The Elements of Statistical Learning - Chapter 4 Exercises

Exercise 4.1

Show how to solve the generalized eigenvalue problem $\max a^T \mathbf{B} a$ subject to $a^T \mathbf{W} a = 1$ by transforming to a standard eigenvalue problem.

Solution

Take the eigendecomposition $\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{U}^T$ where $\mathbf{U}$ is a $p \times p$ orthogonal matrix. Let $\mathbf{W}^{1/2} = \mathbf{U}\mathbf{D}^{1/2}\mathbf{U}^T$ and $a^* = \mathbf{W}^{1/2} a$. The covariance matrix $\mathbf{B}^*$ can be obtained from $\mathbf{B}^* = \mathbf{W}^{-1/2}\mathbf{B}\mathbf{W}^{-1/2}$,

so the constrained optimisation problem is equivalent to $\max a^{*T} \mathbf{B}^* a^*$ subject to $a^{*T} a^* = 1$.

Consider the eigendecomposition $\mathbf{B}^* = \mathbf{V}^* \mathbf{D}_B \mathbf{V}^{*T}$ where the diagonal entries of $\mathbf{D}_B$ are in decreasing order (they are all positive). Write a^* as a linear combination of the eigenvectors $a^* = \sum_i c_i v_i$. This transforms the optimisation problem into the following: $\max \sum_i \lambda_i c_i^2$ subject to $\sum_i c_i^2 = 1$. This is a standard Lagrange multiplier problem with solution $c = e_1$.

Transforming back to the original coordinates gives

$$a = \mathbf{U}\mathbf{D}^{-1/2}\mathbf{U}^T v_1$$

as required.

Exercise 4.2

Suppose we have features $x \in \mathbb{R}^p$, a two-class response, with class sizes N_1, N_2, and the target coded as $-N/N_1, N/N_2$.

(a) Show that the LDA rule classifies to class 2 if

$$x^T \hat\Sigma^{-1}(\hat\mu_2 - \hat\mu_1) > \frac{1}{2}\hat\mu_2^T\hat\Sigma^{-1}\hat\mu_2 - \frac{1}{2}\hat\mu_1^T\hat\Sigma^{-1}\hat\mu_1 + \log(N_1/N) - \log(N_2/N),$$

and class 1 otherwise.

(b) Consider minimization of the least squares criterion

$$\sum_{i=1}^N (y_i - \beta_0 - \beta^T x_i)^2.$$

Show that the solution $\hat\beta$ satisfies

$$\left[(N-2)\hat\Sigma + N\hat\Sigma_B\right]\beta = N(\hat\mu_2 - \hat\mu_1)$$

(after simplification), where $\hat\Sigma_B = \frac{N_1 N_2}{N^2}(\hat\mu_2 - \hat\mu_1)(\hat\mu_2 - \hat\mu_1)^T$.

(c) Hence show that $\hat\Sigma_B \beta$ is in the direction $(\hat\mu_2 - \hat\mu_1)$ and thus

$$\hat\beta \propto \hat\Sigma^{-1}(\hat\mu_2 - \hat\mu_1).$$

Therefore the least-squares regression coefficient is identical to the LDA coefficient, up to a scalar multiple.

(d) Show that this result holds for any (distinct) coding of the two classes.

(e) Find the solution $\hat\beta_0$ (up to the same scalar multiple as in (c)), and hence the predicted value $\hat f(x) = \hat\beta_0 + \hat\beta^T x$. Consider the following rule: classify to class 2 if $\hat f(x) > 0$ and class 1 otherwise. Show this is not the same as the LDA rule unless the classes have equal numbers of observations.

Solution

(a) The discriminant functions for LDA are

$$\delta_k(x) = x^T\hat\Sigma^{-1}\hat\mu_k - \frac{1}{2}\hat\mu_k^T\hat\Sigma^{-1}\hat\mu_k + \ln(\pi_k).$$

