Generative Models for Similarity-based Classification

Luca Cazzanti

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

Doctor of Philosophy

University of Washington

2007

Program Authorized to Offer Degree: Electrical Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Luca Cazzanti

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of the Supervisory Committee:

_____

Maya Gupta

Reading Committee:

_____

Les Atlas

_____

Warren Fox

_____

Maya Gupta

Date: _____

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature_____

Date_____

University of Washington

Abstract

Generative Models for Similarity-based Classification

Luca Cazzanti

Chair of the Supervisory Committee:
Assistant Professor Maya Gupta
Dept. Electrical Engineering

This work proposes a generative framework for similarity-based classification: similarity discriminant analysis (SDA). The classifiers in the SDA framework are similarity-based, because they classify based on the pairwise similarities of samples, and they are generative, because they build class-conditional probability models of the pairwise similarities. The problem of estimating the class-conditional similarity probability models is solved by applying the maximum entropy principle, under the constraint that the mean similarities be equal to the average similarities observed in a set of training samples. Thus, the class-conditional distributions in the SDA framework are exponential functions of the similarities. Within the SDA framework, several classifiers are analyzed in detail: the SDA classifier, the local SDA classifier, the nnSDA classifier, and the mixture SDA classifier. Their performance is evaluated on simulated and benchmark data sets, and compared to the performance of existing similarity-based classifiers which are not generative.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I could not have completed this work without the help of the following people and organizations. I thank them all deeply.

This work was supported in part by the United States Office of Naval Research, Code 321, Grant # N00014-05-1-0843 and by the 2004 AFCEA Ph. D. Fellowship. Additional support was provided by the University of Washington through the employee tuition waiver program and by the Applied Physics Laboratory through discretionary funding.

Thanks to my advisor, Prof. Maya Gupta. I was honored that she picked me as one of her first graduate students. Unfortunately for her, she did not know what she was getting into! Thanks Maya, for sticking by me in spite of my hard-headedness. This work is exponentially better because of your help. I like to think that after all, you are proud of this *testa dura*.

Thanks to the Supervisory Committee: Prof. Les Atlas, Dr. Warren Fox, and Prof. Bill Noble. Thanks to Prof. John Miyamoto, Dr. Santosh Srivastava, Dr. Megan Hazen, Yihua Chen, Sergey Feldman, and all the people in the Information Design Lab for their helpful feedback.

Thanks to the folks at the Applied Physics Laboratory at the University of Washing-

ton for providing a flexible and supportive environment that nurtured my professional goals. Dr. Warren Fox and Dr. Jim Pitton offered enthusiastic encouragement and invaluable perspective throughout my graduate studies. Prof. Bob Odom and Ms. Dian Gay arranged for additional funding for my graduate studies, and Director Jeff Simmen provided lively *chiacchierate* over espresso.

Thanks to my parents Mario and Maria. Their unwavering encouragement and support gave me strength when I needed it. I dedicate this dissertation to them. *Grazie! Vi voglio tanto bene.*

Chapter 1
INTRODUCTION

The contribution of this dissertation is a generative framework for similarity-based classification. The two defining characteristics of the proposed classification framework are *similarity-based* and *generative*. The classifiers in this framework are similarity-based, because they classify based on the pairwise similarities of data samples, and they are generative, because they build class-dependent probability models of the similarities between samples. Similarity-based classifiers already exist; classifiers based on generative models already exist. This work proposes a new family of classifiers that are *both* similarity-based and generative. The new family of classifiers forms a framework for classification termed *similarity discriminant analysis* (SDA).

Metric-space classifiers are the most common classification methods. The samples are represented as vectors of numerical features in Euclidean space and the metric-space classifiers act upon the numerical vectors. Examples of metric-space classifiers are linear and quadratic discriminant analysis (LDA and QDA), Gaussian mixture models (GMMs), support vector machines (SVMs), and k-nearest neighbors(k-NN) [27, 74]. The implicit assumption of metric-space classifiers is that the pairwise similarity between the samples is represented by a metric distance function. For example, the distance between two samples in Euclidean space measures their dissimilarity. However, in some cases metric distance functions fail to capture the full complexity of the similarity (or dissimilarity) relationships between the samples [19, 71–73]. Similarity is more general than distance!

The need to classify samples in similarity space has spurred the development of

similarity-based classifiers. SVMs have been adapted to work in similarity space [30]; k-NN can also be employed in similarity space. However, none of the current similarity-based classifiers are generative. There is an unmet need for generative classifiers that work in similarity space. Generative classifiers such as the metric-learning LDA, QDA, and GMMS, base their results on probabilistic models of the classes, and offer several advantages in terms of interpretability and flexibility. They produce probabilities, which are easily interpretable and can be flexibly incorporated into larger classification systems comprising different types of classifiers. They are natively multiclass and easily take into account classification costs and class prior probabilities. This work proposes classifiers that operate in similarity space *and* retain the powerful flexibility and interpretability properties of metric-space generative classifiers.

The problem of classifying samples based only on their pairwise similarities may be divided into two sub-problems: measuring the similarity between samples and classifying the samples based on their pairwise similarities. **Chapter 2** reviews current approaches to these two components of similarity-based classification. Section 2.1 reviews ways to measure the similarity between samples that are not easily described by numerical vectors in a d-dimensional space, the standard sample description in metric learning. Section 2.2 reviews some existing, non-generative similarity-based classifiers which will be compared to the proposed classifiers in later chapters.

Within the general SDA framework, this dissertation details several classifiers. The *SDA classifier* is at the foundation of SDA. It classifies based on the class-conditional generative models of the similarity of the samples to representative class prototypes, or *centroids*. The SDA framework is introduced, developed, and discussed with the aid of this centroid-based SDA classifier. Then, the centroid-based SDA classifier is generalized beyond class centroids to arbitrary descriptive statistics. One such non-centroidal statistic is the *nearest neighbor similarity*, which gives rise to the *nearest neighbor SDA classifier*, (*nnSDA*). Other possible statistics are described, illustrating

the power and generality of the SDA framework. **Chapter 3** introduces the core of the SDA framework and the SDA and nnSDA classifiers.

The *local SDA classifier* is a local version of the SDA classifier. It builds similarity-based class-conditional generative models within a neighborhood of a test sample to be classified. The local class models are endowed with low bias and retain the powerful quality of interpretability associated with generative probability models. **Chapter 4** introduces local SDA and shows that it is a consistent classifier, in the sense that its error rate converges to the Bayes error rate, which is the best possible error rate attainable by a classifier.

The *mixture SDA classifier* draws from the well-established metric learning mixture model research. It generalizes the single-centroid SDA classifier to a mixture of single-centroid SDA components. The mixture SDA classifier can be trained with an expectation-maximization (EM) algorithm which parallels the standard EM approach for the well-known Gaussian mixture models. **Chapter 5** introduces the mixture SDA classifier, discusses its metric learning analog, and details the EM methods for mixture SDA algorithm.

In **Chapter 6**, the performance of the SDA, local SDA, mixture SDA, and nnSDA classifiers is compared to that of other classifiers in a series of computer experiments. The nearest centroid (NC), local nearest centroid (local NC), k-nearest neighbors (kNN), and condensed nearest neighbors (CNN) are all similarity-based classifiers which are not generative. They help assess the advantages and disadvantages of using generative models in similarity space. A general support vector machine, called PSVM, is also used for comparison. The PSVM is the state-of-the-art in similarity-based classification and provides a good benchmark for assessing the performance of the proposed classifiers. The data sets used in the experiments comprise simulated and real-world data from benchmark databases and previous work published by other authors.

Chapter 7 addresses future directions in similarity-based learning.

Chapter 2

APPROACHES TO SIMILARITY AND SIMILARITY-BASED CLASSIFICATION

The problem of classifying samples based only on their pairwise similarities may be divided into two sub-problems: measuring the similarity between samples and classifying the samples based on their pairwise similarities. This chapter reviews current approaches to these two components of similarity-based classification. Section 2.1 reviews ways to measure the similarity between samples which are not easily described by numerical vectors in a d-dimensional space, the standard sample description in metric learning. However, the focus of this dissertation is on classifying based on given similarities; it is not on measuring the similarity. Thus, Section 2.2 reviews existing similarity-based classifiers which will be compared to this dissertation's contributions in later chapters.

2.1 Measuring Similarity

In standard metric learning, samples are represented by d-dimensional numerical vectors in a Euclidean space. Each element of a vector quantifies a particular feature of the sample. For example, fish width and lightness of fish scales might be two features used in an automatic fish-packing system that groups similar fish together [16]. In this example, the natural way to describe the similarity between the fish is to compute the Euclidean distance, or dissimilarity, between their feature vector representations. More generally, the feature vectors may be thought of as embedded in a linear d-dimensional space endowed with a norm which measures their dissimilarity. Standard metric learning classifiers rely on this numerical feature vector representation to de-

scribe the samples and on the concomitant distance metric to quantify their pairwise similarities.

For example, quadratic discriminant analysis (QDA) classifies feature vectors based on Gaussian models of their class-conditional distributions [16]. Local classifiers such as the k-nearest neighbors (k-NN) classifier rely on Euclidean distance to define a neighborhood of most-similar (nearest) samples to a test sample [27]. Standard support vector machines (SVMs) define similarity between samples by way of kernel functions which rely on the distance between the vector representations of the samples [27].

Judging similarity between samples characterized by many disparate data types poses challenges of data representation and quantitative comparison. For example, modern databases store information from disparate data sources in different formats: multimedia databases store audio, video and text data; proteomics databases store information on proteins, genetic sequences, and related annotations; internet traffic databases store mouse click histories, user profiles, and marketing rules; homeland security databases may store data on individuals and organizations, annotations from intelligence reports, and maritime shipping records. These database objects, or samples, are described by both numerical and non-numerical data. For example, a security database might store cell phone records in textual form and voice parameters for speaker recognition in numerical form. Representing all these different data types with continuous-valued numbers in a geometric feature space is not appropriate. Thus, current metric space classifiers which rely on metric similarity functions may not be applicable.

Furthermore, in some applications, only the pairwise similarities may be observed, and the underlying features may be inaccessible. For example, Chapter 6 discusses a dataset of sonar echoes for which the pairwise similarities are judged by human listeners. For this dataset, the putative perceptual features from which the human similarity ratings are generated are unknown – indeed eliciting the features remains an

ongoing research problem [55] – but the similarity ratings are nonetheless successfully used for classification. Another dataset discussed in Chapter 6 for which the pairwise similarities, but not the features, are available is the protein benchmark dataset used in [30]. In many applications, the similarity relationship between samples may lack the metric properties usually associated with distance (minimality, symmetry, triangle inequality); thus, using a metric function to express the pairwise similarities is suboptimal. Similarities are more general than distances and require more general functions than metrics [71].

The need to fuse disparate data types and to generalize metrics to similarities has spurred research in several general classes of similarity functions. One general class of non-metric similarity functions for pattern recognition is the set-theoretic *linear contrast model* of similarity [19, 71, 72], due to cognitive psychologist Amos Tversky. Tversky's model assumes that each sample can be represented as a set of features, for example the samples shown in Fig. 2.1 can each be represented as a set of up to four features: eyes, nose, mouth, and hair. Then, the set-theoretic similarity between two samples x and z is calculated as some function of the intersection between the two samples' feature sets, and some function of the set-exclusions of the two samples' feature sets [71, 72]:

$$s^l_{Tversky}(x, z) = f(x \cap z) - \alpha f(x \setminus z) - \beta f(z \setminus x), \tag{2.1}$$

where f is a positive *saliency* function that is monotonically increasing with respect to set inclusion, and α and β are fixed positive real numbers. Thus x and z are more similar if their intersection increases, but less similar depending on which features belong exclusively to x or exclusively to z. Tversky also proposed a *ratio* version of the contrast model where similarity is defined

$$s^r_{Tversky}(x, z) = \frac{f(x \cap z)}{f(x \cap z) + \alpha f(x \setminus z) + \beta f(z \setminus x)}. \tag{2.2}$$

Tversky's set-theoretic similarity models have been successful at explaining human similarity judgements in various similarity-assessment tasks, particularly when

the objects are not described well by low-level numerical features, and when the assessment of similarity involves considerable cognitive effort (e.g. similarity of countries or professions) rather than perceptual discrimination (e.g. similarity of audio and visual stimuli) [19, 61, 73].

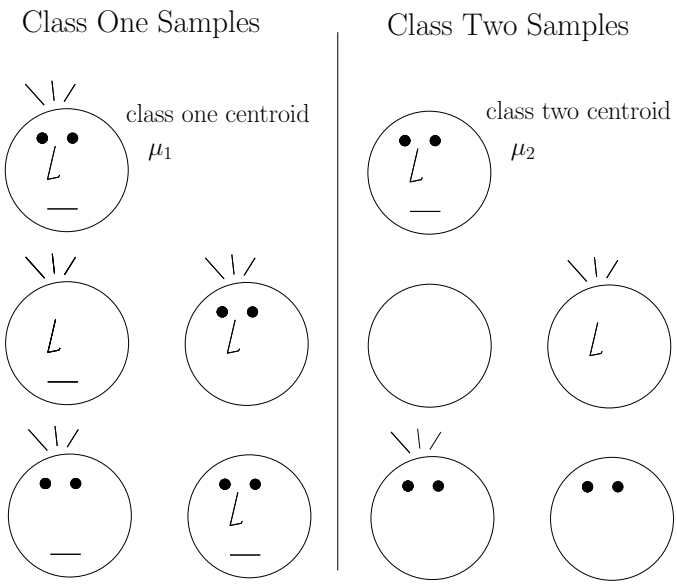

Figure 2.1: Five samples are drawn for class one, and five samples are drawn for class two (note the classes overlap). Each sample can be described as a set of up to four facial features: eyes, nose, mouth, and hair. One sample from each class is labeled as a "centroid" of the class, as per the definition given in (2.10), where the similarity between any two samples is defined to be the number of facial features the two samples have in common.

Tversky's models of similarity are flexible. First, they are not tied to, and indeed overcome the limits of, a vector representation of samples in Euclidean space and the assumption that high similarity and close proximity are inextricably bound to each other. In fact, Tversky and Hutchinson [73] have shown that multidimensional scaling solutions to similarity problems often do not capture faithfully the underlying similarity relationships of data sets. Second, set-theoretic models can explain how context affects the similarity of objects [72]. Finally, Tversky's models allow for

and explain asymmetry in similarity judgments. Metric distances by definition are symmetric, so that if a metric is taken to signify similarity, then it must be that the similarity of x to z is identical to the similarity of z to x. Pseudo-metrics relax this requirement and provide asymmetric measures. Tversky's models allow, but do not mandate, asymmetry, and are thus more flexible.

Tversky's contrast models have been applied outside of psychology. One example is the use of Tversky's ratio contrast model for searching databases that store chemical structure information in binary vectors [76]. In fact, it can be easily shown that Tversky's contrast models generalize the Hamming, Jaccard, and other distance measures commonly used to assess similarity between binary vectors [76, 79]. One simply describes sets as binary vectors in which a 1 (or 0) indicates the presence (or absence) of a feature. The saliency function f is the sum of the elements of a binary vector, or, more generally, the cardinality of a set. For example, the similarity between any two samples in Fig. 2.1 could be calculated with the linear contrast model by setting $\alpha = \beta = 0$ and using set cardinality for f. This choice reduces to measuring similarity using the number of features which the two samples have in common. This simple function is called the *counting*, or Hamming, similarity.

Another general class of similarity functions is designed to measure similarities between objects described by nominal (non-numeric) and unordered features. The Value Difference Metric (VDM) [68] is a dissimilarity metric designed to measure the distance between samples described by non-numeric features; it has been shown very effective in classification problems involving symbolic features and nearest-neighbor classifiers. Given a sample x described by symbolic features – for example, all the unique nouns, verbs, and adjectives in a document – the value difference $\triangle_i$ for the ith feature is a table of pairwise differences of class probabilities conditioned on the features,

$$\triangle_i(x[i] = a, x[i] = b) = |\hat{P}(Y = g|x[i] = a) - \hat{P}(Y = g|x[i] = b)|, \qquad (2.3)$$

where a and b are two possible values of the ith feature of training sample x, and where $\hat{P}_{Y|x[i]}$ is estimated from a training set of samples. Then, the VDM dissimilarity between a sample x and a sample z is computed as

$$d_{vdm}(x,z) = \sum_i (\triangle_i(x[i], z[i]))^q, \tag{2.4}$$

where typically the integer $q = 2$.

Heterogeneous metrics that can deal with both numerical and non-numerical features [13, 77] have been derived from the VDM. Variants of the VDM may be metrics or pseudo-metrics. Other metrics use fuzzy logic to assign numerical truth values to fuzzy predicates and then use set-theoretic models to evaluate similarity based on the assigned truth values [59, 60].

Another general class of similarity functions that are useful in pattern recognition derives from information theory, and includes information content similarity [57], mutual information similarity [28, 41], residual entropy similarity [10], and similarity defined by the compressibility of one sample given another [39].

Lin [41] defines an information-theoretic similarity measure based on the information content of feature vectors, where information of each feature is defined in the standard way $I(x[i]) = -\log P(x[i])$:

$$s_{Lin}(x,z) = \frac{2I(\texttt{common}(x,z))}{I(\texttt{description}(x,z))}.$$

By $\texttt{common}(x,z)$, Lin means the set of features common to both objects x and z; by $\texttt{description}(x,z)$ he means the set of features needed to completely describe both objects x and z. Lin assumes that features are independent in his examples.

Recent work [10] has shown that Lin's similarity is a special case of Tversky's ratio contrast model, using information content as the saliency function, $f = I$, and with $\alpha = \beta = 0.5$. Thus Lin's similarity is both set-theoretic and information-theoretic. Lin's similarity takes into account context by incorporating the probability of features (their "information") into the similarity definition. The similarity is greater if the

common features are less likely. For some applications, the context will be very important and must be captured more strongly by the similarity function. To this end, one might ask, if one knew that a random sample R was at least described by the features in common between objects x and z, then how uncertain would one still be about R? This would specify the similarity of x and z in light of the context of the distribution of R. The *residual entropy* similarity is a set- and information-theoretic similarity that strongly captures context based on this idea [10].

The residual entropy similarity is based on Tversky's linear contrast model, with $\alpha = \beta = 0.5$, and mutual information as the saliency function: $f(x \cap z) = I(R; x \cap z \subset R)$, $f(x \setminus z) = I(R; x \setminus z \subset R)$, and $f(z \setminus x) = I(R; z \setminus x \subset R)$. Then, because $I(R; X) = H(R) - H(R|X)$, the residual entropy similarity is defined:

$$s_{re}(x, z) = -H(R|x \cap z \subset R) + \frac{H(R|x \setminus z \subset R)}{2} + \frac{H(R|z \setminus x \subset R)}{2}, \tag{2.5}$$

where $H(R|x \cap z \in R) = -\sum_{r \in \mathcal{B}} P(R = r|x \cap z \in R) \log P(R = r|x \cap z \in R)$, the terms $H(R|x \setminus z \subset R)$ and $H(R|z \setminus x \subset R)$ are analogously defined, and $\mathcal{B}$ is the sample domain. Research into the residual entropy similarity and its ability to capture context is ongoing.

More comprehensive reviews of similarities can be found in [60] and [18]. The experiments described in Chapter 6 make extensive use of the counting and VDM similarities.

2.2 Similarity-based Classifiers

Similarity-based classifiers are defined as those classifiers that require only a pairwise similarity – a description of the samples themselves is not needed. Similarity-based classifiers classify test samples given a labeled set of training samples, the pairwise similarity values of the training samples, and the similarity of the test sample to the training samples. If the description of the samples in terms of feature vectors is available, an existing or ad hoc similarity function that maps any two samples to a

similarity value may be used [4, 30, 33, 54]. This section reviews current similarity-based classifiers.

2.2.1 Nearest Neighbor

The simplest method for similarity-based classification is the nearest neighbor classifier, which determines the most similar training sample z to the test sample x, and classifies x as z's class:

$$\hat{y} = \arg \max_{h=1,\ldots,G} \left(\max_{z \in \mathcal{X}_h} s(x, z) \right), \tag{2.6}$$

where $\mathcal{X}_h$ is the set of training samples from class h. More generally, the k-nearest neighbor classifier (k-NN) determines a neighborhood of k most similar training samples to the test sample x, and classifies x as the most-frequently occurring class label among the neighbors. Experiments have shown that nearest neighbors can perform well on practical similarity-based classification tasks [2, 13, 54, 66]. For example, nearest neighbor classifiers using a tangent distortion metric and a shape similarity metric have both been shown to achieve very low error on the MNIST character recognition task.

2.2.2 Condensed Nearest Neighbor

Condensed near-neighbor strategies replace the set of training samples for each class with a set of prototypes for that class. Usually the prototype set is an edited set of the original training samples (also called edited nearest neighbors), but the prototypes do not need to be from the original training set. Let c_h be the number of the prototypes $\{\mu_{hl}\}$ for class h; then, the condensed nearest neighbor rule is to classify a test sample x as the class of the prototype to which it is most similar,

$$\hat{y} = \arg \max_{h=1,\ldots,G} \left(\max_{l=1,\ldots,c_h} s(x, \mu_{hl}) \right) \tag{2.7}$$

Many authors have considered strategies for condensing near-neighbors for similarity-based classification to increase classification speed, decrease the required memory, remove outliers, and possibly attain better performance [33,38,43,53,75]. A well-known strategy for condensing nearest neighbors in non-metric spaces is the *k-medoids* algorithm [27]. Given a set of c_h candidate prototypes selected from $\mathcal{X}_h$, the remaining training samples $z \in \mathcal{X}_h$ are assigned to their nearest (most similar) prototype, so that the set $\mathcal{X}_h$ of all training samples from class h is partitioned in c_h mutually-exclusive subsets $\{\mathcal{X}_{hl}\}$, and each $\mathcal{X}_{hl}$ is uniquely associated with candidate prototype μ_{hl}. Then, the lth prototype for the hth class is updated according to the standard maximum similarity update rule, which selects the new μ_{hl} as the training sample in $\mathcal{X}_{hl}$ which is most similar to all other samples in $\mathcal{X}_{hl}$,

$$\mu_{hl}^* = \arg \max_{\mu_{hl} \in \mathcal{X}_{hl}} \sum_{z \in \mathcal{X}_{hl}} s(z, \mu_{hl}). \tag{2.8}$$

The training samples are then reassigned to the updated prototypes, and the update rule (2.8) is repeated. The reassignment and update steps are repeated until a predetermined maximum number of iterations is reached or until the updated prototypes $\mu_{hl}^* = \mu_{hl}$ for all h and l. The number of prototypes in each class c_h is determined by cross-validation; the initial prototypes $\{\mu_{hl}\}$ are selected randomly from the training set.

2.2.3 Nearest Centroid

An extreme form of condensed near-neighbors is to replace each class's training samples by one prototypical sample, often called a *centroid*. The resulting nearest centroid classifier can be considered a simple parametric model [75], though it lacks a probabilistic structure. Let $s(x, z)$ be the similarity between a sample x and a sample z, and let there be a finite set of classes $1, 2, \ldots, G$. The nearest centroid approach classifies x as the class

$$\hat{y} = \arg \max_{h=1,\ldots,G} s(x, \mu_h), \tag{2.9}$$

where μ_h is the representative centroid for the class h. A standard definition for the centroid of a set of training samples is the training sample that has the maximum total similarity to all the training samples of the same class [33, 75]:

$$\mu_h = \arg\max_{\mu \in \mathcal{X}_h} \sum_{z \in \mathcal{X}_h} s(z, \mu). \tag{2.10}$$

For example, if each sample in Fig. 2.1 is described as a set of up to four facial features (eyes, nose, mouth, hair), and the similarity between two samples is defined to be the number of features that the two samples have in common, then the samples marked as μ_1 and μ_2 are the class centroids as per (2.10).

A variation of the nearest centroid classifier is the local nearest centroid classifier, which is an analog to the local nearest means classifier proposed by Mitani and Hamamoto [46, 47]. In this variant, the class centroids (2.10) are computed from a local neighborhood of each test point x; they are not computed from the entire training set. The neighborhood may be defined in many ways. The most common definition is the k-nearest neighbors. In this case, local nearest centroid is like the k-NN classifier, except that it classifies x as the class of its nearest centroid where the centroids are computed from the k-nearest neighbors of x.

The nearest centroid classifier is analogous to the nearest-mean classifier in Euclidean space, which is the optimal Euclidean-based classifier if one assumes that the class-conditional distributions are Gaussian, the class priors are equal, and that each class covariance is the identity matrix [16, 27]. The local nearest centroid classifier is analogous to the local nearest-mean classifier [46, 47] discussed in Chapter 4. Results for the novel methods presented in this dissertation are compared to the nearest centroid and local nearest centroid classifiers in Chapter 6.

2.2.4 Embedding in Euclidean Space — Multi-dimensional Scaling

One approach to similarity-based classification is to embed the training and test samples in Euclidean space using multidimensional scaling (MDS) [78], and then use

standard statistical learning methods in the Euclidean feature space. MDS maps pairwise similarity relationships onto a d-dimensional Euclidean space and is extensively used in psychology to visualize similarity relationships on two- or three-dimensional graphs [71]. More generally, the samples can be embedded in a pseudo-Euclidean space for classification [20, 54]. The embedding approach can also be used for clustering, for example Buhmann and Hofmann embed samples based on pairwise similarities in a low-dimensional Euclidean space by computing a multidimensional scaling solution subject to an entropy constraint [8] . This results in an Euclidean embedding that maximizes the separation between clusters in a data set, while maintaining as much as possible the original pairwise similarity structure of the data. One disadvantage of most nonlinear embedding methods is that classifying a new test sample requires re-computing the metric space embedding for all the data. This is a problem if all the test data or training data are not available at one time. Another disadvantage is that if the underlying similarity relationships are not well represented by a metric distance, then no embedding may be appropriate; for example, forcing samples related by an asymmetric similarity function to be classified using Euclidean distance may be suboptimal. Also, if the underlying similarity relationships are not well represented by a metric distance, the embedding may be relatively high-dimensional, invoking the curse of dimensionality. On the other hand, embedding the training samples in a low-dimensional Euclidean space may fail to sufficiently capture the similarity relationships between the samples [43, 71–73].

