Topic 4: Generative Classifiers — Naive Bayes and DA

The generative classifiers use the Bayes Theorem $\boxed{\mathbb{P}(B|A) = \dfrac{P(A|B)P(B)}{P(A)}}$ to analyse the 'posterior distribution':

$$\mathbb{P}(Y = y|\boldsymbol{X} = \boldsymbol{x}) = \frac{\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|Y = y)\mathbb{P}(Y = y)}{\mathbb{P}(\boldsymbol{X} = \boldsymbol{x})}. \tag{4.1}$$

Note that we have to regard $\mathbb{P}$ as "probability density" function f when $\boldsymbol{x}$ and y are continuous.

In particular, when the response variable Y is categorical and has K distinct values 1, ..., K, then (4.1) becomes

$$\mathbb{P}(Y = j|\boldsymbol{X} = \boldsymbol{x}) = \frac{\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|Y = j)\mathbb{P}(Y = j)}{\sum_{k=1}^{K} \mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|Y = k)\mathbb{P}(Y = k)}, \quad j \in \{1, \cdots, K\} \tag{4.2}$$

where

- $\mathbb{P}(Y = k|\boldsymbol{X} = \boldsymbol{x})$, the *posterior probability*, is the probability that the new observation belongs to the kth class, given the predictor value for that observation;
- $\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}) = \sum_{k=1}^{K} \mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|Y = k)\mathbb{P}(Y = k)$ is regarded as a constant w.r.t the response class j;
- $\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|Y = j)$ is the *likelihood function*, a density function of $\boldsymbol{X}$ for an observation that comes from the jth class. It is relatively large if there is a high probability that an observation in the jth class has $\boldsymbol{X} \approx \boldsymbol{x}$;
- $\mathbb{P}(Y = j)$ is the *prior probability*, i.e. the probability that a randomly chosen observation comes from the jth class of response variable Y.

We are going to explore only two types of generative classifiers in this topic, i.e. the naïve Bayes model in Section 4.2 and discriminate analysis models (e.g. LDA, QDA) in Section 4.3.

4.1 Generative Models

A general probabilistic predictive model is described by a joint probablity of the input and the output

$$\mathbb{P}(X, Y).$$

According to Mitchell [1997],

- when we can estimate $\mathbb{P}(Y|X)$ directly, then the predictive model is called a **discriminative learning** model.
- when we have the prior estimate $P(Y)$ and $\mathbb{P}(X, Y)$, then we have the **generative learning** model with the likelihood function $P(X|Y) = P(X, Y)/P(Y)$.

  The probabilistic framework that underlie the generative models is the **Maximum a Posteriori (MAP)** which assumes θ are random variables being updated based on observed data. The generative classifier derived from (4.2) is

$$\begin{aligned} h_D(\boldsymbol{x}) &= \underset{j\in\{1,\dots,K\}}{\operatorname{argmax}} \mathbb{P}(Y = j|\boldsymbol{X} = \boldsymbol{x}) \\ &= \underset{j\in\{1,\dots,K\}}{\operatorname{argmax}} \frac{\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|Y = j)\mathbb{P}(Y = j)}{\mathbb{P}(\boldsymbol{X} = \boldsymbol{x})} \\ &= \underset{j\in\{1,\dots,K\}}{\operatorname{argmax}} \mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|Y = j)\mathbb{P}(Y = j) \\ &= \underset{j\in\{1,\dots,K\}}{\operatorname{argmax}} [\ln \mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|Y = j) + \ln \mathbb{P}(Y = j)] \end{aligned} \tag{4.3}$$

Note that MAP is only one way to get an estimator. There is much more information in $P(\theta|D)$, and it seems like a shame to simply compute the mode and throw away all other information. A *true Bayesian approach* is to use the posterior predictive distribution directly to make prediction about the label Y of a test sample with features X:

$$\mathbb{P}(Y \mid D, X) = \int_\theta P(Y, \theta \mid D, X) d\theta = \int_\theta P(Y \mid \theta, D, X) P(\theta|D) d\theta$$

Here θ is integrated out, our prediction takes all possible models into account. Unfortunately, the above is generally intractable in closed form and sampling techniques, such as Monte Carlo approximations, are used to approximate the distribution. A pleasant exception are Gaussian Processes.

As always the differences are subtle. In MLE we maximise $\log[P(D; \theta)]$ in MAP we maximise $\log[P(D|\theta)] + \log[P(\theta)]$. So essentially in MAP we only add the term $\log[P(\theta)]$ to our optimisation. This term is independent of the data and penalises if the parameters, θ deviate too much from what we believe is reasonable. The term $\log[P(\theta)]$ is interpreted as a measure of classifier complexity.

4.2 Naïve Bayes Classifier (Supervised Learning)

A `https://en.wikipedia.org/wiki/Naive_Bayes_classifier` is a simple probabilistic classifier (4.2) based on approximating $P(X|Y)$ using the **strong (naive) independence assumption**, i.e. $X_1, X_2, \cdots, X_p$ are **independent**:

$$\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|Y = j) = \mathbb{P}(X_1 = x_1|Y = j) \cdots \mathbb{P}(X_p = x_p|Y = j) = \prod_{i=1}^{p} \mathbb{P}(X_i = x_i|Y = j).$$

Therefore, the generative model (4.3) becomes the naive Bayes model:

$$h_D(\boldsymbol{x}) = \underset{j\in\{1,\dots,K\}}{\operatorname{argmax}} \mathbb{P}(Y=j)\prod_{i=1}^{p}\mathbb{P}(X_i=x_i|Y=j) = \underset{j\in\{1,\dots,K\}}{\operatorname{argmax}} \ln\mathbb{P}(Y=j)+\left[\sum_{i=1}^{p}\ln\mathbb{P}(X_i=x_i|Y=j)\right]. \tag{4.4}$$

The estimate of the prior distribution $\mathbb{P}(Y=j)$ using MLE is as follows:

$$\widehat{\mathbb{P}(Y=j)} = \frac{\#\{i : y_i = j\}}{n}. \tag{4.5}$$

However, it is possible to choose $\mathbb{P}(Y=j) = \frac{1}{K}$ if we know the outcome is theoretically uniformly distributed.

The features X_i can be the following cases:

- One X_i is categorical — Categorical NB
- One X_i is numeric — Gaussian NB, Non-parametric NB (using kernel density approximation rather than normal distribution)
- All X_i are binary — Bernoulli NB
- All X_i are integral — Multinomial NB & Complement NB(?)
- One X_i is non-negative integral — Poisson (distribution) NB

Naive Bayes Algorithm in R

```
library(naivebayes)
nbD   = naive_bayes(Y ~ ., D, options)
Yhat  = predict(nbD,  newX, type="class")
Prob  = predict(nbD,  newX, type="prob")  # Get log-posterior prob
Yhat  = predict(ldaD, newX)$class
Prob  = predict(ldaD, newX)$posterior     # log?
```

Most of the following naïve Bayes models are available in R except for the complement NB. R provides unified functions such as `naivebayes::naive_bayes`, `e1071::naiveBayes`, `bnlearn::naive.bayes` (which can only handle categorical data), `klaR::NaiveBayes`.

4.2.1 Categorical Naïve Bayes

When the feature X_i is categorical and takes on M_i possible values by $\mathscr{C}_i = \{c_1^{(i)}, \cdots, c_{M_i}^{(i)}\}$, the conditional distribution is assumed to be

$$\mathbb{P}(X_i = c|Y=j) = \frac{n_{X_i=c \ \& \ Y=j}}{n_{Y=j}}, \quad c \in \mathscr{C}_i. \tag{4.6}$$

where

- $n_{X_i=c \ \& \ Y=j}$ is the number of times category c appears in the samples X_i, which belong to class j;
- $n_{Y=j}$ is the number of samples with class j.

Example 4.2.1 (Final Exam May 2019, Q3(d)). Table Q3(d) shows a data set containing 7 observations with 3 categorical predictors, X_1, X_2 and X_3.

Observation	X_1	X_2	X_3	Y
1	C	No	0	Positive
2	A	Yes	1	Positive
3	B	Yes	0	Negative
4	B	Yes	0	Negative
5	A	No	1	Positive
6	C	No	1	Negative
7	B	Yes	1	Positive

Table Q3(d)

Without Laplace smoothing, predict the response, Y for an observation with $X_1 = $ B, $X_2 = $ Yes and $X_3 = 1$ using Naïve Bayes approach. (5 marks)

Solution: Let '+' denote 'Positive' and '-' denote 'Negative'.

prior	$\mathbb{P}(X_1\|Y)$	$\mathbb{P}(X_2\|Y)$	$\mathbb{P}(X_3\|Y)$	prop, Π	$\hat{Y}$
$\mathbb{P}(Y=+) = \frac{4}{7}$	$\mathbb{P}(\text{B}\|+) = \frac{1}{4}$	$\mathbb{P}(\text{Yes}\|+) = \frac{1}{2}$	$\mathbb{P}(1\|+) = \frac{3}{4}$	0.0536	
$\mathbb{P}(Y=-) = \frac{3}{7}$	$\mathbb{P}(\text{B}\|-) = \frac{2}{3}$	$\mathbb{P}(\text{Yes}\|-) = \frac{2}{3}$	$\mathbb{P}(1\|-) = \frac{1}{3}$	0.0635	✓, −

Since $\mathbb{P}(Y=-|X) > \mathbb{P}(Y=+|X)$, Y has higher probability to be "Negative".

