Classification

Slides on *Introduction to Statistical Learning*, Chapter 4

Calvin Khor

January 2024

See the Typst source: https://typst.app/project/r9bqFQZQAD5m2ttHgsjSu_

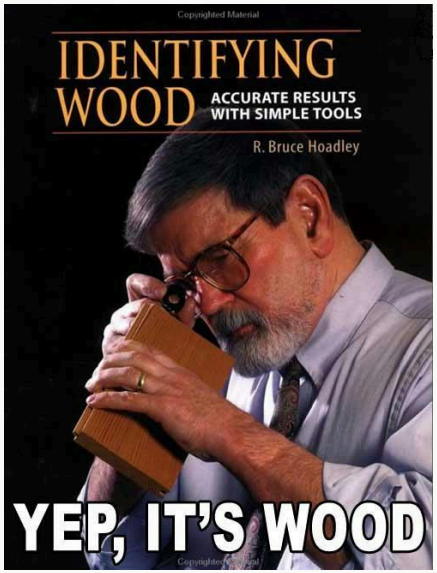
Table of contents

Overview of Classification

What?

- (supervised) **classification**: prediction of a **qualitative** response variable Y
- as opposed to regression, which predicts a *quantitative* response.

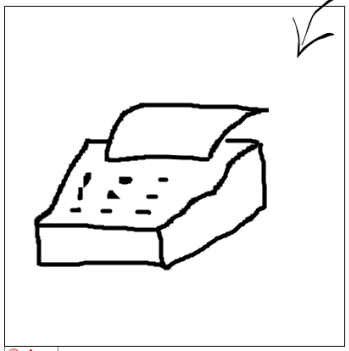
Examples



Detexify^2 - LaTeX symbol classifier

classify | symbols | blog

Draw here!



clear

What is this?

Anyone who works with LaTeX knows how time-consuming it can be to find a symbol in symbols-a4.pdf that you just can't memorize. Detexify is an attempt to simplify this search.

How do I use it?

Just draw the symbol you are looking for into the square area above and look what happens!

My symbol isn't found!

The symbol may not be trained enough or it is not yet in the list of supported symbols. In the first case you can do the training yourself. ~~In the second case just drop me a line (danishkirel@gmail.com)!~~ (I don't really find the

Did this help? Hosting Detexify costs money and if it helps you may consider helping to pay the hosting bill. DONATE pledge.com $1,768.29 Raised!

Score: 0.374649673071808
`\usepackage{ marvosym }`
`\Faxmachine`
textmode

Score: 0.37850351734522
`\usepackage{ marvosym }`
`\Sun`
textmode

Score: 0.37850351734522
`\usepackage{ wasysym }`
`\aquarius`
textmode & mathmode

Score: 0.37850351734522
`\usepackage{ tipa }`
`\textsoftsign`
textmode

Score: 0.383688035121593
`\usepackage{ skull }`
`\skull`
mathmode

The symbol is not in the list? Select from the complete list!

up ↑

Book's Examples

- Detecting illness from symptoms
- Detecting fraud
- Understanding effects of genes on a disease

Real Examples (Kernelised: see Wikipedia)

Improving Kernel Fisher Discriminant Analysis for Face Recognition

Qingshan Liu, Hanqing Lu, and Songde Ma, *Senior Member, IEEE*

***Abstract*—This work is a continuation and extension of our previous research where Kernel Fisher Discriminant Analysis (KFDA), a combination of the kernel trick with Fisher Linear Discriminant Analysis (FLDA), was introduced to represent facial features for face recognition. The main contributions of this work are three-fold to further improve the performance of KFDA. First, a new kernel function, called the Cosine kernel, is proposed to increase the discriminating capability of the original polynomial kernel function. Second, a geometry-based feature vector selection scheme is adopted to reduce the computational complexity of KFDA. Third, a variant of the nearest feature line classifier is employed to further enhance the recognition performance as it can produce virtual samples to make up for the shortage of training samples. Experiments are carried out on a mixed database with 125 persons and 970 images and demonstrate the effectiveness of improvements.**

***Index Terms*—Boosting, face recognition, Kernel Fisher Discriminant Analysis (KFDA), nearest feature line (NFL).**

I. Introduction

FACE recognition has received extensive attention within the past two decades because of its potential applications in many fields, such as identity authentication, information security, surveillance, human–computer interface, and so on. How to

the samples. In Eigenface [3], principal components are called Eigenface images and a face image is represented by a linear combination of these Eigenface images. This is optimal in terms of representation and reconstruction, but not for discriminating one face class from others. Fisher linear discriminant analysis (FLDA) seeks to find a linear transformation that maximizes the between-class scatter and minimizes the within-class scatter, in order to separate one class from others. As for face recognition, because there are not enough samples to make within-class scatter nonsingular, some tricks are demanded in using FLDA for face recognition [4]–[8]. For example, Fisherface [4], [5] applies PCA for dimension reduction at first and then uses FLDA to extract and represent facial features. Belhumeur [4] compared Fisherface with Eigenface on the Harvard and Yale face databases and showed that FLDA performed better than PCA did, especially under illumination variation. FLDA was also given a good evaluation score under the FERET testing framework [5], [9]. Independent Component Analysis (ICA) is another linear subspace analysis method, which separates the high-order moments of the input data besides the second-order moments in PCA. Bartlett [10] first applied it to face recognition, but Moghaddam [11] reported that it just gave the same

"[...] a new kernel function, called the cosine kernel, [...]" DOI link

ELSEVIER

Computer Physics Communications

Volume 106, Issue 3, November 1997, Pages 230-236

An iterative nonlinear discriminant analysis program: IDA 1.0

T.G.M. Malmgren

Show more

+ Add to Mendeley Share Cite

https://doi.org/10.1016/S0010-4655(97)00100-8 Get rights and content

Abstract

A Fortran 77 package of an iterated discriminant analysis multi-background and signal recognition, IDA 1.0, is presented. The primary target is the high energy physics community, but it is general enough to be used in many classification application areas. The package can easily be used as is or may readily be modified for alternative purposes and optimizations.

used for Higgs Boson. DOI Link

Aside: Detexify and its training data is open source

Detexify uses KNN:

"The classification in Detexify is implemented with the k-nearest neighbor algorithm (almost). To classify an unknown pattern the training data is first sorted based on the euclidean distance in the feature space to this pattern. Usually then the k nearest neighbours are examined and the pattern is probably of the class most common among them but since I want 5 entries in the hit list of symbols nearest neighbours are examined until symbols of 5 different classes are found. The score of a symbol (class) is then the number of occurences among these nearest neighbours."

Links: training data, Haskell backend, frontend

What we cover and what we don't

- ✅ Logistic Regression
- ✅ Linear Discriminant Analysis (LDA)
- ✅ Quadratic Discriminant Analysis (QDA)
- ✅ Naïve Bayes
- ✅ Generalised Linear Models (not really classification)

- ❌ Generalised Additive Models
  - that's chapter 7
- ❌ Tree methods
  - that's chapter 8
- ❌ Support Vector Machines
  - that's chapter 9
- ❌ Clustering
  - that's unsupervised (chapter 12)
- ❌ deep learning
  - that's chapter 10

Why not linear regression?

Linear Regression for categorical variables?

- Encoding of categories enforces an order (possibly even weighted)

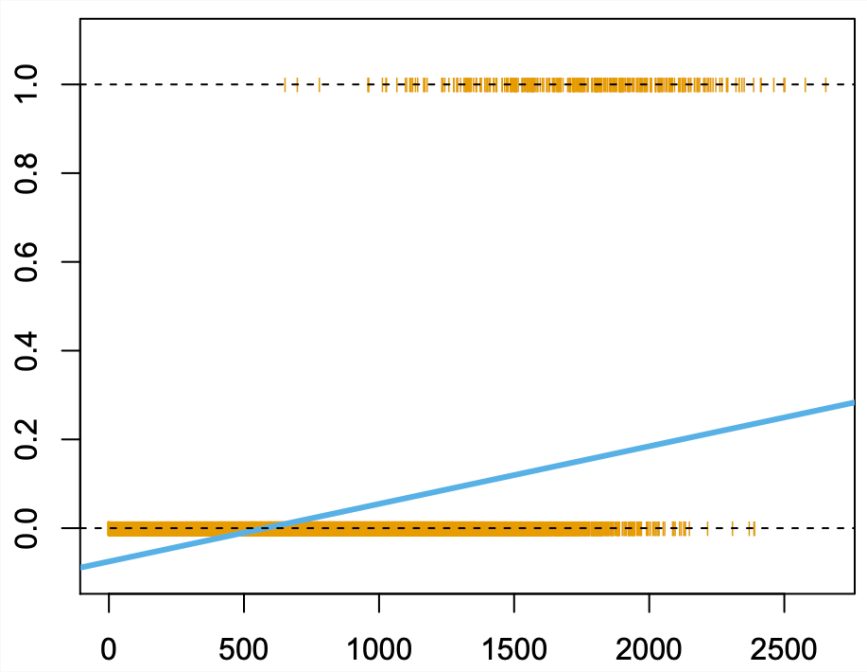
$$Y = \begin{cases} 1 \text{ if } \texttt{stroke} \\ 2 \text{ if } \texttt{drug overdose} \\ 3 \text{ if } \texttt{epileptic seizure} \end{cases} \quad \text{or } Y = \begin{cases} 2 \text{ if } \texttt{stroke} \\ 3 \text{ if } \texttt{drug overdose} \\ 1 \text{ if } \texttt{epileptic seizure} \end{cases} \quad \text{or } \ldots?$$

- More natural for binary response, e.g.

$$Y = \begin{cases} 0 \text{ if } \texttt{stroke} \\ 1 \text{ if } \texttt{drug overdose} \end{cases}$$

- We can model $\mathbb{P}(\texttt{drug overdose} \,|X) = X\beta$.
- Flipping the encoding $0 \leftrightarrow 1$ will give the same predictions
- but linear functions $X \mapsto X\hat{\beta}$ are unbounded in range: how to interpret if $X\hat{\beta} \notin [0, 1]$?

