Springer Series in Statistics

Trevor Hastie
Robert Tibshirani
Jerome Friedman

The Elements of Statistical Learning

Data Mining, Inference, and Prediction

Second Edition

Springer

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

John L. Weatherwax*
David Epstein†

21 June 2013

Introduction

The Elements of Statistical Learning is an influential and widely studied book in the fields of machine learning, statistical inference, and pattern recognition. It is a standard recommended text in many graduate courses on these topics. It is also very challenging, particularly if one faces it without the support of teachers who are expert in the subject matter. For various reasons, both authors of these notes felt the need to understand the book well, and therefore to produce notes on the text when we found the text difficult at first reading, and answers to the exercises. Gaining understanding is time-consuming and intellectually demanding, so it seemed sensible to record our efforts in LaTeX, and make it available on the web to other readers. A valuable by-product of writing up mathematical material, is that often one finds gaps and errors in what one has written.

Now it is well-known in all branches of learning, but in particular in *mathematical* learning, that the way to learn is to *do*, rather than to *read*. It is all too common to read through some material, convince oneself that one understands it well, but then find oneself at sea when trying to apply it in even a slightly different situation. Moreover, material understood only at a shallow level is easily forgotten.

It is therefore our *strong* recommendation that readers of the book should not look at our responses to any of the exercises before making a substantial effort to understand it without this aid. Any time spent in this way, even if it ends without success, will make our solutions

*wax@alum.mit.edu
†David.Epstein@warwick.ac.uk

easier to understand and more memorable. Quite likely you will find better solutions than those provided here—let us know when you do!

As far as teachers of courses based on the text are concerned, our notes may be regarded as a disadvantage, because it may be felt that the exercises in the book can no longer be used as a source of homework or questions for tests and examinations. This is a slight conflict of interest between teachers of courses on the one hand, and independent learners on the other hand. Finding ourselves in the ranks of the independent learners, the position we take is hardly surprising. Teachers of courses might benefit by comparing their own solutions with ours, as might students in classes and independent learners. If you, the reader, find a problem difficult and become stuck, our notes might enable you to unstick yourself. In addition, there is a myriad of materials on any of the topics this book covers. A search for "statistical learning theory" on Google (as of 1 January 2010) gave over 4 million hits. The possible disadvantage of not being able to use the book's problems in an academic course is not really such a large one. Obtaining additional supplementary problems at the right level for an academic course is simply a matter of a visit to the library, or a little time spent surfing the net. And, of course, although one is not supposed to say this, teachers of courses can also get stuck.

Acknowledgments

We are hoping that people will find errors, offer insightful suggestions, and generally improve the understanding of the text, the exercises and of our notes, and that they write to us about this. We will incorporate additions that we think are helpful. Our plan is to gradually add material as we work through the book. Comments and criticisms are always welcome. Special thanks to: Ruchi Dhiman for his comments on chapter 4.

We use the numbering found in the on-line (*second edition*) version of this text. The first edition should be similar but may have some differences.

Chapter 2 (Overview of Supervised Learning)

Statistical Decision Theory

We assume a linear model: that is we assume $y = f(x) + \varepsilon$, where ε is a random variable with mean 0 and variance σ^2, and $f(x) = x^T\beta$. Our expected predicted error (EPE) under the squared error loss is

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

We regard this expression as a function of β, a column vector of length $p + 1$. In order to find the value of β for which it is minimized, we equate to zero the vector derivative with respect to β. We have

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

Now this expression has two parts. The first has integrand yx and the second has integrand $(x^T\beta)x$.

Before proceeding, we make a quick general remark about matrices. Suppose that A, B and C are matrices of size $1 \times p$ matrix, $p \times 1$ and $q \times 1$ respectively, where p and q are positive integers. Then AB can be regarded as a scalar, and we have $(AB)C = C(AB)$, each of these expressions meaning that each component of C is multiplied by the scalar AB. If $q > 1$, the expressions BC, $A(BC)$ and ABC are meaningless, and we must avoid writing them. On the other hand, CAB is meaningful as a product of three matrices, and the result is the $q \times 1$ matrix $(AB)C = C(AB) = CAB$. In our situation we obtain $(x^T\beta)x = xx^T\beta$.

From $\partial \mathrm{EPE}/\partial\beta = 0$ we deduce

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

for the particular value of β that minimizes the EPE. Since this value of 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 Equation 2.16 in the book.

We now discuss some points around Equations 2.26 and 2.27. We have

$$\hat\beta = (X^TX)^{-1}X^T\mathbf{y} = (X^TX)^{-1}X^T(X\beta + \varepsilon) = \beta + (X^TX)^{-1}X^T\varepsilon.$$

So

$$\hat y_0 = x_0^T\hat\beta = x_0^T\beta + x_0^T(X^TX)^{-1}X^T\varepsilon. \tag{5}$$

This immediately gives

$$\hat y_0 = x_0^T\beta + \sum_{i=1}^{N} \ell_i(x_0)\varepsilon_i$$

where $\ell_i(x_0)$ is the i-th element of the N-dimensional column vector $X(X^TX)^{-1}x_0$, as stated at the bottom of page 24 of the book.

We now consider Equations 2.27 and 2.28. The variation is over all training sets $\mathcal{T}$, and over all values of y_0, while keeping x_0 fixed. Note that x_0 and y_0 are chosen independently of $\mathcal{T}$ and so the expectations commute: $\mathrm{E}_{y_0|x_0}\mathrm{E}_{\mathcal{T}} = \mathrm{E}_{\mathcal{T}}\mathrm{E}_{y_0|x_0}$. Also $\mathrm{E}_{\mathcal{T}} = \mathrm{E}_{\mathcal{X}}\mathrm{E}_{\mathcal{Y}|\mathcal{X}}$.

We write $y_0 - \hat{y}_0$ as the sum of three terms

$$\left(y_0 - x_0^T\beta\right) - \left(\hat{y}_0 - \mathrm{E}_{\mathcal{T}}(\hat{y}_0)\right) - \left(\mathrm{E}_{\mathcal{T}}(\hat{y}_0) - x_0^T\beta\right) = U_1 - U_2 - U_3. \tag{6}$$

In order to prove Equations 2.27 and 2.28, we need to square the expression in Equation 6 and then apply various expectation operators. First we consider the properties of each of the three terms, U_i in Equation 6. We have $\mathrm{E}_{y_0|x_0}U_1 = 0$ and $\mathrm{E}_{\mathcal{T}}U_1 = U_1\mathrm{E}_{\mathcal{T}}$. Despite the notation, $\hat{y}_0$ does not involve y_0. So $\mathrm{E}_{y_0|x_0}U_2 = U_2\mathrm{E}_{y_0|x_0}$ and clearly $\mathrm{E}_{\mathcal{T}}U_2 = 0$. Equation 5 gives

$$U_3 = \mathrm{E}_{\mathcal{T}}(\hat{y}_0) - x_0^T\beta = x_0^T\mathrm{E}_{\mathcal{X}}\left((X^TX)^{-1}X^T\mathrm{E}_{\mathcal{Y}|\mathcal{X}}\varepsilon\right) = 0 \tag{7}$$

since the expectation of the length N vector ε is zero. This shows that the bias U_3 is zero.

We now square the remaining part of Equation 6 and then then apply $\mathrm{E}_{y_0|x_0}\mathrm{E}_{\mathcal{T}}$. The cross-term U_1U_2 gives zero, since $\mathrm{E}_{y_0|x_0}(U_1U_2) = U_2\mathrm{E}_{y_0|x_0}(U_1) = 0$. (This works in the same way if $\mathrm{E}_{\mathcal{T}}$ replaces $\mathrm{E}_{y_0|x_0}$.)

We are left with two squared terms, and the definition of variance enables us to deal immediately with the first of these: $\mathrm{E}_{y_0|x_0}\mathrm{E}_{\mathcal{T}}U_1^2 = \mathrm{Var}(y_0|x_0) = \sigma^2$. It remains to deal with the term $\mathrm{E}_{\mathcal{T}}\left(\hat{y}_0 - \mathrm{E}_{\mathcal{T}}\hat{y}_0\right)^2 = \mathrm{Var}_{\mathcal{T}}(\hat{y}_0)$ in Equation 2.27. Since the bias $U_3 = 0$, we know that $\mathrm{E}_{\mathcal{T}}\hat{y}_0 = x_0^T\beta$.

If m is the 1×1-matrix with entry μ, then mm^T is the 1×1-matrix with enty μ^2. It follows from Equation 5 that the variance term in which we are interested is equal to

$$\mathrm{E}_{\mathcal{T}}\left(x_0^T(X^TX)^{-1}X^T\varepsilon\varepsilon^TX(X^TX)^{-1}x_0\right).$$

Since $\mathrm{E}_{\mathcal{T}} = \mathrm{E}_{\mathcal{X}}\mathrm{E}_{\mathcal{Y}|\mathcal{X}}$, and the expectation of $\varepsilon\varepsilon^T$ is $\sigma^2 I_N$, this is equal to

$$\sigma^2 x_0^T\mathrm{E}_{\mathcal{T}}\left((X^TX)^{-1}\right)x_0 = \sigma^2 x_0^T\mathrm{E}_{\mathcal{X}}\left(\left(X^TX/N\right)^{-1}\right)x_0/N. \tag{8}$$

We suppose, as stated by the authors, that the mean of the distribution giving rise to X and x_0 is zero. For large N, X^TX/N is then approximately equal to $\mathrm{Cov}(X) = \mathrm{Cov}(x_0)$, the $p \times p$-matrix-variance-covariance matrix relating the p components of a typical sample vector x—as far as $\mathrm{E}_{\mathcal{X}}$ is concerned, this is a constant. Applying E_{x_0} to Equation 8 as in Equation 2.28, we obtain (approximately)

$$\begin{aligned}
\sigma^2\mathrm{E}_{x_0}\left(x_0^T\mathrm{Cov}(X)^{-1}x_0\right)/N &= \sigma^2\mathrm{E}_{x_0}\left(\mathrm{trace}\left(x_0^T\mathrm{Cov}(X)^{-1}x_0\right)\right)/N \\
&= \sigma^2\mathrm{E}_{x_0}\left(\mathrm{trace}\left(\mathrm{Cov}(X)^{-1}x_0x_0^T\right)\right)/N \\
&= \sigma^2\mathrm{trace}\left(\mathrm{Cov}(X)^{-1}\mathrm{Cov}(x_0)\right)/N \\
&= \sigma^2\mathrm{trace}(I_p)/N \\
&= \sigma^2 p/N.
\end{aligned} \tag{9}$$

This completes our discussion of Equations 2.27 and 2.28.

Notes on Local Methods in High Dimensions

The most common error metric used to compare different predictions of the true (but *unknown*) mapping *function value* $f(x_0)$ is the mean square error (MSE). The unknown in the above discussion is the specific function mapping function $f(\cdot)$ which can be obtained via different methods many of which are discussed in the book. In supervised learning to help with the construction of an appropriate prediction $\hat{y}_0$ we have access to a set of "training samples" that contains the notion of randomness in that these points are not under complete control of the experimenter. One could ask the question as to how much square error at our predicted input point x_0 will have on average when we consider all possible training sets $\mathcal{T}$. We can compute, by inserting the "expected value of the predictor obtained over all training sets", $E_{\mathcal{T}}(\hat{y}_0)$ into the definition of quadratic (MSE) error as

$$\begin{aligned}
\text{MSE}(x_0) &= E_{\mathcal{T}}[f(x_0) - \hat{y}_0]^2 \\
&= E_{\mathcal{T}}[\hat{y}_0 - E_{\mathcal{T}}(\hat{y}_0) + E_{\mathcal{T}}(\hat{y}_0) - f(x_0)]^2 \\
&= E_{\mathcal{T}}[(\hat{y}_0 - E_{\mathcal{T}}(\hat{y}_0))^2 + 2(\hat{y}_0 - E_{\mathcal{T}}(\hat{y}_0))(E_{\mathcal{T}}(\hat{y}_0) - f(x_0)) + (E_{\mathcal{T}}(\hat{y}_0) - f(x_0))^2] \\
&= E_{\mathcal{T}}[(\hat{y}_0 - E_{\mathcal{T}}(\hat{y}_0))^2] + (E_{\mathcal{T}}(\hat{y}_0) - f(x_0))^2 \,.
\end{aligned}$$

Where we have expanded the quadratic, distributed the expectation across all terms, and noted that the middle term vanishes since it is equal to

$$E_{\mathcal{T}}[2(\hat{y}_0 - E_{\mathcal{T}}(\hat{y}_0))(E_{\mathcal{T}}(\hat{y}_0) - f(x_0))] = 0\,,$$

because $E_{\mathcal{T}}(\hat{y}_0) - E_{\mathcal{T}}(\hat{y}_0) = 0$. and we are left with

$$\text{MSE}(x_0) = E_{\mathcal{T}}[(\hat{y}_0 - E_{\mathcal{T}}(\hat{y}_0))^2] + (E_{\mathcal{T}}(\hat{y}_0) - f(x_0))^2 \,. \tag{10}$$

The first term in the above expression $E_{\mathcal{T}}[(\hat{y}_0 - E_{\mathcal{T}}(\hat{y}_0))^2]$ is the *variance* of our estimator $\hat{y}_0$ and the second term $(E_{\mathcal{T}}(\hat{y}_0) - f(x_0))^2$ is the *bias* (squared) of our estimator. This notion of variance and bias with respect to our estimate $\hat{y}_0$ is to be understood relative to possible *training* sets, $\mathcal{T}$, and the specific computational method used in computing the estimate $\hat{y}_0$ given that training set.

Exercise Solutions

Ex. 2.1 (target coding)

The authors have suppressed the context here, making the question a little mysterious. For example, why use the notation $\bar{y}$ instead of simply y? We imagine that the background is something like the following. We have some input data x. Some algorithm assigns to x the probability y_k that x is a member of the k-th class. This would explain why we are told to assume that the sum of the y_k is equal to one. (But, if that reason is valid, then we should

also have been told that each $y_k \geq 0$.) In fact, neither of these two assumptions is necessary to provide an answer to the question. The hyphen in K-classes seems to be a misprint, and should be omitted.

We restate the question, clarifying it, but distancing it even further from its origins. Let $K > 0$ be an integer. For each k with $1 \leq k \leq K$, let t_k be the K-dimensional vector that has 1 in the k-th position and 0 elsewhere. Then, for any K-dimensional vector y, the k for which y_k is largest coincides with the k for which t_k is nearest to y.

By expanding the quadratic we find that

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

since the sum $\sum_{i=1}^{K} {y_i}^2$ is the same for all classes k. Notice that, for each k, the sum $\sum_{k=1}^{K} {(t_k)_i}^2 = 1$. Also $\sum y_i(t_k)_i = y_k$. This means that

$$\begin{aligned}
\text{argmin}_k||y - t_k|| &= \text{argmin}_k \left(-2y_k + 1 \right) \\
&= \text{argmin}_k(-2y_k) \\
&= \text{argmax}_k y_k \, .
\end{aligned}$$

Ex. 2.2 (the oracle revealed)

For this problem one is supposed to regard the points p_i and q_i below as *fixed*. If one does *not* do this, and instead averages over possible choices, then since the controlling points are (1,0) and (0,1), and all probabilities are otherwise symmetric between these points when one integrates out all the variation, the answer must be that the boundary is the perpendicular bisector of the interval joining these two points.

The simulation draws 10 points $p_1, \ldots, p_{10} \in \mathbb{R}^2$ from $N\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, I_2 \right)$ and 10 points $q_1, \ldots, q_{10} \in \mathbb{R}^2$ from $N\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}, I_2 \right)$. The formula for the Bayes decision boundary is given by equating likelihoods. We get an equation in the unknown $z \in \mathbb{R}^2$, giving a curve in the plane:

$$\sum_i \exp(-5\|p_i - z\|^2/2) = \sum_j \exp(-5\|q_j - z\|^2/2) \, .$$

In this solution, the boundary is given as the equation of equality between the two probabilities, with the p_i and q_j constant and fixed by previously performed sampling. Each time one re-samples the p_i and q_j, one obtains a different Bayes decision boundary.

Ex. 2.3 (the median distance to the origin)

We denote the N-tuple of data points by $(x_1, \ldots, x_N)$. Let $r_i = \|x_i\|$. Let $U(A)$ be the set of all N-tuples with $A < r_1 < \ldots < r_N < 1$. Ignoring subsets of measure zero, the set of all N-tuples is the disjoint union of the $N!$ different subsets obtained from $U(0)$ by permuting the indexing set $(1, \ldots, N)$. We will look for $A > 0$ such that the measure of $U(A)$ is half the measure of $U(0)$. The same A will work for each of our $N!$ disjoint subsets, and will therefore give the median for the distance of the smallest x_i from the origin.

We want to find A such that

$$\int_{U(A)} dx_1 \ldots dx_N = \frac{1}{2} \int_{U(0)} dx_1 \ldots dx_N.$$

We convert to spherical coordinates. Since the coordinate in the unit sphere S^{p-1} contributes the same constant on each side of the equality, obtaining

$$\int_{A<r_1<\ldots<r_N<1} r_1^{p-1} \ldots r_N^{p-1}\, dr_1 \ldots dr_n = \frac{1}{2} \int_{0<r_1<\ldots<r_N<1} r_1^{p-1} \ldots r_N^{p-1}\, dr_1 \ldots dr_n.$$

We change coordinates to $s_i = r_i^p$, and the equality becomes

$$\int_{A^p<s_1<\ldots<s_N<1} ds_1 \ldots ds_n = \frac{1}{2} \int_{0<s_1<\ldots<s_N<1} ds_1 \ldots ds_n.$$

In the left-hand integral, we change coordinates to

$$t_0 = s_1 - A^p,\ t_1 = s_2 - s_1,\ \ldots\ , t_{N-1} = s_N - s_{N-1},\ t_N = 1 - s_N.$$

The Jacobian (omitting t_0 which is a redundant variable) is a triangular matrix with -1 entries down the diagonal. The absolute value of its determinant, used in the change of variable formula for integration, is therefore equal to 1.

The region over which we integrate is

$$\sum_{i=0}^{N} t_i = 1 - A^p, \text{ with each } t_i > 0,$$

which is an N-dimensional simplex scaled down by a factor $(1 - A^p)$. The right-hand integral is dealt with in the same way, setting $A = 0$. Since the region of integration is N-dimensional, the measure is multiplied by $(1 - A^p)^N$. We solve for A by solving $(1 - A^p)^N = 1/2$. We obtain $A = (1 - 2^{-1/N})^{1/p}$, as required.

Ex. 2.4 (projections $a^T x$ are distributed as normal $N(0,1)$)

The main point is that $\sum \|x_i\|^2$ is invariant under the orthogonal group. As a consequence the standard normal distribution exists on any finite dimensional inner product space (by fixing an orthonormal basis). Further, if $\mathbb{R}^p$ is written as the orthogonal sum of two vector subspaces, then the product of standard normal distributions on each of the subspaces gives the standard normal distribution on $\mathbb{R}^p$. Everything else follows from this.

The on-line edition is correct except that $\sqrt{10} \approx 3.162278$. So, the figure given should be 3.2 instead of 3.1. Note that the version of this question posed in the first edition makes incorrect claims. The first edition talks of the "center of the training points" and the on-line edition talks of the "origin". The answers are very different. This is shown up best by taking only one training point.

Ex. 2.5 (the expected prediction error under least squares)

Part (a): In order to discuss Equation (2.27), we go back to (2.26) on page 24. We have $y = X\beta + \varepsilon$, where y and ε are $N \times 1$, X is $N \times p$, and β is $p \times 1$. Hence

$$\hat{\beta} = (X^T X)^{-1} X^T y = \beta + (X^T X)^{-1} X^T \varepsilon.$$

Since X and ε are independent variables, $\mathrm{E}_{\mathcal{T}}(\varepsilon) = 0$ and $\mathrm{E}_{\mathcal{T}}(\varepsilon^T \varepsilon) = \sigma^2 I_N$, we have $\mathrm{E}_{\mathcal{T}}(\hat{\beta}) = \beta$ and

$$\mathrm{Var}_{\mathcal{T}}(\hat{\beta}) = \mathrm{E}_{\mathcal{T}}(\hat{\beta}^T \beta) - \mathrm{E}_{\mathcal{T}}(\hat{\beta}) \mathrm{E}_{\mathcal{T}}(\hat{\beta}^T) = (X^T X)^{-1} \sigma^2.$$

Now we prove (2.27) on page 26. Note that y_0 is constant for the distribution $\mathcal{T}$. Note also that, if x_0 is held constant, $\hat{y}_0 = x_0^T \hat{\beta}$ does not depend on y_0 and so the same is true for $\mathrm{E}_{\mathcal{T}}(\hat{y}_0)$ and $\mathrm{Var}_{\mathcal{T}}(\hat{y}_0)$. Let $u = \mathrm{E}_{y_0|x_0}(y_0) = x_0^T \beta$. Then

$$\mathrm{E}_{\mathcal{T}}\left((y_0 - \hat{y}_0)^2\right) = \left(y_0^2 - u^2\right) + \left(\mathrm{E}_{\mathcal{T}}(\hat{y}_0^2) - (\mathrm{E}_{\mathcal{T}} \hat{y}_0)^2\right) + \left((\mathrm{E}_{\mathcal{T}} \hat{y}_0)^2 - 2 y_0 \mathrm{E}_{\mathcal{T}} \hat{y}_0 + u^2\right).$$

Therefore

$$\mathrm{E}_{y_0|x_0} \mathrm{E}_{\mathcal{T}}\left((y_0 - \hat{y}_0)^2\right) = \mathrm{Var}(y_0|x_0) + \mathrm{Var}_{\mathcal{T}}(\hat{y}_0) + \left(\mathrm{E}_{\mathcal{T}}(\hat{y}_0) - u\right)^2.$$

We have

$$\mathrm{E}_{\mathcal{T}}(\hat{y}_0) = x_0^T \mathrm{E}_{\mathcal{T}}(\hat{\beta}) = x_0^T \beta = u.$$

Part (b): If A is a $p \times q$ matrix and B is a $q \times p$ matrix, then it is standard that $\mathrm{trace}(AB) = \mathrm{trace}(BA)$. Note that x_0 is $p \times 1$ and X is $N \times p$, so that $x_0^T (X^T X)^{-1} x_0$ is 1×1 and is therefore equal to its own trace. Therefore $\mathrm{E}_{x_0}\left(x_0^T (X^T X)^{-1} x_0\right) = \mathrm{trace}\left(\mathrm{E}_{x_0}(x_0 x_0^T (X^T X)^{-1})\right)$ which is approximately equal to $\sigma^2 \sigma^{-2} \mathrm{trace}(I_p)/N = p/N$.

Ex. 2.6 (repeated measurements)

To search for parameter θ using least squares one seeks to minimize

$$\mathrm{RSS}(\theta) = \sum_{k=1}^{N} (y_k - f_\theta(x_k))^2 \,, \tag{11}$$

as a function of θ. If there are *repeated* independent variables x_i then this prescription is equivalent to a *weighted* least squares problem as we now show. The motivation for this discussion is that often experimentally one would like to get an accurate estimate of the error ε in the model

$$y = f(x) + \varepsilon \,.$$

One way to do this is to perform many experiments, observing the different values of y produced by the data generating process when the *same* value of x is produced for each experiment. If $\varepsilon = 0$ we would expect the results of each of these experiments to be the same. Let N_u be the number of *unique* inputs x, that is, the number of distinct inputs after discarding duplicates. Assume that if the ith unique x value gives rise to n_i potentially different y values. With this notation we can write the $\mathrm{RSS}(\theta)$ above as

$$\mathrm{RSS}(\theta) = \sum_{i=1}^{N_u} \sum_{j=1}^{n_i} (y_{ij} - f_\theta(x_i))^2 \,.$$

Here y_{ij} is the jth response $1 \le j \le n_i$ to the ith unique input. Expanding the quadratic in the above expression we have

$$\begin{aligned} \mathrm{RSS}(\theta) &= \sum_{i=1}^{N_u} \sum_{j=1}^{n_i} ({y_{ij}}^2 - 2 f_\theta(x_i) y_{ij} + f_\theta(x_i)^2) \\ &= \sum_{i=1}^{N_u} n_i \left(\frac{1}{n_i} \sum_{j=1}^{n_i} {y_{ij}}^2 - \frac{2}{n_i} f_\theta(x_i) \left(\sum_{j=1}^{n_i} y_{ij} \right) + f_\theta(x_i)^2 \right) . \end{aligned}$$

Let's define $\bar{y}_i \equiv \frac{1}{n_i} \sum_{j=1}^{n_i} y_{ij}$, the average of all responses y resulting from the same input x_i. Using this definition and completing the square we have

$$\mathrm{RSS}(\theta) = \sum_{i=1}^{N_u} n_i (\bar{y}_i - f_\theta(x_i))^2 + \sum_{i=1}^{N_u} \sum_{j=1}^{n_i} {y_{ij}}^2 - \sum_{i=1}^{N_u} n_i \bar{y}_i^2 \tag{12}$$

Once the measurements are received the sample points y are fixed and do not change. Thus minimizing Equation 11 with respect to θ is equivalent to minimizing Equation 12 without the term $\sum_{i=1}^{N_u} \sum_{j=1}^{n_i} {y_{ij}}^2 - \sum_{i=1}^{N_u} n_i \bar{y}_i^2$. This motivates the minimization of

$$\mathrm{RSS}(\theta) = \sum_{i=1}^{N_u} n_i (\bar{y}_i - f_\theta(x_i))^2 \,.$$

This later problem is known a *weighted* least squares since each repeated input vector x_i is to fit the value of $\bar{y}_i$ (the average of output values) and each residual error is weighted by how many times the measurement of x_i was taken. It is a *reduced* problem since the number of points we are working with is now $N_u < N$.

Ex. 2.7 (forms for linear regression and k-nearest neighbor regression)

To simplify this problem lets begin in the case of simple linear regression where there is only one response y and one predictor x. Then the standard definitions of y and X state that $y^T = (y_1, \ldots, y_n)$, and $X^T = \begin{bmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_n \end{bmatrix}$.

Part (a): Let's first consider linear regression. We use (2.2) and (2.6), but we avoid just copying the formulas blindly. We have $\hat{\beta} = \left(X^T X\right)^{-1} X^T y$, and then set

$$\hat{f}(x_0) = [x_0\ 1]\hat{\beta} = [x_0\ 1]\left(X^T X\right)^{-1} X^T y.$$

In terms of the notation of the question,

$$\ell_i(x_0; \mathcal{X}) = [x_0\ 1]\left(X^T X\right)^{-1} \begin{bmatrix} 1 \\ x_i \end{bmatrix}$$

for each i with $1 \leq i \leq n$.

More explicitly, $X^T X = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}$ which has determinant $(n-1)\sum_i x_i^2 - 2n\sum_{i<j} x_i x_j$. This allows us to calculate $\left(X^T X\right)^{-1}$ and $\ell_i(x_0; \mathcal{X})$ even more explicitly if we really want to.

In the case of k-nearest neighbor regression $\ell_i(x_0; \mathcal{X})$ is equal to $1/k$ if x_i is one of the nearest k points and 0 otherwise.

Part (b): Here $\mathcal{X}$ is fixed, and $\mathcal{Y}$ varies. Also x_0 and $f(x_0)$ are fixed. So

$$\begin{aligned}
\mathrm{E}_{\mathcal{Y}|\mathcal{X}}\left(\left(f(x_0) - \hat{f}(x_0)\right)^2\right) &= f(x_0)^2 - 2.f(x_0).\mathrm{E}_{\mathcal{Y}|\mathcal{X}}\left(\hat{f}(x_0)\right) + \mathrm{E}_{\mathcal{Y}|\mathcal{X}}\left(\left(\hat{f}(x_0)\right)^2\right) \\
&= \left(f(x_0) - \mathrm{E}_{\mathcal{Y}|\mathcal{X}}\left(\hat{f}(x_0)\right)\right)^2 + \mathrm{E}_{\mathcal{Y}|\mathcal{X}}\left(\left(\hat{f}(x_0)\right)^2\right) - \left(\mathrm{E}_{\mathcal{Y}|\mathcal{X}}\left(\hat{f}(x_0)\right)\right)^2 \\
&= (\text{bias})^2 + \mathrm{Var}(\hat{f}(x_0))
\end{aligned}$$

Part (c): The calculation goes the same way as in (b), except that both $\mathcal{X}$ and $\mathcal{Y}$ vary. Once again x_0 and $f(x_0)$ are constant.

$$\begin{aligned}
\mathrm{E}_{\mathcal{X},\mathcal{Y}}\left(\left(f(x_0) - \hat{f}(x_0)\right)^2\right) &= f(x_0)^2 - 2.f(x_0).\mathrm{E}_{\mathcal{X},\mathcal{Y}}\left(\hat{f}(x_0)\right) + \mathrm{E}_{\mathcal{X},\mathcal{Y}}\left(\left(\hat{f}(x_0)\right)^2\right) \\
&= \left(f(x_0) - \mathrm{E}_{\mathcal{X},\mathcal{Y}}\left(\hat{f}(x_0)\right)\right)^2 + \mathrm{E}_{\mathcal{X},\mathcal{Y}}\left(\left(\hat{f}(x_0)\right)^2\right) - \left(\mathrm{E}_{\mathcal{X},\mathcal{Y}}\left(\hat{f}(x_0)\right)\right)^2 \\
&= (\text{bias})^2 + \mathrm{Var}(\hat{f}(x_0))
\end{aligned}$$

The terms in (b) can be evaluated in terms of the $\ell_i(x_0; \mathcal{X})$ and the distribution of ε_i. We

need only evaluate $\mathrm{E}_{\mathcal{Y}|\mathcal{X}}\left(\hat{f}(x_0)\right) = \sum \ell_i(x_0;\mathcal{X})f(x_i)$ and

$$\begin{aligned}\mathrm{E}_{\mathcal{Y}|\mathcal{X}}\left(\left(\hat{f}(x_0)\right)^2\right) &= \sum_{i,j}\ell_i(x_0;\mathcal{X})\ell_j(x_0;\mathcal{X})\mathrm{E}\left(\left(f(x_i)+\varepsilon_i\right)\left(f(x_j)+\varepsilon_j\right)\right)\\ &= \sum_{i,j}\ell_i(x_0;\mathcal{X})\ell_j(x_0;\mathcal{X})f(x_i)f(x_j) + \sum_i \sigma^2\ell_i(x_0;\mathcal{X})^2.\end{aligned}$$

The terms in (c) can be evaluated in terms of $\mathrm{E}_{\mathcal{X},\mathcal{Y}}\left(\hat{f}(x_0)\right)$ and $\mathrm{E}_{\mathcal{X},\mathcal{Y}}\left(\left(\hat{f}(x_0)\right)^2\right)$. This means multiplying the expressions just obtained by

$$h(x_1)\ldots h(x_n)dx_1\ldots dx_n$$

and then integrating. Even if h and f are known explicitly, it is optimistic to think that closed formulas might result in the case of linear regression. In this case, we have gone as far as is reasonable to go.

In the case of k-nearest-neighbor regression, we can go a bit further. Let

$$U(a,b,c) = \{x : |x-c| \geq \max(|a-c|,|b-c|)\},$$

and let $A(a,b,c) = \int_{U(a,b,c)} h(x)\,dx$. Then $A(a,b,c)$ is the probability of lying further from c than either a or b. Consider the event $F(\mathcal{X})$ where $x_1 < \ldots < x_k$ and, when $i > k$, $x_i \in U(x_1,x_k,x_0)$. There are $n!/(n-k)!$ disjoint events obtained by permuting the indices, and their union covers all possibilities for $\mathcal{X}$, as we see by starting with the k elements of $\mathcal{X}$ that are nearest to x_0.

We have

$$\mathrm{E}_{\mathcal{X},\mathcal{Y}}\left(\hat{f}(x_0)\right) = \frac{n!}{(n-k)!}\int_{x_1<\cdots<x_k} h(x_1)\ldots h(x_k).A(x_1,x_k,x_0)^{n-k}\sum_{i=1}^{k}\frac{f(x_i)}{k}\,dx_1\ldots dx_k$$

and

$$\begin{aligned}\mathrm{E}_{\mathcal{X},\mathcal{Y}}\left(\left(\hat{f}(x_0)\right)^2\right) = \frac{n!}{(n-k)!}\int_{x_1<\cdots<x_k} h(x_1)\ldots h(x_k)\ldots\\ \ldots A(x_1,x_k,x_0)^{n-k}\left(\sum_{1\leq i,j\leq k}\frac{f(x_i)f(x_j)}{k^2}+\frac{\sigma^2}{k}\right)dx_1\ldots dx_k\end{aligned}$$

We have not answered Ex. 2.7(d) (on-line edition) as it doesn't seem to us to mean anything. In particular, the word *Establish* should mean some kind of formal deduction, and we don't think anything like that is available.

Ex. 2.8 (classifying 2's and 3's)

This problem was implemented in `R` , a programming language for statistical computations, in the file `Ex2.8.R`. The program takes a few minutes to run. The results of running this program are given by

```
Training error on linear regression =  0.099
Proportion of training errors on the digit 2 =  0.404
Proportion of training errors on the digit 3 =  0.397
Test error on linear regression =  0.218
Proportion of test errors on the digit 2 =  0.364
Proportion of test errors on the digit 3 =  0.367
End of linear regression
======================

Now we look at nearest neighbour regression.
First the training error:
nhd F% training error
1 0 0
3 0.504 0.014
5 0.576 0.018
7 0.648 0.022
15 0.936 0.033

Test errors:
nhd F% test error
1 2.473 0.099
3 3.022 0.092
5 3.022 0.091
7 3.297 0.094
15 3.846 0.107
```

Note that linear regression is hopeless on this problem, partly, we think, because the pixels for different samples do not align properly. What is unexpected is that the linear regression does better on the test data than on the training data.

Nearest neighbor results are quite reasonable. The training error results are reduced by the fact that there is one direct hit. Note how the amount of error creeps up as the number of neighbors is increased.

Ex. 2.9 (the average training error is smaller than the testing error)

The expectation of the test term $\frac{1}{M}\sum\left(\tilde{y_i} - \hat{\beta}^T x_i\right)^2$ is equal to the expectation of $\left(\tilde{y_1} - \hat{\beta}^T x_1\right)^2$, and is therefore independent of M. We take $M = N$, and then decrease the test expression on replacing $\hat{\beta}$ with a value of β that minimizes the expression. Now the expectations of the two terms are equal. This proves the result. Note that we may have to use the Moore-Penrose pseudo-inverse of $X^T X$, if the rank of X is less than p. This is not a continuous function of X, but it is measurable, which is all we need.

Chapter 3 (Linear Methods for Regression)

Notes on the Text

Linear Regression Models and Least Squares

For this chapter, given the input vector x, the model of how our scalar output y is generated will assumed to be $y = f(x) + \varepsilon = x^T\beta + \varepsilon$ for some fixed vector β of $p+1$ coefficients, and ε a scalar random variable with mean 0 and variance σ^2. With a full data set obtained of N input/output pairs (x_i, y_i) arranged in the vector variables X and $\mathbf{y}$, the space in which we work is $\mathbb{R}^N$. This contains vectors like $\mathbf{y} = (y_1, \ldots, y_N)$, and each column of X. The least squares estimate of β is given by the book's Equation 3.6

$$\hat{\beta} = (X^TX)^{-1}X^T\mathbf{y}\,. \tag{13}$$

We fix X and compute the statistical properties of $\hat{\beta}$ with respect to the distribution $Y|X$. Using the fact that $\mathrm{E}(\mathbf{y}) = X\beta$, we obtain

$$\mathrm{E}(\hat{\beta}) = (X^TX)^{-1}X^TX\beta = \beta\,. \tag{14}$$

Using Equation 14 for $\mathrm{E}(\hat{\beta})$ we get

$$\begin{aligned}\hat{\beta} - \mathrm{E}(\hat{\beta}) &= (X^TX)^{-1}X^T\mathbf{y} - (X^TX)^{-1}X^TX\beta \\ &= (X^TX)^{-1}X^T(\mathbf{y} - X\beta) \\ &= (X^TX)^{-1}X^T\boldsymbol{\varepsilon},\end{aligned}$$

where $\boldsymbol{\varepsilon}$ is a random column vector of dimension N. The variance of $\hat{\beta}$ is computed as

$$\begin{aligned}\mathrm{Var}[\hat{\beta}] &= \mathrm{E}[(\hat{\beta} - \mathrm{E}[\hat{\beta}])(\hat{\beta} - \mathrm{E}[\hat{\beta}])^T] \\ &= (X^TX)^{-1}X^T\mathrm{E}\left(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}\right)^T X(X^TX)^{-1} \\ &= (X^TX)^{-1}X^T\mathrm{Var}(\boldsymbol{\varepsilon})X(X^TX)^{-1}\,.\end{aligned}$$

If we assume that the entries of $\mathbf{y}$ are uncorrelated and all have the same variance of σ^2 (again given X), then $\mathrm{Var}[\boldsymbol{\varepsilon}] = \sigma^2 I_N$, where I_N is the $N \times N$ identity matrix and the above becomes

$$\mathrm{Var}[\hat{\beta}] = \sigma^2(X^TX)^{-1}X^TX(X^TX)^{-1} = (X^TX)^{-1}\sigma^2\,, \tag{15}$$

which is Equation 3.8 in the book.