Estimating μ_k by $\hat\mu_k$, Σ by $\hat\Sigma$, and π_k by N_k/N, we classify to class 2 if and only if

$$\begin{aligned}
&\delta_2(x) > \delta_1(x) \\
\iff\ & x^T\hat\Sigma^{-1}\hat\mu_2 - \frac{1}{2}\hat\mu_2^T\hat\Sigma^{-1}\hat\mu_2 + \ln(N_2/N) > x^T\hat\Sigma^{-1}\hat\mu_1 - \frac{1}{2}\hat\mu_1^T\hat\Sigma^{-1}\hat\mu_1 + \ln(N_1/N) \\
\iff\ & x^T\hat\Sigma^{-1}(\hat\mu_2 - \hat\mu_1) > \frac{1}{2}\hat\mu_2^T\hat\Sigma^{-1}\hat\mu_2 - \frac{1}{2}\hat\mu_1^T\hat\Sigma^{-1}\hat\mu_1 + \ln(N_1/N) - \ln(N_2/N)
\end{aligned}$$

and to class 1 otherwise.

(b) Let $\tilde{\mathbf{X}}$ denote the input data matrix augmented with the intercept column. We know that $\hat\beta$ satisfies $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\hat\beta = \tilde{\mathbf{X}}^T\mathbf{y}$.

We start to solve $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\beta$ and $\tilde{\mathbf{X}}^T\mathbf{y}$. The (i, j) entry of $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$ is $\sum_{n=1}^N x_{ni}x_{nj}$. Taking $k = 0$, the jth entry of the kth column is $\sum_{n=1}^N x_{nj} = N_1(\hat\mu_1)_j + N_2(\hat\mu_2)_j$.

So $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$ is a $(p+1)\times(p+1)$ matrix with $(0,0)$ entry N, $(1,0),\dots,(p,0)$ entries taken up by $N_1\hat\mu_1 + N_2\hat\mu_2$, $(0,1),\dots,(0,p)$ entries taken up by $N_1\hat\mu_1^T + N_2\hat\mu_2^T$, and bottom right $p\times p$ submatrix equal to $\mathbf{X}^T\mathbf{X}$. This implies that $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$ is equal to the block matrix

$$\begin{pmatrix} N & N_1\hat\mu_1^T + N_2\hat\mu_2^T \\ N_1\hat\mu_1 + N_2\hat\mu_2 & \mathbf{X}^T\mathbf{X} \end{pmatrix}$$

where the top expression is a scalar and the bottom is a $p\times p$ matrix.

We have already seen that this is equal to $\tilde{\mathbf{X}}^T\mathbf{y}$. The jth entry of $\tilde{\mathbf{X}}^T\mathbf{y}$ is

$$\sum_{n=1}^N y_n x_{nj} = -\frac{N}{N_1}\sum_{g_n=1}x_{nj} + \frac{N}{N_2}\sum_{g_n=2}x_{nj} = N(\hat\mu_2 - \hat\mu_1)_j$$

so $\tilde{\mathbf{X}}^T\mathbf{y}$ is $N(\hat\mu_2 - \hat\mu_1)$ augmented with a zero in the 0th place. Equating the expressions in the 0th place gives

$$\beta_0 = -\frac{1}{N}(N_1\hat\mu_1^T + N_2\hat\mu_2^T)\beta.$$

Equating the remaining terms and substituting this in gives

$$\left[\mathbf{X}^T\mathbf{X} - \frac{1}{N}(N_1\hat\mu_1 + N_2\hat\mu_2)(N_1\hat\mu_1 + N_2\hat\mu_2)^T\right]\beta = N(\hat\mu_2 - \hat\mu_1).$$

It remains to show that the expression in square brackets equal $(N-2)\hat\Sigma + N\hat\Sigma_B$.

Indeed,

$$\begin{aligned}
(N-2)\hat\Sigma &= \sum_{k=1}^2\sum_{g_i=k}(x_i - \hat\mu_k)(x_i - \hat\mu_k)^T \\
&= \sum_{k=1}^2\sum_{g_i=k}(x_ix_i^T - \hat\mu_kx_i^T - x_i\hat\mu_k^T + \hat\mu_k\hat\mu_k^T) \\
&= \sum_{k=1}^2\left[\left(\sum_{g_i=k}x_ix_i^T\right) - N_k\hat\mu_k\hat\mu_k^T\right] \\
&= \mathbf{X}^T\mathbf{X} - N_1\hat\mu_1\hat\mu_1^T - N_2\hat\mu_2\hat\mu_2^T.
\end{aligned}$$

