Statistical Modeling and Analysis of Audio-Visual Association in Speech

by

Michael Richard Siracusa

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 2004

Author .....
Department of Electrical Engineering and Computer Science
June, 15 2004

Certified by .....
Trevor Darrell
Associate Professor EECS
Thesis Supervisor

Certified by .....
John W. Fisher
Principal Research Scientist EECS
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Statistical Modeling and Analysis of Audio-Visual Association in Speech

by

Michael Richard Siracusa

Submitted to the Department of Electrical Engineering and Computer Science
on August, 31 2004, in partial fulfillment of the
requirements for the degree of
Masters of Science in Computer Science and Engineering

Abstract

Currently, most dialog systems are restricted to single user environments. This thesis aims to promote an untethered multi-person dialog system by exploring approaches to help solve the speech correspondence problem (i.e. who, if anyone, is currently speaking). We adopt a statistical framework in which this problem is put in the form of a hypothesis test and focus on the subtask of discriminating between associated and non-associated audio-visual observations. Various methods for modeling our audio-visual observations and ways of carrying out this test are studied and their relative performance is compared. We discuss issues that arise from the inherently high dimensional nature of audio-visual data and address these issues by exploring different techniques for finding low-dimensional informative subspaces in which we can perform our hypothesis tests. We study our ability to learn a person-specific as well as a generic model for measuring audio-visual association and evaluate performance on multiple subjects taken from MIT's AVTIMIT database.

Thesis Supervisor: Trevor Darrell
Title: Associate Professor EECS

Thesis Supervisor: John W. Fisher
Title: Principal Research Scientist EECS

Acknowledgments

First, I would like to thank my advisors, John Fisher and Trevor Darrell, for their unwavering support and encouragement. I would have been lost without their wisdom and guidance. A special thanks also goes out to Alex Ihler, who has repeatedly helped me gain new insight into the problems discussed in this thesis and donated his KDE Matlab code.

I must also thank the members of the VIP group who were always willing to listen to my problems and discuss different parts of this thesis. In particular, I greatly appreciated the time Kevin Wilson spent helping me and copy editing my text. I salute him for being one of the very few who have actually read the majority of this document. I also would like to thank Mario Christoudias and Kate Saenko for listening to my complaints about my thesis even though they were also trying to finish their own. Kate gets a second thanks, in addition to Jim Glass, for supplying the AVTIMIT database for my experiments.

Two other people I must thank are Kinh Tieu and Carrie McGrory. Kihn was always willing and excited to discuss any research direction I was currently into. Carrie was always ready to get some food or help me calm down when it seemed like everything was going wrong. She was even willing to perform a last minute copy edit to fix as much of my horrible grammar as she could. I also must thank my officemates, Ali Rahimi, Bryan Russell and Ce Liu, for their encouragement and for putting up with my random outbursts of frustration.

Most importantly, I must thank my family. They mean everything to me. John and Tina deserve a special award for letting me eat their food and use their washing machine on the weekends. They were very good about not asking me "are you done yet?." I must also thank my nephews Ethan, Simon and Alex, whose pictures I keep on my desk to remind me of what is truly important and to smile. Last, but most important, I would like to thank my mom and dad. I am so blessed to have such wonderful and loving parents. They have given me so much.

Contents

1 Introduction **17**

1.1 Outline 19

2 Related Work **21**

3 The Speech Correspondence Problem as a Hypothesis Test **27**

3.1 A Generative Approach 28

3.1.1 Audio-Visual Association LRT 28

3.1.2 Speech Correspondence LRT 29

3.1.3 Measuring Performance 31

3.1.4 Aspects of using a learned Generative Model 33

3.1.5 Simple Example 39

3.2 Discriminative Approach 42

3.2.1 Support Vector Machine 43

4 Informative Subspace Learning **47**

4.1 Maximizing Energy 49

4.1.1 PCA 49

4.1.2 Joint PCA 53

4.2 Maximizing Mutual Information 53

4.2.1 CCA 55

4.2.2 Nonparametric Maximization of MI 57

4.2.3 Regularization 60

4.3 Maximizing Discriminability 63
4.3.1 Simple Illustrative Examples 67

5 Dataset and Preprocessing **71**
5.1 Face Tracking 72
5.2 Audio Visual Features 73
5.2.1 Audio 74
5.2.2 Video 76

6 Experiments **79**
6.1 Experiment 1 : Comparing Modeling Techniques for Detecting AV Association 79
6.1.1 Purpose 79
6.1.2 Training and Testing Procedure 80
6.1.3 Techniques Compared 80
6.1.4 Variables 83
6.1.5 Results: Synthetic Data 84
6.1.6 Results: Audio-Visual Data from AVTIMIT 92
6.1.7 Summary and Discussion 105
6.2 Experiment 2 : Techniques for learning Informative Subspaces Comparison 109
6.2.1 Purpose 109
6.2.2 Variables 109
6.2.3 Results : Synthetic Data 110
6.2.4 Results: Data from AVTMIT 116
6.2.5 Summary and Discussion 129
6.3 Experiment 3: Exploring the Importance of Kernel Size 134

7 Discussion and Future Work **137**

A Density Estimation **141**
A.1 Learning a Gaussian Model 141

A.2 Kernel Density Estimation 144
A.2.1 Choosing a Kernel Size 144

B Select Elements of Information Theory **147**
B.1 Entropy 147
B.2 KL Divergence 149
B.3 Mutual Information 149
B.4 Data Processing Inequality 150
B.4.1 Sufficient Statistics 150

C CCA **153**
C.1 Derivation 153
C.2 Implementing CCA with SVD 155

D L2 Regularization **159**
D.1 Least Squares Linear Regression 159
D.1.1 Minimizing Squared Error 160
D.1.2 Maximizing the Likelihood of h 160
D.2 Regularized Least Squares (Ridge Regression) 161
D.2.1 Minimizing Squared Error plus L2 Norm Constraint 162
D.2.2 Regularization as a Prior on h 162
D.2.3 Regularization in relation to an Observation Noise Model . . . 163
D.3 Regularized Canonical Correlation 166

E Entropy Gradient Calculations **169**
E.1 Law of Large Numbers Approximation 169
E.2 ISE Approximation 171

F Experiment 1 Raw Results **175**

List of Figures

2-1 Hershey and Movellan's technique. 22

2-2 Finding an informative subspace 23

3-1 Hypothesis test for Audio-Visual Association 28

3-2 Speech Correspondence Hypothesis Test 30

3-3 General LRT Performance characteristics. The probability of detection is the integral of $\mathcal{L}(\mathbf{a}, \mathbf{v})$'s distribution under $\mathcal{H}_0$ (red,left) from η to ∞.The probability of false alarm is the integral of $\mathcal{L}(\mathbf{a}, \mathbf{v})$'s distribution under $\mathcal{H}_1$ (blue,right) from η to ∞. 33

3-4 Estimating distribution under $\mathcal{H}_0$ 37

3-5 Hypothesis test for single audio and video stream 41

3-6 Grid search results. The contours represent levels of the number of correctly classified samples during cross validation 45

3-7 Sample SVM Classification on Gaussian Data 46

4-1 System Components 48

4-2 Comparison of PCA and JPCA. Top 3 components 52

4-3 Circle Test for MD 69

5-1 Sample Frames from AVTIMIT Corpus 72

5-2 Full System Training and Testing Components 72

5-3 Face Tracking 73

5-4 Extracted Lower Face 74

5-5 Wavelet Pyramid 77

5-6 Differential Video Features: (a) Difference Image, (b) Horizontal Edge Filter (c) Horizontal Edge Difference Image 77
5-7 Optical Flow 78

6-1 Synthetic Data 85
6-2 Performance on Gaussian Data 86
6-3 LR Plots on Gaussian Data. Shows the mean and standard deviation of the likelihood ratio under hypothesis H_1 (associated, plotted in red stars, upper) and H_0 (non-associated, plotted in blue triangles, lower) versus window length. 87
6-4 Performance on Nonlinear Data 89
6-5 LR Plots on Nonlinear Data. Shows the mean and standard deviation of the likelihood ratio under hypothesis H_1 (associated, plotted in red stars, upper) and H_0 (non-associated, plotted in blue triangles, lower) versus window length. 91
6-6 Performance Summary Plots for Person 1 (Pixel Intensity, MFCCs) . 94
6-7 Samples drawn for person 1, 1 PCA coefficient (Pixel Intensity, MFCCs) 95
6-8 PCA Components and the cumulative amount of energy they capture for Person 1 96
6-9 Performance summary plot. Testing and training on different people. (Pixel Intensity, MFCC) 97
6-10 Samples drawn for 10 people. Each person is indicated by a different color/shade, 1 PCA coefficient (Pixel Intensity, MFCCs) 98
6-11 Performance Summary Plots (Image and MFCC differences). (a)(b) Results for Person 2 (c)(d) Results for 10 People 100
6-12 Training samples using 1 principal component. (Image and MFCC differences) 101
6-13 Probability of error versus window length using 5 PCA components and a Gaussian Model 102
6-14 Performance Summary Plots (Flow and MFCC differences) 103

6-15 Top 5 principal components of the optical flow features 103
6-16 Learned Projections (Linear Data) 113
6-17 Linear Data: Pe vs Technique 114
6-18 Learned Projections (Nonlinear) 115
6-19 Nonlinear Data: Pe vs Technique 115
6-20 CCA Learned Projections / Bases (Flow and MFCC differences) . . . 118
6-21 MMI Learned Projections / Bases (Flow and MFCC differences) . . . 119
6-22 Decision Boundaries learned by MMI and CCA for Flow and MFCC Diff features 120
6-23 P(error) vs % Energy kept during PCA Regularization for CCA and MMI on Image and MFCC Differences 122
6-24 CCA and MMI Learned Projections / Bases (Image and MFCC differences). The number above each projection is the % of video energy preserved for regularization. 124
6-25 Data projected into subspaces learned by (a) CCA (b) MMI (d) Random for Image and MFCC Differences. (c) Shows a random linear combination of the top 512 principal components of the training data. 125
6-26 Plot of samples used by KDE Model for the 6 dimensional subspace learned by CCA (Flow and MFCC Differences). Dimensions 1-3 are audio and 4-6 are video. 128
6-27 Plot of samples used by KDE Model for the 6 dimensional subspace learned by CCA (Image and MFCC Differences). Dimensions 1-3 are audio and 4-6 are video. 130
6-28 Comparing Techniques for Choosing Kernel Size 135

List of Tables

6.1 Techniques Compared 81

6.2 Person-specific results summary using pixel intensities and MFCCs. The best performance for each technique is summarized. The number of principal components and window length (in samples) used to achieve the best probability of error are listed. A common separated list of parameters indicates that each parameter achieved similar performances. 94

6.3 Person-specific and generic results summary using image and MFCC differences. 99

6.4 Probability of error for different audio visual features. Technique = Gaussian Model i.i.d. , # PCA = 5, Window Length = 63 samples . 104

6.5 Informative Subspace Learning Techniques and Associated Model Techniques. Details about the subspace techniques can be found in Chapter 4. 109

6.6 Results for linear data 112

6.7 Results for nonlinear data 113

6.8 Results comparison for PCA, CCA and MMI using flow statistics and MFCC differences (10 People) 116

6.9 Results comparison for PCA, CCA and MMI using flow statistics and MFCC differences (10 People) 127

6.10 Results comparison for PCA, CCA and MMI using Image and MFCC differences (10 People) 127

F.1 Experiment 1: Raw Results for Person 1 : Pixels Intensities and MFCCs176
F.2 Experiment 1: Raw Results for Person 2 : Pixels Intensities and MFCCs176
F.3 Experiment 1: Raw Results for 10 people : Pixels Intensities and MFCCs177
F.4 Experiment 1: Raw Results for Person 1 : Image Diffs and MFCC Diffs 178
F.5 Experiment 1: Raw Results for Person 2 : Image Diffs and MFCC Diffs 178
F.6 Experiment 1: Raw Results for 10 People : Image Diffs and MFCC Diffs179
F.7 Experiment 1: Raw Results for Person 1 : Flow and MFCC Diffs . . 180
F.8 Experiment 1: Raw Results for Person 2 : Flow and MFCC Diffs . . 180
F.9 Experiment 1: Raw Results for 10 People : Flow and MFCC Diffs . . 181

Chapter 1

Introduction

Natural human-computer interfaces should place few constraints on the human users. Conversational dialog systems have worked toward achieving this goal by allowing speech to be used as a natural form of input. However, most speech interfaces require that the users be tethered to the system, forcing them to wear a close-talking microphone or talk into a handset. This inconvenience restricts the use of such a system to controlled environments and makes it difficult for multiple people to freely interact with the system and each other. We would like to help promote the development of systems that can participate in multi-person untethered conversation by addressing the problem of associating speech with a particular user.

Determining who is currently speaking in a given scene is not a simple task but the use of multiple sensing modalities makes the problem more tractable. Visual cues can be used to determine if a person is in the field of view, what direction he or she is facing and if his or her lips are moving. However, these cues cannot tell if a subject's lip movement is caused by speaking or by some other process such as a change in expression. Audio cues can tell us when someone is speaking. However, when using only audio, it is often difficult to identify who is speaking the utterance and whether or not that speaker can be seen by the system. It is the fusion of these cues that can identify when a person's lip movements or actions are a result of speech. Consequently, we wish to develop a practical model of the joint audio-visual statistics to help discriminate between consistent and inconsistent observations of speech.

The search for such a model can be biologically motivated. It has been shown that the manner in which humans perceive the location of a sound source is strongly linked to the sound's synchronization with visual events [3]. A television provides a simple example of how this relationship can be exploited. Although the audio may be emanating from the side of a television, the viewer perceives the sound as if it is coming from the mouth of a person shown on the screen. When there is no obvious association between audio and video, the observer concludes that the sound is coming from an "off camera" source or that the show is badly dubbed.

This audio-visual fusion is an automatic process for humans and is hard to break. The Mcgurk Effect [19] shows how our perception of speech can be modified by visual events. When presented simultaneously with the sound /ba/ and the lip movement corresponding to /ga/ the observer hears /da/. The effect is an involuntary action that can only be broken by removing one of the modalities. Such experiments strongly suggest that our perception of human speech involves a low level audio-visual integration. However, it is not entirely clear how much our prior information about human speech plays a role in this illusion.

There has been a variety of work on audio visual integration in the domain of human speech. Some have worked on measuring the low-level statistical relationship between audio and video signals [8, 20, 12], while others have focused more on the creation of phoneme and viseme models for the higher level task of audio-visual speech recognition [21]. The solution to the speech correspondence problem most likely lies somewhere between these two extremes. It is our goal to build a practical joint audio-visual statistical model that incorporates prior knowledge about human speech, while at the same avoids the need of a full audio-visual speech recognition or person identification system.

This thesis adopts a statistical framework in which the speech correspondence problem is put in the form of a hypothesis test. Our primary focus is on the subtask of testing the likelihood of audio-visual association using a limited number of observations from a single camera and microphone. We explain noth generative and discriminative methods for carrying out this test compare the relative performance

of each. We discuss issues that arise from the inherently high dimensional nature of audio-visual observations and address these problems by exploring different techniques for finding low dimensional informative subspaces in which we can perform our hypothesis tests. All experiments are performed on subjects taken from the MIT AVTIMIT database.

1.1 Outline

We begin by discussing related work in Chapter 2. In Chapter 3 we present the speech correspondence problem in terms of a simple hypothesis test. We compare both generative and discriminative approaches for executing this test and discuss how to measure performance. Additionally, we explore how using models learned online or from training data affect performance. In Chapter 4 we discuss the practical issues that arise from using high dimensional audio-visual observations. We discuss some standard techniques for dimensionality reduction and present the concept of informative subspaces. Different techniques for learning these subspaces are explored. Details about the AVTIMIT database and the audio-visual features used for our experiments are discussed in Chapter 5. We show how we use a simple face tracker to obtain segmented faces of the subjects used for testing. Experimental results are presented in Chapter 6. We study our ability to learn person-specific as well as generic models for measuring audio-visual association. Lastly, we conclude with discussion and future work in Chapter 7.

Chapter 2

Related Work

Recently many systems have been developed for speaker detection and segregation using video or audio cues. Here, we summarize a portion of the most relevant previous work. An early system, [14], used cues such as user pose, user proximity and visual speech activity. These cues were combined using simple fusion rules to determine whether a person is speaking to the camera and to enable automatic control of a speech recognition system in a traditional desktop environment. Detection of speech activity was achieved by observing changes in the average illuminance of a tracked mouth region. Several systems for speaker detection using visual cues have been proposed using Bayesian Networks [24] [26]. These systems exploited a sophisticated statistical model for fusion but were primarily designed for a single speaker.

There are many approaches to speaker localization from one or more audio sensors. Microphone array processing can be used to estimate the direction of arrival from one or more sources [15]. [25] demonstrated the use of a microphone array and a single camera to locate a speaker in a scene; they used a time-difference-of-arrival (TDOA)/cross correlation technique to locate the direction of speakers relative to the interface.

All of the above systems improved their performance by combining information from both modalities. However, they were primarily designed for a single user. They did not take advantage of any measurement of low-level audio-visual statistical dependence that could have helped disambiguate who was speaking in tightly packed

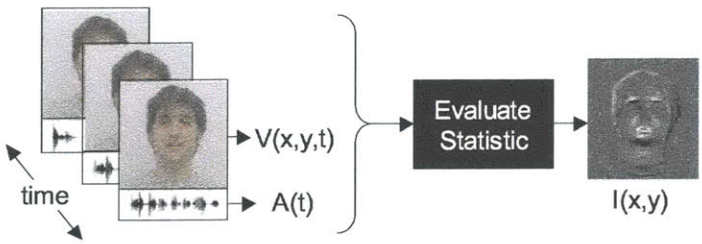


Figure 2-1: Hershey and Movellan's technique.

group of people.

Hershey and Movellan were one of the first to explore the use of audio-visual synchrony/association to locate speaking humans [8]. They define the synchrony as the degree of mutual information,$I(A, V)$, between the audio and video signals:

$$I(A(t), V(x, y, t)) = \frac{1}{2} \log \frac{|C_A(t)||C_V(t)|}{|C_{A,V}(t)|} \tag{2.1}$$

where $A(t)$ and $V(x, y, t)$ are windows of the audio and video signals at time t and at location (x, y) in the image. They make the assumption that audio and video are individually and jointly Gaussian over a small window of time with $|C_A(t)|$, $|C_V(t)|$ and $|C_{A,V}(t)|$ being determinants of the audio, video and joint audio-video covariance matrices. These statistics were calculated recursively as the time window changed. Their audio representation was simply audio energy and the used gray scale intensity values as their video representation. With these one-dimensional representations the mutual information measure is a function of the signals correlation. That is

$$I(A(t), V(x, y, t)) = -\frac{1}{2} \log (1 - \rho(x, y, t)^2) \tag{2.2}$$

where ρ is the Pearson correlation coefficient. They calculate this mutual information or correlation at all locations in the image, treating each pixel independently. This gives a map of audio visual synchrony as shown in Figure 2-1. This map is used by a simple tracker that looks for a localized area of high audio-visual correlation to identify who the current speaker is.

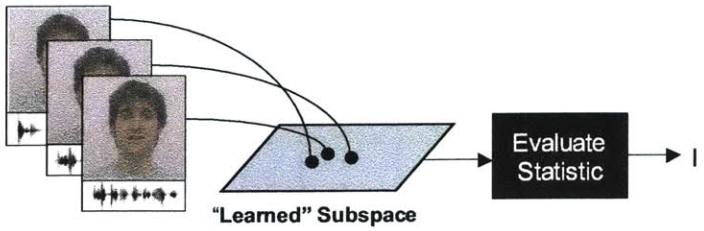


Figure 2-2: Finding an informative subspace

This method works reasonably well and is computationally efficient. However, it makes strong assumptions. The method assumes that each pixel is independent and that they are marginally and jointly Gaussian with respect to the audio. There are many cases in which this is clearly an incorrect assumption. Neighboring pixels around the mouth region will be highly correlated with each other and any head movement unrelated to the audio will cause changes in all pixels on the face.

Improving upon the previous approach Slaney and Covell concentrated on a technique that measured audio-visual synchrony by combining the information from all pixels. Their technique also assumed a jointly Gaussian distribution but exploited canonical correlation analysis to find a projection into a linear subspace that maximized the mutual information between the audio and the video [20]. That is, given audio and video observations, A and V, they found a linear mapping into a lower dimensional subspace.

$$\begin{aligned} Y_A &= h_a^T A \\ Y_V &= h_v^T V \end{aligned} \tag{2.3}$$

This linear mapping, defined by h_a and h_v, was found using canonical correlation analysis [9] which maximizes the correlation, ρ, between Y_A and Y_V. This mapping was learned from a training set composed of aligned faces and their corresponding audio. The synchrony of new data was measured by projecting that data into the learned subspace and measuring correlation. A depiction of this technique is show in Figure 2-2.

Fisher et al proposed an extension of these ideas in [12]. They removed the Gaussian assumption by modeling the joint distribution nonparametrically and they

estimated mutual information using this distribution at each time step. In place of canonical correlation they described an optimization procedure that finds a mapping to a subspace that maximizes their estimate of mutual information.

Both of these learned subspace techniques removed the independent pixel assumption used by Hershey and Movellan. Fisher et al went one step further to remove the Gaussian restriction. All of these techniques are very useful in that they are general enough to be applied to other domains to measure synchrony/association. However, they are strictly online techniques and ignore any information that may be gained from using a model learned from training data. In addition, the experiments presented in [8], [20] and [12] papers focused on the task of localizing speakers and did not address situations in which the audio was not consistent with any part of the video.

Nock et al presented an excellent empirical study of techniques for assessing audio-visual consistency in [23]. In addition to performing experiments dealing with speaker localization, they compared techniques for identifying the "consistent" speaker from a set including multiple confusers. The perform these experiments using a large database of full-face frontal video and audio of speakers, who read sentences from a large vocabulary database.

They compare the use of online measurements of MI to a more sophisticated model based technique that used an HMM trained on held out data. Ultimately they showed that the simple online Gaussian MI technique worked best in the task of determining which speaker most likely produced the observed audio. However, in [22], they concluded that this technique was not suitable for producing an absolute measure that could be used in the task of distinguishing between associated audio and video observations and those that are not. This task of classifying audio and video as being associated or not associated is heavily focused on in this thesis. We aim to explore a more varied set of techniques then tested in [22] in order to extend their search for one better suited to this task.

In most of the previous work presented above, a limited set of audio and visual features were tested. Although Slaney and Covell explored the use of different audio

representations, those working on audio-visual speech recognition have gone further in the search for useful audio-visual features. In the AVSR community there have been differing assumptions about were the relevant visual speech information lies. Some have focused on appearance based features obtained from tracked facial regions. They then use PCA or other techniques borrowed from image compression to preserve the relevant speech information. Others have focused more on shape based features found by tracking lip contours or facial movement. More recently some have explored the use of active appearance models (AAM) [4] which jointly model both shape and texture [18].

There have been a few studies comparing different visual features performance in AVSR systems. [17] showed that AAMs seem to outperform active shape models and give similar performance to purely appearance based features for AVSR. [27] reported that DCT based features were better than lip contours. Although these studies supply some insight into how features affect AVSR performance, which features are best for preserving speech information is still an open question. However, there is a strong argument for the use of appearance-based features in that they can capture information about places or articulation such as tongue and teeth visability. [31] has shown that human speech perception based on lip contours is worse than perception based on the entire mouth region.

Chapter 3

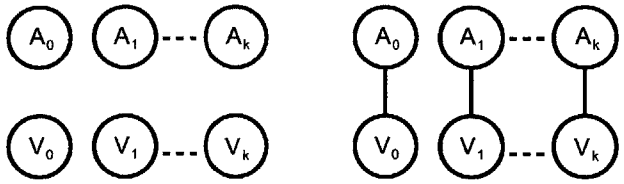
The Speech Correspondence Problem as a Hypothesis Test

Given a scene and multiple people being tracked, we define the speech correspondence problem as identifying who, if anyone, is speaking at any given time. To enable realtime performance, this task must be carried out over a sliding window of time, which limits the number of observations. Observations are obtained using a single camera and microphone and we assume access to a simple tracker that, for every frame, presents us with the face of each individual in the scene.

Decision theory is the natural choice for addressing problems of this form. Thus a solid framework for setting up this problem is in terms of a hypothesis test. This framework assigns each possible configuration of who is speaking as a separate hypothesis. We exploit the fact that each of these hypotheses imply a different model for our observations in order to make a decision.

There are two basic approaches to carrying out this hypothesis test. A fully generative approach uses some parametrization of joint distribution $p(\text{obs}, \text{hyp})$ to form a decision rule that picks the most likely hypothesis. A purely discriminative approach is solely concerned with finding a good decision rule and does not explicitly model the full joint distribution of the observation and hypothesis.

This first half of this chapter discusses the speech correspondence problem in terms of a likelihood ratio test (LRT). Measuring performance of such test is discussed



(a) $\mathcal{H}_0$ (b) $\mathcal{H}_1$

Figure 3-1: Hypothesis test for Audio-Visual Association

in addition to how performance is affected when using different generative model learning techniques. The final section of this chapter explores the use of a specific discriminative approach using Support Vector Machines (SVMs).

3.1 A Generative Approach

We begin with the basic definition of a hypothesis test. The solution to a hypothesis test is described in terms of a decision rule which compares likelihood of the observed data under each hypothesis. This section will pose the speech correspondence problem as a hypothesis test. It will discuss how to measure performance as well as how to obtain a generative model for the audio-visual observations. Lastly this section will show how performance is affected by errors in these models and how one can carry out a hypothesis test when no prior model has been learned.

3.1.1 Audio-Visual Association LRT

We will begin by exploring the subproblem of identifying audio-visual association. Let us define this problem in terms of a binary hypothesis test in which each hypothesis explains a different factorization of a simple graphical model. We assume that our observations will be a series of N consecutive i.i.d. audio and video observations and we will define our hypotheses as

$$\begin{aligned} \mathcal{H}_0 &: \text{Audio and Video are independent} &&: p(\mathbf{a}_k, \mathbf{v}_k) = p_{H_0}(\mathbf{a}_k) p_{H_0}(\mathbf{v}_k) \\ \mathcal{H}_1 &: \text{Audio and Video are dependent} &&: p(\mathbf{a}_k, \mathbf{v}_k) = p_{H_1}(\mathbf{a}_k, \mathbf{v}_k) \end{aligned} \tag{3.1}$$

Figure 3-1 shows the two associated graph factorizations. The graphs clearly show that the audio-visual association corresponds to some audio-visual dependence. We will assume that the prior probability of either of these hypotheses being true is equal. This results in a hypothesis test that compares likelihoods under each hypothesis.

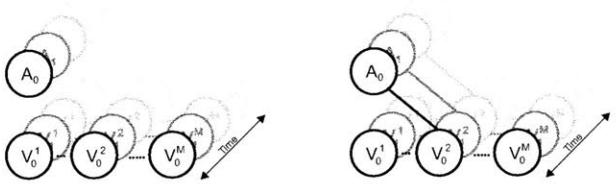
$$\begin{aligned}
p(\{\mathbf{a}\}_N, \{\mathbf{v}\}_N|\mathcal{H}_1) &\underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\lessgtr}} p(\{\mathbf{a}\}_N, \{\mathbf{v}\}_N|\mathcal{H}_0) \\
\prod_{k=1}^{N} p_{H_1}(\mathbf{a}_k, \mathbf{v}_k) &\underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\lessgtr}} \prod_{k=1}^{N} p_{H_0}(\mathbf{a}_k) p_{H_0}(\mathbf{v}_k) \\
\prod_{k=1}^{N} \frac{p_{H_1}(\mathbf{a}_k, \mathbf{v}_k)}{p_{H_0}(\mathbf{a}_k) p_{H_0}(\mathbf{v}_k)} &\underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\lessgtr}} 1 \\
\mathcal{L}(\mathbf{a}, \mathbf{v}) \triangleq \frac{1}{N} \sum_{k=1}^{N} \log \left(\frac{p_{H_1}(\mathbf{a}_k, \mathbf{v}_k)}{p_{H_0}(\mathbf{a}_k) p_{H_0}(\mathbf{v}_k)} \right) &\underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\lessgtr}} 0 \triangleq \eta
\end{aligned} \tag{3.2}$$

We see from the above equation our hypothesis test is simply a likelihood ratio test comparing $\mathcal{L}(\mathbf{a}, \mathbf{v})$ to a threshold η.

This model for our audio-visual observations is clearly a huge simplification. It assumes that speech produces i.i.d audio-visual samples and that it is a stationary process. This conflicts with the approaches taken by most modern speech recognition systems which invest a great deal in modeling the dynamics of speech. However, our goal is much simpler than recognizing speech. Thus, we start with the simplest model and study how well our assumptions hold and what level of performance we can achieve. In addition, this model is consistent with those used in [8], [20] and [12].

3.1.2 Speech Correspondence LRT

The speech correspondence problem is a simple extension of the AV association problem shown above. Given a single audio source we seek to determine which one (or



(a) $\mathcal{H}_0$ (b) $\mathcal{H}_i$

Figure 3-2: Speech Correspondence Hypothesis Test

none) of M video sources $(\mathbf{v}^1..\mathbf{v}^M)$ is jointly dependent with the audio. Our observations will be a series of N consecutive i.i.d observations and we will define our hypotheses as

$$\begin{aligned} \mathcal{H}_0 &: \text{Audio is independent of all M Video sources} \\ \mathcal{H}_i &: \text{Audio and Video source i are dependent.} \\ &\quad \text{All other video sources are independent of the audio} \end{aligned} \tag{3.3}$$

Figure 3-2 shows the two associated graph factorizations. We again assume equal prior probabilities for each hypothesis. Determining which hypothesis is correct requires $M-1$ comparisons / tests. Comparing hypothesis i to j yields.

$$\begin{aligned} p(\{\mathbf{a}\}_k, \{\mathbf{v}^1\}_k, ..., \{\mathbf{v}^M\}_k|\mathcal{H}_i) &\underset{\mathcal{H}_i}{\overset{\mathcal{H}_j}{\lessgtr}} p(\{\mathbf{a}\}_k, \{\mathbf{v}^1\}_k, ..., \{\mathbf{v}^M\}_k|\mathcal{H}_j) \\ \prod_{k=1}^{N} \left(p_{H_i}(\mathbf{a}_k, \mathbf{v}_k^i) \prod_{m \neq i} p_{H_i}(\mathbf{v}_k^m) \right) &\underset{\mathcal{H}_i}{\overset{\mathcal{H}_j}{\lessgtr}} \prod_{k=1}^{N} \left(p_{H_j}(\mathbf{a}_k, \mathbf{v}_k^j) \prod_{m \neq j} p_{H_j}(\mathbf{v}_k^m) \right) \\ \prod_{k=1}^{N} \left(\frac{p_{H_i}(\mathbf{a}_k, \mathbf{v}_k^i) \prod_{m \neq i} p_{H_i}(\mathbf{v}_k^m)}{p_{H_j}(\mathbf{a}_k, \mathbf{v}_k^j) \prod_{m \neq j} p_{H_j}(\mathbf{v}_k^m)} \right) &\underset{\mathcal{H}_i}{\overset{\mathcal{H}_j}{\lessgtr}} 1 \\ \mathcal{L}_{i,j} \triangleq \frac{1}{k} \sum_{k=1}^{N} \log \left(\frac{p_{H_i}(\mathbf{a}_k, \mathbf{v}_k^i) \prod_{m \neq i} p_{H_i}(\mathbf{v}_k^m)}{p_{H_j}(\mathbf{a}_k, \mathbf{v}_k^j) \prod_{m \neq j} p_{H_j}(\mathbf{v}_k^m)} \right) &\underset{\mathcal{H}_i}{\overset{\mathcal{H}_j}{\lessgtr}} 0 \triangleq \eta \end{aligned} \tag{3.4}$$

Comparing $\mathcal{H}_i$ to $\mathcal{H}_0$ is the same as shown in the previous section.

3.1.3 Measuring Performance

We can gain some insight about what this likelihood ratio test is doing by looking at what the expected value of the likelihood statistic is under each hypothesis. The separation of this statistic under each hypothesis gives some indication of how well one can discriminate between them.

Expected Likelihood Statistic for AV Association

When $\mathcal{H}_0$ is true the audio and video are independent and as N becomes large we obtain:

$$\lim_{\mathcal{H}_0, N\to\infty} \mathcal{L}(\mathbf{a}, \mathbf{v}) = E_{p_{H_0}}[\mathcal{L}(\mathbf{a}, \mathbf{v})] = -D\big(p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})||p_{H_1}(\mathbf{a}, \mathbf{v})\big) \tag{3.5}$$

which is the negative KL divergence between the model under $\mathcal{H}_0$ and $\mathcal{H}_1$. Similarly when we look at this ratio when $\mathcal{H}_1$ is true we have:

$$\begin{aligned} \lim_{\mathcal{H}_1, N\to\infty} \mathcal{L}(\mathbf{a}, \mathbf{v}) &= E_{p_{H_1}}[\mathcal{L}(\mathbf{a}, \mathbf{v})] \\ &= D\big(p_{H_1}(\mathbf{a}, \mathbf{v})||p_{H_1}(\mathbf{a})p_{H_1}(\mathbf{v})\big) \\ &\quad + D\big(p_{H_1}(\mathbf{a})p_{H_1}(\mathbf{v})||p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})\big) \\ &= I_{H_1}(\mathbf{a}; V) + D\big(p_{H_1}(\mathbf{a})p_{H_1}(\mathbf{v})||p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})\big) \end{aligned} \tag{3.6}$$

which is the mutual information between the audio and video under hypothesis $\mathcal{H}_1$ plus a model difference term comparing the marginals under each hypothesis. Since KL divergence is strictly positive we see that we get a nice separation between the two hypotheses.

Expected Likelihood Statistic for Speech Correspondence

For the speech correspondence problem we see that under $\mathcal{H}_i$ asymptotically our likelihood ratio becomes:

$$\begin{aligned}
\lim_{\mathcal{H}_i, N\to\infty} \mathcal{L}_{i,j} &= \int p_{H_i}(\mathbf{a}, \mathbf{v}^i) \prod_{m\neq i} p_{H_i}(\mathbf{v}^m) \log\left(\frac{p_{H_i}(\mathbf{a}, \mathbf{v}^i)\prod_{m\neq i} p_{H_i}(\mathbf{v}^m)}{p_{H_j}(\mathbf{a}, \mathbf{v}^j)\prod_{m\neq j} p_{H_j}(\mathbf{v}^m)}\right) d\mathbf{a}d\mathbf{v}^1 ... d\mathbf{v}^m \\
&= \int p_{H_i}(\mathbf{a}, \mathbf{v}^i) \prod_{m\neq i} p_{H_i}(\mathbf{v}^m) \log\left(\frac{p_{H_i}(\mathbf{a}, \mathbf{v}^i_k)}{p_{H_i}(\mathbf{a})p_{H_i}(\mathbf{v}^i)} \frac{p_{H_i}(\mathbf{v}^j)p_{H_i}(\mathbf{a})}{p_{H_j}(\mathbf{a}, \mathbf{v}^j)} \frac{\prod_{m\neq j} p_{H_i}(\mathbf{v}^m)}{\prod_{m\neq j} p_{H_j}(\mathbf{v}^m)}\right) d\mathbf{a}d\mathbf{v}^1 ... d\mathbf{v} \\
&= I_{H_i}(\mathbf{a}; \mathbf{v}^i) + D(p_{H_i}(\mathbf{v}^j)p_{H_i}(\mathbf{a}) || p_{H_j}(\mathbf{a}, \mathbf{v}^j)) + \sum_{m\neq j} D(p_{H_i}(\mathbf{v}^m) || p_{H_j}(\mathbf{v}^m))
\end{aligned} \tag{3.7}$$

When a $\mathcal{H}_i$ is true we obtain a collection of three terms. The first term rewards the statistical dependence between $\mathbf{a}$ and $\mathbf{v}^i$. The second term rewards any divergence between the marginals under $\mathcal{H}_i$ and the joint under $\mathcal{H}_j$ for $\mathbf{a}$ and $\mathbf{v}^j$. The last term adds even more separation from differences in the marginals under each hypothesis. The sum of these terms are strictly non-negative. Furthermore, when $\mathcal{H}_j$ is true we obtain:

$$\begin{aligned}
\lim_{\mathcal{H}_j, N\to\infty} \mathcal{L}_{i,j} &= \int p_{H_j}(\mathbf{a}, \mathbf{v}^j) \prod_{m\neq j} p_{H_j}(\mathbf{v}^m) \log\left(\frac{p_{H_i}(\mathbf{a}, \mathbf{v}^i)\prod_{m\neq i} p_{H_i}(\mathbf{v}^m)}{p_{H_j}(\mathbf{a}, \mathbf{v}^j)\prod_{m\neq j} p_{H_j}(\mathbf{v}^m)}\right) d\mathbf{a}d\mathbf{v}^1 ... d\mathbf{v}^m \\
&= -I_{H_j}(\mathbf{a}; \mathbf{v}^j) - D(p_{H_j}(\mathbf{v}^i)p_{H_j}(\mathbf{a}) || p_{H_i}(\mathbf{a}, \mathbf{v}^i)) - \sum_{m\neq i} D(p_{H_j}(\mathbf{v}^m) || p_{H_i}(\mathbf{v}^m))
\end{aligned} \tag{3.8}$$

which has similar terms but is strictly non-positive.

Receiver Operator Characteristic (ROC)

The amount of expected separation gives us some intuition about the performance of our hypothesis test but is not the whole story. When we are forced to make a decision from a limited set of observations our likelihood ratio will have a different distribution under each hypothesis. In our case $\mathcal{L}(\mathbf{a}, \mathbf{v})$ is the sample mean of a function of i.i.d (the independence is over time) random variables ($\mathbf{a}_k$ and V_k). The central limit theorem tell us as we increase the number of observations, $\mathcal{L}(\mathbf{a}, \mathbf{v})$ will be normally distributed under each hypothesis. The means of these distributions will be

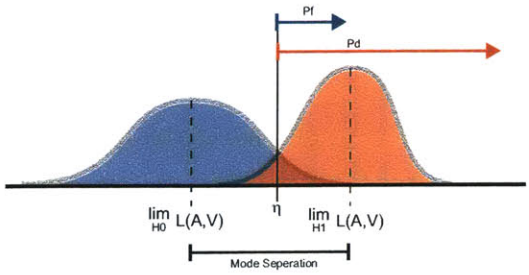


Figure 3-3: General LRT Performance characteristics. The probability of detection is the integral of $\mathcal{L}(\mathbf{a}, \mathbf{v})$'s distribution under $\mathcal{H}_0$ (red,left) from η to ∞.The probability of false alarm is the integral of $\mathcal{L}(\mathbf{a}, \mathbf{v})$'s distribution under $\mathcal{H}_1$ (blue,right) from η to ∞.

Equations 3.5 and 3.6 for hypothesis $\mathcal{H}_0$ and $\mathcal{H}_1$ respectively. In addition the weak law of large numbers says that these distributions will converge to delta functions at their means.Figure 3-3 illustrates these distributions for a generic likelihood ratio test.

The performance of such a test can be described by a receiver-operator characteristic (ROC) which plots the probability of detection, P_d (choosing $\mathcal{H}_1$ when $\mathcal{H}_1$ is true) verses the probability of false alarm, P_f (choosing $\mathcal{H}_1$ when $\mathcal{H}_0$ is true). We obtain different (P_d, P_f) pairs by adjusting the threshold η. Our goal is obtain the pair with the highest P_d and lowest P_f. We know that in the presence of a large number of observations the distribution of likelihood ratio will converge to a delta under each hypothesis and we will obtain a 100% P_d and 0% P_f. However to predict the exact performance using a small number of observations we must know the analytic forms of the distributions shown in Figure 3-3.

3.1.4 Aspects of using a learned Generative Model

The above analysis presents a straightforward method for carrying out this hypothesis test when the forms and parameters of the joint and marginal probability distributions are known. With this knowledge we know the η obtained in Equation 3.2 will produce the best possible (P_d, P_f) pair. In this case we simply plug in our audio and video observations into $\mathcal{L}(\mathbf{a}, \mathbf{v})$ and make a decision.

The problem is that we most likely lack the correct form for these distributions. There are a few ways to address this problem. The simplest thing one can do is to assume a particular parameterizations of the distributions and proceed with the hypothesis test. A second option would be to estimate these distributions from a collection of training data. A third option is to learn the model online from the samples that are being tested.

Using a Model Learned Offline

Appendix A has brief overview two extremes in how to learn a distribution from a set of training samples. It discusses estimating parameters for Gaussian model in addition to a nonparametric approach using a Kernel Density Estimate (KDE). Using one of these techniques we can then carry out our hypothesis test, replacing the densities with our estimates, q. This yields:

$$\mathcal{L}_q(\mathbf{a},\mathbf{v}) = \frac{1}{N}\sum_{k=1}^{N} \log\left(\frac{q_{H_1}(\mathbf{a}_k,\mathbf{v}_k)}{q_{H_0}(\mathbf{a}_k)q_{H_0}(\mathbf{v}_k)}\right) \tag{3.9}$$

Let us see how this will change the performance of our hypothesis test. When $\mathcal{H}_0$ is true (i.e. the correct distribution is $p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})$), asymptotically:

$$\begin{aligned}
\lim_{\mathcal{H}_0, N\to\infty} \mathcal{L}_q(\mathbf{a},\mathbf{v}) &= \int p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v}) \log\left(\frac{q_{H_1}(\mathbf{a},\mathbf{v})}{q_{H_0}(\mathbf{a})q_{H_1}(\mathbf{v})}\right) d\mathbf{a}dV \\
&= \int p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v}) \log\left(\frac{p_{H_1}(\mathbf{a},\mathbf{v})}{p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})}\frac{p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})}{q_{H_0}(\mathbf{a})q_{H_0}(\mathbf{v})}\frac{q_{H_1}(\mathbf{a},\mathbf{v})}{p_{H_1}(\mathbf{a},\mathbf{v})}\right) d\mathbf{a}dV \\
&= -D(p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})||p_{H_1}(\mathbf{a},\mathbf{v})) \\
&\quad + \left(D(p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})||q_{H_0}(\mathbf{a})q_{H_0}(\mathbf{v})) + \int p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v}) \log\left(\frac{p_{H_1}(\mathbf{a},\mathbf{v})}{q_{H_1}(\mathbf{a},\mathbf{v})}\right) d\mathbf{a}dV\right) \\
&= -\{\text{factorization difference}\} + \{\text{model error}\}
\end{aligned} \tag{3.10}$$

and when $\mathcal{H}_1$ is true we will obtain:

$$\begin{aligned}
\lim_{\mathcal{H}_1, N\to\infty} \mathcal{L}_q(\mathbf{a},\mathbf{v}) &= \int p_{H_1}(\mathbf{a},\mathbf{v}) \log\left(\frac{q_{H_1}(\mathbf{a},\mathbf{v})}{q_{H_0}(\mathbf{a})q_{H_0}(\mathbf{v})}\right) dAdV \\
&= \int p_{H_1}(\mathbf{a},\mathbf{v}) \log\left(\frac{p_{H_1}(\mathbf{a},\mathbf{v})}{p_{H_1}(\mathbf{a})p_{H_1}(\mathbf{v})} \frac{p_{H_1}(\mathbf{a})p_{H_1}(\mathbf{v})}{p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})} \frac{q_{H_1}(\mathbf{a},\mathbf{v})}{p_{H_1}(\mathbf{a},\mathbf{v})} \frac{p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})}{q_{H_0}(\mathbf{a})q_{H_0}(\mathbf{v})}\right) dAdV \\
&= I_{H_1}(\mathbf{a};V) + D(p_{H_1}(\mathbf{a})p_{H_1}(\mathbf{v})||p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})) \\
&\quad - \left(D(p_{H_1}(\mathbf{a},\mathbf{v})||q_{H_1}(\mathbf{a},\mathbf{v})) - \int p_{H_1}(\mathbf{a},\mathbf{v}) \log\left(\frac{p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})}{q_{H_0}(\mathbf{a})q_{H_0}(\mathbf{v})}\right) dAdV\right) \\
&= \{\text{factorization difference}\} - \{\text{model error}\}
\end{aligned} \tag{3.11}$$

It is clear from Equations 3.10 and 3.11 that when we pick a model q which is close to the correct, p, we will get the same results shown in Equations 3.5 and 3.6. When q is incorrect, we get extra model error terms which can potentially decrease our separability. However, we know from the previous section that to full quantify our performance we need to know the actual distributions of our likelihood ratio under each hypothesis. If we had these distributions we could search for a threshold η corresponding to the best possible (P_d, P_f) that satisfies an acceptable limit we choose for P_f.