It remains to specify how to determine σ^2. To that end once β is estimated we can compute

$$\hat{\sigma}^2 = \frac{1}{N-p-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \frac{1}{N-p-1}\sum_{i=1}^{N}(y_i - x_i^T\beta)^2\,, \tag{16}$$

and subsequently claim that this gives an unbiased estimate of σ^2. To see this, we argue as follows.

Recall that we assume that each coordinate of the vector $\mathbf{y}$ is independent, and distributed as a normal random variable, centered at 0, with variance σ^2. Since these N samples are independent in the sense of probability, the probability measure of the vector $\mathbf{y} \in \mathbb{R}^N$ is the product of these, denoted by $\mathcal{N}(0, \sigma^2 I_N)$. This density has the following form

$$p(\mathbf{u}) = \frac{1}{\left(\sqrt{2\pi}\sigma\right)^N} \exp\left(-\frac{u_1^2 + \ldots + u_N^2}{2\sigma^2}\right),$$

where the u_i are the coordinates of $\mathbf{u}$ and $\mathbf{u} \in \mathbb{R}^N$. Now any orthonormal transformation of $\mathbb{R}^N$ preserves $\sum_{i=1}^N u_i^2$. This means that the pdf is also invariant under any orthogonal transformation keeping 0 fixed. Note that any function of $u_1, \ldots, u_k$ is probability independent of any function of $u_{k+1}, \ldots, u_N$. Suppose that $\mathbb{R}^N = V \oplus W$ is an orthogonal decomposition, where V has dimension k and W has dimension $N - k$. Let $v_1, \ldots, v_k$ be coordinate functions associated to an orthonormal basis of V and let $w_{k+1}, \ldots, w_N$ be coordinate functions associated to an orthonormal basis of W. The invariance under orthogonal transformations now shows that the induced probability measure on V is $\mathcal{N}(0, \sigma^2 I_k)$. In particular the square of the distance from 0 in V has distribution χ_k^2. (However, this discussion does not appeal to any properties of the χ^2 distributions, so the previous sentence could have been omitted without disturbing the proof of Equation 3.8.)

If x is the standard variable in $\mathcal{N}(0, \sigma^2)$, then $\mathrm{E}(x^2) = \sigma^2$. It follows that the distance squared from 0 in V, $\sum_{i=1}^k v_i^2$, has expectation $k\sigma^2$. We now use the fact that in ordinary least squares $\hat{\mathbf{y}}$ is the orthogonal projection of $\mathbf{y}$ onto the column space of X as a subspace of $\mathbb{R}^N$. Under our assumption of the independence of the columns of X this later space has dimension $p+1$. In the notation above $\mathbf{y} \in V$ with V the column space of X and $\mathbf{y} - \hat{\mathbf{y}} \in W$, where W is the orthogonal complement of the column space of X. Because $y \in \mathbb{R}^N$ and W is of dimension $p + 1$, we know that W has dimension $N - p - 1$ and $\mathbf{y} - \hat{\mathbf{y}}$ is a random vector in W with distribution $\mathcal{N}(0, \sigma^2 I_{N-p-1})$. The sum of squares of the N components of $\mathbf{y} - \hat{\mathbf{y}}$ is the square of the distance in W to the origin. Therefore $\sum_{i=1}^N (y_i - \hat{y}_i)^2$ has expectation $(N - p - 1)\sigma^2$. From Equation 16 we see that $\mathrm{E}(\hat{\sigma}^2) = \sigma^2$. This proves the book's Equation 3.8, which was stated in the book without proof.

Notes on the Prostate Cancer Example

In the `R` script `duplicate_table_3_1_N_2.R` we provide *explicit* code that duplicates the numerical results found in Table 3.1 and Table 3.2 using the above formulas for ordinary least squares. In addition, in that same function we then use the `R` function `lm` to verify the same numeric values. Using the `R` package `xtable` we can display these correlations in Table 1. Next in Table 2 we present the results we obtained for the coefficients of the ordinary least squares fit. Notice that these coefficients are slightly different than the ones presented in the book. These numerical values, generated from the formulas above, agree very closely with the ones generated by the `R` command `lm`. I'm not sure where the discrepancy with the book lies. Once this linear model is fit we can then apply it to the testing data and observe how well we do. When we do that we compute the expected squared prediction error (ESPE)

	lcavol	lweight	age	lbph	svi	lcp	gleason	pgg45
lcavol	1.000	0.300	0.286	0.063	0.593	0.692	0.426	0.483
lweight	0.300	1.000	0.317	0.437	0.181	0.157	0.024	0.074
age	0.286	0.317	1.000	0.287	0.129	0.173	0.366	0.276
lbph	0.063	0.437	0.287	1.000	-0.139	-0.089	0.033	-0.030
svi	0.593	0.181	0.129	-0.139	1.000	0.671	0.307	0.481
lcp	0.692	0.157	0.173	-0.089	0.671	1.000	0.476	0.663
gleason	0.426	0.024	0.366	0.033	0.307	0.476	1.000	0.757
pgg45	0.483	0.074	0.276	-0.030	0.481	0.663	0.757	1.000

Table 1: Duplication of the values from Table 3.1 from the book. These numbers exactly match those given in the book.

	Term	Coefficients	Std_Error	Z_Score
1	(Intercept)	2.45	0.09	28.18
2	lcavol	0.72	0.13	5.37
3	lweight	0.29	0.11	2.75
4	age	-0.14	0.10	-1.40
5	lbph	0.21	0.10	2.06
6	svi	0.31	0.13	2.47
7	lcp	-0.29	0.15	-1.87
8	gleason	-0.02	0.14	-0.15
9	pgg45	0.28	0.16	1.74

Table 2: Duplicated results for the books Table 3.2. These coefficients are slightly different than the ones presented in the book.

loss over the testing data points to be

$$\text{ESPE} \approx \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (y_i - \hat{y}_i)^2 = 0.521274\,.$$

This value is relatively close the corresponding value presented in Table 3.3 of the book which provides a summary of many of the linear techniques presented in this chapter of 0.521. To compute the standard error of this estimate we use the formula

$$\text{se}(\text{ESPE})^2 = \frac{1}{N_{\text{test}}} \text{Var}(\mathbf{Y} - \hat{\mathbf{Y}}) = \frac{1}{N_{\text{test}}} \left(\frac{1}{N_{\text{test}} - 1} \sum_{i=1}^{N_{\text{test}}} (y_i - \hat{y}_i)^2 \right) .$$

This result expresses the idea that if the pointwise error has a variance of σ^2 then the average of N_{test} such things (by the central limit theorem) has a variance given by σ^2/N_{test}. Using this we compute the standard error given by 0.1787240, which is relatively close from the result presented in the book of 0.179.

Notes on the Gauss-Markov Theorem

Let $\tilde{\theta}$ be an estimator of the fixed, non-random parameter θ. So $\tilde{\theta}$ is a function of data. Since the data is (normally) random, $\tilde{\theta}$ is a random variable. We define the mean-square-error (MSE) of our estimator $\tilde{\delta}$ by

$$\text{MSE}(\tilde{\theta}) := \text{E}\left((\tilde{\theta}-\theta)^2\right) .$$

We can expand the quadratic $(\tilde{\theta}-\theta)^2$ to get

$$\begin{aligned}(\tilde{\theta}-\theta)^2 &= (\tilde{\theta}-\text{E}(\tilde{\theta})+\text{E}(\tilde{\theta})-\theta)^2 \\ &= (\tilde{\theta}-\text{E}(\tilde{\theta}))^2 + 2(\tilde{\theta}-\text{E}(\tilde{\theta}))(\text{E}(\tilde{\theta})-\theta) + (\text{E}(\tilde{\theta})-\theta)^2 .\end{aligned}$$

Taking the expectation of this and remembering that θ is non random we have

$$\begin{aligned}\text{MSE}(\tilde{\theta}-\theta)^2 &= \text{E}(\tilde{\theta}-\theta)^2 \\ &= \text{Var}(\tilde{\theta}) + 2(\text{E}(\tilde{\theta})-\text{E}(\tilde{\theta}))(\text{E}(\tilde{\theta})-\theta) + (\text{E}(\tilde{\theta})-\theta)^2 \\ &= \text{Var}(\tilde{\theta}) + (\text{E}(\tilde{\theta})-\theta)^2 , \qquad (17)\end{aligned}$$

which is the book's Equation 3.20.

At the end of this section the book shows that the expected quadratic error in the *prediction* under the model $\tilde{f}(\cdot)$ can be broken down into two parts as

$$\text{E}(Y_0-\tilde{f}(x_0))^2 = \sigma^2 + \text{MSE}(\tilde{f}(x_0)) .$$

The first error component σ^2 is unrelated to what model is used to describe our data. It cannot be reduced for it exists in the true data generation process. The second source of error corresponding to the term $\text{MSE}(\tilde{f}(x_0))$ represents the error in the *model* and is under control of the statistician (or the person doing the data modeling). Thus, based on the above expression, if we minimize the MSE of our estimator $\tilde{f}(x_0)$ we are effectively minimizing the expected (quadratic) prediction error which is our ultimate goal anyway. In this book we will explore methods that minimize the mean square error. By using Equation 17 the mean square error can be broken down into two terms: a model *variance* term and a model *bias* squared term. We will explore methods that seek to keep the total contribution of these two terms as small as possible by explicitly considering the trade-offs that come from methods that might increase one of the terms while decreasing the other.

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

$$\hat{\beta} = (X^T X)^{-1} X^T Y .$$

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

$$(X^T X)^{-1} = (\sum_{i=1}^{N} x_i^2)^{-1} \quad \text{and} \quad X^T Y = \sum_{i=1}^{N} x_i y_i \,,$$

so the least squares estimate of β is therefore given by

$$\hat{\beta} = \frac{\sum_{i=1}^{N} x_i y_i}{\sum_{i=1}^{N} x_i^2} = \frac{x^T y}{x^T x} \,. \tag{18}$$

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 18). 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

$$\hat{\beta}_1 = \frac{\langle \mathbf{x} - \bar{x}\mathbf{1}, \mathbf{y} \rangle}{\langle \mathbf{x} - \bar{x}\mathbf{1}, \mathbf{x} - \bar{x}\mathbf{1} \rangle} \,, \tag{19}$$

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 way 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

$$\hat{\beta}_p = \frac{\langle \mathbf{z}_p, \mathbf{y} \rangle}{\langle \mathbf{z}_p, \mathbf{z}_p \rangle} \,, \tag{20}$$

which is the books equation 3.28. 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

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

From Equation 20 we can derive the variance of $\hat{\beta}_p$ that is stated in the book. We find

$$\begin{aligned} \mathrm{Var}(\hat{\beta}_p) &= \mathrm{Var}\left(\frac{z_p^T y}{\langle z_p, z_p \rangle} \right) = \frac{z_p^T \mathrm{Var}(y) z_p}{\langle z_p, z_p \rangle^2} = \frac{z_p^T (\sigma^2 I) z_p}{\langle z_p, z_p \rangle^2} \\ &= \frac{\sigma^2}{\langle z_p, z_p \rangle} \,, \end{aligned}$$

which is the books equation 3.29.

As stated earlier Algorithm 3.1 is known as the Gram-Schmidt procedure for multiple regression and it has a nice matrix representation that can be useful for deriving results that demonstrate the properties of linear regression. To demonstrate some of these, note that we can write the Gram-Schmidt result in matrix form using the QR decomposition as

$$X = QR \,. \tag{21}$$

In this decomposition Q is a $N \times (p+1)$ matrix with orthonormal columns and R is a $(p+1) \times (p+1)$ upper triangular matrix. In this representation the ordinary least squares (OLS) estimate for β can be written as

$$\begin{aligned}\hat{\beta} &= (X^T X)^{-1} X^T \mathbf{y} \\ &= (R^T Q^T Q R)^{-1} R^T Q^T \mathbf{y} \\ &= (R^T R)^{-1} R^T Q^T \mathbf{y} \\ &= R^{-1} R^{-T} R^T Q^T \mathbf{y} \\ &= R^{-1} Q^T \mathbf{y}\,, \end{aligned} \tag{22}$$

which is the books equation 3.32. Using the above expression for $\hat{\beta}$ the fitted value $\hat{\mathbf{y}}$ can be written as

$$\hat{\mathbf{y}} = X\hat{\beta} = QRR^{-1}Q^T\mathbf{y} = QQ^T\mathbf{y}\,, \tag{23}$$

which is the books equation 3.33. This last equation expresses the fact in ordinary least squares we obtain our fitted vector $\mathbf{y}$ by first computing the coefficients of $\mathbf{y}$ in terms of the basis spanned by the columns of Q (these coefficients are given by the vector $Q^T\mathbf{y}$). We next construct $\hat{\mathbf{y}}$ using these numbers as the coefficients of the column vectors in Q (this is the product $QQ^T y$).

Notes on best-subset selection

While not applicable for larger problems it can be instructive to observe how best-subset selection could be done in practice for small problems. In the `R` script `duplicate_figure_3_5.R` we provide code that duplicates the numerical results found in Figure 3.5 from the book. The results from running this script are presented in Figure 1 . It should be noted that in generating this data we did *not* apply cross validation to selecting the value k that should be used for the optimal sized subset to use for prediction accuracy. Cross validation of this technique is considered in a later section where 10 fold cross validation is used to estimate the complexity parameter (k in this case) with the "one-standard-error" rule.

Notes on various linear prediction methods applied to the prostate data set

In this subsection we present numerical results that duplicate the linear predictive methods discussed in the book. One thing to note about the implementation of methods is that many of these methods "standardize" their predictors and/or subtract the mean from the response before applying any subsequent techniques. Often this is just done once over the entire set of "training data" and then forgotten. For *some* of the methods and the subsequent results presented here I choose do this scaling *as part of the cross validation routine*. Thus in computing the cross validation (CV) errors I would keep the variables in their raw (unscaled) form, perform scaling on the training portion of the cross validation data, apply this scaling to the testing portion of the CV data and then run our algorithm on the scaled CV training data. This should not result in a huge difference between the "scale and forget" method but

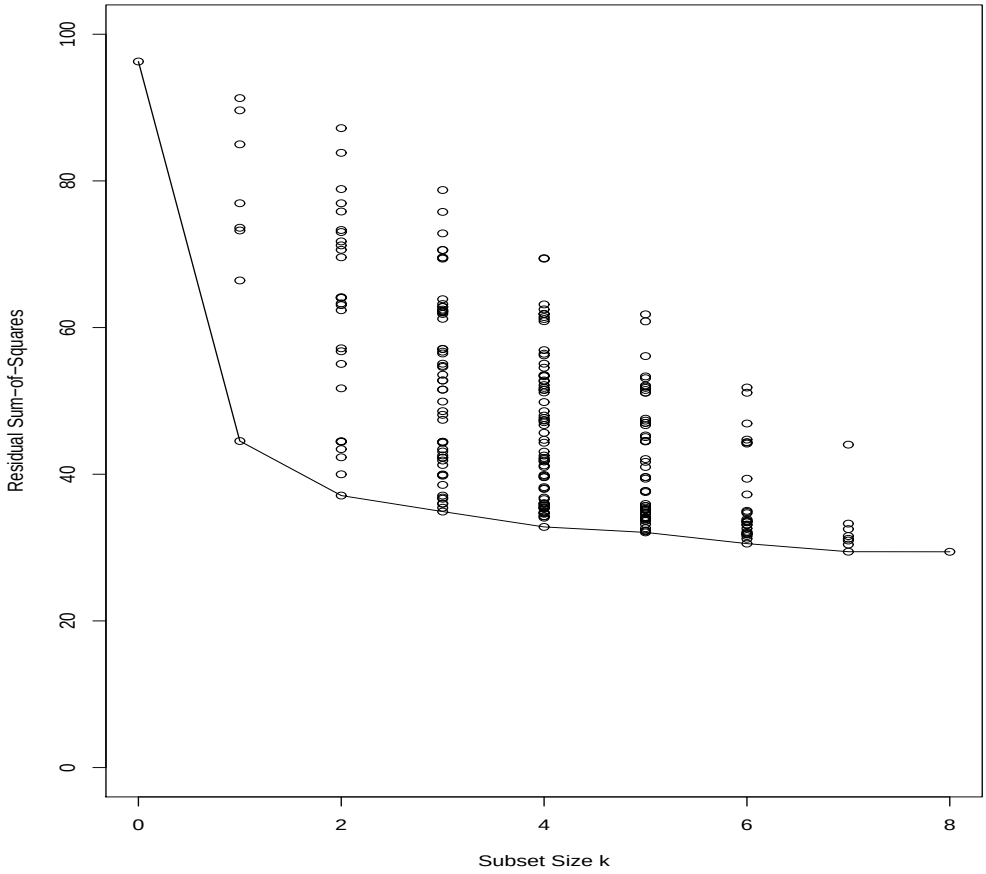

Figure 1: Duplication of the books Figure 3.5 using the code `duplicate_figure_3_5.R`. This plot matches quite well qualitatively and quantitatively the corresponding one presented in the book.

I wanted to mention this point in case anyone reads the provided code for the all subsets and ridge regression methods.

In duplicating the all-subsets and ridge regression results in this section we wrote our own `R` code to perform the given calculations. For ridge regression an alternative approach would have been to use the `R` function `lm.ridge` found in the `MASS` package. In duplicating the lasso results in this section we use the `R` package `glmnet` [5], provided by the authors and linked from the books web site. An interesting fact about the `glmnet` package is that for the parameter settings $\alpha = 0$ the elastic net penalization framework ignores the L_1 (lasso) penalty on β and the formulation becomes equivalent to an L_2 (ridge) penalty on β. This parameter setting would allow one to use the `glmnet` package to do ridge-regression if desired. Finally, for the remaining two regression methods: principal component regression (PCR) and partial least squares regression (PLSR) we have used the `R` package `pls` [7], which is a package tailored to perform these two types of regressions.

As a guide to what `R` functions perform what coding to produce the above plots and table

Term	LS	Best Subset	Ridge	Lasso	PCR	PLS
(Intercept)	2.452	2.452	2.452	2.452	2.452	2.452
lcavol	0.716	0.779	0.432	0.558	0.570	0.436
lweight	0.293	0.352	0.252	0.183	0.323	0.360
age	-0.143		-0.045		-0.153	-0.021
lbph	0.212		0.168		0.216	0.243
svi	0.310		0.235	0.088	0.322	0.259
lcp	-0.289		0.005		-0.050	0.085
gleason	-0.021		0.042		0.228	0.006
pgg45	0.277		0.134		-0.063	0.084
Test Error	0.521	0.492	0.492	0.484	0.448	0.536
Std Error	0.178	0.143	0.161	0.166	0.104	0.149

Table 3: Duplicated results for the books Table 3.3. These coefficients are slightly different than the ones presented in the book but still show the representative ideas.

entries we have:

- The least squares (LS) results are obtained in the script `duplicate_table_3_1_N_2.R`.
- The best subset results are obtained using the script `dup_OSE_all_subset.R`.
- The ridge regression results are obtained using the script `dup_OSE_ridge_regression.R`.
- The lasso results are obtained using the script `dup_OSE_lasso.R`.
- The PCR and PLS results are obtained using the script `dup_OSE_PCR_N_PLSR.R`.

We duplicate figure 3.7 from the book in Figure 2. We also duplicate table 3.3 from the book in Table 3. There are some slight differences in the plots presented in Figure 2, see the caption for that figure for some of the differences. There are also numerical differences between the values presented in Table 3, but the general argument made in the book still holds true. The idea that should be taken away is that all linear methods presented in this chapter (with the exception of `PLS`) produce a linear model that *outperforms* the least squares model. This is important since it is a way for the applied statistician to make further improvements in his application domain.

Notes on various shrinkage methods

After presenting ordinary least squares the book follows with a discussion on subset section techniques: best-subset selection, forward- and backwards-stepwise selection, and forward-stagewise regression. The book's concluding observation is that if it were possible *best-subset selection* would be the optimal technique and should be used in every problem. The book presents two reasons (computational and statistical) why it is not possible to use best-subset

selection in many cases of practical interest. For most problems the computational reason is overpowering since if we can't even compute all of the subsets it will not be practical to use this algorithm on applied problems. The difficulty with best-subset selection is that it is a discrete procedure and there are too many required subsets to search over. The development of the various shrinkage method presented next attempt to overcome this combinatorial explosion of best-subset by converting the discrete problem into a continuous one. The continuous problems then turn out to be much simpler to solve. An example like this of a technique we will study is *ridge regression*. Ridge regression constrains the sum of the squares of the estimated coefficients β_i (except for β_0 which is dealt with separately) to be less than a threshold t. The effect of this constraint is to hopefully "zero out" the same β_i that would have been excluded by a best-subset selection procedure. If one wanted to mimic the result of best-subset selection and truly wanted a fixed number, say M, of non-zero coefficients β_i, one could always simply zero the $p - M$ smallest in magnitude β_i coefficients and then redo the ordinary least squares fit with the retained coefficients. In fact in Table 3.3 we see that best-subset selection selected two predictors `lcavol` and `lweight` as predictors. If instead we had taken the ridge regression result and kept only the two features with the largest values of $|\beta_i|$ we would have obtained the *same* two feature subset. While these are certainly only heuristic arguments hopefully they will make understanding the following methods discussed below easier.

Notes on ridge regression

In this subsection of these notes we derive some of the results presented in the book. If we compute the singular value decomposition (SVD) of the $N \times p$ centered data matrix X as

$$X = UDV^T \,, \tag{24}$$

where U is a $N \times p$ matrix with orthonormal columns that span the column space of X, V is a $p \times p$ orthogonal matrix, and D is a $p \times p$ diagonal matrix with elements d_j ordered such that $d_1 \geq d_2 \geq \cdots d_p \geq 0$. From this representation of X we can derive a simple expression for $X^T X$. We find that

$$X^T X = VDU^T UDV^T = VD^2V^T \,. \tag{25}$$

Using this expression we can compute the least squares fitted values $\hat{y}^{\text{ls}} = X\hat{\beta}^{\text{ls}}$ as

$$\begin{aligned} \hat{y}^{\text{ls}} = X\hat{\beta}^{\text{ls}} &= UDV^T(VD^2V^T)^{-1}VDU^T\mathbf{y} \\ &= UDV^T(V^{-T}D^{-2}V^{-1})VDU^T\mathbf{y} \\ &= UU^T\mathbf{y} \qquad (26) \\ &= \sum_{j=1}^{p} u_j(u_j^T\mathbf{y}) \,, \qquad (27) \end{aligned}$$

where we have written this last equation in a form that we can directly compare to an expression we will derive for ridge regression (specifically the books equation 3.47). To compare how the fitted values $\hat{y}$ obtained in ridge regression compare with ordinary least squares we next consider the SVD expression for $\hat{\beta}^{\text{ridge}}$. In the same way as for least squares

we find

$$\begin{aligned}\hat{\beta}^{\text{ridge}} &= (X^T X + \lambda I)^{-1} X^T \mathbf{y} \qquad (28)\\ &= (V D^2 V^T + \lambda V V^T)^{-1} V D U^T \mathbf{y}\\ &= (V(D^2 + \lambda I) V^T)^{-1} V D U^T \mathbf{y}\\ &= V(D^2 + \lambda I)^{-1} D U^T \mathbf{y}\,. \qquad (29)\end{aligned}$$

Using this we can compute the product $\hat{y}^{\text{ridge}} = X\hat{\beta}^{\text{ridge}}$. As in the above case for least squares we find

$$\hat{y}^{\text{ridge}} = X\hat{\beta}^{\text{ridge}} = UD(D^2 + \lambda I)^{-1} D U^T \mathbf{y}\,. \qquad (30)$$

Now note that in this last expression $D(D^2 + \lambda I)^{-1}$ is a diagonal matrix with elements given by $\frac{d_j^2}{d_j^2+\lambda}$ and the vector $U^T\mathbf{y}$ is the coordinates of the vector $\mathbf{y}$ in the basis spanned by the p-columns of U. Thus writing the expression given by Equation 30 by summing columns we obtain

$$\hat{y}^{\text{ridge}} = X\hat{\beta}^{\text{ridge}} = \sum_{j=1}^{p} u_j \left(\frac{d_j^2}{d_j^2 + \lambda} \right) u_j^T \mathbf{y}\,. \qquad (31)$$

Note that this result is similar to that found in Equation 27 derived for ordinary least squares regression but in ridge-regression the inner products $u_j^T\mathbf{y}$ are now scaled by the factors $\frac{d_j^2}{d_j^2+\lambda}$.

Notes on the effective degrees of freedom $\mathrm{df}(\lambda)$

The definition of the effective degrees of freedom $\mathrm{df}(\lambda)$ in ridge regression is given by

$$\mathrm{df}(\lambda) = \mathrm{tr}[X(X^T X + \lambda I)^{-1} X^T]\,. \qquad (32)$$

Using the results in the SVD derivation of the expression $X\hat{\beta}^{\text{ridge}}$, namely Equation 30 but without the $\mathbf{y}$ factor, we find the eigenvector/eigenvalue decomposition of the matrix inside the trace operation above given by

$$X(X^T X + \lambda I)^{-1} X^T = UD(D^2 + \lambda I)^{-1} D U^T\,.$$

From this expression the eigenvalues of $X(X^T X + \lambda I)^{-1} X^T$ must be given by the elements $\frac{d_j^2}{d_j^2+\lambda}$. Since the trace of a matrix can be shown to equal the sum of its eigenvalues we have that

$$\begin{aligned}\mathrm{df}(\lambda) &= \mathrm{tr}[X(X^T X + \lambda I)^{-1} X^T]\\ &= \mathrm{tr}[UD(D^2 + \lambda I)^{-1} D U^T]\\ &= \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}\,, \qquad (33)\end{aligned}$$

which is the books equation 3.50.

One important consequence of this expression is that we can use it to determine the values of λ for which to use when applying cross validation. For example, the book discusses how to obtain the estimate of y when using ridge regression and it is given by Equation 30 but no mention of the numerical values of λ we should use in this expression to guarantee that we have accurate coverage of all possible regularized linear models. The approach taken in generating the ridge regression results in Figures 2 and 8 is to consider df in Equation 33 a *function* of λ. As such we set $\mathrm{df}(\lambda) = k$ for $k = 1, 2, \cdots, p$ representing all of the possible values for the degree of freedom. We then use Newton's root finding method to solve for λ in the expression

$$\sum_{i=1}^{p} \frac{d_j^2}{d_j^2 + \lambda} = k \,.$$

To implement this root finding procedure recall that d_j in the above expression are given by the SVD of the data matrix X as expressed in Equation 24. Thus we define a function $d(\lambda)$ given by

$$d(\lambda) = \sum_{i=1}^{p} \frac{d_j^2}{d_j^2 + \lambda} - k \,, \tag{34}$$

and we want λ such that $d(\lambda) = 0$. We use Newton's algorithm for this where we iterate given a starting value of λ_0

$$\lambda_{n+1} = \lambda_n - \frac{d(\lambda_n)}{d'(\lambda_n)} \,.$$

Thus we need the derivative of $d(\lambda)$ which is given by

$$d'(\lambda) = -\sum_{i=1}^{p} \frac{d_j^2}{(d_j^2 + \lambda)^2} \,,$$

and an initial guess for λ_0. Since we are really looking for p values of λ (one for each value of k) we will start by solving the problems for $k = p, p-1, p-2, \cdots, 1$. When $k = p$ the value of λ that solves $\mathrm{df}(\lambda) = p$ is seen to be $\lambda = 0$. For each subsequent value of k we use the estimate of λ found in the previous Newton solve as the initial guess for the current Newton solve. This procedure is implemented in the `R` code `opt_lambda_ridge.R`.

Notes on the lasso

When we run the code discussed on Page 20 for computing the lasso coefficients with the `glmnet` software we can also construct the profile of the β_i coefficients as the value of λ changes. When we do this for the prostate data set we obtain Figure 4. This plot agrees quite well with the one presented in the book.

Notes on the three Bayes estimates: subset selection, ridge, and lasso

This would be a good spot for Davids discussion on the various reformulations of the constrained minimization problem for β stating the two formulations and arguing their equivalence.

Below are just some notes scraped together over various emails discussing some facts that I felt were worth proving/discussing/understanding better:

λ is allowed to range over $(0, \infty)$ and all the solutions are different. But s is only allowed to range over an interval (0, something finite). If s is increased further the constrained solution is equal to the unconstrained solution. That's why I object to them saying there is a one-to-one correspondence. It's really a one-to-one correspondence between the positive reals and a finite interval of positive numbers.

I'm assuming by s above you mean the same thing the book does $s = t/\sum_1^p |\hat{\beta}_j|^q$ where q is the "power' in the L_q regularization term. See the section 3.4.3 in the book) . Thus $q = 1$ for lasso and $q = 2$ for ridge. So basically as the unconstrained solution is the least squares one as then all estimated betas approach the least square betas. In that case the largest value for s is 1, so we actually know the value of the largest value of s. For values of s larger than this we will obtain the least squares solution.

The one-to-one correspondence could be easily worked out by a computer program in any specific case. I don't believe there is a nice formula.

Notes on Least Angle Regression (LAR)

To derive a better connection between Algorithm 3.2 (a few steps of Least Angle Regression) and the notation on the general LAR step "k" that is presented in this section that follows this algorithm I found it helpful to perform the first few steps of this algorithm by hand and explicitly writing out what each variable was. In this way we can move from the specific notation to the more general expression.

- Standardize all predictors to have a zero mean and unit variance. Begin with all regression coefficients at zero i.e. $\beta_1 = \beta_2 = \cdots = \beta_p = 0$. The first residual will be $r = y - \bar{y}$, since with all $\beta_j = 0$ and standardized predictors the constant coefficient $\beta_0 = \bar{y}$.
- Set $k = 1$ and begin start the k-th step. Since all values of β_j are zero the first residual is $r_1 = y - \bar{y}$. Find the predictor x_j that is most correlated with this residual r_1. Then as we begin this $k = 1$ step we have the active step given by $\mathcal{A}_1 = \{x_j\}$ and the active coefficients given by $\beta_{\mathcal{A}_1} = [0]$.
- Move β_j from its initial value of 0 and in the direction

$$\delta_1 = (X_{\mathcal{A}_1}^T X_{\mathcal{A}_1})^{-1} X_{\mathcal{A}_1}^T r_1 = \frac{x_j^T r_1}{x_j^T x_j} = x_j^T r_1 \,.$$

  Note that the term $x_j^T x_j$ in the denominator is not present since $x_j^T x_j = 1$ as all variables are normalized to have unit variance. The path taken by the elements in $\beta_{\mathcal{A}_1}$ can be parametrized by

$$\beta_{\mathcal{A}_1}(\alpha) \equiv \beta_{\mathcal{A}_1} + \alpha\delta_1 = 0 + \alpha x_j^T r_1 = (x_j^T r_1)\alpha \quad \text{for} \quad 0 \le \alpha \le 1 \,.$$

This path of the coefficients $\beta_{\mathcal{A}_1}(\alpha)$ will produce a path of fitted values given by

$$\hat{f}_1(\alpha) = X_{\mathcal{A}_1}\beta_{\mathcal{A}_1}(\alpha) = (x_j^T r_1)\alpha\, x_j\,,$$

and a residual of

$$r(\alpha) = y - \bar{y} - \alpha(x_j^T r_1)x_j = r_1 - \alpha(x_j^T r_1)x_j\,.$$

Now at this point x_j itself has a correlation with this residual as α varies given by

$$x_j^T(r_1 - \alpha(x_j^T r_1)x_j) = x_j^T r_1 - \alpha(x_j^T r_1) = (1-\alpha)x_j^T r_1\,.$$

When $\alpha = 0$ this is the maximum value of $x_j^T r_1$ and when $\alpha = 1$ this is the value 0. All other features (like x_k) have a correlation with this residual given by

$$x_k^T(r_1 - \alpha(x_j^T r_1)x_j) = x_k^T r_1 - \alpha(x_j^T r_1)x_k^T x_j\,.$$

Notes on degrees-of-freedom formula for LAR and the Lasso

From the books definition of the degrees-of-freedom of

$$\mathrm{df}(\hat{y}) = \frac{1}{\sigma^2}\sum_{i=1}^{N}\mathrm{cov}(\hat{y}_i, y)\,. \tag{35}$$

We will derive the quoted expressions for $\mathrm{df}(\hat{y})$ under ordinary least squares regression and ridge regression. We begin by evaluating $\mathrm{cov}(\hat{y}_i, y)$ under ordinary least squares. We first relate this scalar expression into a vector inner product expression as

$$\mathrm{cov}(\hat{y}_i, y_i) = \mathrm{cov}(e_i^T\hat{y}, e_i^T y) = e_i^T\mathrm{cov}(\hat{y}, y)e_i\,.$$

Now for ordinary least squares regression we have $\hat{y} = X\hat{\beta}^{\mathrm{ls}} = X(X^TX)^{-1}X^Ty$, so that the above expression for $\mathrm{cov}(\hat{y}, y)$ becomes

$$\mathrm{cov}(\hat{y}, y) = X(X^TX)^{-1}X^T\mathrm{cov}(y, y) = \sigma^2 X(X^TX)^{-1}X^T\,,$$

since $\mathrm{cov}(y, y) = \sigma^2 I$. Thus

$$\mathrm{cov}(\hat{y}_i, y_i) = \sigma^2 e_i^T X(X^TX)^{-1}X^T e_i = \sigma^2(X^Te_i)(X^TX)^{-1}(X^Te_i)\,.$$

Note that the product $X^Te_i = x_i$ the ith samples feature vector for $1 \le i \le N$ and we have $\mathrm{cov}(\hat{y}_i, y_i) = \sigma^2 x_i^T(X^TX)^{-1}x_i$, which when we sum for $i = 1$ to N and divide by σ^2 gives

$$\begin{aligned}
\mathrm{df}(\hat{y}) &= \sum_{i=1}^{N} x_i^T(X^TX)^{-1}x_i \\
&= \sum_{i=1}^{N}\mathrm{tr}(x_i^T(X^TX)^{-1}x_i) \\
&= \sum_{i=1}^{N}\mathrm{tr}(x_i x_i^T(X^TX)^{-1}) \\
&= \mathrm{tr}\left(\left(\sum_{i=1}^{N} x_i x_i^T\right)(X^TX)^{-1}\right)\,.
\end{aligned}$$

Note that this sum above can be written as

$$\sum_{i=1}^{N} x_i x_i^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix} \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix} = X^T X \,.$$

Thus when there are k predictors we get

$$\mathrm{df}(\hat{y}) = \mathrm{tr}\left((X^T X)(X^T X)^{-1}\right) = \mathrm{tr}(I_{k\times k}) = k \,,$$

the claimed result for ordinary least squares.

To do the same thing for ridge regression we can use Equation 28 to show

$$\hat{y} = X\hat{\beta}^{\mathrm{ridge}} = X(X^T X + \lambda I)^{-1} X^T \mathbf{y} \,.$$

so that

$$\mathrm{cov}(\hat{y}, y) = X(X^T X + \lambda I)^{-1} X^T \mathrm{cov}(y, y) = \sigma^2 X(X^T X + \lambda I)^{-1} X^T \,.$$

Again we can compute the scalar result

$$\mathrm{cov}(\hat{y}_i, y_i) = \sigma^2 (X^T e_i)^T (X^T X + \lambda I)^{-1} (X^T e_i) = \sigma^2 x_i^T (X^T X + \lambda I)^{-1} x_i \,.$$

Then summing for $i = 1, 2, \cdots, N$ and dividing by σ^2 to get

$$\begin{aligned} \mathrm{df}(\hat{y}) &= \sum_{i=1}^{N} \mathrm{tr}\left(x_i x_i^T (X^T X + \lambda I)^{-1}\right) \\ &= \mathrm{tr}\left(X^T X (X^T X + \lambda I)^{-1}\right) \\ &= \mathrm{tr}\left(X (X^T X + \lambda I)^{-1} X^T\right) \,, \end{aligned}$$

which is the books equation 3.50 providing an expression for the degrees of freedom for ridge regression.

