The Elements of Statistical Learning: Data Mining, Inference, and Prediction

Trevor Hastie, Robert Tibshirani, Jerome Friedman

Notes by Shravan Ravi, Rohan Potru, Aniruddh Sriram

Contents

Abstract

1 Intro

These are comprehensive notes of ESLII prepared by students.

2 Overview of Supervised Learning

The goal of supervised learning is to use a set of variables or *inputs* to predict some *outputs*. In machine learning, this task leverages different terminology with a similar goal of use a set of predictors or features to generate predictions or responses. In this section, we examine two basic methods of this supervised learning task and their respective applications.

2.1 Linear Models

Linear regression works over a vector of inputs $X^T = (X_1, X_2, X_3, \ldots, X_p)$ to predict an output Y via the model

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j = X^T \beta \tag{2.1.1}$$

In the trivial case, $\hat{\beta}$ is a vector and $\hat{Y}$ is a scalar. We can let $\hat{Y}$ become a k-dimensional vector by letting $\hat{\beta}$ extend from a p-dimentional vector to a $p \times k$ matrix of weights on X^T

In the $p+1$ dimensional input-output space $(X, \hat{Y})$ represents a hyperplane. Thus, viewed as function in the p-dimensional input space, $f(X) = X^T\beta$ is linear and the gradient $f'(X) = \beta$ is a vector in the input space that points in the steepest uphill direction.

2.2 Fitting a Linear Model and Least Squares

In least square regression we select coefficients $\hat{\beta}$ to minimize the residual sum of squares.

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2 \tag{2.2.1}$$

Since $RSS(\beta)$ is a quadratic function, its minimum always exists, however, it may not be unique. Written in matrix notion

$$RSS(\beta) = (y - X\beta)^T (y - X\beta) \tag{2.2.2}$$

where X is an $N \times p$ matrix where each row is an input vector and y is an N-vector of outputs.

Differentiating with respect to β, we get the normal equations

$$X^T(y - X\beta) = 0 \tag{2.2.3}$$

Intuition: Minimum exists where gradient is 0

If $X^T X$ is nonsingular (invertable)

$$\beta = (X^T X)^{-1} X^T y \tag{2.2.4}$$

This produces a fitted surface that is characterized by the p parameters β that maps the function $\hat{y}_i = \hat{y}(x_i) = x_i^T \hat{\beta}$

2.3 Nearest Neighbors

Nearest neighbor methods use observations from the training set closes in input space to x to form $\hat{Y}$ by averaging their responses. For k-nearest neighbor fit

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \tag{2.3.1}$$

where $N_k(x)$ defines the neighborhood of x determined by the k closes points to x in the training set

For k-nearest neighbor models, it seems there is 1 parameter k as opposed to the p parameters that define $\hat{\beta}$ in linear regression for a p-dimensional input space. We should note however that the effective number of parameters (degrees of freedom) is N/k which is generally bigger than p. This is related to the number of neighborhoods - should they be non-overlapping - that are each defined by one parameter (mean of the neighborhood).

2.4 Statistical Decision Theory

Consider a p-dimentional input space $X \in \mathbb{R}^p$ where X is a real values input vector.

Let $Y \in \mathbb{R}$ be a real valued random output variable with the join distribution $Pr(X, Y)$ We look to find $f(X)$ for predicting Y given input X. For this task, we leverage a loss function $L(Y, f(X))$ to penalize the error between predicted and true values.

Considering the most common squared error loss, $L(Y, f(X)) = (Y - f(X))^2$ we can formulate the following criterion for the expected prediction error (EPE)

$$EPE(f) = \mathbb{E}[(Y - f(x)^2]$$

$$= \int [y - f(x)]^2 Pr(dx, dy)$$

By conditioning on X, we see

$$EPE(f) = \mathbb{E}_X \mathbb{E}_{Y|X}[(Y - f(X))^2 | X]$$

Since we look to minimize the expected prediction error, we can find that

$$f(X) = \mathbb{E}[Y|X = x] \tag{2.4.1}$$

The solution for $f(X)$ is the condition expectation or regression function. Thus the best prediction at any point $X = x$ is the conditional mean when using squared error as the loss.

Nearest neighbors by definition directly implements this conditional expectation since at each point x, we find the average of all y with inputs $x_i = x$ which lets us approximate $\mathbb{E}[Y|X = x]$ as $\hat{f}(x) = Ave(y_i|x_i \in N_k(x))$

As the input space increases in dimentionaity, the size of the neighborhood increases. Thus the rate of convergence onto the true mean of each neighborhood will decrease exponentially as dimentionality increases (more on page 23).

Linear regression directly follows from this framework if you make the assumption $f(X)$ is approximately linear such that $f(X) \approx x^T\beta$. We can then use this formulation of $f(X) = x^T\beta$ in the $EPE(f)$ formula and find the minimum of the function through differentiating to solve for β theoretically.

$$\beta = (\mathbb{E}[XX^T])^{-1}\mathbb{E}[XY] \tag{2.4.2}$$

This leverages the assumption of global linearilty of $f(X)$ to pool over values of X and circumvent the conditioning on X previously used in the regression equation $f(X) = \mathbb{E}[Y|X = x]$

2.5 Curse of Dimentionality

As the number of dimensions in the input space increase, in order to maintain the same density of data, and by extension the same level of proximity used to estimate $X = x$ using a neighborhood of points, the number of samples increase exponentially with the relationship

$$d(p, N) = (1 - \frac{1}{2}^{\frac{1}{N}})^{\frac{1}{p}} \tag{2.5.1}$$

where p is the number of dimensions, N is the number of samples and $d(p, N)$ is the density of values in the input space.