Example 4.2.2. Consider the following case given in `https://machinelearningmastery.com/naive-bayes-tutorial-for-machine-learning/`

Weather	Car	Y
sunny	working	go-out
rainy	broken	go-out
sunny	working	go-out
sunny	working	go-out
sunny	working	go-out
rainy	broken	stay-home
rainy	broken	stay-home
sunny	working	stay-home
sunny	broken	stay-home
rainy	broken	stay-home

Construct the categorical Naïve Bayes model for the above data.

Solution: Let X_1=Weather, X_2=Car. The categorical Naïve Bayes model:

$$\mathbb{P}(Y=j|\boldsymbol{X}=\boldsymbol{x}) \propto \mathbb{P}(Y=j)\mathbb{P}(X_1=x_1|Y=j)\mathbb{P}(X_2=x_2|Y=j)$$

where Prior, $\mathbb{P}(Y) = \begin{cases} 0.5, & Y = out \\ 0.5, & Y = stay \end{cases}$

$$\mathbb{P}(X_1|Y=out) = \begin{cases} \frac{4}{5}, & X_1 = sunny \\ \frac{1}{5}, & X_1 = rainy \end{cases}, \qquad \mathbb{P}(X_1|Y=stay) = \begin{cases} \frac{2}{5}, & X_1 = sunny \\ \frac{3}{5}, & X_1 = rainy \end{cases}$$

$$\mathbb{P}(X_2|Y=out) = \begin{cases} \frac{4}{5}, & X_2 = working \\ \frac{1}{5}, & X_2 = broken \end{cases}, \qquad \mathbb{P}(X_2|Y=stay) = \begin{cases} \frac{1}{5}, & X_2 = working \\ \frac{4}{5}, & X_2 = broken \end{cases}$$

In tabular form:

prior	$\mathbb{P}(X_1 = x_1 \vert Y = j)$	$\mathbb{P}(X_2 = x_2 \vert Y = j)$
$\mathbb{P}(Y = out) = 0.5$	$\mathbb{P}(sunny \vert out) = \frac{4}{5}$	$\mathbb{P}(working \vert out) = \frac{4}{5}$
$\mathbb{P}(Y = stay) = 0.5$	$\mathbb{P}(sunny \vert stay) = \frac{3}{5}$	$\mathbb{P}(working \vert stay) = \frac{1}{5}$
$\mathbb{P}(Y = out) = 0.5$	$\mathbb{P}(rainy \vert out) = \frac{1}{5}$	$\mathbb{P}(broken \vert out) = \frac{1}{5}$
$\mathbb{P}(Y = stay) = 0.5$	$\mathbb{P}(rainy \vert stay) = \frac{2}{5}$	$\mathbb{P}(broken \vert stay) = \frac{4}{5}$

In Python, `CategoricalNB` implements the categorical Naïve Bayes algorithm for categorically distributed data. It assumes that each feature, X_i, has its own categorical distribution and the sample matrix is encoded (for instance with the help of `OrdinalEncoder`) such that all categories for each feature are represented with numbers $0, \cdots, M_j - 1$. The probability of category c in feature X_j given class k is estimated as:

$$\mathbb{P}(X_j = c|Y = k;\ \alpha) = \frac{\#(X_j = c \ \&\ Y = k) + \alpha}{\#(Y = k) + \alpha d_j} \tag{4.7}$$

where α is a **smoothing parameter** and d_j is the number of available categories of feature X_j defined above. Setting $\alpha = 1$ is called *Laplace smoothing*, while $\alpha < 1$ is called *Lidstone smoothing*.

Categorical Naïve Bayes (Classifier) in Python

```
from sklearn.naive_bayes import CategoricalNB
CategoricalNB(alpha=1.0, fit_prior=True, class_prior=None)
```

4.2.2 Gaussian Naïve Bayes

When a feature X_j in (4.4) is continuous, we usually approximate $\mathbb{P}(X_j|Y = k)$ for a fixed k using a Gaussian distribution (unless there is a strong proof that it is other kind of distribution):

$$\mathbb{P}(X_j = x|Y = k) = \frac{1}{\sigma_{jk}\sqrt{2\pi}} \exp(-\frac{(x - \mu_{jk})^2}{2\sigma_{jk}^2}). \tag{4.8}$$

The theoretical mean and theoretical standard deviation in (4.8) are

$$\mu_{jk} = \mathbb{E}[X_j|Y = k], \quad \sigma_{jk}^2 = \mathbb{E}[(X_j - \mu_{jk})^2|Y = k], \quad k = 1, \ldots, K. \tag{4.9}$$

The maximum likelihood estimator for (4.9) are

$$\widehat{\mu}_{jk} = \frac{\sum_{i=1}^n x_{ij} I(y_i = k)}{\sum_{i=1}^n I(y_i = k)}, \quad s_{jk}^2 = \frac{\sum_{i=1}^n (x_{ij} - \widehat{\mu}_{jk})^2 I(y_i = k)}{(\sum_{i=1}^n I(y_i = k)) - 1}, \quad k = 1, .., K. \tag{4.10}$$

Here j refers to the jth column of the tabular data while k refers to all rows with $Y = k$ and the i goes through all rows in the tabular data as we will see in the following example.

Example 4.2.3. The table below shows the data collected for predicting whether a customer will default on the credit card or not:

customer	balance	student	Default
1	500	No	N
2	1980	Yes	Y
3	60	No	N
4	2810	Yes	Y
5	1400	No	N
6	300	No	N
7	2000	Yes	Y
8	940	No	N
9	1630	No	Y
10	2170	Yes	Y

(a) Compute the probability density of customer with balance 2080, given Default=Y.

Solution: Let X_1 be `balance`. Then

$$\mathbb{P}(X_1 = 2080 \mid \texttt{Default} = Y) = \frac{1}{s_{1;Y}\sqrt{2\pi}} \exp(-\frac{(2080-\mu_{1;Y})^2}{2s_{1;Y}^2}) = 0.0009162$$

where

$$\mu_{1;Y} = \frac{\sum_{i=1}^{10} x_{i1} I(\texttt{Default} = Y)}{\sum_{i=1}^{10} I(\texttt{Default} = Y)} = \frac{1980+2810+2000+1630+2170}{5} = 2118;$$

$$s_{i;Y} = \sqrt{\frac{\sum_{i=1}^{10}(x_{i1}-\widehat{\mu}_{1;Y})^2 I(y_i = Y)}{(\sum_{i=1}^{10} I(\texttt{Default} = Y)) - 1}} = 433.7857$$

(b) Compute the probability of customer who is a student, given Default=Y.

Solution: $\mathbb{P}(\texttt{student} = Yes \mid \texttt{Default} = Y) = \frac{4}{5}$

(c) Calculate the "probability density" of default for a student customer with balance 2080 by using the Naïve Bayes assumption.

Solution: $\mathbb{P}(\texttt{student} = Yes \mid \texttt{Default} = N) = \frac{0}{5} = 0$

$$\begin{aligned}
&\mathbb{P}(\texttt{Default} = Y \mid \texttt{balance} = 2080,\ \texttt{student} = Yes) \\
=&\frac{\mathbb{P}(\texttt{balance} = 2080,\ \texttt{student} = Yes \mid \texttt{Default} = Y)\mathbb{P}(\texttt{Default} = \texttt{Y})}{\mathbb{P}(\texttt{balance} = 2080,\ \texttt{student} = Yes) =: \mathbb{P}(...)} \\
=&\frac{\mathbb{P}(\texttt{balance} = 2080,\ \texttt{student} = Yes \mid \texttt{Default} = Y)\mathbb{P}(\texttt{Default} = \texttt{Y})}{\mathbb{P}(... \mid \texttt{Default} = Y)\mathbb{P}(\texttt{Default} = \texttt{Y}) + \mathbb{P}(... \mid \texttt{Default} = N)\mathbb{P}(\texttt{Default} = \texttt{N})} \\
=&\frac{0.0009162 \times \frac{4}{5} \times \frac{5}{10}}{0.0009162 \times \frac{4}{5} \times \frac{5}{10} + \mathbb{P}(\texttt{balance} = 2080 | \texttt{Default} = N) \times 0 \times \frac{5}{10}} = 1
\end{aligned}$$

(d) Write down the full naive Bayes model from the data.

