Machine Learning for OR & FE

Introduction to Classification Algorithms

Martin Haugh

Department of Industrial Engineering and Operations Research
Columbia University
Email: `martin.b.haugh@gmail.com`

Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani (JWHT).

Additional References: Chapter 4 of JWHT, Bishop's PRML and HTF's TESL.

Outline

What is Classification?

Goal is to predict a categorical outcome from a vector of inputs

- inputs can be quantitative, ordinal, or categorical
- inputs could also be images, speech, text, networks, temporal data, spatial data etc.

Classification algorithms require inputs to be encoded in quantitative form

- can result in very high dimensional problems!

Simplest and most common type of classification is binary classification

- email classification: spam or not spam?
- sentiment analysis
    - is the movie review good or bad?
    - is this good news for the stock or bad news?
- fraud detection
- revenue management: will a person buy or not?
- medical diagnosis: does a patient have a disease or not?
- will somebody vote for Obama or not?
- is somebody a terrorist or not?

But also have classification problems with multiple categories.

What is Classification?

Classification often used as part of a decision support system.

There are many(!) different classification algorithms

- will cover many of the best known algorithms
- but will not have time to cover all of them such as neural networks, ensemble methods etc.
- will cover support vector machines later in the course.

Most classification algorithms can be categorized as generative or discriminative.
Classification algorithms learn a classifier using training data

- then used to predict category or class for new inputs or test data.

Will use $\mathbf{X}$ or $\mathbf{x}$ to denote vector of inputs and G or Y to denote category or class.

Generative Classification Algorithms

Generative methods focus on modeling $\mathsf{P}(\mathbf{X}, G)$ and then use $\hat{G}(\mathbf{X})$ as a classifier where

$$\begin{aligned} \hat{G}(\mathbf{X}) \;:=\; \underset{G}{\operatorname{argmax}}\, \hat{\mathsf{P}}(G|\mathbf{X}) &= \underset{G}{\operatorname{argmax}}\, \frac{\hat{\mathsf{P}}(\mathbf{X}|G)\hat{\mathsf{P}}(G)}{\sum_i \hat{\mathsf{P}}(\mathbf{X}|G_i)\hat{\mathsf{P}}(G_i)} \\ &= \underset{G}{\operatorname{argmax}}\, \hat{\mathsf{P}}(\mathbf{X}|G)\hat{\mathsf{P}}(G) \end{aligned} \tag{1}$$

Examples include linear discriminant analysis (LDA), quadratic discriminant analysis (QDA) and naive Bayes.

LDA and QDA assume Gaussian densities for $\hat{\mathsf{P}}(\mathbf{X}|G)$.

Naive Bayes assumes

$$\mathsf{P}(\mathbf{X}|G) \;=\; \prod_j \mathsf{P}(X_j|G)$$

so the features are independent conditional on G.

- a strong and generally unrealistic assumption but naive Bayes still often works very well in practice.

Discriminative Classification Algorithms

Discriminative methods focus on modeling $\mathrm{P}(G|\mathbf{X})$ directly

- examples include least squares or regression-based classifiers, logistic regression and Bayesian logistic regression.

Discriminative methods may also focus on minimizing the expected classification error directly without making any assumptions regarding $\mathrm{P}(\mathbf{X}, G)$ or $\mathrm{P}(G|\mathbf{X})$.

Examples include:

- classification trees
- k-nearest neighbors
- support vector machines (SVMs)
- (deep) neural networks.

k-Nearest Neighbors

k-nearest neighbors (k-NN) is a very simple classification algorithm.

Given a new data-point, $\mathbf{x}$, that needs to be classified, we find the k nearest neighbors of $\mathbf{x}$ and use majority voting of these neighbors to classify $\mathbf{x}$.

Need a metric to measure distance between data-points

- easy for quantitative variables and also straightforward for ordinal variables
- generally a good idea (why?) to standardize features so they have mean 0 and variance 1

But constructing a metric not so straightforward for other variables including categorical variables, text documents, images, etc.

- choice of metric can be very important in these situations.

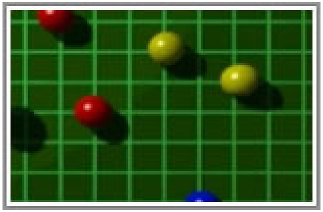
k is usually chosen by using separate training and test sets, or via cross-validation

- to be covered soon.

Despite its simplicity, k-NN often works very well in practice but can be computationally expensive to classify new points.

k-NN's can also be used for regression – k-NN regression

k-Nearest Neighbors


A Demo of k-Nearest Neighbors by Antal van den Bosch

Kernel Classification

k-NN gives equal weight to all k nearest neighbors but it may make more sense to give more weight to the closest neighbors –leads to kernel classification.

With kernel classification every training point, $\mathbf{x}_i$, gets a vote of $K(\mathbf{x}, \mathbf{x}_i)$ when classifying a new point, $\mathbf{x}$.

e.g. A Gaussian-type kernel takes the form

$$K(\mathbf{x}, \mathbf{x}_i) = e^{-d(\mathbf{x},\mathbf{x}_i)/\sigma^2}$$

where $d(\mathbf{x}, \mathbf{x}_i)$ is a distance measure between $\mathbf{x}$ and $\mathbf{x}_i$.

Question: What happens as $\sigma \to \infty$?

Question: What happens as $\sigma \to 0$?

Model Assessment

Goal of model selection: choose the model with the best predictions on new data.

Complex models generally fit a given training set better than less complex models

- follows since more complex models have more flexibility
- but often results in over-fitting in which case the fitted complex model does not generalize well.

This is related to the bias-variance decomposition that we saw earlier when we studied regression.

Training error refers to classification error on the data-set that was used to train the classifier.

Test error refers to classification error on a new or holdout data-set that was not used to train the classifier

- provides an estimate of generalization error.

Training Error Versus Test Error

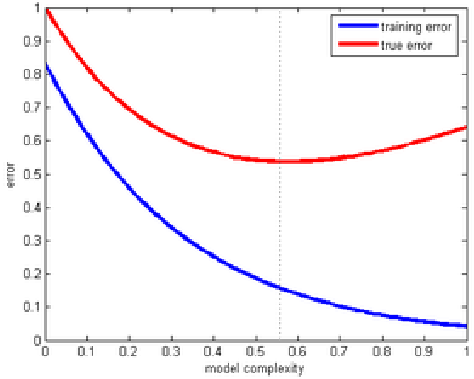


Figure taken from Ben Taskar's web-site at U Penn.

Training error generally declines as model complexity increases. Why?

However, true error, i.e. test or generalization error, tends to decrease for a while and then increase as it begins to over-fit the data.

Approaches to Control Over-Fitting

Many approaches used to control over-fitting:

- The Akaike information criterion (AIC) and Bayesian information criterion (BIC) penalize the effective # of parameters
  - used in MLE settings when we can compute effective # of parameters.
- Bayesian models control over-fitting naturally by modeling parameters as random variables
  - estimation in these models therefore implicitly accounts for parameter uncertainty
  - Bayesian models are very popular in statistics and ML.
- Regularization approaches that explicitly penalize parameter magnitudes along with the misclassification or prediction error in the objective function.

  Smaller magnitude parameters preferred to larger magnitude parameters with degree of regularization controlled via a regularization parameter, λ.

  e.g. ridge regression solves

$$\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2} \left\| \mathbf{y} - \mathbf{X}\boldsymbol{\beta} \right\|^2 + \lambda \cdot \frac{1}{2} \left\| \boldsymbol{\beta} \right\|_2^2 \right\}.$$

Approaches to Control Over-Fitting

- A standard approach is to use training, validation and test sets.
- Another very useful technique is cross-validation
  - often the tool of choice
  - and also used to choose regularization parameters; e.g. λ in ridge regression.

For now will restrict ourselves to the training, validation and test set approach

- will consider cross-validation later.

Training, Validation and Test Sets

To assess model performance we can partition the data into a training set and a validation set.

Training set used to construct the classifier(s) and the validation set is used to assess their performance(s).

The performance of each classifier on validation set used to obtain an unbiased estimator of the classifier's performance.

If we have trained many classifiers then in the model selection stage can choose the classifier that performed best on the validation set.

Question: When we choose a classifier this way, is its performance on the validation set still an unbiased estimator of its performance?

Answer: No. Why?

As a result we would like an additional test set which is used to evaluate the selected classifier. The test set is never used in the training and validation stages.

Optimal Bayes Classifier

Consider again the generative framework where we have the joint distribution function, $\mathsf{P}(\mathbf{X}, G)$, for the feature vector, $\mathbf{X}$, and the associated category, G.

