Lecture 3: Linear methods for classification

Rafael A. Irizarry and Hector Corrada Bravo

February, 2010

Today we describe four specific algorithms useful for classification problems: linear regression, linear discriminant analysis, logistic regression and separating hyperplanes.

Introduction

We now revisit the classification problem and concentrate on linear methods.

Recall that our setting is that we observe for subject i predictors (covariates) x_i, and *qualitative* outcomes (or classes) g_i, which can takes values from a discrete set G.

Since our prediction $\hat{G}(x)$ will always take values in the discrete set G, we can always divide the input space into a collection of regions taking the same predicted values.

The question is: what is the best sub-division of this space?

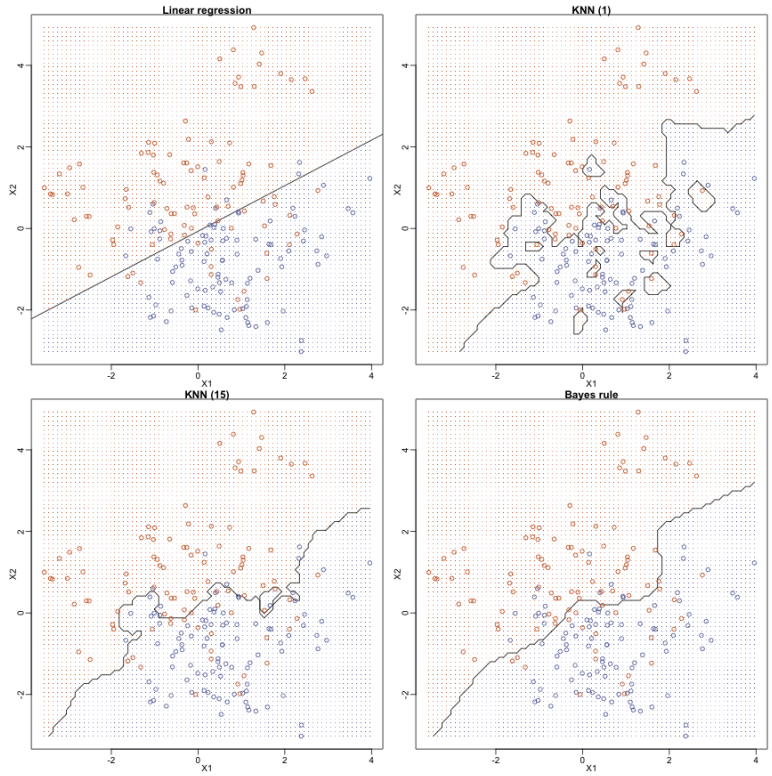
We saw previously that the boundaries can be smooth or rough depending on the prediction function.

For an important class of procedures, these *decision boundaries* are linear. This is what we will refer to as linear methods for classification. We will see that these can be quite flexible (much more than linear regression).

Suppose we have K classes labeled $1, \ldots, K$ and a single predictior X. We can define a $0-1$ indicator for each class k, and perform regression for each of these. We would end up with a regression function $f_k(x) = \hat{\beta}_{0k} + \hat{\beta}_{1k}x$ for each class k.

The decision boundary between class k and l is simply the set for which $\hat{f}_k(x) = \hat{f}_l(x)$, i.e., $\{x : \hat{\beta}_{0k} + \hat{\beta}_{1k}x = \hat{\beta}_{0l} + \hat{\beta}_{1l}x\}$ which is a hyperplane.