Solution: Let x_1=balance, x_2=student, y=Default. The Naive Bayes Model is

$$h_D(x_1, x_2) = \underset{j}{\operatorname{argmax}}\, \mathbb{P}(x_1|y=j)\mathbb{P}(x_2|y=j)\mathbb{P}(y=j).$$

where the prior $\mathbb{P}(y) = \begin{cases} 0.5 & y = N \\ 0.5 & y = Y \end{cases}$

$$\mathbb{P}(x_2|y=N) = \begin{cases} 1 & x_2 = No \\ 0 & x_2 = Yes \end{cases} \qquad \mathbb{P}(x_2|y=Y) = \begin{cases} 1/5 & x_2 = No \\ 4/5 & x_2 = Yes \end{cases}$$

$$\mathbb{P}(x_1|y=N) = \frac{1}{\sqrt{2\pi}(533.6666)} \exp\left\{-\frac{(x_1-640)^2}{2(533.6666)^2}\right\}$$

$$\mathbb{P}(x_1|y=Y) = \frac{1}{\sqrt{2\pi}(433.7857)} \exp\left\{-\frac{(x_1-2118)^2}{2(433.7857)^2}\right\}$$

Note that the $\mathbb{P}(\texttt{balance} = 2080 | \texttt{Default} = N)$ in part (c) was not calculated because it will be multiplied with 0. The calculation is similar to part (a):

$$\mathbb{P}(\texttt{balance} = 2080 \mid \texttt{Default} = N) = \frac{1}{s_{i;N}\sqrt{2\pi}} \exp(-\frac{(2080-\mu_{i;N})^2}{2s_{i;N}^2}) = 1.9616 \times 10^{-5}$$

Example 4.2.4 (Final Exam Jan 2019, Q3(c)). A more efficient marketing strategy can be achieved by targeting the customers who have higher probability to complete a purchase. Hence, you have been asked to predict whether a customer will buy the product based on their demographic data such as age, race, gender and income. Table Q3(c) shows the data collected from previous records.

Cust.	Age	Race	Gender	Income	Result
1	52	Indian	Male	RM 11,500	Not Buy
2	22	Chinese	Female	RM 6,500	Buy
3	30	Chinese	Male	RM 8,000	Buy
4	26	Malay	Male	RM 8,500	Buy
5	27	Indian	Female	RM 6,500	Buy
6	32	Chinese	Female	RM 9,500	Not Buy
7	33	Indian	Male	RM 4,000	Not Buy
8	50	Malay	Female	RM 10,000	Buy
9	31	Chinese	Female	RM 5,500	Buy
10	27	Malay	Male	RM 9,200	Not Buy

Table Q3(c)

(i) State an assumption used in Naïve Bayes approach. (1 mark)

(ii) Using Naïve Bayes approach without Laplace smoothing, predict whether a Malay female customer, aged 29, with income RM7,800, will buy the product. (9 marks)

Gaussian Naïve Bayes (Classifier) in Python

```
from sklearn.naive_bayes import GaussianNB
GaussianNB(priors=None, var_smoothing=1e-09)
```

Here `var_smoothing` is a portion of the largest variance of all numeric features that is added to variances for calculation stability.

Example 4.2.5. The Iris dataset (`https://archive.ics.uci.edu/ml/datasets/Iris`) consists four attributes: sepal length, sepal width, petal length and petal width, all in cm; and a label of 3 classes (Iris Setosa, Iris Versicolour, Iris Virginica).

A script to train and test the dataset in Python is listed below.

```
# https://www.python-course.eu/naive_bayes_classifier_scikit.php
from sklearn import datasets, metrics, naive_bayes
dataset = datasets.load_iris()
model = naive_bayes.GaussianNB()
model.fit(dataset.data, dataset.target)
print(model)
expected  = dataset.target
predicted = model.predict(dataset.data)
# summarise the fit of the model
print(metrics.confusion_matrix(predicted, expected))
print(metrics.classification_report(predicted, expected))
```

4.2.3 Laplace Smoothing

When one of the **categorical feature** has **zero numerator** or one of the **numeric feature** has **zero or extremely small variance**, the Naïve Bayes classifier may perform poorly. A proposal to improve the performance is to apply the **Laplace smoothing** or `https://en.wikipedia.org/wiki/Additive_smoothing` to the Naïve Bayes classifier. For a categorical feature X_j, it is accomplished by adding α (by default, $\alpha = 1$) as shown in (4.7).

The parameter α is **usually chosen** to be 1 because in such case the symmetric Dirichlet prior is equivalent to the uniform distribution and for bigger number of observations in the data such prior has small effect on the estimates. The other popular value for α is 0.5, which corresponds to the popular (non-informative) Jeffreys prior. We can imagine the Laplace smoothing as adding some "pseudosamples" into the data to ensure that each class occurs at least once to avoid zero in the probability/density function.

The situation of a categorical feature having zero probability is illustrated in the Example 4.2.3 and in the following example illustrates how Laplace smoothing can provide an improvement by avoiding the zero probability.

Example 4.2.6. Redo Example 4.2.3 by applying the Laplace smoothing.

Solution: The naive Bayes model for the "continuous" feature is the same:

$$\mathbb{P}(\texttt{balance} = 2080 \mid \texttt{Default} = N) = 1.9616 \times 10^{-5}.$$

because the variance is not too small.

Applying the Laplace smoothing with $\alpha = 1$ (the default) to the categorical variable `student` leads to

- `student`=Yes or No $\Rightarrow \boxed{d = 2}$
- $\mathbb{P}(\texttt{student} = Yes \mid \texttt{Default} = N) = \dfrac{0+1}{5+2}$

Now, the conditional probability will no longer zero:

$$\begin{aligned}
&\mathbb{P}(\texttt{Default} = N \mid \texttt{balance} = 2080,\ \texttt{student} = Yes)\\
&= \tfrac{\mathbb{P}(\texttt{balance}=2080,\texttt{student}=Yes \mid \texttt{Default}=N)\mathbb{P}(\texttt{Default}=N)}{\mathbb{P}(\ldots \mid \texttt{Default}=N)\mathbb{P}(\texttt{Default}=N)+\mathbb{P}(\ldots \mid \texttt{Default}=Y)\mathbb{P}(\texttt{Default}=Y)}\\
&= \tfrac{1.9616\times 10^{-5}\times\frac{1}{7}\times\frac{5}{10}}{1.9616\times 10^{-5}\times\frac{1}{7}\times\frac{5}{10}+0.000916154\times\frac{5}{7}\times\frac{5}{10}}\\
&= \tfrac{1.401151e-06}{1.401151e-06+0.0003271979} = 0.004264014
\end{aligned}$$

Example 4.2.7 (May 2022 Semester Final Exam, Q4(a))**.** The data in Table 4.1 from a study of low birth weight infants in a neonatal intensive care unit is used to examine the development of bronchopulmonary dysplasia (BPD), a chronic lung disease, in a sample of 10 infants weighing less than 1750 grams. The response variable `bpd` is binary (0 denotes 'no' and 1 denotes 'yes'), denoting whether an infant develops BPD by day 28 of life. The predictors are listed below:

- `brthwght`: birth weight in number of grams;
- `gestage`: gestational age in number of weeks;
- `toxemia`: a condition in pregnancy characterised by abrupt hypertension, albuminuria and edema. This is a binary variable with 0 = no, 1 = yes.

`brthwght`	`gestage`	`toxemia`	`bpd`
850	30	0	0
1400	30	0	0
1720	30	0	0
1150	32	0	0
1610	34	0	0
1230	32	1	0
820	26	0	1
840	30	0	1
1060	30	0	1
1140	34	0	0

Table 4.1: Training data for a study of low birth weight infants.

Use the Naïve Bayes classifier model **with Laplace smoothing** to predict if an infant has the BPD problem given that the infant has a birth weight of 1120 grams, has a gestational age 29 weeks and the pregnancy does not have toxemia. (12 marks)

4.2.4 Relation of GNB to Logistic Regression

When Y is Boolean (assuming the values to be 1 and 0 and $\mathbb{P}(Y = 1) + \mathbb{P}(Y = 0) = 1$), the Gaussian naïve Bayes (GNB) classifier is equivalent to the multiple logistic regression (3.2) under suitable assumptions. The illustration is as follows: From (4.2) and (4.8), we have

$$\begin{aligned}&\mathbb{P}(Y=1|\boldsymbol{X}=\boldsymbol{x})\\&=\frac{\mathbb{P}(Y=1)\mathbb{P}(X_1=x_1|Y=1)\cdots\mathbb{P}(X_p=x_p|Y=1)}{\mathbb{P}(Y=1)\mathbb{P}(X_1=x_1|Y=1)\cdots\mathbb{P}(X_p=x_p|Y=1)+\mathbb{P}(Y=0)\mathbb{P}(X_1=x_1|Y=0)\cdots\mathbb{P}(X_p=x_p|Y=0)}\end{aligned}$$

$$= \frac{1}{1 + \frac{\mathbb{P}(Y=0)\mathbb{P}(X_1=x_1|Y=0)\cdots\mathbb{P}(X_p=x_p|Y=0)}{\mathbb{P}(Y=1)\mathbb{P}(X_1=x_1|Y=1)\cdots\mathbb{P}(X_p=x_p|Y=1)}}$$

$$= \frac{1}{1 + \exp\left(\ln \frac{1-\mathbb{P}(Y=1)}{\mathbb{P}(Y=1)} - \frac{(x_1-\mu_{11})^2}{2\sigma_{11}^2} + \frac{(x_1-\mu_{10})^2}{2\sigma_{10}^2} - \cdots - \frac{(x_p-\mu_{p1})^2}{2\sigma_{p1}^2} + \frac{(x_p-\mu_{p0})^2}{2\sigma_{p0}^2}\right)}$$

If we **assume** that $\sigma_{i1} = \sigma_{i0} = \sigma_i$ (which is not true in general based on our earlier calculations), we can have

$$-\frac{(x_i - \mu_{i1})^2}{2\sigma_{i1}^2} + \frac{(x_i - \mu_{i0})^2}{2\sigma_{i0}^2} = \frac{(x_i^2 - 2\mu_{i0}x_i + \mu_{i0}^2) - (x_i^2 - 2\mu_{i1}x_i + \mu_{i1}^2)}{2\sigma_i^2} = \frac{-2(\mu_{i0} - \mu_{i1})x_i + (\mu_{i0}^2 - \mu_{i1}^2)}{2\sigma_i^2}.$$

This can be written as a linear term $b_i - w_i x_i$, $i = 1, \cdots, p$ and

$$\mathbb{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{1 + \exp\left(\ln \frac{1-\mathbb{P}(Y=1)}{\mathbb{P}(Y=1)} + b_1 + \cdots + b_p - w_1 x_1 - \cdots - w_p x_p\right)}$$

which is equivalent to the multiple logistic regression (3.2).