For a given classifier, $\hat{G}(\cdot)$, and a given loss function, $L(G, \hat{G}(\mathbf{X}))$, the expected prediction error (EPE) is given by

$$\begin{aligned}
\text{EPE} &= \mathsf{E}_{\mathbf{X},G}\left[L(G, \hat{G}(\mathbf{X}))\right] \\
&= \mathsf{E}_{\mathbf{X}}\left[\mathsf{E}_{G}\left[L(G, \hat{G}(\mathbf{X}))|\mathbf{X}\right]\right] \\
&= \mathsf{E}_{\mathbf{X}}\left[\sum_{k=1}^{K} L(G_k, \hat{G}(\mathbf{X}))\, \mathsf{P}(G_k|\mathbf{X})\right]
\end{aligned}$$

Wish to minimize EPE as a function of $\mathbf{X}$ and we can do it pointwise to obtain

$$\hat{G}(\mathbf{x}) = \underset{G \in \mathcal{G}}{\operatorname{argmin}} \sum_{k=1}^{K} L(G_k, G)\, \mathsf{P}(G_k|\mathbf{X} = \mathbf{x}) \qquad (2)$$

Optimal Bayes Classifier

A commonly used loss function for classification problems is the $0-1$ loss function which assigns a loss of 0 to correctly classified data-points and 1 to those that are incorrectly classified.

If we now assume this loss function then (2) reduces to

$$\begin{aligned} \hat{G}(\mathbf{x}) &= \underset{G \in \mathcal{G}}{\operatorname{argmin}} \left[1 - \mathsf{P}(G|\mathbf{X} = \mathbf{x})\right] \\ &= \underset{G \in \mathcal{G}}{\operatorname{argmax}} \, \mathsf{P}(G|\mathbf{X} = \mathbf{x}) \end{aligned}$$

so that we classify to the most probable class.

This is the Bayes classifier and error rate of this classifier is the Bayes rate.

The Bayes rate is the best possible error rate for the $0-1$ loss function

- but generally not achievable in practice because we do not know $\mathsf{P}(\mathbf{X}, G)$.

But can be computed in simulated problems where we wish to evaluate the performance of other classification algorithms.

Naive Bayes

Recall naive Bayes is a generative classifier that estimates $\mathsf{P}(\mathbf{X}, G)$ and assumes

$$\mathsf{P}(\mathbf{X}|G) \;=\; \prod_j \mathsf{P}(X_j|G) \tag{3}$$

so the features are independent conditional on G.

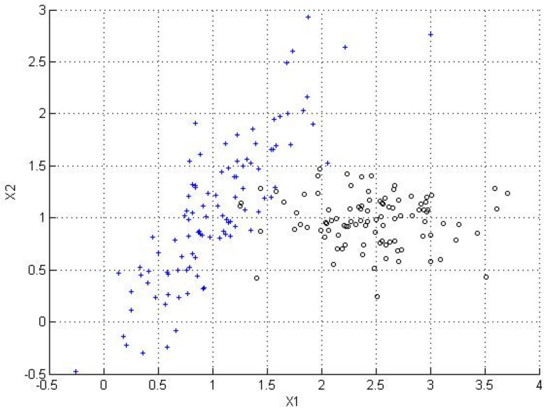
Since $\mathsf{P}(\mathbf{X}, G) = \mathsf{P}(\mathbf{X}|G)\mathsf{P}(G)$, naive Bayes estimates $\mathsf{P}(G)$ and the $\mathsf{P}(X_j|G)$'s separately via MLE and then classifies according to

$$\begin{aligned}\hat{G}(\mathbf{x}) &= \underset{G\in\mathcal{G}}{\operatorname{argmax}}\, \hat{\mathsf{P}}(G|\mathbf{X}=\mathbf{x}) \\ &= \underset{G\in\mathcal{G}}{\operatorname{argmax}}\, \hat{\mathsf{P}}(G)\hat{\mathsf{P}}(\mathbf{x}|G) \\ &= \underset{G\in\mathcal{G}}{\operatorname{argmax}}\, \hat{\mathsf{P}}(G)\prod_i \hat{\mathsf{P}}(x_i|G)\end{aligned}$$

Assumption (3) is strong and generally unrealistic

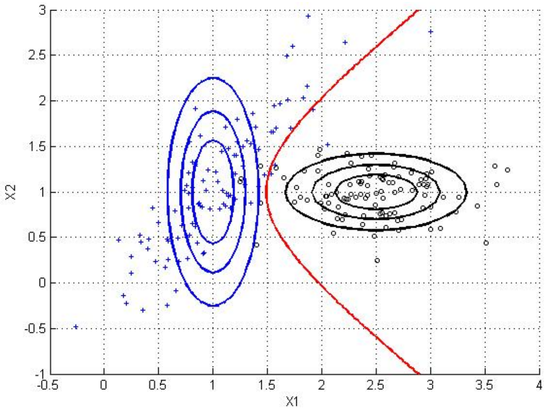
- but when $\mathbf{X}$ is high-dimensional and categorical estimating $\mathsf{P}(\mathbf{X}, G)$ is generally (why?) not possible
- so assumption (3) makes estimation much easier and naive Bayes still often works very well in practice!

Naive Bayes in Action



Naive Bayes: There are 2 equiprobable classes and the class-conditional densities are bivariate Normal. The assumptions of naive Bayes do not apply. Why?

Naive Bayes in Action



Naive Bayes: The contours of the fitted class-conditional densities. These densities are also assumed to be bivariate Normal. The naive Bayes classifier is given by the red curve.

Naive Bayes and Text Classification

Naive Bayes often works well when the data cannot support a more complex classifier – this is the bias-variance decomposition again.

Has been very successful in text classification or sentiment analysis

- e.g. is an email spam or not?

But how would you encode an email into a numerical input vector?

A simple and common way to do this is via the bag-of-words model

- completely ignores the ordering of the words
- stop-words such as “the”, “and”, “of”, “about” etc. are thrown out
- words such as “walk”, “walking”, “walks” etc. all identified as “walk”

Email classification then done by assuming a given email comes from either a “spam” bag or a “non-spam”bag

- naive Bayes assumes $\mathrm{P}(\mathrm{spam}|\mathbf{X}) \propto \mathrm{P}(\mathrm{spam}) \prod_{\mathrm{word} \in \mathrm{email}} \mathrm{P}(\mathrm{word}|\mathrm{spam})$.

Bag-of-words also often used in document retrieval and classification

- leads to the term-document matrix
- also then need a measure of similarity between documents, e.g. cosine distance possibly in TF-IDF version of term-document matrix.

Classification Using Linear Regression

The 1-of-K encoding scheme uses K binary response variables to encode each data-point with

$$Y_k = \begin{cases} 1, & \text{if G = k} \\ 0, & \text{otherwise.} \end{cases}$$

Linear regression approach to classification fits a linear regression with Y_k as the response variable

- so K different linear regressions are performed.

Let $\hat{f}_k(\mathbf{x})$ denote the fitted regression when Y_k is the response variable. Classifier then given by

$$\hat{G}(\mathbf{x}) \ := \ \underset{k \in \mathcal{G}}{\operatorname{argmax}} \, \hat{f}_k(\mathbf{x})$$

so we simply classify to the largest component.

Cons of Doing Classification Via Regression

Lacks robustness to outliers like all least squares methods.

Masking is a major problem when there is more than two classes

- not too surprising given that least-squares solutions correspond to MLE estimates for conditional Gaussian distributions and not conditional Bernoulli distributions.

Figures on slides (26) and (44) demonstrate this masking phenomenon.

According to HTF all polynomial terms up to degree $K-1$ might be needed to avoid this masking problem.

Masking not an issue with other classification techniques that we consider.

