A Solution Manual and Notes for the Text: The Elements of Statistical Learning by Jerome Friedman, Trevor Hastie, and Robert Tibshirani

John L. Weatherwax*

December 15, 2009

*wax@alum.mit.edu

Chapter 2 (Overview of Supervised Learning)

Notes on the Text

Statistical Decision Theory

Our expected predicted error (EPE) under the squared error loss and assuming a linear model for y i.e. $y = f(x) \approx x^T\beta$ is given by

$$\text{EPE}(\beta) = \int (y - x^T\beta)^2 \text{Pr}(dx, dy)\,. \tag{1}$$

Considering this a function of the components of β i.e. β_i to minimize this expression with respect to β_i we take the β_i derivative, set the resulting expression equal to zero and solve for β_i. Taking the vector derivative with respect to the vector β we obtain

$$\frac{\partial \text{EPE}}{\partial \beta} = \int 2\,(y - x^T\beta)\,(-1)x\,\text{Pr}(dx, dy) = -2\int (y - x^T\beta)x\text{Pr}(dx, dy)\,. \tag{2}$$

Now this expression will contain two parts. The first will have the integrand yx and the second will have the integrand $x^T\beta x$. This latter expression in terms of its components is given by

$$\begin{aligned} x^T\beta x &= (x_0\beta_0 + x_1\beta_1 + x_2\beta_2 + \cdots + x_p\beta_p) \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \\ &= \begin{bmatrix} x_0x_0\beta_0 + x_0x_1\beta_1 + x_0x_2\beta_2 + \ldots + x_0x_p\beta_p \\ x_1x_0\beta_0 + x_1x_1\beta_1 + x_1x_2\beta_2 + \ldots + x_1x_p\beta_p \\ \vdots \\ x_px_0\beta_0 + x_px_1\beta_1 + x_px_2\beta_2 + \ldots + x_px_p\beta_p \end{bmatrix} = xx^T\beta\,. \end{aligned}$$

So with this recognition, that we can write $x^T\beta x$ as $xx^T\beta$, we see that the expression $\frac{\partial \text{EPE}}{\partial \beta} = 0$ gives

$$E[yx] - E[xx^T\beta] = 0\,. \tag{3}$$

Since β is a constant, it can be taken out of the expectation to give

$$\beta = E[xx^T]^{-1}E[yx]\,, \tag{4}$$

which gives a very simple derivation of equation 2.16 in the book. Note since $y \in \mathbb{R}$ and $x \in \mathbb{R}^p$ we see that x and y commute i.e. $xy = yx$.

Exercise Solutions

Ex. 2.1 (target coding)

If each of our samples from K classes is coded as a target vector t_k which has a one in the kth spot. Then one way of developing a classifier is by regressing the independent variables onto the target vectors t_k. Then our classification procedure would then become the following. Given the measurement vector X, predict a target vector $\hat{y}$ via linear regression and to select the class k corresponding to the component of $\hat{y}$ which has the largest value. That is $k = \text{argmax}_i(\hat{y}_i)$. Now consider the expression $\text{argmin}_k||\hat{y} - t_k||$, which finds the index of the target vector that is closest to the produced regression output $\hat{y}$. By expanding the quadratic we find that

$$\begin{aligned}
\text{argmin}_k||\hat{y} - t_k|| &= \text{argmin}_k||\hat{y} - t_k||^2 \\
&= \text{argmin}_k \sum_{i=1}^{K} (\hat{y}_i - (t_k)_i)^2 \\
&= \text{argmin}_k \sum_{i=1}^{K} \left((\hat{y}_i)^2 - 2\hat{y}_i (t_k)_i + {(t_k)_i}^2 \right) \\
&= \text{argmin}_k \sum_{i=1}^{K} \left(-2\hat{y}_i (t_k)_i + {(t_k)_i}^2 \right) ,
\end{aligned}$$

since the sum $\sum_{i=1}^{K} \hat{y}_i^2$ is the same for all classes k and we have denoted $(t_k)_i$ to be the ith component of the kth target vector. Continuing with this calculation we have that

$$\begin{aligned}
\text{argmin}_k||\hat{y} - t_k|| &= \text{argmin}_k \left(-2 \sum_{i=1}^{K} \hat{y}_i (t_k)_i + \sum_{i=1}^{K} {(t_k)_i}^2 \right) \\
&= \text{argmin}_k \left(-2 \sum_{i=1}^{K} \hat{y}_i (t_k)_i + 1 \right) \\
&= \text{argmin}_k \left(-2 \sum_{i=1}^{K} \hat{y}_i (t_k)_i \right) ,
\end{aligned}$$

since the sum $\sum_{k=1}^{K} {(t_k)_i}^2 = 1$, for every class k. Thus we see that

$$\text{argmin}_k||\hat{y} - t_k|| = \text{argmax}_k \left(\sum_{i=1}^{K} \hat{y}_i (t_k)_i \right) .$$

