aims to facilitate robust, systematic design of human-robot interactions that are mediated by gesture. In many domains, communication modalities such as voice or physical input are not feasible because of domain restrictions. In these cases, gesture interaction is necessary for explicit communication between a human and a robotic assistant or observer. Because the only input to the robotic platform's gesture recognition system is a continuously tracked hand, there is potential for confusion between the human motions of work and human motions intended to communicate control. Interaction design up to this point has largely been human-centered or ad-hoc in nature, rather than having systematic focus on performance. The design choices of interest to this work are the choice of control gestures (what motions to perform to communicate preset meanings) and feature representations (what collections of measurements or combinations thereof should represent a gesture).

In the absence of constraints on time and resources, the designer could try out all possible combinations of control gestures and feature representations in the final application context. However, that type of exhaustive testing takes a very long time for the researcher, may take

the time of skilled workers, and need to be repeated when processes or environmental factors are changed. The selection of control gestures is based around separability, a motion that a set should be measured against its worst cases that is explored in more detail in Chapter 3. The main contribution of this research is the formalization and validation of tools for systematic design of functional human-robot interaction without exhaustive testing. This is based on a selection method for maximally-separable control gestures and feature representations for any given specific work context.

There are several challenges involved in this goal. Firstly, it is not necessarily obvious what it means to compute the separability of a gesture set even with very simple class representations. How does separability as calculated correlate with overall performance? Secondly, the choice of how to measure "distance" between gesture classes such that correlation with performance measures is preserved is not trivial, particularly in the presence of outliers or small population sizes. Thirdly, there are many different ways to represent "gestures" numerically. How do these different representations interact with separability? How would a more separable representation be chosen? Finally, does this systematic design process generalize to other architectures for sensing human motion?

These challenges are addressed in order. First, a separability metric for systematically selecting a set of control gestures that can be easily distinguished from a given set of work gestures is proposed and examined in Chapter 3. A correlation between the proposed separability metric and test accuracy of gesture sets is found ($R = 0.59$), as well as a negative correlation ($R = -0.35$) between separability and sensitivity of performance to individuals' performances. Results are also presented for a comparative online classification implementation on a hand-following robot of the most- and least-separable gesture sets, where the most-separable gesture set performs an average of 4 times better across all metrics.

Second, a review of the considered measures of distance between gesture classes is pre-

sented, along with numeric and analytic justifications for the final selection of a modified version of the Support Vector Machines (SVM) margin of the model between the gesture classes.

Third, the problem of reduced-order gesture representation is considered and a modification of a method from literature and results are presented. A $>99\%$ reduction in dimensionality is reliably achieved without loss in test accuracy, and on average 17.5% fewer features are required for the same accuracy retention compared to the baseline method of PCA-based selection. Finally, the tools developed for systematic design are applied to a substantially different sensing architecture for human motion to validate the generalization ability of the process.

The main contribution of this research is a method to select gestures that best enable communication between a human and a robotic platform, and a reduced-order representation method to speed up computations without significant loss of accuracy.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

FEATURE: an individual property or characteristic of a gesture, evolving through time in the case of dynamic gestures. This is generally direct measurements of human motion, or some abstracted and processed combination of measurements.

GESTURE: A measured human motion or pose. In the case of control gestures, it is associated with active, explicit communication (examples: waving, beckoning). For work gestures, it is the motions associated with the completion of a task (examples: grasping, carrying).

GESTURE CLASS: Some coherent grouping of performance gestures. In some cases the association is defined by what humans naturally associate together, but it may also mean grouping automatically assigned by a clustering algorithm.

GESTURE SET: Some number of gesture classes that are being evaluated together. For example, one often-considered gesture set in this context is two control gesture classes and seven work gesture classes, for a total of nine gestures in this set. A set is often referred to with the variable s, and $\mathcal{S}$ is the set of all possible gesture sets.

SEPARABILITY: Broadly, the separability of a gesture set is an evaluation of the worst-case scenarios for classification in a gesture set. The concept is explored in Chapter 3 and experimental results on its correlation with test performance presented.

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to her co-advisors, Professor Santosh Devasia and Professor Joseph Garbini, without whose contributions and support this work would not have been possible. She also wishes to thank collaborators, past and present, particularly Maxx Yamasaki, Parker Owan, and Ken Latimer; her labmates in the BARC and the UPC; and the rest of her committee for their valuable insights at key points along the road. Finally, she wishes to thank Johannes, Maureen, Maz, Della, and Reese for their support throughout this process.

This work was supported in part through the Boeing Advanced Research Center (BARC) at the University of Washington.

Chapter 1

INTRODUCTION

1.1 Motivation

1.1.1 Gesture is an important medium for human-robot interaction (HRI)

There are a number of domains today in which humans use gesture to collaborate with each other, as shown in Fig. 1.1. In order to embed robots into these environments or partially

(a) Drivers must be able to comprehend gesture in the event that traffic is directed by an official

(b) Recreational divers and many technical divers use gesture as their primary mode of communication.

(c) Astronauts may need to use gestures as backup communication in the event of an emergency on equipment failure.

Figure 1.1: Examples of domains in which humans use gestural communication to collaborate or navigate.

automate these tasks, robots must be able to understand these gestures. Additionally, gesture is usually used in these domains because of limitations on other communication mediums, and in these domains robots would be subject to these same limitations. Some examples

of these domains include diving, military coordination, spacewalking, traffic control, and many others. A number of these domains are also high-value targets for automation or HRI, because they are strenuous or dangerous for the humans who perform them. However, due to their complex and dynamic nature, they are generally very challenging to fully automate. Thus, they are ideal candidates for gesture-mediated HRI.

There are also alternate domains where although humans may not use gesture now, robots working in that space may be subject to the same constraints. Primary among these is that the ability to communicate via verbal or direct communication may be limited or absent, and that some explicit communication is needed in order to collaborate or navigate. In this scenario, gesture would be critical for automation or semi-autonomous systems. Systematic design of those gestural communication systems becomes critical when surety of action is necessary and there are safety implications for user or public.

1.1.2 Careful design of gesture-mediated HRI is important for online applications

In gesture recognition, the representation space (i.e. how the gestures are represented numerically) is conventionally very high-dimensional and abstracted, and the designer's physical intuition for what gestures are distinct from each other may not necessarily apply. Depending on the measurement system and how error-prone it is, it's also easily possible to pick gestures that are arbitrarily similar to measurement failure states unique the that particular domain or sensor. Using systematic tools and data-driven validation, the system designer can be empowered to make design choices that are optimized for their particular domain and needs.

In the absence of constraints on time and resources, the designer could try out all possible choices to pick the best preferred gesture in the final application context. However, that type of exhaustive testing takes a very long time for the researcher, may take the time of skilled workers, and need to be repeated any time processes or environmental factors are changed.

1.1.3 Why hasn't systematic gesture selection been done before?

Gesture selection and separable design is not something that has been treated systematically before because previous gesture implementations have been focused on facilitating personal computing rather than more risky domains. When humans are interacting with embodied robots in complex domains attempting to perform work collaboratively, the acceptable error rate of the gesture recognition system is much lower than in personal computing applications. Therefore, as the application domains evolve there is a greater need for a systematic approach to design without exhaustive in-situ testing, which may also be risky in these domains.

1.2 Main contributions and associated challenges

The main contribution of this research is the formalization and validation of tools for systematic design of functional HRI without exhaustive testing. This is based on a selection method for maximally-separable control gestures and feature representations in a specific work context. These systematic design tools will then be validated in two online gesture recognition systems.

1.2.1 Introduction of terms

These term definitions are also available in the glossary for easier reference, but they would also benefit from an explicit introduction here.

Feature an individual property or characteristic of a gesture, evolving through time in the case of dynamic gestures. This is generally direct measurements of human motion, or some abstracted and processed combination of measurements.

Gesture A measured human motion or pose. In the case of control gestures, it is associated with active, explicit communication (examples: waving, beckoning). For work ges-

tures, it is the motions associated with the completion of a task (examples: grasping, carrying).

Gesture class Some coherent grouping of performance gestures. In some cases the association is defined by what humans naturally associate together, but it may also mean grouping automatically assigned by a clustering algorithm.

Gesture set Some number of gesture classes that are being evaluated together. For example, one often-considered gesture set in this context is two control gesture classes and seven work gesture classes, for a total of nine gestures in this set. A set is often referred to with the variable s, and $\mathcal{S}$ is the set of all possible gesture sets.

Separability Broadly, the separability of a gesture set is an evaluation of the worst-case scenarios for classification in a gesture set. The concept is explored in Chapter 3 and experimental results on its correlation with test performance presented.

1.2.2 Gesture selection with a given distance measures

The challenge of gesture selection overall is that the notion of "separability" is difficult to tie to specific evaluation methods. In this context it is the degree to which a group of gesture classes are distinguishable or distant from each other, and it is something that humans have an intuitive sense for in low dimensions, but that intuitive reasoning can break down in higher dimensions. Therefore, a method must be constructed that calculates separability of a group of objects (gesture classes) that agrees with human intuition for the notion of separability, predicts improved performance when maximized, and is consistent through a wide range of dimensionality.

The first tool to be tackled is then gesture selection. If an appropriate distance metric between gesture classes is chosen, how does the designer evaluate the separability of a potential set of control gestures in a particular work context? And does selection in this way lead

to improved performance outcomes in an online application? Answering these questions is the main topic of Chapter 3.

1.2.3 Pairwise gesture distance measures

The second challenge buried in the first is that an appropriate measure of "distance" between gesture classes is challenging to identify or formulate. Standard measures for comparatively evaluating distributions with as f-divergence or Support Vector Machines (SVM) margin do not provide a reliable estimate of separability between gestures (see Chapter 4 for explanation). Because it is a high-dimensional problem with many layers of abstraction in modern gesture recognition systems, intuitive spatial reasoning may break down.

The topic of what an appropriate distance metric is and critical evaluations of different possible candidates is the topic of Chapter 4. The goal of this effort is to design a robust measure of separability between gesture classes, according to desired separability properties detailed in the introduction to Chapter 4. The most important characteristics are that an increase in measured distance corresponds to an increase in partitioned test classification performance and that the measures are not brittle against pathological realizations in the training dataset. Ultimately, a modified version of the SVM margin is selected and numerical and analytic justification is presented.

1.2.4 Feature representation

Next, Chapter 5 is dedicated to the evaluation and selection of features from a classification perspective for a specified application. Domain-specific feature engineering is a common practice in gesture recognition [10] and common methods for feature selection may actually have negative impacts on system performance. A modification of a method from literature is presented and validated both in presegmented offline contexts and for online continuous classification.

1.2.5 Application to alternative architecture

Finally, Chapter 6 will (qualitatively) replicate these results on a gesture recognition platform based around direct physiological measurement rather than optical detection of hand pose. Previous chapters present results for an optical-sensor-based platform. The motivation for doing this is is to validate the generalizability of the systematic design process.

Chapter 2
LITERATURE REVIEW

How are communication objects selected for semi-structured interactions?

2.1 Gesture communication

Regardless of how gestures for communication are measured or recognized, there is a finite set of control/communication gestures that are not being performed continually. Unless a preselected set of gestures is dictated by outside requirements, the communicative hand poses or motions must be chosen by the system designer. The question of how to segment out meaningful gestures during online recognition is an area of active research [53, 37], most of which focuses on the classifier as opposed to control gesture selection.

There exists some work dealing with systematic gesture selection with a loose goal of improving robustness, but they are limited in scope. The process developed by Shimada et al. [49] uses some measure of inter-gesture distance or overall classification accuracy as one of five factors informing gesture selection, but the focus of their method is on overall user experience and depends heavily on individual realizations of gestures, necessitated by their sample sizes. It also does not attempt to characterize the underlying distributions of the gesture classes, in order to have some predictive power of their future performance. In describing their methods, Kim et al. [25] states that "gestures should be equally easy to perform and form patterns in the EMG signal which are as discriminative as possible" but then goes on to list the selected gestures without stating how they satisfy these parameters. Similarly, Chen and Kim [6] acknowledge the relevance of picking distinguishable gestures but handle the selection of such on a manual, case-by-case basis.

The most common explicit selection method is participant preference [22, 49], including sourcing the base gesture set itself from study participants. Often experiments will use the base gesture sets of other work [28, 36, 53, 30], particularly when prior datasets are made available. Often when researchers develop their own datasets, the gesture selection process for them is not discussed [16, 44, 7, 12, 40, 31, 51, 50]. It is also worth noting that some gesture recognition work is acting on a pre-fixed set of gestures, such as a sign language [27, 34, 42, 35]. This lack of approaches to systematically select gestures to design a more robust interaction motivates the current work on separability-based gesture selection.

2.2 Verbal communication

There is a relatively large body of work dealing in systematic selection of verbal communication. The older canon deals with communication between humans, perhaps in situations where the signal is degraded. [38] and [48] are older works which experimentally evaluate the intelligibility of the international phonetic military alphabets, but they do not make alternate suggestions for their composition.

In a similar vein, Miller and Nicely [33] collected a large corpus of data on human comprehension of sixteen common English phonemes in the presence of frequency distortions and random masking noise. From this data, Mermelstein [32] informs his choice of distance metrics on speech signals. Similarly, Juola [21] uses the work of Miller and Nicely as the basis selection of the for the PGPfone alphabet, an improved method for exchanging binary communication securely over noisy telephone lines. These works, however, are limited in the sense that they are primarily based on human perception rather than machine recognizability or a validated proxy measure for the same.

There are distance metrics on short-time spectral samples; Sakoe and Chiba [47], in their work on speech recognition, evaluate several distance metrics on frequency spectra of time-aligned signals. Their work is largely concerned with single representative examples,

however, not class characterization or separability. Calculating the distance of multi-featured classes of speech (such as the spectrogram, or any other way to represent sound) then falls under methods described in [13], as they can be productively thought of as multivariate probability distributions. There is also the related concept of instance selection [3], wherein the designer preprocesses classes to find the best representative example of each to use in an improved K-nearest-neighbor classifier. This commonly uses a notion of pairwise distance, but between individual performances rather than full classes.

In summary, there has been some relevant selection work in the area of verbal communication, but it is either ultimately based on human confuseability or not selecting between classes, but within them.

2.3 Miscellaneous related selection

Systematic processes for selection of discrete communication pieces are present in related fields. For example, orthographic distinctiveness was developed as a feature of written word appearance in [54], where increased distinctiveness was found to improve human visual word recognition in [55]. Similarly, distinct features in images has proven a rich area of interest in the field of computer vision, e.g., in [29].

There is a history of selecting gestures for expressiveness of humanoid robotic platforms or animations [17, 45, 4] However, this is primarily a human factors challenge that is optimizing for human perception, not robotic. The success of these metrics developed to improve identification and communication in related fields indicates the potential for applicability to gesture selection. However, since none of these selection methods are suited to this specific class of communication selection problem, a new selection process must be devised, and an appropriate inter-class distance measure selected.

Chapter 3

GESTURE SELECTION BY SEPARABILITY

3.1 Introduction

3.1.1 Robotic collaboration in limited-access manufacturing

Despite the many benefits of automation for manufacturing applications, airplane manufacture still involves a significant number of human workers. This is due to complexity of the assembly environment, variability of process, and difficulty of access making full automation prohibitively time-consuming and expensive. Work in limited-access areas is of particular interest as a target for robotic collaboration with human mechanics. During manufacturing operations in limited-access areas, often mechanics can reach into the working space with a single hand but cannot see it due to the tight space configuration. Typically, airframes have small access holes and these physical-access constraints apply to most work inside, such as fastening, sealing, and inspection. Mechanics are usually forced to work by touch and experience, or to partially peek inside in awkward postures. Lack of easy access can lead to errors requiring later correction, subsequent re-entry into the limited-access area, or even costly repairs.

An appropriately placed camera can provide significant benefit compared to baseline methods (proprioception and experience) and current vision aids (primarily mirrors). In our experience, the efficacy of camera assistance is highly sensitive to viewing orientation, point of view, and potential for occlusion. Using hand following, a robotic assistant could dynamically track the mechanic's working hand with a camera and display the work area local to the hand from a useful perspective (i.e., robot pose).

The robot then would then also be in a position to assist with other facets of the task; in this case, to provide a running record on the results of an inspection task. Consider a situation in which the mechanic's hand and arm is in the limited-access area and the rest of their body is outside the space. The mechanic is performing work with one hand and observing the robotic camera view of the work being performed on a screen outside the limited-access area. As portions of the inspection task are completed, the mechanic may signal to the robotic assistant instead of withdrawing from the space to record results manually.

3.1.2 Gesture control vs. other methods of control

Gesture is not the only possible control interface for a robotic system to receive information from a human. In order for a variety of situations and populations to be served effectively, gesture is just one of many that should be considered [24, 56].

Other methods of control might be more reliable, but they are not well suited to the problem of a robotic manufacturing assistant. For example, setting tools down to manually reposition the robot or withdrawing from the space to log work completion would significantly interrupt the flow of work. Voice commands are challenging in a noisy factory environment and would require additional sensors. The mechanic's other hand is usually occupied with additional tools, so they would be unable to use a touch-based interface at will. Because of these limitations specific to manufacturing in limited-access environments, gesture control is uniquely suited to this application.

3.1.3 The unimodal input problem for gesture control

For the robot collaboration task described above, the robot's information about the intent of the mechanic comes from observing the mechanic's working hand that is in the limited-access space with the robot. This is the unimodal input problem: with only a single input

(the mechanic's hand, also used for work completion), how can the robot reliably tell when a control action is being performed? Compounding this issue is that the movements required for work completion (work gestures) are not simply static or neutral poses. They are complex directed motion sequences with similarities to some commonly used control gestures, such as a grabbing motion. Similarity is a problem, because currently-used sensors also have practical limitations on accuracy and stability problems with resolving fine motions ([52]).

The input to the robotic platform is also continual. Said another way, hands can't be turned off in the same way as other interfaces like commands or haptic interfaces. This is a problem for robotic gesture control because humans can intuit non-communication with another human through social cues or additional input streams such as gaze direction or posture, but robots that only track a subject's hands usually do not have access to these methods. The continual input problem combined with the complexity of work gestures means that for the robot to be able to tell when a control action is desired, the control gestures should be carefully chosen to be maximally separable from the work gestures.

3.2 Methods

3.2.1 Problem formulation

Given a set of "work gestures" $W \subset \mathbb{G}$ with members $W = \{w_1, w_2, \dots\}$ based on the work context, a finite number I of control gestures should be chosen from $C \subset \mathbb{G}$ for maximum separability from W. The control gesture selection space $C \subset \mathbb{G}$ in a practical application is limited to the space which is easily performable and for which data exists to use in recognition. The set of candidate control gestures C in this work is taken to be the 28 gestures in the SHREC dataset ([10]). The sets W and C are not necessarily exclusive. $\mathbb{S}$ is then the set of all possible combinations of I control gestures from C. An individual candidate set of control gestures is defined as $\{s \in \mathbb{S}\} = \{c_{s,1}, \dots, c_{s,I}\}$, where $c_{s,i} \in C$. The challenge, given the problem of choosing maximally distinct recording signals, is to choose a set of control

gestures s such that separation between both i) each member in the control set $c_{s,i} \in s$ from each other, and ii) between members of the control set $c_{s,i} \in s$ and members of the work set $w \in W$ is maximized.

3.2.2 Distance metric

The classifier used is the Support Vector Machine (SVM) classifier, in which training examples are mapped into a high-dimensional feature space and the hyperplane that best separates them is found [8]. The pairwise distance metric used for this work is a modified version of the margin of each pairwise SVM classifier. Further details on this distance metric's calculation and suitability can be found in Chapter 4. Notationally, the pairwise distance between two gestures g_1, g_2 is written $||g_1 - g_2||_{SV}$ or $||g_2 - g_1||_{SV}$ (modified SVM margin distance is symmetric). Additionally, for a single candidate control gesture c_i and a given work gesture set $W = \{w_1, \ldots, w_K\}$:

$$\left\|c_i - W\right\|_{SV} = \min_k \left\|c_i - w_k\right\|_{SV}. \tag{3.1}$$

This is because the separability measure developed is primarily concerned with mitigation of worst-case scenarios.

3.2.3 Separability and Selection Algorithm

Given a possible control set $s \in \mathbb{S}$ and a work gesture set $W = \{w_1, \ldots, w_K\}$, the separability Ψ of $s \cup W$ is calculated like so:

$$\Psi(s) = \min_{i,j} \left\{ \left\|c_{s,i} - c_{s,j}\right\|_{SV} \cdot \left\|c_{s,i} - W\right\|_{SV} \cdot \left\|c_{s,j} - W\right\|_{SV} \right\}. \tag{3.2}$$

Said another way, $\Psi(s)$ is the product distance of the nearest pair of control gestures $\{c_{s,i}, c_{s,j}\} \in s$ to each other and to W. This product distance is visualized in Fig. 3.1.

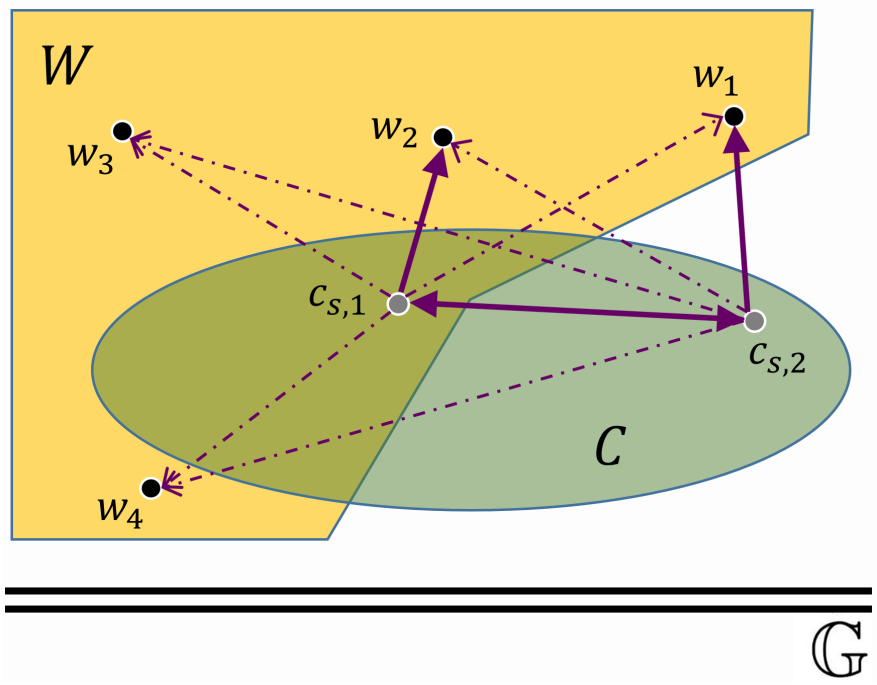


Figure 3.1: Visual explanation of distances that go into the calculation of separability $\Psi(s = \{c_{s,1}, c_{s,2}\})$ for an example task and one possible sample s. The distances between each member of s and all other gestures within $s \cup W$ are measured via the modified SVM margin measure detailed in Chapter 4. The three distances that go into the actual calculation of $\Psi(s)$ (Eqn. 3.2) are shown as solid arrows, while distances that are measured but do not go into the final calculation are shown as dotted arrows.

The separability metric $\Psi(s)$ represents the distance between different control gesture members $c_{s,i} \neq c_{s,j}$ as well as the distance between control gestures and work gestures. The optimal control gesture set s^* is that which maximizes this distance:

$$s^* = \arg\max_s \Psi(s). \tag{3.3}$$

Assignment of control operations for the general case to each selected gesture $g_i \in s^*$ is left to participant preference. An antagonistic control gesture set s° can also be chosen according to

$$s^\circ = \arg\min_s \Psi(s) \tag{3.4}$$

for experimental comparison with s^*, for which a lower classification performance than s^* is predicted. This increase in performance of s^* relative to s° will be referred to as "improvement". As C is drawn from an externally-sourced list of candidate gestures, all possible choices might reasonably be chosen for an online HRI application.

3.3 Structure of Ψ

Though it may seem intuitive for the Ψ metric to be an average of the relevant distances, in practice the average or sum does not provide differentiation for particular but reasonable cases. Here I will detail one such failure case and use it to justify my choice of a distance-product structure for Ψ.

3.3.1 Example setup

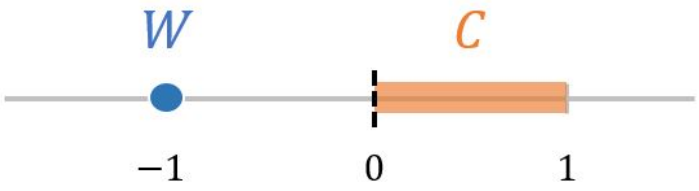
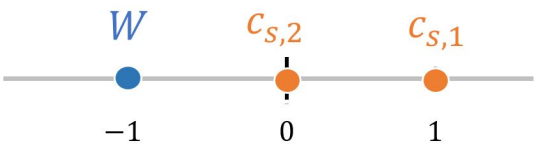
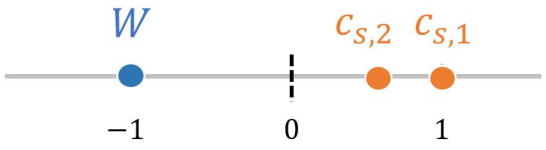


Figure 3.2: Visualization of the Ψ example structure.

In the simplified 1-D example visualized in Fig. 3.2, gesture classes are represented by points and the pairwise distance measure is how far apart they are on the line. There is a single work gesture at $W = -1$ and the selection space for $\mathbb{S}$ is the continuous space between 0 and 1. In this example $I = 2$, meaning that each selection s has two members $c_{s,1}$ and $c_{s,2}$. Note that for this case there are no irrelevant distances. Some specific example cases for comparison that are drawn from this overall framing are shown in Fig. 3.3.



(a) Case A: $s_A = [1,0]$. Intuitively, this would be the most separable combination of two control gestures possible from the setup in Fig. 3.2.



(b) Case B: $s_B = [1,0.7]$. This should be less separable than Case A but still have some degree of separability.

(c) Case C: $s_C = [1,0.95]$. This set should be functionally inseparable, since the two control members are nearly on top of each other relative to the candidate selection space.

Figure 3.3: Specific example cases for comparison of Ψ structures.

3.3.2 Average distance and shortcomings

If the set separability is based on the average distance, then the Ψ calculation for this case becomes

$$\Psi(s) = \frac{1}{3}\left(\left|c_{s,1} - W\right| + \left|c_{s,2} - W\right| + \left|c_{s,1} - c_{s,2}\right|\right).$$

Let the first selection $c_{s,1} = 1$, as in the example cases (Fig. 3.3). Then the previous reduces to

$$\Psi(s)(c_{s,2}) = \frac{1}{3}\left(\left|1 - (-1)\right| + \left|c_{s,2} - (-1)\right| + \left|1 - c_{s,2}\right|\right) = \frac{1}{3}\left(2 + 1 + 1 + c_{s,2} - c_{s,2}\right) = \frac{4}{3}.$$

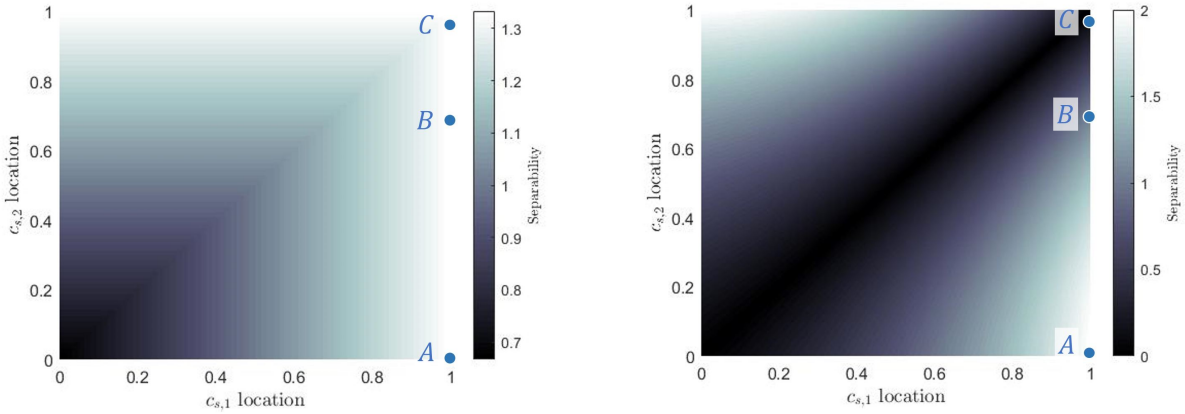
The dependence on $c_{s,2}$ has been completely eliminated. It is intuitive to see that the most separable position for $c_{s,2}$ is 0 (Case A), but in the summation-form equation any losses from moving to the right (closer to $c_{s,1}$) are counterbalanced by the greater distance to W. Thus, the reported separability of cases A, B, and C are all equal despite their obvious differences. This is confirmed in simulation of the above problem (Fig. 3.4 (a)) where the three cases are marked on a heatmap of all possible combinations and their values are all equal.

3.3.3 Product-form equation

By comparison, the problem structure used in [18] does provide better differentiation. The simulation of the same (Fig. 3.4 (b)) highlights that unique optimal positions are chosen for s given this scenario, and that they are in the spots that would be intuitively chosen given this scenario (0 and 1, aka Case A).

3.4 Implementation

There are two types of processes implemented for this work: an offline classification process, as in other work applied to the SHREC database, and an online classification process, where a rolling classifier is applied to continuous timeseries data. Both processes use a time history



(a) Illustration of the problems with using a summation-form separability evaluations for simple 1-D examples A, B, and C in Fig. 3.3

(b) Improvement of the product-form separability on cases A, B, and C.

Figure 3.4: Direct comparison of the average vs. product separability form for example cases A, B and C.

of raw hand measurements as their input and both use a Support Vector Machine classifier, but the acquisition and calculation of these present some nuance as noted where relevant.

3.4.1 Experimental Design

The task under consideration is inspection of fastener installation in a limited-access area, for which the MASCOT-RVA2 (shown in Fig. 3.5) provides visual and record-keeping assistance. The lab experiment version of this experimental setup is shown in Fig. 3.6. The participant checks the installation of each fastener, and after each grouping of four gestures to the robot to check the group off as all done or that it contains needed rework. Simultaneously, the MASCOT-RVA2 tracks the participant's hand to provide visual aid. This combination of functions has the potential to substantially reduce the mental load and cycle time of inspectors. This is a substantially simplified version of the actual inspection task. The "bad" examples in this case are collars that have been insufficiently swaged onto the fasteners

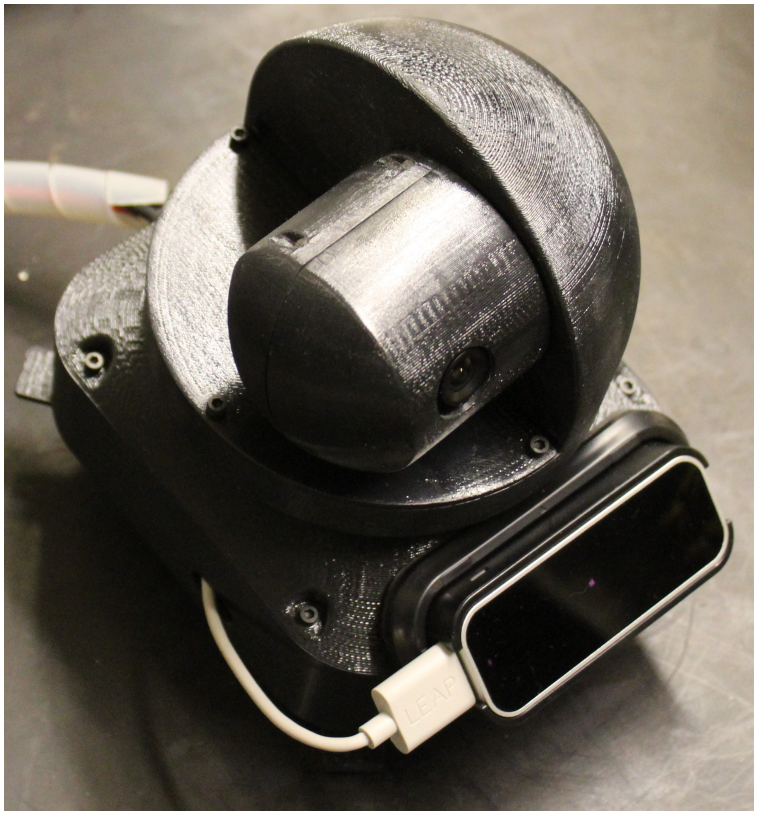


Figure 3.5: Limited-autonomy vision aid/assistive robot, referred to as the MASCOT-RVA2 or simply MASCOT or RVA2. Leap motion hand sensor for interaction and tracking input is shown at front.

before the pintail broke off, and so they are loose and spin freely. The distribution of correctly and incorrectly installed collars is known to a human observer, who manually sets a flag to indicate ground truth for a control gesture being performed by the participant.

