DISCRIMINATIVE VS INFORMATIVE LEARNING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF STATISTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Y. Dan Rubinstein
January 1998

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Trevor Hastie
(Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Jerome Friedman

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Art Owen

Approved for the University Committee on Graduate Studies:

Abstract

The problem of classification can be approached from two points of view: informative - where the classifier learns the class densities, or discriminative - where the focus is on learning the boundaries (logits) between classes without regard to the underlying class densities.

I present a method of modeling classes with linear subspaces called HyperPancakes. Subspace models are useful to represent high-dimensional data. The purpose of HyperPancakes is to find subspaces that best discriminate between classes. Standard subspace methods (eg Watanabe's CLAFIC) find subspaces that best represent the data. However "representation is not discrimination", and this led to the development of the Discriminative/Informative (D/I) framework.

The D/I framework presents two ways of estimating class membership probabilities: one based on class density parameters estimated from the full likelihood, the other based on class logit parameters estimated from the conditional likelihood. I review classical results of Efron and O'Neill comparing Linear Discriminant Analysis vs Logistic Regression and show how the D/I framework applies to modern statistical techniques of GAM and Naive Bayes.

One of the main observations that emerges from the D/I analysis, is that the informative approach, which is an indirect method of learning class memberships, often leads to superior classification. My Separate Basis Procedure (SBP), extends Kooperberg and Stone's Polyclass method by combining discriminative and informative approaches. Polyclass is an adaptive, non-parametric method which models class boundaries as linear in a spline basis, with the same basis for all class boundaries. SBP allows each class boundary to have a separate basis. Simulation examples and experiments on real data motivate why this added flexibility is advantageous.

Acknowledgement

Trevor Hastie, Jerry Friedman, Art Owen, Walter Murray for providing guidance and support throughout my research.

My family. Especially my brother.

Emmanuel Candes. Andrew Huibers. Andrew Gordon. Tzipor Ulman. Barak Kol. Assaf Zeevi.

Contents

Abstract **iv**

Acknowledgement **v**

1 Introduction **1**

1.1 Principal Components 1
1.2 Fisher LDA 2
1.3 Optimal Classification Subspace 3
1.4 Motivation 5
1.5 Single Class case 5
1.6 Likelihood Model 8
1.7 Summary of Algorithm 10
1.8 Simulation Results 10
1.9 Appendix - Matrix function derivatives 10

2 HyperPancakes **11**

2.1 Subspace models in classification 11
2.1.1 Why subspace models? 12
2.1.2 Details 13
2.1.3 Relationship to Mahalanobis Distance 14
2.1.4 Decision boundaries 15
2.1.5 Number of model parameters 15
2.2 Finding the best subspaces 16
2.2.1 Principal Components 16
2.2.2 Fisher LDA 17
2.3 Testing whether the data lie in a subspace 17

2.4 Geometrical approach 18
2.5 Maximum Likelihood Formulation: Hyperpancakes 20
2.5.1 Score equations 20
2.5.2 Direct Optimization via Gradient Descent 21
2.6 Mixed Likelihood: Discrimination and Representation 21
2.7 Learning Subspace Method 21
2.7.1 Mixtures of Probabilistic Principal Component Analysers 22
2.8 Discriminative Training 23
2.8.1 Bayes Vector Quantization 24
2.8.2 Representation vs Discrimination 24
2.9 Choice of weights and centroids 28
2.10 Digit data 30

3 DISCRIMINATION **32**
3.1 Discriminative Training - Introduction 32
3.1.1 Bayes Classifier 32
3.1.2 Discriminative Training 33
3.1.3 LDA vs Logistic Regression 34
3.1.4 Vector Quantization vs Bayes VQ 35
3.1.5 Naive Bayes vs GAM 35
3.2 Combining Representation with Discrimination 36
3.2.1 Maximum Likelihood Estimation 39
3.2.2 Convex Combination of Procedures 39
3.2.3 Simulation of Combined Procedures 42
3.2.4 Analysis of simulation results 46
3.3 Mixed Likelihood as Regularized Discrimination 47
3.4 Regularized Conditional Likelihood 51
3.5 Descent Procedeures 53
3.6 Gaussian Classes 53
3.7 Tangent Distance 53
3.8 Gradient Descent Procedure 53
3.9 Unconstrained MLE 55
3.10 Logistic regression ridged towards MLE 55

4 Lospline Densities **61**
4.1 Naive Bayes vs GAM . . . 62
4.2 Simulation Data . . . 62
4.2.1 Estimating the logspline density . . . 62
4.3 Performance using small training sets . . . 69
4.3.1 Acceptance-Rejection . . . 69

5 O'Neill **70**
5.0.2 Asymptotic Efficiency of Logsplines . . . 70
5.0.3 Example . . . 73

6 Separate Basis Procedure **75**
6.1 POLYCLASS with separate basis . . . 75
6.1.1 Example . . . 76
6.1.2 Choice of constraint . . . 77
6.1.3 Basis Selection . . . 78
6.1.4 Polyclass Model . . . 78
6.1.5 Polyclass Details . . . 79
6.1.6 Model Parameters . . . 79
6.1.7 Constrained Parameters . . . 80
6.1.8 Effect of base class on logit . . . 81
6.2 Separate Basis Procedure (SBP) . . . 82
6.2.1 2D Example: Ugly . . . 82
6.2.2 Logits . . . 83
6.2.3 SB Model Evolution . . . 83
6.2.4 Comparisons . . . 85
6.3 SBP using Informative approach . . . 86
6.3.1 Description of SBP Algorithm . . . 86
6.3.2 Likelihood Gradient and Hessian . . . 87
6.3.3 The Rao statistic . . . 87
6.3.4 Local NR vs full NR . . . 87
6.3.5 Matrix Inverse when submatrix inverse known . . . 90
6.4 $K-1$ Gaussians + Ugly Example . . . 90

6.4.1 L example 97
6.4.2 Vowel Data 99

7 Conclusions and Suggestions for Future Work **103**

A Derivation of score equations **104**

Bibliography **105**

List of Figures

1.1 The simulation data. There are 3 classes, each generated according to a gaussian distribution. 1
1.2 Principal component solution and misclassified observations 2
1.3 Fisher LDA subspace 3
1.4 Projected data onto Fisher LDA subspace and misclassifications 3
1.5 Optimal subspaces and misclassifications 4

2.1 Illustration of subspace classification. 12
2.2 Principal component subspaces and optimal classification subspaces. 20
2.3 Clockwise from top-left: (a) Using gaussian class weights. (b) Using empirical distribution (1/0) weights. (c) Using empirical dist weights and ignoring outliers (based on nearest 0.70 of the data in each class) (d) Based on Lagrangian multiplier weights 30
2.4 The digit data projected onto the first 2 Fisher discriminants, with the principal component subspaces, and the resulting misclassifications. Also shown is the misclassification using the optimal subspaces. 30
2.5 Energy Spectra for the digit data. Each graph shows the spectrum of the subspace using the class covariance matrix eigenvectors as a basis. Each colum corresponds to a class. The top row is for the principal component subspaces, the bottom row is for the optimal classification subspaces. 30
2.6 The distribution of distances to the class subspaces. The left plot is for the 5-principal component subspace, the right plot is for the optimal subspace. The distances are divided into 2 groups: the distances to the correct class, and the distances to the incorrect classes. 31

2.7 Number of misclassifications as a function of subspace dimension, comparing principal components and OCS 31

3.1 Classification Error comparing three techniques: LDA, GLM (logistic regression), naive bayes (with univariate gaussians along each dimension). Two gaussian classes in $\mathcal{R}^2$ were generated with $mu_0 = (0, 0)$, $mu_1 = (2, 0)$, and $\Sigma_{i,j} = 0.7^{|i-j|}$. A test set of 20000 observations was classified according to models obtained using different training set sizes. For each training set size, 25 models were trained, and the error bars indicate the variation of the misclassification rate about the average over the 25 repetitions. 37

3.2 Classification Error comparing three techniques: LDA, GLM (logistic regression), naive bayes (with univariate gaussians along each dimension). Two gaussian classes in $\mathcal{R}^2$ were generated with $mu_0 = (0, 0)$, $mu_1 = (2, 0)$, and $\Sigma_{i,j} = diag(1, 3)$. A test set of 20000 observations was classified according to models obtained using different training set sizes. For each training set size, 25 models were trained, and the error bars indicate the variation of the misclassification rate about the average over the 25 repetitions. 38

3.3 Misclassification rate as function of the mixing parameter for a random sample of 5/50 training sets from each of 3 populations. Top row is for cauchy classes, middle is gaussian, bottom is mixture. $\lambda = 0$ corresponds to LOGREG, $\lambda = 1$ corresponds to NLDA. As expected, for gaussian data, NLDA outperforms LOGREG most of the time. Pay attention to the vertical scale which is different for each plot. Solid lines represent actual test error, dotted lines are cross validated estimates of the test error obtained from the training data by a leave one out procedure. 45

3.4 QQplot for random projections of the training data. The accompanying fitted line is a robust LMS regression line. 47

3.5 Conditioning plot of range vs min location, conditioned on gof across all 150 training sets from all 3 populations. 48

3.6 Pairwise plots of min location, misclassification range, and gof. 49

3.7 Histogram of min location broken down by gof and range. The bottom horizontal bar refers to gof, the upper to range. 50

3.8 The MLE-ridged likelihood profile: $-(1-\lambda)L(y|x) + \lambda||\beta - \hat{\beta_{MLE}}||^2$ about β_{MLE}. Each row corresponds to a different value of the mixing term lambda, each column corresponds to a different dimension of β. The solid vertical line corresponds to β_{LOGREG}, the dashed line to β_{MLE}. This profile corresponds to the 1st cauchy training set (ie top left plot in figure 3.10). Note that the (negative) likelihood is lowest in the $\lambda = 0$ case. So what is the relationship between the ridged likelihood score and classification performance? 57

3.9 The MLE-ridged likelihood profile: $-(1-\lambda)L(y|x) + \lambda||\beta - \hat{\beta_{MLE}}||^2$ about β_{MLE}. Each row corresponds to a different value of the mixing term lambda, each column corresponds to a different dimension of β. The solid vertical line corresponds to β_{LOGREG}, the dashed line to β_{MLE}. This profile corresponds to the 1st gaussian training set (ie 2ndr row, 1st col plot in figure 3.10). Note that the (negative) likelihood is lowest for $\lambda = 0$, around 3000. 58

3.10 Misclassification curves as a function of the mixing parameter. Top row - cauchy data, middle row - gaussian data, bottom row - mixture data (p=5, numclass = 2). Columns correspond to unique training sets. Height of curve is misclassification rate. Solid line - mixed discriminants, x/o - mixed likelihood (converged/not converged), +/= - MLE ridged conditional likelihood (converged/not). The gof score above each graph is the maximum Kolmogorov-Smirnov statistic among 50 randomly chosen univariate projections of the data. 59

4.1 Univariate projections of Class 1's density function. 61

4.2 Effect of NCS exponential tilts on a bivariate gaussian density. 63

4.3 Effect of NCS exponential tilts along both dimensions on a bivariate gaussian density. 64

4.4 The contours of the class densities of the simulation data. 65

4.5 The marginal class densities (dotted), and the Naive Bayes fitted logspline densities (solid). 66

5.1 Class densities for logspline example. 73

6.1 Using logspline density estimation to determine candidate knot locations.. . 86

6.2 Comparison of Rao statistics using full NR vs local NR. 89

6.3 Ugly example: Training data and Bayes classification regions. 91
6.4 Polyclass and linear model. 92
6.5 Hand picked model, and knots used. Top left: common knots used when classes share the same knots. Otherwise, knots are determined using logspline separately in each class for each dimension. 92
6.6 Same knots in each class, candidates shared. Left: Single NR step to evaluate candidate. Right: Exact fit for each candidate. 93
6.7 Left: Same knots in each class, candidates NOT shared. Right: Each class has its own knots, candidates NOT shared. 93
6.8 Left: BIC criterion: penalizing the likelihood by the number of terms in the model. Right: The corresponding model evolution. 94
6.9 L example: Training data and Bayes classification regions. 98
6.10 Polyclass and linear model. 98
6.11 SBP model. 99
6.12 Vowel training data projected onto first 2 principal components, and plotted by class. 100
6.13 The BIC criterion function for the vowel data. 102

Chapter 1

Introduction

We have observations $x_i \in \mathcal{R}^p$ for $i = 1, \ldots, n$, and class membership variables $C_i \in \{1, \ldots, J\}$. There are n_j observations belonging to class j. The centroid (mean) of the j^{th} class is $\mu_j = \frac{1}{n_j} \sum_{C_i=j} x_i$. The class sample covariances are $S_j = \frac{1}{n_j} \sum_{C_i=j} (x_i - \mu_j)(x_i - \mu_j)^T$. We also define the covariance matrix for the m^{th} class centered about the j^{th} class to be

$$R_m^{(j)} = \frac{1}{n_m} \sum_{C_i=m} (x_i - \mu_j)(x_i - \mu_j)^T.$$

Figure 1.1: The simulation data. There are 3 classes, each generated according to a gaussian distribution.

Figure (1.1) shows the simulation data for 3 multivariate gaussian classes. The means of the classes are at (0,0), (4,0), (2,0). The covariances for classes 1 and 2 are $\begin{pmatrix} 1 & 0 \\ 0 & 0.2 \end{pmatrix}$, and for class 3 $\begin{pmatrix} 0.2 & 0 \\ 0 & 1 \end{pmatrix}$.

Before presenting our solution to the optimal subspace method, we review two classical methods of dimensionality reduction.

1.1 Principal Components

Representation not discrimination. Subspace chosen separately for each class. Solution is eigenvectors of class covariance matrices. Discretized version of Karhunen-Loeve.

The principal components of a covariance matrix are those directions along which the data varies the most. They are a solution to the problem: choose $k < p$ vectors for each class that best *represent* it. Formally, for the j^{th} class, find $V = [V_1, \ldots, V_k]$ such that

$$\sum_{m=1}^{k} V_m^T S_j V_m$$

is minimized, subject to the orthonormality constraint: $V^T V = I_k$. The solution can be obtained analytically by the method of Lagrange multipliers by finding the stationary points of (note: you can include constraints for the cross terms $V_m V_n$ but the lagrange multipliers for the cross terms are all zero).

$$\sum_{m=1}^{k} V_m^T S_j V_m - \sum_{m=1}^{k} \lambda_m (V_m^T V_m - 1).$$

Differentiating with respect to V_m, and setting equal to 0 gives us the eigenvector equation

$$S_j V_m = \lambda_m V_m.$$

The original sum $\sum_{m=1}^{k} V_m^T S_j V_m = \sum_{m=1}^{k} \lambda_m$ is maximized by choosing those V_m corresponding to the largest eigenvalues. Thus the best k-dimensional subspace, in terms of *representing* each class, is given by the principal eigenvectors.

Figure 1.2: Principal component solution and misclassified observations

Figure (1.2) shows the subspaces obtained by choosing the first principal component in each class.

1.2 Fisher LDA

Single subspace for all classes, such that between class separation is maximized subject to keeping the within class scatter fixed. Solution is eigenvectors of $W^{-1}B$.

Fisher proposed looking for a single subspace such that the classes are well separated in that subspace. The between class scatter is given by $B = \sum_j^J (\mu_j - \mu_{\text{tot}})(\mu_j - \mu_{\text{tot}})^T$. The pooled within class scatter is given by $W = \sum_j^J n_j S_j$.

Classification is performed by projecting the data onto the subspace, and then finding the nearest projected centroid.

Figure 1.3: Fisher LDA subspace

Figure 1.4: Projected data onto Fisher LDA subspace and misclassifications

Figure (1.4) shows the global subspace given by the Fisher LDA solution.

1.3 Optimal Classification Subspace

We seek subspaces $V^1, \ldots, V^J$ such that the perpendicular distance of observations in class j to V^j is minimized (on average), while simultaneously maximizing distance to V^m for $m \neq j$. Let $V^* = [V : V^\perp]$, a complete, orthnormal basis for $\mathcal{R}^p$, ie $V^{*T}V^* = I_p$. For any x, we can write

$$x = \sum_{a=1}^{p} V_a^* < V_a^*, x >= \sum_{a=1}^{k} V_a < V_a, x > + \sum_{a=1}^{p-k} V_a^\perp < V_a^\perp, x > .$$

The first term is the projection of x onto the subspace spanned by V, the second term is the component of x orthogonal to V. The norm of the second term gives us the perpendicular distance of x to V:

$$d(x, V) = \| \sum_{a=1}^{p-k} V_a^\perp < V_a^\perp, x > \|^2 = \sum_{a=1}^{p-k} < V_a^\perp, x >^2$$

The expected value (wrt to the empirical distribution) of the perpendicular distance of an observation from class m centered about the j^{th} mean to the subspace V is

$$Ed(x-\mu_j, V) = \sum_{a=1}^{p-k} E < V_a^\perp, x-\mu_j >^2 = \sum_{a=1}^{p-k} V_a^{\perp T} E[(x-\mu_j)(x-\mu_j)^T] V_a^\perp = \sum_{a=1}^{p-k} V_a^{\perp T} R_m^{(j)} V_a^\perp.$$

For the j^{th} class we seek a V such that $Ed(x - \mu_j, V)\mathbf{1}(C_i = j)$ is minimized, and $\sum_{m \neq j} Ed(x - \mu_j, V)\mathbf{1}(C_i = m)$ is maximized. This is a hard problem because $d(x - \mu_j, V)$ is convex in V and minimizing a convex function subject to concave constraints is "hard" [DAN: refer Boyd, develop convexity issues, link to score equations].

To make this problem "easy" we convert it to a single minimization of a convex function. The minimum thus obtained will be global. First, for each j we scale $R_1^{(j)}, \ldots, R_J^{(j)}$ so that the maximum eigenvalue is one: $\lambda(R_m^{(j)}) \leq 1$ for $m = 1, \ldots, J$. Note that the $R_m^{(j)}$ are positive definite by construction so $0 \leq \lambda(R_m^{(j)}) \leq 1$. We can therefore bound the average perpendicular distance:

$$Ed(x - \mu_m, V)\mathbf{1}(C_i = m) = \sum_{a=1}^{p-k} V_a^{\perp T} R_m^{(j)} V_a^{\perp} \leq \sum_{a=1}^{p-k} 1 = p - k.$$

Maximizing $Ed(x-\mu_j, V)\mathbf{1}(C_i = m)$ is the same is minimizing $p-k-Ed(x-\mu_j, V)\mathbf{1}(C_i = m)$. Now we can transform our original problem into finding that V which minimizes

$$\begin{aligned} Ed(x - \mu_j, V)\mathbf{1}(C_i = j) + \sum_{m \neq j} (p - k) - Ed(x - \mu_j, V)\mathbf{1}(C_i = m) &= \\ \sum_{a=1}^{p-k} V_a^{\perp T} R_j^{(j)} V_a^{\perp} + \sum_{m \neq j} (p - k) - \sum_{a=1}^{p-k} V_a^{\perp T} R_m^{(j)} V_a^{\perp} &= \\ \sum_{a=1}^{p-k} V_a^{\perp T} R_j^{(j)} V_a^{\perp} + \sum_{m \neq j} \sum_{a=1}^{p-k} V_a^{\perp T} V_a^{\perp} - V_a^{\perp T} R_m^{(j)} V_a^{\perp} &= \\ \sum_{a=1}^{p-k} V_a^{\perp T} [R_j^{(j)} + \sum_{m \neq j} (I - R_m^{(j)})] V_a^{\perp} & \end{aligned}$$

The last equation is again a quadratic form, for which the $V_a^{\perp}$ that minimize it are given by the eigenvectors of $R_j^{(j)} + \sum_{m \neq j}(I - R_m^{(j)})$ corresponding to the smallest eigenvalues. The best classification subspace for class j is then given by the orthogonal space to $V^{\perp}$, namely V, given by the k principal components of $R_j^{(j)} + \sum_{m \neq j}(I - R_m^{(j)})$.

Figure 1.5: Optimal subspaces and misclassifications

Figure (1.5) shows the optimal subspaces solution. Note the improvement in correct classifications. This was actually obtained with a robustified version of the above algorithm, where tokens that are outliers in their own class are ignored. Note also that further improvement is possible if we don't constrain the subspaces to pass through the class mean - we can estimate the point through which the subspace should pass as well.

1.4 Motivation

tangent distance and handwritten digit stuff here, esp choosing linear subspace using SVD. Choosing a subspace such that members of that class are well represented in the subspace while non-members are poorly represented. Relation to constrained logistic regression.

1.5 Single Class case

In order to simplify the discussion in the next section, we first review the computations involved in the case of a single class. We are given data $x_1, \ldots, x_n$ which we suppose was generated by a multivariate normal distribution

$$P(x) = \mathcal{N}(\mu, \Sigma)[x] = \frac{1}{\sqrt{(2\pi)^p \det \Sigma}} \exp -(\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)),$$

where we model

$$\Sigma_{pxp} = \Lambda_{pxk}\Lambda'_{kxp} + \epsilon I_p.$$

The log likelihood of the data, as a function of the parameters Λ and ϵ is (neglecting constants and terms that depend only on the data)

$$\begin{aligned} L(\Lambda, \epsilon) &= \sum_{i=1}^{n} -\frac{1}{2}\left[\log(\det \Sigma) + (x_i-\mu)'\Sigma^{-1}(x_i-\mu))\right] \\ &= -\frac{n}{2}\log(\det \Sigma) - \frac{n}{2}\mathrm{Tr}(\Sigma^{-1}S) \end{aligned}$$

where $S = \frac{1}{n}\sum_{i=1}^{n}(x_i-\hat{\mu})(x_i-\hat{\mu})'$ and $\hat{\mu}$ is typically taken to be $\bar{x} = \sum_{i=1}^{n} x_i$. We will now show that the MLE's for Λ and ϵ are given by:

$$\hat{\epsilon} = \left[\mathrm{Tr}(S) - (\gamma_1 + \cdots + \gamma_k)\right]/(p-k)$$

and that $\hat{\Lambda}$ satisfies the eigenvector equation

$$S\hat{\Lambda} = D_\gamma \hat{\Lambda}$$

where the norm of the eigenvectors $\hat{\Lambda}_i$ satisfies

$$\| \hat{\Lambda}_i \|^2 = \gamma_i - \hat{\epsilon}$$

and the eigenvectors are those corresponding to the k largest out of p eigenvalues γ_i for $i = 1, \ldots, k$.

The MLE's are obtained by setting the partial derivatives of the log likelihood equation to zero. The following matrix function derivatives are needed (see appendix):

$$\begin{aligned}
\frac{\partial \log(\det \Sigma)}{\partial \Lambda} &= 2\Sigma^{-1}\Lambda \\
\frac{\partial \log(\det \Sigma)}{\partial \epsilon} &= 2\mathrm{Tr}(\Sigma^{-1}) \\
\frac{\partial \mathrm{Tr}(\Sigma^{-1}S)}{\partial \Lambda} &= -2\Sigma^{-1}S\Sigma^{-1}\Lambda \\
\frac{\partial \mathrm{Tr}(\Sigma^{-1}S)}{\partial \epsilon} &= -\mathrm{Tr}(\Sigma^{-1}S\Sigma^{-1})
\end{aligned}$$

Differentiating the log likelihood and setting equal to zero,

$$\begin{aligned}
\frac{\partial L}{\partial \Lambda} &= 0 \\
&= \Sigma^{-1}(\Sigma - S)\Sigma^{-1}\Lambda \\
\Rightarrow (\Sigma - S)\Sigma^{-1}\Lambda &= 0
\end{aligned}$$

and using the identity

$$\Sigma^{-1} = \frac{1}{\epsilon} I - \frac{1}{\epsilon^2}\Lambda(I + \Delta)^{-1}\Lambda'$$

where $\Delta = \frac{1}{\epsilon}\Lambda'\Lambda$ is diagonal with $\Delta_{ii} = \frac{1}{\epsilon}\| \Lambda_i \|^2$ (this is one way of specifying Λ which can otherwise be multiplied by any orthogonal matrix without changing Σ), we can write

$$\Sigma^{-1}\Lambda = \frac{1}{\epsilon}\Lambda(I + \Delta)^{-1}$$

so $(\Sigma - S)\Sigma^{-1}\Lambda = (\Sigma - S)\frac{1}{\epsilon}\Lambda(I + \Delta)^{-1} = 0$. Now postmultiply by $(I + \Delta)^{-1}$ and write $\Sigma = \Lambda\Lambda' + \epsilon I$ giving

$$(\Lambda\Lambda' + \epsilon I - S)\frac{1}{\epsilon}\Lambda = 0$$

$$\begin{aligned}
\frac{1}{\epsilon}S\Lambda &= \frac{1}{\epsilon}\Lambda\Lambda'\Lambda + \Lambda \\
&= \Lambda\Delta + \Lambda \\
&= \Lambda(I + \Delta) \\
\Rightarrow S\Lambda &= \Lambda \underbrace{(\epsilon I + \Lambda\Lambda')}_{D_\gamma}
\end{aligned}$$

We have then, as required, that $S\hat{\Lambda} = D_\gamma\hat{\Lambda}$ where $\| \hat{\Lambda}_i \|^2 = \gamma_i - \hat{\epsilon}$.