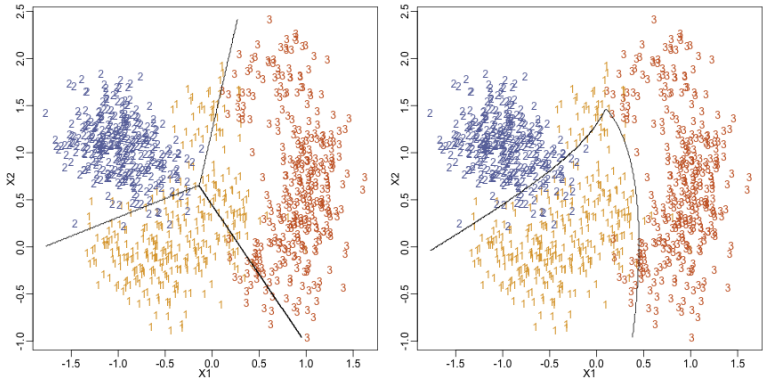
Since the same holds for any pair of classes, the division of the input space is piecewise linear.


Linear regression
KNN (1)
KNN (15)
Bayes rule
X1
X2

This regression approach is one of a number of methods that model a *discriminant function* $\delta_k(x)$ for each class, and classifies X to the class with the largest value of the discriminant function.

Methods that model the posterior probability $Pr(G = k|X = x)$ are also in this class. If this is a linear function of x then we say it is linear. Furthermore, if it is a monotone transform of a linear function of x, we will sometimes say it is linear. An example is *logistic regression*. More on that later.

Both decision boundaries shown in the next Figure are linear: one obtained with linear regression, the other using quadratic terms.



Linear regression of an indicator matrix

Each response is coded as a vector of 0–1 entries. If G has K classes, then Y_k for $k = 1, \ldots, K$ is such that $Y_k = 1$, if $G = k$ and 0 otherwise.

These are collected in a vector $Y = (Y_1, \ldots, Y_k)$ and the N training vectors are collected in an $N \times K$ matrix, we denote by $\mathbf{Y}$.

For example, if we have $K = 5$ classes

$$\begin{array}{cc} 1 & \\ 2 & \\ 3 & \rightarrow \\ 4 & \\ 5 & \end{array} \quad \begin{array}{ccccc} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{array}$$

On the left is the original data, and on the right, the coded data.

To fit with regression to each class simulatneously, we can simply use the same matrix multiplication trick

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

Notice the matrix

$$\hat{\mathbf{B}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

has the coefficients for each regression in the columns. So, for any point x we can get the prediction by

- compute the fitted output $\hat{f}(x) = [(1, x)\hat{\mathbf{B}}]$, a K vector
- identify the largest component, and classify accordingly

$$\hat{G}(x) = \arg\max_{k \in G} \hat{f}_k(x)$$

What is the rationale for this approach?

Recall the *expected prediction error* we discussed last time. What do we do for qualitative data? Because the elements of G are not numbers, taking expectations doesn't mean anything. However, we can define a *loss function*. Say we have three elements $G = \{A, B, C\}$, we can define the loss function like this:

$$L(\hat{G}, G) = \begin{cases} 0 & \text{if } \hat{G} = G \\ 1 & \text{if } \hat{G} \neq G \end{cases}$$

This can be thought of as a distance matrix with 0s in the diagonals and 1s everywhere else.

Notice that in some applications this loss function may not be appropriate. For instance, saying someone has cancer when they don't is not as bad as saying they don't have cancer when they actually do. For now, let's stick to 1 and 0.

We can now write

$$EPE = E_X \sum_{k=1}^{K} L(G_k, \hat{G}(X)) Pr(G = k|X)$$

The minimizer of EPE is known as the *Bayes classifier*, or *Bayes decision rule*.

$$\hat{G}(X) = \max_{g \in G} Pr(g|X = x)$$

So, why don't we use it? Typically we don't know $Pr(g|X = x)$, just like in the regression setting we don't know $f(x) = E[Y|X = x]$.

Note: If we simulate data like we do below, then we can compute the Bayes decision rule since we know $Pr(g|X = x)$.

Note: If we use the encoding 0/1 encoding above for the two-class case, then we see the relationship between the Bayes classifier and the regression function since $Pr(G = g|X = x) = E(Y|X = x)$ in this case.

The real issue here is, how good is the linear approximation here? Alternatively, are the $\hat{f}_k(x)$ good estimates of $Pr(G = k|X = x)$.

