9 Linear Discriminant Analysis

9.1 Introduction

In this chapter, we consider classification models of the following form:

$$p(y = c|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{x}|y = c, \boldsymbol{\theta})p(y = c|\boldsymbol{\theta})}{\sum_{c'} p(\boldsymbol{x}|y = c', \boldsymbol{\theta})p(y = c'|\boldsymbol{\theta})} \tag{9.1}$$

The term $p(y = c|\boldsymbol{\theta})$ is the prior over class labels, and the term $p(\boldsymbol{x}|y = c, \boldsymbol{\theta})$ is called the **class conditional density** for class c.

The overall model is called a **generative classifier**, since it specifies a way to *generate* the features $\boldsymbol{x}$ for each class c, by sampling from $p(\boldsymbol{x}|y = c, \boldsymbol{\theta})$. By contrast, a **discriminative classifier** directly models the class posterior $p(y|\boldsymbol{x}, \boldsymbol{\theta})$. We discuss the pros and cons of these two approaches to classification in Section 9.4.

If we choose the class conditional densities in a special way, we will see that the resulting posterior over classes is a linear function of $\boldsymbol{x}$, i.e., $\log p(y = c|\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{w}^\mathsf{T}\boldsymbol{x} + \text{const}$, where $\boldsymbol{w}$ is derived from $\boldsymbol{\theta}$. Thus the overall method is called **linear discriminant analysis** or **LDA**.[1]

9.2 Gaussian discriminant analysis

In this section, we consider a generative classifier where the class conditional densities are multivariate Gaussians:

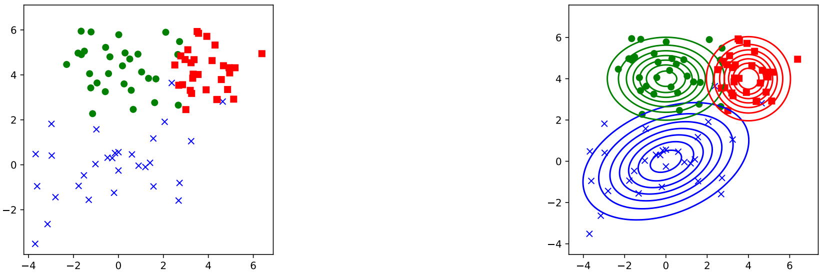
$$p(\boldsymbol{x}|y = c, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{9.2}$$

The corresponding class posterior therefore has the form

$$p(y = c|\boldsymbol{x}, \boldsymbol{\theta}) \propto \pi_c \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{9.3}$$

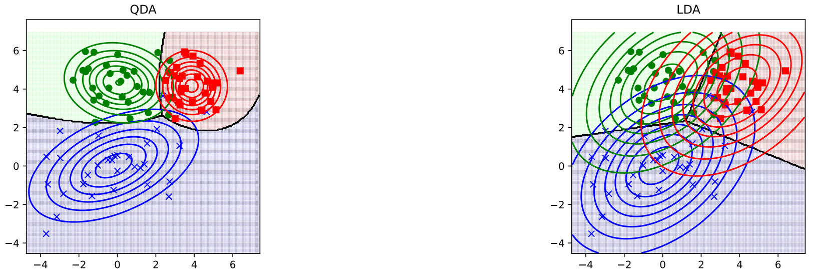
where $\pi_c = p(y = c|\boldsymbol{\theta})$ is the prior probability of label c. (Note that we can ignore the normalization constant in the denominator of the posterior, since it is independent of c.) We call this model **Gaussian discriminant analysis** or **GDA**.

[1] This term is rather confusing for two reasons. First, LDA is a generative, not discriminative, classifier. Second, LDA also stands for "latent Dirichlet allocation", which is a popular unsupervised generative model for bags of words [BNJ03].



(a) *(b)*

Figure 9.1: (a) Some 2d data from 3 different classes. (b) Fitting 2d Gaussians to each class. Generated by discrim_ analysis_ dboundaries_ plot2.ipynb.



(a) *(b)*

Figure 9.2: Gaussian discriminant analysis fit to data in Figure 9.1. (a) Unconstrained covariances induce quadratic decision boundaries. (b) Tied covariances induce linear decision boundaries. Generated by discrim_ analysis_ dboundaries_ plot2.ipynb.

9.2.1 Quadratic decision boundaries

From Equation (9.3), we see that the log posterior over class labels is given by

$$\log p(y = c|\boldsymbol{x}, \boldsymbol{\theta}) = \log \pi_c - \frac{1}{2} \log |2\pi \boldsymbol{\Sigma}_c| - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_c)^{\mathsf{T}} \boldsymbol{\Sigma}_c^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_c) + \text{const} \tag{9.4}$$

This is called the **discriminant function**. We see that the decision boundary between any two classes, say c and c', will be a quadratic function of $\boldsymbol{x}$. Hence this is known as **quadratic discriminant analysis** (QDA).

For example, consider the 2d data from 3 different classes in Figure 9.1a. We fit full covariance Gaussian class-conditionals (using the method explained in Section 9.2.4), and plot the results in Figure 9.1b. We see that the features for the blue class are somewhat correlated, whereas the features for the green class are independent, and the features for the red class are independent and isotropic (spherical covariance). In Figure 9.2a, we see that the resulting decision boundaries are quadratic functions of $\boldsymbol{x}$.

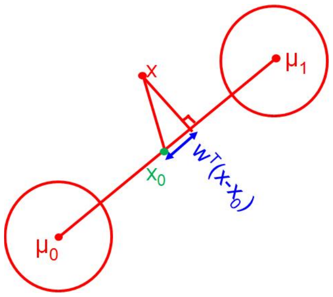


Figure 9.3: Geometry of LDA in the 2 class case where $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{I}$.

9.2.2 Linear decision boundaries

Now we consider a special case of Gaussian discriminant analysis in which the covariance matrices are **tied** or **shared** across classes, so $\mathbf{\Sigma}_c = \mathbf{\Sigma}$. If $\mathbf{\Sigma}$ is independent of c, we can simplify Equation (9.4) as follows:

$$\log p(y = c|\boldsymbol{x}, \boldsymbol{\theta}) = \log \pi_c - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_c)^\mathsf{T} \mathbf{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c) + \text{const} \tag{9.5}$$

$$= \underbrace{\log \pi_c - \frac{1}{2}\boldsymbol{\mu}_c^\mathsf{T} \mathbf{\Sigma}^{-1} \boldsymbol{\mu}_c}_{\gamma_c} + \boldsymbol{x}^\mathsf{T} \underbrace{\mathbf{\Sigma}^{-1} \boldsymbol{\mu}_c}_{\boldsymbol{\beta}_c} \underbrace{+\text{const} - \frac{1}{2}\boldsymbol{x}^\mathsf{T} \mathbf{\Sigma}^{-1} \boldsymbol{x}}_{\kappa} \tag{9.6}$$

$$= \gamma_c + \boldsymbol{x}^\mathsf{T} \boldsymbol{\beta}_c + \kappa \tag{9.7}$$

The final term is independent of c, and hence is an irrelevant additive constant that can be dropped. Hence we see that the discriminant function is a linear function of $\boldsymbol{x}$, so the decision boundaries will be linear. Hence this method is called **linear discriminant analysis** or **LDA**. See Figure 9.2b for an example.