The MLE for ϵ is obtained by noting that

$$\begin{aligned}
\log\det(\Sigma) &= \log\det(\Lambda\Lambda' + \epsilon I) \\
&= \log\epsilon^p \det(\frac{1}{\epsilon}\Lambda\Lambda' + I_p) \\
&\stackrel{(a)}{=} \log\epsilon^p \det(\frac{1}{\epsilon}\Lambda'\Lambda + I_k) \\
&= \log\epsilon^p \det(\Delta + I_k)
\end{aligned}$$

where we have used the fact (a) that the eigenvalues of $A_{pxq}B_{qxp}$, $p > q$ are the same as BA, so that $\det(I_p + AB) = \det(I_q + BA)$. So

$$\begin{aligned}
\log\det(\Sigma) &= p\log\epsilon + \sum_{i=1}^{k}\log(\Delta_{ii} + 1) \\
\frac{\partial\log\det(\Sigma)}{\partial\epsilon} &= \frac{1}{\epsilon}(p - k)
\end{aligned}$$

We also have that

$$\begin{aligned}
S\Sigma^{-1} &= S\Sigma^{-1}(\epsilon I)(\frac{1}{\epsilon}I) \\
&= S\Sigma^{-1}(\Sigma - \Lambda\Lambda')(\frac{1}{\epsilon}I) \\
&= \frac{1}{\epsilon}S - \frac{1}{\epsilon}(S\Sigma^{-1}\Lambda)\Lambda' \\
&\stackrel{(a)}{=} \frac{1}{\epsilon}S - \frac{1}{\epsilon}\Lambda\Lambda' \\
\Rightarrow \mathrm{Tr}(S\Sigma^{-1}) &= \frac{1}{\epsilon}\mathrm{Tr}(S) - \mathrm{Tr}(\Delta)
\end{aligned}$$

$$\Rightarrow \frac{\partial \mathrm{Tr}(S\Sigma^{-1})}{\partial \epsilon} = -\frac{1}{\epsilon^2}\mathrm{Tr}(S) + \frac{1}{\epsilon^2}\sum_{i=1}^{k}\gamma_i$$

where (a) we have used the fact that $S\Sigma^{-1}\Lambda = \Lambda$ which follows from the score equation for Λ where we had $(\Sigma - S)\Sigma^{-1}\Lambda = 0$. Note that Δ is diagonal with $\Delta_{ii} = \frac{1}{\epsilon}\| \Lambda_i \|^2 = \frac{1}{\epsilon}(\gamma_i - \epsilon)$.

Combining these results we obtain

$$\begin{aligned} \frac{\partial L}{\partial \epsilon} &= 0 \\ &= \frac{(p-k)}{\epsilon} - \frac{1}{\epsilon^2}\mathrm{Tr}(S) + \frac{1}{\epsilon^2}\sum_{i=1}^{k}\gamma_i \\ \Rightarrow \hat{\epsilon} &= \left[\mathrm{Tr}(S) - (\gamma_1 + \cdots + \gamma_k)\right]/(p-k) \end{aligned}$$

as required.

1.6 Likelihood Model

DAN: ADD: λ and full/conditional likelihood discuss induced decision surfaces (compare to logistic regression).

We are concerned with classification of $x \in \Re^p$ into one of K classes. We model the class conditional distributions as "flat" Gaussians, or pancakes, where the variance in a k-dimensional ($k < p$) subspace is much larger relative to the remaining $p - k$ dimensions. This can be visualized as a pancake (fig. 1). The distribution of x conditioned on being in class j is given by: $P(x \mid \mathcal{C}_j) = \mathcal{N}(\mu^j, \Sigma^j)[x] = \frac{1}{\sqrt{2\pi^p \det \Sigma^j}} \exp -(\frac{1}{2}(x-\mu^j)'(\Sigma^j)^{-1}(x-\mu^j))$. We model

$$\Sigma^j{}_{pxp} = \Lambda^j{}_{pxk}\Lambda^{j'}{}_{kxp} + \epsilon^j I_p$$

where ϵ can be thought of as the thickness of the pancake and the k columns of Λ span the subspace of interest.

Given the data $x_1, \ldots, x_n$ and the class memberships $\mathcal{C}_{x_i}$, we wish to find Λ^j and ϵ^j such that the multinomial likelihood of the data is maximized. That is, we wish to maximize $P(\mathcal{C}_j \mid x_i)$ for each i. The multinomial likelihood of the data is given by

$$\prod_{i=1}^{n} P(\mathcal{C}_j \mid x_i) = \prod_{i=1}^{n} \frac{P(x_i \mid \mathcal{C}_j)P(\mathcal{C}_j)}{\sum_{l=1}^{K} P(x_i \mid \mathcal{C}_l)P(\mathcal{C}_l)}.$$

Assuming the class priors are equal (so that we can factor out the $P(\mathcal{C}_j)$ terms), and maximizing the log of the likelihood yields

$$\begin{aligned} L(\Lambda_1,\ldots,\Lambda_K,\epsilon_1,\ldots,\epsilon_k) &= \sum_{i=1}^{n}\left[\log P(x_i \mid \mathcal{C}_j) - \log\left(\sum_{l=1}^{K} P(x_i \mid \mathcal{C}_l)\right)\right] \\ &= \sum_{i=1}^{n} -\frac{1}{2}\log(\det(\Sigma^j)) - \frac{1}{2}(x_i-\mu^j)'(\Sigma^j)^{-1}(x_i-\mu^j) - \log\left[\sum_{l=1}^{K} P(x_i \mid \mathcal{C}_l)\right] \\ &= \sum_{j=1}^{K} -\frac{n_j}{2}\log(\det(\Sigma^j)) - \frac{n_j}{2}\mathrm{Tr}(S^j(\Sigma^j)^{-1}) - \sum_{i=1}^{n}\log\left[\sum_{l=1}^{K} P(x_i \mid \mathcal{C}_l)\right] \end{aligned}$$

where $S^j = \frac{1}{n_j}\sum_{i=1}^{n_j}(x_i-\mu^j)(x_i-\mu^j)'$ and n_j is the number of tokens in $\mathcal{C}_j$.

We propose an iterative procedure for finding the optimal Λ^j and ϵ^j that draws on results from Factor Analysis. Differentiating the likelihood and setting equal to zero,

$$\begin{aligned} \frac{\partial L}{\partial \Lambda^j} &= -n_j(\Sigma^j)^{-1}\Lambda^j + n_j(\Sigma^j)^{-1}S^j(\Sigma^j)^{-1}\Lambda^j - \sum_{i=1}^{n}\frac{P(x_i \mid \mathcal{C}_j)}{\sum_{l=1}^{K}P(x_i \mid \mathcal{C}_l)}\frac{\partial(\log(P(x_i \mid \mathcal{C}_j)))}{\partial\Lambda^j} \\ &= -n_j(\Sigma^j)^{-1}\Lambda^j + n_j(\Sigma^j)^{-1}S^j(\Sigma^j)^{-1}\Lambda^j - \sum_{i=1}^{n} w_i^j(-(\Sigma^j)^{-1}\Lambda^j + (\Sigma^j)^{-1}S_i^j(\Sigma^j)^{-1}\Lambda^j) \\ &= (\Sigma^j)^{-1}[(-n_j + \sum_{i=1}^{n} w_i^j)\Sigma^j + n_j\Sigma_j - \sum_{i=1}^{n} w_i^j S_i^j](\Sigma^j)^{-1}\Lambda^j \\ &= 0, \end{aligned}$$

where $S_i^j = (x_i-\mu^j)(x_i-\mu^j)'$. Premutliplying by Σ^j, and using the identity $(\Sigma^j)^{-1} = \frac{1}{\epsilon^j}I - \frac{1}{\epsilon^2}\Lambda^j(I+\Delta)^{-1}\Lambda^{j\prime}$ to write $(\Sigma^j)^{-1}\Lambda^j = \frac{1}{\epsilon^j}\Lambda^j(I+\Delta)^{-1}$ where $\Delta = \frac{1}{\epsilon^j}\Lambda^{j\prime}\Lambda^j$ is diagonal (since any orthonormal transformation of Λ^j leaves Σ^j unchanged, this is one way of specifying Λ^j), we obtain

$$\begin{aligned} [(-n_j + \sum_{i=1}^{n} w_i^j)\Sigma^j + n_j\Sigma_j - \sum_{i=1}^{n} w_i^j S_i^j]\frac{1}{\epsilon^j}\Lambda^j(I+\Delta)^{-1} &= 0 \\ \left[(-n_j + \sum_{i=1}^{n} w_i^j)(\Lambda^j\Lambda^{j\prime} + \epsilon^j) + n_j\Sigma_j - \sum_{i=1}^{n} w_i^j S_i^j\right]\frac{1}{\epsilon^j}\Lambda^j &= 0 \\ \underbrace{\frac{(n_j\Sigma_j - \sum_{i=1}^{n} w_i^j S_i^j)}{(n_j - \sum_{i=1}^{n} w_i^j)}}_{C^j}\Lambda^j &= \Lambda^j \underbrace{\epsilon^j(\Delta+I)}_{D_{\gamma_i}}. \end{aligned}$$

This is an eigenvector equation for the $i = 1, \ldots, k$ columns of Λ^j, $C^j \Lambda_i^j = \gamma_i \Lambda_i^j$. Normally in an eigenvector problem, one is free to choose the scale of the eigenvector arbitrarily since if x is an eigenvector of A, then so is cx for any c. Here, however the norm of the eigenvectors are functions of the eigenvalues $\| \Lambda_i^j \|^2 = \frac{\gamma_i}{n_j - \sum_{i=1}^n w_i^j} - \epsilon^j$.

1.7 Summary of Algorithm

For each class $\mathcal{C}_j$ form

$$C^j = \frac{n_j S_j - \sum_{i=1}^n w_i^j S_i^j}{n_j - \sum_{i=1}^n w_i^j}.$$

Find the eigenvalues γ_i and corresponding eigenvectors Λ_i^j of

$$C^j \Lambda^j = \Lambda^j D_\gamma.$$

Scale $\| \Lambda_i \|^2$ such that

$$\gamma_i = \| \Lambda_i \|^2 + \epsilon.$$

Set

$$\epsilon = \frac{\mathrm{Tr} C^j - \sum_{i=1}^k \gamma_i}{p - k}.$$

Iterate over each class, updating the weights according to

$$\begin{aligned} w_i^j &= \frac{P(x_i \mid \mathcal{C}_j)}{\sum_{l=1}^K P(x_i \mid \mathcal{C}_l)} \\ &= \frac{\mathcal{N}(\mu^j, \Sigma^j)[x]}{\sum_{l=1}^K \mathcal{N}(\mu^l, \Sigma^l)[x]} \end{aligned}$$

where Σ^j is given by the current iteration's estimate $\Sigma^j = \Lambda^j \Lambda^{j'} + \epsilon I$, until convergence.

1.8 Simulation Results

1.9 Appendix - Matrix function derivatives

Chapter 2

HyperPancakes

2.1 Subspace models in classification

The general classification problem consists of categorizing an unknown observation $x \in \mathcal{R}^p$ into one of K classes $C_1, \ldots, C_K$. Each class is represented by a prototype, and we classify the observation to the nearest class according to some distance metric $d(x, C)$, (which can be a function of the class posterior $d(x, C_k) = 1 - p(y = C_k|x)$). Subspace methods of classification represent classes as subspaces. A good overview is given in [23]. Here, we restrict attention to methods for which:

1. Each class is represented by a linear subspace $V_k = [V_{k,1}, \ldots, V_{k,p_k}]$ of dimension $p_k < p$ that passes through the class mean: μ_k. The columns of V_k span the subspace corresponding to the k^{th} class.

2. Classification is performed by finding the nearest subspace by computing perpendicular distance to the subspace:

$$d(x, \mathcal{C}_k) = \|x - \mu_k\|^2 - \|V_k'(x - \mu_k)\|^2)$$

   (see figure 2.1).

More general subspace methods do exist, for example [15], where a special distance to the subspace called "tangent-distance" is used:

$$d(x, \mathcal{C}_k) = \min_{\gamma,\theta} ||\mu_k + T(\mu_k)\gamma - x - T(x)\theta||^2$$

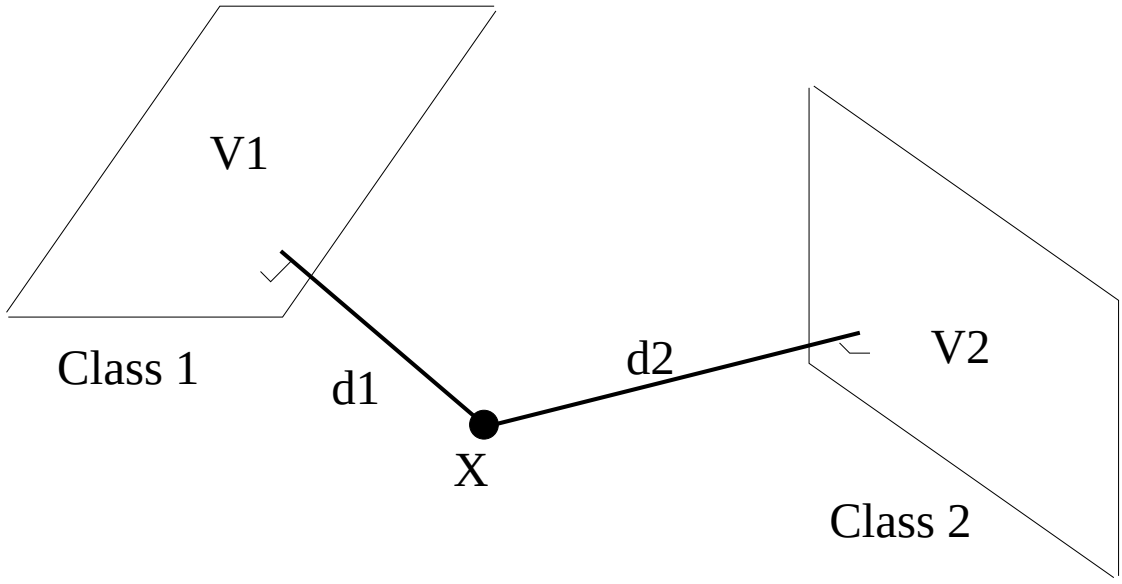

Figure 2.1: Illustration of subspace classification.

where γ, θ are parameters to be determined, μ_k is the centroid for class k, and $T(x)$ generates the tangent basis at the point x.

The challenge is to choose the subspaces $V_1, \ldots, V_K$ judiciously.

2.1.1 Why subspace models?

Subspace models are attractive for three reasons:

1. Parsimonious representation: In high dimensional settings, a low dimensional subspace can be a parsimonious way to represent a class, requiring a total of only $p(1 + \sum_k p_k)$ parameters.

2. Invariance: In certain cases, observations may really lie in a lower dimensional subspace, and this space may depend on the class. The subspace will span those directions in the space along which the data is invariant.

3. Rapid classification: Once the subspaces are specified classification involves computation of Kp subtractions to compute $x - \mu_k$, followed by $p(\sum_k p_k + 1)$ multiplications for the inner products of $x - \mu_k$ with itself and the V_k.

2.1.2 Details

More formally each class is represented by a centroid, μ_k and an affine subspace of dimension p_k passing through that centroid

$$C_k = \{\mu_k + V_k y : y \in \mathcal{R}^{p_k}, V_k^T V_k = I_{p_k}\}$$

The p_k orthonormal columns of V_k are the basis vectors which span the k^{th} subspace. The representation of the subspace is unique up to rotations of the basis vectors in the subspace.

Any point or vector x in the space $\mathcal{R}^p$ can be decomposed into two parts: the component lying in the k^{th} subspace, and the component orthogonal to it

$$x - \mu_k = V_k V_k^T (x - \mu_k) + V_k^{\perp} V_k^{\perp T} (x - \mu_k) \tag{2.1}$$

where $V_k^* = [V_k : V_k^{\perp}]$ is a complete, orthnormal basis for $\mathcal{R}^p$ and $V_k^{*T} V_k^* = I_p$.

The product $V_k^T x$ gives a p_k-vector whose components are the projections of x onto each column of V_k. Multiplying this p_k-vector by V_k gives a weighted sum of the columns of V_k corresponding to the projection of x along each column of V_k.

The perpendicular distance of a point to the k^{th} subspace is given by the Euclidean norm of the orthogonal component

$$\begin{aligned} d(x, C_k) &= \|V_k^{\perp} V_k^{\perp T} (x - \mu_k)\|^2 = \|V_k^{\perp T} (x - \mu_k)\|^2 && (2.2) \\ &= \|x - \mu_k\|^2 - \|V_k^T (x - \mu_k)\|^2 && (2.3) \end{aligned}$$

where we have used the fact that

$$\|V_k V_k^T x\|^2 = (V_k V_k^T x)^T (V_k V_k^T x) = x^T V_k (V_k^T V_k) V_k^T x = x^T V_k V_k^t x = \|V_k^T x\|^2.$$

Classification is performed by computing the distance of an observation to each class and classifying to the the nearest class:

$$\text{Class} = \arg\max_k d(x, C_k)$$

2.1.3 Relationship to Mahalanobis Distance

The Mahalanobis distance is a generalization of Euclidean distance whereby distances along different dimensions are given weights by a weighting matrix A which is positive definite and symmetric. The distance between two points x_0 and x_1 is

$$d(x_0, x_1) = (x_0 - x_1)^T A(x_0 - x_1) = \sum_{i,j}(x_0 - x_1)_i(x_0 - x_1)_j A_{i,j}$$

Euclidean distance is a special case when $A = I$. Perpendicular distance to subspace can be written in a a Mahalanobis distance form as follows

$$\begin{aligned} d(x, C_k) &= \|x - \mu_k\|^2 - \|V_k^T(x - \mu_k)\|^2 \\ &= (x - \mu_k)^T I(x - \mu_k) - (x - \mu_k)^T V_k V_k^T (x - \mu_k) \\ &= (x - \mu_k)^T (I - V_k V_k^T)(x - \mu_k) \end{aligned} \tag{2.4}$$

Later, we will want to embed the Mahalanobis distance in a multidimensional gaussian distribution where the weighting matrix will be the inverse of the covariance matrix Σ

$$\mathcal{N}(\mu_k, \Sigma_k)[x] = \frac{1}{\sqrt{(2\pi)^p \det \Sigma_k}} \exp -(\frac{1}{2}(x - \mu_k)'(\Sigma_k)^{-1}(x - \mu_k))$$

which requires that Σ_k^{-1} be the inverse of some matrix, ie that Σ_k be invertible. Unfortunately, $(I - V_k V_k^T)$ will, in general, not be of full rank and hence will not be invertible. Therefore we will consider the following parametrization of the covariance matrix

$$\Sigma_k = V_k D_\theta V_k^T + \epsilon I_p$$

where D_θ is a diagonal matrix with identical elements $\theta_1, \ldots, \theta_{p_k} = \theta$. The inverse of Σ_k can be derived using the Sherman-Morrison-Woodbury formula [13]

$$\Sigma_k^{-1} = \frac{1}{\epsilon}(I_p - \frac{\theta}{\theta + \epsilon} V_k V_k^T)$$

which, if ϵ goes to some fixed non-zero value, and $\theta \to \infty$, then

$$\Sigma_k^{-1} \to \frac{1}{\epsilon}(I_p - V_k V_k^T)$$

which is a multiple of the perpendicular distance. In practice, it will be enough for $\theta \gg \epsilon$.

2.1.4 Decision boundaries

What are the decision boundaries induced by hyperplane class protoypes? We show that they are quadratic surfaces. Since classification performance ultimately depends on the class decision boundaries, any technique which can generate quadratic boundaries will be a competitor to the hyperplane method.

The decision boundary between two classes is the set of points that is equidistant from both class prototypes

$$\begin{aligned} \text{DB} &= \{x : d(x, C_1) = d(x, C_2)\} \\ &= \{x : (x-\mu_1)^T(I - V_1V_1^T)(x-\mu_1) = (x-\mu_2)^T(I - V_2V_2^T)(x-\mu_2)\end{aligned}$$

Note that if the class prototypes were only points $\mu_1, \mu_2, \ldots$, with Euclidean distance then the decision boundary would be linear in x (this is also true if Mahalanobis distance is used with the same weight matrix for each class):

$$\begin{aligned} (x-\mu_1)^T(x-\mu_1) &= (x-\mu_2)^T(x-\mu_2) \\ x^Tx - 2\mu_1 x + \mu_1^T\mu_1 &= x^Tx - 2\mu_2 x + \mu_2^T\mu_2 \\ -2\mu_1 x + \mu_1^T\mu_1 &= -2\mu_2 x + \mu_2^T\mu_2 \end{aligned}$$

In our case, the presence of the V_k terms prevents us from getting rid of the quadratic component.

2.1.5 Number of model parameters

The hyperplane method requires p_k $p-$vectors plus a mean vector, for a total of $p_k + 1$ $p-$vectors to represent each class. It remains to be seen whether the subspace model is an efficient allocation of these model parameters, or if there exists other methods which dominate the subspace method. For instance, instead of using the $p_k + 1$ vectors to fix a hyperplane, we could find $p_k + 1$ points to represent each class and classify according to the nearest point. Indeed, one of the potentially undesirable artifacts of the subspace method is that the class prototypes extend to regions of the feature space where there may be no

observations from that class but plenty from other classes which will then be misclassified.

2.2 Finding the best subspaces

We have seen how to perform classification given the subspace class prototypes. The question remains how to choose the subspaces so as to optimize classification performance. We first review some of the classical uses of subspaces in dimensionality reduction and classification.

2.2.1 Principal Components

The principal components of a covariance matrix correspond to those directions along which the data varies the most. They are a solution to the problem: choose $p_k < p$ vectors for each class that best *represent* it. Formally, for the k^{th} class, with sample covariance matrix

$$S_k = \frac{1}{n_k} \sum_{x_i \in C_k} (x_i - \mu_k)(x_i - \mu_k)^T$$

find $V = [V_1, \ldots, V_{p(k)}]$ such that

$$\sum_{m=1}^{p_k} V_m^T S_k V_m$$

is minimized, subject to the orthonormality constraint: $V^T V = I_{p_k}$. The solution can be obtained analytically by the method of Lagrange multipliers by finding the stationary points of (note: you can include constraints for the cross terms $V_m V_n$ but the lagrange multipliers for the cross terms turn out to be zero [1]).

$$\sum_{m=1}^{p_k} V_m^T S_k V_m - \sum_{m=1}^{p_k} \lambda_m (V_m^T V_m - 1).$$

Differentiating with respect to V_m, and setting equal to 0 gives us the eigenvector equation

$$S_k V_m = \lambda_m V_m.$$

The original sum $\sum_{m=1}^{p_k} V_m^T S_j V_m = \sum_{m=1}^{p_k} \lambda_m$ is maximized by choosing those V_m corresponding to the largest eigenvalues. Thus the best k-dimensional subspace, in terms of *representing* each class, is given by the principal eigenvectors.

2.2.2 Fisher LDA

A classical technique in dimensionality reduction, Fisher linear discriminant analysis (LDA) finds a common subspace for all the observations, such that the separation between classes is maximized while the within class scatter is minimized for the data projected onto the subspace V.

Formally, let W be the pooled within class scatter, and let B be the between class scatter of the data.

$$W = \sum_k W_k = \sum_k \sum_{y_i=k} (x_i - \mu_k)(x_i - \mu_k)^T \tag{2.5}$$

$$B = \sum_k (\mu_k - \bar{\mu})(\mu_k - \bar{\mu})^T. \tag{2.6}$$

Then the Fisher discriminants are those V which maximize

$$J(V) = \frac{V^T B V}{V^T W V}$$

and are given by the principal eigenvectors of $W^{-1}B$.

2.3 Testing whether the data lie in a subspace

To test whether the data in a class lies in some subspace we want to examine the structure of the covariance matrix. Does the scatter of the data fill all dimensions of the space or are there some directions in which the data has no scatter. Does the high-dimensional cloud of data look like a football, or a pancake?

Let

$$\Sigma_k = \sum_{x \in C_k} (x - \mu_k)(x - \mu_k)^T$$

be the covariance matrix for data in the k^{th} class. By the spectral factorization theorem,

$$\Sigma_k = A D_\theta A^T$$

where the columns of A contain the eigenvectors of Σ_k and D_θ is a diagonal matrix whose elements are the eigenvalues of Σ_k. We can think of D_θ as being the covariance matrix of the original data in a rotated coordinate space, where the basis vectors are aligned with the

axes of the hyperellipsoidal cloud of data.

To check whether the data lie in a subspace, we can examine D_θ and see whether any of the elements are "zero",i.e. either small relative to some threshold or small relative to the larger elements of D_θ.

Large sample hypothesis tests in the case of gaussian data are given in Anderson section 11.7 [1].

Even if the data do not lie in a subspace, we can still use subspaces for classification. The interpretation is that we represent each class with separate means and covariances, but that the covariances are constrained to have a special form.

2.4 Geometrical approach

Intuitively, observations from a class should be close to their own class and far from all other classes. The geometric approach solves for one subspace at a time, seeking to minimize the average distance of observations to their own class, while maximizing the distance of non-members.