2.2.5 Using the Similarities-to-Training-Samples as Features

Similarity-based classification problems can also be turned into standard Euclidean-based learning problems by treating the $N \times 1$ vector of similarities between a test sample x and the N training samples $\{z_i\}, i = 1, 2, \ldots N$ as a feature vector [17,21,54]

$$u = [s(x, z_1)s(x, z_2) \ldots s(x, z_N)]^T. \tag{2.11}$$

Graepel et al. [21] propose a separating hyperplane classifier using this approach. Duin et al. [17, 54] consider various standard learning techniques for this approach, including a regularized Fisher linear discriminant classifier for this space.

An issue with using the vector of similarities as a feature vector is that the feature vector size is equal to the number of training samples, causing curse of dimensionality difficulties for learning. As investigated by Pekalska et al. [54], one way to mitigate the problem that the dimension of the feature space is equal to the number of training samples is to regularize the covariance matrix when applying linear discriminant analysis (LDA) or quadratic discriminant analysis (QDA). Another approach they suggest for solving the dimensionality problem is to use only a subset of the training samples to define the feature vector. The results of Pekalska et al. show that, on average over their different experiments, linear classifiers built on the similarity vectors achieve similar errors as 1-nearest neighbor, except in cases of severe noise, where the 1-nearest neighbor has high error. Also, their similarity-based linear classifiers generally perform slightly better than first embedding the training samples in a metric space and then applying a linear classifier.

2.2.6 Support Vector Machines

The popular support vector machines (SVMs) are a family of classifiers which have been successfully applied to wide a variety of classification problems [9, 27, 49, 74]. A SVM constructs a linear boundary in a high-dimensional Euclidean space such that the distance (or *margin*) of the sample closest to the boundary is maximized. Each sample is described by a d-dimensional feature vector in Euclidean space and its class label is $g \in \{+1, -1\}$. Given training samples $\{z_i\}$ and their corresponding class labels y_i, $i = 1, 2, \ldots N$ the SVM classification rule is to classify a test sample x as

$$\hat{y} = \text{sign}\left(\sum_{i=1}^{N} y_i \hat{\alpha}_i K(x, z_i) + b\right). \tag{2.12}$$

The coefficients $\hat{\alpha} = [\hat{\alpha}_1 \hat{\alpha}_2 \dots \hat{\alpha}_N]^T$, are computed by solving the constrained convex optimization problem

$$\begin{aligned} \hat{\alpha} &= \arg\max_{\alpha} \left(\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(z_i, z_j) \right) \\ &\quad \text{s. t.} \sum_{i=1}^{N} \alpha_i y_i = 0, \\ &\quad 0 \leq \alpha_i \leq C, \end{aligned} \tag{2.13}$$

where $C \in \mathbb{R}^+$ is a problem-dependent constant upper bound on the $\hat{\alpha}$ coefficients. The constant term b is computed by solving $y_i \left(\sum_{j=1}^{N} y_j \hat{\alpha}_j K(z_i, z_j) + b \right) = 1$ for α_i constrained as in (2.13). The kernel $K(x, z_i)$ is a symmetric, positive definite function which relates the test sample x to the training sample z_i.

Thus, a SVM solves a classification problem by first transforming the original d-dimensional feature space into a N-dimensional space by way of the $N \times N$ kernel $K(x, z_i)$, and then producing a linear classification boundary in the transformed space. The boundary is the *separating hyperplane* defined by the coefficients $\hat{\alpha}$. Practitioners choose the kernel function based on the particular classification problem to be solved. Popular choices for the kernel are

$$\begin{aligned} K_{linear}(x, z_i) &= (\langle x, z_i \rangle + 1)^q, \\ K_{Gaussian}(x, z_i) &= \exp(-\|x - z_i\|^2 / \sigma), \\ K_{sigmoid}(x, z_i) &= \tanh(\kappa_1 \langle x, z_i \rangle + \kappa_2), \end{aligned} \tag{2.14}$$

where $\langle x, z_i \rangle$ denotes the inner product between the test feature vector x and the training feature vector z_i.

One can view the kernel $K(x, z_i)$ as the similarity function $s(x, z_i)$ and interpret the SVM approach as a type of similarity-based classifier. For example, one could designate one of the kernels in (2.14) as $s(x, z_i)$. Then the SVM finds the optimum separating hyperplane in the N-dimensional Euclidean feature space where the ith feature is $s(x, z_i)$, the similarity of test sample x to the ith training sample z_i. Furthermore,

the original feature vectors x and z_i may themselves be vectors of application-specific similarities. Liao and Noble [40] classify newly discovered amino acid sequences into families of known proteins using such a composition of similarity functions. First, they use a specialized similarity function $\bar{s}(u, v_i)$ specifically designed to measure how biological sequences are related. The numerical feature vectors x are formed from similarities between an amino acid sequence u and N labeled sequences v_i used as training set, so that $x = [\bar{s}(u, v_1)\ \bar{s}(u, v_2) \dots \bar{s}(u, v_N)]$ and $z_i = [\bar{s}(v_i, v_1)\ \bar{s}(v_i, v_2) \dots \bar{s}(v_i, v_N)]$. The resulting $1 \times N$ numerical vectors are transformed into kernel-based similarities by use of an inner product and an exponential kernel. The resulting $N \times N$ kernel matrix is used for training SVMs to classify new amino acid sequences. The kernel matrix $K(x, z_i)$ may be viewed as a matrix of pairwise amino acid sequence similarities $s(x, z_i)$ derived from numerical vectors of ad-hoc biological similarities.

Viewing the kernel as a matrix of pairwise similarities leads to interpreting SVMs as a particular case of the similarities-to-training-samples approach described in Section 2.2.5. As such, SVMs may suffer from the same curse of dimensionality difficulties. More fundamentally, a difficulty with applying SVMs to similarity-based classification problems is in the definition of the kernels. Most widely-used kernels, like the ones in (2.14), rely on the properties of vector spaces, such as the inner product and the norm, to compute the similarity. This approach is not appropriate when the samples are not easily characterized by numerical vectors in Euclidean space, when the underlying features are not accessible, or when the similarity relationship between objects is not captured by any of the standard kernel functions.

If the kernel function cannot be computed from features, but the pairwise similarities are nonetheless available, then one approach to building similarity-based SVM classifiers is to use the matrix of pairwise training sample similarities as a kernel. If the provided similarity matrix is symmetric and positive definite or conditionally positive definite as defined by [62], then the similarity matrix can be used as a kernel in standard SVMs, so that $K(z_i, z_j) = s(z_i, z_j)$ in (2.13). Modified support vector techniques

must be used if the pairwise similarity matrix does not satisfy these properties.

2.2.7 Generalized Support Vector Machines

A generalized SVM, the *potential support vector machine* (PSVM), has been developed that can be used with any similarity matrix [29,30]. The similarity can be very general and need not rely on the inner product of Euclidean feature vectors. For the PSVM, the $N \times N$ matrix of training samples' pairwise similarities is the SVM kernel matrix $K(z_i, z_j) = s(z_i, z_j)$, where z_i, z_j are training samples. Thus, the classification rule for PSVM is the same as (2.12), but with the similarities directly appearing in the discriminant expression

$$\begin{aligned} \hat{y} &= \text{sign}\left(\sum_{i=1}^{N} \hat{\alpha}_i K(x, z_i) + b\right) \\ &= \text{sign}\left(\sum_{i=1}^{N} \hat{\alpha}_i s(x, z_i) + b\right), \end{aligned} \tag{2.15}$$

where $b = 1/N \sum_{i=1}^{N} y_i$. As for the SVM, the optimum PSVM hyperplane coefficients $\hat{\alpha}$ are computed by solving a constrained optimization problem. However, the objective function is different from the SVM objective (2.13); it depends both on the kernel K and on the squared kernel $Q = K^T K$ [37]:

$$\begin{aligned} \hat{\alpha} = \arg\min_{\alpha} \frac{1}{2}\alpha^T Q \alpha - y^T K \alpha + \epsilon \|\alpha\|_1 \\ \text{s. t.} - C < \alpha_i < C, \end{aligned} \tag{2.16}$$

where $y = [y_1 y_2 \ldots y_N]$ is the vector of training samples class labels, and $\epsilon \in \mathbb{R}^+$ and $C \in \mathbb{R}^+$ are problem-dependent, parameters which respectively regularize the minimization problem and limit the magnitude of the solution. The solution is the vector $\hat{\alpha}$ which optimally weights the similarities of a test sample to the N training samples.

The PSVM is a binary linear classifier in the Euclidean space where the ith dimension is the similarity of a test sample to the ith training sample. One difficulty

with the PSVM is that it requires enumerating $N \times N$ matrices, which is computationally infeasible given large N. Also, the PSVM, like other SVM-based classifiers, naturally works for binary classification problems. While SVMs have been adapted to multi-class problems [32], no multi-class PSVM results are available in the literature. Additionally, the PSVM requires the time-consuming task of cross-validating two parameters, ϵ and C and risks overfitting them. Experiments in this work compare the classification performance of the proposed generative approaches to that of the PSVM.

Chapter 3

SIMILARITY DISCRIMINANT ANALYSIS

This chapter introduces *similarity discriminant analysis* (SDA). In standard metric learning, quadratic discriminant analysis (QDA) is a generative classifier that generalizes the nearest-mean classifier by modeling each class-conditional distribution as a Gaussian. Analogously, SDA is a generative similarity-based classifier that generalizes the nearest-centroid classifier [75] by modeling each class-conditional distribution with a parametric probability model. The SDA class-conditional probability models have exponential form, because they are derived as the maximum entropy distributions subject to constraints on the mean similarities of the data to the class centroids. As with other parametric approaches to classification, the resulting log-linear SDA classifier is powerful when it effectively models the true generating distribution. Section 3.1 introduces SDA and shows how it classifies. Section 3.2 extends SDA from using class centroids to using arbitrary descriptive statistics to discriminate between the classes, including continuous-valued statistics. Section 3.3 details the relationship of SDA to other learning approaches – naive Bayes, QDA, LDA, SVM, and PSVM – with focus on comparing SDA to metric approaches when the similarities are used as metric features. Section 3.4 discusses how one could build mixed models that combine standard metric features and similarity statistics from heterogeneous data, that is samples described by both numeric and categorical features.

3.1 A Generative Centroid-based Classifier

Assume a class centroid μ_h has been determined for the hth class, where $h = 1, \ldots, G$. A problem with the nearest centroid classifier given in (2.9) is that it does not take

into account the variability of the similarities to the centroid within a class. An example of this issue is shown in Fig. 2.1: samples from class one have similarity of 3 or 4 to the class one centroid, whereas samples from class two have similarities in the range $0-3$ to the class two centroid. To take into account this variability, first consider a simple generalization of nearest centroid, here called the *adjusted nearest centroid classifier*: classify a test sample x as class $\hat{y}$ where

$$\hat{y} = \arg \max_{h=1,\ldots,G} \frac{s(x, \mu_h)}{\bar{s}_{hh}}, \tag{3.1}$$

and where $\bar{s}_{hh}$ is the average similarity of class h samples to the class h centroid,

$$\bar{s}_{hh} = \frac{1}{n_h} \sum_{z \in \mathcal{X}_h} s(z, \mu_h),$$

where $n_h = |\mathcal{X}_h|$. The adjusted nearest centroid classifier is analogous to the one-dimensional Gaussian rule of classifying based on the the variance-weighted distances to the class means, $\|x - \tilde{\mu}_h\|/\tilde{\sigma}_h$, where x, $\tilde{\mu}_h$, $\tilde{\sigma}_h \in \mathbb{R}$. The adjusted nearest centroid classifier is more flexible than the nearest centroid classifier, but lacks a probabilistic structure, and takes into account only the similarity of a sample to one class centroid.

Thus, a generative centroid-based classifier that models the probability distribution of the test sample similarity statistics $s(x, \mu_h)$ for each h is proposed. Begin with the Bayes classifier [27], which assigns a test sample x the class $\hat{y}$ that minimizes the expected misclassification cost,

$$\hat{y} = \arg \min_{f=1,\ldots,G} \sum_{g=1}^{G} C(f, g) P(Y = g|x), \tag{3.2}$$

where $C(f, g)$ is the cost of classifying the test sample x as class f if the true class is g and $P(g|x)$ is the probability that sample x belongs in class g . In practice the distribution $P(g|x)$ is generally unknown, and thus the Bayes classifier of (3.2) is an unattainable ideal.

Assume that all test and training samples come from some abstract space of samples $\mathcal{B}$, which might be an ill-defined space, such as $\mathcal{B}$ is the set of all amino

acids, or $\mathcal{B}$ is the set of all terrorist events, or $\mathcal{B}$ is the set of all women who gave birth to twins. Let $x, \mu_h, z \in \mathcal{B}$, and let the similarity function be some function $s : \mathcal{B} \times \mathcal{B} \to \Omega$, where $\Omega \subset \mathbb{R}$. If the set of possible samples $\mathcal{B}$ is finite, then the space of the pairwise similarities Ω will also be finite, and hence discrete. For simplicity, in this section assume that Ω is a finite discrete space. Continuous and possibly infinite spaces $\mathcal{B}, \Omega$ are briefly discussed in Section 3.2.3.

Consider a random test sample X with random class label Y, where x will denote a realization of X. Assume that the relevant information about X's class label is captured by the set $\mathcal{T}(X)$ of G descriptive statistics

$$\mathcal{T}(X) = \{s(X, \mu_1), s(X, \mu_2), \ldots, s(X, \mu_G)\}.$$

That is, the relevant information about x is captured by its similarity to each class centroid. Under this assumption, given a particular test sample x, the classification rule (3.2) becomes: classify x as class $\hat{y}$ that solves

$$\arg \min_{f=1,\ldots,G} \sum_{g=1}^{G} C(f, g) P(Y = g | \mathcal{T}(x)).$$

Using Bayes rule, this is equivalent to the problem

$$\arg \min_{f=1,\ldots,G} \sum_{g=1}^{G} C(f, g) P(\mathcal{T}(x) | Y = g) P(Y = g). \tag{3.3}$$

Note that $P(\mathcal{T}(x)|Y = g)$ is the probability of seeing a particular set of similarities between the test sample x and the G class centroids $\{\mu_1, \mu_2, \ldots, \mu_G\}$ given that x is a class g sample.

Next, assume that each unknown class-conditional distribution $P(\mathcal{T}(x)|Y = g)$ has the same average value as the training sample data from class g. That is, given a random test sample X there will be a random similarity $s(X, \mu_h)$; constrain the class-conditional distribution $P(\mathcal{T}(x)|Y = g)$ such that

$$E_{P(\mathcal{T}(x)|Y=g)}[s(X, \mu_h)] = \frac{1}{n_g} \sum_{z \in \mathcal{X}_g} s(z, \mu_h), \tag{3.4}$$

holds for each g and h where n_g is the number of training samples of class g. Each constraint requires that the class-conditional expectation of one of the elements of $\mathcal{T}(X)$ is equal to the maximum likelihood estimate of that element given the training data. This makes for G constraints for each class-conditional distribution, for a total of $G \times G$ constraints because there are G class-conditional distributions. Given these constraints, there is some compact and convex feasible set of class-conditional distributions. A feasible solution will always exist because the constraints are based on the data.

As prescribed by Jaynes' principle of maximum entropy [34], a unique class-conditional joint distribution is selected by choosing the maximum entropy solution that satisfies (3.4). Maximum entropy distributions have the maximum possible uncertainty, such that they are as uniform as possible while still satisfying given constraints. Given a set of moment constraints, the maximum entropy solution is known to have exponential form [14]. For example, in standard metric learning, the Gaussian class-conditional distribution model used in LDA and QDA is the maximum entropy distribution given a specific mean vector and covariance matrix [14].

The maximum entropy distribution that satisfies the moment constraints specified in (3.4) is

$$\hat{P}(\mathcal{T}(x)|Y=g) = \gamma_g e^{\left(\sum_{h=1}^{G} \lambda_{gh} s(x,\mu_h)\right)}, \tag{3.5}$$

where $\{\gamma_g, \lambda_{g1}, \lambda_{g2}, \ldots, \lambda_{gG}\}$ are a unique set that ensures that the constraints (3.4) are satisfied and that $\hat{P}(\mathcal{T}(x)|Y=g)$ is non-negative and normalized. Rewrite equation (3.5) as

$$\hat{P}(\mathcal{T}(x)|Y=g) = \prod_{h=1}^{G} \gamma_{gh} e^{\lambda_{gh} s(x,\mu_h)} \tag{3.6}$$

where $\prod_h \gamma_{gh} = \gamma_g$. Let

$$\hat{P}(s(x,\mu_h)|Y=g) = \gamma_{gh} e^{\lambda_{gh} s(x,\mu_h)};$$

then (3.6) can be written

$$\hat{P}(\mathcal{T}(x)|Y=g) = \prod_{h=1}^{G} \hat{P}(s(x,\mu_h)|Y=g).$$

That is, under the maximum entropy assumption, the joint distribution on $\mathcal{T}(x)$ is the product of the marginal distributions on each similarity statistic comprising the set $\mathcal{T}(X)$. Thus, the similarity statistics are conditionally independent given the class label under this model. Although one does not expect this conditional independence to be strictly valid, the hypothesis is that it will be an effective model, just as the naive Bayes' model that features are independent is optimistic but useful.

Substituting the maximum entropy solution (3.5) into (3.3) yields the classification rule: classify x as the class $\hat{y}$ which solves

$$\arg \min_{f=1,\ldots,G} \sum_{g=1}^{G} C(f,g) \left(\prod_{h=1}^{G} \gamma_{gh} e^{\lambda_{gh} s(x,\mu_h)} \right) P(Y=g). \tag{3.7}$$

To solve for the parameters $\{\lambda_{gh}, \gamma_{gh}\}$, one solves the G constraints individually for λ_{gh}. Then given $\{\lambda_{gh}\}$, the $\{\gamma_{gh}\}$ are trivially found using the normalization constraint. Solving for λ_{gh} is straightforward; for example, one uses the Nelder-Mead optimizer built into Matlab (version 15) in the `fminsearch()` function [45]. This is the method used throughout this work. As an alternative, one may find the probability mass function with maximum entropy, subject to the constraints, without a priori knowledge that the solution is exponential.

The classifier given in (3.7) is termed the *similarity discriminant analysis* (SDA).

3.1.1 Analysis of the Two Class Case

This section further analyzes the two class case $G = 2$ for equal class prior probabilities and zero-one costs, that is $C(f,g) = 1$ for $f \neq g$ and $C(f,g) = 0$ for $f = g$. Then the classification rule (3.7) becomes: choose class 1 if

$$\frac{P(s(x,\mu_1)|Y=1)}{P(s(x,\mu_1)|Y=2)} \frac{P(s(x,\mu_2)|Y=1)}{P(s(x,\mu_2)|Y=2)} > 1. \tag{3.8}$$

Applying the maximum entropy solution for the class-conditional distributions, (3.8) becomes: choose class 1 if

$$\frac{\gamma_{11}e^{\lambda_{11}s(x,\mu_1)}}{\gamma_{21}e^{\lambda_{21}s(x,\mu_1)}}\frac{\gamma_{12}e^{\lambda_{12}s(x,\mu_2)}}{\gamma_{22}e^{\lambda_{22}s(x,\mu_2)}} > 1. \tag{3.9}$$

The probability distributions of the similarities capture the characteristic average deviation for each class and the average cross-class deviations. In (3.8), the ratio term

$$\frac{P(s(x,\mu_1)|Y=1)}{P(s(x,\mu_1)|Y=2)}$$

calculates whether the similarity between the test sample x and the class one centroid μ_1 is better explained probabilistically by assuming x is from class one or class two. Likewise, in (3.8), the ratio term

$$\frac{P(s(x,\mu_2)|Y=1)}{P(s(x,\mu_2)|Y=2)}$$

establishes whether the similarity $s(x,\mu_2)$ is better explained probabilistically by the hypothesis that x is in class one or class two.

Let us analyze the generative model for the training samples given in Fig. 2.1. As before, each sample is considered to be a set of up to four facial features (eyes, nose, mouth, hair), and the similarity of two samples is calculated to be the number of features they have in common. Recall that in similarity-based classification we assume that we might have only the pairwise similarities to use to classify. For example, applying the nearest centroid classifier as per (2.9) to the training samples, two of the class two samples would be misclassified because they are more similar to the class one centroid than to the class two centroid. In addition, the other class two samples are equidistant from the two class centroids; thus, depending on how ties are resolved, all of the class two samples could be misclassified.

To apply the generative model one must estimate the necessary probabilities. Let us begin with $P(s(x,\mu_1)|Y=1)$, noting it is a pmf over the five possible similarity values. The average similarity of class one samples to the class one centroid is $(4+3+$

$3+3+3)/5 = 3.2$. Thus the generative estimate $\hat{P}(s(x,\mu_1)|Y=1)$ will be exponential with $E[\hat{P}(s(x,\mu_1)|Y=1))] = 3.2$. Since the uniform distribution over the possible similarities $0,1,2,3,4$ has mean 2, it must be that the estimated $\hat{P}(s(x,\mu_1)|Y=1)$ is an increasing exponential pmf over the possible similarities $0,1,2,3,4$ (the estimated parameters are $\lambda_{11} \approx .73, \gamma_{11} \approx .03$). It is intuitive that higher similarities between a class one sample and the class one centroid are more probable.

Next, consider the probability distribution of the similarities between class two samples and the class two centroid. The average similarity shown in Fig. 2.1 is $(3+0+1+1+1)/5 = 1.2$. Thus, the constraint on the estimated probability $\hat{P}(s(x,\mu_2)|Y=2)$ is $E[\hat{P}(s(x,\mu_1)|Y=1))] = 1.2$. Since the estimated probability must be an exponential pmf over the possible similarities $0,1,2,3,4$, the pmf must be a decreasing exponential (the estimated parameters are $\lambda_{22} \approx -.43$, $\gamma_{22} \approx .40$). That is, the model predicts that class two samples will not be very similar to the class two centroid.

It remains to estimate the cross-class pmf's. The estimated pmf $\hat{P}(s(x,\mu_2)|Y=1)$ is constrained to have expectation equal to $(3+2+2+2+3)/5 = 2.4$, and thus is a slightly-increasing exponential pmf (the estimated parameters are $\lambda_{12} \approx .20$, $\gamma_{12} \approx .13$). That is, class one samples are predicted to be somewhat similar to the class two centroid. Last, the estimated pmf $\hat{P}(s(x,\mu_1)|Y=2)$ has expectation equal to $(3+0+2+2+1)/5 = 1.6$, leading to a slightly decreasing exponential pmf (the estimated parameters are $\lambda_{21} \approx -.20, \gamma_{21} \approx .29$). Thus, the model predicts that class two samples will not be very similar to either the class two centroid or the class one centroid, but that they will be actually slightly more similar to the class one centroid.

To classify a given test sample x, the estimated pmf's are used to calculate the discriminant value on the left-hand side of (3.8). In Fig. 3.1, each of the vertices of the surface corresponds to a possible similarity pair $(s(x,\mu_1), s(x,\mu_2))$, and has height equal to the resulting discriminant value. Each vertex is labeled with its predicted class, as per the classification rule (3.8). Some of these test similarity pairs

are not achievable with any set of features, for example, it is not possible to achieve $s(x, \mu_2) = 4$ for any sample x, because μ_2 has only three facial features. However, the SDA classifier does not know this, and it models a pmf over the given set of possible similarities. As an example of how this classifier differs from the nearest centroid classifier, consider the test sample $x = \{hair, mouth\}$, which has $s(x, \mu_1) = 2$, $s(x, \mu_2) = 1$. The nearest centroid classifier would classify x as class one, whereas SDA classifies it as class two.

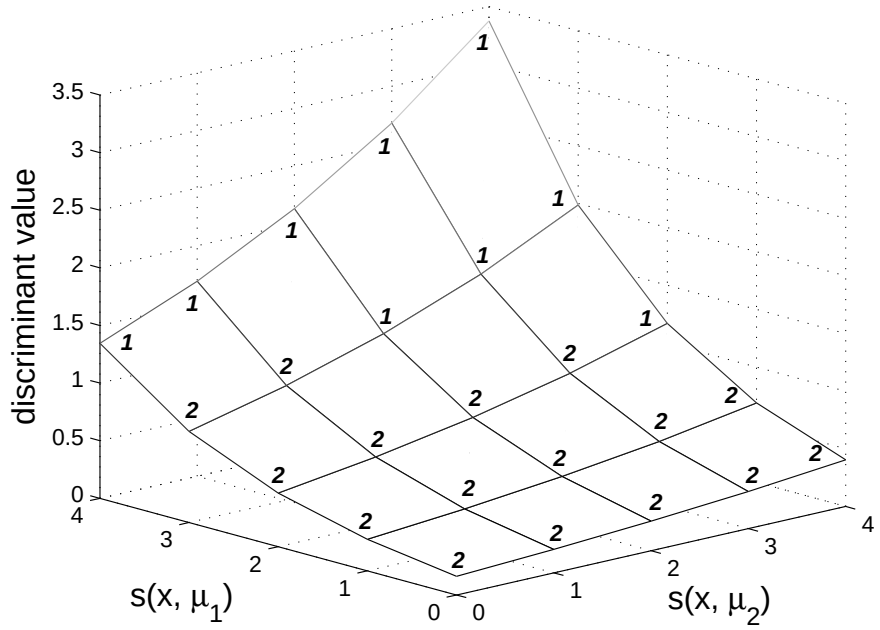

Figure 3.1: Corresponding to the samples shown in Fig. 2.1, for each possible pair of test sample similarities $s(x, \mu_1)$ and $s(x, \mu_2)$, the figure shows the classifier discriminant value and the class label, as per (3.8).

