BUSI97287: Advanced Machine Learning

Classification: Review and Some New Topics

Martin Haugh

Imperial College Business School
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.

Additional References: Bishop's PRML and HTF's TESL.

Required Reading: Sections 4.1, 4.2, 4,4 and 4.5 of *ISLR* by James, Witten, Hastie and Tibshirani

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.
- 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 **x** to denote vector of inputs and G or Y to denote category or class.

Generative Classification Algorithms

Generative methods model $\mathsf{P}(\mathbf{x}, G)$ and then use $\widehat{G}(\mathbf{x})$ as a classifier where

$$\begin{aligned} \widehat{G}(\mathbf{x}) \;:=\; \underset{G}{\operatorname{argmax}}\, \widehat{\mathsf{P}}(G \mid \mathbf{x}) \;&=\; \underset{G}{\operatorname{argmax}}\, \frac{\widehat{\mathsf{P}}(\mathbf{x} \mid G)\widehat{\mathsf{P}}(G)}{\widehat{\mathsf{P}}(\mathbf{x})} \\ &=\; \underset{G}{\operatorname{argmax}}\, \widehat{\mathsf{P}}(\mathbf{x} \mid G)\widehat{\mathsf{P}}(G) \end{aligned} \tag{1}$$

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

- LDA and QDA assume Gaussian densities for $\widehat{\mathsf{P}}(\mathbf{x} \mid G)$.
- Naive Bayes assumes

$$\mathsf{P}(\mathbf{x} \mid G) \;=\; \prod_j \mathsf{P}(x_j \mid G)$$

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 $\mathsf{P}(G \mid \mathbf{x})$ directly

- examples include logistic regression and Bayesian logistic regression.

Discriminative methods may also focus on minimizing the expected classification error directly without ever even modeling $\mathsf{P}(\mathbf{x}, G)$ or $\mathsf{P}(G \mid \mathbf{x})$.

Examples include:

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

The 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 its class G.

$L(G, \widehat{G}(\mathbf{x}))$ denotes loss function when true class is G and we use classifier $\widehat{G}(\mathbf{x})$.

e.g. A common loss function is the $0-1$ loss function where

$$\begin{aligned} L(G, \widehat{G}(\mathbf{x})) &= \begin{cases} 0, & \text{if } G = \widehat{G}(\mathbf{x}) \\ 1, & \text{otherwise} \end{cases} \\ &= 1_{\{G \neq \widehat{G}(\mathbf{x})\}} \end{aligned}$$

Assume now there are a total of K classes.

The Optimal Bayes Classifier

The expected prediction error (EPE) associated with $\widehat{G}$ is

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

Wish to minimize $\mathsf{EPE}_{\widehat{G}}$ w.r.t $\widehat{G}$. Let G^* be optimal so that

$$\begin{aligned}
\mathsf{EPE}_{G^*} &= \min_{\widehat{G}} \mathsf{EPE}_{\widehat{G}} \\
&= \min_{\widehat{G}} \mathsf{E}_{\mathbf{x}}\left[\sum_{k=1}^{K} L(G_k,\widehat{G}(\mathbf{x}))\,\mathsf{P}(G_k \mid \mathbf{x})\right] \\
&= \mathsf{E}_{\mathbf{x}}\left[\min_{\widehat{G}(\mathbf{x})} \sum_{k=1}^{K} L(G_k,\widehat{G}(\mathbf{x}))\,\mathsf{P}(G_k \mid \mathbf{x})\right] \qquad (2)
\end{aligned}$$

The Optimal Bayes Classifier

If we assume the $0-1$ loss function then (2) reduces to

$$\mathsf{EPE}_{G^*} \quad = \quad \mathsf{E}_{\mathbf{x}}\left[\min_{\widehat{G}(\mathbf{x})} \sum_{k=1}^{K} 1_{\{G_k \neq \widehat{G}(\mathbf{x})\}} \mathsf{P}(G_k \mid \mathbf{x})\right]. \tag{3}$$