So,

$$\begin{aligned}
(N-2)\hat\Sigma + N\hat\Sigma_B &= \mathbf{X}^T\mathbf{X} - N_1\hat\mu_1\hat\mu_1^T - N_2\hat\mu_2\hat\mu_2^T + \frac{N_1N_2}{N}(\hat\mu_2 - \hat\mu_1)(\hat\mu_2 - \hat\mu_1)^T \\
&= \mathbf{X}^T\mathbf{X} + \left(\frac{N_1N_2}{N} - N_1\right)\hat\mu_1\hat\mu_1^T + \left(\frac{N_1N_2}{N} - N_2\right)\hat\mu_2\hat\mu_2^T - \frac{N_1N_2}{N}(\hat\mu_1\hat\mu_2^T + \hat\mu_2\hat\mu_1^T) \\
&= \mathbf{X}^T\mathbf{X} - \frac{N_1^2}{N}\hat\mu_1\hat\mu_1^T - \frac{N_2^2}{N}\hat\mu_2\hat\mu_2^T - \frac{N_1N_2}{N}(\hat\mu_1\hat\mu_2^T + \hat\mu_2\hat\mu_1^T) \\
&= \mathbf{X}^T\mathbf{X} - \frac{1}{N}(N_1\hat\mu_1 + N_2\hat\mu_2)(N_1\hat\mu_1 + N_2\hat\mu_2)^T
\end{aligned}$$

as required.

(c) By definition,

$$\hat\Sigma_B\beta = \frac{N_1N_2}{N^2}(\hat\mu_2 - \hat\mu_1)(\hat\mu_2 - \hat\mu_1)^T\beta = \left[\frac{N_1N_2}{N^2}(\hat\mu_2 - \hat\mu_1)^T\beta\right](\hat\mu_2 - \hat\mu_1).$$

So

$$\hat\beta = \frac{N}{N-2}\hat\Sigma^{-1}\left[1 - \frac{N_1N_2}{N^2}(\hat\mu_2 - \hat\mu_1)^T\hat\beta\right](\hat\mu_2 - \hat\mu_1)$$

and thus $\hat\beta \propto \hat\Sigma^{-1}(\hat\mu_2 - \hat\mu_1)$.

(d) Changing the coding makes little difference to our solution to part (b). Suppose that the targets are coded as c_1 and c_2. Then

$$\tilde{\mathbf{X}}^T\mathbf{y} = \begin{pmatrix} c_1N_1 + c_2N_2 \\ c_1N_1\hat\mu_1 + c_2N_2\hat\mu_2 \end{pmatrix}.$$

Taking this through our solution we end up with

$$\left[(N-2)\hat\Sigma + N\hat\Sigma_B\right]\beta = \frac{c_1N_1 + c_2N_2}{N}(N_1\hat\mu_1 + N_2\hat\mu_2) - (c_1N_1\hat\mu_1 + c_2N_2\hat\mu_2).$$

Rearranging this gives

$$\left[(N-2)\hat\Sigma + N\hat\Sigma_B\right]\beta = \frac{(c_2 - c_1)N_1N_2}{N}(\hat\mu_2 - \hat\mu_1).$$

This implies that $\hat\beta \propto \hat\Sigma^{-1}(\hat\mu_2 - \hat\mu_1)$ as before.

(e) By part (b), $\hat\beta_0 = -\frac{1}{N}(N_1\hat\mu_1^T + N_2\hat\mu_2^T)\hat\beta$ so

$$\hat\beta_0 = -\frac{\lambda}{N}(N_1\hat\mu_1 + N_2\hat\mu_2)^T\hat\Sigma^{-1}(\hat\mu_2 - \hat\mu_1)$$

with the same constant of proportionality as $\hat\beta \propto \hat\Sigma^{-1}(\hat\mu_2 - \hat\mu_1)$. Therefore

$$\hat f(x) = \lambda\left[x - \frac{1}{N}(N_1\hat\mu_1 + N_2\hat\mu_2)\right]^T\hat\Sigma^{-1}(\hat\mu_2 - \hat\mu_1).$$

So we classify to class 2 iff

$$x^T\hat\Sigma^{-1}(\hat\mu_2 - \hat\mu_1) > \frac{1}{N}(N_1\hat\mu_1 + N_2\hat\mu_2)^T\hat\Sigma^{-1}(\hat\mu_2 - \hat\mu_1).$$

Since $N = N_1 + N_2$, this is the same as the LDA rule if $N_1 = N_2$. If they are not equal, the rules will be different in general.