The logistic regression directly estimates the parameters of $\mathbb{P}(Y|\boldsymbol{X})$, whereas naïve Bayes directly estimates parameters for $\mathbb{P}(Y)$ and $\mathbb{P}(\boldsymbol{X}|Y)$. The former is known as a ***discriminative classifier***, and the latter is known as a ***generative classifier***. We have shown that the assumptions of a special type of GNB classifier imply the parametric form of $P(Y|\boldsymbol{X})$ used in logistic regression. In fact, if the σ-assumption holds, then asymptotically the GNB and logistic regression converge toward identical classifiers.

According to `https://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf`, the two algorithms also differ in interesting ways:

- When the GNB σ-assumption do not hold, logistic regression and GNB typically learn different classifier functions. In this case, the asymptotic classification accuracy for logistic regression is often better than the asymptotic accuracy of GNB. Although logistic regression is consistent with the GNB's assumption that the input features X_i are **conditionally independent** given Y, it is not rigidly tied to this assumption as is GNB. Given data that disobeys this assumption, the conditional likelihood maximisation algorithm for logistic regression will adjust its parameters to maximise the fit to (the conditional likelihood of) the data, even if the resulting parameters are inconsistent with the Naïve Bayes parameter estimates.

- GNB and logistic regression converge toward their asymptotic accuracies at different rates. GNB parameter estimates converge toward their asymptotic values in order $\log n$ examples, where n is the dimension of X. In contrast, logistic regression parameter estimates converge more slowly, requiring order n examples. In certain datasets, logistic regression outperforms GNB when many training examples are available, but GNB outperforms logistic regression when training data is scarce.

In summary, logistic regression is the discriminative counterpart to Naive Bayes. In Naive Bayes, we first model $P(\mathbf{x}|y)$ for each label y, and then obtain the decision boundary that best discriminates between these two distributions. In logistic regression we do not attempt to model the data distribution $P(\mathbf{x}|y)$, instead, we model $P(y|\mathbf{x})$ directly. We assume the same probabilistic form $P(y|\mathbf{x}_i) = \frac{1}{1+e^{-y(\mathbf{w}^T\mathbf{x}_i+b)}}$, but we do not restrict ourselves in any way by making assumptions about $P(\mathbf{x}|y)$ (in fact it can be any member of the Exponential Family). This allows logistic regression to be more flexible, but such flexibility also requires more data to avoid overfitting. Typically, in scenarios with little data and if the modelling assumption is appropriate, Naive Bayes tends to outperform Logistic Regression. However, as data sets become large logistic regression often outperforms Naive Bayes, which suffers from the fact that the assumptions made on $P(\mathbf{x}|y)$ are probably not exactly correct. If the assumptions hold exactly, i.e. the data is truly drawn from the distribution that we assumed in Naive Bayes, then Logistic Regression and Naive Bayes converge to the exact same result in the limit (but NB will be faster).

4.2.5 Multinomial Naïve Bayes and Its Variants

According to `https://blog.floydhub.com/naive-bayes-for-machine-learning/`, the conventional version of the Naïve Bayes (NB) is the Gaussian NB, which works best for continuous types of data. The underlying assumption of Gaussian NB is that the features follow a normal distribution. The other variants are best used for text classification problems, wherein the data features are discrete. BernoulliNB is the NB version where the features are vectorised in a binary fashion. Whereas, MultinomialNB is the non-binary version of BernoulliNB. As the word implies, multinomial means "many counts". Complement NB (CNB) algorithm is an adaptation of the standard MultinomialNB algorithm that is particularly suited for imbalanced data sets wherein the algorithm uses statistics from the complement of each class to compute the model's weight. The inventors of CNB show empirically that the parameter estimates for CNB are more stable than those for MNB. Furthermore, CNB regularly outperforms MNB (often by a considerable margin) on text classification tasks.

The Naïve Bayes algorithm for multinomially distributed data is called a *multinomial Naïve Bayes (MNB) classifier*:

$$\begin{aligned} h_D(\texttt{document}) &= \underset{k=1,\cdots,K}{\operatorname{argmax}} \mathbb{P}(\texttt{document}|Y=k)\mathbb{P}(Y=k) \\ &= \underset{k=1,\cdots,K}{\operatorname{argmax}} \mathbb{P}(wc_1, wc_2, \cdots, wc_p|Y=k)\mathbb{P}(Y=k) \end{aligned} \tag{4.11}$$

where wc_j is the number of times the word X_j, $j = 1, \cdots, p$, occurred in the `document`, p is the size of the vocabulary (number of features). In real-world, possible entries of "classes" Y for `document` are "scientific", "economic", "management", etc.

The model (4.11) is a Naïve Bayes variant used in **text classification** (where the data are typically represented as word vector counts).

The estimates for the prior probabilities and likelihood fuction are

$$\begin{aligned} \mathbb{P}(Y=k) &\approx \frac{\text{number of documents of class } k}{\text{number of documents, } n}; \\ \mathbb{P}(X_j = wc_j|Y=k) &\approx \frac{\text{total number of the occurrences of the word } X_j \text{ in documents of class } k}{\text{total number of words } X_1, \cdots, X_p \text{ in documents of class } k} \\ &= \frac{\sum_{y_i=k} wc_j + \alpha}{\sum_{j=1}^p \sum_{y_i=k} wc_j + \alpha d_j} = \frac{N_{kj} + \alpha}{N_k + \alpha d_j} =: \theta_{kj}. \end{aligned}$$

where the number of times feature j appears in a sample of class k in the training set D and the total count of all features for class k are respectively

$$N_{kj} = \sum_{y_i=k} wc_j, \quad N_k = \sum_{j=1}^p N_{kj}.$$

The conditional probability of the multinomial naive Bayes model is [Manning et al., 2009, Chapter 13]

$$\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|Y=k) = \frac{(\sum_{j=1}^p wc_j)!}{wc_1! \times \cdots \times wc_p!} \prod_{j=1}^p \theta_{kj}^{wc_j}. \tag{4.12}$$

A *complement Naïve Bayes* (CNB) algorithm is an adaptation of the standard MNB algorithm that is particularly suited for imbalanced data sets. Specifically, CNB uses statistics from the complement of each class to compute the model's weights. The inventors of CNB show empirically that the parameter estimates for CNB are more stable than those for MNB [Rennie et al., 2003]. Further, CNB regularly outperforms MNB (often by a considerable margin) on text classification tasks. The procedure for calculating the weights is as follows:

$$\widehat{\theta}_{kj} = \frac{\sum_{y_i \neq k} wc_j + \alpha}{\sum_{j=1}^{p} \sum_{y_i \neq k} wc_j + \alpha d_j}$$

where the summations are over all documents i not in class k, wc_j is either the count or `https://en.wikipedia.org/wiki/Tf-idf` (term frequency-inverse document frequency) of term j (in document k); α is the Laplace smoothing hyperparameter like that found in MNB. The second normalisation addresses the tendency for longer documents to dominate parameter estimates in MNB. In the CNB, a document is assigned to the class that is the poorest complement match.

Bernoulli Naïve Bayes is used when the data is distributed according to multivariate Bernoulli distributions i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, Boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors.

For Bernoulli Naïve Bayes, the conditional probability (4.12) becomes

$$\mathbb{P}(\boldsymbol{X} = \boldsymbol{x} | Y = k) = \prod_{j=1}^{p} \theta_{kj}^{x_j} (1 - \theta_{kj}^{x_j})^{1-x_j} \tag{4.13}$$

with $x_j = I(wc_j > 0)$ being binary. It differs from MNB's rule in that **it explicitly penalises the non-occurrence of a feature that is an indicator for class** k, where the MNB variant would simply ignore a non-occurring feature.

Multinomial naïve Bayes classifiers are used in *email spam filtering.* They typically use bag of words features to identify spam email, an approach commonly used in text classification. They work by correlating the use of tokens (typically words, or sometimes other things), with spam and non-spam emails. Naïve Bayes spam filtering is a baseline technique for dealing with spam that can tailor itself to the email needs of individual users and give low false positive spam detection rates that are generally acceptable to users. It is one of the oldest ways of doing spam filtering, with roots in the 1990s. Examples are found in a few books such as Lantz [2015, Chapter 4].