It can be shown that for certain situations and incorrect model choice, the hypothesis test will produce results that are opposite of what we want [11]. In addition, finding a good model requires a consistent density estimator and a sufficient amount of training data under both hypotheses.

We can carry out the same analysis for the full speech correspondence problem. In order to simplify the problem we will choose distributions such that the marginals are the same under both hypotheses (i.e. $q_{H_i}(\mathbf{a}) = q_{H_j}(\mathbf{a}) = q(\mathbf{a})$ and $q_{H_i}(\mathbf{v}^m) = q_{H_j}(\mathbf{v}^m) = q(\mathbf{v}^m)$). This produces the likelihood ratio

$$\mathcal{L}_{q:i,j} = \frac{1}{N}\sum_{k=1}^{N} \log\left(\frac{q_{H_i}(\mathbf{a}_k,\mathbf{v}_k^i)q(\mathbf{v}_k^j)}{q_{H_j}(\mathbf{a}_k,\mathbf{v}_k^j)q(\mathbf{v}_k^i)}\right) \tag{3.12}$$

Under $\mathcal{H}_i$ asymptotically our likelihood ratio becomes

$$\begin{aligned}\lim_{\mathcal{H}_i, N\to\infty} \mathcal{L}_{q:i,j} &= I_{H_i}(\mathbf{a};\mathbf{v}^i) + D(p_{H_i}(\mathbf{a})p_{H_i}(\mathbf{v}^j)||p_{H_j}(\mathbf{a},\mathbf{v}^j)) \\ &\quad - \Bigg(D(p_{H_i}(\mathbf{a},\mathbf{v}^i)||q_{H_i}(\mathbf{a},\mathbf{v}^i)) + D(p_{H_i}(\mathbf{v}^j)||q(\mathbf{v}^j)) - D(p_{H_i}(\mathbf{v}^i)||q(\mathbf{v}^i)) \\ &\quad + \int p_{H_i}(\mathbf{a})p_{H_i}(\mathbf{v}^j)\log\left(\frac{p_{H_j}(\mathbf{a},\mathbf{v}^j)}{q_{H_j}(\mathbf{a},\mathbf{v}^j)}\right) dA d\mathbf{v}^j \Bigg)\end{aligned} \tag{3.13}$$

Similarly under $\mathcal{H}_j$

$$\begin{aligned}\lim_{\mathcal{H}_i, N\to\infty} \mathcal{L}_{q:i,j} &= -I_{H_j}(\mathbf{a};\mathbf{v}^j) - D(p_{H_j}(\mathbf{a})p_{H_j}(\mathbf{v}^i)||p_{H_i}(\mathbf{a},\mathbf{v}^i)) \\ &\quad + \Bigg(D(p_{H_j}(\mathbf{a},\mathbf{v}^j)||q_{H_j}(\mathbf{a},\mathbf{v}^j)) + D(p_{H_j}(\mathbf{v}^i)||q(\mathbf{v}^i)) - D(p_{H_j}(\mathbf{v}^j)||q(\mathbf{v}^j)) \\ &\quad + \int p_{H_j}(\mathbf{a})p_{H_j}(\mathbf{v}^i)\log\left(\frac{q_{H_i}(\mathbf{a},\mathbf{v}^i)}{p_{H_i}(\mathbf{a},\mathbf{v}^i)}\right) dA d\mathbf{v}^j \Bigg)\end{aligned} \tag{3.14}$$

Although the above equations are messy we can clearly see that when we choose the wrong model we get a combination of the optimal answer and some extra corrupting terms which have to do with the model differences between p and q.

Using a Model Learned Online

An alternative to the previous approach is to learn these distributions and perform the hypothesis test at the same time, i.e. we can obtain an estimate of our likelihood ratio:

$$\hat{\mathcal{L}}(\mathbf{a},\mathbf{v}) = \frac{1}{N}\sum_{k=1}^{N} log\left(\frac{\hat{p}(\mathbf{a}_k,\mathbf{v}_k)}{\hat{p}(\mathbf{a}_k)\hat{p}(\mathbf{v}_k)}\right) \tag{3.15}$$

where $\hat{p}(\mathbf{a},\mathbf{v})$, $\hat{p}(\mathbf{a})$, and $\hat{p}(\mathbf{v})$ are consistent estimates of the joint and marginal densities from the same N observations. In this case, we will be estimating our densities

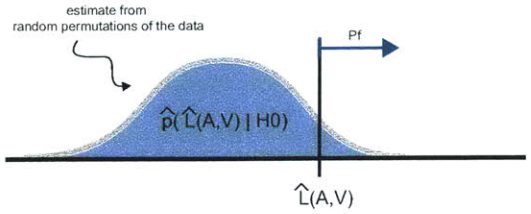


Figure 3-4: Estimating distribution under $\mathcal{H}_0$

from data that comes from one of the two hypotheses, but we do not know which one.

We again look out the asymptotic behavior of our likelihood ratio. When $\mathcal{H}_0$ is true our density estimator will find the joint to be equal to the independent factorization of the marginals, leading to

$$\begin{aligned}\lim_{\mathcal{H}_0, N\to\infty} \hat{\mathcal{L}}(\mathbf{a},\mathbf{v}) &= -D\big(p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})||\hat{p}(\mathbf{a},\mathbf{v})\big) + D\big(p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})||\hat{p}(\mathbf{a})\hat{p}(\mathbf{v})\big) \\ &= -D\big(p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})||\hat{p}(\mathbf{a})\hat{p}(\mathbf{v})\big) + D\big(p_{H_0}(\mathbf{a})p_{H_0}(\mathbf{v})||\hat{p}(\mathbf{a})\hat{p}(\mathbf{v})\big) \\ &= 0\end{aligned} \tag{3.16}$$

Furthermore, $\mathcal{H}_1$ being true results in

$$\begin{aligned}\lim_{\mathcal{H}_1, N\to\infty} \mathcal{L}_q(\mathbf{a},\mathbf{v}) &= D(p_{H_1}(\mathbf{a},\mathbf{v})||\hat{p}(\mathbf{a})\hat{p}(\mathbf{v})) - D(p_{H_1}(\mathbf{a},\mathbf{v})||\hat{p}(\mathbf{a},\mathbf{v})) \\ &= D(p_{H_1}(\mathbf{a},\mathbf{v})||p_{H_1}(\mathbf{a})p_{H_1}(\mathbf{v})) \\ &= I_{H_1}(\mathbf{a};V)\end{aligned} \tag{3.17}$$

where the last step above is due to the fact that if $\hat{p}$ is consistent estimate as N goes to infinity $\hat{p}$ will converge to the true density (under $\mathcal{H}_1$), p_{H_1}. This justifies the use of mutual information estimates from observed data to asses audio-visual association in[8],[20], and [12].

While this shows mutual information is an informative statistic, we still need some way to choose between the two hypotheses. We would like to find some threshold for $\hat{\mathcal{L}}(\mathbf{a},\mathbf{v})$ that gives us the best (P_d, P_f) pair. Again, this requires knowledge of how

$\hat{\mathcal{L}}(\mathbf{a}, \mathbf{v})$ is distributed under each hypothesis. If we had a model of a particular hypothesis, $\mathcal{H}_i$, we could draw random samples and estimate $\hat{p}(\hat{\mathcal{L}}(\mathbf{a}, \mathbf{v})|\mathcal{H}_i)$. The problem is that when are estimating our model online we do not know what hypothesis our observations came from.

Fortunately, however, we should always be able to simulate observations from hypothesis $\mathcal{H}_0$. If we randomly pair our audio and video observations we can simulate independence. We can simply fix the video and randomly permute the order of the audio observations. For each permutation we can calculate $\hat{\mathcal{L}}(\mathbf{a}_{perm}, \mathbf{v})$. Performing multiple permutations we can estimate $\hat{p}(\hat{\mathcal{L}}(\mathbf{a}, \mathbf{v})|\mathcal{H}_0)$. With this knowledge, our hypothesis test would be performed in the following way:

Algorithm 1 Making a decision based on online model estimates

1: Calculate $\hat{\mathcal{L}}(\mathbf{a}, \mathbf{v})$
2: Perform N_{perm} permutations of the observations obtaining a set of $(\mathbf{a}_{perm}, \mathbf{v})$
3: Use these sets to estimate $\hat{p}(\hat{\mathcal{L}}(\mathbf{a}, \mathbf{v})|\mathcal{H}_0)$
4: Find $\hat{P}_f = \int_{\hat{\mathcal{L}}(\mathbf{a},\mathbf{v})}^{\infty} \hat{p}(\hat{\mathcal{L}}(\mathbf{a}, \mathbf{v})|\mathcal{H}_0)$
5: If $\hat{P}_f$ is below some acceptable threshold you choose for false-alarm rate choose $\mathcal{H}_1$, else choose $\mathcal{H}_0$.

Again we carry this analysis further and see how this online estimation method affects the full speech correspondence problem. Using a consistent density estimator $\hat{p}$ gives us our likelihood ratio in the form.

$$\hat{\mathcal{L}}_{i,j} = \frac{1}{N} \sum_{k=1}^{N} \log \left(\frac{\hat{p}(\mathbf{a}_k, \mathbf{v}_k^i)\hat{p}(\mathbf{v}_k^j)}{\hat{p}(\mathbf{a}_k, \mathbf{v}_k^j)\hat{p}(\mathbf{v}_k^i)} \right) \tag{3.18}$$

When $\mathcal{H}_i$ is true $\hat{p}(\mathbf{a}, \mathbf{v}^i) \rightarrow p_{H_i}(\mathbf{a}, \mathbf{v}^i)$, $\hat{p}(\mathbf{a}, \mathbf{v}^j) \rightarrow p_{H_i}(\mathbf{a})p_{H_i}(\mathbf{v}^j)$ and $\hat{p}(\mathbf{v}^m) \rightarrow p_{H_i}(\mathbf{v}^m)$. Using this information and plugging in $\hat{p}$ in place of the q's in Equation 3.13 we see that:

$$\lim_{\mathcal{H}_i, N \rightarrow \infty} \hat{\mathcal{L}}_{i,j} = I_{H_i}(\mathbf{a}, \mathbf{v}^i) \tag{3.19}$$

Similarly under $\mathcal{H}_j$:

$$\lim_{\mathcal{H}_j, N\to\infty} \hat{\mathcal{L}}_{i,j} = -I_{H_j}(\mathbf{a}, \mathbf{v}^j) \tag{3.20}$$

Again we see that when we estimate our densities online our hypothesis test is based solely on the mutual information between the variables. However we are still faced with the problem of how to decide between our hypotheses since we do not know the optimal threshold for decisions in this situation. We can make a simple extension to Algorithm 1 shown in Algorithm 2.

Algorithm 2 Speech Correspondence Hypothesis Test with Online Models

```
S = {}
for i = 0 to M do
  Choose between H_0 and H_i using Algorithm 1
  if Chose H_i then
    S = S ∪ {i}
  end if
end for
if S is empty then
  Choose H_0
else
  Choose H_m̂ where m̂ = arg max_{m∈S} I(a; v^m)
end if
```

3.1.5 Simple Example

Here we will explore the relative performance of the techniques described above through a simple example. We choose a model for our audio and video such that:

$$\begin{aligned} \mathcal{H}_0 &: p(\mathbf{a}, \mathbf{v}) \sim \mathcal{N}(0, C(6, 0)) \\ \mathcal{H}_1 &: p(\mathbf{a}, \mathbf{v}) \sim \mathcal{N}(0, C(4, .6)) \end{aligned} \tag{3.21}$$

where

$$C(\sigma^2, \rho) = \begin{bmatrix} \sigma^2 & \rho\sigma \\ \rho\sigma & \sigma^2 \end{bmatrix} \tag{3.22}$$

We can now study the performance of our hypothesis test by randomly drawing

samples and evaluating $\mathcal{L}(\mathbf{a}, \mathbf{v})$ under each hypothesis. Performing this calculation over many trials will give us a distribution of $\mathcal{L}(\mathbf{a}, \mathbf{v})$ (one for each hypothesis). An ROC curve can then be calculated by adjusting a threshold and calculating (P_d, P_f) pairs from our estimated likelihood distributions.

We follow the same procedure using an incorrect model and evaluating $\mathcal{L}_q(\mathbf{a}, \mathbf{v})$ as our likelihood statistic where:

$$\begin{aligned} \mathcal{H}_0 &: q(\mathbf{a}, \mathbf{v}) \sim \mathcal{N}(0, C(4, 0)) \\ \mathcal{H}_1 &: q(\mathbf{a}, \mathbf{v}) \sim \mathcal{N}(0, C(6, .2)) \end{aligned} \tag{3.23}$$

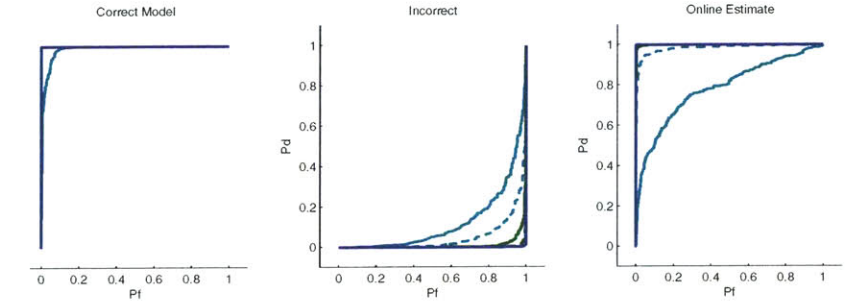
Notice that this model maintains the same graphical factorization but incorrect models the variance and correlation.

Lastly we do the same using $\hat{\mathcal{L}}(\mathbf{a}, \mathbf{v})$ where we estimate our densities online. Here our density estimator assumes a Gaussian distribution and calculates the sample mean and variance.

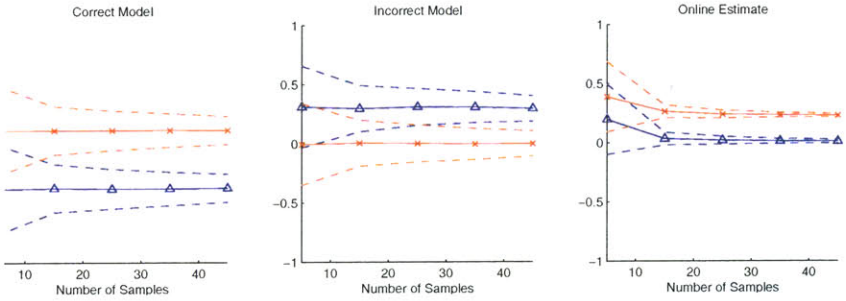
Figure 3-5 shows the relative performance of these tests for varying number of observations. The top row shows ROC curves. Each curve corresponds to a test with a set number of observations. The lighter colors correspond to fewer observations. We see as the number of observations increases the ROC curves converge to the corners.

Using the correct model gives almost perfect performance with more than 10 observations. In addition, for a reasonable number of observations, the online estimation technique performs just as well as having the correct model. However when we incorrectly choose our model we see that our test fails catastrophically, and our results are the opposite of what we would hope for. In this case we could just reverse our decision and obtain reasonable performance. However, when we are performing the test on real data we will not know whether or not we are making correct decisions.

Figure 3-5(b) shows the mean and variance of the likelihood statistics under each hypothesis. The separation between distributions is greatest in the correct model case, and we can clearly see how the two hypothesis switch when we use the incorrect model.



(a) ROC



(b) $L(\mathbf{a}, \mathbf{v})$ distributions

Figure 3-5: Hypothesis test for single audio and video stream

3.2 Discriminative Approach

Much attention can be focused on obtaining the "correct" model under each hypothesis, but if the main goal is to discriminate between consistent and inconsistent speech one may question the necessity of having such a strong generative model.

Thus we can examine this problem in the context of discriminative classification techniques. We define a set of l training vectors $\mathbf{x}_i \in \Re^n, n = N(n_a + n_v), i = 1, ..., l$ with each $\mathbf{x}_i$ containing N audio ($\mathbf{a} \in \Re^{n_a}$) and video ($\mathbf{v} \in \Re^{n_v}$) observations $[\mathbf{a}_1^T \ ... \ \mathbf{a}_N^T \ \mathbf{v}_1^T \ ... \ \mathbf{v}_k^T]^T$. Additionally we have a set of corresponding labels $y_i \in -1, 1$ indicating the class of $\mathbf{x}_i$. The two classes, -1 and 1, will correspond to inconsistent ($\mathcal{H}_0$) and consistent($\mathcal{H}_1$) audio-visual observations. We wish to learn some function, $f(\mathbf{x})$ from training data that can be used to map a new observation $\mathbf{x}$ to the appropriate label with some error criteria we define.

$$f(\mathbf{x}) \underset{\text{class -1}}{\overset{\text{class 1}}{\lessgtr}} b \tag{3.24}$$

Training data for class 1 can easily obtained by recording long video sequences of people speaking, while data for class -1 can be easily created through permutations of the recorded audio and video($\mathbf{x}_i$ shares marginal statistics in each class).

The approach described above discriminates based on a single vector $\mathbf{x}$ which contains all N observations. This removes the i.i.d assumption used in the previous sections. The learned discriminating function may be able to take advantage of correlations over time. However, it would be useful for comparison to define a discriminative approach that is closer to the likelihood ratio test we described using the i.i.d assumption.

We can redefine $\mathbf{x}$ to be a single audio and video observation. The classifier we learn will then give an $f_s(\mathbf{x})$ that can be used to discriminates dependent audio and video from independent based on a single observation. Assuming i.i.d observations

we can define a new classifier based on N samples:

$$\sum_{i=1}^{N} f_s(\mathbf{x}_i) \underset{\text{class -1}}{\overset{\text{class 1}}{\lessgtr}} b \tag{3.25}$$

where, to repeat, each $\mathbf{x}_i$ is an observation $[\mathbf{a}_i^T \mathbf{v}_i^T]^T$.

3.2.1 Support Vector Machine

The particular technique chosen for this paper is Cortes and Vapnik's C-Support Vector Classification (C-SVC) which solves the following problem [32]:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi \geq 0, i = 1, ..., l \end{aligned} \tag{3.26}$$

The basic idea is to find some hyperplane whose normal is $\mathbf{w}$ in some high (perhaps infinite) dimensional space defined by ϕ that separates the training data $\mathbf{x}_i$ into the two defined classes. The optimal hyperplane is the one that maximizes the margin between the classes. Here C is the penalty parameter on the error term. The dual to this problem is:

$$\begin{aligned} \max_{\alpha} \quad & W^2(\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x_j}) \\ & 0 \leq \alpha_i \leq C, \qquad i = 1, ..., l \\ \text{subject to} \quad & \sum_{i=1}^{l} y_i\alpha_i = 0 \end{aligned} \tag{3.27}$$

where:

$$K(\mathbf{x}_i, \mathbf{x_j}) \triangleq \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) \tag{3.28}$$

is called the kernel function. The kernel function defines the inner product in the Reproducing Kernel Hilbert Space (RKHS) that ϕ maps to. To solve the C-SVC we only need to define the higher dimensional space through K rather than explicitly defining ϕ and solve the dual problem. The dual problem shows there is a finite number of α's to find (at most l). The training vectors with nonzero α_i's are called the support vectors. The discriminating function is defined in terms of these support vectors:

$$f(\mathbf{x}) = \sum_{i=1}^{l} y_i \alpha_i K(x, x_i) \tag{3.29}$$

we can also relate the margin M to $\mathbf{w}$ and α:

$$M^2 = \frac{1}{\mathbf{w}^T\mathbf{w}} \tag{3.30}$$

$$\mathbf{w}^T\mathbf{w} = \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \triangleq M^{-2} \tag{3.31}$$

We add the notation M^{-2} to refer to the squared L2-norm of $\mathbf{w}$ (the inverse of the squared margin).

Therefore, to lean an SVM classifier we only need to defined the kernel function, pick a value for C and solve the quadratic program described in Equation 3.27. For this thesis we use a radial basis function (RBF) kernel. This kernel function has single parameter γ and has the form (similar to a Gaussian without a scale factor):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2) \tag{3.32}$$

We use a simple grid search to find good values for C and γ. For each parameter setting in the grid search we perform 4-fold cross validation. This is done by breaking our training data into 4 subsets and iteratively training on 3 and testing on a the remaining subset. We keep the parameters that produce the most correctly classified test samples during cross validation. We then train the SVM will the full training data using these parameters. Figure 3-6 shows a contour plot for a sample grid search.

In Figure 3-7 we show the results of training an SVM on simple Gaussian data.

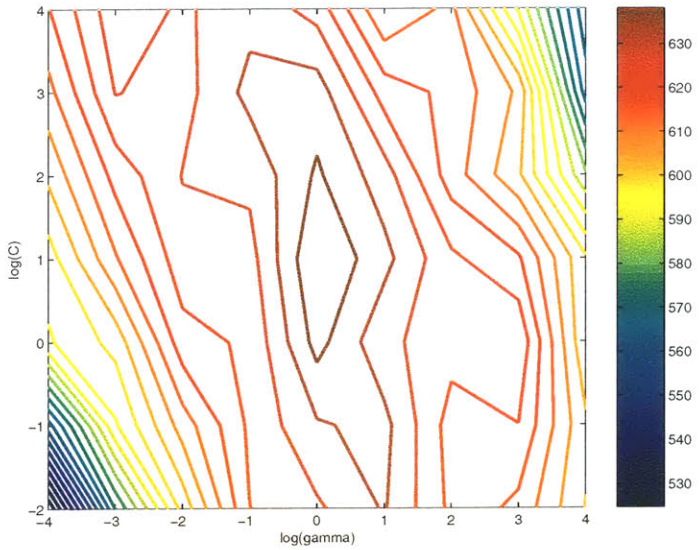


Figure 3-6: Grid search results. The contours represent levels of the number of correctly classified samples during cross validation

Our training data consists of two dimensional samples drawn from a joint Gaussian density with correlation coefficient of .8 for the one class and samples drawn from the product of marginals for the other. Figure 3-7 shows these samples in addition to the SVM's decision boundary, i.e. the boundary described by $f(\mathbf{x}) - b = 0$. For this data we also know the analytic form for the optimal Bayes classifier/hypothesis test. We plot it's decision boundary $\mathcal{L}(\mathbf{x}) - \eta = 0$ as a dotted line in Figure 3-7. We see that the decision boundary found by the SVM is similar to the optimal one. However, this boundary was found with out any prior knowledge of how these samples were generated.

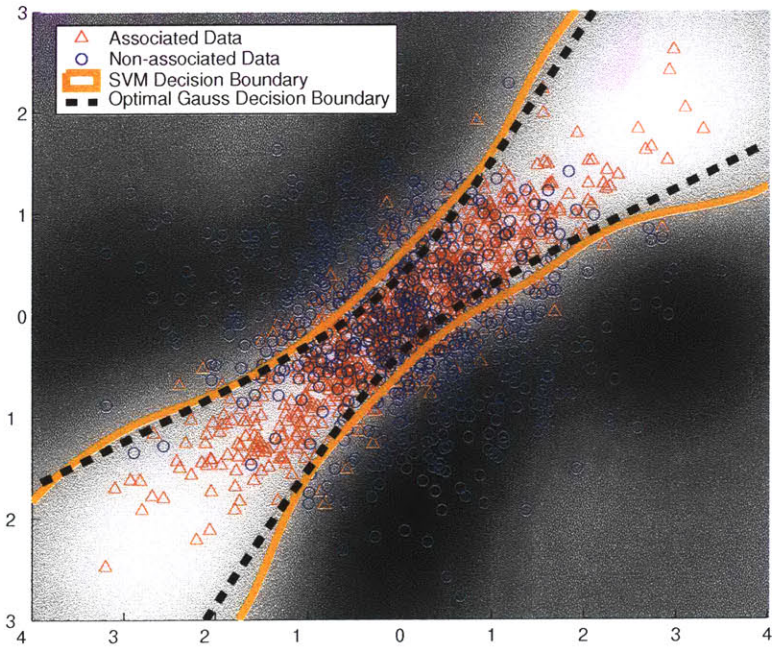


Figure 3-7: Sample SVM Classification on Gaussian Data

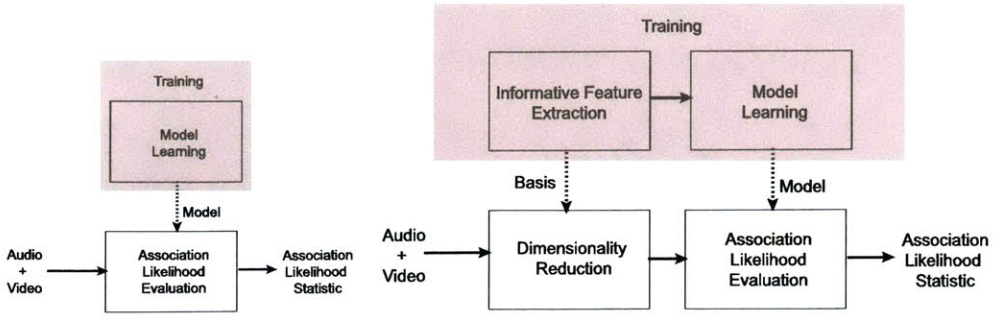
Chapter 4

Informative Subspace Learning

The previous sections have shown how to carry out hypothesis tests to identify audio-visual association and determine speech correspondence. The basic setup for detecting audio visual association is shown in Figure 4-1(a). Audio and video observations come in and a likelihood statistic is calculated. This statistic may be calculated using an online method that estimates mutual information from the input observations, or may make use of a model learned from training examples. The generative approach uses training data to learn distributions for the observations while the discriminative learns a discriminating function.

Unfortunately both of these approaches are sensitive to the dimensionality of the observations. The higher the dimensionality of our observations the more training samples needed to learn a model for them. While theoretically discriminative classifiers should ignore any irrelevant dimensions of the input, in practice as more input features or dimensions are added performance begins to degrade beyond a certain point. These issues lead to having inaccurate models for our observations. It was shown in Section 3.1.4 that the performance of our hypothesis test can be degraded by such inaccuracies. Additionally, high dimensional data comes with the cost of added computational complexity.

Audio and video are inherently high dimensional and thus we must address these issues. We do this by adding a another step to our testing procedure that performs dimensionality reduction prior to evaluating our likelihood statistic, as shown in Fig-



(a) Basic (b) Added Dimensionality Reduction

Figure 4-1: System Components

ure 4-1(b). This step applies separate functions to the audio and video observations that projects them into a lower dimensional informative subspace.

$$\begin{aligned} \hat{\mathbf{a}} &= f_a(\mathbf{a}) \\ \hat{\mathbf{v}} &= f_v(\mathbf{v}) \end{aligned} \tag{4.1}$$

We would like learn these functions not only so that reduce dimensionality but also that they preserve the relevant information for determining audio-visual association. This learning will take place during training.

In this thesis we restrict these functions to be linear. We explore ways to find a compact set of linear bases, $\mathbf{H}_a \in \Re^{n_a \times m_a}$ and $\mathbf{H}_v \in \Re^{n_v \times m_v}$ where $m_a \ll n_a$ and $m_v \ll n_v$, to project the observations onto such that they preserve relative dependency between the audio and video. Using these bases, each feature in the lower dimensional space is a linear combination of the higher dimensional features. This is a reasonable approach if we assume the information in our observations is distributed across dimensions. Such an assumption is likely valid for video of a person's face, where neighboring pixels share information about facial movement. Furthermore, linear bases can ignore particular features in the high dimensional space that may be irrelevant to our problem, such as background pixels in the video.

Only considering linear functions may seem restrictive. However this does not

limit us to measuring linear dependencies. Combining these linear projections with powerful modeling techniques such as KDE or using an SVM classification will still allow us to capture nonlinear relationships in the data.

In this section we discuss different techniques for for learning these linear bases. Section 4.1 discusses techniques for reducing dimensionality such that the dominant axes of variation in the data are preserved. Specifically this is discussed in the context of the standard techniques of principle component analysis (PCA) in Section 4.1.1. Section 4.2 introduces two techniques for finding maximally informative subspaces that preserve dependency between two variables. Lastly Section 4.3 introduces a technique for finding a lower dimensional data representation that preserves the ability to discriminate between dependent and independent data.

4.1 Maximizing Energy

We begin by defining a general goal for dimensionality reduction. In general, given a set of N training examples $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \Re^n$, we would like to find a m dimensional subspace we can project this data onto such that we preserve the most relevant information, where $m < n$. For this thesis we restrict this subspace to be defined by m linear bases $\mathbf{h}_1, ..., \mathbf{h}_m \in \Re^n$. This general description has no meaning unless we define some criteria for what it means to "preserve the most relevant information." In this section we discuss the standard technique of Principle Component Analysis (PCA) where our criteria is to find these bases to best represent our training data in a least squares sense.

4.1.1 PCA

PCA finds a set of linear bases that are ordered by the amount of variation they capture. We can later describe new data by projecting it into a subspace defined by limited set of these bases. We will begin with restricting ourselves to finding a single basis vector such that we minimize its squared distance to each training vector. That

is, we wish to find

$$\boldsymbol{\mu} = \arg\min_{\mathbf{h}} \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{h}||^2 \tag{4.2}$$

It is easy to show that solution to this minimization is simply the sample mean. This is an intuitive result that simple states that if we were restricted to a one dimensional representation of our data we just pick it's projection onto the mean. However this tells us very little about the variation in the data.

In order to ignore the mean, PCA subtracts it from the training data and attempts to represent it as a linear combination of m basis vectors.

$$\mathbf{x}_i - \boldsymbol{\mu} = \sum_{j=1}^{m} a_{j,i}\mathbf{h}_j \tag{4.3}$$

where we restrict $||\mathbf{h}_j|| = 1$ and $\mathbf{h}_j^T\mathbf{h}_k = 0$ for $j \neq k$. Thus $a_{1,i}$, ... , $a_{m,i}$ form m dimensional subspace representation of vector $\mathbf{x}_i$.

The goal of PCA is to minimize the squared distance between the training points and their representation in this m dimensional space. Specifically PCA minimizes the following criteria

$$J_{PCA} = \sum_{i=1}^{N} ||(\mathbf{x}_\mathrm{i} - \boldsymbol{\mu}) - \sum_{j=1}^{m} a_{j,i}\mathbf{h}_j||^2 \tag{4.4}$$

with the same norm and orthogonality constraints on the bases.

If we were given a set of bases it is easy to show that the set of a_j that minimize J_{PCA} are (see [6])

$$a_{i,j} = \mathbf{h}_j^T(\mathbf{x}_i - \boldsymbol{\mu}) \tag{4.5}$$

which is just the projection of the zero-mean training points onto the each basis $\mathbf{h}_j$.

Replacing $a_{i,j}$ with this result and some algebraic manipulation of Equation 4.4

yields new J_{PCA} which is only a function of our basis vectors:

$$J_{PCA}(\mathbf{H}) = -\mathbf{H}^T C_x \mathbf{H} + (N-1) * \sum_{i=1}^{N} ||\mathbf{x_i} - \boldsymbol{\mu}||^2 \tag{4.6}$$

where $\mathbf{H} = [\mathbf{h}_1 \ ... \ \mathbf{h}_m] \in \Re^{n \times m}$ and C_x is the unbiased sample covariance of the training data.

The bases that minimize $J_{PCA}(\mathbf{H})$ are also the eigenvectors of C_x. This is easiest to show for the case where $m = 1$. In this situation it is clear than minimizing $J_{PCA}(\mathbf{H})$ is the same as maximizing $\mathbf{h}_1^T C_x \mathbf{h}_1$. Adding the norm constraint and using a lagrangian multiplier our new objective function is

$$J1_{PCA}(\mathbf{h}_1) = \mathbf{h}_1^T C_x \mathbf{h}_1 + \lambda \mathbf{h}_1^T C_x \mathbf{h}_1 \tag{4.7}$$

Differentiating with respect to $\mathbf{h}_1$ and setting it equal to zero yields

$$\begin{aligned} \frac{\partial J1_{PCA}}{\partial \mathbf{h}_1} &= 2(C_x \mathbf{h}_1 - \lambda \mathbf{h}_1) = 0 \\ C_x \mathbf{h}_1 &= \lambda_1 \mathbf{h}_1 \end{aligned} \tag{4.8}$$

which is a standard eigenvalue problem. Multiplying both sides by $\mathbf{h}_1^T$ and using the unit norm constraint produces

$$\begin{aligned} \mathbf{h}_1^T C_x \mathbf{h}_1 &= \lambda_1 \mathbf{h}_1^T \mathbf{h}_1 \\ \mathbf{h}_1^T C_x \mathbf{h}_1 &= \lambda_1 \end{aligned} \tag{4.9}$$

Thus we see that the $\mathbf{h}_1$ that maximizes $J1_{PCA}$ is the eigenvector of C_x with the largest eigenvalue.

Bringing this analysis back to an m dimensional representation we see that since C_x is real and symmetric the top m eigenvectors will maximize $J_{PCA}(\mathbf{H})$ and will be orthogonal.

A useful geometric interpretation of PCA is that it picks the principal axes of the hyperellipsoid cloud formed by the training examples. These axes are defined by the

(a) PCA on Video Only (b) Joint PCA

Figure 4-2: Comparison of PCA and JPCA. Top 3 components

eigenvectors of the sample covariance matrix C_x. PCA keeps the m axes that capture the most variation. The amount of variation is described by the eigenvalues λ_j. If we were to keep all n principle axes we would be able to described all of the variation in the data. By only keeping the top m we describe

$$pv_x(m) = \frac{\sum_{j=1}^{m} \lambda_j}{e_x} \tag{4.10}$$

percent of the variation/energy, where

$$e_x = tr(C_x) = \sum_{j=1}^{N} \lambda_j \tag{4.11}$$

is the total energy of the training data.

When performing PCA with a particular m we are assuming that the relevant information we wish to keep is contained in the top $pv_x(m)$ percent of the variation. This is not a horrible assumption for audio and video data. For audio we an assume that the major axes of variation will correspond to speaking. For video we hope that lip movements will be captured in the top principal components we keep.

Figure 4-2(a) shows the top three principal components obtained from a video sequence of a single person speaking. For this sequence we see that most variation seemed to come from slight horizontal head movement while speaking. However, this is not where we expect the relevant information for assessing audio-visual synchrony to lie. This is not an unexpected result considering that we did not provide any information about the audio.

4.1.2 Joint PCA

Using PCA we can find separate low dimensional representations for audio and video. This finds the axes of large variation of each modality independent of the other. Ideally we would like to find the axes of maximum covariation between the audio and video. In an attempt to achieve this we can combine our audio and video observations into a single vector and perform PCA.

This is a simple idea, but we must be careful how we combine our observations. It is dangerous to just concatenate our $\mathbf{a}$ and $\mathbf{v}$ vectors on top of each other. We must be aware of the units each modality. If the audio data is represented in units that are larger than the video data PCA will mainly model audio variation and vice versa.

Therefore, when performing Joint PCA (JPCA), we normalize the units of each modality so that they have unit energy. That is, for our audio and video data we perform PCA on

$$\mathbf{av}_k = \left[\frac{\mathbf{a}_k^T}{\sqrt{e_a}} \quad \frac{\mathbf{v}_k^T}{\sqrt{e_v}}\right]^T \tag{4.12}$$

which gives us a set of bases $\mathbf{H}_{av} = [\mathbf{H}_a^T \ \mathbf{H}_v^T]^T$ where $\mathbf{H}_a \in \Re^{n_a \times m}$ and $\mathbf{H}_v \in \Re^{n_v \times m}$. After finding these bases we apply them to new observations as if $\mathbf{H}_a$ and $\mathbf{H}_v \in \Re^{n_v \times m}$ were learned separately.

Figure 4-2(b) shows the top three basis vectors (only the part related to the video) found when performing PCA on the joint AV observations. Here we see the first and third basis emphasis lip movement which is where we expect most of the audio-visual information to lie. This is clearly an improvement. However, the second component still accounts for horizontal head movement. This may be due to the fact that the person was moving synchronously with the audio or that there was so much variation in the video that it ignored the audio component.

4.2 Maximizing Mutual Information

It was shown in the previous section that we can find a lower dimensional representation for our data using PCA. The use of PCA or JPCA was motivated by the

assumption that the important information in our audio-visual observations were found along the major axes of variation or covariation. While this is most likely true, this criteria for preserving variation does not a have a clear link to the hypothesis testing procedure described in Chapter 3. In this chapter we saw how our hypothesis test calculates a likelihood statistic that, as more observations are used, approximates measuring the mutual information $I(\mathbf{a}, \mathbf{v})$. It was also shown that the performance of this test was also dependent on the amount mutual information, particularly when we estimate our generative model online.

As discussed in Appendix B, mutual information is measure of the amount of reduction in uncertainty about one random variable due to another. Furthermore any processing can only reduce the mutual information. That is:

$$\begin{aligned} I(\mathbf{a};\mathbf{v}) &\geq I(f_a(\mathbf{a});\mathbf{v}) \geq I(f_a(\mathbf{a});f_v(\mathbf{v})) \\ &\geq I(\mathbf{a};f_v(\mathbf{v})) \geq I(f_a(\mathbf{a});f_v(\mathbf{v})) \end{aligned} \tag{4.13}$$

where equality is only achieved when $f_a(\mathbf{a})$ and $f_v(\mathbf{v})$ are sufficient statistics for $\mathbf{a}$ and $\mathbf{v}$. Appendix B.4.1 reviews the main criteria for a statistic to be sufficient.

Thus we see that we have a potential conflict. We wish to find a lower dimensional representation for our data in order to improve our ability to learn a good model and thus improve the performance of our hypothesis test. At the same time, we see that by projecting our observations down to this lower dimensional representation we can only throw away information and thus only decrease our maximum performance. The data processing inequality tells us that only way we can avoid this conflict is if the low dimensional representations of our observations are sufficient statistics.

While it may be impossible to find sufficient statistics for our audio and video, we can try to find functions of our data that "nearly sufficient", in that they provide some lower bound on the data processing inequality. In the case where these functions

are restricted to be linear, we wish to find the bases $\mathbf{H}_a$ and $\mathbf{H}_v$ that

$$\begin{aligned} \mathbf{H}_a^*, \mathbf{H}_v^* &= \arg\max_{\mathbf{H}_a, \mathbf{H}_v} I(\mathbf{H}_a^T \mathbf{a}; \mathbf{H}_v^T \mathbf{v}) \\ &= \arg\max_{\mathbf{H}_a, \mathbf{H}_v} I(\hat{\mathbf{a}}; \hat{\mathbf{v}}) \end{aligned} \tag{4.14}$$

which makes $I(\hat{\mathbf{a}}; \hat{\mathbf{v}})$ a lower bound since we know the true mutual information:

$$I(\mathbf{a}; \mathbf{v}) \geq I(\hat{\mathbf{a}}; \hat{\mathbf{v}}) \tag{4.15}$$

In the sections to follow we discuss how to carry out the optimization in Equation 4.14 under different assumptions about the distributions of our observations. Section 4.2.1 discusses the use of Canonical Correlation Analysis (CCA) to solve this optimization when we assume our data is Gaussian. This is approach was used by Slaney and Covell in [20]. Section 4.2.2 discusses the nonparametric approach used by Fisher et al [12]. Lastly Section 4.2.3 discusses how to regularize these techniques.

4.2.1 CCA

We know that for two Gaussian random variables $\mathbf{x}$ and $\mathbf{y}$ that their mutual information is (see Appendix B):

$$I(\mathbf{x}; \mathbf{y}) = \frac{1}{2} \log \left(\frac{|C_x||C_y|}{|C_{[x;y]}|} \right) \tag{4.16}$$

If $\mathbf{x}$ and $\mathbf{y}$ where both 1d random variables then $|C_x| = \sigma_x^2$, $|C_y| = \sigma_y^2$ and $|C_{[x;y]} = \sigma_x^2 \sigma_y^2 - \sigma_{xy}^4 = \sigma_x^2 \sigma_y^2 (1 - \rho^2(\mathbf{x}, \mathbf{y}))$ where $\rho(\mathbf{x}, \mathbf{y})$ is the correlation coefficient between $\mathbf{x}$ and $\mathbf{y}$. Plugging these into Equation 4.16 we obtain

$$\begin{aligned} I(\mathbf{x}; \mathbf{y}) &= \frac{1}{2} \log \left(\frac{\sigma_x^2 \sigma_y^2}{\sigma_x^2 \sigma_y^2 (1 - \rho^2(\mathbf{x}, \mathbf{y}))} \right) \\ &= -\frac{1}{2} \log(1 - \rho^2(\mathbf{x}, \mathbf{y})) \end{aligned} \tag{4.17}$$

which shows us that for 1d Gaussian random variables MI and correlation coefficient, ρ, have a one-to-one mapping.

Using this fact we can show that if we restrict ourselves to single basis vectors for audio and video, $\mathbf{h}_a$ and $\mathbf{h}_v$ our optimization in Equation 4.14 becomes:

$$\begin{aligned}
\mathbf{h}_a^*, \mathbf{h}_v^* &= \arg\max_{\mathbf{h}_a, \mathbf{h}_v} I(\mathbf{h}_a^T \mathbf{a}; \mathbf{h}_v^T \mathbf{v}) \\
&= \arg\max_{\mathbf{h}_a, \mathbf{h}_v} -\frac{1}{2} \log(1 - \rho^2(\mathbf{h}_a^T \mathbf{a}, \mathbf{h}_v^T \mathbf{v})) \\
&= \arg\max_{\mathbf{h}_a, \mathbf{h}_v} \rho^2(\mathbf{h}_a^T \mathbf{a}, \mathbf{h}_v^T \mathbf{v}) \\
&= \arg\max_{\mathbf{h}_a, \mathbf{h}_v} \rho^2(\hat{\mathbf{a}}, \hat{\mathbf{v}}) \\
&= \arg\max_{\mathbf{h}_a, \mathbf{h}_v} \rho((\hat{\mathbf{a}}, \hat{\mathbf{v}})
\end{aligned} \tag{4.18}$$

where the last step can be made since ρ can always be made positive. That is if the extrema of $\rho(\hat{\mathbf{a}}, \hat{\mathbf{v}})$ is negative, and thus a minima, it can always be made positive (and a maxima) by simply negating one of the variables (or projections).