Consider the curse for the linear approximator. Assume the relation ship between Y and X is approximately linear

$$Y = X^T\beta + \epsilon \quad \text{where} \quad \epsilon \sim N(0, \sigma^2) \tag{2.5.2}$$

Now consider the point

$$\hat{y}_0 = \hat{x}_0^T\hat{\beta}$$

$$= x_0^T\beta + \sum_{i=1}^{N} \ell_i(x_0)\epsilon_i \tag{2.5.3}$$

where $\ell_i(x_0)$ is the i_{th} entry of $X(X^TX)^{-1}x_0$
Since the least square estimates are unbiased, we can see that

$$\begin{aligned} EPE(x_0) &= \mathbb{E}_{y_0|x_0}\mathbb{E}_{\mathcal{T}}(y_0 - \hat{y}_0)^2 \\ &= Var(y_0|x_0) + Var_{\mathcal{T}}(\hat{y}_0 + Bias^2(\hat{y}_0) \\ &= \sigma^2 + \mathbb{E}_{\mathcal{T}}x_0^T(X^TX)^{-1}x_0\sigma^2 + 0^2 \end{aligned} \tag{2.5.4}$$

Therefore, from the error term ϵ, we incur an additional variance σ^2 in the prediction error since the target is not deterministic. Since there is no bias, and the variance depends on x_0, if N is large and $\mathcal{T}$ were selected at random, and assuming $\mathbb{E}[X] = 0$ which implies $X^TX = NCov(X)$, then

$$\mathbb{E}_{x_0}EPE(x_0) \sim \mathbb{E}_{x_0}x_0^TCov(X)^{-1}x_0\sigma^2/N + \sigma^2$$

$$= \sigma^2(\frac{p}{N}) + \sigma^2 \tag{2.5.5}$$

Thus EPE increases linearly as a unction of p with slope $\frac{\sigma}{N}$ (More on page 26)

2.6 Maximum Likelihood Estimation

Let y_i be a random sample from a density $Pr_\theta(y)($ indexed by some parameters θ. The log-probability of the observed sample is

$$L(\theta) = \sum_{i=1}^{N} log(Pr_\theta(y_i) \tag{2.6.1}$$

The principle of maximum likelihood assumes that the most reasonable values for θ are those for which the probability of the observed sample is largest. When log-likelihood (cross-entropy) is maximized, we get the describing parameters θ that best conforms to the given data. (More on page 31)

2.7 Function Classes

In order to obtain useful results for finite N training samples. we must restrict the eligible solutions for the functions f used in minimizing the selected loss function $L(y, f(x))$ to a smaller set of functions. This set of constraints imposed by a given learning method is described as a complexity restriction which leverages some kind of regular behavior in small neighborhoods of the input space.

Equivalent kernels describe this local dependence for any method linear in the outputs. These kernels in many cases peak at the target point and fall away smoothly. It is important to note that any method that attempts to produce locally varying functions in small isotropic neighborhoods will run into problems in high dimensions and vice versa, any method that overcomes the dimensionality problem, does not allow neighborhoods to be simultaneously small in all directions.
To control the effect size of the local neighborhood and provide the necessary restrictions on the function space used to fit f, we describe difference function classes that are associated with one or more smoothing parameters.

For example, roughness penalties leverage penalties added to the norms or derivatives of a fitting function work to generate a smoother function that by imposing special structures to a local neighborhood.

Another example is kernel functions that explicitly provide estimates of the regression function by similarly specifying the nature of the local neighborhood. These local neighborhoods are specified by a kernel function $K_\lambda(x_0, x)$ that assigns weights to points x in a region around the giving training point x_0. For example, the Gaussian kernel shown below has a weight function based on the Gaussian density function that assigns weight to the points that reduce exponentially with their squared Euclidean distance from x_0 (more on page 34 & 35)
One final example is basis functions and dictionary methods that adaptively select from an infinitely large set of functions that are constrained based on a given criterion (polynomial, activation functions, splines/piecewise polynomials, etc) and develop models by employing some kind of search mechanism (gradient descent, etc).

$$K_\lambda = \frac{1}{\lambda} exp[-\frac{\|x - x_0\|^2}{2\lambda}] \tag{2.7.1}$$

2.8 Bias - Variance Trade-off

Consider the k-nearest-neighbor regression fit $\hat{f}_k(x_0)$ for data sourced from a model $Y = f(X) + \epsilon$ with $E[\epsilon] = 0$ and $Var(\epsilon) = \sigma^2$. We can formulate the expected prediction error at x_0, also known as the test or generalization error as follows.

$$\begin{aligned} EPE_k(x_0) &= \mathbb{E}[Y - \hat{f}_k(x_0))^2 | X = x_0] \\ &= \sigma^2 + [f(x_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(x_\ell)]^2 + \frac{\sigma^2}{k} \end{aligned} \tag{2.8.1}$$

where ℓ represents the sequence of nearest neighbors to x_0.

This first term σ^2 is the irreducible error. The second and third terms, however, are both controllable and comprise the mean squared error of $\hat{f}_k(x_0)$. This can further be broken down into a bias and variance term.

The bias term is the squared difference between the true mean $f(x_0)$ and the expected value of the estimate.

$$[\mathbb{E}_{\mathcal{T}}(\hat{f}_k(x_0)) - f(x_0)]^2 \tag{2.8.2}$$

This expectation of of the predictor on x_0 work to average the randomness in the training data. As the model complexity increases, the function will be better suited at predicting f_0 and thus the squared bias term should decrease. The variance term is the variance of an average and increases as the model complexity increases.

3 Linear Methods for Regression

Basic linear models can outperform more complex models in situations with a small number of training data, low signal to noise ratios or sparse data.

3.1 Linear Models and Least Squares

Recall from the previous section that using the least squares criterion that we could theoretically solve for the fitting coefficients $\hat{\beta} = (X^T X)^{-1} X^T y$. Thus, when predicting, we can determine the predicted values $\hat{y} = X\hat{\beta}$ The transformation of y onto $\hat{y}$ is accomplished by the matrix $H = X(X^T X)^{-1} X^T$ also known as the hat matrix. We look to minimize $RSS(\beta) = \|y - X\beta\|^2$ by choosing $\hat{\beta}$ such that the residual vectors are orthogonal to the column space of X using H as the projection matrix. Intuition: We look to predict y using some linear combination of X, so the closest guess will be the points perpendicular to y and the possible fitting space (column space of X). Note, if X is not linearly independent, implying X is not full rank, then $X^T X$ is singular (non-invertable) and β is not uniquely defined.