Plot from ISLP 2nd Ed. Page 139



Conclusion

- cannot use linear regression for more than 2 categories
- doesn't provide meaningful estimates for probabilities

Logistic Regression

Logistic Regression

- Instead of modelling $p(X) = \mathbb{P}(Y = 1|X)$ with the linear regression model $p(X) = \beta_0 + \beta_1 X$, we[1] can use the **logistic function** $\sigma(x) = \frac{e^x}{1+e^x} = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{x}{2}\right)$:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} = \sigma(\beta_0 + \beta_1 X)$$

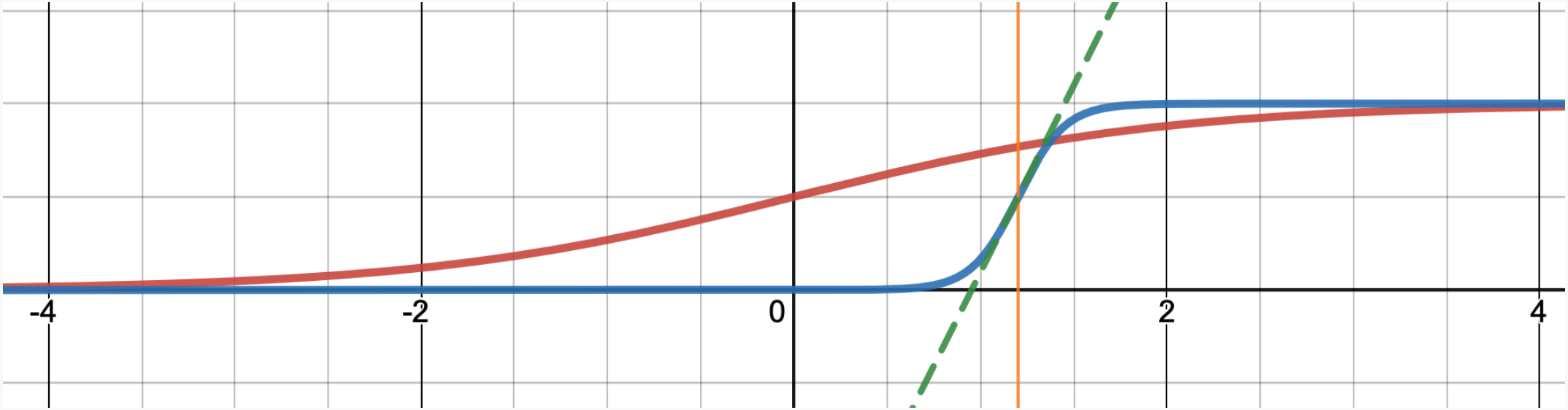
- Since σ takes values in $(0, 1)$, the model will give *interpretable* results.
- The inverse function is called the **logit** function:

$$\operatorname{logit}(y) = \sigma^{-1}(y) = \log \frac{y}{1-y}.$$

- We have $\operatorname{logit}(p(X)) = \beta_0 + \beta_1 X$. The quantity $\operatorname{logit}(p(X))$ is called the **logit**, or **log-odds**.

[1] I got the σ from Wikipedia's Logit page. Presumably sigma for sigmoid.

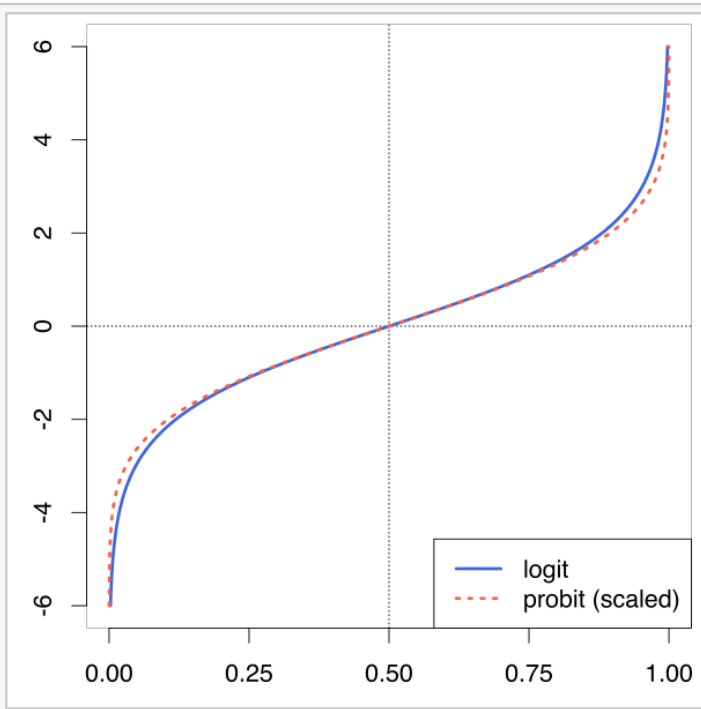
Logistic Plots



$y = \sigma(x) = \frac{e^x}{1+e^x}$ $x = a$ $y = \sigma(m(x-a)), \quad m > 1$ $y = \frac{m}{4}(x-a) + \frac{1}{2}$

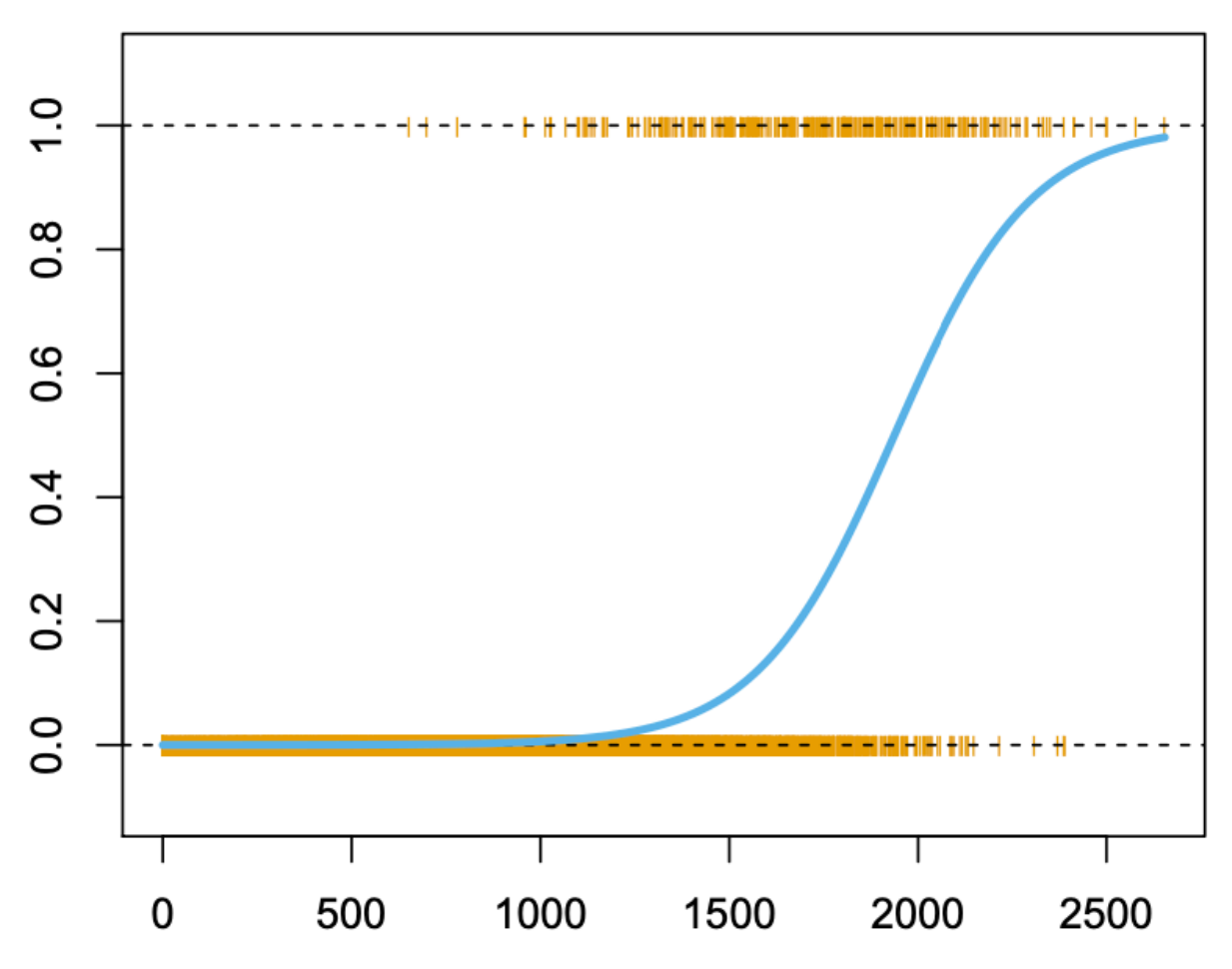
Logit Plot

- plot of $y = \text{logit}(x)$ from Wikipedia.
  - near $x = 0$, logit $\approx$ scaled "probit" a.k.a. gaussian ppf.
  - Link on link functions. More on other "link" functions at the end. (GLM)



Comparison of the logit function with a scaled probit (i.e. the inverse CDF of the normal distribution), comparing $\text{logit}(x)$ vs. $\Phi^{-1}(x)/\sqrt{\frac{\pi}{8}}$, which makes the slopes the same at the origin.

Second Plot from ISLP 2nd Ed. Page 139



Fitting with Maximum Likelihood Estimation

- We could fit $\operatorname{logit} Y$ with least squares or nonlinear least squares on Y but this is unnatural.
- More principled approach: Maximum Likelihood Estimate (MLE).
- The **likelihood function** $\ell(x, \beta)$ is the probability of observing x if we were sampling from the model with parameters β.
- Idea: select the parameter values that make the observed data most probable, i.e. if $\mathcal{B}$ denotes the space of all parameters β then

$$\hat{\beta} = \operatorname*{argmax}_{\beta \in \mathcal{B}} \ell(\beta).$$

- For logistic regression, where $p(X) = \mathbb{P}(Y = 1 | X)$,

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i)).$$

- In python™, we can `from sklearn.linear_model import LogisticRegression`.

Multiple Logistic Regression ($p > 1$)

- Rinse and repeat with $X \in \mathbb{R}^{1\times p}$, $\beta \in \mathbb{R}^p$,

$$p(X) = \sigma(X\beta) = \frac{e^{\beta_0 + X_1\beta_1 + \ldots + X_p\beta_p}}{1 + e^{\beta_0 + X_1\beta_1 + \ldots + X_p\beta_p}}.$$

- With $p > 1$, there can be **CONFOUNDING** (as in multiple linear regression):
  - coefficients can vary wildly as you add features due to lack of independence.