3.2 General Generative Models for Similarity-based Classification

The previous section introduced SDA for the case when the descriptive statistics are the similarities of the samples to the class centroids. This section generalizes SDA to

arbitrary descriptive statistics $\mathcal{T}(x)$ which can be used to discriminate different classes and describes the resulting general generative model for classifying with arbitrary statistics.

3.2.1 Descriptive Statistics

Several possibilities for the descriptive statistics $\mathcal{T}(x)$ are described below.

Centroid Definitions

A standard centroid definition was given in (2.10). Another choice is to allow a class prototype that is not constrained to be a training sample,

$$\mu_h^* = \arg\max_{\mu \in \mathcal{B}} \sum_{z \in \mathcal{X}_h} s(z, \mu). \tag{3.10}$$

In this case the solution μ_h^* requires a description of the entire space of possible samples $\mathcal{B}$. In practice, one may not know the entire sample space $\mathcal{B}$, only the training samples $\mathcal{X}$, so it may not be possible to calculate μ_h^*.

A third definition of a class prototype is based on Tversky's analysis of similarity-based near-neighbor relationships [64, 73], and takes into account the similarity-based ranks of a training sample's near-neighbors. Define the neighborhood $\mathcal{N}(z) \subseteq \mathcal{X}$ of a sample z as the set of training samples whose nearest neighbor in similarity space is z. The popularity of z is the size of its neighborhood $|\mathcal{N}(z)|$. The class centroid is the sample with the highest popularity, that is,

$$\mu_h = \arg\max_{z \in \mathcal{X}_h} |\mathcal{N}(z)|. \tag{3.11}$$

This centroid is the training sample that is most often the closest neighbor of the training samples in the class. Ties in popularity are broken by selecting the sample with the highest total similarity to its neighbors.

Higher Order and Non-Centroidal Descriptive Statistics

Given a set of class centroids $\{\mu_h\}$, higher-order statistics could be used as, or added to, the set of descriptive statistics $\mathcal{T}(X)$, such as $(s(X,\mu_h) - E[s(X,\mu_h)])^2$, or cross-class statistics, such as $(s(X,\mu_h) - E[s(X,\mu_g)])^2$. Or, instead of the centroid-based statistics $\{s(X,\mu_h)\}$, it might be more appropriate to use the nonparametric statistics formed by the total pairwise similarity for each class h, such that the hth descriptive statistic in test set $\mathcal{T}(X)$ is $\sum_{z\in\mathcal{X}_h} s(X,z)$.

Nearest Neighbor Similarity

A descriptive statistic that is not centroid-based is the *nearest neighbor similarity*: a test sample's similarity to its most similar training sample. Given a sample x and the training samples $z \in \mathcal{X}$, the nearest neighbor similarity is defined

$$s_{nn}(x) = \max_{z\in\mathcal{X}} s(x,z). \tag{3.12}$$

The SDA classifier based on nearest neighbor similarity, denoted by *nnSDA*, may be viewed as a generalization of the similarity-based nearest neighbor classifier (1-NN) discussed in Section 2.2.1. That classifier labels x with the same class label as its nearest neighbor without making use of any information about its similarity to such nearest neighbor. The nnSDA classifier, on the other hand, classifies x as the class of its nearest neighbor based on a probabilistic model of $s_{nn}(x)$. The probability model is computed with the mean-constrained maximum entropy approach of Section 3.1, which results in exponential solutions. In this case, the constraint is that the mean of the distribution must be the same as the empirical average of the observed nearest neighbor similarities. Denote by $s_{nn,h}(X)$ the random similarity of a random test sample X to its nearest neighbor in class h. For nnSDA, the constraint is written as

$$E_{P(\mathcal{T}(x)|Y=g)}[s_{nn,h}(X)] = \frac{1}{n_g}\sum_{z\in\mathcal{X}_g} s_{nn,h}(z), \tag{3.13}$$

and the classification rule becomes to classify as the class $\hat{y}$ that solves

$$\arg \min_{f=1,\ldots,G} \sum_{g=1}^{G} C(f,g) \left(\prod_{h=1}^{G} \gamma_{gh} e^{\lambda_{gh} s_{nn,h}(x)} \right) P(Y=g), \tag{3.14}$$

where the parameters λ_{gh} and γ_{gh} are computed with the same numerical optimization method used for SDA.

The experiments in Chapter 6 compare various similarity-based classifiers on various data sets. SDA and local SDA use the similarity to the maximum-sum similarity centroid (2.10) as the descriptive statistic. The nearest-centroid (NC), local nearest centroid (local NC) and the condensed nearest neighbor (CNN) classifiers also use (2.10). The nnSDA classifier uses (3.12). As further discussed in the next section, the SDA framework accommodates any desired set of descriptive statistics $\mathcal{T}(x)$: different similarity functions could be mixed, dissimilarities and similarities can be mixed, and so on.

3.2.2 Generative Classifier from Arbitrary Descriptive Statistics

Given an arbitrary set of M descriptive statistics $\mathcal{T}(x)$, the same reasoning of Section 3.1 produces a generative similarity-based classifier. First, the assumption is that $\mathcal{T}(x)$ is sufficient information to classify x leads to the classification rule given in (3.3). Second, for the mth descriptive statistic $T_m(x) \in \mathcal{T}(x)$, $m = 1, \ldots M$, one assumes that its mean with respect to the class conditional distribution of $\mathcal{T}(x)$ is equal to the training sample mean:

$$E_{P(\mathcal{T}(x)|g)}[T_m(X)] = \frac{1}{n_g} \sum_{z \in \mathcal{X}_g} T_m(z). \tag{3.15}$$

Third, given the $M \times G$ constraints specified by (3.15), one estimates the class-conditional distribution to be the maximum entropy distribution,

$$\begin{aligned} \hat{P}(\mathcal{T}(x)|g) &= \prod_{m=1}^{M} \gamma_{gm} e^{\lambda_{gm} T_m(x)} \\ &= \prod_{m=1}^{M} \hat{P}(T_m(x)|g). \end{aligned} \tag{3.16}$$

Substituting the maximum entropy solution (3.16) into (3.3) yields the SDA classification rule: classify x as the class $\hat{y}$ which solves

$$\arg \min_{f=1,\ldots,G} \sum_{g=1}^{G} C(f,g) P(g) \prod_{m=1}^{M} \gamma_{gm} e^{\lambda_{gm} T_m(x)} \; . \tag{3.17}$$

The parameters $\{\lambda_{gm}, \gamma_{gm}\}$ are calculated as in the centroid-based SDA case described in Section 3.1.

3.2.3 Continuous-valued Statistics

The generative classification models presented in this chapter can be extended to the case in which the statistics $\mathcal{T}(x)$ are from a continuous set Ω. This will be the case, for example, when using an overlap similarity (e.g. $\max\{x[i], z[i]\}$) with real-valued features, or when the similarity between X and z is the Euclidean distance. Then, the expectation in (3.15) is a normalized integral over the continuous set of possible similarity values. Let a and b denote the minimum and maximum possible similarity values (and hence the lower and upper bound on the expectation's integral). Then simplifying (3.15) yields the relationship

$$\frac{e^{\lambda_{gm} b}(\lambda_{gm} b - 1) - e^{\lambda_{gm} a}(\lambda_{gm} a - 1)}{\lambda_{gm}(e^{\lambda_{gm} b} - e^{\lambda_{gm} a})} = \bar{t}_{gm}, \tag{3.18}$$

where $\bar{t}_{gm} = \frac{1}{n_g} \sum_{z \in \mathcal{X}_g} T_m(z)$. The solution to (3.18) can be computed numerically. For the special case $a = 0$ and $b = \infty$, the solution is $\lambda_{gm} = -1/\bar{t}_{gm}$.

3.3 Relationship of SDA to Other Classifiers

This section examines the relationship of the SDA classifier to standard metric classifiers. Section 3.3.1 shows that the generative classifier based on arbitrary statistics described in Section 3.2.2 and the naive Bayes classifier in general are different. However, they are equivalent in the special case of binary feature vectors where the descriptive statistics are the binary feature themselves. Sections 3.3.2 and 3.3.3 detail the relationship of SDA to QDA and LDA for a two-class problem where the metric features are the similarities to the centroids of the classes. The log-linear SDA discriminant is shown to have the same formal expression as the discriminants produced by QDA and LDA, although SDA is more general in that it can seamlessly accommodate discrete or continuous similarities without relying on the assumption that the class models be bivariate Gaussians. Section 3.3.4 compares SDA to the general case in which the metric features are the similarities of a test sample to all the training samples. As a particular case of using similarities as metric features, Section 3.3.5 compares SDA to SVMs with inner product kernels and with the similarities to the class centroids as the metric features. Section 3.3.6 examines SDA and PSVMs, where the PSVM kernel consists of the similarities of the training points to the class centroids, and shows that the two classifiers are fundamentally different. Finally, Section 3.3.7 summarizes the difference between SDA and the other classifiers by examining the decision boundaries produced by the various methods for a simulated binary classification problem.

3.3.1 Relationship of SDA to Naive Bayes

This section shows that for the specific case of binary feature vectors, classifying with the naive Bayes classifier is equivalent to classifying with SDA using the features as the descriptive statistics. However, if the features take on more than two possible values, SDA and naive Bayes are not equivalent.

The fundamental assumption of the naive Bayes classifier is that the features are assumed independent. Thus, the class-conditional probability model for a test sample x is the product of the marginal pmfs of the features:

$$P(X = x|Y) = \prod_{i=1}^{N} P(X[i] = x[i]|Y), \tag{3.19}$$

where N is the number of features. The marginal pmfs are estimated from training data, for example using frequency counts. For the case of continuous data, one often assumes Gaussian marginals and estimates the means and standard deviations from training data.

To investigate the relationship between SDA and naive Bayes, consider the case that each sample is described by a set of binary features, that is, the sample space $\mathcal{B} = \{0, 1\}^N$ for some finite value of N. Given $z \in \mathcal{B}$, let the ith feature be denoted $z[i] \in \{0, 1\}$. Consider the SDA classifier using descriptive statistics $\mathcal{T}(x) = \{x[1], x[2], \ldots, x[N]\}$. In this case, there are N marginal class-conditional distributions to estimate for each class, as per (3.16). For each marginal distribution there are two unknowns, $P(x[i] = 0|Y = g)$ and $P(x[i] = 1|Y = g)$, and two constraints: the normalization constraint and the expectation constraint given in (3.15):

$$\begin{aligned} P(x[i] = 0|g) + P(x[i] = 1|g) &= 1 \\ 0 \times P(x[i] = 0|g) + 1 \times P(x[i] = 1|g) &= \frac{1}{n_g} \sum_{z \in \mathcal{X}_g} I_{z[i]=1}, \end{aligned}$$

where I is the indicator function. These are the same constraints as for naive Bayes. There is only one possible solution for the class-conditional distributions given these constraints. Thus, for binary features and $\mathcal{T}(x) = \{x[1], x[2], \ldots, x[N]\}$, SDA and naive Bayes are equivalent. However, in general SDA and naive Bayes are not equivalent. If any $x[i]$ can take on more than two possible values, then naive Bayes differs from SDA: naive Bayes estimates the probability of each of the possible values of each $x[i]$, whereas SDA will estimate an exponential pmf over the possible values of $x[i]$ based on the empirical mean of the ith feature for the training samples of each class.

3.3.2 Relationship of SDA to QDA

QDA is a generative classifier that models each class by a Gaussian class-conditional distribution in a d-dimensional Euclidean feature space [27]. In standard QDA a mean $\hat{u} \in \mathbb{R}^d$ and covariance matrix $\hat{\Sigma} \in \mathbb{R}^d \times \mathbb{R}^d$ are estimated from the training samples for each class, often using maximum likelihood (ML). The class prior $P(Y = g)$ may be either known or estimated, also with ML. The discriminant function $D_{QDA,g}(x)$ for the gth class is the logarithm of the Gaussian class-conditional distribution with the class prior term added:

$$D_{QDA,g}(x) = -\frac{1}{2}(x - \hat{u}_g)^T \hat{\Sigma}_g^{-1}(x - \hat{u}_g) + \hat{u}_g^T \Sigma_g^{-1} x - \frac{1}{2}\log|\hat{\Sigma}_g| + \log P(Y = g). \quad (3.20)$$

A test point $x \in \mathbb{R}^d$ is classified by determining which class-conditional Gaussian distribution is most likely to have generated the test point. The classification rule is written in terms of $D_{QDA,g}$ as

$$\hat{y} = \arg\max_g D_{QDA,g}(x), \quad (3.21)$$

where for simplicity $(0, 1)$ misclassification costs are assumed.

QDA has a dual nature. It is both a Gaussian random vector model on continuous features and the maximum entropy distribution subject to constraints on $\hat{u}$ and $\hat{\Sigma}$ based on the observed data [14]. In this respect, SDA is like QDA, because it too models a class-conditional generating distribution as the maximum entropy distribution given moment constraints based on the data. However, SDA is more general than QDA. A major difference between QDA and SDA is that QDA is rooted in an Euclidean representation of the feature vectors, and the class-conditional Gaussian distributions directly model the probability of the test point x. SDA does not rely on the Euclidean assumption, and the class-conditional exponential distributions model some set of descriptive statistics $\mathcal{T}(x)$. Thus, in QDA the tested quantity is the probability of the d-dimensional test feature vector x, but in SDA the tested quantity is

the probability of the descriptive statistics $\mathcal{T}(x)$ calculated as functions of the test sample $x \in \mathcal{B}$.

Another major difference between QDA and SDA arises when the Euclidean features are not Gaussian (e. g. they may be discrete or continuous and finite). In this case the QDA assumption that the class-conditional models be Gaussian is incorrect. One may still use (3.20) to model the gth class discriminant and estimate $\hat{u_g}$ and $\hat{\Sigma_g}$ from the data, but the model will be inherently biased. To avoid this problem, one could instead appeal to the dual nature of QDA, and estimate the class-conditional model as the maximum entropy distribution subject to second order constraints. However, this approach still results in a Gaussian class-conditional model which is only an approximation of the true underlying generative distribution; as such, bias is still a problem. Thus, QDA is limited to class-conditional models for which the Gaussian assumption is a good approximation for the underlying distribution. On the other hand, SDA can seamlessly model both discrete and continuous variables. The exponential class-conditional probability models produced by SDA are applicable to continuous or discrete descriptive statistics of any order. In this respect, SDA represents a flexible, more general family of classifiers than QDA. The diagram in Figure 3.3.2 is a simple taxonomy of various generative classifiers based on maximum entropy probability function estimates. SDA is applicable in all situations, whereas other known techniques are limited to specific cases.

One may more closely explore the relationship between QDA and SDA by considering the feature vector u to be a set of descriptive statistics for test sample x. Consider applying QDA to a two-class problem, where the two-dimensional Euclidean feature vector consists of the similarities of the sample x to the class centroids μ_1 and μ_2:

$$u \doteq [s(x, \mu_1)\, s(x, \mu_2)]^T, \tag{3.22}$$

where $s(x, \mu_1), s(x, \mu_2) \in \mathbb{R}$. The class-conditional mean $\hat{u}_1$ and covariance matrix $\hat{\Sigma}_1$ are estimated from a training set of feature vectors $\{z_i\}$ using ML estimation. For

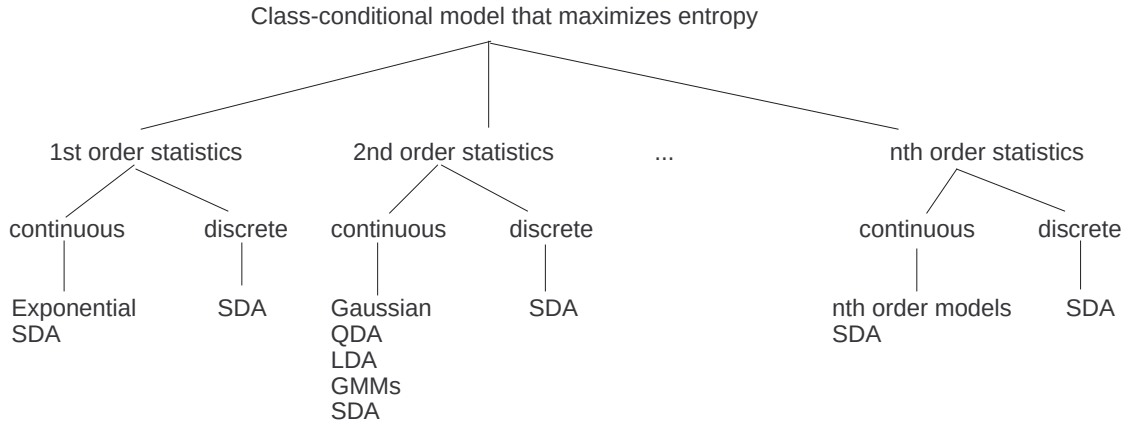

Figure 3.2: Relationship between class-conditional models that maximize entropy, for continuous and discrete constrained statistics of various orders.

the gth class

$$\hat{u}_g = \begin{bmatrix} \hat{u}_{g1} & \hat{u}_{g2} \end{bmatrix}^T = \frac{1}{n_g} \sum_{z_i \in \mathcal{X}_g} [s(z_i, \mu_1)\, s(z_i, \mu_2)]^T, \tag{3.23}$$

$$\hat{\Sigma}_g^{-1} = \begin{bmatrix} a & c \\ c & b \end{bmatrix} = \left[\tfrac{1}{n_g - 1} \textstyle\sum_{z_i \in \mathcal{X}_g} (z_i - \hat{u}_g)(z_i - \hat{u}_g)^T \right]^{-1}, \tag{3.24}$$

where $n_g = |\mathcal{X}_g|$. Then, the quadratic discriminant in (3.20) can be rewritten as

$$-\frac{1}{2} \begin{bmatrix} s(x, \mu_1) & s(x, \mu_2) \end{bmatrix} \begin{bmatrix} a & c \\ c & b \end{bmatrix} \begin{bmatrix} s(x, \mu_1) \\ s(x, \mu_2) \end{bmatrix} + \begin{bmatrix} \hat{u}_{g1} & \hat{u}_{g2} \end{bmatrix} \begin{bmatrix} a & c \\ c & b \end{bmatrix} \begin{bmatrix} s(x, \mu_1) \\ s(x, \mu_2) \end{bmatrix} + u_{g0},$$

where the term

$$u_{g0} = -\frac{1}{2} \begin{bmatrix} \hat{u}_{g1} & \hat{u}_{g2} \end{bmatrix} \begin{bmatrix} a & c \\ c & b \end{bmatrix} \begin{bmatrix} \hat{u}_{g1} \\ \hat{u}_{g2} \end{bmatrix} + \frac{1}{2} \log(ab - c^2) + \log P(Y = g)$$

encompasses the class prior and the terms that do not depend on the feature vector u. Carrying out the matrix multiplications and grouping the terms yields the discriminant expression

$$\begin{aligned} D_{QDA,g}(x) = {} & (a\hat{u}_{g1} + c\hat{u}_{g2}) s(x, \mu_1) + (c\hat{u}_{g1} + b\hat{u}_{g2}) s(x, \mu_2) \\ & - c s(x, \mu_1) s(x, \mu_2) - \frac{1}{2} a s^2(x, \mu_1) - \frac{1}{2} b s^2(x, \mu_2) + u_{g0}. \end{aligned} \tag{3.25}$$

$D_{QDA,g}(x)$ can be expressed in the same form as the log-linear SDA discriminant. Recall the expression (3.17) for the general SDA classifier based on descriptive statistics $\mathcal{T}(x)$ and assume $(0,1)$ misclassification costs; taking the logarithm gives the expression for the SDA discriminant for class g:

$$D_{SDA,g}(x) = \left(\sum_{m=1}^{M} \lambda_{gm} T_m(x)\right) + \left(\sum_{m=1}^{M} \log \gamma_{1m}\right) + \log P(Y = g). \tag{3.26}$$

Inspection of (3.25) and (3.26) reveals that both $D_{QDA,g}$ and $D_{SDA,g}$ are linear combinations of the descriptive statistics $\mathcal{T}(x) = \{T_1(x), T_2(x), T_3(x), T_4(x), T_5(x)\}$. In fact, they are formally the same, as can be seen by making the following term assignments:

$$\begin{aligned} T_1(x), &\leftrightarrow s(x,\mu_1) \\ T_2(x) &\leftrightarrow s(x,\mu_2) \\ T_3(x) &\leftrightarrow s(x,\mu_1)s(x,\mu_2) \\ T_4(x) &\leftrightarrow s^2(x,\mu_1) \\ T_5(x) &\leftrightarrow s^2(x,\mu_2) \end{aligned} \tag{3.27}$$

and

$$\begin{aligned} \lambda_{g1} &\leftrightarrow (a\hat{u}_{g1} + c\hat{u}_{g2}) \\ \lambda_{g2} &\leftrightarrow (c\hat{u}_{g1} + b\hat{u}_{g2}) \\ \lambda_{g3} &\leftrightarrow -c \\ \lambda_{g4} &\leftrightarrow -\frac{1}{2}a \\ \lambda_{g5} &\leftrightarrow -\frac{1}{2}b \\ u_{g0} &\leftrightarrow (\sum_{m=1}^{5} \log \gamma_{gm}) + \log P(Y = g). \end{aligned} \tag{3.28}$$

Thus, the same two-class classification problem may be approached with QDA, using two-dimensional feature vectors $u = [T_1(x)\, T_2(x)]^T$, or with SDA, using the five descriptive statistics comprising the QDA features augmented with the quadratic terms $\{T_3(x), T_4(x), T_5(x)\}$.

This result has a well-established counterpart in metric learning. It is known that the quadratic decision boundaries arising from the bivariate Gaussian class-conditional assumption may be fitted with quadratic discriminants acting on the metric features $x[1], x[2]$ (QDA), or with linear discriminants acting on the enlarged feature set $x[1]$, $x[2]$, $x[1]x[2]$, $x^2[1]$, $x^2[2]$ (LDA) [27]. Effectively, LDA estimates linear decision boundaries in the augmented feature space; these linear boundaries approximate the quadratic boundaries in the original two-dimensional feature space. In this respect, the relationship between QDA and SDA in similarity features space is analogous to the relationship between QDA and LDA in metric feature space. However, the different approaches to parameter estimation taken by QDA and SDA lead to different numerical estimates for the parameters, so the class boundaries estimated by the two techniques will be different. Therefore, one should not misunderstand the parameter correspondences (3.28) as numerical equalities.

Generally, linear decision boundaries can be estimated with lower variance than more complex decision boundaries, at the cost of increased bias. In metric space, the flexibility in choosing QDA or LDA affords the practitioner a trade-off between bias and variance [27]. Both QDA and LDA rely on the assumption that the sample features are jointly Gaussian random variables; given this assumption, choosing between QDA and LDA is merely a matter of matching model complexity to the bias and variance requirements of the particular problem being solved. However, in similarity space, choosing between QDA and SDA is not as straightforward as selecting between QDA and LDA in metric space. While in metric learning QDA and LDA may be viewed as equivalent (albeit in different feature spaces), in similarity-based learning QDA and SDA are complementary. The nature of the available data is an important consideration when choosing QDA or SDA as the most appropriate approach. This underscores the value of developing similarity-based techniques that do not rely on metric assumptions and Euclidean feature spaces: SDA creates a new set of possible classifiers for similarity-based learning.

3.3.3 Relationship of SDA to LDA

LDA is equivalent to QDA when all classes have the same estimated covariance matrix $\hat{\Sigma}$. In this case the quadratic terms are the same for all class discriminants, so they can be ignored. The expression for the linear discriminant for class g becomes:

$$D_{LDA,g}(x) = \hat{u}_g^T \hat{\Sigma}^{-1} x - \frac{1}{2} \hat{u}_g^T \hat{\Sigma}^{-1} \hat{u}_g^{-T} + \log P(Y = g). \tag{3.29}$$

Applying this expression to the analysis of feature vectors $u = [T_1(x)\, T_2(x)]^T$ from Section 3.3.2, the LDA discriminant function $D_{LDA,g}$ can be written directly from (3.25), ignoring the quadratic terms:

$$D_{LDA,g}(x) = (a\hat{u}_{g1} + c\hat{u}_{g2}) s(x, \mu_1) + (c\hat{u}_{g1} + b\hat{u}_{g2}) s(x, \mu_2) + u_{g0}, \tag{3.30}$$

where

$$u_{g0} = -\frac{1}{2} \begin{bmatrix} \hat{u}_{g1} & \hat{u}_{g2} \end{bmatrix} \begin{bmatrix} a & c \\ c & b \end{bmatrix} \begin{bmatrix} \hat{u}_{g1} \\ \hat{u}_{g2} \end{bmatrix} + \log P(Y = g).$$

The same arguments from Section 3.3.2 apply here, and LDA is related to SDA by the following parameter correspondences:

$$\begin{aligned} T_1(x) &\leftrightarrow s(x, \mu_1) \\ T_2(x) &\leftrightarrow s(x, \mu_2) \end{aligned}$$

and

$$\begin{aligned} \lambda_{g1} &\leftrightarrow (a\hat{u}_{g1} + c\hat{u}_{g2}) \\ \lambda_{g2} &\leftrightarrow (c\hat{u}_{g1} + b\hat{u}_{g2}) \\ u_{g0} &\leftrightarrow \left(\sum_{m=1}^{2} \log \gamma_{gm} \right) + \log P(Y = g). \end{aligned}$$

As for the SDA-QDA relationship, the correspondence between the parameters is not a numeric equality, but a formal assignment.