The objective maximization in the last step of Equation 4.18 is the same as the one for CCA, and its solution can be found by solving the following eigenvalue problems (see Appendix C for a full derivation):

$$C_a^{-1} C_{av} C_v^{-1} C_{av}^T \mathbf{h}_a = \rho^2 \mathbf{h}_a \tag{4.19}$$

$$C_v^{-1} C_{av}^T C_a^{-1} C_{av} \mathbf{h}_v = \rho^2 \mathbf{h}_v \tag{4.20}$$

where we pick the $\mathbf{h}_a$ and $\mathbf{h}_v$ with the highest eigenvalue ρ. In fact by finding all the eigenvectors and CCA finds many bases, all ordered by ordering them in terms of their corresponding ρ's. We will find at most $\min(n_a, n_v)$ sets of bases. For each set of bases $\mathbf{h}_{a,i}$ and $\mathbf{h}_{v,i}$ we obtain a set of canonical variates $\hat{a}_i = \mathbf{h}_{a,i}^T \mathbf{a}$ and $\hat{v}_i = \mathbf{h}_{v,i}^T \mathbf{v}$ that have a correlation coefficient of ρ_i.

Each set of canonical variates is uncorrelated and thus, since we are assuming they are Gaussian, independent of all others. That is $\rho(\hat{x}_i, \hat{x}_j) = 0$, $\rho(\hat{y}_i, \hat{y}_j) = 0$ and $\rho(\hat{x}_i, \hat{y}_j) = 0$ for $i \neq j$. Using the fact that information is additive for statistically

independent variables, it can be shown that [2]:

$$\begin{aligned} I(\mathbf{a};\mathbf{v}) &= \sum_i I(\hat{a}_i;\hat{v}_i) \\ &= -\frac{1}{2}\sum_i \log(1-\rho_i^2) \end{aligned} \tag{4.21}$$

Thus, we have some way of telling how much information we lose by only keeping the top m bases given to us by CCA.

4.2.2 Nonparametric Maximization of MI

The previous section showed how to find projections of our audio-visual data that have maximal mutual information under a Gaussian assumption. However, we may want to remove this Gaussian assumption and solve Equation 4.14 using a nonparametric estimate of MI. That is we wish to solve:

$$\begin{aligned} \mathbf{H}_a^*, \mathbf{H}_v^* &= \arg\max_{\mathbf{H}_a,\mathbf{H}_v} \hat{I}(\hat{\mathbf{a}};\hat{\mathbf{v}}) \\ &= \arg\max_{\mathbf{H}_a,\mathbf{H}_v} \hat{I}(\mathbf{H}_a^T\mathbf{a};\mathbf{H}_v^T\mathbf{v}) \end{aligned} \tag{4.22}$$

where

$$\hat{I}(\mathbf{x};\mathbf{y}) = \hat{H}(\mathbf{x}) + \hat{H}(\mathbf{y}) - \hat{H}(\mathbf{x},\mathbf{y}) \tag{4.23}$$

is an estimate of the MI between any two random variables $\mathbf{x}$ and $\mathbf{y}$. The estimates of entropy are of the form

$$\hat{H}(\mathbf{x}) = -\frac{1}{N}\sum_{i=1}^{N} \log \hat{p}(\mathbf{x}_i) \tag{4.24}$$

where $\hat{p}(\mathbf{x})$ is the KDE from N samples of some random variable $\mathbf{x}$.

There is no closed form solution for the problem in Equation 4.22. Fisher et al first showed results on audio visual data using a gradient descent algorithm in [12]. We show the basic flow of the gradient descent algorithm used in this thesis

in Algorithm 3. The input of the algorithm is a set of N pairs of audio and video samples, $\mathbf{a}_i$ and $\mathbf{v}_i$. Throughout this thesis we will refer to this technique as the "Max MI" algorithm. This algorithm involves calculating the gradient of the MI estimate in

Algorithm 3 Basic Max MI Gradient Descent

1: Choose an initial $\mathbf{H}_a = \mathbf{H}_a^0$ and $\mathbf{H}_v = \mathbf{H}_v^0$ (perhaps randomly).
2: Project onto these bases to find $\hat{\mathbf{a}}_i = \mathbf{H}_a^T \mathbf{a}_i$ and $\hat{\mathbf{v}}_i = \mathbf{H}_v^T \mathbf{v}_i$
3: Learn KDEs $\hat{p}(\hat{\mathbf{a}}, \hat{\mathbf{v}})$,$\hat{p}(\hat{\mathbf{a}})$, and $\hat{p}(\hat{\mathbf{v}})$ from these projected samples and calculate $\hat{I}(\hat{\mathbf{a}}; \hat{\mathbf{v}})$
4: **repeat**
5: Calculate $d\hat{\mathbf{a}}_i = \frac{\partial \hat{I}(\hat{\mathbf{a}};\hat{\mathbf{v}})}{\partial \hat{\mathbf{a}}_i}$ and $d\hat{\mathbf{v}}_i = \frac{\partial \hat{I}(\hat{\mathbf{a}};\hat{\mathbf{v}})}{\partial \hat{\mathbf{v}}_i}$
6: Use gradient to define targets for our projected data to move to : $\hat{\mathbf{a}t}_i = \hat{\mathbf{a}}_i + \Delta d\hat{\mathbf{a}}_i$ and $\hat{\mathbf{v}t}_i = \hat{\mathbf{v}}_i + \Delta d\hat{\mathbf{v}}_i$
7: Use LS Regression to find the $\mathbf{H}_a$ and $\mathbf{H}_v$ that minimize $\sum_i ||\hat{\mathbf{a}t}_i - \mathbf{H}_a^T \mathbf{a}_i||^2$ and $\sum_i ||\hat{\mathbf{v}t}_i - \mathbf{H}_v^T \mathbf{v}_i||^2$ respectively
8: Project onto these bases to find $\hat{\mathbf{a}}_i = \mathbf{H}_a^T \mathbf{a}_i$ and $\hat{\mathbf{v}}_i = \mathbf{H}_v^T \mathbf{v}_i$
9: Learn KDEs $\hat{p}(\hat{\mathbf{a}}, \hat{\mathbf{v}})$,$\hat{p}(\hat{\mathbf{a}})$, and $\hat{p}(\hat{\mathbf{v}})$ from these projected samples and calculate $\hat{I}(\hat{\mathbf{a}}; \hat{\mathbf{v}})$
10: **until** We reach our maximum number of iterations N_{iter} or we converge to some maximum $\hat{I}(\hat{\mathbf{a}}; \hat{\mathbf{v}})$

the low dimensional space with respect to each sample of projected audio and video. That is, for the audio we need to calculate

$$d\hat{\mathbf{a}}_i = \frac{\partial \hat{I}(\hat{\mathbf{a}}; \hat{\mathbf{v}})}{\partial \hat{\mathbf{a}}_i} = \frac{\partial \hat{H}(\hat{\mathbf{a}})}{\partial \hat{\mathbf{a}}_i} - \frac{\partial \hat{H}(\hat{\mathbf{a}}, \hat{\mathbf{v}})}{\partial \hat{\mathbf{a}}_i} \tag{4.25}$$

for each of the N samples and an analogous calculation is made for the video. Appendix E details two different approaches for calculating the gradient of the entropy estimate when a KDE is used to represent the distributions.

Targets for our audio and video projections to move to are found by following the calculated gradients in step 6 of Algorithm 3. We then find the $\mathbf{H}_a$ and $\mathbf{H}_v$ that project our original high dimensional samples to these target locations with the least squared error using simple regression. This two step approach of first finding where our projected data should move in order to maximize MI and then finding bases that produce projections as close as possible to those target points, avoids direct calculation of MI gradient with respect to each element of the bases/projections. We

will see in the next section how this setup easily allows us to add regularization to our algorithm.

While Algorithm 3 describes the basic flow of the optimization it does leave out some important details. Notice that we use a step-size of Δ in step 6. In general it is good idea to pick a step-size that is small as possible so we are following the true gradient. If the Δ is made too large we may step over the maximum or oscillate randomly around the solution. However, picking a Δ that is too small means the algorithm will take longer to converge.

In practice we add a step to our algorithm that dynamically adjusts the step-size. At each iteration we compare the previous estimate MI with the most recent. If we haven't improved then we decrease the step size by some percentage and take a step back to use the previous bases. If we have improved, then we increase the step size by the same small percentage for the next step. At the same time we always make sure our Δ is between some $\Delta_{\max}$ and $\Delta_{\min}$. This allows for us to speed up the algorithm when we are improving performance by progressively taking bigger steps. If we stop increasing our MI estimate we then "jam on the brakes" and take smaller steps until we starting seeing improvement again.

It also important to note that the gradient calculations and the MI estimates dependent on the kernel size used when finding the KDEs for the projected data. In practice we have found that using kernel sizes that are too small leads to convergence and local max problems. In general using larger kernels and over-smoothed KDE seem to give better performance. For this reason we either use the Rule of Thumb method (see Appendix A) for picking a kernel size at each iteration or choose some fixed relatively large size.

When using a fixed or relatively constant kernel size we must be take care not to artificially increase our MI estimate by simply scaling our projected data points. We can always maximize MI if we scale our data enough so that each projected point is far enough away from every other point and that the fixed kernel acts like a delta function. To address this problem we restrict our projected data to lie in a hypercube of volume $2^{m_a+m_v}$ so that it can not be arbitrarily scaled to improve the MI estimate.

This restriction is enforced by moving any target locations out side of this hypercube to be on its edge after step 6.

Lastly, an additional change can be made to improve the speed of the algorithm when dealing with a large number of training samples. The more samples we have the longer it will take to learn our KDEs and calculate the gradients. We can reduce computation by starting with a random subsample of the training data and performing the optimization. A new random sampling of the training data is taken every $N_{randiter}$ iterations. Here we are trading off the accuracy of our density estimates and MI gradient estimates for speed. This procedure can also help reduce the probability of getting stuck in a local maximum. However, there is a dangerous risk of overfitting to each subsampling if the number of coefficients for the bases we are learning is greater than the number of samples chosen. We will address this issue in the following section.

4.2.3 Regularization

In the previous section we explored two approaches for finding linear projections of our audio and video data that have maximum MI. CCA provided a closed form solution assuming the data was Gaussian, while the Max MI technique removed this assumption and used gradient descent to find a solution. Both of these approaches use a set of training samples to learn the linear bases. We make the assumption that these training samples are representative of the distribution of observations we will encounter during testing. We expect that when using a large number training samples these algorithms will find linear projections that generalize well to having high mutual information on test data. If, on the other hand, we do not have sufficient training data or it is not representative of our test data our algorithms will not generalize well.

If we have fewer training samples than number of coefficients needed to learn for our bases we can guarantee that CCA and Max MI will overfit. Take, for example, a case when we are looking for single dimensional projections of both audio and video, $\mathbf{h}_a \in \Re^{n_a \times 1}$ and $\mathbf{h}_v \in \Re^{n_v \times 1}$. If we have fewer than $n_a + n_v$ training samples have an undetermined system and we will always be able to find an $\mathbf{h}_a$ and $\mathbf{h}_v$ such that $\mathbf{h}_a^T \mathbf{a}_i = \mathbf{h}_v^T \mathbf{v}_i$ for all i. In other words, we can find a projection that gives infinite

mutual information for the training data.

Even when we have more training samples than the degrees of freedom we may still overfit to our training data. CCA relies on estimating the covariance of our audio and video observations. If we do not have enough training samples we will in turn have inaccurate estimates. In the Max MI algorithm we perform least squares regression to find bases that project our data close to locations that maximize MI. If we have too few samples our regression easily be corrupted by outliers. In general having fewer training samples increase the risk that these algorithms fit to the noise in the data.

To address these problems we introduce regularization as a means to limit the effective number of degrees of freedom. We first discuss a simple technique that uses PCA as a first step dimensionality reduction prior running CCA or Max MI. We then discuss a more theoretically justified approach in which we add L2 regularization to these algorithms.

Using PCA

We have discussed in Section 4.1.1 how PCA can be used for dimensionality reduction. We later argued that its criteria for finding the major axes of variation in the training data does not have a direct connection to the performance of our hypothesis test discussed in Chapter 3 in that the low dimensional subspaces found by PCA may not preserve the information necessary for determining dependence between our observations. However, PCA or JPCA can be used to find a subspace that has a higher dimension than we want, but at the same time throws away the subspace of the training data with very little variation. If we assume that we have not lost the important information in this medium size subspace we can use this as a preprocessing for either CCA or Max MI. This preprocessing step attempts to throw away unwanted noise and reduces the effective degrees of freedom for the algorithms to follow.

We can think of this as a two step dimensionality reduction. The first step forms a subspace that ignores the irrelevant noise in the data, while the second step weights the importance of each basis vector in that subspace in terms of preserving mutual

information. In the end we get a subspace that is a result of linear combinations of the PCA subspace. Choosing the correct number of PCA components to keep in the preprocessing step is an open question. We can try to find the optimal number by using cross validation. Such a procedure is explored in the experimental section of this thesis.

L2 Regularization

Rather than using PCA as described above we can add the more statistically justified L2 regularization to CCA and the Max MI algorithm. As detailed in Appendix D we can interpret L2 regularization in three ways. From a geometric point of view, L2 regularization simply adds a L2 norm constraint on our bases. From a Bayesian perspective this L2 norm constraint is equivalent to having a Gaussian prior on the bases. Lastly, we can see how using this type of regularization corresponds to having a model in which our observations are corrupted by some noise.

Adding regularization to CCA produces Regularized CCA (RCCA) which is the solution to:

$$\begin{aligned} \{\mathbf{h}_a, \mathbf{h}_v\} = \arg\max_{\mathbf{h}_a, \mathbf{h}_v} \ & \mathbf{h}_a^T C_{av} \mathbf{h}_v \\ \text{s.t.} \quad & \mathbf{h}_a^T (C_a + \lambda_a \mathbf{R}_a) \mathbf{h}_a = \mathbf{h}_v^T (C_v + \lambda_v \mathbf{R}_v) \mathbf{h}_v = 1 \end{aligned} \tag{4.26}$$

The main difference from CCA is that L2 regularization terms are added to C_a and C_v. This helps keep these covariance terms full rank and can add prior knowledge to help reduce the effect of having noise estimates.

Adding regularization to the Max MI algorithm changes it's objective function to:

$$\mathbf{H}_a^*, \mathbf{H}_v^* = \arg\max_{\mathbf{H}_a, \mathbf{H}_v} \hat{I}(\mathbf{H}_a^T \mathbf{a}; \mathbf{H}_v^T \mathbf{v}) + \lambda_a \mathbf{H}_a^T \mathbf{R}_a \mathbf{H}_a + \lambda_v \mathbf{H}_v^T \mathbf{R}_v \mathbf{H}_v \tag{4.27}$$

This change in the objective function only changes step 7 in Algorithm 3. Rather than performing Least Squares Regression we perform Ridge Regression to find the the $\mathbf{H}_a$ and $\mathbf{H}_v$ that minimize $\sum_i ||\hat{\mathbf{a}}t_i - \mathbf{H}_a^T \mathbf{a}_i||^2 + \lambda_a \mathbf{H}_a^T R_a \mathbf{H}_a$ and $\sum_i ||\hat{\mathbf{v}}t_i - \mathbf{H}_v^T \mathbf{v}_i||^2 +$

$\lambda_v \mathbf{H}_v^T R_v \mathbf{H}_v$ respectively.

Unfortunately, these algorithms add new unknown parameters λ_a,$\mathbf{R}_a$, λ_v and $\mathbf{R}_v$. While picking the correct values for such regularization parameters is still an open research topic Appendix D gives some insight into how to choose them. In practice, we pick reasonable forms for $\mathbf{R}_a$ and $\mathbf{R}_v$ (the simplest form being identity matrices) and then use cross validation to find the best λ_a and λ_v.

4.3 Maximizing Discriminability

In the previous section we presented techniques for finding an informative subspace that maximizes mutual information. This fit in nicely with the generative approach for detecting audio-visual association. Although we have discussed how discriminative approaches such as the SVM may approximate the likelihood ratio calculated in the generative tests, their primary concern is to discriminate between two classes of data. Thus, it may be useful to look at a more discriminative approach for learning an informative subspace. Here we present a generic technique that searches for a subspace that maximizes the ability to discriminate between two classes of data. It is based on a simple extension of the feature subset selection technique for SVMs presented in [35]. We refer to this approach as Max Discriminate (MD).

We will discuss this approach in terms of a generic classification problem in which we have two classes of observations $\mathbf{x} \in \Re^n$. The goal of MD is to learn a function g parameterized by $\mathbf{H}$ that maps $\Re^n \rightarrow \Re^m$, where $m \lll n$, such that we best preserve the ability to discriminate between classes in this m dimensional subspace. Our new observation are $\hat{\mathbf{x}} = g(\mathbf{x}; \mathbf{H})$. Discussing this problem in the context of an SVM we define :

$$K_{\mathbf{H}}(\mathbf{x}_i, \mathbf{x}_j) \triangleq K(g(\mathbf{x}_i; \mathbf{H}), g(\mathbf{x}_j; \mathbf{H})) \tag{4.28}$$

and

$$W^2(\alpha, \mathbf{H}) \triangleq \sum_{i=1}^{l} \alpha_i - \frac{1}{2} M^{-2}(\alpha, \mathbf{H}) \tag{4.29}$$

with

$$M^{-2}(\alpha, \mathbf{H}) = \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K_{\mathbf{H}}(\mathbf{x}_i, \mathbf{x}_j) \tag{4.30}$$

The goal is to find the best $\mathbf{H}$. However we need to know in what sense these parameters are the "best." Some insight can be obtained from the following theorem [32]:

Theorem 4.3.1 *If images of training data of size l belong to a sphere of size R are separable with the corresponding Margin M, then the expectation of the error probability has the bound:*

$$EP_{err} \leq \frac{1}{l}\left\{\frac{R^2}{M^2}\right\} = \frac{1}{l}E\{R^2 \mathbf{w}^T \mathbf{w}\} = \frac{1}{l}E\{R^2 M^{-2}\}, \tag{4.31}$$

where the expectation is taken over sets of training data of size l.

This theorem shows that the expected error depends on both R and the margin M. This make sense because we can always increase the margin by scaling our data, which will also increase R and not improve our performance. R^2 can be found by maximizing:

$$\begin{aligned} R^2(\mathbf{H}) = \max_{\beta} & \sum_{i=1}^{l} \beta_i K_{\mathbf{H}}(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{l} \sum_{j=1}^{l} \beta_i \beta_j K_{\mathbf{H}}(\mathbf{x}_i, \mathbf{x}_j) \\ & \text{Subject to } \sum_i \beta_i = 1, \beta_i \geq 0 \end{aligned} \tag{4.32}$$

The MD approach performs the following 3 step optimization:

1. For a fixed $\mathbf{H}^0$, find

$$\alpha^0 = \arg\max_{\alpha} \; W^2(\alpha, \mathbf{H}^0) \tag{4.33}$$

2. Using the α^0 found in step 1 find

$$\mathbf{H}^0 = \arg\min_{\mathbf{H}} \; J(\alpha^0, \mathbf{H}) = R^2(\mathbf{H}) M^{-2}(\alpha^0, \mathbf{H}) + \lambda \|\mathbf{H}\|_p \tag{4.34}$$

where $\|\mathbf{H}\|_p$ is some p-norm constraint, and λ is a free regularization parameter.

3. Repeat until some convergence criteria

Step 1 is simply carried out by solving the C-SVM on $\hat{\mathbf{x}}$. What we need to show is how to minimize $J(\alpha, \mathbf{H})$. We do this with a simple gradient descent algorithm. The key components to this gradient descent are as follows:

$$\frac{\partial J(\alpha, \mathbf{H})}{\partial h_k} = R^2(\mathbf{H}) \frac{\partial M^{-2}(\alpha, \mathbf{H})}{\partial h_k} + M^{-2}(\alpha, \mathbf{H}) \frac{\partial R^2(\mathbf{H})}{\partial h_k} \tag{4.35}$$

$$\frac{\partial M^{-2}(\alpha, \mathbf{H})}{\partial h_k} = -\sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \frac{\partial K_{\mathbf{H}}(\mathbf{x}_i, \mathbf{x}_j)}{\partial h_k} \tag{4.36}$$

$$\frac{\partial R^2(\mathbf{H})}{\partial h_k} = \sum_{i=1}^{l} \beta_i \frac{\partial K_{\mathbf{H}}(\mathbf{x}_i, \mathbf{x}_j)}{\partial h_k} - \sum_{i=1}^{l} \sum_{j=1}^{l} \beta_i \beta_j \frac{\partial K_{\mathbf{H}}(\mathbf{x}_i, \mathbf{x}_j)}{\partial h_k} \tag{4.37}$$

For this thesis we restrict ourselves to linear bases such that:

$$\hat{\mathbf{x}}_i = g(\mathbf{x}_i; \mathbf{H}) = H^T \mathbf{x}_i \tag{4.38}$$

where H is in $\mathcal{R}^{n \times m}$. With a RBF kernel this yields:

$$K_{\mathbf{H}}(\mathbf{x}_i, \mathbf{x}_j) = \exp\big(-\gamma (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{H}\mathbf{H}^T (\mathbf{x}_i - \mathbf{x}_j) \big) \tag{4.39}$$

which presents an alternative view of this MD technique. We see that MD is searching over a more flexible space of kernel functions. Normally with an RBF kernel we search over a single parameter γ. With just γ we are restricted to a spherical kernel. In this approach $\mathbf{H}$ can be thought of as the inverse covariance of Gaussian-like kernel, allow more flexibility in the kernel shape. Since in this case we have $\mathbf{H}$ to control the shape and scale of our kernel we γ to be fixed in our optimization.

The RBF kernel has a simple form for its derivative:

$$\frac{\partial K_H(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{H}} = -2\gamma (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{H} \exp\big(-\gamma \| H^T (\mathbf{x}_i - \mathbf{x}_j) \|^2 \big) \tag{4.40}$$

with this, and simple p-norm (we use L2) we can easily calculate the gradient of $J(\alpha, \mathbf{H})$ with respect to $\mathbf{H}$.

Getting back to the original audio video problem, we have an $\mathbf{x}$ which is composed concatenated audio and video frames and we define

$$g(\mathbf{x}_i; \mathbf{H}) = H^T \mathbf{x}_i = \begin{bmatrix} H_a^T & 0 \\ 0 & H_v^T \end{bmatrix} \mathbf{x}_i \tag{4.41}$$

to constrain our $\mathbf{H}$ to be separate audio and video projections.

To improve speed and reduce computation we use stochastic optimization procedure similar to that used for the MMI technique described in the previous section. The optimization picks a random initialization for $\mathbf{H}$ ($\mathbf{H}_a$ and $\mathbf{H}_v$) and follows a simple gradient descent algorithm with a variable step size using Equations 4.35, 4.36 and 4.37. This gradient descent is performed on a random sampling of our training data. A new random sampling of the training data is taken every $N_{randiter}$ iterations. Some other messy details are found in how we choose a fixed value for γ and C. After our first initialization we do a large grid search to find the best values for these parameters. We explored the use of a local grid search every iteration but found the values to remain more or less fixed through the entire optimization. It is important to note that the radius-margin bound described in theorem 4.3.1 is only proven for case in which the training data is separable. However, we found that using this bound on non-separable data in conjunction with the C-SVM and the procedure described above gave us reasonable performance.

This technique is closely related to the feature subset selection procedure in [35]. The only difference is that for feature selection:

$$g(\mathbf{x}_i; \mathbf{H}) = \mathbf{h}. * \mathbf{x}_i \tag{4.42}$$

where $.*$ is the element by element product and the elements of $\mathbf{h} \in [0, 1]$. The optimization procedure is carried ou by relaxing $\mathbf{h}$ to be real valued and imposing an L0-norm constraint to promote sparseness.

4.3.1 Simple Illustrative Examples

In order to help justify the use of MD we present a simple example dealing with a generic classification problem. We define a toy 2 class problem in 2d (u,v) shown in Figure 4-3(a). There are an equal number of training examples for each class (2000 total). We then artificial create a higher dimensional feature representation of

$$\mathbf{x}_i = \begin{bmatrix} u_i + n_1 \\ v_i + n_2 \\ \cdots \\ u_i + n_7 \\ v_i + n_8 \\ n_9 \\ n_{10} \end{bmatrix} \tag{4.43}$$

With $n_1, n_2, n_3, n_4 \sim \mathcal{N}(0, .6)$, $n_5, n_6 \sim \mathcal{N}(0, .7)$, $n_7, n_8 \sim \mathcal{N}(0, 1)$, and $n_8, n_9 \sim \mathcal{N}(0, 2)$. This toy data possess the following key properties:

1. Information inherently lies in a lower dimensional space
2. Information is distributed across multiple dimensions
3. Some features are completely irrelevant
4. The information is not linearly separable

therefore,

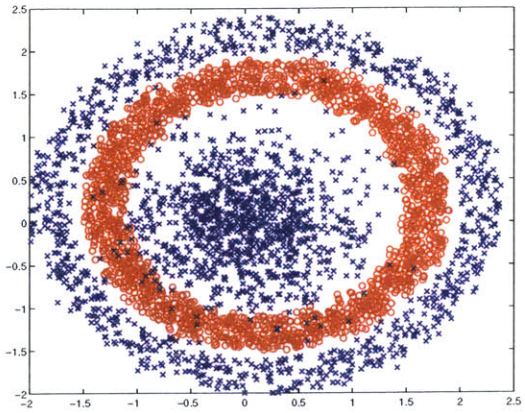
1. Choosing a more compact feature representation makes sense
2. Feature subset selection is not the optimal solution
3. Linear combinations of the features is the correct thing to do in this Gaussian case.

We compared three methods for SVM classification on this data. First we trained on the full feature set, using an RBF kernel and grid search to find the optimal C

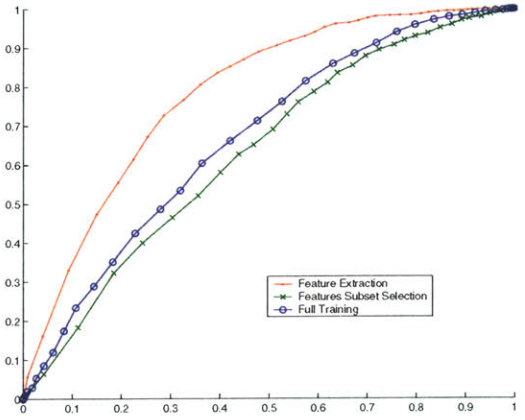
and γ parameters. Second we exhausted all possible subsets of 2 features and found the best solution. Lastly we ran our MD method to extract 2 features (H was 10 x 2). For this simple problem we set λ to zero in Equation 4.34 and rand gradient descent for 100 iterations. Figure 4-3(b) shows the ROC curves for each of the three methods, while Figure 4-3(c) plots the values for the columns of H.

The results show that MD performs significantly better than the other two. The feature subset selection chose the first two features. This makes sense in that they have the smallest amount of noise. What is slightly surprising is that training on the full set was worse than our method. There are two possible explanations for this. Firstly, although we searched over a wide range of γ and C we may not have found the optimal parameters. However, the more likely explanation is that the standard RBF kernel weights each feature equally, but for this toy problem a uniform weighting is suboptimal.

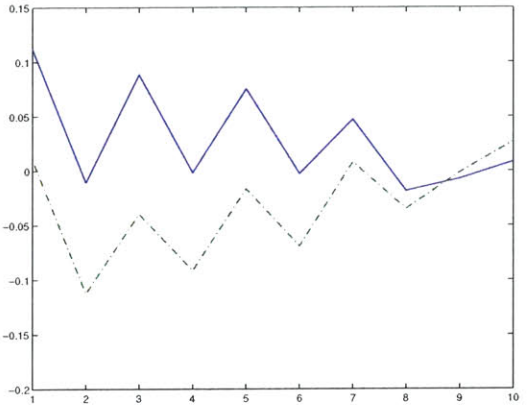
Figure 4-3(c) shows the two features extracted. The first feature is a weighted combination of the u components, while the second is a weighted combination of v components. It also shows how both features have weights close to zero for the irrelevant components in the original feature space. These features are close to optimal, although look a little strange in that the second feature has negative weights on v (Which is fine since the toy problem is symmetric). This example shows a case where MD is the correct method.



(a) 2d data



(b) ROC Curves



(c) Features

Figure 4-3: Circle Test for MD

Chapter 5

Dataset and Preprocessing

For our experiments we use the Massachusetts Institute of Technology's Audio-Visual TIMIT (AVTIMIT) corpus. This corpus contains speakers recorded at 30 frames per second in full DV (720x480) resolution with 16kHz sampled audio. The full corpus contains 223 speakers, 117 male and 106 female. Each speaker is recorded saying 20 utterances/sentences chosen from a total of 450 taken from the TIMIT-SX database. The first sentence is common to all speakers while the other 19 are different for each round. There are 23 different rounds each of which has at least nine different speakers. The last five sentences spoken for each speaker are recorded under different illumination conditions and are ignored in our experiments.

The video in the corpus contains the speaker in front of a constant background. The audio is relatively clean with some background noise giving an average signal to noise ratio of 25 dB. More details about how the corpus was obtained can be found in [7]. Figure 5 shows some sample video frames from this corpus. We consider a single audio-visual observation to be a frame of video and the corresponding audio from the start to end of that frame (1/30 seconds on audio).

Using the raw data from the corpus produces a 345600 (720*480) dimensional video observation and over 500 dimensional audio observation. As discussed in the previous section learning models for data of this dimension is impractical. We introduced methods for finding low dimensional informative subspaces to project this data into. However, these methods would also have poor performance with such extremely

Figure 5-1: Sample Frames from AVTIMIT Corpus

high dimensionality.

It makes more sense to perform some simple preprocessing remove parts of the observations that are clearly irrelevant. Obviously, we should preprocesses the video and only extract the region of the image that contains the person's face. This preprocessing step can also extract useful features commonly used in the domain of audio-visual speech recognition. In the next section we give a quick overview of the face tracker we used. This followed by a brief discussion in Section 5.2 on some of the standard features we tried.

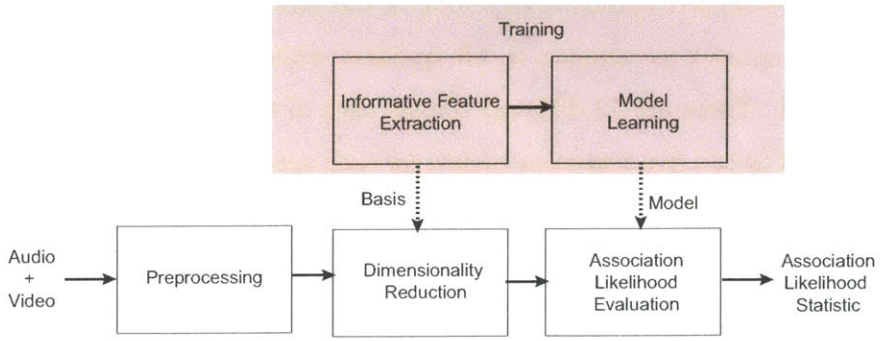


Figure 5-2: Full System Training and Testing Components

5.1 Face Tracking

This thesis focuses on on measuring audio-visual association in human speech. Thus, it makes sense to only use the region of the video that contains a face. We use a simple face tracker to extract the lower facial region of each subject. For each utterance a face detector based on [34] is used to locate the subject's face. This face

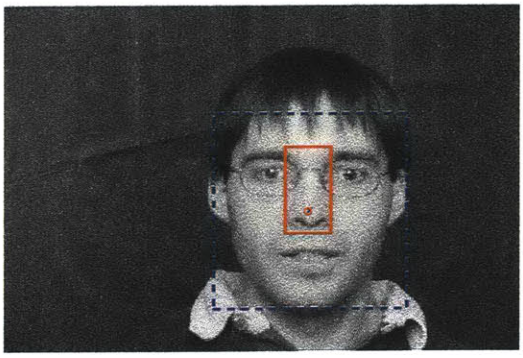


Figure 5-3: Face Tracking

detector supplies the center and width of the face. Once the first detection of a face occurs we use a simple heuristic to extract a small template of the person's nose. This template is then used to track the nose region to the next frame using a sum of squared differences (SSD) error metric. The tracking finds the location in each video frame with the minimum SSD from the template, searching in a window defined by the width of the face and using a quadratic fit for sub-pixel accuracy. Figure 5-3 shows a sample frame with the result of the face detector and the nose template used for tracking.

Using this tracking information we segment out a stabilized view of the lower half of each speakers face and downsample it to a 55 x 100 image per frame. Bi-cubic interpolation is used to keep the subpixel accuracy of the tracker. Figure 5-4 shows some sample extracted lower faces. We see that the tracker does reasonably well at finding and centering the lips and lower jaw of each person. However, the process is completely automated and there was no attempt to align the extracted regions across speakers or even sentences. This provides us a simple tracker that can be run in real time and forces us to consider the effects of alignment problems.

5.2 Audio Visual Features

In addition to tracking the face to help remove any irrelevant information we also make use of some standard audio and video features used in the field of AVSR.

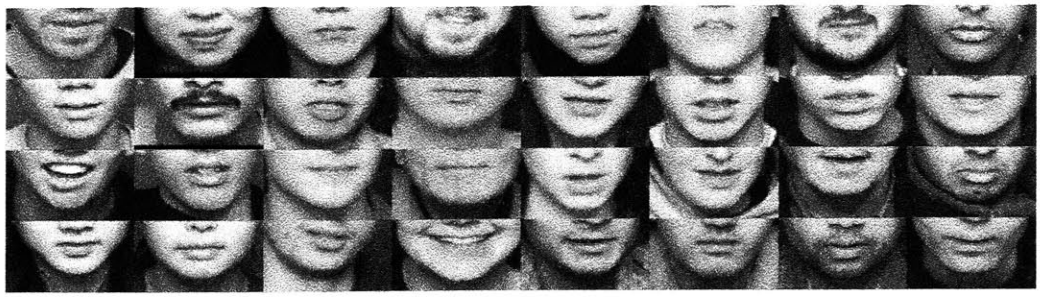


Figure 5-4: Extracted Lower Face

5.2.1 Audio

Our raw audio observation is the sampled intensity waveform obtained by the microphone used for recording. While this raw waveform contains all the information we can capture about what is spoken it also contains any ambient noise or distortions introduced by the microphone and environmental conditions in which the recording took place. We would like to have some representation of the audio that attempts to ignore these unwanted parts of the signal. Furthermore, even if we obtained a noise free recording, there may be information in the speech that can be ignored for our task. There are cues in the speech waveform that can tell us the sex of the speaker, identify who they are, or even if they have a cold. We may be better off finding a transform that ignores this information.

Spectrogram

One of the simplest representation for the audio is a spectrogram. For each frame of audio (corresponding to one video frame) its magnitude spectrum is found by taking the squared absolute value of its discrete-time Fourier transform. This gives us a representation that describes the amount of energy in different frequency bands. The shape of the magnitude spectrum over time can be related to different articulator parameters that describe speech production. A sequence of magnitude spectra frames is referred to as a spectrogram.

This representation is obtained through the invertible fourier transform and is

simply a change of coordinate system. This coordinate system may be better for identifying audio visual association in that we may expect the position of a persons mouth to be related to the amount of energy in particular frequency bands. However, all the information contained in the original waveform is also contained in the spectrogram. In addition, taking the fourier transform of a short window of audio may result in noise estimates of energy at any frequency.

LPC

One way to address the noisy estimates given by a spectrogram is to try and find a smoothed approximation to it. Linear Predicitive Coding (LPC) finds a reduced representation for the shape of a spectrum. LPC finds an autoregressive model for the speech signal which describes a filter that can be used to predict a sample of the speech waveform, $s(t)$, at sample t from a linear combination of previous samples.

$$s(t) \approx c_1 s(t-1) + c_2 s(t-2) + ... + c_k s(t-k) \tag{5.1}$$

These learned coefficients, c_k form an all pole filter. The frequency response of this filter can be viewed as a smoothed version of the signals spectrum. The number of coefficients chosen, k, is the order of the model. Having a lower order model produces smoother estimates, while increasing the order progressively converges to the noisy spectrum. We use a 13th order model, which is commonly used in speech recognition systems.

MFCCs

Another, and perhaps more common, representation used in the speech recognition community is Mel-Frequency Cepstral Coefficients (MFCCs). Much like LPC, MFCCs attempt to obtain a smoothed version of the spectrum. They do this by ignoring large changes in log magnitude of the spectrum. The cepstrum of a signal $s(t)$ is

$$C(q) = F^{-1}\big(\log|F(s(t))|\big) \tag{5.2}$$

where F and F^{-1} represent the Fourier and inverse Fourier transform. In the cepstral domain, q acts as the frequency axis. The cepstral coefficient $C(q)$ at low q represent the power in the slowly varying components of the spectrum while $C(q)$ at higher q represent the rapidly varying components which are assumed to be noise (or perhaps some detailed information about voicing, such as pitch, we wish to ignore). Thus, to obtain a reduced representation for our signal that ignores irrelevant information $C(q)$ is typical truncated to the first 13 cepstral coefficients. MFCCs are formed in similar manor. However, rather than working with a log scale they use a perceptual motivated mel scale prior to taking the inverse Fourier transform in Equation 5.2

5.2.2 Video

Our face tracker does a good job of removing parts of the image outside the mouth and jaw region. The tracker's raw output is simply a sequence of 55 x 100 pixel intensity frames. We assume all the necessary information for determining audio-visual association is contained in these frames. However, we may be able to find a more compact or informative representation with some standard image processing techniques.

DCT

On standard technique for image compression is to use the Discrete Cosine Transform (DCT). The DCT is the real part of the Fourier Transform and transform an image into the frequency domain (for an image we have frequencies in both the x and y directions). For most images most of the energy is concentrated in the lower frequencies. Removing the higher frequency components has a small effect on image reconstruction. The DCT is commonly used in the AVSR community to find a reduced representation for their images by only keeping the top set of DCT coefficients. In this spectral representation it may be easy to identify information about the lips or chin since they would most likely correspond to particular frequencies in the y direction.

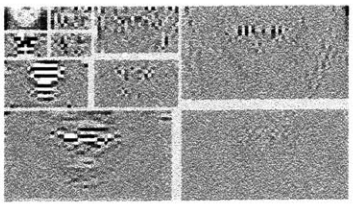


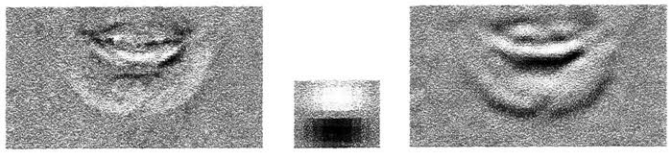
Figure 5-5: Wavelet Pyramid

Wavelet Pyramid

An alternative spectral representation for an image can be found using wavelets. A multi-scale and orientation decomposition of an image can be founding using a set of orthonormal wavelets. Such a decomposition is shown in Figure 5-5. To obtain this decomposition we use Eero Simoncelli's MatlabPyrTools toolbox [30]. We see that the pyramid supplies a representation that separates horizontal and vertical frequencies at various scales. Again we expect the relevant information for speech to lie in the vertical frequency components (horizontal edges).

Difference Features

The features/representations discussed above are applied to each static frame separately. However, it may be useful to incorporate some information about dynamics of the persons mouth. We do this in a naive way by taking finite differences to approximate the derivative of the images over time. We do this over a span of 3 frames in order to keep the same alignment with the static audio frames.



(a) (b) (c)

Figure 5-6: Differential Video Features: (a) Difference Image, (b) Horizontal Edge Filter (c) Horizontal Edge Difference Image

Figure 5-6(a) shows difference image frame. We see that the finite difference

emphasizes motion of the mouth. However, it is somewhat noisy. Since we are primarily interested in the motion of the lips we apply a smooth horizontal edge filter prior to taking the differences. The filter is shown in Figure 5-6(b) and the resulting horizontal edge difference image is shown in Figure 5-6(c).

Optical Flow Statistics

The difference representation described about emphasizes motion of the mouth but also contains information about its position as well. The position of the persons mouth is highly dependent on the tracker as well as the shape of each persons face. Thus, we may want to ignore this location information and find a feature that concentrate solely on representing motion. We do this using an ad hoc optical flow representation.

Figure 5-7: Optical Flow

For each frame we look for a set of 100 good features to track. These features are found using a simple corner detector [28] and are tracked across the next two frames using a Lucas and Kanade tracker [16] using code from OpenCV [13]. For each frame a new set of features are picked an tracked. We summarize this information in a 5 dimensional feature vector containing the mean flow magnitude, the mean vertical flow, the mean horizontal flow, vertical and horizontal flow variance for each frame.

Chapter 6

Experiments

In this chapter we present a series of experiments and their results. Each experiment explores parts of the full training and testing system shown in Figure 5-2. We start by comparing techniques for learning models of AV Association in Section 6.1. We also explore the use of different audio-visual features in this section. We study whether or not performance can be in improved by using techniques for learning an informative subspace in Section 6.2.

6.1 Experiment 1 : Comparing Modeling Techniques for Detecting AV Association

6.1.1 Purpose

In this first set of experiments we compare techniques for detecting AV association. These techniques differ in how observation models are learned and are used. We explore the sensitivity of each technique to the dimensionality and type of input features. Additionally we study the effect of using different window lengths for testing. These tests are performed on synthetic as well as real data from the AVTIMIT database.

6.1.2 Training and Testing Procedure

In this and all other sets of experiments there are two phases, a training phase and testing phase. In the training phase we are given a set of associated audio and video observation pairs. Some of the techniques use this data to learn a model offline, while other online techniques simply ignore it. In the testing phase we are given a new set of audio and video samples and must determine whether or not they are associated. An equal amount of associated and non-associated data is tested.

We produce our training and testing sets using n fold cross validation. This entails breaking our associated audio-visual data into n subsets. For each fold we train on $n-1$ of these subsets and test on the held out data. Given the held out data we derive a test set by making two classes (with corresponding labels). The associated class is simply the held out data, while the non-associated data is generated by pairing the video from the associated class with a randomly chosen segment of audio. This test data is tested by classifying the observations in a sliding a window as associate or non-associated data. Each technique differs in the specifics of how they perform this classification. For each fold we can calculate the probability of error. We record the average probability of error across all folds as well as the variance.

6.1.3 Techniques Compared

As discussed in Chapter 3 we pose the problem of classifying a window of audio-visual observations as associated or non-associated in terms of a simple hypothesis test. This test simply compares a some likelihood statistic of our observations, $L(\mathbf{a}, \mathbf{v})$, to some bias η. The techniques compared in these experiments differ in the way in which they calculate this likelihood statistic and bias. Table 6.1 lists all the techniques compared and their key properties.