Confounding Data Example: Probability of default

Tables 4.1, 4.2, 4.3 in ISLP 2nd Ed. (simplified, pages 142, 143)

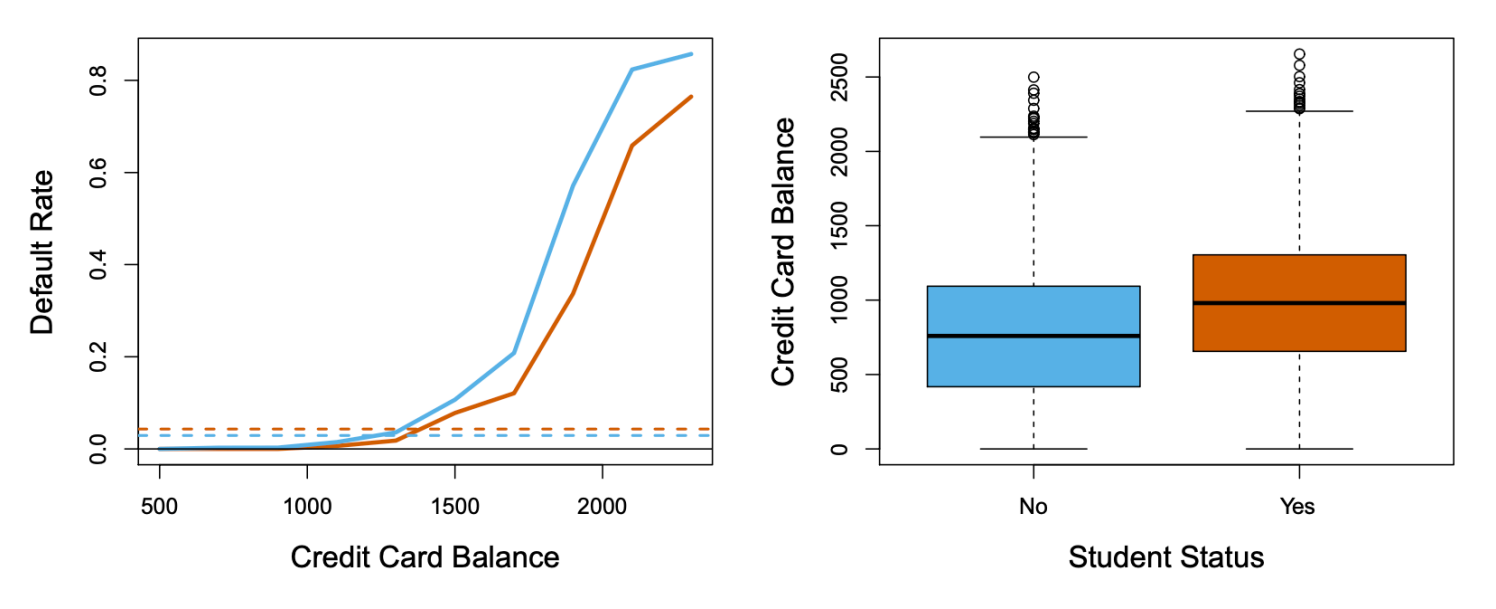
`student[Yes]` is one-hot encoded.

	coeff	std error	p-value
`intercept`	−3.504	0.071	<0.0001
`student[Yes]`	0.405	0.1150	0.0004

	coeff	std error	p-value
`intercept`	−10.651	0.361	<0.0001
`cc balance`	0.0055	0.0002	<0.0001

	coeff	std error	p-value
`intercept`	−10.869	0.492	<0.0001
`cc balance`	0.006	0.0002	<0.0001
`income`	0.003	0.008	0.712
`student[Yes]`	−0.647	0.236	0.006

Confounding Plots (ISLP 2nd Ed. page 143)



(Univariate analysis) Students are more likely to default as a group but

(Multivariate analysis) less likely to default for a given credit card balance

Resolution: students tend to have a higher balance.

Multinomial Logistic Regression (#categories $K > 1$)

- Choose one class to be the "baseline", say the Kth one. Then we model

$$\mathbb{P}(Y = K \mid X) = \frac{1}{1 + \sum_{i=1}^{K-1} e^{X\beta_i}},$$

$$\mathbb{P}(Y = k \mid X) = \frac{e^{X\beta_k}}{1 + \sum_{i=1}^{K-1} e^{X\beta_i}}, \quad k = 1, ..., K-1.$$

where now $X \in \mathbb{R}^{1\times p}$, $\beta_i \in \mathbb{R}^p$, $X\beta_i = \sum_j X_j\beta_{ij} \in \mathbb{R}, i = 1, ..., K$.

- It follows

$$\log \frac{\mathbb{P}(Y = k \mid X)}{\mathbb{P}(Y = K \mid X)} = X\beta_k = \beta_{k0} + \beta_{k1}X_1 + ... + \beta_{kp}X_p.$$

Multinomial Logistic Regression with softmax encoding

- The “softmax” encoding treats each category symmetrically: for $k = 1, ..., K$:

$$\mathbb{P}(Y = k \mid X) = \frac{e^{X\widetilde{\beta}_k}}{\sum_{i=1}^{K} e^{X\widetilde{\beta}_i}}$$

with the obvious link $\beta_k = \widetilde{\beta}_k - \widetilde{\beta}_K$.

- NB The function $\sigma : \mathbb{R}^N \to (0,1)^N$ whose ith component is

$$\sigma_i(v_1, ...v_N) = \frac{e^{v_i}}{\sum_{j=1}^{N} e^{v_j}}$$

is the softmax function used in Machine Learning 🤖

Multinomial Logistic Regression with softmax encoding

- The “softmax” encoding treats each category symmetrically: for $k = 1, ..., K$:

$$\mathbb{P}(Y = k \mid X) = \frac{e^{X\widetilde{\beta}_k}}{\sum_{i=1}^{K} e^{X\widetilde{\beta}_i}} = \frac{e^{X\left(\widetilde{\beta}_k - \widetilde{\beta}_K\right)}}{1 + \sum_{i=1}^{K-1} e^{X\left(\widetilde{\beta}_i - \widetilde{\beta}_K\right)}}.$$

with the obvious link $\beta_k = \widetilde{\beta}_k - \widetilde{\beta}_K$.

- NB The function $\sigma : \mathbb{R}^N \to (0,1)^N$ whose ith component is

$$\sigma_i(v_1, ...v_N) = \frac{e^{v_i}}{\sum_{j=1}^{N} e^{v_j}}$$

is the softmax function used in Machine Learning 🤖

Summary for Logistic Regression

- (Multiple) Logistic Regression models the log-odds $\left(\log \frac{p}{1-p}\right)$, where $p(x) = \mathbb{P}(Y = k|X = x)$, by a linear function $X\beta$.
- Fitting is done by maximum likelihood estimation (*not* OLS)
- Beware of **CONFOUNDING** in the $p > 1$ case
- Can be modified for more than 2 categories (Multinomial Log. Regr.)

Generative Models for Classification

Discriminative vs Generative

- Recall the Bayes classifier is the theoretically optimal classifier (classifying according to which of $\mathbb{P}(Y = k|X = x)$ is largest; assumes we know $\mathbb{P}(Y = k|X = x)$)
- Logistic regression is a "discriminative" model which directly estimates $\mathbb{P}(Y = k|X = x)$ with sigmoids.
- We now consider a different way to approximate the Bayes classifier: we estimate each term that appears in Bayes' Theorem,

$$\mathbb{P}(Y = k|X = x) = \frac{\mathbb{P}(X = x|Y = k)\mathbb{P}(Y = k)}{\mathbb{P}(X = x)}.$$

- Why consider these methods?
  1. The MLE in logistic regression may not exist (not really issue for prediction)
  2. Improved accuracy if X are approximately normal and data is lacking
- These methods will also generalise to $K > 2$, like multinomial logistic regression.

Notation: the prior π_k, the posterior p_k, and the likelihood f_k

Bayes:

$$\mathbb{P}(Y = k|X = x) = \frac{\mathbb{P}(X = x|Y = k)\mathbb{P}(Y = k)}{\mathbb{P}(X = x)}.$$

- For ease of notation, define

$$\pi_k = \mathbb{P}(Y = k), \qquad p_k(x) = \mathbb{P}(Y = k|X = x),$$
$$f_k(x) = \mathbb{P}(X = x|Y = k).$$

Then $\mathbb{P}(X = x) = \sum_\ell \pi_\ell f_\ell(x)$ (law of total probability) and Bayes can be written

$$p_k(x) = \frac{\pi_k f_k(x)}{\sum_\ell \pi_\ell f_\ell(x)}$$

Estimating π_k and f_k

Bayes' Theorem

$$p_k(x) = \frac{\pi_k f_k(x)}{\sum_\ell \pi_\ell f_\ell(x)}$$

- π_k is easy to estimate from the data: just take the proportion that are in class k.
- f_k is much harder. We will make simplifying assumptions and arrive at 3 methods:
  1. Linear Discrimant Analysis (LDA)
  2. Quadratic Discriminant Analysis (QDA)
  3. Naïve Bayes
- approximate Bayes classifier $\rightsquigarrow$ classify according to the largest of the p_ks.

Linear Discriminant Analysis and Quadratic Discriminant Analysis

Fisher's Linear Discriminant Analysis, $p = 1$: the discriminant

- For LDA, we assume that $(X|Y = k) \sim N(\mu_k, \sigma^2)$, i.e.

$$f_k(x) = \frac{1}{(2\pi)^{1/2}\sigma} \exp\left(\frac{-(x-\mu_k)^2}{2\sigma^2}\right)$$

- Plugging into $p_k(x) = \frac{\pi_k f_k(x)}{\sum_\ell \pi_\ell f_\ell(x)}$ and taking logs:

$$\log p_k(x) = \log \pi_k - \frac{(x-\mu_k)^2}{2\sigma^2} - \log\left(\sum_{\ell=1}^{K} \pi_\ell e^{-\frac{x^2}{2\sigma^2} + \frac{x\mu_\ell}{\sigma^2} - \frac{\mu_\ell^2}{2\sigma^2}}\right)$$

$$= \underbrace{\log \pi_k + \frac{x\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2}}_{\delta_k(x)} + \text{stuff independent of } k$$

- So to pick the largest of the $p_k(x)$, it is enough to pick the largest **discriminant** $\delta_k(x)$.