We seek subspaces $V^1, \ldots, V^J$ such that the perpendicular distance of observations in class j to V^j is minimized (on average), while simultaneously maximizing distance to V^m for $m \neq j$. Let $V^* = [V : V^\perp]$, a complete, orthnormal basis for $\mathcal{R}^p$, ie $V^{*T}V^* = I_p$. For any x, we can write

$$x = \sum_{a=1}^{p} V_a^* < V_a^*, x >= \sum_{a=1}^{k} V_a < V_a, x > + \sum_{a=1}^{p-k} V_a^\perp < V_a^\perp, x > .$$

The first term is the projection of x onto the subspace spanned by V, the second term is the component of x orthogonal to V. The norm of the second term gives us the perpendicular distance of x to V:

$$d(x, V) = \| \sum_{a=1}^{p-k} V_a^\perp < V_a^\perp, x > \|^2 = \sum_{a=1}^{p-k} < V_a^\perp, x >^2$$

The expected value (with respect to the empirical distribution) of the perpendicular

distance of an observation from class m centered about the j^{th} mean to the subspace V is

$$Ed(x-\mu_j, V) = \sum_{a=1}^{p-k} E < V_a^{\perp}, x-\mu_j >^2 = \sum_{a=1}^{p-k} V_a^{\perp T} E[(x-\mu_j)(x-\mu_j)^T] V_a^{\perp} = \sum_{a=1}^{p-k} V_a^{\perp T} R_m^{(j)} V_a^{\perp}.$$

For the j^{th} class we seek a V such that

$$Ed(x-\mu_j, V)\mathbf{1}(y_i = j) \tag{2.7}$$

is minimized, and

$$\sum_{m \neq j} Ed(x-\mu_j, V)\mathbf{1}(y_i = m) \tag{2.8}$$

is maximized. Now $d(x, V)$ is quadratic in V, and also convex. A combined criterion such as

$$Ed(x-\mu_j, V)\mathbf{1}(y_i = j) - \sum_{m \neq j} Ed(x-\mu_j, V)\mathbf{1}(y_i = m)$$

being the difference of two convex functions, is not convex.

By bounding $Ed(x-\mu_m, V)\mathbf{1}(C_i = m)$, we can arrive at a criterion function that is convex as follows. First, for each j we scale $R_1^{(j)}, \ldots, R_J^{(j)}$ so that the maximum eigenvalue is one: $\lambda(R_m^{(j)}) \leq 1$ for $m = 1, \ldots, J$. Note that the $R_m^{(j)}$ are positive definite by construction so $0 \leq \lambda(R_m^{(j)}) \leq 1$. We can therefore bound the average perpendicular distance:

$$Ed(x-\mu_m, V)\mathbf{1}(C_i = m) = \sum_{a=1}^{p-k} V_a^{\perp T} R_m^{(j)} V_a^{\perp} \leq \sum_{a=1}^{p-k} 1 = p-k.$$

Maximizing $Ed(x-\mu_j, V)\mathbf{1}(C_i = m)$ is the same is minimizing $p-k-Ed(x-\mu_j, V)\mathbf{1}(C_i = m)$. Now we can transform our original problem into finding that V which minimizes

$$\begin{aligned} Ed(x-\mu_j, V)\mathbf{1}(C_i = j) + \sum_{m \neq j}(p-k) - Ed(x-\mu_j, V)\mathbf{1}(C_i = m) &= \\ \sum_{a=1}^{p-k} V_a^{\perp T} R_j^{(j)} V_a^{\perp} + \sum_{m \neq j}(p-k) - \sum_{a=1}^{p-k} V_a^{\perp T} R_m^{(j)} V_a^{\perp} &= \\ \sum_{a=1}^{p-k} V_a^{\perp T} R_j^{(j)} V_a^{\perp} + \sum_{m \neq j} \sum_{a=1}^{p-k} V_a^{\perp T} V_a^{\perp} - V_a^{\perp T} R_m^{(j)} V_a^{\perp} &= \end{aligned}$$

$$\sum_{a=1}^{p-k} V_a^{\perp T}[R_j^{(j)} + \sum_{m \neq j}(I - R_m^{(j)})]V_a^{\perp}$$

The last equation is again a quadratic form, for which the $V_a^{\perp}$ that minimize it are given by the eigenvectors of $R_j^{(j)} + \sum_{m \neq j}(I - R_m^{(j)})$ corresponding to the smallest eigenvalues. The best classification subspace for class j is then given by the orthogonal space to $V^{\perp}$, namely V, given by the k principal components of $R_j^{(j)} + \sum_{m \neq j}(I - R_m^{(j)})$.

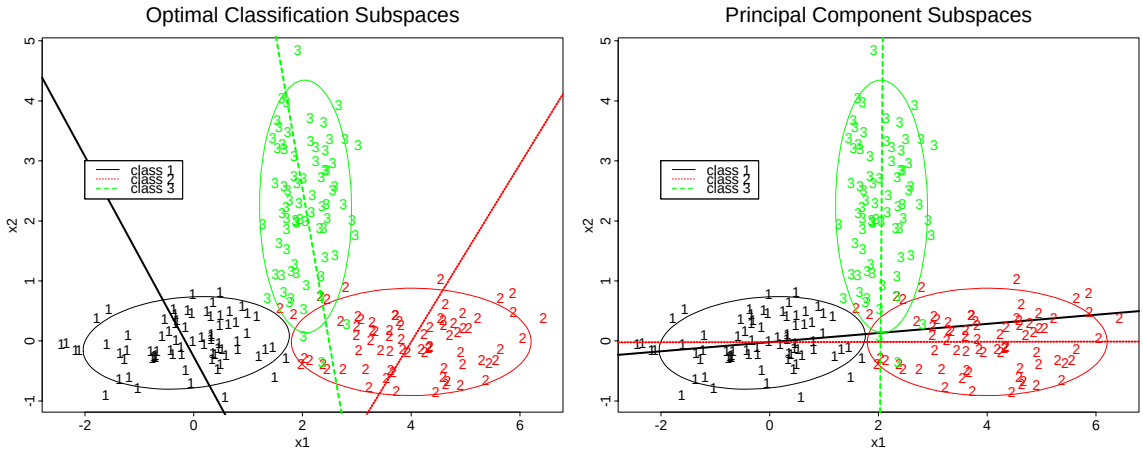

Figure 2.2: Principal component subspaces and optimal classification subspaces.

Figure 2.2 shows subspaces for a 3 class, 2-D example. The classes are bivariate gaussian. The means of the classes are at (0,0), (4,0), (2,0). The covariances for classes 1 and 2 are $\begin{pmatrix} 1 & 0 \\ 0 & 0.2 \end{pmatrix}$, and for class 3 $\begin{pmatrix} 0.2 & 0 \\ 0 & 1 \end{pmatrix}$. The principal component subspaces result in many confusions between observations from class 1 and 2.

2.5 Maximum Likelihood Formulation: Hyperpancakes

By embedding each subspace in a distribution

2.5.1 Score equations

The score equations are given by setting the derivative of the likelihood equal to zero. This leads to a system of coupled eigenvector equations.

2.5.2 Direct Optimization via Gradient Descent

The number of variables is huge and there is a whole family of solutions.

2.6 Mixed Likelihood: Discrimination and Representation

2.7 Learning Subspace Method

Kohonen's Learning Subspace Method [23][p.111-153] is an iterative technique which, at each step, seeks to rotate subspaces closer to the data they are supposed to represent, and farther away from training observations which are misclassified.

The projection matrix onto the subspace orthogonal to x is given by

$$A = I - \frac{xx^T}{x^T x}.$$

When A is applied to a subspace V, it rotates the subspace to a new one V' which is orthogonal to x. For each basis vector in V, the projection retains only that component that is perpendicular to x.

$$Ax = (I - \frac{xx^T}{x^T x})x = x - x = 0$$
$$Av = (I - \frac{xx^T}{x^T x})v = v - x\frac{x^T v}{x^T x}$$

Kohonen's online method [23][p.123] involves starting with a set of subspaces, one for each class $\{V_1, \ldots, V_K\}$. For each training observation, both the correct class' subspace and the closest one are modified

$$V' = (I + \alpha xx^T)V$$

where α is chosen such that the correct class' subspace moves closer to its data, and the incorrect, closest one, is rotated away. The method borrows ideas from the Learning Vector Quantization algorithm, where the class prototypes are points. The LSM is sensitive to the order in which observations are presented, and is described only for non-affine subspaces (ie all the subspaces pass through the origin, $\mu_k = 0$ for $k = 1, \ldots, K$.)

Oja [23][p.146] also presents a modified version of the LSM, called the Averaged Learning Subspace Method (ALSM). It considers the random event that an observation belonging to

class k is miclassified to class j, and averages the LSM update rule (ie the matrix transform on each subspace) conditioned on the following events:

1. An observation from C_k is correctly classified.
2. An observation from C_k is misclassified to C_j.
3. An observation from C_j is misclassified to C_k.

We generalize the algorithm for the case of affine subspaces, where each subspace may pass through a different offset μ_k.

Initialization

$$S_k^0 = \sum_{x \in C_k} (x - \mu_k)(x - mu_k)^T$$

Choose the initial subspaces V_k to be the p_k principal eigenvectors of S_0^k.

Classification Classify the training data using the current subspaces of iteration m, and compute scatter matrices for those observations that were misclassifed

$$S_{k,j}^m = \sum_{\substack{x \in C_k \\ x \mathrm{m} isclassifiedto C_j}} (x - \mu_k)(x - mu_k)^T$$

Update Scatter Matrices Compute

$$S_k^m = S_k^{m-1} + \sum_{k \neq j} \alpha_{k,j} S_{k,j}^m - \sum_{k \neq j} \beta_{j,k} S_{j,k}^m$$

where $\alpha_{k,j}$ and $\beta_{j,k}$ are found not to matter too much and so are taken to be constant.

Update Subspaces The eigenvectors corresponding to the largest eigenvalues of S_k^m are chosen as the basis for subspace V_k^m.

Iteration Increment m and go to Classification step. Stop when steady-state has been reached (ie subspaces don't change or in the case of oscillations, stop at a large fixed m).

2.7.1 Mixtures of Probabilistic Principal Component Analysers

In [31], Tipping and Bishop model the data x as having arisen from a lower dimensional space of latent variables t:

$$x = Wt + \mu + \epsilon$$

where $t \sim \mathcal{N}(0, I)$, and $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. The distribution of x is then also gaussian $x \sim \mathcal{N}(0, \sigma^2 I + WW^T)$.

Tipping and Bishop extend this model (for which I independently arrived at the same solution above) to a mixture of subspaces

$$x \sim \sum_m^M \pi_m \mathcal{N}(0, \sigma^2 I + W_m W_m^T)$$

and present an EM algorithm to find the subspaces W_m based on optimizing the full likelihood. Their formulation is essentially the same as hyperpancakes, except that they maximize the full likelihood $L(x, y)$, whereas hyperpancakes seeks to maximize the conditional likelihood, i.e. the multinomial likelihood $L(y, x)$.

2.8 Discriminative Training

We have observations x_i and class membership variables y_i for $i = 1, \ldots, N$. In the case of training data the y_i are available, but they are hidden when it comes to the classification problem. The data arise from an underlying distribution $p(x, c)$ which we do not know. The Bayes' decision rule minimizes the number of misclassifications and tells us to classify x to that class for which $p(c|x)$ is the largest.

The classical approach is to assume some functional form for the class conditional distributions $p(x|c; \theta)$ (eg multivariate gaussian), and estimate the parameters of the distribution (eg $\theta = \{\mu, \Sigma\}$, the mean and covariance) based on the training data. The resulting performance depends both on how well the parameters are estimated and more importantly, on whether the underlying distributional assumptions are correct.

Discriminative training involves modifying the parametrization of a class based on data from other classes. [DAN: is that it? seems pretty broad. if parameters are tied across classes is that discriminative training? no. (eg common covariance). somehow it's important that data from other classes 'push away'. Is there a link between discriminant functions (eg neural nets) and discriminative training?]

When we speak of discriminative training, we mean a training procedure that seeks to optimize the conditional likelihood: $p(c_1, \ldots, c_n|x_1, \ldots, x_n)$ as opposed to the full likelihood $p(c_1, \ldots, c_n, x_1, \ldots, x_n)$.

2.8.1 Bayes Vector Quantization

Vector quantization [9] is a compression technique whereby a set of prototypes is chosen to represent (lossily) a set of data. Instead of storing the original data, the prototypes and their indices are stored. Various methods exist for coming up with a set of prototypes which largely depend on the choice of distortion measure used (eg minimum squared error (MSE), relative entropy). In [25] a technique called Bayes VQ is investigated. It seeks not only to compress the data, but also to classify it well. A modified distortion measure which adds a term for the Bayes risk is introduced

$$J(\alpha, \beta, \delta) = D(\alpha, \beta) + \lambda B(\alpha, \delta)$$

where α denotes the encoder $\alpha : X \rightarrow \hat{X}_i$, β denotes the decoder $\beta : i \rightarrow \hat{X}_i$, δ is the classifier $\delta(i)$ gives the class label. .

Optimization of this criterion provides prototypes which both compress the data as much as possible, and simultaneously separate the classes.

2.8.2 Representation vs Discrimination

In [5], Efron shows that for the two class case, logistic regression (ie discriminative training) is less efficient than the normal discrimination procedure (ie Fisher's linear discriminant function) but is more robust - it can accomodate any exponential family distribution, not just the gaussian.

The Bayes error is achieved if the classifier is able to correctly estimate the a posteriori class probabilities $p(c|x)$. For several error criteria including squared error and relative entropy, Ney [22] shows that the minimum classification error is achieved if the generalized discriminant functions (eg the outputs of a neural net) are equal to the a posteriori class probabilities. This is interesting because the neural net is not trying to explicitly estimate the class densities, yet the ideal mapping between inputs and outputs is given by $g(x, c) = p(c|x)$.

[DAN: do princomp for digit data]

We are given training data x_i for $i = 1, \ldots, n$ and class membership indicators $z_{ik} = 1$ if x_i is in $\mathcal{C}_k$ and 0 otherwise.

Hastie et al achieve surprisingly good classification results on a handwritten digit recognition problem using the largest-k principal components of the class covariance matrix as the subspace (this is surprising because the principal components are those directions that best *represent* the class in a lower dimension, and not necessarily the ones that best discriminate between classes. See figure). There are other examples where people use the principal components for classification purposes.

We wish to choose the subspaces in a way that best discriminates among the classes. To accomplish this goal, we embed the subspace in a "flat" multivariate Gaussian distribution, or "pancake" and model the class conditional distributions as:

$$P(x \mid x \in \mathcal{C}_k) = \mathcal{N}(\mu^k, \Sigma^k)[x] = \frac{1}{\sqrt{(2\pi)^p \det \Sigma^k}} \exp -(\frac{1}{2}(x-\mu^k)'(\Sigma^k)^{-1}(x-\mu^k))$$

where

$$\Sigma^k{}_{pxp} = (d)V^k{}_{pxk}V^{k'}{}_{kxp} + \epsilon I_p$$

The parameters ϵ and d can be thought of respectively as the thickness of the pancake and its extent and the k columns of V span the subspace of interest. The columns of V_k are orthonormal: $V_k'V_k = I_k$. Note:

$$\Sigma_k^{-1} = \frac{1}{\epsilon}(I_p - \frac{d}{d+\epsilon}V_kV_k').$$

Using Bayes rule we can obtain the class membership probability, given an observation:

$$P(x \in \mathcal{C}_k \mid x) = \frac{P(x \mid x \in \mathcal{C}_k)P(\mathcal{C}_k)}{\sum_{l=1}^K P(x \mid x \in \mathcal{C}_l)P(\mathcal{C}_l)} = \frac{e^{D_k(x)}}{\sum_{l=1}^K e^{D_l(x)}}$$

where we have used the fact that the determinant term depends only on ϵ and d which is common to all classes, and have assumed equal priors on the classes, and defined

$$\begin{aligned} D_k(x_i) &\stackrel{\triangle}{=} -\frac{1}{2}\underbrace{(x_i-\mu_k)'}_{\stackrel{\triangle}{=} y_{ik}}\Sigma_k^{-1}(x_i-\mu_k)) \\ &= -\frac{1}{2}y_{ik}'\frac{1}{\epsilon}(I_p - \frac{d}{d+\epsilon}V_kV_k')y_{ik} \\ &= -\frac{1}{2}\left[\frac{1}{\epsilon}\|y_{ik}\|^2 - \frac{d}{\epsilon(d+\epsilon)}\|V_k'y_{ik}\|^2\right]. \end{aligned}$$

Note that for $\epsilon = 1$ and $d \gg \epsilon$, $D_k(x)$ is equivalent to perpendicular distance to the subspace.

Now that we have embedded our subspaces in these Gaussian pancakes, we propose the multinomial likelihood as a natural way of inducing discrimination between the classes. That is, we consider the class membership probabilities (conditioned on the data), and seek to maximize their log likelihood with respect to the V_k's.

The likelihood of the data assuming multinomial class probabilities given by $P(x \in \mathcal{C}_k \mid x)$ above is:

$$L(V_1, \ldots, V_K) = \prod_{i=1}^{n} \left[\frac{\prod_{k=1}^{K} e^{D_k(x_i) z_{ik}}}{\sum_{l=1}^{K} e^{D_l(x_i)}} \right].$$

Recall that $z_{ik} = 1$ only if x_i is in $\mathcal{C}_k$, so that only one term in the product in the numerator survives. Maximizing the likelihood is equivalent to maximizing the log likelihood:

$$\begin{aligned} l(V_1, \ldots, V_K) &= \sum_{i=1}^{n} \left\{ \log \prod_{k=1}^{K} e^{D_k(x_i) z_{ik}} - \log \sum_{l=1}^{K} e^{D_l(x_i)} \right\} \\ &= \sum_{i=1}^{n} \left\{ \left(\sum_{k=1}^{K} D_k(x_i) z_{ik} \right) - \log \sum_{l=1}^{K} e^{D_l(x_i)} \right\} \\ &= \sum_{i=1}^{n} \left\{ \left(\sum_{k=1}^{K} -\frac{1}{2} z_{ik} \left(\frac{1}{\epsilon} \|y_{ik}\|^2 - \frac{d}{\epsilon(d+\epsilon)} \|V_k' y_{ik}\|^2 \right) \right) - \log \sum_{l=1}^{K} e^{D_l(x_i)} \right\} \end{aligned}$$

Differentiating $l(V_1, \ldots, V_K)$ with respect to V_m gives:

$$\begin{aligned} \frac{\partial l}{\partial V_m} &= \frac{d}{\epsilon(d+\epsilon)} \left[\sum_{i=1}^{n} \left\{ z_{im} y_{im} y_{im}' V_m - w_i^{(m)} y_{im} y_{im}' V_m \right\} \right] \\ &= \frac{d}{\epsilon(d+\epsilon)} \left[n_m S_m - \sum_{i=1}^{n} w_i^{(m)} S_i^{(m)} \right] V_m \end{aligned}$$

where

$$\begin{aligned} w_i^{(m)} &\triangleq \frac{e^{D_m(x_i)}}{\sum_{l=1}^{K} e^{D_l(x_i)}} \\ S_m &\triangleq \frac{1}{n_m} \sum_{i=1}^{n} z_{im} y_{im} y_{im}' \\ S_i^{(m)} &\triangleq y_{im} y_{im}' \end{aligned}$$

Including the constraint that $V_k' V_k = I_k$ and forming the lagrangian involves adding a

term to the derivative equation above of the form: $D_\lambda^k V_k$ where D_λ is diagonal.

Setting $\frac{\partial l}{\partial V_m} = 0$ (including constraint) gives the following stationary condition

$$\underbrace{(n_m S_m - \sum_{i=1}^{n} w_i^{(m)} S_i^{(m)})}_{\triangleq C_m} V_m = D_\lambda^m V_m.$$

This eigenvector equation suggests an iterative scheme for finding V's: given the current estimate of the V's, we can compute the $w_i^{(m)}$, and evaluate the matrix C_m. Finding the principal eigenvectors of C_m provides the next iterations estimate of V_m. Unfortunately this procedure hasn't been obeserved to converge in general.

I've also implemented a conjugate gradient descent where I always normalize the current V's to be orthnormal but it does strange things.

I've implemented a direct minimization using a sequential quadratic programming algorithm, but it can't deal with too many variables, and the whole point of this research is to work in high-dimensions.

Starting from scratch, I have come up with the following idea:

Consider each of the p unit vectors $e_1, \ldots, e_p$ in turn, and examine how well the classes are distinguished in each unit vector direction. More formally, define

$$\begin{aligned} d_m(x_i, \mathcal{C}_k) &\triangleq \|x_i - \mu_k\|^2 - \|e_m'(x_i - \mu_k)\|^2 \\ &= \|x_i - \mu_k\|^2 - \|(x_i - \mu_k)_m\|^2 \\ \alpha_{ikm} &\triangleq d_m(x_i, \mathcal{C}_k) / \sum_{l=1}^{K} d_m(x_i, \mathcal{C}_l) \end{aligned}$$

where $d_m(x_i, \mathcal{C}_k)$ can be thought of as the distance of x_i to the k^{th} subspace, if the unit vector e_m were the subpsace. If e_m is a "good" direction for discrimination, then $\sum_{i=1}^n \alpha_{ikm} z_{ik}$ should be large. How large? Well, if e_m were a terrible direction for discriminating between the classes, then αikm would be approximately $1/K$ for each class (ie for all k). Then we would expect $\sum_{i=1}^n \alpha_{ikm} z_{ik}$ to be roughly equal to the number of observations from $\mathcal{C}_k$ times $1/K$. Deviations away from this indicate good discriminatory potential. So let $\beta_m^k = \sum_{i=1}^n (\alpha_{ikm} - \frac{1}{K}) z_{ik}$ be our measure of how good direction e_m is for discriminating class k.

To find the best subspace, we build it up one vector at a time for each class as follows: Fix k, (class index). Compute β_m^k for $m = 1, \ldots, p$ (discriminatory goodness measure). Let $V_1^k = (\beta_1^k, \ldots, \beta_2^k)/\|(\beta_1^k, \ldots, \beta_2^k)\|$. Repeat this process, using instead of e_m, unit vectors that are orthogonal to this V_1^k.

Experimental results: tried this with a toy example of two gaussians. Got expected results.

2.9 Choice of weights and centroids

The question remains which probability measure to use when computing

$$R_m^{(j)} = E(x_i - \mu_j)(x_i - \mu_j)^T \mathbf{1}(C_i = m).$$

There are several possibilities:

1. Use the empirical distribution which puts equal weight on each observation:

$$R_m^{(j)} = \frac{1}{n_m} \sum_{C_i = m} (x_i - \mu_j)(x_i - \mu_j)^T.$$

   Problem: outliers exert undue influence.

2. Assume Gaussian classes:

$$R_m^{(j)} = \sum_{C_i = m} \mathcal{N}(\mu_m, \Sigma_m)[x_i](x_i - \mu_j)(x_i - \mu_j)^T.$$

   Problem: makes strong assumption about distribution of data.

3. Nonparametric approach: We compute $R_m^{(j)}$ for $m = 1, \ldots, J$. Each $R_m^{(j)}$ has the form

$$R_m^{(j)} = \sum_{C_i = m} w_i^m (x_i - \mu_j)(x_i - \mu_j)^T.$$

   where w_i^m is a weight for the i^{th} observation coming from the m^{th} class. We constrain

$$\sum_{m=1}^{J} w_i^m = 1$$

for each i. The optimal weights are obtained via the method of lagrange multipliers.

Recall that the optimization criterion for the j^{th} class is

$$f_j = \sum_{a=1}^{p-k} V_a^{\perp T}[R_j^{(j)} + \sum_{m \neq j}(I - R_m^{(j)})]V_a^{\perp}$$

which we seek to minimize subject to the constraints $\sum_{m=1}^{J} w_i^m = 1$ for each i. We introduce Lagrange multipliers $\lambda_1, \ldots, \lambda_n$, and consider the Lagrangian:

$$\begin{aligned} L_j &= f_j - \lambda_1(\sum_{m=1}^{J} w_1^m - 1) - \cdots - \lambda_n(\sum_{m=1}^{J} w_n^m - 1) \\ &= \sum_{a=1}^{p-k} V_a^{\perp T}[\sum_{C_i=j} w_i^j (x_i - \mu_j)(x_i - \mu_j)^T + \sum_{m \neq j}(I - \sum_{C_i=m} w_i^m (x_i - \mu_j)(x_i - \mu_j)^T)]V_a^{\perp} - \sum_{i=1}^{n} \lambda_i(\sum_{m=1}^{J} w_i^m(\end{aligned}$$

Using

$$\frac{\partial R_m^{(j)}}{\partial w_i^m} = (x_i - \mu_j)(x_i - \mu_j)^T$$

we obtain

$$\frac{\partial f_j}{\partial w_i^m} = \begin{cases} \sum_a V_a^{\perp T}[(x_i - \mu_j)(x_i - \mu_j)^T]V_a^{\perp} - \lambda_i w_i^m & m = j \\ \sum_a V_a^{\perp T}[-(x_i - \mu_j)(x_i - \mu_j)^T]V_a^{\perp} - \lambda_i w_i^m & m \neq j \end{cases}$$

Setting $\frac{\partial f_j}{\partial w_i^m} = 0$ and summing over m we solve for the Lagrange multiplier

$$\sum_{m=1}^{J} \lambda_i w_i^m = \lambda_i = \sum_a V_a^{\perp T}[(2 - J)(x_i - \mu_j)(x_i - \mu_j)^T]V_a^{\perp}$$

and the weights

$$w_i^m = \begin{cases} \frac{-1}{J-2} & m = j \\ \frac{1}{J-2} & m \neq j \end{cases}$$

This violates non-negativity of the w_i^m. So include a non-negativity constraint. Another idea: minimize $\sum_j f_j + \sum_i \lambda_i(\sum_m s_i^m - 1)$ Problem: this involves all the subspaces of the different classes at once.