This group of techniques can be divided into various categories. The first division separates the generative techniques from the discriminative (SVM). As discussed in Chapter 3 the generative approaches use some explicit model of the joint audio-visual distribution, while the discriminative approaches learn a function (that can be though

Technique	Approach is		Model is learned		Type of model is		Bias, η, is		Assumes
	Generative	Discriminative	Offline	Online	Gaussian	Non-parameteric	Learned	Set significance	i.i.d
Gaussian Model i.i.d.	x		x		x		x		x
KDE Model i.i.d.	x		x			x	x		x
Gaussian MI (Learned Bias)	x			x	x		x		x
KDE MI (Learned Bias)	x			x		x	x		x
RBF SVM i.i.d		x	x			x	x		x
RBF SVM Full		x	x			x	x		
Gaussian MI (w/ Perms)	x			x	x			x	x
KDE MI (w/ Perms)	x			x		x		x	x

Table 6.1: Techniques Compared

of as an approximation of $L(\mathbf{a}, \mathbf{v})$) and a bias that minimize the probability of error. The discriminative approach chosen for this thesis is the C-SVM with a radial basis function (RBF) kernel. A second division can be made between the techniques that learn a model offline from training data, and those that learn the model online. We use model as a generic term for the parameters that control the form of $L(\mathbf{a}, \mathbf{v})$. The online techniques learn these parameters from the samples being tested. For the generative methods the model is the parameters that describe $p(\mathbf{a}, \mathbf{v})$ and for the discriminative approaches the model is the parameters for the SVM. A third division separates the techniques that use Gaussian models from those that have a non-parametric sample-based model (SVM and KDE models). Almost all of the techniques assume that our observations are i.i.d. The one exception is the RBF SVM Full technique which learns a classifier whose input is the entire window of audio-visual observations.

A final difference between the techniques is the way in which they learn and use the bias, η. All the generative techniques which learn their models offline also learn the bias from training data. They do this by calculating the distribution of their parameterized likelihood statistic, $L(\mathbf{a}, \mathbf{v})$ for both associated an non-associated windows of observations and find the bias that minimizes probability of error. However, the training data only contains samples of associated data. Samples of non-associated audio and video are generated through permutations of one of the modalities (i.e. randomly permute the ordering of the audio frames). These synthetic non-associated samples are also used as the non-associated class for training the SVMs.

The online generative techniques define their bias in two different ways. One way is to follow the same procedure for learning the bias that the offline techniques use. The distribution of the likelihood statistic is calculated for both classes of data and a bias is picked to minimize error. However, unlike the offline techniques these techniques calculate their likelihood statistic for each window of data using a model learn solely from that data. In some ways this is not a fully online method in that it requires some training data to learn the bias, but at the same time allows for the model to change according to the samples being tested.

The last two techniques in Table 6.1 are purely online techniques. They use their

bias in a completely different way; the testing procedure is not a simple hypothesis test but rather a significance test. We follow an online testing procedure similar to that described in Algorithm 1. We first estimate MI for data being tested, and then perform a set of permutations on one of the modalities to synthetical produce non-associated data. For each permutation we estimate MI. We define our significance level as 1 minus the number of times the non-permuted MI estimate was less than the permuted MI estimate divided by the number of permutations, or simply $1 - \hat{P}_f$ (*Note that this is the opposite of what is normally referred to as significance in most hypothesis testing literature, but allows us to simply replace our likelihood statistic with this value and threshold so that values higher than the threshold correspond to a detection. We apologize for any confusion this may cause*).

For our tests we choose to perform 15 permutations for each test. This relatively small number of permeations allowed us to maintain a reasonable speed for testing procedure (the computation grows linearly with the number of permutations). We also choose our bias to be a set value of .85. That is, we classified the data as being associated if the estimated $\hat{P}_f$ was less than 15 %. This is a reasonably high P_f threshold to choose, but we must consider that our estimate of $\hat{P}_f$ will be quantized according to the number of permutations we carry out. In addition to reporting results using this significance threshold of .85, we also calculate the performance using all possible significance levels and record the best one.

6.1.4 Variables

In addition to comparing the different techniques described above, we also explore how changes in the following parameters affect performance:

- The dimensionality of our input.
- The window size we test over.
- The type of input features used.

The dimensionality of our input is controlled simply by keeping a set number of PCA features obtained from the training data. Window sizes of 15, 31, 45 and 63 (approximately .5, 1, 1.5 and 2 seconds of data) are tested and compared. A small subset of features is compared in this round of experiments. A more diverse set is later compared using the best performing techniques.

6.1.5 Results: Synthetic Data

We start by testing these techniques on a simple set of synthetic data. This data allows us to explore some basic characteristics of these techniques and acts as a transition between theory and practice. The first set of data is generated from a simple Gaussian model such that:

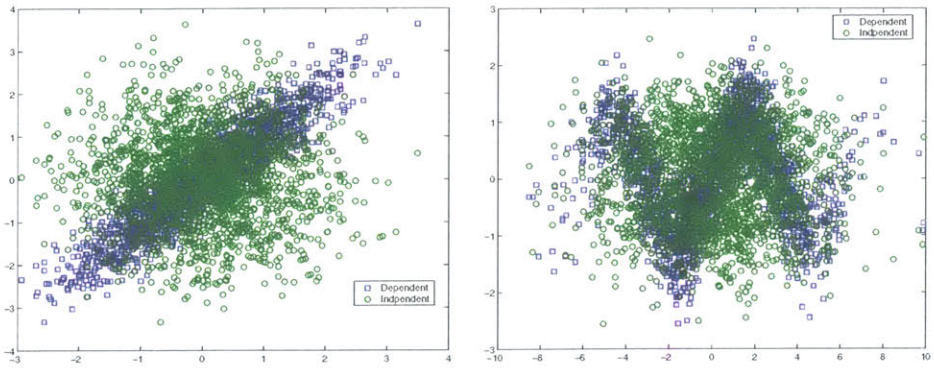
$$\begin{aligned} A &= \phi + n_a \\ V &= \phi + n_v \end{aligned} \tag{6.1}$$

where ϕ is drawn from $N(0,1)$ and n_v is drawn from $N(0,.2)$. For this simple case we set n_a to be zero. From this information we know the mutual information between A and V is .8959 nats and that $-D(p(A)p(V)||p(A,V)$ is -4.1041. The second set of data introduces a nonlinearity such that

$$\begin{aligned} A &= \phi_{nl} + n_a \\ V &= \sin(\phi_{nl}) + n_v \end{aligned} \tag{6.2}$$

In addition to the nonlinearity we increase the variances of ϕ_{nl} to be larger ($N(0,8)$) in order to capture more of the nonlinear structure. Figure 6-1 shows plots of these two datasets with samples drawn from the joint distribution as well as random samples drawn from their marginals.

Training and test data were i.i.d samples drawn from these distributions. We used 1500 training and 500 test samples. We produce non-associated test data through permutations. Thus we have data that is low dimensional, conforms to our i.i.d assumption, and has known distributions. This allows us to explore some of the basic



(a) Synthetic Gaussian Data (b) Synthetic nonlinear Data

Figure 6-1: Synthetic Data

properties of our modeling and testing techniques.The suite of tests were performed for varying window sizes. We plot the minimum probability of error found for each test.

Figure 6-2 shows results for the Gaussian data. We see from the results for this simple dataset that all of the offline model learning techniques performed almost equally well, approaching perfect performance with a window of 10 or more test samples. We know that a Gaussian model is optimal in this situation, but we see that the non-parametric approaches perform equally well for this simple case. This can be attributed to the fact that the dimensionality of the data is extremely low and the non-parametric techniques obtained enough samples to capture the Gaussian structure of the data.

Figure 6-2 also clearly shows a gap between the online and offline techniques. As the number of test samples increase this gap disappears. Since the data is i.i.d, given enough test samples an accurate density can be learned online. In addition, the results show that the online Gaussian MI test outperformed the non-parametric technique (KDE Online MI) for the smaller window sizes. This agrees with theory and ones intuition in that the Gaussian assumption is correct for this situation and the non-parametric technique requires more samples learn an accurate representation of this density. That is, the non-parametric techniques have more capacity to learn

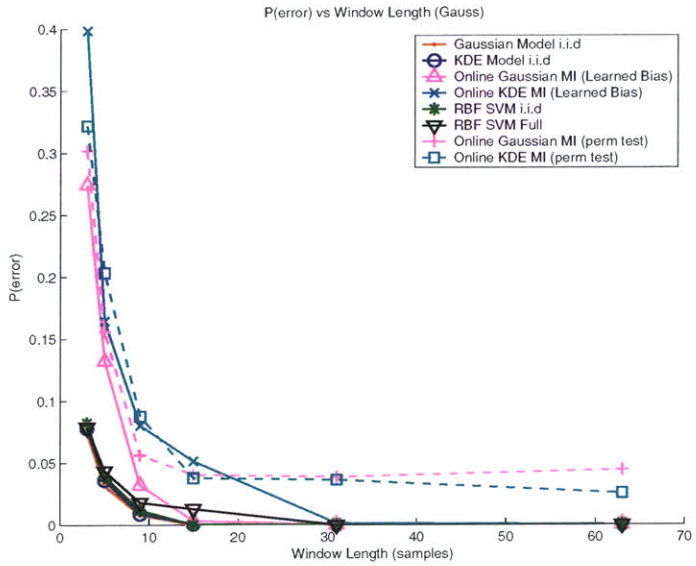


Figure 6-2: Performance on Gaussian Data

complex structures, but pay for this capacity by requiring more data.

It is also interesting to note that the purely online methods that use permutation tests to assess significance have the same performance as the online techniques that learn a bias for the shorter window lengths. However, the performance of the significance testing techniques levels out at 5% probability of error. This is due to the fact that we fix a significance level at 85 %. We achieve perfect performance with these techniques if we raise our significance threshold to 98%. This shows us that we should increase our significance threshold as the window length increases (if we have i.i.d samples).

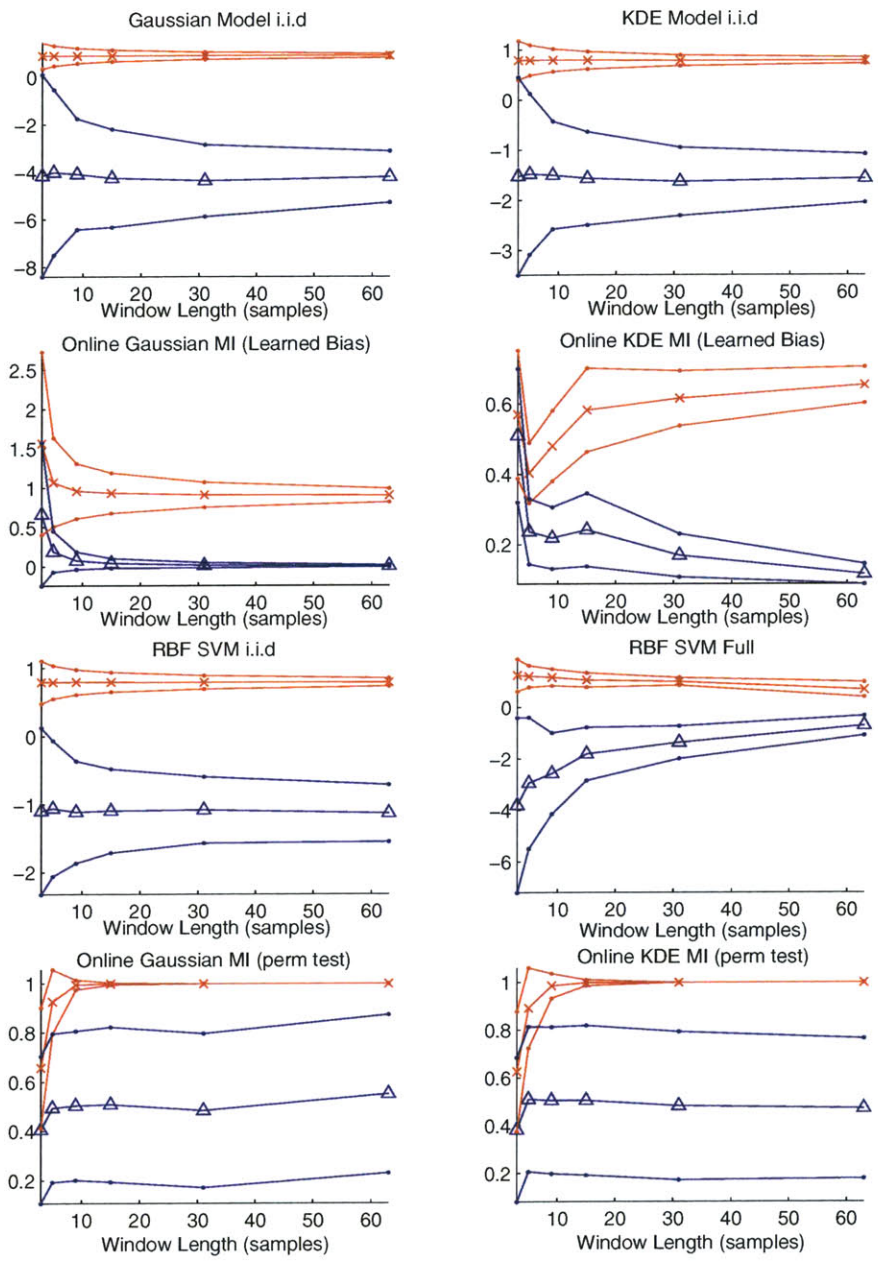


Figure 6-3: LR Plots on Gaussian Data. Shows the mean and standard deviation of the likelihood ratio under hypothesis H_1 (associated, plotted in red stars, upper) and H_0 (non-associated, plotted in blue triangles, lower) versus window length.

We further explore the behavior of these techniques by looking at the distribution of the likelihood statistics for both associated and non-associated data and compare their behavior to the theory outlined in Section 3.1.4. Figure 6-3 shows for each techniques the mean and variances of the likelihood statistic for each class. A clear separation between classes is observed for each test as the window size is increased. The results for the offline and online Gaussian techniques are consistent with theory. The offline Gaussian model's likelihood ratio converges to approximately .9 for associated audio and video and -4.2 for the non-associated class. These numbers are consistent with the actual MI and $-D(p(A)p(V)||p(A,V))$ for this data. Additionally it is shown that the online Gaussian MI technique's likelihood ratio converges to the actual MI for the associated data and zero for the non-associated data as described in Section 3.1.4. It is also interesting to note that the SVM i.i.d and KDE models exhibit almost identical behavior. Both have a smaller separation between likelihood ratio means and have larger variances than the Gaussian techniques. Similarly the online KDE technique has a smaller separation between classes and seems to require more samples before it converges to the offline KDE techniques performance.

We also observe a strange phenomenon in the RBF SVM Full technique. It seems as the number of samples increase the separation between the likelihood statistic gets smaller, but a the same time the variance also decreases. This behavior can be attributed to the difficulty of training an SVM with high dimensional input features. Lastly, in the plots for the online tests with permutations Figure 6-3 displays the distributions of significance rather than the likelihood ratio. We clearly see that as the window length increases the significance when testing the associated class converges to 1. The distribution of significance for the non-associated class has a mean of 50 % and a variance that only decreases slightly with increased window length. This again shows that if we are dealing with i.i.d. samples as we increase the number of samples we test with we should also increase our significance threshold.

Next we look at the performance on the nonlinear synthetic data shown in Figure 6-4. These results clear show that using a Gaussian model is suboptimal in this situation. Both Gaussian techniques have almost random performance. The clear winners

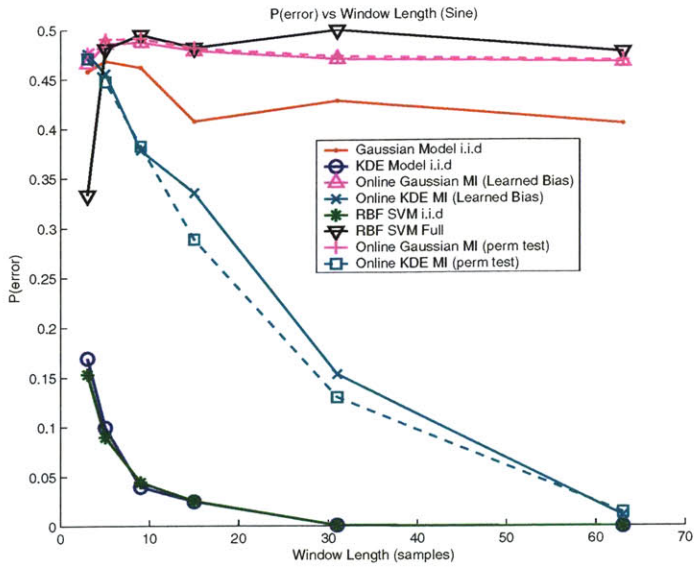


Figure 6-4: Performance on Nonlinear Data

are the KDE and SVM i.i.d techniques. The figure also shows a gap between the online and offline techniques. The online KDE MI techniques dramatically improve as the window size increases. However, even with 31 samples it still has a 14% error while the offline KDE technique is nearly perfect.

The offline RBF SVM Full classifier again seems to exhibit some interesting behavior. It had the third best performance with a small window length, but rapidly became one of the worst performers with window lengths greater than 5 samples. We can again attribute this to the fact that an increase in the window size corresponds to an increase in the input feature dimension for the SVM. We may not have had enough training examples or may not have searched over enough SVM parameters to learn a useful classifier.

Figure 6-5 shows the mean and variances of the likelihood statistic for this data. It is clear from this figure that the non-parametric techniques were the only ones capable of modeling this data. Again we see that the offline KDE and the SVM i.i.d models have similar results. However we see that the SVM i.i.d model likelihood statistic has a slightly larger variance. The online KDE model's likehood statistic becomes

more separable as the window length increases. However, it's behavior is not perfect. Theory tells us that as the window size increases the likelihood ratio should approach zero for the non-associated data and the actual MI for the associated data. While this may be true in the limit, these results show that even with 60 samples of this nonlinear data the online KDE MI calculation has not converged.

Using synthetic data has allowed us explore these various modeling techniques in a controlled environment in which we could predict performance. We have shown experimental results that agree with the theory discussed in Chapter 3. The parametric approaches require fewer training samples and perform better when the implied assumptions (Gaussian) are valid for the data. However, with a sufficient amount of training data the nonparametric approaches worked equally well, and exhibited the added flexability to handle complex non-Gaussian distributions. Lastly we experimentally verified the performance advantage of having learned a prior model over using online density estimation.

All of these results were obtained using data that conforms to i.i.d assumption. Next we will explore how well these modeling techniques will perform on real data. This will give further insight into the tradeoffs between techniques and how well our assumptions hold for audio-visual speech.

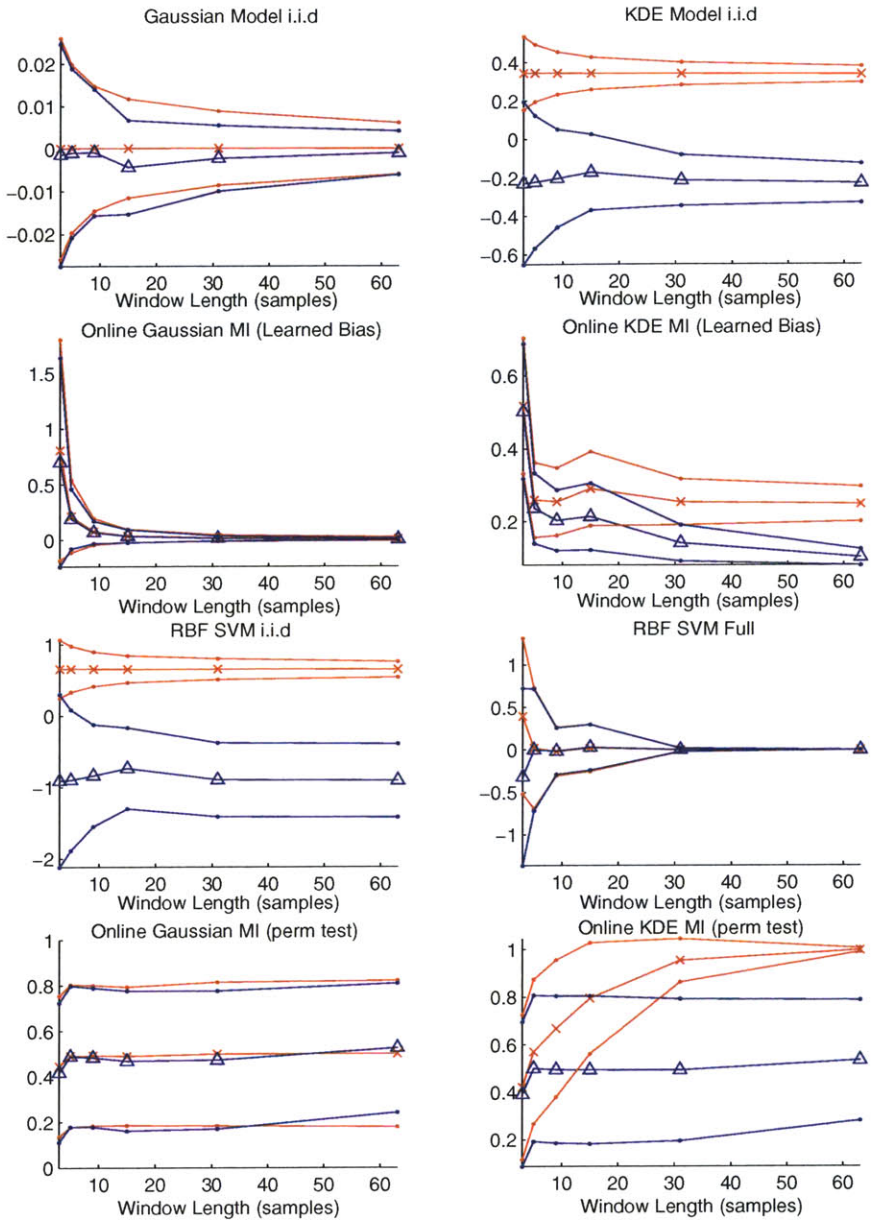


Figure 6-5: LR Plots on Nonlinear Data. Shows the mean and standard deviation of the likelihood ratio under hypothesis H_1 (associated, plotted in red stars, upper) and H_0 (non-associated, plotted in blue triangles, lower) versus window length.

6.1.6 Results: Audio-Visual Data from AVTIMIT

We now turn to testing these techniques on data taken from a small subset of the AVTIMIT dataset. We follow a similar testing procedure to the one used on the synthetic data. One difference is that our non-associated test data is no longer obtained through permutations but instead made by combining video with randomly chosen mismatched audio. In addition, we explore the use of different input features in terms of how the data is preprocessed and the dimensionality. For these experiments we use PCA to control the input dimensionality. We record performance when testing and training on the same individual as well as situations in which the individuals used in training are different than those used in testing.

The goal of these experiments are to answer some basic questions. The first of which is simply to find out what level of performance we can achieve in detecting association for this data. Some other questions we wish to address are: How well does our i.i.d. assumption hold? How does the dimensionality of the observations affect the performance of each technique? Can we use simple Gaussian models or does this data require more complex non-parametric techniques? Can we learn a person-specific model for audio-visual association? Is it possible to learn a generic model for audio-visual association? Does our specific choice of features affect performance?

We test the ability to learn a person-specific model on separate individuals. Each individual contributes around 1500 frames/samples of audio-visual data. We perform 4 fold cross validation on these individuals, each fold training on 3/4 of the data and testing on the held out 1/4. In order to test our ability to learn a generic model we use a set of 10 people in which we perform 5 fold cross validation with each fold training on 8 people and testing on 2.

Note that the PCA components used to control our input dimensionality are learned only from the training data in each fold. Each experiment keeps a specified number, d, of these PCA components. PCA is performed separately for each modality, giving separate d dimensional representations of the audio and video producing a $2d$ joint AV space. The same principal components used to make the $2d$ dimen-

sional training set representation is also applied to the test set in each fold. In these experiments we explore $d = 1$, 2 or 5.

All our raw experimental results are recorded in Appendix F. Below we discuss some of the key results. We organize this discussion in terms of the input features explored.

Using Static Features (Pixel Intensities and MFCCs)

We start by using the raw pixel intensities of the speakers lower face as the video features and MFCCs as the audio features. The raw results can be found in the Appendix Tables F.1, F.2, and F.3. We summarize the best performance achieved by each technique and the associated testing parameters in Table 6.1.6 for the case in which we test and train on the same person.

We see that we achieve the best performance using a Gaussian Model using 5 PCA coefficients and a long window length. This technique produced a 14 % error for person 1 and a 6 % error for person 2. This shows that using pixel intensities and MFCCs it is possible to learn a person-specific model that gives us a reasonable level of performance. We also see that, much like what we saw with the synthetic data, all of the techniques generally did better when using longer window lengths. However, little more can learned from this summarization of the best performances.

We can learn more about the techniques tested by showing their performance as a function of the window length and input dimension. Figure 6-6(a) shows the performance of each technique as a function of the number of PCA components used for a fixed window length of one second. This figure shows us some simple trends. First we notice that the online techniques performance decreases as the input dimension increases. This is easily explained by the fact that with higher dimensional data we need more samples to estimate its density. These online techniques only have 31 samples with a window length of one second. A second trend shows that the offline Gaussian, KDE and SVM i.i.d. techniques improve as the dimensionality increases. The more PCA components used the less information we lose from the original data. The more information we have the better our learned model becomes.

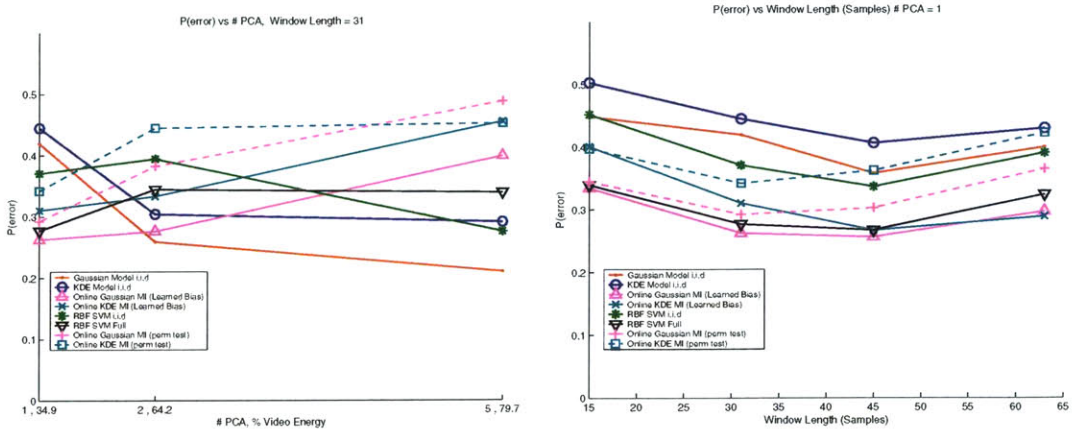
(a) Person 1

Technique	Best Avg P(error)	# PCA	Win Length
Gaussian Model i.i.d.	0.14	5	45,63
KDE Model i.i.d.	0.26	2	63
Gaussian MI (Learned Bias)	0.28	1	45,63
KDE MI (Learned Bias)	0.29	1	45,63
RBF SVM i.i.d.	0.26	5	63
RBF SVM Full	0.30	1	45,63
Gaussian MI (w/ Perms)	0.30	1	63
KDE MI (w/ Perms)	0.30	1	63

(b) Person 2

Technique	Best Avg P(error)	# PCA	Win Length
Gaussian Model i.i.d.	0.06	5	63
KDE Model i.i.d.	0.26	5	63
Gaussian MI (Learned Bias)	0.20	1	45,63
KDE MI (Learned Bias)	0.25	1	45,63
RBF SVM i.i.d.	0.32	2	45,63
RBF SVM Full	0.27	1	45
Gaussian MI (w/ Perms)	0.16	1	45
KDE MI (w/ Perms)	0.23	1	31

Table 6.2: Person-specific results summary using pixel intensities and MFCCs. The best performance for each technique is summarized. The number of principal components and window length (in samples) used to achieve the best probability of error are listed. A common separated list of parameters indicates that each parameter achieved similar performances.



(a) Pe vs Input Dimension

(b) Pe vs Window Length

Figure 6-6: Performance Summary Plots for Person 1 (Pixel Intensity, MFCCs)

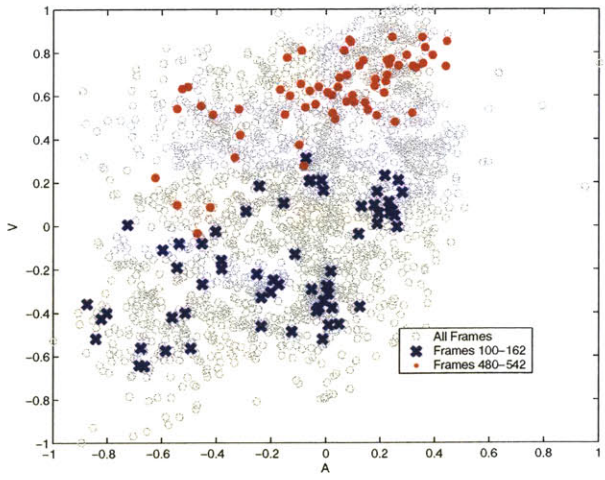
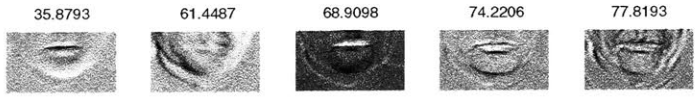


Figure 6-7: Samples drawn for person 1, 1 PCA coefficient (Pixel Intensity, MFCCs)

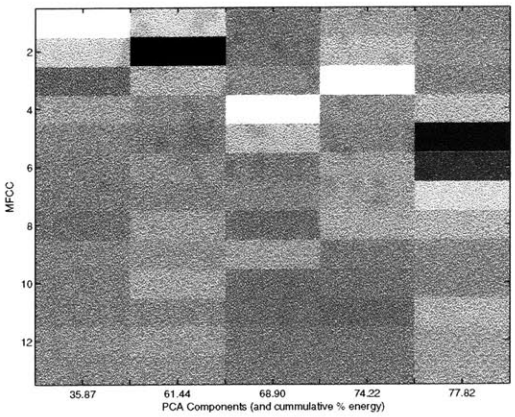
It is interesting to note that when we only use one principal component that the online techniques perform much better than the techniques that learn a model offline. This is clearly evident in Figure 6-6(b) which shows the online techniques doing best with one PCA coefficient over various window lengths. This conflicts with the results we obtained using synthetic i.i.d. data.

An explanation can be found in Figure 6-7. This figure shows shows that, unlike in our synthetic experiments, the i.i.d assumption clearly does not hold for this data when 1 principal component is used. The figure shows the training data using separate one dimensional features for the audio and video. The gray circles show all of the training samples, while the the dots (red,light) and x's (blue,dark) are highlighted samples from different windows of time. It is clear that these samples drawn from different times are not i.i.d samples drawn from the distribution described by the entire training set. This explains the better performance of the online techniques. While the offline techniques were evaluating likelihoods of the test data under distributions that were inconsistent with the data, the online techniques were still able to measure the statistical dependence between the audio and video.

In order to help explain why this data is not i.i.d we look at the the PCA components from a set of training data. Figure 6-8 shows the first 5 PCA components for



(a) Top 5 Video PCA



(b) Top 5 Audio PCA

Figure 6-8: PCA Components and the cumulative amount of energy they capture for Person 1

audio and video for this data and the cumulative amount of energy each component captures. The first, third and fourth video component seem to have to model jaw and lip motion. However, the second and fifth component seem to model translation, and scale of the face. This indicates that the video was not perfectly stabilized over all utterances. By just using the first video component there is no way to represent any shift or scale of the persons lips or mouth region. Depending on where the person's head is in the video frame it may produce different variations in the first PCA coefficient, thus producing audio-visual samples that depend on head location. This clearly shows the importance of having a good face tracker or choosing a representation that is less sensitive to tracking errors for learning models. It also shows an advantage of learning models online which only depend on the local distributions of the windows tested. However, we have seen that, for this situation, the advantage disappears when we use more principal components. By using the top 5 principal components the offline Gaussian technique could learn a model that incorporated information about the person's head position and scale. Even though the data is not i.i.d. this general

model was still able to give us a reasonable level of performance when trained on the individual being tested.

However, we were not able to learn a good generic model with any of the techniques tested when using pixel intensity and MFCCs as our input representations. When training and testing on separate individuals, none of the techniques achieved better than 35 % error (raw results shown in Appendix Table F.3). Figure 6-9 shows the performance of each technique as a function of input dimension with a fixed window length of one second when training and testing on separate people. We see that not only does each technique perform poorly but in some cases they do worse than random.

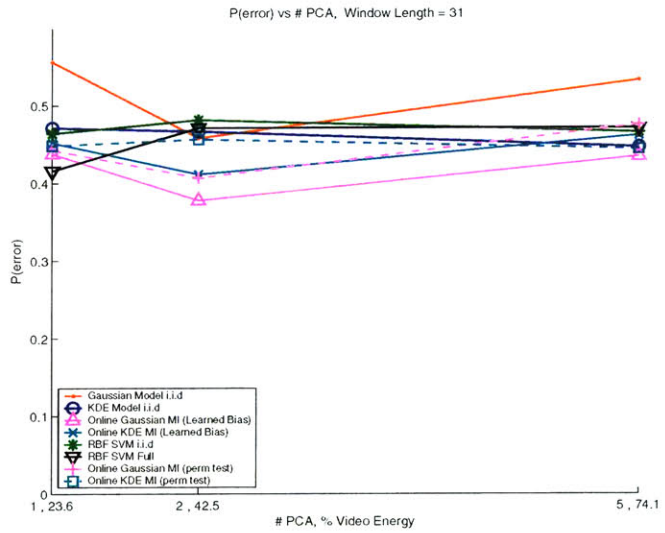


Figure 6-9: Performance summary plot. Testing and training on different people. (Pixel Intensity, MFCC)

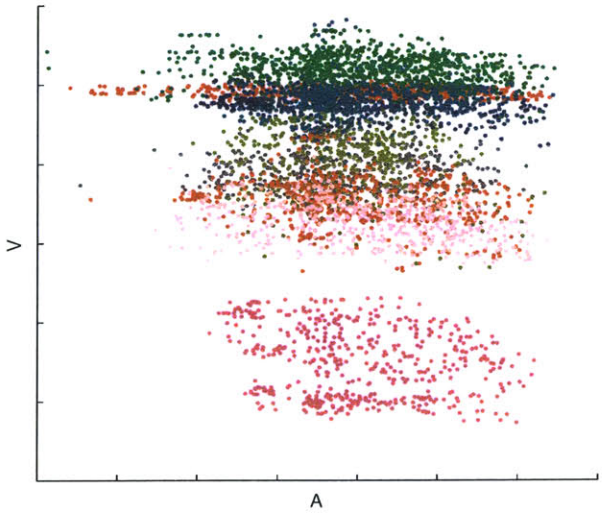


Figure 6-10: Samples drawn for 10 people. Each person is indicated by a different color/shade, 1 PCA coefficient (Pixel Intensity, MFCCs)

We plot the training samples taken from various people using one principal component in Figure 6-10, as a simple explanation for this poor performance. Again we clearly see that the data is not i.i.d. and depends heavily on time and the person being tested. Additionally, we see that it is difficult to identify any dependence between the audio and video within each cluster. Thus, it is difficult, if not impossible, to learn a generic model with PCA and the techniques we are testing.

Using Differential Features (Image and MFCC differences)

Next, we perform a similar set of tests using different audio and video representations. The audio and video representations are adjacent frame differences of intensity frames and MFCCs respectively. These differential features may have an advantage in that they can incorporate dynamics and remove some appearance information that may have caused non i.i.d samples in the previous tests. We test the same suite of techniques using the same parameters explored in the previous set of tests. Raw results are reported in Appendix Tables F.4, F.5, and F.6.

We summarize the best performance achieved by each technique for our person-

(a) Person-Specific (Person 2)

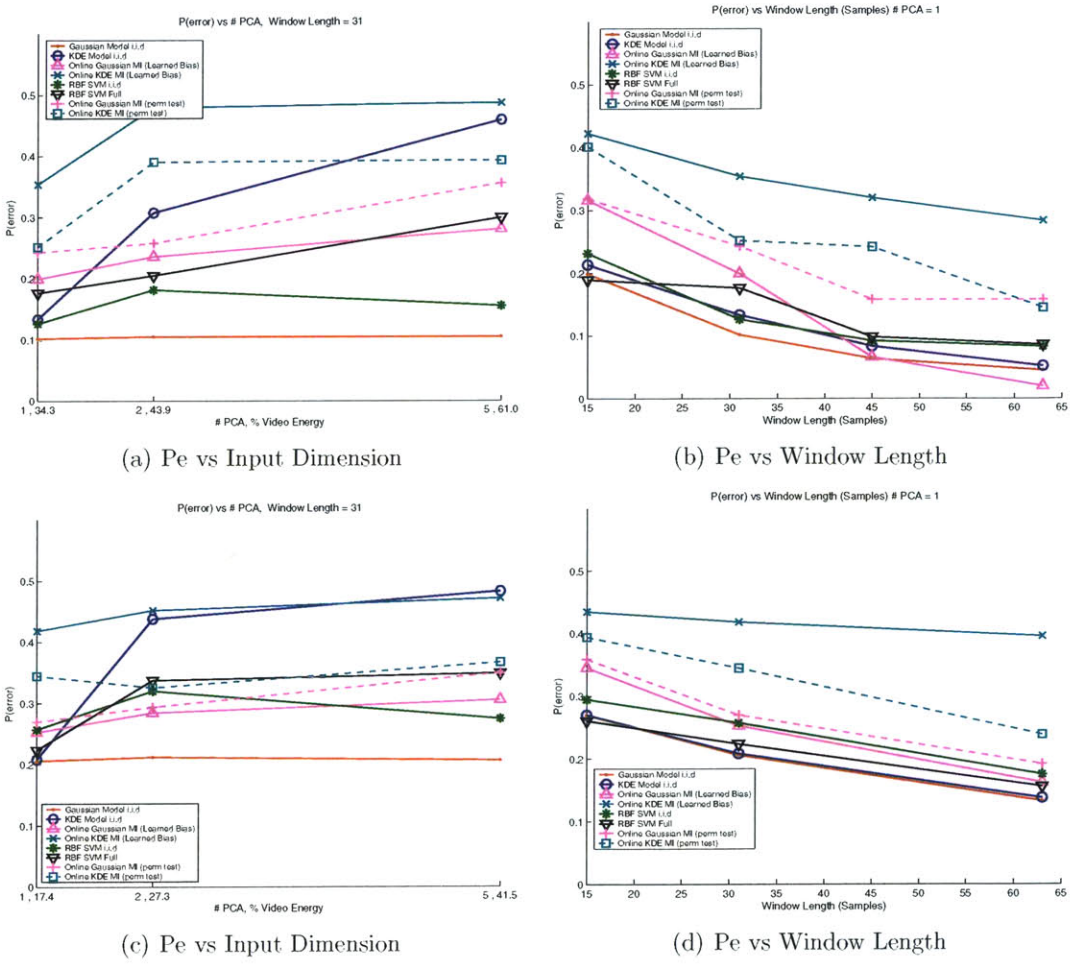
Technique	Best Avg P(error)	# PCA	Win Length
Gaussian Model i.i.d.	0.02	5	63
KDE Model i.i.d.	0.05	1	63
Gaussian MI (Learned Bias)	0.02	1	63
KDE MI (Learned Bias)	0.28	1	63
RBF SVM i.i.d.	0.04	5	63
RBF SVM Full	0.09	1	63
Gaussian MI (w/ Perms)	0.06	1	63
KDE MI (w/ Perms)	0.10	1	63

(b) Generic (10 People)

Technique	Best Avg P(error)	# PCA	Win Length
Gaussian Model i.i.d.	0.13	1	63
KDE Model i.i.d.	0.14	1	63
Gaussian MI (Learned Bias)	0.16	1	63
KDE MI (Learned Bias)	0.40	1	63
RBF SVM i.i.d.	0.17	1	63
RBF SVM Full	0.15	1	63
Gaussian MI (w/ Perms)	0.16	1	63
KDE MI (w/ Perms)	0.21	1	63

Table 6.3: Person-specific and generic results summary using image and MFCC differences.

specific and generic tests in Table 6.3. Again, we see that the offline Gaussian technique did very well at learning a person-specific model. In fact it only had 2% error for person 2. However, unlike in the previous tests, we see that a generic model can also be learned with 13 % probability of error. In addition all the offline techniques did reasonably well using only one principal component (see Figures 6-11(b) and 6-11(d)).



(a) Pe vs Input Dimension

(b) Pe vs Window Length

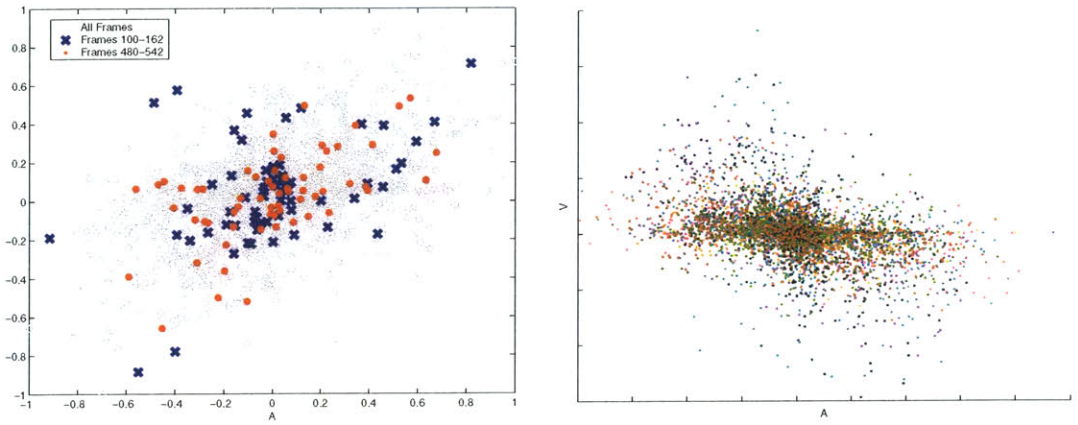
(c) Pe vs Input Dimension

(d) Pe vs Window Length

Figure 6-11: Performance Summary Plots (Image and MFCC differences). (a)(b) Results for Person 2 (c)(d) Results for 10 People

These results are very similar to those shown for the synthetic Gaussian data, indicating that the observations are close to i.i.d. Figure 6-12(a) confirms this by showing different windows of samples selected from training data. Not only does the

data look i.i.d, it also looks somewhat Gaussian. Furthermore, Figure 6-12(b) shows that this i.i.d. assumption seems to hold across multiple people giving us a good explanation for why we could learn a generic model with these features.



(a) Single Person

(b) 10 People

Figure 6-12: Training samples using 1 principal component. (Image and MFCC differences)

Some other interesting observations are that the online KDE MI does the worst and gets even worse as the dimensionality increased. Since this data seems i.i.d and Gaussian, their is no advantages to using the online KDE MI techniques. Figures 6-11(a) and 6-11(c) show all of the non-parametric techniques degrading as the dimensionality of the input increased. However, they also show that the SVM i.i.d is less sensitive to dimensionality than the KDE model in this case. While the KDE techniques attempt to learn a density the SVM only needs to learn a discriminating boundary and may be more robust.

This experiment has shown that we can in fact learn a generic model for audio-visual association, and that are specific choice of audio-visual features plays an important role. Although the best technique produced a 13 % error, we see from Figure 6-13 that by increasing the window length we can achieve even lower error rates.