We now make some stronger assumptions over the true distribution of data to get some sampling properties of β. Assume observations y_i are uncorrelated and have constant variance σ^2 and that x_i are fixed (non-random). We can derive the variance-covariance matrix of β as follows.

$$Var(\hat{\beta}) = (X^T X)^{-1} \sigma^2 \tag{3.1.1}$$

The formula generally states that the variance of β is inversely correlated to the variance of X. Note that $(X^T X)^{-1}$ is the inverse of the variance-covariance matrix of predictions. It is usually also referred to as the precision matrix and provides information regarding how tightly clustered values are around their mean.

We can estimate σ^2 as follows

$$\hat{\sigma}^2 = \frac{1}{N - p - 1} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{3.1.2}$$

Where N denotes the number of samples and p the number of dimensions. Note the $N - p - 1$ as the divisor for the sample variance. This is because the mean must be 0 for all p dimensions of the input space and therefore, the degrees of freedom the sample set can take on is only $N - p - 1$ since after those number of points are selected, each point after must reset the mean of at least one predictor back to 0 and can therefore be solved for. This scaling factor makes $\hat{\sigma}^2$ an unbiased estimator of σ^2, which implies $\mathbb{E}[\hat{\sigma^2}] = \sigma^2$

Now, to make inferences regarding the parameters of the model, we assume that

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j + \epsilon \tag{3.1.3}$$

or put another way, that the conditional expectation of Y is linear in $X_1 \dots X_p$ and that the deviations around Y and its expectation are additive (independent) and Gaussian, meaning $\epsilon \sim N(0, \sigma^2)$. From the above assumptions we can see that

$$\begin{aligned} \hat{\beta} &\sim N(\beta, (X^T X)^{-1}\sigma^2 \\ (N-p-1)\hat{\sigma}^2 &\sim \sigma^2 \chi^2_{N-p-1} \end{aligned} \tag{3.1.4}$$

Note that for the distribution of $\hat{\sigma}^2$, we assume that $N-p-1$ samples - the rest of the samples are constrained since the expected value of the residuals is 0 for all p predictors - come from a normal distribution since we assume the residuals are normally distributed. By definition, a chi square distribution with d degrees of freedom represents sampling d normal distributions and squaring their values. Thus, the sampling distribution for the squared residuals $(y_i - \hat{y}_i)^2$ will be χ^2_{N-p-1}

To test that a coefficient β_j is significant ($H_0 : \beta_j = 0$), we form the $Z - score$

$$z_j = \frac{\hat{\beta}_j}{\sigma\sqrt{v_j}} \tag{3.1.5}$$

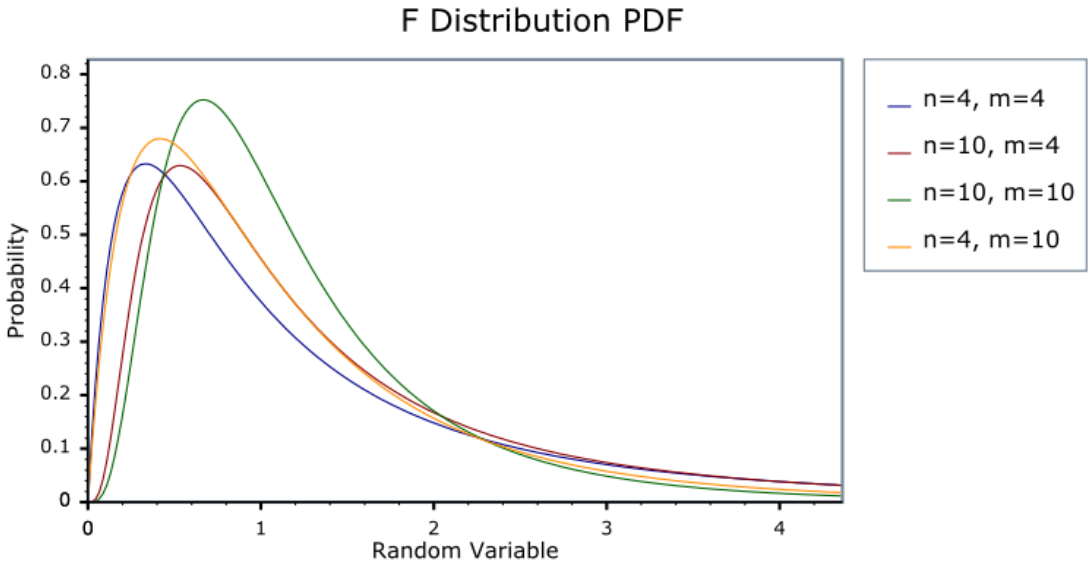
where v_j is the inverse variance of the jth predictor - jth diagonal element of $(X^T X)^{-1}$ Additionally, z_j is distributed according to a $t-$distribution with $N-p-1$ degrees of freedom.

3.1.1 Coefficient Significance Testing

To test for significance in a group coefficients - test whether the coefficients can all be set to zero - we can use an F statistic. (More on page 48)

$$F = \frac{\frac{RSS_0 - RSS_1}{p_1 - p_0}}{\frac{RSS_1}{N - p_1 - 1}} \tag{3.1.6}$$

where RSS_1 is the residual sum-of-squares for the least squares fit of the bigger model with p_1+1 parameters and RSS_0 is the same for the smaller model - using a subset of features - with p_0+1 parameters. Therefore, the test measures change in residual sum-of-squares per additional parameter and is normalized by an estimate of σ^2 of a model that has $p_1 - p_0$ parameters constrained to zero. Under Gaussian assumptions and assuming H_0 - all $p_1 - p_0$ parameters can be set to 0 - the F statistic will have a $F_{p_1-p_0, N-p_1-1}$ distribution. Note that z_j os equivalent to the F statistic for n F distribution is shown below for reference.