Linear Discriminant Analysis (LDA)

LDA a generative model that assumes class-conditional densities, $f_k(\mathbf{x})$, are Gaussian with a common covariance matrix so that

$$f_k(\mathbf{x}) \;=\; \frac{1}{(2\pi)^{M/2}|\mathbf{\Sigma}|^{1/2}}\, e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)}$$

Writing π_k for $\mathrm{P}(G=k)$ we see from (1) that the LDA classifier satisfies

$$\hat{G}(\mathbf{x}) = \underset{k}{\operatorname{argmax}} \frac{\hat{f}_k(\mathbf{x})\hat{\pi}_k}{\sum_{j=1}^{K}\hat{f}_j(\mathbf{x})\hat{\pi}_j} \tag{4}$$

where we have used $\hat{f}_k(\cdot)$ and $\hat{\pi}_k$ to denote MLE estimates of $f_k(\cdot)$ and π_k with

$$\begin{aligned}
\hat{\pi}_k &= N_k/N \\
\hat{\boldsymbol{\mu}}_k &= \sum_{g_i=k} \mathbf{x}_i/N_k \\
\hat{\mathbf{\Sigma}} &= \frac{\sum_{k=1}^{K}\sum_{g_i=k}(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_k)^\top}{N-K}
\end{aligned} \tag{5}$$

Linear Discriminant Analysis (LDA)

Can also calculate log-ratio between two classes, k and l, to get

$$\log \frac{\mathsf{P}(G=k|\mathbf{X}=\mathbf{x})}{\mathsf{P}(G=l|\mathbf{X}=\mathbf{x})} = \log \frac{\hat{\pi}_k}{\hat{\pi}_l} - \frac{1}{2}(\hat{\boldsymbol{\mu}}_k + \hat{\boldsymbol{\mu}}_l)^\top \hat{\boldsymbol{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_l) + \mathbf{x}^\top \hat{\boldsymbol{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}_l) \quad (6)$$

Based on (6) can define the linear discriminant functions

$$\delta_k(\mathbf{x}) := \mathbf{x}^\top \hat{\boldsymbol{\Sigma}}^{-1} \hat{\boldsymbol{\mu}}_k + \underbrace{\log \hat{\pi}_k - \frac{1}{2}\hat{\boldsymbol{\mu}}_k^\top \hat{\boldsymbol{\Sigma}}^{-1} \hat{\boldsymbol{\mu}}_k}_{k^{th}\text{ intercept}} \quad (7)$$

for $k = 1, \ldots, K$.

LDA classifier of (4) then reduces to

$$\hat{G}(\mathbf{x}) = \operatorname*{argmax}_k \delta_k(\mathbf{x}).$$

Linear Discriminant Analysis (LDA)

When there are just 2 classes it can be shown the linear regression and LDA classifiers only differ in their intercepts

- not true for $K > 2$.

But since LDA is based on Gaussian assumptions and linear regression makes no probabilistic assumptions, Hastie et al. suggest choosing the intercepts in (7) to minimize the classification error

- they report this works well in practice.

LDA does not suffer from the masking problem of linear regression.

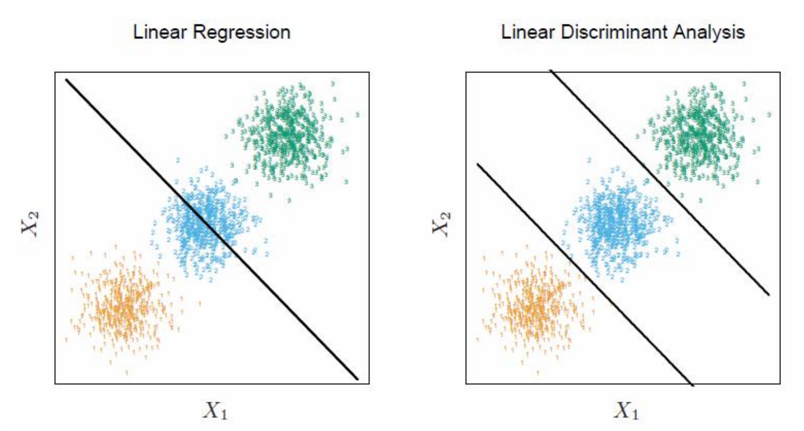


Figure 4.2 from HTF: The data come from three classes in $\mathbb{R}^2$ and are easily separated by linear decision boundaries. The right plot shows the boundaries found by linear discriminant analysis. The left plot shows the boundaries found by linear regression of the indicator response variables.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

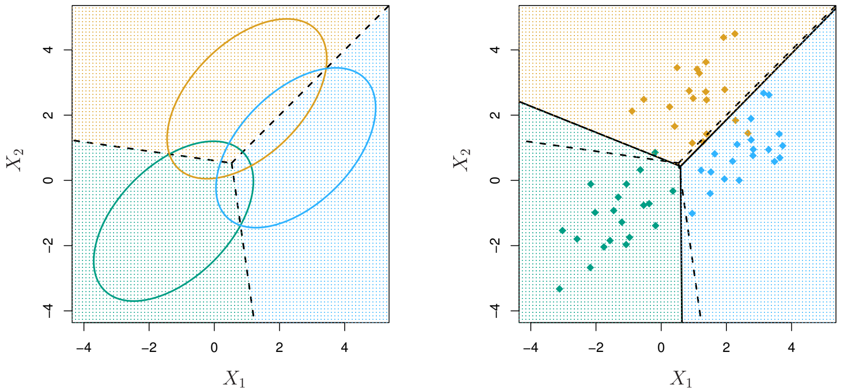


Figure 4.6 from ISLR: An example with three classes. The observations from each class are drawn from a multivariate Gaussian distribution with $p = 2$, with a class-specific mean vector and a common covariance matrix. Left: Ellipses that contain 95% of the probability for each of the three classes are shown. The dashed lines are the Bayes decision boundaries. Right: 20 observations were generated from each class, and the corresponding LDA decision boundaries are indicated using solid black lines. The Bayes decision boundaries are once again shown as dashed lines.

The Default Data from ISLR

ISLR use LDA to obtain a classification rule for default / non-default on the basis of: (i) credit card balance and (ii) student status.

There were 10,000 training samples and the overall default rate was 3.33%.

The training error rate of LDA was 2.75%. But how good is this?

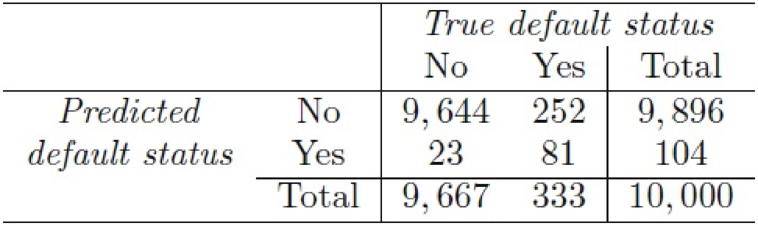
- note training error rate will usually be biased low and therefore lower than test / generalization error
- for comparison, how well does the useless classifier that always predicts non-default do?

We are often interested in breaking out the (training) error rate into the two possible types of error:

1. False positives
2. False negatives.

This leads to the so-called confusion matrix.

The Confusion Matrix



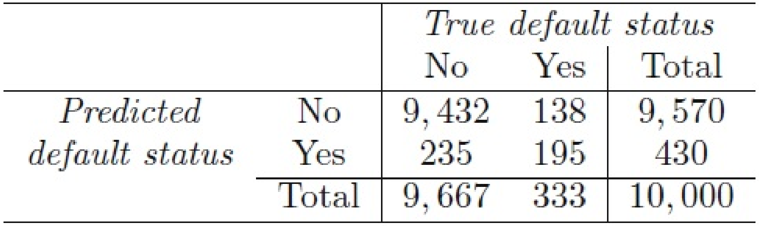
		True default status		
		No	Yes	Total
Predicted default status	No	9,644	252	9,896
	Yes	23	81	104
	Total	9,667	333	10,000

Table 4.4 from ISLR: A confusion matrix compares the LDA predictions to the true default statuses for the $10,000$ training observations in the Default data set. Elements on the diagonal of the matrix represent individuals whose default statuses were correctly predicted, while off-diagonal elements represent individuals that were misclassified. LDA made incorrect predictions for 23 individuals who did not default and for 252 individuals who did default.

Note that the LDA classifier only predicts $81/(81+252) = 24.3\%$ of the true defaults.

Do you think this would be acceptable? If not, do we need to abandon LDA or can we somehow "rescue" it?

The Confusion Matrix for a Different Threshold



		True default status		
		No	Yes	Total
Predicted default status	No	9,432	138	9,570
	Yes	235	195	430
	Total	9,667	333	10,000

Table 4.5 from ISLR: A confusion matrix compares the LDA predictions to the true default statuses for the $10,000$ training observations in the Default data set, using a modified threshold value that predicts default for any individuals whose posterior default probability exceeds 20%.

We can rescue LDA by simply adjusting the **threshold** to emphasize one type of error over another.

In Table 4.5 we “predict“ default if the LDA model has

$$\mathbf{Prob}\,(\text{default} = \text{Yes} \,|\, \mathbf{X} = \mathbf{x}) > 0.2.$$

What happens the overall training error rate with this new rule? Do we care?