Exercise 4.3

Suppose we transform the original predictors $\mathbf{X}$ to $\hat{\mathbf{Y}}$ via linear regression. In detail, let $\hat{\mathbf{Y}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = \mathbf{X}\hat{\mathbf{B}}$, where $\mathbf{Y}$ is the indicator response matrix. Similarly for any input $x \in \mathbb{R}^p$, we get a transformed vector $\hat y = \hat{\mathbf{B}}^T x \in \mathbb{R}^K$. Show that LDA using $\hat{\mathbf{Y}}$ is identical to LDA in the original space.

Solution

Let $\hat\pi_k$, $\hat\mu_k$, and $\hat\Sigma$ denote the parameter estimates for the original predictors. Clearly $\hat\pi$ is unchanged under the transformation. The new estimates of class means are

$$\frac{1}{N_k}\sum_{g_i=k}\hat y_i = \frac{1}{N_k}\sum_{g_i=k}\hat{\mathbf{B}}^T x_i = \hat{\mathbf{B}}^T\hat\mu_k.$$

Since

$$\hat\Sigma = \frac{1}{N-K}\left(\mathbf{X}^T\mathbf{X} - \sum_{k=1}^K N_k\hat\mu_k\hat\mu_k^T\right),$$

the transformed covariance estimate is

$$\frac{1}{N-K}\left(\hat{\mathbf{Y}}^T\hat{\mathbf{Y}} - \sum_{k=1}^K N_k\hat{\mathbf{B}}^T\hat\mu_k\hat\mu_k^T\hat{\mathbf{B}}\right) = \hat{\mathbf{B}}^T\hat\Sigma\hat{\mathbf{B}}.$$

Therefore the new linear discriminant functions are

$$\begin{aligned}
\delta_k(\hat y) &= (x^T\hat{\mathbf{B}})\left(\hat{\mathbf{B}}^{-1}\hat\Sigma^{-1}(\hat{\mathbf{B}}^T)^{-1}\right)(\hat{\mathbf{B}}^T\hat\mu_k) - \frac{1}{2}(\hat\mu_k^T\hat{\mathbf{B}})\left(\hat{\mathbf{B}}^{-1}\hat\Sigma^{-1}(\hat{\mathbf{B}}^T)^{-1}\right)(\hat{\mathbf{B}}^T\hat\mu_k) + \ln(\hat\pi_k) \\
&= x^T\hat\Sigma\hat\mu_k - \frac{1}{2}\hat\mu_k^T\hat\Sigma\hat\mu_k + \ln(\hat\pi_k) \\
&= \delta_k(x),
\end{aligned}$$

as required.

Exercise 4.4

Consider the multilogit model with K classes (4.17). Let β be the $(p+1)(K-1)$-vector consisting of all the coefficients. Define a suitably enlarged version of the input vector x to accommodate this vectorized coefficient matrix. Derive the Newton-Raphson algorithm for maximizing the multinomial log-likelihood, and describe how you would implement this algorithm.

Solution

Given a length p input vector x (we assume the intercept 1 is already included), let $\tilde x$ be the $(p+1)(K-1)\times(K-1)$ block matrix with $K-1$ x's along the diagonal and 0s elsewhere. Then $\tilde x^T\beta$ is the length $K-1$ vector with kth entry $\beta_k^T x$ (again we include the intercept term in β_k). From this the vector of probabilities $p_k(x;\theta)$ can easily be calculated.

We now derive the score equations and Newton-Raphson algorithm. Let $y_i = (y_{i0},\dots,y_{iK})^T$ denote the ith row of the indicator response matrix $\mathbf{Y}$. The multinomial log-likelihood is

$$\begin{aligned}
l(\beta) &= \sum_{i=1}^N\sum_{k=1}^K y_{ik}\ln(p_k(x_i;\theta)) \\
&= \sum_{i=1}^N\left[\sum_{k=1}^{K-1}\left[y_{ik}\left(\beta_k^Tx_i - \ln(1 + \sum_{l=1}^{K-1}\exp(\beta_l^Tx_i))\right)\right] - (1 - \sum_{l=1}^{K-1}y_{il})\ln(1 + \sum_{l=1}^{K-1}\exp(\beta_l^Tx_i))\right] \\
&= \sum_{i=1}^N\left[\sum_{k=1}^{K-1}\left(y_{ik}\beta_k^Tx_i\right) - \ln(1 + \sum_{l=1}^{K-1}\exp(\beta_l^Tx_i))\right].
\end{aligned}$$