Figure (2.3) demonstrates the sensitivity of the method to choice of weights. The $R_m^{(j)} = \sum_{i=1}^{n} w_i^m (x_i - \mu_j)(x_i - \mu_j)^T$, which is a slight modification from before.

Figure 2.3: Clockwise from top-left: (a) Using gaussian class weights. (b) Using empirical distribution (1/0) weights. (c) Using empirical dist weights and ignoring outliers (based on nearest 0.70 of the data in each class) (d) Based on Lagrangian multiplier weights

2.10 Digit data

We compare the performance of Fisher LDA, principal components and the optimal classifier on the digit data: handwritten 3,5,8 digits on a 8x8 grid, 150 observations in each class.

Figure 2.4: The digit data projected onto the first 2 Fisher discriminants, with the principal component subspaces, and the resulting misclassifications. Also shown is the misclassification using the optimal subspaces.

Figure (2.4) gives an idea of what the data looks like when it is projected from 64 dimensions down to 2. In the rest of the analysis we consider 5-dimensional subspaces of 64-space.

As a visualization tool we introduce the notion of a spectrum with respect to a basis. Let $U(p,p)$ be an orthonormal basis. The spectrum of $V(p,k)$, where $k < p$, with respect to U is the vector $\mathcal{E} = \sum_{i=1}^{k} V_i^T U$.

Figure 2.5: Energy Spectra for the digit data. Each graph shows the spectrum of the subspace using the class covariance matrix eigenvectors as a basis. Each colum corresponds to a class. The top row is for the principal component subspaces, the bottom row is for the optimal classification subspaces.

Figure (2.7) shows the misclassification rate as a function of subspace dimension for principal components and OCS.

Figure 2.6: The distribution of distances to the class subspaces. The left plot is for the 5-principal component subspace, the right plot is for the optimal subspace. The distances are divided into 2 groups: the distances to the correct class, and the distances to the incorrect classes.

Figure 2.7: Number of misclassifications as a function of subspace dimension, comparing principal components and OCS

Chapter 3

DISCRIMINATION

3.1 Discriminative Training - Introduction

The pattern recognition problem consists of classifying an observation x into one of K classes. Typically a training set of data (x_i, y_i) $i = 1, \ldots, n$, where the x_i are observations and the $y_i \in \{1, \ldots, K\}$ are class labels, is available from which to build models for future classification. The classifier can be thought of as a function $\gamma : x \to \{1, \ldots, K\}$.

One way of looking at the problem is to consider that the data arise from an unknown, underlying joint distribution $p(x, y)$. The Bayes' decision rule [4] minimizes the number of misclassifications and tells us to classify x to that class for which $p(y|x)$ is the largest

$$\gamma(x) = \arg\max_k p(y = k|x).$$

We briefly review the derivation of the Bayes' classifier.

3.1.1 Bayes Classifier

The risk (expected loss) of classifying to class j given x is

$$R(\gamma(x) = j|x) = \sum_{i=1}^{K} \Lambda(j, i) p(y = i|x)$$

where $\Lambda(j,i)$ is the loss associated with misclassifying an observation from class i into class j. The overall risk is given by

$$R = \int R(\gamma(x)|x)p(x)dx$$

which is minimized if for each x, $R(\gamma(x)|x)$ is made as small as possible. That is, the risk is minimized if $\gamma(x)$ is set to that j for which $R(\gamma(x) = j|x)$ is smallest.

Under 0/1 loss $\Lambda(j,i) = \delta(j,i)$ and the conditional risk

$$R(\gamma(x) = j|x) = \sum_{i \neq j} p(y = i|x) = 1 - p(y = j|x),$$

is minimized if $p(y = j|x)$ is as large as possible. Counting misclassifications is equivalent to a 0/1 loss function (either an observation is misclassified, or it is not), and this gives us the Bayes classification rule: classify to that class for which $p(y = j|x)$ is maximum.

The main problem is that in practice we do not know $p(y|x)$ and must estimate it from the training data.

3.1.2 Discriminative Training

Now,

$$p(y = k|x) = \frac{p(x|y = k)p(y = k)}{p(x)}$$

and if we compare $p(y = k_1|x)$ to $p(y = k_2|x)$ we see that it is enough to consider the ratio

$$\frac{p(y = k_1|x)}{p(y = k_2|x)} = \frac{p(x|y = k_1)p(y = k_1)}{p(x|y = k_2)p(y = k_2)} \tag{3.1}$$

so that the classification decision doesn't depend on the actual form of the conditional densities, but rather, on their *ratio*.

Definition 1 (Discriminative Training) *A training procedure which does not seek to estimate the class conditional densities $p(x|y)$, but rather, concerns itself only with the class posterior probabilities $p(y|x)$.*

The meaning will become clearer with the examples presented below, where the comparison between discriminative versus non-discriminative training is that the latter estimates

the class probabilities and then uses the estimates in computing the ratio, while the former estimates the ratio directly. Neural nets are an example of a system that uses discriminative training: a neural net targets its outputs to classify an observation.

We also investigate combining discriminative and non-discriminative training. This is an idea that has been developed independently in several areas, including Bayesian risk vector quantization, which we describe below.

ASIDE: [DAN: find a good place for this] The Bayes error is achieved if the classifier is able to correctly estimate the a posteriori class probabilities $p(y|x)$. For several error criteria including squared error and relative entropy, Ney [22] shows that the minimum classification error is achieved if the generalized discriminant functions (eg the outputs of a neural net) are equal to the a posteriori class probabilities. This is interesting because the neural net is not trying to explicitly estimate the class densities, yet the ideal mapping between inputs and outputs is given by $g(x, y) = p(y|x)$.

3.1.3 LDA vs Logistic Regression

Consider two gaussian classes with common covariance and class priors π_k

$$p(x|y=k) = \mathcal{N}(\mu^k, \Sigma)[x] = \frac{1}{\sqrt{(2\pi)^p \det \Sigma}} \exp -(\frac{1}{2}(x-\mu^k)'\Sigma^{-1}(x-\mu^k))$$

with probability π_k. For a particular observation, the log of the likelihood ratio gives Fisher's LDA

$$\begin{aligned} \lambda(x) &= \log \frac{p(x|y=0)}{p(x|y=1)} \\ &= \log \frac{\pi_0}{\pi_1} - \frac{1}{2}(\mu_0-\mu_1)^T\Sigma^{-1}(\mu_0-\mu_1) + (\mu_0-\mu_1)^T\Sigma^{-1}x, \end{aligned}$$

which classifies to class 0 or 1 according to whether $\lambda(x)$ is positive or negative.

Efron compared the efficiency of logistic regression to discriminant analysis [5] and found that logistic regression is more robust when the classes are not gaussian, but that LDA is more efficient when they are. O'Neill [24] generalized for the case when the underlying densities were not Gaussian and derived an expression for the asymptotic error rate for an arbitrary estimator of the decision boundary.

3.1.4 Vector Quantization vs Bayes VQ

The goal of vector quantization is to obtain Vector quantization [9] is a compression technique whereby a set of prototypes is chosen to represent (lossily) a set of data. Instead of storing the original data, the prototypes and their indices are stored. Various methods exist for coming up with a set of prototypes which largely depend on the choice of distortion measure used (eg minimum squared error (MSE), relative entropy). In [25] a technique called Bayes VQ is investigated. It seeks not only to compress the data, but also to classify it well. A modified distortion measure which adds a term for the Bayes risk is introduced

$$J(\alpha, \beta, \delta) = D(\alpha, \beta) + \lambda B(\alpha, \delta)$$

where α denotes the encoder $\alpha : X \rightarrow \hat{X}_i$, β denotes the decoder $\beta : i \rightarrow \hat{X}_i$, δ is the classifier $\delta(i)$ gives the class label. The first term is the distortion and the second term is the Bayes Risk. The prototype selected to represent a point depends not only on its Euclidean distance to the nearest prototype, but also on its Bayes risk component to that prototype. The resultant Voronoi regions are not necessarily convex [cite: personal communication with Richard Olshen].

Here we have a mixture of two objectives: one to represent the data (minimizing the squared error distortion), and one to separate the data (using the bayes risk).

3.1.5 Naive Bayes vs GAM

Naive bayes assumes a product form for each class density (ie a log additive model) assuming independence among the covariates. It thus suffers from bias when the covariates are actually not independent, but it's strength lies in its simplicity and potential low variance in its estimates. For classification applications, this low variance behavior may actually improve performance despite the bias [7].

The log of the density in the k^{th} class is given by

$$\log f^k(x) = \sum_a^p \log f_a^k(x_a)$$

where $x = (x_1, \ldots, x_a, \ldots, x_p) \in \mathcal{R}^p$ and $f_a^k(x_a)$ is the univariate density of the a^{th} dimension of x. In our studies we further parametrize the class densities to be an expansion of M

fixed basis functions $b_m(x)$ for $m = 1, \ldots, M$ with weights β_m^k. Note that the weights are class and dimension dependent but that the basis functions are common to all classes and dimensions.

$$\log f_a^k(x_a) = \sum_m^M \beta_m^{k,a} b_k(x_a)$$

The training procedure involves estimating only the $\beta_m^{k,a}$.

Generalized additive models assume that the ratio of class posterior densities is log additive:

$$\log \frac{P(y=k|x)}{P(y=K|x)} = \log g^k(x) = \sum_a^p \log g_a^k(x_a).$$

For comparison purposes, as above, we will assume a basis expansion for the ratio of the densities

$$\log g_a^k(x_a) = \sum_m^M \alpha_m^{k,a} b_k(x_a).$$

Note that this allows for a wide choice of class densities, so long as the ratio of any two is log additive.

For comparing the two methods we will fix the basis functions $b_m(x)$ to be the same, we will fix the number of basis functions M, and we will choose them to be splines with fixed knots. [DAN: use logspline for naive bayes, and ripley for gam]

The Naive Bayes is a non-discriminative approach - the parameters of the class densities are estimated separately for each class assuming a very special form for the class densities. GAM is discriminative - it makes very weak assumptions about the class densities - only that their ratio is log additive.

3.2 Combining Representation with Discrimination

There are cases when learning the class densities leads to good classification, and there are instances when it is better to focus on discrimination. It seems that when the model is correct, maximum likelihood estimates based on the full likelihood $f(x_1, y_1, \ldots, x_n, y_n)$ (eg discriminant analysis, which is misleadingly named) outperform discriminative methods which are based on the conditional likelihood $f(y_1, \ldots, y_n|x_1, \ldots, x_n)$ [24].

[DAN: fix] Intuitively this is because discriminative methods tend to focus on the decision boundary region [DAN: give pictorial example].

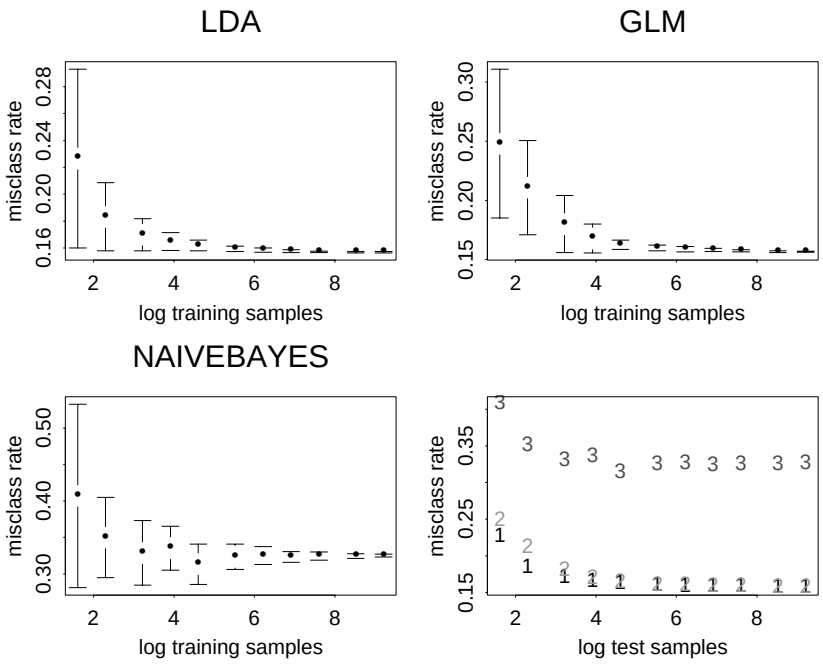

Figure 3.1: Classification Error comparing three techniques: LDA, GLM (logistic regression), naive bayes (with univariate gaussians along each dimension). Two gaussian classes in $\mathcal{R}^2$ were generated with $mu_0 = (0,0)$, $mu_1 = (2,0)$, and $\Sigma_{i,j} = 0.7^{|i-j|}$. A test set of 20000 observations was classified according to models obtained using different training set sizes. For each training set size, 25 models were trained, and the error bars indicate the variation of the misclassification rate about the average over the 25 repetitions.

Consider a 2 class example. The discriminant function $\lambda(x) = \beta_0 + \beta_1 x$ classifies to class 0 or 1 according as $\lambda(x)$ is greater or smaller than 0. When $\beta_1 = (\mu_1 - \mu_0)\Sigma^{-1}$ and $\beta_0 = \log \frac{\pi_0}{\pi_1} - \frac{1}{2}\beta_1(\mu_1 + \mu_0)$ are computed from the full likelihood (ie MLE of Σ, μ_0, μ_1) we are in the situation of LDA. When β_0, β_1 are determined by maximizing the conditional likelihood, we are in the situation of logistic regression. What happens if we optimize a convex combination of the full and conditional likelihoods, assuming gaussian classes with common covariance? The objective function is now

$$L_\lambda = \lambda \sum_{i=1}^{n} \log(\pi_{y_i} \mathcal{N}(\mu^{y_i}, \Sigma)[x_i]) + (1-\lambda) \sum_{i=1}^{n} \log \left(\frac{\pi_{y_i} \mathcal{N}(\mu^{y_i}, \Sigma)[x_i]}{\sum_k \pi_k \mathcal{N}(\mu^k, \Sigma)[x_i])} \right).$$

Denoting

$$\mathcal{N}_k[x_i] = \pi_k \mathcal{N}(\mu^k, \Sigma)[x_i]),$$

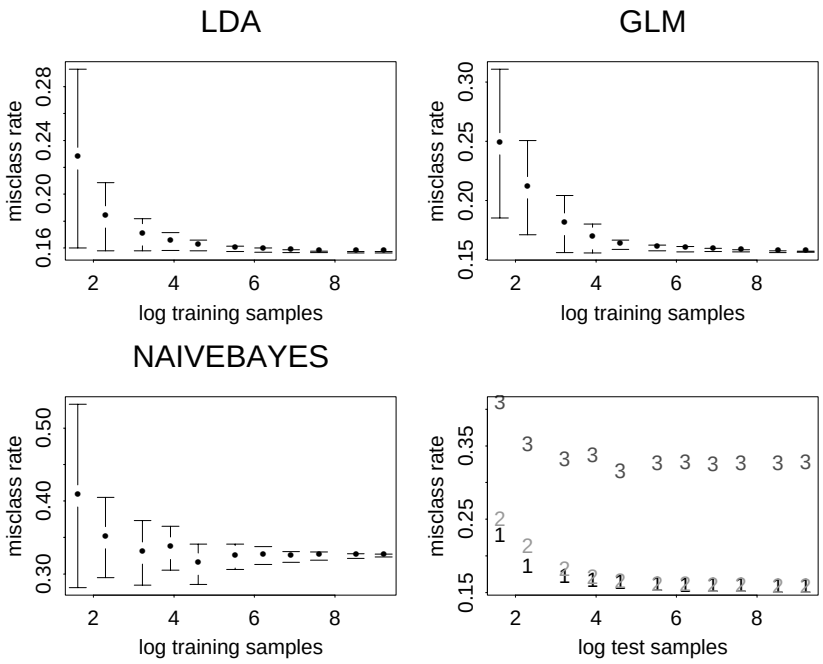

Figure 3.2: Classification Error comparing three techniques: LDA, GLM (logistic regression), naive bayes (with univariate gaussians along each dimension). Two gaussian classes in $\mathcal{R}^2$ were generated with $mu_0 = (0, 0)$, $mu_1 = (2, 0)$, and $\Sigma_{i,j} = diag(1, 3)$. A test set of 20000 observations was classified according to models obtained using different training set sizes. For each training set size, 25 models were trained, and the error bars indicate the variation of the misclassification rate about the average over the 25 repetitions.

and keeping in mind its dependence on μ_k and Σ, we have

$$L_\lambda = \sum_{k=1}^{K} \sum_{\substack{i \\ y_i=k}} \log \mathcal{N}_k[x_i] - (1-\lambda) \log \sum_j \mathcal{N}_j[x_i].$$

Differentiating with respect to μ_k and Σ and setting equal to zero, we obtain the "score" equations

$$\begin{aligned} \mu_j &= \frac{\sum_i x_i \left[\frac{(1-\lambda)\mathcal{N}_j[x_i]}{\sum_k \mathcal{N}_k[x_i]} - 1_{\{y_i=j\}} \right]}{n(1-\lambda) - n_j} \\ \Sigma &= \end{aligned}$$

Note that when $\lambda = 1$ we get the usual maximum likelihood estimates. Because $\mathcal{N}_j[x_i]$

depends on μ_j, Σ_j, we have to solve for μ_j, Σ_j numerically.

3.2.1 Maximum Likelihood Estimation

If the Bayes classifier is optimal, then maximizing the conditional likelihood $\prod_i p_\theta(y_i|x_i)$ with respect to θ seems like the best thing to do in that it seeks to maximize the class posterior probabilities $p(y|x)$, precisely what the Bayes classifier requires. Why then should maximizing the full likelihood ever lead to a better classifier?

The class posterior probablities are estimated based on a training set of observations $T = \{(x_i, y_i) : i = 1, \ldots, n\}$.

$$p_\theta(y = j|x, T) = \frac{p_\theta(x|y = j, T)p_\theta(y = j)}{\sum_k^K p_\theta(x|y = k, T)p_\theta(y = k)}$$

Treating the unknown parameter vector θ as a random variable, we have

$$\begin{aligned} p(x|y = j, T) &= \int p(x, \theta|y = j, T)d\theta & (3.2) \\ &= \int p(x|\theta, y = j, T)p(\theta|y = j, T)d\theta. & (3.3) \end{aligned}$$

If $p(\theta|y = j, T)$ is sharply peaked about the value $\hat{\theta}$, then [4][p.51]

$$p(x|y = j, T) \simeq p(x|\hat{\theta}, y = j, T)$$

ie the MLE for the class conditional densities.

So we see that in the case where the distribution of θ is highly peaked, maximizing the full likelihood leads to resonable plug-in estimates for the class posterior probabilities.

As an aside, the conditional likelihood is related to the Bayes classifier only if the model is correct, ie if the chosen $p(y|x)$ is in fact the true one. As Friedman [7] points out, even if the model is incorrect (ie biased), it can still do well, provided the bias is in the correct direction.

3.2.2 Convex Combination of Procedures

One way of combining logistic regression and NLDA is to take a convex combination of the full and conditional likelihood and optimize this over the same parameter space $\{\mu_1, \mu_2, \Sigma\}$.

Another possibility is to optimize each procedure separately and then combining the resultant discriminant functions. Both procedures produce a linear discriminant $\beta^T x$. Logistic regression's β is obtained by numerically optimizing the conditional likelihood, NLDA's β is obtained by plugging in the MLE's.

Both procedures, NLDA and logistic regression, come up with a discriminant function

$$d(x) = \beta^T x$$

whose value can be mapped into the interval [0,1] by

$$f(x) = \frac{e^{d(x)}}{1 + e^{d(x)}}.$$

Observations whose score is greater than a half are classified into class 1. For a given test set, consider the distribution of the scores, and focus on the misclassified observations. Is there any hope that a convex combination of the scores from the two procedures can improve the misclassification rate over either procedure alone? The error for a given procedure f and test set is

$$error(f) = \sum_{\substack{i \\ y_i=1}} 1\{f(x_i) < \frac{1}{2}\} + \sum_{\substack{i \\ y_i=0}} 1\{f(x_i) > \frac{1}{2}\}.$$

The question is, for $f = \lambda f_1 + (1-\lambda) f_2$ where f_1 is NLDA and f_2 is logistic regression, say, what are the conditions under which

$$error(f) < \ min\{error(f_1), error(f_2)\}$$

How is the solution of the convex combination of the two criteria related to the separate solutions?

Examine the bias and variance of the decision rule using the full and conditional likelihood under different assumptions of the underlying density. Does the combined rule have lower variance?

Whitlark and Geurts [32] examine combined forecasts by looking at the variance and the probability of lowest error of a combined forecast. Let

$$f(x) = \lambda f_1(x) + (1-\lambda) f_2(x)$$

be a combined forecast of procedures f_1 and f_2. First let us consider the case when $\lambda = 0.5$. Then the variance of the combined forecast is

$$var(f(x)) = \frac{var(f_1(x))}{4} + \frac{var(f_2(x))}{4} + \frac{cov(f_1(x), f_2(x))}{2}.$$

The combined forecast will have lower variance than either of the individual procedures if f_1 and f_2 are negatively correlated or even just a bit uncorrelated. Whitlark and Geurts investigate conditions when it pays off to combine forecasts:

> f_1 and f_2 should be negatively correlated (uncorrelated is actually good enough).
>
> Both should be good forecasts (if one forecast is always right and the other is always wrong, they will be highly negatively correlated, but you would do better to always use the single good procedure).

For the general convex combination of procedures $f(x) = \lambda f_1(x) + (1-\lambda) f_2(x)$ can we say what value of λ will yield the best results? What do we mean by "best"? Friedman [7] shows that in classification problems the classical "bias-variance" tradeoff need not apply. He notes that low bias is not critical in classification (so long as the prediction stays on the correct side of the decision boundary), and that it is enough to reduce variance to reduce the misclassification rate. That said, let us examine the $var(f)$ as a function of λ, and minimize $var(f)$.

$$\begin{aligned} var(f(x)) &= var(\lambda f_1(x) + (1-\lambda) f_2(x)) \\ &= \lambda^2 var(f_1(x)) + (1-\lambda)^2 var(f_2) + 2\lambda(1-\lambda) cov(f_1(x), f_2(x)) \end{aligned}$$

Taking derivatives and setting equal to zero

$$\begin{aligned} \frac{\partial var(f(x))}{\partial \lambda} &= 2\lambda var(f_1(x)) - 2(1-\lambda) var(f_2(x)) + 2(1-2\lambda) cov(f_1(x), f_2(x)) \\ \Rightarrow \lambda &= \frac{var(f_2(x)) - 2cov(f_1(x), f_2(x))}{var(f_1(x)) + var(f_2(x)) - 2cov(f_1(x), f_2(x))} \end{aligned}$$

We will want to examine the correlation and variance of each procedure. What sources of variability are there? We are fixing the observation x. There is variability due to the sampling of test data and training data. We discuss each of these below.

3.2.3 Simulation of Combined Procedures

We investigated whether there exists practical situations where combining procedures improves classification performance over using a single procedure. In this simulation study, logistic regression (LOGREG) was combined with normal linear discriminant analysis (NLDA). Three distributional scenarios were examined

- Both classes having a multivariate gaussian distribution with common covariance Σ. Without loss of generality, it is enough to consider $\Sigma = I$ and $\mu_1 = (0, \ldots, 0), \mu_2 = (\delta, 0, \ldots, 0)$ since both procedures are invariant to linear transformations of the space.
- Both classes having a product of univariate cauchy densities $p(x|y = k) = p(x; \text{location} = \mu_k) \prod_{i=2}^{p}(x; 0)$.
- Mixture of gaussian and cauchy data within each class.

Design of Simulation

Design of the simulation required appropriate values for the class separations, the number of training samples, the number of test samples, and the dimension of the feature space.

Class Separation: If the classes are too well separated, very few misclassifications would result from either procedure, and we are interested after all in comparing misclassifications. When the classes are too close together, the ARE (see Efron [5]) of LOGREG to NLDA is close to 1 (ie both procedures give the same results). A separation of 3 was chosen based on the table in Efron which gives an ARE of .641.

Dimensionality: There is a relationship between the dimension of the feature space and the number of training and test samples required. In general, the higher the dimension, the more samples required (the curse of dimensionality) to estimate both the model parameters and the misclassification errors. It would seem that for the class densities under consideration (multivariate Gaussian, product of independent cauchys) it would be enough to consider a univariate feature space, since for example, NLDA reduces to a classification along a single line in the full dimensional space, and the data, projected on this line is also Gaussian. However, it is not true for an arbitrary procedure that the multivariate problem reduces to an equivalent univariate problem (eg a linear combination of cauchys is not cauchy). So it is necessary to examine several values of the dimension of the space.