The Tradeoff from Modifying the Threshold

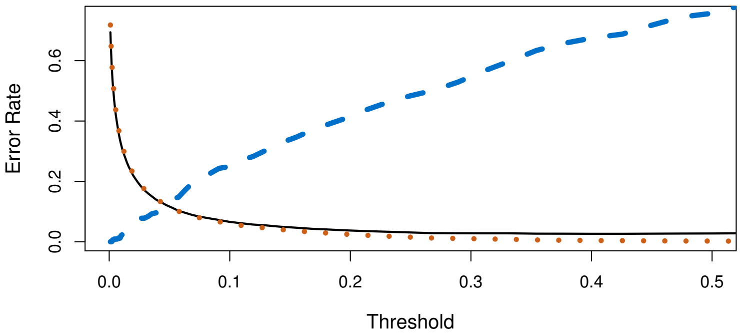
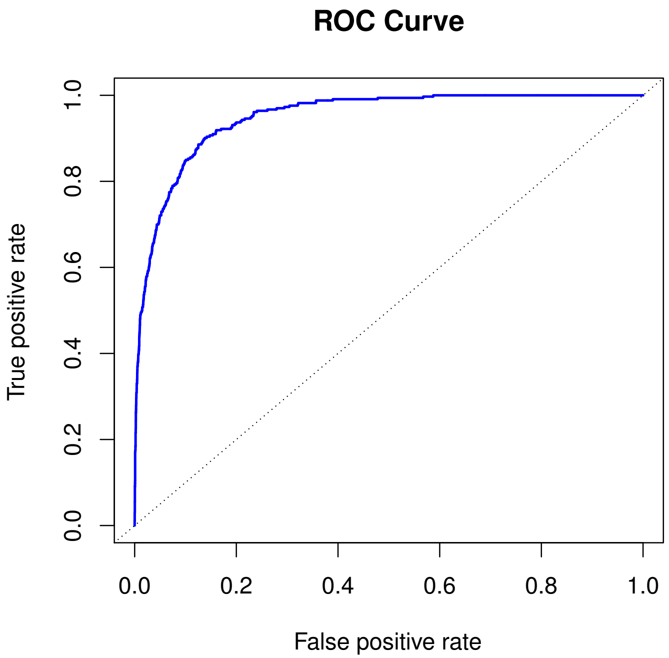


Figure 4.7 from ISLR: For the Default data set, error rates are shown as a function of the threshold value for the posterior probability that is used to perform the assignment. The black solid line displays the overall error rate. The blue dashed line represents the fraction of defaulting customers that are incorrectly classified, and the orange dotted line indicates the fraction of errors among the non-defaulting customers.

Domain specific knowledge required to decide on appropriate threshold
- the ROC curve often used for this task.


ROC Curve
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

Previous slide: Figure 4.8 from ISLR: A ROC curve for the LDA classifier on the Default data. It traces out two types of error as we vary the threshold value for the posterior probability of default. The actual thresholds are not shown. The true positive rate is the sensitivity: the fraction of defaulters that are correctly identified, using a given threshold value. The false positive rate is 1-specificity: the fraction of non-defaulters that we classify incorrectly as defaulters, using that same threshold value. The ideal ROC curve hugs the top left corner, indicating a high true positive rate and a low false positive rate. The dotted line represents the "no information" classifier; this is what we would expect if student status and credit card balance are not associated with probability of default.

Overall performance of the classifier – summarized over all possible thresholds – is given by the AUC or "area under the curve".

The ideal classifier will have an AUC of 1 and will hug top left corner.

ROC curves are useful for comparing classifiers as they factor in all possible thresholds.

Clearly we can alter the threshold for many classifiers and so confusion matrix / ROC curve can be constructed for most binary classifiers.

Quadratic Discriminant Analysis (QDA)

Drop equal covariance assumption and obtain quadratic discriminant functions

$$\delta_k(\mathbf{x}) \;:=\; -\frac{1}{2}\log|\hat{\boldsymbol{\Sigma}}_k| \;-\; -\frac{1}{2}(\mathbf{x}-\hat{\boldsymbol{\mu}}_k)^\top\,\hat{\boldsymbol{\Sigma}}_k^{-1}(\mathbf{x}-\hat{\boldsymbol{\mu}}_k) \;+\; \log\hat{\pi}_k$$

QDA classifier is then

$$\hat{G}(\mathbf{x}) = \underset{k}{\operatorname{argmax}}\,\delta_k(\mathbf{x})$$

So decision boundaries between each pair of classes then given by quadratic functions of **x**.

LDA using linear and quadratic features generally gives similar results to QDA

- QDA generally preferred due to greater flexibility at the cost of more parameters to estimate.

LDA and QDA are popular and successful classifiers

- probably because of the bias-variance decomposition and because the data can often ”only support simple decision boundaries”.

LDA with Quadratic Predictors v QDA

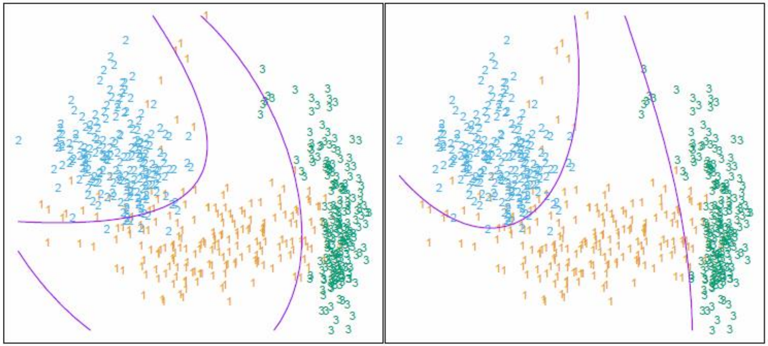


Figure 4.6 from HTF: Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space X_1, X_2, X_1X_2, X_1^2, X_2^2). The right plot shows the quadratic decision boundaries found by QDA. The differences are small, as is usually the case.

LDA v LDA with Quadratic Predictors

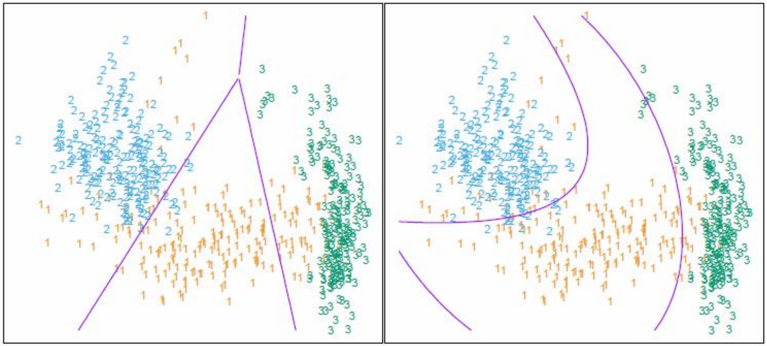


Figure 4.1 from HTF: The left plot shows some data from three classes, with linear decision boundaries found by linear discriminant analysis. The right plot shows quadratic decision boundaries. These were obtained by finding linear boundaries in the five-dimensional space X_1, X_2, $X_1 X_2$,X_1^2, X_2^2. Linear inequalities in this space are quadratic inequalities in the original space.

LDA v QDA

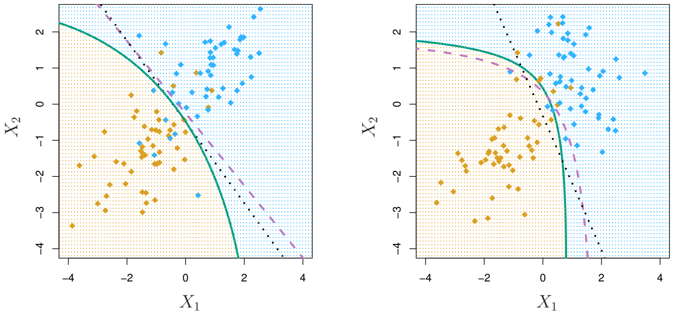


Figure 4.9 from ISLR: Left: The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2$. The shading indicates the QDA decision rule. Since the Bayes decision boundary is linear, it is more accurately approximated by LDA than by QDA. Right: Details are as given in the left-hand panel, except that $\mathbf{\Sigma}_1 \neq \mathbf{\Sigma}_2$. Since the Bayes decision boundary is non-linear, it is more accurately approximated by QDA than by LDA.

Note that LDA is superior (why?!) when true boundary is (close to) linear.