Methods Using Derived Input Directions: Principal Components Regression

Since the linear method of principal components regression (PCR) produced some of the best results (see Table 3) as far as the `prostate` data set it seemed useful to derive this algorithm in greater detail here in these notes. The discussion in the text is rather brief in this section we bring the various pieces of the text together and present the complete algorithm in one location. In general PCR is parametrized by M for $0 \leq M \leq p$ (the number of principal components to include). The values of $M = 0$ imply a prediction based on the mean of the response $\bar{y}$ and when $M = p$ using PCR we duplicate the ordinary least squares solution (see Exercise 3.13 Page 41). The value of M used in an application is can be selected by cross validation (see Figure 2). One could imaging a computational algorithm such that given a value of M would only compute the M principal components needed and no others. Since most general purpose eigenvector/eigenvalue code actually produces the entire eigensystem when supplied a given matrix the algorithm below computes *all* of the possible principal component regressions (for $0 \leq M \leq p$) in one step. The PCR algorithm is then given by the following algorithmic steps:

- Standardize the predictor variables x_i for $i = 1, 2, \ldots, p$ to have mean zero and variance one. Demean the response y.
- Given the design matrix X compute the product $X^T X$.
- Compute the eigendecomposition of $X^T X$ as

$$X^T X = V D^2 V^T .$$

  The columns of V are denoted v_m and the diagonal elements of D are denoted d_m.
- Compute the vectors z_m defined as $z_m = X v_m$ for $m = 1, 2, \ldots, p$.
- Using these vectors z_m compute the regression coefficients $\hat{\theta}_m$ given by

$$\hat{\theta}_m = \frac{< z_m, y >}{< z_m, z_m >} .$$

  Note that we don't need to explicitly compute the inner product $< z_m, z_m >$ for each m directly since using the eigendecomposition $X^T X = V D^2 V^T$ computed above this is equal to

$$z_m^T z_m = v_m^T X^T X v_m = v_m^T V D^2 V^T v_m = (V^T v_m)^T D^2 (V^T v_m) = e_m^T D^2 e_m = d_m^2 ,$$

  where e_m is a vector of all zeros with a one in the mth spot.
- Given a value of M for $0 \leq M \leq p$, the values of $\hat{\theta}_m$, and z_m the PCR estimate of y is given by

$$\hat{y}^{\text{pcr}}(M) = \bar{y}\mathbf{1} + \sum_{m=1}^{M} \hat{\theta}_m z_m .$$

  While the value of $\hat{\beta}^{\text{pcr}}(M)$ which can be used for future predictions is given by

$$\hat{\beta}^{\text{pcr}}(M) = \sum_{m=1}^{M} \hat{\theta}_m v_m .$$

This algorithm is implemented in the `R` function `pcr_wwx.R`, and cross validation using this method is implemented in the function `cv_pcr_wwx.R`. A driver program that duplicates the results from `dup_OSE_PCR_N_PLSR.R` is implemented in `pcr_wwx_run.R`. This version of the PCR algorithm was written to ease transformation from `R` to a more traditional programming language like C++.

Note that the `R` package `pcr` [7] will implement this linear method and maybe more suitable for general use since it allows input via `R` formula objects and has significantly more options.

Notes on Incremental Forward Stagewise Regression

Since the Lasso appears to be a strong linear method that is discussed quite heavily in the book it would be nice to have some code to run that uses this method on a given problem of interest. There is the `R` package `glmnet` which solves a combined L_2 and L_1 constrained minimization problem but if a person is programming in a language other than `R` you would have to write your own L_1 minimization routine. This later task is relatively complicated. Fortunately from the discussion in the text one can get the performance benefit of a lasso type algorithm but using a much simpler computational algorithm: Incremental Forward Stagewise Regression. To verify that we understood this algorithm we first implemented it in the `R` function `IFSR.R`. One interesting thing about this algorithm is that the version given in the parametrized it based on ϵ, but provides no numerical details on how to specify this value. In addition, once ϵ has been specified we need to develop an appropriate stopping criterion. The book suggested to run the code until the residuals are uncorrelated with all of the predictors. In the version originally implemented the algorithm was allowed to loop until the *largest* correlation between the residual r and each feature x_j is smaller than a given threshold. One then has to pick the value of this threshold. Initially the value I selected was too large in that the p value $\text{cor}(x_j, r)$ *never* got small enough. I then added a maximum number of iterations to perform where in each iteration we step an amount ϵ in the j component of β. Again one needs to now specify a step size ϵ and a maximum number of iterations $N_{\max}$. If ϵ is taken very small then one will need to increase the value of $N_{\max}$. If ϵ is taken large one will need to decrease $N_{\max}$. While not difficult to modify these parameters and look at the profile plots of $\hat{\beta}$ for a single example it seemed useful to have a nice way of *automatically* determining ϵ given an value of N_{test}. To do that I found that the simple heuristic of

$$\epsilon = \frac{||\beta^{\text{LS}}||_1}{1.5 N_{\max}} \,. \tag{36}$$

gives a nice way to specify ϵ in terms of $N_{\max}$. Here $||\beta^{\text{LS}}||_1$ is the one norm of the least squares solution for $\hat{\beta}$ given by Equation 13. The motivation for this expression is that this algorithm starts at the value $\hat{\beta} = 0$ and takes "steps" of "size" ϵ towards β^{LS}. If we want a maximum of $N_{\max}$ steps then we should take ϵ of size $\frac{||\beta^{\text{LS}}||_1}{N_{\max}}$ so that we get there at the last step. The factor of 1.5 is to make the values of ϵ we use for stepping somewhat smaller. Another nice benefit of this approach is that the amount of computation for this algorithm then scales as $O(N_{\max})$ so depending on problem size one can pick a value of $N_{\max}$ that is reasonable as far as the required computational time. It is then easy to estimate the computational time if this algorithm was run with $2N_{\max}$. It seems more difficult to do that if the value of ϵ is given a priori and instead we are asked estimate the time to run the algorithm with $\frac{\epsilon}{2}$.

Exercise Solutions

Ex. 3.1 (the F-statistic is equivalent to the square of the Z-score)

Now in the definition of the F-statistic

$$F = \frac{(\mathrm{RSS}_0 - \mathrm{RSS}_1)/(p_1 - p_0)}{\mathrm{RSS}_1/(N - p_1 - 1)}\,, \tag{37}$$

we see from Equation 16 that the expression $\mathrm{RSS}_1/(N - p_1 - 1)$ in the denominator is equal to $\hat{\sigma}^2$. In addition, by just deleting one variable from our regression the difference in degrees of freedom between the two models is one i.e. $p_1 - p_0 = 1$. Thus the F-statistic when we delete the j-th term from the base model simplifies to

$$F_j = \frac{\mathrm{RSS}_j - \mathrm{RSS}_1}{\hat{\sigma}^2}\,.$$

Here the residual sum of squares of the larger model (with all terms) is denoted by RSS_1 and RSS_j is the residual sum of squares of the smaller model, obtained by omitted the j variable as a predictor.

Let v_{ij} be the entry in the i-th row and j-th column of the $(p+1)\times(p+1)$ matrix $(X^TX)^{-1}$. The j-th Z-score is defined (see the book's Equation 3.12) by

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{v_{jj}}}\,. \tag{38}$$

To show that $F_j = {z_j}^2$ we need to show that

$$\mathrm{RSS}_j - \mathrm{RSS}_1 = \frac{\hat{\beta}_j^2}{v_{jj}}\,,$$

which we now proceed to do.

Notice the implicit assumption in the definition of the Z-score that $v_{jj} > 0$. We prove this first. Let u range over all $(p+1)$-dimensional column vectors. Then $w = (X^TX)^{-1}u$ also does so. From the definition of w we have $u = (X^TX)w$ and can then write $u^T(X^TX)^{-1}u$ as

$$w^T(X^TX)(X^TX)^{-1}(X^TX)w = w^TX^TXw = (Xw)^T(Xw) \geq 0 \tag{39}$$

and equality implies that $Xw = 0$, hence $w = 0$ and therefore $u = 0$. So $(X^TX)^{-1}$ is a positive definite matrix. In particular, taking u to be the standard vector with all entries 0, except for 1 in the j-th place, we see that $v_{jj} > 0$.

Next note that $X^TX(X^TX)^{-1} = I_{p+1}$. Let u_j be the j-th column of $X(X^TX)^{-1}$, so that $x_i^Tu_j = \delta_{ij}$, the Kronecker delta. (Here x_i is the i-th column of X.) Using the v_{ij} notation to denote the elements of the matrix $(X^TX)^{-1}$ established above, we have

$$u_j = \sum_{r=1}^{p+1} x_r v_{rj}. \tag{40}$$

We have seen that, for $i \neq j$, u_j/v_{jj} is orthogonal to x_i, and the coefficient of x_j in u_j/v_{jj} is 1. Permuting the columns of X so that the j-th column comes last, we see that $u_j/v_{jj} = z_j$ (see Algorithm 3.1). By Equation 40,

$$\|u_j\|^2 = u_j^T u_j = \sum_{r=1}^{p+1} v_{rj} x_r^T u_j = \sum_{r=1}^{p+1} v_{rj}\delta_{rj} = v_{jj}.$$

Then

$$\|z_j\|^2 = \frac{\|u_j\|^2}{v_{jj}^2} = v_{jj}/v_{jj}^2 = \frac{1}{v_{jj}}. \tag{41}$$

Now $z_j/\|z_j\|$ is a unit vector orthogonal to $x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_{p+1}$. So

$$\begin{aligned} \mathrm{RSS}_j - \mathrm{RSS}_1 &= \langle y, z_j/\|z_j\| \rangle^2 \\ &= \left(\frac{\langle y, z_j \rangle}{\langle z_j, z_j \rangle} \right)^2 \|z_j\|^2 \\ &= \hat{\beta}_j^2 / v_{jj}, \end{aligned}$$

where the final equality follows from Equation 41 and the book's Equation 3.28.

Ex. 3.2 (confidence intervals on a cubic equation)

In this exercise, we fix a value for the column vector $\beta = (\beta_0, \beta_1, \beta_2, \beta_3)^T$ and examine random deviations from the curve

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3.$$

For a given value of x, the value of y is randomized by adding a normally distributed variable with mean 0 and variance 1. For each x, we have a row vector $\mathbf{x} = (1, x, x^2, x^3)$. We fix N values of x. (In Figure 6 we have taken 40 values, evenly spaced over the chosen domain $[-2, 2]$.) We arrange the corresponding values of $\mathbf{x}$ in an $N \times 4$-matrix, which we call X, as in the text. Also we denote by $\mathbf{y}$ the corresponding $N \times 1$ column vector, with independent entries. The standard least squares estimate of β is given by $\hat{\beta} = \left(X^T X\right)^{-1} X^T \mathbf{y}$. We now compute a 95% confidence region around this cubic in two different ways.

In the first method, we find, for each x, a 95% confidence interval for the one-dimensional random variable $u = \mathbf{x}.\hat{\beta}$. Now y is a normally distributed random variable, and therefore so is $\hat{\beta} = (X^T X)^{-1} X^T \mathbf{y}$. Therefore, using the linearity of E,

$$\mathrm{Var}(u) = \mathrm{E}\left(\mathbf{x}\hat{\beta}\hat{\beta}^T\mathbf{x}^T\right) - \mathrm{E}\left(\mathbf{x}\hat{\beta}\right).\mathrm{E}\left(\hat{\beta}^T\mathbf{x}^T\right) = \mathbf{x}\mathrm{Var}(\hat{\beta})\mathbf{x}^T = \mathbf{x}\left(X^T X\right)^{-1}\mathbf{x}^T.$$

This is the variance of a normally distributed one-dimensional variable, centered at $\mathbf{x}.\beta$, and the 95% confidence interval can be calculated as usual as 1.96 times the square root of the variance.

In the second method, $\hat{\beta}$ is a 4-dimensional normally distributed variable, centered at β, with 4×4 variance matrix $\left(X^T X\right)^{-1}$. We need to take a 95% confidence region in 4-dimensional space. We will sample points $\hat{\beta}$ from the boundary of this confidence region, and, for each such sample, we draw the corresponding cubic in green in Figure 6 on page 56. To see what to do, take the Cholesky decomposition $U^T U = X^T X$, where U is upper triangular. Then $(U^T)^{-1}\left(X^T X\right)U^{-1} = I_4$, where I_4 is the 4×4-identity matrix. $U\hat{\beta}$ is a normally distributed 4-dimensional variable, centered at $U\beta$, with variance matrix

$$\mathrm{Var}(U\hat{\beta}) = \mathrm{E}\left(U\hat{\beta}\hat{\beta}^T U^T\right) - \mathrm{E}\left(U\hat{\beta}\right).\mathrm{E}\left(\hat{\beta}^T U^T\right) = U\left(X^T X\right)^{-1} U^T = I_4$$

It is convenient to define the random variable $\gamma = \hat{\beta} - \beta \in \mathbb{R}^4$, so that $U\gamma$ is a standard normally distributed 4-dimensional variable, centered at 0.

Using the R function `qchisq`, we find r^2, such that the ball B centered at 0 in $\mathbb{R}^4$ of radius r has χ^2_4-mass equal to 0.95, and let ∂B be the boundary of this ball. Now $U\gamma \in \partial B$ if and only if its euclidean length squared is equal to r^2. This means

$$r^2 = \|U\gamma\|^2 = \gamma^T U^T U\gamma = \gamma^T X^T X\beta.$$

Given an arbitrary point $\alpha \in \mathbb{R}^4$, we obtain $\hat{\beta}$ in the boundary of the confidence region by first dividing by the square root of $\gamma^T X^T X\gamma$ and then adding the result to β.

Note that the Cholesky decomposition was used only for the theory, not for the purposes of calculation. The theory could equally well have been proved using the fact that every real positive definite matrix has a real positive definite square root.

Our results for one particular value of β are shown in Figure 6 on page 56.

Ex. 3.3 (the Gauss-Markov theorem)

(a) Let b be a column vector of length N, and let $\mathrm{E}(b^T y) = \alpha^T \beta$. Here b is fixed, and the equality is supposed true for all values of β. A further assumption is that X is not random. Since $\mathrm{E}(b^T y) = b^T X\beta$, we have $b^T X = \alpha^T$. We have

$$\mathrm{Var}(\alpha^T\hat{\beta}) = \alpha^T(X^T X)^{-1}\alpha = b^T X(X^T X)^{-1}X^T b,$$

and $\mathrm{Var}(b^T y) = b^T b$. So we need to prove $X(X^T X)^{-1}X^T \preceq I_N$.

To see this, write $X = QR$ where Q has orthonormal columns and is $N \times p$, and R is $p \times p$ upper triangular with strictly positive entries on the diagonal. Then $X^T X = R^T Q^T QR = R^T R$. Therefore $X(X^T X)^{-1}X^T = QR(R^T R)^{-1}R^T Q^T = QQ^T$. Let $[QQ_1]$ be an orthogonal $N \times N$ matrix. Therefore

$$I_N = \begin{bmatrix} Q & Q_1 \end{bmatrix} . \begin{bmatrix} Q^T \\ Q_1^T \end{bmatrix} = QQ^T + Q_1 Q_1^T.$$

Since $Q_1 Q_1^T$ is positive semidefinite, the result follows.

(b) Let C be a constant $p \times N$ matrix, and let $C\mathbf{y}$ be an estimator of β. We write $C = (X^T X)^{-1} X^T + D$. Then $\mathrm{E}(C\mathbf{y}) = \left((X^T X)^{-1} X^T + D\right) X\beta$. This is equal to β for all β if and only if $DX = 0$. We have $\mathrm{Var}(C\mathbf{y}) = CC^T\sigma^2$, Using $DX = 0$ and $X^T D^T = 0$, we find

$$\begin{aligned} CC^T &= \left((X^T X)^{-1} X^T + D\right)\left((X^T X)^{-1} X^T + D\right)^T \\ &= (X^T X)^{-1} + DD^T \\ &= \mathrm{Var}(\hat{\beta})\sigma^{-2} + DD^T. \end{aligned}$$

The result follows since DD^T is a positive semidefinite $p \times p$-matrix.

(a) again. Here is another approach to (a) that follows (b), the proof for matrices. Let c be a length N row vector and let $c\mathbf{y}$ be an estimator of $\alpha^T\beta$. We write $c = \alpha^T \left((X^T X)^{-1} X^T\right) + d$. Then $\mathrm{E}(c\mathbf{y}) = \alpha^T\beta + dX\beta$. This is equal to $\alpha^T\beta$ for all β if and only if $dX = 0$. We have $\mathrm{Var}(c\mathbf{y}) = cc^T\sigma^2$, Using $dX = 0$ and $X^T d^T = 0$, we find

$$\begin{aligned} cc^T &= \left(\alpha^T \left((X^T X)^{-1} X^T\right) + d\right) . \left(\alpha^T \left((X^T X)^{-1} X^T\right) + d\right)^T \\ &= \alpha^T (X^T X)^{-1} \alpha + dd^T \end{aligned}$$

The result follows since dd^T is a non-negative number.

Ex. 3.4 (the vector of least squares coefficients from Gram-Schmidt)

The values of $\hat{\beta}_i$ can be computed by using Equation 22, where Q and R are computed from the Gram-Schmidt procedure on X. As we compute the columns of the matrix Q in the Gram-Schmidt procedure we can evaluate $q_j^T y$ for each column q_j of Q, and fill in the jth element of the vector $Q^T y$. *After* the matrices Q and R are computed one can then solve

$$R\hat{\beta} = Q^T y\,. \tag{42}$$

for $\hat{\beta}$ to derive the entire set of coefficients. This is simple to do since R is upper triangular and is performed with back-substitution, first solving for $\hat{\beta}_{p+1}$, then $\hat{\beta}_p$, then $\hat{\beta}_{p-1}$, and on until $\hat{\beta}_0$. A componentwise version of backwards substitution is presented in almost every linear algebra text.

Note: I don't see a way to compute β_i *at the same time* as one is computing the columns of Q. That is, as *one* pass of the Gram-Schmidt algorithm. It seems one needs to othogonalize X first then solve Equation 42 for $\hat{\beta}$.

Ex. 3.5 (an equivalent problem to ridge regression)

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

$$\sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} \bar{x}_j\beta_j - \sum_{j=1}^{p} (x_{ij} - \bar{x}_j)\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2\,. \tag{43}$$

From this we see that by defining "centered" values of β as

$$\begin{aligned}\beta_0^c &= \beta_0 + \sum_{j=1}^{p} \bar{x}_j \beta_j \\ \beta_j^c &= \beta_i \qquad i = 1, 2, \ldots, p\,,\end{aligned}$$

that the above can be recast as

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

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.

We compute the value of β_0^c in the above expression by setting the derivative with respect to this variable equal to zero (a consequence of the expression being at a minimum). We obtain

$$\sum_{i=1}^{N} \left(y_i - \beta_0^c - \sum_{j=1}^{p} (x_{ij} - \bar{x}_j)\, \beta_j \right) = 0,$$

which implies $\beta_0^c = \bar{\mathbf{y}}$, the average of the y_i. The same argument above can be used to show that the minimization required for the lasso can be written in the same way (with β_j^{c2} replaced by $|\beta_j^c|$). The intercept in the centered case continues to be $\bar{\mathbf{y}}$.

WWX: This is as far as I have proofed

Ex. 3.6 (the ridge regression estimate)

Note: I used the notion in original problem in [6] that has τ^2 rather than τ as the variance of the prior. Now from Bayes' rule we have

$$\begin{aligned} p(\beta|\mathcal{D}) &\propto p(\mathcal{D}|\beta)p(\beta) && (44)\\ &= \mathcal{N}(y - X\beta, \sigma^2 I)\mathcal{N}(0, \tau^2 I) && (45)\end{aligned}$$

Now from this expression we calculate

$$\begin{aligned} \log(p(\beta|\mathcal{D})) &= \log(p(\mathcal{D}|\beta)) + \log(p(\beta)) && (46)\\ &= C - \frac{1}{2}\frac{(y - X\beta)^T(y - X\beta)}{\sigma^2} - \frac{1}{2}\frac{\beta^T\beta}{\tau^2} && (47)\end{aligned}$$

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

$$\hat{\beta} = \text{ArgMin}(-2\sigma^2 \log(p(\beta|\mathcal{D})) = \text{ArgMin}((y - X\beta)^T(y - X\beta) + \frac{\sigma^2}{\tau^2}\beta^T\beta) \tag{48}$$

Since this is the equivalent to Equation 3.43 page 60 in [6] with the substitution $\lambda = \frac{\sigma^2}{\tau^2}$ we have the requested equivalence.

Exs3.6 and 3.7 These two questions are almost the same; unfortunately they are both somewhat wrong, and in more than one way. This also means that the second-last paragraph on page 64 and the second paragraph on page 611 are both wrong. The main problem is that β_0 is does not appear in the penalty term of the ridge expression, but it does appear in the prior for β. . In Exercise 3.6, β has variance denoted by τ, whereas the variance is denoted by τ^2 in Exercise 3.7. We will use τ^2 throughout, which is also the usage on page 64.

With $X = (x_1, \ldots, x_p)$ fixed, Bayes' Law states that $p(\mathbf{y}|\beta).p(\beta) = p(\beta|\mathbf{y}).p(\mathbf{y})$. So, the posterior probability satisfies

$$p(\beta|\mathbf{y}) \propto p(\mathbf{y}|\beta).p(\beta),$$

where p denotes the pdf and where the constant of proportionality does not involve β. We have

$$p(\beta) = C_1 \exp\left(-\frac{\|\beta\|^2}{2\tau^2}\right)$$

and

$$p(\mathbf{y}|\beta) = C_2 \exp\left(-\frac{\|\mathbf{y} - X\beta\|^2}{2\sigma^2}\right)$$

for appropriate constants C_1 and C_2. It follows that, for a suitable constant C_3,

$$p(\beta|\mathbf{y}) = C_3 \exp\left(-\frac{\|\mathbf{y} - X\beta\|^2 + (\sigma^2/\tau^2).\|\beta\|^2}{2\sigma^2}\right) \tag{49}$$

defines a distribution for β given $\mathbf{y}$, which is in fact a normal distribution, though not centered at 0 and with different variances in different directions.

We look at the special case where $p = 0$, in which case the penalty term disappears and ridge regression is identical with ordinary linear regression. We further simplify by taking $N = 1$, and $\sigma = \tau = 1$. The ridge estimate for β_0 is then

$$\text{argmin}_\beta \left\{(\beta_0 - y_1)^2\right\} = y_1.$$

The posterior pdf is given by

$$p(\beta_0|y_1) = C_4 \exp\left(-\frac{(y_1 - \beta_0)^2 + \beta_0^2}{2}\right) = C_5 \exp\left(-\left(\beta_0 - \frac{y_1}{2}\right)^2\right),$$

which has mean, mode and median equal to $y_1/2$, NOT the same as the ridge estimate y_1. Since the integral of the pdf with respect to β_0(for fixed y_1) is equal to 1, we see that $C_5 = 1/\sqrt{2\pi}$. Therefore the log-posterior is equal to

$$-\log(2\pi)/2 + \left(\beta_0 - \frac{y_1}{2}\right) = -\log(2\pi)/2 + (\beta_0 - y_1)^2$$

To retrieve a reasonable connection between ridge regression and the log-posterior, we need to restrict to problems where $\beta_0 = 0$. In that case, the claim of Exercise 3.6 that the ridge estimate is equal to the mean (or the mode) of the posterior distribution becomes true.

We set $\lambda = \sigma^2/\tau^2$. From Equation 49 on page 36 for the posterior distribution of β, we find that the minus log-posterior is not proportional to

$$\sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

as claimed, because the term $\log(C_3)$ has been omitted.

Ex. 3.8 (when is the QR decomposition equivalent to the SVD decomposition)

This exercise is true if X has rank $p+1$, and is false otherwise. Let $X = QR$, where $Q = (q_0, \ldots, q_p)$ is $N \times (p+1)$ with orthonormal columns, and R is upper triangular with strictly positive diagonal entries. We write the entries of R as r_{kj} in the k-th row and j-th column, where $0 \leq k, j \leq p$. Let e be the length N column matrix consisting entirely of ones. Then $e = r_{00}q_0$. We deduce that all the entries of q_0 are equal. Since $\|q_0\| = 1$ and $r_{00} > 0$, we see that $q_0 = e/\sqrt{N}$ and that $r_{00} = \sqrt{N}$. The columns of Q form a basis for the column-space of X. Therefore the columns of Q_2 form a basis for the orthogonal complement of e in the column-space of X. For $1 \leq j \leq p$, we have

$$\bar{q}_j = \sum_{i=1}^{N} q_{ij}/N = e^T.q_j/N = q_0^T.q_j/\sqrt{N} = 0.$$

Let $X = (e, x_1, \ldots, x_p) = QR$. Then $x_j = \sum_{k=0}^{j} r_{kj}q_k$, and so $\bar{x}_j = r_{0j}/\sqrt{N}$. We have

$$\bar{x}_j e = r_{0j}q_0, \text{ and so } x_j - \bar{x}_j e = \sum_{k=1}^{p} r_{kj}q_k. \tag{50}$$

Let R_2 be the lower right $p \times p$ submatrix of R. Then

$$R = \begin{pmatrix} \sqrt{N} & \sqrt{N}(\bar{x}_1, \ldots, \bar{x}_p) \\ 0 & R_2 \end{pmatrix}.$$

Using Equation 50 above, we have

$$Q_2 R_2 = \widetilde{\mathbf{X}} = UDV^T. \tag{51}$$

Since X is assumed to have rank $p+1$, DV^T is a non-singular $p \times p$ matrix. It follows that U, $\widetilde{\mathbf{X}}$ and Q_2 have the same column space.

We assume that, by "up to sign flips", the authors mean that the columns (as opposed to the rows) of Q_2 and U are the same, up to sign. In this case, multiplying Equation 51 by

Q_2^T, we see that $R_2 = D_1 D V^T$, where D_1 is a diagonal matrix with entries ± 1. Since V is an orthogonal matrix, R_2 has orthogonal rows. Also R_2 has strictly positive diagonal entries. It follows that R_2 is a diagonal matrix, and so the columns of $\widetilde{\mathbf{X}}$ are orthogonal to each other. Therefore V is also diagonal, and the entries must all be ± 1.

Now let's look at the converse, where we suppose that $\widetilde{\mathbf{X}}$ has orthogonal columns. Since the QR decomposition is unique (see ???), we see that the QR decomposition of $\widetilde{\mathbf{X}} = Q_2 R_2$, where R_2 is diagonal with strictly positive entries. This is also an SVD, with $U = Q_2$ and $V = I_p$. However, it is not true that $U = Q_2$ for every SVD. Suppose, for example $\widetilde{\mathbf{X}} = I_n$ is the identity matrix. Then $Q_2 = R_2 = I_n$, and we can take $U = V$ to be any orthogonal matrix of the right size.

Ex. 3.9 (using the QR decomposition for fast forward-stepwise selection)

If we fit an ordinary least squares linear model with q terms then the QR decomposition of X_1 or the equation $X_1 = QR$ and Equation 23 expresses $\hat{y}$ as a sum of q columns of Q. Thus $\hat{y}$ is the span of the columns of Q. Since Q is related to X_1 by the Gram-Schmidt orthogonilization process $\hat{y}$ is also in the span of the q columns of X_1. By properties of least square estimates the residual $r = y - \hat{y}$ is in the orthogonal complement of the columns of both Q and X_1. To pick the next column x_j of the $p - q$ possible choices to add next, the one we should pick should be the column that had the largest projection in the direction of r. Thus we want to pick x_j that is most parallel with r to add next. Thus pick j^* such that

$$j^* = \text{argmin}_j \frac{|x_j^T r|}{||x_j||},$$

This x_j will reduce the residual sum of squares the most.

Note: I don't see explicitly why we need the QR algorithm in this expression.

Let $X_1 = QR$ be the QR decomposition, and let z_i be the i-th column of Q ($1 \leq i \leq q$). So $Q^T Q = I_q$ and R is upper triangular with strictly positive diagonal entries. The collection $\{z_1, \ldots, z_q\}$ is an orthonormal basis of C_0, the column space of X_1. C_0 is also the column space of Q. Let $\hat{y}$ be the orthogonal projection of the N-dimensional vector y to C_0.

For each j ($q < j \leq p$), let C_j be the subspace of $\mathbb{R}^N$ spanned by C_0 and x_j, the $(j-q)$-th column of X_2. We define

$$u_j = x_j - \sum_{i=1}^{q} (z_i^T x_j) z_i \text{ and} \tag{52}$$

$$v_j = u_j / \|u_j\| \tag{53}$$

and so $\{z_1, \ldots, z_q, v_j\}$ is an orthonormal basis of C_j. Now $\hat{y}_j = \hat{y} + (v_j^T y) v_j$ is the orthogonal projection of y to C_j, and so the residual sum of squares decreases by $(v_j^T y)^2$ if $\hat{y}$ is replaced by $\hat{y}_j$. It follows that the best variable to include is the k-th, where $k = \text{argmax}_{q<j\leq p} \left| v_j^T y \right|$. We then set $z_{q+1} = v_k$.

Note that, during the computation, we need to know $z_i^T x_j$ for all i and j ($1 \leq i \leq q$ and $q < j \leq p$. We have therefore done already completed of the computation necessary to include yet another variable in our Stepwise Forward Linear Regression. So this seems reasonably efficient, and we know of no more efficient algorithm (but there may be a more efficient algorithm that we don't know about).

Ex. 3.10 (using the z-scores for fast backwards stepwise regression)

The F-statistic

$$F = \frac{(\mathrm{RSS}_0 - \mathrm{RSS}_1)/(p_1 - p_0)}{\mathrm{RSS}_1/(N - p_1 - 1)},$$

when we drop a single term from the *larger* model (with residual sum of squares given by RSS_1) we will have $p_1 - p_0 = 1$, since the change in the degrees of freedom between the two models is only one. In this case when we drop the j-th term from the regression the F-statistic simplifies slightly and becomes

$$F_j = \frac{\mathrm{RSS}_j - \mathrm{RSS}_1}{\mathrm{RSS}_1/(N - p_1 - 1)}.$$

This is a scaled version (scaled by the division of the expression $\mathrm{RSS}_1/(N - p_1 - 1)$) of the *increase* in the residual sum of squares we observe when we drop the j-th term. From Exercise 3.1 this expression is equal to z_j^2 the square of the j-th z-score

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{v_j}}.$$

Thus by picking the index j^* that is associated with the *smallest* z-score and then deleting that variable from our regression we will be selecting the variable x_{j^*} that when deleted from the model will increase the residual sum of squares the least.

Ex. 3.11 (multivariate linear regression with different Σ_i)

The question refers to the book's Equations 3.39 and 3.40. To recall the notation, the following variables refer to matrices of the given sizes: $\mathbf{Y}$ size $N \times K$, $\mathbf{X}$ size $N \times (p+1)$, $\mathbf{B}$ size $(p+1) \times K$. Equation 3.38 gives the sum of squares as

$$\mathrm{RSS}(\mathbf{B}) = \mathrm{tr}\left[(\mathbf{Y} - \mathbf{XB})(\mathbf{Y} - \mathbf{XB})^T\right].$$

In the book, the two factors are written in the other order. We are using the fact that $\mathrm{tr}(UV) = \mathrm{tr}(VU)$ for any matrices U and V, provided both products make sense.

To prove Equation 3.39, we equate to zero the derivative of $\mathrm{RSS}(\mathbf{B})$ with respect to $\mathbf{B}$. Using the fact that the trace of a square matrix is equal to the trace of its transpose, we see that this condition on the derivative is equivalent to the condition that, for all $(p+1) \times K$ matrices $\mathbf{A}$, we have

$$\mathrm{tr}((\mathbf{XA})^T(\mathbf{Y} - \mathbf{X}\hat{\mathbf{B}})) = 0.$$

Let $\mathbf{A}$ be equal to zero, except for a 1 in row j and column k. The condition becomes $\sum_i x_{ij}\left(y_{ik} - \sum_s x_{is}b_{sk}\right) = 0$, for each j and k. But this is the condition $\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\hat{\mathbf{B}}) = 0$. Multiplying on the left by the inverse of $\mathbf{X}^T\mathbf{X}$, we obtain Equation 3.39.

In order to deal with the situation of Equation 3.40, we rewrite it as

$$\mathrm{RSS}(\mathbf{B}, \mathbf{\Sigma}) = \mathrm{tr}\left[(\mathbf{Y} - \mathbf{X}\mathbf{B})\mathbf{\Sigma}^{-1}(\mathbf{Y} - \mathbf{X}\mathbf{B})^T\right].$$

As a positive definite symmetric matrix, $\mathbf{\Sigma}^{-1}$ has a positive definite symmetric square root, which we denote by $\mathbf{S}$, a $K \times K$ matrix. We replace $\mathbf{Y}$ by $\mathbf{YS}$ and $\mathbf{B}$ by $\mathbf{BS}$. This reduces us to the previous case, which, by Equation 3.39, gives a minimum value when $\hat{\mathbf{B}}\mathbf{S} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}\mathbf{S}$, and we multiply on the right by $\mathbf{S}^{-1}$ to obtain the usual formula (Equation 3.39) for $\hat{\mathbf{B}}$.

Now suppose that the correlations $\mathbf{\Sigma}_i$ vary from one sample to the next. We will not go into the question of how to estimate these correlations, which seems to be possible only under some additional assumptions. Instead we assume that these correlations are known. Then there is no closed formula like Equation 3.39 for $\hat{\mathbf{B}}$. However the corresponding sum of squares is, in general, a positive definite quadratic form whose variables are the entries of $\mathbf{B}$. The argmin is therefore easy to find, using the specific values of $\mathbf{X}$, $\mathbf{Y}$. This can be done either with a computer program or analytically by diagonalizing the quadratic form.

Ex. 3.12 (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

$$\hat{X} = \begin{bmatrix} X \\ \sqrt{\lambda} I_{\mathrm{pxp}} \end{bmatrix} \tag{54}$$

and

$$\hat{Y} = \begin{bmatrix} Y \\ \mathcal{O}_{\mathrm{px1}} \end{bmatrix} \tag{55}$$

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

$$\hat{\beta}_{\mathrm{LS}} = (\hat{X}^T\hat{X})^{-1}\hat{X}^T\hat{Y} \tag{56}$$

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

$$\hat{X}^T\hat{X} = \left[X^T\sqrt{\lambda}I_{\mathrm{pxp}}\right]\begin{bmatrix} X \\ \sqrt{\lambda} I_{\mathrm{pxp}} \end{bmatrix} = X^TX + \lambda I_{\mathrm{pxp}} \tag{57}$$

and

$$\hat{X}^T\hat{Y} = \left[X^T\sqrt{\lambda}\right]\begin{bmatrix} Y \\ \mathcal{O}_{\mathrm{px1}} \end{bmatrix} = X^TY \tag{58}$$

Thus equation 56 becomes

$$\hat{\beta}_{\mathrm{LS}} = (X^TX + \lambda I_{\mathrm{pxp}})^{-1}X^TY \tag{59}$$

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

In a slightly different direction, but perhaps related, I was thinking of sending an email to the authors with some comments on some of the questions. For example, I think 3.7 (online) is incorrect, as there should be an additive constant as well as a multiplicative constant. The question is of course almost the same as 3.6, except for the spurious change from τ to τ^2. There are sometimes questions that involve some mind-reading by the reader. For example, I don't like the word "characterize" in 3.5. According to the dictionary, this means "describe the distinctive character of". Mind-reading is required. Another bugbear is the word "Establish" in the online Exercise 2.7(d). And there aren't two cases, there are at least four (nearest neighbour, linear, unconditional, conditional on keeping $\mathcal{X}$ fixed, more if k is allowed to vary.) And do they mean a relationship between a squared bias and variance in each of these four cases?

Ex. 3.13 (principal component regression)

Recall that principal component regression (PCR) using M components, estimates coefficients $\hat{\theta}_m$, based on the top M largest variance principal component vectors z_m. As such it has a expression given by

$$\begin{aligned} \hat{y}^{\text{pcr}}(M) &= \bar{y}\mathbf{1} + \sum_{m=1}^{M} \hat{\theta}_m z_m \\ &= \bar{y}\mathbf{1} + X \sum_{m=1}^{M} \hat{\theta}_m v_m \,, \end{aligned} \tag{60}$$

using the fact that $z_m = X v_m$ and writing the fitted value under PCR as a function of the number of retained components M. The above can be written in matrix form involving the data matrix as

$$\hat{y}^{\text{pcr}}(M) = \begin{bmatrix} \mathbf{1} & X \end{bmatrix} \begin{bmatrix} \bar{y} \\ \sum_{m=1}^{M} \hat{\theta}_m v_m \end{bmatrix} .$$

We can write this as the matrix $\begin{bmatrix} \mathbf{1} & X \end{bmatrix}$ times a vector $\hat{\beta}^{\text{pcr}}$ if we take this later vector as

$$\hat{\beta}^{\text{pcr}}(M) = \begin{bmatrix} \bar{y} \\ \sum_{m=1}^{M} \hat{\theta}_m v_m \end{bmatrix} . \tag{61}$$

This is the same as the books equation 3.62 when we restrict to just the last p elements of $\hat{\beta}^{\text{pcr}}$. It can be shown (for example in [10]) that for multiple linear regression models the estimated regression coefficients in the vector $\hat{\beta}$ can be split into two parts a scalar $\hat{\beta}_0$ and a $p \times 1$ vector $\hat{\beta}^*$ as

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}^* \end{bmatrix} .$$

In addition the coefficient $\hat{\beta}_0$ can be shown to be related to $\hat{\beta}^*$ as

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}^* \bar{x} \,,$$

where $\bar{y}$ is the scalar mean of the response vector y and $\bar{x}$ is the $p \times 1$ vector where each component hold the mean of a predictor say x_i. Since from Equation 61 we see that $\hat{\beta}_0 = \bar{y}$ and can conclude that in principal component regression $\bar{x} = 0$ or that the input predictor variables are standardized.