The participant inspects the space five times during the collection of work data. There are eight clusters of correctly and incorrectly installed collars each (as shown in Fig. 3.6). At this stage, the camera view is not provided to participants and the data being recorded is unlabeled. No control gestures are performed. In order to choose the control gestures systematically for this experiment, the space of expected non-communicative hand motions

Figure 3.6: Experimental setup for inspection task with robotic assistant. Collars (yellow) are inspected for correct installation in their groupings of four.

must be quantified.

3.4.2 Quantification of work gesture space

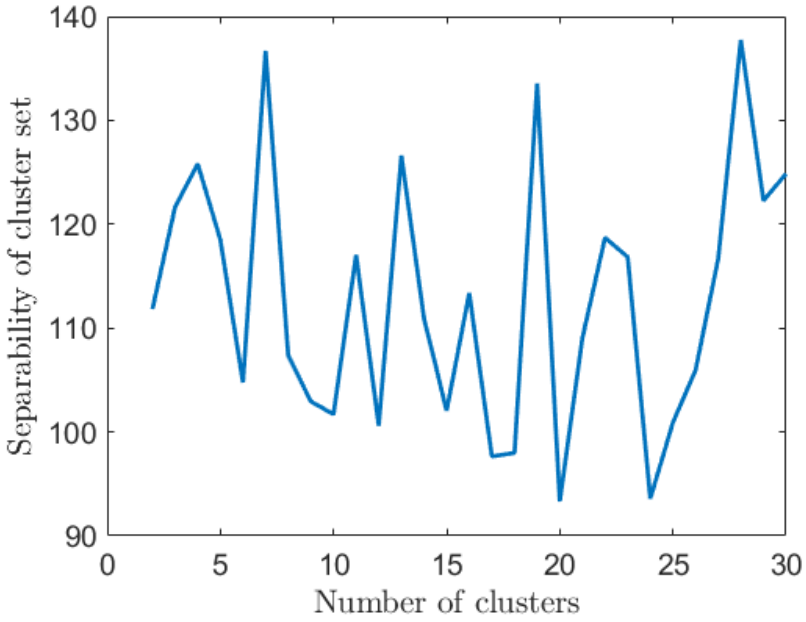


Figure 3.7: Separability evaluation of possible values of K for K-means clustering algorithm applied to randomly segmented work data. For a gesture set consisting exclusively of work gestures, separability of the set is the minimum pairwise distance. $K = 28$ is more separable than $K = 7$ by this analysis, but has several clusters with only one member. This is challenging for implementation, and so $K = 7$ was chosen to go forward. The full pipeline with $K = 30$ to the offline test accuracy vs. Ψ was calculated, though, and results were qualitatively similar to a lower number of clusters.

"Work" gestures will depend on the task performed. To generate representations of the atomic motions for this task, participants were recorded inspecting the entire workspace (in any order of their preference) several times. Inspection record at this stage was collected verbally by the experimenter, which may not be practical in a factory setting. The resulting sequences are randomly sampled in windows of time representative of the time windows of

the SHREC dataset. These shorter clips are then clustered by K-means clustering, using K values up to 30 for comparison (see Fig. 3.7). Several values of K were evaluated to the final results, and qualitatively similar results were achieved for all. $K = 7$ was the most separable set of clusters where all clusters had at least two members, and so was chosen for the implementation. The automatically clustered work gestures can be approximately described thus:

W1: Inspection with hand mostly closed

W2: Inspection with index finger only extended

W3: Inspection with all fingertips

W4: Inspection with index and middle fingers extended

W5: Inspection with open hand, slow motion

W6: Rapid traversal of space, slack hand

W7: Quick transitions between open and closed hand, most common silent sensor failure mode seems to be captured here.

3.4.3 Representation

The hand measurement is reported at every timestep as the Cartesian positions of every joint in the hand. A "gesture" unit $g \in \mathbb{G}$ is a time history of some length of snapshots of raw joint positions. For the SHREC database, these gesture units are pre-segmented into 28 labeled classes of gesture. Conversely, in the online experimental data, the gesture unit that is acted on and classified at time k is the snapshot at time k concatenated with the $N - 1$ previous snapshots for a "buffer length" of N.

The raw measurements are in the coordinate frame of the sensor, and it is desired to represent the hand as a set of spatially-invariant features in order to recognize the same gesture in varying contexts. The positions of the joints relative to the wrist at each timestep are represented using the *Shape of Connected Joints* (SoCJ) feature, sourced from [9]. The gross motion of the hand through space is represented with the velocity of the wrist in the current direction of the thumb at each timestep. The velocity and SoCJ is calculated for the length of a gesture, resampled to be a consistent number of timesteps, and restacked into a vector.

This is a much larger number of features than the original Cartesian frame, but many of the features are redundant or irrelevant. Principal component analysis (PCA) ([26]) is employed to project the full feature space onto a lower-dimensional subspace that preserves the majority of the information and is more suited to rapid computation and available memory. Selection of orthogonalized features according to separability rather than maximized variance is described in Chapter 5.

3.4.4 Classification

For classification, the pairwise one-vs-one SVM classifiers trained in the distance calculation are used with a relative majority voting rule [19]. For each one-vs-one classifier included in the ensemble classification scheme, one is selected and the most commonly occurring gesture label from the component pairwise classifier outputs is chosen as the final classified label for that gesture.

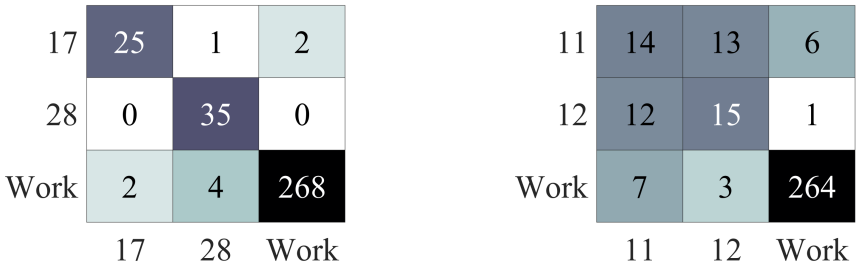
A control gesture will "trigger" in the online process when the content of the recent history buffer of timesteps is recognized as that control gesture. It is evaluated as correct if it is sufficiently close temporally to the ground truth flag that was set manually by a human observer and of the correct class. For a short time after the process triggers, no other gestures can be triggered.

A "false positive" is either when a control gesture is identified incorrectly or when there was no communication intended, and a "false negative" is when a gesture occurs but the classifier does not recognize it. A "true positive" is when a control gesture is correctly triggered.

3.5 Results

Results are now presented for the following claims: (i) proposed separability predicts performance improvement and decreased variability in offline classification; (ii) separability improves online classification performance with multiple subjects.

3.5.1 Separability improves classification



(a) s^* (b) s°

Figure 3.8: Confusion matrices for s^* and s° in offline classification. Note that for this application correct identification between work subtasks is not relevant, and thus all W test gesture samples (separate from training data) have been grouped together. The number of entries on the diagonal divided by the total number of entries is the "Test Accuracy" for an individual control set s. Total iteration number for a complete control gesture class varies somewhat, but is generally around 100.

The main result of this section is that separability of a gesture set is positively correlated with test accuracy and robustness to individual performance of gestures.

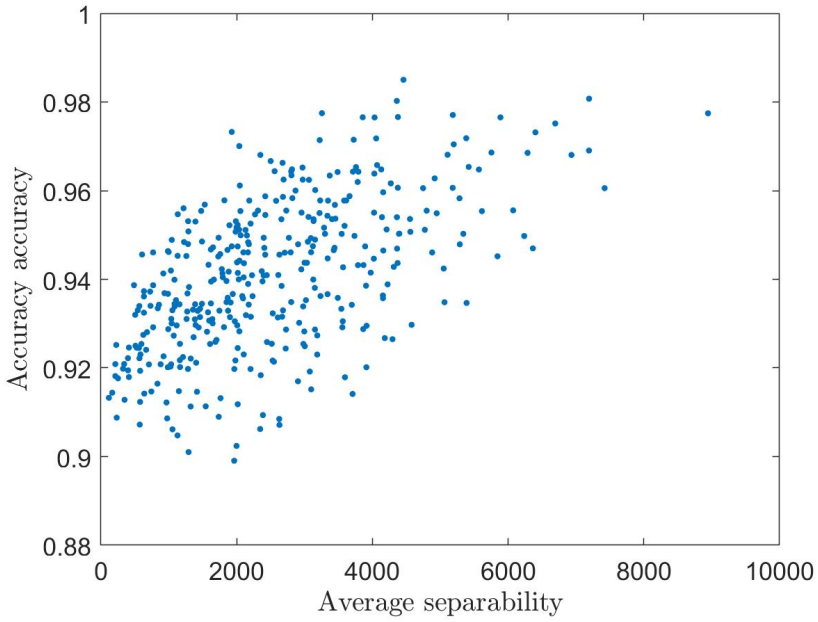


Figure 3.9: Full sample space $\mathbb{S}$ evaluation of each set s's separability ($\Psi(s)$) plotted against its corresponding test accuracy. This selection shows that even for offline classification, maximizing the Ψ metric will predict contextually improved gesture recognition, as there is a positive correlation between Ψ and test accuracy (Pearson correlation coefficient $R = 0.59$).

First, to validate the feature set and classification method, the full set of 28 SHREC control gestures is classified for comparison with existing work. The result is that 71% of presegmented test examples were correctly recognized (i.e. 71% test accuracy), which is comparable to previous work that ranges between 62% and 82% ([10, 40, 41, 11]). A simplified version of the highest-accuracy pipeline in [10] is used in this case study. Test accuracy for offline evaluation of a candidate $s \cup W$ is visualized in confusion matrices, as in Fig. 3.8 for s^* and s°. Each row in a confusion matrix represents the instances of a predicted class while each column represents the instances of an actual class, and is designed to enable visualization of what the common failure modes are of a classifier. The numbers at each index in Fig. 3.8 represent the number of test iterations that truly belong to the

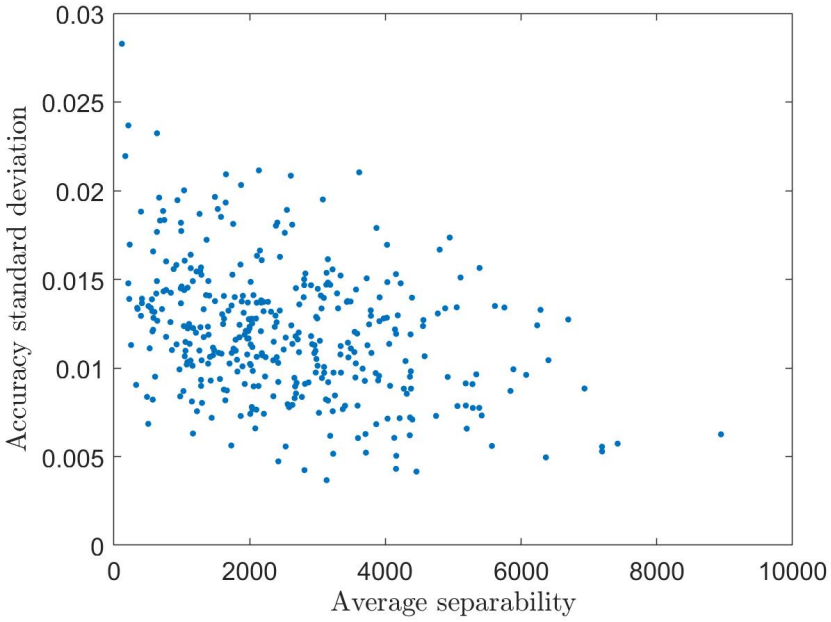


Figure 3.10: Additionally, separability not only positively correlates with test accuracy, it negatively correlates with sensitivity to subject performance (Pearson correlation coefficient $R = -0.35$).

class of their column, but were identified as the class of their row. Thus, the numbers on the diagonal are correctly identified test instances, and all entries off the diagonal are test instances that were misclassified.

Next, the full range of separability Ψ is tested for all possible $s \in \mathbb{S}$ using the test set contained in the SHREC database. $\mathbb{S}$ is constructed using all combinations of 2 from the SHREC dataset, and appending the seven members of W to each set so each $s \cup W$ has 9 members. Test accuracy vs. separability for all combinations is shown in Fig. 3.9. The optimal s^* that maximizes Ψ from (3.3) and antagonistic s° that minimizes Ψ from (3.4) can be seen as image sequences and described in Fig. 3.11. These two combinations are the ones that will be taken forward into online testing.

The main result from Figures 3.9 and 3.10 is that even in the presence of variability

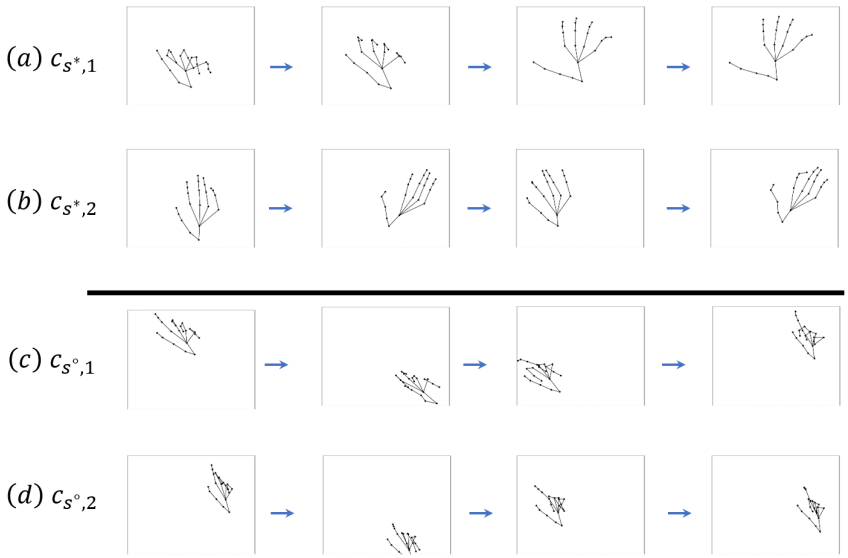


Figure 3.11: Image sequences of members of s^* and s°, which are described as follows: (a) $c_{s^*,1}$ A grabbing motion with only the thumb and forefinger (b) $c_{s^*,2}$ A continuous swipe in the shape of a +, performed with a slack hand with fingers extended (c) $c_{s^\circ,1}$ A pinching motion with only the thumb and forefinger (d) $c_{s^\circ,2}$ A planar shake back and forth of a hand from the wrist with only the thumb and forefinger extended.

in classifier performance for an arbitrary choice of s, separability is positively correlated with test accuracy and robustness to variability in individual participants' performance of a gesture. Choosing an arbitrary control gesture set which does not maximize Ψ does not necessarily result in poor performance, but choosing a maximally-separable s^* is shown here to be more consistently recognizable with a variety of subject performances.

3.5.2 Performance Evaluation

Evaluating the performance of online classification procedures is a problem with different constraints than offline (i.e. temporally presegmented) classification. Two of the most commonly used metrics for online classification of this type are precision and recall ([43]). Briefly, precision is the percentage of returned instances that are correct, and recall is the percentage

of true total instances correctly returned. Using the error types described in Section 3.4.4, they are defined as

$$\text{Precision} = \Pr(s) = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \tag{3.5}$$

and

$$\text{Recall} = \text{Rec}(s) = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}. \tag{3.6}$$

These metrics are often tradeoffs; a more sensitive classifier might have higher recall but lower precision, and a less sensitive classifier might have lower recall but higher precision. These metrics were chosen for direct comparability to similar recent results in continual motion recognition.

Because precision and accuracy measures different types of error, the specifics of the application will determine what measure should be given more weight. For this application, precision should be favored if possible because it is easier to perform a control gesture again than stop the robot from executing an undesired accidental commanded action.

To measure a combination of precision and recall, F-measures are commonly used, defined as

$$F_\alpha\text{-measure} = \frac{(1+\alpha^2) \cdot \text{recall} \cdot \text{precision}}{\alpha^2 \text{recall} + \text{precision}} \tag{3.7}$$

When $\alpha = 1$, precision and recall are weighted equally and the F1 score is the harmonic mean of the two. Otherwise, α is adjusted to weight the more relevant measure appropriately. For this experiment the F1 score is used as all measures are of interest. The expectation is that at minimum the F1 scores will definitively increase between s^* and s° in online subject testing, and potentially all measures will improve.

3.5.3 Online testing with multiple subjects

Once the control-gesture sets have been selected for online testing through maximization and minimization of $\Psi(s)$, the overall performance of the sets is evaluated according to the

Table 3.1: Subject breakdown of performance metrics. There was improvement for all subjects across all measures between s^* and s°, with an average performance improvement of approximately 400%.

Subject	Pr(s^*)	Pr(s°)	Rec(s^*)	Rec(s°)	F1 (s^*)	F1 (s°)
1	0.36	0.03	0.35	0.06	0.35	0.04
2	0.51	0.15	0.48	0.11	0.49	0.12
3	0.35	0.17	0.48	0.23	0.41	0.20
4	0.50	0.07	0.38	0.08	0.43	0.07
5	0.44	0.08	0.55	0.11	0.49	0.09
6	0.51	0.01	0.53	0.02	0.52	0.01
7	0.50	0.11	0.39	0.21	0.43	0.14
8	0.37	0.08	0.42	0.12	0.39	0.09
9	0.37	0.11	0.20	0.09	0.25	0.09
10	0.57	0.10	0.46	0.08	0.50	0.09
Avg.	.45	.09	.42	.11	.43	.10

experiment as detailed in §3.4.1. Each subject performs 32 iterations of each control gesture at intervals while completing the work task, for a total of 128 control gesture instances per subject.

As Table. 3.1 shows, maximizing the separability Ψ results in performance improvement between s^* and s° across all performance measures in online testing. All measures on average were nearly 4 times higher for s^* compared to s° and there was consistent improvement for all subjects, despite individual variations in overall recognizability. In summary, the results show that the separability-based selection improves performance first in offline classification, and then the online classification performance.

3.6 Conclusion

In both online and offline contexts, systematic selection of gestures is shown to have performance benefits. A correlation between the proposed separability metric and test accuracy of gesture sets is found ($R = 0.59$), as well as a negative correlation ($R = -0.35$) between separability and sensitivity of performance to individuals' performances. Results are also presented for a comparative online classification implementation on a hand-following robot of the most- and least-separable gesture sets, where the most-separable gesture set performs an average of 4 times better across all metrics.

Chapter 4

DISTANCE MEASURES

In order to calculate separability as the problem has been framed, a meaningful measure of pairwise distance between gesture classes is necessary. A "gesture class" is any meaningful group of performed motions, either from human-assigned perceptual meaning or an automatic clustering algorithm. There are many possible candidates for such a distance measure between classes, both new and from literature, each with advantages and disadvantages. In prior work [18] the margin of the SVM classifier was used, but that metric has some shortcomings. Desirable properties of a separability-capturing distance measure between gesture classes would include, in decreasing order of importance:

P1: Larger distance corresponds to lower likelihood of misclassification,

P2: the distance measure is applicable to many different data distributions, and

P3: outliers or low population sizes do not significantly affect the outcomes for the same underlying distributions of gesture realizations.

In this section, a number of candidate measures for the distance between gesture classes will be introduced and discussed. First, the measure that is determined to fulfill these properties best is presented, then the measures that were considered and discarded.

4.1 Ultimately selected method: modified SVM margin

4.1.1 SVM Background - Conventional Solution

In support vector machines, there are two classes of data labeled as +1 or -1 for which the designer wishes to be able to classify future examples. If the data classes are linearly separable, then the solution found is referred to as a *hard-margin* case. Else, the problem is a *soft-margin* case, where the margin is the boundary region between the classes.

(a) Example of a data realization that has a hard-margin solution.

(b) This data is not linearly separable, and so its solution is limited to the soft-margin case.

Figure 4.1: Visualization of the two types of SVM cases, hard-margin and soft-margin.

The solution methods for these are closely related, and here laid out sequentially. This subsection will conclude with concrete examples of the solution method for each case. This subsection is designed to introduce all relevant terminology and notation for the SVM-based distance measures.

Formal Problem: Hard-margin

In this formulation for data $\mathbf{x}$ and labels $\vec{y}$, the i'th data point is a pair (x_i, y_i) where x_i is the data, an n-dimensional vector, and y_i is the class label where $y_i \in [-1, 1]$. The designer wants to find the separating hyperplane $(a'x + b = 0)$ with the maximum margin. The margin is formally defined as the region where $y_i(a'x_i + b) < 1$ and is visually represented for the 1-dimensional case in Fig. 4.2. The width of margin (grey region) is $\frac{2}{||a||}$. In order to classify new data $x_{j>i}$, the classifier score is calculated $\hat{y}_j = (a'x_j + b)$ and the data is assigned

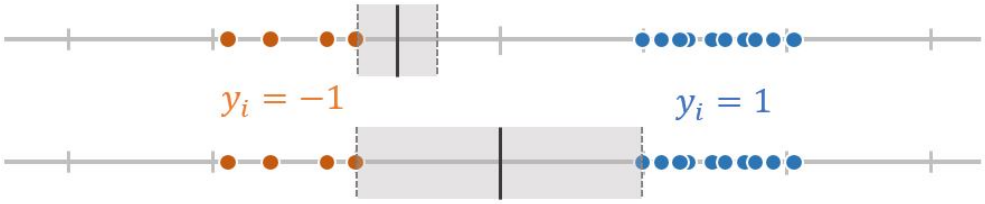


Figure 4.2: For this data realization, two possible hyperplanes that produce zero in-sample clasification error are shown, with the margin that satisfies the requirement $\min |(a'x_i+b)| = 1$ for that hyperplane (black line) shown as a grey box. Both solutions are equally feasible for the training data; however, the second solution has a much larger margin and it is intuitive to see that this solution also has a lower likelihood of misclassification.

to the class that matches the sign of $\hat{y}_j$. In order to find the best classification boundary (hyperplane with the widest margin and no in-sample error) the optimization problem to solve is

$$\begin{aligned} \underset{a\in\Re^k,\, b\in\Re}{\text{maximize}} \quad & \frac{1}{||a||_2} \\ \text{subject to} \quad & \min |(a'x_i + b)| = 1,\ i = 1, \ldots, n \end{aligned}$$

which is equivalent to

$$\begin{aligned} \underset{a\in\Re^k,\, b\in\Re}{\text{minimize}} \quad & \frac{1}{2}a'a \\ \text{subject to} \quad & y_i(a'x_i + b) \geq 1,\ i = 1, \ldots, n \end{aligned} \tag{4.1}$$

without loss of optimality. To solve this problem efficiently and with commonly available software for quadratic programming (such as quadprog in Matlab) the dual problem is constructed using Lagrangian duality and KKT conditions [14]:

$$L(a, b, \lambda) = \frac{1}{2}a'a - \sum_{i=1}^{n} \lambda_i (y_i(a'x_i + b) - 1) \tag{4.2}$$

which when minimized, will solve the original problem. To achieve this, the gradients w.r.t. a and b are

$$\nabla_a L = a - \sum_{i=1}^{n} \lambda_i y_i x_i \tag{4.3}$$

and

$$\frac{\delta L}{\delta b} = -\sum_{i=1}^{n} \lambda_i y_i. \tag{4.4}$$

Setting them both to zero and substituting back into 4.2 gives

$$L^*(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \lambda_i \lambda_j (x_i)' x_j \tag{4.5}$$

which then is a convex maximization problem

$$\begin{aligned} \underset{\lambda \in \Re^n}{\text{maximize}} \quad & L^*(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \lambda_i \lambda_j (x_i)' x_j \\ \text{subject to} \quad & \sum_{i=1}^{n} \lambda_i y_i = 0, \\ & \lambda_i \geq 0, \ i = 1, \ldots, n \end{aligned} \tag{4.6}$$

and is computer-solvable for realized data in the equivalent matrix-vector form

$$\begin{aligned} \min_{\lambda \in \Re^n} \quad & \frac{1}{2} \lambda' \begin{bmatrix} y_1 y_1 (x_1)' x_1 & y_1 y_2 (x_1)' x_2 & \ldots & y_1 y_n (x_1)' x_n \\ y_2 y_1 (x_2)' x_1 & y_2 y_2 (x_2)' x_2 & \ldots & y_2 y_n (x_2)' x_n \\ \vdots & \vdots & \ddots & \vdots \\ y_n y_1 (x_n)' x_1 & y_n y_2 (x_n)' x_2 & \ldots & y_n y_n (x_n)' x_n \end{bmatrix} \lambda \\ \text{subject to} \quad & y' \lambda = 0, \\ & \lambda_i \geq 0, \ i = 1, \ldots, n. \end{aligned} \tag{4.7}$$

The separating hyperplane is then recoverable from λ^* by recalling from the gradients that $a^* = \sum_{i=1}^{n} \lambda_i^* y_i x_i$. The optimal b^* is recovered by solving $y_i((a^*)' x_i + b) - 1 = 0$ using any support vector, where a support vector is a pair (x_i, y_i) whose optimal Lagrange multiplier $\lambda_i^* > 0$.

Example solution

A test case is shown in Fig. 4.3. In this example of a hard-margin problem, it is intuitive to see that the boundary is near 0 and the approximate scaling to force the score of the

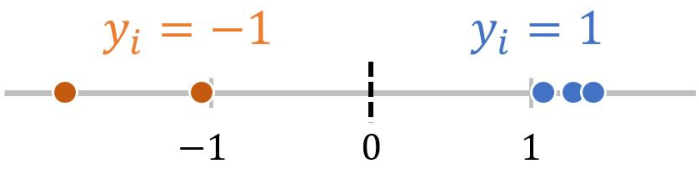


Figure 4.3: Visualization of the SVM numerical solution example.

support vectors to 1 is 1. In the appropriate terminology, $b^* \sim 0$ and $a^* \sim 1$. Fig. 4.3 is an approximate visualization; $\vec{x} = [-2.0, -1.01, 1.01, 1.3, 1.5]$ and $\vec{y} = [-1, -1, 1, 1, 1]$. The convex dual problem for this with problem data is then

$$\begin{aligned} \min_{\lambda \in \Re^n} \quad & \frac{1}{2}\lambda' \begin{bmatrix} 4 & 2.02 & 2.02 & 2.6 & 3 \\ 2.02 & 1.0201 & 1.0201 & 1.313 & 1.515 \\ 2.02 & 1.0201 & 1.0201 & 1.313 & 1.515 \\ 2.6 & 1.313 & 1.313 & 1.69 & 1.95 \\ 3 & 1.515 & 1.515 & 1.95 & 2.25 \end{bmatrix} \lambda \\ \text{subject to} \quad & y'\lambda = 0, \\ & \lambda_i \geq 0, \ i = 1, \ldots, n. \end{aligned}$$

Using any off-the-shelf quadratic programming solver such as `MATLAB`'s function `quadprog`, the recovered optimal values are $\vec{\lambda}^* = [0, 0.4901, 0.4901, 0, 0]$. This indicates that the support vectors are $x_2 = -1.01$ and $x_3 = 1.01$, as expected. Recall that $a^* = \sum_{i=1}^{n} \lambda_i^* y_i x_i = 0.4901 * (-1) * (-1.01) + 0.4901 * (1) * (1.01) = 0.99$ and $b^* = y_i((a^*)'x_i + b) - 1 = 0 \implies b^* = 1 * 0.9901 * 1.01 + b - 1 = 0 \implies b = 0$. The assigned scores $\hat{y}_i = a'x_i + b$ for the training set are $\vec{\hat{y}} = [-1.9802, -1.0000, 1.0000, 1.2871, 1.4851]$, confirming that the in-sample error rate for this example is zero.

Formal Problem: Soft-margin

For soft-margin SVM classification, the problem formulation in Eq. 4.1 changes in ways that are highlighted in red text. There is a new term ξ_i, which represents the margin violation of the i'th point. ξ_i is visualized in Fig. 4.4. The new constant C represents the gain on the margin violation penalty; if C is large, the margin violations are more important to the solution, and vice versa. In a hard-margin problem $\xi_i = 0 \; \forall i$ and so this term is negligible. The primal convex quadratic problem is (red terms are soft-margin changes):

$$\begin{aligned} \underset{a\in\Re^k,\, b\in\Re}{\text{minimize}} \quad & \frac{1}{2}a'a + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(a'x_i + b) \geq 1 - \xi_i \; i = 1, \ldots, n \\ & \xi_i \geq 0. \end{aligned} \tag{4.8}$$

Lagrangian and KKT conditions:

$$L(a, b, \lambda, \xi, \beta) = \frac{1}{2}a'a + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i(y_i(a'x_i + b) - 1 + \xi_i) - \sum_{i=1}^{n}\beta_i\xi_i \tag{4.9}$$

The gradients taken to maximize the Lagrangian are the same except that there is one additional gradient taken wrt ξ_i, the margin violation of the i'th point:

$$\nabla_\xi L = C - \lambda_i - \beta_i. \tag{4.10}$$

In the dual problem this gives λ_i the additional constraint that $\lambda_i \leq C$, but otherwise the dual problem is in fact the same as in the hard-margin case. This simplification, while convenient for model optimization, makes it challenging to reason about the effect of enmeshed vs separable data on the eventual margin width. For this reason the forthcoming examination of regional inversion will take place in the primal rather than dual formulation.