Logistic Regression

Two possible classes encoded as $y \in \{0,1\}$ and assume

$$\mathsf{P}(y=1|\mathbf{x},\mathbf{w}) \;=\; \frac{\exp\left(\mathbf{w}^\top\mathbf{x}\right)}{1+\exp\left(\mathbf{w}^\top\mathbf{x}\right)}$$

where **w** an $m+1$-parameter vector and first element of **x** is the constant 1.

Follows that

$$\mathsf{P}(y=0|\mathbf{x},\mathbf{w}) \;=\; 1-\mathsf{P}(y=1|\mathbf{x},\mathbf{w}) \;=\; \frac{1}{1+\exp\left(\mathbf{w}^\top\mathbf{x}\right)}.$$

Given N independently distributed data-points can write likelihood as

$$L(\mathbf{w}) \;=\; \prod_{i=1}^{N} p_i(\mathbf{w})^{y_i}\left(1-p_i(\mathbf{w})\right)^{1-y_i}$$

where $p_i(\mathbf{w}) := \mathsf{P}(y_i=1|\mathbf{x}_i,\mathbf{w})$. Obtain **w** by maximizing the log-likelihood

$$l(\mathbf{w}) = \sum_{i=1}^{N}\left(y_i\mathbf{w}^\top\mathbf{x}_i - \log\left(1+\exp(\mathbf{w}^\top\mathbf{x}_i)\right)\right) \tag{8}$$

MLE Estimation

To maximize $l(\mathbf{w})$ we set its derivatives to 0 and obtain

$$\frac{\partial l(\mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^{N} \mathbf{x}_i (y_i - p_i(\mathbf{w})) = 0 \tag{9}$$

– so have $m+1$ non-linear equations in $\mathbf{w}$.

First component of each $\mathbf{x}_i$ is 1 so at MLE solution have $\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p_i(\mathbf{w})$.

Can solve (9) iteratively using Newton-Raphson steps

$$\mathbf{w}_{new} = \mathbf{w}_{old} - \left(\frac{\partial^2 l(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^\top} \right)^{-1} \frac{\partial l(\mathbf{w})}{\partial \mathbf{w}} \tag{10}$$

where the partial derivatives are evaluated at $\mathbf{w}_{old}$.

Let $\mathbf{V}$ be the $N \times N$ diagonal matrix with $\mathbf{V}_{i,i} = p_i(\mathbf{w})(1 - p_i(\mathbf{w}))$.
And let $\mathbf{X}$ be the $N \times (m+1)$ matrix of $\mathbf{x}_i$'s.

MLE Via Iteratively Reweighted Least Squares

Can then write

$$\begin{aligned}\frac{\partial l(\mathbf{w})}{\partial \mathbf{w}} &= \mathbf{X}^\top(\mathbf{y}-\mathbf{p}) \\ \frac{\partial^2 l(\mathbf{w})}{\partial \mathbf{w}\partial \mathbf{w}^\top} &= -\mathbf{X}^\top \mathbf{V}\mathbf{X}\end{aligned}$$

so that (10) becomes

$$\begin{aligned}\mathbf{w}_{new} &= \mathbf{w}_{old} + \left(\mathbf{X}^\top \mathbf{V}_{old}\mathbf{X}\right)^{-1}\mathbf{X}^\top(\mathbf{y}-\mathbf{p}_{old}) \\ &= \left(\mathbf{X}^\top \mathbf{V}_{old}\mathbf{X}\right)^{-1}\mathbf{X}^\top \mathbf{V}_{old}\left(\mathbf{X}\mathbf{w}_{old} + \mathbf{V}_{old}^{-1}(\mathbf{y}-\mathbf{p}_{old})\right) \\ &= \left(\mathbf{X}^\top \mathbf{V}_{old}\mathbf{X}\right)^{-1}\mathbf{X}^\top \mathbf{V}_{old}\mathbf{z}_{old} \qquad (11)\end{aligned}$$

where:

- $\mathbf{z}_{old} := \mathbf{X}\mathbf{w}_{old} + \mathbf{V}_{old}^{-1}(\mathbf{y}-\mathbf{p}_{old})$
- and where $\mathbf{V}_{old}$ and $\mathbf{p}_{old}$ are $\mathbf{V}$ and $\mathbf{p}$, respectively, evaluated at $\mathbf{w}_{old}$.

MLE Via Iteratively Reweighted Least Squares

Now iterate (11) until convergence which typically occurs

- but convergence fails if the two classes are linearly separable.

If classes are linearly separable, then can handle this via regularization.

Note that $\mathbf{w}_{new}$ as given by (11) also satisfies

$$\mathbf{w}_{new} \; = \; \operatorname*{argmin}_{\mathbf{w}} \left(\mathbf{z}_{old} - \mathbf{X}\mathbf{w}\right)^\top \mathbf{V}_{old} \left(\mathbf{z}_{old} - \mathbf{X}\mathbf{w}\right)$$

– a weighted least-squares problem and hence iterating (11) is often called iteratively reweighted least squares.

Even in the 2-class case, logistic regression has many advantages over classification by least squares

- **e.g.** not sensitive to extreme or outlying points.

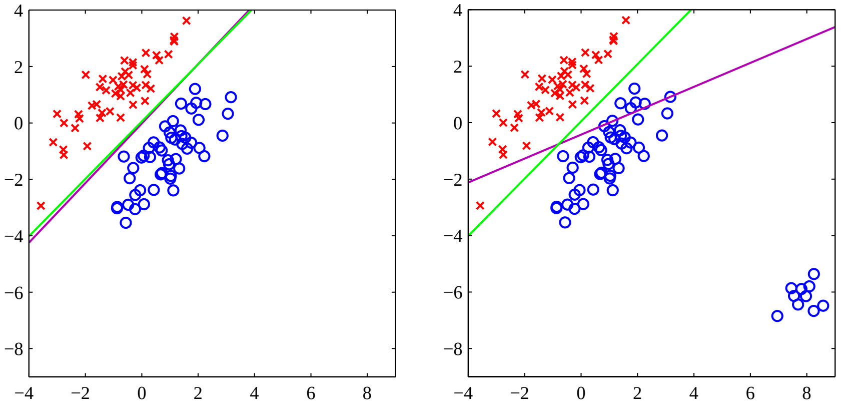


Figure 4.4 from Bishop: The left plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundary found by least squares (magenta curve) and also by the logistic regression model (green curve). The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

Multinomial Logistic Regression

When there are $K > 2$ classes can use multinomial or multi-class logistic regression.

Let $G_1, \ldots, G_K$ denote the K categories. Then assume

$$\mathsf{P}(G_k|\mathbf{x}, \mathbf{w}) \;=\; \frac{\exp\left(\mathbf{w}_k^\top \mathbf{x}\right)}{\sum_j \exp\left(\mathbf{w}_j^\top \mathbf{x}\right)}$$

and as before can use maximum likelihood to estimate the $\mathbf{w}_k$'s.

As with 2-class case, this can be done via an iterative numerical scheme such as Newton-Raphson.

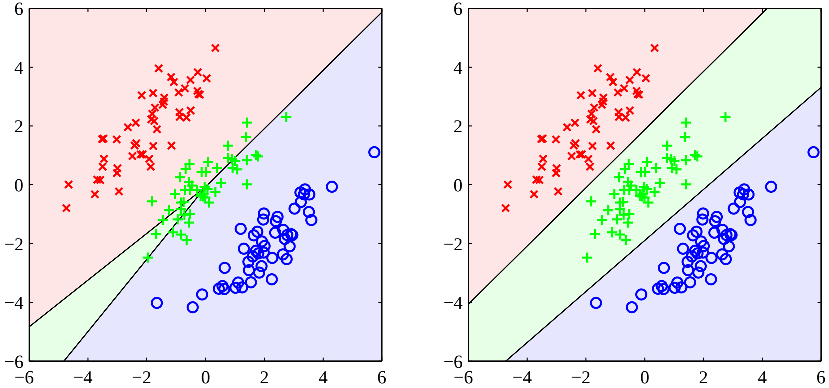


Figure 4.5 from Bishop: Example of a synthetic data set comprising three classes, with training data points denoted in red ($\times$), green ($+$), and blue ($\circ$). Lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. On the left is the result of using a least-squares discriminant. We see that the region of input space assigned to the green class is too small and so most of the points from this class are misclassified. On the right is the result of using logistic regressions showing correct classification of the training data.

Feature Space Expansion With Basis Functions

Have already seen how expanding the feature space can provide much greater flexibility

- **e.g.** using LDA with quadratic basis functions on slide 35

Non-separable data in original feature space may become separable when features are projected into a higher-dimensional space.