To evaluate $\hat{\beta}^{\rm pcr}(M)$ when $M = p$ recall that in principal component regression the vectors v_m are from the SVD of the matrix X given by $X = UDV^T$. This later expression is equivalent to $XV = UD$ from which if we equate the columns of the matrices on both sides we get

$$\begin{aligned} Xv_1 &= d_1 u_1 \\ Xv_2 &= d_2 u_2 \\ &\vdots \\ Xv_p &= d_p u_p \,. \end{aligned}$$

Since the vectors z_m are given by $z_m = Xv_m$ for $1 \le m \le M \le p$, by the above we have the vectors z_m in terms of the vectors u_m as $z_m = d_m u_m$. Since for this problem we are to show that when $M = p$ the estimate $\hat{\beta}^{\rm pcr}(M)$ above becomes the least squares estimate $\hat{\beta}^{\rm ls}$ it is helpful to see what this later estimate looks like interms of the SVD matrices U, D, and V. Recognizing that when $\lambda = 0$ the ridge regression estimate for β is equal to the least squares estimate (when $\lambda = 0$ we have no $\lambda||\beta||_2^2$ ridge penalty term) we can use Equation 29 to obtain

$$\hat{\beta}^{\rm ls} = \hat{\beta}^{\rm ridge}(\lambda = 0) = VD^{-2}DU^T y = VD^{-1}U^T y \,. \tag{62}$$

We now see if we can transform $\hat{\beta}^{\rm pcr}(p)$ into this expression. We have

$$\hat{\beta}^{\rm pcr}(p) = \sum_{m=1}^{p} \hat{\theta}_m v_m = V \begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \\ \vdots \\ \hat{\theta}_p \end{bmatrix} = V \begin{bmatrix} \frac{<z_1,y>}{<z_1,z_1>} \\ \frac{<z_2,y>}{<z_2,z_2>} \\ \vdots \\ \frac{<z_p,y>}{<z_p,z_p>} \end{bmatrix} .$$

Using $z_m = d_m u_m$ derived above we have

$$< z_m, z_m >= d_m^2 < u_m, u_m >= d_m^2 \,,$$

since the vectors u_m are assumed to be orthonormal. Also $< z_m, y >= d_m < u_m, y >$, so the estimate for $\hat{\beta}^{\rm pcr}(p)$ becomes

$$\hat{\beta}^{\rm pcr}(p) = V \begin{bmatrix} \frac{<u_1,y>}{d_1} \\ \frac{<u_2,y>}{d_2} \\ \vdots \\ \frac{<u_p,y>}{d_p} \end{bmatrix} = VD^{-1}U^T y \,,$$

which by Equation 62 equals $\hat{\beta}^{\rm ls}$ as we were to show.

Ex. 3.14 (when the inputs are orthogonal PLS stops after $m = 1$ step)

Observe that in Algorithm 3.3 on partial least squares (PLS) if on any given step m we compute that $\hat{\phi}_{mj} = 0$ for all j then it follows that the algorithm must stop. For this problem when the x_j's are orthonormal we will explicitly walk through the first step of the algorithm by hand and show that $\hat{\phi}_{2j} = 0$ for all $1 \leq j \leq p$.

Note that the x_j's to be orthogonal means that $x_i^T x_j = 0$ for all $i \neq j$. Following each step of Algorithm 3.3 and starting with $m = 1$ we have

(a) We have that z_1 is given by

$$z_1 = \sum_{j=1}^{p} \hat{\phi}_{1j} x_j^{(0)} \quad \text{with} \quad \hat{\phi}_{1j} =< x_j^{(0)}, y > \, .$$

(b) Next we compute $\hat{\theta}_1 = \frac{<z_1, y>}{<z_1, z_1>}$. The denominator in the above fraction is given by

$$\begin{aligned} < z_1, z_1 > &= \langle \sum_{j=1}^{p} \hat{\phi}_{1j} x_j^{(0)}, \sum_{j=1}^{p} \hat{\phi}_{1j} x_j^{(0)} \rangle = \sum_{j=1}^{p} \sum_{j'=1}^{p} \hat{\phi}_{1j} \hat{\phi}_{1j'} \langle x_j^{(0)}, x_{j'}^{(0)} \rangle \\ &= \sum_{j=1}^{p} \sum_{j'=1}^{p} \hat{\phi}_{1j} \hat{\phi}_{1j'} \delta_{jj'} = \sum_{j=1}^{p} \hat{\phi}_{1j}^2 \, , \end{aligned}$$

since the vectors $x_j^{(0)}$ are orthogonal. The numerator in the above expression for $\hat{\theta}_1$ is given by

$$< z_1, y >= \sum_{j=1}^{p} \hat{\phi}_{1j} < x_j^{(0)}, y >= \sum_{j=1}^{p} \hat{\phi}_{1j}^2 \, ,$$

and thus we have $\hat{\theta}_1 = 1$.

(c) Next we find $\hat{y}^{(1)}$ given by

$$\hat{y}^{(1)} = \hat{y}^{(0)} + z_1 = \hat{y}^{(0)} + \sum_{j=1}^{p} \hat{\phi}_{1j} x_j^{(0)} \, .$$

(d) Next we compute $x_j^{(1)}$ for each value of $j = 1, 2, \ldots, p$ using

$$x_j^{(1)} = x_j^{(0)} - \frac{< z_1, x_j^{(0)} >}{< z_1, z_1 >} z_1 \, .$$

Since the vectors $x_j^{(0)}$ are orthogonal the inner product in the numerator above becomes

$$< z_1, x_k^{(0)} >= \sum_{j=1}^{p} \hat{\phi}_{1j} < x_j^{(0)}, x_k^{(0)} >= \hat{\phi}_{1k} \, .$$

Using this result $x_j^{(1)}$ becomes

$$\begin{aligned} x_j^{(1)} &= x_j^{(0)} - \frac{\hat{\phi}_{1j}}{\sum_{j=1}^p \hat{\phi}_{1j}^2} z_1 \\ &= x_j^{(0)} - \left(\frac{\hat{\phi}_{1j}}{\sum_{j=1}^p \hat{\phi}_{1j}^2} \right) \sum_{j=1}^p \hat{\phi}_{1j} x_j^{(0)} . \end{aligned}$$

Having finished the first loop of Algorithm 3.3 we let $m = 2$ and compute $\hat{\phi}_{2j}$ to get

$$\begin{aligned} \hat{\phi}_{2j} &= < x_j^{(1)}, y > = < x_j^{(0)}, y > - \left(\frac{\hat{\phi}_{1j}}{\sum_{j=1}^p \hat{\phi}_{1j}^2} \right) \sum_{j=1}^p \hat{\phi}_{1j}^2 \\ &= 0 . \end{aligned}$$

Thus as discussed at the beginning of this problem since $\hat{\phi}_{2j} = 0$ the algorithm must stop.

Ex. 3.15 (PLS seeks directions that have high variance and high correlation)

This problem has not yet been worked.

Ex. 3.16 (explicit expressions for $\hat{\beta}_j$ when the features are orthonormal)

When the predictors are orthonormal $X^T X = I$ and the ordinary least squared estimate of β is given by

$$\hat{\beta} = X^T Y . \tag{63}$$

In best-subset selection we will take the top M predictors that result in the smallest residual sum of squares. Since the columns of X are orthonormal we can construct a basis for $\mathbb{R}^N$ by using the first p columns of X and then extending these with $N - p$ linearly independent additional orthonormal vectors. The Gram-Schmidt procedure guarantees that we can do this. Thus in this extended basis we can write y as

$$y = \sum_{j=1}^p \hat{\beta}_j x_j + \sum_{j=p+1}^N \gamma_j \tilde{x}_j . \tag{64}$$

Where $\hat{\beta}_j$ equal the components of $\hat{\beta}$ in Equation 63, $\tilde{x}_j$ are the extended basis vectors required to span $\mathbb{R}^N$, and γ_j are the coefficients of y with respect to these extended basis vectors. Then if we seek to approximate y with a subset of size M as in best subset selection our $\hat{y}$ can be written as $\hat{y} = \sum_{j=1}^p I_j \hat{\beta}_j x_j$, with $I_j = 1$ if we keep the predictor x_j and zero

otherwise. Now since all the vectors x_j and $\tilde{x}_j$ are orthonormal we have

$$\begin{aligned} ||y-\hat{y}||_2^2 &= ||y-X\hat{\beta}||_2^2 = \left|\left|\sum_{j=1}^{p}\hat{\beta}_j(1-I_j)x_j + \sum_{j=p+1}^{N}\gamma_j\tilde{x}_j\right|\right|_2^2 \\ &= \sum_{j=1}^{p}\hat{\beta}_j^2(1-I_j)^2\,||x_j||_2^2 + \sum_{j=p+1}^{N}{\gamma_j}^2\,||\tilde{x}_j||_2^2 \\ &= \sum_{j=1}^{p}\hat{\beta}_j^2(1-I_j)^2 + \sum_{j=p+1}^{N}{\gamma_j}^2\,. \end{aligned}$$

Thus to minimize $||y-\hat{y}||_2^2$ we would pick the M values of I_j to be equal to one that have the largest $\hat{\beta}_j^2$ values. This is equivalent to sorting the values $|\hat{\beta}_j|$ and picking the indices of the largest M of these to have $I_j = 1$. All other indices j would be taken to have $I_j = 0$. Using an indicator function this is equivalent to the expression

$$\hat{\beta}_j^{\text{best-subset}} = \hat{\beta}_j I[\text{rank}(|\hat{\beta}_j|) \leq M]\,. \tag{65}$$

For ridge regression, since X has orthonormal columns we have

$$\begin{aligned} \hat{\beta}^{\text{ridge}} &= (X^TX+\lambda I)^{-1}X^Ty \\ &= (I+\lambda I)^{-1}X^Ty = \frac{1}{1+\lambda}X^Ty \\ &= \frac{1}{1+\lambda}\hat{\beta}^{\text{ls}}\,, \end{aligned}$$

which is the desired expression.

For the lasso regression procedure we pick the values of β_j to minimize

$$\text{RSS}(\beta) = (y-X\beta)^T(y-X\beta) + \lambda\sum_{j=1}^{p}|\beta_j|\,.$$

Expanding $\hat{y}$ as $\hat{y} = \sum_{j=1}^{p}\beta_j x_j$ and with y expressed again as in Equation 64 we have that $\text{RSS}(\beta)$ in this case becomes

$$\begin{aligned} \text{RSS}(\beta) &= \left|\left|\sum_{j=1}^{p}(\hat{\beta}_j-\beta_j)x_j + \sum_{j=p+1}^{N}\gamma_j\tilde{x}_j\right|\right|_2^2 + \lambda\sum_{j=1}^{p}|\beta_j| \\ &= \sum_{j=1}^{p}(\hat{\beta}_j-\beta_j)^2 + \sum_{j=p+1}^{N}\gamma_j^2 + \lambda\sum_{j=1}^{p}|\beta_j| \\ &= \sum_{j=1}^{p}\{(\hat{\beta}_j-\beta_j)^2 + \lambda|\beta_j|\} + \sum_{j=p+1}^{N}\gamma_j^2\,. \end{aligned}$$

We can minimize this expression for each value of β_j for $1 \leq j \leq p$ independently. Thus our vector problem becomes that of solving p scalar minimization problems all of which look like

$$\beta^* = \text{argmin}_\beta\left\{(\hat{\beta}-\beta)^2 + \lambda|\beta|\right\}\,. \tag{66}$$

In this expression $\hat{\beta}$ and λ are assumed fixed. This expression can be represented as the sum of two terms $(\hat{\beta}-\beta)^2$ and $\lambda|\beta|$. The first expression $(\hat{\beta}-\beta)^2$ is symmetric about the least squares estimate $\hat{\beta}$ while the second expression is symmetric about $\beta = 0$. To get an idea of what this objective function looks like we take some representative values of $\hat{\beta}$ and λ and plot of the sum of these two functions we get the plots in Figure 7.

Then the objective function $F(\beta)$ in Equation 66 we want to minimize is

$$F(\beta) = \begin{cases} (\beta-\hat{\beta})^2 - \lambda\beta & \text{when} \quad \beta < 0 \\ (\beta-\hat{\beta})^2 + \lambda\beta & \text{when} \quad \beta > 0 \end{cases} .$$

To find the minimum of this function take the derivative with respect to β and set the result equal to zero and solve for β. We find the derivative of $F(\beta)$ given by

$$F'(\beta) = \begin{cases} 2(\beta-\hat{\beta}) - \lambda & \beta < 0 \\ 2(\beta-\hat{\beta}) + \lambda & \beta > 0 \end{cases} .$$

When we set $F'(\beta)$ equal to zero we get two possible solutions for β given by

$$\begin{array}{ll} \beta = +\frac{\lambda}{2} + \hat{\beta} & \text{and} \quad \beta < 0 \\ \beta = -\frac{\lambda}{2} + \hat{\beta} & \text{and} \quad \beta > 0 \end{array} .$$

Note: I want this to be equal to the desired expression but I seem to be off by a factor of 1/2... did I do something wrong?

Ex. 3.17 (linear methods on the spam data set)

For this problem we were asked to apply various linear methods to predict whether or not a piece of email is spam or not. This is very similar to the example presented in the book on the prostate data set. We implemented each of the various methods: Ordinary Least Squares (OLS), ridge regression, the lasso (a regression that imposes an L_1 penalty on the vector of least squares coefficient β), principal component regression (PCR), and partial least squares (PLS). As a guide to what `R` functions perform what computations and produce the above plots and table entries we have:

- The least squares (LS) results are obtained in the script `spam_OLS.R`.
- The ridge regression results are obtained using the script `spam_ridge.R`.
- The lasso results are obtained using the script `spam_lasso.R`.
- The PCR and PLS results are obtained using the script `spam_PCR_N_PLSR.R`.

Each of these script generates one of the plots given in Figure 8 Note that since the spam data set has so many predictor variables (57) it is not possible (with the code written for the prostate data set) to perform an exhaustive search over all possible subsets of size $1 \leq k \leq 57$

as in performed in `dup_OSE_all_subset.R` for the prostate data set. One could drop some variables from the spam data to produce a smaller set of variables and then run such an exhaustive search procedure if desired. The fact that we *can* run the other routines on this problem is an argument for their value. When the above codes are run we obtain Table 4 comparing their performance.

	LS	Ridge	Lasso	PCR	PLS
Test Error	0.121	0.117	0.123	0.123	0.119
Std Error	0.007	0.004	0.007	0.007	0.005

Table 4: Duplicated results for the books Table 3.3 but applied to the `spam` data set. Note that ridge regression and partial least squares outperforms ordinary least squares.

Ex. 3.18 (conjugate gradient methods)

The conjugate gradient method is an algorithm that can be adapted to find the minimum of nonlinear functions [8]. It also provides a way to solve certain linear algebraic equations of the type $Ax = b$ by formulating them as a minimization problem. Since the least squares solution $\hat{\beta}$ is given by the solution of a quadratic minimization problem given by

$$\hat{\beta} = \text{argmin}_\beta \text{RSS}(\beta) = \text{argmin}_\beta (y - X\beta)^T (y - X\beta) \,. \tag{67}$$

which has the explicit solution given by the normal equations or

$$(X^T X)\beta = X^T Y \,. \tag{68}$$

One of the properties of the conjugate gradient algorithm is that when it is used to solve $Ax = b$ where x is an $p \times 1$ vector it will terminate with the exact solution after p iterations. Thus during is operation we get a sequence of p approximate values of the solution vector x.

One obvious way to use the conjugate gradient algorithm to derive refined estimates of $\hat{\beta}$ the least squared solution is to use this algorithm to solve the normal Equations 68. Thus after finishing each iteration of the conjugate gradient algorithm we have a new approximate value of $\hat{\beta}^{(m)}$ for $m = 1, 2, \cdots, p$ and when $m = p$ this estimate $\hat{\beta}^{(p)}$ corresponds to the least squares solution. The value of m is a model selection parameter and can be selected by cross validation. To derive the algorithm just described one would then need to specify the conjugate gradient algorithm so that it was explicitly solving the normal Equations 68.

Warning: The problem with the statements just given is that while viewing the conjugate gradient algorithm as a iterative algorithm to compute $\hat{\beta}$ seems to be a reasonable algorithm, the algorithm given in the book for partial least squares does not seem to be computing iterative estimates for $\hat{\beta}^{\text{LS}}$ but instead each iteration seems to be computing an improved estimates of $\mathbf{y}$. While these two related as $\hat{y}^{(m)} = X\hat{\beta}^{(m)}$, I don't see how to modify the partial least squares algorithm presented in the book into one that is approximating $\hat{\beta}$. If anyone sees a simple way to do this or knows of a paper/book that describes this transformation please let the authors know.

Ex. 3.19 (increasing norm with decreasing λ)

To begin with we will use Equation 29 to compute $||\hat{\beta}^{\text{ridge}}||_2^2$. We find

$$\begin{aligned} ||\hat{\beta}^{\text{ridge}}||_2^2 &= y^T U D (D^2 + \lambda I)^{-1} V^T V (D^2 + \lambda I)^{-1} D U^T y \\ &= y^T U D (D^2 + \lambda I)^{-2} D U^T y \\ &= (U^T y)^T [D (D^2 + \lambda I)^{-2} D] (U^T y) \,. \end{aligned}$$

The matrix in the middle of the above is a diagonal matrix with elements $\frac{d_j^2}{(d_j^2+\lambda)^2}$. Thus

$$||\hat{\beta}^{\text{ridge}}||_2^2 = \sum_{i=1}^{p} \frac{d_j^2 (U^T y)_j{}^2}{(d_j^2 + \lambda)^2} \,.$$

Where $(U^T y)_j$ is the jth component of the vector $U^T y$. As $\lambda \to 0$ we see that the fraction $\frac{d_j^2}{(d_j^2+\lambda)^2}$ increases, and because $||\hat{\beta}^{\text{ridge}}||_2^2$ is made up of sum of such terms, it too must increase as $\lambda \to 0$.

To determine if the same properties hold For the lasso, note that in both ridge-regression and the lasso can be represented as the *minimization* problem

$$\begin{aligned} \hat{\beta} &= \text{argmin}_\beta \left(\sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 \right) \\ &\quad \text{subject to} \quad \sum_j |\beta_j|^q \le t \,. \end{aligned} \tag{69}$$

When $q = 1$ we have the lasso and when $q = 2$ we have ridge-regression. This form of the problem is equivalent to the *Lagrangian* from given by

$$\hat{\beta} = \text{argmin}_\beta \left(\sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|^q \right) \,. \tag{70}$$

Because as $\lambda \to 0$ the value of $\sum |\beta_j|^q$ needs to *increase* to have the product $\lambda \sum |\beta_j|^q$ stay constant and have the same error minimum value in Equation 70. Thus there is an *inverse* relationship between λ and t in the two problem formulations, in that as λ decreases t increases and vice versa. Thus the *same* behavior of t and $\sum_j |\beta_j|^q$ with decreasing λ will show itself in the lasso. This same conclusion can also be obtained by considering several of the other properties of the lasso

- The explicit formula for the coefficients $\hat{\beta}_j$ estimated by the lasso when the features are orthogonal is given. In that case it equals $\text{sign}(\hat{\beta}_j)(|\hat{\beta}_j| - \lambda)_+$. From this expression we see that as we decrease λ we increase the value of $\hat{\beta}_j$, and correspondingly the norm of the vector $\hat{\beta}$.

- In the books figure 3.11 since smaller λ (i.e. larger t values) correspond to larger neighborhoods about the origin and correspondingly an lasso based estimate for β that gets closer to $\hat{\beta}$ and correspondingly gets larger in magnitude.
- Numerical solutions for the lasso estimates of β for the `prostate` data set are presented in figure 3.10 as a function of $s = \frac{t}{\sum_1^p |\hat{\beta}_j|}$. We see that as t increases more and more β_j become "active" or non-zero. Correspondingly the norm increases.

Ex. 3.20 ()

This problem has not been worked.

Ex. 3.21 ()

This problem has not been worked.

Ex. 3.22 ()

This problem has not been worked.

Ex. 3.23 ($(X^TX)^{-1}X^Tr$ keeps the correlations tied and decreasing)

Part (a): Now in the expression $\frac{1}{N}|\langle x_j, y - u(\alpha)\rangle|$, $u(\alpha)$ is parametrized as $u(\alpha) = \alpha X\hat{\beta}$ where $\hat{\beta}$ is given by Equation 13 or the least squares fit. Then $\frac{1}{N}|\langle x_j, y - u(\alpha)\rangle|$ is the absolute value of the j component of

$$\begin{aligned}\frac{1}{N}X^T(y - u(\alpha)) &= \frac{1}{N}X^T(y - \alpha X(X^TX)^{-1}X^Ty) \\ &= \frac{1}{N}(X^Ty - \alpha X^Ty) \\ &= \frac{1}{N}(1-\alpha)X^Ty\,. \end{aligned} \tag{71}$$

Since in this problem we are told that the absolute value of each element of X^Ty is equal to $N\lambda$ we have from the above that $\frac{1}{N}X^T(y - u(\alpha)) = (1-\alpha)\lambda$, or looking at the jth row and taking absolute values of this expression we conclude that

$$\frac{1}{N}|\langle x_j, y - u(\alpha)\rangle| = (1-\alpha)\lambda\,,$$

for $j = 1, 2, \cdots, p$ as we were to show. In words the magnitude of the projections of x_j onto the residual $y - u(\alpha) = y - \alpha X\hat{\beta}$ is the same for every value of j.

Part (b): Warning: I don't see how to derive this expression for in Part (a) above when $\alpha = 0$, using the hypothesis for this problem we have that

$$\frac{1}{N}|\langle x_j, y\rangle| = \lambda\,.$$

In addition, as we more α from 0 to 1 the correlations given above are given by Equation 71 and so the functional form of $\lambda(\alpha)$ should just be *linear* in α and not have this square-root dependence. If anyone sees anything wrong with this logic please email me.

Part (c): From the given expression derived in Part (a) and (b) one sees that when $\alpha = 0$ we have $\lambda(0) = \lambda$, when $\alpha = 1$ we have that $\lambda(1) = 0$, where all correlations are tied and decrease from λ to zero as α moves from 0 to 1.

Ex. 3.24 (LAR directions)

From the definition of the LAR direction vector u_k we see that

$$\begin{aligned} X_{\mathcal{A}_k}^T u_k &= X_{\mathcal{A}_k}^T (X_{\mathcal{A}_k}\delta_k) \\ &= X_{\mathcal{A}_k}^T X_{\mathcal{A}_k}(X_{\mathcal{A}_k}^T X_{\mathcal{A}_k})^{-1} X_{\mathcal{A}_k}^T r_k \\ &= X_{\mathcal{A}_k}^T r_k\,. \end{aligned}$$

Since the cosign of the angle of u_k with each predictor x_j in $\mathcal{A}_k$ is given by $\frac{x_j^T u}{||x_j||\,||u||} = \frac{x_j^T u}{||u||}$ each element of the vector $X_{\mathcal{A}_k}^T u_k$ corresponds to a cosign of an angle between a predictor x_j and the vector u_k. Since the *procedure* for LAR adds the predictor x_j exactly when the absolute value of $x_j^T r$ equals that of $x_k^T r$ for all predictors x_k in $\mathcal{A}_k$, this direction u_k makes an equal angle with all predictors in $\mathcal{A}_k$.

Ex. 3.25 (LAR look-ahead)

Warning: I was unable to solve this problem. If anyone sees a way to solve it please let me know.

Ex. 3.26 (forward stepwise regression vs. LAR)

Warning: I was unable to solve this problem. If anyone sees a way to solve it please let me know.

Ex. 3.27 (Lasso and LAR)

Warning: I was unable to solve this problem. If anyone sees a way to solve it please let me know.

Ex. 3.28-3.29 ()

These problems have not yet been worked.

Ex. 3.30 (solving the elastic net optimization problem with the lasso)

For this problem note that if we augment X with a multiple of the $p \times p$ identity to get

$$\tilde{X} = \begin{bmatrix} X \\ \gamma I \end{bmatrix},$$

then $\tilde{X}\beta = \begin{bmatrix} X\beta \\ \gamma\beta \end{bmatrix}$. If we next augment y with p zero values as

$$\tilde{y} = \begin{bmatrix} y \\ 0 \end{bmatrix}.$$

Then we have

$$||\tilde{y} - \tilde{X}\beta||_2^2 = \left|\left|\begin{bmatrix} y - X\beta \\ \gamma\beta \end{bmatrix}\right|\right|_2^2 = ||y - X\beta||_2^2 + \gamma^2||\beta||_2^2. \tag{72}$$

Now in the this augmented space a lasso problem for β is

$$\hat{\beta} = \text{argmin}_\beta \left(||\tilde{y} - \tilde{X}\beta||_2^2 + \tilde{\lambda}||\beta||_1 \right).$$

Writing this using Equation 72 we get in the original variables the following

$$\hat{\beta} = \text{argmin}_\beta \left(||y - X\beta||_2^2 + \gamma^2||\beta||_2^2 + \tilde{\lambda}||\beta||_1 \right).$$

To make this match the requested expression we take $\gamma^2 = \lambda\alpha$ or $\gamma = \sqrt{\lambda\alpha}$, and $\tilde{\lambda} = \lambda(1-\alpha)$. Thus to solve the requested minimization problem given y, X, λ and α perform the following steps

- Augment y with p additional zeros to get $\tilde{y} = \begin{bmatrix} y \\ 0 \end{bmatrix}$.
- Augment X with the multiple of the $p \times p$ identity matrix $\sqrt{\lambda\alpha}I$ to get $\tilde{X} = \begin{bmatrix} X \\ \sqrt{\lambda\alpha}I \end{bmatrix}$.
- Set $\tilde{\lambda} = \lambda(1 - \alpha)$.
- Solve the lasso minimization problem with input $\tilde{y}$, $\tilde{X}$, and $\tilde{\lambda}$.

The solution $\hat{\beta}$ is the desired solution to the entire problem.

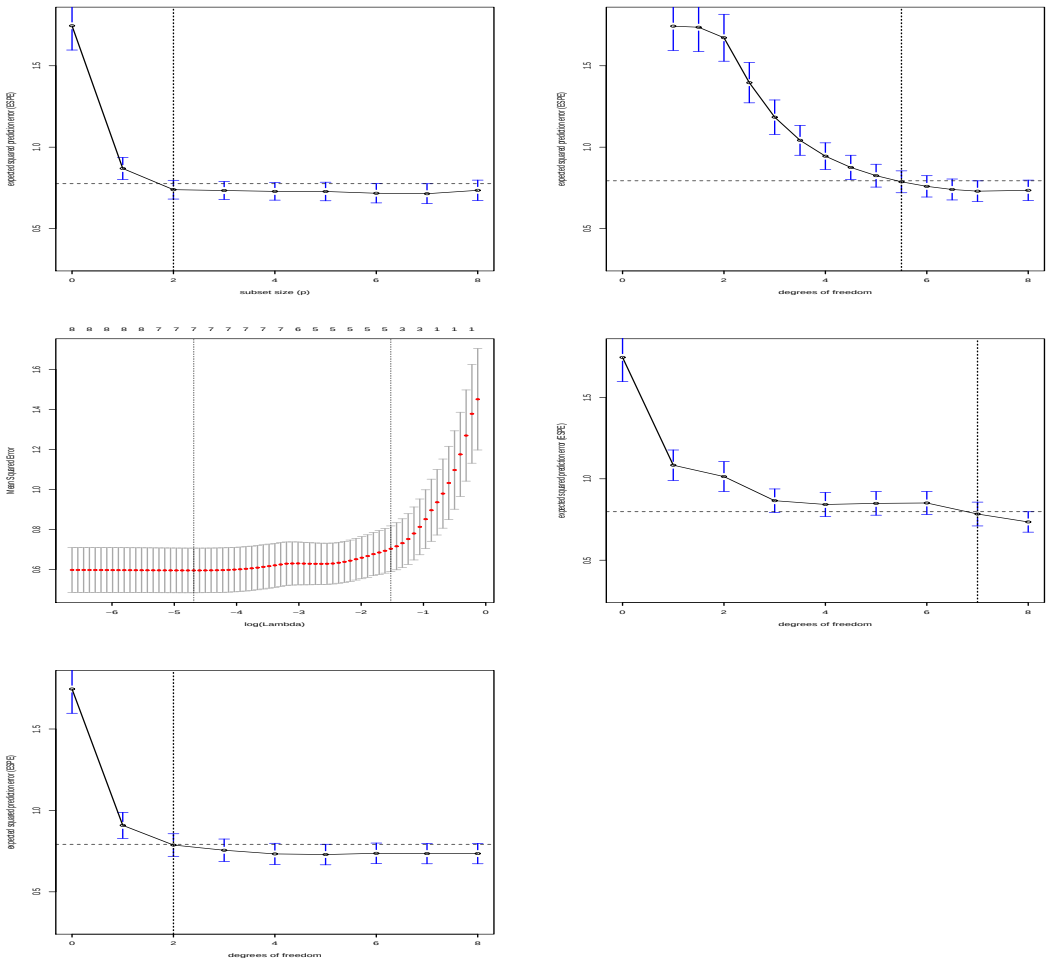

Figure 2: Numerical duplication of the books Figure 3.7. Estimated prediction error curves and their standard errors for various linear methods. From top to bottom, left to right the methods displayed are: Best-Subset Selection, Ridge Regression, The Lasso, Principal Components Regression (PCR), and Partial Least Squares (PLS). The lasso results look different than that presented in the book since we present the default plot result using the `glmnet` plot command. In addition, the PCR and PLS results don't return cross validation results for the case of no predictors (predicting with the constant mean $\bar{y}$ only). These later results should duplicate the first point in the best-subset selection.

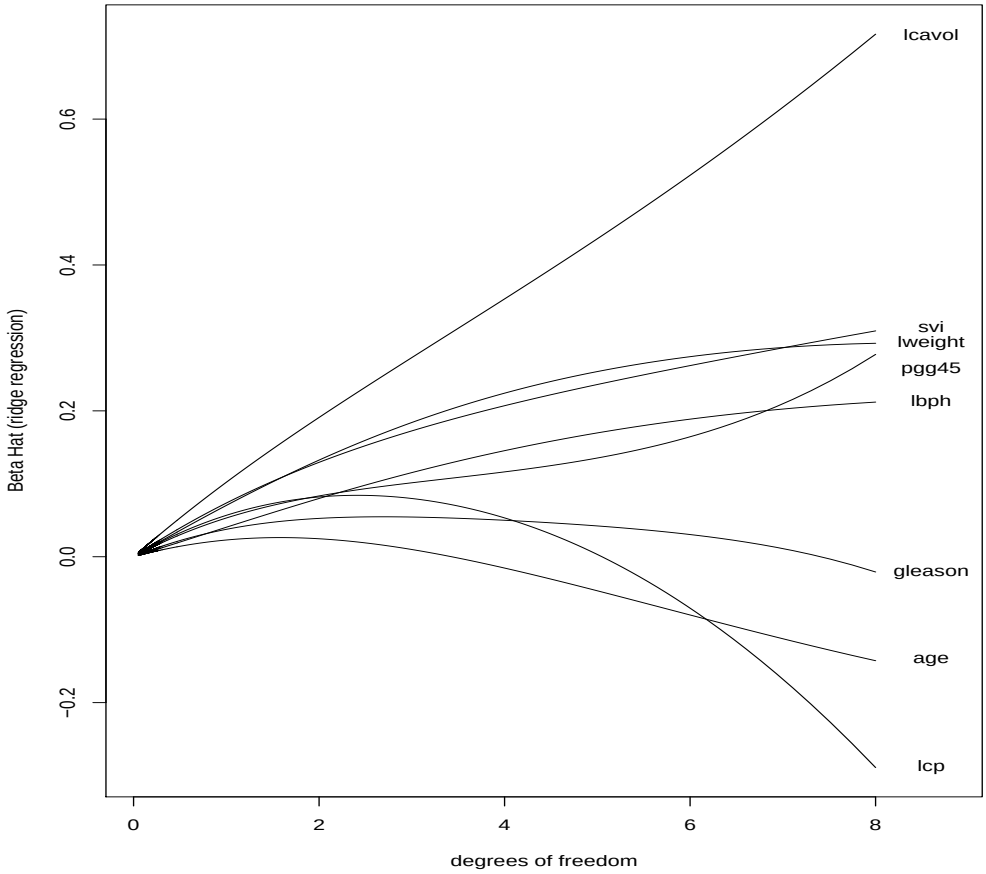

Figure 3: Duplication of the books Figure 3.8 using the code `duplicate_figure_3_8.R`.

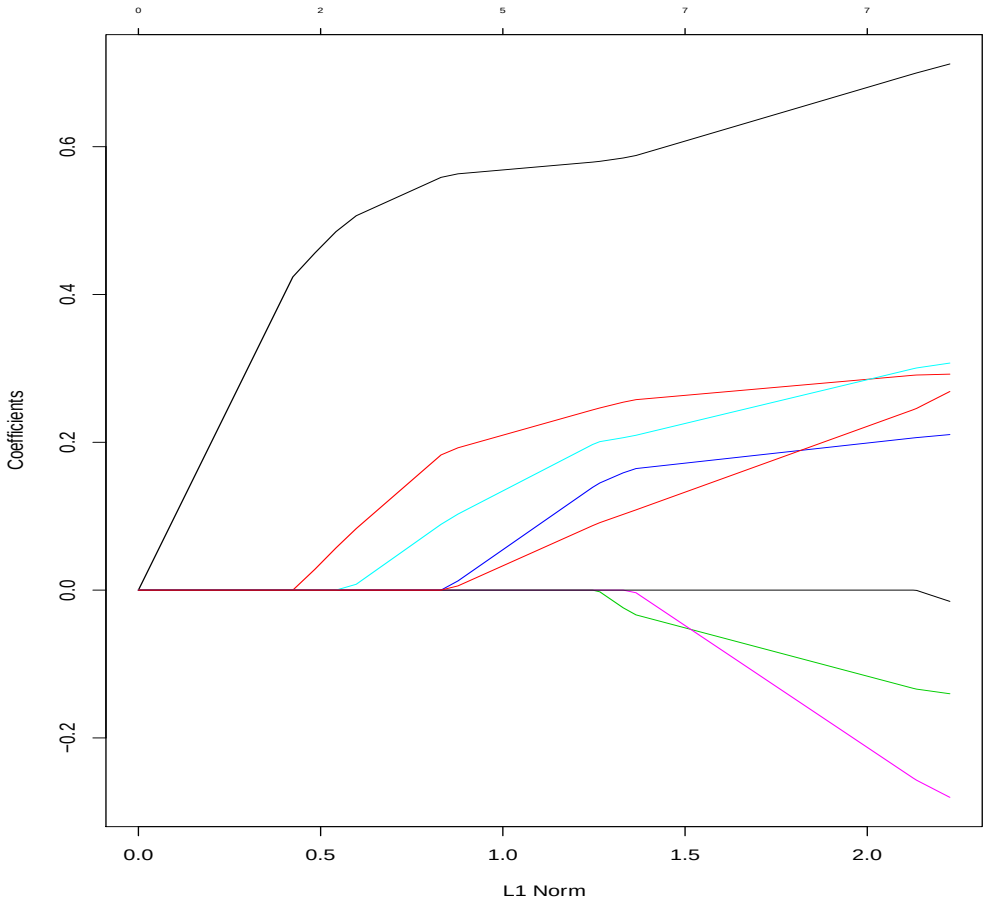

Figure 4: Duplication of the books Figure 3.10 using the code `dup_OSE_lasso.R`. This plot matches quite well qualitatively and quantitatively the corresponding one presented in the book.