3.3.4 Relationship of SDA to Discriminant Analysis on Similarity Features

The previous sections explored the relationship between SDA and quadratic and linear discriminant analysis for a two-class problem in which the two-dimensional feature vector consists of the similarities of the test sample x to the class centroids μ_1 and μ_2. More generally, similarity-based classification problems can be turned into standard Euclidean-based learning problems by taking the $N \times 1$ vector of similarities between a test sample and the N training samples, and using it as an N-dimensional Euclidean feature vector [17, 21, 54]. This approach turns the similarity-based classification into a standard metric statistical learning problem with N training samples in an N-dimensional feature space, with the concomitant *curse of dimensionality* difficulties [27]. Duin et al. proposed dealing with the resulting curse of dimensionality problem by using a regularized linear discriminant analysis classifier on the n-dimensional feature space [17, 54]. Refer to this method as *discriminant analysis on similarity features*. Their results show that, on average over their different experiments, linear classifiers built on the similarity vectors achieve similar errors as the 1-nearest neighbor similarity-based classifier, except in cases of severe noise, where the 1-nearest neighbor has high error.

This section further analyzes discriminant analysis on similarity features, and compares it with SDA by building on the previous two-class case. In [54], the problem is formulated in terms of dissimilarities. Here, it is formulated in terms of similarities, as the results do not depend on the chosen similarity measure. The descriptive statistics are the set of similarities $s(x, z_i)$, where s is a function $\mathcal{B} \times \mathcal{B} \to \Omega$, $\Omega \subset \mathbb{R}$. The feature vector is

$$u \doteq [s(x, z_1)s(x, z_2) \dots s(x, z_N)]^T, \tag{3.31}$$

where z_i is a training sample and x the test sample.

Discriminant analysis on similarity features classifies a test sample based on the discriminant (3.29), where the ith component of the mean vector for the gth class is

estimated by ML

$$\hat{u}_g[i] = \frac{1}{n_g} \sum_{x_j \in \mathcal{X}_g} s(x_j, z_i) \tag{3.32}$$

and the pooled covariance matrix is estimated as

$$\hat{\Sigma} = \frac{1}{(N-2)} \sum_{g=1}^{2} \sum_{x_j \in \mathcal{X}_g} (u_j - \hat{u}_g)(u_j - \hat{u}_g)^T. \tag{3.33}$$

If there are N training samples, then there are $N \times N$ parameters to estimate for $\hat{\Sigma}$, which is generally ill-posed. Duin et al. suggest regularizing $\hat{\Sigma}$ by forming a convex combination between the empirical within-class pooled covariance and the identity matrix. To simplify the present analytic comparison, suppose that the true pooled covariance is the identity matrix I, and optimistically suppose that the pooled covariance is estimated perfectly, so that $\hat{\Sigma} = I$. Then the classification rule becomes: classify as class one if $D_{LDA,1} > D_{LDA,2}$, that is, if

$$\hat{u}_1^T u - \frac{1}{2}\hat{u}_1^T \hat{u}_1 > \hat{u}_2 - \frac{1}{2}\hat{u}_2^T \hat{u}_2. \tag{3.34}$$

This decision rule is based on whether the feature vector of dissimilarities u is better correlated with $\hat{u}_1$, the vector of mean distances to each training sample from class one training samples, or better correlated to $\hat{u}_2$, the vector of mean distances to each training sample from class two training samples, where these correlations are offset by the self-correlations of $\hat{u}_1$ and $\hat{u}_2$.

Consider now applying SDA using the descriptive statistics vector u. Then the log-linear SDA decision rule becomes: classify as class one if

$$\sum_{i=1}^{n} (\log(\gamma_{1i}) + s(x, z_i)\lambda_{1i}) > \sum_{i=1}^{n} (\log(\gamma_{2i}) + s(x, z_i)\lambda_{2i}), \tag{3.35}$$

or equivalently, where λ_1 is a vector with ith component λ_{1i},

$$\left(\sum_{i=1}^{n} \log(\gamma_{1i}) \right) + u^T \lambda_1 > \left(\sum_{i=1}^{n} \log(\gamma_{2i}) \right) + u^T \lambda_2. \tag{3.36}$$

Thus, using the dissimilarities u as descriptive statistics, the SDA rule given in (3.36) and the discriminant analysis on similarity features rule (3.34) (assuming the estimated covariance was the identity matrix) have the same form. The constants in both rules are due to the normalization of the underlying probability model. However, in (3.34) the tested correlations are between the test feature vector given in (3.31) and each class mean $\hat{u}$, whereas in (3.36) the tested correlations are between the test feature vector u and the parameter vector λ.

To show how the classifiers differ, consider a concrete example which compares the discriminant analysis on similarity features rule given in (3.34) to the SDA rule using the similarity-to-the-training-samples as descriptive statistics (3.36), and to SDA using the centroid descriptive statistics set $\mathcal{T}(x) = \{s(x, \mu_1), s(x, \mu_2)\}$ where the centroids μ_1, μ_2 are defined in (2.10).

Consider two classes whose elements are generated as additive noise on a prototypical set of three features, where the class one prototype is [0 0 0] and the class two prototype is [1 1 1]. Illustrative training samples were then chosen:

Class 1 training samples: [0 0 0], [0 0 1], [1 0 0], [0 1 0], [1 1 0]

Class 2 training samples: [1 1 1], [1 0 1], [1 1 0], [0 1 1], [0 0 1]

The dissimilarity metric used is Hamming distance (the number of features that differ), and all eight possible feature combinations were considered as test samples. The classifiers differ in their classification of [0 0 1] and [1 1 0], which appear as training samples in both classes. The discriminant analysis on similarity features as defined in (3.34), classifies [0 0 1] as class 2 and [1 1 0] as class 1, which are incorrect given the underlying model. The proposed SDA with (3.9) and (2.10) correctly classifies all test samples. SDA with the decision rule (3.36) incorrectly classifies [0 0 1] as class 2, but correctly classifies [1 1 0] as class 2. Over a number of similar focused simulations with simple unimodal models for each class, SDA with centroid-based similarities performed better than the SDA decision rule (3.36) or the discriminant analysis on similarity features (3.34).

3.3.5 Relationship of SDA to Support Vectors of Similarity Features

Section 2.2.6 discussed how the matrix of pairwise similarities between training samples can be used as the kernel in a SVM classifier, $K(z_i, z_j) = s(z_i, z_j)$ as long as $s(z_i, z_j)$ is symmetric and positive definite. In this case, the SVM classification rule (2.12) becomes a linear discriminant in similarity feature space. Separate the terms associated with each of the two classes $g = \{-1, +1\}$, and substitute the feature vector $u = [s(z, x_1), s(z, x_2), \ldots s(z, x_N)]^T$ of similarities of the test sample x to each training sample z_i for the kernel. The classification rule (2.12) becomes to classify as the class

$$\hat{y} = \text{sign}\left(\sum_{i=1}^{N} \hat{\alpha}_i y_i s(x, z_i) + b\right),$$

and may be written in terms of linear discriminants for the classes $g \in \{+1, -1\}$,

$$\begin{aligned} \sum_{i=1}^{N} \hat{\alpha}_i s(x, z_i) I_{y_i=1} + b_1 &> \sum_{i=1}^{N} \hat{\alpha}_i s(x, z_i) I_{y_i=-1} + b_{-1} \\ \hat{\alpha}_{g=1}^T u_{g=1} + b_1 &> \hat{\alpha}_{g=-1}^T u_{g=-1} + b_{-1} \\ D_{SVM,x}(z) &> D_{SVM,-1}(x), \end{aligned} \tag{3.37}$$

where $b_1 - b_{-1} = b$, I is the indicator function, and where the ith elements of vectors α_g, u_g and y_g are $\alpha_i I_{y_i=g}$, $s(z, x_i) I_{y_i=g}$, and $I_{y_i=g}$ respectively. The decision rule (3.37) is based on whether the feature vector of similarities u is better correlated with $\hat{\alpha}_{g=1}$, the vector of SVM coefficients associated with support vectors from class $g = 1$, or better correlated with $\hat{\alpha}_{g=-1}$, the vector of SVM coefficients associated with support vectors from class $g = -1$, where these correlations are offset by the contribution of the coefficients b_g associated with each class.

Expression (3.37) is analogous to expression (3.34). Thus, using the matrix of pairwise similarities as the SVM kernel is simply a case of discriminant analysis on similarity features, discussed in Section 3.3.4. However, the correlations in (3.37) are computed from the similarities of a test sample to the training samples associated

with one of the classes, whereas the correlations in (3.34) are computed from the similarities to the training samples in both classes.

Applying SDA to this problem leads to expression (3.36), and the observations of Section 3.3.4 apply here. To further explore the relationship between SVMs and SDA, consider the same two-class problem described in Section 3.3.2, where a sample is characterized by a feature vector of similarities to the two class centroids (3.22), and where the class labels are $y = \{-1, +1\}$. SVMs can be constructed from this description of the samples. In particular, consider the linear kernel (2.14) of Section 2.2.6 with a quadratic exponent $q = 2$. The SVM classification rule (2.12) for a test sample $v = [s(x, \mu_1)\, s(x, \mu_2)]^T$ given N training samples $u_i = [s(z_i, \mu_1)\, s(z_i, \mu_2)]^T$ becomes

$$\begin{aligned} \hat{y} &= \text{sign}\left(\sum_{i=1}^{N} \hat{\alpha}_i y_i [\langle v, u_i \rangle + 1]^2 + b\right) \\ &= \text{sign}\left(\sum_{i=1}^{N} \hat{\alpha}_i I_{y_i=1} [\langle v, u_i \rangle + 1]^2 - \sum_{i=1}^{N} \hat{\alpha}_i I_{y_1=-1} [\langle v, u_i \rangle + 1]^2 + b\right). \end{aligned} \quad (3.38)$$

Expand the kernel expression and rewrite the term associated with class label $y = g$:

$$\begin{aligned} D_{SVM,g}(x) &= \left(2\sum_{i=1}^{N} \hat{\alpha}_i I_{y_i=g} s(z_i, \mu_1)\right) s(x, \mu_1) \\ &+ \left(2\sum_{i=1}^{N} \hat{\alpha}_i I_{y_i=g} s(z_i, \mu_2)\right) s(x, \mu_2) \\ &+ \left(\sum_{i=1}^{N} \hat{\alpha}_i I_{y_i=g} s(z_i, \mu_1) s(z_i, \mu_2)\right) s(x, \mu_1) s(x, \mu_2) \\ &+ \left(\sum_{i=1}^{N} \hat{\alpha}_i I_{y_i=g} s(z_i, \mu_1)\right) s^2(x, \mu_1) \\ &+ \left(\sum_{i=1}^{N} \hat{\alpha}_i I_{y_i=g} s(z_i, \mu_2)\right) s^2(x, \mu_2) \\ &+ b_g. \end{aligned} \quad (3.39)$$

Following the same argument of Sections 3.3.2 and 3.3.3, (3.39) may be expressed in the same form as the log-linear SDA discriminant by relating the similarities to the

descriptive statistics as in (3.27), and by making the formal assignments

$$\begin{aligned}
\lambda_{g1} &\leftrightarrow 2\sum_{i=1}^{N}\hat{\alpha}_i I_{y_i=g} s(z_i,\mu_1) \\
\lambda_{g2} &\leftrightarrow 2\sum_{i=1}^{N}\hat{\alpha}_i I_{y_i=g} s(z_i,\mu_2) \\
\lambda_{g3} &\leftrightarrow \sum_{i=1}^{N}\hat{\alpha}_i I_{y_i=g} s(z_i,\mu_1) s(z_i,\mu_2) \\
\lambda_{g4} &\leftrightarrow \sum_{i=1}^{N}\hat{\alpha}_i I_{y_i=g} s(z_i,\mu_1) \\
\lambda_{g5} &\leftrightarrow \sum_{i=1}^{N}\hat{\alpha}_i I_{y_i=g} s(z_i,\mu_2) \\
u_{g0} &\leftrightarrow b_g.
\end{aligned} \tag{3.40}$$

Thus, the linear SVM kernel generates a linear discriminant expression, much like the expressions for QDA (3.25), LDA (3.30) and SDA (3.26). However, in general the SVM hyperplane coefficients $\hat{\alpha}$ computed by solving (2.13) will produce different discriminant boundaries than those produced by the other methods.

3.3.6 Relationship of SDA to PSVM

The PSVM classifier can accept any matrix as the kernel K. Thus, any pairwise similarity matrix, even non-symmetric or non-positive definite, may be used for K. Given $K(z_i, z_j) = s(z_i, z_j)$, the approach of Section 3.3.5 leads to the same linear discriminant on similarity features (3.37) result for PSVM.

Next, consider a different set of statistics for the two-class case where the kernel is a $N \times 2$ matrix of similarities of the N training samples z_i to the class centroids μ_1 and μ_2. The ith row of K is the vector $u_i = [s(z_i, \mu_1)\, s(z_i, \mu_2)]^T$ and a test sample x is described by the vector $v = [s(x, \mu_1)\, s(x, \mu_2)]^T$. Following the approach in Section 3.3.5 one may write the PSVM discriminant for class g from the PSVM classification

rule (2.15),

$$D_{PSVM,g}(x) = \hat{\alpha}_g s(x, \mu_g) + b_g, \tag{3.41}$$

where $b_g = \frac{1}{N} \sum_{i=1}^{N} y_i I_{y_i=g}$.

A correspondence between SDA and PSVM analogous to the correspondences between SDA, QDA, LDA and SVM cannot be made in this case. The value of the class g PSVM linear discriminant for a test point x is simply the similarity of x to the class centroid μ_g, scaled by the factor $\hat{\alpha}$ and offset by the fraction of training samples from class g. Unlike SDA, QDA, LDA, and SVMs, here the discriminant for class g depends only on the similarity to the class g centroid. For the other classifiers the discriminant for class g depends also on the similarities to the centroids of the classes other than g. In this sense, for this case, the PSVM is more like a scaled nearest-centroid (most-similar centroid) classifier than a discriminant-based classifier. In spite of this difference, the PSVM classifier can estimate classification boundaries very close to the boundaries produced by QDA, LDA, and SVM, as discussed in the following section.

3.3.7 Comparison of Discriminants for Different Classifiers

Sections 3.3.1-3.3.6 described the relationship between the SDA and metric classifiers. In particular, it was shown that when the sample features are the similarities of the samples to the centroids of each class, the expressions for the discriminants produced by QDA, LDA, and SVM are formally the same as the log-linear SDA discriminant. The discriminants for these classifiers may be written as linear combinations of similarity features, but in general the linear combination coefficients differ from one classifier to the other, because each classifier estimates the coefficients differently. The PSVM was shown to produce different discriminant expressions, more like weighted nearest centroid similarity than the log-linear SDA discriminants. This section examines the classification boundaries produced by various classifiers by building on the examples

in the previous sections, in which the feature vectors consist of the sample similarities to the class centroids, $u = [s(x, \mu_1)\ s(x, \mu_2)]$. In this example, LDA and QDA do not apply because the class-dependent covariance matrices are singular. SVM, PSVM, naive Bayes, and SDA produce class boundaries very close to each other.

Each feature vector u is formed as the similarity between a random binary vector and two fixed binary vectors, which are the class centroids. For this example, class one is characterized by the centroid $\mu_1 = [0\,0\,0]$ and class two by centroid $\mu_2 = [1\,1\,1]$. The class one samples are perturbations of μ_1, where each element is perturbed (flipped) with probability $p_1 = 1/3$; for class two the probability that each element of μ_2 is flipped is $p_2 = 1/4$. The data consist of 1000 binary vector realizations drawn from each class with prior probability $P(Y = g) = 1/2$, where $g = \{1, 2\}$. The similarity function is the Hamming (or counting) similarity, which is the number of identical features shared by two vectors. In this example, the similarity takes on the values $\{0, 1, 2, 3\}$.

In this very simple example QDA and LDA cannot be applied because the covariance matrices are singular. The elements of the feature of vector $[s(x, \mu_1)\ s(x, \mu_2)]$ are linearly dependent such that $s(x, \mu_1) = 3 - s(x, \mu_2)$, due to the fact that the centroids μ_1 and μ_2 do not have any element in common. Thus, in spite of the problem's simplicity, the standard QDA and LDA techniques are not applicable. SDA, SVM, PSVM, and naive Bayes, however, may still be applied because their discriminants do not rely on covariance matrices.

The discriminant parameters for (3.37), (3.41), and (3.26) are estimated from the data using each classifier's estimation method: for SVM and PSVM the coefficients $\hat{\alpha}$ are computed by numerical optimization of the two corresponding objective functions (3.37) and (3.41); for SDA, λ_{gm} and γ_{gm}, $m = 1, 2, \ldots 5$ are computed numerically from the constraints (3.15). For naive Bayes, the class-conditional feature pmfs are computed using ML estimation. The class boundaries in the dissimilarity feature space are the curves defined by $D_{SDA,1} =, D_{SDA,2},\ D_{SVM,1} =, D_{SVM,2}$, etc. These

curves are plotted in Figure 3.3. All classifiers produce similar classification boundaries and, for this example, produce identical class estimates. The small differences in the boundaries are due to the different numerical approaches used to estimate the coefficients of the linear discriminants.

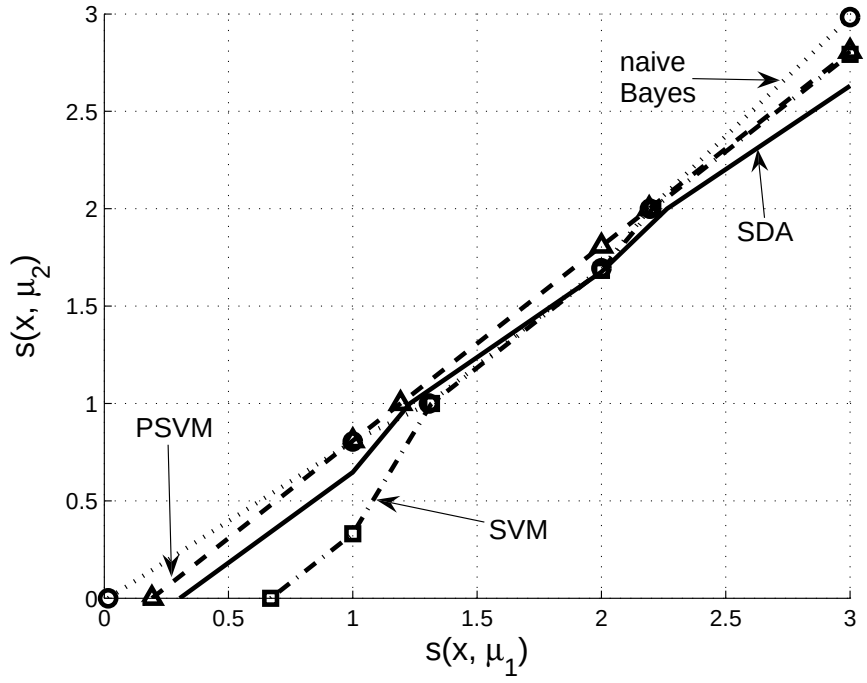

Figure 3.3: Class boundaries produced by SDA (-), SVM (□), PSVM (△), and naive Bayes (o) applied to the similarity.

3.4 Mixed Numerical and Categorical Features

In some applications, samples are characterized by mixed – that is both numerical and categorical – features. One way to deal with mixed features is to use similarity functions that can handle both types of features and then use SDA for classification. Examples of these similarity functions are the heterogeneous VDM [13] and the residual entropy [10]. Given a similarity measure appropriate for mixed features, any of

the similarity-based classification methods discussed thus far can be applied.

A different way to deal with mixed features is to build mixed models that combine standard Euclidean feature space-based probability models with similarity-based models. This alternative method is briefly discussed below, and can be further pursued in future work. Consider the sample x, which is a random realization of the random sample X composed of random features $X[1], X[2], \ldots, X[d]$. Assume without loss of generality that the first j features of x are categorical, and its last $d - j$ features are numerical: x is written as the concatenation of a categorical feature vector and a numeric feature vector, $x = [x_{cat}\, x_{num}]$. Using the chain rule of probability, the class-conditional probability of sample x may be written as the product of two terms, one based only on the categorical features and the other based on the numerical features additionally conditioned on the categorical features:

$$\begin{aligned} P(X = x|Y = g) &= P(X_{1,\ldots,j} = x_{cat}, X_{j+1,\ldots,d} = x_{num}|Y = g) \\ &= P(X_{j+1,\ldots,d} = x_{num}|X_{1,\ldots,j} = x_{cat}, Y = g)P(X_{1,\ldots,d} = x_{cat}|Y = g). \end{aligned} \tag{3.42}$$

The class-conditional and categorical feature-conditional term $P(X_{j+1,\ldots,d} = x_{num}|X_{1,\ldots,j} = x_{cat}, Y = g)$ for the numerical features may be estimated using any known probability model estimation technique. For example, Gaussian models can be used [67]. The categorical class-conditional probability may also be estimated with known techniques. For example, a multinomial naive Bayes model can be easily estimated. Or, SDA could be used to model the class-conditional probabilities with similarity-based techniques. The same reasoning of Section 3.1 allows using the descriptive statistics $\mathcal{T}(x_{cat})$ instead of the categorical features, so that the test sample $x = [\mathcal{T}(x_{cat})\, x_{num}]$ and one substitutes $P(\mathcal{T}(x_{cat})|Y = g)$ for $P(x_{cat}|Y = g)$. More specifically, one could use a relevant similarity function to measure the pairwise similarities between samples based only on the x_{cat} features, and analogously compute the class centroids μ_h based on x_{cat}. Applying Bayes rule to (3.42) the mixed Euclidean- and similarity-based

classifier rule is: classify x as the class $\hat{y}$ which maximizes the expression

$$\arg \max_{f=1,\ldots,G} \sum_{g=1}^{G} C(f,g)\hat{P}(g)\hat{P}(X_{j+1,\ldots,d} = x_{num} | \mathcal{T}(x_{cat}) = T_{1,\ldots,m}(x_{cat}), Y = g) \times \prod_{m=1}^{M} \gamma_{gm} e^{\lambda_{gm} T_m(x_{cat})} \quad (3.43)$$

A potential problem with (3.43) is the rapid growth of the number of conditional probability models that must be estimated to model the continuous features given different conditions of class and categorical features. If the ith categorical feature takes on c_i values, then the number of models to estimate is $\prod_i c_i \times G$, which can grow rapidly. When this happens, one may be left with too few samples per class to accurately estimate the model parameters.

To simplify the classifier complexity and further reduce the estimation variance (at some increase in model bias) one could make the additional assumption that the numerical features and categorical features are class-conditionally independent:

$$\arg \max_{f=1,\ldots,G} \sum_{g=1}^{G} C(f,g)\hat{P}(g)\hat{P}(X_{j+1,\ldots,d} = x_{num} | Y = g) \prod_{m=1}^{M} \gamma_{gm} e^{\lambda_{gm} T_m(x_{cat})}. \quad (3.44)$$

Chapter 4

LOCAL SIMILARITY DISCRIMINANT ANALYSIS

This chapter introduces *local SDA*, a similarity-based classifier that is both generative and local. An advantage of generative classifiers is their interpretability: classes are modeled by conditional probability distributions which are assumed to have generated the observed data. An advantage of local classifiers it that they reduce the estimation bias problem which affects generative classifiers. Local SDA combines the qualities of both generative and local classifiers.

For the SDA classifier described in Chapter 3, the class-conditional generative distributions are exponentials which model the similarities between samples – or more generally the descriptive statistics of the sample. The exponentials are the maximum entropy distributions subject to constraints on the mean values of the similarities. However, when the underlying distributions are complex, a particular set of empirical statistics may fail to capture the necessary information about a sample's class membership. In fact, in SDA, constraining the means of the class-conditional distributions may result in too much model bias, just as the QDA model of one Gaussian per class causes model bias [27]. In standard metric learning, one way to address the bias problem while retaining the advantages of a generative approach is to form more flexible Gaussian mixture models. In similarity-based learning, mixture models may also be formed, as discussed in Chapter 5.

This chapter addresses the bias in SDA by using local classifiers in similarity space. In metric learning, one way to avoid the bias problem is to use local classifiers, e.g. k-NN, which classify test samples based on the class labels of their nearest neighbors. Local classifiers do not estimate probabilistic models for the sample classes

and consequently lack the interpretability of generative models. Even so, they provide an intuitive framework for classification through the concepts of nearest-neighbor and neighborhood. In this chapter, SDA is applied to a local neighborhood about the test sample. The resulting *local SDA* classifier trades-off model bias and estimation variance depending on the neighborhood size, while retaining the power of a generative classifier. To the author's knowledge, local SDA is the first example of a classifier that is both generative and local. The only arguable contender is the local nearest-mean classifier [46, 47] for metric learning; however that classifier was not proposed as a generative model. In Section 4.1, local SDA is introduced as a straightforward local version of SDA. In Section 4.2 local SDA is proven to be a consistent classifier, in the sense that its error rate asymptotically converges to the Bayes error rate, which is the best possible error rate attainable by a classifier.