In fact the kernel "trick" allows us to project into infinite-dimensional spaces

- will discuss kernel trick later in context of support vector machines (SVMs) and principal component analysis (PCA).

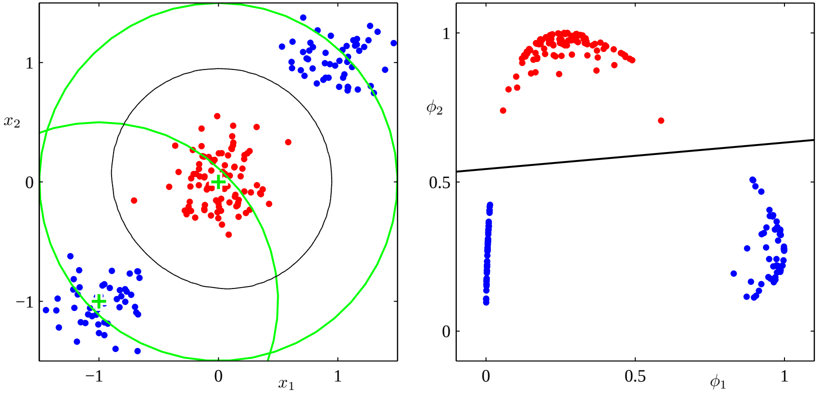


Figure 4.12 from Bishop: Illustration of the role of nonlinear basis functions in linear classification models. The left plot shows the original input space (x_1, x_2) together with data points from two classes labeled red and blue. Two 'Gaussian' basis functions $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$ are defined in this space with centers shown by the green crosses and with contours shown by the green circles. The right-hand plot shows the corresponding feature space (ϕ_1, ϕ_2) together with the linear decision boundary obtained given by a logistic regression model of the form discussed in Section 4.3.2. This corresponds to a nonlinear decision boundary in the original input space, shown by the black curve in the left-hand plot.

A Comparison of Classification Methods

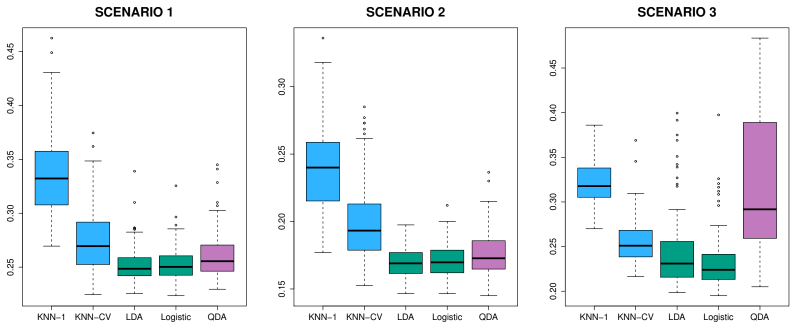


Figure 4.10 from ISLR: Boxplots of the test error rates for each of the linear scenarios described in the main text.

See Section 4.5 of ISLR for description of scenarios.

Here we simply note that in Fig 4.10 the true boundary in each scenario is linear
- and the linear classifiers perform best.

A Comparison of Classification Methods

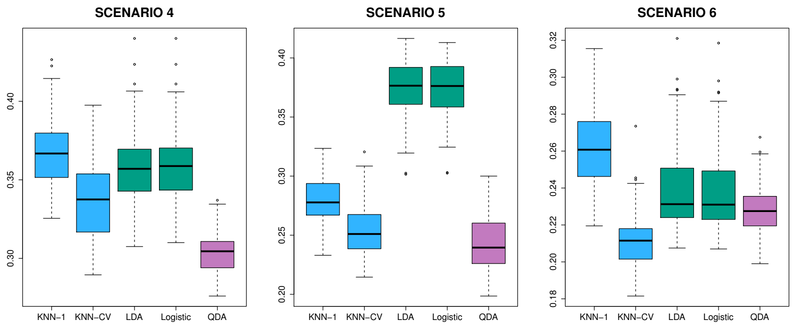


Figure 4.11 from ISLR: Boxplots of the test error rates for each of the non-linear scenarios described in the main text.

Here we note that in Fig 4.11 the true boundary in each scenario is **non-linear**
- and the linear classifiers are no longer the best.

Aside: KNN-CV refers to KNN with K chosen via cross-validation
- to be studied soon!

Appendix: Reduced-Rank LDA

Recall our LDA analysis but now rewrite log-ratio in (6) as

$$\log \frac{\mathsf{P}(G=k|\mathbf{X}=\mathbf{x})}{\mathsf{P}(G=l|\mathbf{X}=\mathbf{x})} = \log \frac{\hat{\pi}_k}{\hat{\pi}_l} - \frac{1}{2}\left(\mathbf{x}-\hat{\boldsymbol{\mu}}_k\right)^\top \hat{\boldsymbol{\Sigma}}^{-1}\left(\mathbf{x}-\hat{\boldsymbol{\mu}}_k\right) + \frac{1}{2}\left(\mathbf{x}-\hat{\boldsymbol{\mu}}_l\right)^\top \hat{\boldsymbol{\Sigma}}^{-1}\left(\mathbf{x}-\hat{\boldsymbol{\mu}}_l\right) \tag{12}$$

Now let $\mathbf{S}_W := \hat{\boldsymbol{\Sigma}} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^\top$ be the eigen decomposition of $\hat{\boldsymbol{\Sigma}}$ where:

- $\boldsymbol{U}$ is an $m \times m$ orthornormal matrix
- $\boldsymbol{D}$ is a diagonal matrix of the positive eigen values of $\hat{\boldsymbol{\Sigma}}$.

Consider now a change of variable with $\mathbf{x}^* := \boldsymbol{D}^{-1/2}\boldsymbol{U}^\top\mathbf{x}$

- so that variance-covariance matrix of $\mathbf{x}^*$ is the identity matrix
- often called sphering the data.

Also implies $\hat{\boldsymbol{\mu}}_k^* := \boldsymbol{D}^{-\frac{1}{2}}\mathbf{U}^\top\hat{\boldsymbol{\mu}}_k$ are now the estimated class centroids and

$$\left(\mathbf{x}-\hat{\boldsymbol{\mu}}_k\right)^\top \mathbf{S}_W^{-1}\left(\mathbf{x}-\hat{\boldsymbol{\mu}}_k\right) = \|\mathbf{x}^* - \hat{\boldsymbol{\mu}}_k^*\|_2^2 .$$

Appendix: Reduced-Rank LDA

Can then rewrite (12) as

$$\begin{aligned}
\log \frac{\mathsf{P}(G=k|\mathbf{X}=\mathbf{x})}{\mathsf{P}(G=l|\mathbf{X}=\mathbf{x})} &= \log \frac{\hat{\pi}_k}{\hat{\pi}_l} - \frac{1}{2}\left(\mathbf{x}^* - \hat{\boldsymbol{\mu}}_k^*\right)^\top \left(\mathbf{x}^* - \hat{\boldsymbol{\mu}}_k^*\right) \\
&\quad + \frac{1}{2}\left(\mathbf{x}^* - \hat{\boldsymbol{\mu}}_l^*\right)^\top \left(\mathbf{x}^* - \hat{\boldsymbol{\mu}}_l^*\right) \\
&= \left(\log \hat{\pi}_k - \frac{1}{2}\|\mathbf{x}^* - \hat{\boldsymbol{\mu}}_k^*\|_2^2\right) \\
&\quad - \left(\log \hat{\pi}_\ell - \frac{1}{2}\|\mathbf{x}^* - \hat{\boldsymbol{\mu}}_\ell^*\|_2^2\right). \qquad (13)
\end{aligned}$$

From (13) it follows that the LDA classifier satisfies

$$k^* = \underset{1 \leq k \leq K}{\operatorname{argmax}} \left\{ \log \hat{\pi}_k - \frac{1}{2}\|\mathbf{x}^* - \hat{\boldsymbol{\mu}}_k^*\|_2^2 \right\}$$

Appendix: LDA in Spherical Coordinates

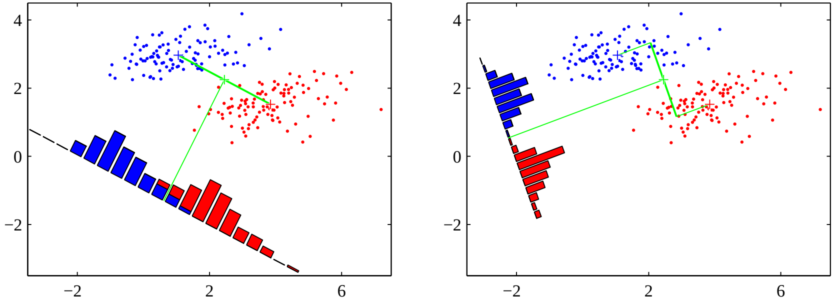


Figure 4.6 from Bishop: The left plot shows samples from two classes (depicted in red and blue) along with the histograms resulting from projection onto the line joining the class means. Note that there is considerable class overlap in the projected space. The right plot shows the corresponding projection based on the Fisher linear discriminant, showing the greatly improved class separation.

