Linear Boundaries

Consider the first example (similar to the one in Figure 2.1 of HTF).

We used regression to form a simple dividing line between groups.

We have two features, $x_1$ and $x_2$, and a binary response $y$; that is, there are two groups.

We used 0’s and 1’s for the $y$’s and fitted the linear model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon.$$ 

We predicted the group based on whether or not $\hat{y} > 0.5$. 
Separating Hyperplanes

We consider planes of the form

\[ \{ x : f(x) = x^T \beta + \beta_0 = 0 \}, \]

where \( \| \beta \| = 1 \).

For a given \( x \), the decision rule for the two groups -1 and 1 (Not eht coding!) is

\[ G(x) = \text{sign}(x^T \beta + \beta_0). \]

There may be a plane that actually separates the data; that is, with \( Y \) taking values -1 and 1 (Note the coding!!), we want a hyperplane such that

\[ y_i(x_i^T \beta + \beta_0) > 0. \]
Separating Hyperplanes

We represent a hyperplane $H$ in the feature space as

$$\beta_0 + \beta^T x = 0.$$  

Notice that for two points $x_1$ and $x_2$ lying on $H$, we have

$$\beta^T (x_1 - x_2) = 0.$$  

This means that $\beta$ is perpendicular to $H$; in particular,

$$\beta^* = \beta / \|\beta\|$$

is normal to $H$.

Note that if $x_0$ is a point on $H$, then $\beta^T x_0 = -\beta_0$.

Furthermore, the signed distance of any point $x$ to $H$ is given by

$$\beta^* (x - x_0) = \frac{1}{\|\beta\|} (\beta^T x + \beta_0).$$
Separating Hyperplanes

The last equation says that the distance of any point to the hyperplane $\beta_0 + \beta^T x = 0$ is proportional to $\beta_0 + \beta^T x$, and the constant of proportionality is $1/\|\beta\|.$

Now, let’s consider a given set of data.

If we code the responses as -1 and 1, and $\beta_0 + \beta^T x = 0$ is a separating hyperplane, then $y_i(\beta_0 + \beta^T x_i) > 0$ for all $i$.

If a hyperplane in the feature space separates the two groups, then we consider the two most separated hyperplanes that separate the two groups.

Because the distance of any point $x$ to $H$ is given by

$$\beta^*(x - x_0) = \frac{1}{\|\beta\|}(\beta^T x + \beta_0),$$

there are parallel hyperplanes at a distance of $2/\|\beta\|$ from each other that still separate all of the given points.
If there is one, there are probably others.
The Optimal Separating Hyperplane

Maximize the “margin”,

\[ y_i(x_i^T \beta + \beta_0) \geq M. \]

We have the optimization problem

\[
\max_{\beta_0, \|\beta\|=1} M
\]

subject to the constraint above.
The Optimal Separating Hyperplane

Determining the maximum margin is to solve the optimization problem

$$\max_{\beta, \beta_0, \|\beta\|=1} M$$

subject to

$$y_i(\beta_0 + \beta^T x_i) \geq M \quad \forall i.$$  

This optimization problem is the same as

$$\min_{\beta, \beta_0} \|\beta\|$$

subject to

$$y_i(\beta_0 + \beta^T x_i) \geq 1 \quad \forall i.$$
The Optimal Separating Hyperplane

Now, we consider the case when there is no separating hyperplane.

We modify the objective function to include the distances $\xi_i$ of the misclassified points. (Draw picture.)

We have the optimization problem

$$\min_{\beta, \beta_0} \left( \frac{1}{2}\|\beta\|^2 + C \sum_{i=1}^{n} \xi_i \right)$$

subject to

$$\xi_i \geq 0, \ y_i(\beta_0 + \beta^T x_i) \geq 1 - \xi_i \quad \forall i.$$
Support Vector “Machines”

The basic idea in a support vector “machine” (SVM) is to apply a nonlinear mapping to the feature space, and then use a linear model to form the decision boundary.

(A support vector “machine” is an algorithm — not a machine.)

The nonlinear mapping may increase the number of features from $p$ to some much larger number.

The decision boundary is linear in the space of the transformed (and “new”) features.

When projected back onto the original space, however, the boundary is nonlinear.
A Nonlinear Mapping

Suppose we follow the same approach, still with 0's and 1's for the $y$'s, but adding some features:

$$T(x_1, x_2) = x_1, x_2, x_1^2, x_2^2, x_1, x_1x_2$$

Now, we fit the linear model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2 + \epsilon.$$  

We again predict the group based on whether or not $\hat{y} > 0.5$. 