9.2.3 The connection between LDA and logistic regression

In this section, we derive an interesting connection between LDA and logistic regression, which we introduced in Section 2.5.3. From Equation (9.7) we can write

$$p(y = c|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{e^{\boldsymbol{\beta}_c^\mathsf{T} \boldsymbol{x} + \gamma_c}}{\sum_{c'} e^{\boldsymbol{\beta}_{c'}^\mathsf{T} \boldsymbol{x} + \gamma_{c'}}} = \frac{e^{\boldsymbol{w}_c^\mathsf{T} [1, \boldsymbol{x}]}}{\sum_{c'} e^{\boldsymbol{w}_{c'}^\mathsf{T} [1, \boldsymbol{x}]}} \tag{9.8}$$

where $\boldsymbol{w}_c = [\gamma_c, \boldsymbol{\beta}_c]$. We see that Equation (9.8) has the same form as the multinomial logistic regression model. The key difference is that in LDA, we first fit the Gaussians (and class prior) to maximize the joint likelihood $p(\boldsymbol{x}, y|\boldsymbol{\theta})$, as discussed in Section 9.2.4, and then we derive $\boldsymbol{w}$ from $\boldsymbol{\theta}$. By contrast, in logistic regression, we estimate $\boldsymbol{w}$ directly to maximize the conditional likelihood $p(y|\boldsymbol{x}, \boldsymbol{w})$. In general, these can give different results (see Exercise 10.3).

To gain further insight into Equation (9.8), let us consider the binary case. In this case, the posterior is given by

$$p(y=1|\boldsymbol{x},\boldsymbol{\theta}) = \frac{e^{\boldsymbol{\beta}_1^\mathsf{T}\boldsymbol{x}+\gamma_1}}{e^{\boldsymbol{\beta}_1^\mathsf{T}\boldsymbol{x}+\gamma_1}+e^{\boldsymbol{\beta}_0^\mathsf{T}\boldsymbol{x}+\gamma_0}} = \frac{1}{1+e^{(\boldsymbol{\beta}_0-\boldsymbol{\beta}_1)^\mathsf{T}\boldsymbol{x}+(\gamma_0-\gamma_1)}} \tag{9.9}$$

$$= \sigma\left((\boldsymbol{\beta}_1-\boldsymbol{\beta}_0)^\mathsf{T}\boldsymbol{x}+(\gamma_1-\gamma_0)\right) \tag{9.10}$$

where $\sigma(\eta)$ refers to the sigmoid function.

Now

$$\gamma_1-\gamma_0 = -\frac{1}{2}\boldsymbol{\mu}_1^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1+\frac{1}{2}\boldsymbol{\mu}_0^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0+\log(\pi_1/\pi_0) \tag{9.11}$$

$$= -\frac{1}{2}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_0)+\log(\pi_1/\pi_0) \tag{9.12}$$

So if we define

$$\boldsymbol{w} = \boldsymbol{\beta}_1-\boldsymbol{\beta}_0 = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0) \tag{9.13}$$

$$\boldsymbol{x}_0 = \frac{1}{2}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_0)-(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)\frac{\log(\pi_1/\pi_0)}{(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)} \tag{9.14}$$

then we have $\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_0 = -(\gamma_1-\gamma_0)$, and hence

$$p(y=1|\boldsymbol{x},\boldsymbol{\theta}) = \sigma(\boldsymbol{w}^\mathsf{T}(\boldsymbol{x}-\boldsymbol{x}_0)) \tag{9.15}$$

This has the same form as binary logistic regression. Hence the MAP decision rule is

$$\hat{y}(\boldsymbol{x}) = 1 \text{ iff } \boldsymbol{w}^\mathsf{T}\boldsymbol{x} > c \tag{9.16}$$

where $c = \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_0$. If $\pi_0 = \pi_1 = 0.5$, then the threshold simplifies to $c = \frac{1}{2}\boldsymbol{w}^\mathsf{T}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_0)$.

To interpret this equation geometrically, suppose $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$. In this case, $\boldsymbol{w} = \sigma^{-2}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)$, which is parallel to a line joining the two centroids, $\boldsymbol{\mu}_0$ and $\boldsymbol{\mu}_1$. So we can classify a point by projecting it onto this line, and then checking if the projection is closer to $\boldsymbol{\mu}_0$ or $\boldsymbol{\mu}_1$, as illustrated in Figure 9.3. The question of how close it has to be depends on the prior over classes. If $\pi_1 = \pi_0$, then $\boldsymbol{x}_0 = \frac{1}{2}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_0)$, which is halfway between the means. If we make $\pi_1 > \pi_0$, we have to be closer to $\boldsymbol{\mu}_0$ than halfway in order to pick class 0. And vice versa if $\pi_0 > \pi_1$. Thus we see that the class prior just changes the decision threshold, but not the overall shape of the decision boundary. (A similar argument applies in the multi-class case.)

9.2.4 Model fitting

We now discuss how to fit a GDA model using maximum likelihood estimation. The likelihood function is as follows

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N}\text{Cat}(y_n|\boldsymbol{\pi})\prod_{c=1}^{C}\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c)^{\mathbb{I}(y_n=c)} \tag{9.17}$$

Hence the log-likelihood is given by

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \left[\sum_{n=1}^{N}\sum_{c=1}^{C}\mathbb{I}\left(y_n = c\right)\log \pi_c\right] + \sum_{c=1}^{C}\left[\sum_{n:y_n=c}\log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)\right] \tag{9.18}$$

Thus we see that we can optimize $\boldsymbol{\pi}$ and the $(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ terms separately.

From Section 4.2.4, we have that the MLE for the class prior is $\hat{\pi}_c = \frac{N_c}{N}$. Using the results from Section 4.2.6, we can derive the MLEs for the Gaussians as follows:

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{N_c}\sum_{n:y_n=c}\boldsymbol{x}_n \tag{9.19}$$

$$\hat{\boldsymbol{\Sigma}}_c = \frac{1}{N_c}\sum_{n:y_n=c}(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_c)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_c)^{\mathsf{T}} \tag{9.20}$$

Unfortunately the MLE for $\hat{\boldsymbol{\Sigma}}_c$ can easily overfit (i.e., the estimate may not be well-conditioned) if N_c is small compared to D, the dimensionality of the input features. We discuss some solutions to this below.