Can solve minimisation problem inside expectation in (3) to obtain

$$\begin{aligned} G^*(\mathbf{x}) &= \underset{\widehat{G}(\mathbf{x})}{\operatorname{argmin}} \sum_{k=1}^{K} 1_{\{G_k \neq \widehat{G}(\mathbf{x})\}} \mathsf{P}(G_k \mid \mathbf{x}) \\ &= \underset{\widehat{G}(\mathbf{x})}{\operatorname{argmin}} \left[1 - \mathsf{P}(\widehat{G}(\mathbf{x}) \mid \mathbf{x})\right] \\ &= \underset{\widehat{G}(\mathbf{x})}{\operatorname{argmax}} \mathsf{P}(\widehat{G}(\mathbf{x}) \mid \mathbf{x}) \end{aligned} \tag{4}$$

– so optimal to classify to the most probable class given $\mathbf{x}$.

The Optimal Bayes Classifier

Call G^* the Bayes classifier and EPE_{G^*} 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.

Generative classifiers also use (3) (in the case of 0-1 loss) except they use approximations / estimates for $\mathsf{P}(G(\mathbf{x}) \mid \mathbf{x})$

e.g. Naive Bayes, LDA and QDA.

Linear Discriminant Analysis (LDA)

Recall a generative model classifies according to

$$\widehat{G}(\mathbf{x}) \;:=\; \underset{G}{\operatorname{argmax}}\, \widehat{\mathsf{P}}(G \mid \mathbf{x}) \;=\; \underset{G}{\operatorname{argmax}}\, \widehat{\mathsf{P}}(\mathbf{x} \mid G)\widehat{\mathsf{P}}(G). \tag{5}$$

LDA is a generative model which assumes that

$$\mathsf{P}(\mathbf{x} \mid G = k) \sim \mathsf{MVN}\left(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}\right)$$

That is, LDA assumes the class-conditional densities $f_k(\mathbf{x})$ are multivariate normal with a common covariance matrix so that

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

Writing $\widehat{\pi}_k$ for $\widehat{\mathsf{P}}(G = k)$, (5) then reduces to

$$\widehat{G}(\mathbf{x}) = \underset{k}{\operatorname{argmax}}\; \widehat{f}_k(\mathbf{x})\, \widehat{\pi}_k. \tag{6}$$

Estimating LDA Model Parameters

Suppose we have N training points $(\mathbf{x}_1, g_1), \ldots, (\mathbf{x}_N, g_N)$.

We use these points to construct the MLE estimates of $f_k(\cdot)$ and π_k to obtain

$$\begin{aligned} \widehat{\pi}_k &= N_k/N \\ \widehat{\boldsymbol{\mu}}_k &= \sum_{g_i=k} \mathbf{x}_i/N_k \\ \widehat{\boldsymbol{\Sigma}} &= \frac{\sum_{k=1}^{K} \sum_{g_i=k} (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)^\top}{N-K} \end{aligned} \tag{7}$$

where $N_k = \#$ of training points in class k.

Note: Have assumed a common covariance matrix so must use a pooled estimator for $\boldsymbol{\Sigma}$ in (7).

Linear Discriminant Analysis (LDA)

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

$$\begin{aligned}
\log \frac{\widehat{\mathsf{P}}(G = k \mid \mathbf{x})}{\widehat{\mathsf{P}}(G = l \mid \mathbf{x})} &= \log \frac{\widehat{\mathsf{P}}(\mathbf{x} \mid G = k)\widehat{\mathsf{P}}(G = k)}{\widehat{\mathsf{P}}(\mathbf{x} \mid G = l)\widehat{\mathsf{P}}(G = l)} \\
&= \log \frac{\widehat{f}_k(\mathbf{x})\,\widehat{\pi}_k}{\widehat{f}_l(\mathbf{x})\,\widehat{\pi}_l} \\
&= \log \frac{\widehat{\pi}_k}{\widehat{\pi}_l} - \frac{1}{2}\,(\widehat{\boldsymbol{\mu}}_k + \widehat{\boldsymbol{\mu}}_l)^\top\,\widehat{\boldsymbol{\Sigma}}^{-1}\,(\widehat{\boldsymbol{\mu}}_k - \widehat{\boldsymbol{\mu}}_l) \\
&\quad + \mathbf{x}^\top\widehat{\boldsymbol{\Sigma}}^{-1}\,(\widehat{\boldsymbol{\mu}}_k - \widehat{\boldsymbol{\mu}}_l) \qquad (8) \\
&= \left(\log\widehat{\pi}_k - \frac{1}{2}\widehat{\boldsymbol{\mu}}_k^\top\widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\boldsymbol{\mu}}_k + \mathbf{x}^\top\widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\boldsymbol{\mu}}_k\right) \\
&\quad - \left(\log\widehat{\pi}_l - \frac{1}{2}\widehat{\boldsymbol{\mu}}_l^\top\widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\boldsymbol{\mu}}_l + \mathbf{x}^\top\widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\boldsymbol{\mu}}_l\right). \qquad (9)
\end{aligned}$$