An F distribution is used to test for the equality of variances from two normal populations. It is described by the two degrees of freedom for n and m of the two compared populations and models the distribution of

$$F = \frac{\frac{X_1}{\theta_1}}{\frac{X_2}{\theta_2}} \qquad X_1 \sim \chi^2(\theta_1) \quad X_2 \sim \chi^2(\theta_2) \tag{3.1.7}$$

X_1 is a chi-squared random variable with degrees of freedom θ_1 and X_2 is a chi-squared random variable with degrees of freedom θ_2

Using these sampling properties, we can form a $1 - 2\alpha$ confidence interval for the predictor β_j using the above distribution as follows.

$$(\hat{\beta}_j - z^{1-\alpha} v_j^{\frac{1}{2}} \hat{\sigma}, \hat{\beta}_j - z^{1+\alpha} v_j^{\frac{1}{2}} \hat{\sigma}) \tag{3.1.8}$$

Note that $z^{1-\alpha}$ of the 1 - α percentile of the normal distribution.

3.1.2 Gauss-Markov Theorem

Least square estimates of β will have the smallest variance amongst all linear unbiased estimators. However, there may be a biased estimator with smaller mean squared error that trades a little bias for a larger reduction in variance.

3.1.3 Multiple Regression from Univariate Regression

Recall for least squares

$$\beta = \frac{\sum_1^N x_i y_i}{\sum_1^N x_i^2} = \frac{< x, y >}{< x, x >} \tag{3.1.9}$$

with the residual $r_i = y_i - x_i \hat{\beta}$

To extend to a data matrix, we can set multiple least square estimates to $\hat{\beta}_j = \frac{<x_j, y>}{<x_j, x_j>}$ - key assumption is linear independence between features.

We orthogonalize the input space $X \to z_0 \dots z_j$ to produce the coefficients $\hat{\beta}_p = \frac{<z_p, y>}{<z_p, z_p>}$ where z_p is the residual from regressing X_j onto the new orthogonal basis $z_0 \dots z_{j-1}$ (successive orthogonalization or Gram-Schmit decomposition). Note that if x_p is correlated with some other predictor x_k, the residual vector z_p will be close to zero. This implies $\hat{\beta}_p$ may become unstable since $< z_p, z_p >$ is the denominator.

$$Var(\hat{\beta}_p) = \frac{\sigma^2}{< z_p, z_p >} = \frac{\sigma^2}{\|z_p\|^2} \tag{3.1.10}$$

This implies that precision with which we can estimate $\hat{\beta}_p$ to depends on the length of the residual vector z_p which itself represent how much of x_p is unexplained by the other x_k.

We can represent the multiple regression algorithm of orthogonalize each input vector into residual vectors as follows.

$$X = Z\Gamma \tag{3.1.11}$$

where Z has columns $z_0 \dots z_{j-1}$ and Γ is the upper triangular matrix.

$$\begin{aligned} X &= ZD^{-1}D\Gamma \\ X &= QR \end{aligned} \tag{3.1.12}$$

where Q is an $N \times (p+1)$ orthogonal matrix and R is an $(p+1) \times (p+1)$ upper trianglular matrix. Together, the QR decomposition represent a convenient orthogonal basis for the column space of X - run-time $\sim O(n^3)$. The QR decomposition is useful in solving linear least squares problems and estimating eigen-values. The least squares solutions are shown below using the QR components (More on page 55)

$$\begin{aligned} \hat{\beta} &= R^{-1}Q^T y \\ \hat{y} &= QQ^T y \end{aligned} \tag{3.1.13}$$

3.2 Subset Selection

The motivation behind subset selection is related to the variability of β being inversely correlated to the variability of the features in X. Therefore, removing low variance features will reduce variance for low amounts of bias.

Best Subset Selection
Find subsets of size k that yield the smallest residual sum (leaps and bounds algorithm). Select k based on desired relationship between bias and variance.

Forward Step Selection Sequentially add parameters to model to improve fit. Use QR decomposition on current fit to find next candidate features. Potentially more bias but less variance than best fit.

Backward Step Selection Start with full model with all parameters and sequentially delete predictors that have least impact on fit. Drop predictors with lowest z score. Only use when $N > p$

Forward-Stagewise Regression Begin with the intercept equal to $\bar{y}$ and all coefficients set to 0. t each step find variable most correlated with residual. Compute linear regression coefficients of the residual and add to coefficients for that variable. The algorithm that's more than p steps and can be infeasible in high dimensions.

3.3 Shrinkage Models

By retaining a subse of predictors, subset selection produces a model that is interpretable and has possibly lower prediction error, however, may exhibit higher variance.

3.3.1 Ridge Regression

Shrinks the regression coefficients by imposing a penalty on the size of regression coefficients. Model then converges by minimizing the penalized residual sum.

$$\hat{\beta}_{ridge} = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{3.3.1}$$

The parameter λ controls the amount of shrinkage of the model. Note that one should scale and normalize inputs since ridge solutions are not equivariant under scaling - features with different magnitudes would be penalized differently. (Penalizing by the sum of squares of the parameters used in neural networks *weight decay*)

Once all features have been scaled and centered (mean of feature is subtracted from all sample inputs) we can use the following learning criterion.

$$RSS(\lambda) = (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta \tag{3.3.2}$$

The ridge regression solutions can also be easily found as follows.

$$\hat{\beta}_{ridge} = (X^T X + \lambda I)^{-1} X^T y \tag{3.3.3}$$

We can also formulate the ridge regression solutions from the singular value decomposition (SVD) of a matrix.
Recall the singular value decomposition of matrix (Book refers to Σ as D page 66)

$$X = U\Sigma V^T \tag{3.3.4}$$

We can trivially formulate the solutions for the least squares criterion.

$$X\hat{\beta}_{ls} = X(X^TX)^{-1}X^Ty = UU^Ty \tag{3.3.5}$$

This can further be extended to discover the ridge regression solutions as shown below

$$X\hat{\beta}_{ridge} = X(X^TX + \lambda I)^{-1}X^Ty = U\Sigma(\Sigma^2 + \lambda I)^{-1}\Sigma U^Ty$$

$$X\hat{\beta}_{ridge} = \sum_{j=1}^{p} u_j \frac{\Sigma_j^2}{\Sigma_j^2 + \lambda} u_j^T y \tag{3.3.6}$$

We can see from the above formulation that ridge regression works to shrink the jth singular value Σ_j by a factor of $\frac{\Sigma_j^2}{\Sigma_j^2+\lambda}$. This means that for smaller values of Σ_j^2, we shrinkage is greater. Consider the following implications.