As the target vector t_k has elements consisting of only ones and zeros such that

$$(t_k)_i = \delta_{ki} = \begin{cases} 1 & k = i \\ 0 & k \neq i \end{cases} ,$$

we see that the above becomes

$$\text{argmax}_k \left(\sum_{k=1}^{K} \hat{y}_i (t_k)_i \right) = \text{argmax}_k \left(\sum_{i=1}^{K} \hat{y}_i \delta_{ik} \right) = \text{argmax}_k (\hat{y}_k) ,$$

showing that the above formulation is equivalent to selecting the class k corresponding to component of $\hat{y}$ with the largest value.

Note: In this derivation, I don't see the need to have the elements of $\hat{y}$ sum to one.

Ex. 2.2 (the oracle reveled)

Note: I don't see anything incorrect with what I have done here but the answer derived thus far will not reproduce the decision boundary given in the text. In fact it result in a linear decision boundary. If anyone sees anything incorrect with this below please let me know.

From the book we are told how the class conditional densities are generated and using this information we can derive the Bayes' decision boundary for this problem as follows. Recognizing that if we are *given* the values of the 10, class dependent randomly drawn "centering" points $\mathbf{m}$ the observed data points for the class ω_c are generated according to the following mixture density

$$p(x|\mathbf{m},\omega_c) = \sum_{i=1}^{10} \frac{1}{10} N(x; m_i, \frac{1}{5}I) \quad \text{for} \quad c \in \{\text{GREEN}, \text{RED}\}\,.$$

Here we have denoted the values of the 10 centering points as the vector $\mathbf{m}$ when we want to consider them together or as m_i for $i = 1, 2, \cdots, 10$ when we want to consider them individually. Since these m_i are actually unknown, to evaluate the full conditional density $p(x|\omega_c)$ without knowing the values of the m_i we will marginalize them out. That is we compute

$$\begin{aligned} p(x|\omega_c) &= \int p(x|\mathbf{m},\omega_c)p(\mathbf{m}|\omega_c)d\mathbf{m} \\ &= \int \left(\sum_{i=1}^{10} \frac{1}{10} N(x; m_i, \frac{1}{5}I) \right) p(\mathbf{m}|\omega_c)d\mathbf{m} \\ &= \sum_{i=1}^{10} \frac{1}{10} \int N(x; m_i, \frac{1}{5}I) N(m_i; l_c, I) dm_i\,. \end{aligned}$$

Where l_c is the mean of the normal distribution from which we draw our random sampling point for class c. That is

$$l_{\text{GREEN}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad l_{\text{RED}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\,.$$

Thus to evaluate $p(x|\omega_c)$ we now need to evaluate the product integrals

$$\int N(x; m_i, \frac{1}{5}I) N(m_i; l_c, I) dm_i\,.$$

To do this we will use a convolution like identity which holds for Gaussian density functions. This identity is given by

$$\int_y N(y-a_i;0,\Sigma_1)N(y-a_j;0,\Sigma_2)dy = N(a_i-a_j;0,\Sigma_1+\Sigma_2)\,. \tag{5}$$

Using this expression we find that the integrals above evaluate as

$$\begin{aligned}\int N(x;m_i,\frac{1}{5}I)N(m_i;l_c,I)dm_i &= \int N(x-m_i;0,\frac{1}{5}I)N(m_i-l_c;0,I)dm_i \\ &= N(x-l_c;0,\frac{1}{5}I+I) = N(x-l_c;0,\frac{6}{5}I)\,.\end{aligned}$$

Thus with this result we find our conditional densities given by

$$p(x|\omega_{\text{GREEN}}) = \frac{1}{10}\sum_{i=1}^{10} N(x-(1,0)^T;0,\frac{6}{5}I) = N(x-(1,0)^T;0,\frac{6}{5}I) \tag{6}$$

$$p(x|\omega_{\text{RED}}) = \frac{1}{10}\sum_{i=1}^{10} N(x-(0,1)^T;0,\frac{6}{5}I) = N(x-(0,1)^T;0,\frac{6}{5}I)\,. \tag{7}$$

The Bayes' decision boundary when both classes are equally probable is given by the values of x that satisfy

$$p(x|\omega_{\text{GREEN}}) = p(x|\omega_{\text{RED}})\,.$$

To plot this decision boundary we sample Equations 6 and 7, select the larger of the two and classify that point as either GREEN or RED depending.

Note: As stated above, this cannot be correct since each classes conditional density is now a Gaussian with equal covariance matrices and the optimal decision boundary is therefore a line. If someone sees what I have done incorrectly please email me.