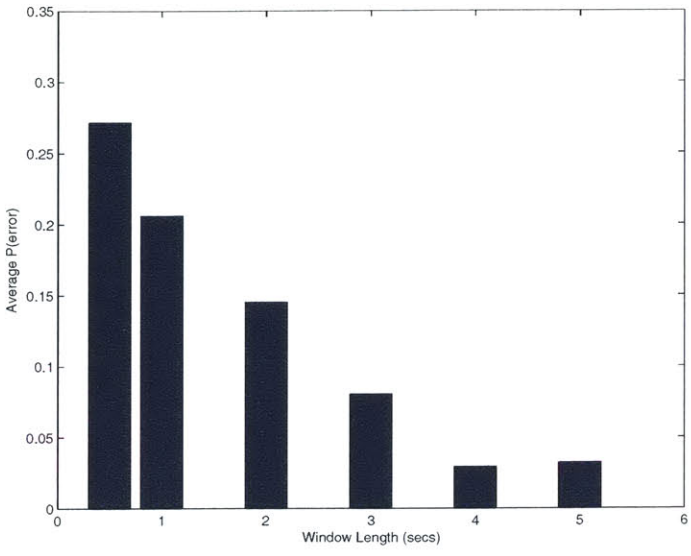


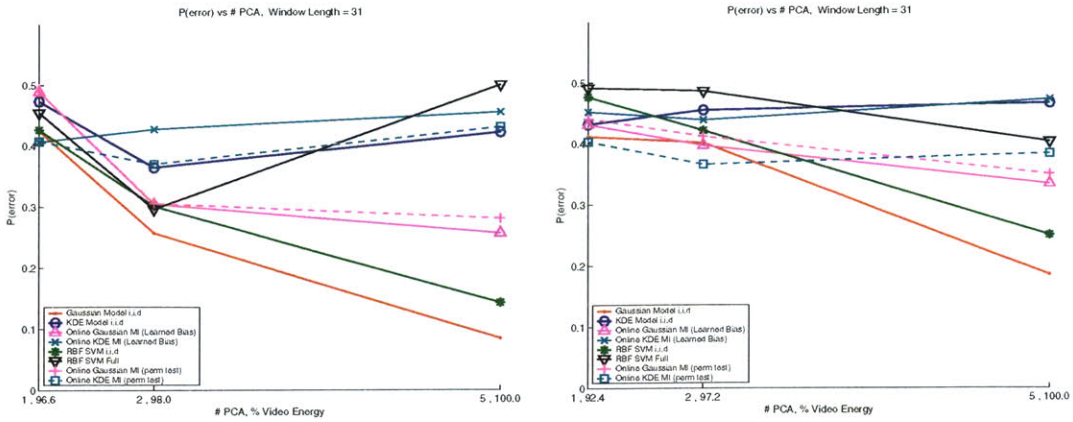
Figure 6-13: Probability of error versus window length using 5 PCA components and a Gaussian Model

Using spatially invariant features (Optical flow)

Another set of tests are carried out on a the same individuals using optical flow statistics for the video representation. These flow statistics throw out any spatial information and represent the overall amount of motion, the average x and y motion as well as the motion variance in each direction. Appendix Tables F.4, F.5, and F.6 show the raw results.

The results for these features were very similar to those when using the image differences. The Gaussian model technique performed the best for the person-specific as well as the multi-person tests. It had a 1 to 2 % error on the person-specific tests and a 10 % error on the 10 person dataset. However, to get any decent performance we needed to use at least 5 principal components. Figures 6-11(a) and 6-14(a) show how the probability of error decreases as the input dimension increases for the Gaussian and SVM i.i.d techniques. At the same time the performance of the other non-parametric and online techniques degrades.

Figure 6-15 shows the PCA components for the video representation from training data of a single person. We see the first PCA component picks out the variance of



(a) Pe vs Input Dimension (1 Person)

(b) Pe vs Input Dimension (10 People)

Figure 6-14: Performance Summary Plots (Flow and MFCC differences)

the horizontal optical flow. This should capture any non ridged horizontal motion. One would expect this to be a good indicator of a person opening or rounding their lips, but the results show that using this feature and the first PCA component of the MFCC differences is not particularly informative. This shows the danger of just using PCA to pick a lower dimensional representation. In this particular case the video representation was already reasonably compact and the highest variance components found with PCA may not be the most informative features.

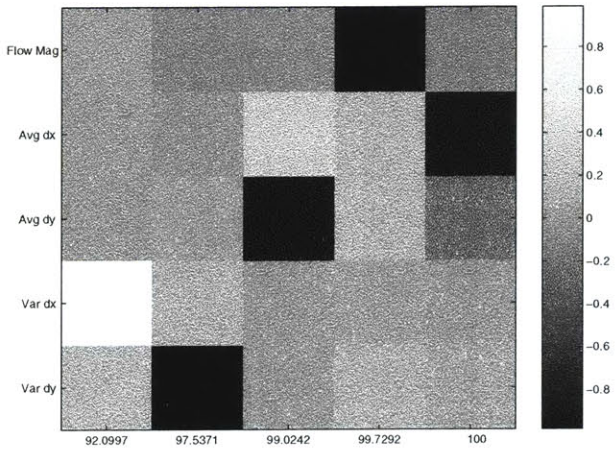


Figure 6-15: Top 5 principal components of the optical flow features

Other audio-visual features

We have shown above that the audio-visual features we choose impact our performance. Although we were able to achieve decent performance when training and testing on the same person with all of the features tested, only the differential features produced samples that were close to i.i.d. and could be used to measure association on individuals who were not seen in the training data. Overall the offline Gaussian model technique had the best performance. Here we test the rest of the audio-visual features discussed in Chapter 5 using this technique with 5 principal components and a two second window length. We only test the case in which we train and test on separate individuals (using 10 people and 5 fold cross validation). Results are shown in Table 6.1.6.

	MFCC	MFCC Diff	LPC	Sgram
Intensity	0.35	0.43	0.35	0.33
Difference	0.28	0.13	0.37	0.37
DCT	0.37	0.40	0.41	0.28
Flow	0.20	0.10	0.26	0.27
H Edge Diff	0.34	0.15	0.38	0.38
W Pyr	0.33	0.42	0.38	0.27

Table 6.4: Probability of error for different audio visual features. Technique = Gaussian Model i.i.d. , # PCA = 5, Window Length = 63 samples

We see that the differential features outperform the static features. However, it seems to make little difference which type of differential features is used. Using image differences, optical flow statistics or horizontal edge differences as the video feature with MFCC differences for audio results in a probability of error around 13 %. It is interesting to note that the optical flow features do the best given any fixed audio feature. This is may be due to the fact that the flow statistics throw out all information about appearance and only capture motion information. This may be a disadvantage when tracking errors cause non-associated motion in the video. We leave a more rigorous study of audio-visual features for future work. All that can be taken away from this simple experiment is that throwing away information about appearance and using features with some dynamics helps, particularly when making

a Gaussian i.i.d. assumption across different people.

6.1.7 Summary and Discussion

The goal of this set of experiments was to study the tradeoffs and performance of different modeling and testing techniques for detecting audio-visual association. Both offline and online learning techniques were compared. Most of the techniques used a generative approach while discriminative approaches were explored through the use of two different SVM techniques. The experiments were designed to explore how these techniques are affected by the dimensionality and type of the input features used as well as the number of samples used for testing. Synthetic data was tested in addition to real data taken from the AVTIMIT database.

The first set of synthetic data tested showed that when the data is i.i.d there was a performance advantage for the offline models that could learn a prior model from training data. The performance of these techniques agreed with the theory outlined in Chapter 3. It was also shown how the simple Gaussian techniques completely fail on the nonlinear data, while the non-parametric techniques were are capture more complex relationships. The SVM i.i.d technique and the KDE techniques had almost identical performance on the synthetic data. However for the linear Gaussian test data both seemed produce likelihood statistics that underestimated the true MI of the data. This may be due to the methods chosen for picking kernel sizes. We deliberately choose rule of thumb for KDE and cross validation for the SVM techniques to help foster quicker training times. However, other techniques may have led to better performance. We leave this question open for future work.

Prior to running experiments on real audio-visual data we outlined a set of questions we would like to have answered. Given the results of our experiments we do our best to answer these questions here:

What level of performance we can achieve in detecting audio-visual association for this data? We have shown that when we have training data for the person being tested we can learn a model that gives us excellent performance. Using differential features and a window length of 2 seconds we can achieve less than 5 % error in

detecting when the observed audio and video are associated. Detecting association without a model built specifically for the person tested was more difficult. However, we still showed that we could achieve less than 10 % error using only a 2 second window of data. Using longer window lengths only improves performance.

How well does our i.i.d. assumption hold? This question needs to be clarified. The audio-visual information that results from speech is inherently non-stationary. Depending on what is being said and how it is spoken we receive different audio-visual observations. The real question should be how well can we approximate this non-stationary processes using an i.i.d. assumption. Our results have shown that this depends on the audio and video representations that we choose. Using a simple representation such as pixel intensities and MFCCs produces samples that are clearly not i.i.d.. The distribution of these samples not only varies over time but also across different individuals' appearances. While we were able to learn a useful person-specific model with this data it was impossible to learn a generic model. However, we showed that by using differential features our observations looked more i.i.d. and that a generic model could be obtained. Therefore, while our i.i.d. assumption is not correct for audio-visual speech data we can find features that gives us reasonable performance.

How does the dimensionality of the observations affect the performance of each technique? In these experiments we controlled the dimensionality of our data using PCA. The more principal components we keep the more information we preserve. We showed that the offline Gaussian model technique, in general, benefited from keeping more principal components. This was particularly evident in the results for the optical flow features. To get any decent performance we needed to keep at least 5 principal components. However, most of the other techniques' performances degraded as the dimensionality increased. In particular all the online techniques had difficulty with a joint audio-visual space greater than 2 dimensions. This was to be expected in that even when we use a two second window we only have 63 samples to estimate a covariance or form a KDE. This is a limiting factor even when restricting ourselves to a Gaussian model. If we have a n dimensional joint audio-visual space we need

to learn $n^2/2$ parameters for a covariance matrix. With 63 samples and $n = 4$ we already have less than 8 samples per parameter. This situation becomes much worse as n increases.

Our results also showed that the offline non-parametric techniques were very sensitive to dimensionality. As shown in Figures 6-11(a), 6-11(c), 6-14(a), and 6-14(b) the gap between the offline Gaussian and KDE techniques grew as the input dimensionality increased. This gap seems to be larger than it should be considering that the offline KDE technique learns its model from a large amount of training data. It should have had enough samples to give a robust estimate of the joint density, even if it was truly Gaussian. Perhaps the best explanation for this is that our method for choosing a kernel size was suboptimal. Similarly the SVM i.i.d. technique may had difficult with higher dimensional data because we did not search enough parameters (C and the kernel size) when training. We leave exploration of these potential problems as future work.

Can we use simple Gaussian models or does this data require more complex non-parametric techniques? Overall we found that using a Gaussian model gave us the best performance. This may be because we needed more than a two dimensional joint audio-visual feature space to achieve good performance in most our tests and that our non-parametric techniques failed for the reasons discussed above. However, we found features that looked i.i.d. and somewhat Gaussian and thus a Gaussian model was the correct choice.

Can we learn a person-specific model for audio-visual association? Is it possible to learn a generic model for audio-visual association? Our results showed that learning a model for a particular person gives us better performance than using an online technique or training on different individuals. We also showed that it was possible to learn a generic model for audio-visual association if our features were close to i.i.d., i.e. each person has similar statistics that do not vary are large amount over time.

Does our specific choice of features affect performance? As discussed in the answers to the above questions, the choice of features greatly affects performance. We showed that using differential features that remove information about appearance

produced samples that were more i.i.d. and gave us lower error rates.

6.2 Experiment 2 : Techniques for learning Informative Subspaces Comparison

6.2.1 Purpose

In the previous set of experiments we compared different techniques for modeling and detecting audio-visual association in human speech. We showed how the performance of these techniques varied with the dimensionality of our observations. PCA was used a standard approach to control dimensionality. However, in Chapter 4 we discussed alternatives to PCA for learning informative low dimensional subspaces in which to represent our data. In this set of experiments we compare these alternative subspace learning techniques.

6.2.2 Variables

We compare the main techniques discussed in Chapter 4. Each subspace learning technique has its own set of assumptions. Thus, for each technique we restrict ourself to a particular set of model learning and testing techniques that are consistent with these assumptions. In addition, we do not consider purely online techniques in these tests. By assuming we can learn an informative subspace from training data we are also implying that some model for our data can be learned. Thus, we ignore the online Gaussian and KDE techniques that use permutations rather than a learned bias. The subspace learning techniques and their associated modeling techniques are shown in Table 6.5.

Subspace Learning Method	Abbreviation	Modeling and Testing Techniques
Joint PCA	JPCA	Gaussian Model i.i.d , Gaussian MI (Learned Bias)
Canonical Correlation Analysis	CCA	Gaussian Model i.i.d , Gaussian MI (Learned Bias)
Max MI	MMI	KDE Model i.i.d, KDE MI (Learned Bias)
Max Discriminant	MD	RBF SVM i.i.d, RBF SVM Full

Table 6.5: Informative Subspace Learning Techniques and Associated Model Techniques. Details about the subspace techniques can be found in Chapter 4.

For the majority of our experiments we restrict ourself to learning a two dimensional subspace, separate one dimensional projections of the audio and video. For the MMI and MD techniques we pick a reasonable set of optimization parameters. We run these optimizations with four different random initializations of the projection coefficients and keep the results with the highest resulting objective criteria. To help improve the speed of these techniques we only use a stochastically sampled set of 800 data points during each iteration. A new set of samples is chosen every 6 iterations. We use a variable step size and the optimization is run for 100 iterations.

6.2.3 Results : Synthetic Data

We begin with testing the informative subspace learning techniques on synthetic data for which we can calculate a "optimal" subspace. The first set of data is generated using the following linear model:

$$\begin{aligned} A &= \mathbf{M}_a\phi + \mathbf{n}_a \\ V &= \mathbf{M}_v\phi + \mathbf{n}_v \end{aligned} \tag{6.3}$$

with

$$\begin{aligned} \mathbf{M}_a &= [1\ .1]^T \\ \mathbf{M}_v &= [1\ 1\ 1\ 1\ .1]^T \end{aligned} \tag{6.4}$$

A low dimensional Gaussian random variable ϕ is lifted into a higher dimension using $\mathbf{M}_a$ and $\mathbf{M}_v$. Noise is then added in this higher dimension with independent random variables $\mathbf{n}_a$ and $\mathbf{n}_v$. For this data ϕ is drawn from $N(0,1)$, $\mathbf{n}_a$ from $N(0, \text{diag}([.01\ .5]))$ and $\mathbf{n}_v$ from $N(0, \text{diag}([.5\ .75\ 1\ .5\ .5]))$. A second set of data is generated with an added nonlinearity such that

$$\begin{aligned} \mathbf{a} &= \mathbf{M}_a\phi + \mathbf{n}_a \\ \mathbf{v} &= \mathbf{M}_v \sin(2\pi f\phi) + \mathbf{n}_v = \mathbf{M}_v\phi_{\text{sin}} + \mathbf{n}_v \end{aligned} \tag{6.5}$$

using $f = .5$.

Each synthetic data set produces an audio and video signal that share a common underlying low dimensional variable. Information about this variable is distributed amongst the components of the higher dimensions. Since all the additive noise is Gaussian and $\mathbf{M}_a$ and $\mathbf{M}_v$ are fixed, it can be shown that a linear projection will produce a sufficient statistic for ϕ. For the linear Gaussian case:

$$\begin{aligned} \mathbf{h}_a &= C_{n_a}^{-1}\mathbf{M}_a \\ \hat{\phi}(\mathbf{a}) &= \mathbf{h}_a^T\mathbf{a} = \mathbf{M}_a^T C_{n_a}^{-1}\mathbf{a} \end{aligned} \tag{6.6}$$

and

$$\begin{aligned} \mathbf{h}_v &= C_{n_v}^{-1}\mathbf{M}_v \\ \hat{\phi}(\mathbf{v}) &= \mathbf{h}_v^T\mathbf{v} = \mathbf{M}_v^T C_{n_v}^{-1}\mathbf{v} \end{aligned} \tag{6.7}$$

The same projections, $\mathbf{h}_a$ and $\mathbf{h}_v$, are optimal for the nonlinear case. However, rather than $\mathbf{h}_v^T\mathbf{v}$ producing a sufficient statistic for ϕ it gives sufficient statistic for ϕ_{sin}. That is $\mathbf{h}_v^T\mathbf{v} = \hat{\phi}_{\text{sin}}(\mathbf{v})$.

For both the linear and nonlinear models we generate 2000 samples. We then perform 4 fold cross validation in which we train on 1500 of those samples and then test using the remaining 500. The testing phase takes those 500 dependent samples and generates another 500 independent samples through random permutations of $\mathbf{a}$. We then perform an association hypothesis test using the model and subspace learned in the training phase. A short window length of 5 samples is used for this simple synthetic data. The training phase consistent of two parts. First we use one of the methods in Table 6.5 to learn an $\mathbf{h}_a$ and $\mathbf{h}_v$. Second we learn a model for projections of the training data in the subspace defined by $\mathbf{h}_a$ and $\mathbf{h}_v$.

We begin by using the full data for training and testing. This is compared to training and testing using the optimal projections described above. These results were compared to the results from using all of the methods described in Table 6.5 and their corresponding modeling techniques. Table 6.6 shows results for the linear data.

		P(error)					
		Gaussian Model i.i.d	KDE Model i.i.d	Gaussian MI	KDE MI	RBF SVM i.i.d	RBF SVM Full
Method	None	.0217 ± .0066	.0503 ± .0079	.5000 ± .0000	.4611 ± .0164	.0343 ± .0032	.0685 ± .0085
	Optimal	.0242 ± .0083	.0301 ± .0109	.0885 ± .0147	.1515 ± .0186	.0186 ± .0093	.0313 ± .0030
	JPCA	.0348 ± .0070	-	.1076 ± .0057	-	-	-
	CCA	.0207 ± .0029	-	.0882 ± .0113	-	-	-
	MMI	-	.0364 ± .0145	-	.1558 ± .0155	-	-
	MD	-	-	-	-	.0320 ± .0055	.0234 ± .0054

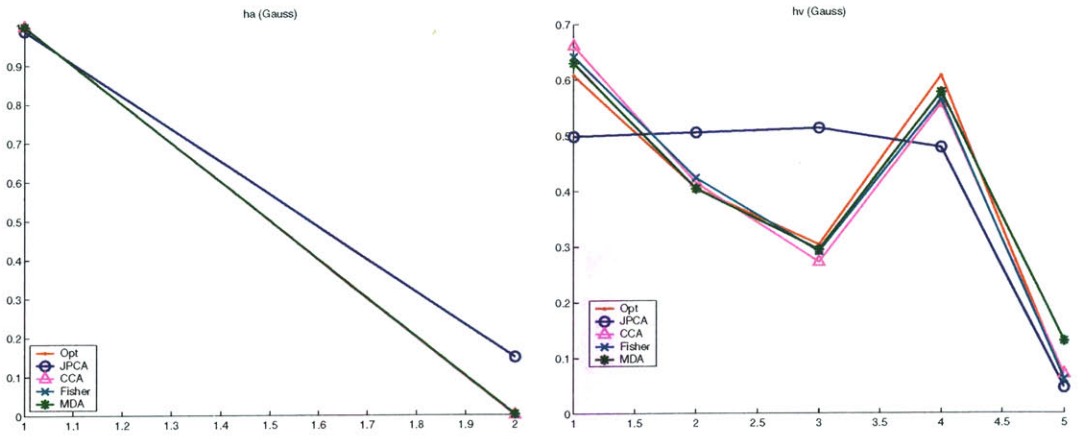
Table 6.6: Results for linear data

When training and testing on the full data we see that, unsurprisingly, using a Gaussian model works the best (2% error). Table 6.6 also shows that the SVM and offline KDE techniques also work reasonably well. However, we clearly see that the online techniques do very poorly with the full dimensional data and only 5 samples to estimate it's distribution.

Using the optimal projections we see a significant improvement in the online MI techniques. We also observe that using the optimal projection and a Gaussian model has the same performance as using the full data. Similarly, so does the i.i.d SVM technique, verifying that our optimal projections produce sufficient statistics. Thus, if we can learn the optimal projection we will have the benefit of lower dimensional distributions for the online techniques to estimate and possibly be able match performance of using the full data.

Figure 6-16 shows the projections learned by the subspace learning techniques and the optimal. We normalize the projections to have a unit norm since measuring MI and learning models for the data is invariant to scale. Clearly each of the methods found projections close to the optimal in this simple linear case. The main exception is JPCA. This is an expected result. As discussed in Chapter 4, the objective of JPCA has little to do with the performance of the association hypothesis test.

Table 6.6 and Figure 6-17 show that the methods other than JPCA have similar performance to the optimal projection. For this simple linear case we know that using CCA and a Gaussian model is the correct thing to do. Although CCA performed the best we also see that the other more general and computationally expensive MMI and MD techniques had similar performance. Next, we explore the performance of these techniques on nonlinear data where we expect to see CCA fail and the nonparametric



(a) $\mathbf{h}_a$ (b) $\mathbf{h}_v$

Figure 6-16: Learned Projections (Linear Data)

		P(error)					
		Gaussian Model i.i.d	KDE Model i.i.d	Gaussian MI	KDE MI	RBF SVM i.i.d	RBF SVM Full
IFE Method	None	.4694 ± .0179	.3290 ± .0251	.4871 ± .0144	.4833 ± .0141	.3967 ± .0438	.4844 ± .0054
	Optimal	.4515 ± .0467	.1162 ± .0086	.4786 ± .0119	.4336 ± .0223	.1116 ± .0161	.4609 ± .0340
	JPCA	.4614 ± .0521	-	.4682 ± .0250	-	-	-
	CCA	.4556 ± .0166	-	.4882 ± .0106	-	-	-
	Max MI	-	.1568 ± .1357	-	.4396 ± .0172	-	-
	MD	-	-	-	-	.1568 ± .1401	.3374 ± .2253

Table 6.7: Results for nonlinear data

techniques to show their worth.

The results for the nonlinear data are shown in Table 6.7. Clearly, we see that using a Gaussian model is wrong for this data. There is a significant improvement for the KDE and SVM i.i.d models when using the optimal projections. The lack of improvement in the KDE MI technique is most likely due the fact we are only using 5 samples for test. Figure 6-18 shows the projections learned for the nonlinear data. We see that JPCA and CCA did poorly while MMI and MD found projections closer to the optimal. However, there is more variation in their performance than CCA and JPCA. Figure 6-19 also shows this variation in the performance. While MMI and MD have the best performance they also have the largest error bars.

This set of experiments on synthetic data has reenforced much of the theory discussed in Chapter 4. It was clear that using JPCA was suboptimal for learning an informative subspace well suited to measuring association. We saw that, with

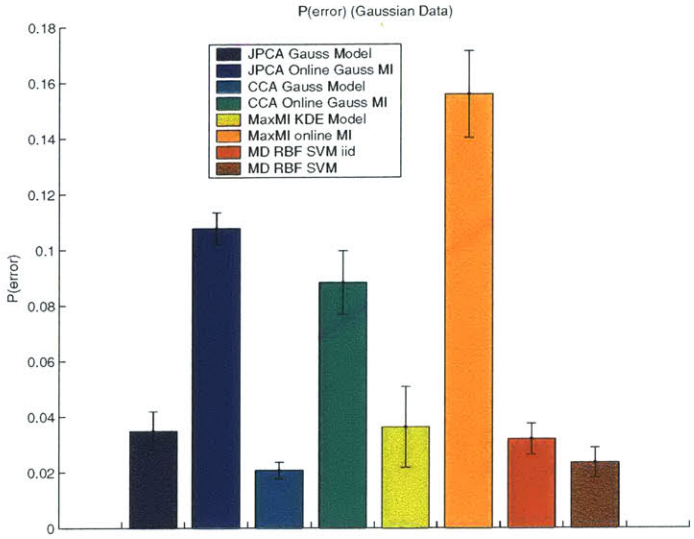
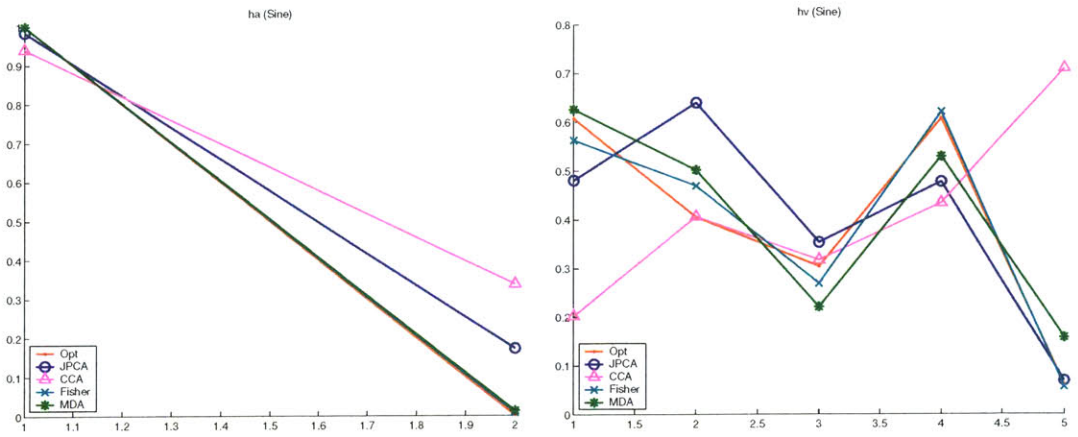


Figure 6-17: Linear Data: Pe vs Technique

the simple linear data, using CCA and a Gaussian model was optimal. It was also shown that if we are able to find a projections that produce sufficient statistics for the underlying dependency between the audio and video we can achieve the same performance as if we were using the full data. The advantage of finding the subspace is that we may not need as many samples to learn a good model and our computational cost is reduced.

The experiments on nonlinear data presented a simple example where CCA fails and the nonparametric approaches were necessary. We saw similar performance from the MMI and MD techniques in both sets of experiments. While they did a good job at finding the optimal projections they also seemed to have to most variation across cross-validation folds. Unfortunately, these techniques were also an order of magnitude slower in their training phase. This is mainly a result of us trying multiple initializations. However, in the case of the MD technique each iteration took a considerable amount of time even when only 600 samples were used per iteration. Experiments on high dimensional data were impractical with the MD technique. Thus, in this thesis we only show results on synthetic data and leave exploration of improvement to this algorithm for future work.



(a) $\mathbf{h}_a$ (b) $\mathbf{h}_v$

Figure 6-18: Learned Projections (Nonlinear)

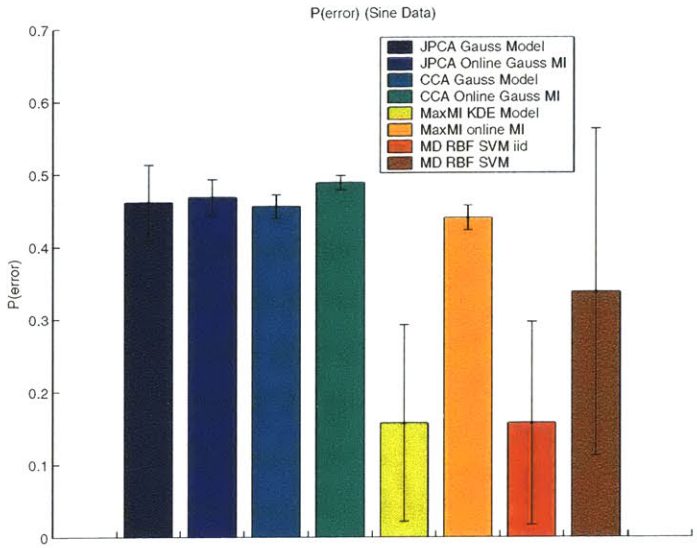


Figure 6-19: Nonlinear Data: Pe vs Technique

Modeling Technique	Average P(error)	Subspace Learned w/	Number of Bases per Modality	Window Length
Gaussian Model i.i.d.	0.10	PCA	5	63
Gaussian Model i.i.d.	0.36	PCA	1	63
Gaussian Model i.i.d.	0.12	CCA	1	63
KDE Model i.i.id.	0.48	PCA	5	63
KDE Model i.i.id.	0.44	PCA	1	63
KDE Model i.i.id.	0.08	MMI	1	63

Table 6.8: Results comparison for PCA, CCA and MMI using flow statistics and MFCC differences (10 People)

6.2.4 Results: Data from AVTMIT

Now that we have reenforced some of our theory for subspace learning on synthetic data, we are ready to test these techniques on real data. We concentrate on learning a generic model for association. Our training and testing sets are mutually exclusive. No individuals or sentences are shared between them. In the previous experimental section we found that differential features performed the best. Specifically, we were able to learn a good generic model for measuring association using image differences and our flow statistics for the video representation.

CCA and MMI on Flow Features

We begin our experiments in this section using flow statistics as the video representation and MFCC differences for the audio. These representations are already relatively low dimensional, producing an 18 dimensional joint AV space. Previously we have shown that using the top 5 principle components for both modalities and a Gaussian model produced 90 % accuracy in detecting audio-visual association. Here, rather than using PCA, we test the use of both CCA and MMI for learning an informative subspace. We use the non-regularized versions of these methods since we have many more training examples than number of input dimensions. Table 6.8 presents the results for this experiment in addition to the previous results obtained using PCA.

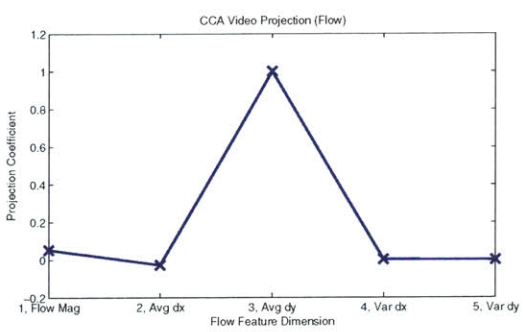
When using a single CCA projection for each modality and a Gaussian model we obtain 12% error. This is a huge improvement over using a single principal component learned by PCA, but similar in performance to using 5 principal components.

Clearly CCA better preserves the relevant information than simply using PCA. A more dramatic improvement can be seen when using MMI for the subspace learning and using a KDE model. We see in Table 6.8 that the simple 2 dimensional subspace learned by MMI gave us 8% error. With PCA we never did better than 40% error. It is also interesting that using MMI and a non-parametric model did best overall, when previously the Gaussian techniques were doing the best.

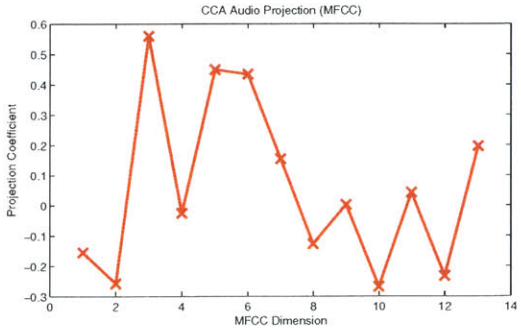
To help explain these results we can study the bases learned by both CCA and MMI. Figure 6-20 and 6-21 show the learned bases and projected data for CCA and MMI respectively. The first thing we notice is that the bases learned for the video strongly emphasizes the average flow in the y direction. This is true for both CCA and MMI. This is a nice intuitive result. We expect the vertical flow to be related to lip motion. If we look back at the bases learned using PCA in Figure 6-15 we see that this feature, (Average dy), was mainly modeled by the 3rd basis. The bases that described the variance in the x and y directions had more variation and thus PCA picked bases to described those features first. This is why we required so many principal components to obtain decent performance.

The audio projections found CCA and MMI are difficult to compare. They are both different and do not have a simple intuitive explanation like the bases found for the video. However, if we look at the scatter plots in Figure 6-20(c) and 6-21(c) we see that projecting our data onto these bases learned by CCA and MMI reveals some dependency structure. We also see that the bases produced similar projections for both training and testing data. That is, it doesn't look like we have an overfitting problem.

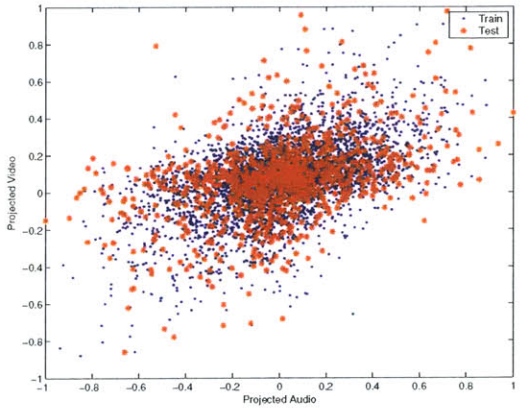
It is difficult to judge the amount of dependency information contained in the data from these simple scatter plots. The MMI projections look a little less dependent than those learned for CCA and neither technique seemed to produce particularly Gaussian looking data. However, we found that MMI combined with a non-parametric did the best. Since we are only dealing with 2 dimensional projections here, it may be useful to look at the decision boundaries described by the models we learned for these projections. That is, plot $L(\hat{\mathbf{a}}, \hat{\mathbf{v}}) = \eta$ for a single observation. Figure 6-22 shows



(a) CCA Video Projection

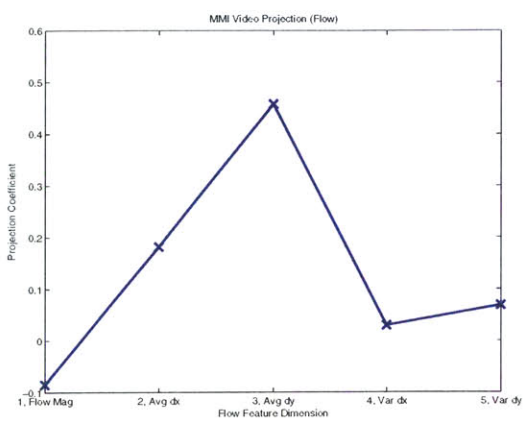


(b) CCA Audio Projection

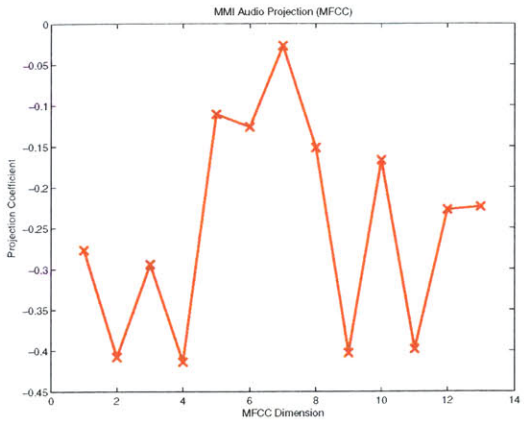


(c) CCA Projection Scatter Plot

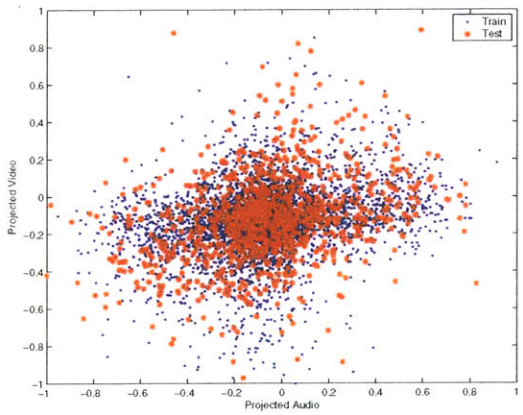
Figure 6-20: CCA Learned Projections / Bases (Flow and MFCC differences)



(a) MMI Video Projection



(b) MMI Audio Projection



(c) MMI Projection Scatter Plot

Figure 6-21: MMI Learned Projections / Bases (Flow and MFCC differences)

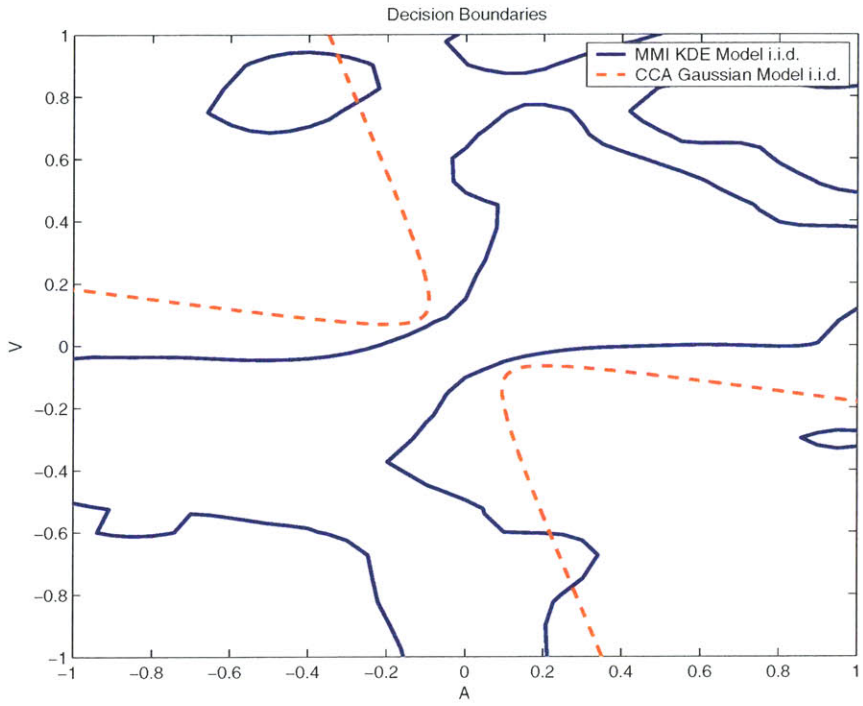


Figure 6-22: Decision Boundaries learned by MMI and CCA for Flow and MFCC Diff features

these decision boundaries on aligned axes.

We see that both boundaries have somewhat similar hyperboloid shapes (with some artifacts from finite sampling near the corners for the KDE model). However, the boundary learned using MMI and a KDE model clearly exploits some non-linear / non-Gaussian nature of the data. This gives us some explanation for its improved performance of 8% error.

CCA and MMI on Image Differences

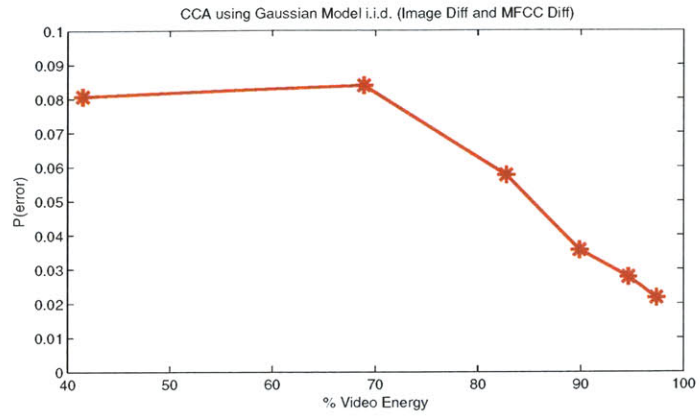
Next we apply CCA and MMI to a higher dimensional video representation. Previously, using image and MFCC differences, we obtained 13% error in detecting audio-visual association. This was using a single PCA bases for each modality. Furthermore, we found that adding more PCA bases did little to improve performance and actually hurt the non-parametric modeling techniques. Here, as we did previously with the

flow representation, we see if we can improve performance by replacing PCA with one of our informative subspace learning techniques.

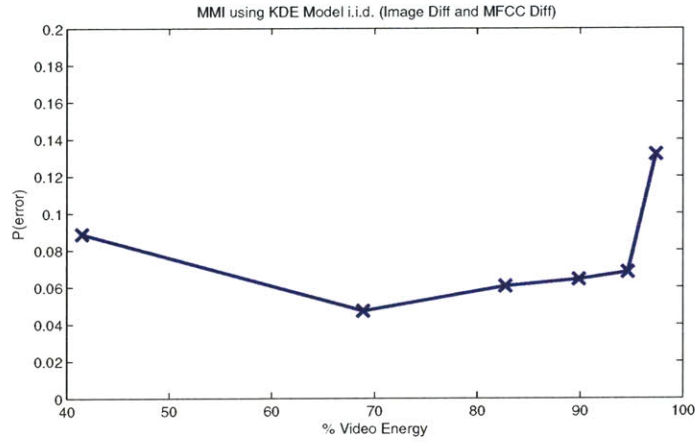
The difference here is that our video representation is much higher dimensionality than our simple flow statistics. Each video frame has 5500 features (55 x 100 image). The simplest of our subspace learning method, CCA, requires an estimate of the covariance of these features. For this representation, that means we estimate more than 15 million parameters for the covariance matrix. Thus, it makes sense to impose some regularization. Here we simply use PCA to first reduce the dimensionality to reasonable size and then apply our subspace learning techniques in this lower dimensional space. Since both PCA and our subspace learning methods produce linear bases, we can combine them into a single set of linear bases to project our original data onto. For our experiments we regularize by first reducing dimensionality of our video with 512, 256, 128, 64, 24, or 5 PCA bases. Using this number of bases, on average, preserved 97, 95, 90, 82, 69, and 41% of the energy in the video respectively. We assume that by preserving over 90% of the variation in the video data we are also preserving the relevant information for detecting audio-visual association.

Again, we use 10 people for this experiment, with 5 fold cross-validation (training on 8 and testing on 2). We restrict our informative subspace learning techniques to learning a single basis for each modality. Figure 6-23(a) shows performance using CCA and a Gaussian model as a function of energy preserved by our PCA regularization. Surprisingly, we see that regularization only hurts us. The technique did better as more principal components were kept. The more information we gave CCA the better it did. There was no sign of overfitting to the training data. The average probability of error across folds was only 2% when using 512 principal components. This a significant improvement over the 13% error previously obtained using PCA alone.

However, we see in Figure 6-23(b), that the when using MMI and a non-parametric model the PCA regularization did improve performance. This model's performance was the same or better than CCA and the Gaussian model using 5, 24 or 64 principal components for regularization, but rapidly became worse using 128 or more. It



(a) CCA



(b) MMI

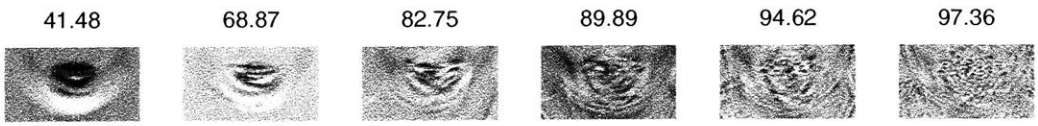
Figure 6-23: P(error) vs % Energy kept during PCA Regularization for CCA and MMI on Image and MFCC Differences

performed best when using 24 bases for regularization, with only 5% error.

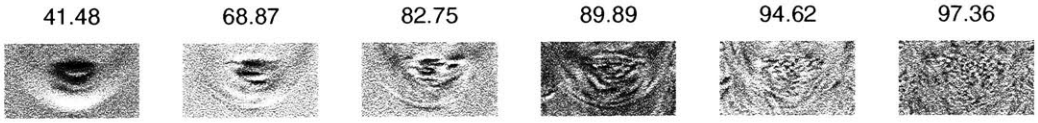
This behavior can possible be explained by our optimization procedure for MMI. Unlike CCA, which finds a global maximum for it's objective criteria, MMI's optimization procedure may get stuck in local maximum. We use a simple gradient descent starting at random initializations to find a solution. The higher the dimensionality of the data input to MMI, the larger our search space becomes for finding informative projections/bases. Furthermore, in order to improve speed, we employ a subsampling technique which only uses a subset of the training data in each iteration. If our data dimensionality is too high, we may overfit to a particular subsampling of the data. We try to combat these problems by trying at multiple random initializations, and uses over 800 subsamples per iteration. Our results show that this was not enough to avoid local maxima when using over 64 principal components for regularization. While this highlights some of the main disadvantages of MMI, it also shows the importance of regularization with such a technique. It is also important to note that MMI still did an excellent job of finding an informative subspace with regularization, resulting in 5% error. Again, this is a significant improvement over just using PCA.