4.1 Local SDA

Local SDA is a straightforward variation of SDA. The local SDA classifier model is that all of the relevant information about classifying a test sample x depends only on the k nearest (most similar) training samples to x. Thus, the local SDA classifier computes the descriptive statistics from a neighborhood of a test sample. More specifically, local SDA is a log-linear generative classifier that models the probability distribution of the similarity $s(x, \mu_h)$ between the test sample x and the class centroids $\{\mu_h\}$, just like SDA. Unlike SDA, the class centroids, the class-conditional similarity probability models, and the estimates of the class priors are computed from a neighborhood of the test sample rather than from the entire training set. Thus, the class centroid definition (2.10) used for SDA still holds for local SDA; one simply redefines $\mathcal{X}_h$ as the subset of the k nearest neighbors from class h. The class priors are estimated using normalized class membership counts of the neighbors of x, that is $\hat{P}(Y = h) = |\mathcal{X}_h|/k$. The mean similarity constraints (3.4) for the SDA maximum entropy optimization are formally the same for local SDA, except that the mean is

computed from the neighbors of test sample x rather than the whole training set. Thus, the optimized parameters λ_{gh} and γ_{gh} are local. Given the set of local class centroids $\{\mu_h\}$, the local class priors $\hat{P}(Y = g)$, and the local class-conditional model parameters γ_{gh} the local SDA classification rule is identical to the SDA rule (3.7):

$$\arg \max_{f=1,\ldots,G} \sum_{g=1}^{G} C(f,g) \left(\prod_{h=1}^{G} \gamma_{gh} e^{\lambda_{gh} s(x,\mu_h)} \right) \hat{P}(Y = g).$$

A problem can occur if the hth class has few training samples in the neighborhood of test sample x. In this case, the local SDA model for class h is difficult to estimate. To avoid this problem, if the number of local training samples in any of the classes is very small, for example $n_h < 3$, the local SDA classifier reverts to the local nearest centroid classifier discussed in Section 2.2.3. If $n_h = 0$ so that $\mathcal{X}_h$ is the empty set, then the probability of class h is locally zero, and that class is not considered in the classification rule (3.7). This strategy enables local SDA to gracefully handle small k and very small class priors.

Local classification algorithms have traditionally been weighted voting methods, including classifying with local linear regression, which can be formulated as a weighted voting method [27]. These methods are by their nature non-parametric and their use arises in situations when the available training samples are too few to accurately build class models. On the other hand, it is known that the number of training samples required by nonparametric classifiers to achieve low error rates grows exponentially with the number of features [47]. Thus, when only small training sets are available, nonparametric classifiers are negatively impacted by outliers. In 2000, Mitani and Hamamoto [46,47] were the first ones to propose a classifier that is both model-based and local. However, they did not develop it as a local generative method; instead, they proposed the classifier as a local weighted-distance method. Their nearest-means classifier can be interpreted as a local QDA classifier with identity covariances. In experiments with simulated and real data sets, the local nearest-means classifier was competitive with, and often better than, nearest neighbor, the Parzen classifier, and

an artificial neural network, especially for small training sets and for high dimensional problems.

Local nearest-means differs from local SDA in several aspects. First, the classifier by Mitani and Hamamoto in [47] learns a metric problem, not a similarity problem: the class prototypes are the local class-conditional means of the features and a weighted Euclidean distance is used to classify a test sample as the class of its nearest class mean. Second, the neighborhood definition is different than the usual k nearest neighbors: they select k nearest neighbors from each class, so that the total neighborhood size is $k \times G$.

More recently, it was proposed to apply a support vector machine to the k nearest neighbors of the test sample [80]. The SVM-KNN method was developed to address the robustness and dimensionality concerns that afflict nearest neighbors and SVMs. Similarly to the nearest-means classifier, the SVM-KNN is a hybrid local and global classifier developed to mitigate the high variance typical of nearest neighbor methods and the curse-of-dimensionality. However, unlike the nearest means classifier of Mitani and Hamamoto, which is rooted in Euclidean space, the SVM-KNN can be used with any similarity function, as it assumes that the class information about the samples is captured by their pairwise similarities without reference to the underlying feature space. Experiments on benchmark datasets using various similarity functions showed that SVM-KNN outperforms k-NN and its variants especially for cases with small training sets and large number of classes. SVM-KNN differs from local SDA because it is not a generative classifier.

Finally, note that different definitions of neighborhood may be used with local SDA. One could use the Mitani and Hamamoto [47] definition described above, or radius-based definitions. For example, the neighborhood of a test sample x may be defined as all the samples that fall within a factor of $1 + \alpha$ of its similarity to its most similar neighbor, and α is cross-validated. This work employs the traditional definition of neighborhood, as the k nearest neighbors. Results for local SDA are in

Chapter 6.

4.2 Consistency of the Local SDA Classifier

Generative classifiers with a finite number of model parameters, such as QDA or SDA, will not asymptotically converge to the Bayes classifier due to the model bias. This section shows that, like k-NN, the local SDA classifier is consistent such that its expected classification error $E[L]$ converges to the Bayes error rate L^* under the usual asymptotic assumptions that the number of training samples $N \to \infty$, the neighborhood size $k \to \infty$, but that the neighborhood size grows relatively slowly such that $k/N \to 0$. First a lemma is proven that will be used in the proof of the local SDA consistency theorem. Also, the known result that k-NN is a consistent classifier is reviewed in terms of similarity.

Let the similarity function be $s : \mathcal{B} \times \mathcal{B} \to \Omega$, where $\Omega \subset \mathbb{R}$ is discrete and let the largest element of Ω be termed s_{max}. Let X be a test sample and let the training samples $\{X_1, X_2, \ldots, X_N\}$ be drawn identically and independently. Re-order the training samples according to decreasing similarity and label them $\{Z_1, Z_2, \ldots Z_N\}$ such that Z_k is the kth most similar neighbor of X.

Lemma 1. *Suppose $s(x, Z) = s_{max}$ if and only if $x = Z$ and $P(s(x, Z) = s_{max}) > 0$ where Z is a random training sample. Then $P(s(x, Z_k) = s_{max}) \to 1$ as $k, N \to \infty$ and $k/N \to 0$.*

Proof: The proof is by contradiction and is similar to the proof of Lemma 5.1 in [15]. Note that $s(x, Z_k) \neq s_{max}$ if and only if

$$\frac{1}{N} \sum_{i=1}^{N} I_{\{s(x,Z_i)=s_{max}\}} < \frac{k}{N}, \tag{4.1}$$

because if there are less than k training samples whose similarity to x is s_{max}, the similarity of the kth training sample to x cannot be s_{max}. The left-hand side of (4.1) converges to $P(s(x, Z) = s_{max})$ as $N \to \infty$ with probability one by the strong law of

large numbers, and by assumption $P(s(x, Z) = s_{max}) > 0$. However, the right-hand side of (4.1) converges to 0 by assumption. Thus, assuming $s(x, Z_k) \neq s_{max}$ leads to a contradiction in the limit. Therefore, it must be that $s(x, Z_k) = s_{max}$.

Theorem 1. *Assume the conditions of Lemma 1. Define L to be the probability of error for test sample X given the training sample and label pairs* $\{(Z_1, Y_1), (Z_2, Y_2), \ldots, (Z_N, Y_N)\}$, *and let* L^* *be the Bayes error. If* $k, N \to \infty$ *and* $k/N \to 0$, *then for the local SDA classifier* $E[L] \to L^*$.

Proof: By Lemma 1, $s(x, Z_i) = s_{max}$ for $i \leq k$ in the limit as $N \to \infty$, and thus in the limit the centroid μ_h of the subset of the k neighbors that are from class h must satisfy $s(x, \mu_h) = s_{max}$, for every class h which is represented by at least one sample in the k neighbors. By definition of the local SDA algorithm, any class $\bar{h}$ that does not have at least one sample in the k neighbors is assigned the class prior probability $P(Y = \bar{h}) = 0$, so it is effectively eliminated from the possible classification outcomes. Then, the constraint (3.4) on the expected value of the class-conditional similarity for every class g that is represented in the k neighbors of x is

$$E_{P(s(x,\mu_h)|Y=g)}[s(X, \mu_h)] = s_{max}, \tag{4.2}$$

which is solved by the pmf $P(s(x, \mu_h)|Y = g) = 1$ if $s(x, \mu_h) = s_{max}$, and zero otherwise. Thus the local SDA classifier (3.7) becomes

$$\hat{y} = \arg \max_{g=1,\ldots G} \hat{P}(Y = g), \tag{4.3}$$

where the estimated probability of each class $\hat{P}(Y = g)$ is calculated using a maximum likelihood estimate of the class probabilities for the neighborhood. Then, $\hat{P}(Y = g) \to P(Y = g|x)$ as $k \to \infty$ with probability one by the strong law of large numbers. Thus the local SDA classifier converges to the Bayes classifier, and the local SDA average error $E[L] \to L^*$.

The known result that k-NN is a consistent classifier can be stated in terms of similarity as a direct consequence of Lemma 1:

Lemma 2. *Assume the conditions of Lemma 1 and define L and L^* as in Theorem 1. For the similarity-based k-NN classifier $E[L] \to L^*$.*

Proof. It follows directly from Lemma 1 that within the size-k neighborhood of x, $Z_i = x$ for $i \leq k$. Thus, the k-NN classifier (2.6) estimates the most frequent class among the k samples maximally similar to x:

$$\begin{aligned} \hat{y} &= \arg \max_{g=1,\ldots,G} \sum_{i=1}^{k} I(Y_i = g) \\ &= \hat{P}(Y = g). \end{aligned}$$

The summation converges to the class prior $P(Y = g|x)$ as $k \to \infty$ with probability one by the strong law of large numbers, and the k-NN classifier becomes that in (4.3). Thus the similarity-based k-NN classifier is consistent.

Chapter 5

MIXTURE MODELS FOR SIMILARITY DISCRIMINANT ANALYSIS

This chapter discusses generative mixture models for similarity-based learning. The mixture similarity discriminant analysis (*mixture SDA*) classifier draws from the well-established metric learning mixture model research and generalizes the SDA classifier of Chapter 3. Section 5.1 reviews mixture models from metric learning, focusing on the popular Gaussian mixture models (GMMs). The steps in the well-known expectation-maximization (EM) algorithm for estimating GMM parameters are also reviewed. Section 5.2 introduces SDA mixture models as similarity-based analogs of GMMs which generalize SDA. Section 5.3 discusses how mixture SDA parameters are learned using an EM algorithm which parallels the standard EM approach for GMM parameter estimation.

5.1 Review of Mixture Models for Classification in Metric Spaces

In metric learning, mixture models for class-conditional distributions are a robust but flexible generative approach to classification [27]. The probability of a test sample x conditioned on its class label $Y = h$ is modeled as a weighted sum of c_h probability components $P_l(X = x|Y = h)$:

$$P(X = x|Y = h) = \sum_{l=1}^{c_h} w_{hl} P_l(X = x|Y = h), \tag{5.1}$$

where $\sum_{l=1}^{c_h} w_{hl} = 1$, and $w_{hl} > 0$. The classification rule is to classify as

$$\hat{y} = \arg \max_{f=1...G} \sum_{h=1}^{G} C(f, h) P(X = x|Y = h), \tag{5.2}$$

where G is the number of possible classes.

Typically, the shape of each component $P_l(X = x|Y = h)$ is assumed a priori. Gaussian mixture models (GMMs) assume that each mixture component $P_l(X = x|Y = h) = \mathcal{N}(u_{hl}, \Sigma_{hl})$, where $\mathcal{N}(u_{hl}, \Sigma_{hl})$ is a Gaussian probability distribution with mean u_{hl} and covariance Σ_{hl}. For each class h, the mixture parameters $\{w_{hl}\}$, $\{u_{hl}\}$, and $\{\Sigma_{hl}\}$ are estimated from training data. For GMMs, the standard estimation method is the well-known expectation-maximization (EM) algorithm [6,16,27], which leads to maximum-likelihood (ML) estimates for the parameters. EM iteratively maximizes the expected log-likelihood of the training data, given current estimates of the desired model parameters. During each iteration, EM executes two steps: the expectation (E) step and the maximization (M) step. The E step computes the *responsibilities* for each sample, that is the contribution of each Gaussian component to the sample's total posterior probability given by the mixture (5.1). The M step finds new values of the desired parameters which maximize the expected log-likelihood of the data. The new values are computed by taking the partial derivatives of the expected log-likelihood with respect to the parameters and setting them to zero. These new parameter estimates are used in the E step at the next iteration of the EM algorithm.

For GMMs, the EM algorithm gives explicit expressions for updating the component weights, means and covariances [27]. Denote by C a random component of the Gaussian mixture and by $P(C = l|X = z_i, Y = h)$ the responsibility of the lth Gaussian component of class h for the training sample $z_i \in \mathcal{X}_h$, $i = 1, 2 \ldots n_h$. The EM algorithm for GMMs is:

1. Initialize the parameters $\{w_{hl}\}$, $\{u_{hl}\}$, and $\{\Sigma_{hl}\}$.

2. E step: compute the responsibilities

$$P(C = l|X = z_i, Y = h) = \frac{w_{hl} P_l(X = z_i|Y = h)}{\sum_{l=1}^{c_h} w_{hl} P_l(X = z_i|Y = h)}. \tag{5.3}$$

3. M step: compute model parameters

$$u_{hl} = \frac{\sum_{i=1}^{n_h} P(C = l|X = z_i, Y = h) z_i}{\sum_{i=1}^{N} P(C = l|X = z_i, Y = h)}, \tag{5.4}$$

$$\Sigma_{hl} = \frac{\sum_{i=1}^{N} P(C = l|X = z_i, Y = h)(z_i - u_{hl})(z_i - u_{hl})^T}{\sum_{i=1}^{n_h} P(C = l|X = z_i, Y = h)}, \tag{5.5}$$

$$w_{hl} = \frac{\sum_{i=1}^{n_h} P(C = l|X = z_i, Y = h)}{n_h}. \tag{5.6}$$

4. Iterate E and M steps until convergence criterion is satisfied.

The parameters may be initialized in several ways. A common initialization randomly assigns one of c_h training samples $z_i \in \mathcal{X}_h$ to each component mean vector u_{hl}, sets all the component covariance matrices Σ_{hl} equal to the overall within-class sample covariance Σ_h, and uniformly initializes the component weights $w_{hl} = 1/c_h$ [27]. Another common strategy is to run the K-means algorithm to find the initial values of the parameters. The component means are assigned to the K-means centroids, each component covariance matrix is set to the sample covariances of each K-means cluster, and the component weights are initialized to the fractions of training samples in each cluster [36].

The number of components c_h in each mixture may be optimized using penalized likelihood methods such as the Bayesian information criterion (BIC) [11, 27, 63], the minimum description length (MDL) [14,44], or the Akaike information criterion (AIC) [1]. Often, in practice the number of components is cross-validated by independently training several class-conditional models with different number of components and selecting the set of number of components $\{c_h\}$ for $h = 1, \ldots, G$ which gives the best classification performance.

The EM algorithm for mixture SDA closely parallels the EM algorithm for GMMs reviewed above. In particular, the expression for the component weights of an SDA mixture model is identical to (5.6); the other mixture SDA parameters are computed by solving a constrained optimization problem, where the constraints are empirical

weighted averages of observed similarities. The mixture SDA parameter estimation procedure is discussed in detail in Section 5.3.

GMMs have been successful in a variety of different statistical learning applications [27]. In the area of speaker identification Gaussian mixtures have been successfully applied to model each speaker's vocal features [58]. More recently, GMMs have been applied to the problem of classifying computational predictions of three-dimensional protein structures according to how likely it is that the modeled structure might appear in nature [11]. Genomic data, such as microarray gene expression data, can also be succesfully analyzed with GMMs [69]. In the area of image processing, GMMs have successfully been employed to segment aerial images [56] and to classify small objects [7]. GMMs have found such broad applicability because they form smooth approximations of arbitrary distributions by weighted sums of Gaussian functions. Thus, they are well suited to model non-trivial feature spaces which exhibit multi-modal distributions, such as the feature spaces which arise in many real-world applications. For such multi-modal cases, modeling a class distribution with a single Gaussian component, for example using LDA or QDA, produces class-conditional models that are too biased for useful classification.

5.2 Mixture SDA

Like LDA and QDA, basic SDA may be too biased if the similarity space – or more generally the descriptive statistics space – is multi-modal. In analogy to metric space mixture models, the bias problem in similarity space may be alleviated by generalizing the SDA formulation of Chapter 3 with similarity-based mixture models. In the *mixture SDA* models, the class-conditional probability distribution of the descriptive statistics $\mathcal{T}(x)$ for a test sample x is modeled as a weighted sum of exponential components. Generalizing the single centroid-based SDA classifier of Section 3.1, and drawing from the metric mixture models reviewed in Section 5.1, each class h is characterized by c_h centroids $\{\mu_{hl}\}$. The descriptive statistics for test sample x are

its similarities to the centroids of class h, $\{s(x, \mu_{h1}), s(x, \mu_{h2}), \ldots, s(x, \mu_{hc_h})\}$, for each class h. The mixture SDA model for the probability of the similarities, assuming that test sample x is drawn from class g, is written as

$$P(s(x, \mu_{h1}), s(x, \mu_{h2}), \ldots, s(x, \mu_{hc_h})|Y = g) = \sum_{l=1}^{c_h} w_{ghl}\gamma_{ghl}e^{\lambda_{ghl}s(x,\mu_{hl})}, \tag{5.7}$$

where $\sum_{l=1}^{c_h} w_{ghl} = 1$ and $w_{ghl} > 0$. Then, the SDA classification rule (3.7) for mixture SDA becomes to classify x as the class $\hat{y}$ that solves the maximum a posteriori problem

$$\arg \max_{f=1,\ldots,G} \sum_{g=1}^{G} C(f, g) \left(\prod_{h=1}^{G} \sum_{l=1}^{c_h} w_{ghl}\gamma_{ghl}e^{\lambda_{ghl}s(x,\mu_{hl})} \right) P(Y = g). \tag{5.8}$$

Note how the mixture SDA generative model (5.7) parallels the metric mixture formulation (5.1), with the exponentials $\gamma_{ghl}e^{\lambda_{ghl}s(x,\mu_{hl})}$ in place of $P_l(x|y)$. However, there are deep differences between mixture SDA and metric mixture models. In metric learning, the mixtures model the underlying generative probability distributions of the features. Due to the curse of dimensionality, high-dimensional, multi-modal feature spaces require many training samples for robust model parameter estimation. For example, for d features, GMMs require that a $d \times 1$ mean vector and a $d \times d$ covariance matrix be estimated for each component in each class, for a total of $c_h \times (d^2 + 3d)/2$ parameters per mixture. Constraining each Gaussian covariance to be diagonal, at the cost of an increased number of mixture components, alleviates the robust estimation problem, but does not solve it [58].

When relatively few training samples are available, robust parameter estimation becomes particularly difficult. In similarity-based learning the modeled quantity is the similarity of a sample to a class centroid. The estimation problem is essentially univariate and reduces to estimating the exponent λ_{ghl} in each component of the mixture, for a total of $c_h \times G \times 2$ parameters per mixture (the scaling parameter γ_{ghl} follows trivially). This simpler classifier architecture allows robust parameter estimation from smaller training set depending on the number of centroids per class, or, more generally, the number of descriptive statistics.

Another major difference between mixture SDA and metric mixture models is in the number of class-conditional probability models that must be estimated. In metric learning, G mixtures are estimated, one for each of the G possible classes from which a sample x may be drawn. In mixture SDA, G^2 mixture models are estimated. Each sample x is hypothesized drawn from class $g = 1, 2, \ldots G$, and its similarities to each of the G classes are modeled by the mixture (5.7), with $h = 1, 2, \ldots G$. When the number of classes grows, or when the number of components in each mixture model grows, the quadratic growth in the number of needed models presents a challenge in robust parameter estimation, especially when the number of available training samples is relatively small. However, this problem is mitigated by the fact that the component SDA parameters may be robustly estimated with smaller training sets than in metric mixture models due to the simpler, univariate estimation problem at the heart of SDA classification. The next section discusses the mixture SDA parameter estimation procedure.

5.3 Estimating the Parameters for Mixture SDA Models

Computing the SDA mixture model for the similarities of samples $x \in \mathcal{X}_g$ to class h requires estimating the number of components c_h, the component centroids $\{\mu_{hl}\}$, the component weights $\{w_{ghl}\}$ and the component SDA parameters $\{\lambda_{ghl}\}$ and $\{\gamma_{ghl}\}$. This section describes an EM algorithm for estimating these mixture parameters. The algorithm parallels the EM approach for estimating GMM parameters described in Section 5.1; it is first summarized below, and then explained in detail in the following sections.

Let $\theta_{gh} = \{\{w_{ghl}\}, \{\gamma_{ghl}\}, \{\lambda_{ghl}\}\}$ for $l = 1, 2 \ldots c_h$ be the set of parameters for the class h mixture model to be estimated under the assumption that the training samples z_i, for $i = 1, 2, \ldots n_g$ are drawn identically and independently. Denote by C a random component of the mixture and by $P(C = l|s(z_i, \mu_{hl}), \theta_{gh})$ the responsibility [27] of the lth component for the ith training sample similarity $s(z_i, \mu_{hl})$. Also write

$P(s(z_i, \mu_{hl})|C = l, \theta_{gh}) = \gamma_{ghl} e^{\lambda_{ghl} s(z_i, \mu_{hl})}$. The proposed EM algorithm for mixture SDA is

1. Compute the centroids $\{\mu_{hl}\}$ with K-medoids algorithm.

2. Initialize the parameters $\{w_{ghl}\}$ and the components $P(s(z_i, \mu_{hl})|C = l, \theta_{gh})$.

3. E step: compute the responsibilities

$$P(C = l|s(z_i, \mu_{hl}), \theta_{gh}) = \frac{w_{ghl} P(s(z_i, \mu_{hl})|C = l, \theta_{gh})}{\sum_{l=1}^{c_h} w_{ghl} P(s(z_i, \mu_{hl})|C = l, \theta_{gh})}. \tag{5.9}$$

4. M step: compute model parameters

   (a) Find the λ_{ghl} which solves

$$E_{P(\mathcal{T}(x)|Y=g)}[s(X, \mu_{hl})] = \frac{\sum_{i=1}^{n_g} s(z_i, \mu_{hl}) P(C = l|s(z_i, \mu_{hl}), \theta_{gh})}{\sum_{i=1}^{n_g} P(C = l|s(z_i, \mu_{hl}), \theta_{gh})}. \tag{5.10}$$

   (b) Compute the corresponding scaling factor

$$\gamma_{ghl} = \frac{1}{\sum\limits_{s(X, \mu_{hl}) \in \Omega} e^{\lambda_{ghl} s(X, \mu_{hl})}}. \tag{5.11}$$

   (c) Compute the component weights

$$w_{ghl} = \frac{1}{n_g} \sum_{i=1}^{n_g} P(C = l|s(z_i, \mu_{hl}), \theta_{gh}). \tag{5.12}$$

5. Repeat E and M steps until convergence criterion is satisfied.

Note that, just like EM for GMMs, the EM algorithm for mixture SDA involves iterating the E step, which estimates the responsibilities, and the M step, which estimates the parameters that maximize the expected log-likelihood of the training data. At each iteration of the M step, the explicit expression (5.12) updates the component weights. However, unlike EM for GMMs, the update expression for the

component parameters (5.10) is implicit and must be solved numerically. Another difference between the GMM and SDA EM algorithms is in how the centroids are estimated. For GMMs, the component means $\{u_{hl}\}$, which are the metric centroids, are updated at each iteration of the M step. For mixture SDA, the centroids $\{\mu_{hl}\}$ are estimated at the beginning of the algorithm and kept constant throughout the iterations.

The update expressions for the mixture SDA parameters are derived from the expression of the expected log-likelihood of the observed similarities. A standard assumption in EM is that the observed data are independent and identically distributed given the class and mixture component. For mixture SDA, this assumption means that the training sample similarities $\{\mathcal{T}_g(z_i)\} = \{s(z_i, \mu_{hl})\}$, $z_i \in \mathcal{X}_g$ to the component centroids are identically distributed and conditionally independent given the lth class component. Then, the expected log-likelihood of $\{\mathcal{T}_g(z_i)\}$ is

$$L(\{\mathcal{T}_g(z_i)\}|\theta_{gh}) = \sum_{i=1}^{n_g}\sum_{h=1}^{G}\sum_{l=1}^{c_h} \log\left(w_{ghl}\gamma_{ghl}e^{\lambda_{ghl}s(z_i,\mu_{hl})}\right) P(C = l|s(z_i, \mu_{hl}), \theta_{gh}). \tag{5.13}$$

Using the properties of the logarithm and rearranging the terms, $L(\{\mathcal{T}_g(z_i)\}|\theta_{gh})$ splits into the terms depending on w_{ghl} and the terms depending on λ_{ghl} and γ_{ghl}:

$$\begin{aligned} L(\{\mathcal{T}_g(z_i)\}|\theta_{gh}) &= \sum_{i=1}^{n_g}\sum_{h=1}^{G}\sum_{l=1}^{c_h} \log(w_{ghl}) P(C = l|s(z_i, \mu_{hl}), \theta_{gh}) \\ &+ \sum_{i=1}^{n_g}\sum_{h=1}^{G}\sum_{l=1}^{c_h} \log(\gamma_{ghl}) P(C = l|s(z_i, \mu_{hl}), \theta_{gh}) \\ &+ \sum_{i=1}^{n_g}\sum_{h=1}^{G}\sum_{l=1}^{c_h} \lambda_{ghl} s(z_i, \mu_{hl}) P(C = l|s(z_i, \mu_{hl}), \theta_{gh}). \end{aligned} \tag{5.14}$$

The standard EM approach to maximizing (5.14) is to set its partial derivatives with respect to the parameters to zero and solve the resulting equations. This is the approach adopted here for estimating the mixture SDA parameters θ_{gh} for all g, h.