4.1.2 Examining margin width in soft-margin and hard-margin cases

The SVM algorithm maximizes the margin ($\frac{2}{||a||}$), the area between the bounding planes x_1, x_{-1} which satisfy $ax_{-1} + b = -1$ and $ax_1 + b = 1$. Violating the margin does not

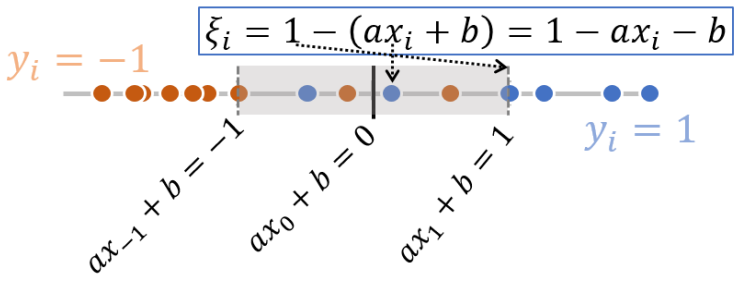


Figure 4.4: SVM terms visualization for the 1-D enmeshed case. This is not the optimal margin for this data, just an example a and b and their associated terms.

have to include being over the decision boundary; the point merely needs to be within the margin defined by a and b to be considered as violating the margin. In Fig. 4.4, any point within the grey region (or outside of it among the bulk of the opposite-colored class) are margin-violating points with an associated nonzero ξ_i. Their ξ_i term is the distance between the outer edge of the margin and the score of the margin-violating point, i.e. $\hat{y}_i = ax_i + b$. As illustrated in Fig. 2: for the i'th point (with $y_i = 1$, i.e. the class on the right), the margin violation distance is $\max(0, 1 - ax_i - b)$, where x_1 is the right margin edge and satisfies $x_1 = \frac{1-b}{a}$. Similarly, $x_{-1} = \frac{-1-b}{a}$. The margin violation for a point of class with label +1 is

$$\xi_i = 1 - \hat{y}_i = 1 - ax_i - b \tag{4.11}$$

and of class -1:

$$\xi_i = \hat{y}_i - (-1) = ax_i + b + 1, \tag{4.12}$$

illustrated visually in Fig. 4.4.

With a linear kernel, the decision boundary hyperplane can be written as the set of support vectors $\mathbf{x}$ that satisfies

$$\mathbf{a} \cdot \mathbf{x} + b = 0 \tag{4.13}$$

where $\mathbf{a}$ is the vector normal to the hyperplane and b is the hyperplane's bias from the origin.

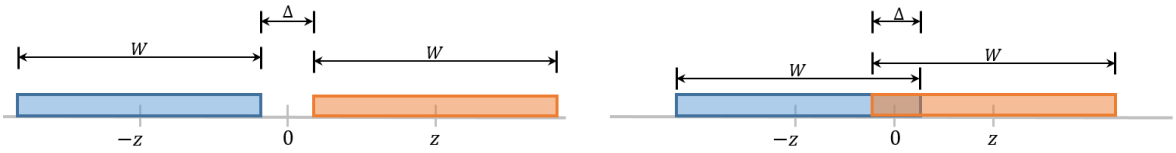


Figure 4.5: Two uniformly distributed classes of data with width W and whose centers are at $\pm z$. The distance between them is of width Δ (defined in Eqn. 4.14) as shown in figure. When the classes are not overlapping ($z > \frac{W}{2}$), as on the left, Δ is negative. The cases studied are for a constant W as $z \to 0$.

Two-region behavior of the margin

The margin of the classifier is commonly understood as the "width" of the margin between the classes, since the optimization is explicitly maximizing the margin and the intuitive leap to positive performance correlation holds where data is enmeshed. However, the change in optimization formulation between the hard-margin and soft-margin cases means that the margin is capturing qualitatively different information about the relevant data and empirical risk in the different regions. In the soft-margin case, it can be understood as the distance that best captures the margin violations. To illustrate this behavior, a case study is presented and illustrated in Fig. 4.5. For notational convenience, the distance between the right edge of the blue class and the left edge of the orange class is defined as Δ. In terms of z and W,

$$\Delta(z, W) = \left(-z + \frac{W}{2}\right) - \left(z - \frac{W}{2}\right) = -2z + W. \tag{4.14}$$

In the region with no overlap ($z < W/2$), Δ is always negative. The numerically-solved results for the margin edges for this case as Δ flips from negative to positive are shown in Fig. 4.6.

As shown there, the margin width has a highly structured relationship to empirical risk, but the behavior has two distinct regions. If this behavior can be shown to behave consistently, it is exploitable to achieve a consistent predictor of empirical risk. The remaineder of this section will be dedicated to establishing the consistency of this behavior and the method

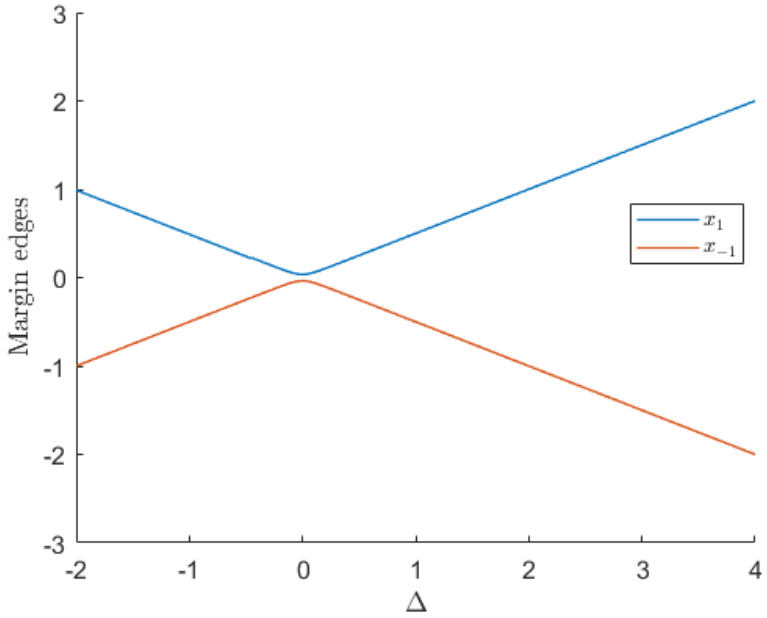


Figure 4.6: Infinitely-sampled approximations of the classification problem shown in Fig. 4.5, solved by conventional SVM. The margin width for a given Δ is the distance between x_1 and x_{-1}. As $\Delta \to 0$ the margin edges contract and then expand when when $\Delta > 0$, because the margin width is capturing different information about the model in this region.

of exploitation to develop a distance metric between gesture classes with a consistent relationship to empirical risk.

Separable (Hard-Margin) Case

At the outer edges of classes 1 and 2, the constraints of Eq. 4.1 reduce to a system of equations. They are:

$$\begin{aligned} y_1(a(x_{\Delta/2}) + b) &= 1 \\ a\frac{\Delta}{2} + b &= 1. \end{aligned} \tag{4.15}$$

at the right edge of the orange class and

$$\begin{aligned} y_{-1}(a(x_{-\Delta/2}) + b) &= 1 \\ a\frac{\Delta}{2} - b &= 1 \end{aligned} \tag{4.16}$$

at the left edge of the blue class. Summing (4.15) and (4.16),

$$a = \frac{2}{\Delta} \implies \text{margin} = \frac{2}{||a||} = ||\Delta||. \tag{4.17}$$

This agrees with the numerical solution presented in Fig. 4.6.

Enmeshed (Soft-Margin) Case

To discover structure between Δ and the optimal margin width where data is enmeshed, want to find an analytic solution to the soft-margin SVM optimization for where $z < W/2$.

The objective function of Eq. 4.9 in summation-form for the examined case where $z < W/2$ is

$$R = \frac{1}{2}a^2 + C\sum \xi_i = \frac{1}{2}a^2 + C\sum_{i=1} \xi_{\text{class }1} + C\sum_{i=1} \xi_{\text{class }-1} \tag{4.18}$$

and in an infinitely-sampled approximation with point density n/W, can be written with integrals

$$R \;=\; \frac{1}{2}a^2 + C\int_{x_{-1}}^{\Delta/2} \frac{n}{W}(ax + b + 1)\, dx \;+ C\int_{-\Delta/2}^{x_1} \frac{n}{W}(1 - ax - b)\, dx \tag{4.19}$$

Expanded out and changing variables entirely to x_1, x_{-1},

$$R \;= \frac{2}{(x_1 - x_{-1})^2} + \;\; \frac{Cn}{W}\left[\frac{1}{x_1 - x_{-1}}\left(x_1^2 + x_{-1}^2 + \frac{\Delta^2}{2}\right) + \Delta\right]. \tag{4.20}$$

Using symmetry to say that $x = x_1 = -x_{-1}$ ($\implies$ margin width $= 2x$) this reduces to

$$R = \frac{1}{2x^2} + \frac{Cn}{W}\left(\Delta + \frac{\Delta^2}{4x} + x\right) \tag{4.21}$$

whose derivative is

$$\frac{dR}{dx} = \frac{-1}{x^3} + \frac{Cn}{W}\left(\frac{-\Delta^2}{4x^2} + 1\right). \tag{4.22}$$

From this $x^* \approx \Delta/2 + \delta$, where $\delta \to 0$ as $\frac{Cn}{W}$ becomes large. This puts a lower threshold on the margin width at Δ, and for reasonably-sized datasets $\frac{Cn}{W}$ is large.

This analytic result agrees with the numerical result that as Δ increases (starting negative, indicating the classes are not enmeshed) the margin edges contract, and then expand with Δ when positive (shown in Fig. 4.6). This result is consistent with varying distribution types and higher-dimensional spaces.

4.1.3 Mitigation of regional inversion

While the inverted behavior on the margin width's predictive power is a challenge to using it to select gestures, the behavior is consistent, structured, and therefore exploitable. In order to achieve a consistent relationship between the final distance metric and empirical risk, the distance for any models that are soft-margin is the negative of its margin. Detection of this with sparse data is done via assessing if the in-sample error of the training data is greater than 1%. Shown in Fig. 4.7 is the SHREC data (pairwise only and without inclusion of work data) plot of separability vs. test accuracy with and without this regional inversion adjustment.

However, having a negative distance measure creates issues for the product form of the overall separability calculation. This is addressed by adding a constant positive offset O to the entire pairwise model population, where O is sufficiently large to ensure the smallest distance is 1.

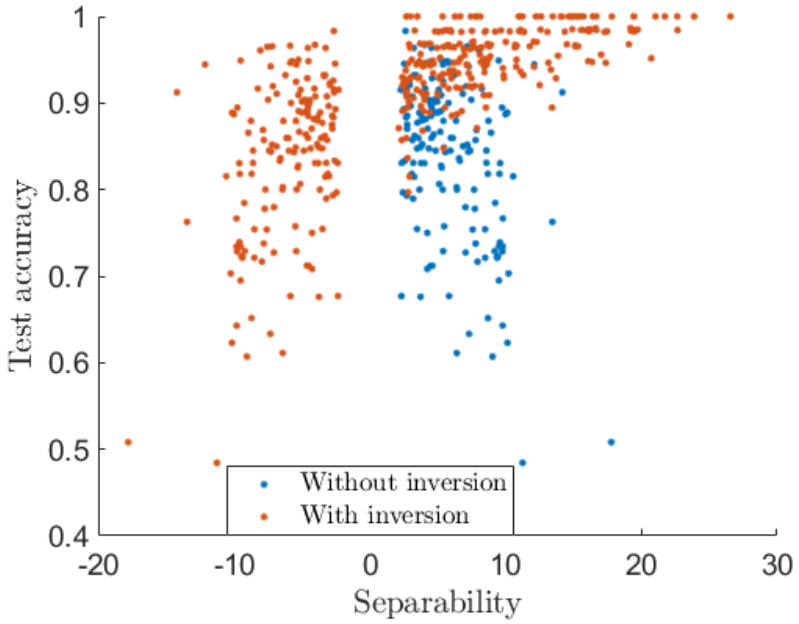


Figure 4.7: Every possible pair of control gestures from the SHREC dataset (no work data included), test accuracy vs. margin width with and without regional inversion adjustment applied.

4.2 Considered measures: metrics on probability distributions

4.2.1 f-divergence measures

Another potentially promising method for measuring distance between gesture classes is types of f-divergence, which measure differences between two probability distributions. Some commonly used examples in ML and data science include Kullback-Leibler (KL) divergence, χ^2-divergence, Hellinger distance, Total Variance distance, and α-divergence. Formally, the f-divergence between two distributions P and Q over a common space is

$$D_f(P \parallel Q) \equiv \int_\Omega f\left(\frac{dP}{dQ}\right) dQ$$

for particular choices of the function f. f must be convex and $f(1) = 0$. The named examples are differentiated by their f-functions and each serves a distinct purpose in their application domains. However, the problem with these is that they do not necessarily penalize the appropriate things. For example, if two normally-distributed gesture classes had exactly the same mean but very different variances, then their distances according to each of these metrics might be large. In some limited cases this is appropriate, but in a large selection population this is unlikely to truly provide an optimal answer.

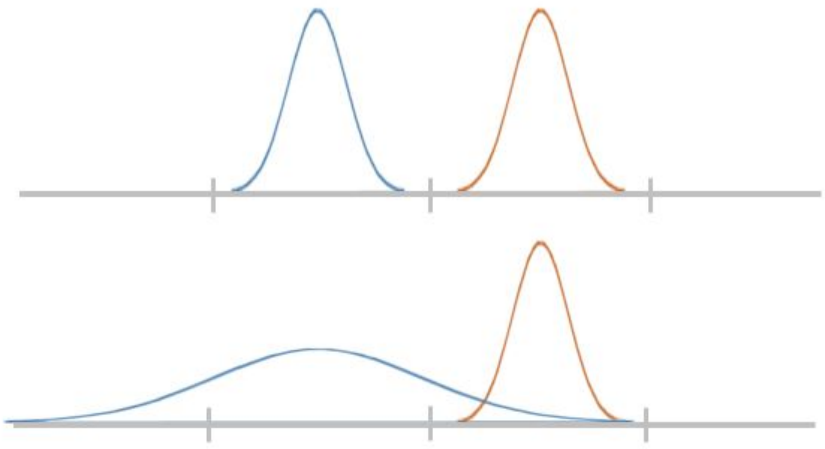


Figure 4.8: A comparison between these two cases should intuitively show that Case 1 (top) is more separable in the sense relevant to this chapter than Case 2 (bottom). However, because f-divergences penalize difference between distributions, the KL divergence is larger in Case 2 (see Eqn. 4.23 for details).

As an example, the function f for the KL divergence is $f(t) = t \log t$. Shown in Fig. 4.8 is two cases that are the same, except for one class having a much larger variance. Intuitively we would expect the second case to be less separable, but it in fact has a higher KL divergence. A common discretization of KL divergence between distributions P and Q is

$$\mathcal{D}_{KL}(p, q) = \sum_{i}^{N} p(x_i) \log \left(\frac{p(x_i)}{q(x_i)} \right). \tag{4.23}$$

When probability density functions for distributions $P_1 \sim \mathcal{N}(-0.5, 1)$, $Q_1 \sim \mathcal{N}(0.5, 1)$, $P_2 \sim \mathcal{N}(-0.5, 3)$, and $Q_2 \sim \mathcal{N}(0.5, 1)$ are generated, the calculations show that $\mathcal{D}_{KL}(p_1, q_1) = 0.5$ and $\mathcal{D}_{KL}(p_2, q_2) = 3.35$. Simple code to execute this check may be found in Appendix A. Though different f-functions may penalize differently, they all have this core problem of seeking divergence rather than separability.

4.2.2 Earth mover's distance and similar

There are other measures of distance in literature that do not necessarily fit into the f-divergence model, but may have interesting properties. One of the most common is the Earth Mover's Distance, whichb is closely realted to the transportation metric. For multivariate normal sample gestures g_a, g_b with expected values μ_a, μ_b and covariances Γ_a, Γ_b, the Earth Mover's distance $V(g_a, g_b)$ at time t is defined as

$$\mathcal{D}_{EMD}(g_a, g_b) = \left\|\mu_a - \mu_b\right\|_2^2 + \text{trace}\left(\Gamma_a + \Gamma_b - 2(\Gamma_b^{\frac{1}{2}}\Gamma_a\Gamma_b^{\frac{1}{2}})^{\frac{1}{2}}\right) \tag{4.24}$$

It does not have the properties of a true norm but has seen relevant applications, particularly in the field of computer vision. It has the same shortcomings as the measures of ϕ-divergence, i.e. that it would consider two distributions on top of each other with different variances to be "distant". For the distributions in Fig. 4.8 are $\mathcal{D}_{\text{EMD}}(\text{case 1}) = 1$, $\mathcal{D}_{EMD}(\text{case 2}) = 1.5359$, indicating the same issue as in f-divergence.

Many of the measures in [13] have this same issue, with one potential exception. The "separation" distance, originally advocated in work on Markov chains [1] is promising, though it does not have the properties of a metric. It is defined as

$$\mathcal{D}_{sep}(P, Q) := \max_i \left(1 - \frac{P(i)}{Q(i)}\right) \tag{4.25}$$

and it is less vulnerable to the issues of f-divergence and EMD; for example, for the distributions in Fig. 4.8, it finds that $\mathcal{D}_{\text{sep}}(\text{case 1}) = 1$, and $\mathcal{D}_{\text{sep}}(\text{case 2}) = 0.6869$. However, it has several weaknesses: firstly, the order of the arguments can substantially change the

answer. For the same two cases, $\mathcal{D}_{\text{sep}}(Q_1, P_1) = \mathcal{D}_{\text{sep}}(Q_2, P_2) = 1$ (note order of P and Q). Though the other properties of a norm are not critical, it is important to get a consistent answer when comparing two gesture classes. Secondly, because it uses an absolute maximum on observed probabilities rather than parameters of a best-fit characterization, this method may be particularly vulnerable to low population sizes and histogram scaling. When higher-dimensional data is involved this problem may compound, as the population sizes required to adequately characterize high-dimensional data increase and many datasets begin to appear sparse (known as the "look-elsewhere effect"). Ultimately, the direct connection between SVM modified margin and empirical risk and the symmetry of that measure made it a stronger choice.

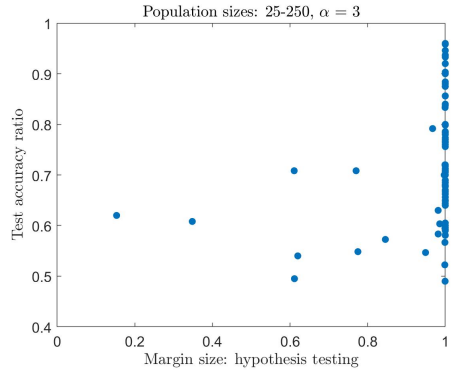
4.2.3 Hypothesis testing on class score distributions

As in §4.1.1, the method for classification of a new data point depends on its score $\hat{y}^i = (a'x^i + b)$. The idea behind this method is to use two-tailed t-test hypothesis testing on the $\hat{y}$ distribution for each class. Using a null hypothesis of $\mathrm{E}[\vec{\hat{y}}^1] = \mathrm{E}[\vec{\hat{y}}^2]$, a larger p-value would indicate that the null hypothesis is more likely given the data realization. Thus, a smaller p-value would indicate more separability. To transform this into a separability calculation that satisfies property 1, the hypothesis-testing distance measure is defined as

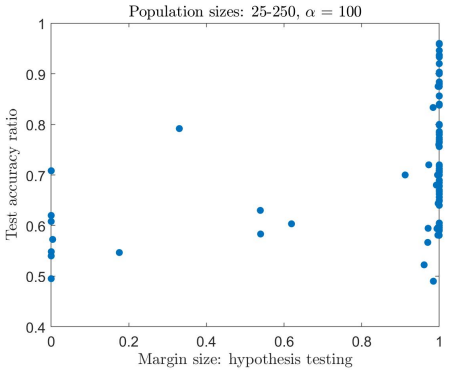
$$\mathcal{D}_{hypoth}(\text{class 1}, \text{class 2}) := (1 - p)^{\alpha} \tag{4.26}$$

where p is the p-value for the null hypothesis that $\mathrm{E}[\vec{\hat{y}}^1] = \mathrm{E}[\vec{\hat{y}}^2]$. The problem with the p-value in this context is that it tends to saturate at zero because the score distributions produced by SVM will always be at least somewhat distinct. The α parameter is intended to address this; by exponentiating many near-1 numbers, maybe they can be separated out.

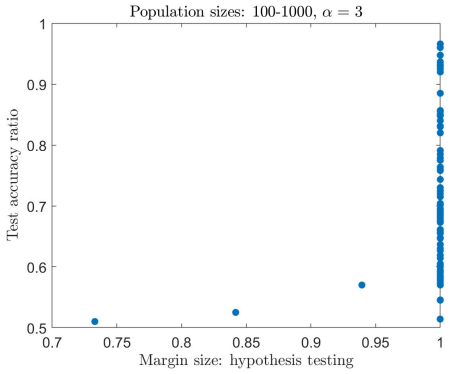
However, this turns out not to work as well as hoped, as demonstrated in Fig. 4.9. Even with the numerical precision of p elevated to 64 bits, the large majority of all p-values are sufficiently saturated at identity such that even a large exponent ($\alpha = 100$, Fig. 4.9 (b) and



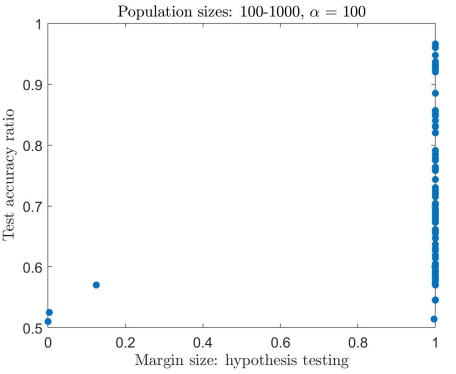
(a) With relatively low sample sizes and $\alpha = 3$, there is some spread in $\mathcal{D}_{hypoth}$ but the evaluated distance is 1 for most of the 90 cases represented here.



(b) A much higher α value leads to a somewhat better spread in evaluated distance, but still with a large cluster at $\mathcal{D} = 1$.



(c) Larger sample sizes exacerbate this saturation problem.



(d) This holds true again for large values of α.

Figure 4.9: Study of hypothesis testing on scores as a distance measure, with various values of α and population size ranges. Overall the distance measure does not have a strong positive correlation with performance (represented by test accuracy) due to the saturation of (1-p) at 1.

(d)) does not achieve notable discernment between instances. This saturation is even more pronounced with a higher range of sample sizes (Fig. 4.9 (c) and (d)), as the t-test becomes more certain when the score distributions are better characterized.

4.3 Considered measures: alternate SVM-based

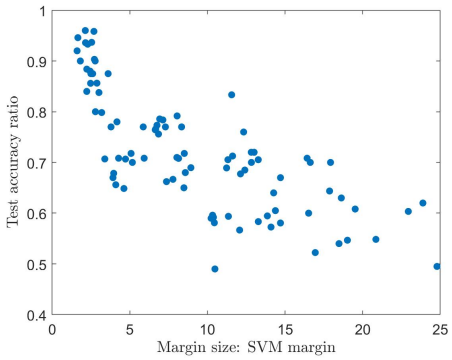
4.3.1 Unmodified SVM Margin

This method was used for the first published work on this topic [18] and it does provide good results under certain assumptions, but the regional inversion of the margin correlation is a serious barrier to direct use. If all data can be confined to the separable region, performance is acceptable, but otherwise positive correlation between separability and performance can disappear or even reverse as shown in Fig. 4.10.

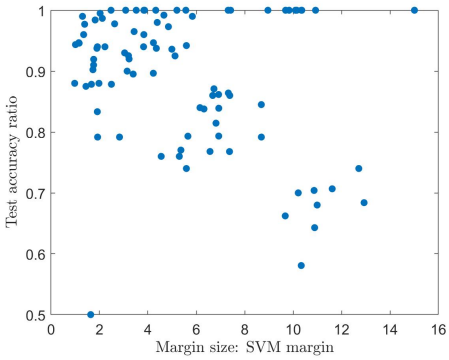
4.3.2 Robustness Distribution

It would be desireable to be able to select the most separable case with statistical certainty for an online application. One method for achieving guarantees is to transform the problem into a ranking and selection problem, for which validated machinery already exists. To do this, the two labeled classes and their interaction with the model separating them must be combined into a single distribution that is characterized by a higher mean indicating a more robust separation between the classes. With this achieved, the main question is the differing sample requirements of the available methods of ranking and selection for this problem and if they reliably select the case that is known to be most separable.

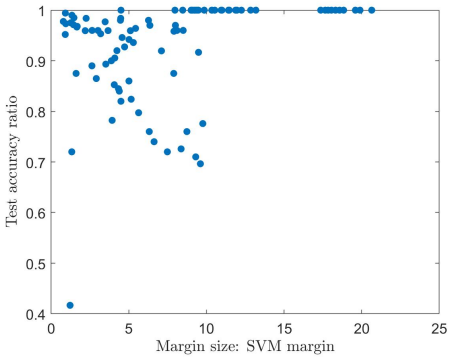
A comparative study is made of several methods of ranking and selection with statistical guarantees for unknown and unequal variances, with a detailed treatment in Appendix D. Worth mentioning is the highly restrictive assumption placed on the gesture class distributions in order to use this method, which under these restrictions performs well. The machinery entirely breaks down if the classes are not multivariate normal with symmetric



(a) In one dimension on generated data, the margin width actually has an inverse correlation with the test accuracy, because this data is mostly enmeshed.



(b) As the dimensionality gets higher (this is on 10-dimensional multivariate normal data), the inverse relationship disappears.



(c) Finally, on high-dimensional data, higher distance corresponds to higher test accuracy, because data is separable and so the classifiers can be hard-margin..

Figure 4.10: Illustration of the dimensional sensitivity of the margin as performance measure. More details on this result and the code used to generate these plots is in Appendix A.

distributions. It is not obvious that a real case exists where these assumptions are remotely reasonable or that these conditions can be relaxed, so this work is primarily an exercise in what statistical guarantees might look like rather than a serious proposal.

4.4 Conclusion

Each method has strengths and weaknesses (summarized in Table 4.1) but none discussed besides the modified SVM margin is successful. All have nontrivial cases in which they would fail or mislead a designer. Ultimately, the modified SVM margin fulfills all three properties by virtue of the core SVM algorithm (fulfilling P2 and P3) and the analytically backed regional inversion (fulfilling P1).

	SVM Margin	R. Distribution	f-Divergence	EMD & Similar	Hypothesis Testing
P1: Positive correlation between distance and performance	✗ Fig. 4.10	✓	✗ Fig. 4.8	✗ Fig. 4.8	✗ Fig. 4.9
P2: Applicable to different data distributions	✓	✗ Appx. D	✓	✓	✓
P3: Robust with outliers and small populations	✓	✓	✓	✗ §4.2.2	✓

Table 4.1: Summary of the degree to which each distance measure satisfies the separability properties outlined at the beginning of the chapter. An ✗ doesn't mean that the property is never satisfied, rather it means that the method has some large vulnerability or pathological case regarding that property that is detailed in its section. The figure or section which represents the failure mode is referenced in each box along with the ✗.

Chapter 5

GESTURE REPRESENTATIONS

A performance of a "gesture" is a collection of measurements of human motion, sometimes static and sometimes over a period of time. These measurements may take a variety of forms; it could be the color values of every pixel in a video stream, electrical signals for muscle activation, the output of a lower-level pose identification algorithm, some combination or nonlinear transformation of these, or many other possibilities. In order to apply the previously developed tools to gesture classes, these measurements must be transformed into consistent, coherent mathematical objects.

5.1 Structure of gesture representation

First, n measurements of human motion at t timesteps are concatenated into a $n \times t$ matrix. Typically, this matrix is very large. At this stage, preliminary operations such as smoothing, filtering, normalizing, and resampling are easiest.

Repetitions are necessary for gesture class characterization, ideally a large population of them. In order to represent a class as a matrix, which enables the usage of machinery such as PCA and SVMs, individual performances are stacked into a vector of dimension nt and k performance vectors are concatenated into a matrix. Thus, a labeled class with k repetitions is in fact an $nt \times k$ matrix. Different gestures may have significantly different distributions even with the same actors, as they may depend on individual physical configurations, variable interpretations of performance notes (ex: wave with open hand), and environmental factors at time of measurement.

5.2 Dimensionality reduction

There are a number of problems with these gesture class matrices as they now stand. First and most simply, they're too big, particularly as the number of time steps required increases. If the measurement is the pixels in a low-resolution 640×480 video taken at a standard 30 frames per second for two seconds, that means that a single performance of a single gesture has over 18 million features ($640 * 480 * 60 = 18,432,000$). This size can cause memory and efficiency issues, particularly relevant in an online application, but more importantly this many features per example means that the "curse of dimensionality" is in play [20]. This is a term for the problems that arise when a dataset has so many dimensions that all the data become sparse, which has implications for how this data can be efficiently characterized. The amount of data that is required to support data characterizations statistically increases dramatically at this scale, and there tends to not be enough variation in calculated distances to make meaningful inferences.

Secondly, there is a high degree of collinearity in the dataset. This refers to the situation in which one feature can linearly predict another with a high degree of accuracy. Even if the base set of measurements is completely uncorrelated (an unlikely scenario in gesture; as an example, the positions of the first and second knuckle on a finger are highly correlated), the temporal stacking of the features to make a class matrix means that there is strong periodic correlation between the same feature through time. High collinearity can cause the values of individual regression coefficients (or in this case, components of a) to change erratically and significantly with small changes to the data, and can make the overall model less robust.

5.2.1 Feature extraction

Feature extraction is the process of building derived values from a raw set of measurements. In theory, these derived features will be more suited to solving the problem at hand than the base measurements. This process can be highly complex, such as the PoseNet algorithm [23]

for extracting human positions from raw visual data, or it can be finding relatively simple such as linear combinations of raw measurements as in PCA. The the process of using specialized domain knowledge to manually build a set of custom features is also used extensively in the field of gesture recognition. This process is generally called feature engineering.

5.2.2 Feature selection

Feature selection is the process of taking a large number of features and through some appropriate method selecting the ones that are the most relevant to the problem at hand. Sometimes this can be done simultaneously with feature extraction (as in PCA), but there are also many other ways to achieve this goal. Using feature selection may prevent overfitting of data, lead to more interpretable models, and better facilitate online applications due to lower memory and processing requirements.

5.3 Feature reduction method

5.3.1 Custom features and preprocessing

Particularly for visual measurement systems, there are a number of custom feature sets calculated from raw joint position data that researchers use from the raw data. Some examples include the relative joint angles of the finger bones, Euclidian distance between each possible joint pair in 3-D space, and more abstract measures like the *Shape of Connected Joints* [9]. There is often also preprocessing such as filtering, normalizing, and resampling the data to achieve a consistent dt.

5.3.2 PCA

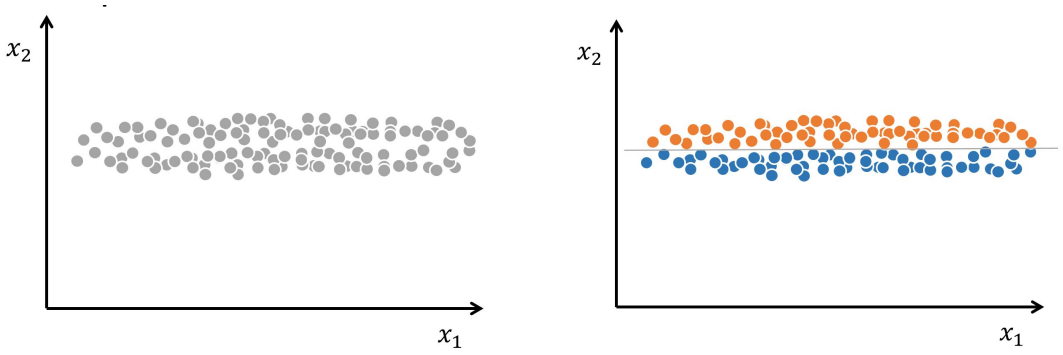
Principal components analysis (PCA) is a process that forms an orthogonal (linearly uncorrelated) basis set, organized in order of decreasing variance. Formally, the recursion to find

the k'th principal component is

$$\begin{aligned} \max_{p_k} \quad & \mathrm{Var}(p_k' x) \\ \text{subject to} \quad & p_k' p_k = 1, \\ & \mathrm{Cov}(p_k' x, p_j' x) = 0, \ j = 1, \ldots, k-1. \end{aligned} \tag{5.1}$$

This is commonly used and very functional. In particular, the inducement of an orthogonal feature set helps correct the strong temporal correlation that is introduced to the data by stacking individual gesture performances. However, it does have some limitations.