Recall the sample covariance matrix $S = \frac{X^TX}{N}$ and the following eigenvalue decomposition $X^TX = V\Sigma^2V^T$. The eigenvectors v_j are the principle components directions of X. The first principle component v_1 has the property that $z_1 = Xv_1$ has the largest sampling variance amongst all normalized linear combinations of the columns of X. A generalized sampling variance is shown below

$$Var(z_j) = Var(Xz_j) = \frac{\Sigma_j^2}{N} \tag{3.3.7}$$

Thus, smaller values for Σ_j correspond to directions in the columns space of X with small variance that ridge regression shrinks the most.
Finally, consider the effective degrees of freedom of the ridge regression model.

$$\begin{aligned} df(\lambda) &= tr[X(X^TX + \lambda I)^{-1}X^T] \\ &= tr[H_\lambda] \\ &= \sum_{j=1}^{p} \frac{\Sigma_j^2}{\Sigma_j^2 + \lambda} \end{aligned} \tag{3.3.8}$$

Recall a degree p polynomial has by definition p degrees of freedom. Ridge regression fit would also contain p coefficients, however depending on the restrictive nature of λ on the fitting process, the degrees of freedom will drop according to the above equation.

3.3.2 The Lasso (Basis Pursuit)

Consider the following criterion

$$\hat{\beta}_{lasso} = \arg\min_{\beta} \frac{1}{2} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_i j_i \beta_j)^2 + \lambda \sum_{j=1}^{p} \|\beta_j\| \tag{3.3.9}$$

Note the key difference between the lasso and ride regression being the L_1 penalty $\lambda \sum_{j=1}^{p} \|\beta_j\|$ added to the coefficients vector β as opposed to the L_2 penalty ($\sum_{j=1}^{p} \beta_j^2$) used in ridge regression. This L_1 constraint makes the solutions non-linear in y_i while the L_2 constraint - as shown previously - remains linear in y. Computing the lasso implements a kind of continuous subset selection and is solved as a quadratic programming problem. Another key differentiator is that while ridge regression does a proportional shrinkage of the

singular values ($\frac{\sigma_j}{\sigma_j+\lambda}$), the lasso translates each coefficient by a constant factor.

Furthermore, we can generalize the lasso and ridge regression using the following Bayes estimate (pg 72)

$$\beta = \arg\min_{\beta} \sum_{i=1}^{N}(y_i\beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \|\beta_j\|^q \tag{3.3.10}$$

where $\|\beta_j\|^q$ can be considered as the log-prior density of β_j.

3.3.3 Elastic Net Penalty

The Elastic Net Penalty similarly selects variables like lasso and then shrinks together coefficients with correlated predictors - similar to ridge regression.

$$\lambda \sum_{j=1}^{p}(\alpha\beta_j^2 + (1-\alpha)\|\beta_j\|) \tag{3.3.11}$$

Elastic net regression closely estimates L_q penalties ($\alpha = L_q - 1$)with considerable computational advantages.

3.3.4 Least Angle Regression

Least angle regression (pg 73) identifies variables most correlated to the response and moves the coefficients of said variables continuously towards its least square value. When another variabels "catches up" in terms of correlation with the residual, the process is paused and the second variable joins the active set of updated coefficients. The process is continued until all variables are in the model. (Algorithm shown on pg 74)

Let A_k denote the active set of the algorithm at the k_th step. Let β_{A_k} be the coefficient vector for the variables at the step. There will be $k-1$ nonzero values and the one value that recently entered the active set will be set to 0. Given the current residual $r_k = y - X_{A_k}\beta_{A_k}$, the direction all the coefficients will be updated in for this step is as follows

$$\delta_k = (X_{A_k}^T X_{A_k})^{-1} X_{A_k}^T r_k \tag{3.3.12}$$

with the coefficient profiles be updates such that $\beta_{A_k}(\alpha) = \beta_{A_k} + \alpha \cdot \delta_k$. This works to keep the coefficients tied and decreasing as desired. The fit vector $\hat{f}_k(\alpha) = \hat{f}_k + \alpha \cdot \mu_k$ where $\mu_k = X_{A_k}\delta_k$

Looking at the geometric representation of μ_k, we can see that it makes the smallest and equal angle with each of the predictors in A_k.

The LAR algorithm is incredibly efficient, requiring the same number of steps as single least squares fit. Additionally, it is important to note that the lasso and LAR are identical when the sign of β_j matches the sign of the inner product. Put in other terms, this means LAR and lasso differ when an active coefficient passes through zero. (More on pg 76)

Finally, when finding the degrees of freedom for LAR or lasso, we can use the following formulation.

$$df(\hat{y}) = \frac{1}{\sigma^2}\sum_{i=1}^{N} Cov(\hat{y}_i, y_i) \tag{3.3.13}$$

where $\hat{y}$ represents the fitted vector and $Cov(\hat{y}_i, y_i)$ refers to the sample - response covariance

3.4 Derived Input Directions

In situations with large numbers of inputs that are often correlated, it becomes useful to work in a smaller representation space with linear combinations of the original inputs.

3.4.1 Principle Components Regression

Form the derived input columns z_m by regressing X_{v_m} onto y

4 Linear Methods for Classification

In this chapter we investigate classification problems where decision boundaries are linear. That is, we have a predictor $G(x)$ that divides the input space into a discrete collection of regions. For many problems, the decision boundaries between these regions are linear, and this is what is meant by linear methods for classification.