The derivation of the expression for the component weights $\{w_{ghl}\}$ follows directly

from (5.14); both the derivation of and the final expression for the component weights are identical to the metric mixtures case. Section 5.3.1 re-derives the well-known expression for w_{ghl}.

Applying the EM approach, however, does not lead to explicit expressions for $\{\lambda_{ghl}\}$ and $\{\gamma_{ghl}\}$. Instead, it leads to many single-parameter constraint expressions for the mean similarities of the training data to the mixture component centroids. These expressions are solved with the same numerical solver used in the single-centroid SDA classifier of Section 3.1. Section 5.3.2 derives the constraints from the EM approach and shows that they are weighted versions of the constraints which appear in the maximum entropy formulation of the single-centroid SDA.

Section 5.3.3 describes how the centroids are estimated and Section 5.3.4 discusses initialization strategies for the mixture SDA parameters.

5.3.1 Estimating the Component Weights

To compute the log-likelihood-maximizing weights w_{ghl}, one uses the standard technique of taking the derivative of the log-likelihood with respect to w_{ghl}, setting it to zero, and solving the resulting expression for w_{ghl}. The constraint $\sum_{l=1}^{c_h} w_{ghl} = 1$ is taken into account with the Lagrange multiplier η:

$$\frac{\partial}{\partial w_{ghl}} \left\{ L(\{\mathcal{T}_g(z_i)\}|\theta_{gh}) + \eta \left(\sum_{l=1}^{c_h} w_{ghl} - 1 \right) \right\} = \sum_{i=1}^{n_g} \frac{1}{w_{ghl}} P(C = l|s(z_i, \mu_{hl}), \theta_{gh}) + \eta = 0,$$

which gives the well-known expression for the component weights of a mixture model in terms of the responsibilities:

$$w_{ghl} = \frac{1}{n_g} \sum_{i=1}^{n_g} P(C = l|s(z_i, \mu_{hl}), \theta_{gh}). \tag{5.15}$$

5.3.2 Estimating γ_{ghl} and λ_{ghl}

The same approach used for estimating the component weights $\{w_{ghl}\}$ is adopted to estimate the SDA parameters $\{\gamma_{ghl}\}$ and $\{\lambda_{ghl}\}$: Find the likelihood-maximizing

values of the parameters by setting the corresponding partial derivatives to zero and solving the resulting equations. First, since each γ_{ghl} is simply a scaling factor that ensures that each mixture component is a probability mass function, one rewrites

$$\gamma_{ghl} = \frac{1}{\sum_{s(X,\mu_{hl})\in\Omega} e^{\lambda_{ghl}s(X,\mu_{hl})}}, \tag{5.16}$$

where $X \in \mathcal{X}_g$ is a random sample from class g, $s(X, \mu_{hl})$ is its corresponding random similarity to component centroid μ_{hl}, and Ω is the set of all possible similarity values. Substituting (5.16) into (5.14), setting the partial derivative of $L(\{\mathcal{T}_h(z_i)\}|\theta_{gh})$ with respect to λ_{ghl} to zero, and rearranging the terms gives

$$\frac{\sum_{s(X,\mu_{hl})\in\Omega} s(X,\mu_{hl})e^{\lambda_{ghl}s(X,\mu_{hl})}}{\sum_{s(X,\mu_{hl})\in\Omega} e^{\lambda_{ghl}s(X,\mu_{hl})}} \sum_{i=1}^{n_g} P(C = l|s(z_i,\mu_{hl}),\theta_{gh}) = \sum_{i=1}^{n_g} s(z_i,\mu_{hl})P(C = l|s(z_i,\mu_{hl}),\theta_{gh}). \tag{5.17}$$

The first term on the left side of (5.17) is simply the definition of the expected value of the similarity of samples in class g to the lth centroid of class h. Thus, one rewrites (5.17)

$$E_{P(\mathcal{T}(x)|Y=g)}[s(X,\mu_{hl})] = \frac{\sum_{i=1}^{n_g} s(z_i,\mu_{hl})P(C = l|s(z_i,\mu_{hl}),\theta_{gh})}{\sum_{i=1}^{n_g} P(C = l|s(z_i,\mu_{hl}),\theta_{gh})}. \tag{5.18}$$

Expression (5.18) is an equality constraint on the expected value of the similarity of samples $z_i \in \mathcal{X}_g$ to the component centroids μ_{hl} of class h. This is the same type of constraint that must be solved in the mean-constrained, maximum entropy formulation of single-centroid SDA (3.4). In (3.4), the mean similarity of samples from class g to the single centroid of class h is constrained to be equal to the observed average similarity. Analogously, in (5.18), the mean similarity of the samples from class g to the lth centroid of class h is constrained to be equal to the weighted sum of the observed similarities, where each similarity is weighted by its normalized responsibility. To solve for λ_{ghl}, one uses the same numerical procedure used to solve

(3.4) and described in Section 3.1. Thus, solving for all the $\{\lambda_{ghl}\}$ requires solving the $G \times \sum_{h=1}^{G} c_h$ expressions of (5.18).

It is not surprising that taking the EM approach to estimating λ_{ghl} has lead to the same expressions for the mean constraints in the maximum entropy approach to density estimation. It is known that maximum likelihood (ML) – the foundation for EM – and maximum entropy are dual approaches to estimating distribution parameters which lead to the same unique solution based on the observed data [36]. The ML approach assumes exponential distributions for the similarities, maximizes the likelihood, and arrives at constraint expressions whose solutions give the desired values for the parameters. The maximum entropy approach assumes the constraints, maximizes the entropy, and arrives at exponential distributions whose parameters satisfy the given constraints. This powerful dual relationship between ML and maximum entropy extends from metric problems to similarity-based problems; for this reason it leads to the the constraint expression (5.18), from which λ_{ghl} is numerically computed. The corresponding γ_{ghl} is found by applying (5.16).

5.3.3 Estimating the centroids

Estimating the centroids of a mixture model encompasses two problems: estimating the number of components (i.e. centroids) $\{c_h\}$, and estimating the centroids $\{\mu_{hl}\}$. This work adopts the common metric learning practice of cross-validating the number of mixture components $\{c_h\}$. The centroids $\{\mu_{hl}\}$ are estimated with the K-medoids algorithm [27], using the maximum-sum-similarity criterion (2.8). The initial centroids are selected randomly from the training set samples $z_i \in \mathcal{X}_h$. K-medoids was described in Section 2.2.2.

5.3.4 Initializing EM for SDA

In this work, the component weights $\{w_{ghl}\}$ are uniformly initialized to $w_{ghl} = 1/c_h$ and the components are assigned uniform initial probability $P(s(z_i, \mu_{hl})|C = l, \theta_{gh}) =$

$1/c_h$. This initialization reflects the assumption that initially the mixture components equally contribute to a sample's class-conditional probability: it is the least-assumptive initialization. Another strategy would be to initialize the weights by the fraction of training samples assigned to the clusters which result from estimating the centroids with K-medoids. The component probabilities may also be initialized by estimating the SDA parameters $\{\lambda_{ghl}\}$ and $\{\gamma_{ghl}\}$ from the K-medoids clusters. This is analogous to the GMM initialization strategy based on the results of the K-means algorithm. In practice, as shown in Section 5.3.5 and in Chapter 6, the simple uniform initialization works well.

5.3.5 Example of SDA Parameter Estimation with EM

As a concrete example of SDA parameter estimation with EM, consider two classes whose elements are random perturbations of prototypical binary vectors in 12 dimensions. Each class has two prototypical vectors, so that $c_h = 2$ for both $h = 1, 2$. The component prototypes are

$$\begin{aligned}
\mu_{11} &= [0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0] \\
\mu_{12} &= [0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1] \\
\mu_{21} &= [1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1] \\
\mu_{22} &= [1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0]
\end{aligned}$$

The elements of each class are generated by randomly drawing one of the prototypes with probability 1/4, and by independently perturbing (flipping) each dimension of the prototypes with probability 1/10. The mixture SDA model parameters are estimated from 1000 training samples, keeping the number of components per class fixed at 2. The Hamming similarity is the adopted similarity function. Figure 5.1 shows the log-likelihood of the class one samples for 30 iterations of the EM algorithm. As expected, the likelihood increases at each iteration and converges to a maximum value.

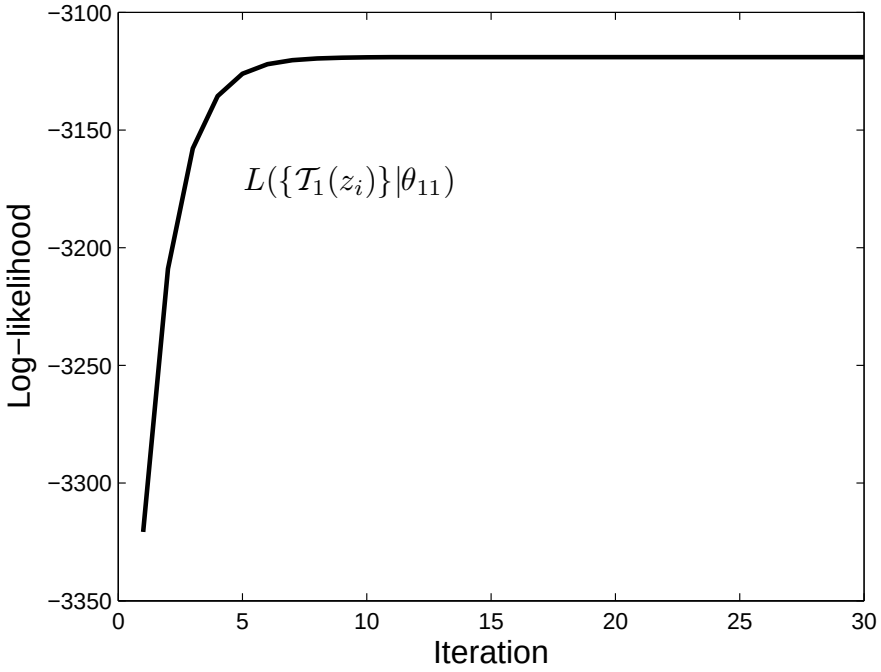

Figure 5.1: Log-likelihood of the training samples $z_i \in \mathcal{X}_1$ as a function of iteration number.

Figure 5.2 shows the component weights as they converge to their ML values. All the final values for the weights are close to the true prior weight of 0.5.

Figure 5.3 shows the λ_{1hl} parameters during the EM iterations. As for the likelihood and the component weights, they converge to their ML values. Note that for the similarities of the samples in class $g = 1$ to the centroids of class $h = 1$ the corresponding λ_{111} and λ_{112} converge to positive values, but for the similarities of class $g = 1$ samples to the centroids of class $h = 2$, the corresponding λ_{121} and λ_{122} converge to negative values. These final values are consistent with the distribution of the similarities: The within-class one similarities are on average higher than the inter-class similarities. The positive exponents capture the information about the higher average similarity and cause their corresponding exponential mixture components to assign higher probability to higher similarity. Conversely, negative exponents

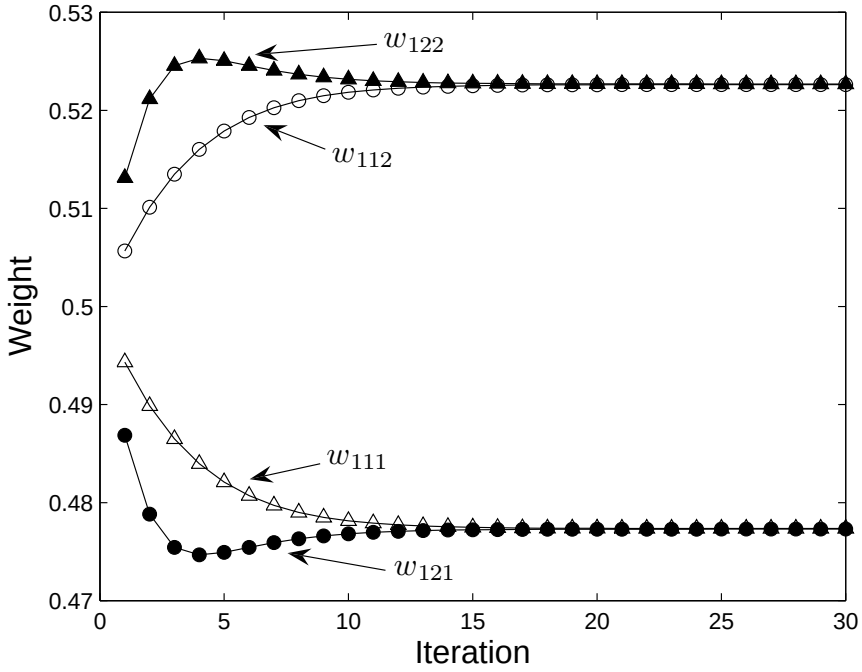

Figure 5.2: Mixture weights $\{w_{1hl}\}$ for $h = 1, 2$ and $l = 1, 2$ as a function of iteration number.

correctly favor assigning higher probabilities to lower similarities. The end result is consistent with the intuitive idea of similarity: the mixture SDA model for class $g = 1$ assigns a test sample $x \in \mathcal{X}_1$ high probability to be in class $g = 1$ if its similarities to μ_{11l} are high *and* its similarities to μ_{12l} are low. Finally, the K-medoids algorithm correctly identified the true component centroids. Analogous results, not shown here for conciseness, are obtained for class two.

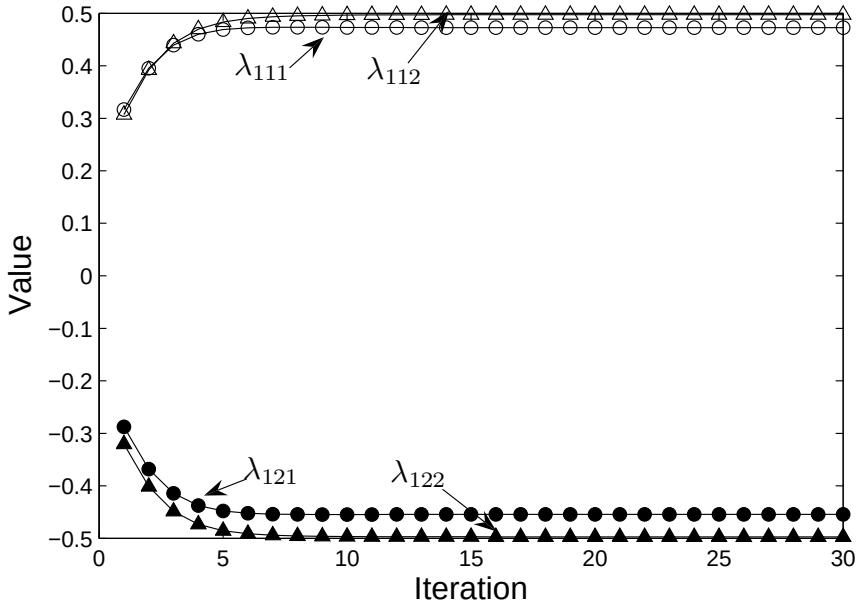

Figure 5.3: Mixture exponential parameters $\{\lambda_{1hl}\}$ for $h = 1, 2$ and $l = 1, 2$ as a function of iteration number.

Chapter 6

EXPERIMENTS

SDA (Section 3.1), local SDA (Section 4.1), mixture SDA (Section 5.2), and nnSDA (Section 3.2.1) are compared to other similarity-based classifiers in a series of experiments: the tested classifiers are the nearest centroid (NC), local nearest centroid (local NC), k-nearest neighbors (k-NN) in similarity space, condensed nearest neighbor (CNN) [27] in similarity space, and the potential support vector machine (PSVM) [30]. When the features underlying the similarity are available, the classifiers are also compared to the naive Bayes classifier [27]. The counting and the VDM similarities are used to compute the similarities on which the classifiers operate, except for cases in which similarity is provided as part of benchmark datasets.

The first set of comparisons involves simulated binary data, where each class is generated by random perturbations of one or two centroids. Illustrative, smaller-scale experiments based on randomly perturbed binary vectors were described in Chapters 3 and 5. This chapter presents results from more extensive experiments. The *perturbed centroids* simulation is a scenario where each class is characterized by one or two prototypical samples (centroids), but samples have random perturbations that make them different from their class centroid in some features. Thus, this simulation fits the centroid-based SDA models, in that each class is defined by perturbations around one or two prototypical centroids.

Then, three benchmark datasets are investigated: the protein dataset, the voting dataset, and the sonar dataset. The results on the simulated and benchmark datasets show that the proposed similarity-based classifiers are effective in classification problems spanning several application domains, including cases when the similarity mea-

sures do not possess the metric properties usually assumed for metric classifiers and when the underlying features are unavailable.

For local SDA and local NC, the class prior probabilities are estimated as the empirical frequency of each class in the neighborhood; for SDA, mixture SDA, nnSDA, NC, and CNN they are estimated as the empirical frequency of each class in the entire training data set. The k-NN classifier is implemented in the standard way, with the neighborhood defined by the test sample's k most similar training samples, irrespective of the training samples class. Ties are broken by assigning a test sample to class one.

6.1 Perturbed Centroids

In this two-class simulation, each sample is described by d binary features such that $\mathcal{B} = \{0, 1\}^d$. Each class is defined by one or two prototypical sets of features (one or two centroids). Every sample drawn from each class is a class centroid with some features possibly changed, according to a feature perturbation probability. Several variants of the simulation are presented, using different combinations of number of class centroids, feature perturbation probabilities, and similarity measures. Given samples $x, z \in \mathcal{B}$, $s(x, z)$ is either the counting or the VDM similarity. The simulations span several values for the feature dimensions d and are run several times to better estimate mean error rates. For each run of the simulation and for each number of features considered, the neighborhood size k for local SDA, local NC, and k-NN is determined independently for the three classifiers by leave-one-out cross-validation on the training set of 100 samples; the range of tested values for k is $\{1, 2, \ldots 20, 29, 39, \ldots, 99\}$. The optimum k is then used to classify 1000 test samples. Similarly, the candidate numbers of components for mixture SDA and for CNN are $\{2, 3, 4, 5, 7, 10\}$. To keep the experiment run time within a manageable practical limit, five-fold cross validation was used to determine the number of components for mixture SDA, and the mixture SDA EM algorithm was limited to 30 iterations for each cross-validated mixture

model. The parameters for the PSVM classifier are cross-validated over the range of possible values $\epsilon = \{0.1, 0.2, \ldots 1\}$ and $C = \{1, 51, 101, \ldots 951\}$.

The perturbed centroid simulation results are in Tables 6.1-6.8. For each value of d, the lowest mean cross-validation error rate is in bold. Also in bold for each d are the error rates which are not statistically significantly different from the lowest mean error rate, as determined by the Wilcoxon signed rank test for paired differences, with a significance level of 0.05. The naive Bayes classifier results are also included for reference.

6.1.1 Perturbed Centroids — One Centroid Per Class

Each class is generated by perturbing one centroidal sample. There are two, equally likely classes, and each class is defined by one prototypical set of d binary features, c_1 or c_2, where c_1 and c_2 are each drawn uniformly and independently from $\{0, 1\}^d$. A training or test sample z drawn from class g has the ith feature $z[i] = c_g[i]$ with probability $1 - p_g$, and $z[i] \neq c_g[i]$ with perturbation probability p_g. In one set of simulation results $p_1 = 1/3$ and $p_2 = 1/30$; thus, class two is well-clustered around its generating centroid and the two classes are well-separated. In another set of simulation results, $p_1 = 1/3$ and $p_2 = 1/4$ and the two classes are not as well separated. Classifiers are trained on 100 training samples and tested on 1000 test samples per run; twenty runs are executed for a total of $20,000$ test samples. The number of features d ranges from $d = 2$ to $d = 200$ in the simulation, but the number of training samples is kept constant at 100, so that $d = 200$ is a sparsely populated feature space. This procedure was repeated for the counting and for the VDM similarities, so there are four sets of results for the one centroid simulation, depending on the perturbation probabilities and the similarity measure used. The results are in Tables 6.1-6.4.

The performance of all classifiers increases as d increases. For large d, the feature space is sparsely populated by the training and test samples, which are segregated around their corresponding generating centroids. This leads to good classification

Table 6.1: Perturbed centroids experiment - One centroid per class. Misclasssification percentage for **counting** similarity, perturbation probabilities $\mathbf{p_1 = 1/3}$ and $\mathbf{p_2 = 1/30}$.

d	Local SDA	Local NC	SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**15.58**	**15.58**	35.13	23.47	49.80	**15.58**	19.22	**16.07**	15.58
4	**11.74**	**11.82**	23.97	22.54	39.08	12.05	13.85	13.01	11.98
8	**4.88**	6.10	**12.85**	14.07	**6.35**	6.19	7.86	6.21	4.63
12	**3.35**	3.97	10.16	11.50	5.00	4.26	6.01	3.74	2.46
25	**2.27**	3.50	**7.36**	11.49	**2.27**	3.49	5.30	**2.16**	1.77
40	1.86	2.79	**3.65**	8.79	**1.38**	2.79	4.38	**1.33**	1.24
50	2.02	2.37	**2.71**	7.94	**1.60**	2.31	3.24	**1.33**	1.29
75	1.90	2.58	2.56	7.83	**1.31**	2.27	3.82	**1.43**	1.43
100	2.11	2.32	2.05	5.92	**1.03**	2.16	3.69	1.65	1.65
125	**1.86**	**2.07**	**1.67**	6.21	**1.47**	**1.96**	3.56	**1.58**	1.58
150	**1.40**	**1.50**	**1.23**	4.86	**1.08**	**1.44**	2.55	**1.20**	1.20
175	**1.63**	**1.64**	**1.37**	4.28	**1.29**	**1.60**	2.59	**1.33**	1.33
200	**1.41**	**1.42**	**1.26**	4.20	**0.99**	**1.38**	2.67	**1.26**	1.26

performance for all classifiers. For small d, the feature space is densely populated by the samples, and the two classes considerably overlap, negatively affecting the classification performance.

Across all four sets of results, the naive Bayes classifier almost always gives the best performance. Its assumption that the features are independent captures the true underlying relationship of the sample features makes the naive Bayes classifier well suited for these particular data sets: indeed the samples are generated as random vectors of independent binary features. The consequent excellent performance of the naive Bayes classifier provides a reference point for the other classifiers. More

Table 6.2: Perturbed centroids experiment - One centroid per class. Misclasssification percentage for **counting** similarity, perturbation probabilities $\mathbf{p_1 = 1/3}$ and $\mathbf{p_2 = 1/4}$.

d	Local SDA	Local NC	SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**30.88**	**30.88**	48.48	**30.29**	49.70	**30.88**	31.07	**30.66**	30.62
4	31.19	**30.30**	35.56	**29.83**	44.41	30.92	32.63	**29.25**	29.18
8	22.63	22.56	**23.30**	**21.95**	33.13	23.12	24.13	**21.18**	21.02
12	18.11	18.58	**18.39**	**16.99**	29.52	18.42	20.04	**17.03**	16.56
25	12.16	13.90	13.40	13.17	26.21	10.40	14.82	**8.84**	7.96
40	7.87	11.42	10.33	12.45	17.59	7.26	11.58	**5.67**	4.91
50	6.59	8.98	9.47	11.32	19.36	6.42	10.37	**4.43**	3.69
75	5.32	6.96	6.06	8.42	12.29	4.17	7.89	**2.67**	2.19
100	4.84	6.56	5.66	6.96	9.09	3.93	5.61	**2.88**	2.69
125	3.23	5.10	3.98	6.25	11.82	2.65	4.81	**2.08**	2.01
150	3.03	3.84	3.07	5.13	**6.38**	2.61	4.50	**1.97**	1.94
175	3.56	3.86	3.86	6.30	**4.81**	2.83	4.33	**2.38**	2.38
200	2.61	2.66	2.78	3.66	**2.42**	2.08	3.15	**1.75**	1.75

generally, when a classification problem involves samples natively embedded in an Euclidean space, as in these perturbed centroids experiments, metric-space classifiers like naive Bayes can perform well. In these cases, the similarity-based classification framework provides no clear advantage.

On the other hand, naive Bayes cannot be used when the samples are not described by vectors of independent features, either because the features are not known, the independence assumption is too restrictive for effective performance, or because the Euclidean representation does not sufficiently capture the pairwise relationships of the samples. In these cases, the similarity-based techniques provide solutions to

Table 6.3: Perturbed centroids experiment - One centroid per class. Misclasssification percentage for **VDM** similarity, perturbation probabilities $\mathbf{p_1 = 1/3}$ and $\mathbf{p_2 = 1/30}$.

d	Local SDA	Local NC	SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**16.36**	**16.36**	34.13	26.41	48.90	**16.36**	22.17	**16.87**	16.36
4	**11.28**	**11.18**	15.22	19.10	38.16	**11.37**	12.23	12.20	11.23
8	**6.71**	7.51	**9.89**	14.52	**8.09**	**7.44**	**8.71**	**6.89**	5.42
12	**4.69**	**6.17**	**5.20**	12.99	**4.48**	5.85	7.36	**4.78**	3.33
25	2.96	3.46	2.56	9.87	4.08	3.35	4.90	**2.09**	1.59
40	2.36	2.60	2.62	7.28	4.65	2.49	4.37	**1.78**	1.67
50	2.60	2.86	2.61	7.02	4.45	2.80	4.70	**1.97**	1.94
75	2.42	2.59	2.11	6.09	2.96	2.47	4.03	**1.93**	1.93
100	1.88	1.90	1.74	3.97	2.38	1.88	2.46	**1.68**	1.68
125	1.67	1.68	**1.54**	3.25	**2.03**	1.67	2.39	**1.52**	1.52
150	1.68	1.68	**1.65**	2.92	**1.89**	1.68	2.17	**1.64**	1.64
175	1.60	1.61	**1.57**	2.59	**1.63**	1.61	2.08	**1.56**	1.56
200	**1.63**	**1.63**	**1.62**	2.25	**2.14**	**1.63**	1.84	**1.62**	1.62

classification problems. Thus, in these perturbed centroids experiments, the naive Bayes classifier is a good reference for assessing the effectiveness of the similarity-based classifiers, but it is not considered for the Wilcoxon significance tests because it is not generally applicable to similarity-based classification.