We know they are not great since we know $\hat{f}(x)$ can be greater than 1 or less than 0. However, as we have discussed, this may not matter if we get good predictions.

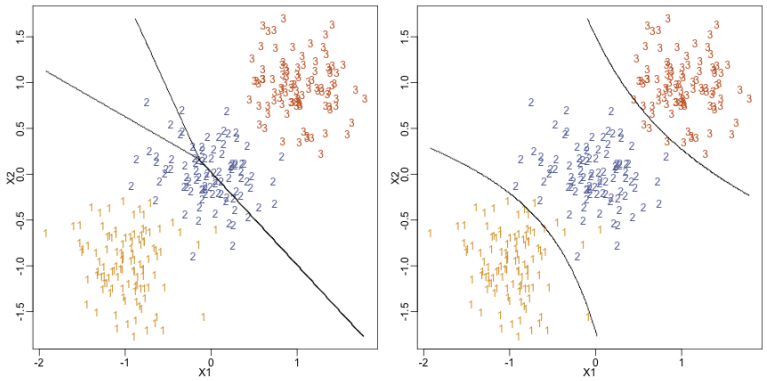
A more conventional way of fitting this model is by defining target t_k as the vector with a 1 in the kth entry and 0 otherwise, such that $y_i = t_k$ if $g_i = k$. Then the above approach is equivalent to minimizing

$$\min_{\mathbf{B}} \sum_{i=1}^{N} \|y_i - \{(1, x_i)\mathbf{B}\}'\|^2.$$

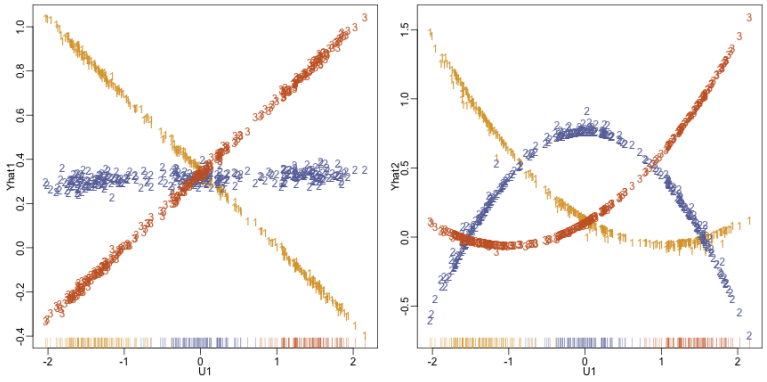
A new observation is classified by computing $\hat{f}(x)$ and choosing the closes target

$$\arg\min_k \|\hat{f}(x) - t_k\|^2$$

Because of the rigid nature of linear regression, a problem arises for linear regression with $K >= 3$. The next figure shows an extreme situation. Notice that decision boundaries can be formed by eye to perfectly discriminate the three classes. However, the decision boundaries from linear regression do not do well.



Why does linear regression miss the classification in this case? Notice that the best direction to separate these data is a line going through the centroids of the data. Notice there is no information in the projection orthogonal to this one.



If we then regress Y on the transformed X, then there is barely any information about the second class. This is seen clearly in the left panel of the next Figure.

However, by making the regression function a bit more flexible, we can do a bit better. One way to do this is to use quadratic terms (there are 3, what are they?) In the last two Figures, this linear regression version including quadratic terms does much better.

However, if we increase the number of classes to $K = 4$ we would then need to start adding the cubic terms and now are dealing with lots of variables.

A data set we may be working on later will be vowel sound data. The next Figure contains a plot of the first 2 coordinates and the classes.

Linear Discriminant Analysis

Decision theory tells us that we should know the class posteriors $Pr(G|X = x)$ for optimal classification.