9.2.4.1 Tied covariances

If we force $\boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}$ to be tied, we will get linear decision boundaries, as we have seen. This also usually results in a more reliable parameter estimate, since we can pool all the samples across classes:

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{c=1}^{C}\sum_{n:y_n=c}(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_c)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_c)^{\mathsf{T}} \tag{9.21}$$

9.2.4.2 Diagonal covariances

If we force $\boldsymbol{\Sigma}_c$ to be diagonal, we reduce the number of parameters from $O(CD^2)$ to $O(CD)$, which avoids the overfitting problem. However, this loses the ability to capture correlations between the features. (This is known as the naive Bayes assumption, which we discuss further in Section 9.3.) Despite this approximation, this approach scales well to high dimensions.

We can further restrict the model capacity by using a shared (tied) diagonal covariace matrix. This is called “diagonal LDA” [BL04].

9.2.4.3 MAP estimation

Forcing the covariance matrix to be diagonal is a rather strong assumption. An alternative approach is to perform MAP estimation of a (shared) full covariance Gaussian, rather than using the MLE. Based on the results of Section 4.5.2, we find that the MAP estimate is

$$\hat{\boldsymbol{\Sigma}}_{\text{map}} = \lambda \text{diag}(\hat{\boldsymbol{\Sigma}}_{\text{mle}}) + (1-\lambda)\hat{\boldsymbol{\Sigma}}_{\text{mle}} \tag{9.22}$$

where λ controls the amount of regularization. This technique is known as **regularized discriminant analysis** or RDA [HTF09, p656].

9.2.5 Nearest centroid classifier

If we assume a uniform prior over classes, we can compute the most probable class label as follows:

$$\hat{y}(\boldsymbol{x}) = \underset{c}{\operatorname{argmax}} \log p(y = c|\boldsymbol{x}, \boldsymbol{\theta}) = \underset{c}{\operatorname{argmin}} (\boldsymbol{x} - \boldsymbol{\mu}_c)^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_c) \tag{9.23}$$

This is called the **nearest centroid classifier**, or **nearest class mean classifier** (**NCM**), since we are assigning $\boldsymbol{x}$ to the class with the closest $\boldsymbol{\mu}_c$, where distance is measured using (squared) Mahalanobis distance.

We can replace this with any other distance metric to get the decision rule

$$\hat{y}(\boldsymbol{x}) = \underset{c}{\operatorname{argmin}} \, d^2(\boldsymbol{x}, \boldsymbol{\mu}_c) \tag{9.24}$$

We discuss how to learn distance metrics in Section 16.2, but one simple approach is to use

$$d^2(\boldsymbol{x}, \boldsymbol{\mu}_c) = ||\boldsymbol{x} - \boldsymbol{\mu}_c||^2_{\mathbf{W}} = (\boldsymbol{x} - \boldsymbol{\mu}_c)^\mathsf{T} (\mathbf{W}\mathbf{W}^\mathsf{T})(\boldsymbol{x} - \boldsymbol{\mu}_c) = ||\mathbf{W}(\boldsymbol{x} - \boldsymbol{\mu}_c)||^2 \tag{9.25}$$

The corresponding class posterior becomes

$$p(y = c|\boldsymbol{x}, \boldsymbol{\mu}, \mathbf{W}) = \frac{\exp(-\frac{1}{2}||\mathbf{W}(\boldsymbol{x} - \boldsymbol{\mu}_c)||^2_2)}{\sum_{c'=1}^{C} \exp(-\frac{1}{2}||\mathbf{W}(\boldsymbol{x} - \boldsymbol{\mu}_{c'})||^2_2)} \tag{9.26}$$

We can optimize $\mathbf{W}$ using gradient descent applied to the discriminative loss. This is called **nearest class mean metric learning** [Men+12]. The advantage of this technique is that it can be used for **one-shot learning** of new classes, since we just need to see a single labeled prototype $\boldsymbol{\mu}_c$ per class (assuming we have learned a good $\mathbf{W}$ already).

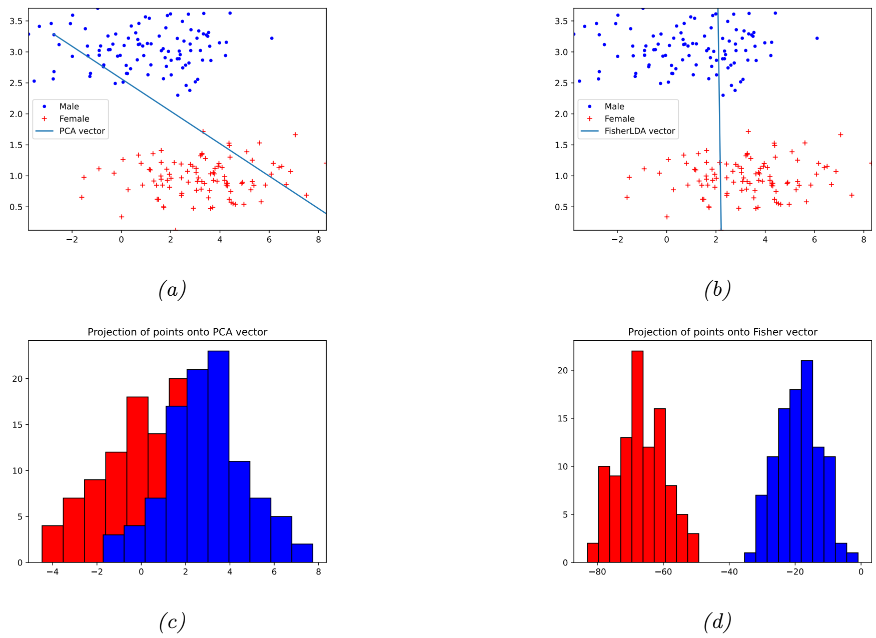
9.2.6 Fisher's linear discriminant analysis *

Discriminant analysis is a generative approach to classification, which requires fitting an MVN to the features. As we have discussed, this can be problematic in high dimensions. An alternative approach is to reduce the dimensionality of the features $\boldsymbol{x} \in \mathbb{R}^D$ and then fit an MVN to the resulting low-dimensional features $\boldsymbol{z} \in \mathbb{R}^K$. The simplest approach is to use a linear projection matrix, $\boldsymbol{z} = \mathbf{W}\boldsymbol{x}$, where $\mathbf{W}$ is a $K \times D$ matrix. One approach to finding $\mathbf{W}$ would be to use principal components analysis or PCA (Section 20.1). However, PCA is an unsupervised technique that does not take class labels into account. Thus the resulting low dimensional features are not necessarily optimal for classification, as illustrated in Figure 9.4.