Multinomial Naïve Bayes Classifier and Its Variants in Python

```
from sklearn.naive_bayes import *
MultinomialNB(alpha=1.0, fit_prior=True, class_prior=None)
ComplementNB(alpha=1.0, fit_prior=True, class_prior=None, norm=False)
BernoulliNB(alpha=1.0, binarize=0.0, fit_prior=True, class_prior=None)
```

Example 4.2.8. Real-world application is complex. There are a lot of information on the Internet, so I will not create a "fake" example, but just use the a Python example.

```
# https://jakevdp.github.io/PythonDataScienceHandbook/05.05-naive-bayes.html
from sklearn.datasets import fetch_20newsgroups
# Downloads https://ndownloader.figshare.com/files/5975967 and
# put in under ~/scikit_learn_data/
data = fetch_20newsgroups()
data.target_names

categories = ['talk.religion.misc', 'soc.religion.christian',
              'sci.space', 'comp.graphics']
```

```
train = fetch_20newsgroups(subset='train', categories=categories)
test  = fetch_20newsgroups(subset='test',  categories=categories)

from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline

# https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn
# https://scikit-learn.org/stable/modules/feature_extraction.html
vectoriser = CountVectorizer()
vectoriser.fit(train.data)
print(vectoriser.vocabulary_)
feature_table = vectoriser.transform(train.data)
model = make_pipeline(TfidfVectorizer(), MultinomialNB())
model.fit(train.data, train.target)
labels = model.predict(test.data)

from sklearn.metrics import confusion_matrix
import seaborn as sns, matplotlib.pylab as plt
mat = confusion_matrix(test.target, labels)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=train.target_names,
            yticklabels=train.target_names)
plt.xlabel('true label')
plt.ylabel('predicted label')
```

4.3 Discriminant Analysis

Discriminant Analysis (DA) is a generative model (4.3) which assumes that the predictors $\boldsymbol{X} = (X_1, X_2, \cdots, X_p)$ are numeric and are drawn from a *multivariate Gaussian* (or a *multivariate normal*) distribution, $Normal(\boldsymbol{\mu}, \boldsymbol{C})$. Therefore, the "$\mathbb{P}$" in (4.3) should be regarded as "probability density" because the predictors are numeric.

DA parallels multiple regression analysis and is closely related to analysis of variance (ANOVA), logistic regression and principle component analysis (PCA). The main difference between DA and regression is that regression analysis deals with a continuous dependent variable, while discriminant analysis must have a discrete dependent variable.

Two important DA models are the *linear discriminant analysis (LDA)* models and *quadratic discriminant analysis (QDA)* models.

In the **QDA models**, $\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|Y = k)$ is assumed to have the following form:

$$\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|Y = k) = \frac{1}{(2\pi)^{p/2}\sqrt{|\boldsymbol{C}_k|}} \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^T \boldsymbol{C}_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k)\right\} \tag{4.14}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{C}_k$ are the class-specific mean vector and the class-specific covariance matrix respectively.

By substituting (4.14) into (4.3), we have

$$
\begin{aligned}
h_D(\boldsymbol{x}) &= \underset{k\in\{1,\ldots,K\}}{\operatorname{argmax}} \left[\ln \mathbb{P}(Y=k) + \ln \frac{1}{(2\pi)^{p/2}\sqrt{|\boldsymbol{C}_k|}} \exp\left\{ -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{C}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) \right\} \right] \\
&= \underset{k\in\{1,\ldots,K\}}{\operatorname{argmax}} \left[\ln \mathbb{P}(Y=k) - \frac{1}{2}\ln|\boldsymbol{C}_k| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{C}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) \right]
\end{aligned} \tag{4.15}
$$

From here, we can obtain the "discriminant functions" for QDA:

$$
\delta_k(\boldsymbol{x}) = \ln \mathbb{P}(Y=k) - \frac{1}{2}\ln|\boldsymbol{C}_k| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{C}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k), \quad k = 1,\cdots,K. \tag{4.16}
$$

Note that the "boundary" between classes is a quadratic surface according to (4.16).

The prior distribution $\mathbb{P}(Y=k)$ is usually estimated according to (4.5) while the class-specific covariance matrix, $\boldsymbol{C}_k$, has the following estimation:

$$
\widehat{\boldsymbol{C}}_k = \frac{1}{n_k - 1}\sum_{i=1}^{n}(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)^T I(y_i = k). \tag{4.17}
$$

In the **LDA models**, $\mathbb{P}(\boldsymbol{X}=\boldsymbol{x}|Y=k)$ is assumed to be multivariate normal distribution (4.14) with a common covariance matrix $\boldsymbol{C}$:

$$
\boldsymbol{C}_1 = \boldsymbol{C}_2 = \cdots = \boldsymbol{C}_K =: \boldsymbol{C}.
$$

In such a case, the derivation is similar to QDA but the generative model (4.15) will become

$$
\begin{aligned}
h_D(\boldsymbol{x}) &= \underset{k\in\{1,\ldots,K\}}{\operatorname{argmax}} \left[\ln \mathbb{P}(Y=k) - \frac{1}{2}\ln|\boldsymbol{C}| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) \right] \\
&= \underset{k\in\{1,\ldots,K\}}{\operatorname{argmax}} \left[\ln \mathbb{P}(Y=k) - \frac{1}{2}\left(\boldsymbol{x}^T\boldsymbol{C}^{-1}\boldsymbol{x} - \boldsymbol{\mu}_k^T\boldsymbol{C}^{-1}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{C}^{-1}\boldsymbol{\mu}_k + \boldsymbol{\mu}_k^T\boldsymbol{C}^{-1}\boldsymbol{\mu}_k \right) \right] \\
&= \underset{k\in\{1,\ldots,K\}}{\operatorname{argmax}} \left[\ln \mathbb{P}(Y=k) - \frac{1}{2}\left(-2\boldsymbol{\mu}_k^T\boldsymbol{C}^{-1}\boldsymbol{x} + \boldsymbol{\mu}_k^T\boldsymbol{C}^{-1}\boldsymbol{\mu}_k \right) \right] \\
&= \underset{k\in\{1,\ldots,K\}}{\operatorname{argmax}} \left[\ln \mathbb{P}(Y=k) + \boldsymbol{\mu}_k^T\boldsymbol{C}^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{C}^{-1}\boldsymbol{\mu}_k \right].
\end{aligned} \tag{4.18}
$$

From here, we obtain the *discriminant function* for LDA:

$$
\delta_k(\boldsymbol{X}) = \ln \mathbb{P}(Y=k) - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{C}^{-1}\boldsymbol{\mu}_k + \boldsymbol{\mu}_k^T\boldsymbol{C}^{-1}\boldsymbol{x}. \tag{4.19}
$$

Note that the "boundary" between classes is a linear space according to (4.19). The name of LDA is from the fact that the discriminant functions, $\delta_k(\boldsymbol{x})$ are linear functions of $\boldsymbol{x}$.

The estimation of the parameters in the discriminant function (4.19) using the **standard moment estimators** is as follows [Hastie et al., 2008, p109].

- The prior probability $\mathbb{P}(Y=k)$ is approximated by the fraction of training samples of class k, (4.5).

- The centre of each class $\boldsymbol{\mu}_k$ has the estimation

$$\widehat{\boldsymbol{\mu}}_k = \frac{\sum_{i=1}^n \boldsymbol{x}_i I(y_i = k)}{\sum_{i=1}^n I(y_i = k)}. \tag{4.20}$$

- The common covariance matrix $\boldsymbol{C}$ is an unbiased estimate of its covariance matrix of the vectors of deviations $(\boldsymbol{x}_1 - \widehat{\boldsymbol{\mu}}_{y_1})$, $(\boldsymbol{x}_2 - \widehat{\boldsymbol{\mu}}_{y_2})$, $\cdots$, $(\boldsymbol{x}_n - \widehat{\boldsymbol{\mu}}_{y_n})$:

$$\widehat{\boldsymbol{C}} = \frac{1}{n-K} \sum_{k=1}^K \sum_{i=1}^n (\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)^T I(y_i = k). \tag{4.21}$$

Computer implementation of LDA is more complex than the steps mentioned earlier. Here is some description of algorithms used in computer implementations.

The dimensionality reduction procedure of LDA involves the within-class variance, $W = \hat{\Sigma}$, and the between-class variance, B. The between-class variance indicates the deviation of centroids from the overall mean, $\hat{\mu} = \sum_{k=1}^K \hat{\pi}_k \hat{\mu}_k$, and is defined as:

$$B = \sum_{k=1}^K \hat{\pi}_k (\hat{\mu}_k - \hat{\mu})(\hat{\mu}_k - \hat{\mu})^T .$$

Finding a sequence of optimal substeps involves three steps:

1. Compute the $K \times p$ matrix M containing the centroids, μ_k, and determine the common covariance matrix W.
2. Compute $M^* = MW^{-\frac{1}{2}}$ using the eigen-decomposition of W.
3. Compute B^* (the between-class covariance) and its eigen-decomposition $B^* = V^* D_B V^{*T}$. The columns v_l^* of V^* define the coordinates of the reduced subspace.

The l-th discriminant variable (one of the $K-1$ new dimensions) is determined by $Z_l = v_l^T X$ with $v_l = W^{-\frac{1}{2}} v_l^*$.

Fisher's LDA optimisation criterion states that the centroids of the groups should be spread out as far as possible. This amounts to finding a linear combination $Z = a^T X$ such that a^T maximizes the between-class variance relative to the within-class variance.