Appendix: Reduced-Rank LDA

ML estimates, $\hat{\boldsymbol{\mu}}_k^*$, often called the class centroids and they lie in an $K-1$-dimensional subspace, H_{K-1} say, of $\mathbb{R}^m$.

Now consider an arbitrary point, $\mathbf{x}^*$, and let:

- $\mathbf{z}^* :=$ projection of $\mathbf{x}^*$ onto H_{K-1}
- $\mathbf{y}^* :=$ projection orthogonal to H_{k-1}

Then see that

$$\|\mathbf{x}^* - \hat{\boldsymbol{\mu}}_k^*\|_2^2 = \|\mathbf{y}^*\|_2^2 + \|\mathbf{z}^* - \hat{\boldsymbol{\mu}}_k^*\|_2^2$$

– this is just Pythagoras Theorem!

Note $\mathbf{y}^*$ does not discriminate between classes so can ignore it(!) to obtain

$$k^* = \underset{1 \leq k \leq K}{\operatorname{argmax}} \left\{ \log \hat{\pi}_k - \frac{1}{2} \|\mathbf{z}^* - \hat{\boldsymbol{\mu}}_k^*\|_2^2 \right\}$$

Often $K << m$ so substantial dimension reduction can be achieved.

Appendix: Reduced-Rank LDA

Can also look for an L-dimensional subspace $H_L \subseteq H_{K-1}$ for $L < K-1$ with H_L chosen to maximize between-class variance relative to within-class variance.

Goal is to find dimensions that contribute to dispersion of the $\boldsymbol{\mu}_k^*$'s.

Particular procedure is:

1. First sphere the variables: $\mathbf{x}_i^* := \boldsymbol{D}^{-1/2}\boldsymbol{U}^\top \mathbf{x}_i$ for $i = 1, \ldots, N$.
2. Let M^* be the $m \times K$ matrix of class centroids, $\hat{\boldsymbol{\mu}}_1^*, \ldots, \hat{\boldsymbol{\mu}}_K^*$.
3. Compute the $m \times m$ covariance matrix, $\boldsymbol{B}^*$, of the K centroids.
4. Let $\boldsymbol{B}^* = \boldsymbol{V}\boldsymbol{D_B}\boldsymbol{V}^\top$ be its eigen decomposition with $\boldsymbol{D_B} = \text{diag}(d_1, \ldots, d_{K-1}, 0, \ldots, 0)$ where $d_1 \geq d_2 \geq \cdots \geq d_{K-1} \geq 0$.
5. Let $\boldsymbol{v}_l$ be l^{th} column of $\boldsymbol{V}$ corresponding to eigen value, d_l.
6. Then the l^{th} discriminant or (canonical) variable for $1 \leq l \leq K-1$ is

$$\begin{aligned} c_l &:= \boldsymbol{v}_l^\top \boldsymbol{D}^{-1/2}\boldsymbol{U}^\top \mathbf{x} \\ &= \boldsymbol{v}_l^\top \mathbf{x}^*. \end{aligned}$$

Appendix: Reduced-Rank LDA

These steps can be understood via:

- A principal components analysis (PCA)
- Or by solving a generalized eigen-value problem
  - the original approach of Fisher.

Discriminant function can now be written as

$$k^* = \underset{1 \leq k \leq K}{\operatorname{argmax}} \left\{ \log(\hat{\pi}_k) - \frac{1}{2} \sum_{\ell=1}^{m} \left(\mathbf{v}_l^\top (\mathbf{x}^* - \boldsymbol{\mu}_k^*) \right)^2 \right\}$$

On Slide 55 we plot pairs of canonical variates for the "vowel data" of HTF.

Notice that the lower the coordinates, the more spread out the centroids.

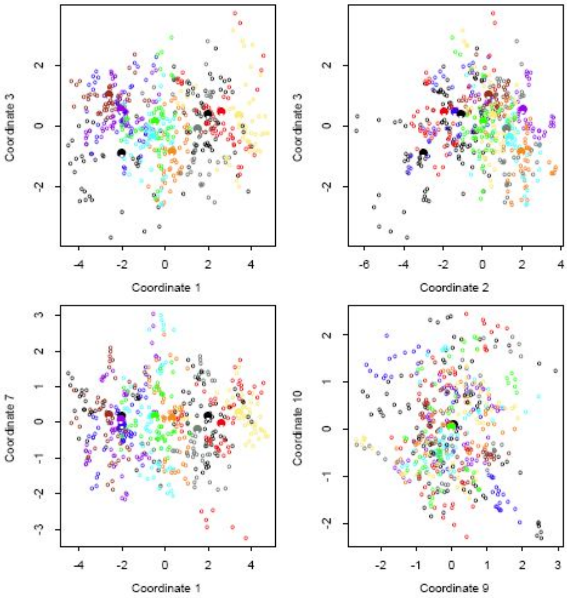


Figure 4.8 from HTF: Four projections onto pairs of canonical variates. Notice that as the rank of the canonical variates increases, the centroids become less spread out. In the lower right panel they appear to be superimposed, and the classes most confused.

Appendix: Classification Using Discriminant Subspaces

Projecting class centroids onto 2- or 3-dimensional discriminant subspaces aids in visualization.

However, also possible that performing classification in H_L (instead of H_{K-1}) for $L < K-1$ might yield a superior classifier. Why?

Figure on Slide 57 plots training and test error against L, the dimension of the discriminant subspace used for LDA, using the vowel data of Hastie et al.

- $K = 11$ classes and $M = 10$ variables
- we see that $L = 2$ yields the best test error.

Figure on Slide 58 shows the corresponding optimal classification regions in H_2.

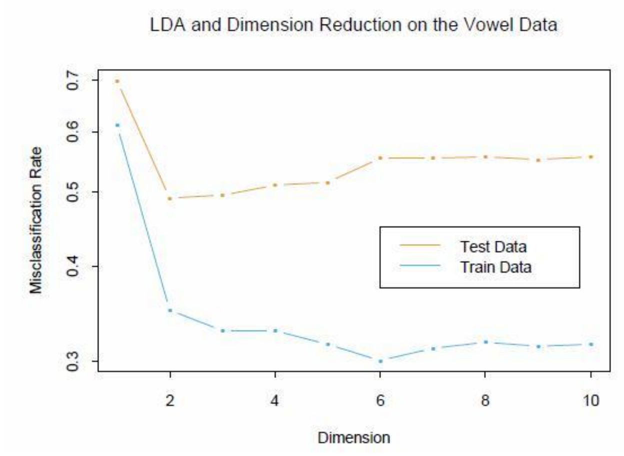


Figure 4.10 from HTF: Training and test error rates for the vowel data, as a function of the dimension of the discriminant subspace. In this case the best error rate is for dimension 2. Figure 4.11 shows the decision boundaries in this space.

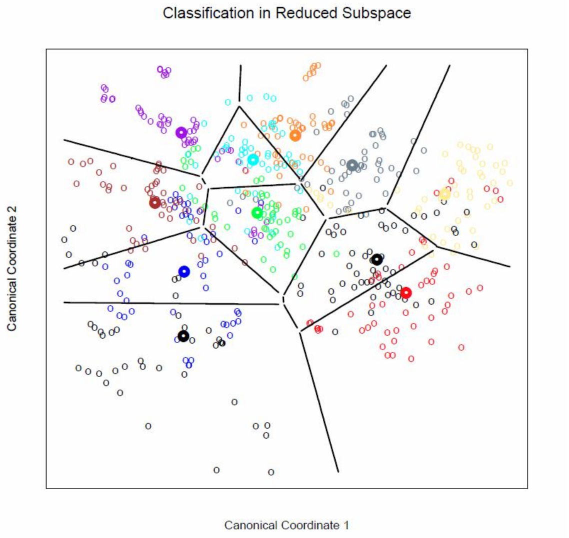


Figure 4.11 from HTF: Decision boundaries for the vowel training data, in the two-dimensional subspace spanned by the first two canonical variates. Note that in any higher-dimensional subspace, the decision boundaries are higher-dimensional affine planes, and could not be represented as lines.