An alternative approach is to use gradient based methods to optimize the log likelihood, derived from the class posterior in the low dimensional space, as we discussed in Section 9.2.5.

A third approach (which relies on an eigendecomposition, rather than a gradient-based optimizer) is to find the matrix $\mathbf{W}$ such that the low-dimensional data can be classified as well as possible using a Gaussian class-conditional density model. The assumption of Gaussianity is reasonable since we are computing linear combinations of (potentially non-Gaussian) features. This approach is called **Fisher's linear discriminant analysis**, or **FLDA**.

FLDA is an interesting hybrid of discriminative and generative techniques. The drawback of this technique is that it is restricted to using $K \leq C - 1$ dimensions, regardless of D, for reasons that we will explain below. In the two-class case, this means we are seeking a single vector $\boldsymbol{w}$ onto which we can project the data. Below we derive the optimal $\boldsymbol{w}$ in the two-class case. We then generalize to the multi-class case, and finally we give a probabilistic interpretation of this technique.



(a) (b)

(c) (d)

Figure 9.4: Linear discriminant analysis applied to two class dataset in 2d, representing (standardized) height and weight for male and female adults (a) PCA direction. (b) FLDA direction. (c) Projection onto PCA direction shows poor class separation. (d) Projection onto FLDA direction shows good class separation. Generated by fisher_lda_demo.ipynb.

9.2.6.1 Derivation of the optimal 1d projection

We now derive this optimal direction $\boldsymbol{w}$, for the two-class case, following the presentation of [Bis06, Sec 4.1.4]. Define the class-conditional means as

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n:y_n=1} \boldsymbol{x}_n, \ \boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n:y_n=2} \boldsymbol{x}_n \tag{9.27}$$

Let $m_k = \boldsymbol{w}^\mathsf{T} \boldsymbol{\mu}_k$ be the projection of each mean onto the line $\boldsymbol{w}$. Also, let $z_n = \boldsymbol{w}^\mathsf{T} \boldsymbol{x}_n$ be the projection of the data onto the line. The variance of the projected points is proportional to

$$s_k^2 = \sum_{n:y_n=k} (z_n - m_k)^2 \tag{9.28}$$

The goal is to find $\boldsymbol{w}$ such that we maximize the distance between the means, $m_2 - m_1$, while also ensuring the projected clusters are "tight", which we can do by minimizing their variance. This suggests the following objective:

$$J(\boldsymbol{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \tag{9.29}$$

We can rewrite the right hand side of the above in terms of $\boldsymbol{w}$ as follows

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^{\mathsf{T}}\mathbf{S}_B\boldsymbol{w}}{\boldsymbol{w}^{\mathsf{T}}\mathbf{S}_W\boldsymbol{w}} \tag{9.30}$$

where $\mathbf{S}_B$ is the between-class scatter matrix given by

$$\mathbf{S}_B = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^{\mathsf{T}} \tag{9.31}$$

and $\mathbf{S}_W$ is the within-class scatter matrix, given by

$$\mathbf{S}_W = \sum_{n:y_n=1}(\boldsymbol{x}_n - \boldsymbol{\mu}_1)(\boldsymbol{x}_n - \boldsymbol{\mu}_1)^{\mathsf{T}} + \sum_{n:y_n=2}(\boldsymbol{x}_n - \boldsymbol{\mu}_2)(\boldsymbol{x}_n - \boldsymbol{\mu}_2)^{\mathsf{T}} \tag{9.32}$$

To see this, note that

$$\boldsymbol{w}^{\mathsf{T}}\mathbf{S}_B\boldsymbol{w} = \boldsymbol{w}^{\mathsf{T}}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^{\mathsf{T}}\boldsymbol{w} = (m_2 - m_1)(m_2 - m_1) \tag{9.33}$$

and

$$\begin{aligned}\boldsymbol{w}^{\mathsf{T}}\mathbf{S}_W\boldsymbol{w} = &\sum_{n:y_n=1}\boldsymbol{w}^{\mathsf{T}}(\boldsymbol{x}_n - \boldsymbol{\mu}_1)(\boldsymbol{x}_n - \boldsymbol{\mu}_1)^{\mathsf{T}}\boldsymbol{w}+ \\ &\sum_{n:y_n=2}\boldsymbol{w}^{\mathsf{T}}(\boldsymbol{x}_n - \boldsymbol{\mu}_2)(\boldsymbol{x}_n - \boldsymbol{\mu}_2)^{\mathsf{T}}\boldsymbol{w} &\quad (9.34)\\ = &\sum_{n:y_n=1}(z_n - m_1)^2 + \sum_{n:y_n=2}(z_n - m_2)^2 &\quad (9.35)\end{aligned}$$

Equation (9.30) is a ratio of two scalars; we can take its derivative with respect to $\boldsymbol{w}$ and equate to zero. One can show (Exercise 9.1) that $J(\boldsymbol{w})$ is maximized when

$$\mathbf{S}_B\boldsymbol{w} = \lambda\mathbf{S}_W\boldsymbol{w} \tag{9.36}$$

where

$$\lambda = \frac{\boldsymbol{w}^{\mathsf{T}}\mathbf{S}_B\boldsymbol{w}}{\boldsymbol{w}^{\mathsf{T}}\mathbf{S}_W\boldsymbol{w}} \tag{9.37}$$

Equation (9.36) is called a **generalized eigenvalue** problem. If $\mathbf{S}_W$ is invertible, we can convert it to a regular eigenvalue problem:

$$\mathbf{S}_W^{-1}\mathbf{S}_B\boldsymbol{w} = \lambda\boldsymbol{w} \tag{9.38}$$

However, in the two class case, there is a simpler solution. In particular, since

$$\mathbf{S}_B\boldsymbol{w} = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^{\mathsf{T}}\boldsymbol{w} = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)(m_2 - m_1) \tag{9.39}$$

then, from Equation (9.38) we have

$$\begin{aligned}\lambda\ \boldsymbol{w} &= \mathbf{S}_W^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)(m_2 - m_1) &\quad (9.40)\\ \boldsymbol{w} &\propto \mathbf{S}_W^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) &\quad (9.41)\end{aligned}$$