Linear Discriminant Analysis (LDA)

Based on (9) can define the linear discriminant functions

$$\delta_k(\mathbf{x}) := \mathbf{x}^\top \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_k + \underbrace{\log \widehat{\pi}_k - \frac{1}{2} \widehat{\boldsymbol{\mu}}_k^\top \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_k}_{k^{th} \text{ intercept}} \tag{10}$$

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

Then (9) reduces to

$$\log \frac{\widehat{\mathsf{P}}(G = k \mid \mathbf{x})}{\widehat{\mathsf{P}}(G = l \mid \mathbf{x})} = \delta_k(\mathbf{x}) - \delta_l(\mathbf{x})$$

so that the LDA classifier of (6) reduces to

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

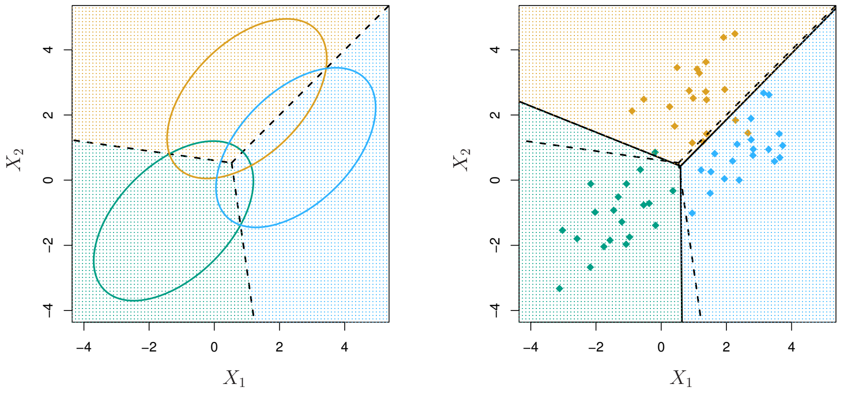


Figure 4.6 from ISLR: An example with three classes. The observations from each class are drawn from a multivariate Gaussian distribution with $p = 2$, with a class-specific mean vector and a common covariance matrix. Left: Ellipses that contain 95% of the probability for each of the three classes are 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?

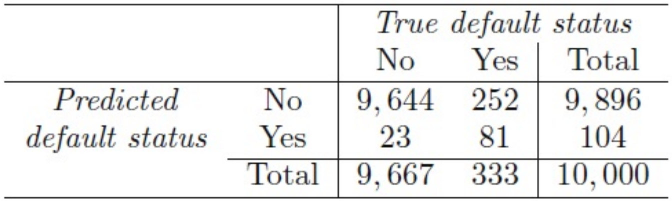
- note training error rate will usually be biased low and therefore lower than test / generalization error
- for comparison, how well does the useless classifier that always predicts non-default do?

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



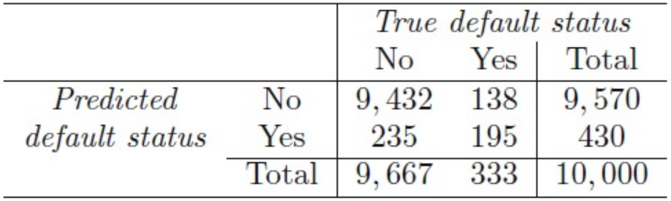
		True default status		
		No	Yes	Total
Predicted default status	No	$9,644$	252	$9,896$
	Yes	23	81	104
	Total	$9,667$	333	$10,000$

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%.