Previously, when using CCA and MMI on our optical flow features, we found that the learned video bases mainly modeled vertical motion. This was a nice intuitive result. Let's see if the projections/bases learned for our image differences emphasize any particular portions of the video. Figure 6-24 shows the projections learned by CCA and MMI as function of energy preserved during regularization. We notice that the more regularization we use (less energy preserved) the more our bases seem to emphasize the jaw and lip motion. Less regularization seems to produce more noisy looking bases. However, we found that CCA did the best with those "noisy" looking bases. What is even more surprising is that the CCA and MMI projections look almost identical. This result is odd considering that we concluded that MMI must have found local maxima when we used less regularization. These local maxima look very similar to the optimal bases CCA learned.

We do not claim to have a solid explanation for this results. However, it turns out, if we look at any random projection of the top 512 or 256 principal components



(a) CCA Video Projection



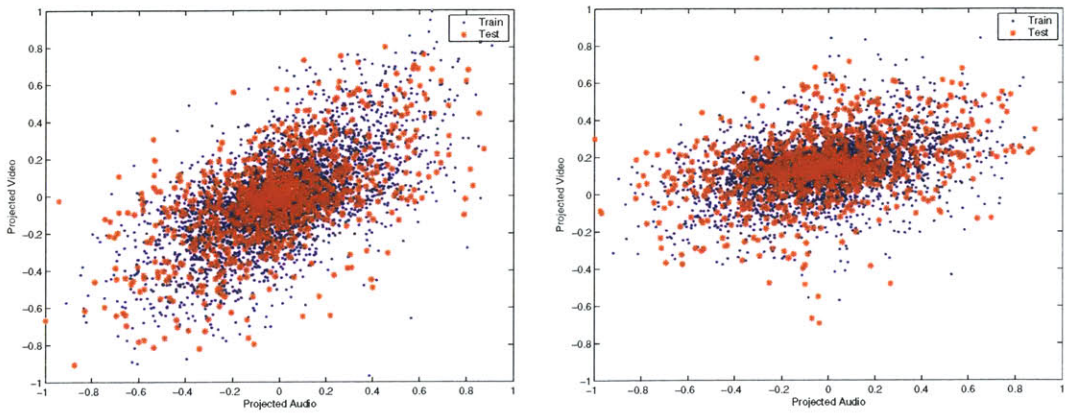
(b) MMI Video Projection

Figure 6-24: CCA and MMI Learned Projections / Bases (Image and MFCC differences). The number above each projection is the % of video energy preserved for regularization.

of our training data we get bases similar to the last two CCA and MMI bases shown in Figure 6-24. This brings into question whether or not CCA and MMI are just learning "random" projections. We see that this is not the case, specifically for CCA, in Figure 6-25. Figure 6-25(a) shows the training and testing data projected onto the subspace learned using 512 principal components for regularization. Clearly we see some dependency between the audio and video. Additionally we see that the training and testing data have similar distributions. Thus, it seems we do not overfit.

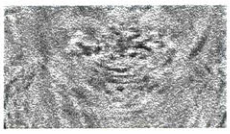
Figure 6-25(b) shows the projection our our data onto the subspace learned by MMI under the same conditions (512 principal components for regularization). Visually there seems to be less dependency between the audio and video in this case. Thus, it makes sense that we did worst with this MMI projection they we did with the one learned by CCA Although the testing and training data seem to have similar distributions in this figure, it may be that we overfit to last subsampling of data used by MMI, which is mixed into the rest of the training data, or it may be that this subspace provided a local maxima.

Lastly we show what our subspace would look like if we used a random combination of the principal components rather than CCA or MMI in Figures 6-25(c) and 6-25(d). Clearly we see that while our random bases looks visual similar to those learned by MMI and CCA, it provides us with much less informative subspace. Therefore, we

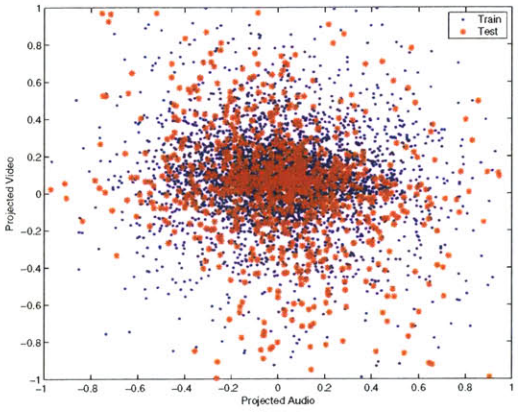


(a) CCA Subspace

(b) MMI Subspace



(c) Random Basis



(d) Random Subspace

Figure 6-25: Data projected into subspaces learned by (a) CCA (b) MMI (d) Random for Image and MFCC Differences. (c) Shows a random linear combination of the top 512 principal components of the training data.

cannot always extract useful information from the projections learned by CCA or MMI. The information they exploit to best capture the dependency between the audio and video may not be visually obvious.

In addition to using PCA for regularization, we also ran a series of experiments using L2-regularization. We tried setting R_a and R_v to be the identity in Equations 4.27, 4.26 and swept λ_a and λ_v from $1e-10$ to $1e-2$. However, we could not identify any trends. We could never match the performance we achieved by simply using PCA for regularization. In Appendix D we show that the parameters for L2 regularization

can be interpreted as defining priors on the bases we learn or the type of noise we have in our training data. As discussed above, we found that the informative bases learned by CCA and MMI do not always have an intuitive form. The bases that looked smooth and seemed to model jaw motion did worse than the bases that had a more "noisy" appearance. Thus we do not have a good intuition about what how we should choose our regularization parameters. We leave a more detailed study of L2 regularization as future work.

Combining CCA with a non-parametric model

We have shown that using CCA and MMI to learn an informative subspace improves performance over simply using PCA. Both techniques gave very similar performance, and sometimes learned similar bases. We saw this when using CCA and MMI on our flow features. However, while both techniques produced similar bases, we saw that using a KDE Model had better performance than using a Gaussian one. This led us to try mixing CCA for subspace learning with a KDE for modeling our data. CCA provides us with a simple subspace learning algorithm that avoids the local maximum problems that can occur with MMI. It finds a subspace that has maximum mutual information if our data is Gaussian. If our data is not Gaussian, the subspace learned by CCA is only guaranteed to have first and second order statistics such that the correlation coefficient is maximized. However, this in no way implies that the projected data will be Gaussian or have maximum mutual information. Thus, it may be a good idea to model this projected data with a non-parametric model. In addition, CCA has the added bonus of always finding a set of orthogonal bases. Therefore, we can look at how keeping multiple bases to form a higher dimensional subspace affects performance.

We ran a set of experiments on the same dataset of 10 people exploring these ideas. We looked at keeping 1, 2 or 3 bases (1,4 or 6 dimensional subspace) learned by CCA and compared performance using a Gaussian or non-parametric model for the projected data. We begin by using flow and MFCC differences for our representation. Results are summarized in Table 6.9.

Modeling Technique	Average P(error)	Subspace Learned w/	Number of Bases per Modality	Window Length
Gaussian Model i.i.d.	0.104	CCA	1	63
Gaussian Model i.i.d.	0.125	CCA	2	63
Gaussian Model i.i.d.	0.098	CCA	3	63
KDE Model i.i.id.	0.064	CCA	1	63
KDE Model i.i.id.	0.028	CCA	2	63
KDE Model i.i.id.	0.005	CCA	3	63

Table 6.9: Results comparison for PCA, CCA and MMI using flow statistics and MFCC differences (10 People)

Modeling Technique	Average P(error)	Subspace Learned w/	Number of Bases per Modality	Window Length
Gaussian Model i.i.d.	0.017	CCA	1	63
Gaussian Model i.i.d.	0.007	CCA	2	63
Gaussian Model i.i.d.	0.007	CCA	3	63
KDE Model i.i.id.	0.013	CCA	1	63
KDE Model i.i.id.	0.005	CCA	2	63
KDE Model i.i.id.	0.004	CCA	3	63

Table 6.10: Results comparison for PCA, CCA and MMI using Image and MFCC differences (10 People)

The first thing we notice is that using CCA with a Gaussian model we see no improvement in keeping multiple bases. The performance stays close to 10% error, with a strange bump up to 13% when using 2 bases. The more interesting results is that combining CCA with a non-parametric model has improved performance. We get 6% error using a single CCA bases and improve to under 1% error as we use up to 3 bases. These are our best results so far. Clearly, CCA has provided an informative subspace for this problem, but we needed a non-parametric model to properly describe the data in this space. Using 3 bases for audio and video learned by CCA gives us a 6 dimensional joint space. We show the nonlinear nature of the data in this space by plotting the training samples used by the KDE model learned in Figure 6-26. We see some linear relationships between the first audio and video dimensions (1 and 4). However, there are clearly many nonlinear relationships, particularly between the video dimensions that would be poorly modeled by a Gaussian.

We ran the same set of experiments using image differences for our video repre-

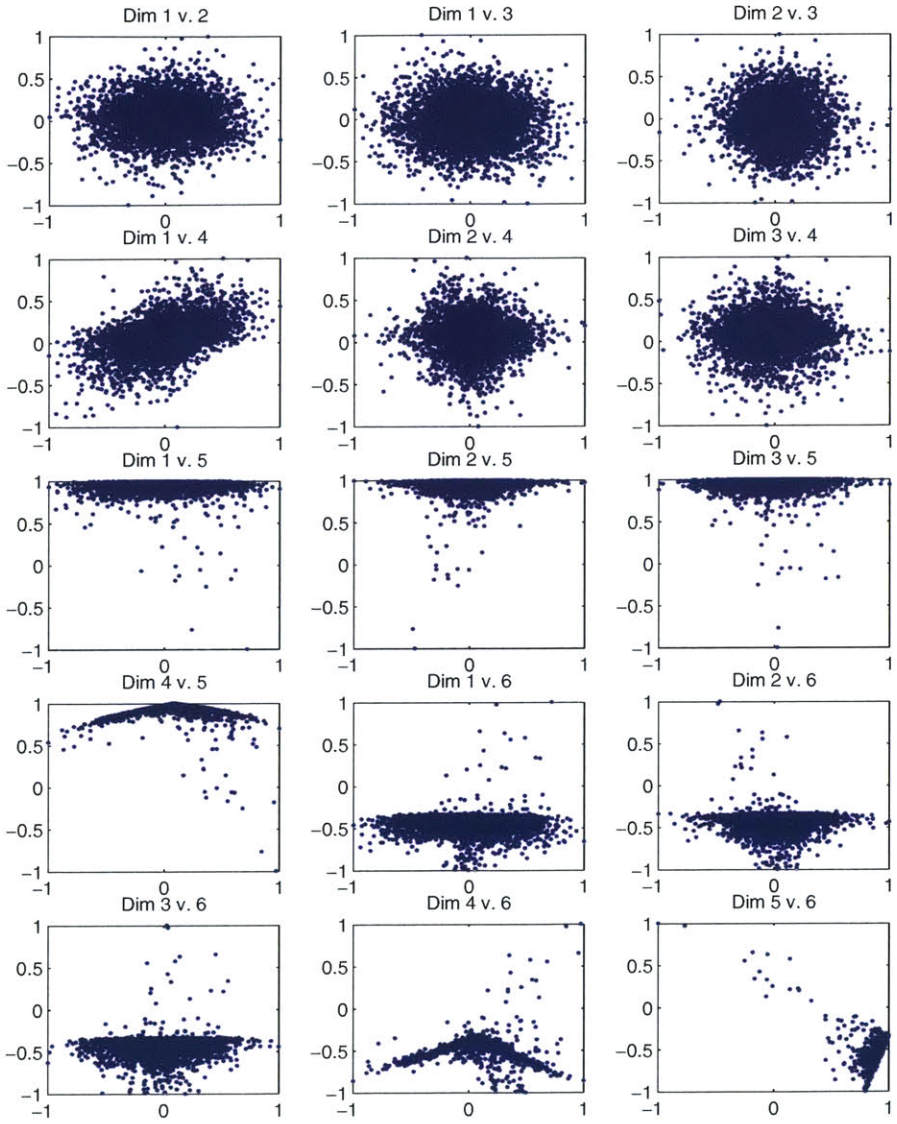


Figure 6-26: Plot of samples used by KDE Model for the 6 dimensional subspace learned by CCA (Flow and MFCC Differences). Dimensions 1-3 are audio and 4-6 are video.

sentation, using 256 PCA components for regularization. Table 6.10 summarizes the results. This time we see that even with a Gaussian model we benefit from using multiple CCA bases, obtaining 0.7% error. However, again we see that the KDE Model did the best with 0.4% error using a 6 dimensional subspace learned by CCA. Figure 6-27 shows the KDE model learned on the 6 dimensional subspace. This time we see more linear relationships, particularly between dimensions 1 and 4, 2 and 5, and 3 and 6. These are the relationships between the first, second and third sets of bases learned by CCA.

6.2.5 Summary and Discussion

In this section we explored the use of informative subspace learning techniques to help reduce the dimensionality of our data and improve performance. All of the techniques introduced in Chapter 4 were tested; JPCA, CCA, MMI and our MD method. We began by creating a synthetic dataset in which the audio and video observations shared a common, underlying low dimensional variable. Information about this variable was lifted and distributed amongst the higher dimensions of our observations. We tested two cases; one in which this lifting was completely linear, and a second case in which we added a nonlinearity. For both cases we knew the optimal subspace. We obtain sufficient statistics for the common,underlying variable by projecting onto the basis of this subspace without losing any information.

The results of the experiments carried out on this data reenforced the theory discussed in Chapter 4. We saw that when our data was linear and Gaussian, CCA was optimal. However, it was clear that CCA was the wrong choice for non-linear data. In the nonlinear case MMI and MD proved their worth finding subspaces that were close to optimal and outperforming all other techniques. It was also shown that for both sets of data, JPCA was suboptimal. Its objective criteria tries to preserve energy rather than preserving the information shared between modalities. Lastly, we discussed some of the potential problems with the MMI and MD techniques that resulted from their gradient descent optimization routines. Unfortunately, we found that our routine for MD was too slow and unstable to use on real, higher dimensional

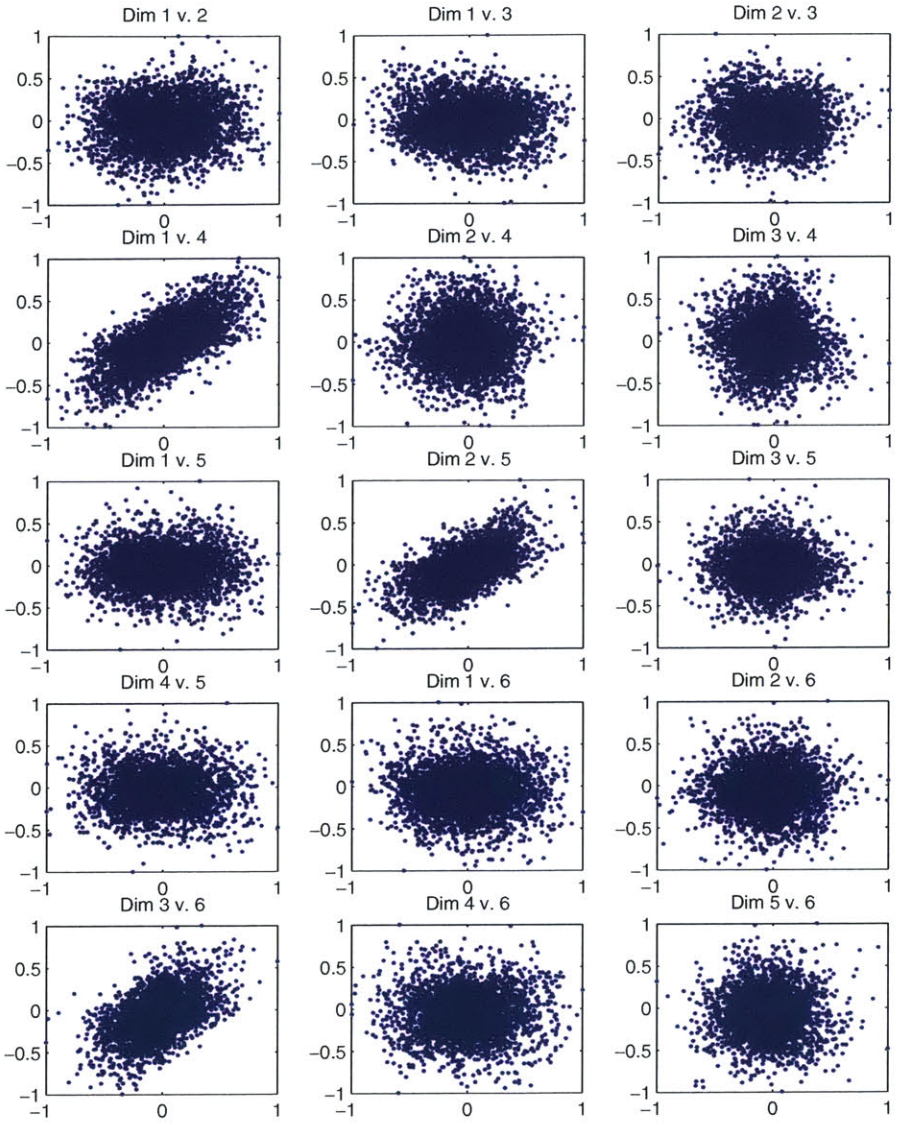


Figure 6-27: Plot of samples used by KDE Model for the 6 dimensional subspace learned by CCA (Image and MFCC Differences). Dimensions 1-3 are audio and 4-6 are video.

data. Thus, we left further study of this technique for future work.

Subsequently, we were left to compare the use of CCA and MMI on real data taken from the AVTIMIT database. We concentrated on learning a generic model for measuring audio-visual association using these informative subspace learning techniques for dimensionality reduction rather than PCA which was used in the previous experimental section. Immediately we saw the advantage of these techniques over PCA when using flow and MFCC differences as our video and audio representations. Previously, we needed to keep over 3 principle components for each modality to get any decent performance. However, both CCA and MMI found a single set of bases that resulted in equivalent or better performance than using PCA. The video bases learned by both CCA and MMI showed that the average vertical flow statistic was the most informative feature. PCA found this feature to be less important than the variance of flow in the x and y directions since these variance features had more variation. The experiments on these flow features using MMI combined with a KDE model in training were the first to show a potential advantage of using a non-parametric approach giving us 92% accuracy in our association detection task. This technique resulted in better performance than simply using CCA and a Gaussian model.

Next, we applied these techniques using image differences as our video representation. This representation provided the challenge of having much higher dimensionality than our optical flow statistics, giving us the opportunity to explore the effect of regularization. We showed results in which regularization was carried out in a naive way, using PCA as an initial dimensionality reduction prior to running CCA or MMI. Surprisingly we found that regularization only hurt CCA. CCA did the best when our PCA step only threw away 3% of the energy in our video data (512 principal components). This PCA plus CCA combination gave us 2% error, a significant improvement over the results in the previous experiment section when we obtained 13% using PCA only. We saw no signs of overfitting to the training data. It is important to note that we did not attempt to run CCA on the raw 5500 dimensional for practical reasons. Overfitting would most likely have occured in such a case. However, it is clear that by using at most 512 principal components we did not lose much information and

the results have shown that CCA had a sufficient amount of training data to avoid overfitting.

Our results for MMI were different. It was clear that this simple PCA regularization helped the MMI technique. By first reducing the dimensionality of the input, we helped restrict the search space for MMI and thus helped its optimization procedure avoid local maximums. MMI combined with a KDE model had equivalent or better performance than CCA and a Gaussian model when the regularization limited the input dimension to less than 64. Once we went beyond that threshold, MMI's performance began to degrade. Thus, we see that MMI is a more powerful technique, but has an optimization procedure that limits its practicality. We leave exploration of improvements to this algorithm for future work.

Another very interesting result when using our image differences as our video representation was that the bases learned by CCA and MMI were not easy to interpret visually. The bases that looked more informative, emphasizing the jaw and lip, provided less information than those that seemed to be "noisy" and appeared to be overfitting. However, this was not the case. The smoothly varying and visually appealing bases were less informative. This makes it difficult to apply regularization that imposes a particular prior on our learned bases, i.e. L2-regularization.

Lastly, we explored combining the robust subspace learning method of CCA with a non-parametric model rather than assuming Gaussianity. This combination produced our best results. CCA avoids local maxima problems and finds a two dimensional subspace in which projected data has maximal correlation coefficient. While this may not correspond to the subspace with the maximal mutual information, it does capture some dependency. Using a non-parametric approach such as a KDE to model the projected data in this subspace allows us to potentially capture other dependencies that are not captured in the second order statistics CCA concentrates on. This was particular useful when using optical flow features which are highly non-Gaussian. In some ways CCA is imposing its own form or regularization or capacity control by only looking at second or statistics. It makes sense to ignore higher order statistics if we do not have enough data to properly estimate them. Furthermore, CCA provides

multiple, orthogonal, informative 2 dimensional subspaces that can be combined to model more of the dependency between the modalities. We obtained less than 1% error in using this mixture of techniques for both image differences and optical flow.

6.3 Experiment 3: Exploring the Importance of Kernel Size

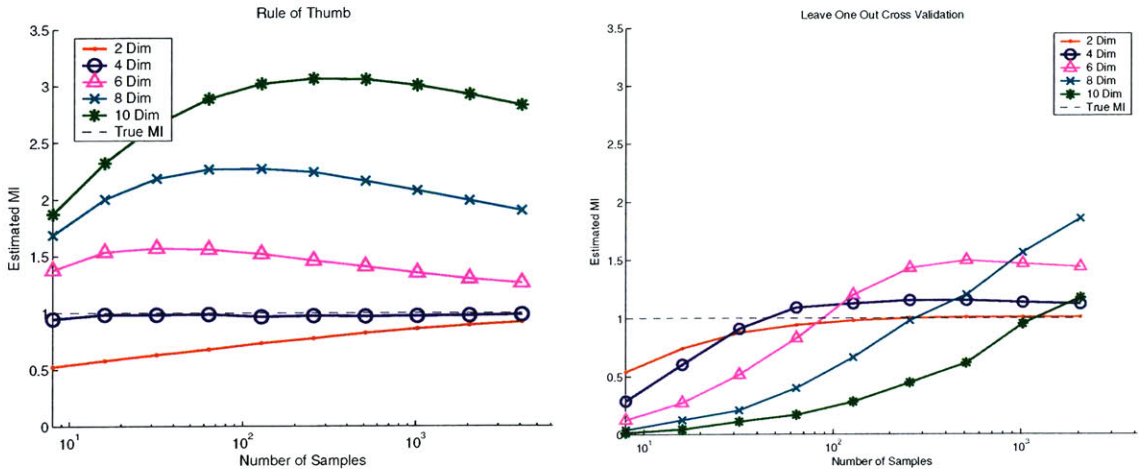
In Section 6.1 we saw that our KDE modeling techniques were extremely sensitive to dimensionality. A large degradation in performance occured when the data we were modeling had more than 4 dimensions. In our discussion of this section we conjectured that this poor performance may have been a result of having choosen a bad kernel size. In this section we briefly explore how different methods for choosing kernel size, and the dimensionality of our data affect our ability to estimate mutual information using a KDE.

We use our ability to estimate MI as our metric since, in the limit, both our online and offline modeling techniques are measuring MI as their likelihood statistic in our audio-visual association hypothesis test. We also simplify our experiments but using Gaussian data and fixing the amount of mutual information. That is, our synthetic n dimensional samples of $\mathbf{a}$ and $\mathbf{v}$ from $N(0, C_{av})$, with C_a and C_v being identity I_n, and choosing C_{av} so that $I(\mathbf{a}; \mathbf{v}) = 1$.

We test both the rule-of-thumb (ROT) and leave-one-out cross validation methods for choosing a kernel size described in Appendix A. For each dimensionality n tested, we draw a set of k samples and estimate its density and MI with a KDE using either ROT or LCV. We do this 100 times for each setting of n and k and record the average MI estimate. Figure 6-28 shows results for both ROT and LCV. Each line represents a different n, ranging from 1 to 5 (1 to 10 joint dimension). The estimated MI is plotted as a function of the number samples used.

These plots lead to some simple conclusions. First, it clear that the way in which we pick our kernel size has a large affect on our ability to estimate MI. We see two completely different trends for ROT and LCV. Second, we see that as we increase the dimensionality of our data, with these kernel picking techniques, our MI estimate generally gets worse.

The results for ROT show that with 2 dimensions we pick an over-smoothed kernel and under-estimate MI estimate. As we increase dimensionality, we go from over-



(a) (b)

Figure 6-28: Comparing Techniques for Choosing Kernel Size

smoothing to under-smoothing and overestimating MI. As we increase the number of samples used in our estimate, the trend of slowly converges to the true MI. However, visually following the trend, it seems that once we have over 6 dimensions it may take an extreme number of samples before we converge to the true MI estimate.

The results for LCV show a different trend. As the dimensionality increases, we are more prone to under-estimating MI. As the number of samples increase, we seem to converge to the true MI when our dimensionality is below 6. Once we get higher than that we do progressively worse.

This quick experiment has clearly shown that both kernel size and dimensionality of our data play an important role in our ability to estimate MI when using a KDE model. It is important to note that these methods for choosing kernel size are based on minimizing either the mean integrated squared error (MISE) or KL divergence between the estimate density and the true density. However, we may be willing to sacrifice our ability to match the true density if we could have a better estimate of MI, particularly when using an online method. We leave such exploration for future work.

Chapter 7

Discussion and Future Work

A main motivation for this thesis is to help promote the development of multi-person dialog systems by addressing the speech correspondence problem. We simplify this problem to its core task of detecting audio-visual association. Given pairs of audio and video observations of human speech, our task is to determine whether not they belong together. This seemingly simple task is attended by a set of interesting questions: What is the proper framework in which to discuss such a problem? Can we learn a simple model for our audio-visual observations. If so, how should they be modeled? How do we intelligently deal with the high dimensional nature of audio-visual data?

In Chapter 3 we answered the first of these questions, showing how this task can be naturally addressed using decision theory. We discussed how a simple hypothesis testing framework can be used for detecting audio-visual association. This statistical framework makes use of the fact that associated and non-associated audio-visual data have different distributions. For simplicity, we restrict ourselves to an observation model in which we assume our audio-visual samples are i.i.d.. We discussed how generative approaches explicitly model the different distributions for the data while discriminative approaches exploit these differences to directly learn an optimal decision function. In addition, for the generative approach, we discussed the tradeoffs between learning a model from training data and learning one online. We showed that, no matter how or when we learn our generative model, carrying out our hypothesis test ultimately boils down to some measurement of mutual information.

In Chapter 4 we addressed the practical issues that arise from using high dimensional audio-visual data. We discussed how standard techniques for dimensionality reduction, such as PCA, are suboptimal for our problem. We presented the concept of an informative subspace and introduced CCA and MMI as the proper methods for dimensionality reduction, in that they preserve the relevant dependency information between our observations. We show how CCA is optimal when the data is Gaussian and that MMI provides a more general non-parametric approach. Lastly, we presented a new technique we refer to as maximum discriminant (MD), which learns a low dimensional subspace which best perseveres the ability to discriminate between two different sets of data.

The AVTIMIT database used in our experiments was discussed in Chapter 5. We showed how we use a simple correlation based tracker to extract the lower facial region of speaking subjects. Different preprocessing techniques for extracting audio-visual features were also discussed. We argued that by preprocessing our raw audio-visual with techniques commonly used by the AVSR community we eliminate obvious irrelevant information and find features that are closely tied to speech production.

Chapter 6 presented a series of experiments on both synthetic data and real data. The synthetic data help reenforce our understanding of the theory presented in Chapters 3 and 4. Real data taken from the AVTIMIT database was also tested, allowing us to see how well this theory held up in practice. We primarily discuss our results in sections 6.1.7 and 6.2.5, but present a summary of some of our key results here: We saw how the features we extract during preprocessing can greatly affect our performance. Static audio-visual features produced non-i.i.d. observations and were only suitable for learning a person-specific model of audio-visual association. Differential features varied less across subjects, allowing us to learn a decent generic model for association. We saw that when our observations were close to being i.i.d. it was advantageous to learn some model from training data rather than relying on a online testing technique. Both non-parametric (discriminative and generative) and Gaussian modeling techniques worked well when dealing with low dimensional observations. However, performance using the non-parametric techniques rapidly degraded as we increased

dimensionality. Using PCA for dimensionality reduction, in many cases, we needed to keep close to 5 principal components to obtain any decent level of performance. With this 10 dimensional joint subspace our non-parametric techniques were outperformed by the simpler Gaussian model.

In section 6.2 we showed that by using the informative subspace learning techniques described in Chapter 4 instead of PCA, we could obtain similar performance with a much lower dimensional representation. While we demonstrated with synthetic data that the MMI technique had an advantage of being able to capture non-linear audio-visual relationships, both CCA and MMI were shown to learn similar subspaces for our audio-visual data. When our input representation was already relatively low-dimensional (such as flow statistics), MMI combined with a non-parametric KDE model outperformed CCA combined with a Gaussian model. However, when higher dimensionality input was used (such as image differences) the optimization routine for MMI seemed to find local maxima. We improved performance by imposing capacity control with PCA prior to running MMI, but could not beat the performance of CCA. This led us to combine CCA with a non-parametric model. That is, we let CCA find our informative subspace and then modeled our data in that subspace using a KDE rather than a simple Gaussian model. We argued that CCA imposes some form of regularization or capacity control by ignoring higher order statistic that are difficult to estimate with a finite set of training data. Using this approach we obtained close to 99% accuracy in detecting audio-visual association. This was the average performance on a test set of 10 people in which we performed 5 fold cross-validation, iteratively training on 8 people and testing on the 2 held out.

In summary, this thesis has presented a solid framework for addressing the problem of detecting audio-visual association. We have broken up our testing and training procedure into a set of simple components (see Figure5-2); one for preprocessing, one for dimensionality reduction, and one for the carrying out the final hypothesis testing. We developed each component separately and explored how each component affects the overall performance of the system. Preprocessing allowed us to remove irrelevant information in our data and supply features which are somewhat i.i.d..

The dimensionality reduction component used a set of bases learned from training data to supply a low-dimensional, informative representation of our data for the hypothesis testing component. This leaves the hypothesis testing component free to measure the likelihood of audio-visual association by treating its input as generic low dimensionality random variables. We were ultimately able to obtain close to 99% accuracy in detecting association with this structure (with preprocessing providing differential features, dimensionality reduction using bases learned with CCA and a hypothesis test which uses a non-parametric model of the data).

While we have obtained some promising results, our work on this problem is far from being complete. In the very near future, we plan on continuing our exploration of this problem by testing a larger portion of the AVTIMIT database. In addition, we wish to address the many interesting questions raised in our experimental sections that we were forced to leave unanswered. In particular, we would like to explore new methods for choosing kernel size to help improve performance of non-parametric modeling techniques when dealing with high dimensional data. We also plan on working toward improving our optimization routines for both MMI and MD. Furthermore, we would like to more thoroughly study the role of regularization in the context of the informative subspace learning techniques used in this thesis.

Lastly we wish to incorporate our framework for detecting association into an actual multi-person dialog system. Our framework was already designed with a real-time implementation in mind, detecting association using a sliding window of audio-visual observations. However, future work must also consider the potential problems caused by tracking errors and non-frontal faces. We also wish to study how our framework can be extended to better incorporate dynamics.

Appendix A

Density Estimation

In Section 3.1 we saw how to design a hypothesis test for our speech correspondence problems if we had access to the correct models for our audio visual data under each hypothesis. However, all we have to work with is a set of sample observations and some general knowledge about their nature (i.e. we know that when the audio and video are associated their joint density should exhibit some dependency).

In the following sections we will discuss two different approaches to learning a model for our observations from these training samples. First, we assume a Gaussian model and show how to estimate its parameters. We then show how to obtain a non-parametric model using a Kernel Density Estimate. We discuss these techniques in terms of estimating a probability distribution for some random variable $\mathbf{x} \in \Re^n$ when giving a set of N training examples, $\mathbf{x}_1$ through $\mathbf{x}_N$.

A.1 Learning a Gaussian Model

If we assume our random variable $\mathbf{x}$ is Gaussian it has a density of the following form:

$$p_x(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|C_x|^{1/2}} \exp\Big[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_x)^T C_x^{-1}(\mathbf{x}-\boldsymbol{\mu}_x)\Big] \tag{A.1}$$

which is characterized by two quantities; its mean $\boldsymbol{\mu}_x \in \Re^n$ and covariance $C_x \in Re^{n \times n}$

$$\boldsymbol{\mu}_x = E[\mathbf{x}] = \int_x \mathbf{x} p_x(\mathbf{x}) d\mathbf{x} \tag{A.2}$$

$$C_x = E[(\mathbf{x} - \boldsymbol{\mu}_x)(\mathbf{x} - \boldsymbol{\mu}_x)^T] = E[\mathbf{x}\mathbf{x}^T] - \boldsymbol{\mu}_x \boldsymbol{\mu}_x^T \tag{A.3}$$

Therefore learn a Gaussian model is equivalent to estimating these two quantities. Typical the means is estimated using the following unbiased estimator (which also happens to be the estimate of the mean that maximizes the likelihood of our data assuming Gaussianity):

$$\hat{\boldsymbol{\mu}}_x = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i \tag{A.4}$$

It is unbiased because:

$$E[\hat{\boldsymbol{\mu}}_x] = \frac{1}{N} \sum_{i=1}^{N} E[\mathbf{x}_i] = \boldsymbol{\mu}_x \tag{A.5}$$

Similarly there exists an unbiased estimator for the covariance.

$$\hat{C}_x = \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_x)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_x)^T \tag{A.6}$$

Although this estimator is unbiased it is not the maximum likelihood (ML) estimate. The ML estimate uses a factor of $1/N$ rather than $1/(N-1)$ and produces a bias. What follows is a simple proof that the estimator in Equation A.6 is unbiased:

$$E[\hat{C}_x] = \frac{1}{N-1} \sum_{i=1}^{N} (E[\mathbf{x}_i \mathbf{x}_i^T] - E[\mathbf{x}_i \hat{\boldsymbol{\mu}}_x^T] - E[\hat{\boldsymbol{\mu}}_x \mathbf{x}_i^T] + E[\hat{\boldsymbol{\mu}}_x \hat{\boldsymbol{\mu}}_x^T]) \tag{A.7}$$

Note that:

$$\begin{aligned} E[\mathbf{x}_i\hat{\boldsymbol{\mu}}_x^T] &= \frac{1}{N}\sum_{q=1}^{N} E[\mathbf{x}_i\mathbf{x}_q^T] \\ &= \frac{1}{N}(E[\mathbf{x}_i\mathbf{x}_i^T] + (N-1)E[\mathbf{x}_{q\neq i}]E[\mathbf{x}_i^T]) \\ &= \frac{1}{N}(C_x + \boldsymbol{\mu}_x\boldsymbol{\mu}_x^T + (N-1)\boldsymbol{\mu}_x\boldsymbol{\mu}_x^T) \\ &= \frac{1}{N}(C_x + N\boldsymbol{\mu}_x\boldsymbol{\mu}_x^T) \\ &= E[\hat{\boldsymbol{\mu}}_x\mathbf{x}_i^T] \end{aligned} \tag{A.8}$$

and

$$\begin{aligned} E[\hat{\boldsymbol{\mu}}_x\hat{\boldsymbol{\mu}}_x^T] &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} E[\mathbf{x}_i\mathbf{x}_j^T] \\ &= \frac{1}{N^2}(NE[\mathbf{x}_i\mathbf{x}_i^T] + N(N-1)E[\mathbf{x}_{i\neq j}]E[\mathbf{x}_j^T]) \\ &= \frac{1}{N}(C_x + N\boldsymbol{\mu}_x\boldsymbol{\mu}_x^T) \end{aligned} \tag{A.9}$$

Plugging Equations A.8, A.9 back into Equation A.7 yields:

$$\begin{aligned} E[\hat{C}_x] &= \frac{1}{N-1}\sum_{i=1}^{N}(C_x + \boldsymbol{\mu}_x\boldsymbol{\mu}_x^T - \frac{1}{N}(C_x + N\boldsymbol{\mu}_x\boldsymbol{\mu}_x^T)) \\ &= \frac{1}{N-1}(NC_x + N\boldsymbol{\mu}_x\boldsymbol{\mu}_x^T - C_x - N\boldsymbol{\mu}_x\boldsymbol{\mu}_x^T) \\ &= \frac{N-1}{N-1}C_x \\ &= C_x \end{aligned} \tag{A.10}$$

Thus, completing the proof. Note that if $\mathbf{x}$ is zero mean than the ML estimate, $\sum_i x_i x_i^T/N$ is also unbiased.

Putting everything together our density estimate is:

$$\hat{p}_x(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\hat{C}_x|^{1/2}} \exp\Big[-\frac{1}{2}(\mathbf{x}-\hat{\boldsymbol{\mu}}_x)^T C_x^{-1}(\mathbf{x}-\hat{\boldsymbol{\mu}}_x)\Big] \tag{A.11}$$

A.2 Kernel Density Estimation

There are many cases in which our data may not fit a Gaussian model. While we may be able to find some other parametric model that better describes our data, sometimes it is preferable to use samples of the data describe its own distribution. This is the goal Parzen window or kernel density estimators. They attempt to approximate $p_x(\mathbf{x})$ using N data samples:

$$\hat{p}_x(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} K(\mathbf{x} - \mathbf{x}_i; \mathbf{h}) = \frac{1}{N}\sum_{i=1}^{N} K(\mathbf{x}_i - \mathbf{x}; \mathbf{h}) \tag{A.12}$$

where $K(\mathbf{x}; \mathbf{h})$ is the kernel function with a bandwidth described by $\mathbf{h}$. The kernel function is usually symmetric and integrates to 1 so that $\hat{p}$ is a valid density. For this thesis we restrict ourself to product kernels in which each dimension is treated independently:

$$K(\mathbf{x}; \mathbf{h}) = \prod_{j=1}^{n} \frac{1}{h_{(j)}} K\left(\frac{x_{(j)}}{h_{(j)}}\right) \tag{A.13}$$

where $K()$ is a unit-variance kernel and $\mathbf{h} = [h_{(1)}, ..., h_{(n)}]^T$ is a vector containing kernel sizes / bandwidth for each dimension. For our experiments we simply use a Gaussian kernel:

$$k(x) = \frac{1}{\sqrt{2pi}} \exp(-\frac{1}{2}x^2) \tag{A.14}$$

This provides us we a smooth, easily evaluated kernel with an infinite region of support.

A.2.1 Choosing a Kernel Size

After choosing a form for our kernel all that is left for us to control is the kernel size. Choosing an optimal kernel size is difficult task and is an active area of research. Here we described two popular methods. However, we must first described what we mean by "optimal." A simple cost functions for measuring the fit or optimality of a kernel

density estimate is the Mean Integrated Squared Error (MISE):

$$MISE = E_{p_x}\Big[\int (p_x(\mathbf{x}) - \hat{p}_x(\mathbf{x}))^2 d\mathbf{x}\Big] \tag{A.15}$$

where $p_x()$ is the true density. (Note that, for simplicity, we restrict ourselves to single dimensional densities). If we assume the true density is close to Gaussian and we use a Gaussian kernel for $\hat{p}_x()$ it can be shown that the following "plug-in-estimate" minimizes an approximation to the MISE:

$$h_{(j),rot} = \Big(\frac{4\hat{\sigma}^5}{3N}\Big)^{1/5} \tag{A.16}$$

where

$$\hat{\sigma} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \frac{1}{N}\sum_{m=1}^{N} x_m)^2} \tag{A.17}$$

This "rule-of-thumb" for picking a kernel size was originally introduced by Silverman [29] and has the advantage of being simple and efficient to calculate. However, one may question the Gaussian assumption used to obtain this estimate. If we are assuming a Gaussian density why bother with this kernel density estimation? While it is true that this kernel size is optimal for a Gaussian density, it still may give a reasonable estimate for distributions that are more or less symmetric, unimodal, and have small tails. When dealing with multidimensional data we simply treat each dimension independently, estmating a different $\hat{\sigma}$ for each dimension.

An alternative cost function for measuring correctness of fit is the Kullback-Leibler divergence:

$$D(p_x||\hat{p}_x) = E_{p_x}\Big[\log p_x(x) - log\hat{p}_x(x)\Big] \tag{A.18}$$

In order to minimize this quantity we maximize the part we have control over, $E_{p_x}\big[log\hat{p}_x(x)\big]$. We estimate this with:

$$CV = \frac{1}{N}\sum_{i=1} N \log \hat{p}_{x:i}(x_i) \tag{A.19}$$

where

$$\hat{p}_{x:j}(x) = \frac{1}{N} \sum_{i \neq j}^{N} K(x - x_i; h) \tag{A.20}$$

is a leave-one-out estimator of the density. A line search is performed to find the h that maximizes CV. We refer to this method as maximum likelihood leave-one-out cross validation. The reason for the leave-one-out estimate is that if all the sample points were included the solution would be a kernel size of 0, giving use a kernel that acts as a delta function.

For our experiments we use Alex Ihler's excellent KD-tree based Kernel Density Estimation class for MATLAB [10].

Appendix B

Select Elements of Information Theory

This thesis makes judicious use of concepts described by information theory. In the sections below we quickly highlight the most relevant material. An excellent resource for further investigation is [5].

B.1 Entropy

Given a continuous random variable $\mathbf{x} \in \Re^n$ with probability density $p_x(\mathbf{x})$ and region of support S, differential entropy $H(\mathbf{x})$ is defined as:

$$H(\mathbf{x}) = E[-\log(p_x(\mathbf{x}))] = -\int_S p_x(\mathbf{x}) \log p_x(\mathbf{x}) d\mathbf{x} \tag{B.1}$$

and is sometimes alternatively written as $H(p_x)$ since it dependent only on the density.

Entropy describes the average uncertainty in a random variable. When using the natural log it is measured in nats and using log base 2 results in units of bits. A useful interpretation is that entropy is related to the number of bits on average required to describe the random variable.

As a quick example we can calculate the entropy of a Gaussian random variable

with zero mean and covariance C_x. Using the natural log for convenience we get:

$$\begin{aligned} H(\mathbf{x}) &= -\int p_x(\mathbf{x})\Big(-\frac{1}{2}\mathbf{x}^T C_x^{-1}\mathbf{x} - \frac{1}{2}\log(|2\pi C_x|)\Big) \\ &= \frac{1}{2}E[\mathbf{x}^T C_x^{-1}\mathbf{x}] + \frac{1}{2}\log((2\pi)^n|C_x|) \\ &= \frac{1}{2}(n + \log(2\pi^n|C_x|)) \\ &= \frac{1}{2}(\log(e^n) + \log((2\pi)^n|C_x|)) \\ &= \frac{1}{2}\log((2e\pi)^n|C_x|) \end{aligned} \tag{B.2}$$

where in the third step we use the linear and cyclic properties of the trace (and the fact that expectation is linear):

$$\begin{aligned} E[\mathbf{x}^T C_x^{-1}\mathbf{x}] &= tr(E[\mathbf{x}^T C_x^{-1}\mathbf{x}]) \\ &= E[tr(\mathbf{x}^T C_x^{-1}\mathbf{x})] \\ &= E[tr(C_x^{-1}\mathbf{x}\mathbf{x}^T)] \\ &= tr(C_x^{-1}E[\mathbf{x}\mathbf{x}^T]) \\ &= tr(C_x^{-1}C_x) \\ &= n \end{aligned} \tag{B.3}$$

It is interesting to note that a Gaussian random variable has the largest possible entropy for any other continuous random variable with the same mean and variance.