Test samples: We are interested in studying the misclassification rate as a function of the mixing parameter λ. For a given λ, and training set, the estimate of the misclassification rate will vary due to the random sampling of the test set. We want to choose the test set to be large so that this variation is very small. After all, the test set is being used to effectuate a Monte Carlo integration. The misclassification error due to a procedure g is given by

$$err(g) = \pi_1 \int_{g(x)<0.5} p(x|y=1)dx + \pi_0 \int_{g(x)>0.5} p(x|y=0)dx.$$

If we knew the class conditional densities, we could compute $err(g)$ exactly. For a given procedure g, then the test error variance is due to variance in estimating the class conditional densities. In order to keep computation at a reasonable level we want to use the smallest test set that gives us acceptable test error variance.

Training samples: If the number of training samples was sufficiently large, then, provided both classification procedures were unbiased, there would not be much difference in the resultant decision boundaries. We are primarily interested in the domain of small training samples.

Raudys and Pikelis [26] derive exact expressions for the expected misclassification error as a function of training size, class separation, dimensionality for several discriminant functions including NLDA. They tabulate values of $E(P_N)/P_\infty$ where P_N is the probability of misclassification using a given procedure with N training samples. Table 3.1 gives some values for the NLDA procedure. Note that in particular, for $p = 5$ dimensions, a training sample of 250 observations is enough to obtain a misclassification rate that on average (taken across possible training sets) is 1.02 times the Bayes rule (since NLDA is unbiased for gaussian classes).

p	3	5	8	12	20	50
N	150	250	400	600	1000	2500

Table 3.1: Training samples (N) and dimension (p) for a separation of $\delta = 2.56$ and $E(P_N)/P_\infty = 1.02$

Using a fixed test set of 10000 observations, 25 training sets of (10, 25, 50, 125, 250, 500) observations were generated according to a multivariate-5 gaussian distirbution with a separation of 3. The misclassification rate using both LOGREG and NLDA was recorded,

as well as the correlation between the 2 predictors.

Training samples	NLDA misclass	LOGREG misclass	cor(LOGREG,NLDA)
10	0.105 +/- 0.0487	0.140 +/- 0.0679	0.85 +/- 0.12248
25	0.075 +/- 0.0071	0.097 +/- 0.0198	0.95 +/- 0.03633
50	0.071 +/- 0.0046	0.079 +/- 0.0109	0.98 +/- 0.02546
125	0.067 +/- 0.0019	0.069 +/- 0.0028	1.00 +/- 0.00256
250	0.065 +/- 0.0013	0.066 +/- 0.0023	1.00 +/- 0.00219
500	0.065 +/- 0.0011	0.065 +/- 0.0012	1.00 +/- 0.00051

Table 3.2: Misclassification rate as a function of training sample size for LOGREG and NLDA, simulation results. Misclassifications and SD's are averaged across 25 repetitions. p = 5. Test set = 10000. Class densities are gaussian.

Using a test set of 10000 observations, 50 training sets each containing 26 observations (13 from each class) were used to train both LDA and LOGREG classifiers. For each training model, a convex combination of the discriminant function was used for varying values of the mixing parameter, and the misclassification rate on the test set computed. This procedure was carried out on three types of data where both the training and test sets followed the same distribution:

1. 5-variate gaussian classes with a separation of 3 along one direction (note invariance to affine transformations).

2. 5-variate independent cauchys all centered at 0 except for one dimension in which the class separation as 3.

3. A mixture of the above two distributions.

Figure 3.3 shows some typical misclassification curves as a function of the mixing parameter for various training sets. For each training set, an NLDA and LOGREG model was fitted. This produced 2 linear discriminants β_{LDA} and β_{LOGREG}. The combined discriminant is

$$\begin{aligned} d(x) &= \lambda d_{LDA}(x) + (1-\lambda) d_{LOGREG}(x) \\ &= \lambda \frac{e^{\beta_{LDA}^T x}}{1+e^{\beta_{LDA}^T x}} + (1-\lambda)\frac{e^{\beta_{LOGREG}^T x}}{1+e^{\beta_{LOGREG}^T x}}. \end{aligned}$$

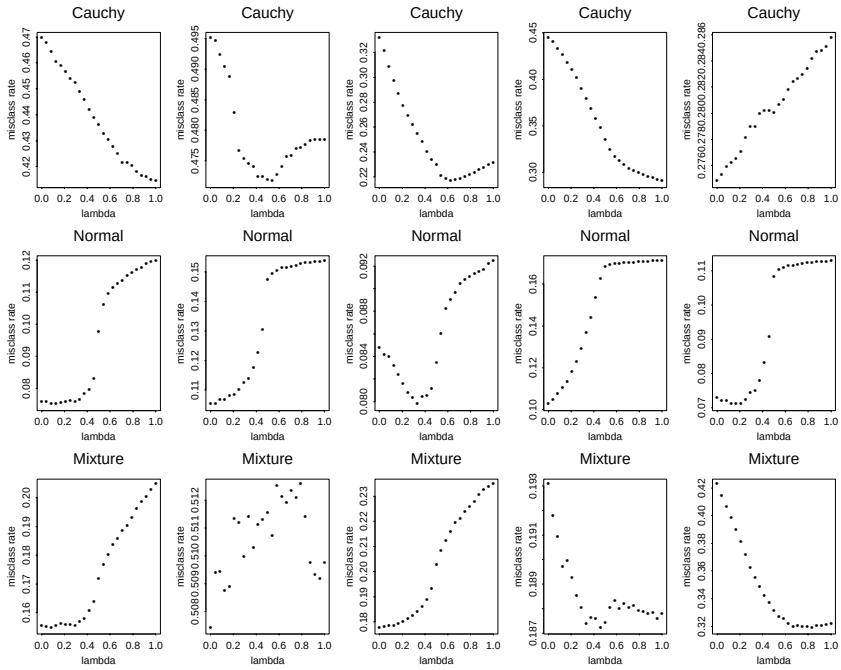

Figure 3.3: Misclassification rate as function of the mixing parameter for a random sample of 5/50 training sets from each of 3 populations. Top row is for cauchy classes, middle is gaussian, bottom is mixture. $\lambda = 0$ corresponds to LOGREG, $\lambda = 1$ corresponds to NLDA. As expected, for gaussian data, NLDA outperforms LOGREG most of the time. Pay attention to the vertical scale which is different for each plot. Solid lines represent actual test error, dotted lines are cross validated estimates of the test error obtained from the training data by a leave one out procedure.

For each value of the mixing parameter λ, the misclassification rate on the test set was computed. The cases that are especially interesting are those for which a mixture of the NLDA and LOGREG procedures outperforms either one.

3.2.4 Analysis of simulation results

Under what circumstances is it better to use a mixture of the two procedures? We analyzed the misclassification curves by looking at the value of λ at which the minimum misclassification occured. It is also important to consider the magnitude of the improvement in the misclassification rate. Hence, the misclassification curves were characterized by the misclassification rate range (max - min), and the location of the minimum.

The training sets were characterized by goodness of fit (GOF) to a gaussian. Twenty-five random directions were chosen, and the data projected onto each of them. If the data were multivariate gaussian, any such projection should be univariate gaussian. A Kolmogorov-Smirnov goodness of fit was carried out on each of the 25 projections, and the maximum value of the KS statistic across the projections was used as a measure of the GOF for the multivariate training set.

Figure 3.4 shows qqplots for several projections of training sets (the same ones for which misclassification curves are presented in figure 3.3). The general trend seems to be that the non-gaussian multivariate data indeed gives rise to a non-gaussian univariate projection.

Our hypothesis is that for gaussian-like data the NLDA procedure will lead to a better classifier, whereas for non-gaussian data LOGREG will be better. Asymptotically (ie for large training sets) NLDA is the optimal procedure for gaussian classes. In the small sample case, however, even if the true underlying class distributions are normal, the observed training set may not be.

Figure 3.5 shows that for data that looks gaussian, NLDA (minloc = 0) works best. As the GOF measure increases (less gaussian), LOGREG or a mixed criterion classifier work better (ie minloc ¿ 0), especially in cases where there is a large benefit to be obtained (range ¿ 0.05). Note that for extremely non-gaussian cases, the range distribution is concentrated in [0,0.1], ie a 0 to 10% improvement can be achieved by optimizing the mixing parameter. For moderately non-gaussian cases the improvement range goes up to 30% and this is for a mixed criterion.

Figure 3.6 shows

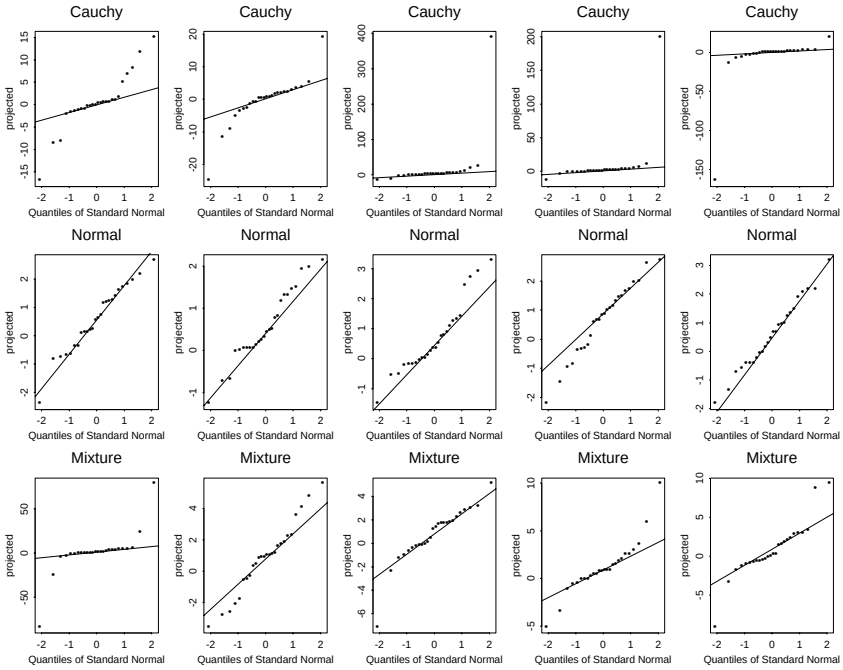

Figure 3.4: QQplot for random projections of the training data. The accompanying fitted line is a robust LMS regression line.

Figure 3.7 shows min location broken down by range and gof. Going from left to right: gof measure increases (ie less gaussian on right), bottom to top: range increases. We see that for gaussian data the min location is often at 0 (ie NLDA is the best procedure) and the penalty for not using NLDA can be quite steep (large range). For severely non-gaussian data the same is true with respect to LOGREG. However for intermediate levels of the gof measure, it can often pay to choose a mixed procedure.

3.3 Mixed Likelihood as Regularized Discrimination

We have seen that combining two discriminants, in particular one based on the full likelihood $\prod_i p(x_i, y_i)$ with one based on the conditional likelihood $\prod_i p(y_i|x_i)$, can lead to improved classification performance. The expected error using classifier $g : x \to \{1, \ldots, K\}$ is

$$\begin{aligned} E(\text{error using } g) &= \int\int p(x,y) 1\{g(x) \neq y\} dxdy \\ &= \int\int p(x|y)p(y) 1\{g(x) \neq y\} dxdy \end{aligned}$$

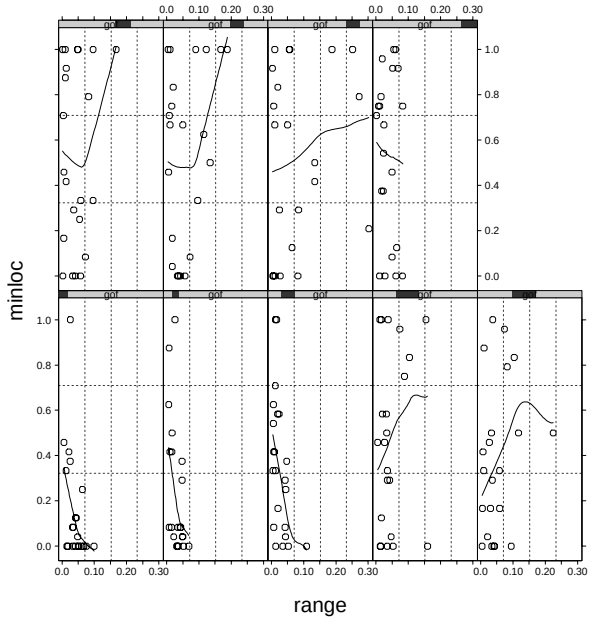

Figure 3.5: Conditioning plot of range vs min location, conditioned on gof across all 150 training sets from all 3 populations.

$$= \sum_k p(y=k) \int p(x|y=k) 1\{g(x) \neq k\} dx$$

This is curious: the Bayes rule tells us to classify x to the class for which $p(y|x)$ is greatest. Maximizing the conditional likelihood provides us with parameters which maximize the product of observed $p(y_i|x_i)$. It would therefore seem that any other classification procedure would be suboptimal. In particular, we should be surprised that maximizing the full likelihood should lead to a procedure which outperforms one based on the conditional likelihood.

To gain some insight into why maximum full likelihood might outperform maximum conditional likelihood likelihood, we show that the full likelihood is the conditional likelihood plus an unsupervised learning term which regularizes it.

The log likelihood of the training data is

$$L[(x_1, y_1), \ldots, (x_n, y_n)] = \sum_i \log p(x_i, y_i)$$

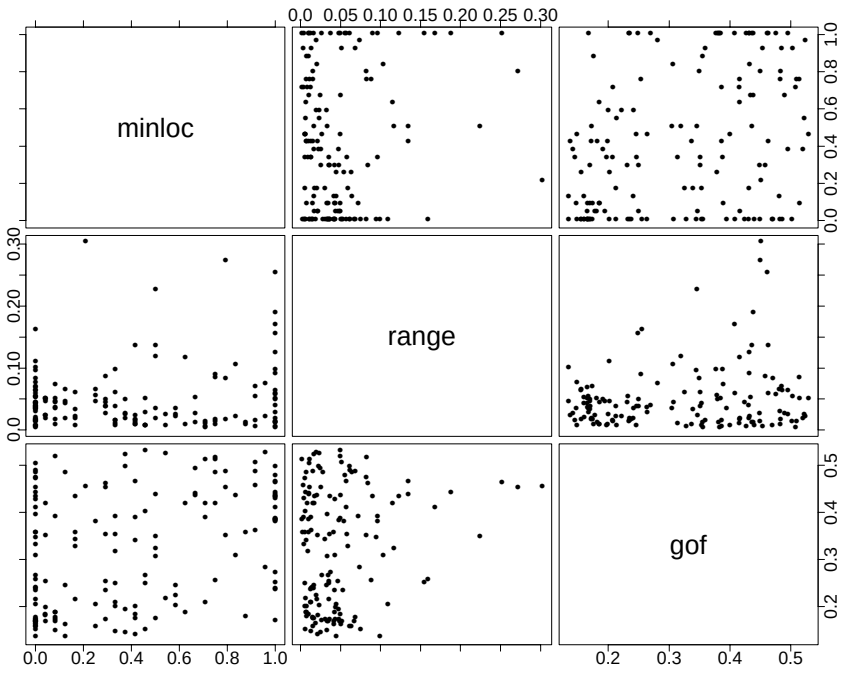

Figure 3.6: Pairwise plots of min location, misclassification range, and gof.

$$= \sum_i \log p(y_i|x_i) + \sum_i \log p(x_i).$$

We introduce a parameter $\lambda \in \mathcal{R}^+$ which will control the degree of regularization of the likelihood

$$\begin{aligned} L(\lambda)[(x_1, y_1), \ldots, (x_n, y_n); \theta] &= \sum_i \log p(y_i|x_i; \theta) + \lambda \sum_i \log p(x_i; \theta) \\ &= L_{y|x}(\theta) + \lambda L_x(\theta) \\ &= (L_{x,y}(\theta) - L_x(\theta)) + \lambda L_x(\theta) \\ &= L_{x,y}(\theta) + (\lambda - 1) L_x(\theta). \end{aligned}$$

$\lambda = 1$ corresponds to the usual full likelihood.

$\lambda = 0$ corresponds to the conditional likelihood case.

$\lambda = \infty$ corresponds to the unsupervised learning case, ie class labels are completely ignored.

The last form of the mixed likelihood $L(\lambda)$ suggests an iterative method for solving for

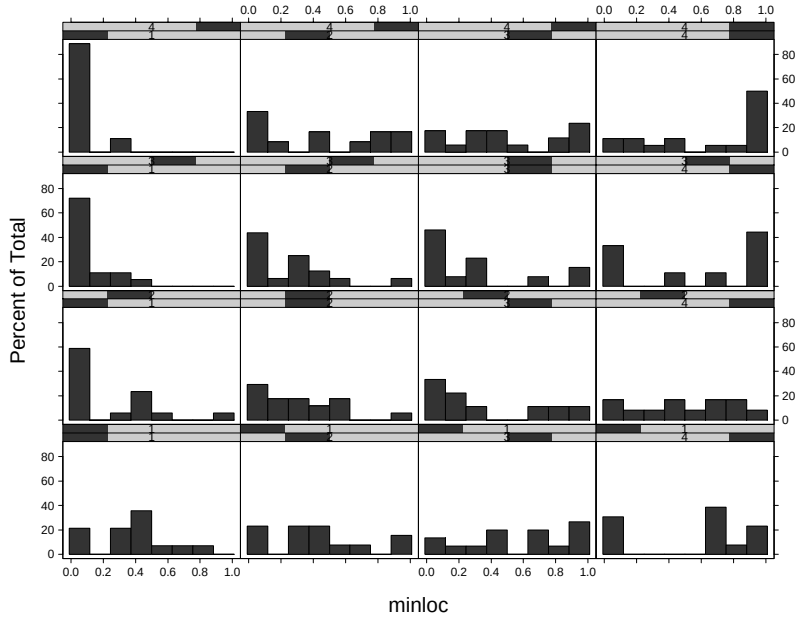

Figure 3.7: Histogram of min location broken down by gof and range. The bottom horizontal bar refers to gof, the upper to range.

the parameters θ based on EM. Suppose the MLE's of the full likelihood $L_{x,y}$ are easy to obtain. Then the parameters of L_x can be obtained via the EM algorithm:

E step Given current value of θ, estimate $p(y = j|x_i; \theta) = [w_{i,1}, \ldots, w_{i,K}]$.

M step Given $p(y = j|x_i; \theta)$ update θ by finding the

$$\begin{aligned} & \max_\theta \sum_i^n \sum_j^K y_{i,j} \log f_j(x_i, \theta) + (\lambda - 1) \sum_i^n \sum_j^K w_{i,j} \log f_j(x_i, \theta) \\ = \quad & \max_\theta \sum_i^n \sum_j^K (y_{i,j} + (\lambda - 1) w_{i,j} \log f_j(x_i, \theta). \end{aligned}$$

This will be the usual MLE but with each observation weighted by $y_{i,j} + (\lambda - 1)w_{i,j}$.

The form of the weighted MLE also provides some insight as to why maximizing the full likelihood might lead to better classification than maximizing the conditional likelihood, even though the latter is more directly concerned with classification. An observation x_i that clearly belong to its class j will have $w_{i,j} \approx 1$ so that its weight in the weighted likelihood

will be $y_{i,j}+(\lambda-1)w_{i,j} = 1+(\lambda-1)1 = \lambda$. Now, the conditional likelihood case corresponds to $\lambda = 0$, so that such an unconfusable observation will get weight 0. The only observations that enter the likelihood then are those which are confusable, ie on the class boundaries. The effective training set size then is greatly diminished. (A similar conclusion follows from analyzing the iterative procedure used in logistic regression, see below). The full likelihood takes into account all observations, even those far removed from the decision boundary. If the form of the distribution is correct, then this helps the classification which is based on the plug-in rule using the MLE's.

There is a problem when $\lambda \leq 1$ because then the effective weight might be negative which can lead to MLE's that don't make sense (eg non-positive definite covariance estimates). One option is to limit attention to the case where $\lambda \geq 1$. Another is to force any negative weights to be 0. Yet another is to rescale the weights to lie in [0,1]. The minimum value $y_{i,j} + (\lambda - 1)w_{i,j}$ can take is -1. The maximum value is 1 if $\lambda \leq 1$ and λ if $\lambda \geq 1$. All these solutions are a bit ad hoc and unsatisfactory. For this reason the EM algorithm is unsuitable for maximizing the mixed likelihood when $\lambda \leq 1$.

3.4 Regularized Conditional Likelihood

The full likelihood can be decomposed as

$$L(x, y) = L(y|x) + L(x)$$

Introducing a parameter λ, we obtain

$$L_\lambda(x, y) = L(y|x) + \lambda L(x) = L(x, y) + (\lambda - 1)L(x)$$

where the interpretation is that the marginal distribution of x regularizes the conditional likelihood.

In the 2 class, normal case (common covariance),

$$p(x|y = k) = \mathcal{N}(\mu^k, \Sigma)[x] \text{with probability } \pi_k$$

the full likelihood, $L(x, y)$ gives rise to MLE's $\hat{\pi_1}, \hat{\pi_2}, \hat{\mu_1}, \hat{\mu_2}, \hat{\Sigma}$ which are then used to obtain

a linear discriminant β via a plug-in rule (ie Fisher's linear discriminant function).

$$\begin{aligned}\hat{\beta}_0 &= \log\frac{\pi_1}{\pi_2} + \frac{1}{2}(\hat{\mu_1} + \hat{\mu_2})\hat{\Sigma}^{-1}(\hat{\mu_1} - \hat{\mu_2} \\ \hat{\beta}_1 &= \hat{\Sigma}^{-1}(\hat{\mu_1} - \hat{\mu_2})\end{aligned}$$

The plug-in rule is derived by examining the likelihood ratio

$$\begin{aligned}\log\frac{p(x|y=1)}{p(x|y=2)} &= \log\frac{\pi_1 \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu 1)\right)}{\pi_2 \exp\left(-\frac{1}{2}(x-\mu_2)^T\Sigma^{-1}(x-\mu 2)\right)} \\ &= x^T\Sigma^{-1}(\mu_1 - \mu_2) + \log\frac{\pi_1}{\pi_2} + \frac{1}{2}(\mu_1 + \mu_2)\Sigma^{-1}(\mu_1 - \mu_2)\end{aligned}$$

Three different ways of combining discriminative and non-discriminative training:

1. Take a convex combination of their discriminants, ie find the MLE's and the optimal β via logistic regression and then form

$$\gamma(x) = \lambda\frac{\exp(\beta_{LDA}^T x)}{1+\exp(\beta_{LDA}^T x)} + (1-\lambda)\frac{\exp(\beta_{LOGREG}^T x)}{1+\exp(\beta_{LOGREG}^T x)}.$$

2. Optimize the mixed likelihood directly. This will yield a single β (or equivalently a set of μ_i, Σ, π_i) which maximizes

$$L_\lambda(x,y) = L(y|x) + \lambda L(x).$$

   Problem with gradient descent is that we need to introduce nonlinear constraints on Σ to ensure postive definiteness and symmetry, as well as potential unidentifiability of the parameters when $\lambda = 0$.

3. Optimize the criterion

$$L_\lambda(x,y) = L(y|x) + \lambda||\beta - \hat{\beta}||^2$$

   where $\hat{\beta}$ is obtained from the plug-in rule. This can easily be solved as an unconstrained gradient descent.

3.5 Descent Procedeures

Newton-Raphson. IRLS. Gradient Descent.

3.6 Gaussian Classes

Class probs. Gradients. Problems: Σ symmetric and positive definite. Unidentifiability of original parameters when in pure conditional likelihood situation. Solution: reparametrize (eg $S = VDV'$ with $D = diag(\exp(d_j))$) introduce constraints

3.7 Tangent Distance

Tangent distance: $D(x_i, x_j) = \min_{\theta_i, \theta_j)} ||x_i + T_{X_i}\theta_i - x_j + T_{X_j}\theta_j||$.

Comparison of Hastie to Sona: Hastie et al. [15] find a centroid M and a subspace V that minimize

$$\sum_i^n \min_{\gamma_i, \theta_i} ||M + V\gamma_i - x_i(\theta_i)||^2$$

where $x_i(\theta_i) = x_i + T_i\theta$. The centroid and subspace are estimated independently within each class using an iterative method with an SVD decomposition at its core. Sona et al. [29] investigate a 2 class discriminant approach where a centroid M and a subspace $V = (V_1, \ldots, V_p)$ is sought such that the error function

$$E = \frac{1}{2}\sum_i^n (y_i - \hat{y}_i)^2$$

is minimized where $y_i \in \{0, 1\}$ is the actual class of the i^{th} observation and

$$\hat{y}_i = \left[1 + \alpha(||x_i - M||^2 - \sum_i^p [(x_i - M)'V_i]^2)\right]^{-1}.$$

A gradient descent is used to find the minimizing M, V, α

3.8 Gradient Descent Procedure

The full likelihood is a concave function of μ, Σ. [DAN show details] A gradient descent procedure can therefore be used to find the maximum. The problem is that Σ is constrained

to be symmetric and positive definite, ie $\Sigma = \Sigma^T > 0$. So we cannot simply follow the gradient of the likelihood because this may lead us to a Σ that is not positive definite (symmetry will always obtain because the gradient with respect to Σ is also symmetric).