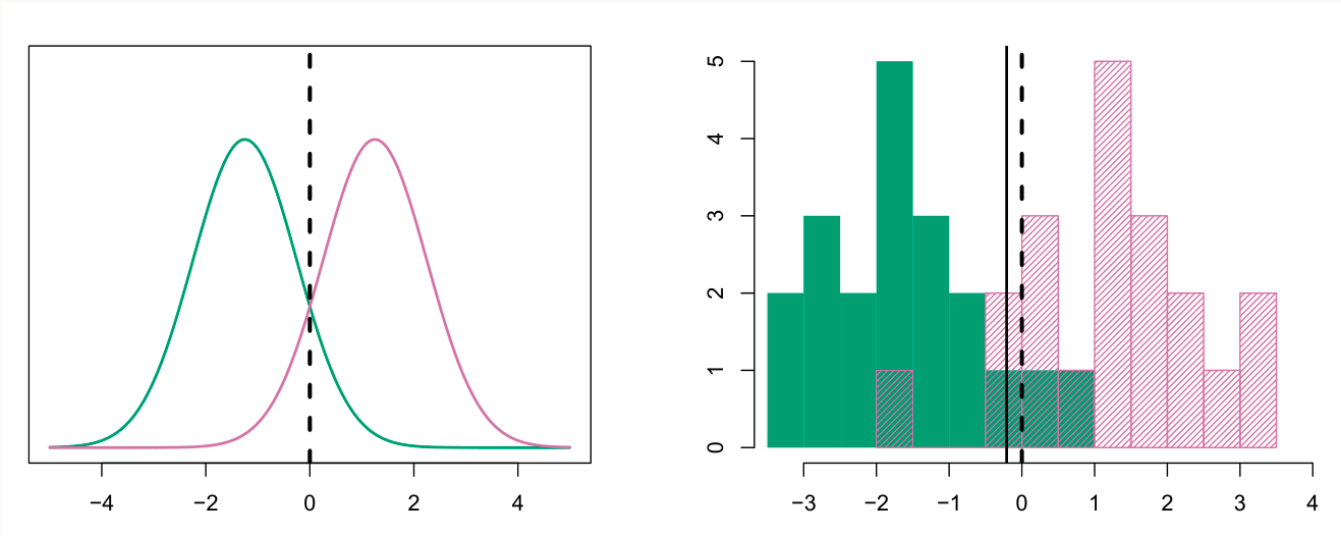
Linear Discriminant Analysis, $p = 1$: estimating parameters

- *linear* discriminant because $\delta_k(x) = \log \pi_k + \frac{x\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2}$ is linear.
- we put (simplest case) $n_k = \#\{i : y_i = k\}$, and

$$\hat{\pi}_k = \frac{n_k}{n}, \qquad \hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i,$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} \left(x_i - \hat{\mu}_k\right)^2.$$

LDA Example (ISLP Fig 4.4, page 148)



Two one-dimensional normal density functions are shown. The dashed vertical line represents the Bayes decision boundary.

20 observations were drawn from each of the two classes, and are shown as histograms. The Bayes decision boundary is again shown as a dashed vertical line. The solid vertical line represents the LDA decision boundary estimated from the training data.

LDA, $p \geq 1$

- We assume $(X \mid Y = k) \sim N(\mu_k, \Sigma)$ i.e.

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

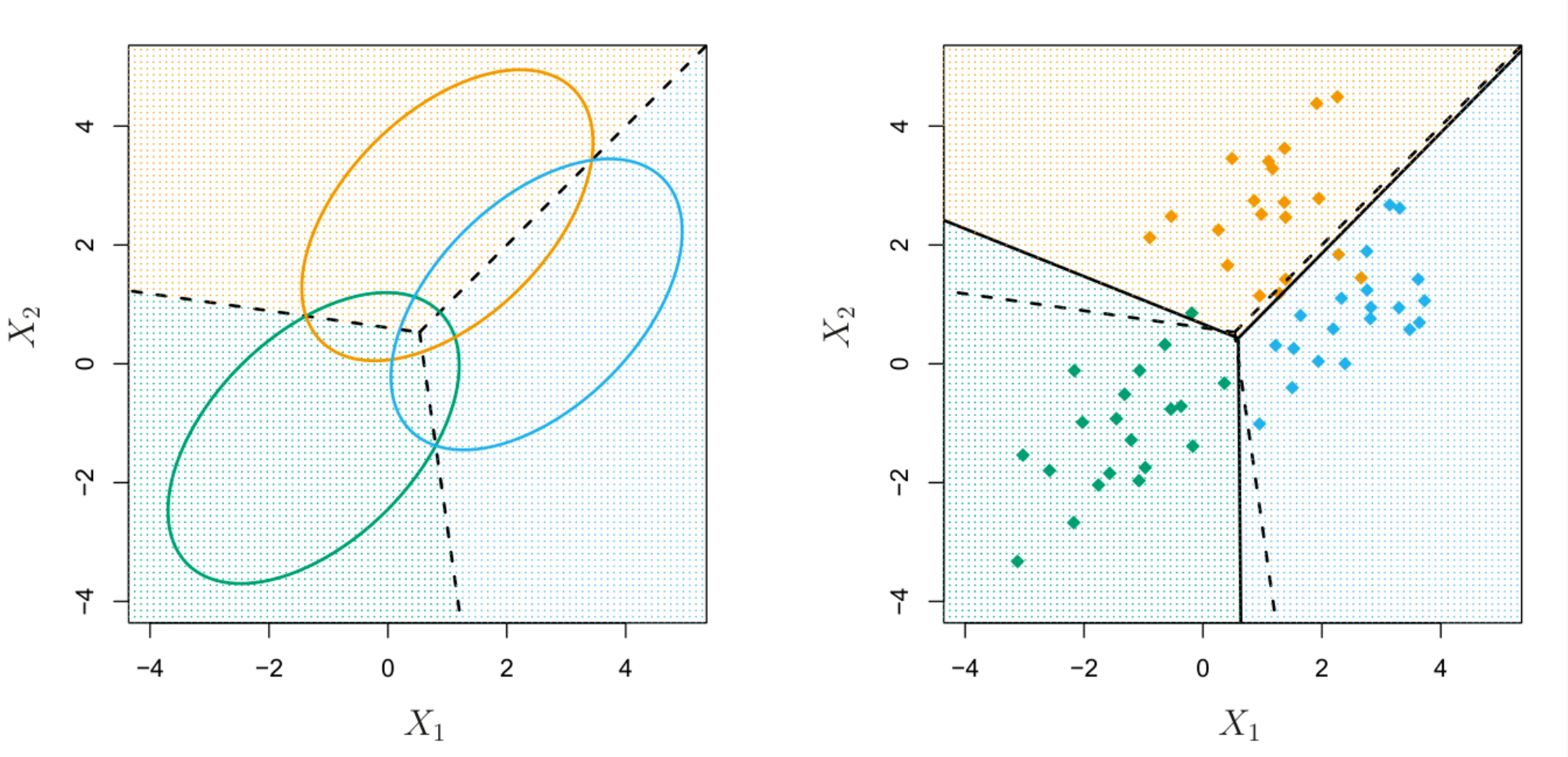
- leads again to the linear discriminant

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

- and we can similarly estimate

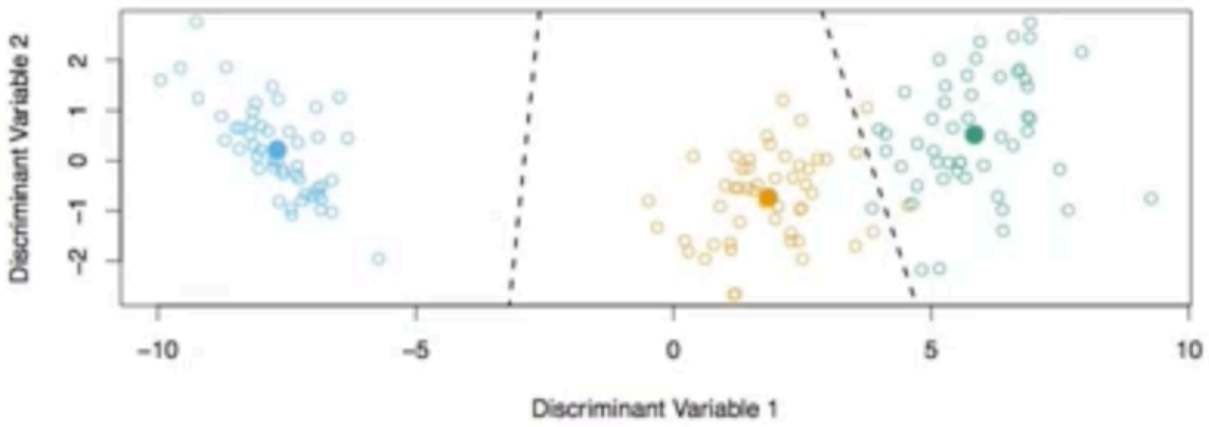
$$\hat{\pi}_k = \frac{n_k}{n}, \qquad \hat{\mu}_k = \frac{1}{n_k}\sum_{i:y_i=k} x_i, \qquad \hat{\Sigma} = \frac{1}{n-K}\sum_{k=1}^{K}\sum_{i:y_i=k}\left(x_i - \hat{\mu}_k\right)\left(x_i - \hat{\mu}_k\right)^T.$$

LDA plot, $p = 2, K = 3$ (ISLP Fig 4.6, page 151)



(ISLR video on LDA)

Fisher's Discriminant Plot



When there are K classes, linear discriminant analysis can be viewed exactly in a $K-1$ dimensional plot.

Why? Because it essentially classifies to the closest centroid, and they span a $K-1$ dimensional plane.

Even when $K>3$, we can find the "best" 2-dimensional plane for vizualizing the discriminant rule.

LDA example: predicting default from balance and student status

- Model evaluation on training set with a **confusion matrix** (ISLP Table 4.4, page 152):

	True No	True Yes	Total
Predict No	9644	252	9896
Predict Yes	23	81	104
Total	9667	333	10,000

- Overall training error rate is $\frac{23+252}{10000} = 2.75\%$.
  - not *test* error
  - the trivial classifier has error rate $\frac{333}{10000} = 3.33\%$
- For binary classifiers, can consider false positive and false negative rates:
  - $\mathrm{FPR} = \frac{23}{9667} = 0.2\%$, and $\mathrm{FNR} = \frac{252}{333} = 75.7\%$.
  - $\mathrm{specificity} = 1 - \mathrm{FPR} = 99.8\%$ and $\mathrm{sensitivity} = 1 - \mathrm{FNR} = 24.3\%$.
- LDA $\approx$ Bayes which minimises total error: no guarantees on false pos/negative rate.