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

e.g. In Table 4.5 we can "predict" default if the LDA model has

$$\mathsf{P}\left(\text{default} = \text{Yes} \mid \mathbf{x}\right) > 0.2.$$

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

The Tradeoff from Modifying the Threshold

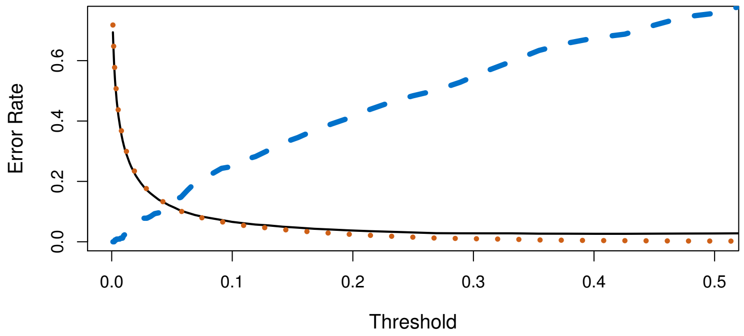
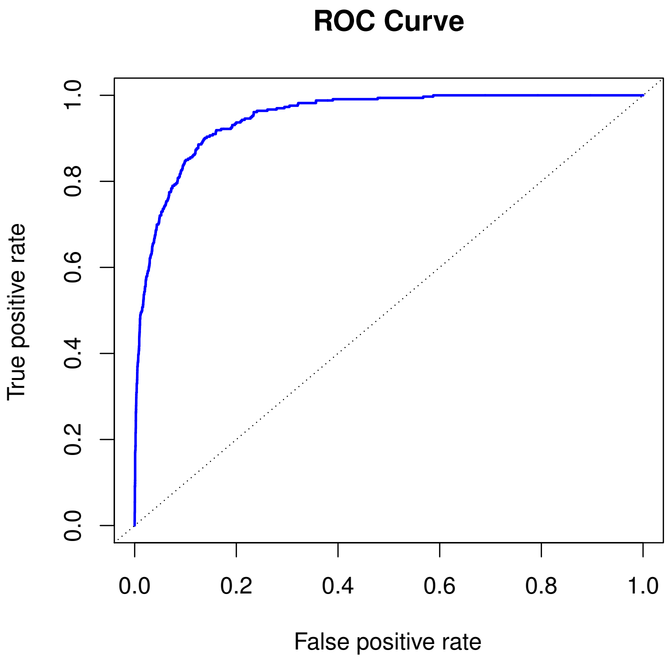


Figure 4.7 from ISLR: For the Default data set, error rates are shown as a function of the threshold value for the posterior probability that is used to perform the assignment. The black solid line displays the overall error rate. The blue dashed line represents the fraction of defaulting customers that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

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

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

QDA classifier is then

$$\widehat{G}(\mathbf{x}) = \underset{k}{\operatorname{argmax}}\;\delta_k(\mathbf{x})$$

So decision boundaries between each pair of classes then given by quadratic functions of $\mathbf{x}$.

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

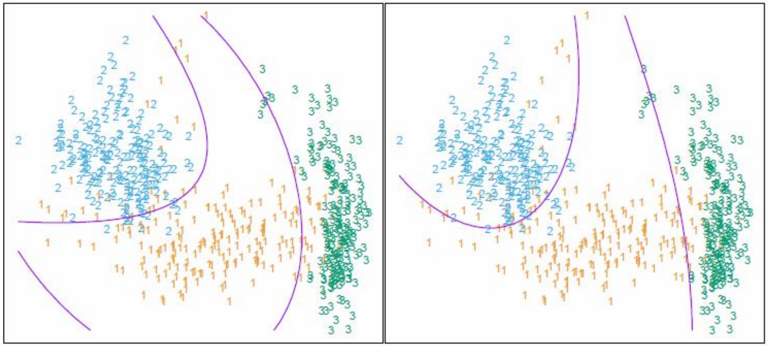


Figure 4.6 from HTF: Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space X_1, X_2, X_1X_2, X_1^2, X_2^2). The right plot shows the quadratic decision boundaries found by QDA. The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