With few exceptions the PSVM performs best on the four sets of results on a wide range of d. This is likely because the PSVM classifies a test sample based on its similarities to the entire training set. Recall from (2.15) that the class of a test sample is determined from the weighted sum of the similarities $s(x, z_i)$ of the test sample to all the training samples, where the weights $\hat{\alpha}$ associated with each similarity are computed with numerical optimization on the training set. In contrast,

Table 6.4: Perturbed centroids experiment - One centroid per class. Misclasssification percentage for **VDM** similarity, perturbation probabilities $\mathbf{p_1 = 1/3}$ and $\mathbf{p_2 = 1/4}$.

d	Local SDA	Local NC	SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**34.38**	**34.38**	42.72	**34.17**	48.50	**34.38**	**33.76**	**34.56**	34.66
4	29.85	30.10	30.04	28.47	44.27	29.59	30.55	**27.85**	27.46
8	25.66	26.05	24.38	**24.16**	26.99	25.52	26.47	**24.24**	23.41
12	18.71	19.28	**17.99**	**17.98**	22.86	19.95	20.82	18.26	16.90
25	10.75	11.49	9.92	10.58	14.21	11.01	12.04	**9.10**	8.03
40	8.00	8.28	6.91	7.88	8.68	7.57	9.10	**6.02**	5.00
50	6.86	7.97	5.93	6.98	8.03	5.98	8.81	**4.81**	3.82
75	4.18	5.24	3.45	4.81	4.08	3.53	4.22	**2.63**	2.02
100	3.76	4.06	3.19	4.02	**2.71**	3.26	3.62	2.50	2.22
125	2.67	3.02	2.03	2.73	2.15	2.13	2.86	**1.65**	1.52
150	**2.90**	2.97	2.56	3.41	**1.73**	2.68	3.16	**2.25**	2.13
175	**2.56**	**2.75**	2.25	2.62	**2.07**	2.46	2.54	**2.12**	2.10
200	**2.31**	**2.39**	2.18	2.56	**1.77**	2.27	2.90	**2.02**	1.98

local methods such as local SDA, local NC, nnSDA, k-NN, and CNN make use of a subset of the training samples and thus have less information available to classify. Global methods based on the similarity-to-class-centroid summary statistic such as SDA, NC, and CNN also use less information. It is plausible that the ability to make use of all the similarity information in the training set and to optimally weight the similarities to the training samples gives the PSVM a performance advantage over the other techniques. However, in spite of this advantage, the results show that for low and high values of d the SDA-based techniques yield statistically equivalent performance to the PSVM, and in some cases match or exceed its results. When the PSVM statistically produces significantly different results from the other techniques,

its performance does not hugely surpass them. Thus the similarity-based techniques possess the ability to produce good classification results using less information. This quality can be immensely useful when few training samples are available.

In all four sets of results, the SDA-based algorithms generally perform better than their non-generative counterparts: local SDA performs better than local NC and SDA performs better than NC. This shows that generative models based on the similarity of samples to local or global class centroids provide increased discriminative power over the non-generative centroid-based similarity models. Furthermore, in almost all cases across the four sets of results, local SDA performs better than SDA. While the classification performance of SDA is good, its inherent model bias prevents it from achieving even better performance; local SDA is not as susceptible to model bias, and is able to perform very well. Still, the SDA performance is close to that of the local SDA in all cases and sometimes it surpasses it (VDM similarity with $p_2 = 1/4$), a confirmation that the single-centroid generative model at the heart of SDA matches well the perturbed single-centroid experimental setup for these sets of results.

The similarity-space k-NN performs well, albeit not as well as the PSVM. Compared to SDA, k-NN performs better only for the counting similarity and $p_2 = 1/4$. Since SDA matches well the class models for the generated samples, it is not surprising that it performs better than k-NN, which does not rely on class models. However, k-NN does better when the class two perturbed samples are more likely to differ from their generating class two centroid ($p_2 = 1/4$), that is when the classes overlap more. In this case, it is more difficult to estimate the class centroids, and the SDA performance is affected. On the other hand, SDA is better than k-NN for the VDM similarity, for both $p_2 = 1/30$ and $p_2 = 1/4$. The VDM similarity is calculated from class-dependent lookup tables pre-computed from the training set, and this additional information seems to favor the SDA classifier more than the k-NN. Local SDA, performs slightly better than k-NN when $p_2 = 1/30$ for both counting and VDM similarities.

The CNN classifier generally does not perform as well as k-NN. This is expected, because, as for its metric learning analog, the condensing process primarily aims to reduce the size of large training sets and possibly eliminate outliers rather than to improve classification performance. The observed lower performance of CNN compared to k-NN reflects the expectation that classification performance will degrade when using the condensed training set instead of the full set of available training samples.

The nnSDA classifier performs well for the counting similarity when $p_2 = 1/30$, and in general for higher values of d. For low values of d the performance is particularly poor: for $d = 2$ the error rate is essentially equal to that of a random classifier (50%) and for $d = 4$ it is only slightly better. In fact, the nnSDA performance is limited by the interplay of its asymptotic behavior and the value of d. Recall that by Lemma (1) from Section 4.2, $P(s(x, Z_k) = s_{max}) \to 1$ as $k, N \to \infty$ and $k/N \to 0$, where k is the neighborhood size, N is the number of available training samples, and Z_k is the k-th nearest neighbor of test sample x. Then, it follows that $P(s_{nn,h}(x) = s_{max}) \to 1$ for all h as $k, n \to \infty$, because $s_{nn,h}(x) = s(x, Z_1)$ for $Z_1 \in \mathcal{X}_h$ as $k \to \infty$. Thus, for nnSDA, the similarities of a test sample to its nearest neighbors in each class are all identical in the limit of infinite number of training samples. Consequently, for a large training set, all class discriminants in the nnSDA classification rule (3.14) are identical and therefore uninformative. The classification rule (3.14) reduces to the trivial rule that classifies according to the cost-adjusted class priors,

$$\hat{y} = \arg \min_{f=1,\ldots,G} \sum_{g=1}^{G} C(f, g) P(Y = g). \tag{6.1}$$

When 0-1 costs are used, as in this simulation, the rule (6.1) always classifies as the class g with the highest prior probability $\hat{P}(Y = g)$, estimated as the empirical frequency from the training data:

$$\hat{y} = \arg \max_{g=1,\ldots,G} \hat{P}(y = g). \tag{6.2}$$

In this experiment, the samples are generated from two, a priori equally likely classes,

so the limit misclassification rate is $1 - \max_g \hat{P}(Y = g) \approx 0.5$.

The limit error rate is noticeable when d is small. In this case the similarity can take on values in a limited range bounded by d ($s(x, z) \in [0, 1 \ldots d]$ for the counting similarity) and the training set is highly redundant. Thus, a test sample x is very likely to be maximally similar to its nearest neighbor from each class, and $s_{nn,h}(x)$ is uninformative. In higher dimensions, the experimental results show that the training set is sufficiently sparse for effective classification. Thus nnSDA is a viable classifier for sparse training sets which do not cover the entire range of possible values for the chosen similarity. In applications when few training samples are available, nnSDA can be a valuable tool for achieving actionable classification results.

6.1.2 Perturbed Centroids — Two Centroids Per Class

In this variation of the perturbed centroids simulation, each class is characterized by two prototypical samples, c_{11}, c_{12} for class one, and c_{21}, c_{22} for class two. Each time the simulation is run, the centroids $c_{11}, c_{12}, c_{21}, c_{22}$ are drawn independently and identically using a uniform distribution over $\mathcal{B}$.

Every sample drawn from each class is a perturbed version of one of the two class prototypes, where the class labels are drawn independently and identically with probability 1/2. A training or test sample z drawn from class one is randomly selected to be $z = c_{11}$ or $z = c_{12}$ with probability 1/2, and then for each $i = 1, \ldots, d$, z's ith feature is probabilistically perturbed so that $z[i] \neq c_{11}[i]$ with probability p_{11} (or $z[i] \neq c_{12}[i]$ with probability p_{12}). Thus on average, a randomly drawn sample based on c_{11} will have dp_{11} features that are different from the class prototype c_{11}'s features. Likewise, a training or test sample v drawn from class two starts out as $v = c_{21}$ or $v = c_{22}$ with probability 1/2, but then for each $i = 1, \ldots, d$, v's ith feature is changed so that $v[i] \neq c_{21}[i]$ with probability p_{21} (or $v[i] \neq c_{22}[i]$ with probability p_{22}).

The number of features d ranges from $d = 2$ to $d = 200$ in the simulation, but the number of training samples is kept constant at 100, so that $d = 200$ is a sparsely

populated feature space. Two different sets of values of the perturbation probabilities $p_{11}, p_{12}, p_{21}, p_{22}$ were used: in the first case $p_{11} = p_{12} = 1/3$ and $p_{21} = p_{22} = 1/30$, so that the class two samples are much more tightly clustered around c_{21} and c_{22} than the class one samples are with respect to c_{11} and c_{12}. In the second case, $p_{11} = p_{12} = 1/3$ and $p_{21} = p_{22} = 1/4$, resulting in a higher Bayes error. Each simulation was run twenty times, for a total of 20,000 test samples. The resulting mean error rates are given in Tables 6.5-6.8.

Table 6.5: Perturbed centroids experiment - Two centroids per class. Misclasssification percentage for **counting** similarity, perturbation probabilities $\mathbf{p_{11} = p_{12} = 1/3}$ and $\mathbf{p_{21} = p_{22} = 1/30}$.

d	Local SDA	Local NC	SDA	Mixture SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**26.41**	**26.41**	47.52	**28.93**	38.98	49.30	**26.41**	**27.51**	**27.16**	29.09
4	**13.80**	**13.68**	34.77	18.32	34.84	38.04	**13.26**	16.84	17.18	17.95
8	**9.23**	**9.25**	29.32	**13.96**	26.77	**9.54**	**9.29**	12.82	12.62	9.96
12	**5.61**	6.47	31.20	**10.56**	27.05	**7.35**	**6.25**	10.87	8.72	7.96
25	3.11	4.37	28.75	**2.39**	25.90	3.21	4.03	9.45	4.08	2.67
40	2.88	4.25	30.84	**6.54**	28.23	**1.91**	3.94	8.69	**2.21**	1.39
50	2.94	4.89	27.77	**1.73**	30.12	**1.32**	4.35	9.10	**1.77**	1.16
75	2.04	3.21	26.38	3.69	27.74	**1.61**	2.75	7.61	**0.95**	1.12
100	2.21	3.03	25.39	**2.30**	24.58	**1.37**	2.60	5.25	**1.52**	1.08
125	2.46	2.96	25.51	**4.74**	24.83	**1.42**	2.68	5.33	**1.59**	1.47
150	1.55	1.80	25.00	4.54	26.55	**1.54**	1.76	5.34	**1.00**	0.78
175	1.93	2.38	25.32	**2.72**	21.40	**1.16**	2.02	4.17	**1.29**	1.21
200	1.44	1.61	23.87	**1.63**	19.28	**1.38**	1.49	4.45	**1.10**	0.95

For all four sets of results, the local SDA classifier performs better than the local NC classifier. This result agrees with the analogous case for the single centroid

Table 6.6: Perturbed centroids experiment - Two centroids per class. Misclasssification percentage for **counting** similarity, perturbation probabilities $\mathbf{p_{11} = p_{12} = 1/3}$ and $\mathbf{p_{21} = p_{22} = 1/4}$.

d	Local SDA	Local NC	SDA	Mixture SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**40.02**	**40.19**	49.36	40.21	42.87	49.90	**39.88**	**37.64**	**39.85**	39.91
4	**33.43**	33.77	39.96	39.13	37.95	46.61	**33.17**	**34.91**	**34.67**	32.62
8	**29.81**	31.84	36.82	33.52	34.94	40.20	**29.10**	35.09	**30.12**	28.43
12	**27.27**	29.42	35.24	39.11	33.19	38.37	**27.19**	30.30	**27.51**	25.90
25	19.89	22.91	29.83	39.86	28.81	33.77	**17.09**	22.75	**16.79**	17.51
40	14.05	16.91	28.60	34.62	26.70	24.45	**11.49**	18.14	13.10	12.72
50	11.65	14.64	26.82	34.22	25.61	31.04	**9.04**	15.90	10.19	9.68
75	8.16	9.01	24.61	30.99	24.48	20.37	**5.84**	12.71	**7.51**	6.00
100	7.67	8.00	23.59	30.20	21.68	17.09	**4.83**	9.68	**4.37**	3.96
125	6.05	6.79	23.70	26.82	22.50	15.18	**3.52**	7.87	**3.94**	2.99
150	5.05	6.31	22.36	26.24	21.62	11.50	**3.13**	6.13	**2.90**	2.79
175	3.72	4.15	25.02	23.29	23.79	10.43	**2.14**	6.29	2.39	1.81
200	3.45	3.85	21.86	21.74	21.83	9.41	**2.19**	5.28	**2.36**	2.24

experiments and attests to the advantage that similarity-based generative models provide over simpler nearest-centroid classifiers. However, the SDA classifier yields better classification than its counterpart NC classifier only for the VDM similarity. For the counting similarity, SDA does not provide an advantage over NC. There are two causes that contribute to this outcome. First, the single-centroid SDA is a biased model that does not match the true two-centroids-per-class experimental setup. Consider class one and its centroids, c_{11} and c_{12}. SDA at best correctly estimates one of the two centroids per class, let's say $\hat{c}_{11}$. Thus, the estimated centroid-based generative model for class one is a good match for the samples which are generated as

Table 6.7: Perturbed centroids experiment - Two centroids per class. Misclasssification percentage for **VDM** similarity, perturbation probabilities $\mathbf{p_{11} = p_{12} = 1/3}$ and $\mathbf{p_{21} = p_{22} = 1/30}$.

d	Local SDA	Local NC	SDA	Mixture SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**25.83**	**25.61**	30.60	34.05	39.07	49.00	**25.61**	**30.83**	**28.64**	27.95
4	15.22	**13.46**	22.30	17.71	26.00	40.72	**13.59**	20.42	**16.87**	18.01
8	**10.96**	**11.81**	**11.77**	**13.66**	22.53	16.63	**11.15**	**12.14**	14.16	11.76
12	**8.17**	9.46	**7.92**	**9.41**	19.07	**9.41**	**8.52**	12.60	10.11	7.58
25	4.52	6.19	3.77	**4.39**	16.32	**5.95**	5.92	8.30	**4.23**	3.63
40	2.96	3.73	**2.30**	**2.77**	16.41	4.59	3.79	6.66	**2.79**	2.25
50	2.59	3.86	**1.74**	**2.58**	15.96	3.50	3.56	6.61	1.79	1.80
75	2.33	3.40	**1.57**	**1.57**	13.69	**3.20**	2.90	5.59	**1.15**	1.45
100	2.17	3.06	**1.52**	**1.03**	11.35	2.81	2.79	5.62	1.24	1.52
125	2.51	2.90	**1.63**	**1.36**	11.36	2.25	2.74	5.05	**1.39**	1.62
150	2.10	2.50	**1.39**	**1.44**	11.45	2.32	2.30	4.83	**1.03**	1.38
175	2.12	2.33	**1.47**	**1.44**	10.82	**1.62**	2.20	4.21	**1.31**	1.46
200	1.80	1.99	**1.19**	**1.88**	10.46	2.04	1.93	3.28	**1.32**	1.18

random perturbations of c_{11}. The model, however, is not a good match for samples generated as random perturbations of c_{12}. The model cannot distinguish the similarities of these class one samples to $\hat{c}_{11}$ from their similarities to the centroids of class two. The result is that the c_{12}-generated samples are classified according to the class priors, that is half as class one and half as class two. The same argument applies to class two, so that overall about 25% of the samples are misclassified. Indeed, the SDA error rates quickly settle to $\approx 25\%$ for the counting similarity for medium to large values of d. For lower d, the class overlap due to the density of the feature space dominates the misclassification rate.

Table 6.8: Perturbed centroids experiment - Two centroids per class. Misclasssification percentage for **VDM** similarity, perturbation probabilities $\mathbf{p_{11} = p_{12} = 1/3}$ and $\mathbf{p_{21} = p_{22} = 1/4}$.

d	Local SDA	Local NC	SDA	Mixture SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**39.98**	**40.01**	**42.57**	**40.26**	**40.77**	48.00	**40.01**	**39.66**	**41.08**	38.89
4	**37.28**	**37.45**	**37.98**	**34.99**	**38.53**	48.95	**37.21**	**37.09**	**36.19**	37.34
8	**30.80**	**32.80**	**30.62**	**31.26**	**30.99**	36.88	31.84	33.43	**30.23**	29.37
12	**27.26**	**28.85**	**27.87**	**26.97**	**28.59**	31.06	**27.65**	29.82	**29.68**	25.15
25	**21.87**	**21.86**	**20.88**	**22.03**	21.77	23.96	**20.82**	23.27	21.55	17.63
40	16.56	18.50	16.41	18.01	17.96	19.08	16.91	18.98	**15.20**	12.44
50	14.92	17.22	16.04	16.11	17.65	16.89	14.96	16.07	**13.92**	11.21
75	11.98	13.40	12.41	11.68	13.91	12.16	10.57	11.65	**8.99**	7.53
100	8.54	9.94	9.04	9.01	11.09	8.83	7.55	9.66	**6.87**	4.66
125	7.24	8.31	7.61	8.04	9.68	8.45	6.09	8.24	**6.07**	3.64
150	6.64	8.04	7.03	6.17	9.68	6.41	**5.03**	6.36	**5.15**	3.02
175	5.00	5.57	5.78	5.32	8.38	5.51	**4.03**	7.18	**4.15**	2.04
200	4.46	5.08	5.00	4.31	6.77	4.86	**3.39**	4.81	**3.91**	2.31

The second cause contributing to the observed SDA results stems from the way the class centroids are generated. Each class centroid is generated randomly from a multivariate uniform distribution over the feature space. Thus, there is no guarantee that two centroids from the same class be more similar to each other than two centroids from different classes, that is there is no guarantee that $s(c_{1i}, c_{1j}) < s(c_{1i}, c_{2j})$ for $i, j = 1, 2$. On the contrary, on average over many draws from the sample space, the centroids are equally similar, and consequently the samples generated as perturbations of c_{12}, c_{21}, and c_{22} are approximately equally similar to c_{11}. This amplifies the detrimental effect of the bias in the SDA model. If the condition on the similarities

between centroids $s(c_{1i}, c_{1j}) < s(c_{1i}, c_{2j})$ were enforced, then even the biased SDA model would produce better classification results.

The performance of mixture SDA is comparable to that of SDA if not slightly better. For the particularly simple case of the counting similarity with $p_{21} = p_{22} = 1/30$, the mixture SDA provides an order of magnitude improvement over SDA, showing that it is able to alleviate the bias problem inherent to the single-centroid SDA. However, in all other perturbed centroids results the comparison between the performance of mixture SDA and SDA is inconclusive. For $p_{21} = p_{22} = 1/4$, the overlap between the classes overshadows any performance gains mixture SDA might obtain; for the VDM results, the advantage provided by the optimized similarity measure brings the performance of SDA and mixture SDA closer together, and thus limits the gains of mixture SDA. Given the increase in complexity of the mixture SDA classifier and its inconclusive performance advantages, for these experiments it might be more advantageous to use local classifiers such as local SDA to obtain improved performance. The results show that local SDA consistently performs very well, and with only a few exceptions outperforms SDA and mixture SDA.

Note that for the VDM similarity, SDA produces excellent classification results which are very competitive with local SDA and local NC, and consistently outperform NC. The large improvement is attributable to the fact that the VDM undergoes a training phase, performed on the training set, in which the class information is used to optimize the similarity measure for class discrimination. This training step greatly benefits the SDA classifier and yields improved classification results for all classifiers when compared to the counting similarity, which does not rely on such pre-computations.

As for the single-centroid results, nnSDA is most effective at higher values of d, when the feature space is sparsely populated by the samples. A consistently good performer is the k-NN classifier, which is very competitive with local SDA, local NC, and the PSVM when $p_{21} = p_{22} = 1/30$, and often outperforms them when

$p_{21} = p_{22} = 1/4$. Using a subset of the training samples, as with CNN, negatively impacts the classification performance for all sets of simulations, consistently with the single-centroids results discussed in the previous section.

6.2 Benchmark Data Sets

Three benchmark data sets are used to analyze further the performance of various similarity-based classifiers: a data set of protein similarities, a data set of congressional voting records, and a data set of aural sonar similarities. The tested classifiers are the local SDA, local NC, SDA, NC, nnSDA, k-NN, and PSVM classifiers. Based on their comparatively lackluster performance on the perturbed centroids, the mixture SDA and CNN classifiers are not tested on these data sets, as the long time required to cross-validate their parameters does not justify their attainable performance.

The performance of the classifiers on all three benchmark data sets is evaluated as the *leave-one-out error*, as follows. One sample is set aside as the test sample, and all other $N-1$ samples are used for training. The parameters for each classifier are cross-validated on the $N-1$ training samples using leave-one-out cross validation. The resulting best parameters are used to train each classifier on the entire $N-1$ training samples, and the trained classifier finally classifies the test sample. The process is repeated until all available samples are tested by the trained classifiers. For local SDA, local NC and k-NN, the neighborhood size is cross-validated on the set of possible sizes $\{1, 2, \ldots 20, 30 \ldots 100, 150, 200\}$. The PSVM parameters are cross-validated over the sets of possible values $C = \{1, 51, \ldots 951\}$, and $\epsilon = \{0.1, 0.2, \ldots 1\}$. The class priors are estimated to be the empirical probability of seeing a sample from each class, with Laplace correction [35]. Table 6.9 shows the percent leave-one-out error for each classifier evaluated on the protein data set; Table 6.10 shows the percent leave-one-out error for each classifier evaluated on the voting and sonar echoes data sets. The benchmark data sets experiments are discussed in more detail in the following sections.

Table 6.9: Percentage of leave-one-out misclassifications on the protein data set.

Local SDA	Local NC	SDA	NC	k-NN	PSVM
8.92	37.09	29.58	41.78	20.66	**1.41**

Table 6.10: Percentage of leave-one-out misclassifications on the voting and sonar echoes benchmark data sets.

	Local SDA	Local NC	SDA	NC	nnSDA	k-NN	PSVM
Voting	9.66	8.05	11.72	12.87	12.18	9.20	**6.44**
Sonar	22	14	16	26	24	18	**10**

6.2.1 Protein Data

Many bioinformatics prediction problems are formulated in terms of pairwise similarities or dissimilarities. An example is the protein data set used by [30]. For this data set, pairwise dissimilarity values are calculated using the evolutionary distance, which is the probability that an amino acid sequence transforms into another one [31]. The sample space $\mathcal{B}$ is not enumerated, so classification must be done based only on the pairwise dissimilarity values. The dataset contains 213 proteins with class labels "HA" (72 samples), "HB" (72 samples), "M" (39 samples) , and "G" (30 samples). The SDA, local SDA, nearest centroid, local nearest centroid, and k-NN classifiers natively support multiclass classification problems, so they can be applied directly to this four-class experiment. The PSVM, however, is a binary classifier and cannot be applied directly to this multiclass data set. To compare to the PSVM, this experiment adopts the same standard practice commonly used for binary support vector classifiers: the PSVM classification problem is cast into four independent one-class-vs.-the-rest binary classification tasks and the classification rule becomes to classify as the class which yields the highest of the four resulting support vector margins [50].

The set of possible similarities Ω is needed to solve for the SDA parameters λ and γ, but is not directly available, so Ω was approximated as the set of empirical similarities that occur in the training samples' similarity matrix.

In this experiment, the PSVM performs best. Guessing that all samples were from the most prevalent class would yield a 66.2% error rate. The simple one-centroid per class model of SDA achieves half that error, and works better than the more flexible local nearest centroid classifier. Local SDA, local nearest centroid and k-NN all have the same free parameter, the neighborhood size k. Of these, local SDA is seen to be best suited to this problem.

6.2.2 Voting Data Set

The UCI voting data set [48] records the voting record of 435 members of the US House of Representatives on 16 bills. The binary classification problem is to predict each member's political party affiliation given the voting records. Each of the 16 votes is either a yes, a no, or "neither", so there are 16 features which can each take on 3 possible values. This classification problem can be treated as a similarity-based classification problem by applying a similarity function to the trinary feature space. The adopted similarity in this experiment is the counting similarity.

6.2.3 Aural Sonar Echoes Classification

In the sonar echoes classification experiment, the data consist of 100 pairwise similarities assessed by human listeners. The listeners rated the pairwise similarities of digitized active sonar echoes from two classes – clutter or target – without knowledge of the class labels, and based their evaluation of similarity only on their perceptual judgement of how the echoes sounded similar; thus, the underlying features of similarity are inaccessible. Each listener assigned a discrete similarity value between 1 and 5 to each pair of echoes; each pair was rated by two different listeners, and the two assigned similarity scores were added, so that the range of possible values for

the similarity is $[2, 10]$. The target and clutter classes are equally likely, each one containing 50 echoes. This set of echoes is particularly difficult to classify in that metric-space classifiers produced incorrect results. Further details on this data set are in [55].