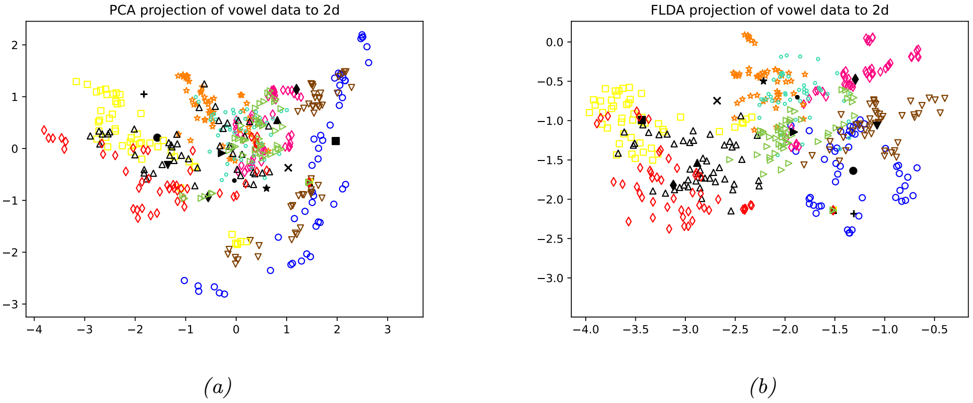


Figure 9.5: (a) PCA projection of vowel data to 2d. (b) FLDA projection of vowel data to 2d. We see there is better class separation in the FLDA case. Adapted from Figure 4.11 of [HTF09]. Generated by fisher_discrim_vowel.ipynb.

Since we only care about the directionality, and not the scale factor, we can just set

$$\boldsymbol{w} = \mathbf{S}_W^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) \tag{9.42}$$

This is the optimal solution in the two-class case. If $\mathbf{S}_W \propto \mathbf{I}$, meaning the pooled covariance matrix is isotropic, then $\boldsymbol{w}$ is proportional to the vector that joins the class means. This is an intuitively reasonable direction to project onto, as shown in Figure 9.3.

9.2.6.2 Extension to higher dimensions and multiple classes

We can extend the above idea to multiple classes, and to higher dimensional subspaces, by finding a projection *matrix* $\mathbf{W}$ which maps from D to K. Let $\boldsymbol{z}_n = \mathbf{W}\boldsymbol{x}_n$ be the low dimensional projection of the n'th data point. Let $\boldsymbol{m}_c = \frac{1}{N_c}\sum_{n:y_n=c} \boldsymbol{z}_n$ be the corresponding mean for the c'th class and $\boldsymbol{m} = \frac{1}{N}\sum_{c=1}^{C} N_c \boldsymbol{m}_c$ be the overall mean, both in the low dimensional space. We define the following scatter matrices:

$$\tilde{\mathbf{S}}_W = \sum_{c=1}^{C} \sum_{n:y_n=c} (\boldsymbol{z}_n - \boldsymbol{m}_c)(\boldsymbol{z}_n - \boldsymbol{m}_c)^{\mathsf{T}} \tag{9.43}$$

$$\tilde{\mathbf{S}}_B = \sum_{c=1}^{C} N_c(\boldsymbol{m}_c - \boldsymbol{m})(\boldsymbol{m}_c - \boldsymbol{m})^{\mathsf{T}} \tag{9.44}$$

Finally, we define the objective function as maximizing the following:[2]

$$J(\mathbf{W}) = \frac{|\tilde{\mathbf{S}}_B|}{|\tilde{\mathbf{S}}_W|} = \frac{|\mathbf{W}^{\mathsf{T}}\mathbf{S}_B\mathbf{W}|}{|\mathbf{W}^{\mathsf{T}}\mathbf{S}_W\mathbf{W}|} \tag{9.45}$$

[2] An alternative criterion that is sometimes used [Fuk90] is $J(\mathbf{W}) = \text{tr}\left\{\tilde{\mathbf{S}}_W^{-1}\tilde{\mathbf{S}}_B\right\} = \text{tr}\left\{(\mathbf{W}\mathbf{S}_W\mathbf{W}^{\mathsf{T}})^{-1}(\mathbf{W}\mathbf{S}_B\mathbf{W}^{\mathsf{T}})\right\}$.

where $\mathbf{S}_W$ and $\mathbf{S}_B$ are defined in the original high dimensional space in the obvious way (namely using $\boldsymbol{x}_n$ instead of $\boldsymbol{z}_n$, $\boldsymbol{\mu}_c$ instead of $\boldsymbol{m}_c$, and $\boldsymbol{\mu}$ instead of $\boldsymbol{m}$). The solution can be shown [DHS01] to be $\mathbf{W} = \mathbf{S}_W^{-\frac{1}{2}}\mathbf{U}$, where $\mathbf{U}$ are the K leading eigenvectors of $\mathbf{S}_W^{-\frac{1}{2}}\mathbf{S}_B\mathbf{S}_W^{-\frac{1}{2}}$, assuming $\mathbf{S}_W$ is non-singular. (If it is singular, we can first perform PCA on all the data.)

Figure 9.5 gives an example of this method applied to some $D = 10$ dimensional speech data, representing $C = 11$ different vowel sounds. We project to $K = 2$ dimensions in order to visualize the data. We see that FLDA gives better class separation than PCA.

Note that FLDA is restricted to finding at most a $K \leq C - 1$ dimensional linear subspace, no matter how large D, because the rank of the between class scatter matrix $\mathbf{S}_B$ is $C - 1$. (The -1 term arises because of the $\boldsymbol{\mu}$ term, which is a linear function of the $\boldsymbol{\mu}_c$.) This is a rather severe restriction which limits the usefulness of FLDA.

9.3 Naive Bayes classifiers

In this section, we discuss a simple generative approach to classification in which we assume the features are conditionally independent given the class label. This is called the **naive Bayes assumption**. The model is called "naive" since we do not expect the features to be independent, even conditional on the class label. However, even if the naive Bayes assumption is not true, it often results in classifiers that work well [DP97; HY01a]. One reason for this is that the model is quite simple (it only has $O(CD)$ parameters, for C classes and D features), and hence it is relatively immune to overfitting.

More precisely, the naive Bayes assumption corresponds to using a class conditional density of the following form:

$$p(\boldsymbol{x}|y = c, \boldsymbol{\theta}) = \prod_{d=1}^{D} p(x_d|y = c, \boldsymbol{\theta}_{dc}) \tag{9.46}$$

where $\boldsymbol{\theta}_{dc}$ are the parameters for the class conditional density for class c and feature d. Hence the posterior over class labels is given by

$$p(y = c|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{p(y = c|\boldsymbol{\pi}) \prod_{d=1}^{D} p(x_d|y = c, \boldsymbol{\theta}_{dc})}{\sum_{c'} p(y = c'|\boldsymbol{\pi}) \prod_{d=1}^{D} p(x_d|y = c', \boldsymbol{\theta}_{dc'})} \tag{9.47}$$

where π_c is the prior probability of class c, and $\boldsymbol{\theta} = (\boldsymbol{\pi}, \{\boldsymbol{\theta}_{dc}\})$ are all the parameters. This is known as a **naive Bayes classifier** or **NBC**.