LDA example: tuning for better false negative rate (ISLP pg. 154)

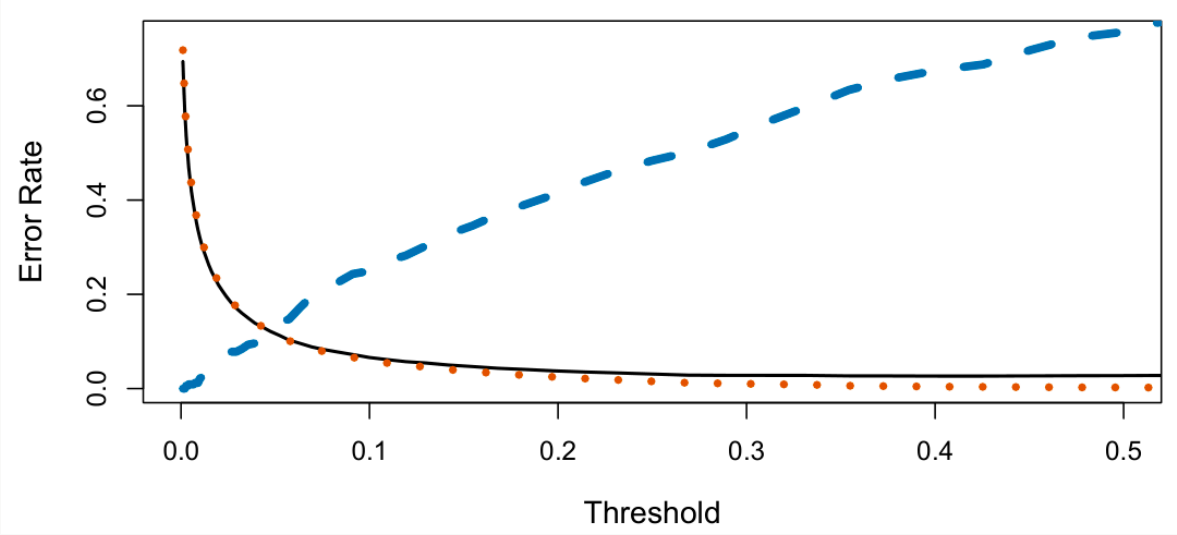
- LDA for 2 classes assigns to the k for which $\mathbb{P}(Y = k \mid X) > p_0 = 0.5$.
- We can adjust the threshold p_0 to improve the sensitivity.

Training confusion matrix, $p_0 = 0.2$

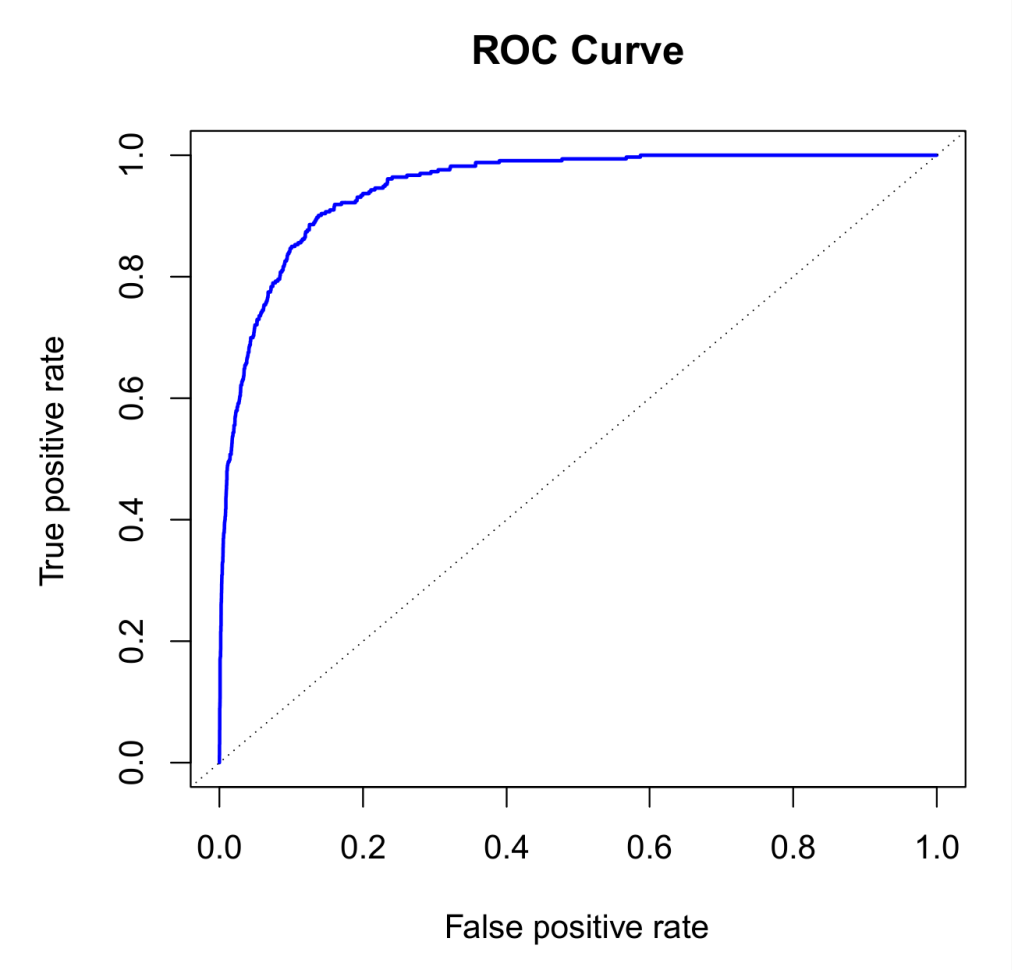
	No	Yes	Total
Pred No	9432	138	9570
Pred Yes	235	195	430
Total	9667	333	10,000

$\text{sensitivity} = 41\%,$

$\text{total error} = 3.73\%.$



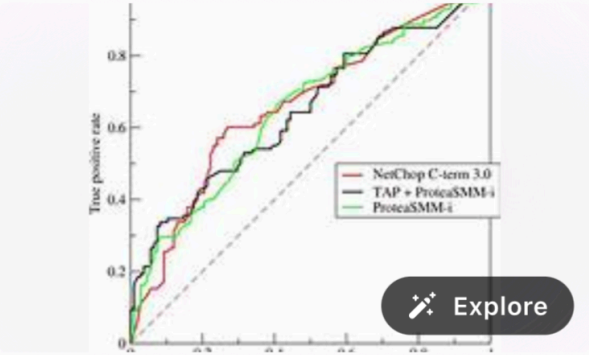
ROC curve



- best classifier will have a ⌜ shape
- $y = x$ is the completely useless classifier
- Area Under Curve (AUC): number that summarises the model across all thresholds

Receiver Operating Characteristic

The name “ROC curve” stands for “receiver operating characteristic curve”. It is a graphical plot that shows the performance of a binary classifier model at different threshold values. The ROC curve was first developed by electrical engineers and radar engineers during World War II for detecting enemy objects in battlefields, starting in 1941 [1]. The “receiver” refers to the device that picks up the signal, and the “operating characteristic” refers to how well it works [2]. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at each threshold setting, which are also known as sensitivity and fall-out, respectively. The ROC curve can help compare different models and choose the optimal threshold for a given task [1].



[1]: Receiver operating characteristic - Wikipedia [2]: What is the origin of the “receiver operating characteristic” (ROC) terminology? - Cross Validated

Learn more 1 en.wikipedia.org 2 stats.stackexchange.com 3 en.wikipedia.org

1 of 5

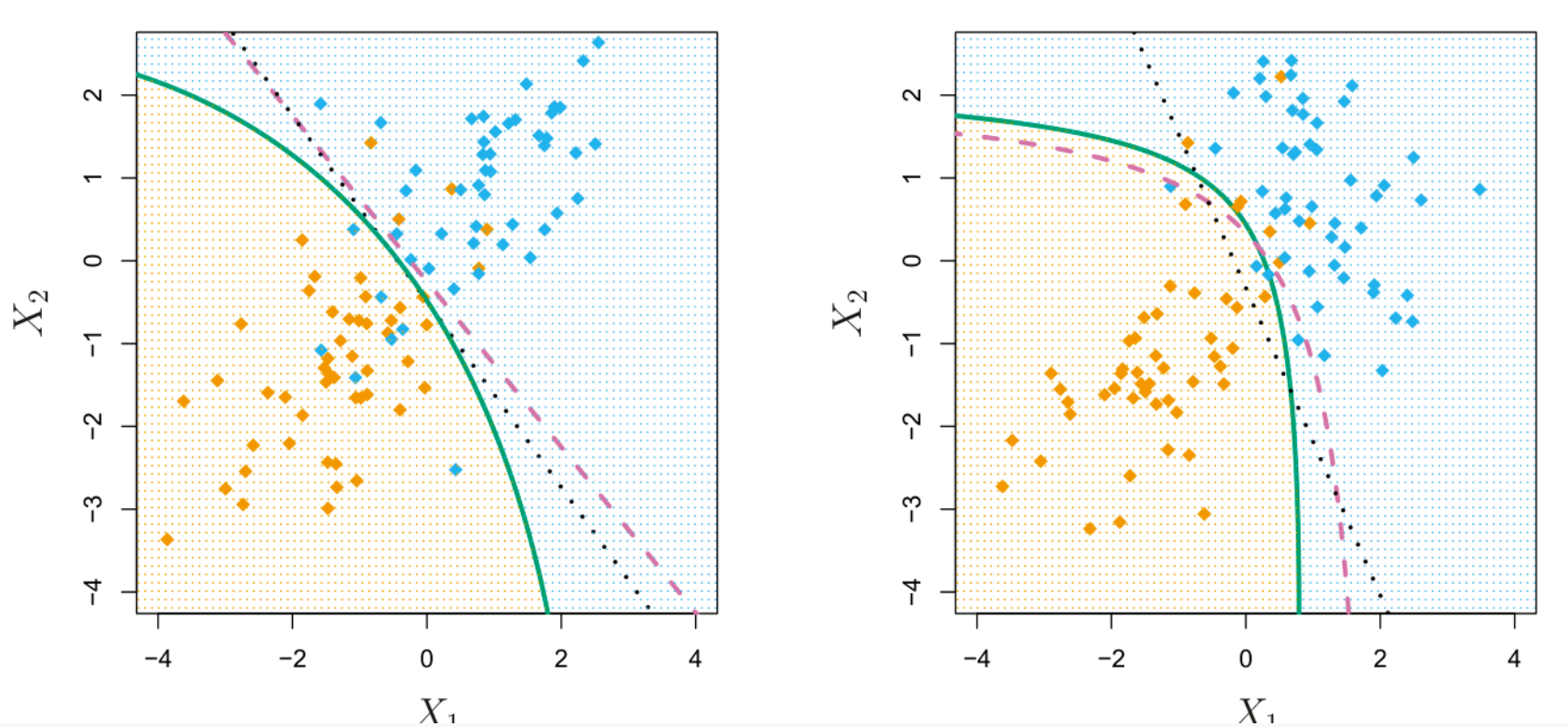
Quadratic Discriminant Analysis

- For LDA, we assume that $(X|Y = k) \sim N(\mu_k, \Sigma)$.
- For QDA, we assume that $(X|Y = k) \sim N(\mu_k, \Sigma_k)$. Then we cannot cancel the common factor of $x^T\Sigma^{-1}x/2$, which means the discriminant is quadratic:

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

- LDA is a special case of QDA
- QDA is more flexible than LDA
- QDA has low bias, LDA has low variance

QDA plot, ISLP page 157



QDA plot, ESL 2nd Ed. page 103

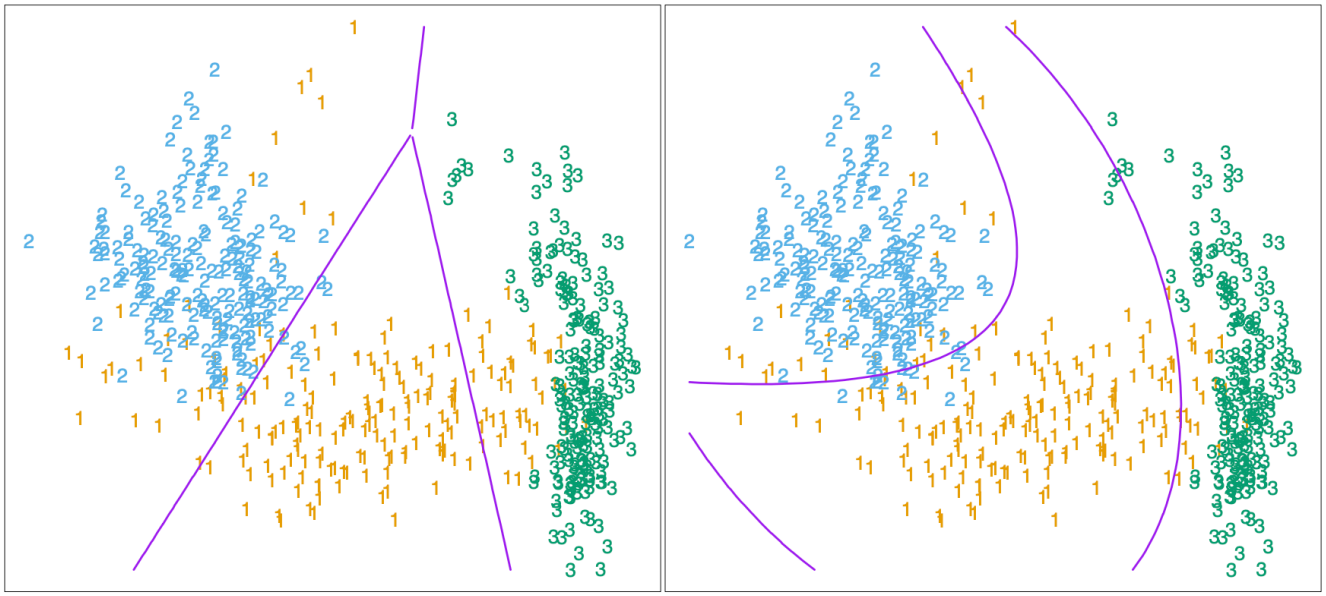


FIGURE 4.1. *The left plot shows some data from three classes, with linear decision boundaries found by linear discriminant analysis. The right plot shows quadratic decision boundaries. These were obtained by finding linear boundaries in the five-dimensional space* $X_1, X_2, X_1X_2, X_1^2, X_2^2$. *Linear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

Naïve Bayes

Naïve Bayes

- for LDA or QDA, we assumed f_k was a Gaussian density.
- for Naïve Bayes, we instead assume:

Within the kth class, the p predictors are independent.

i.e. for each $k = 1, ..., K$, we assume f_k takes the form:

$$f_k(x) = f_{k1}(x_1) f_{k2}(x_2) \cdots f_{kp}(x_p).$$

- great simplification because how to model the interactions?
  - for MV normal distribution, given by the $\frac{p(p-1)}{2}$ off-diagonal terms of Σ
  - In general, need $n \gg 1$ to estimate full joint distribution.

$$\rightsquigarrow \quad p_{k(x)} = \frac{\pi_k f_{k1}(x_1) f_{k2}(x_2) \cdots f_{kp}(x_p)}{\sum_\ell \pi_\ell f_{\ell 1}(x_1) f_{\ell 2}(x_2) \cdots f_{\ell p}(x_p)}$$

Naïve Bayes: Estimation of marginals

- The easiest thing to do is to assume a parametric form for f_{ki}, e.g. Gaussian
  - independence $\rightsquigarrow$ f_{ki} is the density of some $N(\mu, \sigma^2 I_p)$ rv. Special case of QDA
- Non-parametric estimation for quantitative predictor:
  - histograms or "smoothed histograms" i.e. kernel density estimators
- qualitative predictors: estimate as proportions. E.g. if
  - the jth predictor x_j takes values in $\{$🍕, 🍔, 🍫$\}$, and
  - for the kth class we observed 100 training data points, for which

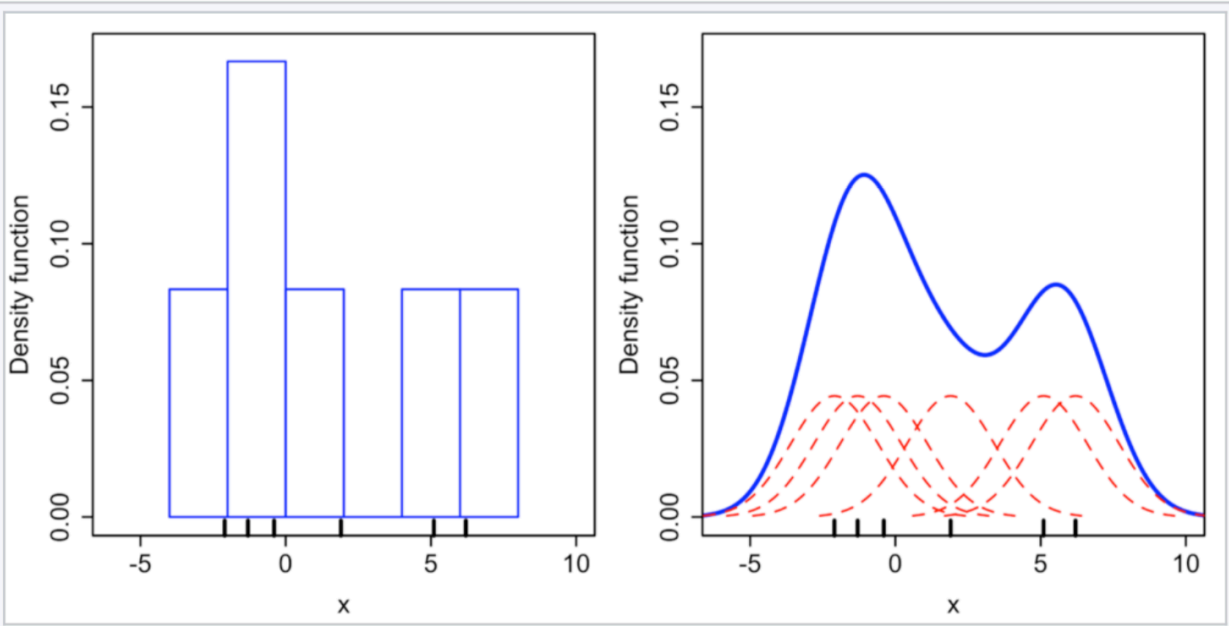
$$\#\{x_j = 🍕\} = 32 \qquad \#\{x_j = 🍔\} = 55 \qquad \#\{x_j = 🍫\} = 13$$

Then we can put

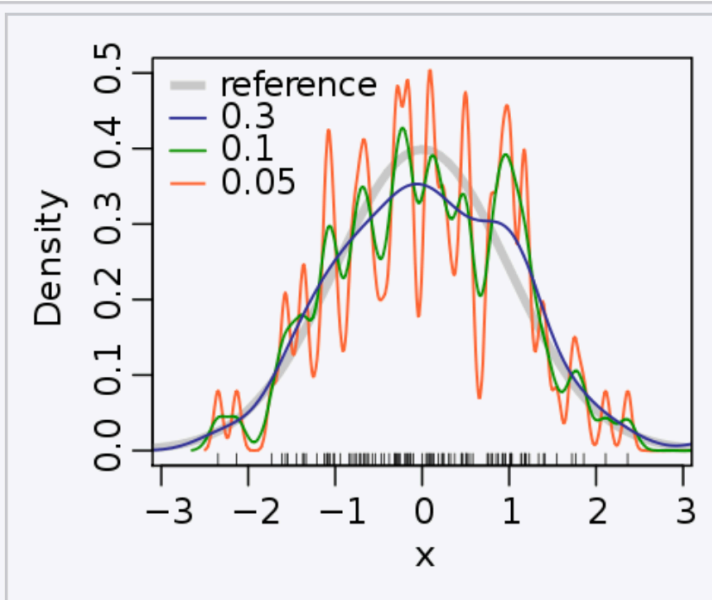
$$f_{kj}(x_j) = \begin{cases} 0.32 \text{ if } x_j = 🍕 \\ 0.55 \text{ if } x_j = 🍔 \\ 0.13 \text{ if } x_j = 🍫 \end{cases}$$

Kernel Density Estimators

Wikipedia pics on KDEs



Comparison of the histogram (left) and kernel density estimate (right) constructed using the same data. The six individual kernels are the red dashed curves, the kernel density estimate the blue curves. The data points are the rug plot on the horizontal axis.