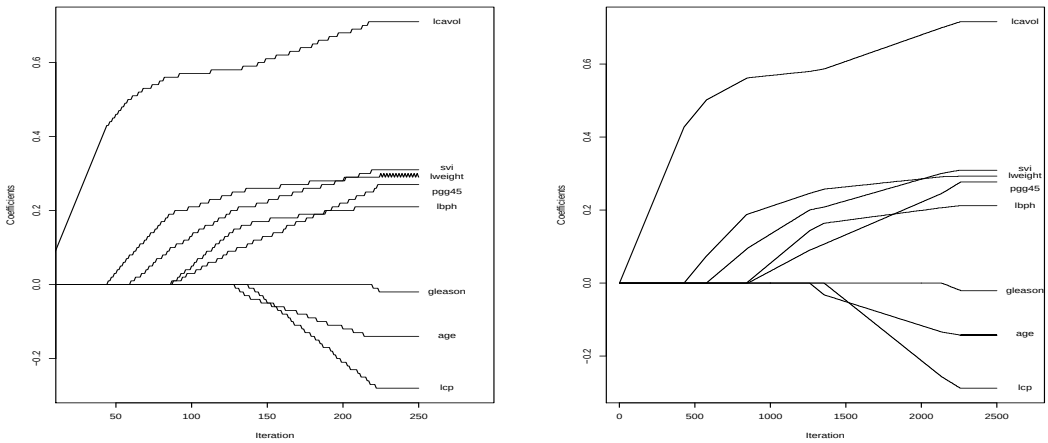

Figure 5: Numerical duplication of the books Figure 3.19. **Left:** Using the Incremental Forward Stagewise Regression (IFSR) with $\epsilon = 0.01$ and 250 iterations of the algorithm. **Right:** IFSR with $\epsilon = 0.001$ and 2500 iterations. With ϵ this small the curves are so smooth they approximate the lasso paths quite well.

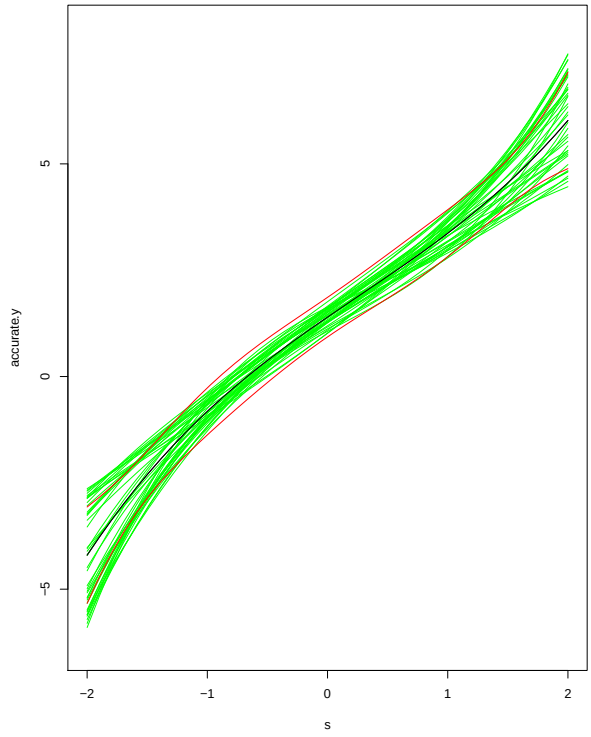

Figure 6: The middle black curve is the curve

$$y = 1.3965849 + 1.9407724x - 0.1215529x^2 + 0.1535441x^3.$$

The two red curves come from the upper and lower limits of the 95% confidence interval, taken separately at each value of x. The green curves are the result of sampling values from the boundary of the 4-dimensional 95% confidence region for values of $\hat{\beta}$, as determined by the χ^2_4 distribution, and then drawing the corresponding curve. Note that the green curves do not lie entirely between the red curves.

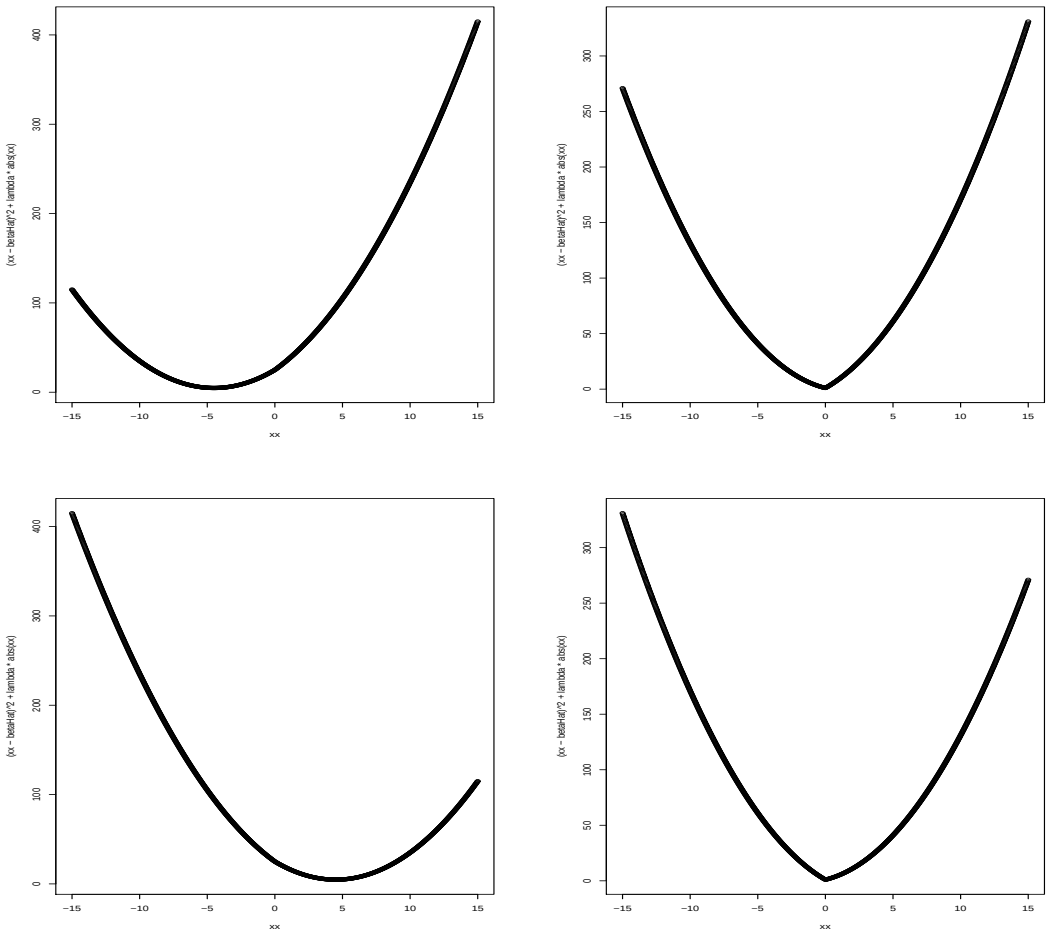

Figure 7: Plots of $(\hat{\beta} - \beta)^2 + \lambda|\beta|$ for various values of $\hat{\beta}$ and λ.
Upper Left: For $\hat{\beta} = -5.0$ and $\lambda = 1.0$. The minimum appears to be at $\beta = -5.0$.
Upper Right: For $\hat{\beta} = -1.0$ and $\lambda = 5.0$. The minimum appears to be at $\beta = 0.0$.
Lower Left: For $\hat{\beta} = 5.0$ and $\lambda = 1.0$. The minimum appears to be at $\beta = 5.0$.
Lower Right: For $\hat{\beta} = 1.0$ and $\lambda = 5.0$. The minimum appears to be at $\beta = 0.0$.

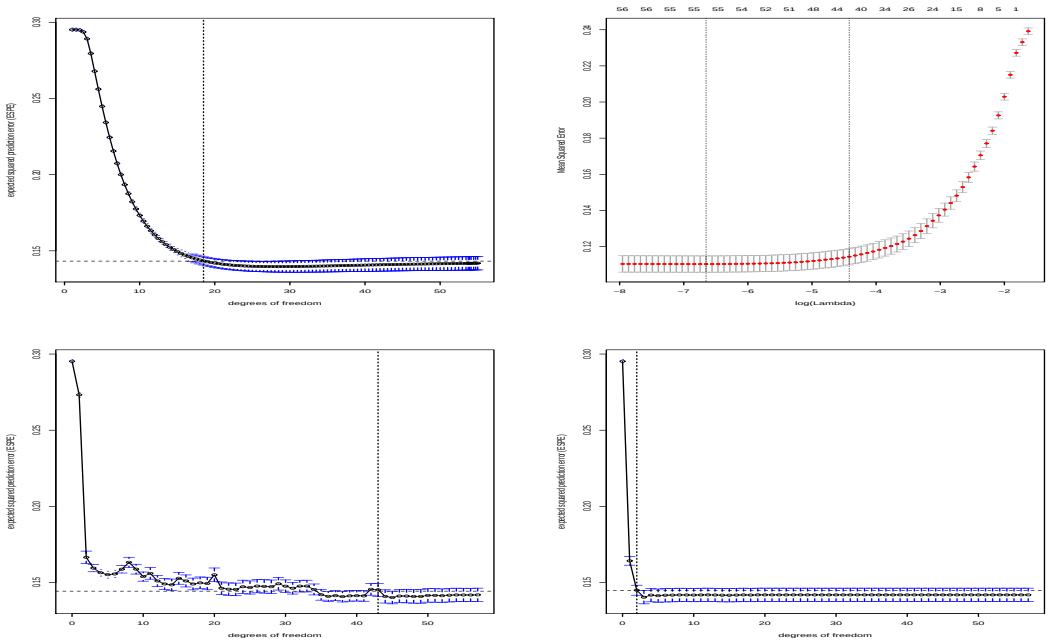

Figure 8: Estimated prediction error curves and their standard errors for various linear methods applied to the `spam` data set. From top to bottom, left to right the methods displayed are: Ridge Regression, The Lasso, Principal Components Regression (PCR), and Partial Least Squares (PLS). The lasso results look different than that presented in the book since we present the default plot result using the `glmnet` plot command. In addition, the PCR and PLS results don't return cross validation results for the case of no predictors (predicting with the constant mean $\bar{y}$ only).

Chapter 4 (Linear Methods for Classification)

Notes on the Text

Notes on using LDA as a dimensionality reduction technique

In the `R` code `dup_fig_4.4.R` we duplicate the results in the book where we project the vowel data set into a two dimensional space. When we run the above `R` code we get the result shown in Figure 9. This result looks very much like the result presented in the book.

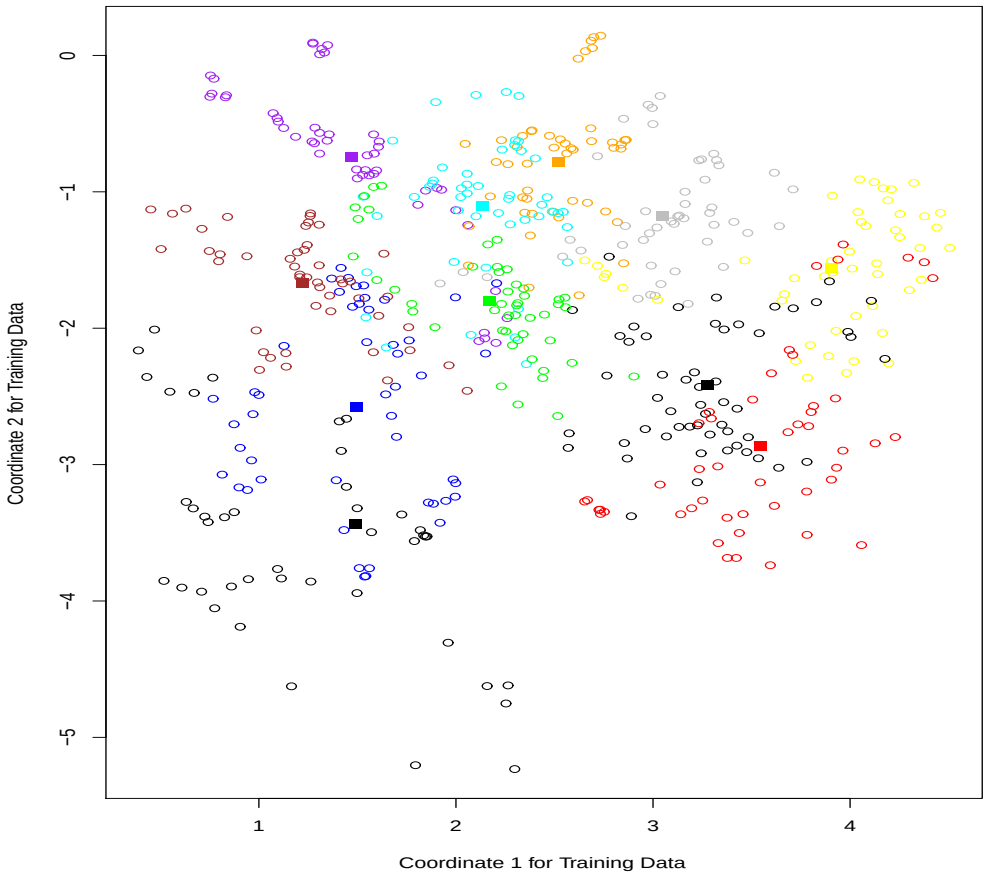

Figure 9: Duplication of the books Figure 4.4 using the code `dup_fig_4_4.R`.

Notes on Linear Discriminant Analysis

In this subsection of the book there were comments on a two class method for classification that was claimed to work quite well in practice. This novel method is based on an observation

made during the derivation of LDA in that the cut classification threshold point or the scalar value of the expression

$$\frac{1}{2}\hat{\mu}_2^T\hat{\Sigma}^{-1}\hat{\mu}_2 - \frac{1}{2}\hat{\mu}_1^T\hat{\Sigma}^{-1}\hat{\mu}_1 + \log\left(\frac{N_1}{N}\right) - \log\left(\frac{N_2}{N}\right) , \tag{73}$$

is only Bayes optimal when the two class conditional densities are Gaussian and have the same covariance matrix. Thus the conclusion suggested by the authors is that one might be able to improve classification by specifying a *different* value for this cut point. In the `R` function entitled `two_class_LDA_with_optimal_cut_point.R` we follow that suggestion and determine the cut point that minimizes the classification error rate over the training set. We do this by explicitly enumerating several possible cut points and evaluating the in-sample error rate of each parametrized classifier[1]. To determine the range of cut point to search for this minimum over, we first estimate the common covariance matrix $\hat{\Sigma}$, and the two class means $\hat{\mu}_1$ and $\hat{\mu}_2$ in the normal ways and then tabulate (over all of the training samples x) the left-hand-side of the book's equation 4.11 or

$$x^T\hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) . \tag{74}$$

Given the range of this expression we can sample cut points a number of times between the minimum and maximum given value, classify the points with the given cut point and estimating the resulting classifier error rate. The above code then returns the cut point threshold that produces the minimum error rate over the in-sample data. This code is exercised using the `R` script `two_class_LDA_with_optimal_cut_point_run.R` where we perform pairwise classification on two vowels. While the book claimed that this method is often superior to direct use of LDA running the above script seems to indicate that the two methods are very comparable. Running a pairwise comparison between the optimal cut point method shows that it is only better then LDA 0.29 of the time. Additional tests on different data sets is certainly needed.

Regularized Discriminant Analysis

Some `R` code for performing regularized discriminant analysis can be found in `rda.R`. This code is exercised (on the vowel data) in the `R` code `dup_fig_4_7.R` which duplicates the plot given in the book. When that script is run we get

```
[1] "Min test error rate=   0.478355; alpha=   0.969697"
```

and the plot shown in Figure 10. This plot matches well with the one given in the book. In addition, we can shrink $\hat{\Sigma}$ towards a scalar and consider the more general three term model

$$\hat{\Sigma}_k(\alpha,\gamma) = \alpha\hat{\Sigma}_k + (1-\alpha)(\gamma\hat{\Sigma} + (1-\gamma)\hat{\sigma}^2 I) .$$

With this form for $\hat{\Sigma}_k$ we can again run a grid search over the parameters α and γ and select values that optimize out of sample performance. When we do that (at the bottom of the `dup_fig_4_7.R`) we get

[1] A better algorithm but one that requires more computation would be to specify the classification point and estimate the error rate using leave-one-out cross-validation.

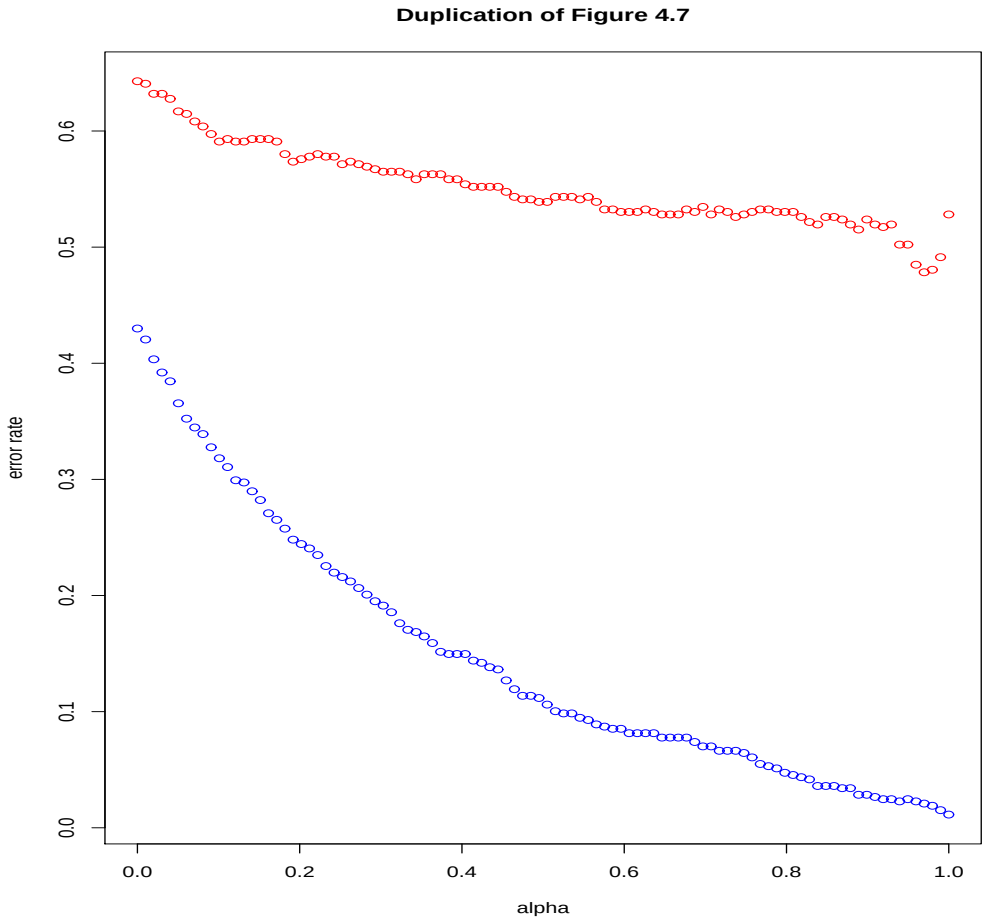

Figure 10: Duplication of the books Figure 4.7 using the code `dup_fig_4_7.R`.

```
[1] "Min test error rate=   0.439394; alpha=   0.767677; gamma=   0.050505"
```

Notes on duplication of Table 4.1

In this `dup_table_4_1.R` we call various `R` functions created to duplicate the results in the book from Table 4.1. This code uses the `R` routine `linear_regression_indicator_matrix.R` which does classification based on linear regression (as presented in the introduction to this chapter). When we run that code we get results given by

```
                                        Technique    Training        Test
[1] "                          Linear Regression:    0.477273;    0.666667"
[1] "      Linear Discriminant Analysis (LDA):    0.316288;    0.556277"
[1] "   Quadratic Discriminant Analysis (QDA):    0.011364;    0.528139"
[1] "                         Logistic Regression:    0.221591;    0.512987"
```

These results agree well with the ones given in the book.

Notes on Reduced-Rank Linear Discriminant Analysis

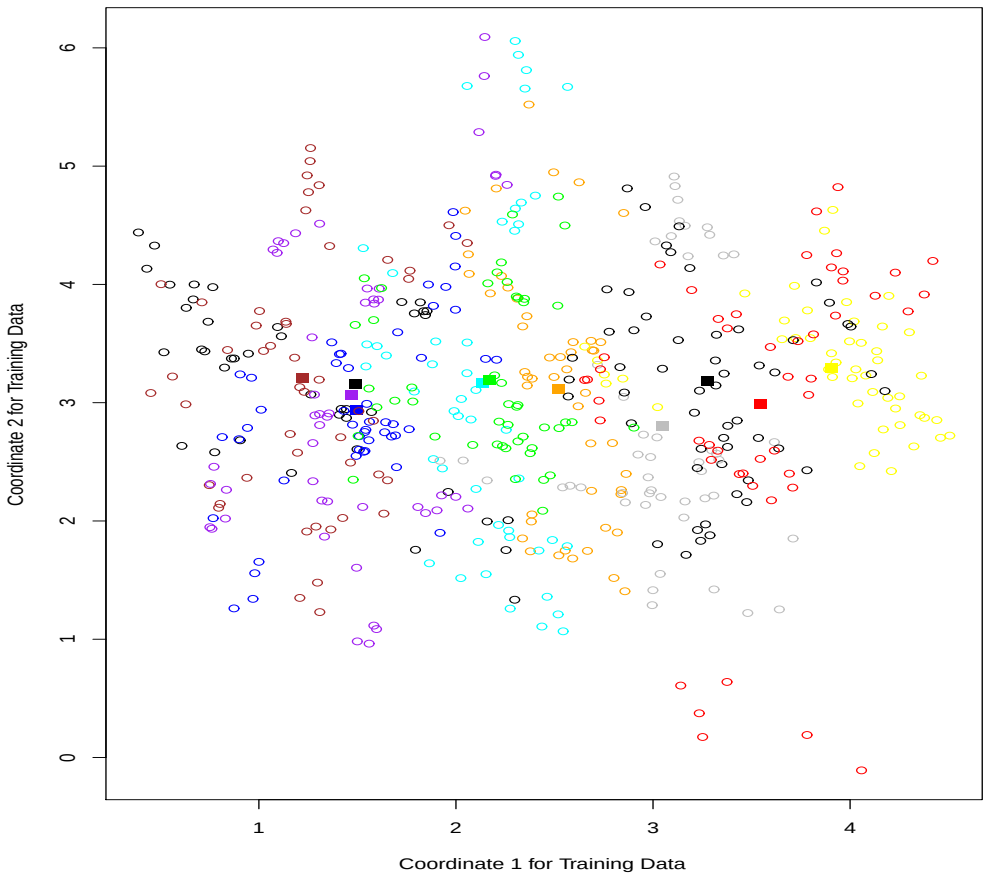

Figure 11: Duplication of the coordinate 1 - 7 projection found in the books Figure 4.8.

In this section of these notes we implement the reduced-rank linear discriminant analysis in the R code `reduced_rank_LDA.R`. Using this code in the R code `dup_fig_4_8.R` we can then duplicate the projections presented in figure 4.8 from the book. When we run that code we get the plot given in Figure 11. This plot matches the results in the book quite well. In the R code `dup_fig_4.10.R` we have code that duplicates figure 4.10 from the book. When that code is run we get the plot shown in Figure 12.

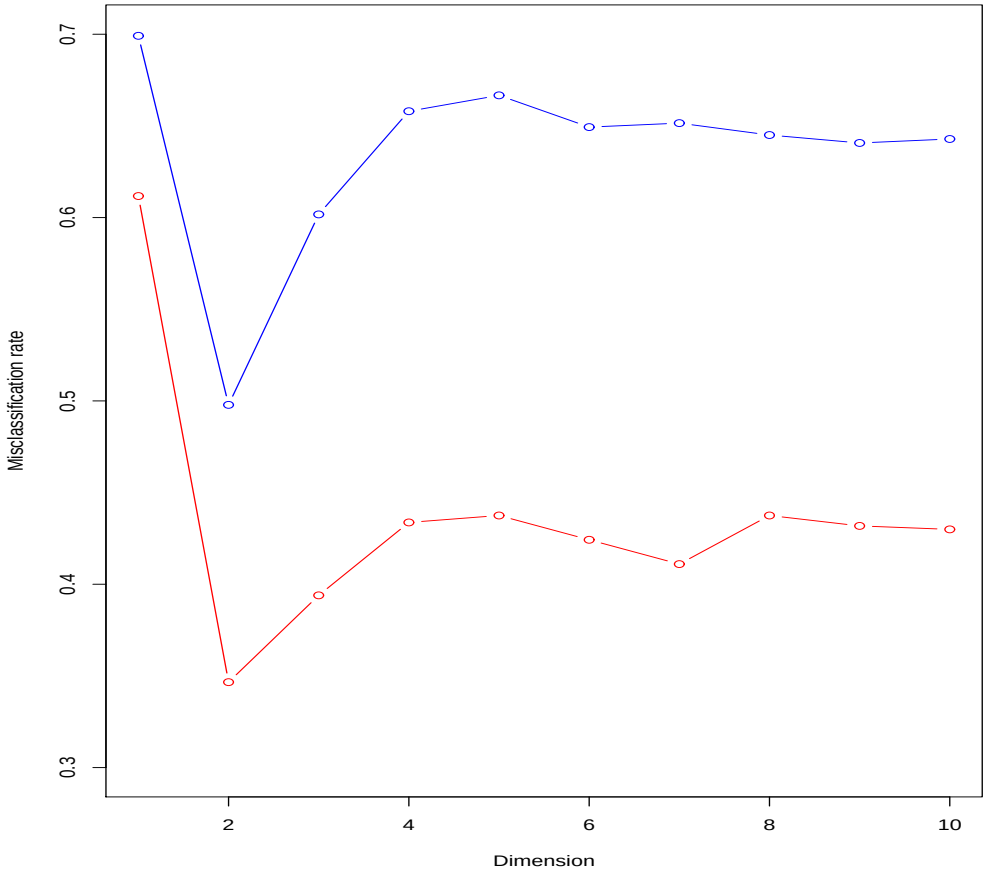

Figure 12: Duplication of the books Figure 4.10.

Logistic Regression

From the given specification of logistic regression we have

$$\begin{aligned}
\log\left(\frac{\Pr(G=1|X=x)}{\Pr(G=K|X=x)}\right) &= \beta_{10}+\beta_1^T x \\
\log\left(\frac{\Pr(G=2|X=x)}{\Pr(G=K|X=x)}\right) &= \beta_{20}+\beta_2^T x \\
&\vdots \\
\log\left(\frac{\Pr(G=K-1|X=x)}{\Pr(G=K|X=x)}\right) &= \beta_{(K-1)0}+\beta_{K-1}^T x\,.
\end{aligned} \tag{75}$$

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 75 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

$$\begin{aligned}
\Pr(G=1|X=x) &= \Pr(G=K|X=x)\exp(\beta_{10}+\beta_1^T x) \\
\Pr(G=2|X=x) &= \Pr(G=K|X=x)\exp(\beta_{20}+\beta_2^T x) \\
&\vdots \\
\Pr(G=K-1|X=x) &= \Pr(G=K|X=x)\exp(\beta_{(K-1)0}+\beta_{(K-1)}^T x)\,.
\end{aligned} \tag{76}$$

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

$$\Pr(G=K|X=x)=\frac{1}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l^T x)}\,. \tag{77}$$

When we put this expression in the proceeding $K-1$ in Equations 76 we find

$$\Pr(G=k|X=x)=\frac{\exp(\beta_{k0}+\beta_k^T x)}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l^T x)}\,, \tag{78}$$

which are equations 4.18 in the book.

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 posterior 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 measurement 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 succinctly represent the probability that x_i is a member of its class with the following notation

$$p_{g_i}(x_i)=\Pr(G=1|X=x_i)^{y_i}\Pr(G=2|X=x)^{1-y_i}\,. \tag{79}$$

Since only one of the values y_i or $1-y_i$ will in fact be non-zero. Using this notation given an entire data set of measurements and their 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-likelihood 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 79 we obtain

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

If we now use Equations 75 to express the log-posterior odds in terms of the parameters we desire to estimate β we see that

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

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_{10}. 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

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i (y_i - p(x_i))\,. \tag{80}$$

The score is a column vector and the Hessian or second derivative is a matrix. We can denote this notationally by taking the derivative $\frac{\partial}{\partial \beta^T}$ of the score column vector (note the transpose). We find

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

This expression in the summation is a row vector and is given by

$$\frac{\partial p(x_i)}{\partial \beta^T} = \frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}} x_i^T - \frac{(e^{\beta^T x_i})^2}{(1+e^{\beta^T x_i})^2} x_i^T = p(x_i)(1-p(x_i)) x_i^T\,.$$

Thus we get

$$\frac{\partial l(\beta)}{\partial \beta \partial \beta^T} = -\sum_{i=1}^{N} x_i x_i^T p(x_i)(1-p(x_i))\,, \tag{81}$$

for the Hessian.

Notes on the South African heart disease data

In this `dup_table_4_2.R` we call various `R` functions created to duplicate the results in the book from Table 4.2. This gives a result that matches the books result quite well. Next in the `R` code `dup_table_4_3.R` we duplicate the results found in Table 4.3 in the book. We use the `R` function `step` to systematically remove predictors from the heart disease model to end with the model `chd ~ tobacco + ldl + famhist + age`. After this model is found the `step` function tries to remove the remaining variables one at a time as can seen from the `step` output

```
Step:  AIC=495.44
chd ~ tobacco + ldl + famhist + age

          Df Deviance    AIC
<none>         485.44 495.44
- ldl      1   495.39 503.39
- tobacco  1   496.18 504.18
- famhist  1   502.82 510.82
- age      1   507.24 515.24
```

This shows that removing each of the three variables one at a time produces models that have a *larger* Akaike information criterion (AIC) (and deviance) vs. the model above where $AIC = 495.44$. Running the `summary` command on the resulting model gives

```
> summary(stepped_model)

Call:
glm(formula = chd ~ tobacco + ldl + famhist + age, family = binomial(),
    data = SAheart)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7559  -0.8632  -0.4545   0.9457   2.4904

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    -4.204275   0.498315  -8.437  < 2e-16 ***
tobacco         0.080701   0.025514   3.163  0.00156 **
ldl             0.167584   0.054189   3.093  0.00198 **
famhistPresent  0.924117   0.223178   4.141 3.46e-05 ***
age             0.044042   0.009743   4.521 6.17e-06 ***
```

These results match well with the ones given in the book.

Notes on Optimal Separating Hyperplanes

For the Lagrange (primal) function to be minimized with respect to β and β_0 given by

$$L_P = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{N} \alpha_i [y_i(x_i^T \beta + \beta_0) - 1] \,, \tag{82}$$

we have derivatives (set equal to zero) given by

$$\frac{\partial L_p}{\partial \beta} = \beta - \sum_{i=1}^{N} \alpha_i y_i x_i = 0 \tag{83}$$

$$\frac{\partial L_p}{\partial \beta_0} = 0 - \sum_{i=1}^{N} \alpha_i y_i = 0 \,. \tag{84}$$

Note that by expanding we can write L_p as

$$L_p = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{N} \alpha_i y_i x_i^T \beta - \beta_0 \sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i \,.$$

From Equation 84 the third term in the above expression for L_p is zero. Using Equation 83 to solve for β we find that the first term in the above expression for L_p is

$$||\beta||^2 = \beta^T \beta = \left(\sum_{i=1}^{N} \alpha_i y_i x_i^T \right) \left(\sum_{j=1}^{N} \alpha_j y_j x_j \right) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \,,$$

and that the second terms in the expression for L_p is

$$\sum_{i=1}^{N} \alpha_i y_i x_i^T \beta = \sum_{i=1}^{N} \alpha_i y_i x_i^T \left(\sum_{j=1}^{N} \alpha_j y_j x_j \right) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \,.$$

Thus with these two substitutions the expression for L_p becomes (we now call this L_D)

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \,. \tag{85}$$

Exercise Solutions

Ex. 4.1 (a constrained maximization problem)

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

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

Here λ is the Lagrange multiplier. 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

$$Ba + \lambda Wa = 0\,,$$

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

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

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. From the form of the objective function we seek to maximize we would select a to be the eigenvector corresponding to the maximum eigenvalue.

Ex. 4.2 (two-class classification)

Part (a): Under zero-one classification loss, for each class ω_k the Bayes' discriminant functions $\delta_k(x)$ take the following form

$$\delta_k(x) = \ln(p(x|\omega_k)) + \ln(\pi_k)\,. \tag{86}$$

If our conditional density $p(x|\omega_k)$ is given by a multidimensional normal then its function form is given by

$$p(x|\omega_k) = \mathcal{N}(x;\mu_k,\Sigma_k) \equiv \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right\}\,. \tag{87}$$

Taking the logarithm of this expression as required by Equation 86 we find

$$\ln(p(x|\omega_k)) = -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - \frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma_k|)\,,$$

so that our discriminant function in the case when $p(x|\omega_k)$ is a multidimensional Gaussian is given by

$$\delta_k(x) = -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - \frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma_k|) + \ln(\pi_k)\,. \tag{88}$$

We will now consider some specializations of this expression for various possible values of Σ_k and how these assumptions modify the expressions for $\delta_k(x)$. Since linear discriminant analysis (LDA) corresponds to the case of equal covariance matrices our decision boundaries (given by Equation 88), but with equal covariances ($\Sigma_k = \Sigma$). For decision purposes we can drop the two terms $-\frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma|)$ and use a discriminant $\delta_k(x)$ given by

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

Expanding the quadratic in the above expression we get

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

Since $x^T\Sigma^{-1}x$ is a common term with the same value in all discriminant functions we can drop it and just consider the discriminant given by

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

Since $x^T\Sigma^{-1}\mu_k$ is a scalar, its value is equal to the value of its transpose so

$$x^T\Sigma^{-1}\mu_k = \left(x^T\Sigma^{-1}\mu_k\right)^T = \mu_k^T(\Sigma^{-1})^T x = \mu_k^T\Sigma^{-1}x\,,$$

since Σ^{-1} is symmetric. Thus the two linear terms in the above combine and we are left with

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \ln(\pi_k)\,. \tag{89}$$

Next we can estimate π_k from data using $\pi_i = \frac{N_i}{N}$ for $i = 1, 2$ and we pick class 2 as the classification outcome if $\delta_2(x) > \delta_1(x)$ (and class 1 otherwise). This inequality can be written as

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

or moving all the x terms to one side

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

as we were to show.