9.3.1 Example models

We still need to specify the form of the probability distributions in Equation (9.46). This depends on what type of feature x_d is. We give some examples below:

- In the case of binary features, $x_d \in \{0, 1\}$, we can use the Bernoulli distribution: $p(\boldsymbol{x}|y = c, \boldsymbol{\theta}) = \prod_{d=1}^{D} \text{Ber}(x_d|\theta_{dc})$, where θ_{dc} is the probability that $x_d = 1$ in class c. This is sometimes called the **multivariate Bernoulli naive Bayes** model. For example, Figure 9.6 shows the estimated parameters for each class when we fit this model to a binarized version of MNIST. This approach does surprisingly well, and has a test set accuracy of 84.3%. (See Figure 9.7 for some sample predictions.)

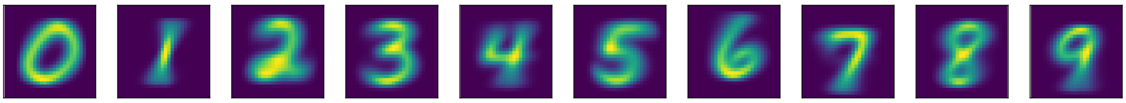


Figure 9.6: Visualization of the Bernoulli class conditional densities for a naive Bayes classifier fit to a binarized version of the MNIST dataset. Generated by naive_bayes_mnist_jax.ipynb.

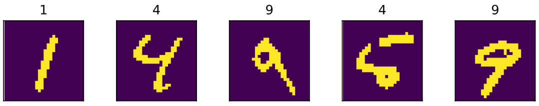


Figure 9.7: Visualization of the predictions made by the model in Figure 9.6 when applied to some binarized MNIST test images. The title shows the most probable predicted class. Generated by naive_bayes_mnist_jax.ipynb.

- In the case of categorical features, $x_d \in \{1, \ldots, K\}$, we can use the categorical distribution: $p(\boldsymbol{x}|y = c, \boldsymbol{\theta}) = \prod_{d=1}^{D} \mathrm{Cat}(x_d|\boldsymbol{\theta}_{dc})$, where θ_{dck} is the probability that $x_d = k$ given that $y = c$.
- In the case of real-valued features, $x_d \in \mathbb{R}$, we can use the univariate Gaussian distribution: $p(\boldsymbol{x}|y = c, \boldsymbol{\theta}) = \prod_{d=1}^{D} \mathcal{N}(x_d|\mu_{dc}, \sigma_{dc}^2)$, where μ_{dc} is the mean of feature d when the class label is c, and σ_{dc}^2 is its variance. (This is equivalent to Gaussian discriminant analysis using diagonal covariance matrices.)

9.3.2 Model fitting

In this section, we discuss how to fit a naive Bayes classifier using maximum likelihood estimation.

We can write the likelihood as follows:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \left[\mathrm{Cat}(y_n|\boldsymbol{\pi}) \prod_{d=1}^{D} p(x_{nd}|y_n, \boldsymbol{\theta}_d) \right] \tag{9.48}$$

$$= \prod_{n=1}^{N} \left[\mathrm{Cat}(y_n|\boldsymbol{\pi}) \prod_{d=1}^{D} \prod_{c=1}^{C} p(x_{nd}|\boldsymbol{\theta}_{dc})^{\mathbb{I}(y_n=c)} \right] \tag{9.49}$$

so the log-likelihood is given by

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \left[\sum_{n=1}^{N} \sum_{c=1}^{C} \mathbb{I}(y_n = c) \log \pi_c \right] + \sum_{c=1}^{C} \sum_{d=1}^{D} \left[\sum_{n:y_n=c} \log p(x_{nd}|\boldsymbol{\theta}_{dc}) \right] \tag{9.50}$$

We see that this decomposes into a term for $\boldsymbol{\pi}$, and CD terms for each $\boldsymbol{\theta}_{dc}$:

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \log p(\mathcal{D}_y|\boldsymbol{\pi}) + \sum_c \sum_d \log p(\mathcal{D}_{dc}|\boldsymbol{\theta}_{dc}) \tag{9.51}$$

where $\mathcal{D}_y = \{y_n : n = 1 : N\}$ are all the labels, and $\mathcal{D}_{dc} = \{x_{nd} : y_n = c\}$ are all the values of feature d for examples from class c. Hence we can estimate these parameters separately.

In Section 4.2.4, we show that the MLE for $\boldsymbol{\pi}$ is the vector of empirical counts, $\hat{\pi}_c = \frac{N_c}{N}$. The MLEs for $\boldsymbol{\theta}_{dc}$ depend on the choice of the class conditional density for feature d. We discuss some common choices below.

- In the case of discrete features, we can use a categorical distribution. A straightforward extension of the results in Section 4.2.4 gives the following expression for the MLE:

$$\hat{\theta}_{dck} = \frac{N_{dck}}{\sum_{k'=1}^{K} N_{dck'}} = \frac{N_{dck}}{N_c} \tag{9.52}$$

  where $N_{dck} = \sum_{n=1}^{N} \mathbb{I}\left(x_{nd} = k, y_n = c\right)$ is the number of times that feature d had value k in examples of class c.

- In the case of binary features, the categorical distribution becomes the Bernoulli, and the MLE becomes

$$\hat{\theta}_{dc} = \frac{N_{dc}}{N_c} \tag{9.53}$$

  which is the empirical fraction of times that feature d is on in examples of class c.

- In the case of real-valued features, we can use a Gaussian distribution. A straightforward extension of the results in Section 4.2.5 gives the following expression for the MLE:

$$\hat{\mu}_{dc} = \frac{1}{N_c} \sum_{n:y_n=c} x_{nd} \tag{9.54}$$

$$\hat{\sigma}_{dc}^2 = \frac{1}{N_c} \sum_{n:y_n=c} (x_{nd} - \hat{\mu}_{dc})^2 \tag{9.55}$$

Thus we see that fitting a naive Bayes classifier is extremely simple and efficient.