Consider first the case of a single gaussian $x_i \sim \mathcal{N}(\mu^k, \Sigma)$. The MLE's are well know to be

$$\begin{aligned} \hat{\mu} &= \bar{x} = \frac{1}{n}\sum_i x_i \\ \hat{Sigma} &= S = \frac{1}{n}\sum_i (x_i - \bar{x})(x_i - \bar{x})^T \end{aligned}$$

Suppose you were to try and obtain the MLEs via gradient ascent. In this simple case, this would be a silly thing to do, but we illustrate it to show how to incorporate the constraints on Σ in a simple way.

The log likelihood of the data (ignoring an additive constant) is given by

$$L = \sum_i \frac{1}{2}\log\det(\Sigma^{-1}) - \frac{1}{2}(x_i - \mu)^T\Sigma^{-1}(x_i - \mu).$$

The gradient is

$$\begin{aligned} \frac{\partial L}{\partial \mu} &= \sum_i (x_i - \mu)\Sigma^{-1} \\ \frac{\partial L}{\partial \Sigma^{-1}} &= n\left[\Sigma - \frac{1}{2}\mathrm{diag}(\Sigma) - \mathrm{S} + \frac{1}{2}\mathrm{diag}(\S)\right] \end{aligned}$$

which gives update equations ($\alpha, \beta \geq 0$)

$$\mu_1 = \mu_0 + \alpha(\sum_i x_i - \mu_0)\Sigma_0^{-1}\Sigma_1^{-1} = \Sigma_0^{-1} + \beta n\left[\Sigma - \frac{1}{2}\mathrm{diag}(\Sigma) - \mathrm{S} + \frac{1}{2}\mathrm{diag}(\S)\right]$$

As an aside, setting the gradient equal to 0 yields the expressions for the MLE's given above. Under what circumstances will our iteration on Σ cause us to obtain a value that violates the constraint $\Sigma > 0$. Note first that the set of all symmetric, positive definite matrices

$$\mathcal{M} = \{\Sigma | \Sigma = \Sigma^T > 0\}$$

is a convex cone

$$\alpha_1, alpha_2 \geq 0, \Sigma_1, \Sigma_2 \in \mathcal{M} \Rightarrow \alpha_1\Sigma_1 + \alpha_2\Sigma_2 \in \mathcal{M}.$$

Let $A = \frac{\partial L}{\partial \Sigma^{-1}}$. Then $\Sigma_1^{-1} = \Sigma_0^{-1} + \beta A$ will be positive definite if $\beta \geq 0, A \geq 0$. If $A \leq 0$ then we must be careful when choosing the step size β to ensure that $\Sigma_1^{-1} \geq 0$. [DAN: is there an easy way to determine the maximum such β?]

Let us return to the 2 gaussian class, mixed likelihood case.

3.9 Unconstrained MLE

Tried minimizing the negative log likelihood of a single gaussian using S-plus' nlmin (see trymin.s). Starting away from the MLE, the procedure quickly converged to a (μ, Σ) near the MLE. Starting at the MLE, the procedure did not converge and actually walked to a singular solution (ie a Σ with one of its eigenvalues approaching 0).

In [27], Redner and Walker describe the application of the EM algorithm to estimating the parameters of mixture densities. The problem with our set up is that the "unlabeled" observations enter via $L(x)$ which is always subtracted when considering additive combinations of $L(x, y)$ and $L(y|x) = L(x, y) - L(x)$ which can give rise to negative weights (as we have seen).

3.10 Logistic regression ridged towards MLE

We view the problem as finding β that minimizes

$$-L(y|x) + \lambda||\beta - \hat{\beta}_{MLE}||^2.$$

We derive an iteratively reweighted least squares (IRLS) algorithm for obtaining β. The procedure is based on the Newton-Raphson procedure which is solved at each iteration using a (weighted) linear regression.

Suppose we seek the stationary point of a function $f(x)$, ie that point at which $f'(x) = 0$. The Taylor expansion of $f'(x)$ about a point x_0 is

$$f'(x) = f'(x_0) + (x - x_0)f''(x_0).$$

Solving for $f'(x) = 0$ and rearranging terms suggests the iterative procedure

$$x_1 = x_0 - [f''(x_0)]^{-1} f'(x_0)$$

which is the Newton-Raphson algorithm. IRLS involves manipulating the Newton-Raphson equations so that they look like the normal equations for linear regression. In linear regression we are given Y (n,1) and X (n,p), and possible weights W (diagonal (n,n) and we seek β (p,1) that minimizes

$$f_\beta = (Y - X\beta)^T W (Y - X\beta).$$

The normal equations are

$$\begin{aligned} \frac{df_\beta}{d\beta} &= -2X^T W(Y - X\beta) \\ \Rightarrow X^T WY &= X^T WX\beta \\ \Rightarrow Z^T(W^{-\frac{1}{2}}Y) &= Z^T Z\beta. \end{aligned}$$

In our case the function we seek to minimize is

$$f = L(y|x) + \lambda||\beta - \hat{\beta}_{MLE}||^2 = \sum_i y_i x_i'\beta - \log(1 + e^{\beta' x}) + \lambda||\beta - \hat{\beta}_{MLE}||^2.$$

The score equation is

$$\begin{aligned} \frac{\partial f}{\partial \beta} &= \sum_i x_i(y_i - p_i) - 2\lambda(\beta - \hat{\beta}_{MLE}) \\ &= X^T(Y - p) - 2\lambda(\beta - \hat{\beta}_{MLE}) \end{aligned}$$

where $p_i = 1/(1 + e^{-\beta' x_i})$.

The hessian is

$$\frac{\partial^2 f}{\partial \beta \partial \beta} = -X^T WX - 2\lambda I$$

The Newton-Raphson equations are then

$$-\frac{\partial^2 f}{\partial \beta \partial \beta}(\beta_0)(\beta_1 - \beta_0) = \frac{\partial f}{\partial \beta}(\beta_0)$$

$$
\begin{aligned}
(X^T W X + 2\lambda I)(\beta_1 - \beta_0) &= X^T(Y - p) - 2\lambda(\beta_0 - \hat{\beta}_{MLE}) \\
Z^T Z \beta_1 &= Z^T(Z\beta_0 + [W^{-\frac{1}{2}}(Y - p), 0_{p,p}] - Z^T \begin{pmatrix} 0_{n,n} & 0_{n,p} \\ 0_{p,n} & I_p \end{pmatrix} Z(\beta_0 - \hat{\beta}_{MLE}))
\end{aligned}
$$

where both the hessian and the gradient are evaluated at the current estimate of β, and

$$Z_{n+p,p} = [W^{\frac{1}{2}}X, \sqrt{2\lambda}I_p]^T.$$

The last form of the equation can be identified with the normal regression equations which we iteratively solve for β. [DAN: had trouble with IRLS, although Newton-Raphson worked fine and agreed with glm()]

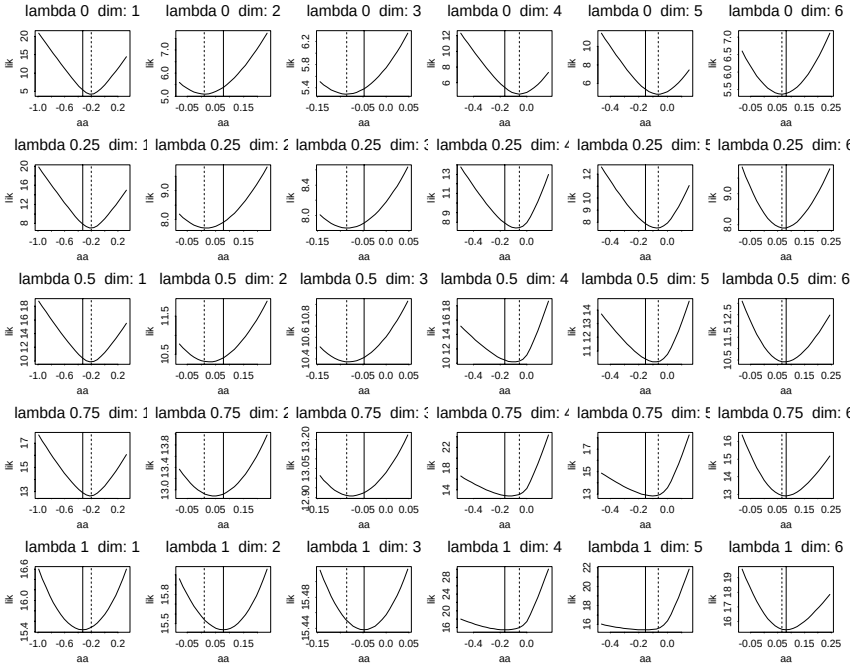

Figure 3.8: The MLE-ridged likelihood profile: $-(1-\lambda)L(y|x) + \lambda||\beta - \hat{\beta_{MLE}}||^2$ about β_{MLE}. Each row corresponds to a different value of the mixing term lambda, each column corresponds to a different dimension of β. The solid vertical line corresponds to β_{LOGREG}, the dashed line to β_{MLE}. This profile corresponds to the 1st cauchy training set (ie top left plot in figure 3.10). Note that the (negative) likelihood is lowest in the $\lambda = 0$ case. So what is the relationship between the ridged likelihood score and classification performance?

Figure 3.10 illustrates the misclassification curves for the 3 mixing procedures. No one method dominates the others. An analysis of the probability of error using the 3 procedured

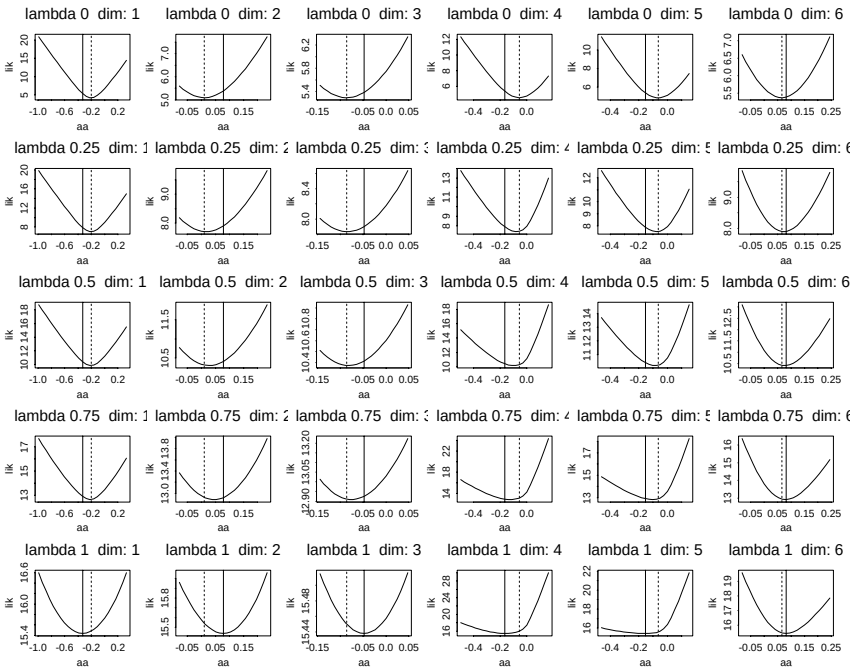

Figure 3.9: The MLE-ridged likelihood profile: $-(1-\lambda)L(y|x) + \lambda||\beta - \hat{\beta_{MLE}}||^2$ about β_{MLE}. Each row corresponds to a different value of the mixing term lambda, each column corresponds to a different dimension of β. The solid vertical line corresponds to β_{LOGREG}, the dashed line to β_{MLE}. This profile corresponds to the 1st gaussian training set (ie 2ndr row, 1st col plot in figure 3.10). Note that the (negative) likelihood is lowest for $\lambda = 0$, around 3000.

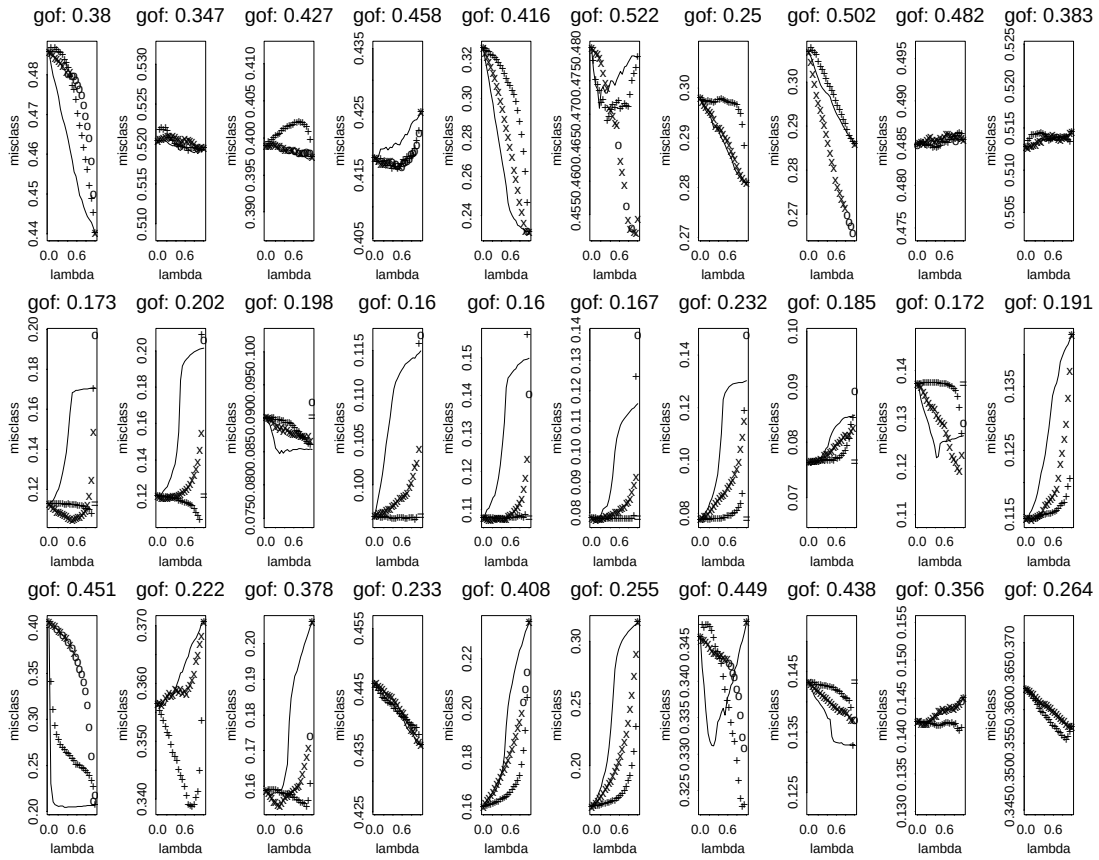

Figure 3.10: Misclassification curves as a function of the mixing parameter. Top row - cauchy data, middle row - gaussian data, bottom row - mixture data (p=5, numclass = 2). Columns correspond to unique training sets. Height of curve is misclassification rate. Solid line - mixed discriminants, x/o - mixed likelihood (converged/not converged), +/= - MLE ridged conditional likelihood (converged/not). The gof score above each graph is the maximum Kolmogorov-Smirnov statistic among 50 randomly chosen univariate projections of the data.

may yield insight into the circumstances when one method is superior to another.

In cases where the range of the misclassification curve is large, and the minimum occurs using a non-trivial mixed procedure (ie λ away from 0 or 1), the MLE-ridged method seems to give the best performance. Why? When the training data are perfectly separable, the logistic regression procedure has a hard time choosing a coefficient vector, because there are many that will give the same conditional likelihood (on the training data. Of course the test data might be poorly classified when β is so free). By regularizing with the MLE ridge, we are in effect "borrowing strength" from the marginal distribution, while still optimizing the quantity of interest: namely the class posteriors (via the conditional likelihood). The mixture of discriminants only hurts us by including the poorly estimated logistic regression discriminant. The mixed likelihood also borrows strength, but it has the disadvantage of having too many parameters (ie $|\{\mu_i, \Sigma, \pi_i\}| = pK+p(p+1)/2+K$ parameters vs $|\beta| = p+1$).

Chapter 4

Lospline Densities

Figure 4.1: Univariate projections of Class 1's density function.

4.1 Naive Bayes vs GAM

4.2 Simulation Data

We first construct a data set for which we expect the Naive Bayes approach to perform poorly relative to GAM. Naive Bayes models the multivariate class densities as a product of univariate densities

$$\log f^j(x) = \sum_i^p \log f_i^j(x_i).$$

We set the functional form of the univariate densities to be logspline densities:

$$\log f_i^j(x_i) = \alpha_0 + \alpha_1 x_i + \sum_{m=2}^{M} \alpha_m (x_i - t_m)_+^3 = \sum_m^M \beta_{i,m}^j B_m(x_i)$$

where the $B_m(x)$ are a natural cubic spline basis [14]. Given a set of real numbers t_m (called knots) in an interval $[a, b]$, with $a = t_0 < t_1 < \ldots < t_M < b = t_M + 1$, a natural cubic spline is a function on $[a, b]$ satisfying:

1. on each interval (t_m, t_{m+1}) g is a cubic polynomial, ie

$$g(x) = \alpha_{m,3}(x - t_m)^3 + \alpha_{m,2}(x - t_m)^2 + \alpha_{m,1}(x - t_m) + \alpha_{m,0} \quad \text{for} \quad t_m \leq x \leq t_{m+1}.$$

2. $g(x), g'(x), g''(x)$ are continuous at each t_m.

3. $g'(a) = g'(b) = g''(a) = g''(b) = 0$.

The reason for choosing a logspline density is that it is a rich family capable of modeling very complex, but smooth, densities.

4.2.1 Estimating the logspline density

[get Stone and Kooperberg reference here and clean up] The logspline density is estimated using constrained Maximum Likelihood. The data is supposed to arise from a logspline density

$$\exp(B^T(x)\beta + c)$$

In our initial comparisons, the knots, and hence the spline basis, are fixed to be the same in all dimensions, across all classes.

Figure 4.2: Effect of NCS exponential tilts on a bivariate gaussian density.

Figure 4.2 demonstrates the effect of tilting a bivariate gaussian. The original density was

$$f_A(x_1, x_2) = \mathcal{N}(\mu = (0,0)^T, \Sigma = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix})[x_1, x_2]$$

The contours of this density appear in all the subplots. For each of the spline bases $B_m(x)$ we tilted the original density and plotted the resulting contours

$$f_B^m(x_1, x_2) = \exp(\beta B_m(x_1)) f_A(x_1, x_2).$$

As can be seen in figure 4.2, either the tilted density f_B lies on top of the original density f_A and is hence confusable, or it can be distinguished by a Naive Bayes density, ie the tilted density and the original are distinguishable in at least one of the univariate marginals.

Figure 4.3 illustrates the effect of tilting along 2 dimensions simultaneously. Again, the density for class A is bivariate gaussian as above. Here $\beta_0 = (-25, -5, 5, 25), \beta_1 =$

Figure 4.3: Effect of NCS exponential tilts along both dimensions on a bivariate gaussian density.

$(0, \beta_0[i], 0, 0, 0), \beta_2 = (0, \beta_0[j], 0, 0, 0)$, and the tilting function was

$$\exp(\beta_1^T B(x_1) + \beta_2^T B(x_2))$$

where $B(x) = (B_1(x), \ldots, B_M(x))$. Again, in any case in which the 2 classes are not confusable, the Naive Bayes procedure would also seem to do well.

As confirmation, we chose the $i = 1, j = 4$ case to compare Naive Bayes and GAM. We know the exact densities for both classes. Our comparison across simulated training sets consists of

1. Kullback-Leibler distance (assuming $\hat{p}(y) = p(y)$)

$$D(p(y|x), p(\hat{y}|x)) \quad = \quad \int_x p(x)dx \int_y \log \frac{p(y|x)}{\hat{p}(y|x)} p(y|x)dy$$

2. James-Hastie misclassification bias/var measure

Figure 4.4: The contours of the class densities of the simulation data.

The simulation data was generated by choosing a mixture of 3 gaussians for the first class

$$P(x|y=1) = \sum_k \alpha_k \mathcal{N}(\mu^k, \Sigma)[x]$$

with $\mu_1 = (-2.5, -2.5), \mu_2 = (2.5, -2.5), \mu_3 = (0, 2.5), \Sigma = I_2, \alpha_k = \frac{1}{3}$. Then, a random search over spline coefficients β was carried out to find a tilting density

$$\exp((B(x_1), B(x_2))^T \beta)$$

that will determine class 2's density

$$P(x|y=2) = c \exp((B(x_1), B(x_2))^T \beta) P(x|y=1)$$

in such a way that the Bayes probability of error

$$P(\text{error}) = \int_{d(x)>0} P(x|y=1)dx + \int_{d(x)<0} P(x|y=2)dx$$

is roughly 0.2. Here, $d(x)$ is the discriminant

$$d(x) = \log \frac{P(Y=1|x)}{P(Y=2|x)} = \log \frac{P(x|Y=1)P(Y=1)}{P(x|Y=2)P(Y=2)}$$

which decides class 1 if $d(x) > 0$ and class 2 otherwise.

Figure 4.4 illustrates the contours for the the 2 class densities, as well as the level contours of the log odds. When the log odds are negative class 2 is decided, otherwise, class 1 is. Note that where the 2 classes overlap, the 2nd class' density dominates the first.

Figure 4.5: The marginal class densities (dotted), and the Naive Bayes fitted logspline densities (solid).

Performing the exact multidimensional integration using software in Splus adapted from [2] on a Silicon Graphics machine with an R10000 CPU (rev 2.5) and R10010 FPU (rev 0.0) took 5593.6 secs or over 1.5 hours. Evaluating the probability of error integral over a grid was considerably faster. Table 4.1 demonstrates how the P(error) estimates depend on the grid resolution. A grid resolution of 300 gives a reasonably good approximation (to 0.1%)

$$P(\text{error}) = \int_{d(x)=1} P(x|y=2)P(y=2)dx + \int_{d(x)=2} P(x|y=1)P(y=1)dx$$

Given the marginal densities, we can get a bound on how well the Naive Bayes procedure will do in this case. The Bayes error was computed by sampling on a finite grid and approximating the integrals in the probability of error formula with finite sums. The probability of error for the Naive Bayes classifier was also computed using a finite grid approximation. Table 4.1 shows that when perfect knowledge of the class densities is available the Naive Bayes classifier suffers XX% more misclassifications as compared to the GAM classifier. We also compute the probability of error using a classifier (called Marginal) which assumes that the multivariate density is a product of univariate densities, ie independence among the covariates. This is the same assumption that Naive Bayes makes, except here Marginal doesn't impose a logspline density on the marginals. Figure 4.5 shows that with the fixed knots, the logspline estimate of the density is biased. We expect this will handicap the Naive Bayes procedure, and indeed the third column in table 4.1 indicates that were it not for the logspline density constraint, Naive Bayes would have done bettery.

Grid size	Error % Naive Bayes	Error % GAM	Error % Marginal
25	8.874	6.791	8.832
50	8.964	6.914	7.841
100	9.201	7.050	7.712
200	9.247	7.120	7.662
300	9.316	7.144	7.662
400	9.321	7.156	7.670
500	9.330	7.163	7.671
EXACT	8.970	7.188	
Abs Err in Exact	0.536	0.167	

Table 4.1: Exact probability of errors for three classification procedures as a function of grid resolution. The area of a grid element is $[-8, 8]^2/N^2$ where N is given in the first column above.

In practice of course, the exact class densities are unavailable and we must estimate them using the supplied training observations. Naive Bayes is biased for this choice of class densities, ie the class densities are not a product of the marginals, and the marginals themselves are not actual logspline densities (with basis fixed at the knots (-10,-5,0,5,10)).

Naive Error	Gam Error	Naive KL	GAM KL
9.83	14.44	69.71	162.83
10.93	7.50	Inf	858.86
9.86	9.63	50.97	561.16
9.38	12.97	36.46	Inf
11.00	10.80	67.61	436.37
10.50	12.65	29.65	Inf
12.65	9.38	Inf	Inf
9.12	17.15	24.45	Inf
13.85	14.11	79.95	133.26
9.63	8.99	38.13	19.13
11.35	8.30	72.24	20.67
9.20	16.95	32.68	440.83
13.59	12.94	72.10	501.23
9.24	9.81	50.86	75.98
9.25	7.38	21.32	356.38
9.66	9.07	53.03	Inf
14.46	9.79	111.95	20.16
10.64	12.40	60.31	Inf
9.48	9.46	27.58	39.10
14.60	12.46	81.69	542.55
18.59	14.87	Inf	243.68
13.71	10.93	Inf	851.69
9.46	8.69	46.76	19.19
8.63	15.46	14.27	Inf
9.12	8.15	24.55	8.08
11.1	11.4	Average	
2.44	2.92	SD	

Table 4.2: Error rates for different training sets of size 25, and KL distance.