If we have two random variables $\mathbf{x}$ and $\mathbf{y}$ we can talk about joint and conditional entropy. The joint entropy is simply the same as the entropy defined on new vector valued random variable $[\mathbf{x};\mathbf{y}]$:

$$H(\mathbf{x},\mathbf{y}) = E_{p_{[x;y]}}[-\log p_{[x;y]}(\mathbf{x},\mathbf{y})] \tag{B.4}$$

Conditional entropy is:

$$H(\mathbf{y}|\mathbf{x}) = E_{p_{[x;y]}}[-\log p_{x|y}(\mathbf{x}|\mathbf{y})] \tag{B.5}$$

It can also easily be shown the following chain rule holds:

$$H(\mathbf{x},\mathbf{y}) = H(\mathbf{x}) + H(\mathbf{y}|\mathbf{x}) = H(\mathbf{y}) + H(\mathbf{x}|\mathbf{y}) \tag{B.6}$$

B.2 KL Divergence

Given two densities $p_x(\mathbf{x})$ and $q_x(\mathbf{x})$ for the same variable the relative entropy or Kullback-Leibler divergence is defined as:

$$\begin{aligned} D(p_x||q_x) &= E_p\left[\log\left(\frac{p_x(\mathbf{x})}{q_x(\mathbf{x})}\right)\right] \\ &= \int p_x(\mathbf{x}) \log\left(\frac{p_x(\mathbf{x})}{q_x(\mathbf{x})}\right) d\mathbf{x} \end{aligned} \tag{B.7}$$

which some measure of distance between distributions. It is strictly positive and zero only when $p_x() = q_x()$. However it is not a "true" distance metric in that it is not symmetric and does not satisfy the triangle inequality. between two probability densities.

B.3 Mutual Information

Given two random variables $\mathbf{x}$ and $mathbfy$, we can discuss the amount of shared information between the two in terms of their mutual information. Mutual information is defined as

$$\begin{aligned} I(\mathbf{x};\mathbf{y}) &= D(p_{[x;y]}(\mathbf{x},\mathbf{y})||p_x(\mathbf{x})p_y(\mathbf{y})) \\ &= H(\mathbf{x}) - H(\mathbf{x}|\mathbf{y}) \\ &= H(\mathbf{x}) + H(\mathbf{y}) - H(\mathbf{x},\mathbf{y}) \\ &= I(\mathbf{y};\mathbf{x}) \end{aligned} \tag{B.8}$$

and can be viewed as the reduction in uncertainty about one random variable due to another. Since it is a special case of KL divergence it is strictly positive. It is also

symmetric, implying that $\mathbf{x}$ tells us as much about $\mathbf{y}$ as $\mathbf{y}$ does about $\mathbf{x}$.

It is also interesting to note that the mutual information of a variable with its self $I(\mathbf{x};\mathbf{x})$ is equal to it's entropy $H(\mathbf{x})$. This is why entropy is sometimes referred to as self information.

Using Equations B.8 and B.2 we can easily show that the mutual information between two Gaussian random variables is

$$\begin{aligned} I(\mathbf{x};\mathbf{y}) &= \frac{1}{2}\Big(\log((2e\pi)^n|C_x|) + \log((2e\pi)^m|C_x|) - \log((2e\pi)^{n+m}|C_{[x;y]}|)\Big) \\ &= \frac{1}{2}\log\left(\frac{|C_x||C_y|}{|C_{[x;y]}|}\right) \end{aligned} \tag{B.9}$$

B.4 Data Processing Inequality

One important concept used by this thesis is the found in the data processing inequality. It formally states that intuitive notion that any processing carried out on data can only serve to reduce and at beast preserve the amount of information it can convey. That is:

$$I(\mathbf{x};\mathbf{y}) \geq I(\mathbf{x};f(\mathbf{y})) \tag{B.10}$$

and is equal if $f()$ is an invertible function or $f(\mathbf{y})$ is a sufficient statistic for $\mathbf{y}$ (as discussed in the next section).

B.4.1 Sufficient Statistics

The concept of a sufficient statistic is very important in stochastic estimation and detection as well as information theory. If we have some random variable $\mathbf{y}$ related to some $\mathbf{x}$ we say that $T(\mathbf{y})$ is a sufficient statistic if

$$p(\mathbf{x}|\mathbf{y}) = p(\mathbf{x}|T(\mathbf{y})) \tag{B.11}$$

which is equivalent to saying we can write the probability of $\mathbf{y}$ conditioned on $\mathbf{x}$ as the product of two independent functions:

$$p(\mathbf{y}|\mathbf{x}) = f(T(\mathbf{y}); \mathbf{x})g(\mathbf{y}) \tag{B.12}$$

Having this form produces:

$$\begin{aligned} p(\mathbf{y}) &= \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x} \\ &= g(\mathbf{y}) \int f(T(\mathbf{y}); \mathbf{x})p(\mathbf{x})d\mathbf{x} \end{aligned} \tag{B.13}$$

and therefore

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}) &= \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} \\ &= \frac{f(T(\mathbf{y}); \mathbf{x})p(\mathbf{x})}{\int f(T(\mathbf{y}); \mathbf{x})p(\mathbf{x})d\mathbf{x}} \\ &= p(\mathbf{x}|T(\mathbf{y})) \end{aligned} \tag{B.14}$$

which shows that Equation B.11 and B.12 are equivalent conditions.

Any statistic of $\mathbf{y}$ forms the Markov chain $\mathbf{x} \rightarrow \mathbf{y} \rightarrow T(\mathbf{y})$ or equivalently $\mathbf{x} \leftarrow \mathbf{y} \leftarrow T(\mathbf{y})$ which means the joint density can be written as

$$\begin{aligned} p(\mathbf{x}, \mathbf{y}, T(\mathbf{y})) &= p(\mathbf{x})p(\mathbf{y}|\mathbf{x})p(T(\mathbf{y})|\mathbf{y}) \\ &= p(\mathbf{x}, \mathbf{y})p(T(\mathbf{y})|\mathbf{y}) \\ &= p(\mathbf{x}|\mathbf{y})p(\mathbf{y})p(T(\mathbf{y})|\mathbf{y}) \\ &= p(\mathbf{x}|\mathbf{y})p(T(\mathbf{y}), \mathbf{y}) \\ &= p(\mathbf{x}|\mathbf{y})p(\mathbf{y}|T(\mathbf{y}))p(T(\mathbf{y})) \end{aligned} \tag{B.15}$$

which implies $\mathbf{x}$ and $T(\mathbf{y})$ are independent given $\mathbf{y}$. That is

$$p(\mathbf{x}, T(\mathbf{y})|\mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y}, T(\mathbf{y}))}{p(\mathbf{y})} = \frac{p(\mathbf{x}, \mathbf{y})p(T(\mathbf{y})|\mathbf{y})}{p(\mathbf{y})} = p(\mathbf{x}|\mathbf{y})p(T(\mathbf{y})|\mathbf{y}) \tag{B.16}$$

which means $I(\mathbf{x}; \mathbf{y}|T(\mathbf{y})) = 0$. This allows us to prove the data processing inequality

for any function $T()$. Using the chain rule mutual information can be expanded in the following two ways:

$$\begin{aligned} I(\mathbf{x};\mathbf{y},T(\mathbf{y})) &= I(\mathbf{x};T(\mathbf{y})) + I(\mathbf{x};\mathbf{y}|T(\mathbf{y})) \\ &= I(\mathbf{x};\mathbf{y}) + I(\mathbf{x};T(\mathbf{y})|\mathbf{y}) \end{aligned} \tag{B.17}$$

Since mutual information is strictly positive, setting these results to equal each other

$$\begin{aligned} I(\mathbf{x};T(\mathbf{y})) + I(\mathbf{x};\mathbf{y}|T(\mathbf{y})) &= I(\mathbf{x};\mathbf{y}) + I(\mathbf{x};T(\mathbf{y})|\mathbf{y}) \\ I(\mathbf{x};T(\mathbf{y})) + I(\mathbf{x};\mathbf{y}|T(\mathbf{y})) &= I(\mathbf{x};\mathbf{y}) + 0 \\ I(\mathbf{x};T(\mathbf{y})) &= I(\mathbf{x};\mathbf{y}) - I(\mathbf{x};\mathbf{y}|T(\mathbf{y})) \\ I(\mathbf{x};T(\mathbf{y})) &\leq I(\mathbf{x};\mathbf{y}) \end{aligned} \tag{B.18}$$

which shows we have equality if $I(\mathbf{x};\mathbf{y}|T(\mathbf{y})) = 0$.

However, if $T(\mathbf{y})$ is a sufficient statistic we can plug Equation B.11 into the line 3 of the Equations B.15 and show that

$$\begin{aligned} p(\mathbf{x},\mathbf{y},T(\mathbf{y})) &= p(\mathbf{x}|\mathbf{y})p(\mathbf{y})p(T(\mathbf{y})|\mathbf{y}) \\ &= p(\mathbf{x}|T(\mathbf{y}))p(T(\mathbf{y})|\mathbf{y})p(\mathbf{y}) \end{aligned} \tag{B.19}$$

This implies that when $T(\mathbf{y})$ is sufficient we have the Markov chains $\mathbf{x} \leftarrow T(\mathbf{y}) \leftarrow \mathbf{y}$ and $\mathbf{x} \rightarrow T(\mathbf{y}) \rightarrow \mathbf{y}$. Thus $I(\mathbf{x};\mathbf{y}|T(\mathbf{y})) = 0$, proving

$$I(\mathbf{x};T(\mathbf{y})) = I(\mathbf{x};\mathbf{y}) \tag{B.20}$$

Therefore when $T(\mathbf{y})$ is a sufficient statistic we achieve the maximum mutual information with $\mathbf{x}$.

Appendix C

CCA

Canonical Correlation was originally proposed by Hotelling in 1936. Given two zero-mean random variables $X \in \Re^n$ and $Y \in \Re^m$, canonical correlation is defined over one-dimensional projections of the random variable as:

$$\{h_x, h_y\} = \arg\max_{h_x, h_y} \frac{E\left\{h_x^T X Y^T h_y\right\}}{\sqrt{E\left\{h_x^T X X^T h_x\right\}}\sqrt{E\left\{h_y^T Y Y^T h_y\right\}}} = \arg\max_{h_x, h_y} \frac{h_x^T C_{xy} h_y}{\sqrt{h_x^T C_x h_x}\sqrt{h_y^T C_y h_y}} \quad \text{(C.1)}$$

where $h_x \in \Re^m$ and $h_y \in \Re^m$. It looks for the projections of X and Y that have maximum correlation coefficient ρ.

C.1 Derivation

Notice that scaling h_x or h_y will have no effect on the maximum in Equation C.1 This allows the problem to be reformulated as:

$$\begin{aligned} \{h_x, h_y\} &= \arg\max_{h_x, h_y} \; h_x^T C_{xy} h_y \\ \text{s.t.} \quad & h_x^T C_x h_x = h_y^T C_y h_y = 1 \end{aligned} \quad \text{(C.2)}$$

Using Lagrangian multipliers the solution for CCA becomes the stationary points of the following objective function:

$$J_{CCA}(h_x, h_y, \rho_x, \rho_y) = h_x^T C_{xy} h_y - \frac{\rho_x}{2}(h_x^T C_x h_x - 1) - \frac{\rho_y}{2}(h_y^T C_y h_y - 1) \tag{C.3}$$

Differentiating with respect h_x and *beta* yields

$$\frac{\partial J_{CCA}}{\partial h_x} = C_{xy} h_y - \rho_x C_x h_x = 0 \tag{C.4}$$

$$\frac{\partial J_{CCA}}{\partial h_y} = C_{xy}^T h_x - \rho_y C_y h_y = 0 \tag{C.5}$$

Using Equation C.4 and solving for h_x gives:

$$h_x = \frac{C_x^{-1} C_{xy} h_y}{\rho_x} \tag{C.6}$$

Plugging this into Equation C.5

$$\begin{aligned} (\frac{1}{\rho_x} C_{xy}^T C_x^{-1} C_{xy} - \rho_y C_y) h_y &= 0 \\ C_{xy}^T C_x^{-1} C_{xy} h_y &= \rho_y \rho_x C_y h_y \\ C_y^{-1} C_{xy}^T C_x^{-1} C_{xy} h_y &= \rho_y \rho_x h_y \end{aligned} \tag{C.7}$$

Note that multiplying Equation C.4 by h_x^T and Equation C.5 by h_y^T and setting them equal results in

$$\begin{aligned} h_x^T C_{xy} h_y - \rho_x (h_x^T C_x h_x) &= h_y^T C_{xy}^T h_x - \rho_y (h_y^T C_y h_y) \\ h_x^T C_{xy} h_y - \rho_x (1) &= h_x^T C_{xy} h_y - \rho_y (1) \\ \rho_x &= \rho_y = \rho \end{aligned} \tag{C.8}$$

where the second step used the constraints of Equation C.2 and the fact that $h_y^T C_{xy}^T h_x = h_x^T C_{xy} h_y$. This yields the following eigenvalue problem

$$C_y^{-1} C_{xy}^T C_x^{-1} C_{xy} h_y = \rho^2 h_y \tag{C.9}$$

and similarly

$$C_x^{-1}C_{xy}C_y^{-1}C_{xy}^T h_x = \rho^2 h_x \tag{C.10}$$

With some rearranging the solution to CCA can be expressed as a single generalized eigenvalue problem:

$$\begin{pmatrix} 0 & C_{xy} \\ C_{xy}^T & 0 \end{pmatrix} \begin{pmatrix} h_{x,i} \\ h_{y,i} \end{pmatrix} = \rho_i \begin{pmatrix} C_x & 0 \\ 0 & C_y \end{pmatrix} \begin{pmatrix} h_{x,i} \\ h_{y,i} \end{pmatrix} \tag{C.11}$$

Solving these equations for all eigenvectors yields a set of orthogonal $h_{x,i}$s,(and $h_{y,i}$s) and can be sorted in order of their eigenvalues (or ρ_i). Therefore instead of solving for the single best h_x and h_y a set of the top $p = min(m, n)$ solutions can be found and we will refer to them as p columns of H_x and H_y.

However, in practice, it may be simpler to just solve eigenvalue problem in Equation C.9 for the h_ys and then use Equation C.6 to solve for the h_xs. Note that $\rho = h_x^T C_{xy} h_y$ and therefore is the correlation coefficient.

C.2 Implementing CCA with SVD

It has been shown above that the canonical correlations between data X and Y can be found by solving the eigenvalue Equations C.9 and C.10. The solution to these equations require computing the full covariance of XY to obtain C_x,C_{xy}, etc. These costly computations can be avoided by taking advantage of a handy little tool, Singular Value Decomposition (SVD). Performing SVD on the data gives us

$$X = U_x S_x V_x^T \tag{C.12}$$

$$Y = U_y S_y V_y^T \tag{C.13}$$

which allows the covariances to be expressed as

$$\begin{aligned} C_x = E[XX^T] &= \tfrac{1}{N-1} U_x S_x V_x^T V_x S_x U_x^T = \tfrac{1}{N-1} U_x S_x^2 U_x^T \\ C_{xy} = E[XY^T] &= \tfrac{1}{N-1} U_x S_x V_x^T V_y S_y U_y^T \end{aligned} \tag{C.14}$$

and so on.

Using the SVD and applying it to Equation C.10 we obtain

$$\begin{aligned}(U_x S_x V_x^T V_y S_y U_y^T U_y S_y^{-2} U_y^T U_y S_y V_y^T V_x S_x U_x^T - U_x S_x^2 U_x^T \rho^2) h_x &= 0 \\ (U_x S_x V_x^T V_y V_y^T V_x S_x U_x^T - U_x S_x^2 U_x^T \rho^2) h_x &= 0\end{aligned} \tag{C.15}$$

Multiplying both sides by $S_x^{-1} U_x^T$ gives

$$\begin{aligned}(V_x^T V_y V_y^T V_x S_x U_x^T - S_x U_x^T \rho^2) h_x &= 0 \\ (V_x^T V_y V_y^T V_x - I\rho^2) S_x U_x^T h_x &= 0 \\ (KK^T - I\rho^2) S_x U_x^T h_x &= 0\end{aligned} \tag{C.16}$$

where

$$K = V_x^T V_y = U_k S_k V_k^T \tag{C.17}$$

which leads to

$$\begin{aligned}(U_k S_k^2 U_k^T - I\rho^2) S_x U_x^T h_x &= 0 \\ (S_k^2 U_k^T - \rho^2 U_k^T) S_x U_x^T h_x &= 0 \\ (S_k^2 - \rho^2) U_k^T S_x U_x^T h_x &= 0\end{aligned} \tag{C.18}$$

In order to maximize ρ we set

$$h_x = U_x S_x^{-1} U_k \tag{C.19}$$

which gives

$$\rho^2 I = S_k \tag{C.20}$$

Similarly it can be shown

$$h_y = U_y S_y^{-1} V_k^T \tag{C.21}$$

Appendix D

L2 Regularization

In this chapter we will discuss different interpretations of L2 regularization. The main goal of imposing regularization in learning techniques is to limit the risk of overfitting to training data. This risk of overfitting occurs when supplied with less training examples than degrees of freedom. Regularization is a way to deal with finite sampling problems such as this in order help our learning procedures generalize better.

One interpretation of regularization is that it imposes a prior on the learned parameters, thus reducing the degrees of freedom. Another view is that regularization provides a robust solution when training data is corrupted by noise. We will review these and other interpretations of L2 regularization in the context linear regression and canonical correlation analysis. The majority of this chapter summarizes and extends the regularization discussion found in [1].

D.1 Least Squares Linear Regression

We start with the extensively studied problem of linear least squares regression. Given a set N x_i and y_i pairs observations of linear least squares regression finds the best linear function of the x_is to predict the y_is in a least squared error sense. Here we restrict the problem such that $x_i \in \Re^n$ and $y_i \in \Re^1$ for simplicity. The linear function is represented by the vector h.

D.1.1 Minimizing Squared Error

Linear regression can be viewed as solving the following:

$$\begin{aligned} h_{LS} &= \arg\min_h \sum_{i=1}^{N} (y_i - h^T x_i)^2 = \arg\min_h J_{ls}(h) \\ h_{LS}* &= \arg\min_h (y - h^T X)(y - h^T X)^T \end{aligned} \tag{D.1}$$

where $y = [y_1 ... y_k] \in \Re^{1 \times N}$ and $X = [x_1 ... x_N] \in \Re^{n \times N}$. The above equation shows that we are trying to find the linear function h that minimize the sum of squared errors. The solution to this problem is very straight forward. Solving for the stationary point of $J_{ls}(h)$ leads to:

$$\begin{aligned} \frac{\partial J_{ls}(h)}{\partial w} &= 0 \\ \frac{\partial}{\partial w}(yy^T - 2h^T X y^T + h^T X X^T h) &= 0 \\ -2(Xy^T - XX^T h) &= 0 \\ h_{LS} &= (XX^T)^{-1} X y^T \\ h_{LS} &= C_x^{-1} C_{xy} \end{aligned} \tag{D.2}$$

C_x is the sample covariance of x_i, XX^T/N and C_{xy} is the sample cross covariance between x_i and y_i, Xy^T/N.

D.1.2 Maximizing the Likelihood of h

A second interpretation of least squares regression is that it finds the maximum likelihood estimate of h give data X and y and the following model:

$$y = h^T X + n \tag{D.3}$$

with $n = [n_1 ... n_k] \in \Re^{1 \times k}$ and each i.i.d n_i being a gaussian random variable with zero mean and a variance of σ_n^2. Given this model and h the probability of the data

is simply

$$\begin{aligned} p(X, y|h) &= p(n) \\ &= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(- \frac{(y_i - h^T x_i)^2}{2\sigma_n^2} \right) \\ &= \frac{1}{\sqrt{2\pi\sigma_n^2}^k} \exp\left(- \frac{(y - h^T X)(y - h^T X)^T}{2\sigma_n^2} \right) \end{aligned} \tag{D.4}$$

Finding the h which maximizes this probability leads to

$$\begin{aligned} h_{LS} &= \arg\max_h p(X, y|h) \\ &= \arg\max_h \log\left(p(X, y|h)\right) \\ &= \arg\min_h - \log\left(p(X, y|h)\right) \\ &= \arg\min_h k \log\sqrt{2\pi\sigma_n^2} + \frac{(y - h^T X)(y - h^T X)^T}{2\sigma_n^2} \\ &= \arg\min_h (y - h^T X)(y - h^T X)^T \end{aligned} \tag{D.5}$$

which is the same objective function shown previously. Therefore we now have two equivalent interpretations of least squares. It can either be viewed as minimizing the squared error between y and a linear function of X or as the maximum likelihood estimate of h in which y is generated by the Gaussian model in Equation D.3.

D.2 Regularized Least Squares (Ridge Regression)

The solution shown in Equation D.2 depends on the sample covariance of the data. This can be problematic when we have a small number of samples (small k) or our data is corrupted by noise or outliers. In fact the sample covariance C_x may not be invertible. In an attempt to combat these problems Ridge Regression adds L2 regularization to least squares linear regression. The role of this regularization has various interpretations.

D.2.1 Minimizing Squared Error plus L2 Norm Constraint

One view of Ridge Regression is that it simply adds a norm constraint to the least squares objective function:

$$h_{RR} = \arg\min_h (y - h^T X)(y - h^T X) + \lambda h^T R h \tag{D.6}$$

where λ is a parameter that controls the tradeoff between the squared error term and the norm constraint defined by the positive semidefinte matrix R. In its most simple form R is the identity matrix and the norm is simply the squared L2 norm. The solution to this problem is as follows

$$\begin{aligned} \frac{\partial J_{RR}(h)}{\partial w} &= 0 \\ \frac{\partial}{\partial w}(yy^T - 2h^T X y^T + h^T X X^T h + \lambda h^T R h) &= 0 \\ -2(Xy^T - XX^T h - \lambda R h) &= 0 \\ h_{RR} &= (XX^T + \lambda R)^{-1} X y^T \\ h_{RR} &= (C_x + \lambda R)^{-1} C_{xy} \end{aligned} \tag{D.7}$$

D.2.2 Regularization as a Prior on h

Looking at Ridge Regression from a Bayesian point of view we can think about the problem in terms of the model in Equation D.3, but this time treat h as a random variable with some prior probability. Under the assumption h is Gaussian we have

$$p(h) = \frac{1}{\sqrt{|2\pi C_h|}} \exp\Big(-.5 h^T C_h^{-1} w \Big) \tag{D.8}$$

and the posterior probability of h given the data is

$$\begin{aligned} p(h|X,y) &\propto p(X,y|h)p(h) \\ &\propto p(n)p(h) \\ &\propto \exp\left(-\frac{(y - h^T X)(y - h^T X)^T}{2\sigma_n^2} - \frac{h^T C_h^{-1} w}{2} \right) \end{aligned} \tag{D.9}$$

where we have removed all the constants that don't depend on h. Maximizing this posterior probability leads to

$$\begin{aligned} h_{RR} &= \arg\max_h p(h|X,y) \\ &= \arg\max_h \log\left(p(h|X,y)\right)) \\ &= \arg\min_h -\log\left(p(h|X,y)\right) \\ &= \arg\min_h \frac{(y-h^TX)(y-h^TX)^T}{\sigma_n^2} + h^TC_h^{-1}w \\ &= \arg\min_h (y-h^TX)(y-h^TX)^T + \sigma_n^2 h^TC_h^{-1}w \end{aligned} \tag{D.10}$$

This is equivalent to the solution shown in Equation D.7 with λ set to be σ_n^2 and R is the inverse covariance of h. This shows us that Ridge Regression finds the h that maximizes the posterior probability of our data when we have a Gaussian prior on h.

D.2.3 Regularization in relation to an Observation Noise Model

A third usefully interpretation of Ridge Regression is that is address the problem of having noisy observations of data. The model in Equation D.3 has noise on y but none on X. If we assume we actually observe $\tilde{X}$ rather than the true X and $\tilde{X} = X + S$ where s_i is i.i.d zero mean Gaussian noise with covariance C_s the model is changed to:

$$y = h^T(\tilde{X} - S) + n \tag{D.11}$$

It can be shown that h_{RR} maximizes of the expected log likelihood of the data under this model [1]. That is

$$\begin{aligned} h_{RR} &= \arg\max_h E_S\big[\log(p(\tilde{X}, y|h, S))\big] \\ &= (C_{\tilde{x}} + C_s)^{-1}C_{\tilde{x}y} \end{aligned} \tag{D.12}$$

While [1] shows a full proof of this we present a simple alternative here. The model in Equation D.11 can be compared with the simple Least squares model in Equation D.3. We see in this new model we observe $\tilde{X}$ rather than the noise free X. The solution of the least squares problem involves the sample covariances C_x and C_xy. Under this model we see that:

$$\begin{aligned} C_x &= \frac{XX^T}{N} \\ &= \frac{1}{N}(\tilde{X} - S)(\tilde{X} - S)^T \\ &= \frac{1}{N}\Big(\tilde{X}\tilde{X}^T - S\tilde{X}^T - \tilde{X}S^T + SS^T\Big) \end{aligned} \tag{D.13}$$

Here the covariance of the noise free data X depends on our observations $\tilde{X}$ and knowledge of what noise S has corrupted it. Taking the expected value with respect to S yeilds

$$\begin{aligned} E_S[C_x] &= \frac{1}{N}\Big(\tilde{X}\tilde{X}^T - E_S[S]\tilde{X}^T - \tilde{X}E_S[S^T] + E_S[SS^T]\Big) \\ &= C_{\tilde{x}} + C_s \end{aligned} \tag{D.14}$$

where we use the fact that the sample covariance,SS^T/N is unbiased since S is zero mean. That is the expected value of the sample covariance is the true covariance. Similarly we show that the expected sample cross covariance is

$$\begin{aligned} E_S[C_{xy}] &= \frac{E_S[Xy^T]}{N} \\ &= \frac{1}{N}E_S[(\tilde{X} - S)y^T] \\ &= \frac{1}{N}\tilde{X}y^T - E_S[S]y^T \\ &= C_{\tilde{x}y} \end{aligned} \tag{D.15}$$

The cross covariance is unaffected by the added noise on X. Substituting these expected covariances into the least squares solution shows that

$$\begin{aligned} h_{RR} &= E_S[C_x]^{-1} E_S[C_{xy}] \\ &= (C_{\tilde{x}} + C_s)^{-1} C_{\tilde{x}y} \end{aligned} \tag{D.16}$$

which is equivalent to D.7 with λR set to be C_s and change of notation from observing X to observing $\tilde{X}$.

In summary, Ridge Regression yields the general solution shown in Equation D.7 and has the following useful interpretations:

1. Ridge Regression is the solution to a least squares problem with an added norm constraint. The tradeoff between minimizing squared error and the norm defined by $h^T R h$ is controlled by the parameter λ

2. Ridge Regression produces the h that maximizes the posterior probability $P(h|X, y)$ under the model in Equation D.3 when there is Gaussian prior on h. In this situation λR is equivalent to the inverse covariance of the prior on h, C_h^{-1}. A stronger prior implies a small covariance and large in covariance and thus the C_h^{-1} dominates. Having a weak prior implies the opposite, with the extreme case when there is an improper uniform prior on h and h_{RR} becomes equivalent to h_{LS}.

3. The h given by Ridge Regression maximizes the expected log likelihood of the data under the model in Equation D.11. This expectation is taken with respect to the noise S added to X. In this interpretations λR represents the covariance of the noise , C_S.

These interpretations can help in choosing the appropriate λ and R for a particular problem. In most cases, however, simple forms are chosen for R and cross validation is used to find the best λ. In such situations it is sometimes more useful to limit λ to be between 0 and 1 and change the form of the Ridge Regression solution to be

$$h_{RR} = (1-\lambda)((1-\lambda)C_x + \lambda R)^{-1} X y^T \tag{D.17}$$

which is equivalent to Equation D.7 with λ replaced with $\lambda/(1-\lambda)$.

D.3 Regularized Canonical Correlation

A full derivation of CCA was given in Appendix C. This derivation was straight forward and showed how to find vectors h_x and h_y to maximizes the correlation coefficient between $h_x^T X$ and $h_y^T Y$, where X is a set of N i.i.d samples of $x_i \in \Re^n$ and Y a set of N samples of $y_i \in \Re^m$.

Finding a solution to CCA requires calculating sample covariances C_x, C_y, and C_{xy}. Therefore, the solution is sensitive to the same problems with linear regression such as finite sampling, noisy observations and outliers. This was the inspiration of Regularized Canonical Correlation (RCCA) or Canonical Ridge Analysis originally proposed in [33]. In it's most general form RCCA finds to following:

$$\{h_x, h_y\} = \frac{h_x^T C_{xy} h_y}{\sqrt{h_x^T (C_x + \lambda_x R_x) h_x}\sqrt{h_y^T (C_y + \lambda_y R_y) h_y}} \tag{D.18}$$

which like CCA is invariant to the scale of h_x and h_y which leads to the following equivalent optimization

$$\begin{aligned} \{h_x, h_y\} &= \arg\max_{h_x, h_y} \; h_x^T C_{xy} h_y \\ \text{s.t.} \quad & h_x^T (C_x + \lambda_x R_x) h_x = h_y^T (C_y + \lambda_y R_y) h_y = 1 \end{aligned} \tag{D.19}$$

Following the same procedure outlined in the derivation shown in Section C.1 shows that solving RCCA is equivalent to solving the following eigenvalue problems.

$$(C_y + \lambda_y R_y)^{-1} C_{xy}^T (C_x + \lambda_x R_x)^{-1} C_{xy} h_y = \rho^2 h_y \tag{D.20}$$

$$(C_x + \lambda_x R_x)^{-1} C_{xy} (C_y + \lambda_y R_y)^{-1} C_{xy}^T h_x = \rho^2 h_x \tag{D.21}$$

Comparing the solution to CCA in derived in Section C.1 to that of RCCA hints at relationship to how one goes from Least Squares to Ridge Regression. A probabilistic interpretation of CCA is presented in [1] and a link between RCCA and Ridge Regression is made.

The role of regularization in RCCA can be interpreted in a similar manor to that described in the previous section. That is, if we assume that our observations of X and Y are corrupted by noise S_x and S_y only our estimates of C_x and C_y will be affected (As shown in Equations D.14 and D.15). Therefore, we see that RCCA simply plugs in the expected value of C_x and C_y with respect to the noise, and $\lambda_x R_x$ and $\lambda_y R_y$ represent C_{s_x} and C_{s_y} respectively.

Appendix E

Entropy Gradient Calculations

E.1 Law of Large Numbers Approximation

Differential entropy is defined as follows:

$$h(X) = -E[\log(p_X(\mathbf{x}))] = -\int p_X(\mathbf{x}) \log(p_X(\mathbf{x}))dx \tag{E.1}$$

We can estimate $h(X)$ by using a kernel density estimate

$$\hat{h}(X) = -E[\log(\hat{p}_X(\mathbf{x}))] \tag{E.2}$$

Using the law of large numbers we can approximate the expectation in equation E.2 as:

$$\hat{h}(X) = -\frac{1}{N}\sum_{i=1}^{N} \log(\hat{p}_X(\mathbf{x}_i)) \tag{E.3}$$

To find the gradient of $\hat{h}(X)$ at a particular sample point $\mathbf{x}_s$ we must calculate

$$\frac{\partial}{\partial \mathbf{x}_s}\hat{h}(X) = -\frac{1}{N}\frac{1}{\hat{p}_X(\mathbf{x}_s)}\frac{\partial}{\partial \mathbf{x}_s}\hat{p}_X(\mathbf{x}_s) \tag{E.4}$$

where

$$\frac{\partial}{\partial \mathbf{x}_s}\hat{p}_X(\mathbf{x}_s) = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial \mathbf{x}_s}K(\mathbf{x}_i - \mathbf{x}_s; \mathbf{h}) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{K}'(\mathbf{x}_s - \mathbf{x}_i; \mathbf{h}) \tag{E.5}$$

with $\mathbf{K}'()$ is the gradient of the kernel. (The switch of variable positions in the second equality of Equation E.5 is a result of $\mathbf{K}'()$ being an odd function).

Let us assume the kernel function $K()$ is a product kernel (w/ independent dimensions) such that:

$$K(\mathbf{x}; \mathbf{h}) = \prod_{i=1}^{d}\frac{1}{h_i}K_i(\frac{x_i}{h_i}) \tag{E.6}$$

where $K_i()$ is a unit-variance kernel and $\mathbf{h} = [h_1, ..., h_d]^T$ is a vector containing kernel sizes for each dimension. Then our kernel gradient is:

$$\mathbf{K}'(\mathbf{x}; \mathbf{h}) = \frac{\partial}{\partial \mathbf{x}}K(\mathbf{x}; \mathbf{h}) = \left[\frac{\partial}{\partial x_1}K(\mathbf{x}; \mathbf{h}) \quad \frac{\partial}{\partial x_2}K(\mathbf{x}; \mathbf{h}) \quad ... \quad \frac{\partial}{\partial x_d}K(\mathbf{x}; \mathbf{h})\right] \tag{E.7}$$

with

$$\frac{\partial}{\partial x_k}K(\mathbf{x}; \mathbf{h}) = \left(\prod_{i\neq k}^{d}\frac{1}{h_i}K_i(\frac{x_i}{h_i})\right)\left(\frac{\partial}{\partial x_k}\frac{1}{h_k}K_k(\frac{x_k}{h_k})\right) = K(\mathbf{x}; \mathbf{h})\frac{\frac{\partial}{\partial x_k}\frac{1}{h_k}K_k(\frac{x_k}{h_k})}{\frac{1}{h_k}K_k(\frac{x_k}{h_k})} \tag{E.8}$$

Equation E.5 is $O(N)$ which leads to $O(N^2)$ calculations for the estimating the gradient for the entire set of points.

Here is some help with this not-so-nice notation:

- $\mathbf{x}$ = a sample data point.
- x_k = kth dimension of sample data point.
- $\mathbf{x}_i$ = the ith data point.
- $x_{i,k}$ = the kth dimension of the ith data point

E.2 ISE Approximation

We can approximate the $p\log(p)$ term in Equation E.1 with its second order Taylor series expansion about some value q.

$$f(p) \approx f(q) + f^{'}(q)(p-q) + \frac{f^{''}(q)}{2}(p-q)^2 \tag{E.9}$$

$$p\log(p) \approx q\log(q) + (\log(q)+1)(p-q) + \frac{1}{2q}(p-q)^2 \tag{E.10}$$

$$p\log(p) \approx p\log(q) + p - q + \frac{1}{2q}(p-q)^2 \tag{E.11}$$

Substituting in p and q being a function of $\mathbf{x}$ and plugging into the formula for differential entropy we get the following approximation:

$$\hat{h}(X) = -\int p(\mathbf{x})\log(q(\mathbf{x}))d\mathbf{x} - \int p(\mathbf{x})d\mathbf{x} + \int q(\mathbf{x})d\mathbf{x} - \int \frac{1}{2q(\mathbf{x})}(p(\mathbf{x})-q(\mathbf{x}))^2 d\mathbf{x} \tag{E.12}$$

$$\hat{h}(X) = -\int (p(\mathbf{x})\log\left(p(\mathbf{x})\right) - p(\mathbf{x})\log(\frac{p(\mathbf{x})}{q(\mathbf{x})}))d\mathbf{x} - \int \frac{1}{2q(\mathbf{x})}(p(\mathbf{x})-q(\mathbf{x}))^2 d\mathbf{x} \tag{E.13}$$

$$\hat{h}(X) = h(X) + D(p(\mathbf{x})||q(\mathbf{x})) - \int \frac{1}{2q(\mathbf{x})}(p(\mathbf{x})-q(\mathbf{x}))^2 d\mathbf{x} \tag{E.14}$$

We can choose $q(\mathbf{x})$ to be the uniform distribution $u(\mathbf{x}) = V_{\Omega_x}^{-1}$ where V_{Ω_x} is the volume of $\mathbf{x}$'s region of support. This simplifies the above equation with:

$$h(X) + D(p(\mathbf{x})||q(\mathbf{x})) = -\int p(\mathbf{x})\log(u(\mathbf{x}))d\mathbf{x} = -\log(V_{\Omega_x})\int p(\mathbf{x})d\mathbf{x} = \log(V_{\Omega_x}) \tag{E.15}$$

thus giving us

$$\hat{h}(X) = \log(V_{\Omega_x}) - \frac{V_{\Omega_x}}{2}\int \left(\hat{p}(\mathbf{x}) - \frac{1}{V_{\Omega_x}}\right)^2 d\mathbf{x} \tag{E.16}$$

where here we use our kernel density estimate for $p(\mathbf{x})$ since we are assuming we do not have the functional form.