Differentiating this with respect to one of the β_k gives

$$\begin{aligned}
\frac{\partial l(\beta)}{\partial\beta_k} &= \sum_{i=1}^N\left[y_{ik}x_i - \frac{\exp(\beta_k^Tx)}{1 + \sum_{l=1}^{K-1}\exp(\beta_l^Tx_i)}x_i\right] \\
&= \sum_{i=1}^N(y_{ik} - p_k(x_i;\theta))x_i
\end{aligned}$$

and differentiating again yields

$$\frac{\partial^2 l(\beta)}{\partial\beta_k\partial\beta_l^T} = \begin{cases} -\sum_{i=1}^N p_k(x_i;\theta)p_l(x_i;\theta)x_ix_i^T & \text{if } k \neq l \\ -\sum_{i=1}^N p_k(x_i;\theta)(1 - p_k(x_i;\theta))x_ix_i^T & \text{if } k = l \end{cases}$$

This gives us enough to implement Newton-Raphson: concatenate the $\frac{\partial l(\beta)}{\partial\beta_k}$ and $\frac{\partial^2 l(\beta)}{\partial\beta_k\partial\beta_l^T}$ into a single vector and block matrix $\frac{\partial l(\beta)}{\partial\beta}$ and $\frac{\partial^2 l(\beta)}{\partial\beta\partial\beta^T}$, respectively. Then set

$$\beta^{\text{new}} = \beta^{\text{old}} - \left(\frac{\partial^2 l(\beta)}{\partial\beta\partial\beta^T}\right)^{-1}\frac{\partial l(\beta)}{\partial\beta}.$$

We will now describe how to use the enlarged vector $\tilde x$ to put everything in terms of matrix products. The idea is to vectorise natural matrices like $\mathbf{Y}$ in a similar fashion to β.

Let $\tilde{\mathbf{X}}$ denote the $(k-1)N\times(p+1)(K-1)$-matrix obtained by stacking $\tilde x_1^T,\dots,\tilde x_N^T$ on top of each other, or equivalently obtained from $\mathbf{X}$ by replacing each x_i^T with its tilde counterpart. Let $\tilde{\mathbf{y}}$ be the length $(k-1)N$ vector obtained by stacking the observations y_i on top of each other in order. Similarly, let $\mathbf{p}_i$ denote the length $K-1$ vector with entries $p_1(x_i;\theta),\dots p_{K-1}(x_i;\theta)$ and let $\tilde{\mathbf{p}}$ be the length $(k-1)N$ vector obtained by stacking the $\mathbf{p}_i$. Then

$$\frac{\partial l(\beta)}{\partial\beta} = \sum_{i=1}^N\tilde x_i(y_i - \mathbf{p}_i) = \tilde{\mathbf{X}}^T(\tilde{\mathbf{y}} - \tilde{\mathbf{p}})$$

For $i \in \{1,\dots,N\}$, let $\mathbf{W}_i$ denote the $(K-1)\times(K-1)$ matrix with (k,l) entry $p_k(x_i;\theta)p_l(x_i;\theta)$ if $k \neq l$ and $p_k(x_i;\theta)(1 - p_k(x_i;\theta)$ if $k = l$. Then

$$\frac{\partial^2 l(\beta)}{\partial\beta\partial\beta^T} = \sum_{i=1}^N\tilde x_i\mathbf{W}_i\tilde x_i^T = \tilde{\mathbf{X}}^T\mathbf{W}\tilde{\mathbf{X}},$$

where $\mathbf{W}$ is the $(K-1)N\times(K-1)N$ block matrix with $\mathbf{W}_1,\dots,\mathbf{W}_N$ along the diagonal.