4.3 Performance using small training sets

4.3.1 Acceptance-Rejection

Given the class densities, simulated training sets are generated using the acceptance-rejection method: First, a uniform vector X is generated in the space where the class densities have most of their support. Then a uniform variate U taking values in $[0, 1]$ is generated. If $U < f(x)/T$ where $T = \max_x f(x)$ then $Z = X$ is accepted as an observation, otherwise it is rejected and we repeat the procedure with starting with a new X.

Theorem: The rejection-acceptance generates variates Z with density $f(z)$, where WLOG the support of f(z) is on [0,a]. Proof:

$$\begin{aligned}
P(Z < z) &= P(X < z | U < \frac{f(X)}{T}) \\
&= \frac{P(X < z, U < \frac{f(X)}{T})}{P(U < \frac{f(X)}{T})} \\
&= \frac{\int_0^z dx \frac{1}{a} \int_0^{\frac{f(X)}{T}} du}{\int_0^a dx \frac{1}{a} \int_0^{\frac{f(X)}{T}} du} \\
&= \frac{\int_0^z dx \frac{1}{a} \frac{f(X)}{T}}{\int_0^a dx \frac{1}{a} \frac{f(X)}{T}} \\
&= \frac{\frac{1}{aT} F(z)}{\frac{1}{aT}} = F(z).
\end{aligned}$$

Chapter 5

O'Neill

5.0.2 Asymptotic Efficiency of Logsplines

For convenience, we restrict attention to the 1D, 2 class case. The class densities are modeled as logspline densities

$$\begin{aligned} p(x|y=k) &= \exp(\gamma_k(x;\beta_k) - \log \int_L^U \exp \gamma_k(x;\beta_k) \\ \gamma_k(x;\beta_k) &= a_{k,0} + a_{k,1} + \sum_m^M \beta_{k,m} B_{k,m} \\ B_{k,m} &= (x-x_m)_+^3 - \frac{x_M - x_m}{x_M - x_{M-1}}(x-x_{M-1})_+^3 + \frac{x_{M-1} - x_m}{x_M - x_{M-1}}(x-x_M)_+^3 \quad m = 1,\ldots,M-2. \end{aligned}$$

over $[L,U]$. The class densities are members of the exponential family with natural parameters β

$$p(x|y=k) = \exp \sum_m \beta_{k,m} B_{k,m} - A(\beta).$$

The $B_{k,m}$ are a natural cubic spline basis with knots x_m.

ONeill [24] derives the asymptotic distribution of the error rate of a classification procedure. Let

$$g(x;\alpha) = \log \frac{\pi_1 p(x|y=1)}{\pi_0 p(x|y=0)}$$

be the discriminant with error rate

$$ER(\alpha) = \pi_0 \int_{g(x,\alpha)>0} p(x|y=0) + \pi_1 \int_{g(x,\alpha)0} p(x|y=1)$$

for a particular choice of the parameters α.

Theorem 1 (O'Neill) *Under suitable regularity conditions (satisfied by exponential families), suppose there is an estimator $\hat{\alpha}$ such that*

$$\sqrt{n}(\hat{\alpha} - \alpha) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \Sigma_\alpha)$$

Then

$$n\{ER(\hat{\alpha}) - ER(\alpha)\} \xrightarrow{\mathcal{L}} z'Wz$$

where

$$z \sim \mathcal{N}(0, \Sigma_\alpha)$$

and

$$W = \frac{1}{4} \int_{g(x,\alpha)=0} |\nabla_x g(x;\alpha)|^{-1} \nabla_\alpha g(x;\alpha) \nabla'_\alpha g(x;\alpha) f(x) d\lambda(x)$$

where $d\lambda(x)$ is Lebesque measure on $\{g(x,\alpha) = 0\}$, and $|\nabla_x|^2 = \sum_j^p \left(\frac{\partial}{\partial x_j}\right)^2$

Asymptotically, the distribution of the MLE $\hat{\beta}$ has a normal distribution [3][ch 9.2]

$$\sqrt{n}(\hat{\beta} - \beta) \xrightarrow{\mathcal{L}} \mathcal{N}(0, i(\beta)^{-1})$$

where $i(\beta) = E[\frac{\partial \log f}{\partial \beta}]^2$ is the Fisher information matrix. For the exponential family case

$$i(\beta)_{j,k} = \frac{\partial^2 A}{\partial \beta_j \partial \beta_k}.$$

For the logspline case (ignoring the class label for convenience)

$$i(\beta)_{j,k} = \frac{\int \exp \gamma(x;\beta) dx \int B_j(x) B_k(x) \exp \gamma(x;\beta) dx - \int B_j(x) \exp \gamma(x;\beta) dx \int B_k(x) \exp \gamma(x;\beta) dx}{[\int \exp \gamma(x;\beta) dx]^2}$$

Now α is a function of β, and in order to apply O'Neill's result **??** we need to obtain the covariance matrix for α. We consider two cases: when the knots for the two classes are distinct, and when they are the same.

When the knots are distinct, the discriminant is

$$g(x;\alpha) = \log \frac{\pi_1 p(x|y=1)}{\pi_0 p(x|y=0)}$$

$$
\begin{aligned}
&= \gamma_1(x;\beta_1) - \gamma_0(x;\beta_0) - \log \int \exp \gamma_1(x;\beta_1)dx + \log \int \exp \gamma_0(x;\beta_0)dx \\
&= \alpha_0 + \alpha_1 x + \sum_{m=2}^{M} \alpha_{1,m} B_{1,m} - \sum_{m=2}^{M} \alpha_{0,m} B_{0,m} - \alpha_{2m+1}
\end{aligned}
$$

where $\alpha = (\beta_{1,1} - \beta_{0,1}, \beta_{1,2} - \beta_{0,2}, \beta_{0,3}^M, \beta_{1,3}^M, \log \int \exp \gamma_1(x;\beta_1)dx + \log \int \exp \gamma_0(x;\beta_0)dx)$ and $\beta = (\beta_0, \beta_1)$.

We will need the matrix

$$
\frac{\partial \alpha}{\partial \beta} = M = \begin{pmatrix}
1 & 0 & 0 & \cdots & 0 & -1 & 0 & 0 & \cdots & 0 \\
0 & 1 & 0 & \cdots & 0 & 0 & -1 & 0 & \cdots & 0 \\
0 & 0 & 1 & 0 & \cdots & & & & & 0 \\
0 & 0 & 0 & 1 & 0 & \cdots & & & & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
C_{0,1} & \cdots & & & C_{0,M} & -C_{1,1} & \cdots & & & -C_{1,M}
\end{pmatrix}
$$

where

$$
C_{k,m} = \frac{\int B_{k,m}(x) \exp \gamma_k(x;\beta_k)dx}{\int \exp \gamma_k(x;\beta_k)dx}.
$$

The distribution of $\hat{\alpha}$ is then

$$
\sqrt{n}(\hat{\alpha} - \alpha) \xrightarrow{\mathcal{L}} \mathcal{N}(0, Mi(\beta)^{-1}M')
$$

For the logistic regression estimate $\overline{\alpha}$ obtained by maximizing the conditional log likelihood

$$
\sum_i \sum_k z_{i,k} \log p(y_i = k|x_i) = \sum_i g(x_i;\alpha)y_i - \log(1 + \exp g(x_i;\alpha)
$$

the asymptotic distribution is given by [24][Theorem 2]

$$
\sqrt{n}(\overline{\alpha} - \alpha) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \Sigma_L)
$$

where

$$
\Sigma_L^{-1} = \int \frac{\exp g(x;\alpha)}{(1 + \exp g(x;\alpha))^2} \nabla_\alpha g(x;\alpha) \nabla'_\alpha g(x;\alpha) p(x)dx.
$$

O'Neill defines the asymptotic error rate (AER) of a procedure with discriminant parameters a as

$$AER(a) = Ez'Wz = \mathrm{Tr}W\Sigma_a$$

and the efficiency of two procedures the ratio

$$\mathrm{eff} = \frac{AER(a_1)}{AER(a_2)} = \frac{\mathrm{Tr}W\Sigma_1}{\mathrm{Tr}W\Sigma_2}.$$

We will now compare the efficiency of informative to discriminative training for the logspline class density case. This will require evaluating the matrices $W, \Sigma_{\hat{\alpha}}, \Sigma_{\overline{\alpha}}$.

5.0.3 Example

Figure 5.1: Class densities for logspline example.

We choose as our first example the densities depicted in figure 5.1. The knots for the classes are indicated in the figure. The coefficients and knots for the two densities are

$$\begin{aligned}
\beta_0 &= (79.4, 7.5; 0.7, -2.2, -13.8, 45.4, -58.4, 57.1, -61.3, 65.1, -61.3, 33.9, 16.0, -27.8, 3.0, 9.3, \\
&\quad -8.6, 3.0, -0.2, 0.2, -0.1) \\
\beta_1 &= (7.8, 6.2; -0.1, 0.2, -0.3, 3.8, -10.7, 11.2, 4.6, -37.2, 28.2, 27.5, -60.1, 65.9, \\
&\quad -62.7, 58.0, -53.7, 32.0, 1.4, -9.3, 1.4) \\
x_{0,n} &= (-13.0, -12.0, -11.5, -11.0, -10.5, -10.0, -9.5, -9.0, -8.5,
\end{aligned}$$

$$
\begin{aligned}
& -8.0, -7.5, -7.0, -6.0, -5.0, -4.0, -3.0, -1.0, 2.0, 4.5) \\
x_{1,n} \;=\; & (-4.5, -2.0, 1.0, 3.0, 4.0, 5.0, 6.0, 7.0, \\
& 7.5, 8.0, 8.5, 9.0, 9.5, 10.0, 10.5, 11.0, 11.5, 12.0, 13.0)
\end{aligned}
$$

The first two coefficients are the global constant and linear term, the rest multiply $(x-x_m)_+^3$. In all calculations, the unconstrained NCS basis is used.

Chapter 6

Separate Basis Procedure

6.1 POLYCLASS with separate basis

The standard POLYCLASS framework models the discriminant

$$\gamma(k|x) = \log \frac{P(y = k|x)}{P(y = K|x)}.$$

Note that the choice of denominator is arbitrary, but fixed for all classes, so that for class K, $\gamma(K|x) \equiv 0$. It is necessary to do this because otherwise the model is overparametrized since $\sum_k P(y = k|x) = 1$.

Summing over all classes

$$1 = \sum_k P(y = k|x) = P(y = K|x) \sum_k \exp \gamma(k|x)$$

gives

$$P(y = k|x) = \frac{\exp \gamma(k|x)}{\sum_j \exp \gamma(j|x)}.$$

In [18], the discriminant is assumed to be a linear combination of natural cubic splines $B(x)$ an m–vector, common to all classes

$$\gamma(k|x) = \beta_k^T B(x).$$

We propose an extension of POLYCLASS which admits a separate basis for each class

$$\gamma(k|x) = \beta_k^T B_k(x) = \sum_m^{m_k} \beta_{k,m} B_{k,m}(x).$$

6.1.1 Example

Before continuing, we give an example which motivates the desirability of a separate basis in each class, and also touches on the problems with arbitrarily setting the discriminant's denominator to the K^{th} class.

Suppose the class densities are given by

$$P(x|y=k) = \begin{cases} \mathcal{N}(x;\mu_k,\Sigma) & : \quad k \neq K \\ \text{Ugly}(x) & : \quad k = K \end{cases}$$

Assuming equal priors, the discriminants with respect to class K

$$\gamma_K(k|x) = \log \frac{P(y=k|x)}{P(y=K|x)} = \log \frac{P(x|y=k)}{P(x|y=K)}$$

will all be complicated functions because $P(x|y=K)$ is a complicated function. A procedure with a common basis B (which includes a constant and linear term) for each class will demand a large number of basis functions to model the complicated discriminants. The coefficients β_k will all be significantly non-zero.

What happens if we consider the discriminant with respect to class 1, which has a gaussian distribution? Recall that the discriminant between two gaussians with the same covariance is linear, so that

$$\gamma_1(k|x) = \begin{cases} 0 & : \quad k = 1 \\ (1, \beta_{k,1}, 0, \ldots, 0)^T x & : \quad k \neq K, 1 \\ \text{Ugly}_1(x) & : \quad k = K \end{cases}$$

Most of the coefficients are zero, ie we have much fewer parameters to estimate if we know ahead of time to ignore certain bases.

Of course, for a fixed, common basis, there is no difference which parametrization is used (ie which class has $\gamma(k|x) = 0$) since the log likelihood depends only on $p(y=k|x) =$

$\exp \gamma(k|x) / \sum_j^K \exp \gamma(j|x)$ and

$$\begin{aligned} \gamma_j(k|x) &= \log \frac{P(y=k|x)}{P(y=j|x)} \\ &= \gamma_K(k|x) + \log \frac{P(y=K|x)}{P(y=j|x)}. \end{aligned}$$

6.1.2 Choice of constraint

We saw above that due to the overparametrization in POLYCLASS, one of the $\gamma(k|x)$ is set to zero, and that the estimation is invariant to the particular choice of k when the basis B is common to all classes.

However, when we consider a separate basis $B_k(x)$ for each class, the choice of which $\gamma(k|x) = 0$ can determine whether the discriminants are simple or complicated. This is especially relevant in a procedure which tries to find a basis for each class.

Is there a way to build in the constraint that $\sum_k P(y=k|x) = 1$ without having to select a class? We first consider a simple case that demonstrates the principles involved.

Consider the case where $B_k(x) = 1, m_k = 1 \ \ \forall k$, ie the discriminants $\gamma(k|x) = \beta_k$ are all constants. The class posterior

$$P(y=k|x) = \frac{e^{\beta_k}}{\sum_j e^{\beta_j}}$$

is invariant to translations of $\beta \to \beta + c$

$$\frac{e^{\beta_k + c}}{\sum_j e^{\beta_j + c}} = \frac{e^c e^{\beta_k}}{e^c \sum_j e^{\beta_j}}.$$

The standard way of dealing with this is to choose $c = -\beta_K$ and then estimate the remaining β_k. But in fact, there are many other ways to constrain this overparametrization

- Standard: $c = -\beta_K$. Problem: assymetric - requires choice of class K.
- linear constraint: $c = -\sum_j \beta_j$. Advantage: symmetric - does not treat any particular class specially.
- quadratic constraint: choose c such that $||\beta + c|| = 1$, ie the resulting coefficient vector

has unit norm. This gives rise to a quadratic equation in c with solutions

$$c = \frac{-\sum_j^K \beta_j \pm \sqrt{(\sum_j \beta_j)^2 - 4K(\sum_j \beta_j^2 - 1)}}{2K}$$

We like the second approach the best: it is symmetric, and linear. We propose a method which will involve maximization of the conditional log likelihood subject to linear constraints.

In general, the bases $B_k(x)$ will consist of several terms beyond the constant. For every basis that is common to all classes, we constrain the coefficients across classes for that basis to sum to zero. That is, suppose the m^{th} basis $B_{k,m}(x) = h(x) \forall k$, then we constrain $\sum_k \beta_{k,m} = 0$.

[Dan: insert details from Jun 5 notes about orthogonality of the bases and WLOG]

This gives rise to a nonlinear program with linear constraints: Maximize $L(\beta) = \sum_i \log P(y_i|x_i)$ subject to $\sum_k \beta_{k,m} = 0$ for all m for which $B_{k,m}(x)$ is common to all classes. Implement using firt order Kuhn-Tucker conditions ie project current descent direction onto constraint surface. [Discuss algorithmic options here: penalty barrier methods etc...]

6.1.3 Basis Selection

- Basis Pursuit flavour: considers all dictionary elements simultaneously (look at problem as K function estimation problems of p(y=k—x)). so use a very large basis space and perhaps as part of the final iterations throw away basis that aren't contributing much to the likelihood
- stepwise regression using rao stat

6.1.4 Polyclass Model

Polyclass: adaptive classification procedure (Kooperberg, Stone et al). Borrows ideas from MARS.

Model class logits as linear in derived basis $B(x)$

$$\gamma(k|x) = \sum_m \beta_{k,m} B_m(x) = \beta_k^T B(x)$$

Same $B(x)$ for all classes.

$B(x)$: tensor product of linear splines, e.g. $B(x) = x_1$ or $B(x) = x_3x_7$ or $B(x) = x_2(x_4 - t_1)_+$.

Knots induce partition on feature space, logits are piecewise polynomials over subregions. *Same partition for all class logits.*

6.1.5 Polyclass Details

1. Parameters obtained by maximizing multinomial likelihood - a convex function: $l(\beta) = \sum_i \log P(y_i|x_i;\beta)$.

2. Stepwise *hierarchical* addition

   - up to 2 term interactions
   - Rao Stat: $\nabla l(\beta_1)^T I^{-1}(\beta_1)\nabla l(\beta_1)$, which is small if $\beta_1 = (\beta_0, 0)$ lies in subspace. Add term with largest Rao stat.
   - Optimize on knot location.

3 Stepwise deletion using Wald stat: $(A\beta)^T[AI^{-1}(\beta)A^T]^{-1}(A\beta)$, tests whether $\beta \in \mathcal{B}_0$, where $\mathcal{B}_0$ is subspace for which $A\beta = 0$.

4 Estimate a sequence of models, each with different number of parameters ν. Choose best model by

   - AIC (a=2) (or BIC ($a = \log n$)): $AIC(a) = -2\hat{l} + a\nu$.
   - Independent test set
   - Cross validation

6.1.6 Model Parameters

Matrix of coefficients $\beta_{k,m}$: k-th class, m-th term.

Model is overspecified since

$$\sum_j P(y = j|x) = 1$$

Invariant to translations of the i-th column of β $\beta_{k,i} \to \beta_{k,i} + c$.

$$\frac{\exp\left[\sum_{m\neq i}\beta_{k,m}B_m(x)+(\beta_{k,i}+c)B_i(x)\right]}{\sum_k \exp\left[\sum_{m\neq i}\beta_{k,m}B_m(x)+(\beta_{k,i}+c)B_i(x)\right]} = \frac{\exp(cB_i(x))\exp\left[\sum_m \beta_{k,m}B_m(x)\right]}{\exp(cB_i(x))\sum_k \exp\left[\sum_m \beta_{k,m}B_m(x)\right]}$$

Constrain β. POLYCLASS fixes $\beta_{K,m} = 0 \ \ \forall\ m$ so that $\gamma(K|X) = 0$.

Makes estimation easy: just ignore class K parameters, and optimize unconstrained likelihood.

6.1.7 Constrained Parameters

Constraining $\gamma(K|X) = 0$, means we are modeling the logits using class K as a "base" class:

$$\begin{aligned}
\exp\gamma_k(x) &= P(y=k|x)\sum_j \exp\gamma_j(x) \\
\Rightarrow \gamma_k(x) &= \log P(y=k|x) - \log\frac{1}{\sum_j \exp\gamma_j(x)} \\
&= \log\frac{P(y=k|x)}{P(y=K|x)}
\end{aligned}$$

When $B(x)$ is common to all classes, the choice of base class is arbitrary. Coefficients transform, although logits may look dramatically different.

$$\begin{aligned}
\theta_k^T B(x) &= \log\frac{P(y=k|x)}{P(y=j|x)} \\
&= \log\frac{P(y=k|x)}{P(y=K|x)} + \log\frac{P(y=K|x)}{P(y=j|x)} \\
&= \beta_k^T B(x) - \beta_j^T B(x)
\end{aligned}$$

So $\theta_k = \beta_k - \beta_j$.

6.1.8 Effect of base class on logit

Using class 6, a gaussian, as base class:

Using class 15, an ugly density, as base class:

6.2 Separate Basis Procedure (SBP)

Idea: Combine Informative and Discriminative approaches.

Let each class suggest its basis, fit using Discriminative (conditional likelihood) criterion.

Achieved by allowing separate basis for each class:

$$\gamma(k|x) = \sum_m \beta_{k,m} B_{k,m}(x) = \beta_k^T B_k(x)$$

Choose knots separately for each class based on class density (to do).

1. Validated idea by handpicking a basis. Implemented in S. On special case, improved on Polyclass.

   Issue: *Model selection:* would SBP be able to find the right basis for each class?

2. Implemented SBP in C. Solves exactly at candidates. Knots are prespecified.

6.2.1 2D Example: Ugly

Goal: come up with an example for which SBP will do much better than Polyclass, and so *validate SB*.

Many classes $\Rightarrow$ force Polyclass to pay heavy price for each term it adds.

Make most class logits simple (eg linear), but one that is really complicated, ie *Ugly*:

$$P(x|y=k) = \begin{cases} \mathcal{N}(x; \mu_k, \Sigma) & : \quad k \neq K \\ \text{Ugly}(x) & : \quad k = K \end{cases}$$

Took Ugly(x) to be a mixture of Gaussians.

6.2.2 Logits

When $B(x)$ common to all classes, choice of base class is unimportant.

But if each class has its own basis $B_k(x)$ it can make a huge difference.

Ugly example: Using a gaussian class as a base, (K-1) linear logits, one "ugly" logit.

$$\begin{aligned} B_k(x) &= [1, x_1, x_2] \text{ for } k \neq K \\ B_K(x) &= [1, x_1, x_2, x_1x_2, x_1 - t_1, \ldots, x_2 - s_1, \ldots] \end{aligned}$$

Using the Ugly class as a base, all logits are "ugly" - need many basis terms to represent each one.

Adaptive model selection with AIC (penalizes number of terms) might go the "wrong" route and stop prematurely.

6.2.3 SB Model Evolution

1D example shows how SB model evolves.

There are 4 classes, 10 training obs/class.

Start with a constant model, then:

Class	Term	Lik	Numterms
3	Lin	32.962	5
0	Lin	13.887	6
2	Lin	13.886	7
2	Knot(-0.25)	11.921	8

Out of the box, Polyclass gets a linear model for all classes, with Perror = 0.358.

SB gets Linear model for classes 0,2,3,(1) plus a knot term in class 2, (total of 3+3+1=7 terms). Perror = 0.2345.

Lowering penalty term for Polyclass ⇒ knot term (total of 9 terms). Perror = 0.2354. Each term Polyclass adds, costs it (K-1) parameters.

Polyclass reluctant to add terms.

Estimated class posteriors:

6.2.4 Comparisons

Figure 6.1: Using logspline density estimation to determine candidate knot locations..

6.3 SBP using Informative approach

Figure 6.1 shows the knots that are selected by the logspline density estimation procedure [19]. These are based on estimating the univariate density separately in each dimension, and separately for each class. The knots obtained are then used as candidates in the SBP.

6.3.1 Description of SBP Algorithm

Initially, the SBP starts with a constant model in each class, i.e. $\gamma_k(x) = \beta_{k,0} \quad \forall k$. Each class has a list of candidate terms which may be added to that class' discriminant. Initially, the candidate list consists of pure linear terms in all dimensions

$$\text{Candidates}(k) = \{x_1, x_2, \ldots, x_p\}.$$

Now begin a set of iterations to build the model by adding terms, and then fitting the current iteration's model. Each candidate across all classes is scored, and the one that seems the most promising is added to the model. There are several ways to gauge a candidate's promise:

1. Exact fit: Fit the newest model with the latest candidate exactly, and then compare

likelihoods and choose the candidate that gave the best likelihood.

2. Single Newton-Raphson: Rather than fit the likelihood exactly, take a single descent step starting at the last best fit model, and evaluate the likelihood after this single step. Choose the candidate that gave the best likelihood.

3. Rao statistic: Approximate the likelihood by its quadratic Taylor expansion and choose the candidate that causes the greatest change in the quadratic approximation to the likelihood.

These are listed in order of decreasing computational complexity. The exact fit method is very slow. It appears from empirical evidence that doing a rough model selection is good enough.

All the above are greedy strategies in the sense that they optimize the current candidate selection rather than the entire candidate sequence. One heuristic for overcoming this drawback is to iterate stepwise addition with stepwise deletion of model terms.

Once the best candidate is selected, the model is fit exactly via a gradient descent procedure (a quasi-Newton method was much faster and stabler than a conjugate gradient method).

This procedure produces a nested set of models along with their likelihoods which are monotonically decreasing.

6.3.2 Likelihood Gradient and Hessian

6.3.3 The Rao statistic

6.3.4 Local NR vs full NR

Each new candidate term is evaluated by adding it to the current, fitted model with a coefficient equal to 0. The "old" coefficient vector is then $\beta_{\text{old}} = (\beta_{\text{fit}}, 0)$. A single Newton-Raphson (NR) step is taken, and the likelihood evaluated at this new point

$$\beta_{\text{new}} = \beta_{\text{old}} - H(\beta_{\text{old}})^{-1} g(\beta_{\text{old}})$$

where H is the negative of the hessian matrix.

Since β_{fit} maximized the old model the gradient at that point should be near zero $g(\beta_{\text{old}}) = 0$. It may not be exactly zero because the optimization routine stops when the

norm of the gradient is below some small threshold, so it is only "numerically" zero. For the purposes of this discussion, we will assume it is exactly zero.

We now consider two methods of evaluating candidates. The first, will be called the local method. Only that class whose candidate is being evaluated undergoes a NR step. Coefficients belonging to other classes stay fixed at their previously fitted values. Below we explore the relationship between density estimation and this local update.