Ex. 2.6 (forms for linear regression and k-nearest neighbor regression)

Part (a): Linear regression computes its estimate at x_0 by

$$\begin{aligned}\hat{f}(x_0) &= x_0^T\beta \\ &= x_0^T\left((X^TX)^{-1}X^Ty\right) \\ &= \left(x_0^T(X^TX)^{-1}X^T\right)y\,.\end{aligned}$$

This will be a function linear in the components of y_i if we can express the vector $x_0^T(X^TX)^{-1}X^T$ in terms of the components of X. Now let x_{ij} be the jth component variable $1 \le j \le p$ of the ith observed instance of the variable X. Then X^TX has an (i,j) element given by

$$(X^TX)_{ij} = \sum_{k=1}^{N}(X^T)_{ik}X_{kj} = \sum_{k=1}^{N}X_{ki}X_{kj} = \sum_{k=1}^{N}x_{ki}x_{kj}\,,$$

so the element of $(X^T X)^{-1}$ are given by the cofactor expansion coefficient (i.e. Crammer's rule). That is, since we know the elements of the matrix $X^T X$ in terms of the component points then $(X^T X)^{-1}$ has an ijth element given by see [3],

$$(A^{-1})_{ij} = \frac{C_{ji}}{\det(A)} ,$$

Where C_{ij} is the ijth element of the cofactor matrix

$$C_{ij} = (-1)^{i+j} \det(M_{ij}) ,$$

with M_{ij} the ijth "minor" of the matrix A, that is the matrix from A obtained by deleting the i row and the jth column from the A matrix.

$$\hat{f}(x_0) = \sum_{i=1}^{N} l_i(x_0; \mathcal{X}) y_i .$$

From the given expression since $x_0^T (X^T X)^{-1} X^T$ is a $1 \times N$ vector we see that in terms of the ith component of this vector is $(x_0^T (X^T X)^{-1} X^T)_i$ then

$$\hat{f}(x_0) = \sum_{i=1}^{N} (x_0^T (X^T X)^{-1} X^T)_i y_i ,$$

thus for linear regression we have

$$l_i(x_0; \mathcal{X}) = (x_0^T (X^T X)^{-1} X^T)_i .$$

Note that this might be able to be simplified directly into the components of the data matrix X using the cofactor expansion of the inverse $(X^T X)^{-1}$.

Chapter 3 (Linear Methods for Regression)

Notes on the Text

Linear Regression Models and Least Squares

With a data set arranged in the variables X and y, our estimate of the linear coefficients β for the model $y \approx f(x) = x^T\beta$ is given by equation 3.6 given by

$$\hat{\beta} = (X^TX)^{-1}X^Ty\,. \tag{8}$$

Then taking the expectation with respect to Y and using the fact that $E[y] = \mu_y$, we obtain

$$E[\hat{\beta}] = (X^TX)^{-1}X^T\mu_y\,. \tag{9}$$

The covariance of this estimate $\hat{\beta}$ can be computed as

$$\begin{aligned}
\mathrm{Cov}[\hat{\beta}] &= E[(\hat{\beta} - E[\hat{\beta}])(\hat{\beta} - E[\hat{\beta}])^T] = E[\hat{\beta}\hat{\beta}^T] - E[\hat{\beta}]E[\hat{\beta}]^T \\
&= E[(X^TX)^{-1}X^Tyy^TX(X^TX)^{-1}] - (X^TX)^{-1}X^T\mu_y\mu_y^TX(X^TX)^{-1} \\
&= (X^TX)^{-1}X^TE[yy^T - \mu_y\mu_y^T]X(X^TX)^{-1} \\
&= (X^TX)^{-1}X^T\left(E[yy^T] - E[y]E[y]^T\right)X(X^TX)^{-1} \\
&= (X^TX)^{-1}X^T\mathrm{Cov}[y]X(X^TX)^{-1}\,.
\end{aligned}$$

If we assume that the elements of y are uncorrelated then $\mathrm{Cov}[y] = \sigma^2 I$ and the above becomes

$$\mathrm{Cov}[\hat{\beta}] = \sigma^2(X^TX)^{-1}X^TX(X^TX)^{-1} = (X^TX)^{-1}\sigma^2\,, \tag{10}$$

which is the equation. 3.8 in the book.

Multiple Regression from Simple Univariate Regression

As stated in the text we begin with a *univariate* regression model with no intercept i.e. no β_0 term as

$$Y = X\beta + \epsilon\,.$$

The ordinary least square estimate of β are given by the normal equations or

$$\hat{\beta} = (X^TX)^{-1}X^TY\,.$$

Now since we are regressing a model with no intercept the matrix X is only a *column* matrix and the products X^TX and X^TY are scalars

$$(X^TX)^{-1} = (\sum_{i=1}^N x_i^2)^{-1} \quad \text{and} \quad X^TY = \sum_{i=1}^N x_iy_i\,,$$

so the least squares estimate of β is therefore given by

$$\hat{\beta} = \frac{\sum_{i=1}^N x_i y_i}{\sum_{i=1}^N x_i^2} = \frac{x^T y}{x^T x}\,. \tag{11}$$

Which is equation 3.24 in the book. The residuals r_i of any model are defined in the standard way and for this model become $r_i = y_i - x_i\hat{\beta}$.