Part (b): To minimize the expression $\sum_{i=1}^N (y_i - \beta_0 - \beta^T x_i)^2$ over $(\beta_0, \beta)'$ we know that the solution $(\hat{\beta}_0, \hat{\beta})'$ must satisfy the normal equations which in this case is given by

$$X^T X \begin{bmatrix} \beta_0 \\ \beta \end{bmatrix} = X^T\mathbf{y}\,.$$

Our normal equations have a block matrix $X^T X$ on the left-hand-side given by

$$\begin{bmatrix} 1 & 1 & \cdots & 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_{N_1} & x_{N_1+1} & x_{N_1+2} & \cdots & x_{N_1+N_2} \end{bmatrix} \begin{bmatrix} 1 & x_1^T \\ 1 & x_2^T \\ \vdots & \\ 1 & x_{N_1}^T \\ 1 & x_{N_1+1}^T \\ 1 & x_{N_1+2}^T \\ \vdots & \\ 1 & x_{N_1+N_2}^T \end{bmatrix}.$$

When we take the product of these two matrices we find

$$\begin{bmatrix} N & \sum_{i=1}^N x_i^T \\ \sum_{i=1}^N x_i & \sum_{i=1}^N x_i x_i^T \end{bmatrix}. \tag{90}$$

For the case where we code our response as $-\frac{N}{N_1}$ for the first class and $+\frac{N}{N_2}$ for the second class (where $N = N_1 + N_2$), the right-hand-side or $X^T y$ of the normal equations becomes

$$\begin{bmatrix} 1 & 1 & \cdots & 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_{N_1} & x_{N_1+1} & x_{N_1+2} & \cdots & x_{N_1+N_2} \end{bmatrix} \begin{bmatrix} -N/N_1 \\ -N/N_1 \\ \vdots \\ -N/N_1 \\ N/N_2 \\ N/N_2 \\ \vdots \\ N/N_2 \end{bmatrix}.$$

When we take the product of these two matrices we get

$$\begin{bmatrix} N_1\left(-\frac{N}{N_1}\right) + N_2\left(\frac{N}{N_2}\right) \\ \left(\sum_{i=1}^{N_1} x_i\right)\left(-\frac{N}{N_1}\right) + \left(\sum_{i=N_1+1}^{N} x_i\right)\left(\frac{N}{N_2}\right) \end{bmatrix} = \begin{bmatrix} 0 \\ -N\mu_1 + N\mu_2 \end{bmatrix}.$$

Note that we can simplify the $(1,2)$ and the $(2,1)$ elements in the block coefficient matrix $X^T X$ in Equation 90 by introducing the class specific means (denoted by μ_1 and μ_2) as

$$\sum_{i=1}^{N} x_i = \sum_{i=1}^{N_1} x_i + \sum_{i=N_1+1}^{N} x_i = N_1\mu_1 + N_2\mu_2\,,$$

Also if we pool all of the samples for this two class problem ($K = 2$) together we can estimate the pooled covariance matrix $\hat{\Sigma}$ (see the section in the book on linear discriminant analysis) as

$$\hat{\Sigma} = \frac{1}{N-K}\sum_{k=1}^{K}\sum_{i:g_i=k}(x_i - \mu_k)(x_i - \mu_k)^T\,.$$

When $K = 2$ this is

$$\begin{aligned} \hat{\Sigma} &= \frac{1}{N-2}\left[\sum_{i:g_i=1}(x_i-\mu_1)(x_i-\mu_1)^T + \sum_{i:g_i=2}(x_i-\mu_2)(x_i-\mu_2)^T\right] \\ &= \frac{1}{N-2}\left[\sum_{i:g_i=1} x_i x_i^T - N_1\mu_1\mu_1^T + \sum_{i:g_i=1} x_i x_i^T - N_2\mu_2\mu_2^T\right]. \end{aligned}$$

From which we see that the sum $\sum_{i=1}^{N} x_i x_i^T$ found in the $(2,2)$ element in the matrix from Equation 90 can be written as

$$\sum_{i=1}^{N} x_i x_i^T = (N-2)\hat{\Sigma} + N_1\mu_1\mu_1^T + N_2\mu_2\mu_2^T\,.$$

Now that we have evaluated both sides of the normal equations we can write them down again as a linear system. We get

$$\begin{bmatrix} N & N_1\mu_1^T + N_2\mu_2^T \\ N_1\mu_1 + N_2\mu_2 & (N-2)\hat{\Sigma} + N_1\mu_1\mu_1^T + N_2\mu_2\mu_2^T \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta \end{bmatrix} = \begin{bmatrix} 0 \\ -N\mu_1 + N\mu_2 \end{bmatrix}. \quad (91)$$

In more detail we can write out the first equation in the above system as

$$N\beta_0 + (N_1\mu_1^T + N_2\mu_2^T)\beta = 0\,,$$

or solving for β_0, in terms of β, we get

$$\beta_0 = \left(-\frac{N_1}{N}\mu_1^T - \frac{N_2}{N}\mu_2^T\right)\beta\,. \tag{92}$$

When we put this value of β_0 into the second equation in Equation 91 we find the total equation for β then looks like

$$(N_1\mu_1 + N_2\mu_2)\left(-\frac{N_1}{N}\mu_1^T - \frac{N_2}{N}\mu_2^T\right)\beta + \left((N-2)\hat{\Sigma} + N_1\mu_1\mu_1^T + N_2\mu_2\mu_2^T\right)\beta = N(\mu_2 - \mu_1)\,.$$

Consider the terms that are outer products of the vectors μ_i (namely terms like $\mu_i\mu_j^T$) we see that taken together they look like

$$\begin{aligned}
\text{Outer Product Terms} &= -\frac{N_1^2}{N}\mu_1\mu_1^T - \frac{2N_1N_2}{N}\mu_1\mu_2^T - \frac{N_2^2}{N}\mu_2\mu_2^T + N_1\mu_1\mu_2^T + N_2\mu_2\mu_2^T \\
&= \left(-\frac{N_1^2}{N} + N_1\right)\mu_1\mu_1^T - \frac{2N_1N_2}{N}\mu_1\mu_2^T + \left(-\frac{N_2^2}{N} + N_2\right)\mu_2\mu_2^T \\
&= \frac{N_1}{N}\left(-N_1 + N\right)\mu_1\mu_1^T - \frac{2N_1N_2}{N}\mu_1\mu_2^T + \frac{N_2}{N}\left(-N_2 + N\right)\mu_2\mu_2^T \\
&= \frac{N_1N_2}{N}\mu_1\mu_1^T - \frac{2N_1N_2}{N}\mu_1\mu_2^T + \frac{N_2N_1}{N}\mu_2\mu_2^T \\
&= \frac{N_1N_2}{N}(\mu_1\mu_1^T - 2\mu_1\mu_2 - \mu_2\mu_2) = \frac{N_1N_2}{N}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T\,.
\end{aligned}$$

Here we have used the fact that $N_1 + N_2 = N$. If we introduce the matrix $\hat{\Sigma}_B$ as

$$\hat{\Sigma}_B \equiv (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T\,, \tag{93}$$

we get that the equation for β looks like

$$\left[(N-2)\hat{\Sigma} + \frac{N_1N_2}{N}\hat{\Sigma}_B\right]\beta = N(\mu_2 - \mu_1)\,, \tag{94}$$

as we were to show.

Part (c): Note that $\hat{\Sigma}_B\beta$ is $(\mu_2 - \mu_1)(\mu_2 - \mu_1)^T\beta$, and the product $(\mu_2 - \mu_1)^T\beta$ is a scalar. Therefore the vector *direction* of $\hat{\Sigma}_B\beta$ is given by $\mu_2 - \mu_1$. Thus in Equation 94 as both the right-hand-side and the term $\frac{N_1N_2}{N}\hat{\Sigma}_B$ are in the direction of $\mu_2 - \mu_1$ the solution β must be in the direction (i.e. proportional to) $\hat{\Sigma}^{-1}(\mu_2 - \mu_1)$.

Ex. 4.4 (multidimensional logistic regression)

In the case of $K > 2$ classes, in the same way as discussed in the section on fitting a logistic regression 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 response vector variable $\mathbf{y}$ of size $K-1$ where the l-th component of $\mathbf{y}$ is equal to one if this sample is drawn from the l-th class and zero otherwise. That is

$$y_i = \begin{cases} 1 & \mathbf{x} \text{ is from class } l \text{ and } i = l \\ 0 & \text{otherwise} \end{cases} .$$

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

$$\begin{aligned} p_{\mathbf{y}}(\mathbf{x}) &= \Pr(G=1|X=\mathbf{x})^{y_1} \Pr(G=2|X=\mathbf{x})^{y_2} \cdots \Pr(G=K-1|X=\mathbf{x})^{y_{K-1}} \\ &\times (1 - \Pr(G=1|X=\mathbf{x}) - \Pr(G=2|X=\mathbf{x}) - \cdots - \Pr(G=K-1|X=\mathbf{x}))^{1-\sum_{l=1}^{K-1} y_l} . \quad (95) \end{aligned}$$

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

$$l = \sum_{i=1}^{N} \log(p_{\mathbf{y}_i}(\mathbf{x}_i)) .$$

Using the Equation 95 in the above expression we find $\log(p_{\mathbf{y}}(\mathbf{x}))$ for any given training pair $(\mathbf{x}_i, \mathbf{y}_i)$ is given by

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

The total log-likelihood is then given by summing the above expression over all data points

$$l(\theta) = \sum_{i=1}^{N} \left[\sum_{l=1}^{K-1} y_{il} \beta_l^T x_i + \log(\Pr(G=k|X=x_i)) \right] .$$

- Here x_i is the ith vector sample $1 \leq i \leq N$ with a leading one prepended and so is of length $p+1$.
- y_{il} is the lth component of the ith response vector i.e. if the sample x_i came from class l when $1 \leq l \leq K-1$ the lth element of y_i is one and all the other elements are zero. If x_i is a sample from class K then all elements of the vector y_i are zero.
- β_l is a vector of coefficients for the lth class $1 \leq l \leq K-1$ with the leading β_{0l} prepended and thus is of length $p+1$.
- $\Pr(G=k|X=x_i)$ is the a posteriori probability that x_i comes from class $G=K$ and is given in terms of the parameters $\{\beta_l\}_{l=1}^{K-1}$ as

$$\Pr(G=k|X=x_i) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_l^T x_i)} .$$

- The total parameter set that we need to solve for of θ can be thought of as the "stacked" vector of β's or

$$\theta \equiv \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{K-2} \\ \beta_{K-1} \end{bmatrix},$$

this is a vector of size $(K-1)(p+1)$. Since each sub vector β_l has $p+1$ components.

Using the expression for $\Pr(G=k|X=x_i)$ the expression for $l(\theta)$ can be written as

$$l(\theta) = \sum_{i=1}^{N} \left[\sum_{l=1}^{K-1} y_{il} \beta_l^T x_i - \log\left(1 + \sum_{l=1}^{K-1} \exp(\beta_l^T x_i)\right) \right].$$

Once we have the objective function $l(\theta)$ defined we can develop an algorithm to maximize it $l(\theta)$ as a function of θ. To develop a procedure for this maximization we will use the Newton-Raphson algorithm in terms of θ (which is a block column vector in β) as

$$\theta^{\text{new}} = \theta^{\text{old}} - \left(\frac{\partial^2 l(\theta)}{\partial \theta \partial \theta^T} \right)^{-1} \frac{\partial l(\theta)}{\partial \theta}. \tag{96}$$

We need to evaluate the derivatives in the Newton-Raphson method. We will do this in block form (which is the same way that θ is constructed). The expression $\frac{\partial l(\theta)}{\partial \theta}$ is a block vector with blocks given by the derivatives of $l(\theta)$ with respect to β_l or

$$\begin{aligned} \frac{\partial l(\theta)}{\partial \beta_l} &= \sum_{i=1}^{N} y_{il} \mathbf{x}_i - \frac{\exp(\beta_l^T \mathbf{x}_i)}{1 + \sum_{l'=1}^{K-1} \exp(\beta_{l'}^T \mathbf{x}_i)} \mathbf{x}_i \\ &= \sum_{i=1}^{N} (y_{il} - \Pr(G=l|X=\mathbf{x}_i)) \mathbf{x}_i. \end{aligned}$$

The argument of the summation above are each column vectors of dimension $p+1$ (since the vectors $\mathbf{x}_i$ are) and we to create the full vector $\frac{\partial l(\theta)}{\partial \theta}$ we would stack the $K-1$ vectors above (one for each of l in $1 \le l \le K-1$) on top of each other. That is we form the full gradient vector $\frac{\partial l(\theta)}{\partial \theta}$ as

$$\frac{\partial l(\theta)}{\partial \theta} = \begin{bmatrix} \frac{\partial l}{\partial \beta_1} \\ \frac{\partial l}{\partial \beta_2} \\ \vdots \\ \frac{\partial l}{\partial \beta_{K-1}} \end{bmatrix}.$$

If we write the above β_l derivative as two terms as

$$\frac{\partial l(\theta)}{\partial \beta_l} = \sum_{i=1}^{N} y_{il} \mathbf{x}_i - \sum_{i=1}^{N} \Pr(G=l|X=\mathbf{x}_i) \mathbf{x}_i, \tag{97}$$

and introduce the $N \times 1$ column vectors $\mathbf{t}_l$ and $\mathbf{p}_l$ for $1 \leq l \leq K-1$ as

$$\mathbf{t}_l = \begin{bmatrix} y_{1,l} \\ y_{2,l} \\ \vdots \\ y_{N,l} \end{bmatrix} \quad \text{and} \quad \mathbf{p}_l = \begin{bmatrix} \Pr(G = l|X = \mathbf{x}_1) \\ \Pr(G = l|X = \mathbf{x}_2) \\ \vdots \\ \Pr(G = l|X = \mathbf{x}_N) \end{bmatrix} .$$

With these definitions we can then write $\frac{\partial l(\theta)}{\partial \beta_l}$ as

$$\mathbf{X}^T \mathbf{t}_l - \mathbf{X}^T \mathbf{p}_l = \mathbf{X}^T (\mathbf{t}_l - \mathbf{p}_l) .$$

The above can be verified by writing out components of the above products. When we stack these vectors to form the full derivative we find

$$\frac{\partial l(\theta)}{\partial \theta} = \begin{bmatrix} \mathbf{X}^T(\mathbf{t}_1 - \mathbf{p}_1) \\ \mathbf{X}^T(\mathbf{t}_2 - \mathbf{p}_2) \\ \vdots \\ \mathbf{X}^T(\mathbf{t}_{K-1} - \mathbf{p}_{K-1}) \end{bmatrix} = \begin{bmatrix} \mathbf{X}^T & 0 & \cdots & 0 \\ 0 & \mathbf{X}^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{X}^T \end{bmatrix} \begin{bmatrix} \mathbf{t}_1 - \mathbf{p}_1 \\ \mathbf{t}_2 - \mathbf{p}_2 \\ \vdots \\ \mathbf{t}_{K-1} - \mathbf{p}_{K-1} \end{bmatrix} . \tag{98}$$

We see the appearance of a $(K-1) \times (K-1)$ block diagonal matrix with blocks given by the $(p+1) \times N$ matrix $\mathbf{X}^T$. Lets denote this matrix by $\hat{\mathbf{X}}^T$ or

$$\hat{\mathbf{X}}^T \equiv \begin{bmatrix} \mathbf{X}^T & 0 & \cdots & 0 \\ 0 & \mathbf{X}^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{X}^T \end{bmatrix} ,$$

which we see is a $(K-1)(p+1) \times (K-1)N$ dimensioned matrix.

Next we have to evaluate the second derivative of $l(\theta)$. As we did when we evaluated the first derivative we will evaluate this expression in block form. From the expression for $\frac{\partial l(\theta)}{\partial \beta_l}$ given in Equation 97 we have that

$$\frac{\partial^2 l(\theta)}{\partial \beta_l \partial \beta_{l'}^T} = -\sum_{i=1}^{N} \frac{\partial \Pr(G = l|X = \mathbf{x}_i)}{\partial \beta_{l'}^T} \mathbf{x}_i .$$

The above derivative on the right-hand-side depends on whether $l = l'$ or not. The case where $l \neq l'$ is slightly easier to compute and the derivative of $\Pr(G = l|X = \mathbf{x}_i)$ with respect to $\beta_{l'}^T$ in that case is

$$\frac{\partial \Pr(G = l|X = \mathbf{x}_i)}{\partial \beta_{l'}^T} = \Pr(G = l|X = \mathbf{x}_i) \Pr(G = l'|X = \mathbf{x}_i) \mathbf{x}_i^T .$$

From this we have that the block *off-diagonal* second derivative terms are given by

$$\frac{\partial^2 l(\theta)}{\partial \beta_l \partial \beta_{l'}^T} = -\sum_{i=1}^{N} \Pr(G = l|X = \mathbf{x}_i) \Pr(G = l'|X = \mathbf{x}_i) \mathbf{x}_i^T \mathbf{x}_i \quad \text{for} \quad l \neq l' . \tag{99}$$

Note that the right-hand-side of this expression evaluates to a $(p+1)\times(p+1)$ matrix (as it should). If $l = l'$ we find the derivative of $\Pr(G = l|X = \mathbf{x}_i)$ with respect to $\beta_{l'}^T$ given by

$$\begin{aligned}\frac{\partial \Pr(G = l|X = \mathbf{x}_i)}{\partial \beta_{l'}^T} &= \frac{\partial}{\partial \beta_{l'}^T}\left(\frac{e^{\beta_{l'}\mathbf{x}_i}}{1+\sum_{l''=1}^{K-1} e^{\beta_{l'}\mathbf{x}_i}}\right) \\ &= \frac{e^{\beta_{l'}\mathbf{x}_i}}{1+\sum_{l''=1}^{K-1} e^{\beta_{l'}\mathbf{x}_i}}\mathbf{x}_i - \frac{e^{\beta_{l'}\mathbf{x}_i}}{(1+\sum_{l''=1}^{K-1} e^{\beta_{l'}\mathbf{x}_i})^2}e^{\beta_{l'}\mathbf{x}_i}\mathbf{x}_i \\ &= \Pr(G = l'|X = \mathbf{x}_i)\mathbf{x}_i - \Pr(G = l'|X = \mathbf{x}_i)^2\mathbf{x}_i \\ &= \Pr(G = l'|X = \mathbf{x}_i)(1 - \Pr(G = l'|X = \mathbf{x}_i))\mathbf{x}_i\,.\end{aligned}$$

From this we have that the block *diagonal* second derivative terms are given by

$$\frac{\partial^2 l(\theta)}{\partial \beta_l \partial \beta_l^T} = -\sum_{i=1}^{N} \Pr(G = l|X = \mathbf{x}_i)(1 - \Pr(G = l|X = \mathbf{x}_i))\mathbf{x}_i^T\mathbf{x}_i\,. \tag{100}$$

The right-hand-side of the above again evaluates to a $(p+1)\times(p+1)$ matrix. To compute the full Hessian we will assemble the block pieces (computed above) and form the full matrix as

$$\frac{\partial^2 l(\theta)}{\partial\theta\partial\theta^T} = \begin{bmatrix} \frac{\partial^2 l}{\partial\beta_1\partial\beta_1^T} & \frac{\partial^2 l}{\partial\beta_1\partial\beta_2^T} & \cdots & \frac{\partial^2 l}{\partial\beta_1\partial\beta_{K-1}^T} \\ \frac{\partial^2 l}{\partial\beta_2\partial\beta_1^T} & \frac{\partial^2 l}{\partial\beta_2\partial\beta_2^T} & \cdots & \frac{\partial^2 l}{\partial\beta_2\partial\beta_{K-1}^T} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 l}{\partial\beta_{K-1}\partial\beta_1^T} & \frac{\partial^2 l}{\partial\beta_{K-1}\partial\beta_2^T} & \cdots & \frac{\partial^2 l}{\partial\beta_{K-1}\partial\beta_{K-1}^T} \end{bmatrix}.$$

As we did in the case of the first derivative of $l(\theta)$ we will write the second derivative of $l(\theta)$ in matrix notation. To do this, we first introduce $K-1$, $N\times N$ diagonal matrices $\mathbf{Q}_l$ for $1 \le l \le K-1$ with diagonal elements given by $\Pr(G = l|X = \mathbf{x}_i)(1 - \Pr(G = l|X = \mathbf{x}_i))$ where $1 \le i \le N$. Then with these definitions we can write Equation 100 in matrix form by

$$\frac{\partial^2 l(\theta)}{\partial \beta_l \partial \beta_l^T} = -\mathbf{X}^T\mathbf{Q}_l\mathbf{X}\,.$$

We next introduce $K-1$, $N\times N$ diagonal matrices $\mathbf{R}_l$ for $1 \le l \le K-1$ with diagonal elements given by $\Pr(G = l|X = \mathbf{x}_i)$ where $1 \le i \le N$. Then with these definitions we can write Equation 99 in matrix form by

$$\frac{\partial^2 l(\theta)}{\partial \beta_l \partial \beta_l^T} = -\mathbf{X}^T\mathbf{R}_l\mathbf{R}_{l'}\mathbf{X}\,.$$

With these definitions we get that the Hessian can be written as

$$\frac{\partial^2 l(\theta)}{\partial\theta\partial\theta^T} = \begin{bmatrix} -\mathbf{X}^T\mathbf{Q}_1\mathbf{X} & -\mathbf{X}^T\mathbf{R}_1\mathbf{R}_2\mathbf{X} & \cdots & -\mathbf{X}^T\mathbf{R}_1\mathbf{R}_{K-1}\mathbf{X} \\ -\mathbf{X}^T\mathbf{R}_2\mathbf{R}_1\mathbf{X} & -\mathbf{X}^T\mathbf{Q}_2\mathbf{X} & \cdots & -\mathbf{X}^T\mathbf{R}_2\mathbf{R}_{K-1}\mathbf{X} \\ \vdots & & \ddots & \vdots \\ -\mathbf{X}^T\mathbf{R}_{K-1}\mathbf{R}_1\mathbf{X} & -\mathbf{X}^T\mathbf{R}_{K-1}\mathbf{R}_2\mathbf{X} & \cdots & -\mathbf{X}^T\mathbf{Q}_{K-1}\mathbf{X} \end{bmatrix}$$

This is a block $(K-1)\times(K-1)$ matrix with each block matrix of size $(p+1)\times(p+1)$ giving a total matrix size of $(K-1)(p+1)\times(K-1)(p+1)$. We can introduce the matrix

$\hat{\mathbf{X}}^T$ by writing the above Hessian as

$$\frac{\partial^2 l(\theta)}{\partial\theta\partial\theta^T} = -\begin{bmatrix} \mathbf{X}^T & 0 & \cdots & 0 \\ 0 & \mathbf{X}^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{X}^T \end{bmatrix} \begin{bmatrix} \mathbf{Q}_1 & \mathbf{R}_1\mathbf{R}_2 & \cdots & \mathbf{R}_1\mathbf{R}_{K-1} \\ \mathbf{R}_2\mathbf{R}_1 & \mathbf{Q}_2 & \cdots & \mathbf{R}_2\mathbf{R}_{K-1} \\ \vdots & & \ddots & \vdots \\ \mathbf{R}_{K-1}\mathbf{R}_1 & \mathbf{R}_{K-1}\mathbf{R}_2 & \cdots & \mathbf{Q}_{K-1} \end{bmatrix} \begin{bmatrix} \mathbf{X} & 0 & \cdots & 0 \\ 0 & \mathbf{X} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{X} \end{bmatrix}.$$

Define the *non-diagonal* matrix $\mathbf{W}$ as

$$\mathbf{W} \equiv \begin{bmatrix} \mathbf{Q}_1 & \mathbf{R}_1\mathbf{R}_2 & \cdots & \mathbf{R}_1\mathbf{R}_{K-1} \\ \mathbf{R}_2\mathbf{R}_1 & \mathbf{Q}_2 & \cdots & \mathbf{R}_2\mathbf{R}_{K-1} \\ \vdots & & \ddots & \vdots \\ \mathbf{R}_{K-1}\mathbf{R}_1 & \mathbf{R}_{K-1}\mathbf{R}_2 & \cdots & \mathbf{Q}_{K-1} \end{bmatrix},$$

and we have

$$\frac{\partial^2 l(\theta)}{\partial\theta\partial\theta^T} = -\hat{\mathbf{X}}^T\mathbf{W}\hat{\mathbf{X}}. \tag{101}$$

Using Equation 98 and 101 we can write the Newton-Raphson method 96 for θ in matrix form in the same way as was done in the text for the two-class case. Namely we find

$$\begin{aligned} \theta^{\text{new}} &= \theta^{\text{old}} + \left(\hat{\mathbf{X}}^T\mathbf{W}\hat{\mathbf{X}}\right)^{-1}\hat{\mathbf{X}}^T \begin{bmatrix} \mathbf{t}_1 - \mathbf{p}_1 \\ \mathbf{t}_2 - \mathbf{p}_2 \\ \vdots \\ \mathbf{t}_{K-1} - \mathbf{p}_{K-1} \end{bmatrix} \\ &= \left(\hat{\mathbf{X}}^T\mathbf{W}\hat{\mathbf{X}}\right)^{-1}\hat{\mathbf{X}}^T\mathbf{W}\left(\hat{\mathbf{X}}\theta^{\text{old}} + \mathbf{W}^{-1}\begin{bmatrix} \mathbf{t}_1 - \mathbf{p}_1 \\ \mathbf{t}_2 - \mathbf{p}_2 \\ \vdots \\ \mathbf{t}_{K-1} - \mathbf{p}_{K-1} \end{bmatrix}\right). \end{aligned}$$

This shows that θ^{new} is the solution of a *non-diagonal* weighted least squares problem with a response $\mathbf{z}$ given by

$$\mathbf{z} \equiv \hat{\mathbf{X}}\theta^{\text{old}} + \mathbf{W}^{-1}\begin{bmatrix} \mathbf{t}_1 - \mathbf{p}_1 \\ \mathbf{t}_2 - \mathbf{p}_2 \\ \vdots \\ \mathbf{t}_{K-1} - \mathbf{p}_{K-1} \end{bmatrix}.$$

The fact that the weight matrix $\mathbf{W}$ is non-diagonal means that there are not efficient algorithms for solving the weighted least squares algorithm. In this case it is better perhaps to form the matrices above and to numerically adjust θ using the above matrix products starting at some initial value. Alternatively, another multidimensional optimization algorithm could be used (rather than Newton-Raphson) for example the conjugate-gradient or Powell's method to find the optimal θ.

Ex. 4.6 (proving the convergence of the perceptron algorithm)

Part (a): By definition, if the points are separable then there exists a vector β such that

$$\begin{aligned} \beta^T x_i^* > 0 \quad \text{when} \quad y_i = +1 \quad \text{and} \\ \beta^T x_i^* < 0 \quad \text{when} \quad y_i = -1\,, \end{aligned}$$

for all $i = 1, 2, \cdots, N$. This is equivalent to the expression that $y_i \beta^T x_i^* > 0$ for all i. Equivalently we can divide this expression by $||x_i^*||$ (a positive number) to get

$$y_i \beta^T z_i > 0\,,$$

for all i. Since each one of these N values of $y_i \beta^T z_i$ is positive let $m > 0$ be the smallest value of this product observed over all our training set. Thus by definition of m we have

$$y_i \beta^T z_i \geq m\,,$$

for all i. When we divide both sides of this inequality by this positive value of m we get

$$y_i \left(\frac{1}{m} \beta \right)^T z_i \geq 1\,.$$

If we define $\beta_{\text{sep}} \equiv \frac{1}{m}\beta$ we have shown that $y_i \beta_{\text{sep}}^T z_i \geq 1$ for all i.

Part (b): From $\beta_{\text{new}} = \beta_{\text{old}} + y_i z_i$ we have that

$$\beta_{\text{new}} - \beta_{\text{sep}} = \beta_{\text{old}} - \beta_{\text{sep}} + y_i z_i\,.$$

When we square this result we get

$$||\beta_{\text{new}} - \beta_{\text{sep}}||^2 = ||\beta_{\text{old}} - \beta_{\text{sep}}||^2 + y_i^2 ||z_i||^2 + 2 y_i (\beta_{\text{old}} - \beta_{\text{sep}})^T z_i\,.$$

Since $y_i = \pm 1$ and $||z_i||^2 = 1$ we have that $y_i^2 ||z_i||^2 = 1$ for the second term on the right-hand-side. Note that the third term on the right-hand-side is given by

$$2(y_i \beta_{\text{old}}^T z_i - y_i \beta_{\text{sep}}^T z_i)\,.$$

Since the "point" y_i, z_i was misclassified by the vector β_{old} we have $y_i \beta_{\text{old}}^T z_i < 0$ (if it was positive we would have classified it correctly). Since β_{sep} is the vector that can correctly classify all points we have $y_i \beta_{\text{sep}}^T z_i \geq 1$. With these two facts we can write

$$2(y_i \beta_{\text{old}}^T z_i - y_i \beta_{\text{sep}}^T z_i) \leq 2(0 - 1) = -2\,.$$

Thus we have just shown that

$$||\beta_{\text{new}} - \beta_{\text{sep}}||^2 \leq ||\beta_{\text{old}} - \beta_{\text{sep}}||^2 + 1 - 2 = ||\beta_{\text{old}} - \beta_{\text{sep}}||^2 - 1\,.$$

Thus we can drive any initial vector β_{start} to β_{sep} in at most $||\beta_{\text{start}} - \beta_{\text{sep}}||^2$ steps.

Ex. 4.9 (classification of the vowel data set)

See the notes on the text on Page 61 where a comparison of various classification methods on the vowel data set was done.

Chapter 5 (Basis Expansions and Regularization)

Notes on the Text

There is a nice chapter on using B-splines in the R language for regression problems in the book [11]. There are several regression examples using real data and a discussion of the various decisions that need to be made when using splines for regression. The chapter also provides explicit R code (and discussions) demonstrating how to solve a variety of applied regression problems using splines.

Notes on piecewise polynomials and splines

If the basis functions $h_m(X)$ are local piecewise constants i.e. if

$$h_1(X) = I(X < \xi_1)\,, \quad h_2(X) = I(\xi_1 \leq X < \xi_2)\,, \quad h_3(X) = I(\xi_2 \leq X)\,,$$

and taking our approximation to $f(X)$ of

$$f(X) \approx \sum_{m=1}^{3} \beta_m h_m(X)\,,$$

then the coefficients β_m are given by solving the minimization problem

$$\begin{aligned}
\beta &= \text{ArgMin}_\beta \left\| f(X) - \sum_{m=1}^{3} \beta_m h_m(X) \right\|^2 \\
&= \text{ArgMin}_\beta \int \left(f(x) - \sum_{m=1}^{3} \beta_m h_m(X) \right)^2 dX \\
&= \text{ArgMin}_\beta \left(\int_{\xi_L}^{\xi_1} (f(X) - \beta_1)^2 dX + \int_{\xi_1}^{\xi_2} (f(X) - \beta_2)^2 dX + \int_{\xi_2}^{\xi_R} (f(X) - \beta_3)^2 dX \right) .
\end{aligned}$$

Here ξ_L and ξ_R are the left and right end points of the functions domain. Since the above is a sum of three positive independent (with respect to β_m) terms we can minimize the total sum by minimizing each one independently. For example, we pick β_1 that minimizes the first term by taking the derivative of that term with respect to β_1, and setting the result equal to zero and solving for β_1. We would have

$$\frac{d}{d\beta_1} \int_{\xi_L}^{\xi_1} (f(X) - \beta_1)^2 dX = 2 \int_{\xi_L}^{\xi_1} (f(X) - \beta_1) dX = 0\,.$$

When we solving this for β_1 we get

$$\beta_1 = \frac{1}{\xi_1 - \xi_L} \int_{\xi_L}^{\xi_1} f(X) dX = \overline{Y}_1\,,$$

as claimed by the book. The other minimizations over β_2 and β_3 are done in the same way.

Note on R programming with splines

If one is working in the R language, much of the complexity of coding splines has already been done by using the bs command (for B-splines) or the ns command (for natural splines). For example, if one has data in the vector X the following given various spline approximations that could be used as a first modeling attempt

- bs(x,degree=1,df=1) gives a single linear fit over the entire domain of X
- bs(x,degree=1,df=2) gives two linear fits with a break point at the median of the data in X
- bs(x,degree=1,df=3) gives three linear fits separated at the 1/3 and 2/3 percentile points of the date
- bs(x,degree=2,df=2) gives a single quadratic fit over the entire domain
- bs(x,degree=2,df=3) gives two quadratic fits with a break point at the median of the data X
- bs(x,degree=2,df=4) gives three quadratic fits with a break points at the 1/3 and 2/3 percentile points of the date

The way to reason about the values passed to each of the arguments of the bs functions is that the degree argument is the degree of the polynomial that will be fit in each region and that additional df values (above the minimum necessary) inserts additional polynomials into the domain. For example, to use linear fits requires degree=1 polynomials. Then the value of df must be at least one (for the slope) thus each additional value for df introduces a new region in which a line is fit. To use quadratic polynomials we use degree=2 and then df must be larger than 2. As an example, in the R command dup_fig_5_1.R we perform the suggested fittings above on the function

$$Y = \cos(X) + 0.3n\,.$$

where $n \sim N(0,1)$. When that command is run we get the plots shown in Figure 13.

Notes on the South African Heart Disease Example

In the R code dup_table_5_1.R we have code that duplicates logistic regression using the South African heart disease data. When we run that code we first fit a logistic regression model and then use the R function step1 to experiment with removing various terms. The output from this command gives

```
> drop1( m, scope=form, test="Chisq", data=SAheart )
Single term deletions
```

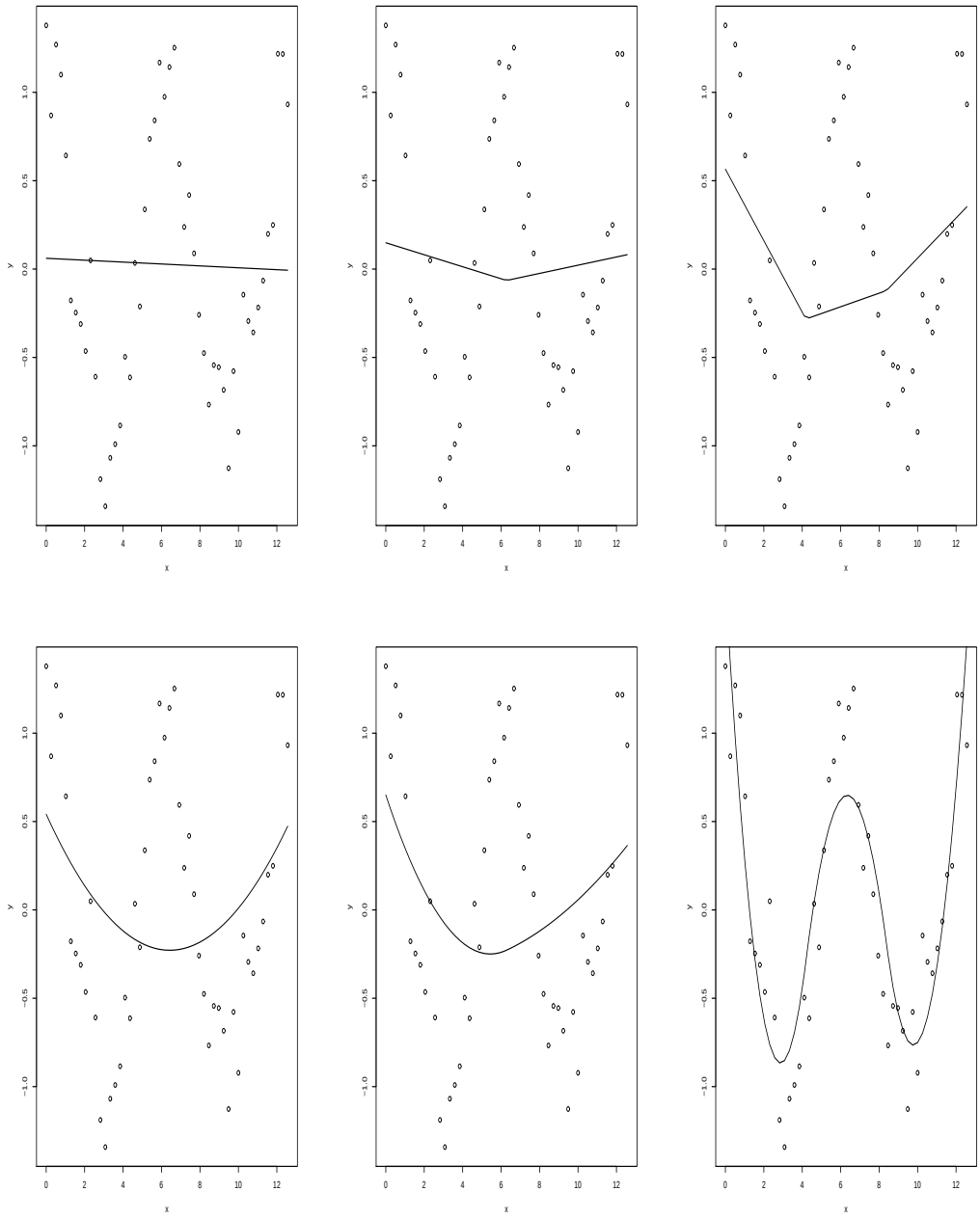

Figure 13: Piecewise regression using B-splines. The first row shows three linear models, while the second row shows three quadratic models.