For this data set, the PSVM performs best. The local NC and SDA are the best similarity-based classifiers. It is interesting that the local SDA provides decreased performance compared to SDA. In this data set the clutter class is very diverse making the local SDA classifier more prone to high variance estimates than SDA.

Chapter 7
CONCLUSIONS

This chapter summarizes the contributions of this dissertation in Section 7.1. Some ideas for further developing similarity-based classification techniques are outlined in Section 7.2.

7.1 Discussion

The contribution of this dissertation is a new framework for classification that is both *similarity-based* and *generative*: *similarity discriminant analysis*, or *SDA*. The experimental results in this dissertation show that the classifiers resulting from the proposed SDA framework have practical advantages in terms of performance, interpretability, and ease of use. SDA is *similarity-based* in that it classifies samples based on their pairwise similarities and does not require that the samples be described by numerical feature vectors, the standard sample description method in metric learning. SDA is *generative*, in that it estimates probabilistic models based on descriptive statistics of the classes. Having access to probability estimates is important. A probabilistic framework seamlessly accommodates multi-class classifiers, asymmetric misclassification costs, and class priors. Furthermore, probability estimates are easily fused into into larger systems, and can be used to identify abnormal samples that have low probability of any class. The generative models in the SDA family are solutions to constrained maximum entropy problems where the constraints are placed on the mean values of the similarity-based descriptive statistics. As dictated by the principle of maximum entropy, the resulting generative class models are exponential functions of the similarity statistics.

Different choices for the descriptive statistics lead to different SDA classifiers. This dissertation focused on the centroid-based SDA classifiers: each class is described by a prototypical sample, a *centroid*, and the generative models are based on the similarities of the samples to each class centroid. SDA accommodates various definitions of centroid; this dissertation focused on the maximum-sum-similarity centroid. The nearest neighbor similarity is also explored as a descriptive statistic, yielding the nnSDA classifier.

The SDA classifier may be thought of as creating log-linear boundaries between the classes in a similarity space where each dimension maps a function of the similarity of a sample to a class centroid. This interpretation is analogous to the standard view of linear metric-space classifiers, which create linear boundaries in a metric feature space. Metric-space classifiers such as LDA, QDA, and SVM-type methods, can be applied to similarity-based classification by using the similarities to the class centroids as the metric feature vectors. The resulting mathematical expressions for the linear class boundaries are formally the same as the log-linear boundaries produced by SDA: they are linear combinations of similarity statistics. However, SDA is more general than the metric generative classifiers LDA and QDA, as it does not rely on the descriptive similarity statistics being Gaussian. This assumption limits the applicability of LDA and QDA to similarity-based classification. Thus the SDA framework gives rise to a new family of classifiers which can more generally be applied to similarity-based classification problems.

As with LDA and QDA, the power of the SDA generative classifier depends on how well its model matches the true class-conditional distributions. A mismatched model will be biased and produce erroneous classifications. The centroid-based SDA classifier is a good match for single-centroid distributions of objects, but is a biased model for multi-centroidal distributions. This dissertation proposes *local SDA* and *mixture SDA* as similarity-based generative classifiers with reduced bias that can be used for multimodal distributions. Local SDA is the SDA classifier applied to a local

neighborhood of a test sample. A local class centroid can be viewed as a representative prototype for the class in the neighborhood of a test sample and the class-conditional models provide an estimate of the local distribution of the similarities to the local centroid. Local SDA shown to be a Bayes error-consistent classifier and is the first classifier to be similarity-based, generative, *and* local. Mixture SDA builds on the metric-learning mixture models by modeling each class as a linear combination of several single-centroid SDA models. The parameters for the mixture SDA classifier can be estimated with the EM algorithm; the resulting mixture SDA EM algorithm is shown to maximize the likelihood of the similarities of the samples to the class centroids.

The family of SDA classifiers is very competitive with, and often outperforms, their corresponding non-generative similarity-based classifier. SDA competes with nearest centroid; local SDA competes with local NC. The SDA classifiers are also competitive with the PSVM, the state-of-the-art support vector machine for similarity-based classification. The PSVM bases its classification on the entire training set of pairwise similarities. This requires enumeration of size $N \times N$ similarity matrices, thus posing computational challenges for large data sets. Furthermore, PSVM is a non-generative, intrinsically binary classifier: it is difficult to view it in probabilistic framework where there are more than two possible classes for the data samples. The SDA classifiers remain competitive while relying on more parsimonious representations of the underlying similarity relationships between the samples. Furthermore, the generative quality of the SDA family of classifiers provides intuitive information about the similarity characteristics of the data. The SDA-generated probability estimates are useful for interpreting the results in a probabilistic framework, and allow for class priors and costs to be seamlessly integrated into the classification rules.

7.2 The Road Ahead

The ideas put forth in this dissertation – generative models for similarity-based classification – provide the starting point for further investigation of several challenges in the rich area of similarity-based classification. In this section, ideas for future developments in mixture SDA and local SDA are presented first. Then, some ideas are proposed for solving the challenging problem of similarity-based classification when samples are described by mixed features.

7.2.1 Jointly Estimating the Centroids and the Component Parameters for Mixture SDA

This dissertation introduced the mixture SDA classifier as a way to reduce the SDA model bias, in analogy to metric learning mixture models such as GMMs. In the proposed EM algorithm for mixture SDA, the class centroids are first estimated in a single step. Then, the other parameters are iteratively estimated while the centroids are kept constant. A modified mixture SDA EM algorithm that jointly estimates the class centroids and the component parameters during each iteration can lead to improved performance. However, the improved class models would only partially address the ongoing challenge of decreasing estimation variance in mixture models, which results from the large number of parameters that must be estimated (the number of components, the relative component weights $\{w\}$, the component exponents $\{\lambda\}$ and scaling factors $\{\gamma\}$).

7.2.2 Adaptive Neighborhood Selection for Local SDA

This dissertation proposed the *local SDA* classifier as a way to make SDA more flexible and less prone to estimation bias. The innovation in local SDA is that it is the first classifier that is both local and generative. Local SDA applies SDA to a local neighborhood of the test sample. In this work, the neighborhood was chosen according

to the common definition adopted for metric local learning: select the k most similar samples to the test sample from the training set; the fixed neighborhood size k is cross-validated. A key question that has hardly been researched is "which similar examples?" That is, when applying a local learning algorithm (be it nearest neighbors, local SDA, etc.), how should one choose the neighborhood? The standard answer is to choose some fixed number of neighbors k, where k is chosen by cross-validation. This is not a satisfying approach because cross-validation can lead to high estimation variance, and is theoretically based on the often untrue assumption that training and test samples are independent and identically distributed. Also, the assumption that every test sample should be learned from the same number k of neighbors may not be generally true. Research into methods to choose optimal neighborhoods for local similarity-based learning can produce improved versions of the local SDA classifier.

Recent work by Gupta et al. has advanced the field of defining adaptive neighborhoods given Euclidean features [12, 24]. An ideal neighborhood method would automatically adaptively determine the number of samples and which samples to include in the neighborhood. For example, Nock et al. [51] defined the neighborhood for a test sample to be its nearest neighbor, and all training samples for which the test sample is the nearest neighbor. Another automatically spatially-adaptive neighborhood definition is Sibson's natural neighbors [52, 65]. The natural neighbors are defined by the Voronoi tessellation $\mathcal{V}$ of the training set and test point x. Given $\mathcal{V}$, the natural neighbors of x are defined to be those training points z_i whose Voronoi cells are adjacent to the cell containing x. Natural neighbors is an effective neighborhood for linear interpolation in two and three dimensions [52, 65]. However, to find the natural neighbors the entire Voronoi tessellation must be computed, which is computationally problematic in higher dimensions [3].

To guide the design and analysis of neighborhood algorithms, one could investigate design goals that relate to minimizing estimation error. One design goal is that asymptotically as $n \to \infty$, the neighborhood size $k \to \infty$ and $k/n \to 0$. This design

goal is important for the algorithm to asymptotically achieve the Bayes' error [15, 70]. The local SDA classifier was shown to satisfy this design goal in Chapter 4 for the neighborhood defined by the k most similar samples to the test sample. Researchers in ranking have proposed that the top ranked neighbors should be both similar to the test sample (*affinity*) but also different from each other (*diversity*) [81]. These design goals can be useful in guiding future research into defining neighborhoods for similarity-based learning, and that high affinity can keep estimation bias low, and high diversity can reduce estimation variance.

Related to the goal of diverse neighbors, Gupta defined the term *enclosing neighborhood* for neighborhood definitions for Euclidean features where the test sample is contained in the convex hull of its neighborhood if possible [22]. For example, one can show that the natural neighbors described above are an enclosing neighborhood. Gupta et al. proved that using an enclosing neighborhood results in a bounded estimation variance for local linear regression of a noisy linear function with additive Gaussian noise [12, 24]. To satisfy the design goals of affinity and diversity, they proposed the *enclosing k-NN neighborhood* to be the smallest k such that the k neighbors enclosed a test sample (or adding an additional neighbor does not decrease the distance from the test sample to the convex hull of its neighborhood). They also investigated an inclusive version of the natural neighbors such that the neighborhood includes all training samples closer than the furthest natural neighbor [12, 24].

7.2.3 Weighting Nearest Neighbors for Local SDA

The local SDA classifier proposed in this dissertation bases its local generative models on a local neighborhood of a test sample x. The training samples within the neighborhood equally contribute to the constraints on the mean local similarities (3.4) and equally affect the choice of the local centroid (2.10). The equal importance of the neighbors implicitly assigns uniform weights to their similarities to the test sample x. Adaptively weighting the similarity of a test sample to its neighbors can provide

improved classification power and deeper insight into the similarity relationships of the data. Recent progress in weighting strategies for metric nearest neighbor learning can guide future research into adaptive weights for similarity-based local learning.

Weighted k-NN assigns weight w_i to the training sample pair z_i and its corresponding class label y_i, where the weights are normalized such that $\sum_i w_i = 1$ and the test sample x is classified as the class assigned the most weight. The design goals of *affinity* and *diversity* that can guide neighborhood selection can also guide the definition of novel weighting schemes for the neighbors. An intuitive approach to weighting nearest neighbors is to give larger weight to neighbors that are more similar to the test sample. This is the design goal of *affinity*, which states that w_i should be an increasing function of $s(x, z_i)$. For example, affinity can be satisfied by $w_i = \gamma s(z, x_i)$, where γ is a normalization term. However, it was recently shown that such weighted nearest-neighbor classifiers can have severe bias problems given finite training samples due to changes in the class-conditional distributions in the neighborhood of the test sample [25]. An extreme example of this problem happens with repeated training samples. For example, when learning from web data, the same story originating from a news source can appear on multiple web pages. As a specific illustration of the problem, consider the case of ten training samples, all equally similar to the test sample, but seven of the training samples are repeats, or highly similar to each other. Uniform weighting or weights that were only a function of the $\{s(x, z_i)\}$ would cause the seven repeated samples to receive 7/10 of the weight, and might bias the classification. This problem of correlated or highly similar training examples leads to the *diversity* design goal for weighting nearest-neighbors: w_i and w_j should be decreasing functions of $s(z_i, z_j)$. These two goals are similar to goals in ranking [81].

7.2.4 Defining Similarity given Heterogeneous Features for the Purpose of Learning

The choice of similarity measure is an important aspect of similarity-based classification. For example, [13] found that using nearest neighbors with a modified version of the value difference metric [68] resulted in generally lower error rates than using Hamming distance (termed "overlap metric" in their paper). Section 2.1 described several ways to define similarity. Among them is the recently proposed *residual entropy similarity* $s_{re}(x, z)$ [10], defined in (2.5). Based on both set theory and information theory, the residual entropy similarity strongly captures context in assessing the similarity between samples by measuring the residual entropy in a random sample R. In the context of the random sample R, two rarely occurring samples x and z that share many features are judged more similar than more-frequently occurring samples: their features specify the random sample R with more certainty, and thus the residual entropy in R is lower for more uniquely similar samples.

Similarity functions that are information theoretic and/or set theoretic can be useful for learning from heterogeneous (or mixed) data. These similarity functions often require estimating the probabilities of different types of features (numerical, categorical, discrete, etc.), and estimating such probabilities from a small set of given training data is a difficult problem that can significantly affect their utility for learning. There are many open research questions about the interplay between similarity-based classifiers and similarity measures. A possible future research direction is exploring how to use different set-theoretic and information-theoretic similarity functions for similarity-based learning, with particular focus on the difficult case of small training sets.

7.2.5 Generative Classifiers for Mixed Features

An effective way to classify samples described by heterogeneous features is to build mixed models that combine traditional Euclidean feature space-based probability

models with similarity-based models. One solution to this is use mixtures of naive Bayes models, as proposed by Lowd and Domingos [42]. To make their solution work for mixed features, they quantized all Euclidean features to five quantization levels, then treated it as a categorical variable. Section 3.4 discussed a different approach which mixes Euclidean-space and similarity-space generative models. A research challenge will be estimating the parameters for these generative models which often require estimating many parameters. In particular, based on the success of Bayesian and regularization techniques [5, 27] and on the recent work by Srivastava et al. in Bayesian estimation for Gaussian generative models [67], Bayesian approaches can provide a practical method for regularization.

7.2.6 Weighting Nearest-Neighbors Given Mixed Features

Given some Euclidean features, basing nearest-neighbor weightings only on similarities may not be as effective as explicitly using the information in the Euclidean features. One approach would be to combine the weights based on the similarities with the weights based on the Euclidean features (using, for example, weights formed by local linear regression [27, 70] or by LIME). This approach poses the question of how to combine the weights. An effective way to solve for weights is to formulate weights to minimize first-order error or satisfy appropriate design goals [23, 26].

BIBLIOGRAPHY

[1] H. Akaike. A new look at the statistical identification model. *IEEE Trans. Auto Control*, AC-19:716–723, December 1974.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002.

[3] B. Bhattacharya, K. Mukherjee, and G. Toussaint. Geometric decision rules for high dimensions. *Proc. of the 55th Session of Intl. Statistics Institute*, pages 1–4, 2005.

[4] M. Bicego, V. Murino, M. Pelillo, and A. Torsello. Special issue on similarity-based classification. *Pattern Recognition*, 39, October 2006.

[5] P. J. Bickel and B. Li. Regularization in statistics. *Test*, 15(2):271–344, 2006.

[6] J. A. Bilmes. A gentle tutorial of the EM algorithm and its applications to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, International Computer Science Institute, April 1998.

[7] D. B. O 'Brien and R. M. Gray. Gaussian mixture model classifier for small objects in images. In *IEEE Int. Conf. on Image Processing*, volume 2, pages 850–853, September 2005.

[8] J. M. Buhmann and T. Hofmann. A maximum entropy approach to pairwise data clustering. In *Proc. of the Intl. Conf. on Pattern Recognition*, volume 2, pages 207–212, October 1994.

[9] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[10] L. Cazzanti and M. R. Gupta. Information-theoretic and set-theoretic similarity. In *Proc. of the IEEE Intl. Symposium on Information Theory*, pages 1836–1840, 2006.

[11] L. Cazzanti, M. R. Gupta, L. Malmström, and D. Baker. Quality assessment of low free-energy protein structure predictions. In *Proc. of the IEEE Workshop on Machine Learning for Signal Processing*, 2005.

[12] E. M. Chin, E. K. Garcia, and M. R. Gupta. Color management of printers by regression over enclosing neighborhoods. *Proc. of the IEEE Intl. Conf. on Image Processing*, 2007.

[13] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):57–78, 1993.

[14] T. Cover and J. Thomas. *Elements of Information Theory.* John Wiley and Sons, New York, 1991.

[15] L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer-Verlag Inc., New York, 1996.

[16] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley-Interscience, 2001.

[17] R. P. W. Duin, E. Pekalska, and D. de Ridder. Relational discriminant analysis. *Pattern Recognition Letters*, 20:1175–1181, 1999.

[18] B. S. Everitt and S. Rabe-Hesketh. *The Analysis of Proximity Data.* Arnold, London, 1997.

[19] I. Gati and A. Tversky. Weighting common and distinctive features in perceptual and conceptual judgments. *Cognitive Psychology*, (16):341–370, 1984.

[20] L. Goldfarb. A new approach to pattern recognition. *Progress in Pattern Recognition*, 2:241–402, 1985.

[21] T. Graepel, R. Herbrich, and K. Obermayer. Classification on pairwise proximity data. *Advances in Neural Information Processing Systems 11*, pages 438–444, 1999.

[22] M. R. Gupta. Custom color enhancements. *Proc. of the IEEE Intl. Conf. on Image Processing*, pages 968–971, 2005.

[23] M. R. Gupta, Y. Chen, and A. Marin. Weighted nearest neighbor similarity-based classifiers. *In review by the Advances in Neural Information Processing Systems 2007.*

[24] M. R. Gupta, E. K. Garcia, and E. M. Chin. Adaptive local linear regression with application to printer color management. *To appear in IEEE Trans. on Image Processing.*

[25] M. R. Gupta, R. M. Gray, and R. A. Olshen. Nonparametric supervised learning with linear interpolation and maximum entropy. *IEEE Trans. on Pattern Analysis and Machine Learning*, 28(5):766–781, 2006.

[26] M. R. Gupta and W. H. Mortensen. Weighted nearest neighbor classification. *In review by the Journal of Machine Learning Research.*

[27] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer-Verlag, New York, 2001.

[28] D. Hindle. Noun classification from predicate-argument structures. *Proc. of the ACL*, pages 268–275, 1990.

[29] S. Hochreiter, M. C. Mozer, and K. Obermayer. Coulomb classifiers: Generalizing support vector machines via an analogy to electrostatic systems. *Advances in Neural Information Processing Systems 15*, pages 545–552, 2003.

[30] S. Hochreiter and K. Obermayer. Support vector machines for dyadic data. *Neural Computation*, 18(6):1472–1510, 2006.

[31] T. Hofmann and J.M. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(1), January 1997.

[32] C.-W Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines.

[33] D. W. Jacobs, D. Weinshall, and Y. Gdalyahu. Classification with nonmetric distances: Image retrieval and class representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(6):583–600, June 2000.

[34] E. T. Jaynes. On the rationale for maximum entropy methods. *Proc. of the IEEE*, 70(9):939–952, September 1982.

[35] E. T. Jaynes. *Probability theory: the logic of science.* Cambridge University Press, 2003.

[36] M. I. Jordan. *An Introduction to Probabilistic Graphical Models*. To be published, 20xx.

[37] T. Knebel, S. Hochreiter, and K. Obermeyer. An SMO algorithm for the potential support vector machine. Technical report, Technische Univesität Berlin, 2007.

[38] W. Lam, C. Keung, and D. Liu. Discovering useful concept prototypes for classification based on filtering and abstraction. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(8):1075–1090, August 2002.

[39] M. Li, X. Chen, X. Li, B. Ma, and P M. B. Vitányi. The similarity metric. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 50(12):3250–3264, December 2004.

[40] L. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *Journal of Computational Biology*, 10(6):857–868, 2003.

[41] D. Lin. An information-theoretic definition of similarity. *Proc. of the Intl. Conf. on Machine Learning*, 1998.

[42] D. Lowd and P. Domingos. Naive Bayes models for probability estimation. In *Proc. 22nd Intl. Conference on Machine Learning (ICML)*, pages 529–536, Bonn, Germany, 2005. ACM Press.

[43] M. Lozano, J. M. Sotoca, J. S. Sánchez, F. Pla, E. Pekalska, and R. P. W. Duin. Experimental study on prototype optimisation algorithms for prototype-based classification in vector spaces. *Pattern Recognition*, 39:1827–1838, 2006.

[44] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, New York, 2003.

[45] The MathWorks, Natick, MA. *MATLAB: The Language of Technical Computing*, 2006 edition.

[46] Y. Mitani and Y. Hamamoto. Classifier design based on the use of nearest neighbor samples. *Proc. of the Intl. Conf. on Pattern Recognition*, pages 769–772, 2000.

[47] Y. Mitani and Y. Hamamoto. A local mean-based nonparametric classifier. *Pattern Recognition Letters*, 27(10):1151–1159, July 2006.

[48] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998.

[49] W. S. Noble. *Kernel Methods in Computational Biology*, chapter Support Vector Machine Applications in Computational Biology, pages 71–92. MIT Press, 2004.

[50] W. S. Noble. What is a support vector machine. *Nature Biotechnology*, 24(12):1565–1567, December 2006.

[51] R. Nock, M. Sebban, and D. Bernard. A simple locally adaptive nearest neighbor rule with application to pollution forecasting. *Intl. Journal of Pattern Recognition and Artificial Intelligence*, 17(8):1–14, 2003.

[52] A. Okabe, B. Boots, K. Sugihara, and S. Chiu. *Spatial Tessellations*, chapter 6, pages 418–421. John Wiley and Sons, Ltd., Chichester, England, 2000.

[53] E. Pekalska, R. P. W. Duin, and P. Paclík. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition Letters*, 39:189–208, 2006.

[54] E. Pekalska, P. Paclíc, and R. P. W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research*, pages 175–211, 2001.

[55] S. Philips, J. Pitton, and L. Atlas. Perceptual feature identification for active sonar echoes. In *IEEE OCEANS*, 2006.

[56] K. Pyun, J. Lim, C. S. Won, and R. M. Gray. Image segmentation using hidden Markov Gauss models. *IEEE Trans. Image Processing*, 16(7):1902–1911, July 2007.

[57] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. *Proc. of the Intl. Joint Conf. on Artificial Intelligence*, pages 448–453, 1995.

[58] D. A. Reynolds and R. C. Rose. Robust text-independent speaker-identification using Gaussian mixture speaker models. *IEEE Trans. on Speech and Audio Processing*, 3(1), 1995.

[59] S. Santini and R. Jain. Similarity is a geometer. *Multimedia Tools and Applications*, 5:277–306, 1997.

[60] S. Santini and R. Jain. Similarity measures. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(9):871–883, September 1999.

[61] S. Sattath and A. Tversky. On the relation between common and distinctive feature models. *Psychological Review*, (94):16–22, 1987.

[62] B. Schölkopf. The kernel trick for distances. *Advances in Neural Information Processing Systems*, 13, 2001.

[63] G. Schwartz. Estimating the dimension of a model. *The Annals of Statistics*, 5(2):461–464, 1978.

[64] G. Schwartz and A. Tversky. On the reciprocity of proximity relations. *Journal of Mathematical Psychology*, 22(3):301–307, September 1980.

[65] R. Sibson. *Interpreting multivariate data*, chapter A brief description of natural neighbour interpolation, pages 21–36. John Wiley, 1981.

[66] P. Simard, Y. Le Cun, and J. Denker. Efficient pattern recognition using a new transformation distance. *Advances in Neural Information Processing Systems 5*, pages 50–68, 1993.

[67] S. Srivastava, M. R. Gupta, and B. A. Fryigik. Bayesian quadratic discriminant analysis. *Journal of Machine Learning Research*, 8:1287–1314, 2007.

[68] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, December 1986.

[69] C. Steinhoff, T. Müller, U. A. Nuber, and M. Vingron. Gaussian mixture density estimation applied to microarray data. In M. R. Berthold, H.-J. Lenz, E. Bradley, and R. Kruse, editors, *Advances in Intelligent Data Analysis V - Proc. Int. Symp. Intelligent Data Analysis*, volume 5 of *Lecture Notes in COmputer Science*, pages 418–429, Berlin, Germany, August 2003. Springer.

[70] C. J. Stone. Consistent nonparametric regression. *The Annals of Statistics*, 5(4):595–645, 1977.

[71] A. Tversky. Features of similarity. *Psychological Review*, (84):327–352, 1977.

[72] A. Tversky and I. Gati. Studies of similarity. In E. Rosch and B. Lloyd, editors, *Cognition and Categorization*. Earlbaum, Hillsdale, N.J., 1978.

[73] A. Tversky and J. W. Hutchinson. Nearest neighbor analysis of psychological spaces. *Psychological Review*, 93:3–22, 1986.

[74] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, NY, 1998.

[75] D. Weinshall, D. W. Jacobs, and Y. Gdalyahu. Classification in non-metric spaces. *Advances in Neural Information Processing Systems 11*, pages 838–844, 1999.

[76] P. Willet, J. M. Barnard, and G. M. Downs. Chemical similarity searching. *Journal of Chemical Information and Computer Sciences*, 38(6):983–996, 1998.

[77] D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.

[78] G. Young and A. S. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19–22, 1938.

[79] B. Zhang and S. Shrihari. Discovery of the tri-edge inequality with several binary dissimilarity measures. *Proc. of the Intl. Conf. on Pattern Recognition*, 4:669–672, August 2004.

[80] H. Zhang, A. C. Berg, M. Maire, and J. Malik. SVM-KNN: discriminative nearest neighbor classification for visual category recognition. *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2126 – 2136, 2006.

[81] X. Zhu, A. B. Goldberg, J. Van Gael, and D. Andrzejewski. Improving diversity in ranking using absorbing random walks. In *Proc. Human Language Technologies*, pages 97–104, 2007.

VITA

Luca Cazzanti conducts research in statistical learning and signal processing at the Applied Physics Laboratory – University of Washington, in Seattle, USA. Before moving to Seattle, he attended the University of Wisconsin in Madison, where he received his B.S. and M.S. degrees in electrical engineering in 1996 and 1998. He began his engineering studies at the Politecnico di Milano in Milano, Italy.