All we require is that some monotone transformation of $\Pr(G = k|X = x)$ be linear for the decision boundaries to be linear. For example, if there are two classes, a popular model for the posterior probabilities (see 3.3) is

$$Pr(G = 0|X = k) = \frac{\exp(\beta^T x)}{1 + \exp(\beta^T x)}$$
$$Pr(G = 1|X = k) = \frac{1}{1 + \exp(\beta^T x)}$$

Here the monotone transformation is the logit transformation: $\log[\frac{p}{1-p}]$ and we see that

$$\log \frac{Pr(G = 0|X = k)}{Pr(G = 1|X = k)} = \beta^T x$$

which implies the decision boundary is the set of points where the *log-odds are zero*. In other words, no one class is preferred more than the other, which is how we'd want to model a decision boundary.

Two methods that result in log-odds (logits):

- Linear Discriminant Analysis (LDA)
- Logistic Regression

A more direct approach is to explicitly model the boundaries between the classes as linear - finding a separating hyperplane. Some approaches are:

- Perceptron
- SVM

4.1 Linear Regression of an Indicator Matrix

Here each of the response categories are coded via a one hot encoding/indicator variable. The fit is

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

Here we view the regression as an estimate of conditional expectation. For the random variable Y_k, $E(Y_k|X = x) = Pr(G = k|X = x)$, so conditional expectation of each of the Y_k seems a sensible goal.

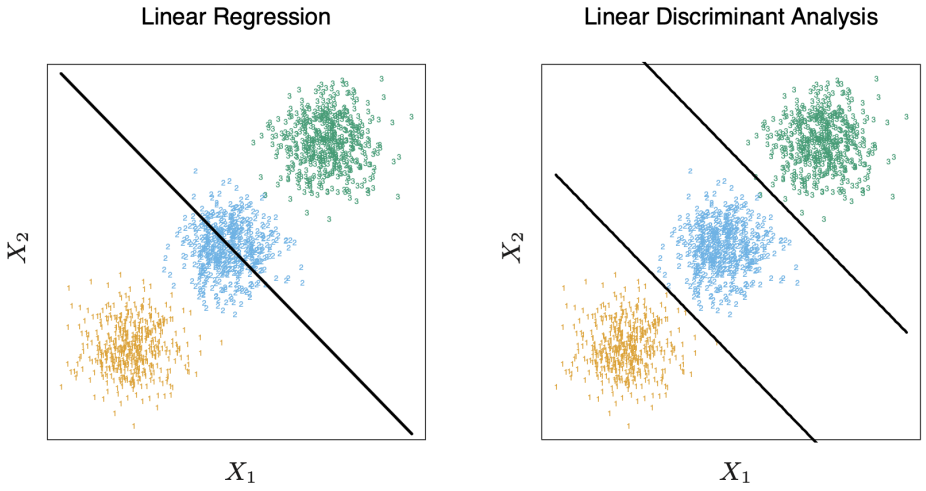


FIGURE 4.2. *The data come from three classes in* $\mathbb{R}^2$ *and are easily separated by linear decision boundaries. The right plot shows the boundaries found by linear discriminant analysis. The left plot shows the boundaries found by linear regression of the indicator response variables. The middle class is completely masked (never dominates).*

However, linear regression can *mask* certain classes when $K \geq 3$ - without polynomial terms (quadratic regression), simple linear regression cannot separate classes. A loose but general rule is that if $K \geq 3$ classes are lined up, polynomial terms up to degree $K - 1$ might be needed to resolve them.

4.2 Linear Discriminant Analysis (LDA)

LDA is a technique used best with continuous predictors that models the class-conditional density of X - if we assume a prior distribution over the classes, we can simply apply Bayes rule with the learned class-conditional density from LDA to model the posterior distribution. In LDA we typically model each class density as multivariate Gaussian:

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}$$

LDA arises when we have a common covariance matrix $\Sigma_k = \Sigma \forall k$. We also maintain a prior probability for each class given by π_k - then we can apply Bayes rule to get

$$Pr(G = k|X = x) = \frac{\pi_k f_k(x)}{\sum_j \pi_j f_j(x)}$$

Note that when comparing two classes, it is sufficient to look at the log ratio $\log \frac{Pr(G=k|X=x)}{Pr(G=l|X=x)}$ which results in an equation linear in x. This implies the decision boundary between classes k and l is a hyperplane in p-dimensional space (or equivalently linear in x).

From this we can derive the discriminant function. We have

$$\hat{G}(x) = \arg\max_k Pr(G = k|X = x)$$
$$\hat{G}(x) = \arg\max_k \pi_k f_k(x)$$

Interpreting as a likelihood function, let $\delta_k(x) = \log \pi_k f_k(x)$ - then the predicted class is simply $\hat{G}(x) = \arg\max_k \delta_k(x)$.

$$\begin{aligned}
\delta_k(x) &= \log \pi_k + \log f_k(x) \\
&= \log \pi_k - \log((2\pi)^{p/2}|\Sigma|^{1/2}) - \frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) \\
&= \log \pi_k - \frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) && \text{(no k term)} \\
&= \log \pi_k - \frac{1}{2}(x^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_k - \mu_k^T\Sigma^{-1}x + \mu_k^T\Sigma^{-1}\mu_k) \\
&= \log \pi_k - \frac{1}{2}(x^T\Sigma^{-1}x - 2x^T\Sigma^{-1}\mu_k + \mu_k^T\Sigma^{-1}\mu_k) \\
&= \log \pi_k - \frac{1}{2}(-2x^T\Sigma^{-1}\mu_k + \mu_k^T\Sigma^{-1}\mu_k) && \text{(no k term)} \\
&= \log \pi_k + x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k
\end{aligned}$$

In practice, we do not know the parameters of the Gaussian distributions. We estimate them using our train data.

- $\hat{\pi_k} = N_k/N$ where N_k is the number of class k observations
- $\hat{\mu_k} = \sum_{g_i=k} x_i/N_k$
- $\hat{\Sigma} = \sum_{k=1}^{K}\sum_{g_i=k}(x_i - \hat{\mu_k})(x_i - \hat{\mu_k})^T/(N-K)$

If the Σ_k are not equal, then the terms quadratic in x remain and we get *quadratic discriminant analysis* (QDA). The estimates for QDA are similar to those for LDA, except that separate covariance matrices must be estimated for each class.