5.3.3 SVM-based subselection



(a) Here, x_1 is ranked as more important by PCA because it captures more variance. However, classes are not considered in the PCA algorithm.

(b) When classes are revealed, it is obvious that x_1 has no bearing on classification. If the PCA cumulative contribution is used as a cutoff and only x_1 included, the classes would be wholly inseparable.

Figure 5.1: Illustration of the problems with using PCA as both feature extractor and feature selector.

The 2-D example in Fig. 5.1 illustrates the problem with using PCA for both feature extraction and feature selection. Two classes of data are contained within the visualized

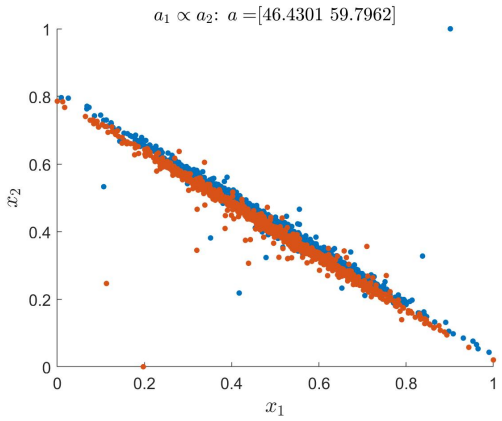
data, but the PCA algorithm does not distinguish between them. If PCA was used to pick the most important feature, it would select x_1. Because of the way PCA operates on data, the classes are not knowable to the algorithm as it generates features. However, when the classes that make up this data are revealed (Fig. 5.1 (b)), x_1 is shown to be irrelevant for classification and were the data to be reduced to just that feature, these classes would appear intractable for classification.

This problem is recognized in SVM literature [15, 5, 2]. One way of determining relative importance of the n'th feature is to look at the magnitude of its weight $a(n)$ of the SVM solution. For this purpose, it is productive to visualize a as the vector normal to the hyperplane. The relative scaling of the components (assuming data is correctly normalized) of a defines the orientation of the hyperplane and thus the relative importance of the variable to classification; if the hyperplane normal does not have a large component corresponding to a variable, that variable may not be significant to classification. In the example above, the hyperplane normal is oriented vertically, correctly indicating that the most relevant feature for classification is x_2. An example of this relative weight scaling compared to PCA follows.

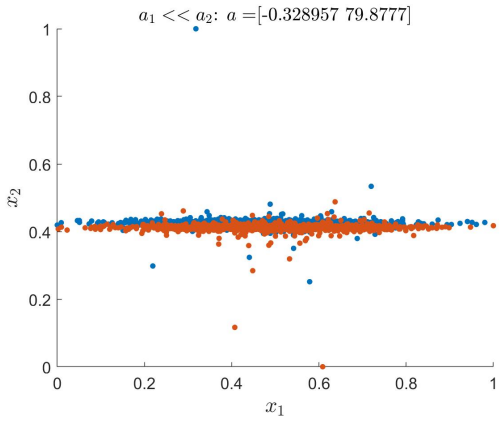
Realized SVM-based selection example

In a simple 2D example with generated data (Fig. 5.2) the selection method does behave as expected. The distributions represented are normal along the long axis $\sim \mathcal{N}(\mu = 0, \sigma = 1)$ and along the short axis, the distribution is mostly the same but with the stability parameter α changed from 2 to 1.5. Said another way, the two distributions are stable$(2, 0, 1, 0)$ and stable$(1.5, 0, 1, \pm 2)$. The effect of this change is to have heavier tails to an otherwise similar distribution. The reported values for a are in Fig. 5.2 and the corresponding PCA results to support the example's conclusion are:

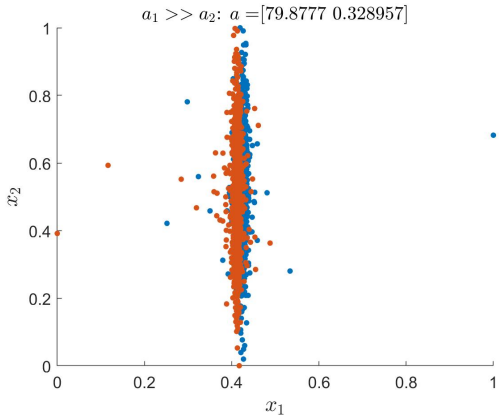
Fig. 5.2 (a): $U = \begin{bmatrix} 0.7883 & -0.6153 \\ -0.6153 & -0.7883 \end{bmatrix}$ and $S = \begin{bmatrix} 0.1981 & 0 \\ 0 & 0.0262 \end{bmatrix}$, indicating roughly



(a) Both features are equally important to classification in these multivariate normal distributions, and the classifier scaling reflects that. PCA agrees, saying that x_1 and x_2 contribute equally to the dominant principal component.



(b) Here, however, the PCA reports that x_1 contributes 99% of the dominant feature, which contributes 97% of the variance. The SVM-based selection is scaled appropriately, and identifies x_2 as by far the most important feature from a classification perspective.



(c) The same happens in reverse, when x_1 is the most important feature for classification.

Figure 5.2: 2D example of SVM-based feature selection in action, in a pathological example for PCA-based feature selection. U and S for each example and details in generating distributions are in §5.3.3.

equal contribution of the features.

Fig. 5.2 (b): $U = \begin{bmatrix} 1.0 & -0.0046 \\ -0.0046 & -1.0 \end{bmatrix}$, indicating that x_1 contributes over 99% of the first feature. Next, $S = \begin{bmatrix} 0.1574 & 0 \\ 0 & 0.0214 \end{bmatrix}$ indicates that the information contribution of that first feature is over 98%. In the common PCA scenario of a cumulative contribution threshold, below which all features are discarded, x_2 would likely be removed from the dataset and the remaining data functionally inseparable.

Fig. 5.2 (c): $U = \begin{bmatrix} 0.0046 & 1.0000 \\ 1.0000 & -0.0046 \end{bmatrix}$ and $S = \begin{bmatrix} 0.1574 & 0 \\ 0 & 0.0214 \end{bmatrix}$, again indicating that the most relevant feature for classification (here x_1) would likely be removed from the dataset if PCA thresholding was used for feature selection.

5.3.4 Separability-based multi-class extension

In literature, these weights of a are used directly to determine the relative importance of features [15] and used for recursive feature selection. When extended to a multi-class classification scheme with a similar recursive elimination [5], squared magnitudes are summed across models to get an combined importance ranking. That algorithm is described below, for K models and starting with the maximum number of features.

1. Calculate the optimal $a_k, b_k \; \forall \; k \in K$, and let $a_{i,k}$ be the i'th component of the k'th model.

2. Calculate the importance criterion IC for each feature i:

$$\mathrm{IC}_i = \sum_{k=1}^{K} a_{i,k}^2$$

3. Remove feature $\arg\min \mathrm{IC}_i$

4. Until desired number of features is achieved, repeat steps 1-3.

I propose two modifications to this method for this gesture classification task, henceforth referred to as Modified Chapelle. First, instead of directly summing the squared weights, a better importance criterion for a feature would be the relative orientation of the hyperplane to that feature. This is calculated by the dot product of the hyperplane normal a and the feature vector. Since the features have been orthogonalized by PCA prior to feature selection, the importance for the $i'th$ feature for a single model reduces to $\frac{a_i}{||a||}$. This is the cosine of the angle between the feature and the hyperplane normal; large values indicate alignment and thus importance. Negative angles are equivalent alignment as positive angles, and so an absolute value is taken. The second modification is to, instead of summing across all models, to exploit the structure of the problem and only sum across models of interest to ultimate performance. In this case, that means all pairwise models between work gestures.

The modified algorithm description follows. This is begun with the all features. The subset of relevant models R is identified as $R = \{k \in K \mid g(k) \cup C \neq 0\}$, where $g(k)$ is the gesture pair that comprises the k'th model labels.

1. Calculate the optimal $a_k, b_k \ \forall k \in K$, and let $a_{i,k}$ be the i'th component of the k'th model.

2. Calculate the importance criterion IC for each feature i:

$$\mathrm{IC}_i = \sum_{k \in R} \left| \frac{a_{i,k}}{||a_k||} \right|$$

3. Remove feature $\arg\min \mathrm{IC}_i$

4. Until desired number of features is achieved, repeat steps 1-3.

5. Compute cosine of orientation of each feature to each model's hyperplane normal ($|\frac{a_i}{||a||}|$)

This can be accelerated for computational efficiency by removing all features below a steadily incrementing cutoff criterion instead, which will then remove several thousand features in the first step of this test problem.

5.4 Results

5.4.1 Offline results

Table 5.1: Number of features required for the top ten most separable gesture sets to retain 99.9% of test accuracy obtained when training with all features. The modified Chapelle method achieves on average 17.5% reduction in required features for accuracy retention relative to PCA-based selection. The average dimensionality reduction with the modified Chapelle method with 99.9% accuracy retention is 99.3%.

Gesture Pair	PCA	Mod. Chap.
[1,17]	18	23
[4,17]	56	20
[5,17]	63	35
[6,17]	39	38
[6,26]	39	41
[6,28]	39	44
[14,17]	18	26
[17,24]	109	24
[17,26]	79	37
[17,28] (s^*)	31	24
Average	49.1	31.2

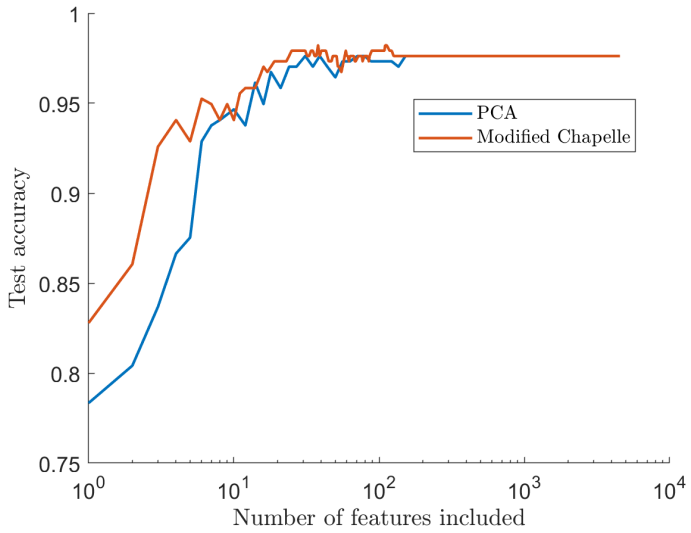


Figure 5.3: Feature selection comparison for s^*, offline, relative to PCA-based selection. As noted in Table 5.1, eight fewer features are needed in the modified Chapelle method to retain 99.9% of full-featured accuracy. Additionally, there is a 6% increase in accuracy using only one feature when selected according to the modified Chapelle method rather than by PCA.

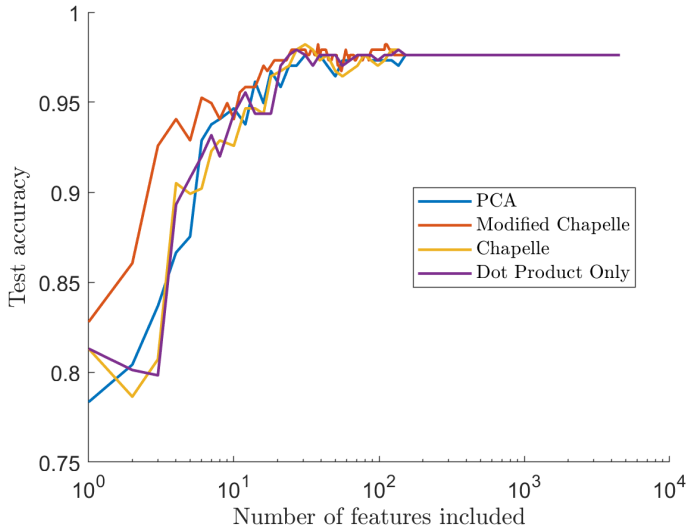


Figure 5.4: The Modified Chapelle not only outperforms PCA, it also outperforms other SVM-based methods of feature selection for s^*.

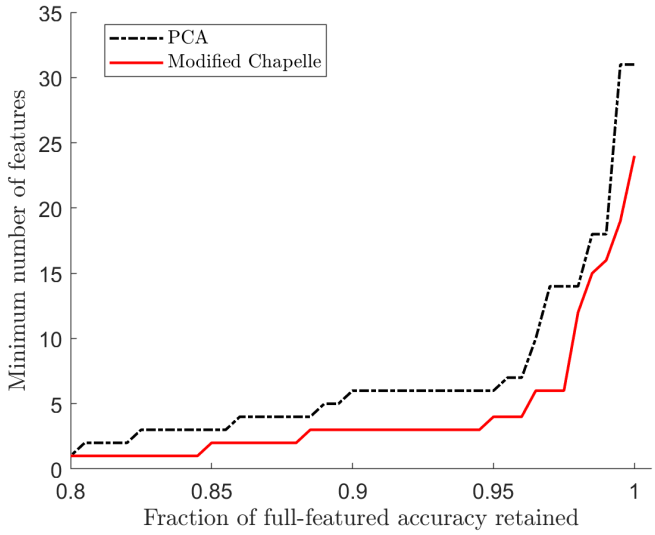


Figure 5.5: As performance deteriorates at very low feature counts, the modified Chapelle method consistently needs fewer features to achieve the same performance.

Extremely low-dimensional feature sets generally do not have good performance on these datasets. Area of interest for results is twofold:

1. Does this method identify and eliminate features with less classification ability and so retain or improve classification relative to the full-featured representation, and if yes does it do so better than other methods of feature reduction, and

2. When exchanging dimensionality for performance, does it retain classification ability?

Fig. 5.3 shows the recursive feature selection and the equivalent PCA features for the maximally-separable gesture set s^*. The minimum number of features to retain 99.9% of the test accuracy when using all features still reduces the dimensionality by over 99%. Results for the top 10 most-separable control gesture pairs are shown in Table 5.1. The modified Chapelle method achieves on average 17.5% reduction in required features for accuracy retention relative to PCA-based selection. Additionally, it compares well to the original Chapelle

method and non-selective dot product, as shown in Fig. 5.4. When accuracy begins to be compromised by the extremely low feature count, this method consistently requires fewer features to retain a given fraction of the full-featured accuracy than PCA selection (Fig. 5.5).

5.4.2 Online results

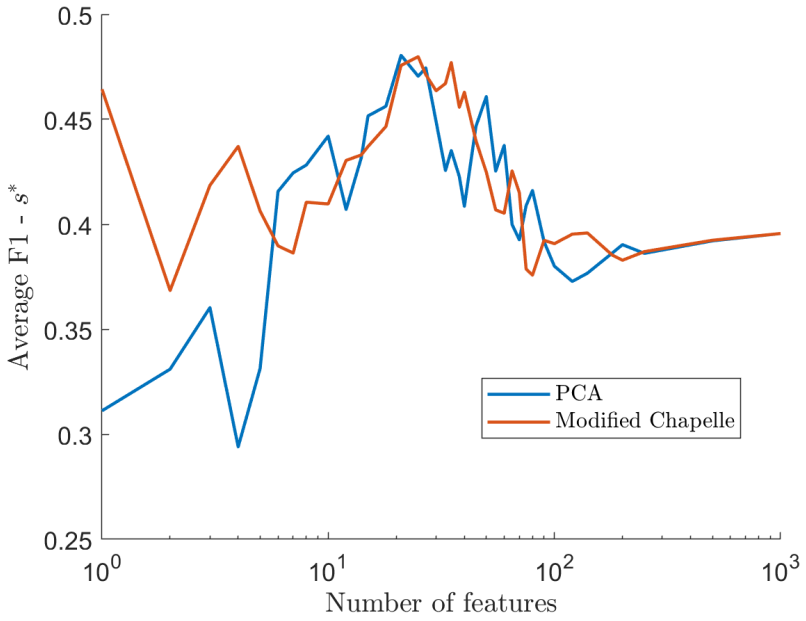


Figure 5.6: Recursive feature selection performance results for the online classification task, measured in average F1 measure across all ten subjects. Performance at the 99.9% accuracy retention cutoff established in Table 5.1 is near a peak of 15% better F1 performance relative to the full-featured data, indicating that problematic features have been removed effectively. Additionally, in the extreme case of using only a single feature out of 4551 total there is a 49% performance improvement if that feature is selected by the modified Chapelle method rather than by PCA.

A similar pattern is seen in the online classification results for various levels of feature reduction, as shown in Fig. 5.6 for s^* and Fig. 5.7 for s°. Here the removal of problematic

features has a proportionally much greater effect, and the overall classification ability is better preserved in the low-feature region by the modified Chapelle method.

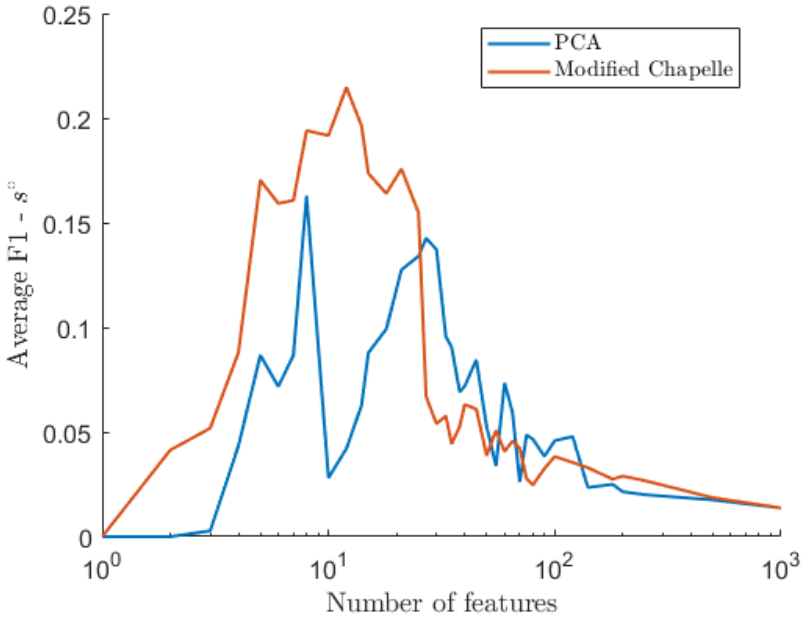


Figure 5.7: The positive effect of feature selection is even more substantial when applied to the minimally-separable set s°. Removal of problematic features means the modified Chapelle method achieves a high of ten times higher average F1 than the full-featured representation, 30% higher than achieved by PCA selection with any number of features.

5.5 Conclusion

The online and offline results of this evaluation indicate that selecting a small population of highly discriminative features provides a substantial performance benefit in online classification, both relative to PCA-based selection and in absolute terms. The proposed modified Chapelle method achieves on average 17.5% reduction in required features for 99.9% test accuracy retention relative to PCA-based selection. The average dimensionality reduction with the modified Chapelle method with 99.9% accuracy retention is 99.3%.

Online performance at the 99.9% accuracy retention cutoff established in the offline results for s^* is near a peak of 15% better F1 performance relative to the full-featured data, indicating that problematic features have been removed effectively. Additionally, in the extreme case of using only a single feature out of 4551 total there is a 49% performance improvement if that feature is selected by the modified Chapelle method rather than by PCA. For s°, problematic features are removed more effectively by the modified Chapelle method compared to the PCA method, as exemplified by a 30% higher peak performance in features selected by the modified Chapelle method.

Chapter 6

VALIDATION ON ALTERNATE SENSING ARCHITECTURE

6.1 Motivation

Different sensor architectures have very different strengths and weaknesses, even for the same task. A system designer might chose between two major modalities for recognizing gesture in the same context. Presented in this chapter is the same task as considered in previous chapters, but sensed using an architecture from the other major modality: direct measurement. The success of applying the machinery built up in previous chapters and the striking qualitative similarity of results also indicates that separability-based structural design tools can generalize to other classification problems.

6.2 Sensing architecture

6.2.1 Visual sensing - visible and structured light

Visual sensing was used in prior work [18] and is overall a good sensing modality to use for gesture in particular, as it mimics the way humans perceive them and thus will also be able to shortcut domain restrictions in the same way. No explicit connection is needed between the observer and observed besides line-of-sight. Some examples of this modality include the Leap motion, raw images, Kinect/structured light, output of a lower-level pose estimation algorithm, etc. Benefits include the lack of a need for signal communication, which is very useful in some considered domains (underwater, hard vacuum, otherwise adverse conditions), as well as no need for any objects or coverings attached to the sensed human's hand. Drawbacks of this sensing architecture include that it is highly sensitive to many environmental

factors, including light, distance to target, presence of multiple targets, background, motion of platform, and unique physical configurations. It usually requires direct line-of-sight within a specified envelope and can have highly nonlinear failure modes that mask as unaffected data.

6.2.2 Direct physiological measurement

This is a common alternate method to visual measurement, encompassing myoelectric signals, accelerometers, bend sensors, pressure sensors, and more. There does usually need to be some signal communication between human and observer, but these methods also tend to be much less sensitive to environmental factors. Benefits of this method include substantially less sensitivity to environmental factors, tracking loss is not an issue, tracting stability is invariant to skin tone, light and line of sight are not necessary, and the sensing can generally be a self-contained unit. This method also lends itself well to directly sensing contact forces, including tool presence and/or some measure of applied grip strength. Drawbacks include that the need for worn sensing apparatus can impact ability to complete work or be uncomfortable for long periods of time. Additionally, visual intuition breaks down for what gestures will look similar in this measurement space - two bend positions that are visually very different can appear the same in measurement. This can also be more sensitive to subject-specific differences, while optical methods will generally impose strong assumptions about hand size and shape.

6.2.3 Implications of new sensing architecture

This transition to the direct physiological measurement scheme means that nothing beyond the systematic design tools themselves can be reused from chapters 3 and 5. The data for candidate control gestures and observations of work data must be captured and processed for the new platform in order to determine the most and least separable sets for online

comparison, and the online classification implemented. Similarly, the feature selection must be implemented for the new data. The main motivation for this chapter is to, by carrying over the design tools and nothing else, show that the results presented in Chapters 3 and 5 are not an artifact of the datasets used or the optical sensing platform, but the result of systematic design according to the presented method.

6.3 Implementation

The platform under consideration is a glove with attached sensors: six bend sensors and a 6-DOF accelerometer.

6.3.1 Sensors

Velostat is a conductive plastic originally used for electronics transport. Its electrical resistance changes with applied pressure. When a conductor is placed on either side, it can be used as a cheap and lightweight pressure sensor. The bend sensors were designed iteratively to achieve the widest possible range of resistance corresponding to the sensors range of motion and to achieve consistent results despite the gloves flexibility. Modular sensor strips attach to the glove with industrial Velcro, allowing for faster sensor iteration and adjustable placement.

Alternate sensor designs to the final are shown in Fig. 6.1. Prior experience with the sensor on the far right led to that being considered the "baseline" sensor, but some investigation was made of increasing the number of layers of Velostat, varying the widths of copper tape, and material preparation. More layers of Velostat increased precision of resistance response but added stiffness to glove, which would result in a less comfortable glove. A single layer of Velostat gives low-noise results that are consistent across trials, but with a low overall resistance range (0-2 ohms). Thinner copper strips increased resistance range (0-10 ohms) but increased noise substantially. Strips of copper and Velostat are layered, each adhered

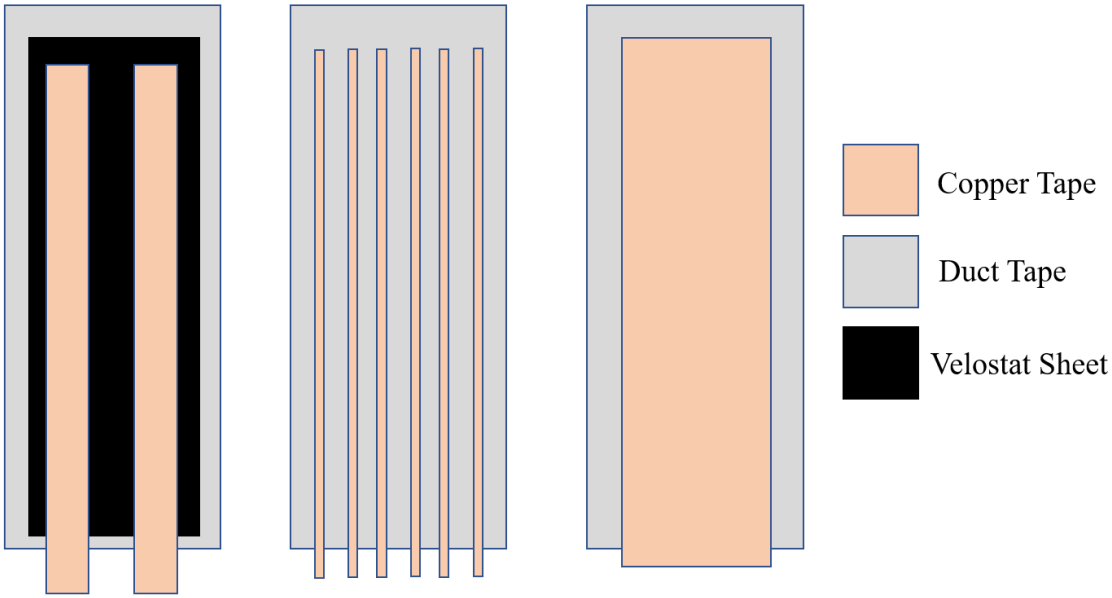


Figure 6.1: Three candidate sensors were designed and constructed, shown above. Each of these sensor designs is sandwiched with its mirror image and a layer of Velostat between them. In the case of the first design, this brings the total layers of Velostat to three. Ultimately, the sensor on the far right was selected as the final design. Design considerations are detailed in §6.3.1.

to duct tape but not to each other. Ultimately preparing the Velostat before assembly by pressing evenly with a 50-lb. weight substantially increased range without an equally-scaled increase in noise. The final sensor used wide copper strips and a single layer of Velostat, and had a range of 25-1k ohm. Its design is shown in Fig. 6.2. The usage of copper tape conductive adhesive to adhere the Velostat and copper layers was attempted but reduced range substantially (on the order of .01-6 ohm vs 25-1k ohm) compared to no adhesive.

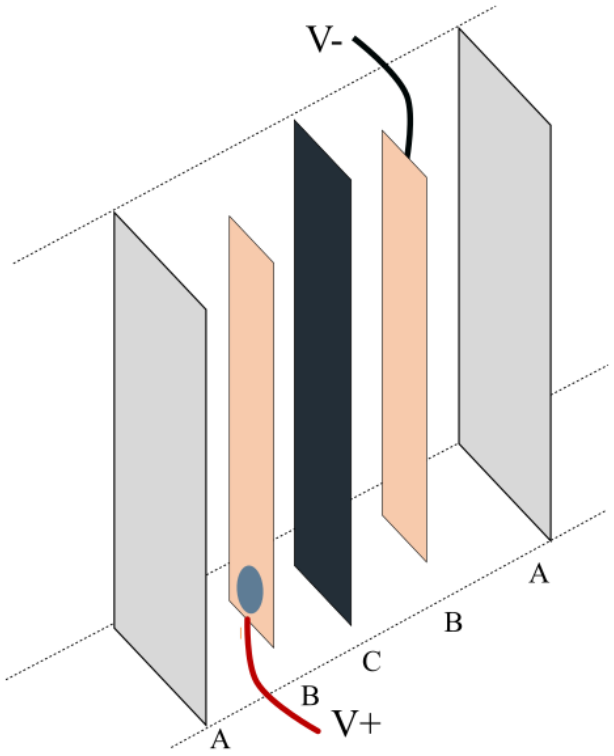


Figure 6.2: Final sensor design, exploded view. The materials labels are as follows: A - duct tape, B - copper tape (adhesive side away from Velostat), and C - Velostat. Each sensor is driven at 5V by the microcontroller and the resistance range is on the order of 25-1k ohm.

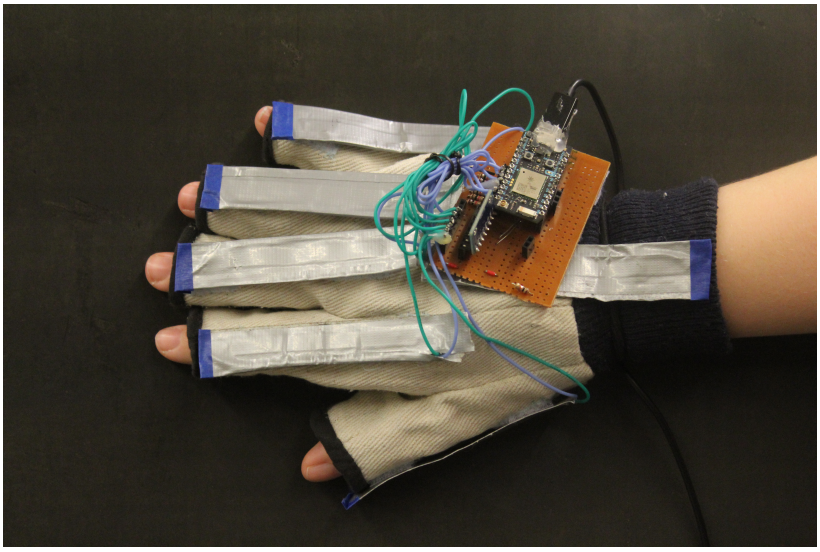


Figure 6.3: Final design of glove used to collect data. Front of glove has no attachments or modifications, to minimize motion impediment.

6.3.2 Glove design

Bend sensors are attached with one on each finger, on back only to minimally interfere with dexterous manipulation. For orientation and acceleration sensing, a M9050 combination accelerometer has linear and rotational acceleration (6 DOF). Minimum interference and consistent fit across a range of participants would suggest a stretchy glove, but rigid glove provided the best ability for sensors to move consistently with hand motion. Fingertips were removed for ease of collar manipulation and general participant dexterity. Final glove design is shown in in Fig. 6.3.

Initially sensors were sewn directly to glove but for ease of sensor comparison and replacement, attachment method transitioned to industrial velcro. The microprocessor and breakout for associated electronics were also held onboard glove with velcro. The Particle Photon STM32 processor and generic STM32 board was chosen as the microprocessor for its 12 bit ADC and potential for wireless communication. Data transmission via Wifi had

issues with packet loss and noise, so ultimately wired communication was used for these experiments but it would be certainly possible to move past wired communication, given more development time.