LDA v LDA with Quadratic Predictors

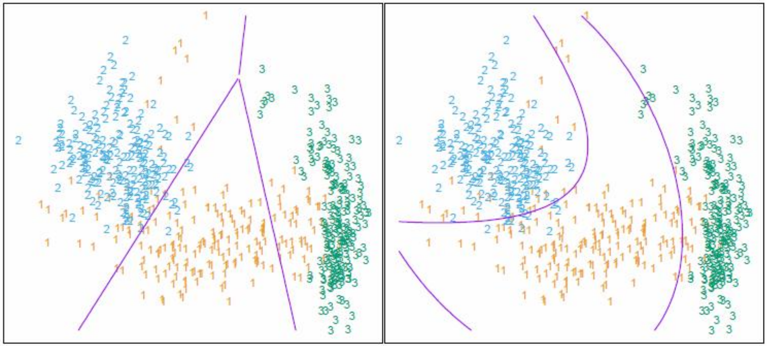


Figure 4.1 from HTF: The left plot shows some data from three classes, with linear decision boundaries found by linear discriminant analysis. The right plot shows quadratic decision boundaries. These were obtained by finding linear boundaries in the five-dimensional space X_1, X_2, X_1X_2,X_1^2, X_2^2. Linear inequalities in this space are quadratic inequalities in the original space.

LDA v QDA

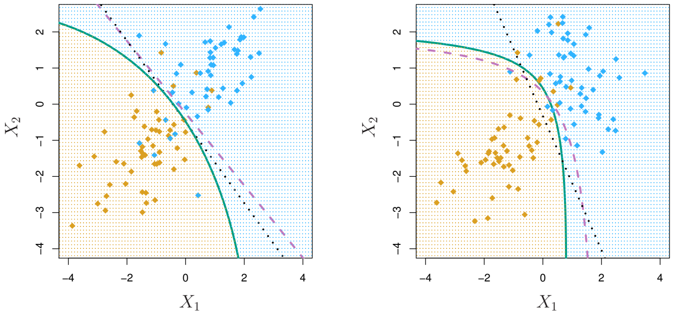


Figure 4.9 from ISLR: Left: The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2$. The shading indicates the QDA decision rule. Since the Bayes decision boundary is linear, it is more accurately approximated 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
- this is the bias-variance tradeoff story of supervised learning.

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 27

Non-separable data in original feature space may become separable when new features are used in an alternative (often higher-dimensional) space.

e.g. See Figure 4.12 from Bishop on slide 32:

- For $i = 1, 2$ the Gaussian basis functions are

$$\phi_i(\mathbf{x}) := e^{-(\mathbf{x}-\boldsymbol{\mu}_i)^\top(\mathbf{x}-\boldsymbol{\mu}_i)} \in (0, 1)$$

  with $\boldsymbol{\mu}_1^\top = [-1 \ \ -1]$ and $\boldsymbol{\mu}_2^\top = [0 \ \ 0]$.

- Blue points in top r.h. corner of left panel correspond to blue points in bottom l.h. corner of right panel.

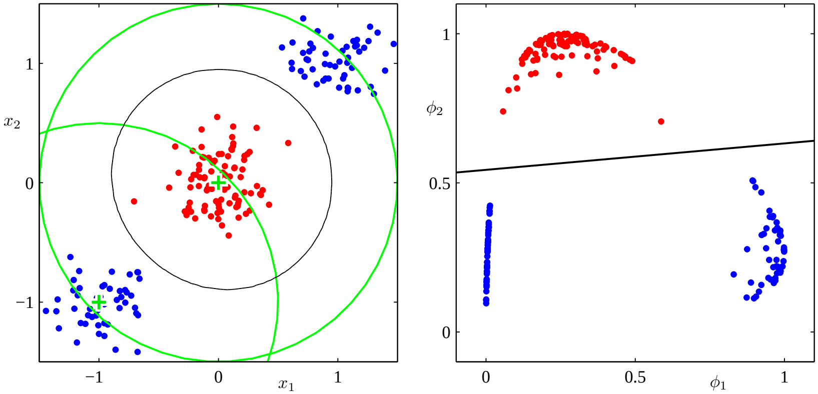


Figure 4.12 from Bishop: Illustration of the role of nonlinear basis functions in linear classification models. The left plot shows the original input space (x_1, x_2) together with data points from two classes labeled red and blue. Two 'Gaussian' basis functions $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$ 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