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log \pi_k$$

Computations for LDA can be achieved by diagonalizing $\hat{\Sigma}$ or $\hat{\Sigma_k}$ which can be achieved through an eigen-decomposition of the matrices.

4.2.1 Regularized Discriminant Analysis

Regularized Discriminant Analysis allows one to shrink the separate covariances of QDA toward a common covariance as in LDA. The regularized covariance matrices are simply

$$\hat{\Sigma_k} = \alpha\hat{\Sigma_k} + (1-\alpha)\hat{\Sigma}$$

where $\hat{\Sigma}$ is the pooled covariance matrix as used in LDA.

4.2.2 Reduced-Rank Linear Discriminant Analysis

Part of why LDA is so popular is that it embeds an informative low-dimensional projection of the data. The K centroids in the p-dimensional input space lie in an affine subspace of dimension $\leq K-1$, which is nice when $p >> K$. Also, when classifying a new point to its closest centroid, notice that we can ignore all distances perpendicular to the subspace spanned by the K centroids - this is because perpendicular distances are equally relevant to all groups and do not help us differentiate between them. Thus we might just as well project the data onto this centroid-spanning subspace H_{k-1}, and make distance comparisons there.

Fisher's idea was that if we wanted to find one direction, good classification should be obtained based on the projected data. His idea was to maximize the ratio of the between-class variance and the within- class variance. Roughly speaking, the "spread" of the centroids of every class is maximized relative to the "spread"

of the data within class.

Consider projecting our data onto some vector a, given by $a^T X$. Note the variance of our projected data is given by $a^T W a$ where W is the common within-class covariance matrix of X (equivalent to Σ in LDA).

We also need a between-class covariance matrix denoted by B. This is the covariance matrix you compute using only the mean vectors.

$$\mu = \sum_{k=1}^{K} \pi_k \mu_k \qquad (\mu_k \text{ is mean vector of class K})$$
$$B = \sum_{k=1}^{K} \pi_k (\mu_k - \mu)(\mu_k - \mu)^T$$

Therefore the within-class variance is $a^T W a$ and the between-class variance is $a^T B a$. The optimization becomes

$$\max_a \frac{a^T B a}{a^T W a}$$
$$\max_a \frac{a^T W^{-1} B a}{a^T a}$$

a is given by the largest eigenvector of $W^{-1}B$. Similarly you can find the second, third, fourth largest eigenvectors to obtain $a_1, a_2, a_3, a_4, \ldots$ for each input. This defines the projection to a lower dimensional space while maximizing between-class variance relative to within-class variance.

4.3 Logistic Regression

In logistic regression, we directly model the posterior distribution as linear functions of x.

4.3.1 Binary Logistic Regression

To motivate this, consider simple linear regression on a dataset where the target Y is either 0 or 1. We want to estimate the probability of success (1), given by p. Linear regression works well with continuous values, but in this case we need a *link function* to map our targets Y to the outputs of a linear model. We can apply the logit (log of the odds ratio) transform to map values in $(0, 1)$ to $(-\inf, \inf)$ as desired.

$$\begin{aligned}
\text{logit}(E[Y]) &= \text{logit}(p) \\
&= \log \frac{p}{1-p} \\
&= \log \frac{Pr(G=1|X=x)}{Pr(G=0|X=x)} \\
&= \beta_0 + \beta^T x \qquad \text{(def link function)}
\end{aligned}$$

Taking the inverse we find

$$\frac{p}{1-p} = e^{\beta_0 + \beta^T x}$$
$$p = \frac{e^{\beta_0 + \beta^T x}}{1 + e^{\beta_0 + \beta^T x}}$$

where $p = E[Y] = Pr(G = 1|X = x)$. This is how the *sigmoid* transformation used in logistic regression is derived. Therefore, by learning β_0, β and applying sigmoid we can classify binary targets.

4.4 Multiclass Logistic Regression

In multiclass logistic regression, we have $K-1$ logit transformations. From this we find

$$Pr(G=k|X=x) = \frac{\exp \beta_{k0} + \beta_k^T x}{1 + \sum_{l=0}^{K-1} \exp \beta_{l0} + \beta_l^T x}$$
$$Pr(G=K|X=x) = \frac{1}{1 + \sum_{l=0}^{K-1} \exp \beta_{l0} + \beta_l^T x}$$

Denote $Pr(G=k|X=x)$ as $p_k(x;\theta)$ where θ is the set of all parameters.

Logistic regression is usually fit by maximum likelihood, where the log-likelihood of N observations is

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

which can be maximized via Newton-Rhapson or gradient descent.

4.5 Logistic Regression vs. LDA

Both logistic regression and LDA have linear logits, or linear functions of x in their log-odds. However, the difference lies in the way the linear coefficients are estimated. The logistic regression model is more general, in that it makes less assumptions.

We can write the joint density of X and G as

$$Pr(X, G=k) = Pr(X)Pr(G=k|X)$$

for both LDA and logistic regression, $Pr(G=k|X)$ is the same (just sigmoid/softmax). However, logstic regression leaves $Pr(X)$ as an arbitrary density function, and fits $Pr(G|X)$ by maximizing *conditional likelihood*. With LDA we fit the parameters by maximizing the full log-likelihood, based on the joint density

$$Pr(X, G=k) = \phi(X;\mu_k,\Sigma)\pi_k$$

where ϕ is multivariate Gaussian. Unlike logistic regression, $Pr(X)$ plays a role here - it is a mixture density

$$Pr(X) = \sum_{k=1}^{K} \phi(X;\mu_k,\Sigma)\pi_k$$

which involves the parameters we are optimizing over. Note since the parameters are from Gaussian distributions, we can get MLE values by plugging in corresponding estimates $\hat{\pi_k}, \hat{\mu_k}, \hat{\Sigma}$ from standard normal theory.