As before, the within-class variance is W is the pooled covariance matrix, $\hat{\Sigma}$, which indicates the deviation of all observations from their class centroids. The between-class variance is defined according to the deviation of the centroids from the overall mean, as defined earlier. For Z, the between class variance is $a^T Ba$ and the within-class variance is $A^T Wa$. Thus, LDA can be optimized through the Rayleigh quotient

$$\max_a \frac{a^T Ba}{a^T Wa} ,$$

which defines an optimal mapping of X to the new space Z. Note that $Z \in \mathbb{R}^{1\times p}$, that is, the observations are mapped to a single dimension. To obtain additional dimensions, we need to solve the optimization problem for $a_1, \ldots, a_{K-1}$ where each successive a_k is constructed to be orthogonal in W to the previous discriminant coordinates. This leads to the linear transformation $G = (Z_1^T, Z_2^T, \ldots, Z_{K-1}^T) \in \mathbb{R}^{p\times q}$ with which we can map from p to q dimension via XG. Why do we consider $K-1$ projections? This is because the affine subspace that is spanned by the K centroids has a rank of at most $K-1$.

Using Fisher's formulation of LDA, classification involves two steps:

1. Sphere the data using the common covariance matrix $\hat{\Sigma} = UDU^T$ (eigendecomposition) such that $X^* = D^{-\frac{1}{2}}U^TX$. In this way, the covariance of X^* becomes the identity matrix. By eliminating the covariance between the variables in this way, it becomes much easier to separate the classes in the transformed space.

2. Classify observations x_i to the closest class centroid in the transformed space, taking into account the class priors π_k. Here, the intuition is that an observation with equal distance to two centroids would be assigned to the class with the greater prior.

LDA performs classification in a reduced subspace. When performing classification, we do not need to use all $K-1$ dimensions and instead can choose a smaller subspace H_l with $l < K-1$. When $l < K-1$ is used, this is called *reduced-rank LDA*. The motivation for reduced-rank LDA is that classification basd on a reduced number of discriminant variables can improve performance on the test set when the model is overfitted.

The number of effective parameters of LDA can be derived in the following way. There are K means, $\hat{\mu}_k$ that are estimated. The covariance matrix does not require additional parameters because it is already defined by the centroids. Since we need to estimate K discriminant functions (to obtain the decision boundaries), this gives rise to K calculations involving the p elements. Additionally, we have $K-1$ free parameters for the K priors. Thus, the number of effective LDA parameters is $Kp + (K-1)$.

Example 4.3.1 (Final Exam May 2019, Q4)**.** (a) State the advantages of using linear discriminant analysis in classification as compared to logistic regression. (3 marks)

Solution:

- LDA is more stable when the classes are well-separated.
- If n is small and the distribution of the predictors $\boldsymbol{X}$ is approximately normal in each of the classes, LDA is more stable than logistic regression.
- When we have more than two response classes, LDA is more popular.

If by any chance, the assumptions of LDA hold for the true distribution of the data, then LDA is optimal in the sense that it **converges** to the Bayes classifier (Section 2.2) when the number of data tends towards infinity (at which point the parameter estimates coincide with the true distribution parameters).

(b) (i) State two assumptions made in linear discriminant analysis. (2 marks)

(ii) Bayes' theorem states that posterior probability to estimate probability of a new observation belongs to the jth class can be written as

$$\mathbb{P}(Y = j|X = x) = \frac{\pi_j \mathbb{P}(x|Y = j)}{\sum_{i=1}^{K} \pi_i \mathbb{P}(x|Y = i)}.$$

where $\pi_j = \mathbb{P}(Y = j)$. Classification is done by assigning an observation to the class which posterior probability, $\mathbb{P}(Y = j|X = x)$ is the largest. Linear discriminant analysis involves assigning the observation to the class for which discriminant function, $\delta_j(X)$ is the largest. For linear discriminant analysis with one predictor, the discriminant function is

$$\delta_j(x) = x \cdot \frac{\mu_j}{\sigma^2} - \frac{\mu_j^2}{2\sigma^2} + \ln(\pi_j).$$

With assumptions stated in Q4(b)(i), show how discriminant function, $\delta_j(x)$ can be equivalent to posterior probability, $\mathbb{P}(Y = j|X = x)$ in linear discriminant analysis with one predictor. (10 marks)

(c) A teacher is preparing an extra class for the students who are predicted to fail in their final exam. The teacher would like to predict the performance of the current students in final exam (fail/pass). You are to build a model for the teacher by using 500 previous students' record. Below shows some information and analysis of the previous record:

(I) The coursework consisted of Assignment, Quiz and Test.

(II) Average mark for students who passed in Assignment was 73.9; whereas average mark for students who failed in Assignment was 51.4.

(III) Average mark for students who passed in Quiz was 68.2; whereas average mark for students who failed in Quiz was 42.3.

(IV) Average mark for students who passed in Test was 63.7; whereas average mark for students who failed in Test was 35.6.

(V) There were 380 students passed the final exam.

(VI) The inverse of the group covariance matrix for the collected data is

$$C^{-1} = \begin{bmatrix} 0.0022 & 0.0132 & 0.0095 \\ 0.0132 & 0.0074 & 0.0108 \\ 0.0095 & 0.0108 & 0.0180 \end{bmatrix}$$

where x_1 = Assignment; x_2 = Quiz; x_3 = Test.

Using linear discriminant analysis, predict the final exam performance (pass/fail) of a current student who scored 55.7 marks in Assignment, 49.8 marks in Quiz and 52.6 marks in Test. (10 marks)

Solution: Let pass = 1 and fail = 0.

Prior probability (given in item (V) and using (4.5)):

$$\widehat{\pi}_1 = \frac{380}{500} = 0.76; \quad \widehat{\pi}_0 = 1 - 0.76 = 0.24$$

Mean vector (given in items (II), (III) and (IV) and calculate using (4.20)):

$$\widehat{\mu}_1 = [73.9,\ 68.2,\ 63.7]; \quad \widehat{\mu}_0 = [51.4,\ 42.3,\ 35.6]$$

Discriminant function, $\delta_j(X)$:

$$\widehat{\delta}_j(X) = \widehat{\mu}_j C^{-1} \boldsymbol{x}^T - \frac{1}{2}\widehat{\mu}_j C^{-1} \widehat{\mu}_j^T + \ln \pi_j,$$

$$\widehat{\delta}_1(X) = [1.6680,\ 2.1681,\ 2.5852]\boldsymbol{x}^T - 435.8066 + \ln(0.76)$$
$$\widehat{\delta}_0(X) = [1.0096,\ 1.3760,\ 1.5859]\boldsymbol{x}^T - 166.5589 + \ln(0.24)$$

For a new observation, $\boldsymbol{x}^* = [55.7,\ 49.8,\ 52.6]$,

$$\widehat{\delta}_1(\boldsymbol{x}^*) = 118.6826$$
$$\widehat{\delta}_0(\boldsymbol{x}^*) = 123.4746$$

Since $\delta_1(\boldsymbol{x}^*) < \delta_0(\boldsymbol{x}^*)$, the new observation should be assigned to class 0, the student will "more likely to" fail in final exam.

In the one-dimensional case, the estimation formulae for the LDA parameters $\{\pi_1, \cdots, \pi_K\}$, $\{\mu_1, \cdots, \mu_K\}$ and σ^2 in Example 4.3.1 are

$$\hat{\pi}_j = \frac{n_j}{n}, \quad \hat{\mu}_j = \frac{1}{n_j} \sum_{i:y_i=j} x_i, \quad \hat{\sigma}^2 = \frac{1}{n-K} \sum_{j=1}^{K} \sum_{i:y_i=j} (x_i - \hat{\mu}_j)^2 \tag{4.24}$$

where n is the total number of training observations, and n_j is the number of training observations in the jth class.

Example 4.3.2 (May 2022 Semester Final Exam, Q2)**.** The data in Table 2.1 contains size measurements for two Raisin classes, i.e. Besni and Kecimen.

Area	Perimeter	Class
78883	1092.709	Kecimen
49336	909.681	Kecimen
53890	957.132	Kecimen
143386	1422.014	Besni
158808	1624.343	Besni
134303	1497.515	Besni
116198	1328.070	Besni

Table 2.1: Seven data from the UCI Raisin dataset.

Construct a linear discriminant analysis (LDA) classifier for the given data in Table 2.1 by following the following steps.

(a) Write down the general mathematical formula of the LDA classifier in terms of the discriminant functions. (2 marks)

Solution: The general mathematical formula for the LDA classifier in terms of the discriminant functions is

$$h_D(\boldsymbol{x}) = \underset{j\in\{1,\ldots,K\}}{\operatorname{argmax}} \left[\ln P(Y=j) + \boldsymbol{\mu}_j^T \boldsymbol{C}^{-1}(\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_j) \right] \qquad [2 \text{ marks}]$$

which is based on the posterior probability is proportional to Gaussian distribution:

$$P(Y|X=x) \propto \frac{1}{(2\pi)^{p/2}\sqrt{|\boldsymbol{C}|}} \exp(-\frac{1}{2}(\boldsymbol{x}-\mu_k)^T \boldsymbol{C}^{-1}(\boldsymbol{x}-\mu_k)) \cdot \hat{P}(Y)$$

where $\widehat{P}(Y)$ is the prior distribution.