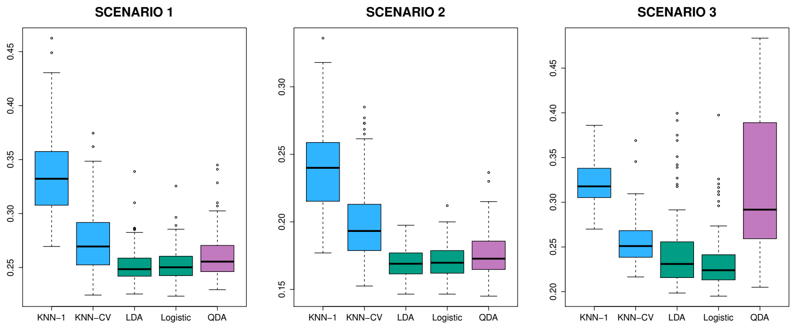


Figure 4.10 from ISLR: Boxplots of the test error rates for each of the linear scenarios described in the main text.

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

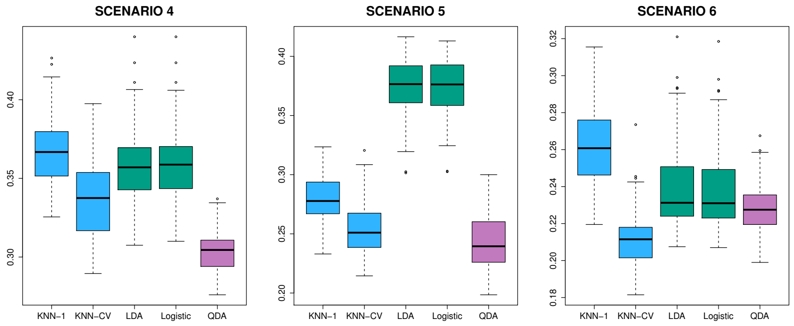


Figure 4.11 from ISLR: Boxplots of the test error rates for each of the non-linear scenarios described in the main text.

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.

Appendix: Naive Bayes

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

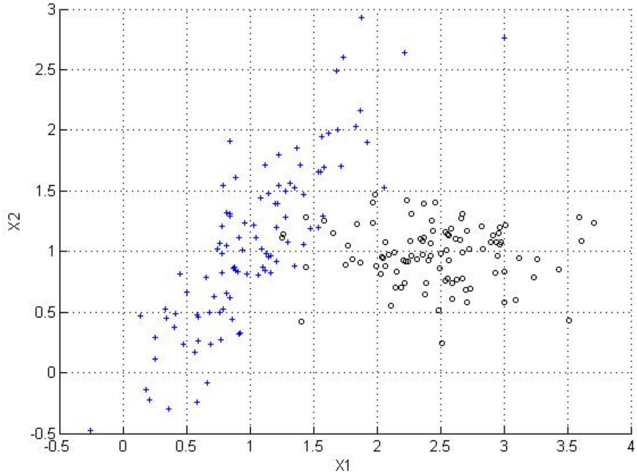
$$\mathsf{P}(\mathbf{x} \mid G) \;=\; \prod_j \mathsf{P}(x_j \mid G) \tag{11}$$

- so the features are independent conditional on G.

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

$$\begin{aligned} \widehat{G}(\mathbf{x}) &= \operatorname*{argmax}_{G \in \mathcal{G}} \widehat{\mathsf{P}}(G \mid \mathbf{x}) \\ &= \operatorname*{argmax}_{G \in \mathcal{G}} \widehat{\mathsf{P}}(G)\widehat{\mathsf{P}}(\mathbf{x} \mid G) \\ &= \operatorname*{argmax}_{G \in \mathcal{G}} \widehat{\mathsf{P}}(G) \prod_i \widehat{\mathsf{P}}(x_i \mid G). \end{aligned}$$