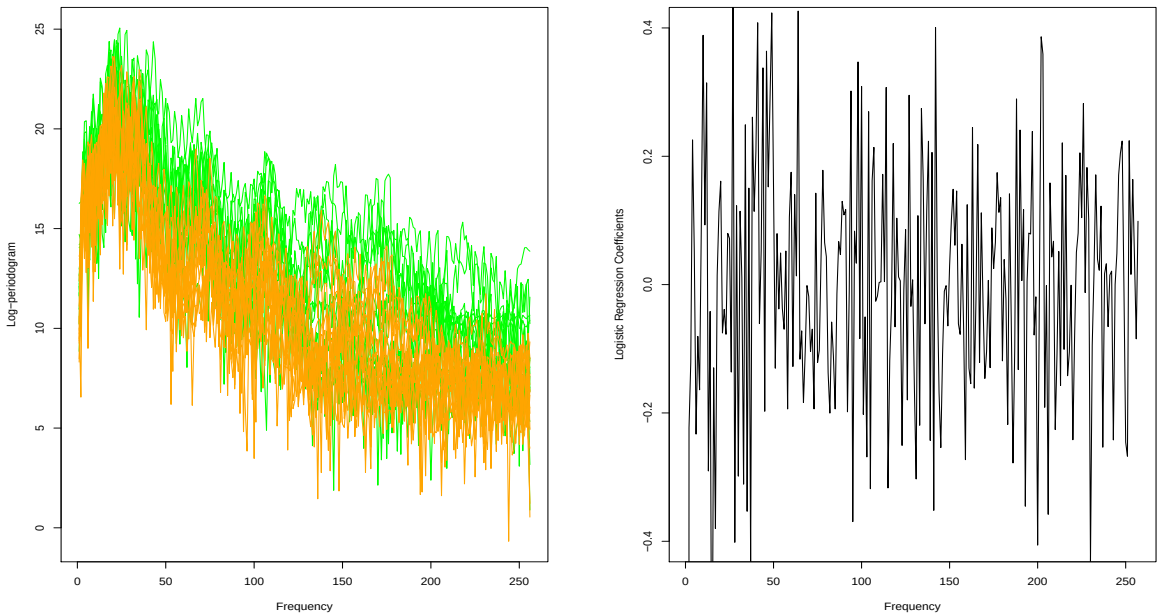

Figure 14: Duplication of plots in Figure 5.5 from the book.

```
Model:
chd ~ ns(sbp, df = 4) + ns(tobacco, df = 4) + ns(ldl, df = 4) +
    famhist + ns(obesity, df = 4) + ns(alcohol, df = 4) + ns(age,
    df = 4)
                    Df Deviance    AIC     LRT   Pr(Chi)
<none>                   457.63 509.63
ns(sbp, df = 4)      4   466.77 510.77  9.1429 0.0576257 .
ns(tobacco, df = 4)  4   469.61 513.61 11.9753 0.0175355 *
ns(ldl, df = 4)      4   470.90 514.90 13.2710 0.0100249 *
famhist              1   478.76 528.76 21.1319 4.287e-06 ***
ns(obesity, df = 4)  4   465.41 509.41  7.7749 0.1001811
ns(alcohol, df = 4)  4   458.09 502.09  0.4562 0.9776262
ns(age, df = 4)      4   480.37 524.37 22.7414 0.0001426 ***
```

These numbers match well with the ones given in the book.

Notes on the phoneme classification example

In the `R` code `dup_fig_5_5.R` we duplicate some of the plots shown in the book. When we run that script we get the two plots shown in Figure 14. The error rate of the "raw" logistic regression classifier (without any smoothing) gives for training and testing

```
[1] "err_rate_train=   0.093114; err_rate_test=    0.24374"
```

These numbers are not that different for the ones given in the book. If we train a quadratic discriminant classifier on these two phonemes we get

```
[1] "err_rate_train=   0.000000; err_rate_test=    0.33941"
```

Finally, just to compare algorithms performance, we can fit (using cross validation) a regularized discriminant analysis model where we find

```
[1] "err_rate_train=   0.075781; err_rate_test=    0.19590"
```

the optimal parameters found were $\alpha = 0.1111111$ and $\gamma = 0.7474747$. While this testing error rate is not as good as the regularized result the book gives 0.158 regularized discriminant analysis gives the best result of the three classifiers tested.

Notes on the bone mineral density (smoothing splines example)

In the `R` code `dup_fig_5_6.R` we duplicate some of the plots shown in the book. When we run that script we get the two plots shown in Figure 15. This plot uses the `R` command `smooth.spline`.

Exercise Solutions

Ex. 5.9 (deriving the Reinsch form)

Given

$$S_\lambda = N(N^T N + \lambda \Omega_N)^{-1} N^T \,,$$

when N is invertible then we have

$$\begin{aligned} S_\lambda &= (N^{-T}(N^T N + \lambda \Omega_N) N^{-1})^{-1} \\ &= (N^{-T} N^T N N^{-1} + \lambda N^{-T} \Omega_N N^{-1})^{-1} \\ &= (I + \lambda N^{-T} \Omega_N N^{-1})^{-1} \,, \end{aligned}$$

so $K \equiv N^{-T} \Omega_N N^{-1}$.

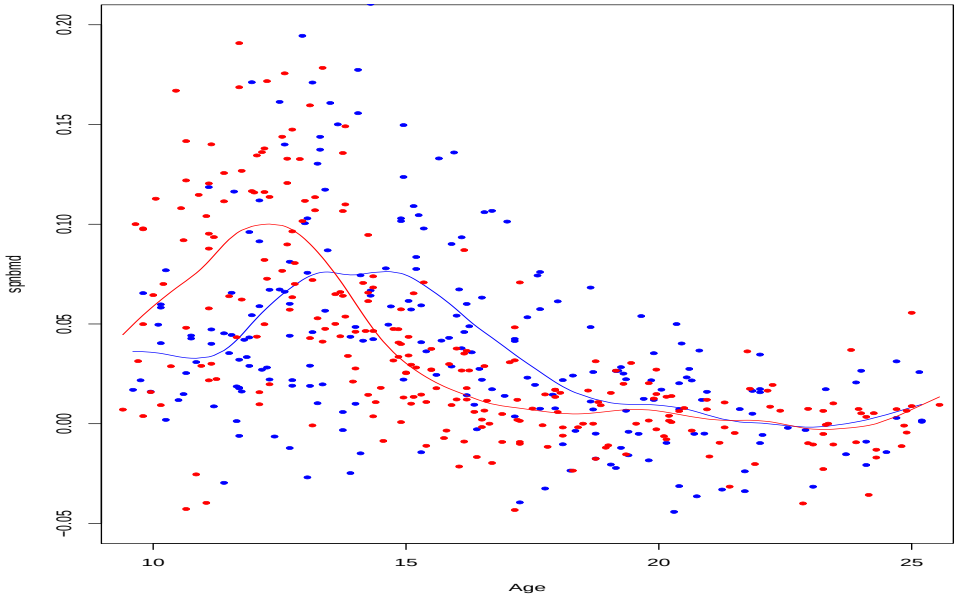

Figure 15: Duplication of plot in Figure 5.6 from the book.

Chapter 7 (Model Assesment and Selection)

Notes on the Text

Note: This chapter needs to be proofed.

Notes on the bias-variance decomposition The *true model* is $Y = f(X) + \epsilon$ with $E(\epsilon) = 0$ and $\text{Var}(\epsilon) = \sigma_\epsilon^2$. Define $\text{Err}(x_0)$ as the test error we are after is generaliztaion error i.e. how well we will do on samples from *outside* the given data set.

$$\begin{aligned}
\text{Err}(x_0) &= E[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= E[(f(x_0) + \epsilon - \hat{f}(x_0))^2 | X = x_0] \\
&= E[(f(x_0) - \hat{f}(x_0))^2 + 2\epsilon(f(x_0) - \hat{f}(x_0)) + \epsilon^2 | X = x_0] \\
&= E[(f(x_0) - \hat{f}(x_0))^2 | X = x_0] \\
&+ 2E[\epsilon(f(x_0) - \hat{f}(x_0)) | X = x_0] \\
&+ E[\epsilon^2 | X = x_0] \,.
\end{aligned}$$

Now

$$E[\epsilon(f(x_0) - \hat{f}(x_0)) | X = x_0] = E[\epsilon | X = x_0] E[(f(x_0) - \hat{f}(x_0)) | X = x_0] = 0 \,.$$

and so the above becomes

$$\text{Err}(x_0) = \sigma_\epsilon^2 + E[(f(x_0) - \hat{f}(x_0))^2 | X = x_0] \,.$$

Now $\hat{f}$ is *also* random since it depends on the random selection of the initial training set. To evalaute this term we can consider the expected value of $\hat{f}(x_0)$ taken over all random training sets. We have

$$E[(f(x_0) - \hat{f}(x_0))^2 | X = x_0] = E[(f(x_0) - E\hat{f}(x_0) + E\hat{f}(x_0) - \hat{f}(x_0))^2 | X = x_0] \,.$$

This later expression expands to

$$E[(f(x_0) - E\hat{f}(x_0))^2 + 2(f(x_0) - E\hat{f}(x_0))(E\hat{f}(x_0) - \hat{f}(x_0)) + (E\hat{f}(x_0) - \hat{f}(x_0))^2] \,.$$

Note $f(x_0)$ and $E\hat{f}(x_0)$ given $X = x_0$ are *not* random since $f(x_0)$ in our universie output and $E\hat{f}(x_0)$ has already taken the expectation. Thus when we take the expectation of the above we get

$$E[E\hat{f}(x_0) - \hat{f}(x_0) | X = x_0] = 0 \,,$$

and the second term in the above vanishes. We end with

$$\begin{aligned}
E[(f(x_0) - \hat{f}(x_0))^2 | X = x_0] &= (f(x_0) - E\hat{f}(x_0))^2 + E[(E\hat{f}(x_0) - \hat{f}(x_0))^2] \\
&= \sigma_\epsilon^2 + \text{model bias}^2 + \text{model variance} \,.
\end{aligned}$$

$Y = f(X) + \epsilon$ $\text{Var}(\epsilon) = \sigma_\epsilon^2$

$$\begin{aligned}
\text{Err}(x_0) &= E[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= E[(f(x) + \epsilon - \hat{f}(x_0))^2 | X = x_0] \\
&= E[(f(x) - \hat{f}(x_0) + \epsilon)^2 | X = x_0] \\
&= E[(f(x_0) - Ef(x_0) + Ef(x_0) - \hat{f}(x_0) + \epsilon)^2] \\
&= E[(f(x_0) - Ef(x_0))^2 + 2(f(x_0) - Ef(x_0))(Ef(x_0) - \hat{f}(x_0)) + (Ef(x_0) - \hat{f}(x_0))^2 + \epsilon^2]
\end{aligned}$$

We take the expectation over possible Y values via ϵ variabliligy and possible training sets that go into constants of $\hat{f}(x_0)$. Note that all $E[(\cdot)\epsilon] = E[(\cdot)]E[\epsilon] = 0$. The middle terms vanishes since $E[f(x_0) - Ef(x_0)] = 0$.

Introduce $E[\hat{f}(x_0)]$ expected value over all training sets predicition using the median

For k nearest neighbor

$$\begin{aligned}
\hat{f}(x_0) &= \frac{1}{k} \sum_{l=1}^{k} f(x_{(l)}) \\
&\approx \frac{1}{k} \sum_{l=1}^{k} (f(x_0) + \varepsilon_{(l)}) = f(x_0) + \frac{1}{k} \sum_{l=1}^{k} \varepsilon_{(l)} \,.
\end{aligned}$$

Now the variance of the estimate $\hat{f}(x_0)$ comes from the $\frac{1}{k} \sum_{l=1}^{k} \varepsilon_{(k)}$ random term. Taking the variance of this summation we have

$$\frac{1}{k^2} \sum_{l=1}^{k} \text{Var}(\varepsilon_{(l)}) = \frac{1}{k^2} k \text{Var}(\varepsilon) \frac{1}{k} \sigma_\varepsilon^2 \,. \tag{102}$$

Which is the equation in the book.

$\hat{f}_p(x) = \hat{\beta}^T x$ we have

$$\begin{aligned}
\text{Err}(x_0) &= E[(Y - \hat{f}_p(x_0))^2 | X = x_0] \\
&= \sigma_\varepsilon^2 + [f(x_0) - E\hat{f}_p(x_0)]^2 + ||h(x_0||^2 \sigma_\varepsilon^2 \,.
\end{aligned}$$

$\hat{f}_p(x_0) = x_0^T (X^T X)^{-1} X^T y$ where y is random since $y = y_0 + \varepsilon$

$$\frac{1}{N} \sum \text{Err}(x_0) = \sigma_\varepsilon^2 + \frac{1}{N} \sum_{i=1}^{N} [f(x_i) - E\hat{f}(x_i)]^2 + \frac{p}{N} \sigma_\varepsilon^2 \,. \tag{103}$$

Chapter 10 (Boosting and Additive Trees)

Notes on the Text

Notes on the Sample Classification Data

For AdaBoost.M1 the books suggests using a "toy" ten-dimensional data set where the individual elements $X_1, X_2, \cdots, X_{10}$ of the vector $\mathbf{X}$ are standard independent Gaussian random draws and the classification labels (taken from $\{-1, +1\}$) are assigned as

$$Y = \begin{cases} +1 & \text{if } \sum_{j=1}^{10} X_j^2 > \chi^2_{10}(0.5) \\ -1 & \text{otherwise} \end{cases} .$$

Here $\chi^2_{10}(0.5)$ is the median of a chi-squared random variable with 10 degrees of freedom. In [2] it is stated that the X_i are standard independent Gaussian and their 10 values are squared and then summed one *gets* a chi-squared random variable (by definition) with 10 degrees of freedom. Thus the threshold chosen of $\chi^2_{10}(0.5)$ since it is the *median* will split the data generated exactly (in the limit of infinite samples) into two equal sets. Thus when we ask for N samples we approximately equal number of $\frac{N}{2}$ of samples in each class and it is a good way to generate testing data. Code to generate data from this toy problem in Matlab is given by the function `gen_data_pt_b.m` and in the `R` language in the function `gen_eq_10_2_data.R`.

Notes on duplicated Figure 10.2

See the `R` code `dup_fig_10_2.R` where we use the `R` package `gbm` do duplicate the books Figure 10.2. That package has an option (`distribution='adaboost'`) that will perform gradient boosting to minimize the exponential adaboost loss function. Since this package does gradient boosting (a more general form of boosting) to get the results from the book one needs to set the learning rate to be 1. When that code is run we get the results given in Figure 16, which matches quite well the similar plot given in the text. For a "home grown" Matlab implementation of AdaBoost see the problem on Page 94.

Notes on Why Exponential Loss

From the text and Equation 105 we have

$$f^*(x) = \text{argmin}_{f(x)} E_{Y|x}(e^{-Yf(x)}) = \frac{1}{2}\log\left(\frac{\Pr(Y=+1|x)}{\Pr(Y=-1|x)}\right)$$

Solving the above for $\Pr(Y=+1|x)$ in terms of $f^*(x)$ we first write the above as

$$\frac{\Pr(Y=+1|x)}{1-\Pr(Y=+1|x)} = e^{2f^*(x)} ,$$

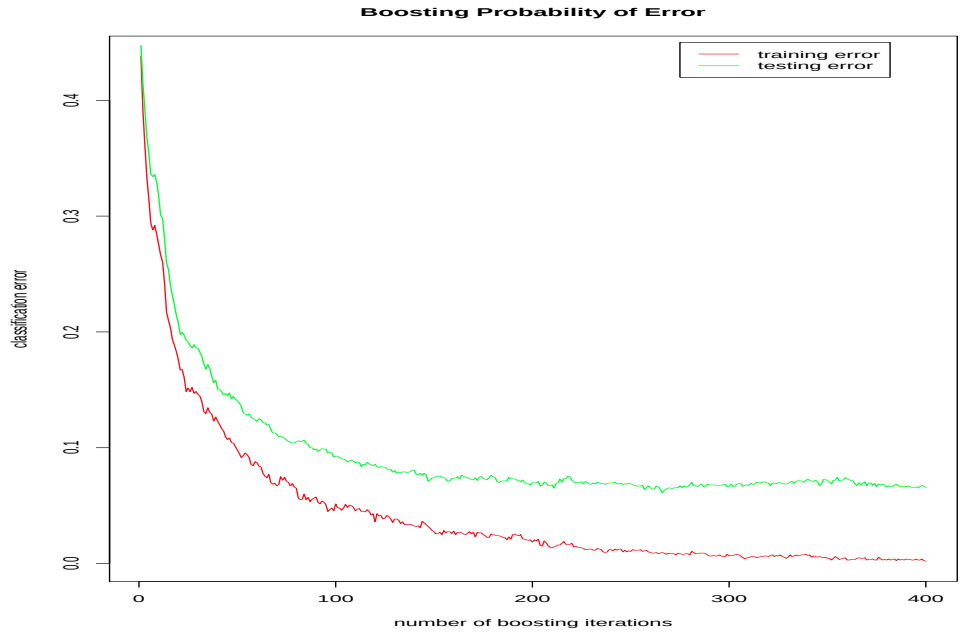

Figure 16: Duplication of plots in Figure 10.2 from the book. See the main text for details.

or

$$(1 + e^{2f^*(x)}) \Pr(Y = +1|x) = e^{2f^*(x)} .$$

Solving for $\Pr(Y = +1|x)$ we get

$$\Pr(Y = +1|x) = \frac{e^{2f^*(x)}}{1 + e^{2f^*(x)}} = \frac{1}{1 + e^{-2f^*(x)}} ,$$

the expression given in the book.

Notes on the Loss Function and Robustness

To compare different loss functions for robustness in the `R` code `dup_fig_10_4.R` we plot the "modified" loss functions discussed in the book. Namely we consider the typical loss functions discussed thus far but scaled so that they pass through the point $(0, 1)$ in (fy, loss) space. Since our targets are mapped to $y \in \{-1, +1\}$ and our predictions are given by $G(x) = \text{sign}(f(x))$ data points are misclassified when the product $f(x)y < 0$ and are correctly classified when $f(x)y > 0$. Following the book, we consider the following losses as functions of the product fy:

- Misclassification: $I(\text{sign}(f) \neq y)$.
- Exponential: e^{-yf}.
- Binomial deviance: $\log(1+e^{-2yf})$ which when scaled to pass through $(0, 1)$ this becomes $\log(1 + e^{-2yf})/\log(2)$.

- Squared error: $(y-f)^2$ when scaled to pass through $(0,1)$ this becomes $\frac{1}{y^2}(1-yf)^2$. Note that we can drop the factor $\frac{1}{y^2}$ since $y \in \{-1,+1\}$.
- Support vector $(1-yf)_+$.

When we run the above `R` script we get the plot shown in Figure 17. Note that all curves but the one for binomial deviance (the orange curve) seem to be at their correct locations. There must be something wrong since the location of the binomial deviance curve in this plot would indicate that binomial deviance is *less* robust to outlying classification examples than is exponential loss (the cyan curve). This follows from the fact that the binomial deviance curve lies above the exponential loss curve when $yf < 0$ indicating greater relative penalty applied when using the binomial deviance as a loss for incorrectly classified samples. This is in contrast to what the book states. There must be an error in this discussion but I'm not sure where it is. I've tried to "scale" the binomial functional form of $\log(1+e^{-2yf})$ by:

- Shifting the function along the fy axis
- Shifting the function along the *loss* axis
- Scaling the function by dividing by $\log(2)$

and none of the resulting plots match the ones given in the book. Note that all of the above transformations pass through the point $(0,1)$. You can view the other plots by uncommenting lines in the `dup_fig_10_4.R` script. If anyone sees a problem with my logic or knows what I am doing incorrectly please contact me. For quadratic loss

$$f^*(x) = \text{argmin}_{f(x)} E_{Y|x}(Y-f(x))^2 = E(Y|x)\,,$$

where the last step is via the Gauss Markov theorem (namely that the minimizer of the squared error is the conditional expectation). Since the mapping for Y takes the form $Y \in \{-1,+1\}$ we can evaluate the above expectation explicitly using the definition of expectation

$$\begin{aligned} E(Y|x) &= +1\Pr(Y=+1|x) - \Pr(Y=-1|x) \\ &= +1\Pr(Y=+1|x) - (1-\Pr(Y=+1|x)) \\ &= 2\Pr(Y=+1|x) - 1\,. \end{aligned} \tag{104}$$

Notes on Duplicating the Spam Data Example

See the `R` code `dup_spam_classification.R` where we use the `R` package `gbm` do duplicate the books Spam data example. Following the documentation for the package `gbm` I selected `shrinkage=0.005` (a value be between 0.01 and 0.001), `interaction.depth=1`, `cv.folds=5`, and `n.trees=50000` boosting trees to get estimate of the learning curves associated with the `gbm` method applied to this data set. After training when one then calls the function `gbm.perf` one gets the plot shown in Figure 18 (left) and the fact that the best number

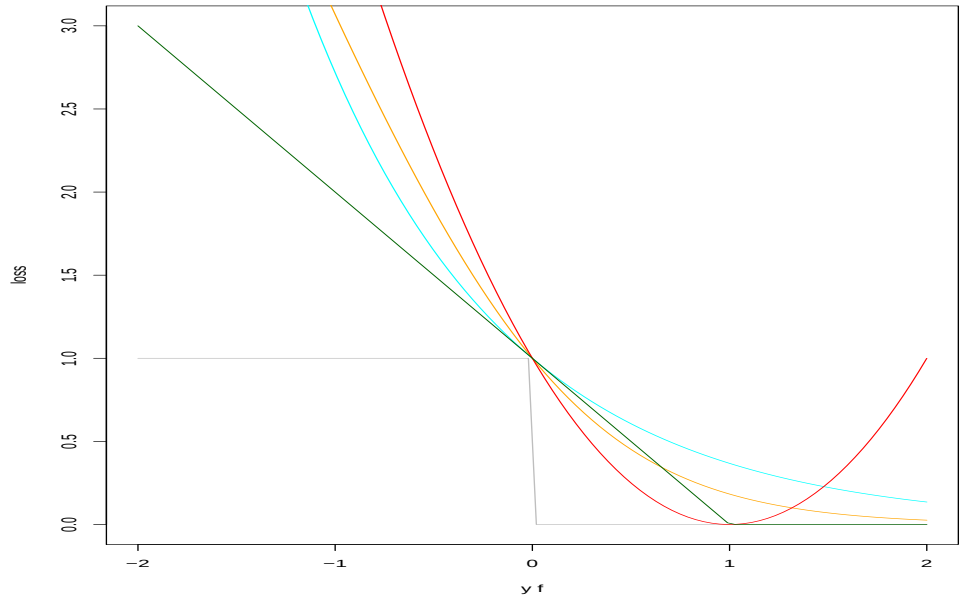

Figure 17: An attempted duplication of plots in Figure 10.4 from the book. See the main text for details.

of boosting trees is 10563. The error rate with this tree is 5.14%. This is a bit larger than the numbers reported in the book but still in the same ballpark. If we retrain with `interaction.depth=3` we get the learning curve given in Figure 18 (right). The optimal number of boosting trees in this case 7417, also given a smaller testing error of 4.75%. The `R` routine `summary` can be use to plot a relative importance graph. When we do that we get the plot shown in Figure 19.

While the labeling of the above plot is not as good in the books we can print the first twenty most important words (in the order from most important to less important) to find

```
> most_important_words[1:20]
 [1] "$"                          "!"
 [3] "remove"                     "free"
 [5] "hp"                         "your"
 [7] "capital_run_length_average" "capital_run_length_longest"
 [9] "george"                     "capital_run_length_total"
[11] "edu"                        "money"
[13] "our"                        "you"
[15] "internet"                   "1999"
[17] "will"                       "email"
[19] "re"                         "receive"
```

This ordering of words that are most important for the classification task agrees very well with the ones presented in the book.

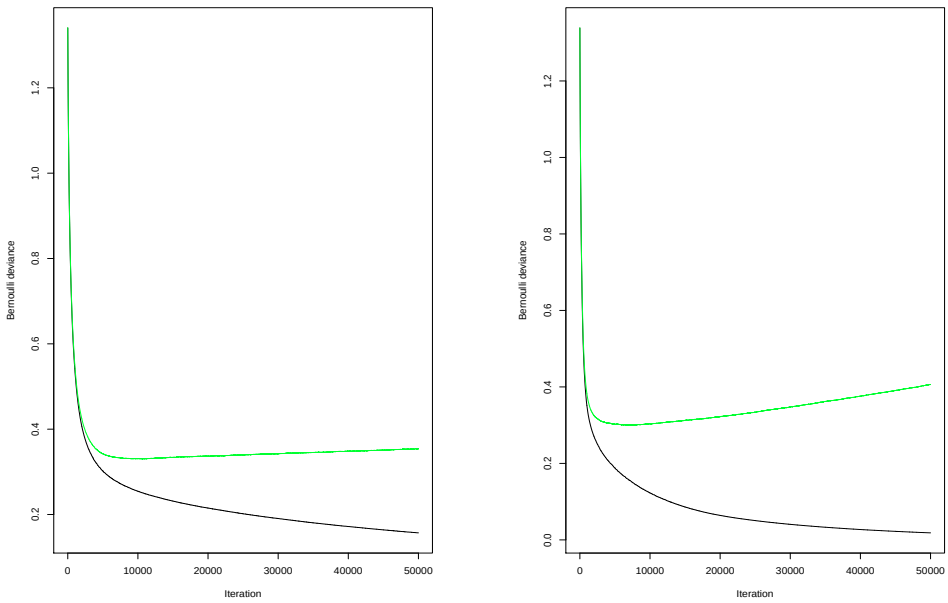

Figure 18: **Left:** The learning curves of the `gbm` method on the spam data when `interaction.depth=1`. Note that after we have performed enough boosting iterations we steadily decrease the loss in sample (the black curve) while the loss out-of-sample stays nearly constant. **Right:** The learning curves of the `gbm` method on the spam data when `interaction.depth=3`. In this case as we perform learning by using more boosts we start to perform worse on the testing data set. This indicates that we are overfitting and learning (memorizing) the training data set. We should stop learning at the point where the error when using the cross validation data set starts to increase i.e. the location of the minimum of the green curve. Note that the y-axis between the two plots are not the same.

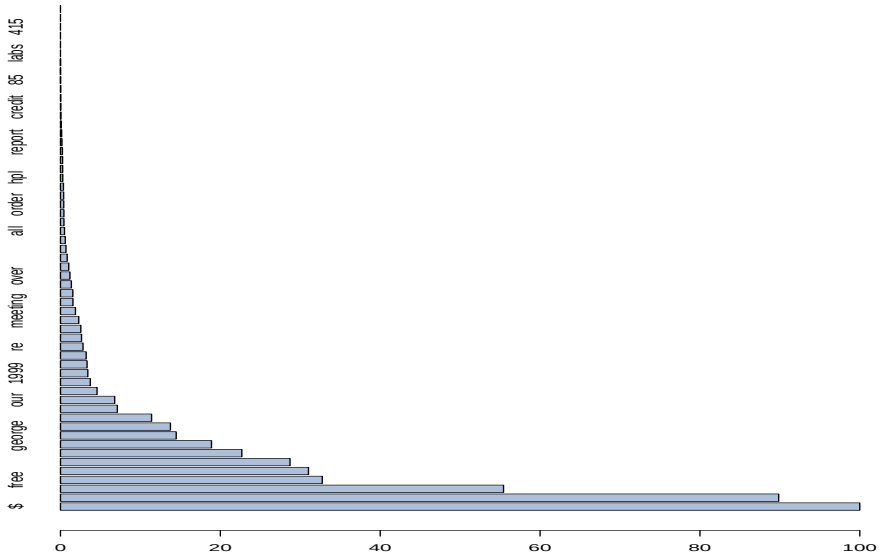

Figure 19: A plot of the relative importance of the various words in the spam classification example using the R code `summary(gbm)`.

Notes on Duplicating the Boosting with Different Sized Trees Example

See the R code `dup_fig_10_9.R` where we use the R package `gbm` do duplicate the plot in the book on comparing boosting with different sized trees. I could not get the `gbm` package to perform boosting with 100 node trees due to memory problems in the `gbm` package. I think the book was using the MART package to do these plots and thus might have been able to perform boosting with these larger trees. The largest tree size I could boost with using `gbm` was around 20. When I run the above script, I get the plot shown in Figure 20. This plot shows that AdaBoost and stumps seem to perform best on this problem in that they have out of sample learning curves that continue to improve as we perform boosting iterations. Gradient boosting with 10 vs. 20 nodes seems to show qualitatively different behaviour. The 10 node boosting algorithm seems to seem to be overfitting as we continue boosting, while boosting with 20 node trees does not show this behaviour.

Exercise Solutions

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

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

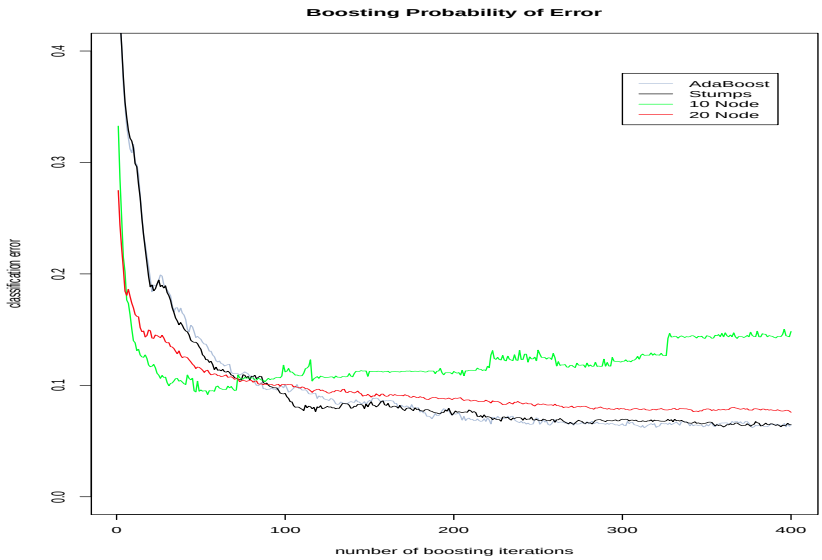

Figure 20: An attempted duplication of the Figure 10.9 from the book. The book's learning curve for AdaBoost does not perform as well as the result shown here.

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

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

Finally we have that β is given by

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

which is the expression Eq. 10.12 as desired.

Ex. 10.2 (minimize the AdaBoost loss)

We want to find $f^*(x)$ such that

$$f^*(x) = \text{argmin}_{f(x)} E_{Y|x}(e^{-Yf(x)}) .$$

To find $f(x)$, we take the derivative of the above objective function with respect to $f(x)$, set the resulting expression equal to zero, and solve for $f(x)$. This procedure would give the equation

$$\frac{\partial}{\partial f} E_{Y|x}(e^{-Yf(x)}) = E_{Y|x}(-Ye^{-Yf(x)}) = 0 .$$

Now evaluating the above expectation when our targets are $Y = \pm 1$ gives

$$-(-1)e^{-(-1)f(x)}\text{Pr}(Y = -1|x) - 1(+1)e^{-f(x)}\text{Pr}(Y = +1|x) = 0 .$$

Multiplying the above by $e^{f(x)}$ gives

$$e^{2f(x)}\text{Pr}(Y = -1|x) - \text{Pr}(Y = +1|x) = 0 ,$$

or

$$e^{2f(x)} = \frac{\text{Pr}(Y = +1|x)}{\text{Pr}(Y = -1|x)} ,$$

or solving for $f(x)$ we get

$$f(x) = \frac{1}{2} \log \left(\frac{\text{Pr}(Y = +1|x)}{\text{Pr}(Y = -1|x)} \right) . \tag{105}$$

the expression we were to show.

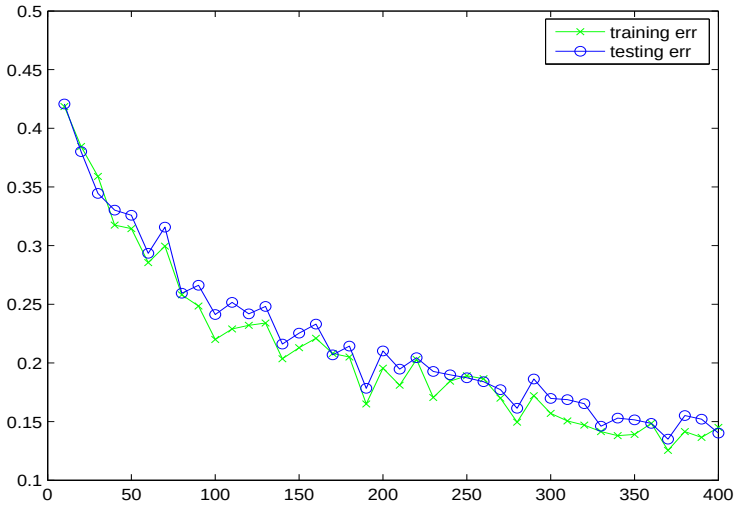

Figure 21: A duplication of the Figure 10.2 from the book.

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 21 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 21 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 purported advantages of the AdaBoost algorithm.

Ex. 10.5 (Zhu's multiclass exponential loss function)

Part (a): We want to find the vector function $f^*(x)$ such that

$$f^*(x) = \text{argmin}_{f(x)} E_{Y|x}[L(Y,f)] = \text{argmin}_{f(x)} E_{Y|x}\left[\exp\left(-\frac{1}{K}Y^T f\right)\right] ,$$

and where the components of $f(x)$ satisfy the constraint $\sum_{k=1}^{K} f_k(x) = 0$. To do this we will introduce the Lagrangian $\mathcal{L}$ defined as

$$\mathcal{L}(f;\lambda) \equiv E_{Y|x}\left[\exp\left(-\frac{1}{K}Y^T f\right)\right] - \lambda\left(\sum_{k=1}^{K} f_k - 0\right) .$$

Before applying the theory of continuous optimization to find the optima of $\mathcal{L}$ we will first evaluate the expectation

$$\mathcal{E} \equiv E_{Y|x}\left[\exp\left(-\frac{1}{K}Y^T f\right)\right] .$$

Note that by expanding the inner products the above expectation can be written as

$$E_{Y|x}\left[\exp\left(-\frac{1}{K}(Y_1 f_1 + Y_2 f_2 + \cdots + Y_{K-1} f_{K-1} + Y_K f_K\right)\right] .$$

We then seek to evaluate the above expression by using the law of the unconscious statistician

$$E[f(X)] \equiv \sum f(x_i)p(x_i) .$$

In the case we will evaluate the above under the encoding for the vector Y given by

$$Y_k = \begin{cases} 1 & k = c \\ -\frac{1}{K-1} & k \neq c \end{cases} .$$

This states that when the true class of our sample x is from the class c, the response vector Y under this encoding has a value of 1 in the c-th component and the value $-\frac{1}{K-1}$ in all other components. Using this we get the conditional expectation given by

$$\begin{aligned}
\mathcal{E} &= \exp\left\{-\frac{1}{K}\left(-\frac{1}{K-1}f_1(x) + f_2(x) + \cdots + f_{K-1}(x) + f_K(x)\right)\right\}\text{Prob}(c=1|x) \\
&+ \exp\left\{-\frac{1}{K}\left(f_1(x) - \frac{1}{K-1}f_2(x) + \cdots + f_{K-1}(x) + f_K(x)\right)\right\}\text{Prob}(c=2|x) \\
&\vdots \\
&+ \exp\left\{-\frac{1}{K}\left(f_1(x) + f_2(x) + \cdots - \frac{1}{K-1}f_{K-1}(x) + f_K(x)\right)\right\}\text{Prob}(c=K-1|x) \\
&+ \exp\left\{-\frac{1}{K}\left(f_1(x) + f_2(x) + \cdots + f_{K-1}(x) - \frac{1}{K-1}f_K(x)\right)\right\}\text{Prob}(c=K|x) .
\end{aligned}$$

Now in the exponential arguments in each of the terms above by using the relationship

$$-\frac{1}{K-1} = \frac{K-1-K}{K-1} = 1 - \frac{1}{K-1} ,$$

we can write the above as

$$\begin{aligned}
\mathcal{E} &= \exp\left\{-\frac{1}{K}\left(f_1(x) + f_2(x) + \cdots + f_{K-1}(x) + f_K(x) - \frac{K}{K-1}f_1(x)\right)\right\}\text{Prob}(c=1|x) \\
&+ \exp\left\{-\frac{1}{K}\left(f_1(x) + f_2(x) + \cdots + f_{K-1}(x) + f_K(x) - \frac{K}{K-1}f_2(x)\right)\right\}\text{Prob}(c=2|x) \\
&\vdots \\
&+ \exp\left\{-\frac{1}{K}\left(f_1(x) + f_2(x) + \cdots + f_{K-1}(x) + f_K(x) - \frac{K}{K-1}f_{K-1}(x)\right)\right\}\text{Prob}(c=K-1|x) \\
&+ \exp\left\{-\frac{1}{K}\left(f_1(x) + f_2(x) + \cdots + f_{K-1}(x) + f_K(x) - \frac{K}{K-1}f_K(x)\right)\right\}\text{Prob}(c=K|x) .
\end{aligned}$$

Using the constraint is $\sum_{k'=1}^{K} f_{k'}(x) = 0$ the above simplifies to

$$\begin{aligned}\mathcal{E} &= \exp\left\{\frac{1}{K-1}f_1(x)\right\}\text{Prob}(c=1|x) + \exp\left\{\frac{1}{K-1}f_2(x)\right\}\text{Prob}(c=2|x) \\ &\vdots \\ &+ \exp\left\{\frac{1}{K-1}f_{K-1}(x)\right\}\text{Prob}(c=K-1|x) + \exp\left\{\frac{1}{K-1}f_K(x)\right\}\text{Prob}(c=K|x)\,.\end{aligned}$$

Then to find the vector $f(x)$, we take the derivative of the Lagrangian objective function $\mathcal{L}$ with respect to each component $f_k(x)$ (and the Lagrangian multiplier λ), set the resulting expressions equal to zero, and solve the resulting system of equations for $f_k(x)$. Recall that to get the full function form for $\mathcal{L}$ we need to add $-\lambda\sum_{k=1}^{K} f_k$ to $\mathcal{E}$. With this the derivative with respect to $f_k(x)$ of our Lagrangian gives

$$\frac{\partial\mathcal{L}}{\partial f_k} = \frac{\partial\mathcal{E}}{\partial f_k} - \lambda = \frac{1}{K-1}\exp\left\{\frac{1}{K-1}f_k(x)\right\}\text{Prob}(c=k|x) - \lambda\,, \tag{106}$$

for $1 \le k \le K$. The derivative of the Lagrangian with respect to λ gives back the constrain equation $\sum_{k'=1}^{K} f_{k'}(x) = 0$. To solve these equations we will solve each of Equation 106 for $f_k(x)$ in terms of λ and then put these equations back into the constraint that all f_k must sum to zero. We find that $f_k(x)$ in terms of λ is given by

$$f_k(x) = -(K-1)\log\left(\frac{-(K-1)\lambda}{\text{Prob}(c=k|x)}\right)\,,$$

or

$$f_k(x) = -(K-1)\log(\text{Prob}(c=k|x)) - (K-1)\log(-(K-1)\lambda)\,. \tag{107}$$

Enforcing that this expression must sum to zero means that

$$(K-1)\sum_{k'=1}^{K}\log(\text{Prob}(c=k'|x)) - K(K-1)\log(-(K-1)\lambda) = 0\,.$$

If we divide by $(K-1)K$ we get

$$\log(-(K-1)\lambda) = \frac{1}{K}\sum_{k'=1}^{K}\log(\text{Prob}(c=k'|x))\,,$$

or solving for λ we get

$$\lambda = -\frac{1}{K-1}\exp\left(\frac{1}{K}\sum_{k'=1}^{K}\log(\text{Prob}(c=k'|x))\right)\,.$$

When we put this expression for λ in Equation 107 we get

$$\begin{aligned}f_k(x) &= (K-1)\log(\text{Prob}(c=k|x)) - \frac{K-1}{K}\sum_{k'=1}^{K}\log(\text{Prob}(c=k'|x)) \\ &= (K-1)\left(\log(\text{Prob}(c=k|x)) - \frac{1}{K}\sum_{k'=1}^{K}\log(\text{Prob}(c=k'|x)\right)\,,\end{aligned}$$

for $1 \leq k \leq K$. We can view the above as K equations for the K unknowns $\text{Prob}(c = k|x)$. To find these probabilities in terms of $f_k(x)$ we first write the above as

$$\frac{1}{K-1} f_k(x) = \log(\text{Prob}(c = k|x)) + \log\left(\left[\prod_{k'=1}^{K} \text{Prob}(c = k'|x) \right]^{-1/K} \right) .$$

From this we can solve for $\text{Prob}(c = k|x)$ to get

$$\text{Prob}(c = k|x) = \left[\prod_{k'=1}^{K} \text{Prob}(c = k'|x) \right]^{1/K} e^{\frac{f_k(x)}{K-1}} . \tag{108}$$

If we sum both sides of this equation from $k' = 1$ to $k' = K$ the left-hand-side must sum to one, the term in brackets is constant with respect to the summation index, and we get

$$1 = \left[\prod_{k'=1}^{K} \text{Prob}(c = k'|x) \right]^{1/K} \sum_{k'=1}^{K} e^{\frac{f_{k'}(x)}{K-1}} .$$

Using this expression we can solve for the term in brackets to find

$$\left[\prod_{k'=1}^{K} \text{Prob}(c = k'|x) \right]^{1/K} = \frac{1}{\sum_{k'=1}^{K} e^{\frac{f_{k'}(x)}{K-1}}} .$$

Using this expression in Equation 108 we get

$$\text{Prob}(c = k|x) = \frac{e^{\frac{f_k(x)}{K-1}}}{\sum_{k'=1}^{K} e^{\frac{f_{k'}(x)}{K-1}}} ,$$

one of the desired expressions.