Kernel density estimation of 100 normally distributed random numbers using different smoothing bandwidths.

Naïve Bayes on the Default-Balance-Student Data

Training confusion matrix, $p_0 = 0.5$

	No	Yes	Total
Pred No	9621	244	9865
Pred Yes	46	89	135
Total	9667	333	10,000

$\text{sensitivity} = 26.7\%,$

$\text{total error} = 2.9\%.$

Training confusion matrix, $p_0 = 0.2$

	No	Yes	Total
Pred No	9339	130	9469
Pred Yes	328	203	531
Total	9667	333	10,000

$\text{sensitivity} = 60.9\%,$

$\text{total error} = 4.58\%.$

- Performs about the same as LDA: $n = 10,000$ and $p = 2$.
- We expect Naïve Bayes to be better when n is small and p is large.

Logistic regression on the Default-Balance-Student Data

```
from ISLP import load_data
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression

data = load_data("Default")
ny = {"No": 0, "Yes": 1}
for col in ["default", "student"]:
    data.loc[:, col] = data[col].map(ny)
X = data.loc[:,["student", "balance", "income"]]
y = data.loc[:,"default"]
model = LogisticRegression(penalty=None)
model.fit(X, y)
y_pred = model.predict(X)
confusion_matrix(y, y_pred).T
```

Output in notebook:

```
array([[9608,  269],
       [  59,   64]])
```

Threshold $p_0 = 0.2$:

```
y_pred2 = (model.predict_proba(X)[:,1] >=
0.2).astype(bool)
confusion_matrix(y, y_pred2).T
```

Output:

```
array([[9308,  177],
       [ 359,  156]])
```

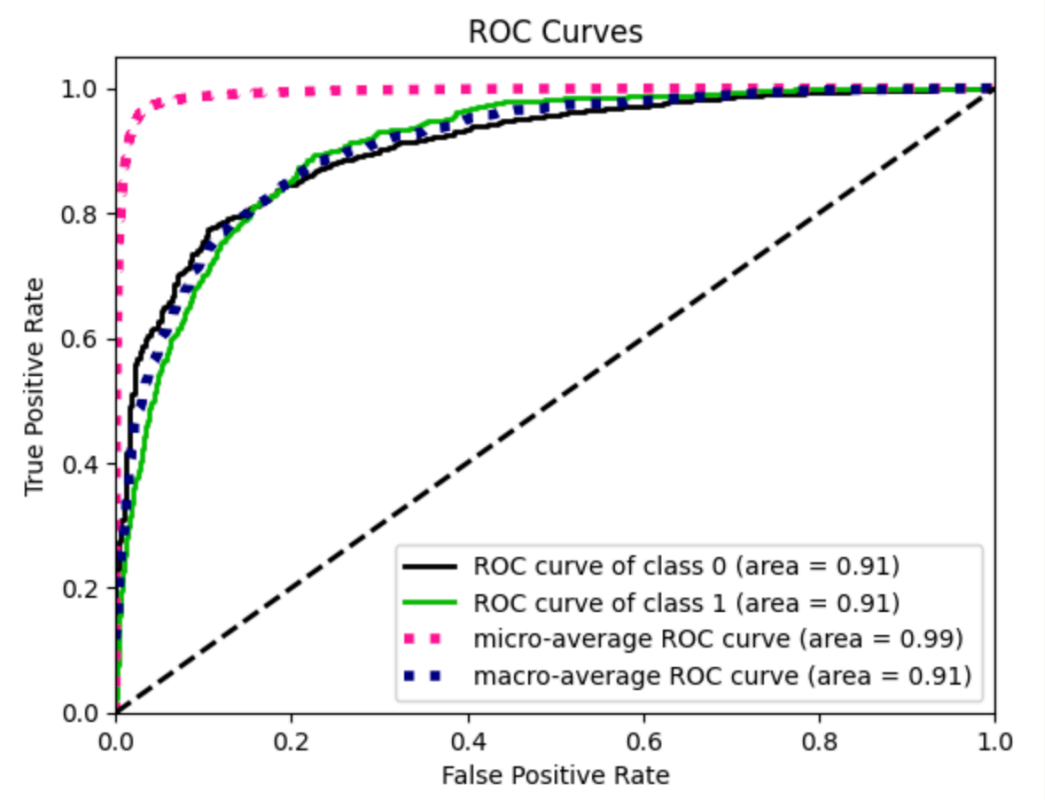
Logistic Regression on Default: ROC curve

```
pip install scikit-plot
```

or `pdm add scikit-plot` or ... then

```
import scikitplot as skplt
import matplotlib.pyplot as plt

y_probas = model.predict_proba(X)
skplt.metrics.plot_roc(y, y_probas)
plt.show()
```



Summary for Generative Models

- assume a computationally convenient form for the joint distribution by estimating $\pi_k = \mathbb{P}(Y = k)$ and $f_{k(x)} = \mathbb{P}(X = x|Y = k)$.
- Recover $\mathbb{P}(Y = k|X = x)$ by Bayes' Theorem
- For LDA, $X|Y = k \sim N(\mu_k, \Sigma)$: linear boundaries
- For QDA, $X|Y = k \sim N(\mu_k, \Sigma_k)$: curved boundaries
  - μ_k, Σ_k can be independently identified or estimated
- For Naive Bayes, assume $\{X_i|Y = k\}_{i=1}^{p}$ are (mutually) independent.

Tuning for sensitivity / specificity

- Adjust decision boundary by changing the threshold values
- e.g. if $K = 2$ then can classify to class 2 if $\mathbb{P}(Y = k|X = x) > t \in [0, 1]$
- Will increase overall error but can improve false pos/negative rate
- Consult ROC curve for sweetspot, AUC to evaluate model

Comparison of Classification Methods

Formula for log probability ratios

For each classification method, we will compare

$$\mathrm{lr}(x) := \log \frac{\mathbb{P}(Y = k \mid X = x)}{\mathbb{P}(Y = K \mid X = x)} = \log \frac{\pi_k f_k(x)}{\pi_K f_K(x)}.$$

- We have $\mathrm{lr}_{\mathrm{LDA}}(x) = a_k + b_k x$ for some a_k, b_k, like $\mathrm{lr}_{\log}(x)$,
- $\mathrm{lr}_{\mathrm{QDA}}(x) = a_k + b_k x + x^T C_k x$.
- Naïve Bayes:

$$\mathrm{lr}_{\mathrm{NB}}(x) = \log \frac{\pi_k \prod_{j=1}^{p} f_{kj}(x_j)}{\pi_K \prod_{j=1}^{p} f_{Kj}(x_j)}$$

$$= \log \frac{\pi_k}{\pi_K} + \sum_{j=1}^{p} \log \frac{f_{kj}(x_j)}{f_{Kj}(x_j)} = a_k + \sum_{j=1}^{p} g_{kj}(x_j)$$

- So Naïve Bayes takes the form of an Generalised Additive Model (GAMs: Chapter 7)

Comparison of classification methods

- $\mathrm{lr}_{\mathrm{LDA}}(x) = a_k + b_k x = \mathrm{lr}_{\log}(x)$,
- $\mathrm{lr}_{\mathrm{QDA}}(x) = a_k + b_k x + x^T C_k x$.
- $\mathrm{lr}_{\mathrm{NB}}(x) = a_k + \sum_{j=1}^{p} g_{kj}(x_j)$.

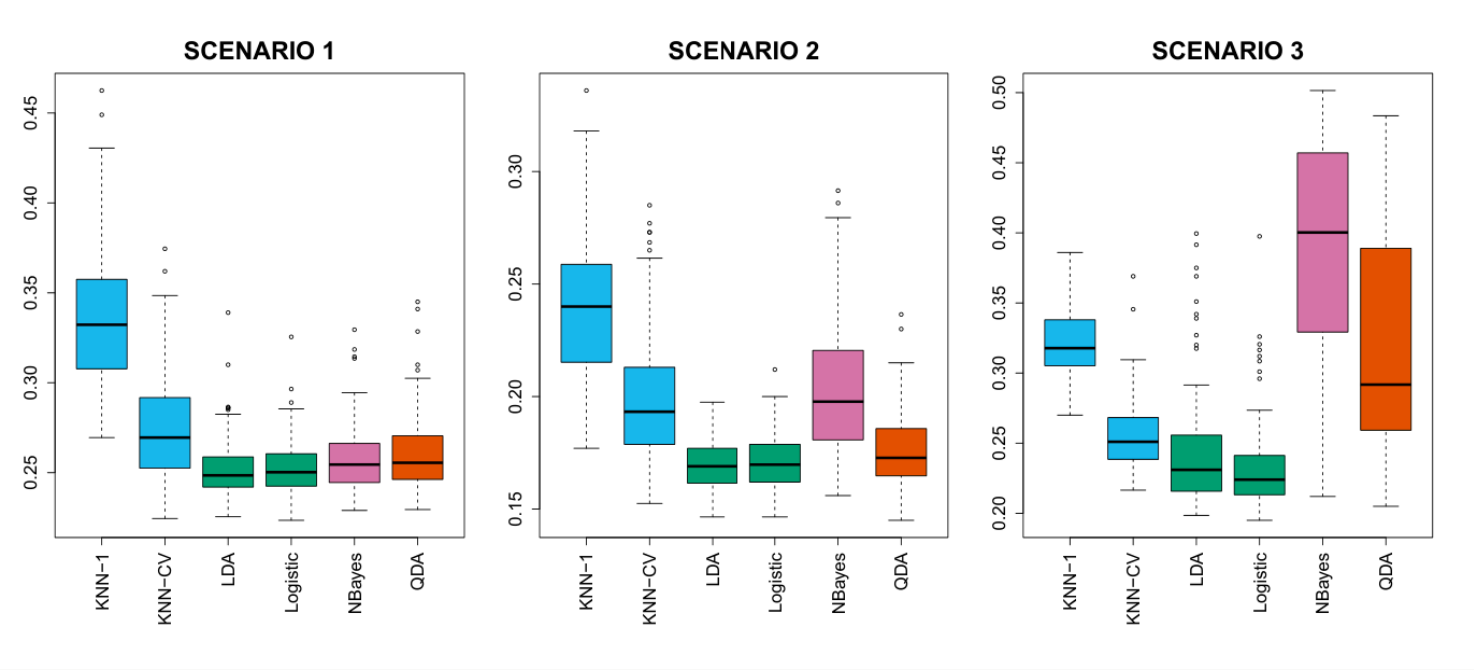
- LDA is a special case of QDA
- lr of any classifier with linear boundaries has the form $\mathrm{lr}_{\mathrm{NB}}$
  $\rightsquigarrow$ LDA is a special case of NB
- QDA with diagonal Σ is a special case of NB
- generally, $\mathrm{lr}_{\mathrm{QDA}} \neq \mathrm{lr}_{\mathrm{NB}}$
  $\rightsquigarrow$ QDA can have interaction terms.
- $\mathrm{lr}_{\mathrm{LDA}}(x) = \mathrm{lr}_{\log}(x)$ but classifier is trained differently based on different assumptions.
  $\rightsquigarrow$ LDA should be better when the normality assumption holds, LR otherwise