With this in hand, the Newton-Raphson algorithm can be expressed as an iterated reweighted least squares procedure as in the $K = 2$ case:

$$\beta^{\text{new}} = (\tilde{\mathbf{X}}^T\mathbf{W}\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\mathbf{W}\tilde{\mathbf{z}}, \qquad \text{where} \qquad \tilde{\mathbf{z}} = \tilde{\mathbf{X}}\beta^{\text{old}} + \mathbf{W}^{-1}(\tilde{\mathbf{y}} - \tilde{\mathbf{p}})$$

This can be implemented directly, updating $\mathbf{W}$ and $\tilde{\mathbf{p}}$ at each step, or as the solution to the weighted least squares problem

$$\min_\beta(\tilde{\mathbf{z}} - \tilde{\mathbf{X}}\beta)^T\mathbf{W}(\tilde{\mathbf{z}} - \tilde{\mathbf{X}}\beta).$$

Exercise 4.5

Consider a two-class logistic regression problem with $x \in \mathbb{R}$. Characterize the maximum-likelihood estimates of the slope and intercept parameter if the sample x_i for the two classes are separated by a point $x_0 \in \mathbb{R}$. Generalize this result to **(a)** $x \in \mathbb{R}^p$ (see Figure 4.16), and **(b)** more than two classes.

Solution

In all these situations the MLEs are undefined. In the first case the log-likelihood is

$$l(\beta) = \sum_{i=1}^N\left[y_i\ln(p(x_i;\theta)) + (1 - y_i)\ln(1 - p(x_i;\theta))\right].$$

Since $p(x,\theta) \in [0,1]$, this is negative. If the two classes are separated then we can approach 0 with suitable choices of θ, but θ will diverge to infinity.

To be more precise, suppose that $y_i = 1$ if $x_i < x_0$ and $y_i = 0$ if $x_i > x_0$. Take $\theta = (\beta_0, \beta) = (-rx_0, r)$ for $r > 0$. Then $\beta_0 + \beta x > 0$ if $x > x_0$ and $\beta_0 + \beta x > 0$ if $x < x_0$. So as $r \to \infty$,

$$\beta_0 + \beta x_i \to \begin{cases} -\infty & \text{if } y_i = 1 \\ \infty & \text{if } y_i = 0 \end{cases} \implies p(x_i;\theta) \to \begin{cases} 0 & \text{if } y_i = 1 \\ 1 & \text{if } y_i = 0 \end{cases}.$$

and $l(\beta) \to 0$. Clearly this path isn't unique, for example we could take a different separating point x_0.

(a) If $x \in \mathbb{R}^p$ for $p > 1$ the situation is very similar. Suppose that the two classes are separated by a hyperplane $\tilde\beta_0 + \tilde\beta^T x = 0$ with $y_i = 1$ if $\tilde\beta_0 + \tilde\beta^T x < 0$ and $y_i = 0$ if $\tilde\beta_0 + \tilde\beta^T x > 0$. Then $\theta = (r\tilde\beta_0, r\tilde\beta)$ diverges to infinity as $r \to \infty$ but $l(\beta)$ converges to its supremum zero$.

(b) There are a few ways to conceive of generalising to $K > 2$ classes, but the point is that the log-likehihood has supremum 0 which can't be attained with finite values of θ. For a simple formulation, suppose that for $k = 1,\dots,K-1$, the kth class can be separated from *all other* classes by a hyperplane $\tilde\beta_{0k} + \tilde\beta_k^T x = 0$. More precisely, suppose that $g_i = k$ if $\tilde\beta_{0k} + \tilde\beta_k^T x > 0$ and $g_i \neq k$ if $\tilde\beta_{0k} + \tilde\beta_k^T x < 0$. Let θ have kth component $(r\tilde\beta_{0k}, r\tilde\beta_k)$. Then the log-likelihood

$$l(\beta) = \sum_{i=1}^N\sum_{k=1}^K y_{ik}\ln(p_k(x_i;\theta))$$

converges to 0 as $r \to \infty$ but θ diverges to infinity.

Note that this means that logistic regression fails for arguably the *simplest* type of classification problem. This motivates separating hyperplanes.