Appendix: Adaptive k-Nearest Neighbors

In high-dimensions, nearest neighbors of a given point are typically **very far** away

- so k-nearest neighbors can perform quite poorly.

e.g. Consider N data-points uniformly distributed in unit cube $[-\frac{1}{2}, \frac{1}{2}]^p$.

Let R be the radius of a 1-NN centered at the origin. Then can be shown that

$$\text{median}(R) \;=\; v_p^{-1/p} \left(1 - \frac{1}{2}^{1/N}\right)^{1/p}$$

where $v_p r^p$ is the volume of the sphere of radius r in p dimensions.

Figure 13.12 from HTF shows median(R) as a function of p for various sample sizes, N

- note that median(R) approaches .5.

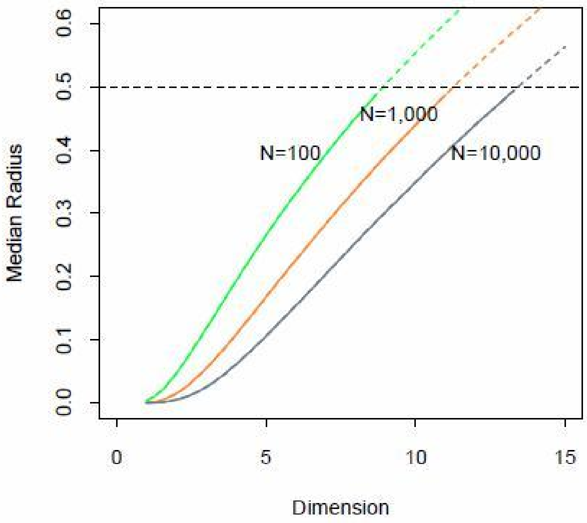


Figure 13.12 from HTF: Median radius of a 1-nearest-neighborhood, for uniform data with N observations in p dimensions.

Appendix: Adaptive k-Nearest Neighbors

Can partly address this problem by allowing metric used at a given point to depend on the point

- yields adaptive k-nearest-neighbors.

Can motivate adaptive k-NN by considering Figure 13.13 from HTF:

- There are 2 classes and 2 features but clearly one of the features is unrelated to class.
- See that class probabilities vary only in the x-direction and not in the y-direction
- So it makes sense to stretch the neighborhood in the y-direction.

Can generalize this insight to apply in higher dimensions by adapting the metric at each point.

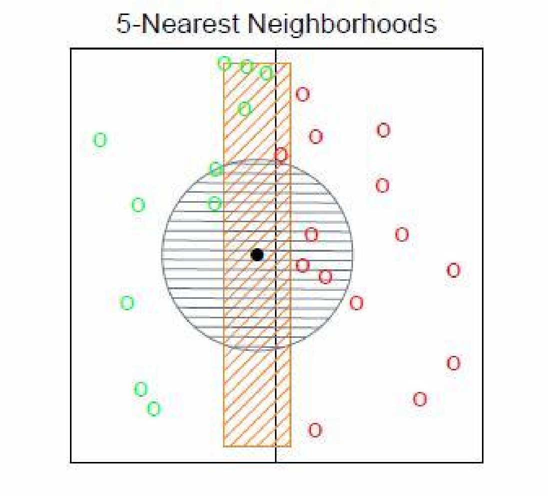


Figure 13.13 from HTF: The points are uniform in the cube, with the vertical line separating class red and green. The vertical strip denotes the 5-nearest-neighbor region using only the horizontal coordinate to find the nearest-neighbors for the target point (solid dot). The sphere shows the 5-nearest-neighbor region using both coordinates, and we see in this case it has extended into the class-red region (and is dominated by the wrong class in this instance).

Appendix: Adaptive k-Nearest Neighbors

The discriminant adaptive nearest neighbor (DANN) rule works as follows:

1. At a query point $\mathbf{x}_0$, form the neighborhood of the n nearest points
   - will use these n points to determine the metric used at $\mathbf{x}_0$.
2. Using these n points compute:
   (i) $\boldsymbol{W} = \sum_{k=1}^{K} \pi_k \boldsymbol{W}_k$, the pooled within-class covariance matrix.
   (ii) $\boldsymbol{B} = \sum_{k=1}^{K} \pi_k (\bar{\mathbf{x}}_k - \bar{\mathbf{x}})(\bar{\mathbf{x}}_k - \bar{\mathbf{x}})^\top$, the between-class covariance matrix.
3. Define
$$\boldsymbol{\Sigma} \; := \; \boldsymbol{W}^{-1/2} \left[\boldsymbol{W}^{-1/2} \boldsymbol{B} \boldsymbol{W}^{-1/2} \; + \; \epsilon \boldsymbol{I} \right] \boldsymbol{W}^{-1/2}. \qquad (14)$$
4. Now classify $\mathbf{x}_0$ using k-NN with $k < n$ and the metric
$$D(\mathbf{x}, \mathbf{x}_0) \; = \; (\mathbf{x} - \mathbf{x}_0)^\top \boldsymbol{\Sigma} (\mathbf{x} - \mathbf{x}_0).$$

HTF suggest using $n = 50$ and $\epsilon = 1$.

Question: What is the geometric interpretation behind (14)?

Question: Are there any obvious down-sides to DANN?

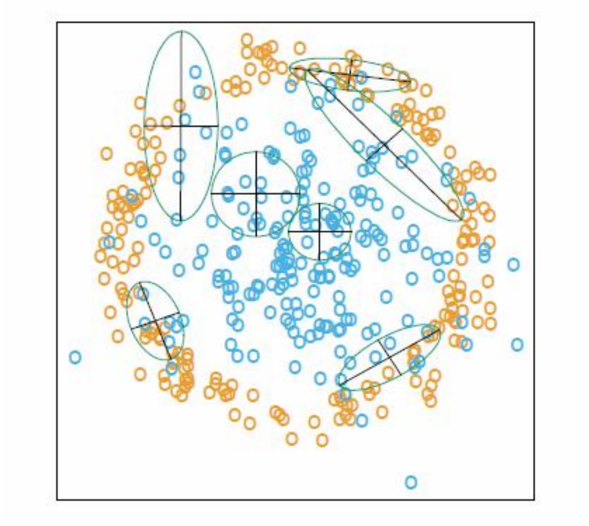


Figure 13.14 from HTF: Neighborhoods found by the DANN procedure, at various query points (centers of the crosses). There are two classes in the data, with one class surrounding the other. 50 nearest-neighbors were used to estimate the local metrics. Shown are the resulting metrics used to form 15-nearest-neighborhoods.

Appendix: Adaptive k-Nearest Neighbors

Question: What is the geometric interpretation behind (14)?

Solution: (14) arises naturally from the following steps:

1. First sphere the data using the pooled within-class variance, $\mathbf{W}$, so that $\mathbf{x}_i \to \mathbf{W}^{-1/2}\mathbf{x}_i$ for each point $\mathbf{x}_i$ in the n-neighborhood of $\mathbf{x}_0$.
2. Now note that $\mathbf{B}^* := \mathbf{W}^{-1/2}\mathbf{B}\mathbf{W}^{-1/2}$ is the between-class covariance matrix under these new coordinates.
3. The metric in (14) now reduces to

$$\begin{aligned} D(\mathbf{x}, \mathbf{x}_0) &= (\mathbf{x} - \mathbf{x}_0)^\top \boldsymbol{W}^{-1/2} \left[\boldsymbol{W}^{-1/2} \boldsymbol{B} \boldsymbol{W}^{-1/2} + \epsilon \boldsymbol{I} \right] \boldsymbol{W}^{-1/2} (\mathbf{x} - \mathbf{x}_0) \\ &= (\mathbf{x}^* - \mathbf{x}_0^*)^\top \left[\boldsymbol{B}^* + \epsilon \boldsymbol{I} \right] (\mathbf{x}^* - \mathbf{x}_0^*). \end{aligned} \tag{15}$$

Intuitively, (15) now makes sense.

e.g. Suppose $(\mathbf{x}^* - \mathbf{x}_0^*) \approx a\mathbf{e}$ where $\mathbf{e}$ is an eigen vector of $\mathbf{B}^*$ with small eigen value. Then (ignoring $\epsilon \boldsymbol{I}$ term) we see that (15) will be small so $\mathbf{x}^*$ more likely to be included among the k nearest neighbors (and to have same class as $\mathbf{x}_0$).

Role of $\epsilon \boldsymbol{I}$ is simply to round any infinite strips to ellipses in new coordinates.