6.3.3 Data Collection

The candidate control gestures and their unique identifiers are as follows:

1. G7 - Index finger swipe right
2. G11 - Trace X, index finger
3. G12 - Trace +, index finger
4. G14 - Index finger shake/wave
5. G17 - Fist to fully extended fingers
6. G21 - Full/open hand swipe right
7. G22 - Full/open hand swipe left
8. G23 - Full/open hand swipe up with palm upward
9. G24 - Full/open hand swipe down with palm downward
10. G28 - Shake/wave with open hand
11. GM1 - Fist then hard flick of index finger up
12. GM2 - Fist then index and middle hard flick up

13. GM3 - Raise index and middle from fist without flick

14. GM4 - Raise index and middle fingers and cross them (like ASL “R”)

15. GM5 - Slack hand palm downward transitioning to all fingers together, palm upward

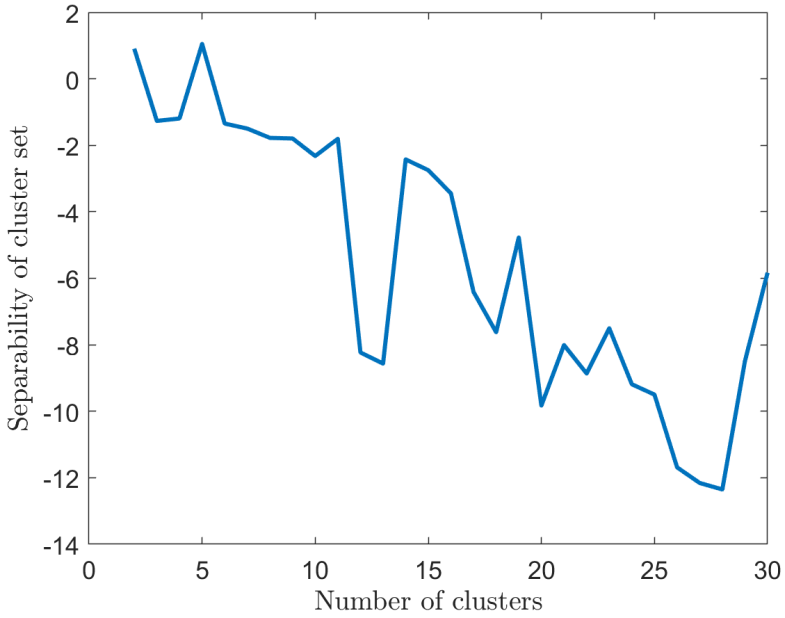


Figure 6.4: Separability analysis to determine how many clusters the work data should be separated into. For a gesture set consisting exclusively of work gestures, separability of the set is the minimum pairwise distance. The separability for the work set is the minimum pairwise distance in the whole population of models. 5 clusters was chosen to move forward.

These gestures were chosen to get a range of what glove could do, comparability with optical sensor dataset, and comparability with MASCOT-RVA2 experiment. The first ten gestures in this set are drawn directly from the SHREC dataset (G# gestures). To collect data for these gestures, twelve subjects performed each gesture 15 times, with labeling of relevant time segments done by an observer. The work data on the task was collected the same

way as in Chapter 3: continuous timeseries examples are observed, randomly sectioned, and clustered with K-means. 5 clusters was chosen as the most separable; work separability evaluation is shown in Fig. 6.4.

6.4 Results

6.4.1 Offline Recognition and Separability

Separability results are qualitatively similar to the previous case. High separability is positively correlated with test accuracy and negatively correlated with sensitivity to individual subject performance. (Results in Figures 6.5 and 6.6 are for leave-one-out cross-validation on all eleven members of the control gesture dataset.) The optimal gestures s^* and the least

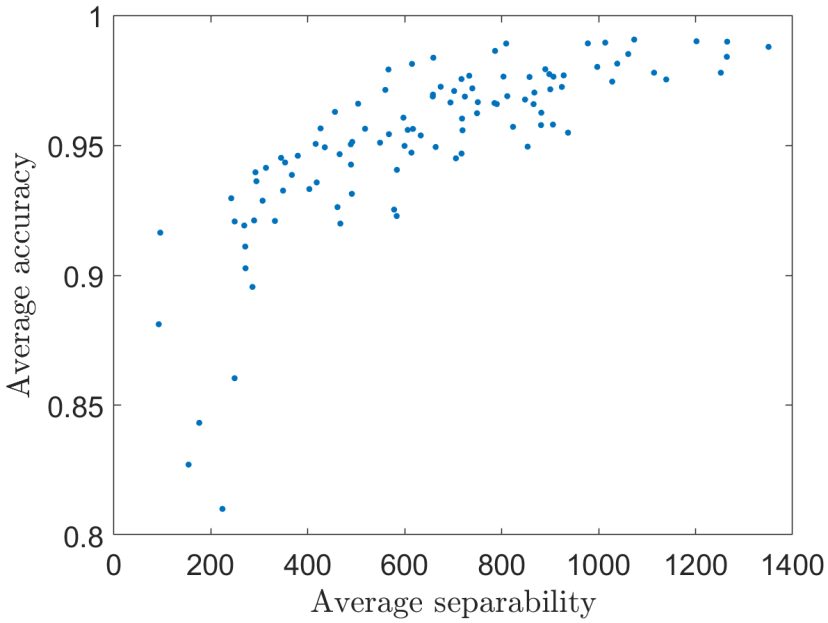


Figure 6.5: As in the previous experiment, separability is positively correlated with accuracy (Pearson correlation coefficient $R = 0.76$). Here, separability and accuracy are averaged over the results of a leave-one-out cross-validation; individual fold results are all qualitatively similar.

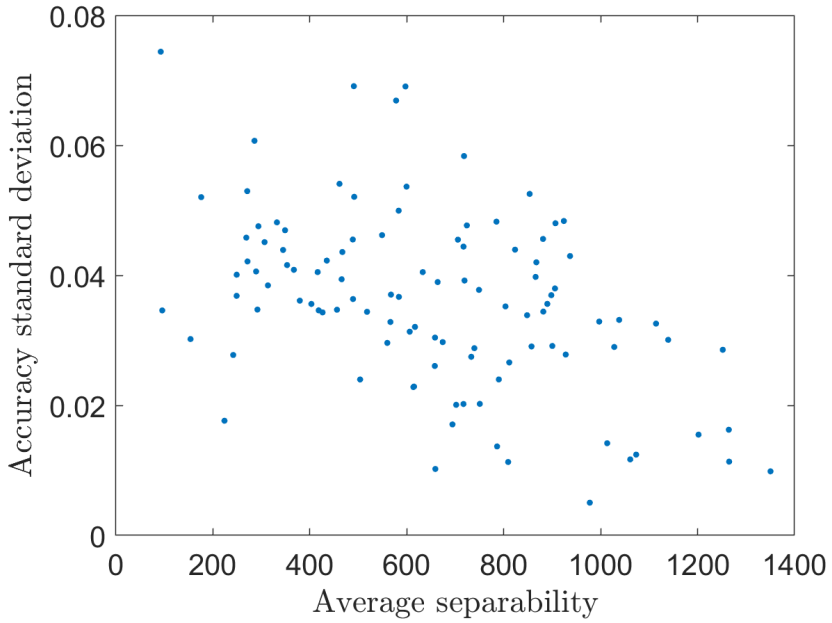


Figure 6.6: For the same leave-one-out cross-validation, separability is also inversely correlated to subject performance sensitivity (Pearson correlation coefficient $R = -0.46$), again qualitatively similar to the previous experiment.

separable set s° chosen for online testing from this offline analysis are described and shown in sequential images in Fig. 6.7.

6.4.2 Online results

In online subject testing, s^* performed substantially better that s° in all metrics, for all subjects, as seen in Table 6.1. The minimum difference in F1 is greater than 150% improvement. The conclusion from this results is that online results validate the utility of separability-based gesture selection in a substantially different sensing paradigm than originally designed for.

An interesting artifact of the result presented in Table 6.1 that the precision of s^* begins to degrade after subject 7. One difference in data collection for subjects 7-10 is that down time between these subjects of interest was effectively zero, while there was generally a 10-15

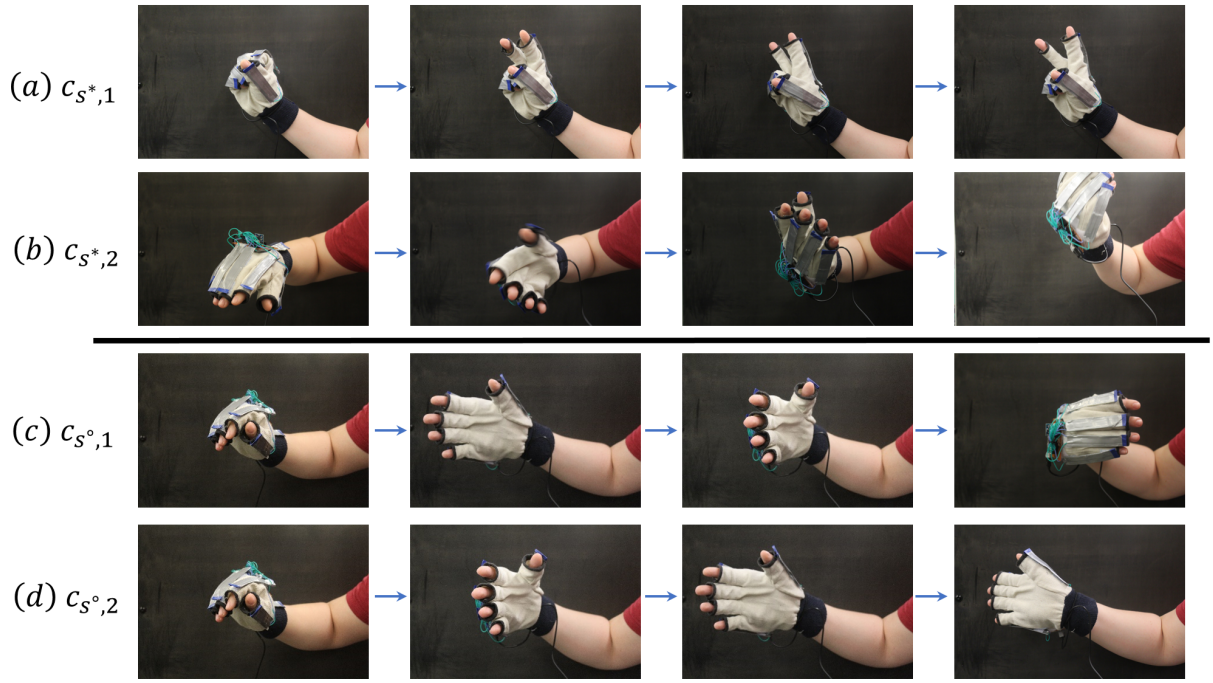


Figure 6.7: Image sequences of members of s^* and s° selected for online testing, which are described as follows: (a) $c_{s^*,1}$ Raise index and middle from fist without flick (b) $c_{s^*,2}$ Full/open hand swipe up with palm upward (c) $c_{s^\circ,1}$ Full/open hand swipe left (d) $c_{s^\circ,2}$ Full/open hand swipe right.

Table 6.1: Subject breakdown of performance metrics. There was improvement for all subjects across all measures between s^* and s°, and on average s^* performs over twice as well and s° in all measures.

Subject	Pr(s^*)	Pr(s°)	Rec(s^*)	Rec(s°)	F1 (s^*)	F1 (s°)
1	0.92	0.10	1	0.31	0.96	0.15
2	0.86	0.22	1	0.44	0.92	0.29
3	0.92	0.56	1	0.44	0.96	0.49
4	0.86	0.45	0.91	0.72	0.88	0.55
5	0.86	0.42	1	0.32	0.92	0.37
6	0.68	0.22	0.90	0.47	0.77	0.29
7	0.82	0.33	0.78	0.33	0.79	0.32
8	0.54	0.29	0.89	0.48	0.67	0.36
9	0.41	0.31	0.80	0.22	0.54	0.25
10	0.44	0.29	0.98	0.44	0.60	0.35
Avg.	0.73	0.32	0.93	0.42	0.80	0.34

minute break between previous subjects. There is still performance improvement between s^* and s° for the affected subjects, but the degradation in performance is concerning. No such large differences in data manifested in the other two rounds of data collection with this platform, so the working hypothesis for this effect is that sustained usage of the glove produced a degradation in data. Three possible causes of this include:

(i) slotted connections on prototype board loosening,

(ii) increase in bend sensor temperature from sustained exposure to body heat could cause change in duct tape insulation or adhesive properties, or

(iii) the Velostat layer itself could build up residual stresses from sustained usage.

In the event more experiments would be performed with this platform or a true online implementation pursued, a more detailed study of this effect would be necessary. However, for this result that examination may be left to future work, as the performance improvement for the F1 measure between s^* and s° for the affected subjects is on the order of 200%, continuing to demonstrate the benefit of separable selection even when there are unforeseen issues with the sensor platform.

6.4.3 Feature selection

Offline results

Results for this sensing platform are qualitatively similar to those presented in Chapter 5, as shown in Fig. 6.8. Classification ability plummets at less than three features but with more, classification ability is better preserved by the modified Chapelle feature selection than PCA-based feature selection with far fewer features needed to retain full-featured test accuracy.

Table 6.2: Number of features required for the top ten most separable gesture sets to retain 99.9% of test accuracy obtained when training with all features. The modified Chapelle method achieves on average 37.8% reduction in required features for accuracy retention. The average dimensionality reduction with the modified Chapelle method with 99.9% accuracy retention is 98.2%.

Gesture Pair	PCA	Mod. Chap.
[1,5]	16	3
[2,5]	14	8
[2,8]	44	4
[2,15]	6	14
[3,8]	44	5
[5,8]	44	27
[8,11]	44	22
[8,12]	50	4
[8,13] (s^*)	44	5
[13,15]	5	7
Average	31.1	9.9

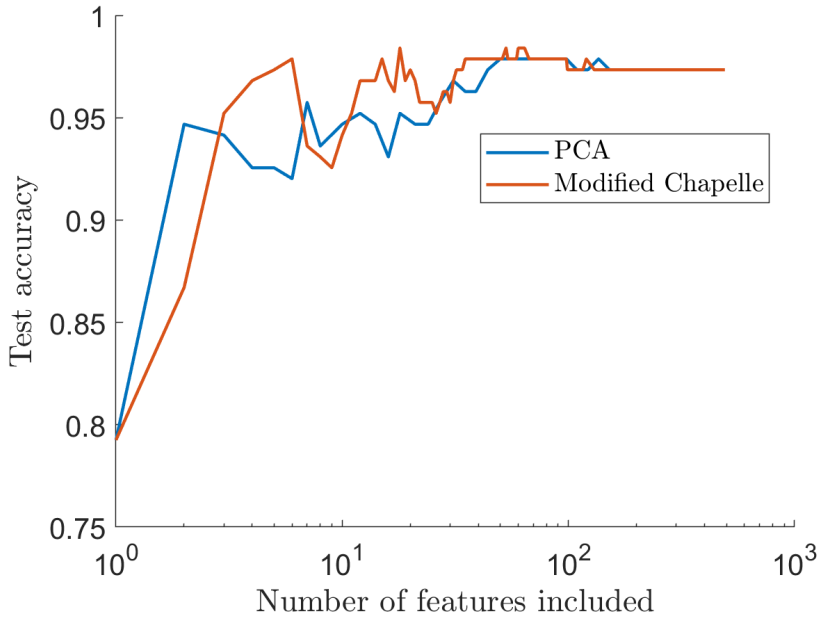


Figure 6.8: Feature selection comparison for s^*, offline, relative to PCA-based selection. As noted in Table 6.2, 88.6% fewer features are needed in the modified Chapelle method to retain 99.9% of full-featured accuracy.

Online results

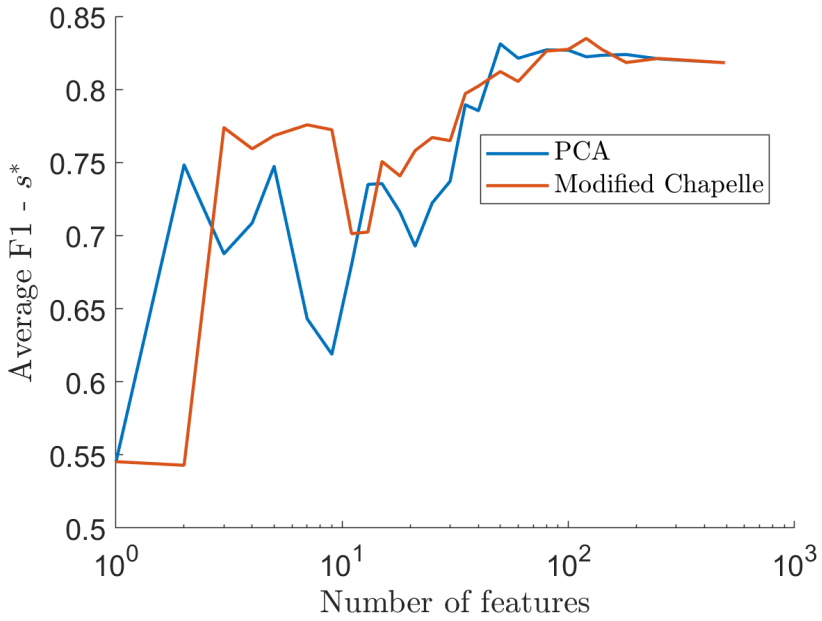


Figure 6.9: Online feature selection comparison results show again that before performance rapidly degrades at extremely low feature counts, modified Chapelle preserves performance better than PCA as features are progressively eliminated.

Online results (Fig. 6.9 for s^* and Fig. 6.10 for s°) are similar, showing that for non-extreme feature reduction the modified Chapelle method degrades more slowly as dimensionality is reduced.

6.5 Conclusions

This parallel experiment on an very different sensing architecture has shown that all of the machinery developed in Chapters 3-5 is, at least to some degree, transferable with no modification to achieve very similar performance benefits. This dataset was substantially less refined, with no feature engineering, no software filtering, and largely homebrew sensors, but

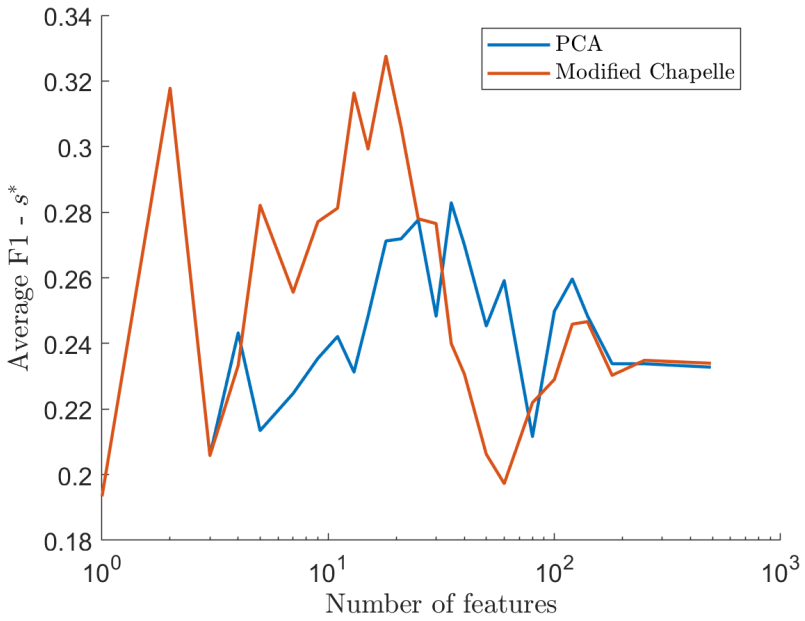


Figure 6.10: Results are similar. Note that for the minimally-separable feature sets, like in the SHREC experiment, the full-featured accuracy is much lower than what can be achieved with more carefully chosen feature representations, and thus there is more to be gained in discarding problematic features and performance can be substantially improved by separable feature selection.

with the highly-separable gestures and features excellent performance was achievable. An even higher correlation between the proposed separability metric and test accuracy of gesture sets than in Chapter 3 was found ($R = 0.76$), as well as a negative correlation ($R = -0.46$) between separability and sensitivity of performance to individuals' performances. In feature selection on this dataset, the modified Chapelle method achieves on average 37.8% reduction in required features for accuracy retention. The average dimensionality reduction with the modified Chapelle method with 99.9% accuracy retention is 98.2%. Future work in this area might explore how the systematic selection process and feature selection might further generalize into other application domains.

Chapter 7

DISCUSSIONS AND FUTURE WORK

Separable selection is positively correlated with performance improvement ($R = 0.59$ and $R = 0.76$, respectively for optical-based and glove-based systems) in all presented applications, including online, offline, datasets with differing sensors, datasets with the same set of sensors, low-cost custom implementations, and off-the-shelf sensors. Separability correlates inversely with sensitivity to subject performance ($R = -0.35$ and $R = -0.46$). Online performance is on average 2-4x higher when using the s^* control gestures rather than the s° gestures, and there was an improvement between gesture sets for every subject on every individual metric. The proposed method of feature selection consistently requires fewer features than PCA to prevent performance degradation (17.5% and 37.8% less, respectively).

The tools for systematic design of human-robot interaction presented in this work have consistently proved to be beneficial to performance. Future work would include looking at finer distinctions in online implementation than highest vs lowest separability, though all gestures chosen in this work were externally-sourced reasonable gestures for a generic online implementation.

In qualitative feedback from experimental participants a correlation emerged between gesture pairs which are easy to remember and which have low separability, though inference of that kind is challenging with ten subjects. Given that the main current gesture selection method is participant preference, it might be interesting to investigate if or how participant preference or memorability long-term is correlated to separability. Qualitative feedback from my own experiments suggests that minimally-constrained participant preference is more likely to choose poorly-separable gestures. It might also be productive to explore how a mea-

sure of memorability or cultural appropriateness could be built into the process of selection via separability, possibly as a weighting system.

Another interesting line of further research could be the construction of a ideal gesture or gesture set from a work context. All work up until now has used externally-sourced lists of gestures, but this is not an inherent limitation of separability analysis. Careful thought would have to go into how to encode the limitations of the human hand into the selection space and how to encode a measure of expected variability in performance.

Additionally, this work has the potential to generalize to arbitrary classification tasks. Future work could include many other types of human-system interactions; I am particularly interested in how the separability concept could extend to the adversarial interactions, given the overriding concern for mitigating the worst cases. Separability overall could be a powerful tool for evaluating how classification systems might perform in online applications prior to deployment.

BIBLIOGRAPHY

[1] David Aldous and Persi Diaconis. Strong uniform times and finite random walks. *Advances in applied mathematics*, 8(1):69–97, 1987.

[2] Paul S Bradley and Olvi L Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, volume 98, pages 82–90, 1998.

[3] Krisztian Buza, Alexandros Nanopoulos, and Lars Schmidt-Thieme. Insight: efficient and effective instance selection for time-series classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 149–160. Springer, 2011.

[4] Justine Cassell, Hannes Hgni Vilhjlmsson, and Timothy Bickmore. BEAT: The Behavior Expression Animation Toolkit. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 477–486, New York, NY, USA, 2001. ACM.

[5] Olivier Chapelle and S Sathiya Keerthi. Multi-class feature selection with support vector machines. In *Proceedings of the American statistical association*, volume 58, 2008.

[6] Jinfa Chen and Won-jong Kim. A human-following mobile robot providing natural and universal interfaces for control with wireless electronic devices. *IEEE/ASME Transactions on Mechatronics*, 24(5):2377–2385, 2019.

[7] Y. Chen, Z. Ding, Y. L. Chen, and X. Wu. Rapid recognition of dynamic hand gestures using leap motion. In *2015 IEEE International Conference on Information and Automation*, pages 1419–1424, August 2015.

[8] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[9] Quentin De Smedt, Hazem Wannous, and Jean-Philippe Vandeborre. Skeleton-based dynamic hand gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–9, 2016.

[10] Quentin De Smedt, Hazem Wannous, Jean-Philippe Vandeborre, Joris Guerry, Bertrand Le Saux, and David Filliat. Shrec'17 track: 3d hand gesture recognition using a depth and skeletal dataset. In *10TH EUROGRAPHICS WORKSHOP ON 3D OBJECT RETRIEVAL*, 2017.

[11] Maxime Devanne, Hazem Wannous, Stefano Berretti, Pietro Pala, Mohamed Daoudi, and Alberto Del Bimbo. 3-d human action recognition by shape analysis of motion trajectories on riemannian manifold. *IEEE transactions on cybernetics*, 45(7):1340–1352, 2015.

[12] Dong-Luong Dinh, Jeong Tai Kim, and Tae-Seong Kim. Hand gesture recognition and interface via a depth imaging sensor for smart home appliances. *Energy Procedia*, 62:576–582, 2014.

[13] Alison L Gibbs and Francis Edward Su. On choosing and bounding probability metrics. *International statistical review*, 70(3):419–435, 2002.

[14] Steve R Gunn et al. Support vector machines for classification and regression. *ISIS technical report*, 14(1):5–16, 1998.

[15] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[16] G. Hackenberg, R. McCall, and W. Broll. Lightweight palm and finger tracking for real-time 3d gesture control. In *2011 IEEE Virtual Reality Conference*, pages 19–26, March 2011.

[17] B. Hartmann, M. Mancini, S. Buisine, and C. Pelachaud. Design and Evaluation of Expressive Gesture Synthesis for Embodied Conversational Agents. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '05, pages 1095–1096, New York, NY, USA, 2005. ACM.

[18] Owan P. Garbini J. & Devasia S. Hendrix, R. Gesture selection for control of a robotic manufacturing assistant. In *Proc. Of the 2nd IFAC Conference on Cyber-Physical & Human-Systems, CPHS*, 2018.

[19] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

[20] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[21] Patrick Juola. Whole-word phonetic distances and the pgpfone alphabet. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 1, pages 98–101. IEEE, 1996.

[22] Juha Kela, Panu Korpip, Jani Mntyjrvi, Sanna Kallio, Giuseppe Savino, Luca Jozzo, and Sergio Di Marca. Accelerometer-based gesture control for a design environment. *Personal and Ubiquitous Computing*, 10(5):285–299, August 2006.

[23] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[24] Dae-Jin Kim, Zhao Wang, Nicholas Paperno, and Aman Behal. System design and implementation of ucf-manusan intelligent assistive robotic manipulator. *IEEE/ASME transactions on mechatronics*, 19(1):225–237, 2012.

[25] Jonghwa Kim, Stephan Mastnik, and Elisabeth André. Emg-based hand gesture recognition for realtime biosignal interfacing. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 30–39. ACM, 2008.

[26] J Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

[27] Shao-Zi Li, Bin Yu, Wei Wu, Song-Zhi Su, and Rong-Rong Ji. Feature learning based on SAEPCA network for human gesture recognition in RGBD images. *Neurocomputing*, 151:565–573, March 2015.

[28] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[29] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[30] W. Lu, Z. Tong, and J. Chu. Dynamic Hand Gesture Recognition With Leap Motion Controller. *IEEE Signal Processing Letters*, 23(9):1188–1192, September 2016.

[31] Robert McCartney, Jie Yuan, and Hans-Peter Bischof. Gesture recognition with the leap motion controller. In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, page 3. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015.

[32] Paul Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern recognition and artificial intelligence*, 116:374–388, 1976.

[33] George A Miller and Patricia E Nicely. An analysis of perceptual confusions among some english consonants. *The Journal of the Acoustical Society of America*, 27(2):338–352, 1955.

[34] M. Mohandes, S. Aliyu, and M. Deriche. Arabic sign language recognition using the leap motion controller. In *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pages 960–965, June 2014.

[35] Mohamed A Mohandes. Recognition of two-handed arabic signs using the cyberglove. *Arabian Journal for Science and Engineering*, 38(3):669–677, 2013.

[36] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz. Hand gesture recognition with 3d convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–7, 2015.

[37] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4207–4215, 2016.

[38] Henry M. Moser and John J. Dreher. Evaluation of the military alphabets. *Speech Monographs*, 22(5):256–265, November 1955.

[39] Barry L Nelson, Julie Swann, David Goldsman, and Wheyming Song. Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research*, 49(6):950–963, 2001.

[40] E. Ohn-Bar and M. M. Trivedi. Hand Gesture Recognition in Real Time for Automotive Interfaces: A Multimodal Vision-Based Approach and Evaluations. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2368–2377, December 2014.

[41] Omar Oreifej and Zicheng Liu. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *Computer vision and pattern recognition (CVPR), 2013 IEEE conference on*, pages 716–723. IEEE, 2013.

[42] Leigh Ellen Potter, Jake Araullo, and Lewis Carter. The leap motion controller: a view on sign language. In *Proceedings of the 25th Australian computer-human interaction conference: augmentation, application, innovation, collaboration*, pages 175–178. ACM, 2013.

[43] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[44] Zhou Ren, Jingjing Meng, Junsong Yuan, and Zhengyou Zhang. Robust Hand Gesture Recognition with Kinect Sensor. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, pages 759–760, New York, NY, USA, 2011. ACM.

[45] L. D. Riek, T. C. Rabinowitch, P. Bremner, A. G. Pipe, M. Fraser, and P. Robinson. Cooperative gestures: Effective signaling for humanoid robots. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 61–68, March 2010.

[46] Yosef Rinott. On two-stage selection procedures and related probability-inequalities. *Communications in Statistics-Theory and methods*, 7(8):799–811, 1978.

[47] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[48] Astrid Schmidt-Nielsen. Intelligibility of ICAO (International Civil Aviation Organization) Spelling Alphabet Words and Digits Using Severely Degraded Speech Communication Systems. Technical Report NRL-9035, Naval Research Lab Washington DC, March 1987.

[49] Atsushi Shimada, Takayoshi Yamashita, and Rin-ichiro Taniguchi. Hand gesture based tv control systemtowards both user-& machine-friendly gesture applications. In *Frontiers of Computer Vision,(FCV), 2013 19th Korea-Japan Joint Workshop on*, pages 121–126. IEEE, 2013.

[50] Nicola Vanello, Valentina Hartwig, Mario Tesconi, Emiliano Ricciardi, Alessandro Tognetti, Giuseppe Zupone, Roger Gassert, Dominique Chapuis, Nicola Sgambelluri, Enzo P Scilingo, et al. Sensing glove for brain studies: design and assessment of its

compatibility for fmri with a robust test. *IEEE/Asme Transactions on Mechatronics*, 13(3):345–354, 2008.

[51] B. Wang, C. Yang, and Q. Xie. Human-machine interfaces based on EMG and Kinect applied to teleoperation of a mobile humanoid robot. In *Proceedings of the 10th World Congress on Intelligent Control and Automation*, pages 3903–3908, July 2012.

[52] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(5):6380–6393, 2013.

[53] D. Wu, L. Pigou, P. J. Kindermans, N. D. H. Le, L. Shao, J. Dambre, and J. M. Odobez. Deep Dynamic Neural Networks for Multimodal Gesture Segmentation and Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(8):1583–1597, August 2016.

[54] Eugene B Zechmeister. Orthographic distinctiveness. *Journal of Verbal Learning and Verbal Behavior*, 8(6):754–761, 1969.

[55] Eugene B Zechmeister. Orthographic Distinctiveness as a Variable in Word Recognition. *The American Journal of Psychology*, 85(3):425–430, 1972.

[56] Hongying Zhang, A Senthil Kumar, Feifei Chen, Jerry YH Fuh, and Michael Yu Wang. Topology optimized multimaterial soft fingers for applications on grippers, rehabilitation, and artificial hands. *IEEE/ASME Transactions on Mechatronics*, 24(1):120–131, 2018.