Comparison with KNN

- KNN is non-parametric and flexible.
  - ⇝ Should be best when boundaries are highly nonlinear
  - ⇝ requires a lot of data
  - ⇝ If boundary is nonlinear but data is small, consider QDA
  - ⇝ Logistic regression gives coefficients (interpretation)

Simple empirical tests 1,2,3 ($K = 2$)

Different methods on 100 random training sets

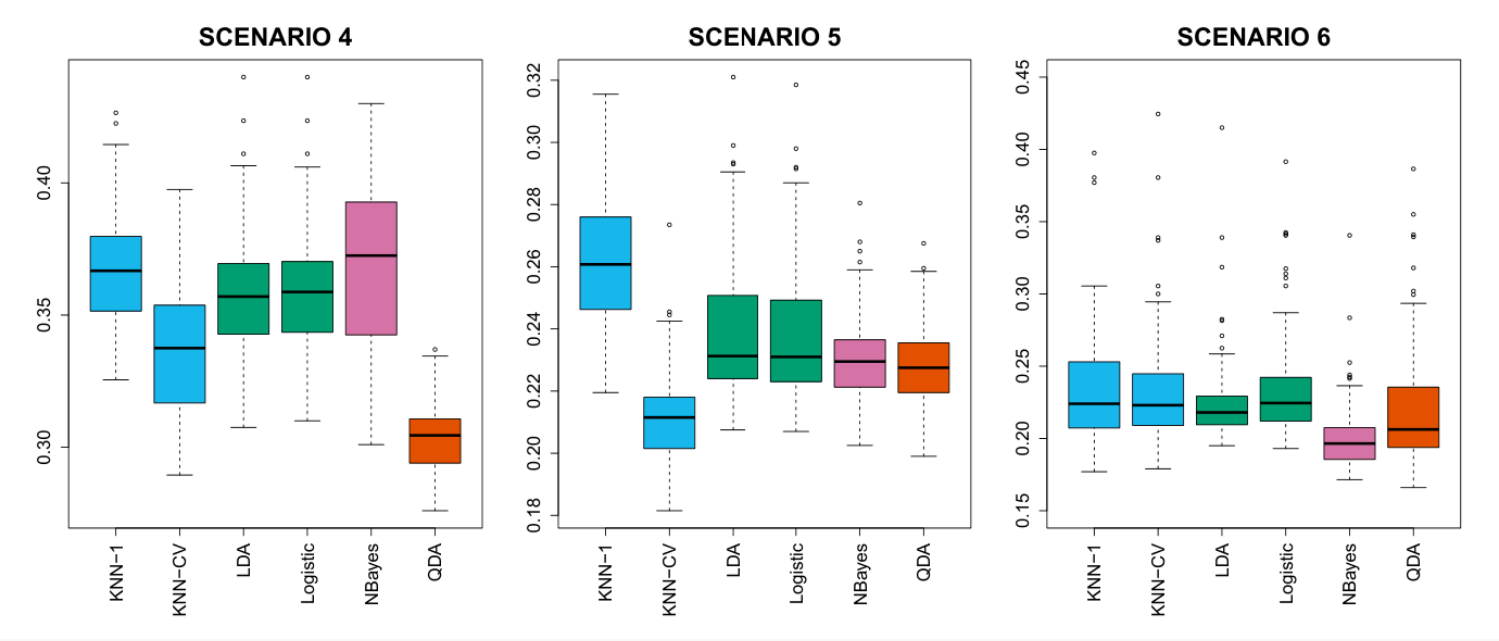


1. $n = 20$, uncorrelated gaussians
2. $n = 20$, $\text{corr}_{k=1,2} = 1/2$
3. $n = 50$, $\text{corr}_{k=1,2} = 1/2$, drawn from t dist.

Simple empirical tests 4,5,6 ($K = 2$)

4. $n = 20,$

$\text{corr}_{k=1} = 1/2$
$\text{corr}_{k=2} = -1/2$



5. Highly nonlinear distribution
6. $n = 6, (X|Y = k) \sim N(\mu_k, \sigma_k^2 I)$

Generalised Linear Models

The `Bikeshare` dataset

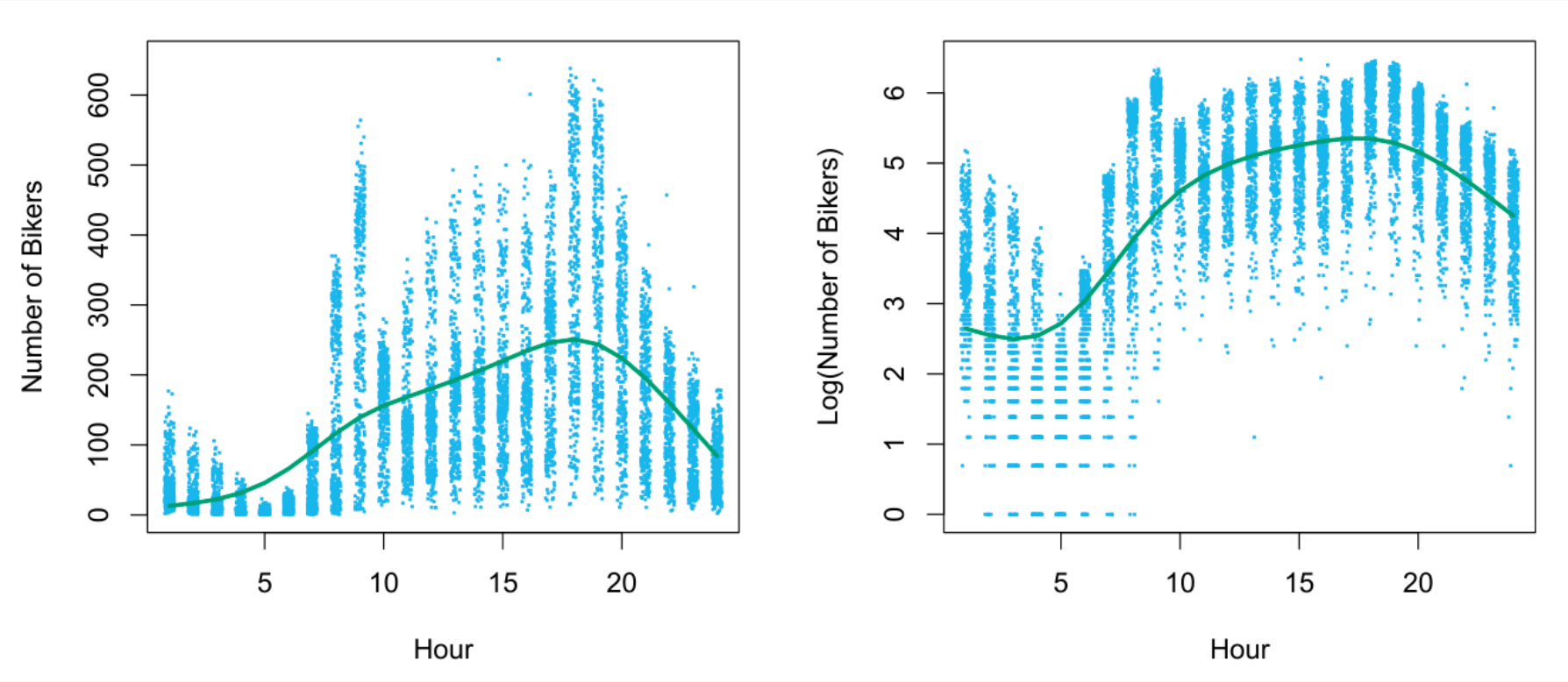
- Response: number of `bikers` $\in \mathbb{Z}_{\geq 0}$.

```
from ISLP import load_data
> Bike = load_data('Bikeshare')
> Bike.shape, Bike.columns
((8645, 15), Index(['season', 'mnth', 'day', 'hr', 'holiday', 'weekday',
'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed', 'casual',
'registered', 'bikers'], dtype='object'))
```

- Linear regression has similar issues to classification:
  - negative `bikers`?
  - fractional `bikers`?

Heteroskedasticity

🚲 In addition, data is heteroskedastic:



Poisson Regression

- A standard distribution on $\mathbb{Z}_{\geq 0}$ to model counts is the Poisson distribution: let $\lambda > 0$. We say $N \sim \mathrm{Poi}(\lambda)$ if $\mathbb{P}(N = k) = e^{-\lambda}\frac{\lambda^k}{k!}$ for $k = 0, 1, 2, \ldots$, and $\mathbb{P}(N = k) = 0$ otherwise.
- Note $\mathbb{E}N = \mathrm{Var}\, N = \lambda$ (models heteroskedasticity)
- Modelling assumption:

$$Y|X \sim \mathrm{Poi}(\lambda(X)), \qquad \lambda(X) = \exp(X\beta).$$

- Use MLE approach, i.e. choose β to maximise

$$\ell(\beta) = \prod_{j=1}^{n} e^{-\lambda(x_i)} \frac{\lambda(x_i)^{y_i}}{y_i!}.$$

Linear, Logistic and Poisson as Generalised Linear Models

- assume $Y|X$ belongs to some family of distributions
- use a **link function** η so that $\eta(\mathbb{E}(Y|X)) = X\beta$
- Fit by MLE estimation.

Regression type	Linear	Logistic	Poisson
$Y\|X$	Gaussian	Bernoulli	Poisson
link function	$\eta(y) = y$	$\eta(y) = \operatorname{logit} y$	$\eta(y) = \log y$