When we attempt to take this example from $p = 1$ to higher dimensions, lets assume that the columns of our data matrix X are *orthogonal* that is we assume that $\langle x_j^T x_k\rangle = x_j^T x_k = 0$, for all $j \neq k$ then the outer product in the normal equations becomes quite simple

$$\begin{aligned} X^T X &= \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_p^T \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_p \end{bmatrix} \\ &= \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & \cdots & x_1^T x_p \\ x_2^T x_1 & x_2^T x_2 & \cdots & x_2^T x_p \\ \vdots & \vdots & \cdots & \vdots \\ x_p^T x_1 & x_p^T x_2 & \cdots & x_p^T x_p \end{bmatrix} = \begin{bmatrix} x_1^T x_1 & 0 & \cdots & 0 \\ 0 & x_2^T x_2 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & x_p^T x_p \end{bmatrix} = D\,. \end{aligned}$$

So using this, the estimate for β becomes

$$\hat{\beta} = D^{-1}(X^T Y) = D^{-1} \begin{bmatrix} x_1^T y \\ x_2^T y \\ \vdots \\ x_p^T y \end{bmatrix} = \begin{bmatrix} \frac{x_1^T y}{x_1^T x_1} \\ \frac{x_2^T y}{x_2^T x_2} \\ \vdots \\ \frac{x_p^T y}{x_p^T x_p} \end{bmatrix}\,.$$

And each beta is obtained as in the univariate case (see Equation 11). Thus when the feature vectors are orthogonal they have no effect on each other.

Because orthogonal inputs x_j have a variety of nice properties it will be advantageous to study how to obtain them. A method that indicates how they can be obtained can be demonstrated by considering regression onto a single intercept β_0 and a single "slope" coefficient β_1 that is our model is of the given form

$$Y = \beta_0 + \beta_1 X + \epsilon\,.$$

When we compute the least squares solution for β_0 and β_1 we find (with some simple manipulations)

$$\begin{aligned} \hat{\beta}_1 &= \frac{n\sum x_t y_t - (\sum x_t)(\sum y_t)}{n\sum x_t^2 - (\sum x_t)^2} = \frac{\sum x_t y_t - \bar{x}(\sum y_t)}{\sum x_t^2 - \frac{1}{n}(\sum x_t)^2} \\ &= \frac{\langle \mathbf{x} - \bar{x}\mathbf{1}, \mathbf{y}\rangle}{\sum x_t^2 - \frac{1}{n}(\sum x_t)^2}\,. \end{aligned}$$

See [1] and the accompanying notes for this text where the above expression is explicitly derived from first principles. Alternatively one can follow the steps above. We can write the

denominator of the above expression for β_1 as $\langle \mathbf{x} - \bar{x}\mathbf{1}, \mathbf{x} - \bar{x}\mathbf{1} \rangle$. That this is true can be seen by expanding this expression

$$\begin{aligned} \langle \mathbf{x} - \bar{x}\mathbf{1}, \mathbf{x} - \bar{x}\mathbf{1} \rangle &= \mathbf{x}^T\mathbf{x} - \bar{x}(\mathbf{x}^T\mathbf{1}) - \bar{x}(\mathbf{1}^T\mathbf{x}) + \bar{x}^2 n \\ &= \mathbf{x}^T\mathbf{x} - n\bar{x}^2 - n\bar{x}^2 + n\bar{x}^2 \\ &= \mathbf{x}^T\mathbf{x} - \frac{1}{n}(\sum x_t)^2 \,. \end{aligned}$$

Which in matrix notation is given by

$$\hat{\beta}_1 = \frac{\langle \mathbf{x} - \bar{x}\mathbf{1}, \mathbf{y} \rangle}{\langle \mathbf{x} - \bar{x}\mathbf{1}, \mathbf{x} - \bar{x}\mathbf{1} \rangle} \,, \tag{12}$$

or equation 3.26 in the book. Thus we see that obtaining an estimate of the second coefficient β_1 is really two one-dimensional regressions followed in succession. We first regress $\mathbf{x}$ onto $\mathbf{1}$ and obtain the residual $\mathbf{z} = \mathbf{x} - \bar{x}\mathbf{1}$. We next regress $\mathbf{y}$ onto this residual $\mathbf{z}$. The direct extension of these ideas results in Algorithm 3.1: Regression by Successive Orthogonalization or Gram-Schmidt for multiple regression.