Appendix A

CASE STUDY ON MARGIN WIDTH FOR ASYMMETRIC CLASSES

A.1 Formulation

Goal: Find an analytic solution to the SVM minimization in a particular structured case:

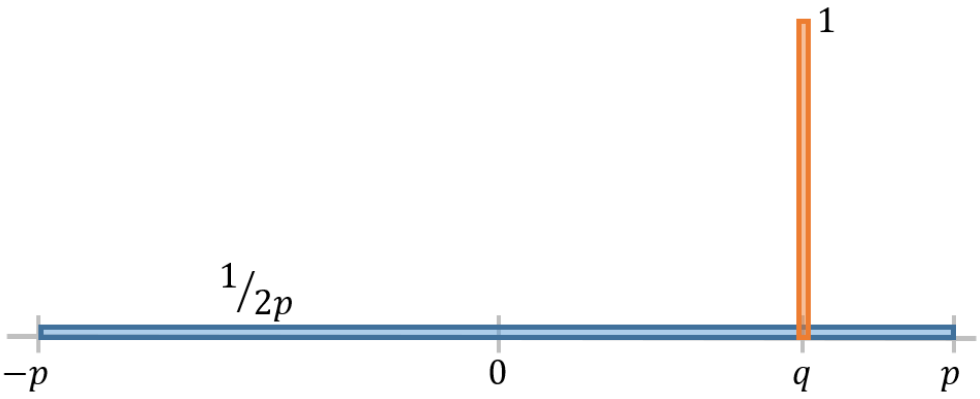


Figure A.1: Two classes of data, one with a uniform distribution between $-p$ and p and one where every point in the class has the value q. This case is referred to as the "uniform-delta case" and is fully described by p and q.

Recall the base support vector machines optimization problem is:

$$\begin{aligned} \underset{a\in\Re^k,\, b\in\Re}{\text{minimize}} \quad & \frac{1}{2}a'a + C\sum_{i=1}\xi_i \\ \text{subject to} \quad & y_i(a'x_i + b) \geq 1 - \xi_i \; i = 1, \ldots, n \\ & \xi_i \geq 0. \end{aligned} \tag{A.1}$$

The first term is the margin term, encoding that the optimal solution will have the widest

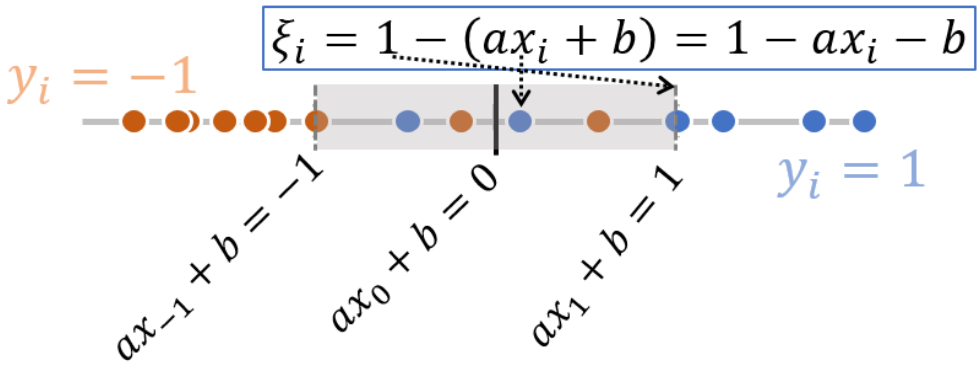


Figure A.2: SVM terms visual explainer for the 1-D enmeshed case

margin possible. The second term deals with any points that violate the margin. Violating the margin does not have to include being over the decision boundary; the point merely needs to be within the margin defined by a and b to be considered as violating the margin. In Fig. 2, any point within the grey region (or outside of it among the bulk of the opposite-colored class) are margin-violating points with an associated nonzero ξ_i. Their ξ^i term is the distance between the outer edge of the margin and the score of the margin-violating point, i.e. $\hat{y_i} = ax_i + b$. As illustrated in Fig. 2: for the i'th point (with $y^i = 1$, i.e. the class on the right), the margin violation distance is $\max(0, 1 - ax_i - b)$, where x_1 is the right margin edge and satisfies $x_1 = \frac{1-b}{a}$.

A.1.1 Conditions on a

For every set of data, there are two optimal solutions for the cases $a > 0$ and $a < 0$. These cases have the same margin ($\pm a$) and in order to restrict to the case where $x_{-1} < x_1$, we restrict a to the nonnegative case. (n.b for $a = 0$, $x_{-1} = x_1$.)

A.2 Objective function

The general objective function to minimize over a and b in terms of p and q is

$$R = \frac{1}{2}a^2 + C\sum \xi_i = \frac{1}{2}a^2 + C\sum_{i=1}^{n} \xi_q + C\sum_{i=1}^{N} \xi_p \tag{A.2}$$

where ξ_q are the margin-violating points in class 2, ξ_p are the margin-violating points in class 1, and N is the number of these points ξ_p. Not every term appears in every region defined by a, b; see section A.3 for details.

For the uniform-delta case in Fig. 1, let class 1 be the blue class ($y = -1$) one with a uniform distribution between $-p$ and p and class 2 be the orange class ($y = 1$), where every point in this class has the value q. Each class has a total of n points in it, and it is assumed that in class 1 there are points at exactly p and $-p$ and the remainder of the class is evenly spread between those two extremes. The spacing between each point is then $\frac{2p}{n-1}$ and the region of interest is of width $p - x_{-1} = p + \frac{1+b}{a}$. Therefore, the number of points N that contribute to $C\sum_{i=1}^{N} \xi_p$ is the width divided by the spacing, or

$$N = \frac{n}{2p}\left(p + \frac{1+b}{a}\right). \tag{A.3}$$

N must be bounded between $[0, n]$ to be a meaningful quantity. Where $\frac{n}{2p}\left(p + \frac{1+b}{a}\right) > n$, there $N = n$, and where $\frac{n}{2p}\left(p + \frac{1+b}{a}\right) < 0$, there $N = 0$.

From numerical results presented in Fig. A.3 on the uniform-delta case with conventional solving of the SVM algorithm, it is known that the optimal margin as a function of p and q in the enmeshed region should be $p - q$, and that the optimal $x_1 = q$, i.e. class 2 does not violate the margin.

The integration-form of this same equation is the same except for the ξ_p term, which instead of being manually summed and bounds put on N, $\xi_{ip} = (ax_i + b) - (-1) = ax_i + b + 1$ is integrated from $\frac{-1-b}{a}$ to p in terms of x and the result multiplied by the point density. For

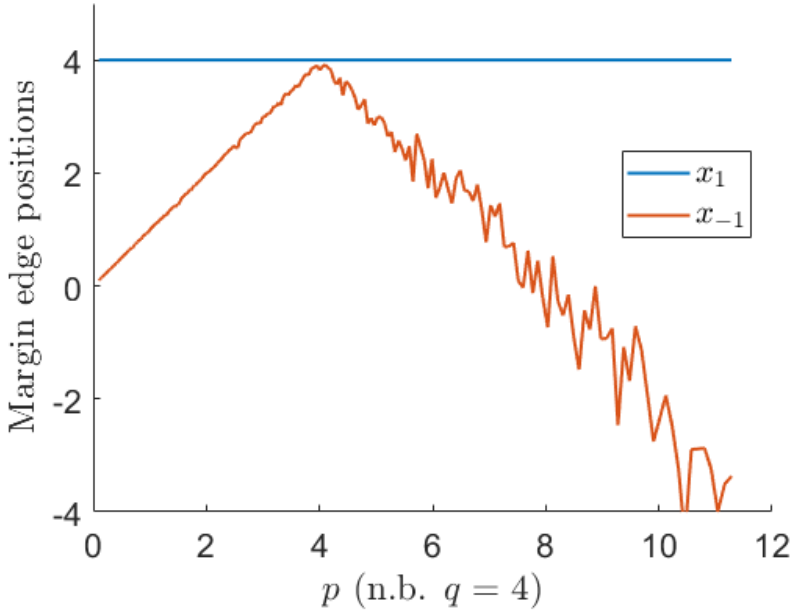


Figure A.3: Optimal margin edges for the uniform-delta case, in one dimension and a population size of 200.

this uniform distribution, the point density is a constant $\frac{n}{2p}$. Numerical validation for the integral approximation of the objective function is present in included scripts.

A.3 Regions of the problem for varying a, b, p, q

For a margin defined by varying values of a, b, the objective function R varies because different margin violation terms are included and the bounds on summation or integration change. (This is implicit within the summation form, though the inclusion of the second term varies.) The integration form of the problem has four regions, potential forms of which are represented in Fig. A.4 and based around the conditions in the previous section. The regions and their integration form objective functions are as follows:

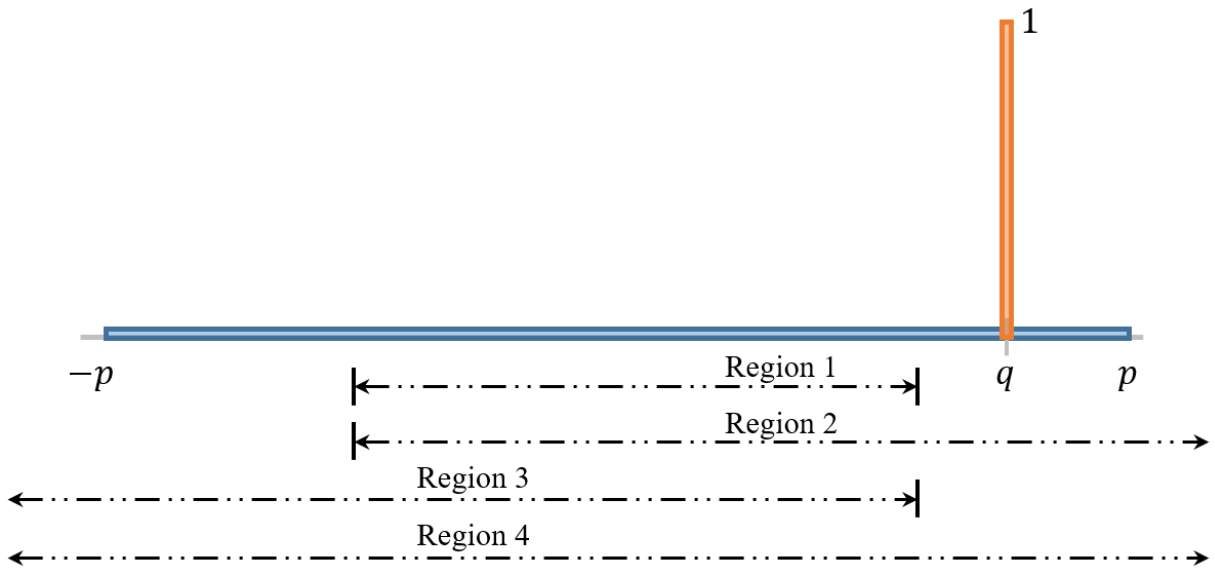


Figure A.4: Representative margins for each of the six possible regions for the integral form of the objective function. There are many possible a, b that satisfy the conditions laid out for each region, but each is subject to the same objective function and the transitions between the objective functions are smooth.

Region 1

Conditions: $x_{-1} > -p\,;\ x_1 < q$

$$\begin{aligned} R_1 &= \frac{1}{2}a^2 + C\int_{x_{-1}}^{p} \frac{n}{2p}(ax+b+1)\,dx \\ &= \frac{1}{2}a^2 + \frac{Cn}{2p}\left[\frac{a}{2}(p^2 - x_{-1}^2) + (b+1)(p - x_{-1})\right] \end{aligned} \tag{A.4}$$

Region 2

Conditions: $x_{-1} > -p\,;\ q < x_1$

$$R_2 = \frac{1}{2}a^2 + Cn(1 - aq - b) + C\int_{x_{-1}}^{p} \frac{n}{2p}(ax+b+1)\,dx \tag{A.5}$$

Region 3

Conditions: $x_{-1} < -p\,;\ x_1 < q$

$$R_3 = \frac{1}{2}a^2 + C\int_{-p}^{p} \frac{n}{2p}(ax + b + 1)\,dx \tag{A.6}$$

Region 4

Conditions: $x_{-1} < -p\,;\ q < x_1$

$$R_4 = \frac{1}{2}a^2 + Cn(1 - aq - b) + C\int_{-p}^{p} \frac{n}{2p}(ax + b + 1)\,dx \tag{A.7}$$

A.4 Minimization for Region 1

Since the region boundaries are in terms of x_1, x_{-1}, change objective function into those variables:

$$\begin{aligned} 1 &= ax_1 + b \\ -1 &= ax_{-1} + b \\ &\implies 2 = a(x_1 - x_{-1}) \\ &\qquad a = \frac{2}{x_1 - x_{-1}} \end{aligned} \tag{A.8}$$

Substituting back in,

$$\begin{aligned} 1 &= \frac{2}{x_1 - x_{-1}} + b \\ b + 1 &= \frac{-2x_{-1}}{x_1 - x_{-1}} \end{aligned} \tag{A.9}$$

Recall that the objective function for this region in the original variables was

$$R_1 = \frac{1}{2}a^2 + \frac{Cn}{2p}\left[\frac{a}{2}(p^2 - x_{-1}^2) + (b+1)(p - x_{-1})\right]. \tag{A.10}$$

Substituting these new variables in,

$$\begin{aligned} R_1 &= \frac{1}{2}a^2 + \frac{Cn}{2p}\left[\frac{a}{2}(p^2 - x_{-1}^2) + (b+1)(p - x_{-1})\right] \\ &= \frac{1}{2}\left(\frac{2}{x_1 - x_{-1}}\right)^2 + \frac{Cn}{2p}\left[\frac{1}{x_1 - x_{-1}}(p^2 - x_{-1}^2) + \frac{-2x_{-1}}{x_1 - x_{-1}}(p - x_{-1})\right] \\ &= \frac{2}{(x_1 - x_{-1})^2} + \frac{Cnp}{2(x_1 - x_{-1})} - \frac{Cnx_{-1}}{(x_1 - x_{-1})} + \frac{Cnx_{-1}^2}{2p(x_1 - x_{-1})}. \end{aligned} \tag{A.11}$$

The partial derivative of this region's objective function with respect to x_1 is

$$\frac{\partial R_1}{\partial x_1} = -\frac{4}{(x_1 - x_{-1})^3} - \frac{Cnp}{2(x_1 - x_{-1})^2} + \frac{Cnx_{-1}}{(x_1 - x_{-1})^2} - \frac{Cnx_{-1}^2}{2p(x_1 - x_{-1})^2}. \tag{A.12}$$

If this derivative is always negative, then the optimal value in this region is at the upper boundary: $x_1 = q$, which is known to be true from numerical results (see Fig. A.3) and is intuitively true, from an examination of the original problem. So now I want to show that $\frac{\partial R_1}{\partial x_1} < 0$. Using the fact that $(x_1 - x_{-1}) > 0$ and factoring out $\frac{1}{(x_1 - x_{-1})^2}$ for simplicity,

$$\begin{aligned} \frac{4}{(x_1 - x_{-1})} + \frac{Cnp}{2} + \frac{Cnx_{-1}^2}{2p} - Cnx_{-1} &> 0 \\ \frac{4}{(x_1 - x_{-1})} + \frac{Cn}{2p}(x_{-1}^2 - 2x_{-1}p - p^2) &> 0 \\ \frac{4}{(x_1 - x_{-1})} + \frac{Cn}{2p}(x_{-1} - p)^2 &> 0 \end{aligned} \tag{A.13}$$

which is true as C, n and p are all positive, and so the optimal value for this region is either q or in Region 2.

A.5 Minimization for Region 2

The objective function for Region 1 and 2 are the same except for the additional term $Cn(1 - aq - b)$. Substituting in the same expressions for a and b, this becomes

$$\begin{aligned} Cn(1 - aq - b) &= Cn(1 - \frac{2}{x_1 - x_{-1}}q - (1 - \frac{2x_1}{x_1 - x_{-1}})) \\ &= \frac{2Cn}{x_1 - x_{-1}}(x_1 - q) \end{aligned} \tag{A.14}$$

and so the complete objective function for R_2 in the new variables is

$$R_2 = \frac{2}{(x_1 - x_{-1})^2} + \frac{Cnp}{2(x_1 - x_{-1})} - \frac{Cnx_{-1}}{(x_1 - x_{-1})} + \frac{Cnx_{-1}^2}{2p(x_1 - x_{-1})} + \frac{2Cn}{x_1 - x_{-1}}(x_1 - q) \quad \text{(A.15)}$$

The partial derivative of this wrt x_{-1} is

$$\begin{aligned} \frac{\partial R_2}{\partial x_{-1}} =& \frac{4}{(x_1 - x_{-1})^3} + \frac{Cnp}{2(x_1 - x_{-1})^2} - \frac{Cnx_{-1}}{(x_1 - x_{-1})^2} - \frac{Cn}{x_1 - x_{-1}} \cdots \\ &+ \frac{Cnx_{-1}^2}{2p(x - x_{-1})^2} + \frac{Cnx_{-1}}{p(x - x_{-1})} + \frac{2Cn}{(x_1 - x_{-1})^2}(x_1 - q) \end{aligned} \quad \text{(A.16)}$$

The partial derivative wrt x_1 is

$$\begin{aligned} \frac{\partial R_2}{\partial x_1} =& - \frac{4}{(x_1 - x_{-1})^3} - \frac{Cnp}{2(x_1 - x_{-1})^2} + \frac{Cnx_{-1}}{(x_1 - x_{-1})^2} - \frac{Cnx_{-1}^2}{2p(x_1 - x_{-1})^2} \cdots \\ &+ \frac{2Cn}{(x_1 - x_{-1})^2}(q - x_1) + \frac{2Cn}{(x_1 - x_{-1})} \end{aligned} \quad \text{(A.17)}$$

Numerical investigation of this objective function and its derivatives show that the optimal value for this region is also at q, but the multivariate optimization proved intractable in the time frame dedicated to its solution. After showing that the optimal value for this region is at the boundary (q), next steps would be to show that $\frac{\partial R_3}{\partial x_{-1}} < 0$ and $\frac{\partial R_4}{\partial x_{-1}} < 0$, ruling out the optimal a, b being in Regions 3 or 4. Finally, the optimal $x_1^* = q$ would be substituted into R_2 and minimized to show that $x_{-1}^* = 2q - p$.

Appendix B

NUMERICAL RESULTS FOR REGIONAL INVERSION WITH VARYING DIMENSIONALITY AND DISTRIBUTION TYPE AND SELECTION OF 1% INVERSION THRESHOLD

B.1 Margin edge results

Presented in this appendix is results that show that the margin edges contract and then expand when data moves from separable to inseparable, as well as the code used to generate each result.

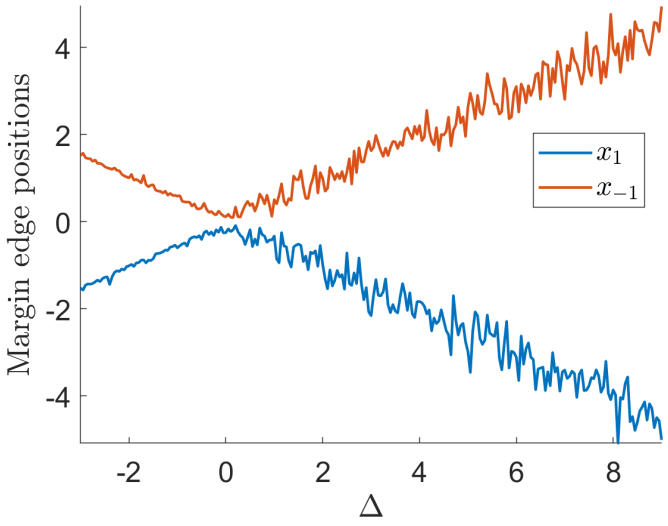


Figure B.1: Margin edge positions for the symmetric uniform case.

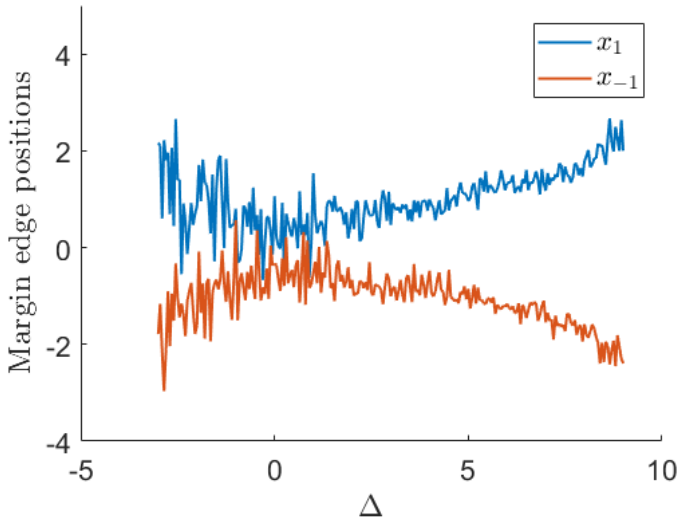


Figure B.2: Margin edge positions for symmetric Gaussian distributions

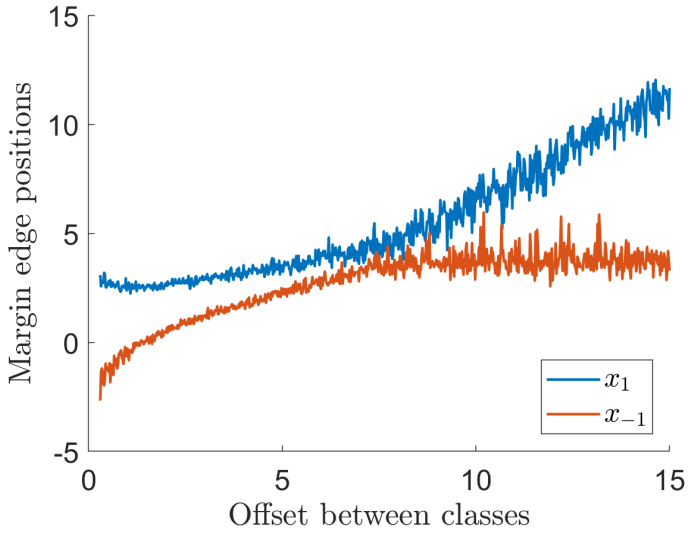


Figure B.3: Margin edge positions for symmetric bimodal Gaussian distributions with an increasing offset from each other.

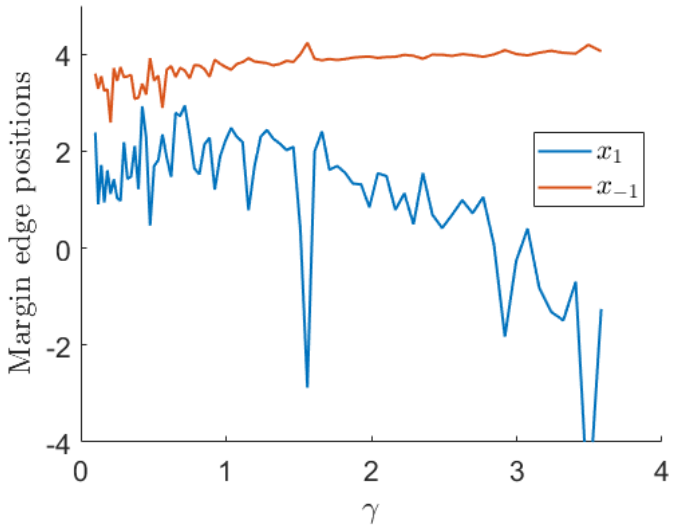
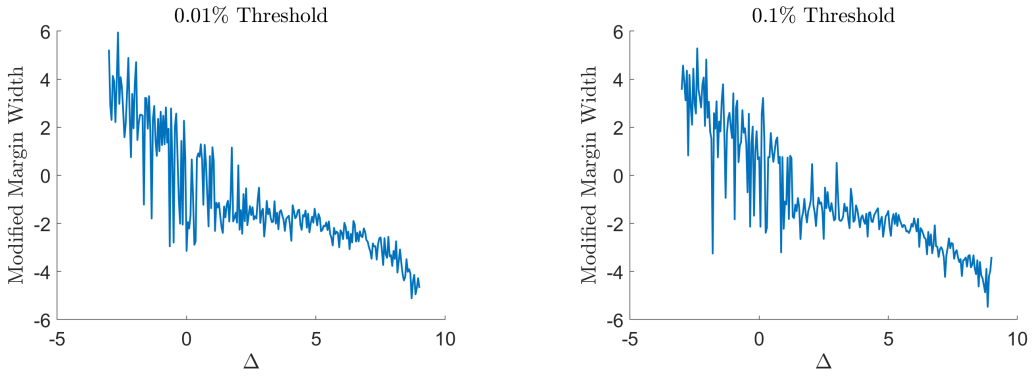


Figure B.4: Margin edge positions for stable distributions distributions with an increasing offset from each other. Independent variable is γ parameter of one distribution. Other parameters are $\alpha = 1.7$, $\beta = 0$, γ of the first distribution 0.1, and $\delta_1 = 0$, $\delta_2 = 4$.

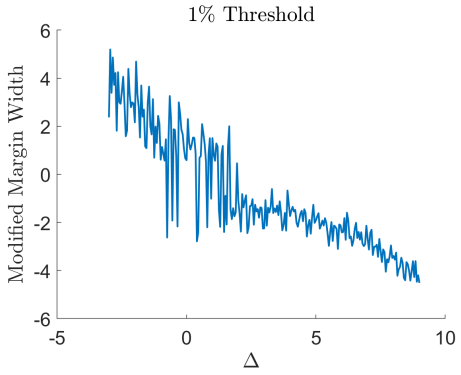
B.2 Selection of 1% threshold

Shown in §B.1, for cases not involving a uniform distribution the transition region between the contracting and expanding margin is larger, raising the question of when exactly when applied to data the decision should be made to consider the inter-class distance the negative margin. 1% in-sample error was chosen by examining the effect of different cutoff thresholds for the cases presented in §B.1. The desired outcome is to choose a cutoff that achieves the most smoothly linear relationship between the varied parameter of that distribution type (on the x-axis; Δ in the double uniform distribution case) and the modified margin width.

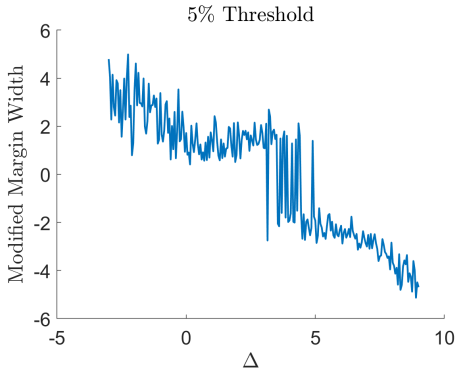


(a) 0.01% training error threshold.

(b) 0.1% training error threshold.

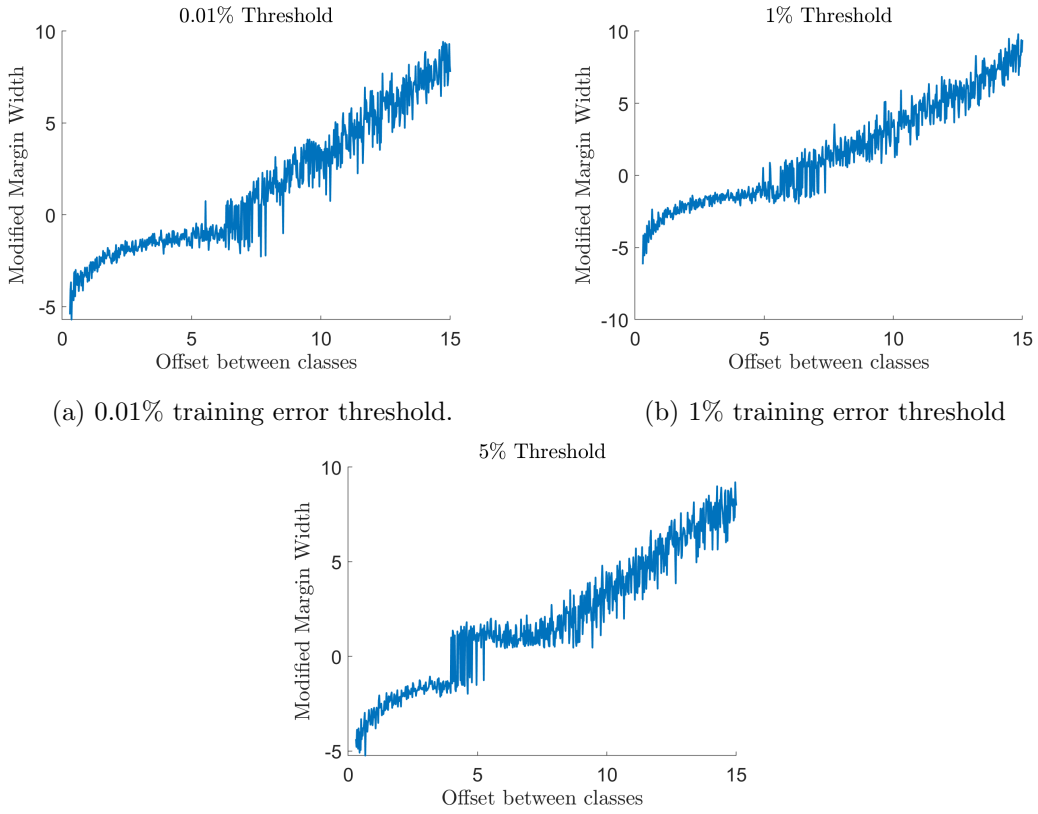


(c) 1% training error threshold



(d) 5% training error threshold

Figure B.5: The modified margin width for different thresholds of in-sample error in training data for two Gaussian distributions. The margin edge positions for a similar case are visualized in Fig. B.2. Population sizes are 100 iterations per class, which is approximately an upper limit on how many iterations there are in the gesture classes of interest. The goal is to achieve the most smoothly linear relationship between the varied parameter Δ and the modified margin width.

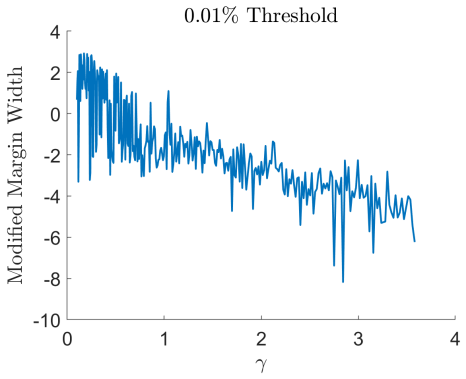


(a) 0.01% training error threshold.

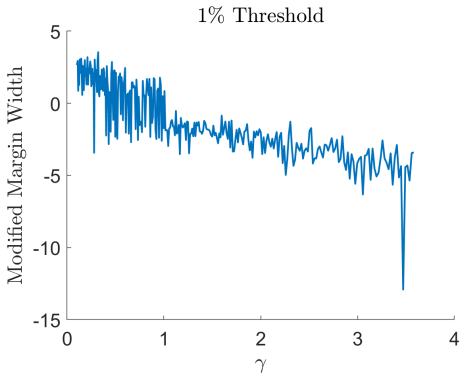
(b) 1% training error threshold

(c) 5% training error threshold

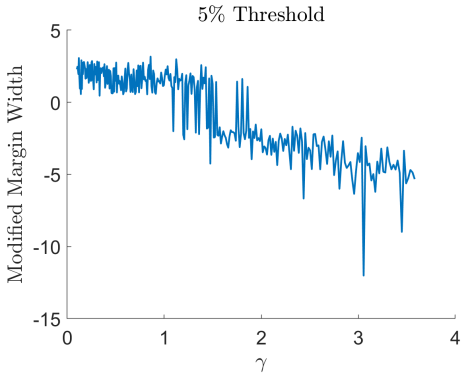
Figure B.6: The modified margin width for different thresholds of in-sample error in training data for two bimodal Gaussian distributions. The margin edge positions for a similar case are visualized in Fig. B.3. Population sizes are 100 iterations per class, which is approximately an upper limit on how many iterations there are in the gesture classes of interest. The goal is to achieve the most smoothly linear relationship between the varied parameter offset between classes and the modified margin width.



(a) 0.01% training error threshold.