Exercise 4.6

Suppose we have N points x_i in $\mathbb{R}^p$ in general position, with class labels $y_i \in \{-1, 1\}$. Prove that the perceptron learning algorithm converges to a separating hyperplane in a finite number of steps:

(a) Denote a hyperplane by $f(x) = \beta_1^T + \beta_0 = 0$, or in more compact notation $\beta^T x^* = 0$, where $x^* = (x, 1)$ and $\beta = (\beta_1, \beta_0)$. Let $z_i = x_i^*/\|x_i^*\|$. Show that separability implies the existence of a β_{sep} such that $y_i\beta_{\text{sep}}^T z_i \geq 1\ \forall i$

(b) Give a current β_{old}, the perceptron algorithm identifies a point z_i that is misclassified, and produces the update $\beta_{\text{new}} \leftarrow \beta_{\text{old}} + y_iz_i$. Show that $\|\beta_{\text{new}} - \beta_{\text{sep}}\|^2 \leq \|\beta_{\text{old}} - \beta_{\text{sep}}\|^2 - 1$, and hence the algorithm converges to a separating hyperplane in no more than $\|\beta_{\text{start}} - \beta_{\text{sep}}\|^2$ steps (Ripley, 1996).

Solution

(a) If the data are separable then there exists a β such that $\beta^T x_i > 0$ if $y_i = 1$ and $\beta^T x_i < 0$ if $y_i = -1$. Now it is just a case of scaling β.

(b) Note that this is just stochastic gradient descent with a changing learning rate ρ. We have

$$\begin{aligned}
\|\beta_{\text{new}} - \beta_{\text{sep}}\|^2 &= \|\beta_{\text{old}} + y_iz_i - \beta_{\text{sep}}\|^2 \\
&= \|\beta_{\text{old}} - \beta_{\text{sep}}\|^2 + 2\langle y_iz_i, \beta_{\text{old}} - \beta_{\text{sep}}\rangle + \|y_iz_i\|^2 \\
&= \|\beta_{\text{old}} - \beta_{\text{sep}}\|^2 + 2\langle y_iz_i, \beta_{\text{old}}\rangle - 2\langle y_iz_i, \beta_{\text{sep}}\rangle + 1.
\end{aligned}$$

By the definition of β_{sep}, $\langle y_iz_i, \beta_{\text{sep}}\rangle \geq 1$ and since the point z_i is misclassified $\langle y_iz_i, \beta_{\text{old}}\rangle < 0$. This implies that

$$\|\beta_{\text{new}} - \beta_{\text{sep}}\|^2 \leq \|\beta_{\text{old}} - \beta_{\text{sep}}\|^2 - 1$$

and so the algorithm will converge in no more than $\|\beta_{\text{start}} - \beta_{\text{sep}}\|^2$ steps.

Exercise 4.7

Consider the criterion

$$D^*(\beta, \beta_0) = -\sum_{i=1}^N y_i(x_i^T\beta + \beta_0),$$

a generalization of (4.41) where we sum over all the observations. Consider minimizing D^* subject to $\|\beta\| = 1$. Describe this criterion in words. Does it solve the optimal separating hyperplane problem?

Solution

This is minimising a sum of signed distances to the separating hyperplane, where the sign is positive for misclassified points and negative for correctly classified points. First of all note that the problem isn't well-posed unless there are the same number of observations of each class. If not then the coefficient of β_0 is non-zero so we are minimising a linear function over $\mathbf{R}$ and the infimum is $-\infty$. It makes more sense to remove β_0, use D^* to find the direction β, and choose β_0 another way.

If we remove β_0 (or there are the same number of observations of each class) then we get $D^* = -\sum y_ix_i^T\beta$. This has solution

$$\beta = \frac{1}{N}\left(\sum_{i=1}^N y_ix_i\right)\Big/\left\|\frac{1}{N}\sum_{i=1}^N y_ix_i\right\|.$$

So β is a signed average of the observations. Both this and an optimal separating hyperplane might be said to be trying to find a hyperplane 'half-way' between the classes, but an optimal separating hyperplane looks for something halfway between the boundaries of the classes, whereas this considers all observations equally.

Exercise 4.8

Consider the multivariate Gaussian model $X|G = k \sim N(\mu_k, \Sigma)$, with the additional restriction that $\text{rank}\{\mu_k\}_1^K = L < \max(K-1, p)$. Derive the constrained MLEs for the μ_k and Σ. Show that the Bayes classification rule is equivalent to classifying in the reduced subspace computed by LDA (Hastie and Tibshirani, 1996b).