Another was to view Algorithm 3.1 is to take our design matrix X, form an orthogonal basis by performing the Gram-Schmidt orthogonilization procedure (learned in introductory linear algebra classes) on its column vectors, and ending with an orthogonal basis $\{\mathbf{z}_i\}_{i=1}^p$. Then using this basis linear regression can be done simply as in the *univariate* case by by computing the inner products of $\mathbf{y}$ with $\mathbf{z}_p$ as

$$\hat{\beta}_p = \frac{\langle z_p, y \rangle}{\langle z_p, z_p \rangle} \,, \tag{13}$$

which is the books equation 3.27. Then with these coefficients we can compute predictions at a given value of $\mathbf{x}$ by first computing the coefficient of $\mathbf{x}$ in terms of the basis $\{\mathbf{z}_i\}_{i=1}^p$ (as $\mathbf{z}_p^T\mathbf{x}$) and then evaluating

$$\hat{f}(\mathbf{x}) = \sum_{i=0}^{p} \hat{\beta}_i (\mathbf{z}_i^T \mathbf{x}) \,.$$

Exercise Solutions

Ex. 3.5 (an equivalent problem to ridge regression)

Consider that the ridge expression given can be written as (inserting a zero as $\bar{x}_j - \bar{x}_j$

$$\sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} \bar{x}_j\beta_j - \sum_{j=1}^{p} (x_{ij} - \bar{x}_j)\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{14}$$

We see that by defining

$$\beta_0^c = \beta_0 + \sum_{j=1}^{p} \bar{x}_j\beta_j \tag{15}$$

$$\beta_j^c = \beta_i \qquad i = 1, 2, \ldots, p \tag{16}$$

The above can be recast as

$$\sum_{i=1}^{N}(y_i - \beta_0^c - \sum_{j=1}^{p}(x_{ij} - \bar{x}_j)\beta_j^c)^2 + \lambda\sum_{j=1}^{p}\beta_j^{c2} \tag{17}$$

The equivalence of the minimization results from the fact that if β_i minimize its respective functional the β_i^c's will do the same.

A heuristic understanding of this procedure can be obtained by recognizing that by shifting the x_i's to have zero mean we have translated all points to the origin. As such only the "intercept" of the data or β_0 is modified the "slope's" or β_j^c for $i = 1, 2, \ldots, p$ are not modified.

Ex. 3.6 (the ridge regression estimate)

Note: I used the notion in original problem in [2] that has τ^2 rather than τ as the variance of the prior. Now from Bayes' rule we have

$$\begin{aligned} p(\beta|\mathcal{D}) &\propto p(\mathcal{D}|\beta)p(\beta) &(18)\\ &= \mathcal{N}(y - X\beta, \sigma^2 I)\mathcal{N}(0, \tau^2 I) &(19)\end{aligned}$$

Now from this expression we calculate

$$\begin{aligned} \log(p(\beta|\mathcal{D})) &= \log(p(\mathcal{D}|\beta)) + \log(p(\beta)) &(20)\\ &= C - \frac{1}{2}\frac{(y - X\beta)^T(y - X\beta)}{\sigma^2} - \frac{1}{2}\frac{\beta^T\beta}{\tau^2} &(21)\end{aligned}$$

here the constant C is independent of β. The mode and the mean of this distribution (with respect to β) is the argument that maximizes this expression and is given by

$$\hat{\beta} = \text{ArgMin}(-2\sigma^2 \log(p(\beta|\mathcal{D})) = \text{ArgMin}((y - X\beta)^T(y - X\beta) + \frac{\sigma^2}{\tau^2}\beta^T\beta) \tag{22}$$

Since this is the equivalent to Equation 3.43 page 60 in [2] with the substitution $\lambda = \frac{\sigma^2}{\tau^2}$ we have the requested equivalence.

Ex. 3.10 (ordinary least squares to implement ridge regression)

Consider the input centered data matrix X (of size pxp) and the output data vector Y both appended (to produce the new variables $\hat{X}$ and $\hat{Y}$) as follows

$$\hat{X} = \begin{bmatrix} X \\ \sqrt{\lambda} I_{\text{pxp}} \end{bmatrix} \tag{23}$$

and

$$\hat{Y} = \begin{bmatrix} Y \\ \mathcal{O}_{\text{px1}} \end{bmatrix} \tag{24}$$

with I_{pxp} and $\mathcal{O}_{\text{pxp}}$ the pxp identity and px1 zero column respectively. The the classic least squares solution to this *new* problem is given by

$$\hat{\beta}_{\text{LS}} = (\hat{X}^T \hat{X})^{-1} \hat{X}^T \hat{Y} \tag{25}$$