Next we want to find it's gradient. Plugging in our kernel density estimate we have:

$$\frac{\partial}{\partial \mathbf{x}}\hat{h}(X) = -\frac{V_{\Omega_x}}{2}\int \frac{\partial}{\partial \mathbf{x}}\left(\hat{p}(\mathbf{x}) - \frac{1}{V_{\Omega_x}}\right)^2 d\mathbf{x} \tag{E.17}$$

$$\frac{\partial}{\partial \mathbf{x}}\hat{h}(X) = -V_{\Omega_x}\int \left(\hat{p}(\mathbf{x}) - \frac{1}{V_{\Omega_x}}\right)\left(\frac{\partial}{\partial \mathbf{x}}\hat{p}(\mathbf{x})\right) d\mathbf{x} \tag{E.18}$$

To be consistent with John Fisher's thesis let:

$$\epsilon_u(\mathbf{x}) = \frac{1}{V_{\Omega_x}} - \hat{p}(\mathbf{x}) = u(\mathbf{x}) - \hat{p}(\mathbf{x}) \tag{E.19}$$

Combining E.17 with E.19 and E.5 gives us:

$$\frac{\partial}{\partial \mathbf{x}}\hat{h}(X) = -\frac{V_{\Omega_x}}{N}\sum_{i=1}^{N}\int \epsilon_u(\mathbf{x})\mathbf{K}'(\mathbf{x}_i - \mathbf{x}; \mathbf{h})d\mathbf{x} \tag{E.20}$$

(Note the negative is from $\mathbf{K}'()$ being an odd function and the way $\epsilon_u()$ is defined). A different notation can express the gradient as a convolution:

$$\frac{\partial}{\partial \mathbf{x}}\hat{h}(X) = -\frac{V_{\Omega_x}}{N}\sum_{i=1}^{N}\left(\epsilon_u(\mathbf{x}) * \mathbf{K}'(\mathbf{x})\right)\Big|_{\mathbf{x}=\mathbf{x}_i} = -\frac{V_{\Omega_x}}{N}\sum_{i=1}^{N}\mathbf{f}_\epsilon(\mathbf{x}_i) \tag{E.21}$$

we can further expand $\mathbf{f}_\epsilon(\mathbf{x}_i)$:

$$\mathbf{f}_\epsilon(\mathbf{x}_i) = \int \left(u(\mathbf{x}) - \frac{1}{N}\sum_{j=1}^{N}K(\mathbf{x} - \mathbf{x}_j; \mathbf{h})\right)\mathbf{K}'(\mathbf{x}_i - \mathbf{x}; \mathbf{h})d\mathbf{x} \tag{E.22}$$

$$\mathbf{f}_\epsilon(\mathbf{x}_i) = \int u(\mathbf{x})\mathbf{K}'(\mathbf{x}_i - \mathbf{x}; \mathbf{h})d\mathbf{x} - \frac{1}{N}\sum_{j=1}^{N}\int K(\mathbf{x}_j - \mathbf{x}; \mathbf{h})\mathbf{K}'(\mathbf{x}_i - \mathbf{x}; \mathbf{h})d\mathbf{x} \tag{E.23}$$

$$\mathbf{f}_\epsilon(\mathbf{x}_i) = \mathbf{f}_r(\mathbf{x}_i) - \frac{1}{N}\sum_{j=1}^{N}\int K(\mathbf{z}; \mathbf{h})\mathbf{K}'(\mathbf{x}_i - \mathbf{x}_j - \mathbf{z}; \mathbf{h})d\mathbf{z} \tag{E.24}$$

$$\mathbf{f}_\epsilon(\mathbf{x}_i) = \mathbf{f}_r(\mathbf{x}_i) - \frac{1}{N}\sum_{j=1}^{N}\mathbf{f}_a(\mathbf{x}_i - \mathbf{x}_j) \tag{E.25}$$

where there was a substitution of $\mathbf{z}$ for $\mathbf{x}_j - \mathbf{x}$. The two functions $\mathbf{f}_a()$ and $\mathbf{f}_r()$ are simply:

$$\mathbf{f}_r(\mathbf{z}) = u(\mathbf{x}) * \mathbf{K}'(\mathbf{x};\mathbf{h})\Big|_{\mathbf{z}} \tag{E.26}$$

$$\mathbf{f}_a(\mathbf{z}) = K(\mathbf{x};\mathbf{h}) * \mathbf{K}'(\mathbf{x};\mathbf{h})\Big|_{\mathbf{z}} = \left(\frac{\partial}{\partial \mathbf{x}}\big(K(\mathbf{x};\mathbf{h}) * K(\mathbf{x};\mathbf{h})\big)\right)\Big|_{\mathbf{z}} \tag{E.27}$$

Appendix F

Experiment 1 Raw Results

Win Length	15	31	45	63	15	31	45	63	15	31	45	63
Input Dim	5	5	5	5	2	2	2	2	1	1	1	1
Gaussian Model i.i.d												
Avg Pe	0.2552	0.2096	0.1373	0.1620	0.3291	0.2586	0.2107	0.2017	0.4481	0.4189	0.3574	0.3990
Stdv Pe	0.0570	0.1355	0.1156	0.1025	0.0937	0.1350	0.2286	0.1143	0.0572	0.0373	0.1035	0.1126
Avg Bias	-0.01	-0.05	-0.03	-0.02	0.01	0.00	0.00	-0.01	0.02	0.02	0.02	0.00
KDE Model i.i.d												
Avg Pe	0.3194	0.2904	0.3078	0.3110	0.3627	0.3037	0.2639	0.2609	0.5019	0.4444	0.4057	0.4290
Stdv Pe	0.0255	0.0929	0.1363	0.1551	0.1134	0.1351	0.1261	0.1114	0.0417	0.1494	0.1051	0.0884
Avg Bias	0.38	0.00	0.00	0.00	0.13	0.12	0.07	0.05	0.02	0.02	0.02	0.02
Online Gaussian MI (Learned Bias)												
Avg Pe	0.4672	0.3981	0.4111	0.4268	0.3526	0.2755	0.2586	0.2957	0.3340	0.2621	0.2557	0.2962
Stdv Pe	0.0438	0.0307	0.0373	0.0602	0.0308	0.0560	0.0502	0.0794	0.1108	0.1503	0.1828	0.0685
Avg Bias	2.21	0.96	0.67	0.50	0.31	0.20	0.17	0.11	0.08	0.06	0.05	0.03
Online KDE MI (Learned Bias)												
Avg Pe	0.4937	0.4545	0.4500	0.4447	0.3731	0.3327	0.3123	0.3497	0.4000	0.3100	0.2664	0.2883
Stdv Pe	0.0041	0.0264	0.0286	0.0336	0.0247	0.0493	0.0885	0.0533	0.0702	0.1313	0.1301	0.0930
Avg Bias	1.24	1.54	1.53	1.61	0.62	0.56	0.51	0.49	0.24	0.18	0.15	0.14
RBF SVM i.i.d												
Avg Pe	0.3690	0.2750	0.2574	0.2678	0.4237	0.3938	0.3493	0.3377	0.4513	0.3703	0.3362	0.3898
Stdv Pe	0.1009	0.1480	0.1705	0.1821	0.0923	0.0334	0.0501	0.0771	0.0671	0.0304	0.1098	0.1321
Avg Bias	0.05	0.16	0.15	0.03	0.01	0.19	0.13	0.07	0.11	0.15	0.16	0.16
RBF SVM Full												
Avg Pe	0.3326	0.3391	0.4620	0.4692	0.3196	0.3445	0.3484	0.3329	0.3393	0.2770	0.2672	0.3238
Stdv Pe	0.0999	0.0326	0.0308	0.0481	0.0354	0.0932	0.0985	0.1418	0.0655	0.0695	0.0558	0.0699
Avg Bias	0.28	0.52	0.25	0.08	0.06	0.42	0.21	0.18	0.07	0.28	0.11	0.13
Online Gaussian MI (perm test)												
Avg Pe	0.4851	0.4875	0.4926	0.4996	0.3657	0.3817	0.4127	0.4029	0.3444	0.2919	0.3020	0.3641
Stdv Pe	0.0175	0.0125	0.0094	0.0009	0.0311	0.0423	0.0414	0.0564	0.1052	0.1191	0.1415	0.0520
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4552	0.4538	0.4697	0.4935	0.3451	0.3013	0.3307	0.3336	0.3313	0.2614	0.2586	0.2962
Stdv Best Pe	0.0397	0.0177	0.0182	0.0074	0.0312	0.0554	0.0355	0.0742	0.1109	0.1361	0.1637	0.0653
Avg Best Bias	0.93	1.00	1.00	1.00	0.95	1.00	1.00	1.00	0.73	0.95	1.00	1.00
Online KDE MI (perm test)												
Avg Pe	0.4675	0.4514	0.4980	0.4970	0.3910	0.4444	0.4529	0.4713	0.3970	0.3417	0.3619	0.4216
Stdv Pe	0.0293	0.0434	0.0025	0.0050	0.0702	0.0410	0.0296	0.0529	0.0700	0.1085	0.0972	0.0388
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4481	0.4173	0.4877	0.4843	0.3821	0.4060	0.4057	0.4503	0.3746	0.2864	0.2881	0.3436
Stdv Best Pe	0.0466	0.0508	0.0143	0.0098	0.0751	0.0686	0.0415	0.0759	0.0813	0.1444	0.1236	0.0296
Avg Best Bias	0.92	1.00	0.75	1.00	0.93	0.87	1.00	1.00	0.75	0.75	1.00	1.00

Table F.1: Experiment 1: Raw Results for Person 1 : Pixels Intensities and MFCCs

Win Length	15	31	45	63	15	31	45	63	15	31	45	63
Input Dim	5	5	5	5	2	2	2	2	1	1	1	1
Gaussian Model i.i.d												
Avg Pe	0.1716	0.1015	0.1135	0.0614	0.3481	0.2986	0.2488	0.2805	0.3410	0.3484	0.3061	0.2526
Stdv Pe	0.0518	0.0864	0.0601	0.0553	0.0662	0.0355	0.0426	0.0788	0.0564	0.0906	0.1092	0.1468
Avg Bias	-0.02	-0.04	-0.04	-0.06	0.03	0.02	0.03	0.01	0.01	0.03	0.01	0.02
KDE Model i.i.d												
Avg Pe	0.3403	0.2888	0.2750	0.2666	0.4034	0.3425	0.3045	0.3528	0.3963	0.3578	0.3951	0.3558
Stdv Pe	0.0508	0.0800	0.0732	0.0771	0.0630	0.1003	0.1234	0.1619	0.1727	0.2163	0.2918	0.1922
Avg Bias	0.40	0.10	0.00	0.00	0.10	0.07	0.09	0.10	0.03	0.03	0.02	0.02
Online Gaussian MI (Learned Bias)												
Avg Pe	0.4459	0.4757	0.4336	0.4429	0.3668	0.3017	0.3143	0.2901	0.3045	0.1787	0.1418	0.2365
Stdv Pe	0.0471	0.0156	0.0561	0.0368	0.0238	0.0534	0.0351	0.0506	0.0215	0.0492	0.0521	0.1003
Avg Bias	2.31	0.91	0.65	0.46	0.33	0.15	0.13	0.11	0.10	0.08	0.06	0.04
Online KDE MI (Learned Bias)												
Avg Pe	0.4955	0.4545	0.3975	0.3136	0.4295	0.3762	0.3283	0.3371	0.3325	0.2429	0.2389	0.2796
Stdv Pe	0.0044	0.0334	0.0325	0.0323	0.0447	0.0765	0.0800	0.0926	0.0243	0.0264	0.0872	0.0192
Avg Bias	1.33	1.84	1.78	1.92	0.67	0.62	0.59	0.59	0.27	0.21	0.17	0.16
RBF SVM i.i.d												
Avg Pe	0.3497	0.3133	0.3627	0.3103	0.3977	0.4207	0.3064	0.3403	0.3895	0.3496	0.3864	0.3728
Stdv Pe	0.0336	0.0612	0.1691	0.0843	0.0735	0.0653	0.1288	0.1233	0.0894	0.1280	0.1633	0.1099
Avg Bias	0.00	0.12	0.00	0.00	0.08	0.24	0.23	0.19	0.06	0.05	0.05	0.08
RBF SVM Full												
Avg Pe	0.4267	0.3516	0.4792	0.4987	0.3717	0.3941	0.4538	0.4006	0.2478	0.2676	0.2635	0.3186
Stdv Pe	0.0684	0.0812	0.0164	0.0026	0.0658	0.0795	0.0849	0.1174	0.0529	0.0333	0.0427	0.1214
Avg Bias	0.06	0.58	0.26	0.00	0.32	0.28	0.15	0.34	0.35	0.32	0.27	0.08
Online Gaussian MI (perm test)												
Avg Pe	0.4627	0.4976	0.4959	0.4978	0.3903	0.3370	0.4037	0.4264	0.3254	0.2480	0.2414	0.3105
Stdv Pe	0.0313	0.0020	0.0071	0.0026	0.0204	0.0375	0.0396	0.0350	0.0141	0.0324	0.0805	0.0480
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4440	0.4867	0.4799	0.4865	0.3612	0.2931	0.3348	0.3567	0.2985	0.1920	0.1619	0.2487
Stdv Best Pe	0.0420	0.0049	0.0186	0.0140	0.0354	0.0405	0.0332	0.0399	0.0312	0.0607	0.0788	0.0511
Avg Best Bias	0.95	1.00	1.00	0.75	0.98	0.98	1.00	1.00	1.00	1.00	1.00	1.00
Online KDE MI (perm test)												
Avg Pe	0.4112	0.4957	0.4889	0.5000	0.4235	0.4463	0.4578	0.4983	0.3526	0.3084	0.3668	0.4277
Stdv Pe	0.0278	0.0066	0.0162	0.0000	0.0222	0.0333	0.0218	0.0020	0.0323	0.0197	0.0499	0.0218
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.3914	0.4851	0.4701	0.4983	0.3970	0.3938	0.4143	0.4839	0.3340	0.2292	0.2619	0.3436
Stdv Best Pe	0.0246	0.0165	0.0458	0.0014	0.0329	0.0615	0.0171	0.0141	0.0348	0.0370	0.0635	0.0295
Avg Best Bias	0.97	0.75	0.50	0.75	0.98	1.00	1.00	0.75	0.98	1.00	1.00	1.00

Table F.2: Experiment 1: Raw Results for Person 2 : Pixels Intensities and MFCCs

Win Length	15	31	63	15	31	63	15	31	63
Input Dim	5	5	5	2	2	2	1	1	1
Gaussian Model i.i.d									
Avg Pe	0.5449	0.5333	0.5482	0.4832	0.4583	0.4737	0.5127	0.5567	0.5172
Stdv Pe	0.1403	0.0721	0.0594	0.0475	0.0604	0.0910	0.0259	0.0519	0.0508
Avg Bias	0.04	0.00	-0.00	0.02	0.02	0.01	0.00	0.00	0.00
KDE Model i.i.d									
Avg Pe	0.4783	0.4470	0.4721	0.4214	0.4671	0.4893	0.5133	0.4716	0.4847
Stdv Pe	0.0780	0.0707	0.0747	0.0474	0.0463	0.0692	0.0431	0.0547	0.0691
Avg Bias	0.21	0.21	0.00	0.03	0.05	0.04	0.01	0.01	0.00
Online Gaussian MI (Learned Bias)									
Avg Pe	0.4333	0.4349	0.4857	0.3916	0.3782	0.4260	0.4329	0.4381	0.4185
Stdv Pe	0.0239	0.0303	0.0178	0.0504	0.0507	0.0529	0.0526	0.0780	0.0652
Avg Bias	2.23	0.88	0.44	0.33	0.17	0.09	0.09	0.05	0.02
Online KDE MI (Learned Bias)									
Avg Pe	0.4978	0.4620	0.4234	0.4361	0.4118	0.4320	0.4494	0.4522	0.4334
Stdv Pe	0.0016	0.0167	0.0387	0.0301	0.0292	0.0270	0.0380	0.0343	0.0502
Avg Bias	0.79	1.59	1.79	0.68	0.65	0.59	0.25	0.18	0.14
RBF SVM i.i.d									
Avg Pe	0.4969	0.4660	0.4906	0.4782	0.4818	0.4773	0.5096	0.4643	0.4840
Stdv Pe	0.0483	0.0465	0.0558	0.0162	0.0215	0.0265	0.0306	0.0601	0.0277
Avg Bias	0.02	0.14	0.15	-0.03	0.29	0.16	0.29	0.29	0.29
RBF SVM Full									
Avg Pe	0.4884	0.4714	0.4731	0.4521	0.4711	0.4648	0.4684	0.4159	0.4408
Stdv Pe	0.0403	0.0466	0.0649	0.0559	0.0382	0.1025	0.0528	0.0774	0.0473
Avg Bias	0.40	0.32	0.43	0.07	0.16	0.25	0.02	0.02	0.08
Online Gaussian MI (perm test)									
Avg Pe	0.4514	0.4762	0.4998	0.4044	0.4071	0.4556	0.4377	0.4429	0.4287
Stdv Pe	0.0145	0.0121	0.0005	0.0382	0.0362	0.0412	0.0449	0.0694	0.0535
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4276	0.4526	0.4971	0.3881	0.3801	0.4302	0.4342	0.4332	0.4134
Stdv Best Pe	0.0175	0.0203	0.0022	0.0412	0.0510	0.0543	0.0435	0.0736	0.0645
Avg Best Bias	1.00	1.00	0.80	0.99	0.99	0.80	0.87	0.73	0.75
Online KDE MI (perm test)									
Avg Pe	0.4423	0.4446	0.4956	0.4284	0.4569	0.4896	0.4414	0.4486	0.4545
Stdv Pe	0.0213	0.0279	0.0040	0.0464	0.0110	0.0098	0.0363	0.0401	0.0277
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4343	0.4176	0.4842	0.4132	0.4175	0.4756	0.4303	0.4265	0.4318
Stdv Best Pe	0.0282	0.0360	0.0083	0.0418	0.0271	0.0210	0.0384	0.0541	0.0341
Avg Best Bias	0.97	1.00	1.00	0.95	1.00	1.00	0.84	0.95	1.00

Table F.3: Experiment 1: Raw Results for 10 people : Pixels Intensities and MFCCs

Win Length	15	31	45	63	15	31	45	63	15	31	45	63
Input Dim	5	5	5	5	2	2	2	2	1	1	1	1
Gaussian Model i.i.d												
Avg Pe	0.2000	0.0842	0.0758	0.0736	0.2351	0.1179	0.0631	0.0205	0.2433	0.0745	0.0906	0.0675
Stdv Pe	0.0577	0.0631	0.0937	0.1190	0.0311	0.0554	0.0456	0.0314	0.0468	0.0640	0.0686	0.0708
Avg Bias	0.02	0.00	-0.01	-0.01	0.00	-0.00	0.01	0.00	0.00	0.00	0.00	0.00
KDE Model i.i.d												
Avg Pe	0.4336	0.4675	0.4926	0.5000	0.3802	0.3942	0.4184	0.4591	0.2366	0.0960	0.0730	0.0497
Stdv Pe	0.0392	0.0190	0.0085	0.0000	0.0365	0.0260	0.0401	0.0165	0.0727	0.0565	0.0479	0.0554
Avg Bias	0.18	0.00	0.00	0.00	0.11	0.10	0.11	0.02	0.03	0.00	0.01	0.02
Online Gaussian MI (Learned Bias)												
Avg Pe	0.4243	0.2582	0.2098	0.1620	0.3459	0.2578	0.1807	0.1503	0.3478	0.2057	0.1016	0.0771
Stdv Pe	0.0235	0.0959	0.0383	0.0719	0.0329	0.0612	0.0442	0.0311	0.0829	0.0766	0.0771	0.0388
Avg Bias	2.20	0.83	0.55	0.42	0.27	0.14	0.10	0.09	0.06	0.04	0.04	0.03
Online KDE MI (Learned Bias)												
Avg Pe	0.4750	0.4886	0.4201	0.3275	0.4851	0.4181	0.3689	0.3262	0.4351	0.3907	0.3135	0.2827
Stdv Pe	0.0230	0.0123	0.0444	0.0953	0.0111	0.0292	0.0461	0.0751	0.0513	0.0373	0.0645	0.0437
Avg Bias	2.08	2.35	2.29	2.58	0.89	1.18	1.06	1.16	0.33	0.34	0.26	0.26
RBF SVM i.i.d												
Avg Pe	0.2262	0.1977	0.1581	0.1680	0.2827	0.2492	0.1908	0.1623	0.3062	0.1719	0.1883	0.0981
Stdv Pe	0.0703	0.0447	0.0648	0.0156	0.0317	0.0651	0.0377	0.0540	0.0592	0.1032	0.0908	0.0665
Avg Bias	0.20	0.33	0.33	0.19	0.19	0.29	0.27	0.26	0.25	0.26	0.25	0.27
RBF SVM Full												
Avg Pe	0.3769	0.3742	0.4032	0.5000	0.2902	0.2258	0.2810	0.4466	0.2418	0.1512	0.1556	0.1289
Stdv Pe	0.0789	0.0831	0.1242	0.0000	0.0327	0.0456	0.0797	0.0708	0.0256	0.0561	0.0630	0.0639
Avg Bias	0.08	0.14	0.02	0.00	0.29	0.28	0.00	-0.02	0.05	0.36	0.23	0.34
Online Gaussian MI (perm test)												
Avg Pe	0.4496	0.3676	0.3504	0.3624	0.3507	0.2602	0.2357	0.2578	0.3418	0.2245	0.1439	0.1755
Stdv Pe	0.0211	0.0486	0.0418	0.0763	0.0275	0.0511	0.0342	0.0415	0.0662	0.0616	0.0621	0.0875
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4205	0.2908	0.2447	0.2578	0.3381	0.2308	0.1918	0.1760	0.3396	0.2116	0.1221	0.1054
Stdv Best Pe	0.0249	0.0801	0.0414	0.0958	0.0253	0.0516	0.0250	0.0354	0.0676	0.0689	0.0678	0.0530
Avg Best Bias	1.00	0.98	1.00	1.00	0.78	0.97	0.98	1.00	0.87	0.92	0.95	0.97
Online KDE MI (perm test)												
Avg Pe	0.4828	0.3503	0.2844	0.2391	0.4123	0.3041	0.2598	0.1908	0.4287	0.2927	0.1832	0.1742
Stdv Pe	0.0207	0.0185	0.1273	0.1218	0.0284	0.0436	0.0248	0.1769	0.0393	0.0623	0.0585	0.0955
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4612	0.3217	0.2672	0.2204	0.4082	0.2982	0.2529	0.1821	0.4138	0.2868	0.1693	0.1581
Stdv Best Pe	0.0234	0.0262	0.1271	0.1282	0.0296	0.0435	0.0223	0.1778	0.0211	0.0590	0.0539	0.0781
Avg Best Bias	0.68	0.77	0.90	0.90	0.73	0.87	0.83	0.92	0.85	0.92	0.90	0.93

Table F.4: Experiment 1: Raw Results for Person 1 : Image Diffs and MFCC Diffs

Win Length	15	31	45	63	15	31	45	63	15	31	45	63
Input Dim	5	5	5	5	2	2	2	2	1	1	1	1
Gaussian Model i.i.d												
Avg Pe	0.1493	0.1042	0.0295	0.0240	0.1750	0.1042	0.0434	0.0301	0.1985	0.1011	0.0631	0.0444
Stdv Pe	0.0482	0.0227	0.0299	0.0392	0.0538	0.0507	0.0381	0.0400	0.0714	0.0232	0.0120	0.0675
Avg Bias	0.02	-0.01	-0.06	-0.05	0.01	0.00	-0.01	-0.01	0.00	0.00	0.00	-0.00
KDE Model i.i.d												
Avg Pe	0.4313	0.4577	0.4840	0.4939	0.3213	0.3064	0.3074	0.3541	0.2131	0.1328	0.0828	0.0514
Stdv Pe	0.0324	0.0407	0.0229	0.0122	0.0720	0.1190	0.1512	0.1474	0.0571	0.0120	0.0450	0.0382
Avg Bias	0.10	0.00	0.00	0.00	0.03	0.09	0.10	0.00	0.00	0.00	0.05	0.06
Online Gaussian MI (Learned Bias)												
Avg Pe	0.4209	0.2798	0.2488	0.1834	0.3903	0.2347	0.1119	0.0919	0.3157	0.1987	0.0656	0.0192
Stdv Pe	0.0546	0.0437	0.0142	0.0698	0.0588	0.0933	0.0559	0.0616	0.0216	0.0611	0.0606	0.0157
Avg Bias	2.04	0.80	0.53	0.38	0.28	0.16	0.12	0.11	0.08	0.05	0.05	0.05
Online KDE MI (Learned Bias)												
Avg Pe	0.4877	0.4855	0.4660	0.4355	0.4851	0.4788	0.3984	0.3541	0.4220	0.3534	0.3193	0.2831
Stdv Pe	0.0136	0.0178	0.0301	0.0511	0.0144	0.0311	0.0410	0.0654	0.0380	0.0139	0.0743	0.0529
Avg Bias	2.03	2.38	2.35	2.65	0.92	1.15	1.08	1.17	0.32	0.33	0.32	0.35
RBF SVM i.i.d												
Avg Pe	0.2113	0.1543	0.0735	0.0425	0.2448	0.1805	0.1230	0.0729	0.2310	0.1258	0.0915	0.0825
Stdv Pe	0.0608	0.0309	0.0354	0.0340	0.0156	0.0746	0.0889	0.0250	0.0749	0.0483	0.0365	0.0792
Avg Bias	0.20	0.16	0.17	0.14	0.25	0.33	0.29	0.32	0.39	0.35	0.33	0.35
RBF SVM Full												
Avg Pe	0.3456	0.2988	0.4996	0.2930	0.2385	0.2043	0.1462	0.1272	0.1894	0.1762	0.0980	0.0859
Stdv Pe	0.1097	0.1396	0.0008	0.1386	0.0570	0.0846	0.0818	0.0174	0.0654	0.0282	0.0431	0.0610
Avg Bias	0.12	0.16	-0.00	0.15	0.56	0.42	0.04	0.10	0.32	0.31	0.35	0.24
Online Gaussian MI (perm test)												
Avg Pe	0.4340	0.3546	0.3754	0.3558	0.3918	0.2567	0.1803	0.2670	0.3179	0.2422	0.1574	0.1572
Stdv Pe	0.0484	0.0193	0.0428	0.0260	0.0570	0.0787	0.0391	0.1116	0.0040	0.0525	0.0554	0.0335
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4075	0.2931	0.2869	0.2483	0.3847	0.2347	0.1225	0.1455	0.3056	0.1971	0.0980	0.0605
Stdv Best Pe	0.0564	0.0458	0.0196	0.0242	0.0584	0.0685	0.0314	0.0769	0.0148	0.0445	0.0647	0.0301
Avg Best Bias	0.98	1.00	1.00	1.00	0.87	0.98	0.98	1.00	0.87	1.00	1.00	1.00
Online KDE MI (perm test)												
Avg Pe	0.4705	0.3918	0.3332	0.3223	0.4369	0.3895	0.2934	0.2787	0.4007	0.2512	0.2410	0.1442
Stdv Pe	0.0272	0.0911	0.0486	0.0680	0.1080	0.1037	0.0766	0.1213	0.0452	0.0319	0.0315	0.0698
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4571	0.3781	0.3270	0.3062	0.4224	0.3777	0.2816	0.2483	0.3955	0.2457	0.2123	0.1037
Stdv Best Pe	0.0312	0.0813	0.0499	0.0737	0.1049	0.0932	0.0731	0.1178	0.0463	0.0285	0.0422	0.0313
Avg Best Bias	0.42	0.63	0.87	0.72	0.48	0.78	0.90	0.97	0.88	0.87	0.95	0.93

Table F.5: Experiment 1: Raw Results for Person 2 : Image Diffs and MFCC Diffs

Win Length	15	31	63	15	31	63	15	31	63
Input Dim	5	5	5	2	2	2	1	1	1
Gaussian Model i.i.d									
Avg Pe	0.2716	0.2060	0.1448	0.2579	0.2114	0.1289	0.2693	0.2054	0.1322
Stdv Pe	0.0617	0.0689	0.0852	0.0607	0.0756	0.0589	0.0716	0.0453	0.0711
Avg Bias	0.01	0.01	0.00	0.02	0.01	0.00	0.01	0.00	0.00
KDE Model i.i.d									
Avg Pe	0.4745	0.4821	0.5004	0.4065	0.4373	0.4583	0.2695	0.2078	0.1370
Stdv Pe	0.0094	0.0156	0.0016	0.0382	0.0298	0.0236	0.0583	0.0365	0.0593
Avg Bias	0.23	0.09	0.00	0.07	0.08	0.01	0.01	0.00	0.00
Online Gaussian MI (Learned Bias)									
Avg Pe	0.4396	0.3046	0.1996	0.3593	0.2837	0.1964	0.3449	0.2522	0.1616
Stdv Pe	0.0082	0.0533	0.0668	0.0281	0.0401	0.0766	0.0659	0.0837	0.0863
Avg Bias	2.10	0.76	0.36	0.26	0.12	0.07	0.08	0.05	0.03
Online KDE MI (Learned Bias)									
Avg Pe	0.4855	0.4708	0.4501	0.4740	0.4513	0.4432	0.4343	0.4179	0.3955
Stdv Pe	0.0099	0.0269	0.0458	0.0108	0.0289	0.0516	0.0389	0.0500	0.0557
Avg Bias	1.80	2.20	2.45	0.85	1.14	1.13	0.30	0.32	0.29
RBF SVM i.i.d									
Avg Pe	0.3336	0.2737	0.1908	0.3173	0.3194	0.2754	0.2944	0.2569	0.1747
Stdv Pe	0.0667	0.0513	0.0890	0.0779	0.0952	0.0955	0.0589	0.0489	0.0800
Avg Bias	0.24	0.36	0.31	0.18	0.25	0.25	0.18	0.18	0.17
RBF SVM Full									
Avg Pe	0.3954	0.3487	0.3696	0.3324	0.3373	0.2493	0.2606	0.2236	0.1559
Stdv Pe	0.0965	0.0904	0.0941	0.1145	0.1115	0.0702	0.0613	0.0655	0.0812
Avg Bias	0.21	0.03	-0.22	0.09	0.10	-0.01	0.19	0.02	0.12
Online Gaussian MI (perm test)									
Avg Pe	0.4437	0.3490	0.3235	0.3592	0.2931	0.2363	0.3584	0.2692	0.1911
Stdv Pe	0.0061	0.0434	0.0442	0.0308	0.0358	0.0659	0.0614	0.0744	0.0687
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4271	0.3097	0.2252	0.3556	0.2787	0.1848	0.3515	0.2580	0.1610
Stdv Best Pe	0.0120	0.0519	0.0460	0.0292	0.0280	0.0737	0.0600	0.0822	0.0816
Avg Best Bias	0.96	1.00	1.00	0.91	0.93	1.00	0.81	0.91	0.96
Online KDE MI (perm test)									
Avg Pe	0.4572	0.3657	0.2511	0.4019	0.3250	0.2624	0.3938	0.3444	0.2383
Stdv Pe	0.0180	0.0381	0.0745	0.0223	0.0420	0.0708	0.0630	0.0673	0.0423
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4475	0.3573	0.2449	0.3966	0.3211	0.2556	0.3905	0.3388	0.2186
Stdv Best Pe	0.0169	0.0374	0.0679	0.0208	0.0435	0.0678	0.0572	0.0631	0.0438
Avg Best Bias	0.69	0.76	0.85	0.76	0.85	0.89	0.87	0.92	0.91

Table F.6: Experiment 1: Raw Results for 10 People : Image Diffs and MFCC Diffs

Win Length	15	31	45	63	15	31	45	63	15	31	45	63
Input Dim	5	5	5	5	2	2	2	2	1	1	1	1
Gaussian Model i.i.d												
Avg Pe	0.2075	0.1540	0.0746	0.0109	0.3067	0.2453	0.2455	0.1328	0.4410	0.4397	0.3840	0.4055
Stdv Pe	0.0427	0.0523	0.0277	0.0195	0.0658	0.0953	0.0680	0.0997	0.0484	0.0785	0.1627	0.0810
Avg Bias	0.00	-0.01	-0.00	-0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
KDE Model i.i.d												
Avg Pe	0.4090	0.4326	0.4770	0.4970	0.3634	0.3495	0.3926	0.4438	0.3951	0.3440	0.3496	0.4125
Stdv Pe	0.0212	0.0258	0.0233	0.0061	0.0394	0.0676	0.0611	0.0606	0.0255	0.0853	0.0854	0.1103
Avg Bias	0.25	0.11	0.00	0.00	0.05	0.10	0.00	0.04	0.01	0.00	0.02	0.00
Online Gaussian MI (Learned Bias)												
Avg Pe	0.4519	0.3053	0.2574	0.1145	0.3698	0.2806	0.2082	0.1394	0.4884	0.4373	0.3668	0.3998
Stdv Pe	0.0384	0.0202	0.1177	0.0649	0.0627	0.0658	0.0526	0.1102	0.0162	0.0369	0.0293	0.0754
Avg Bias	2.02	0.70	0.46	0.33	0.20	0.13	0.08	0.06	0.07	0.02	0.02	0.02
Online KDE MI (Learned Bias)												
Avg Pe	0.4903	0.4655	0.4266	0.3859	0.4306	0.3926	0.3414	0.3110	0.4425	0.3582	0.3070	0.3001
Stdv Pe	0.0128	0.0354	0.0288	0.0466	0.0089	0.0627	0.0737	0.0576	0.0375	0.0645	0.0539	0.0536
Avg Bias	1.91	2.01	1.93	2.09	0.84	0.90	0.83	0.90	0.37	0.33	0.26	0.24
RBF SVM i.i.d												
Avg Pe	0.2173	0.1941	0.1254	0.0629	0.2846	0.2629	0.2014	0.1571	0.4178	0.3883	0.3476	0.3776
Stdv Pe	0.0500	0.0138	0.0189	0.0347	0.0754	0.1084	0.0821	0.1190	0.0642	0.1271	0.1562	0.1652
Avg Bias	0.20	0.30	0.27	0.26	0.23	0.28	0.29	0.31	0.28	0.35	0.33	0.33
RBF SVM Full												
Avg Pe	0.5000	0.4121	0.4203	0.4336	0.4546	0.4207	0.4028	0.4848	0.4851	0.4742	0.4653	0.4939
Stdv Pe	0.0000	0.1111	0.0721	0.1328	0.0662	0.0889	0.1056	0.1450	0.0255	0.0285	0.0240	0.0145
Avg Bias	0.03	0.00	0.03	0.00	0.00	0.01	-0.14	-0.00	0.13	0.06	0.14	-0.00
Online Gaussian MI (perm test)												
Avg Pe	0.4526	0.3350	0.2984	0.1777	0.3638	0.2908	0.2115	0.1581	0.4799	0.4165	0.3869	0.3776
Stdv Pe	0.0456	0.0095	0.1239	0.0844	0.0518	0.0495	0.0536	0.0430	0.0194	0.0422	0.0347	0.0741
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4299	0.3147	0.2553	0.1328	0.3545	0.2810	0.2004	0.1233	0.4444	0.3887	0.3643	0.3406
Stdv Best Pe	0.0524	0.0182	0.0994	0.0445	0.0513	0.0527	0.0597	0.0358	0.0417	0.0429	0.0225	0.0824
Avg Best Bias	0.80	0.97	0.95	0.97	0.82	0.92	0.90	0.93	0.35	0.77	0.80	0.78
Online KDE MI (perm test)												
Avg Pe	0.4612	0.4028	0.2648	0.1960	0.4104	0.3111	0.2299	0.1773	0.4511	0.3139	0.2713	0.3001
Stdv Pe	0.0454	0.0503	0.0480	0.0945	0.0288	0.0678	0.0531	0.1049	0.0141	0.0260	0.0369	0.0701
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4571	0.3868	0.2475	0.1842	0.3948	0.2951	0.2299	0.1616	0.4276	0.2955	0.2643	0.2696
Stdv Best Pe	0.0457	0.0382	0.0472	0.0914	0.0300	0.0599	0.0531	0.0837	0.0098	0.0352	0.0366	0.0522
Avg Best Bias	0.45	0.78	0.87	0.88	0.68	0.77	0.85	0.80	0.78	0.77	0.90	0.97

Table F.7: Experiment 1: Raw Results for Person 1 : Flow and MFCC Diffs

Win Length	15	31	45	63	15	31	45	63	15	31	45	63
Input Dim	5	5	5	5	2	2	2	2	1	1	1	1
Gaussian Model i.i.d												
Avg Pe	0.1750	0.0854	0.0168	0.0122	0.2433	0.2574	0.1865	0.2230	0.3974	0.4255	0.3955	0.3406
Stdv Pe	0.0475	0.0250	0.0315	0.0232	0.1173	0.2576	0.2673	0.2343	0.0560	0.1224	0.1370	0.0200
Avg Bias	0.00	-0.02	-0.01	-0.02	0.00	0.01	0.00	0.02	-0.00	0.01	-0.00	0.00
KDE Model i.i.d												
Avg Pe	0.4000	0.4228	0.4504	0.4882	0.3451	0.3648	0.4152	0.4085	0.4164	0.4737	0.4799	0.4708
Stdv Pe	0.0227	0.0528	0.0429	0.0161	0.0480	0.0372	0.0626	0.0365	0.0427	0.0419	0.1092	0.0699
Avg Bias	0.05	0.00	0.00	0.00	0.03	0.05	0.12	0.02	0.01	0.01	0.01	0.01
Online Gaussian MI (Learned Bias)												
Avg Pe	0.3948	0.2574	0.1443	0.1672	0.3825	0.3045	0.2090	0.1398	0.4701	0.4894	0.4947	0.4660
Stdv Pe	0.0491	0.0651	0.0373	0.0984	0.0635	0.1226	0.1377	0.1566	0.0444	0.0212	0.0107	0.0580
Avg Bias	1.87	0.73	0.48	0.36	0.22	0.11	0.09	0.07	0.06	0.03	0.02	0.01
Online KDE MI (Learned Bias)												
Avg Pe	0.4821	0.4557	0.3480	0.3262	0.4459	0.4275	0.3869	0.3510	0.4731	0.4071	0.4230	0.4242
Stdv Pe	0.0201	0.0257	0.0512	0.0409	0.0300	0.0485	0.0843	0.0880	0.0312	0.0652	0.0339	0.0672
Avg Bias	1.75	2.04	2.06	2.20	0.90	0.90	0.85	0.92	0.38	0.33	0.26	0.26
RBF SVM i.i.d												
Avg Pe	0.2113	0.1437	0.0596	0.0282	0.3356	0.3012	0.2586	0.2326	0.4282	0.4270	0.4518	0.4358
Stdv Pe	0.0458	0.0829	0.0412	0.0323	0.1431	0.2101	0.2062	0.2452	0.0246	0.0562	0.0536	0.1296
Avg Bias	0.30	0.35	0.32	0.27	0.17	0.17	0.17	0.18	0.44	0.49	0.50	0.41
RBF SVM Full												
Avg Pe	0.3672	0.5000	0.4342	0.4470	0.3389	0.2961	0.3819	0.4570	0.4799	0.4547	0.4947	0.4783
Stdv Pe	0.1552	0.0000	0.1315	0.1059	0.1570	0.2076	0.2040	0.0689	0.0104	0.0262	0.0220	0.0302
Avg Bias	0.12	0.02	-0.02	0.00	0.06	0.15	0.09	-0.03	0.03	0.04	0.01	-0.00
Online Gaussian MI (perm test)												
Avg Pe	0.3993	0.2817	0.2127	0.2753	0.4000	0.3056	0.2381	0.1860	0.4821	0.4894	0.4816	0.4678
Stdv Pe	0.0489	0.0763	0.0132	0.0473	0.0519	0.1141	0.1101	0.1306	0.0190	0.0172	0.0369	0.0409
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.3858	0.2571	0.1500	0.1751	0.3806	0.2962	0.2135	0.1646	0.4601	0.4753	0.4561	0.4307
Stdv Best Pe	0.0549	0.0661	0.0440	0.0593	0.0513	0.1062	0.1233	0.1466	0.0405	0.0153	0.0558	0.0650
Avg Best Bias	0.85	0.95	1.00	1.00	0.77	0.82	0.88	0.92	0.47	0.67	0.23	0.52
Online KDE MI (perm test)												
Avg Pe	0.4459	0.4314	0.2500	0.2186	0.4433	0.3703	0.3398	0.2465	0.4522	0.4079	0.4135	0.3645
Stdv Pe	0.0427	0.0701	0.0418	0.0993	0.0670	0.1474	0.1141	0.0975	0.0148	0.0838	0.0730	0.0560
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4287	0.3981	0.2348	0.1995	0.4299	0.3103	0.3324	0.2287	0.4306	0.3844	0.3709	0.3567
Stdv Best Pe	0.0415	0.0577	0.0501	0.1026	0.0625	0.1092	0.1125	0.0860	0.0245	0.0865	0.0807	0.0675
Avg Best Bias	0.68	0.65	0.77	0.83	0.57	0.65	0.90	0.72	0.68	0.53	0.70	0.82

Table F.8: Experiment 1: Raw Results for Person 2 : Flow and MFCC Diffs

Win Length	15	31	63	15	31	63	15	31	63
Input Dim	5	5	5	2	2	2	1	1	1
Gaussian Model i.i.d									
Avg Pe	0.2517	0.1864	0.0982	0.4237	0.4025	0.3328	0.4226	0.4121	0.3617
Stdv Pe	0.0698	0.0796	0.0543	0.0233	0.0436	0.0539	0.0445	0.0535	0.1131
Avg Bias	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
KDE Model i.i.d									
Avg Pe	0.4345	0.4676	0.4822	0.4323	0.4560	0.4746	0.4257	0.4322	0.4413
Stdv Pe	0.0391	0.0165	0.0303	0.0203	0.0462	0.0421	0.0352	0.0450	0.0813
Avg Bias	0.22	0.28	0.11	0.00	0.00	0.03	0.00	0.00	0.00
Online Gaussian MI (Learned Bias)									
Avg Pe	0.4418	0.3351	0.2742	0.4379	0.3981	0.3939	0.4761	0.4318	0.4138
Stdv Pe	0.0184	0.0448	0.0757	0.0272	0.0296	0.0735	0.0351	0.0542	0.0794
Avg Bias	1.91	0.70	0.32	0.20	0.08	0.05	0.08	0.03	0.01
Online KDE MI (Learned Bias)									
Avg Pe	0.4914	0.4738	0.4407	0.4786	0.4399	0.4003	0.4678	0.4528	0.4620
Stdv Pe	0.0083	0.0241	0.0608	0.0119	0.0274	0.0442	0.0195	0.0433	0.0385
Avg Bias	1.82	2.02	2.11	0.96	0.91	0.85	0.38	0.33	0.28
RBF SVM i.i.d									
Avg Pe	0.2955	0.2514	0.1958	0.4355	0.4228	0.4011	0.4834	0.4773	0.4670
Stdv Pe	0.0653	0.0823	0.0690	0.0703	0.0832	0.0969	0.0483	0.0724	0.0301
Avg Bias	0.17	0.22	0.21	0.13	0.12	0.10	-0.17	-0.17	-0.17
RBF SVM Full									
Avg Pe	0.4018	0.4041	0.3688	0.4941	0.4870	0.5002	0.4747	0.4914	0.4821
Stdv Pe	0.0944	0.1042	0.0943	0.0134	0.0124	0.0094	0.0329	0.0245	0.0298
Avg Bias	0.06	-0.29	0.05	-0.16	0.27	-0.02	-0.11	-0.10	-0.13
Avg Best Bias	0.07	-0.14	0.00	-0.14	0.37	-0.04	-0.19	-0.19	-0.20
Online Gaussian MI (perm test)									
Avg Pe	0.4386	0.3514	0.3065	0.4457	0.4131	0.4021	0.4806	0.4414	0.4324
Stdv Pe	0.0162	0.0437	0.0577	0.0187	0.0318	0.0686	0.0309	0.0406	0.0714
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4316	0.3379	0.2710	0.4359	0.4022	0.3756	0.4758	0.4313	0.4072
Stdv Best Pe	0.0114	0.0304	0.0639	0.0218	0.0300	0.0726	0.0368	0.0508	0.0821
Avg Best Bias	0.93	0.93	1.00	0.75	0.67	0.83	0.48	0.81	0.57
Online KDE MI (perm test)									
Avg Pe	0.4632	0.3851	0.3375	0.4304	0.3671	0.2655	0.4457	0.4039	0.3570
Stdv Pe	0.0245	0.0251	0.0924	0.0297	0.0460	0.1261	0.0258	0.0523	0.0820
Avg Bias	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
Avg Best Pe	0.4509	0.3800	0.3177	0.4135	0.3545	0.2565	0.4322	0.4002	0.3485
Stdv Best Pe	0.0241	0.0260	0.0904	0.0241	0.0431	0.1163	0.0245	0.0501	0.0766
Avg Best Bias	0.80	0.80	0.93	0.71	0.71	0.75	0.64	0.72	0.71

Table F.9: Experiment 1: Raw Results for 10 People : Flow and MFCC Diffs

Bibliography

[1] T.D. Bie and B.D. Moor. On the regularization of canonical correlation analysis. In *ICA*, 2003.

[2] M. Borga, T. Landelius, and Knutsson L. A unified approach to pca, pls, mlr and cca. In *LiTH-ISY-R*, 1992.

[3] P. Bretelson, G. Vroomen, J. Wiegeraad, and B. de Gelder. Exploring the relation between the mcgurk interference and ventriloquism. In *International Conference on Spoken Language Processing*, volume 2, pages 559–562, 1994.

[4] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *Lecture Notes in Computer Science*, 1407:484–??, 1998.

[5] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* John Wiley and Sons, Inc., 1991.

[6] R. Duda, P. Hart, and D. Stork. *Patter Classification.* John Wiley & Sons, Inc., 2000.

[7] T.J. Hazen, K. Saenko, C. La, and J.R. Glass. A segment-based audi-visual speech recognizer: Data collection, development, and initial experiments. In *ICMI*, 2004.

[8] J. Hershey and J. Movellan. Audio-vision: Using audio-visual synchrony to locate sounds. In *Neural Information Processing Systems*, pages 813–819, 1999".

[9] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(321-377), 1936.

[10] A. Ihler. Kernel density estimation class for matlab. Web, 04.

[11] A. Ihler, J. Fisher, and A. Wilsky. Nonparameteric hypothesis tests for statistical dependency. In *Trans. on signal processing, special issue on machine learning*, 2004.

[12] J.W. Fisher III, T.Darrell, W.T. Freeman, and P.A. Viola. Learning joint statistical models for audio-visual fusion and segregation. In *NIPS*, pages 772–778, 2000".

[13] Intel. Open source computer vision libaray. Web.

[14] G. Iyengar and C. Neti. A vision-based microphone switch for speech intent detection. In *IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, 2001".

[15] D.H. Johnson and D.E. Dudgeon. *Array Signal Processing: Concepts and Techniques*. Prentice Hall, 1993".

[16] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

[17] I. Mathews, T. Cootes, S. Cox, R. Harvey, and J.A. Bangham. Lipreading using shape, shading and scale. In *Workshop on Audio Visual Speech Processing*, pages 73–78, Terrigal, Australia, 1998.

[18] I. Mathews, G. Potamianos, C. Neti, and J. Luettin. A comparison of model and transform-based visual features for audio-visual lvcsr. In *International Conference on Multimedia and Expo*, Tokyo, Japan, 2001.

[19] H. McGurk and J. MacDonald. Hearing lips and seeing voices. *Nature*, 264, December 1976.

[20] M.Slaney and M.Covell. Facesync: A linear operator for measuring synchronization of video facial images and audio tracks. In *Neural Information Processing Systems*, 2000".

[21] C. Neti, G. Potamianos, J. Luettin, H. Glotin, D. Vergyri, A. Sison, J. Mashari, and J. Zhou. Audio-visual speech recognition. In *Final Workshop*, Baltimore, MD, 2000. Center for Language and Speech Processing, The Johns Hopkins University.

[22] H.J. Nock, G. Iyengar, and Neti C. Speaker localisation using audio-visual synchrony: An empirical study. In *CIVR*, 2003.

[23] H.J. Nock, G. Iyengar, and C. Netic. Assessing face and speech consistency for monologue detection in video. In *ACM Multimedia*, 2002.

[24] V. Pavlovic, A. Garg, J. Rehg, and T. Huang. Multimodal speaker detection using error feedback dynamic bayesian networks. In *Computer Vision and Pattern Recognition*, 2000.

[25] G. Pingali, G. Tunali, and I. Carlbom. Audio-visual tracking for natural interactivity. In *Proceedings of the seventh ACM international conference on Multimedia*, pages 373–382, 1999.

[26] J. M. Rehg, K.P. Murphy, and P. W. Fieguth. Vison-based speaker detection using bayesian networks. In *Computer Vision and Pattern Recognition*, 1999".

[27] P. Scanlon and R. Reilly. Feature analysis for automatic speechreading. In *Workshop on Multimedia Signal Processing*, pages 625–630, Cannes, France, 2001.

[28] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994.

[29] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, 1998.

[30] E. Simoncelli. Matlabpyrtools : Matalb source code for multi-scale image processing. Web, 2004.

[31] Q. Summerfield, A. MacLeod, M. McGrath, and M. Brooke. Lips, teeth, and the benefits of lipreading. *Handbook of Research on Face Processing.*, pages 223–233, 1989. Elsevier Science Publishers.

[32] Vladimir N. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, Inc., 1998.

[33] H.D. Vinod. Canonical ridge and econometrics of joint proudction. In *J. Econometrics*, 1976.

[34] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. Conf. on Computer Vision and Pattern Recognition*, 2001".

[35] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. In *NIPS*, pages 668–674, 2000.