In [ ]:

Exercise 4.9

Write a computer program to perform a quadratic discriminant analysis by fitting a separate Gaussian model per class. Try it out on the vowel data, and compute the misclassification error for the test data. The data can be found in the book website www-stat.stanford.edu/ElemStatLearn.

Solution

Our implementation of quadratic discriminant analysis is contained in the module `qda.py`. It consists of estimating class priors $\hat\pi_k$, class means $\hat\mu_k$, and class covariances $\hat\Sigma_k$ for each $k = 1,\dots,K$, and using quadratic discriminant functions $\delta_k(x)$ to classify a data point x.

To implement this we vectorise, calculating the matrix with (i,k) entry $\delta_k(x_i)$ all at once. To calculate a fixed δ_k acting on a whole matrix $\mathbf{X}$ we begin by computing the eigendecomposition $\hat\Sigma_k = \mathbf{U}_k\mathbf{D}_k\mathbf{U}_k^T$. To obtain a matrix with ith row $(x_i - \hat\mu_k)^T\hat\Sigma_k^{-1}(x_i - \hat\mu_k)$ we subtract $\hat\mu_k$ from each row of $\mathbf{X}$, multiply by $\mathbf{U}_k\mathbf{D}_k^{-1/2}$, and then take the square norm of each row. We apply linear adjustments to get $\delta_k(x_i)$ for all i. We use three dimensional arrays to do this simultaneously for all k.

Now we try the module out on the vowel data.

In [1]:
```
# Packages
import numpy as np
import pandas as pd
import qda
```

We begin by importing and organising the data.

In [2]:
```
# Import data
train = pd.read_csv('vowel_train.csv', header=None, index_col=0)
test = pd.read_csv('vowel_test.csv', header=None, index_col=0)

# Split into inputs and outputs
y_train = train.iloc[:, 0].to_numpy()
X_train = train.iloc[:, 1:].to_numpy()

y_test = test.iloc[:, 0].to_numpy()
X_test = test.iloc[:, 1:].to_numpy()

# Some useful constants
N_train, p = X_train.shape
N_test = len(y_test)
print('{} training samples'.format(N_train))
print('{} test samples'.format(N_test))
print('{} features'.format(p))

# Class labels
print('Class labels: ', np.sort(np.unique(y_train)))
```

```
528 training samples
462 test samples
10 features
Class labels:  [ 1  2  3  4  5  6  7  8  9 10 11]
```

We generate codes for each class to map them to $0,\dots,K-1$

In [3]:
```
# Class labels
K = len(np.unique(y_train))
print('{} classes'.format(K))
print('Class labels: ', np.sort(np.unique(y_train)))

# Generating coding for classes and dictionaries to pass between them
code_to_class, class_to_code = qda.gen_class_codes(y_train)

# Print codes
print('Class coding: ', code_to_class)

# Transform output data with this encoding
y_train_coded = qda.encode_classes(y_train, class_to_code)
y_test_coded = qda.encode_classes(y_test, class_to_code)
```

```
11 classes
Class labels:  [ 1  2  3  4  5  6  7  8  9 10 11]
Class coding:  {0: 1, 1: 2, 2: 3, 3: 4, 4: 5, 5: 6, 6: 7, 7: 8, 8: 9, 9: 10, 10: 11}
```

In this case the coding is simple: each class label is mapped to the number one less.

Now we train the K Guassian models and calculate the training and test classification errors.

In [10]:
```
# Generate indicator response matrix for training responses
Y_train_coded = qda.gen_indicator_responses(y_train_coded, K)

# Estimate parameters
prior_prob, sample_means, sample_vars = qda.est_params(X_train, Y_train_coded)

# Classification error for training data
y_train_est_coded = qda.classify(X_train, prior_prob, sample_means, sample_vars)
train_err = qda.classification_error(y_train_coded, y_train_est_coded)
print('Training classification error: {:.4f}'.format(train_err))

# Classification error for test data
y_test_est_coded = qda.classify(X_test, prior_prob, sample_means, sample_vars)
test_err = qda.classification_error(y_test_coded, y_test_est_coded)
print('Test classification error: {:.4f}'.format(test_err))
```

```
Training classification error: 0.0133
Test classification error: 0.5238
```

A 52% error rate is high, but we had a relatively small training set. The info file `vowel_info.txt` gives the classification error for a variety of Neural Networks trained on the data and most have a 45-55% error rate.