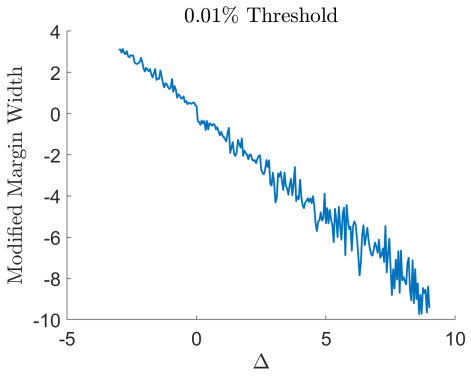


(b) 1% training error threshold

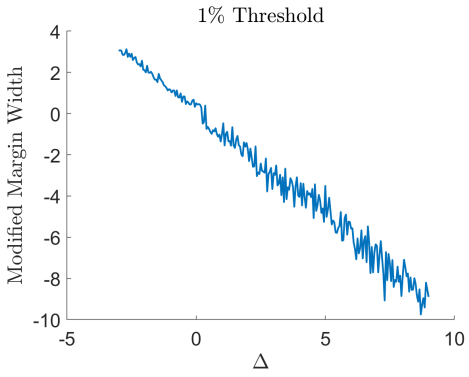


(c) 5% training error threshold

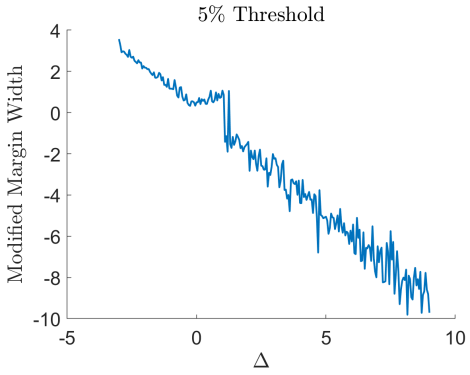
Figure B.7: The modified margin width for different thresholds of in-sample error in training data for two stable (heavy-tailed, otherwise similar to Gaussian) distributions. The margin edge positions for a similar case are visualized in Fig. B.4. Population sizes are 100 iterations per class, which is approximately an upper limit on how many iterations there are in the gesture classes of interest. The goal is to achieve the most smoothly linear relationship between the varied parameter γ and the modified margin width.



(a) 0.01% training error threshold.



(b) 1% training error threshold



(c) 5% training error threshold

Figure B.8: The modified margin width for different thresholds of in-sample error in training data for two uniform distributions, as discussed extensively in Chapter 4. The margin edge positions for a similar case are visualized in Fig. B.1. Population sizes are 100 iterations per class, which is approximately an upper limit on how many iterations there are in the gesture classes of interest. The goal is to achieve the most smoothly linear relationship between the varied parameter Δ and the modified margin width.

Across the considered test distribution types, a cutoff of 1% broadly achieved the best management of the transition region to achieve a globally linear relationship between the varied parameters and the margin width. However, this cutoff is something that could be revisited in light of possible assumptions on true data distributions or very different amounts of training data, as 1% is not necessarily a universally applicable choice.

B.3 Script to generate results

```
% Rose Hendrix
% 4-30-2019
% script to test all candidate measures at different population
    sizes and
% distribution types

set(0,'DefaultTextInterpreter','Latex',...
    'DefaultLegendInterpreter','Latex','DefaultAxesFontSize',16,'
       DefaultTextFontSize',16,...
    'defaultlinemarkersize',16)

clearvars; close all

addpath ../helper-functions
addpath ../measures

CaseName = 'double uniform'; % Put the section name here to run

switch CaseName
    case 'uniform-delta'
        fprintf('Calculating margin edges for the uniform-delta
           case.\n')
        types = repmat([2 4],1,1);

        % uniform-delta distribution parameters
```

```
23          q = 4; maxp = 10; minp = 0.1; qspread = 0.05;
24          P = exp(0:.01:2.5) -0.9;
25          numcases = length(P);
26          xlabelstring = '$p$ (n.b. $q=4$)';
27      case 'double uniform'
28          fprintf('Calculating margin edges for the double uniform
                case.\n')
29          types = repmat([2  2],1,1);
30          % double uniform  or double gaussian distribution
                parameters
31          deltas=-3:.05:9;
32          P=deltas;
33          W = 10;
34          if types(1) ==6 ||types(1)==2
35              inputparam = W/2; % uniform distributions
36          else
37              inputparam = W/3; % NOTE: in this formulation W = 3*
                    sigma for Gaussian
38          end
39          center1=(W - deltas) ./2;
40          center2=-1.*center1;
41          numcases=length(deltas);
42          xlabelstring = '$\Delta$';
43      case 'double Gaussian'
44          fprintf('Calculating margin edges for the double Gaussian
                case.\n')
```

```
45          types = repmat([1  1],1,1);
46          % double uniform  or double gaussian distribution
                parameters
47          deltas=-3:.05:9;
48          P=deltas;
49          W = 10;
50          if types(1) ==6 ||types(1)==2
51              inputparam = W/2; % uniform distributions
52          else
53              inputparam = W/3; % NOTE: in this formulation W = 3*
                    sigma for Gaussian
54          end
55          center1=(W - deltas) ./2;
56          center2=-1.*center1;
57          numcases=length(deltas);
58          xlabelstring = '$\Delta$';
59      case 'bimodal Gaussian'
60          fprintf('Calculating margin edges for the bimodal Gaussian
                case.\n')
61          types = repmat([3  3],1,1);
62
63          % bimodal distribution case parameters
64          bimodaloffsetarray = 15:-.02:0.3;
65          mu1 = 0;
66          sig1 = 2;
67          mu2 = 1;
```

```
68          sig2 = 1;
69          numcases = length(bimodaloffsetarray);
70          P=bimodaloffsetarray;
71          xlabelstring = 'Offset between classes';
72      case 'stable'
73          fprintf('Calculating margin edges for the stable case.\n')
74          types = repmat([5 5],1,1);
75          % stable distribution parameters
76          alph = 1.7; %
77          bet = 0; %
78          G = exp(0:0.02:1.5)-0.9;
79          g2 = 0.1;
80          delta1 = 0; % center of class1
81          delta2 = 4; % center of class2
82          numcases = length(G);
83          P=G;
84          xlabelstring = '$\gamma$';
85      otherwise
86          error('This is not a valid case. Please edit and try again
                .\n');
87  end
88
89  % 1. Gaussian, shape params are mu, sigma^2.
90  % 2. Uniform, shape params are center, spread
91  % 3. Bimodal Gaussian, shape params are (mu1, sig1, mu2, sig2)
92  % 4. Delta, shape param is location
```

```
93 % 5. Stable. shape params are alpha, beta, gamma, delta
94 % 6. Dummy Uniform (evenly sampled). shape params are center,
       spread
95
96 minsample = 10;
97 idx = [200];% round([ (20:2:30).^2 + 1]);
98 popcap = idx(end);
99
100 numfeatures = 1*ones(1,1);
101 numruns = 1;
102
103 for nn = 1:numcases
104 switch CaseName
105     case 'uniform-delta'
106         p = P(nn)*ones(popcap,1);
107         g = p;
108         shapeparams(:,2,1,1:max(numfeatures)) = q*ones(popcap,1,1,
              max(numfeatures));
109         shapeparams(:,2,2,1:max(numfeatures)) = qspread*ones(
              popcap,1,1,max(numfeatures));
110         shapeparams(:,1,1,1:max(numfeatures)) = 0*ones(popcap,1,1,
              max(numfeatures));
111         shapeparams(:,1,2,1:max(numfeatures)) = repmat(p,[1 1 1
              max(numfeatures)]);
112     case 'double uniform'
113         % double uniform or double gaussian
```

```
114         c2=center2(nn)*ones(popcap,1);
115         c1=center1(nn)*ones(popcap,1);
116
117         shapeparams(:,2,1,1:max(numfeatures)) = repmat(c2,[1 1 1
                max(numfeatures)]);
118         shapeparams(:,2,2,1:max(numfeatures)) = inputparam*ones(
                popcap,1,1,max(numfeatures));
119         shapeparams(:,1,1,1:max(numfeatures)) = repmat(c1,[1 1 1
                max(numfeatures)]);
120         shapeparams(:,1,2,1:max(numfeatures)) = inputparam*ones(
                popcap,1,1,max(numfeatures));
121     case 'double Gaussian'
122         % double uniform or double gaussian
123         c2=center2(nn)*ones(popcap,1);
124         c1=center1(nn)*ones(popcap,1);
125
126         shapeparams(:,2,1,1:max(numfeatures)) = repmat(c2,[1 1 1
                max(numfeatures)]);
127         shapeparams(:,2,2,1:max(numfeatures)) = inputparam*ones(
                popcap,1,1,max(numfeatures));
128         shapeparams(:,1,1,1:max(numfeatures)) = repmat(c1,[1 1 1
                max(numfeatures)]);
129         shapeparams(:,1,2,1:max(numfeatures)) = inputparam*ones(
                popcap,1,1,max(numfeatures));
130     case 'bimodal Gaussian'
131         % bimodal gaussian
```

```
132         bimodaloffset = bimodaloffsetarray(nn);
133
134         shapeparams(:,1,1,1:max(numfeatures)) = mu1*ones(popcap
                ,1,1,max(numfeatures));
135         shapeparams(:,1,2,1:max(numfeatures)) = sig1*ones(popcap
                ,1,1,max(numfeatures));
136         shapeparams(:,1,3,1:max(numfeatures)) = repmat(mu2,[1 1 1
               max(numfeatures)]);
137         shapeparams(:,1,4,1:max(numfeatures)) = repmat(sig2,[1 1 1
                max(numfeatures)]);
138
139         shapeparams(:,2,1,1:max(numfeatures)) = (bimodaloffset+mu1
                )*ones(popcap,1,1,max(numfeatures));
140         shapeparams(:,2,2,1:max(numfeatures)) = sig1*ones(popcap
                ,1,1,max(numfeatures));
141         shapeparams(:,2,3,1:max(numfeatures)) = bimodaloffset+
                repmat(mu2,[1 1 1 max(numfeatures)]);
142         shapeparams(:,2,4,1:max(numfeatures)) = repmat(sig2,[1 1 1
                max(numfeatures)]);
143
144     case 'stable'
145         % stable
146         g = G(nn)*ones(popcap,1);
147         shapeparams(:,1,1,1:max(numfeatures)) = alph*ones(popcap
                ,1,1,max(numfeatures));
148         shapeparams(:,1,2,1:max(numfeatures)) = bet*ones(popcap
```

```
        ,1,1,max(numfeatures));
149     shapeparams(:,1,3,1:max(numfeatures)) = repmat(g,[1 1 1
            max(numfeatures)]);
150     shapeparams(:,1,4,1:max(numfeatures)) = delta1*ones(popcap
            ,1,1,max(numfeatures));
151
152     shapeparams(:,2,1,1:max(numfeatures)) = alph*ones(popcap
            ,1,1,max(numfeatures));
153     shapeparams(:,2,2,1:max(numfeatures)) = bet*ones(popcap
            ,1,1,max(numfeatures));
154     shapeparams(:,2,3,1:max(numfeatures)) = g2*ones(popcap
            ,1,1,max(numfeatures));
155     shapeparams(:,2,4,1:max(numfeatures)) = delta2*ones(popcap
            ,1,1,max(numfeatures));
156 otherwise
157     error('This is not a valid case. Please edit and try again
            .\n');
158 end
159
160 populations = [popcap popcap];
161 testpopulation = [1000 1000]; % always test with 1k
162
163 %% Generate params object, generate data, models, and test
        performance
164 dataparams = generateDataParams(1,numfeatures,types,
        shapeparams,populations);
```

```
165     testdataparams = generateDataParams(1,numfeatures,types,
            shapeparams,testpopulation);
166     boxconstraint =10;
167
168     for jj = 1:numruns
169         TD(nn) = generateData(dataparams); % NOTE: does not work
                properly for type 6
170         TD2(nn) = generateData(testdataparams);
171
172         for ii = idx % fix this for speed
173             trainingData(ii,jj).x1 = TD(nn).x1(1:ii,:);
174             trainingData(ii,jj).y1 = TD(nn).y1(1:ii);
175             trainingData(ii,jj).x2 = TD(nn).x2(1:ii,:);
176             trainingData(ii,jj).y2 = TD(nn).y2(1:ii);
177             testingData(ii,jj).x1 = TD2(nn).x1(1:ii,:);
178             testingData(ii,jj).y1 = TD2(nn).y1(1:ii);
179             testingData(ii,jj).x2 = TD2(nn).x2(1:ii,:);
180             testingData(ii,jj).y2 = TD2(nn).y2(1:ii);
181
182         end
183         %     fprintf('Done making the data sets. \n');
184
185         %% Generate models - can sub in any solver
186         [models(nn,:),scores(:,nn,:),aEff,bEff,trainacc(:,jj,:)] =
                generateModelsInbuilt(trainingData(idx,jj),
              boxconstraint);
```

```
187
188         psi(:,2,jj) = psi_margin_2region(models(nn,:),scores(:,nn
                ,:));
189         psi(:,1,jj) = psi_margin(models(nn,:));
190
191         % calculate test accuracy
192         testacc(nn,jj) = testAccuracy(models(nn,:),testingData(idx
                ,jj));
193
194     end
195     % uncomment this to visualize data
196     % figure; histogram(trainingData(end,end).x1); hold on;
            histogram(trainingData(end,end).x2);
197     psiavg = nanmean(psi,3);
198     final_psi(nn,:,:)= psiavg;
199 end
200
201 %% visualize
202 for ii = 1:size(psi,2)
203     figure;
204     hold on
205     for jj = 1:length(idx)
206         plot(P,final_psi(:,jj,ii))
207     end
208     xlabel(xlabelstring);
209     if ii==2
```

```
210         ylabel('Modified Margin Width')
211     else
212         ylabel('Margin Width');
213     end
214 end
215
216 % flipped and unflipped vs acc
217 figure;
218 hold on;
219 plot(final_psi(:,1,1),testacc,'.');
220 plot(final_psi(:,1,2),testacc,'.');
221 legend('Without Inversion','With Inversion','Location','best');
222 xlabel('Separabiity'); ylabel('Test Accuracy');
223
224
225 for nn=1:size(models,1)
226     a(nn) = models(nn).a;
227     b(nn) = models(nn).b;
228     xRight(nn) = (1-models(nn).b)/models(nn).a;
229     xLeft(nn) = (-1-models(nn).b)/models(nn).a;
230 end
231 figure;
232 hold on;
233 plot(P,xRight,'LineWidth',1.5);
234 plot(P,xLeft,'LineWidth',1.5);
235 xlabel(xlabelstring); ylabel('Margin edge positions');
```

```
236 legend('$x_1$','$x_{-1}$','Location','best','FontSize',16)
237 axis tight
```

Appendix C

FEATURE SELECTION ALGORITHM IMPLEMENTATION

C.1 Algorithm summary

An overview of the code for comparing different methods of feature selection follows.

1. For every gesture set of interest:
    (a) Compute the full-featured orthogonal feature representation
    (b) While total number of features is above some preset final count:
        i. Compute SVM models for every gesture pair in the set
        ii. Calculate test performance of set for current feature representation
        iii. Calculate importance criterion for each feature as appropriate for the current method
        iv. Decide which feature(s) to remove based on this performance criterion
        v. Remove those features and repeat until done.
    (c) Save results for that method and gesture set

This is executed for all three methods of recursive feature evaluation as independent recursions. PCA is evaluated alongside the Modified Chapelle method, as it is not recursive and does not need its own loop.

C.2 Code

```
% Rose Hendrix
% 06FEB2020
% reduce representation by PCA and by various recursive methods

clearvars; close all;

set(0,'DefaultTextInterpreter','Latex',...
    'DefaultLegendInterpreter','Latex',...
    'DefaultTextFontSize',12,...
    'DefaultAxesFontSize',12,...
    'DefaultLineLineWidth',1.5,...
    'DefaultLineMarkerSize',7.75)

addpath ../../helper_functions/
addpath ../../measures/

load ../../helper_functions/options
load ../../data/Features.mat
ff=1;
% apply orthogonalization to individual gestures
trainX_all_modes = NaN(size(Features(ff).X_norm));
testX_all_modes = NaN(size(TestFeatures(ff).X_all));
for gesidx = 1:size(Features(ff).X,3)
    trainX_all_modes(:,:,gesidx) = Features(ff).U(:,1:end).' * (
        Features(ff).X_norm(:,:,gesidx));
    testX_all_modes(:,:,gesidx) = Features(ff).U(:,1:end).' * (
```

```
        TestFeatures(ff).X_all(:,:,gesidx));
end

% most separable according to k-fold validation results, hardcoded
list_of_control_pairs = [1 17; 4 17; 5 17; 6 17; 6 26; 6 28; 14
    17; 17 24; 17 26; 17 28];
% least separable gesture pair for comparison:
    list_of_control_pairs = [11 12];

frac_to_remove = 10; % 1/frac_to_remove is the actual fraction
totalloopnum = size(Features(ff).X,1); % preallocate as if single-
    elimination

for pairidx = 1:size(list_of_control_pairs,1)
    controlgests = list_of_control_pairs(pairidx,:);%[17 28];
    W = optn.workLocation;

    % for 1:end of ordered modes, zero out the unused features,
        reconstruct
    % X with the irrelevant features zeroed out, and test
        performance

    %% Modified Chapelle method and PCA
    currentfeat = 1:size(Features(ff).X,1);
    acc_sep = NaN(1,length(currentfeat));
    feats_kept = NaN(totalloopnum,np);
```

```
46      loopnumber = 1;
47      finalfeat = 1;
48      while length(currentfeat) >= finalfeat
49          clear trainX_red_sep_reduced testX_red_sep_reduced
               trainX_red_pca_reduced testX_red_pca_reduced
               importance_crit
50          testgestures = [controlgests W];
51
52          for g = testgestures
53              trainX_red_sep_reduced(:,:,g) = trainX_all_modes(
                   currentfeat,:,g);
54              testX_red_sep_reduced(:,:,g) = testX_all_modes(
                   currentfeat,:,g);
55          end
56
57          % models
58          [M,S] = mySVMTrain_Modified(trainX_red_sep_reduced ,
               testgestures);%
59          pairlist = nchoosek(testgestures,2);
60          for ii = 1:size(pairlist,1)
61              modellist(ii) = M(pairlist(ii,1),pairlist(ii,2));
62          end
63
64          %% calculate perf for current set
65          yhat_sep = [];
66          y_sep = [];
```

```
67
68          for gt = 1:length(testgestures)
69              % Define a class index to test
70              testClassIndex = testgestures(gt);
71
72              % Number of tests
73              Ntest = run_idx(testClassIndex)-1;
74
75              [pred_sep,testY_sep] = mySVMTest_Modified(
                    testX_red_sep_reduced(:,1:Ntest,testClassIndex),...
76                   pairlist, testClassIndex, modellist);
77
78              pred_sep(pred_sep>optn.sizeC) = optn.sizeC+1;
79              testY_sep(testY_sep>optn.sizeC) = optn.sizeC+1;
80
81              yhat_sep(end+1:end+Ntest) = pred_sep;
82              y_sep(end+1:end+Ntest) = testY_sep;
83          end
84          acc_sep(length(currentfeat)) = 1 - (nnz(yhat_sep - y_sep)
                / length(yhat_sep));
85
86          %% make decisions about what features to keep with
                importance criterion
87          % find sep - both types
88          [m,n] = size(M);
89          StandardSep = nan(size(M));
```

```
90          FeatLevelSep = nan([size(M) length(currentfeat)]);
91          for ii =1:m
92              for jj = 1:n
93                  if isempty(M(ii,jj).a)
94                      continue;
95                  end
96                  StandardSep(ii,jj) = psi_margin_2region(M(ii,jj),S
                        (ii,jj));
97                  for featiter = 1:length(currentfeat)
98                      %NEW - dot product, works because features
                            are orthogonal
99                      FeatLevelSep(ii,jj,featiter) = abs(M(ii,jj).a(
                            featiter) / norm(M(ii,jj).a));
100                 end
101             end
102         end
103
104         tmpimportancrit = zeros(length(currentfeat),1);
105         votingpairs = identifyRelevantModels(controlgests,W,
                pairlist,StandardSep);
106         for ii=1:length(votingpairs)
107             currentpair = pairlist(votingpairs(ii),:);
108             featvec = squeeze(FeatLevelSep(currentpair(1),
                    currentpair(2),:));
109             tmpimportancrit = [tmpimportancrit (featvec)];
110         end
```

```
111         importance_crit = sum(tmpimportancrit,2);
112
113         % if all features have attained the minimum importandce
                level, stop
114         if 0 % possible stopping criterion goes here
115             finalfeat = length(currentfeat);
116             break;
117         else
118             % drop the ones below loopnumber * .001
119             tokeep = find(importance_crit > (loopnumber*0.01));
120             if length(tokeep) == length(currentfeat)
121                 tokeep = tokeep(~(importance_crit == min(
                        importance_crit)));
122             end
123         end
124
125         feats_kept(loopnumber,1:length(tokeep)) = currentfeat(
                tokeep);
126         currentfeat = currentfeat(tokeep);
127         loopnumber = loopnumber + 1;
128     end
129     acc_sep_sepleveldotprod = acc_sep;
130     feats_kept_sepleveldotprod = feats_kept;
131
132     %% Feature-Level Separability - Dot Product
133     currentfeat = 1:size(Features(ff).X,1);
```

```
134     acc_sep = NaN(1,length(currentfeat));
135     acc_pca = NaN(1,length(currentfeat));
136     feats_kept = NaN(totalloopnum,np);
137     loopnumber = 1;
138     while length(currentfeat) >= finalfeat
139         clear trainX_red_sep_reduced testX_red_sep_reduced
              trainX_red_pca_reduced testX_red_pca_reduced
              importance_crit
140         testgestures = [controlgests W];
141
142         for g = testgestures
143             trainX_red_sep_reduced(:,:,g) = trainX_all_modes(
                  currentfeat,:,g);
144             testX_red_sep_reduced(:,:,g) = testX_all_modes(
                  currentfeat,:,g);
145             trainX_red_pca_reduced(:,:,g) = trainX_all_modes(1:
                  length(currentfeat),:,g);
146             testX_red_pca_reduced(:,:,g) = testX_all_modes(1:
                  length(currentfeat),:,g);
147         end
148
149         % models
150         [M,S] = mySVMTrain_Modified(trainX_red_sep_reduced ,
              testgestures);
151         [M2_pca,~] = mySVMTrain_Modified(trainX_red_pca_reduced ,
              testgestures);
```

```
152
153
154
155         pairlist = nchoosek(testgestures,2);
156         for ii = 1:size(pairlist,1)
157             modellist(ii) = M(pairlist(ii,1),pairlist(ii,2));
158             modellist_pca(ii) = M2_pca(pairlist(ii,1),pairlist(ii
                    ,2));
159         end
160
161         %% calculate perf for current set
162         yhat_sep = [];
163         y_sep = [];
164         yhat_pca = [];
165         y_pca = [];
166
167         for gt = 1:length(testgestures)
168             % Define a class index to test
169             testClassIndex = testgestures(gt);
170
171             % Number of tests
172             Ntest = run_idx(testClassIndex)-1;
173
174             [pred_sep,testY_sep] = mySVMTest_Modified(
                    testX_red_sep_reduced(:,1:Ntest,testClassIndex),...
175                  pairlist, testClassIndex, modellist);
```

```
176
177             pred_sep(pred_sep>optn.sizeC) = optn.sizeC+1;
178             testY_sep(testY_sep>optn.sizeC) = optn.sizeC+1;
179
180             yhat_sep(end+1:end+Ntest) = pred_sep;
181             y_sep(end+1:end+Ntest) = testY_sep;
182
183             [pred_pca,testY_pca] = mySVMTest_Modified(
                    testX_red_pca_reduced(:,1:Ntest,testClassIndex),...
184                  pairlist, testClassIndex, modellist_pca);
185
186             pred_pca(pred_pca>optn.sizeC) = optn.sizeC+1;
187             testY_pca(testY_pca>optn.sizeC) = optn.sizeC+1;
188
189             yhat_pca(end+1:end+Ntest) = pred_pca;
190             y_pca(end+1:end+Ntest) = testY_pca;
191
192         end
193         acc_sep(length(currentfeat)) = 1 - (nnz(yhat_sep - y_sep)
                / length(yhat_sep));
194         acc_pca(length(currentfeat)) = 1 - (nnz(yhat_pca - y_pca)
                / length(yhat_pca));
195
196         %% make decisions about what features to keep with
                importance criterion
197         % find sep - both types
```

```
198         [m,n] = size(M);
199         StandardSep = nan(size(M));
200         FeatLevelSep = nan([size(M) length(currentfeat)]);
201         for ii =1:m
202             for jj = 1:n
203                 if isempty(M(ii,jj).a)
204                     continue;
205                 end
206                 StandardSep(ii,jj) = psi_margin_2region(M(ii,jj),S
                        (ii,jj));
207                 for featiter = 1:length(currentfeat)
208                     %                   FeatlevelSep(ii,jj,featiter)
                            = norm(M(ii,jj).a(featiter));
209                     % NEW - dot product, works because features
                            are already
210                     % orthogonal
211                     FeatLevelSep(ii,jj,featiter) = abs(M(ii,jj).a(
                            featiter) / norm(M(ii,jj).a));
212                 end
213             end
214         end
215
216         importance_crit = zeros(length(currentfeat),1);
217         for ii=1:length(pairlist)
218             currentpair = pairlist(ii,:);
219             featvec = squeeze(FeatLevelSep(currentpair(1),
```

```
                currentpair(2),:));
220             importance_crit = importance_crit + featvec;
221         end
222
223         [~,critorder] = sort(importance_crit,'descend');
224         cutoff = length(critorder) - floor(length(critorder)/
                frac_to_remove);
225         r = 1:length(critorder);
226         r(critorder) = r;
227         tokeep = find(r<cutoff);
228         feats_kept(loopnumber,1:length(tokeep)) = currentfeat(
                tokeep);
229         currentfeat = currentfeat(tokeep);
230         loopnumber = loopnumber + 1;
231
232     end
233     acc_pca_mine = acc_pca;
234     acc_dotprod = acc_sep;
235     feats_kept_dotprod = feats_kept;
236
237
238     %% Regular Chapelle/Keerthi recursion
239     currentfeat = 1:size(Features(ff).X,1);
240     acc_sep = NaN(1,length(currentfeat));
241     acc_pca = NaN(1,length(currentfeat));
242     feats_kept = NaN(totalloopnum,np);
```

```
243     loopnumber = 1;
244     while length(currentfeat) >= finalfeat
245         clear trainX_red_sep_reduced testX_red_sep_reduced
              trainX_red_pca_reduced testX_red_pca_reduced
              importance_crit
246         testgestures = [controlgests W];
247
248         for g = testgestures
249             trainX_red_sep_reduced(:,:,g) = trainX_all_modes(
                  currentfeat,:,g);
250             testX_red_sep_reduced(:,:,g) = testX_all_modes(
                  currentfeat,:,g);
251             trainX_red_pca_reduced(:,:,g) = trainX_all_modes(1:
                  length(currentfeat),:,g);
252             testX_red_pca_reduced(:,:,g) = testX_all_modes(1:
                  length(currentfeat),:,g);
253         end
254
255         % models
256         [M,S] = mySVMTrain_Modified(trainX_red_sep_reduced ,
              testgestures);
257         [M2_pca,~] = mySVMTrain_Modified(trainX_red_pca_reduced ,
              testgestures);
258
259         pairlist = nchoosek(testgestures,2);
260         for ii = 1:size(pairlist,1)
```

```
261             modellist(ii) = M(pairlist(ii,1),pairlist(ii,2));
262             modellist_pca(ii) = M2_pca(pairlist(ii,1),pairlist(ii
                    ,2));
263         end
264
265         %% calculate perf for current set
266         yhat_sep = [];
267         y_sep = [];
268         yhat_pca = [];
269         y_pca = [];
270
271         for gt = 1:length(testgestures)
272             % Define a class index to test
273             testClassIndex = testgestures(gt);
274
275             % Number of tests
276             Ntest = run_idx(testClassIndex)-1;
277
278             [pred_sep,testY_sep] = mySVMTest_Modified(
                    testX_red_sep_reduced(:,1:Ntest,testClassIndex),...
279                  pairlist, testClassIndex, modellist);
280
281             pred_sep(pred_sep>optn.sizeC) = optn.sizeC+1;
282             testY_sep(testY_sep>optn.sizeC) = optn.sizeC+1;
283
284             yhat_sep(end+1:end+Ntest) = pred_sep;
```

```
285             y_sep(end+1:end+Ntest) = testY_sep;
286
287             [pred_pca,testY_pca] = mySVMTest_Modified(
                    testX_red_pca_reduced(:,1:Ntest,testClassIndex),...
288                  pairlist, testClassIndex, modellist_pca);
289
290             pred_pca(pred_pca>optn.sizeC) = optn.sizeC+1;
291             testY_pca(testY_pca>optn.sizeC) = optn.sizeC+1;
292
293             yhat_pca(end+1:end+Ntest) = pred_pca;
294             y_pca(end+1:end+Ntest) = testY_pca;
295
296         end
297         acc_sep(length(currentfeat)) = 1 - (nnz(yhat_sep - y_sep)
                / length(yhat_sep));
298         acc_pca(length(currentfeat)) = 1 - (nnz(yhat_pca - y_pca)
                / length(yhat_pca));
299
300
301         %% base alg: rank according to summed squared model
                weights
302         for fiter = 1:length(currentfeat)
303             tmpimport = 0;
304             % for every pair
305             for ii = 1:size(pairlist,1)
306                 tmpimport = tmpimport + (M(pairlist(ii,1),pairlist
```

```
                    (ii,2)).a(fiter))^2;
307             end
308             importance_crit(fiter) = tmpimport;
309         end
310
311         [~,critorder] = sort(importance_crit,'descend');
312         cutoff = length(critorder) - floor(length(critorder)/
                frac_to_remove);
313         r = 1:length(critorder);
314         r(critorder) = r;
315         tokeep = find(r<cutoff);
316         feats_kept(loopnumber,1:length(tokeep)) = currentfeat(
                tokeep);
317         currentfeat = currentfeat(tokeep);
318         loopnumber = loopnumber + 1;
319
320     end
321     acc_sep_Chapelle = acc_sep;
322     feats_kept_Chapelle = feats_kept;
323
324
325     accstruct(pairidx).sepdot = acc_sep_sepleveldotprod;
326     accstruct(pairidx).pca = acc_pca;
327     accstruct(pairidx).regulardot = acc_dotprod;
328     accstruct(pairidx).chapelle = acc_sep_Chapelle;
329
```

```
330     featstruct(pairidx).sepdot = feats_kept_sepleveldotprod;
331     featstruct(pairidx).regulardot = feats_kept_dotprod;
332     featstruct(pairidx).chapelle = feats_kept_Chapelle;
333     featstruct(pairidx).geslist = controlgests;
334
335     save('recursivefeatselectmethods_badset','accstruct','
            featstruct');
336 end
337
338 %% Final Visualization
339 featlocations = find(~isnan(acc_pca));
340 featlocations_sep = find(~isnan(acc_sep_sepleveldotprod));
341
342 figure;
343 hold on;
344 semilogx(featlocations,acc_pca_mine(featlocations));
345 semilogx(featlocations,acc_sep_Chapelle(featlocations));
346 semilogx(featlocations,acc_dotprod(featlocations));
347 semilogx(featlocations_sep,acc_sep_sepleveldotprod(
      featlocations_sep));
348 legend('PCA selection','Chapelle Sep selection','Dot Product','Dot
       Product, Relevant Models','Location','best')
349 % legend('PCA','Chapelle','Separability','Location','best')
350 set(gca, 'XScale', 'log');
351 xlabel('Number of features included')
352 ylabel('Test accuracy')
```

```
353
354 %% Calculate minimum num features
355 maxthresh = max(acc_sep_sepleveldotprod./acc_sep_sepleveldotprod(
      end));
356 threshes = [maxthresh 1:-.005:.8];
357
358 for ii=1:length(threshes)
359     minfeat_pca(ii) = minFeatForRetention(acc_pca_mine,threshes(ii
        ));
360     minfeat_chapelle(ii) = minFeatForRetention(acc_sep_Chapelle,
        threshes(ii));
361     minfeat_dotprod(ii) = minFeatForRetention(acc_dotprod,threshes
        (ii));
362     minfeat_dotprodsep(ii) = minFeatForRetention(
        acc_sep_sepleveldotprod,threshes(ii));
363 end
364 figure;
365 hold on;
366 plot(threshes, minfeat_pca,'k-.');
367 plot(threshes, minfeat_dotprodsep,'r');
368
369
370 % plot(threshes, minfeat_chapelle);
371 % plot(threshes, minfeat_dotprod);
372 % legend('PCA selection','Chapelle Sep selection','Dot Product','
      Dot Product, Relevant Models','Location','best')
```

```
373 legend('PCA','Modified Chapelle','Location','best')
374 xlabel('Fraction of full-featured accuracy retained')
375 ylabel('Minimum number of features')
376 % axis tight
377 xlim([.8 maxthresh])
378
379 %% Functions
380 function sep = Psi(c,W,sepmat)
381 numctrl = 1:size(c,2);
382 combos = nchoosek(numctrl,2); % permutations of combos
383
384 dist = Inf(size(combos,1),1);
385 for kk = 1:size(combos,1)
386     c1 = c(:,combos(kk,1))';
387     c2 = c(:,combos(kk,2))';
388
389     if isempty(W)
390         dist(kk) = sepmat(c1,c2);
391     else
392         dist(kk) = sepmat(c1,c2) * min(dW(c1,W,sepmat)) * min(dW(
                c2,W,sepmat));
393     end
394
395 end
396 sep = min(dist);
397
```

```
398 end
399
400 function votingmodels = identifyRelevantModels(c,W,pairlist,sepmat
    )
401 c1models = find(ismember(pairlist(:,1),c(1),'rows'));
402 c2models = find(ismember(pairlist(:,1),c(2),'rows'));
403 votingmodels = [c1models' c2models'];
404 end
405
406 function numfeat = minFeatForRetention(acc,thresh)
407 finalacc = acc(end);
408 qualifyingnumfeat = find(acc>=(thresh*finalacc)); % this gets
    indicies
409 numfeat = min(qualifyingnumfeat);
410 if isempty(numfeat)
411     numfeat = nan;
412 end
413 end
414
415 function distW = dW(g,W,sepmat)
416 tmpdistances = inf(size(W,2),1);
417 for ii = 1:size(W,2)
418     tmpdistances(ii) = sepmat(g,W(ii));
419 end
420 distW = (tmpdistances);
421 end
```

Appendix D

ROBUSTNESS DISTRIBUTION

A comparative study is made of several methods of ranking and selection with statistical guarantees for unknown and unequal variances. A minimum of two stages is required for unknown variance. The two methods implemented are the original, Rinott's two-stage method from 1978 [46], and a later partially-sequential method by Nelson, Swann, Goldsman, and Song in 2001 [39]. The first stage is always the estimation step, when initial samplings of the distributions are taken and decisions made about necessary further sampling, and the second stage is when selection is completed.