5 Kernel Smoothing Methods

We can learn complex regression function in $\mathbb{R}^p$ by fitting more simple, different models totarget points x_0 using observations close to x_0 and finding the estimated functions $\hat{f}(x)$ that is smooth in $\mathbb{R}^p$. For this task, we can leverage kernel function $K_\lambda(x_0, x_i)$ that weigh x_i based on proximity to x_0.

5.1 One Dimensional Kernel Functions

Recall k nearest neighbors

$$\hat{f}(x) = Ave(y_i|x_i \in N_k(x)) \tag{5.1.1}$$

This results in lots of jitter since there are discrete steps when points fall in and out of the neighborhoods. To get a smooth/differentiable functions, we assign weights that die smoothly with distance.

Consider the Nadaraya-Watson kernel weighted average.

$$\hat{f}(x_0) = \frac{\sum_{i=1}^{N} K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)} \tag{5.1.2}$$

with the Epanechnikov quadratic kernel

$$K_\lambda(x_0, x) = D(\frac{|x - x_0}{\lambda}) \tag{5.1.3}$$

with

$$D(t) = \begin{cases} \frac{3}{4}(1 - t^2) & \text{if } |t| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.1.4}$$

We can more generally define $h_\lambda(x_0)$ to be a width function that determine to width of the neighborhood at x_0. The implementations related to this class of functions is discussed more on pages 193 and 194.

5.1.1 Local Linear Regression

Locally weighted averages can be badly biased on the boundaries of the domain because of the asymmetry of the kernel. To combat this, we can fit straight lines rather than constants locally to remove this bias to first order. This bias that is present in most standard kernels is complex to modify and is only approximate for finite sample sizes. Local linear regression automatically modifies the kernel to correct the bias exactly to first order (automatic kernel carpentry). This phenomenon is discussed further on pages 195 and 196. Local linear regression tends to be biased in regions of curvature of the true function, leading to a phenomenon referred to as trimming the hills and filling the valleys. Local quadratic regression is generally able to correct this bias at the expense of increased variance.

5.1.2 Local Polynomial Regression

We can also locally fit polynomials of arbitrary degree d and constrain the bias to only having components of degree $d + 1$ and higher. Assuming the model $y_i = f(x_i) + \epsilon_i$ where ϵ_i is independently and identically distributed with mean zero and variance σ^2, $Var(\hat{f}(x_0)) = \sigma^2 \|l(x_0)\|^2$ where $l(x_0)$ is the vector of equivalent kernel weights at x_0. It can be shown that $\|l(x_0)\|$ increases with d and so there is a bias variance tradeoff in selecting the polynomial degree. More takeaways/discussion about local regression is shared on page 198.

5.2 Selecting the Width of the Kernel

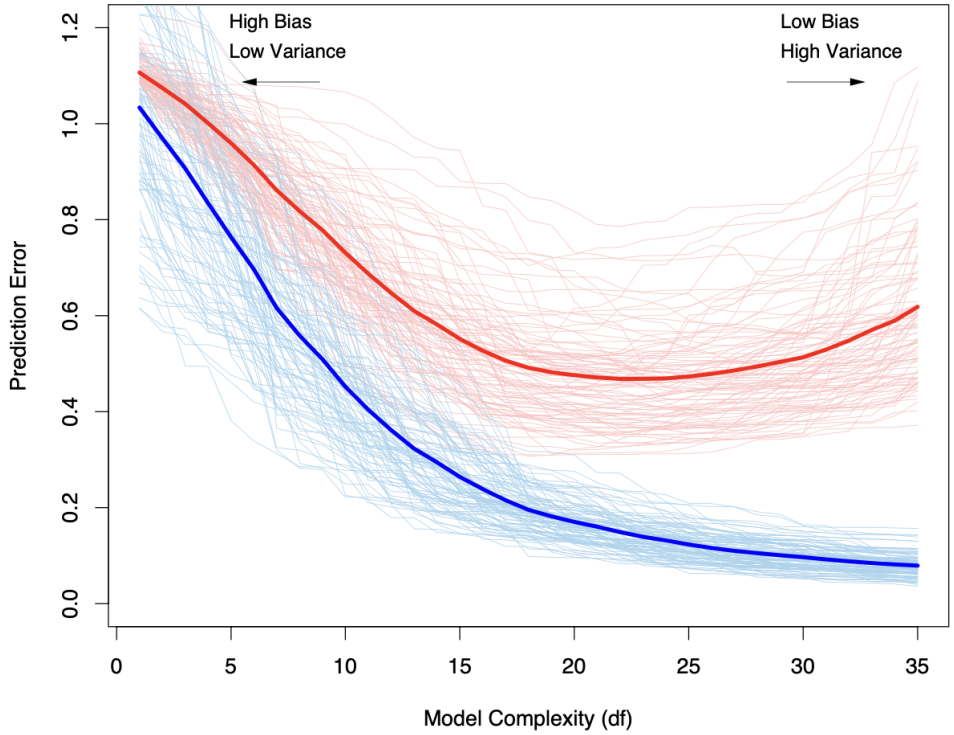
For each of the kernels K_λ, the λ parameter controls its width, The smooth matrix

6 Model Assessment and Selection

The generalization performance is a model's prediction capability on independent test data. Assessment of this performance is extremely important in practice, since it guides the choice of learning method or model, and gives us a measure of the quality of the ultimately chosen model.

6.1 Bias Variance Tradeoff

The bias-variance tradeoff describes a model's tendency to overfit to training data as it becomes more complex, thereby harming test error.



Consider the example of $Y = f(X) + \epsilon$ where $E(\epsilon) = 0$ and $Var(\epsilon) = \sigma_\epsilon^2$. We can derive the expected prediction error of a regression fit $\hat{f}(X)$ at an input point x_0 using squared error loss

$$\begin{aligned} Err(x_0) &= E[(f(x_0) - \hat{f}(x_0)^2] \\ &= \sigma_\epsilon^2 + [E\hat{f}(x_0) - f(x_0)]^2 + E[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\ &= \sigma_\epsilon^2 + \text{Bias}^2 + \text{Variance} \end{aligned}$$

7 Model Inference and Averaging

7.1