(b) Write down the estimates of all the parameters of discriminant functions of the LDA model in part (a) and write down the mathematical formulas of the discriminant functions. Note that $\vec{\mu}_{Kecimen}^T C^{-1} \approx \begin{bmatrix} -0.001665 \\ 0.304976 \end{bmatrix}^T$. (13 marks)

Solution: The prior probability estimates for both classes are

$$\widehat{P}(Y=\text{Kecimen}) = \frac{3}{7} = 0.4285714, \quad \widehat{P}(Y=\text{Besni}) = \frac{4}{7} = 0.5714286. \quad [2 \text{ marks}]$$

The mean vectors for both classes are

$$\begin{aligned}
\vec{\mu}_{\text{Kecimen}} &= \frac{1}{3}((78883, 1092.709) + (49336, 909.681) + (53890, 957.132)) \\
&= (60703, 986.5073) \\
\vec{\mu}_{\text{Besni}} &= \frac{1}{4}((143386, 1422.014) + (158808, 1624.343) + \\
&\quad (134303, 1497.515) + (116198, 1328.070)) \\
&= (138173.8, 1467.9855)
\end{aligned} \qquad [4 \text{ marks}]$$

We now estimate the "unscaled" covariance matrix estimate for Kecimen:

$$X_{\text{Kecimen}} - \vec{\mu}_{\text{Kecimen}} = \begin{bmatrix} 78883 & 1092.709 \\ 49336 & 909.681 \\ 53890 & 957.132 \end{bmatrix} - [60703, 986.5073]$$

$$= \begin{bmatrix} 18180 & 106.20167 \\ -11367 & -76.82633 \\ -6813 & -29.37533 \end{bmatrix} \qquad \text{[1.5 marks]}$$

$$C_{\text{Kecimen}} = (X_{\text{Kecimen}} - \vec{\mu}_{\text{Kecimen}})^T (X_{\text{Kecimen}} - \vec{\mu}_{\text{Kecimen}})$$

$$= \begin{bmatrix} 506138058 & 3004165.38 \\ 3004165 & 18043.99 \end{bmatrix}$$

We now estimate the "unscaled" covariance matrix estimate for Besni:

$$X_{\text{Besni}} - \vec{\mu}_{\text{Besni}} = \begin{bmatrix} 143386 & 1422.014 \\ 158808 & 1624.343 \\ 134303 & 1497.515 \\ 116198 & 1328.070 \end{bmatrix} - [138173.8, 1467.985]$$

$$= \begin{bmatrix} 5212.25 & -45.9715 \\ 20634.25 & 156.3575 \\ -3870.75 & 29.5295 \\ -21975.75 & -139.9155 \end{bmatrix} \qquad \text{[1.5 marks]}$$

$$C_{\text{Besni}} = (X_{\text{Besni}} - \vec{\mu}_{\text{Besni}})^T (X_{\text{Besni}} - \vec{\mu}_{\text{Besni}})$$

$$= \begin{bmatrix} 950856117 & 5947151.53 \\ 5947152 & 47009.39 \end{bmatrix}$$

The group covariance matrix estimate is

$$\boldsymbol{C} = \frac{1}{7-2}(C_{\text{Kecimen}} + C_{\text{Besni}}) = \begin{bmatrix} 291398835 & 1790263.38 \\ 1790263 & 13010.67 \end{bmatrix} \qquad \text{[1 mark]}$$

Note that using scientific calculator or linear algebra, we can obtain

$$\vec{\mu}_{\text{Kecimen}}^T \boldsymbol{C}^{-1} = \begin{bmatrix} -0.001665359 \\ 0.304975610 \end{bmatrix}, \quad \vec{\mu}_{\text{Besni}}^T \boldsymbol{C}^{-1} = \begin{bmatrix} -0.001416364 \\ 0.307720382 \end{bmatrix}. \qquad \text{[1 mark]}$$

The discriminant functions (of the LDA model) for both classes are

$$\delta_{\text{Kecimen}}(\boldsymbol{x}) = \ln\frac{3}{7} + \begin{bmatrix} -0.001665 \\ 0.304976 \end{bmatrix}^T (\boldsymbol{x} - \frac{1}{2}\begin{bmatrix} 60703 \\ 986.5073 \end{bmatrix}),$$

$$\delta_{\text{Besni}}(\boldsymbol{x}) = \ln\frac{4}{7} + \begin{bmatrix} -0.001416 \\ 0.307720 \end{bmatrix}^T (\boldsymbol{x} - \frac{1}{2}\begin{bmatrix} 138173.8 \\ 1467.9855 \end{bmatrix}) \qquad \text{[2 marks]}$$

(c) Based on the calculations in part (b), comment on the need for data preprocessing with justification. (2 marks)

(d) Suppose the number of samples n is less than or equal to the number of features p, can the linear discriminant analysis model still be applied? Write down your answer by providing justification based on the mathematical formulation from linear algebra. (Hint: rank of matrix) (3 marks)

Solution: When $n \leq p$, the matrix X_k corresponding to class k is an $n_k \times p$-matrix and the matrix $X_k - \vec{\mu}_k$ is also an $n_k \times p$-matrix of rank $n_k < p$ since $n_k < n$. [1 mark]

When the unscaled covariance matrices for class k is computed:

$$C_k = (X_k - \vec{\mu}_k)^T (X_k - \vec{\mu}_k)$$

it is found to be $p \times p$ but the matrix has a rank lower than p. The group covariance matrix will also have a rank lower than p leading to the impossibility of inverting the group covariance matrix C which is required by the LDA's discriminant functions. [2 marks]

(e) The CART tree trained on a subset of the UCI Raisin dataset is shown in Figure 2.1.

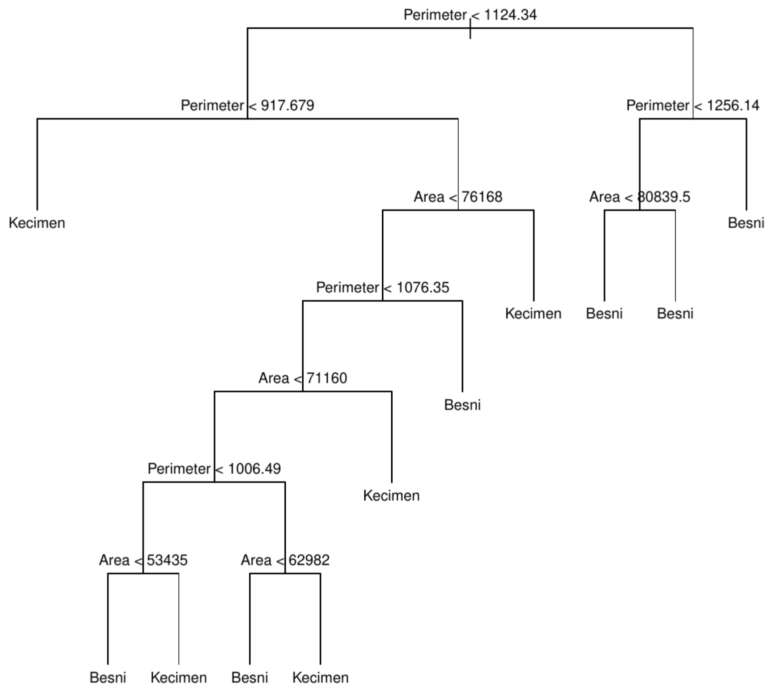


Figure 2.1: Fitted CART tree.

(i) Predict the class for a raisin with an area 87524, a major axis length 442.2460, a minor axis length 253.2912, an eccentricity 0.8197384, a convex-area 90546, an extent 0.7586506 and a perimeter 1184.040. (3 marks)

Solution: Based on the CART tree, only the Area and Perimeter are important.

- The perimeter is $1184.040 \geq 1124.34$, go right.
- The perimeter is $1184.040 < 1256.14$, go left.
- The area $87524 \geq 80839.5$, go right .... [1.5 marks]

Prediction of the class: **Besni** .... [1.5 marks]

(ii) Predict the class for a raisin with an area 49242, and a perimeter 881.836. (2 marks)

Example 4.3.3. Table below shows the data collected:

Customer	Balance	Default
1	500	N
2	1980	Y
3	60	N
4	2810	Y
5	1400	N
6	300	N
7	2000	Y
8	940	N
9	1630	Y
10	2170	Y

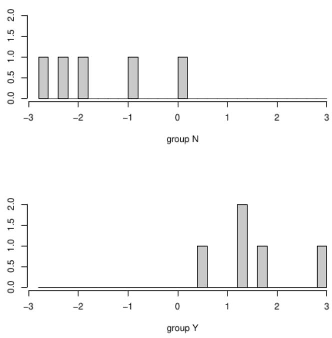
Use these data and the predictive model LDA to predict if a customer with balance 1500 will default in his credit card?

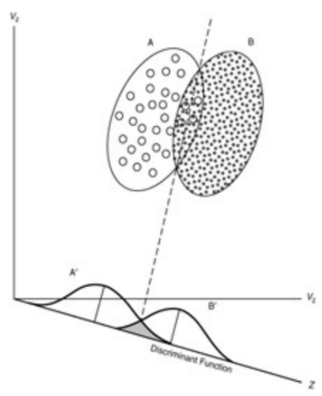
So far, we have been applying the LDA using hand-calculation. LDA and QDA are available in R's `MASS` package as `lda` and `qda` respectively [Venables and Ripley, 2002]. They are available in Python's `sklearn.discriminant_analysis` as `LinearDiscriminantAnalysis` and `QuadraticDiscriminantAnalysis`.