Rinott's Two-Stage

First, an initial sample size of n_0 is taken from all k distributions. The sampled realizations from the i'th distribution are $\{\hat{Y}_{i1}, \hat{Y}_{i2}, \ldots, \hat{Y}_{in_0}\}$. For convenience, let $\nu = n_0 - 1$, $f_\nu(\cdot)$ be a χ^2 pdf with ν degrees of freedom, and $\Phi(\cdot)$ be a standard normal cdf. To approximate the distributions, the sample mean for the i'th distribution is

$$\bar{Y}_i^0 = \frac{\sum_{j=1}^{n_0} \hat{Y}_{ij}}{n_0}$$

and the sample variance is

$$\bar{S}_i^2 = \frac{\sum_{j=1}^{n_0} (\hat{Y}_{ij} - \bar{Y}_0^i)^2}{n_0 - 1}.$$

Next, the value of Rinott's constant h must be calculated. With a chosen confidence P^* that the correct distribution will be chosen and that it will be higher than the next-best by not

less than δ^*, the Rinott's constant for this problem is the h that satisfies

$$\int_0^\infty \int_0^\infty \left[\Phi\left(\frac{h}{\nu((1/x)+(1/y))} \right) f_\nu(x) \right]^{k-1} f_\nu(y)\, dydx = P^*.$$

See Appendix B for the script to calculate $h(P^*, n_0, k)$ for arbitrary inputs and the overall set selection script.

Once the initial estimation (stage 1) is complete, the Rinott's constant is used to calculate the number of measurements it would take to appropriately characterize the distributions in order to select the highest mean. The $n_i - n_0$ additional measurements needed (if any) is found by $n_i = \max\{n_0, \lceil (hS_i/\delta^*)^2 \rceil\}$. The requisite batch of additional samples are taken from each distribution and the final sample means recalculated:

$$\bar{Y}_i = \frac{\sum_{j=1}^{n_i} \hat{Y}_{ij}}{n_i}.$$

and then the distribution with the highest $\bar{Y}_i$ is selected.

Nelson, Swann, Goldsman, Song (NSGS)

This method has a lot in common with Rinott's method, but it eliminates some distributions from consideration before the second sampling. This can allow the number of required samples to be much lower than in Rinott's method.

In the first stage, the initial sampling with n_0 realizations and calculation of sample statistics proceeds mostly as before, with a chosen P_0, P_1 satisfying $(P_0 + P_1) - 1 = P^*$ and the Rinott's constant being $h(P_1, n_0, k)$. Once that's complete, the decisions about what distributions may be cut can be made. To do this, some measure of the possible expected increase in mean is required. This allows identification of which distributions can't surpass another by at least δ^* with probability P_0 regardless of how many additional samples are taken. This measure W_{ij} satisfies

$$W_{ij} = t\sqrt{\frac{S_i^2 + S_j^2}{n_0}},\ i \in 1:k,\ j \in 1:k,\ i \neq j$$

where $t_{\beta,nu}$ is the β quantile of a t-distribution of ν degrees of freedom and $t = t_{P_0^{1/(k-1)},n_0-1}$. Then, the criterion for if a distribution remains in contention is if its potential improvement with more samples could improve it more than any other distribution. This is formalized by saying that that i'th distribution moves on to stage 2 if it satisfies

$$\bar{Y}_i^0 \geq \bar{Y}_j^0 - [W_{ij} - \delta^*]^+ \; \forall j \neq i.$$

If there is only one distribution remaining, then the process is over; otherwise, the remaining distributions proceed to Stage 2. Stage 2 is the same as in Rinott's procedure: calculate the number of additional samples needed, sample the distributions, recalculate the final means, and report the maximizer.

Data & Solution Procedure

For simplicity, generated data will be used. This allows the imposition of normality conditions and the ability to theoretically sample an arbitrary number of times and with an arbitrary (but finite) number of features.

Using generated data and their true labels (class 1 vs. class 2), the first task is to synthesize a single distribution for each case which represents the projected robustness of this case. This measure should have a higher mean in cases where the data clusters are well-separated. To achieve all this, the new distribution for the i'th case is $R^i \sim \hat{y}_m^i y_m^i \; \forall m$, where y_m is the class label for the m'th data point $\in \{-1, 1\}$ and $\hat{y}_m$ is the assigned classifier score that satisfies $\hat{y}_m = a' x_m + b$ with the (a, b) that represent the support vector machine model. Note that for direct comparisons, this method is limited to soft-margin SVM classifiers.

When the above assumptions hold, R has the desired characteristics listed above. When $\hat{y}$ and y agree in sign and x is not near the dividing hyperplane, $\hat{y} \times y$ is large and positive. If this is true of many points, it indicates that the two classes are well-defined in feature space and also means that R will have a large mean. Large numbers of points very near to the hyperplane or whose $\hat{y}$ and y disagree in sign would be indicative of very enmeshed

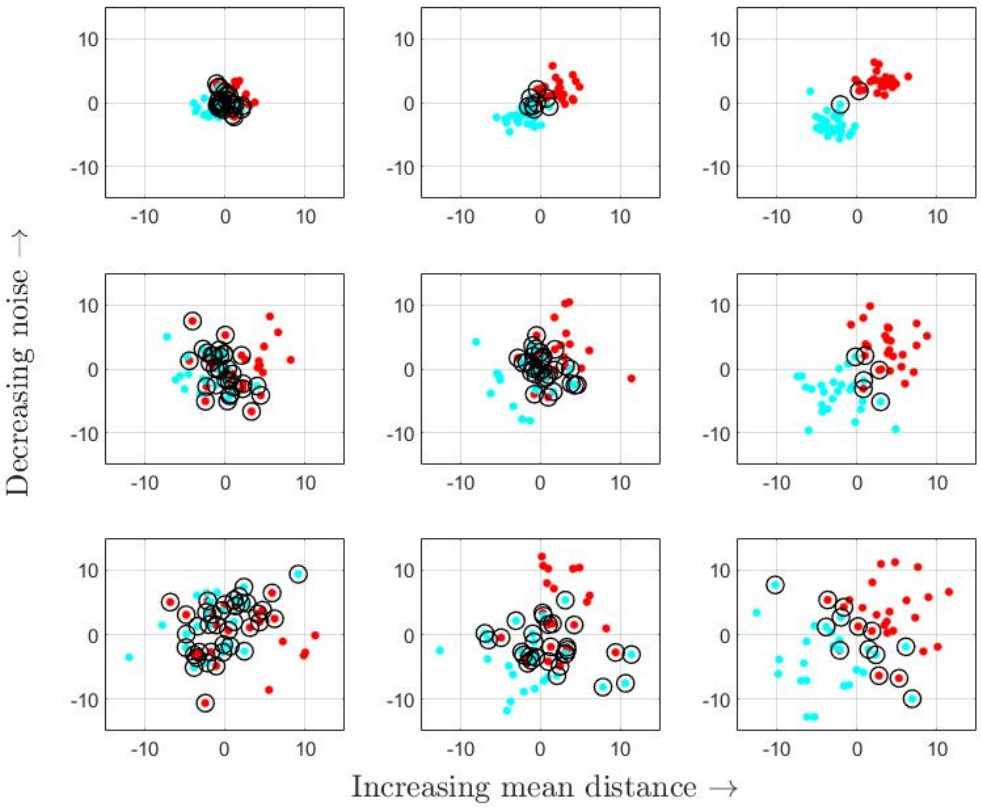


Figure D.1: All nine classes, two features with selected support vectors circled. For visual clarity, this is with far fewer data points than are actually used to define the model.

classes, and R would have a lower mean. In order for this distribution to be normal, the classes themselves must be normal and their variances must be equal. These are artificially-imposed conditions of the generated data, and will generally not be true for real gesture data. Additionally, the SVM algorithm must be able to find a decent solution with the data realizations available; informally, most of the $y = 1$ points should have a positive $\hat{y}$, and vice versa. If this is not the case, the distribution R will turn out to be bimodal. Once these distributions are generated, the two ranking and selection methods are applied as described

in the previous section.

Results

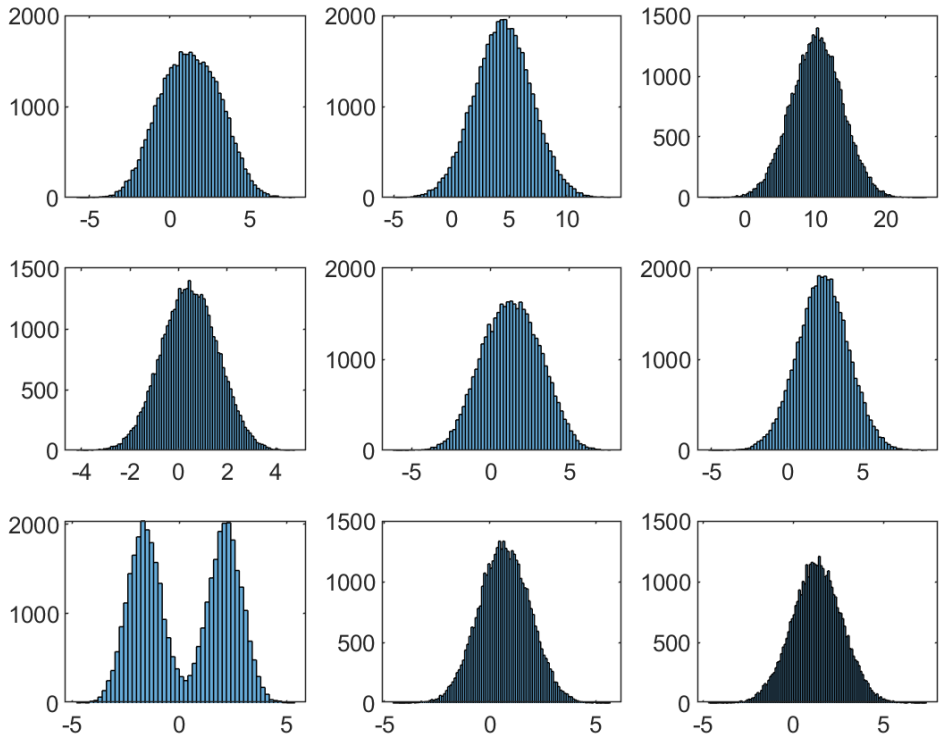


Figure D.2: The R-distributions of all nine classes, appearing in the order they appear in Fig. D.1. It is visually clear that Case 3 in the upper-right hand corner has the highest mean. Additionally illustrated is what R looks like when the SVM solver does not find the ground-truth optimal hyperplane ($b = 0$) but instead b is large and most of a class is misclassified.

Case 3, the known most-separable case (upper-right corner in Fig. D.1), is reliably chosen by all methods but with widely varying sample requirements between the two methods. In several realizations, the NSGS method did not even need to proceed to Stage 2, while Rinott's

method required thousands of additional realizations. The R distributions for each case are shown in Fig. D.2, including a visualization of the failure mode wherein the SVM classifier does not find a good solution and the distribution is bimodal. This only ever appears in the highly enmeshed cases, though, so it has a minimal effect on the selection. In the realization shown in Fig. D.2, 10 initial samples was sufficient for the NSGS method but Rinott's required 17615 additional samples of Case 3.

Discussion

Worth mentioning is the highly restrictive assumption placed on the gesture class distributions in order to use this method. The machinery entirely breaks down if the classes are not multivariate normal with symmetric distributions. It is not obvious that a real case exists where these assumptions are remotely reasonable or that these conditions can be relaxed, so this work is primarily an exercise in what statistical guarantees might look like rather than a serious proposal.

Appendix E

GENERATION OF SUPPORTING EXAMPLES

E.1 Nine-case comparisons additional details and code

As an addendum to Fig. 4.10 (a), a direct visualization of the best and worst cases is shown in Fig. E.1 The Matlab code used to generate this and other 9-class comparison figures

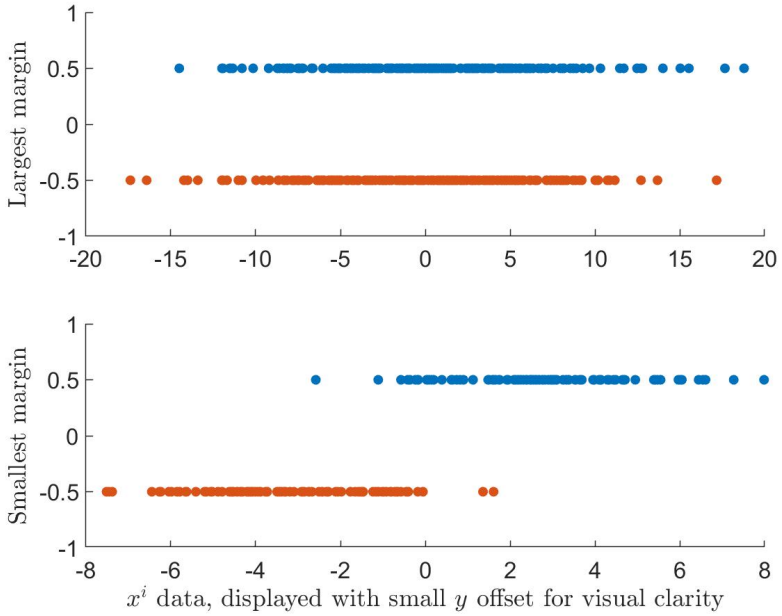


Figure E.1: As expected, the "largest" reported margin is the most enmeshed classes and vice versa, as the margin is explicitly solving for the worst cases.

is below. A consistent rng seed is included so the figures should appear the same on any machine, but the generated data is also saved in case later analysis is needed. The qualitative

results are consistent for many rng seed values.

```
% Rose Hendrix
% Experimentation with SVM margin and related distance measures with
% generated data

clearvars; close all;

set(0,'DefaultTextInterpreter','Latex',...
    'DefaultLegendInterpreter','Latex','DefaultAxesFontSize',12,'DefaultTextFontSize',14,...
    'defaultlinemarkersize',16)

rng(2) % for repeatability. generated data from proposal also saved in a backup .mat file

% all the considered means and noises
I = [1 -1; 2 -2; 3 -3; 1 -1; 2 -2; 3 -3; 1 -1; 2 -2; 3 -3];
N = 4*[.5 .5; .5 .5; .5 .5; 1 1 ; 1 1; 1 1; 1.5 1.5 ; 1.5 1.5; 1.5 1.5];
casenums = [1:9]'; % convenience variable; nine cases compared corresponding to I and N
numfeatures = 1; % dimensionality
basenumsamples = 4*[25 25]; % lower bound on population sizes

```

```
20 numgenerations = 10; % iterations of the nine cases, each
       successive one with larger populations
21 alph_EM = [0,0.5,1,2]; % tuneable alpha parameter for various
       methods
22 alph_HT = [3,100];
23
24 Ifull = repmat(I,numgenerations,1);
25 Nfull = repmat(N,numgenerations,1);
26 casesfull = repmat(casenums,numgenerations,1);
27
28 % preallocate the save arrays with NaNs to accomodate population
       size
29 % differences
30 Xsave = NaN(numgenerations * 2*basenumsamples(1),size(Ifull,1),
       numfeatures);
31 Ysave = NaN(numgenerations * 2*basenumsamples(1),size(Ifull,1));
32 xsave = NaN(numgenerations * 2*basenumsamples(1),size(Ifull,1),
       numfeatures);
33 ysave = NaN(numgenerations * 2*basenumsamples(1),size(Ifull,1));
34
35
36 for jj = 1:size(Ifull,1)
37     genmultiplier = (floor((jj-1)/9) + 1);
38     classtype = mod(jj,9);
39     if(classtype == 0); classtype = 9; end % minor change for
           comprehensibility
```

```
40
41      numsamples = genmultiplier * basenumsamples;
42      ii = Ifull(jj,:);
43      nn = Nfull(jj,:);
44
45      % make the training data and train models
46      [X,Y] = generateData(ii,nn,numfeatures,numsamples);
47      m = fitcsvm(X,Y,'KernelFunction','linear');
48      sv = m.SupportVectors;
49
50
51      marginsep(jj) = psimargin(m);
52      for aa = 1:length(alph_EM)
53          [EMsep(jj,aa)]= psiEffMarg(X,numsamples,m,alph_EM(aa));
54      end
55
56      for aa = 1:length(alph_HT)
57          [HTsep(jj,aa)]= psiHypoth(X,numsamples,m,alph_HT(aa));
58      end
59      % generate test data
60      [x,y] = generateData(ii,nn,numfeatures,floor(0.5*numsamples));
61      % classify test data
62      [labels,~] = predict(m,x);
63      accuracy(jj) = 1 - nnz(y-labels)/numel(y);
64
65      % save realizations
```

```
66      Xsave(1:2*numsamples,jj,:) = X;
67      Ysave(1:2*numsamples,jj) = Y;
68      xsave(1:2*floor(0.5*numsamples),jj,:) = x;
69      ysave(1:2*floor(0.5*numsamples),jj,:) = y;
70
71 end
72 %% Plotting and visual interpretation
73 % margin
74 sepused = marginsep;
75 [~,idx] = sort(sepused,'descend');
76
77 figure; plot(sepused(idx),accuracy(idx),'.','MarkerSize',20)
78 ylabel('Test accuracy ratio')
79 xlabel('Margin size: SVM margin')
80 % xlabel('$\mathcal{D} = (1-p)^{\alpha}$')
81 % title(['\# of features: ' num2str(numfeatures) ', $\alpha$ = '
     num2str(alph)])
82 % title(['Population sizes: ' num2str(basenumsamples(1)), '-',
     num2str(numsamples(1)), ', $\alpha$ = ' num2str(alph)])
83
84 % plot best and worst raw data
85 figure;
86 subplot(2,1,1)
87 Xloc = Xsave(:, idx(1),1); Yloc = Ysave(:, idx(1));
88 Xloc = Xloc(~isnan(Xloc)); Yloc = Yloc(~isnan(Yloc));
```

```
89 half = (length(Yloc)/2); % should always be an even number so half
       a valid index
90 hold on;
91 plot(Xloc(1:half),0.5+zeros(1,half),'.');
92 plot(Xloc(half+1:end),-0.5+zeros(1,half),'.');
93 ylabel('Largest margin')
94 hold off;
95 ylim([-1 1])
96
97 subplot(2,1,2)
98 Xloc = Xsave(:, idx(end),1); Yloc = Ysave(:, idx(end));
99 Xloc = Xloc(~isnan(Xloc)); Yloc = Yloc(~isnan(Yloc));
100 half = (length(Yloc)/2); % should always be an even number so half
       a valid index
101 hold on;
102 plot(Xloc(1:half),0.5+zeros(half,1),'.');
103 plot(Xloc(half+1:end),-.5+zeros(1,half),'.');
104 ylabel('Smallest margin')
105 xlabel('$x^i$ data, displayed with small $y$ offset for visual
      clarity')
106 hold off;
107 ylim([-1 1])
108
109 % hypothersis testing
110 for aa = 1:length(alph_HT)
111 sepused = HTsep(:,aa);
```

```
112 [~,idx] = sort(sepused,'descend');
113
114 figure; plot(sepused(idx),accuracy(idx),'.','MarkerSize',20)
115 ylabel('Test accuracy ratio')
116 xlabel('Margin size: hypothesis testing')
117 % xlabel('$\mathcal{D} = (1-p)^{\alpha}$')
118 % title(['\# of features: ' num2str(numfeatures) ', $\alpha$ = '
      num2str(alph_HT(aa))])
119 title(['Population sizes: ' num2str(basenumsamples(1)), '-',
      num2str(numsamples(1)), ', $\alpha$ = ' num2str(alph_HT(aa))])
120 end
121
122
123 % effective margin
124 for aa = 1:length(alph_EM)
125
126 sepused = EMsep(:,aa);
127 [~,idx] = sort(sepused,'descend');
128
129 figure; plot(sepused(idx),accuracy(idx),'.','MarkerSize',20)
130 ylabel('Test accuracy ratio')
131 xlabel('Margin size: effective margin')
132 % xlabel('$\mathcal{D} = (1-p)^{\alpha}$')
133 title(['\# of features: ' num2str(numfeatures) ', $\alpha$ = '
      num2str(alph_EM(aa))])
```

```
134 % title(['Population sizes: ' num2str(basenumsamples(1)), '-',
      num2str(numsamples(1)), ', $\alpha$ = ' num2str(alph_EM(aa))])
135 % axis([0 1 0.4 1])
136 end
137
138 %% save the generated data just in case
139 filename = ['thesisproposal_savedata_basesamples' num2str(
      basenumsamples(1)) '_numfeatures' num2str(numfeatures) '.mat'];
140 save(filename, 'Xsave','Ysave','xsave','ysave');
141
142 %% functions
143
144 function d = psimargin(m)
145 d = 2/norm(m.Beta);
146 end
147
148 function [d] = psiEffMarg(X,numsamples,m,alph)
149 [~,scores] = predict(m,X);
150 class1 = scores(1:numsamples(1));
151 class2 = scores(numsamples(1)+1:numsamples(1)+numsamples(2));
152 avgs = [mean(class1) mean(class2)];
153 stdev = [std(class1) std(class2)];
154
155 d = sum(abs(avgs))-alph*(stdev(1)+stdev(2));
156 end
157
```

```
158 function [d] = psiHypoth(X,numsamples,m,alph)
159
160 [~,scores] = predict(m,X);
161 class1 = scores(1:numsamples(1));
162 class2 = scores(numsamples(1)+1:numsamples(1)+numsamples(2));
163
164 digits(64) % increase the precision of p
165 [~,p] = ttest(class1,class2);
166 d = vpa(1-vpa(p))^alph;
167 end
168
169
170 function [X,y] = generateData(means,noises,timesteps,numsamples)
171 class1 = repmat(means(1),numsamples(1),timesteps) + noises(1).*
      randn(numsamples(1),timesteps);
172 class2 = repmat(means(2),numsamples(2),timesteps) + noises(2).*
      randn(numsamples(2),timesteps);
173 X = [class1;class2];
174 y(1:numsamples(1)) = 1;
175 y(numsamples(1)+1:numsamples(1)+numsamples(2)) = 2;
176 y=y';
177 end
```

E.2 Divergence calculations

Also included is the code for the representative examples calculated on Fig. 4.8, including KL divergence, Earth Mover's Distance, and Separation distance.

```
% Rose Hendrix
% script for calculating divergences between prob dists

clearvars;
dx = 0.01;

% KL divergence
A1 = normpdf([-10:dx:10], -0.5, 1);
B1 = normpdf([-10:dx:10],  0.5, 1);
KL1 = dx * sum(A1 .* (log(A1)-log(B1)))

A2 = normpdf([-10:dx:10], -0.5, 3);
B2 = normpdf([-10:dx:10],  0.5, 1);
KL2 = dx * sum(A2 .* (log(A2)-log(B2)))

% Earth mover's distance: multivariate normal
V1 = norm(-0.5-0.5)^2 +(1 + 1 - 2*((sqrt(1)*1*sqrt(1))^(1/2)))

V2 = norm(-0.5-0.5)^2 +(3 + 1 - 2*((sqrt(1)*3*sqrt(1))^(1/2)))

% separation distance
s1 = max(1-A1./B1)
s2 = max(1-A2./B2)

% separation distance: reverse arguments
s1_rev = max(1-B1./A1)
```

```
27 s2_rev = max(1-B2./A2)
```

E.3 SVM margin details and feature selection

Included here is the code for the low-population 1-D examples of the SVM margin case, as well as the code for generating the feature selection examples in Fig. 5.2. That example also has a consistent rng seed for repeatability.

```
1 % Rose Hendrix
2 % simple 1-D numerical SVM example for thesis
3
4 clearvars; close all;
5
6 set(0,'DefaultTextInterpreter','Latex',...
7     'DefaultLegendInterpreter','Latex','DefaultAxesFontSize',12,'
        DefaultTextFontSize',14)
8
9 addpath('helper_functions');
10
11 load generatedsimple.mat
12
13 x =  [-1.0,-0.75,0.75,1]';
14 y = [-1,1,-1,1]';
15
16 x1 = X([1:10])+0.5;
17 x2 = [X([2501:2510])-0.5 ; 1];
18 x = [x1;x2];
19 y = Y([1:10,2501:2511]);
```

```
20 [a,b,yhat,lambda,~] = mySVMsoft(x,y,2)
21
22
23 x =  [-1.0,-0.01,0.01,1]';
24 y = [-1,-1,1,1]';
25 C = inf;
26 [a,b,yhat,lambda,~] = mySVMsoft(x,y,C)
27
28 % figure
29 % hold on;
30 % plot(x1,0*y(1:10),'.','MarkerSize',20)
31 % plot(x2,0*y(11:end),'.','MarkerSize',20)
32 % axis tight
33 %
34 clearvars;
35 rng(1)
36 N = 1000;
37 alph1 = 2;
38 alph2 = 1.5;
39
40 x = [random('Stable',alph1,0,1,0,N,1) ; random('Stable',alph1
      ,0,1,0,N,1) ];
41 x(:,2) = [random('Stable',alph2,0,1,2,N,1) ;  random('Stable',
      alph2,0,1,-2,N,1) ];
42 x0=x;
43
```

```
44 figure; hold on;
45 plot(x0(1:N,1),x0(1:N,2),'.')
46 plot(x0(N+1:end,1),x0(N+1:end,2),'.')
47 % axis([-20 20 -20 20]);
48 axis tight
49
50 theta = pi/4;
51 R = [cos(theta) -sin(theta); sin(theta) cos(theta)];
52 x2 = x0*R*R;
53 y = [ones(N,1) ; 2*ones(N,1)];
54
55 % rescale to 0-1
56 for featnum = 1:size(x0,2)
57     x0(:,featnum) = (x0(:,featnum) - min(x0(:,featnum))) ./ (max(
          x0(:,featnum))- min(x0(:,featnum)));
58     x2(:,featnum) = (x2(:,featnum) - min(x2(:,featnum))) ./ (max(
          x2(:,featnum))- min(x2(:,featnum)));
59 end
60 x1 = x0*R;
61 for featnum = 1:size(x0,2)
62     x1(:,featnum) = (x1(:,featnum) - min(x1(:,featnum))) ./ (max(
          x1(:,featnum))- min(x1(:,featnum)));
63 end
64
65 % close all
66 C = 10;
```

```
67 [a0,b0,~,~,~] = mySVMsoft(x0,y,C);
68 [a1,b1,~,~,~] = mySVMsoft(x1,y,C);
69 [a2,b2,~,~,~] = mySVMsoft(x2,y,C);
70
71 [~,L0,U0,S0,~] = myPCA(x0',100);
72 [~,L1,U1,S1,~] = myPCA(x1',100);
73 [~,L2,U2,S2,~] = myPCA(x2',100);
74
75
76 figure
77 hold on
78 title(['$a_1 << a_2$: $a = $[' num2str(a0) ']' ],'FontSize',14)
79 xlabel('$x_1$','FontSize',16)
80 ylabel('$x_2$','FontSize',16)
81 plot(x0(1:N,1),x0(1:N,2),'.','MarkerSize',14)
82 plot(x0(N+1:end,1),x0(N+1:end,2),'.','MarkerSize',14)
83 figure
84 hold on
85 title(['$a_1 \propto a_2$: $a = $[' num2str(a1) ']' ],'FontSize'
      ,14)
86 xlabel('$x_1$','FontSize',16)
87 ylabel('$x_2$','FontSize',16)
88 plot(x1(1:N,1),x1(1:N,2),'.','MarkerSize',14)
89 plot(x1(N+1:end,1),x1(N+1:end,2),'.','MarkerSize',14)
90 figure
91 hold on
```

```
92 title(['$a_1 >> a_2$: $a = $[' num2str(a2) ']' ],'FontSize',14)
93 xlabel('$x_1$','FontSize',16)
94 ylabel('$x_2$','FontSize',16)
95 plot(x2(1:N,1),x2(1:N,2),'.','MarkerSize',14)
96 plot(x2(N+1:end,1),x2(N+1:end,2),'.','MarkerSize',14)
```