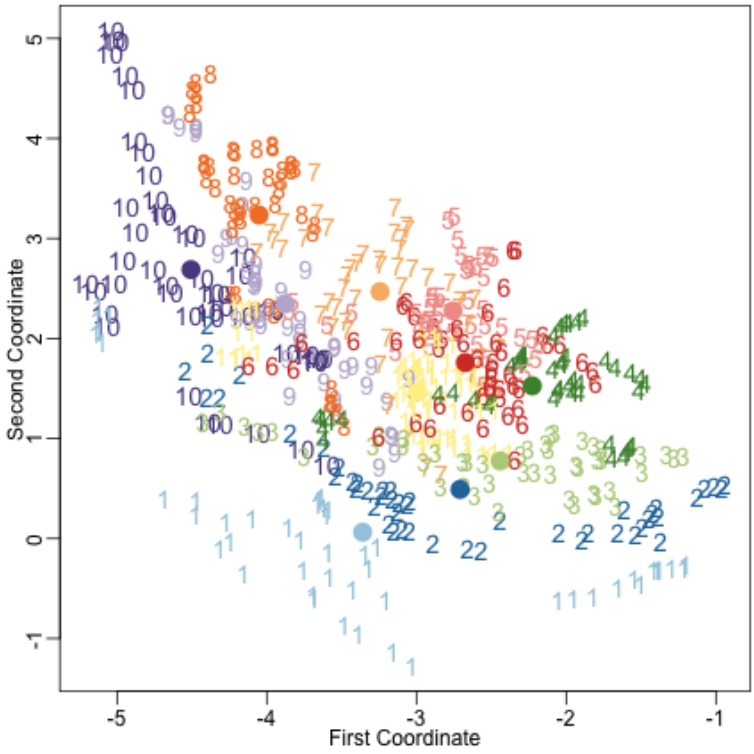
Suppose $f_k(x)$ is the class conditional density of X in class $G = k$, and let π_k be the prior probability of class k, with $\sum_{k=1}^{K} \pi_k = 1$. A simple application of Bayes' theorem gives us

$$Pr(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$$

Notice that havign quantities $f_k(x)$ is almost equivalent to having the $Pr(G = k|X = x)$ provided by Bayes' rule.

Suppose we model each class conditional density as multivariate Gaussian:

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma_k}^{-1}\|^{1/2}} \exp\{-\frac{1}{2}(x - \mu_k)'\mathbf{\Sigma_k}^{-1}(x - \mu_k).$$


Second Coordinate
First Coordinate
5
4
3
2
1
0
-1
-5
-4
-3
-2
-1

The next Figure shows regions that contain 95% of the data for three bivariate distributions with different means, but the same covariance structure. The covariance structure make these ellipses rather than circles.

The lines are the Bayes decision rules.

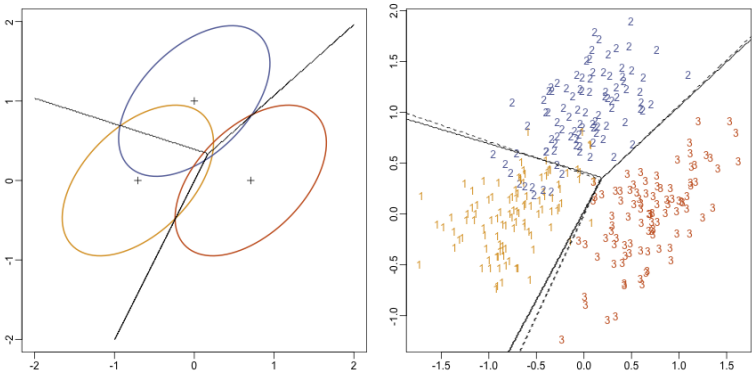
Linear Discriminant Analysis (LDA) arises when we assume that the covariance is the same for all classes. In this case, we see that discriminant functions are simply

$$\delta_k(x) = x'\mathbf{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k\mathbf{\Sigma}^{-1}\mu_k + \log \pi_k$$

Notice: if we assume $\pi_k = 1/K$ then the last term is not needed. In any case, notice this is a linear function of x!.

In practice, we do not have the means μ_k or the covariance structure $\mathbf{\Sigma}$. The strategy is to use training data to estimate these.