The Nonlinear Boundary

In the $x_1-x_2$ plane, we choose a grid on say $x_1$, and solve for $x_2$ such that

$$0.5 = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2$$

This quadratic equation may have non-real roots, which we just set to NA.

That corresponds to the boundary going off of the plane.

The choice of sign on the square root of the discriminant is more difficult.

That corresponds to whether the boundary is coming or going.
SVM

Does the nonlinear mapping work?

One problem is computational complexity.

The other problem is overfitting.

SVM addresses the problem slightly differently.

The linear model is the **maximum-margin hyperplane**, and it is fitted by determining the **support vectors**.

The ideas of the SVM method are based on a two-class learning problem in which there is a hyperplane that forms a complete separation between the two classes.
SVM

We seek a solution to the optimization problem after applying nonlinear transformations on the features.

We generally limit ourselves to polynomials of a given degree.

Another nonlinear transformation is the inner product of two normal PDFs.

SVM in R:

In the package e1071 the function is svm.

In the package kernlab the function is ksvm.

Both functions allow a parameter specifying the maximum cost.

svm uses the product of two normal PDFs, so another parameter is the reciprocal of the variance.
Discriminant Analysis

The techniques of discriminant analysis are based on relative probabilities, just as the Bayes classifier.

Given an observation on a predictor variable \( X \), our interest is in the conditional probability distribution of the class variable \( G \).

If \( f_k(x) \) is the probability the \( G = k \) and \( X = x \), then we have

\[
\Pr(G = k | X = x) = \frac{\Pr(G = k \text{ and } X = x)}{\Pr(X = x)} = \frac{f_k(x) \pi_k}{\sum_{j=1}^{K} f_j(x) \pi_j},
\]

where \( \pi_j \) are “prior” probabilities of being in one class or another.
Discriminant Analysis

An important first step is to extend the idea of probability to probability density.

The prior probabilities can arise either from prior “beliefs” (from a Bayesian perspective) or from some known or assumed distribution of the relative probabilities of being in each class.

The most common way of assigning prior probabilities is to use the relative proportion of a class in the training set as the prior probability of that class.
Discriminant Analysis

Suppose that the predictor variables in each class have a $p$-variate normal distribution with the same variance-covariance matrix, but just with different means; that is, the “probability” is the PDF

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

This leads to linear discriminant analysis, or LDA.

For two classes, $k$ and $j$, we want to compare $\Pr(G = k|X = x)$ with $\Pr(G = j|X = x)$, but that is just comparing $f_k(x)$ with $f_j(x)$. 

Discriminant Analysis

The ratio \( \Pr(G = k | X = x) / \Pr(G = j | X = x) \) is just \( f_k(x) / f_j(x) \), and that simplifies because the constant out front cancels; furthermore, if we take the log of the ratio, we have, after some rearrangement,

\[
\log \left( \frac{\Pr(G = k | X = x)}{\Pr(G = j | X = x)} \right) = -\frac{1}{2} (\mu_k - \mu_j)^T \Sigma^{-1} (\mu_k - \mu_j) + x^T \Sigma^{-1} (x - \mu_k)
\]

This means that the decision boundary between classes \( k \) and \( j \) is just the hyperplane

\[
x^T \Sigma^{-1} (\mu_k - \mu_j) = \frac{1}{2} (\mu_k - \mu_j)^T \Sigma^{-1} (\mu_k - \mu_j).
\]

The decision function for the \( k^{th} \) class is

\[
\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k.
\]
Discriminant Analysis

What next?

Well, suppose that the variance-covariance matrices are different.

No problem, we can estimate them separately from the individual classes in the training data.

The constant terms in the ratio do not cancel, however, and so the discriminant function has an additional term, and the $x$ enters quadratically:

$$
\delta_k(x) = \frac{1}{2} \log(|\Sigma_k|) - \frac{1}{2} (x - \mu_k)^T \Sigma^{-1} (x - \mu_k).
$$
Discriminant Analysis

LDA or QDA still may not do so well.

What next?

Use the same things we’ve discussed for linear regression.

Three things:
• add powers of variables (also same idea as in SVM)
• penalize/regularize
• use mixtures: different discriminators in different regions

“flexible discriminant analysis” (FDA)
Flexible Discriminant Analysis

First, we note that the linear discriminant functions tessellate the $p$ space of the predictors into $K$ regions.