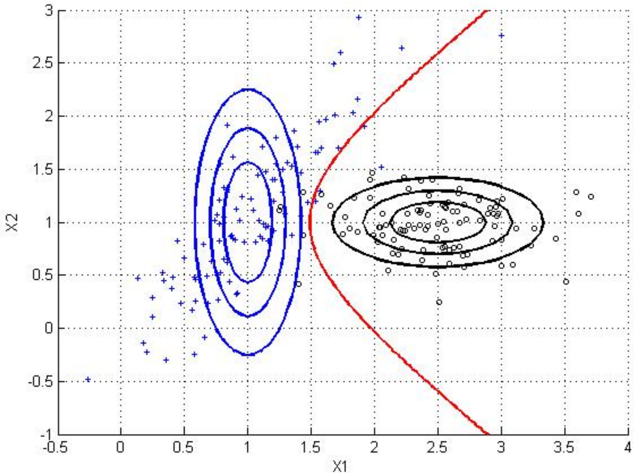
Appendix: Naive Bayes in Action



- There are 2 equiprobable classes and the class-conditional densities are bivariate normal.

Question: The assumptions of naive Bayes do not apply. Why?

Appendix: 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 with points to the left (right) of the curve classified to blue (black).

Appendix: Naive Bayes

Assumption (11) is strong and generally unrealistic

- but when $\mathbf{x}$ high-dimensional estimating $\mathsf{P}(\mathbf{x} \mid G)$ accurately not possible unless very large amounts of data are available.

e.g. Suppose $\mathbf{x} \in \{0,1\}^p$. How many parameters must we estimate to estimate $\mathsf{P}(\mathbf{x} \mid G)$ if we don't assume (11)? And how many if we do assume (11)?

Assumption (11) therefore makes estimation much easier.

Even when (11) not justified, naive Bayes still often works very well in practice!

- **e.g.** see its performance on slide 39
- this is an example of trading bias off against variance - a common theme in supervised learning.

Appendix: 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 $\mathrm{P}(\text{spam} \mid \mathbf{x}) \propto \mathrm{P}(\text{spam}) \prod_{\text{word} \in \text{email}} \mathrm{P}(\text{word} \mid \text{spam})$.

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.

Appendix: Logistic Regression

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

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

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

Follows that

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

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

$$l(\mathbf{w}) = \sum_{i=1}^{N} \left(y_i \mathbf{w}^\top \mathbf{x}_i - \log\left(1 + \exp(\mathbf{w}^\top \mathbf{x}_i)\right)\right) \tag{12}$$

Appendix: MLE Estimation

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

$$\frac{\partial l(\mathbf{w})}{\partial \mathbf{w}} \;=\; \sum_{i=1}^{N} \mathbf{x}_i(y_i - p_i(\mathbf{w})) \;=\; 0 \tag{13}$$

– 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 (13) iteratively using Newton-Raphson steps

$$\mathbf{w}_{new} \;=\; \mathbf{w}_{old} - \left(\frac{\partial^2 l(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^\top}\right)^{-1} \frac{\partial l(\mathbf{w})}{\partial \mathbf{w}} \tag{14}$$

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

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

Appendix: 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 (14) becomes

$$\begin{aligned}\mathbf{w}_{new} &= \mathbf{w}_{old} + \left(\mathbf{X}^\top \mathbf{V}_{old}\mathbf{X}\right)^{-1}\mathbf{X}^\top(\mathbf{y}-\mathbf{p}_{old}) \\ &= \left(\mathbf{X}^\top \mathbf{V}_{old}\mathbf{X}\right)^{-1}\mathbf{X}^\top \mathbf{V}_{old}\left(\mathbf{X}\mathbf{w}_{old} + \mathbf{V}_{old}^{-1}(\mathbf{y}-\mathbf{p}_{old})\right) \\ &= \left(\mathbf{X}^\top \mathbf{V}_{old}\mathbf{X}\right)^{-1}\mathbf{X}^\top \mathbf{V}_{old}\mathbf{z}_{old} \qquad (15)\end{aligned}$$

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 $\mathbf{w}_{old}$.

Appendix: MLE Via Iteratively Reweighted Least Squares

Now iterate (15) 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 (15) also satisfies

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

– a weighted least-squares problem and hence iterating (15) is often called iteratively reweighted least squares.

Appendix: 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

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

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.