In the next figure we see some outcomes in a three class simulation. The distributions used to create them are those shown on the left panel. The Bayes decision rules are shown dashed and the estimated LDA discrimant functions are shown as solid lines.



To estimate the parameters we simply:

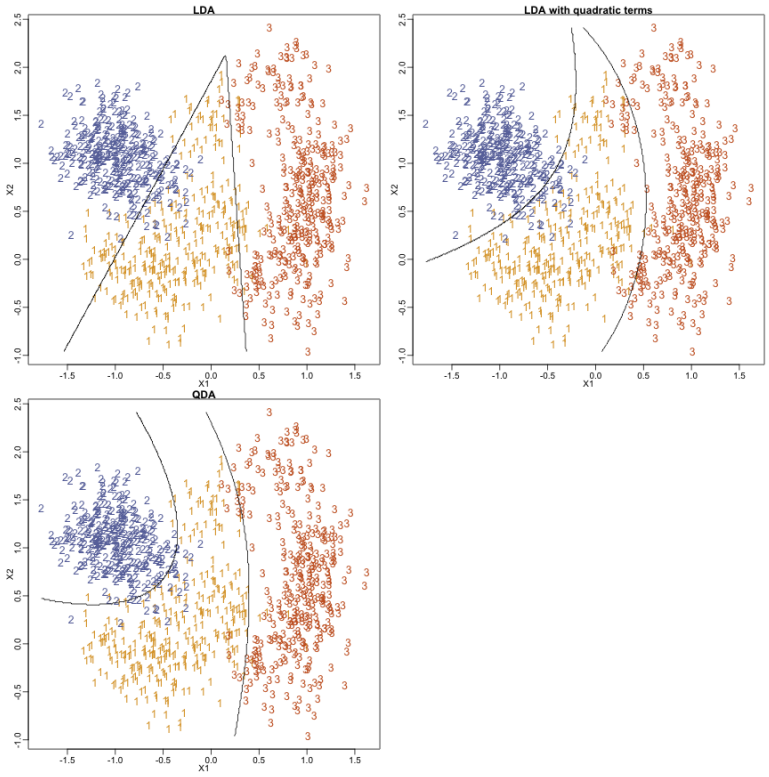
- $\hat{\pi}_k = \frac{N_k}{N}$, where N_k is the observed number of subjects in class k
- $\hat{\mu}_k = \frac{1}{N_k}\sum_{g_i=k} x_i$
- $\hat{\mathbf{\Sigma}} = \frac{1}{N-K}\sum_{k=1}^{K}\sum_{g_i=k}(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)'$

Technical note: for two classes LDA is almost the same as regression, the coefficients of each are proportional, but unless $N_1 = N_2$ the intercepts are different.

If we assume that each class has its own correlation structure, the discriminant functions are no longer linear. Instead, we get

$$\delta_k(x) = -\frac{1}{2}\log|\mathbf{\Sigma}_k|^{-1} - \frac{1}{2}(x-\mu_k)'\mathbf{\Sigma}_k^{-1}(x-\mu_k)$$

The decision boundary is now described with a quadratic function. This is therefore called *quadratic discriminant analysis* (QDA). Next we plot LDA and QDA decision boundaries for the same data.



Note: when the number of covariates grow, the number of things to estimate in the covariance matrix gets very large. One needs to be careful.

Computations for LDA and QDA

Suppose we compute the eigen-decomposition of each $\hat{\mathbf{\Sigma}}_k = \mathbf{U}_k\mathbf{D}_k\mathbf{U}_k'$, where $\mathbf{U}_k$ is a $p \times p$ matrix with orthonormal columns and $\mathbf{D}_k$ is a diagonal matrix of positive eigenvalues d_{kl}. The ingredients for $\delta_k(x)$ are:

- $(x-\hat{\mu}_k)'\mathbf{\Sigma}_k^{-1}(x-\hat{\mu}_k) = [\mathbf{U}_k'(x-\hat{\mu}_k)']\mathbf{D}_k^{-1}[(x-\hat{\mu}_k)\mathbf{U}_k]$

- $\log |\mathbf{\Sigma}_k|^{-1} = \sum_l \log d_{kl}$

Notice this is much easier to compute since $\mathbf{D}_k$ is a diagonal matrix!