Ex. 10.7 (the optimal offset $\hat{\gamma}_{jm}$)

We want to find the optimal constants γ_{jm} in each region of the tree. We have

$$\begin{aligned}
\hat{\gamma}_{jm} &= \text{argmin}_{\gamma_{jm}} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma_{jm}) \\
&= \text{argmin}_{\gamma_{jm}} \sum_{x_i \in R_{jm}} e^{-y_i f_{m-1}(x_i) - y_i \gamma_{jm}} \quad \text{since we assume exponential loss} \\
&= \text{argmin}_{\gamma_{jm}} \sum_{x_i \in R_{jm}} w_i^{(m)} e^{-y_i \gamma_{jm}} \quad \text{since } w_i^{(m)} \equiv e^{-y_i f_{m-1}(x_i)} .
\end{aligned}$$

Define a function of γ_{jm} by

$$F(\gamma_{jm}) \equiv \sum_{x_i \in R_{jm}} w_i^{(m)} e^{-y_i \gamma_{jm}} ,$$

then to find the minimum of this we solve

$$F'(\gamma_{jm}) = \sum_{x_i \in R_{jm}} w_i^{(m)} e^{-y_i \gamma_{jm}} (-y_i) = 0 \,.$$

We can write the above summation in two parts by introducing indicator functions. When we use these the above equation can be written as

$$- \sum_{x_i \in R_{jm}} w_i^{(m)} e^{-\gamma_{jm}} I(y_i = +1) + \sum_{x_i \in R_{jm}} w_i^{(m)} e^{\gamma_{jm}} I(y_i = -1) = 0 \,.$$

Notice that $e^{\gamma_{jm}}$ in the sums above are independent of the summation index. Next multiply the above equation by $e^{\gamma_{jm}}$ to get

$$e^{2\gamma_{jm}} \sum_{x_i \in R_{jm}} w_i^{(m)} I(y_i = -1) = \sum_{x_i \in R_{jm}} w_i^{(m)} I(y_i = +1) \,,$$

or solving for γ_{jm} we find

$$\gamma_{jm} = \frac{1}{2} \log \left(\frac{\sum_{x_i \in R_{jm}} w_i^{(m)} I(y_i = +1)}{\sum_{x_i \in R_{jm}} w_i^{(m)} I(y_i = -1)} \right) \,.$$

Note that in the above I have a factor of $1/2$ that the book does not have. Note also that a factor of $1/2$ appears in a similar calculation that results in Equation 105. I wonder if this missing $1/2$ in the book might be a typo. If anyone sees anything wrong with what I have done please contact me.

Ex. 10.8

Part (a): Note that if we approximate the class conditional probabilities using

$$p_k(x) = \frac{e^{f_k(x)}}{\sum_{l=1}^K e^{f_l(x)}} \,, \tag{109}$$

then $p_k(x) > 0$ and $\sum_{k=1}^K p_k(x) = 1$ as required for a probability model. Note that as specified there is redundancy in this model since we can add an arbitrary function $h(x)$ to each function $f_k(x)$ and the value of $p_k(x)$ does not change. Thus we will impose the constraint that $\sum_{l=1}^K f_l(x) = 0$ to eliminate this redundancy. For this problem we are asked to consider the log-likelihood (which is the *negative* multinomial deviance) and is specified in the book. For a single training example the log-likelihood is given by

$$L(y, p(x)) = \sum_{k=1}^K I(y = \mathcal{G}_k) \log(p_k(x)) = \sum_{k=1}^K I(y = \mathcal{G}_k) f_k(x) - \log \left(\sum_{l=1}^K e^{f_l(x)} \right) \,. \tag{110}$$

Part (b): If in the region R we want to increment $f_k(x)$ by some amount γ_k we should consider the total log-likelihood over all samples in the region R and the total log-likelihood then becomes (using the encoding of y specified)

$$LL \equiv \sum_{x_i \in R} \sum_{k=1}^K y_{ik} f_k(x_i) - \sum_{x_i \in R} \log \left(\sum_{l=1}^K e^{f_l(x_i)} \right) \,.$$

When we increment $f_k(x)$ by γ_k our log-likelihood becomes

$$LL(\boldsymbol{\gamma}) = \sum_{x_i \in R} \sum_{k=1}^{K} y_{ik}(f_k(x_i) + \gamma_k) - \sum_{x_i \in R} \log \left(\sum_{l=1}^{K} e^{f_l(x_i)+\gamma_l} \right) .$$

As we are going to use Newton's algorithm to find the maximum of the log-likelihood (or the minimum of the deviance) with respect to the $K-1$ values γ_k (the Kth value is taken to be zero) we will need to evaluate the first and second derivatives of LL with respect to these variables. We find

$$\frac{\partial}{\partial \gamma_k} LL(\boldsymbol{\gamma}) = \sum_{x_i \in R} y_{ik} - \sum_{x_i \in R} \left(\frac{e^{f_k(x_i)+\gamma_k}}{\sum_{l=1}^{K} e^{f_l(x_i)+\gamma_l}} \right) ,$$

for $1 \leq k \leq K-1$. Next we need to take the derivative of the above with respect to $\gamma_{k'}$. We have two cases to consider, when $k' = k$ and when $k' \neq k$. We find when $k' \neq k$ that

$$\frac{\partial^2}{\partial \gamma_k \partial \gamma_{k'}} LL(\boldsymbol{\gamma}) = - \sum_{x_i \in R} \frac{e^{f_k(x)+\gamma_k} e^{f_{k'}(x)+\gamma_{k'}}}{\left(\sum_{l=1}^{K} e^{f_l(x_i)+\gamma_l} \right)^2} ,$$

and when $k' = k$ that

$$\frac{\partial^2}{\partial \gamma_k \partial \gamma_k} LL(\boldsymbol{\gamma}) = \sum_{x_i \in R} \left(- \frac{e^{2f_k(x)+2\gamma_k}}{\left(\sum_{l=1}^{K} e^{f_l(x_i)+\gamma_l} \right)^2} + \frac{e^{f_k(x)+\gamma_k}}{\left(\sum_{l=1}^{K} e^{f_l(x_i)+\gamma_l} \right)} \right) ,$$

One step of Newton's method will start with a value for $\boldsymbol{\gamma}_0$ and update to get $\boldsymbol{\gamma}_1$ using

$$\boldsymbol{\gamma}_1 = \boldsymbol{\gamma}_0 - \left[\frac{\partial^2 LL(\boldsymbol{\gamma})}{\partial \gamma_k \partial \gamma_{k'}} \right]^{-1} \frac{\partial LL(\boldsymbol{\gamma})}{\partial \gamma_k} .$$

If we start our Newton iterations with $\boldsymbol{\gamma}_0 = 0$ then the gradient and the Hessian simplify. We find

$$\begin{aligned}
\frac{\partial}{\partial \gamma_k} LL(\boldsymbol{\gamma} = 0) &= \sum_{x_i \in R} y_{ik} - \sum_{x_i \in R} p_{ik} = \sum_{x_i \in R} (y_{ik} - p_{ik}) \\
\frac{\partial^2}{\partial \gamma_k \partial \gamma_{k'}} LL(\boldsymbol{\gamma} = 0) &= - \sum_{x_i \in R} p_{ik} p_{ik'} \quad \text{for} \quad k' \neq k \\
\frac{\partial^2}{\partial \gamma_k \partial \gamma_k} LL(\boldsymbol{\gamma} = 0) &= \sum_{x_i \in R} (-p_{ik}^2 + p_{ik}) \quad \text{for} \quad k' = k .
\end{aligned}$$

Here we have defined $p_{ik} = \frac{e^{f_k(x_i)}}{\sum_{l=1}^{K} f_l(x_i)}$. If we assume that the Hessian is diagonal the the matrix inverse becomes a sequence of scalar inverses, and our first Newton iterations become

$$\gamma_k^1 = \frac{\sum_{x_i \in R} (y_{ik} - p_{ik})}{\sum_{x_i \in R} p_{ik}(1 - p_{ik})} ,$$

for $1 \leq k \leq K-1$.

Part (c): The above update will produce values of γ_k^1 that do not sum to zero as our original functions $f_k(x)$ do. I'm not sure how to argue using symmetry for the formula given in the book. It is easy to show that the formula suggested does satisfy the desired requirement that $\sum_{k=1}^K \hat{\gamma}_k = 0$ by simply summing each of the terms. If you assume that the form for the normalized gammas, i.e. $\hat{\gamma}_k$, is an *affine* function of first Newton update γ_k^1 namely

$$\hat{\gamma}_k = a\gamma_k^1 + b\,,$$

then to make sure that $\hat{\gamma}_k$ sums to zero requires that

$$\sum_{k=1}^K \hat{\gamma}_k = a\sum_{k=1}^K \gamma_k^1 + bK = 0 \quad \text{so} \quad b = -\frac{a}{K}\sum_{k=1}^K \gamma_k^1\,.$$

Thus we have shown that

$$\hat{\gamma}_k = a\left[\gamma_k^1 - \frac{1}{K}\sum_{k=1}^K \gamma_k^1\right]\,,$$

but I'm not sure how to argue for the expression given in the book for a. If anyone knows an argument for why the given expression is a good one to use please contact me.

Chapter 14 (Unsupervised Learning)

Notes on the Text

Notes on Unsupervised as Supervised Learning

I believe there is a slight typo in this section of the book or at least the results they present are valid for the case where we duplicate (using the proposed density $g_0(x)$ exactly the same number of cases as we are presented with originally, that is $N_0 = N$. If we do actually only generate $N_0 \neq N$ cases from $g_0(x)$ then defining $w = \frac{N}{N_0+N}$ and $w_0 = \frac{N_0}{N_0+N}$ the density of points x should be given by the two component mixture model

$$f(x) = wg(x) + w_0 g_0(x)\,.$$

Then the expression for $\mu(x) = \mathrm{E}(Y|x)$ is given by

$$\begin{aligned}\mu(x) &= \mathrm{E}(Y|x) = 0P(Y=0|x) + 1P(Y=1|x) = P(Y=1|x) \\ &= \frac{P(x|Y=1)P(Y=1)}{P(x)} = \frac{g(x)w}{wg(x)+w_0 g_0(x)} \\ &= \frac{g(x)}{g(x) + \left(\frac{w_0}{w}\right) g_0(x)}\,.\end{aligned}$$

Note this equals the result in the book if $N_0 = N$.

Notes on Object Dissimilarity

From the definition of $\bar{d}_j$ of

$$\bar{d}_j = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}(x_{ij} - x_{i'j})^2\,,$$

we can transform this to an expression relating $\bar{d}_j$ to the variance of x_j. To do this we first expand the quadratic in the double summation and write $\bar{d}_j$ as

$$\begin{aligned}\bar{d}_j &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}(x_{ij}^2 - 2x_{i'j}x_{ij} + x_{i'j}^2) \\ &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}x_{ij}^2 - \frac{2}{N^2}(N^2\bar{x}_j^2) + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}x_{i'j}^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}x_{ij}^2 - 2\bar{x}_j^2 + \frac{1}{N}\sum_{i'=1}^{N}x_{i'j}^2\,.\end{aligned}$$

To continue, recall that if X is a random variable with N samples x_i then one can show that

$$\mathrm{Var}(X) = \frac{1}{N}\sum_{i=1}^{N} {x_i}^2 - \left(\frac{1}{N}\sum_{i=1}^{N} x_i\right)^2 = \frac{1}{N}\sum_{i=1}^{N} {x_i}^2 - \bar{x}^2 \,,$$

so that

$$\frac{1}{N}\sum_{i=1}^{N} x_i^2 = \mathrm{Var}(X) + \bar{x}^2 \,.$$

Thus using this we see that $\bar{d}_j$ can be written as

$$\bar{d}_j = \mathrm{Var}(x_j) + \bar{x}_j - 2\bar{x}_j^2 + \mathrm{Var}(x_j) + \bar{x}_j = 2\mathrm{Var}(x_j) \,, \tag{111}$$

as claimed in the books equation 14.27.

Notes on the k-means algorithm

Recall the definition of $W(C)$

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k} ||x_i - x_{i'}||^2 \,, \tag{112}$$

then since

$$||x_i - x_{i'}||^2 = (x_i - x_{i'})^T(x_i - x_{i'}) = x_i^T x_i - 2x_i^T x_{i'} + x_{i'}^T x_{i'} \,,$$

then we can write $\sum_{C(i')=k} ||x_i - x_{i'}||^2$ as

$$\sum_{C(i')=k} (x_i^T x_i - 2x_i^T x_{i'} + x_{i'}^T x_{i'}) = N_k x_i^T x_i - 2x_i^T(N_k \bar{x}_k) + \sum_{C(i')=k} x_{i'}^T x_{i'} \,.$$

Next perform the summation of this expression over the points x_i such that $C(i) = k$ to get

$$N_k \sum_{C(i)=k} x_i^T x_i - 2(N_k\bar{x}_k)^T(N_k\bar{x}_k) + N_k \sum_{C(i')=k} x_{i'}^T x_{i'} = 2N_k \left[\sum_{C(i)=k} x_i^T x_i - N_k \bar{x}_k^T \bar{x}_k\right] \,. \tag{113}$$

We would like to show that the expression in brackets is equal to $\sum_{C(i)=k} ||x_i - \bar{x}_k||^2$. To do that we next consider

$$\begin{aligned}
\sum_{C(i)=k} ||x_i - \bar{x}_k||^2 &= \sum_{C(i)=k} (x_i^T x_i - 2x_i^T \bar{x}_k + \bar{x}_k^T \bar{x}) \\
&= \sum_{C(i)=k} x_i^T x_i - 2(N_k\bar{x}_k)^T \bar{x}_k + N_k \bar{x}_k^T \bar{x}_k \\
&= \sum_{C(i)=k} x_i^T x_i - N_k \bar{x}_k^T \bar{x}_k \,,
\end{aligned}$$

so they are equal. Putting $\sum_{C(i)=k} ||x_i - \bar{x}_k||^2$ where the expression in brackets is in Equation 113, and then into Equation 112 we find

$$W(C) = \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - \bar{x}_k||^2 , \tag{114}$$

which is the books equation 14.31.

The book also claims that the solution m to the following minimization problem

$$\text{argmin}_m \sum_{i \in S} ||x_i - m||^2$$

is $\bar{x}_S$ or the mean over the points in the set S. This can be shown by defining $f(m) \equiv \sum_{i \in S} ||x_i - m||^2$ taking the derivative with respect to m, setting the result equal to zero and then solving for m. Recalling that

$$\frac{\partial}{\partial m}(x_i - m)^T A(x_i - m) = -(A + A^T)(x_i - m) ,$$

so $\frac{\partial}{\partial m} \sum_{i \in S} ||x_i - m||^2$ is given by

$$-2 \sum_{i \in S} (x_i - m) .$$

When we set this expression equal to zero and solve for m we have

$$\sum_{i \in S} x_i = \sum_{i \in S} m = |S| m ,$$

or

$$m = \frac{1}{|S|} \sum_{i \in S} x_i = \bar{x}_S ,$$

showing the claimed result. Here we have denoted $|S|$ the number of points in the set S.

Exercise Solutions

Ex. 14.1 (the weighted Euclidean distance)

Consider the expression for $d_e^{(w)}(x_i, x_i')$. We see that

$$d_e^{(w)}(x_i, x_i') = \frac{\sum_{l=1}^{p} w_l (x_{il} - x_{i'l})^2}{\sum_{l=1}^{p} w_l} = \sum_{l=1}^{p} \left(\frac{w_l}{\sum_{l=1}^{p} w_l} \right) (x_{il} - x_{i'l})^2 .$$

Define $s_l = \frac{w_l}{\sum_{l=1}^{p} w_l}$ then the above can be written as

$$d_e^{(w)}(x_i, x_i') = \sum_{l=1}^{p} (\sqrt{s_l} x_{il} - \sqrt{s_l} x_{i'l})^2 .$$

If we define vectors z_i with components given by

$$z_{il} = x_{il}\sqrt{s_l} = x_{il}\left(\frac{w_l}{\sum_{l=1}^{p} w_l}\right)^{1/2} ,$$

then the above shows that $d_e^{(w)}(x_i, x_i')$ is equal to $d_e(z_i, z_i')$ as we were to show.

Ex. 14.2 (k-means as the EM algorithm)

Part 1: The likelihood function for the data set $\{x_i\}_{i=1}^N$ is given by

$$\prod_{i=1}^{N} g(x_i) = \prod_{i=1}^{N}\left(\sum_{k=1}^{K} \pi_k g_k(x)\right) .$$

The loglikelihood is then given by

$$\sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k g_k(x)\right) .$$

Ex. 14.5 (k-means and the SOM algorithm on some semi-spherical data)

Since the `R` programming envionment provides a function for k-means (called `kmean`) and a function for Kohonen's self-organizing maps (called `SOM`) we will use these functions to study this problem.

Ex. 14.6 (k-means and the SOM algorithm on the tumor microarray data)

Ex. 14.7 (PC minimization)

We want to find μ and V_q that minimize

$$\sum_{i=1}^{N} ||x_i - \mu - V_q\lambda_i||^2 . \tag{115}$$

We take the derivative with respect to μ, set the result equal to zero, and solve for μ. We find this derivative given by

$$\frac{\partial}{\partial\mu}\left(\sum_{i=1}^{N}(x_i - \mu - V_q\lambda_i)^T(x_i - \mu - V_q\lambda_i)\right) = \sum_{i=1}^{N} -2(x_i - \mu - V_q\lambda_i) .$$

Setting this equal to zero and solving for μ we find

$$\mu = \bar{x} - V_q \left(\frac{1}{N} \sum_{i=1}^{N} \lambda_i \right) , \tag{116}$$

where $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$. Next we take the derivative of Equation 115, with respect to λ_i, set the result equal to zero and then solve for λ_i. To do this we write the sum in Equation 115 as

$$\sum_{i=1}^{N} [(x_i - \mu)^T - 2(x_i - \mu)^T V_q \lambda_i + \lambda_i^T V_q^T V_q \lambda_i] .$$

From which we see that taking, the $\frac{\partial}{\partial \lambda_i}$ derivative of this expression and setting the result equal to zero is given by

$$-2((x_i - \mu)^T V_q)^T + +(V_q^T V_q + V_q^T V_q)\lambda_i = 0 ,$$

or

$$V_q^T (x_i - \mu) = V_q^T V_q \lambda_i = \lambda_i , \tag{117}$$

since $V_q^T V_q = I_p$. When we put this value of λ_i into Equation 116 to get

$$\mu = \bar{x} - V_q V_q^T (\bar{x} - \mu) .$$

Thus μ must satisfy

$$V_q V_q^T (\bar{x} - \mu) = \bar{x} - \mu ,$$

or

$$(I - V_q V_q^T)(\bar{x} - \mu) = 0 .$$

Now $I - V_q V_q^T$ is the orthogonal projection onto the subspace spanned by the columns of V_q thus $\mu = \bar{x} + h$, where h is an arbitray p dimensional vector in the subspace spanned by V_q. Since V_q has a rank of q the vector h is selected from an $p - q$ dimensional space (the space spanned by the orthogonal complement of the columns of V_q). If we take $h = 0$ then $\mu = \bar{x}$ and we get

$$\lambda_i = V_q^T (x_i - \bar{x}) ,$$

from Equation 117.

Ex. 14.8 (the procrustes transformation)

For the Procrustes transformation we want to evaluate

$$\text{argmin}_{\mu,R} ||X_2 - (X_1 R + \mathbf{1}\mu^T)||_F . \tag{118}$$

This has the same solution when we square the same norm as above

$$\text{argmin}_{\mu,R} ||X_2 - (X_1 R + \mathbf{1}\mu^T)||_F^2 .$$

When we use the fact that $||X||_F^2 = \text{trace}(X^T X)$ we see that the above norm equals

$$\text{trace}((X_2 - X_1 R - \mathbf{1}\mu^T)^T (X_2 - X_1 R - \mathbf{1}\mu^T)) .$$

To minimize this expression with respect to μ and R we next expand the quadratic in the above expression as follows

$$\begin{aligned}((X_2 - X_1R) - \mathbf{1}\mu^T)^T((X_2 - X_1R) - \mathbf{1}\mu^T) &= ((X_2 - X_1R)^T - \mu\mathbf{1}^T)((X_2 - X_1R) - \mathbf{1}\mu^T) \\ &= (X_2 - X_1R)^T(X_2 - X_1R) \\ &- (X_2 - X_1R)^T\mathbf{1}\mu^T - \mu\mathbf{1}^T(X_2 - X_1R) \\ &+ \mu\mathbf{1}^T\mathbf{1}\mu^T\,. \end{aligned} \tag{119}$$

Note that when dealing with scalar inner products the terms linear in μ are equal to each other and therefore double. In this case these two terms $(X_2 - X_1R)^T\mathbf{1}\mu^T$ and $\mu\mathbf{1}^T(X_2 - X_1R)$ are matrices and are not nessisarily equal to each other. Note that some of the expressions in the above simplify. We have $\mathbf{1}^T\mathbf{1} = N$ and

$$\mathbf{1}^T X_2 = N\bar{x}_2^T\,,$$

where $\bar{x}_2^T$ is the columnwise mean of the matrix X_2 i.e.

$$\bar{x}_2^T = \left[\frac{1}{N}\sum_{i=1}^N (X_2)_{i1}, \frac{1}{N}\sum_{i=1}^N (X_2)_{i2}\right]$$

In the same way $\mathbf{1}^T X_1 = N\bar{x}_1$, where $\bar{x}_1^T$ is the columnwise mean of X_1. Recalling that our objective function to minimize is the trace of the four terms in Equation 119 above we can minimize this expression by taking the μ derivative, setting the result equal to zero and solving for μ. To take the derivative of the trace of Equation 119 we need to use Equations 127, 126, and 129 to get

$$-(X_2 - X_1R)^T\mathbf{1} - (X_2 - X_1R)^T\mathbf{1} + 2N\mu\,.$$

Setting this equal to zero and solving for μ gives

$$\mu = \frac{1}{N}(X_2^T - R^T X_1^T)\mathbf{1} = \bar{x}_2 - R^T\bar{x}_1\,. \tag{120}$$

Note: that this somewhat different than the result in the book where the R appears without the transpose. If anyone sees an error in my calculations please let me know. Using this result we see that our minimization argument in Equation 118 now becomes

$$\begin{aligned} X_2 - X_1R - \mathbf{1}\mu^T &= X_2 - X_1R - \mathbf{1}(\bar{x}_2^T - \bar{x}_1^T R) \\ &= X_2 - \mathbf{1}\bar{x}_2^T - (X_1 - \mathbf{1}\bar{x}_1^T)R \\ &= \tilde{X}_2 - \tilde{X}_1 R\,, \end{aligned}$$

where we have introduced the mean reduced matrices $\tilde{X}_i$ as defined in the book. Using this expression in Equation 118 and expanding the quadratic as before the minimization problem we are trying to solve becomes

$$\text{argmin}_R \text{trace}(\tilde{X}_2^T\tilde{X}_2 - \tilde{X}_2^T\tilde{X}_1 R - R^T\tilde{X}_1^T\tilde{X}_2 + R^T\tilde{X}_1^T\tilde{X}_1 R)\,.$$

To minimize this we take the derivative with respect to R, set the result equal to zero, and solve for R. Taking this derivative and using Equations 125, 128, and 129 we get

$$-\tilde{X}_1^T\tilde{X}_2 - \tilde{X}_1^T\tilde{X}_2 + 2\tilde{X}_1^T\tilde{X}_1 R = 0\,.$$

when we solve for R we get

$$R = (\tilde{X}_1^T \tilde{X}_1)^{-1}(\tilde{X}_1^T \tilde{X}_2)\,. \tag{121}$$

Warning: this result is different that what is claimed in the book and I don't see how to make the two the same. If anyone knows how to make the results equivalent please let me know. One piece of the puzzel is that R is supposted to be orthogonal. I have not explicitly enforeced this constraint in any way. I think I need to modify the above minimization to add the constraint that $R^T R = I$.

This problem is discussed in greater detail in [9].

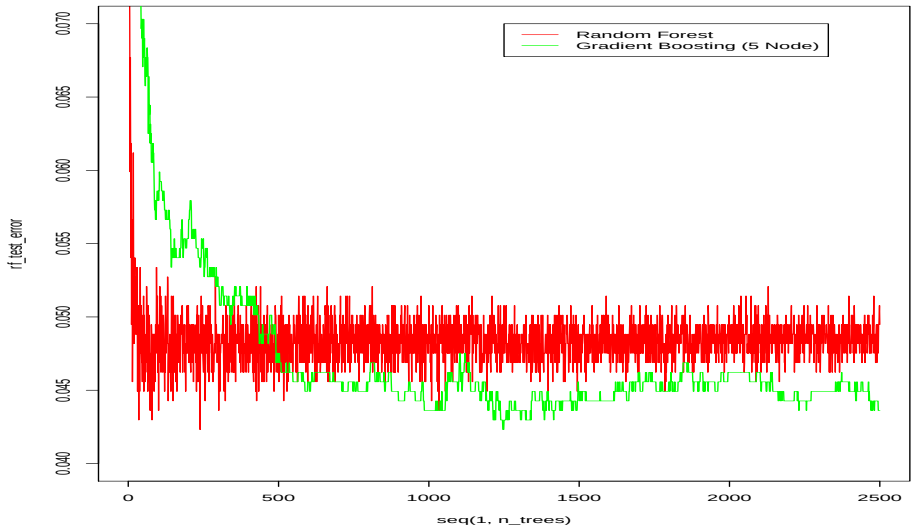

Figure 22: A duplication of the books Figure 15.1, comparing random forests with gradient boosting on the "spam" data.

Chapter 15 (Random Forests)

Notes on the Text

Duplicating Figure 15.1 (classification performance on the spam data)

In the `R` script `dup_fig_15_1.R` we duplicate the results from the books Figure 15.1 comparing the classification performance of random forests and gradient boosting on the "spam" data. When that script is run it generates the results given in Figure 22. This plot is qualitatively the same as given in the book. My random forest test error rate seems more variable than the one that the book presents. The general conclusion presented in the book still seems to hold in that the gradient boosting algorithm seems to reach a lower error rate than random forest algorithm.

Duplicating Figure 15.2 (classification of the nested spheres data)

In the `R` script `dup_fig_15_2.R` we duplicate the results from the books Figure 15.2 comparing random forests with gradient boosting on the "nested spheres" data. When that script is run it generates the results given in Figure 23. This plot looks quantitatively very similar to the one given in the book.

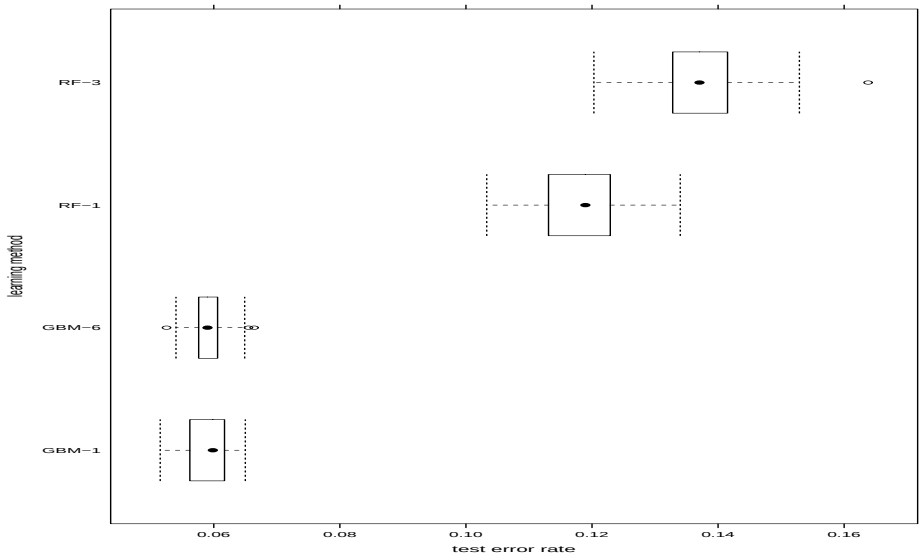

Figure 23: A duplication of the books Figure 15.2, comparing random forests with gradient boosting on the "nested spheres" data.

Exercise Solutions

Ex. 15.1 (the variance of the mean of correlated samples)

We are told to assume that $x_i \sim N(m, \sigma^2)$ for all i and that x_i and x_j are correlated with a correlation coefficient of ρ. We first compute some expectation of powers of x_i. First recall from probability that the variance in terms of the moments is given by

$$E[X^2] = \mathrm{Var}(X) + E[X]^2 \,.$$

Next the definition of the correlation coefficient ρ we have that

$$\frac{1}{\sigma^2} E[(x_i - m)(x_j - m)] = \rho > 0 \,.$$

We can expand the above expectation to get

$$E[x_i x_j] = \rho\sigma^2 + m^2 \,.$$

Thus we now have shown that

$$\begin{aligned} E[x_i] &= m \\ E[x_i x_j] &= \begin{cases} \rho\sigma^2 + m^2 & i \neq j \\ \sigma^2 + m^2 & i = j \end{cases} \,. \end{aligned}$$

The variance of the estimate of the mean is now given by

$$\mathrm{Var}\left(\frac{1}{B}\sum x_i\right) = \frac{1}{B^2}\mathrm{Var}\left(\sum x_i\right) = \frac{1}{B^2}\left[E\left[\left(\sum x_i\right)^2\right] - E\left[\sum x_i\right]^2\right]$$

The second expectation is easy to evaluate

$$E\left[\sum x_i\right] = \sum E[x_i] = Bm\,.$$

For the first expectation note that

$$\left(\sum_{i=1}^{n} x_i\right)^2 = \sum_{i,j} x_i x_j\,.$$

Thus we have

$$\begin{aligned} E\left[\left(\sum x_i\right)^2\right] &= \sum_{i,j} E[x_i x_j] = BE[x_i^2] + (B^2 - B)E[x_i x_j] \\ &= B(\sigma^2 + m^2) + B(B-1)(\rho\sigma^2 + m^2) \\ &= B\sigma^2 + B^2\rho\sigma^2 - B\rho m^2 + B^2 m^2\,. \end{aligned}$$

With this expression we can now compute $\mathrm{Var}\left(\frac{1}{B}\sum x_i\right)$ and find

$$\mathrm{Var}\left(\frac{1}{B}\sum x_i\right) = \frac{\sigma^2}{B} + \rho\sigma^2 - \frac{\rho\sigma^2}{B} = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2\,. \tag{122}$$

Which matches the expression in the book.

A Appendix

A.1 Matrix and Vector Derivatives

In this section of the appendix we enumerate several matrix and vector derivatives that are used in the previous document. We begin with some derivatives of scalar forms

$$\frac{\partial \mathbf{x}^T\mathbf{a}}{\partial \mathbf{x}} = \frac{\mathbf{a}^T\mathbf{x}}{\partial \mathbf{x}} = \mathbf{a} \tag{123}$$

$$\frac{\partial \mathbf{x}^T\mathbf{B}\mathbf{x}}{\partial \mathbf{x}} = (\mathbf{B}+\mathbf{B}^T)\mathbf{x}\,. \tag{124}$$

Next we present some derivatives involving traces. We have

$$\frac{\partial}{\partial \mathbf{X}}\mathrm{trace}(\mathbf{A}\mathbf{X}) = \mathbf{A}^T \tag{125}$$

$$\frac{\partial}{\partial \mathbf{X}}\mathrm{trace}(\mathbf{X}\mathbf{A}) = \mathbf{A}^T \tag{126}$$

$$\frac{\partial}{\partial \mathbf{X}}\mathrm{trace}(\mathbf{A}\mathbf{X}^T) = \mathbf{A} \tag{127}$$

$$\frac{\partial}{\partial \mathbf{X}}\mathrm{trace}(\mathbf{X}^T\mathbf{A}) = \mathbf{A} \tag{128}$$

$$\frac{\partial}{\partial \mathbf{X}}\mathrm{trace}(\mathbf{X}^T\mathbf{A}\mathbf{X}) = (\mathbf{A}+\mathbf{A}^T)\mathbf{X}\,. \tag{129}$$

Derivations of expressions of this form are derived in [3, 4].

References

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

[2] M. H. DeGroot. *Optimal Statistical Decisions*. 2004.

[3] P. A. Devijver and J. Kittler. *Pattern recognition: A statistical approach*. Prentice Hall, 1982.

[4] P. S. Dwyer and M. S. Macphail. Symbolic matrix derivatives. *Annals of Mathematical Statistics*, 19(4):517–534, 1948.

[5] J. H. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 12 2009.

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

[7] B.-H. Mevik and R. Wehrens. The pls package: Principal component and partial least squares regression in R. *Journal of Statistical Software*, 18(2):1–24, 1 2007.

[8] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, August 2000.

[9] P. Schnemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[10] S. Weisberg. *Applied Linear Regression*. Wiley, New York, 1985.

[11] D. Wright and K. London. *Modern regression techniques using R: a practical guide for students and researchers*. SAGE, 2009.