Performing the block matrix multiplications required by this expression we see that

$$\hat{X}^T \hat{X} = \begin{bmatrix} X^T \sqrt{\lambda} I_{\text{pxp}} \end{bmatrix} \begin{bmatrix} X \\ \sqrt{\lambda} I_{\text{pxp}} \end{bmatrix} = X^T X + \lambda I_{\text{pxp}} \tag{26}$$

and

$$\hat{X}^T \hat{Y} = \begin{bmatrix} X^T \sqrt{\lambda} \end{bmatrix} \begin{bmatrix} Y \\ \mathcal{O}_{\text{px1}} \end{bmatrix} = X^T Y \tag{27}$$

Thus equation 25 becomes

$$\hat{\beta}_{\text{LS}} = (X^T X + \lambda I_{\text{pxp}})^{-1} X^T Y \tag{28}$$

This expression we recognize as the solution to the regularized least squares proving the equivalence.

Chapter 4 (Linear Methods for Classification)

Notes on the Text

Logistic Regression

From the given specification of logistic regression we have

$$\begin{aligned}
\log\left(\frac{\Pr(G=1|X=x)}{\Pr(G=K|X=x)}\right) &= \beta_{10}+\beta_1^T x \\
\log\left(\frac{\Pr(G=2|X=x)}{\Pr(G=K|X=x)}\right) &= \beta_{20}+\beta_2^T x \\
&\vdots \\
\log\left(\frac{\Pr(G=K-1|X=x)}{\Pr(G=K|X=x)}\right) &= \beta_{(K-1)0}+\beta_{K-1}^T x\,.
\end{aligned} \tag{29}$$

The reason for starting with expressions of this form will become more clear when we look at the log-likelihood that results when we use the *multinomial* distribution for the distribution satisfied over the class of each sample once the probabilities $\Pr(G=k|X=x)$ are specified. Before we discuss that, however, lets manipulate the Equations 29 above by taking the the exponential of both sides and multiplying everything by $\Pr(G=K|X=x)$. When we do this we find that these equations transform into

$$\begin{aligned}
\Pr(G=1|X=x) &= \Pr(G=K|X=x)\exp(\beta_{10}+\beta_1^T x) \\
\Pr(G=2|X=x) &= \Pr(G=K|X=x)\exp(\beta_{20}+\beta_2^T x) \\
&\vdots \\
\Pr(G=K-1|X=x) &= \Pr(G=K|X=x)\exp(\beta_{(K-1)0}+\beta_{(K-1)}^T x)\,.
\end{aligned} \tag{30}$$

Adding the value of $\Pr(G=K|X=x)$ to both sides of the sum of all of the above equations and enforcing the constraint that $\sum_l \Pr(G=l|X=x)=1$, we find

$$\Pr(G=K|X=x)\left(1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l^T x)\right)=1\,.$$

On solving for $\Pr(G=K|X=x)$ we find

$$\Pr(G=K|X=x)=\frac{1}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l^T x)}\,. \tag{31}$$

When we put this expression in the proceeding $K-1$ in Equations 30 we find

$$\Pr(G=k|X=x)=\frac{\exp(\beta_{k0}+\beta_k^T x)}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l^T x)}\,, \tag{32}$$

which are equations 4.18 in the book.

WWX: Note below this point these notes have not been proofed

Fitting Logistic Regression

In the case where we have only two classes, for the ith sample *given* the feature vector $X = x_i$ the probability of that this sample comes from either of the two classes is given by the two values of the posteriori probabilities. That is given x_i the probability we are looking at a sample from the first class is $\Pr(G = 1|X = x)$, and from the second class is $\Pr(G = 2|X = x) = 1 - \Pr(G = 1|X = x)$. If for each sample x_i for $i = 1, 2, \cdots, N$ in our training set we include with the measurment vector x_i a "coding" variable denoted y_i, that takes the value 1 if the ith item comes from the first class and is zero otherwise we can sussinctly represent the probability that x_i is a member of its class with the following notation

$$p_{g_i}(x_i) = \Pr(G = 1|X = x_i)^{y_i} \Pr(G = 2|X = x)^{1-y_i} . \tag{33}$$

Since only one of the values y_i or $1 - y_i$ will infact be non-zero. Using this notation given an entire data set of measurments and thier class encoding $\{x_i, y_i\}$ the *total* likelihood of this data set is given by

$$L = \prod_{i=1}^{N} p_{g_i}(x_i) ,$$

the log-liklihood for this set of data is then given by taking the logarithm of this expression as

$$l = \sum_{i=1}^{N} \log(p_{g_i}(x_i)) .$$

When we put in the expression for $p_{g_i}(x_i)$ defined in Equation 33 we obtain

$$\begin{aligned} l &= \sum_{i=1}^{N} y_i \log(p_{g_i}(x_i)) + (1 - y_i) \log(1 - p_{g_i}(x_i)) \\ &= \sum_{i=1}^{N} y \log(\frac{p_{g_i}(x_i)}{1 - p_{g_i}(x_i)}) + \log(1 - p_{g_i}(x_i)) \end{aligned}$$