Given this, we can now compute and interpret the LDA classifier as follows:

- *Sphere* the data with respect to the common covariance estimate $\hat{\mathbf{\Sigma}}$ to get $X^* = \mathbf{D}^{1/2}\mathbf{U}'X$. The common covariance estimate of X^* is now the identity matrix!
- Classify to the closes class centroid (μ_ks) in the transformed space, correcting for the effect of the class prior probabilities π_k.

Section 4.3.3 of Hastie, Tibshirani and Friedman, has a nice discussion of how LDA is related to the solution of the problem: *find the linear combination $Z = a'X$ such that the between-class variance is maximized relative to the within-class variance.*

Logistic regression

Assume

$$\begin{aligned}
\log \frac{Pr(G=1|X=x)}{Pr(G=K|X=x)} &= \beta_{10} + \beta_1' x \\
\log \frac{Pr(G=2|X=x)}{Pr(G=K|X=x)} &= \beta_{20} + \beta_2' x \\
&\vdots \\
\log \frac{Pr(G=K-1|X=x)}{Pr(G=K|X=x)} &= \beta_{(K-1)0} + \beta_{K-1}' x.
\end{aligned}$$

Notice $g(p) = \log \frac{p}{1-p}$ is the logistic link and is $g : (0,1) \to \mathbf{R}$.

A simple calculation gives

$$\begin{aligned}
Pr(G=k|X=x) &= \frac{\exp(\beta_{k0} + \beta_k' x)}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l' x)}, k = 1, \ldots, K-1, \\
Pr(G=K|X=x) &= \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l' x)}
\end{aligned}$$

When $K = 2$ this has a very simple form (only one set of covariates) and is a very popular model used in biostatistical applications.

With this probability model we can now write the log-likelihood

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

where $p_{g_i}(x_i; \theta) = Pr(G = k|X = x; \theta)$. In the two-class case, use $y_i = 1$ for $g_i = 1$ and $y_i = 0$ for $g_i = 2$; let $p_1(x_i; \theta) = p(x_i; \theta)$, so $p_2(x_i; \theta) = 1 - p(x_i; \theta)$. The log-likelihood is then

$$\begin{aligned} l(\beta) &= \sum_{i=1}^{N} \{y_i \log p(x_i; \beta) + (1 - y_i) \log(1 - p(x_i; \beta))\} \\ &= \sum_{i=1}^{N} \left\{ y_i \beta' x_i - \log(1 + e^{\beta' x_i}) \right\} \end{aligned}$$

Our estimate of β will be the maximum likelihood estimate (MLE), obtained by maximizing the log-likelihood with respect to β.

We do this by setting the partial derivatives of the log-likelihood to zero:

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i (y_i - p(x_i; \beta)) = 0$$

This results in a nonlinear system of $p + 1$ equations. These are also called the *score equations*. Notice that for the intercept ($x_0 = \mathbf{1}$), its score equation ($\sum_{i=1}^{N} y_i = \sum_{i=1} p(x_i; \beta)$ states that for β to be an MLE solution, the expected number of observations in class 1, must match the observed number of observations.