9.3.3 Bayesian naive Bayes

In this section, we extend our discussion of MLE estimation for naive Bayes classifiers from Section 9.3.2 to compute the posterior distribution over the parameters. For simplicity, let us assume we have categorical features, so $p(x_d|\boldsymbol{\theta}_{dc}) = \text{Cat}(x_d|\boldsymbol{\theta}_{dc})$, where $\theta_{dck} = p(x_d = k|y = c)$. In Section 4.6.3.2, we show that the conjugate prior for the categorical likelihood is the Dirichlet distribution, $p(\boldsymbol{\theta}_{dc}) = \text{Dir}(\boldsymbol{\theta}_{dc}|\boldsymbol{\beta}_{dc})$, where β_{dck} can be interpereted as a set of "**pseudo counts**", corresponding to counts N_{dck} that come from prior data. Similarly we use a Dirichlet prior for the label frequencies, $p(\boldsymbol{\pi}) = \text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha})$. By using a conjugate prior, we can compute the posterior in closed form, as we explain in Section 4.6.3. In particular, we have

$$p(\boldsymbol{\theta}|\mathcal{D}) = \text{Dir}(\boldsymbol{\pi}|\,\widehat{\boldsymbol{\alpha}}) \prod_{d=1}^{D} \prod_{c=1}^{C} \text{Dir}(\boldsymbol{\theta}_{dc}|\,\widehat{\boldsymbol{\beta}}_{dc}) \tag{9.56}$$

where $\widehat{\alpha}_c = \breve{\alpha}_c + N_c$ and $\widehat{\beta}_{dck} = \breve{\beta}_{dck} + N_{dck}$.

Using the results from Section 4.6.3.4, we can derive the posterior predictive distribution as follows. For the label prior (before seeing $\boldsymbol{x}$, but after seeing $\mathcal{D}$), we have $p(y|\mathcal{D}) = \text{Cat}(y|\overline{\boldsymbol{\pi}})$, where $\overline{\pi}_c = \widehat{\alpha}_c / \sum_{c'} \widehat{\alpha}_{c'}$. For the feature likelihood of $\boldsymbol{x}$ (given y and $\mathcal{D}$), we have $p(x_d = k|y = c, \mathcal{D}) = \overline{\theta}_{dck}$, where

$$\overline{\theta}_{dck} = \frac{\widehat{\beta}_{dck}}{\sum_{k'=1}^{K} \widehat{\beta}_{dck'}} = \frac{\breve{\beta}_{dck} + N_{dck}}{\sum_{k'=1}^{K} \breve{\beta}_{dck'} + N_{dck'}} \tag{9.57}$$

is the posterior mean of the parameters. (Note that $\sum_{k'=1}^{K} N_{dck'} = N_{dc} = N_c$ is the number of examples for class c.)

If $\breve{\beta}_{dck} = 0$, this reduces to the MLE in Equation (9.52). By contrast, if we set $\breve{\beta}_{dck} = 1$, we add 1 to all the empirical counts before normalizing. This is called **add-one smoothing** or **Laplace smoothing**. For example, in the binary case, this gives

$$\overline{\theta}_{dc} = \frac{\breve{\beta}_{dc1} + N_{dc1}}{\breve{\beta}_{dc0} + N_{dc0} + \breve{\beta}_{dc1} + N_{dc1}} = \frac{1 + N_{dc1}}{2 + N_{dc}} \tag{9.58}$$

We can finally compute the posterior predictive distribution over the label as follows:

$$p(y = c|\boldsymbol{x}, \mathcal{D}) \propto p(y = c|\mathcal{D}) \prod_{d} p(x_d|y = c, \mathcal{D}) = \overline{\pi}_c \prod_{d} \prod_{k} \overline{\theta}_{dck}^{\mathbb{I}(x_d = k)} \tag{9.59}$$

This gives us a fully Bayesian form of naive Bayes, in which we have integrated out all the parameters. (In this case, the predictive distribution can be obtained merely by plugging in the posterior mean parameters.)

9.3.4 The connection between naive Bayes and logistic regression

In this section, we show that the class posterior $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ for a NBC model has the same form as multinomial logistic regression. For simplicity, we assume that the features are all discrete, and each has K states, although the result holds for arbitrary feature distributions in the exponential family.

Let $x_{dk} = \mathbb{I}(x_d = k)$, so $\boldsymbol{x}_d$ is a one-hot encoding of feature d. Then the class conditional density can be written as follows:

$$p(\boldsymbol{x}|y = c, \boldsymbol{\theta}) = \prod_{d=1}^{D} \text{Cat}(x_d|y = c, \boldsymbol{\theta}) = \prod_{d=1}^{D} \prod_{k=1}^{K} \theta_{dck}^{x_{dk}} \tag{9.60}$$

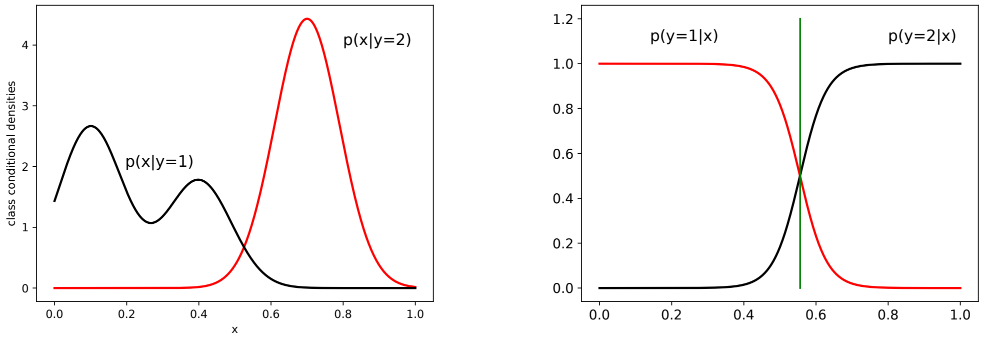
Hence the posterior over classes is given by

$$p(y = c|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{\pi_c \prod_d \prod_k \theta_{dck}^{x_{dk}}}{\sum_{c'} \pi_{c'} \prod_d \prod_k \theta_{dc'k}^{x_{dk}}} = \frac{\exp[\log \pi_c + \sum_d \sum_k x_{dk} \log \theta_{dck}]}{\sum_{c'} \exp[\log \pi_{c'} + \sum_d \sum_k x_{dk} \log \theta_{dc'k}]} \tag{9.61}$$

This can be written as a softmax

$$p(y = c|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{e^{\boldsymbol{\beta}_c^{\mathsf{T}} \boldsymbol{x} + \gamma_c}}{\sum_{c'=1}^{C} e^{\boldsymbol{\beta}_{c'}^{\mathsf{T}} \boldsymbol{x} + \gamma_{c'}}} \tag{9.62}$$

by suitably defining $\boldsymbol{\beta}_c$ and γ_c. This has exactly the same form as multinomial logistic regression in Section 2.5.3. The difference is that with naive Bayes we optimize the joint likelihood $\prod_n p(y_n, \boldsymbol{x}_n|\boldsymbol{\theta})$, whereas with logistic regression, we optimize the conditional likelihood $\prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta})$. In general, these can give different results (see Exercise 10.3).