If we now use Equations 29 to express the log-posteriori odds in terms of the parameters we desire to estimate β we see that

$$\log(\frac{p_{g_i}(x_i)}{1 - p_{g_i}(x_i)}) = \beta 10 + \beta_1^T = \beta^T \mathbf{x} ,$$

and

$$\log(1 - p_{g_i}(x_i)) = \frac{1}{1 + e^{\beta^T \mathbf{x}}} .$$

Here we have extended the definition of the vector x to include a constant value of one to deal naturally with the constant value term β_0. Thus in terms of β the log-likelihood becomes

$$l(\beta) = \sum_{i=1}^{N} \left(y_i \beta^T x_i - \log(1 + e^{\beta^T x_i}) \right) .$$

Now to maximize the log-likelihood over our parameters β we need to take the derivative of l with respect to β. We find

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} \left(y_i x_i - \frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}} x_i \right) .$$

Since $p(x_i) = \frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}$ the score (or derivative of l with respect to β becomes

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i (y_i - p(x_i)) .$$

The remaining derivations presented in the book appear reasonable.

Exercise Solutions

Ex. 4.1 (a constraned maximization problem)

This problem is not finished!

To solve constainted maximization or minimization problems we want to use the idea of Legrangian multipliers. Define the lagragian $\mathcal{L}$ as

$$\mathcal{L}(a; \lambda) = a^T B a + \lambda (a^T W a - 1) .$$

Here λ is the lagrange multipler. Taking the a derivative of this expression and setting it equal to zeros gives

$$\frac{\partial \mathcal{L}(a; \lambda)}{\partial a} = 2Ba + \lambda(2Wa) = 0 .$$

This last equation is equivalent to

$$Ba + \lambda W a = 0 ,$$

or multiplying by W^{-1} on both sides and moving the expression with B to the left hand side gives the

$$W^{-1} B a = \lambda a ,$$

Notice this is a *standard* eigenvalue problem, in that the solution vectors a must be an eigenvector of the matrix $W^{-1}B$ and λ is its corresponding eigenvalue.

WWX: Note above this point these notes have not been proofed

Ex. 4.4 (mulidimensional logistic regression)

In the case of $K > 2$ classes, in the same way as discussed in the section on fitting a logistic regresion model, for each sample point with a given measurement vector $\mathbf{x}$ (here we are

implicitly considering one of the samples from our training set) we will associate a position coded responce vector variable $\mathbf{y}$ of size $K-1$ where the l-th component of $\mathbf{y}$ are equal to one if this sample is drawn from the l-th class and zero otherwise. That is

$$y_l = \begin{cases} 1 & \mathbf{x} \text{ is from class } l \\ 0 & \text{otherwise} \end{cases}$$

With this notation the likelihood that this particular measured vector $\mathbf{x}$ is from its known class can be written as

$$\begin{aligned} p_{\mathbf{y}}(\mathbf{x}) &= \Pr(G=1|X=x)^{y_1}\Pr(G=2|X=x)^{y_2}\cdots\Pr(G=K-1|X=x)^{y_{K-1}} \\ &\times (1-\Pr(G=1|X=x)-\Pr(G=2|X=x)-\cdots-\Pr(G=K-1|X=x))^{1-\sum_{l=1}^{K-1} y_l}\,. \end{aligned}$$

Since this expression is for *one* data point the log-likelihood for an entire data set will be given by

$$l = \sum_{i=1}^{N} \log(p_y(x_i))$$

Using the Equation 34 in the above expression we find $\log(p_y(x))$ is given by

$$\begin{aligned} \log(g_y(x)) &= y_1\log(\Pr(G=1|X=x)) + y_2\log(\Pr(G=2|X=x)) + \cdots + y_{K-1}\log(\Pr(G=K-1|X=x \\ &+ (1-y_1-y_2-\cdots-y_{K-1})\log(\Pr(G=K|X=x)) \\ &= \log(\Pr(G=K|X=x)) + y_1\log(\frac{\Pr(G=1|X=x)}{\Pr(G=K|X=x)}) + y_2\log(\frac{\Pr(G=2|X=x)}{\Pr(G=K|X=x)}) + \cdots \\ &+ y_{K-1}\log(\frac{\Pr(G=K-1|X=x)}{\Pr(G=K|X=x)}) \\ &= \log(\Pr(G=K|X=x)) + y_1(\beta_{01}+\beta_1^T x) + y_2(\beta_{02}+\beta_2^T x) + \cdots \\ &+ y_{K-1}(\beta_{(K-1)0}+\beta_{K-1}^T x)\,. \end{aligned}$$