To solve the set of score equations, we can use the Newton-Raphson method, which starting from an initial guess β_{old} iteratively updates the estimate using:

$$\beta^{\text{new}} \leftarrow \beta^{\text{old}} - \left(\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta'} \right)^{-1} \frac{\partial l(\beta)}{\partial \beta},$$

with derivatives evaluated at β_{old}. The matrix of second derivatives (Hessian matrix) is given by

$$\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta'} = -\sum_{i=1}^{N} x_i x_i' p(x_i; \beta)(1 - p(x_i; \beta)).$$

By writing the gradient and Hessian in matrix notation, we can see a neat by-product of using the Newton method in this case. The gradient and Hessian are given by

$$\frac{\partial l(\beta)}{\partial \beta} = \mathbf{X}'(\mathbf{y} - \mathbf{p})$$
$$\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta'} = -\mathbf{X}'\mathbf{W}\mathbf{X}$$

where vector $\mathbf{y}$ is the vector of y_i values, $\mathbf{X}$ the $N \times (p+1)$ matrix of x_i values, $\mathbf{p}$ the vector of fitted probabilities and $\mathbf{W}$ a $N \times N$ diagonal matrix with ith entry $p(x_i; \beta)(1 - p(x_i; \beta))$.

With this we can rewrite the Newton update as

$$\begin{aligned}\beta^{\text{new}} &= \beta^{\text{old}} + (\mathbf{X}'\mathbf{X}\mathbf{W})^{-1}\mathbf{X}'(\mathbf{y} - \mathbf{p}) \\ &= (\mathbf{X}'\mathbf{X}\mathbf{W})^{-1}\mathbf{X}'\mathbf{W}(\mathbf{X}\beta^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p}))\end{aligned}$$

Introducing notation $\mathbf{z} = \mathbf{X}\beta^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})$, we can see that the Newton update is the solution to a *weighted* least squares problem

$$\beta^{\text{new}} \leftarrow \arg\min_{\beta}(\mathbf{z} - \mathbf{X}\beta)'\mathbf{W}(\mathbf{z} - \mathbf{X}\beta)$$

This connection to least squares also gives us the following:

- The weighted residual-sum-of-squares is the Pearson chi-square statistic

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

a quadratic approximation to the deviance

- Asymptotic likelihood theory says that if the model is correct then $\hat{\beta}$ converges to the true β.
- A CLT then shows that the distribution of $\hat{\beta}$ converges to $N(\beta, (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1})$

Logistic regression versus LDA

Notice logistic regression provides a similar solution to LDA.

Using Bayes' theory we can show that

$$\log \frac{Pr(G = k|X = x)}{Pr(G = K|X = x)} = \log \frac{\pi_k}{\pi_K} - \frac{1}{2}(\mu_k + \mu_K)'\mathbf{\Sigma}^{-1}(\mu_k - \mu_K) + x'\mathbf{\Sigma}^{-1}(\mu_k - \mu_K)$$

which can be re-written as

$$\log \frac{Pr(G = k|X = x)}{Pr(G = K|X = x)} = \alpha_{0k} + \alpha_k' x.$$

For logistic regression, we explicitly write

$$\log \frac{Pr(G = k|X = x)}{Pr(G = K|X = x)} = \beta_{0k} + \beta_k' x.$$

The difference comes from how the parameters are estimated. The estimate of the αs assumes that the conditional distribution of x is multivariate Gaussian.

Thus, the main difference is that LDA imposes a distributional assumption on X which, if it holds, yields more efficient estimates. Logistic regression is *conditional* methodology. We condition on X and do not specify a distribution for it. This presents a big advantage in cases where we know X can not be normal, e.g. categorical variables.

However, many times in practice both methods perform similarly. Even in extreme cases with categorical variables.

Separating hyperplanes

So far we have seen methods that use a probabilistic argument to estimate parameters. For example, in LDA we assume the class conditional density is Gaussian and find maximum likelihood estimates. In this section, we look at methods that make no probabilistic arguments, but instead rely entirely on geometric arguments.