The same tessellation can be achieved by linear regression, after we have mapped the group indicators into the reals:

$$\theta(g_j) \in \mathbb{R}.$$

But how?

We do it in such a way as to get a good fit.

Least squares (there could be other ways):

$$\min_{\theta, \beta} \sum_{i=1}^{N} (\theta(g_i) - x_i^T \beta)^2$$
Flexible Discriminant Analysis

What about the $\theta$?

We have a overdetermined problem; there are two many classes to do this for each. (One line separates two groups; two lines separate three groups; etc.)

Determine $\theta$ for $K - 1$ groups.

Minimize the **average squared residual**:

$$\text{ASR} = \frac{1}{N} \sum_{k=1}^{K-1} \sum_{i=1}^{N} (\theta_k(g_i) - x_i^T \beta_k)^2.$$
Flexible Discriminant Analysis

Minimizing the ASR yields the same separators (now called $\beta_k$) as the straightforward LDA (called discriminant vectors).

What have we achieved by this reformulation?

This is what allows us to do the same kinds of things we did in regression.

Regularization:

$$\min_{\theta, \beta} \left( \sum_{k=1}^{K-1} \sum_{i=1}^{N} \left( (\theta_k(g_i) - x_i^T \beta_k)^2 + \lambda J(\beta_k) \right) \right),$$

where $J$ is a regularizer, most often a norm.
Flexible Discriminant Analysis

Other types of smoothers, instead of just $x^T \beta$, together with regularization:

$$\min_{\theta, \eta} \left( \sum_{k=1}^{K-1} \sum_{i=1}^{N} \left( (\theta_k(g_i) - \eta_k(x_i))^2 + \lambda J(\eta_k) \right) \right).$$
Prototypes

Classification is basically the identification of various regions in the feature space that are associated with the various classes.

This is often based on a tessellation.

The tessellations can be very general.

A simple tessellation is based on distances of points to the centroid; for example, in $K$-means clustering.

The regions of the feature space are defined in terms closeness to prototype points.

If there is just one prototype for each class, then the dividing lines are straight lines.
Multiple Prototypes

Each class may be associated with multiple prototype points.

Prototype points could just be randomly sampled from the training data, but of course, this is likely not to be very good.

A point in the feature space is classified based on its closest prototype.

Of course, it could also be based on its total (or average) distance to the closest \( m \) prototype points.

This yields boundaries between classes that are curvilinear.
Prototypes

Prototype points could be defined in terms of mixture models.

Each class may be associated with multiple subclasses.

The means of a mixture of normals can be taken as prototype points.

How to fit them? EM algorithm.
Prototypes

Another approach to establishing good prototype points is to randomly sample some from the training data, and then use the training data to move them around so that they are more effective.

We want to push prototypes of the wrong class away and attract prototypes of the correct class.

For each training point, the distance of movement is called the “learning rate”, $\epsilon$. 
Learning Vector Quantization

1. Choose $R$ initial prototypes for each class by random sampling of the training data. Call these $m_1(k), \ldots, m_R(k)$ for given $k$.

2. Randomly choose a set of training points, and for each $x_i$ in the set:

   (a) Determine the closest prototype $m_j(k)$.

   (b) If $g_i = k$, then move the prototype closer to the training point:

   $$m_j(k) \leftarrow m_j(k) + \epsilon(x_i - m_j(k))$$

   (c) If $g_i \neq k$, then move the prototype closer to the training point:

   $$m_j(k) \leftarrow m_j(k) - \epsilon(x_i - m_j(k))$$

3. Decrease $\epsilon$ and go back to step 2, unless some stopping criterion is satisfied.

Self-organizing maps.
Nearest Neighbors

\(k\)-nearest neighbor classifiers are “memory-based”; there is no real “model”.

A point is classified based on the predominant class of its \(k\) nearest neighbors.

In high dimensions, data become very sparse.

A “circular” metric may not be very good.

A circular metric may also not be very good if the division between the groups is circular (e.g., Figure 13.14 in HTF).

An adaptive metric may be better.
Adaptive Metrics for Nearest Neighbors

One simple way, under the assumption that within each group the “variance-covariance” matrices are the same is to sphere the data with respect to that variance (that is, the “within” variance), and then at each point in the feature space, use the “between” variance based on a fixed number of nearest neighbors.

“discriminant adaptive nearest-neighbor” (DANN)