Chapter 10 (Boosting and Additive Trees)

Ex. 10.1 (deriving the β update equation)

From the book we have that for a fixed β the solution $G_m(x)$ is given by

$$G_m = \text{ArgMin}_G \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i)) \,,$$

which states that we should select our classifier G_m such that $G_m(x_i) = y_i$ for the largest weights $w_i^{(m)}$ values, effectively "nulling" these values out. Now in AdaBoost this is done by selecting the training samples according to a discrete distribution $w_i^{(m)}$ specified on the training data. Since $G_m(x)$ is then specifically trained using these samples we expect that it will correctly classify many of these points. Thus lets select the $G_m(x)$ that appropriately minimizes the above expression. Once this $G_m(x)$ has been selected we now seek to minimize our exponential error with respect to the β parameter.

Then by considering Eq. 10.11 (rather than the recommended expression) with the derived G_m we have

$$(e^{\beta} - e^{-\beta}) \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G_m(x_i)) + e^{-\beta} \sum_{i=1}^{N} w_i^{(m)}$$

Then to minimize this expression with respect to β, we will take the derivative with respect to β, set the resulting expression equal to zero and solve for β. The derivative (and setting our expression equal to zero) we find that

$$(e^{\beta} + e^{-\beta}) \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G_m(x_i)) - e^{-\beta} \sum_{i=1}^{N} w_i^{(m)} = 0 \,.$$

To facilitate solving for β we will multiply the expression above by e^{β} to give

$$(e^{2\beta} + 1) \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G_m(x_i)) - \sum_{i=1}^{N} w_i^{(m)} = 0 \,.$$

so that $e^{2\beta}$ is given by

$$e^{2\beta} = \frac{\sum_{i=1}^{N} w_i^{(m)} - \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G_m(x_i))} \,.$$

Following the text we can define the error at the m-th stage (err_m) as

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i^{(m)}} \,,$$

so that in terms of this expression $e^{2\beta}$ becomes

$$e^{2\beta} = \frac{1}{\text{err}_m} - 1 = \frac{1 - \text{err}_m}{\text{err}_m} \,.$$

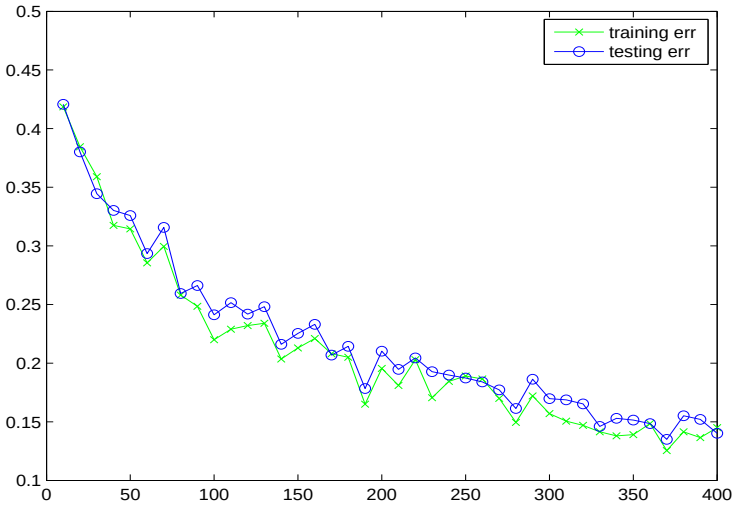

Figure 1: A duplication of the Figure 10.2 from the book.

Finally we have that β is given by

$$\beta = \frac{1}{2}\log(\frac{1-\text{err}_m}{\text{err}_m})\,,$$

which is the expression Eq. 10.12 as desired.

Ex. 10.4 (Implementing AdaBoost with trees)

Part (a): Please see the web site for a suite of codes that implement AdaBoost with trees. These codes were written by Kelly Wallenstein under my guidance.

Part (b): Please see the Figure 1 for a plot of the training and test error using the provide AdaBoost Matlab code and the suggested data set. We see that the resulting plot looks very much like the on presented in Figure 10.2 of the book helping to verify that the algorithm is implemented correctly.

Part (c): I found that the algorithm proceeded to run for as long as I was able to wait. For example, Figure 1 has 800 boosting iterations which took about an hour train and test with the Matlab code. As the number of boosting iterations increased I did not notice any significant rise in the test error. This was one of the proported advantages of the AdaBoost algorithm.

References

[1] B. Abraham and J. Ledolter. *Statistical Methods for Forecasting*. Wiley, Toronto, 1983.

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

[3] G. Strang. *Linear Algebra and Its Applications*. Brooks/Cole, 3 edition, 1988.