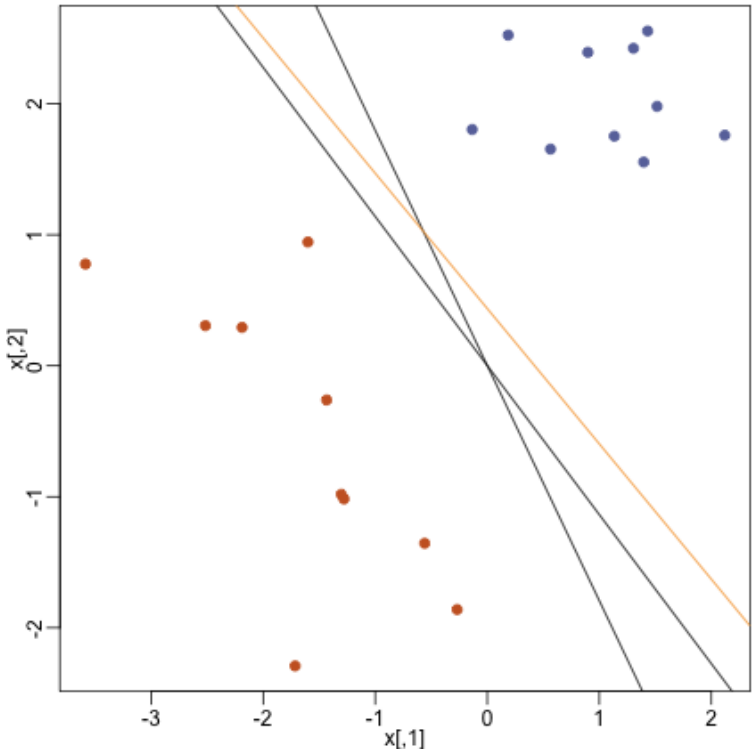
In all of the linear classification we have seen, we are looking for discriminant functions that are *linear* with respect to the covariates $X_1, \ldots, X_p$.

In p-dimensional space $\mathbf{R}^p$ these are described by vectors β. The *decision boundary* is thus

$$L = \{x : \beta' x = 0\}.$$

Technical note: The literature on separating hyperplanes traditionally uses w (they call it a *weight vector*) in place of β.

Notice that this boundary partitions the input space into two sets on each side of the line. If we restrict estimates to those for which $\beta'\beta = \|\beta\|^2 = 1$, then the *signed* distance of any point x to the decision boundary L is $\beta' x$. With this we can easily describe the two partitions as


x[,2]
x[,1]
2
1
0
-1
-2
-3
-2
-1
0
1
2

$$L^+ = \{x : \beta' x > 0\},$$
$$L^- = \{x : \beta' x < 0\}$$

Intuitively, the β we want as an estimate is one that separates the training data as perfectly as possible. If we code our classes as $y = -1$ if $g = 1$ and $y = +1$ if $g = 2$, we can describe our intuitive requirement for estimate β as:

$$y_i(x'\beta) > 0, i = 1, \ldots, N$$

Rosenblatt's algorithm

Rosenblatt's algorithm is one way of finding a vector β that satisfies the separation requirement as much as possible. The goal is to penalize β by how far into the wrong side misclassified points are:

$$D(\beta) = -\sum_{i \in M} y_i x'\beta$$

where M is the set points misclassified (on the wrong side of the line) by β.

Thus, we estimate β by minimizing D. Assuming M is constant, the gradient of D is

$$\frac{\partial D(\beta)}{\partial \beta} = -\sum_{i \in M} y_i x_i$$

Rosenblatt's algorithm uses *stochastic gradient descent*:

1. Initialize β
2. Cycle through training points i, if it is misclassified, update β as

$$\beta \leftarrow \beta + \rho y_i x_i$$

3. Stop when converged (or get tired of waiting)

Parameter ρ is used to control how much we update β in each step.

There are a few problems with this algorithm:

- When the data are separable, i.e. there exists a β that separates the training points perfectly, then there are an infinite number of βs that also separate the data perfectly

- Although this algorithm will converge in a finite number of steps if the training data is separable, the number of finite steps can be *very* large
- When the training data is *not* separable, the algorithm will not converge.

In a future lecture we will discuss Support Vector Machines (SVMs) which are based on addressing these problems.

For example, when the data are separable, SVMs will choose among the infinite number of βs that separate the data perfectly, a single optimal β that maximizes the distance between the decision boundary and the closest point in each class. Why is this a good idea?