(a) *(b)*

Figure 9.8: The class-conditional densities $p(x|y = c)$ (left) may be more complex than the class posteriors $p(y = c|x)$ (right). Adapted from Figure 1.27 of [Bis06]. Generated by generativeVsDiscrim.ipynb.

9.4 Generative vs discriminative classifiers

A model of the form $p(\boldsymbol{x}, y) = p(y)p(\boldsymbol{x}|y)$ is called a **generative classifier**, since it can be used to generate examples $\boldsymbol{x}$ from each class y. By contrast, a model of the form $p(y|\boldsymbol{x})$ is called a **discriminative classifier**, since it can only be used to discriminate between different classes. Below we discuss various pros and cons of the generative and discriminative approaches to classification. (See also [BT04; UB05; LBM06; BL07a; Rot+18].)

9.4.1 Advantages of discriminative classifiers

The main advantages of discriminative classifiers are as follows:

- **Better predictive accuracy**. Discriminative classifiers are often much more accurate than generative classifiers [NJ02]. The reason is that the conditional distribution $p(y|\boldsymbol{x})$ is often much simpler (and therefore easier to learn) than the joint distribution $p(y, \boldsymbol{x})$, as illustrated in Figure 9.8. In particular, discriminative models do not need to "waste effort" modeling the distribution of the input features.

- **Can handle feature preprocessing**. A big advantage of discriminative methods is that they allow us to preprocess the input in arbitrary ways. For example, we can perform a polynomial expansion of the input features, and we can replace a string of words with embedding vectors (see Section 20.5). It is often hard to define a generative model on such pre-processed data, since the new features can be correlated in complex ways which are hard to model.

- **Well-calibrated probabilities**. Some generative classifiers, such as naive Bayes (described in Section 9.3), make strong independence assumptions which are often not valid. This can result in very extreme posterior class probabilities (very near 0 or 1). Discriminative models, such as logistic regression, are often better calibrated in terms of their probability estimates, although they also sometimes need adjustment (see e.g., [NMC05]).

9.4.2 Advantages of generative classifiers

The main advantages of generative classifiers are as follows:

- **Easy to fit.** Generative classifiers are often very easy to fit. For example, in Section 9.3.2, we show how to fit a naive Bayes classifier by simple counting and averaging. By contrast, logistic regression requires solving a convex optimization problem (see Section 10.2.3 for the details), and neural nets require solving a non-convex optimization problem, both of which are much slower.

- **Can easily handle missing input features.** Sometimes some of the inputs (components of $\boldsymbol{x}$) are not observed. In a generative classifier, there is a simple method for dealing with this, as we show in Section 1.5.5. However, in a discriminative classifier, there is no principled solution to this problem, since the model assumes that $\boldsymbol{x}$ is always available to be conditioned on.

- **Can fit classes separately.** In a generative classifier, we estimate the parameters of each class conditional density independently (as we show in Section 9.3.2), so we do not have to retrain the model when we add more classes. In contrast, in discriminative models, all the parameters interact, so the whole model must be retrained if we add a new class.

- **Can handle unlabeled training data.** It is easy to use generative models for semi-supervised learning, in which we combine labeled data $\mathcal{D}_{xy} = \{(\boldsymbol{x}_n, y_n)\}$ and unlabeled data, $\mathcal{D}_x = \{\boldsymbol{x}_n\}$. However, this is harder to do with discriminative models, since there is no uniquely optimal way to exploit $\mathcal{D}_x$.

- **May be more robust to spurious features**. A discriminative model $p(y|\boldsymbol{x})$ may pick up on features of the input $\boldsymbol{x}$ that can discriminate different values of y in the training set, but which are not robust and do not generalize beyond the training set. These are called **spurious features** (see e.g., [Arj21; Zho+21]). By contrast, a generative model $p(\boldsymbol{x}|y)$ may be better able to capture the causal mechanisms of the underlying data generating process; such causal models can be more robust to distribution shift (see e.g., [Sch19; LBS19; LN81]).

9.4.3 Handling missing features

Sometimes we are missing parts of the input $\boldsymbol{x}$ during training and/or testing. In a generative classifier, we can handle this situation by marginalizing out the missing values. (We assume that the missingness of a feature is not informative about its potential value.) By contrast, when using a discriminative model, there is no unique best way to handle missing inputs, as we discuss in Section 1.5.5.

For example, suppose we are missing the value of x_1. We just have to compute

$$p(y = c|\boldsymbol{x}_{2:D}, \boldsymbol{\theta}) \propto p(y = c|\boldsymbol{\pi})p(\boldsymbol{x}_{2:D}|y = c, \boldsymbol{\theta}) \tag{9.63}$$

$$= p(y = c|\boldsymbol{\pi}) \sum_{x_1} p(x_1, \boldsymbol{x}_{2:D}|y = c, \boldsymbol{\theta}) \tag{9.64}$$

In Gaussian discriminant analysis, we can marginalize out x_1 using the equations from Section 3.2.3.

If we make the naive Bayes assumption, things are even easier, since we can just ignore the likelihood term for x_1. This follows because

$$\sum_{x_1} p(x_1, x_{2:D}|y=c, \boldsymbol{\theta}) = \left[\sum_{x_1} p(x_1|\boldsymbol{\theta}_{1c})\right] \prod_{d=2}^{D} p(x_d|\boldsymbol{\theta}_{dc}) = \prod_{d=2}^{D} p(x_d|\boldsymbol{\theta}_{dc}) \tag{9.65}$$

where we exploited the fact that $p(x_d|y=c,\boldsymbol{\theta}) = p(x_d|\boldsymbol{\theta}_{dc})$ and $\sum_{x_1} p(x_1|\boldsymbol{\theta}_{1c}) = 1$.

9.5 Exercises

Exercise 9.1 [Derivation of Fisher's linear discriminant]

Show that the maximum of $J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \mathbf{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \mathbf{S}_W \boldsymbol{w}}$ is given by $\mathbf{S}_B \boldsymbol{w} = \lambda \mathbf{S}_W \boldsymbol{w}$
where $\lambda = \frac{\boldsymbol{w}^T \mathbf{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \mathbf{S}_W \boldsymbol{w}}$. Hint: recall that the derivative of a ratio of two scalars is given by $\frac{d}{dx}\frac{f(x)}{g(x)} = \frac{f'g - fg'}{g^2}$, where $f' = \frac{d}{dx}f(x)$ and $g' = \frac{d}{dx}g(x)$. Also, recall that $\frac{d}{d\boldsymbol{x}}\boldsymbol{x}^T \mathbf{A} \boldsymbol{x} = (\mathbf{A} + \mathbf{A}^T)\boldsymbol{x}$.