The second method, the "full" NR, is the common one in which all coefficients are updated by the NR step. What is the connection between the local and the full NR updates, especially with regard to the Rao statistic

$$g(\beta_{\text{old}})^T H(\beta_{\text{old}})^{-1} g(\beta_{\text{old}}).$$

Since $g(\beta_{\text{fit}}) = 0$, the Rao statistic reduces to

$$g(\beta_{\text{old}})^T H(\beta_{\text{old}})^{-1} g(\beta_{\text{old}}) = g_M^2 H_{M,M}^{-1},$$

so it is sufficient to examine the structure of the inverse of the hessians in the local vs the full cases.

The full hessian can be expressed in block form

$$H_{MxM} = \begin{pmatrix} H_0 & U \\ U^T & H_K \end{pmatrix}$$

where H_K is the (m_K, m_K) hessian for the local NR, which we assume without loss of generality is with respect to class K. We can derive the inverse of H by determining the sequence of elementary operations that reduce H to I

$$E_m \cdots E_2 E_1 H = I$$

and then identifying $H^{-1} = E_m \cdots E_1$. Letting $B = H_O^{-1}$, and $C = (-U^T BU + H_K)^{-1}$,

$$H^{-1} = \begin{pmatrix} I & -Bu \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & C \end{pmatrix} \begin{pmatrix} I & 0 \\ -U^T & I \end{pmatrix} \begin{pmatrix} B & 0 \\ 0 & I \end{pmatrix} \tag{6.1}$$

$$= \begin{pmatrix} B + BUCU^T B & -BUC \\ -CU^T B & C \end{pmatrix} \tag{6.2}$$

The Rao statistic involves only $H_{M,M}$. For the full case this is

$$H_{M,M} = C_{m_K,m_K} = [-U^T BU + H_K]^{-1}_{m_K,m_K}$$

while the local case involves only $[H_K]^{-1}_{m_K,m_K}$.

Additional insight can be gained by considering the case when there is only a single term, namely the new one, in class K so that $m_K = 1$. Since the hessian is negative definite, its negative H is positive definite (and so are the submatrices H_0 and H_K)

$$x^T Hx > 0$$

for all x. Since H_0 is positive definite, so is $B = H_0^{-1}$ and C. Then

$$0 < -U^T BU + H_K < H_K \Rightarrow (-U^T BU + H_K)^{-1} > H_K^{-1}$$

so that the Rao statistic for the full case will be larger than for the local case.

Figure 6.2: Comparison of Rao statistics using full NR vs local NR.

Figure 6.2 shows the Rao statistics computed using the local NR and the full NR. The statistics were collected during the model building for the "ugly example" described below. Note the close agreement between the full and the local methods.

6.3.5 Matrix Inverse when submatrix inverse known

Suppose

$$A = \begin{pmatrix} B & v \\ u & x \end{pmatrix} \tag{6.3}$$

where $B(n-1,n-1), v(n-1,1), u(1,n-1), x(1,1)$, and $B^{-1} = C$ which is known. We can express A^{-1} in terms of C, u, v, x by determining the sequence of row operations that reduce A to I

$$E_m \cdots E_2 E_1 A = I$$

and then identifying $A^{-1} = E_m \cdots E_1$.

$$\begin{aligned}
\begin{pmatrix} C & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} B & v \\ u & x \end{pmatrix} &= \begin{pmatrix} I & Cv \\ u & x \end{pmatrix} \\
\begin{pmatrix} I & 0 \\ -u & 1 \end{pmatrix} \begin{pmatrix} I & Cv \\ u & x \end{pmatrix} &= \begin{pmatrix} I & Cv \\ 0 & -uCv + x \end{pmatrix} \\
\begin{pmatrix} I & -Cv/(-uCv+x) \\ 0 & 1/(-uCv+x) \end{pmatrix} \begin{pmatrix} I & Cv \\ 0 & -uCv+x \end{pmatrix} &= \begin{pmatrix} I & 0 \\ 0 & 1 \end{pmatrix} \\
\Rightarrow A^{-1} &= \begin{pmatrix} C + \frac{CvuC}{-uCv+x} & \frac{-Cv}{-uCv+x} \\ \frac{-uC}{-uCv+x} & \frac{1}{-uCv+x} \end{pmatrix}
\end{aligned}$$

6.4 $K-1$ Gaussians + Ugly Example

A simulation example was chosen with the goal of exemplifying a case in which the separate basis procedure would have a significant advantage over Polyclass.

First, there are many classes ($K = 15$). For every additional basis term Polyclass wants to add to the model, $K - 1 = 14$ coefficients must be estimated. Polyclass will pay a heavy price for adding basis terms and so will be reluctant to do so.

The training data consisted of 15 observations from the 14 bivariate, spherical gaussians, along with 150 observations from the ugly class (a mixture of spherical gaussians). The variance of the gaussians was $\sigma^2 = 0.005$.

```
The class centroids:

       1     2    3     4     5     6      7
X1  0.69 -0.57 0.06  0.27  1.11 -0.15  -0.99
X2 -0.15 -0.15 0.06 -0.15 -0.15 -0.15  -0.15

       8    9   10    11   12    13    14
X1  0.27 0.06 0.48 -0.15 0.27 -0.57 -0.36
X2  0.69 0.27 0.06  0.27 0.48 -0.99  0.06

        U     U     U     U     U     U
X1  -1.20 -0.78 -0.36  0.06  0.48  0.90
X2  -0.15 -0.15 -0.15 -0.15 -0.15 -0.15
```

Figure 6.3: Ugly example: Training data and Bayes classification regions.

Figure 6.8 demonstrates that "less is more" when it comes to adding terms to the model. The additional terms in the model are overfitting the training data - the misclassification rate actually begins to increase. The minimum of the BIC criterion, which penalizes the

Figure 6.4: Polyclass and linear model.

Figure 6.5: Hand picked model, and knots used. Top left: common knots used when classes share the same knots. Otherwise, knots are determined using logspline separately in each class for each dimension.

likelihood by the number of terms in the model,

$$\text{BIC} = -\text{Likelihood} + \text{Numterms}\log(n)$$

determines the "best" model to report.

All the results described here are for basis functions involving only a single linear spline term. Second order terms involving a tensor product of splines did not help much and entered the models only at advanced stages where they were over fitting the data. The best model as selected by the BIC criterion was at a stage before any second order terms entered the models.

Table 6.1 summarizes the classification results and the training time required for various

Figure 6.6: Same knots in each class, candidates shared. Left: Single NR step to evaluate candidate. Right: Exact fit for each candidate.

Figure 6.7: Left: Same knots in each class, candidates NOT shared. Right: Each class has its own knots, candidates NOT shared.

procedures applied to the ugly example. Note the speed up obtained, by a factor of 75, from evaluating candidates with a single Newton-Raphson step versus doing an exact fit for each candidate, at negligible cost in terms of misclassifications. The best classification results (Build 2) were obtained for the case when each class had its own candidate list and knots, determined by logspline applied to the margins of each class density separately. Because fewer candidates are generated, this was also the fastest procedure to fit.

All the SBP builds were carried out for a maximum model containing 45 terms. Compared to Polyclass, the training times for the SBP builds took $O(K)$ longer to run.

Build 0:

$$\gamma_1(x) \quad = \quad -23.89 + 67.17x_1 - 87.59(x_1 - 0.63)_+$$

Figure 6.8: Left: BIC criterion: penalizing the likelihood by the number of terms in the model. Right: The corresponding model evolution.

$$
\begin{aligned}
\gamma_2(x) &= 13.01 - 105.65(x_1 + 0.47)_+ + 23.63(x_1 + 0.77)_+ - 67.97(x_2 + 0.11)_+ \\
\gamma_3(x) &= 9.10 - 30.23(x_2 - 0.02)_+ \\
\gamma_4(x) &= 10.51 - 91.08(x_2 + 0.17)_+ + 45.67(x_2 + 0.28)_+ \\
\gamma_5(x) &= -25.87 + 40.87x_1 \\
\gamma_6(x) &= 10.39 - 56.20(x_2 + 0.11)_+ - 121.50(x_1 + 0.07)_+ + 22.03(x_1 + 0.47)_+ \\
\gamma_7(x) &= 15.57 - 40.38(x_1 + 0.77)_+ - 63.04(x_2 + 0.11)_+ \\
\gamma_8(x) &= -3.89 + 305.38(x_2 - 0.52)_+ \\
\gamma_9(x) &= 4.53 \\
\gamma_{10}(x) &= -2.39 - 0.17(x_2 - 0.02)_+ + 31.39x_1 - 105.71(x_1 - 0.63)_+ \\
\gamma_{11}(x) &= 7.92 - 72.21(x_1 + 0.07)_+ \\
\gamma_{12}(x) &= -26.15 + 69.69x_2 + 42.07(x_1 - 0.08)_+ \\
\gamma_{13}(x) &= -9.05 - 41.28x_2 \\
\gamma_{14}(x) &= 10.47 - 55.13(x_1 + 0.30)_+ \\
\gamma_{15}(x) &= 16.50 - 2.25(x_2 + 0.17)_+ - 54.79(x_2 + 0.11)_+
\end{aligned}
$$

Description	Error Rate	Num terms	Training Time (secs)
Polyclass	25.07	56	21.0
Linear	31.86	30	4.9
Hand	19.98	45	10.1
Build 0	20.07	43	346.6
Build 0 Exact	20.26	40	25960.3
Build 1	21.53	40	299.5
Build 2	17.28	39	177.5

Table 6.1: Ugly example results

Build 0 Exact:

$$
\begin{aligned}
\gamma_1(x) &= -119.44 + 94.03(x_1 + 1.16)_+ \\
\gamma_2(x) &= 41.24 - 110.29(x_1 + 0.47)_+ - 0.07(x_1 + 1.16)_+ \\
\gamma_3(x) &= 6.16 - 126.71(x_2 - 0.02)_+ + 83.60(x_2 + 0.28)_+ \\
\gamma_4(x) &= 23.24 - 24.79(x_2 + 0.11)_+ + 26.13x_1 \\
\gamma_5(x) &= -50.39 + 137.18x_1 \\
\gamma_6(x) &= 33.64 - 104.73(x_2 - 0.02)_+ - 93.56(x_1 + 0.07)_+ \\
\gamma_7(x) &= 50.53 - 33.11(x_1 + 0.77)_+ - 19.94(x_1 + 1.16)_+ \\
\gamma_8(x) &= 15.79 + 385.78(x_2 - 0.52)_+ \\
\gamma_9(x) &= 24.96 \\
\gamma_{10}(x) &= 1.62 + 66.03x_1 + 21.38(x_2 + 0.28)_+ \\
\gamma_{11}(x) &= 30.28 - 101.82(x_1 + 0.07)_+ \\
\gamma_{12}(x) &= -9.58 + 80.07(x_2 - 0.02)_+ + 89.41(x_1 - 0.08)_+ \\
\gamma_{13}(x) &= 5.71 - 51.55x_2 \\
\gamma_{14}(x) &= 34.52 - 82.26(x_1 + 0.30)_+ \\
\gamma_{15}(x) &= 30.14 - 331.53(x_2 - 0.02)_+ + 47.73(x_1 - 0.63)_+ - 17.91x_1 + 15.83(x_1 - 0.41)_+ + 61.81(x_1 - 0.
\end{aligned}
$$

Build 1:

$$
\begin{aligned}
\gamma_1(x) &= -54.12 + 58.40x_1 - 76.79(x_1 - 0.63)_+ \\
\gamma_2(x) &= -8.24 + 18.79x_1 - 96.76(x_1 + 0.47)_+ \\
\gamma_3(x) &= -22.40 + 48.08x_2 \\
\gamma_4(x) &= -31.64 + 69.60x_1 - 95.91(x_1 - 0.22)_+ \\
\gamma_5(x) &= -58.75 + 37.70x_1 \\
\gamma_6(x) &= -22.33 \\
\gamma_7(x) &= 500.27 + 446.44x_1 - 454.12(x_1 + 1.16)_+ \\
\gamma_8(x) &= -76.25 + 176.71x_2 \\
\gamma_9(x) &= -28.43 + 82.80x_2 \\
\gamma_{10}(x) &= -25.69 + 14.76x_1 - 62.84(x_1 - 0.63)_+ + 38.77x_2 \\
\gamma_{11}(x) &= -26.39 + 93.40x_2 - 0.89x_1 - 90.84(x_1 + 0.07)_+ \\
\gamma_{12}(x) &= -42.14 + 120.26x_2 \\
\gamma_{13}(x) &= -38.25 - 30.87x_2 \\
\gamma_{14}(x) &= -14.03 + 76.05x_2 + 18.60x_1 - 91.97(x_1 + 0.30)_+ \\
\gamma_{15}(x) &= -19.61 + 1.83x_2 - 9.96(x_2 + 0.11)_+
\end{aligned}
$$

Build 2:

$$
\begin{aligned}
\gamma_1(x) &= -13.93 + 37.32x_1 - 62.04(x_1 - 0.68)_+ \\
\gamma_2(x) &= 36.73 + 43.25x_1 - 100.60(x_1 + 0.53)_+ \\
\gamma_3(x) &= 6.64 + 49.35x_2 \\
\gamma_4(x) &= -1.94 + 62.45x_1 - 103.06(x_1 - 0.25)_+ \\
\gamma_5(x) &= -29.79 + 37.56x_1 \\
\gamma_6(x) &= 6.44 \\
\gamma_7(x) &= 79.59 + 65.49x_1 - 123.08(x_1 + 1.00)_+ \\
\gamma_8(x) &= -52.76 + 186.27x_2
\end{aligned}
$$

$$
\begin{aligned}
\gamma_9(x) &= 0.22 + 86.49x_2 \\
\gamma_{10}(x) &= -3.19 + 32.70x_1 - 51.21(x_1 - 0.47)_+ + 49.28x_2 \\
\gamma_{11}(x) &= 6.03 + 102.22x_2 + 11.76x_1 - 99.96(x_1 + 0.12)_+ \\
\gamma_{12}(x) &= -14.32 + 115.67x_2 + 18.34x_1 \\
\gamma_{13}(x) &= -11.48 - 36.48x_2 \\
\gamma_{14}(x) &= 8.45 + 80.41x_2 + 2.21x_1 - 86.64(x_1 + 0.26)_+ \\
\gamma_{15}(x) &= 8.45 - 2.02x_2
\end{aligned}
$$

6.4.1 L example

The class densities are three bivariate spherical gaussians $f_i(x) = \mathcal{N}(\mu_i, \sigma^2 I)$, with $\sigma = 0.75$ and $\mu_1 = (-1, -1), \mu_2 = (1, -1), \mu_3 = (-1, 1)$. A training sample of 10 was drawn from each class. The class centroids form the letter "L", hence the name for this example.

Since the true underlying densities are known, it is easy to obtain the Bayes model with class 1 as the base class

$$
\begin{aligned}
\gamma_1(x) &= 0 \\
\gamma_2(x) &= \log \frac{f_2(x)}{f_1(x)} = 4\sigma^2 x_1 = 2.25x_1 \\
\gamma_3(x) &= \log \frac{f_3(x)}{f_1(x)} = 4\sigma^2 x_2 = 2.25x_2
\end{aligned}
$$

Note that at most only one dimension enters each of the discriminants. The resulting decision regions are simple to describe

$$
\begin{aligned}
x_1 < 0, x_2 < 0 &\Rightarrow \text{Class}_1 \\
x_1 > 0, x_1 > x_2 &\Rightarrow \text{Class}_2 \\
x_2 > 0, x_1 < x_2 &\Rightarrow \text{Class}_3
\end{aligned}
$$

and are show in figure 6.9.

Polyclass comes up with a linear model in all dimensions. This is to be expected, because a term for x_1 is required for $\gamma_2(x)$ and a term for x_2 is needed for $\gamma_3(x)$. Once a term enters, it must enter for all class discriminants.

Figure 6.9: L example: Training data and Bayes classification regions.

Figure 6.10: Polyclass and linear model.

SBP model

$$\begin{aligned} \gamma_1(x) &= 0.01 \\ \gamma_2(x) &= -0.17 + 2.64x_1 \\ \gamma_3(x) &= 0.97 + 5.30x_2 \end{aligned}$$

Figure 6.11: SBP model.

Polyclass model

$$\begin{aligned}\gamma_1(x) &= -2.25 - 0.89x_1 - 5.78x_2 \\ \gamma_2(x) &= -2.02 + 1.97x_1 - 5.57x_2 \\ \gamma_3(x) &= 0\end{aligned}$$

Linear Model coefficients:

	1	X1	X2
class 1:	-2.257736	-0.8937562	-5.813452
class 2:	-2.028876	1.9718518	-5.596449
class 3:	0.000000	0.0000000	0.000000

Table 6.2: Coefficients of linear model, fixing $\gamma_3(x) = 0$.

6.4.2 Vowel Data

From Tony Robinson:

The vowel data used in this experiment was collected by D. H. Deterding, who recorded examples of the eleven steady state vowels of English spoken by fifteen speakers for a speaker

normalisation study.

The speech signals were low pass filtered at 4.7kHz and then digitised to 12 bits with a 10kHz sampling rate. Twelfth order linear predictive analysis was carried out on six 512 sample Hamming windowed segments from the steady part of the vowel. The reflection coefficients were used to calculate 10 log area parameters, giving a 10 dimensional input space. For a general introduction to speech processing and an explanation of this technique see Rabiner and Schafer [RabinerSchafer78].

[RabinerSchafer78] L. R. Rabiner and R. W. Schafer, Englewood Cliffs, New Jersey, 1978, Prentice Hall, "Digital Processing of Speech Signals".

Each speaker thus yielded six frames of speech from eleven vowels. This gave 528 frames from the eight speakers used to train the models and 462 frames from the seven speakers used to test the models.

Figure 6.12: Vowel training data projected onto first 2 principal components, and plotted by class.

Figure 6.12 gives a sense of the high degree of overlap among the classes.

The model selected for the SBP was found by inspecting the BIC curve, which was very flat over a wide range (figure 6.13). SBP improves over Polyclass by 9% with fewer terms in the model.

Description	Error Rate	Num terms	Training Time (secs)
Nearest Neighbour	56	528	–
Linear	51.30	110	9.59
Polyclass	54.98	99	26.0
SBP	46.32	89	2771.0

Table 6.3: Vowel data results.

Polyclass confusion matrix

	1	2	3	4	5	6	7	8	9	10	11
1	28	0	0	0	0	0	0	0	0	5	0
2	14	12	0	0	0	0	1	0	0	8	0
3	0	2	11	1	0	0	3	0	1	0	0
4	0	9	16	21	0	5	0	0	2	0	5
5	0	0	0	0	33	21	8	0	0	0	6
6	0	0	8	20	3	10	0	0	0	0	2
7	0	2	1	0	6	0	29	18	5	0	15
8	0	0	0	0	0	0	0	24	17	6	0
9	0	5	0	0	0	0	0	0	13	0	9
10	0	6	0	0	0	0	0	0	4	23	1
11	0	6	6	0	0	6	1	0	0	0	4

SBP confusion matrix

	1	2	3	4	5	6	7	8	9	10	11
1	32	1	0	0	0	0	0	0	0	4	0
2	8	29	6	0	0	0	0	0	0	7	0
3	1	4	19	5	0	0	0	0	0	0	1
4	0	0	8	25	1	3	0	0	2	0	6
5	0	1	4	0	19	7	7	6	1	1	0
6	0	1	3	6	15	23	7	0	0	0	7
7	0	0	0	0	7	0	28	10	7	0	13
8	0	0	0	0	0	0	0	22	6	6	0
9	0	0	0	0	0	0	0	4	21	3	6
10	1	2	0	0	0	0	0	0	1	21	0
11	0	4	2	6	0	9	0	0	4	0	9

Linear model

	1	2	3	4	5	6	7	8	9	10	11
1	33	13	0	0	0	0	0	0	0	6	1
2	9	23	4	0	0	0	4	0	0	3	4
3	0	2	21	2	0	5	0	0	1	6	1
4	0	0	10	24	6	8	0	0	0	0	0
5	0	0	0	0	19	6	11	2	0	0	0
6	0	1	7	16	5	13	6	0	0	0	6
7	0	0	0	0	12	2	9	7	1	0	1
8	0	0	0	0	0	0	0	20	4	6	0
9	0	0	0	0	0	2	10	7	26	2	5
10	0	0	0	0	0	0	0	6	4	14	1
11	0	3	0	0	0	6	2	0	6	5	23

Table 6.4: Confusion matrices for vowel data using linear, Polyclass and SBP classifiers.

Figure 6.13: The BIC criterion function for the vowel data.

Chapter 7

Conclusions and Suggestions for Future Work

Suggetsion for future work.

Appendix A

Derivation of score equations

This is a test.

Bibliography

[1] T. W. Anderson. *AN INTRODUCTION TO MULTIVARIATE STATISTICAL ANALYSIS*. Wiley, 2 edition, 1984.

[2] Jarle Berntsen, Terje O. Espelid, and Alan Genz. Algorithm 698: Dcuhre: An adaptive multidimensional integration routine for a vector of integrals. *ACM Transactions of Mathematical Software*, 17(4):452–456, 1991.

[3] D.R. Cox and D.V. Hinkley. *Theoretical Statistics*. Chapman & Hall, 1974.

[4] Duda and Hart. *Pattern Recognition and Scene Analysis*. Wiley, 1992.

[5] Bradley Efron. The efficiency of logistic regression compared to normal discriminant analysis. *Journal of the American Statistical Association*, 70(352):892–898, December 1975.

[6] Charles J. Stone et al. Polynomial splines and their tensor products in extended linear modeling. *Annals of Statistics*, to appear.

[7] J. Friedman. On bias, variance, 0/1-loss and the curse of dimensionality. Technical report, Dept. of Statistics, Stanford University, April 1996.

[8] D. Geiger, D. Heckerman, and Christopher Meek. Asymptotic model selection for directed networks with hidden variables. Technical Report MSR-TR-96-07, Microsoft Research, May 1996.

[9] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Acadenic Publishers, Boston, 1992.

[10] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, 1981.

[11] V. P. Godambe. *Estimating Functions*. Oxford University Press, 1991.

[12] V.P. Godambe. An optimum porperty of regular maximum likelihood. *Annals of Mathematical Statistiscs*, 31:1208–1212, 1960.

[13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1983.

[14] P.J Green and B.W Silverman. *Nonparametric Regression and Genralized Linear Models*. Chapman & Hall, 1994.

[15] T. Hastie, P.Y Simard, and E.Sackinger. Learning prototype models for tangent distance. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 999–1006. Morgan Kaufmann, 1995.

[16] D. Heckerman and D. Geiger. Likelihoods and parameter priors for bayesian networks. Technical Report MSR-TR-95-54, Microsoft Research, November 1995.

[17] B. H. Juang and S. Katagari. Discriminative training. Technical Report 311402-2616, AT&T Bell Laboratories, 1992.

[18] C. Kooperberg, S. Bose, and C.J. Stone. Polychotomous regression. Technical Report 288, Dept. of Statistics, University of Washington, May 1996.

[19] C. Kooperberg and C. J. Stone. Logspline density estimation for censored data. *J. Comput. Graph. Statist.*, 1:301–328, 1992.

[20] P. A. Lachenbruch and M. R. Mickey. Estimation of error rates in discriminant analysis. *Technometrics*, 10:1–11, 1968.

[21] Geoffrey J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, 1992.

[22] Hermann Ney. On the probabilistic interpretation of neural network classifiers and discriminative training criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):107–119, February 1995.

[23] E. Oja. *Subspace Methods of Pattern Recognition*. John Wiley & Sons Inc, New York, NY, 1983.

[24] T. J. O'Neill. The general distribution of the error rate of a classification procedure with application to logistic regression discrimination. *Journal of the American Statistical Association*, 75(369):154–160, March 1980.

[25] K. O. Perlmutter, S. M. Perlmutter, R. M. Gray, R. A. Olshen, and K. L. Oehler. Bayes risk weighted vector quantization with posterior estimation for image compression and classification. *IEEE Transactions on Image Processing*, 0, February 1996.

[26] S. Raudys and V. Pikelis. On dimensionality, sample size, classification error, and complexity of classification algorithm in pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:242–252, 1980.

[27] R. A. Redner and H.F. Walker. Mixture densities, maximum likelihood and the em algorithm. *SIAM Review*, 26(2):195–239, April 1984.

[28] Holger Schwenk and Maurice Milgram. Learning discriminant tangent models for handwritten character recognition. Technical report, PARC, Universite Paris 6, boite 164, 75252 Paris cedex 05, France, 1996.

[29] D. Sona, A. Sperduti, and A. Starita. A constructive learning algorithm for discriminant tangent models. In *Advances in Neural Information Processing Systems*, 1996.

[30] Charles W. Therrien. *Decision Estimation and Classification.* John Wiley & Sons, 1989.

[31] Michael E. Tipping and Christopher M. Bishop. Mixtures of probabilistic principal component analysers. Technical report, Neural Computing Research Group, Dept of Computer Science & Appleid Mathematics, Aston University, Birmingham, UK, http://www.ncrg.aston.ac.uk, June 1997.

[32] D Whitlark and M. Geurts. When and how much do average forecasts improve predictive accuracy. *J. Statist. Comput. Simul.*, 42:39–45, 1992.