According to Tim [2018], The `MASS` package's `lda` function produces coefficients in a different way from (4.20), (4.21) and to most other LDA software. The alternative approach computes one set of coefficients for each group and each set of coefficients has an intercept. With the discriminant function (scores) computed using these coefficients, classification is based on the highest score and there is no need to compute posterior probabilities in order to predict the classification. A modification of the MASS function that produces these more convenient coefficients is available as `Displayr/flipMultivariates`. If an object is created using LDA, we can extract the coefficients using `obj$original$discriminant.functions`.

The theory behind this function is "Fisher's Method for Discriminating among Several Population" [Johnson and Wichern, 2007, §11.6].

Example 4.3.4. Redo Example 4.3.3 using R. Are you able to find the discriminant functions?





Example 4.3.5 (Final Exam Jan 2023, Q2. Part (a) is in Tut3)**.** (b) Suppose the parameters of the quadratic discriminant analysis (QDA) predictive model

$$P(Y=k|\boldsymbol{X}=\boldsymbol{x}) \propto \left[\frac{1}{(2\pi)^{p/2}\sqrt{\det(C_k)}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)C_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\right)\right] P(Y=k)$$

trained the data D are given in Table 2.1.

k	DOWN	UP
$\#\{y_i = k\}$	18228	13489
$\text{mean}(X_1 \mid Y = k)$	4.0659	3.8932
$\text{mean}(X_2 \mid Y = k)$	0.4634	0.5515
$\text{mean}(X_3 \mid Y = k)$	0.5166	0.4775
$\det(C_k)$	0.00923456	0.00560849
C_k	$\begin{bmatrix} 3.8891 & 0.0316 & -0.0226 \\ 0.0316 & 0.0989 & -0.0083 \\ -0.0226 & -0.0083 & 0.0249 \end{bmatrix}$	$\begin{bmatrix} 4.1010 & -0.0204 & -0.0189 \\ -0.0204 & 0.0656 & 0.0004 \\ -0.0189 & 0.0004 & 0.0210 \end{bmatrix}$
C_k^{-1}	$\begin{bmatrix} 0.2589 & -0.0648 & 0.2139 \\ -0.0648 & 10.4136 & 3.4063 \\ 0.2139 & 3.4063 & 41.5542 \end{bmatrix}$	$\begin{bmatrix} 0.2452 & 0.0750 & 0.2197 \\ 0.0750 & 15.2625 & -0.2162 \\ 0.2197 & -0.2162 & 47.9087 \end{bmatrix}$

Table 2.1: Parameters of the trained QDA model.

Determine the value of p in the trained QDA model and then calculate the posterior probabilities for DOWN and UP when the day is 2, the period is 0.042553 and the transfer is 0.414912 based on the trained QDA model. (4 marks)

Solution: Since there are three attributes, $p = 3$. ....[1 mark]

$$P(Y = \text{DOWN} \mid X_1 = 2, X_2 = 0.042553, X_3 = 0.414912) \propto \frac{18228}{18228 + 13489} \times$$
$$\frac{1}{(2\pi)^{3/2} \times \sqrt{0.00923456}} \exp\left(-\frac{1}{2} \begin{bmatrix} -2.0659 \\ -0.4208 \\ -0.1017 \end{bmatrix}^T C_{\text{DOWN}}^{-1} \begin{bmatrix} -2.0659 \\ -0.4208 \\ -0.1017 \end{bmatrix} \right)$$
$$= 0.5747076 \times \frac{1}{1.513484} \exp(-1.823913) = 0.06128495$$

[1 mark]

$$P(Y = \text{UP} \mid X_1 = 2, X_2 = 0.042553, X_3 = 0.414912) \propto \frac{13489}{18228 + 13489} \times$$
$$\frac{1}{(2\pi)^{3/2} \times \sqrt{0.005609738}} \exp\left(-\frac{1}{2} \begin{bmatrix} -1.8932 \\ -0.5089 \\ -0.0626 \end{bmatrix}^T C_{\text{UP}}^{-1} \begin{bmatrix} -1.8932 \\ -0.5089 \\ -0.0626 \end{bmatrix} \right)$$

[1 mark]

$$= 0.4252924 \times \frac{1}{1.179617} \exp(-2.601496) = 0.02673813$$

The posterior probabilities for DOWN and UP are $\dfrac{(0.061285, 0.026738)}{0.061285 + 0.026738} = (0.6962,$ $0.3038)$ [1 mark]

Average: 0.97 / 4 marks in Jan 2023; 71% below 2 marks.

(c) By using the coefficients from Table 2.1, use the notion of linear algebra to prove that the inverse of the covariance matrix C in the linear discriminant analysis (LDA) predictive model is as follows:

$$C^{-1} = \begin{bmatrix} 0.2525 & -0.0160 & 0.2258 \\ -0.0160 & 11.9250 & 2.3438 \\ 0.2258 & 2.3438 & 43.7657 \end{bmatrix}.$$

Use the trained LDA model with C^{-1}, $\det(C) = 0.007705512$ and the parameters from Table 2.1 to calculate the posterior probalities for DOWN and UP when the day is 2, the period is 0.042553 and the transfer is 0.414912. [Hint: You may reuse some of results from part (b).] (7 marks)

Solution: To prove the values of C^{-1}, we need to calculate

$$C = ((18228-1)C_{\text{DOWN}} + (13489-1)C_{\text{UP}})/(18228+13489-2)$$
$$\approx \begin{bmatrix} 3.9792 & 0.0095 & -0.0210 \\ 0.0095 & 0.0847 & -0.0046 \\ -0.0210 & -0.0046 & 0.0232 \end{bmatrix} \qquad \text{[1 mark]}$$

and the show that the following product is close to identity:

$$C \times C^{-1} \approx \begin{bmatrix} 1 & 0.0004 & 0.0015 \\ 0 & 0.9991 & -0.0007 \\ 0 & -0.0001 & 0.9998 \end{bmatrix} \approx I_3. \qquad \text{[1 mark]}$$

$$P(Y=\text{DOWN}|X_1=2, X_2=0.042553, X_3=0.414912) \propto \frac{18228}{18228+13489} \times$$
$$\frac{1}{(2\pi)^{3/2} \times \sqrt{0.007705512}} \exp\left(-\frac{1}{2} \begin{bmatrix} -2.0659 \\ -0.4208 \\ -0.1017 \end{bmatrix}^T C^{-1} \begin{bmatrix} -2.0659 \\ -0.4208 \\ -0.1017 \end{bmatrix} \right)$$
$$= 0.5747076 \times \frac{1}{179.4193} \exp(-1.95504) = 0.0588455$$

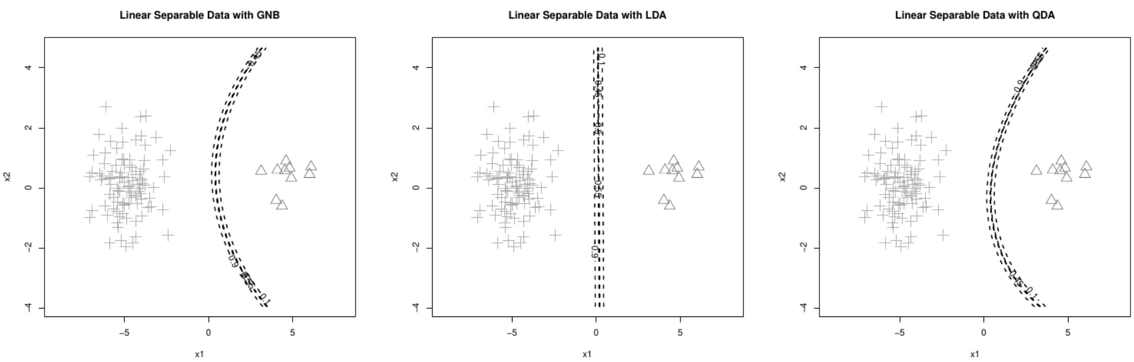
[2 marks]

$$P(Y=\text{UP}|X_1=2, X_2=0.042553, X_3=0.414912) \propto \frac{13489}{18228+13489} \times$$
$$\frac{1}{(2\pi)^{3/2} \times \sqrt{0.007705512}} \exp\left(-\frac{1}{2} \begin{bmatrix} -1.8932 \\ -0.5089 \\ -0.0626 \end{bmatrix}^T C^{-1} \begin{bmatrix} -1.8932 \\ -0.5089 \\ -0.0626 \end{bmatrix} \right) \qquad \text{[2 marks]}$$
$$= 0.4252924 \times \frac{1}{179.4193} \exp(-2.168739) = 0.0351678$$

The posterior probabilities for DOWN and UP are $\dfrac{(0.0588455, 0.0351678)}{0.0588455 + 0.0351678} = (0.6259, 0.3741)$ [1 mark]

Average: 1.10 / 7 marks in Jan 2023; 86% below 3.5 marks.

Example 4.3.6 (Completely Separable 2D Data). For the separable data in Example 3.1.6, unlike LR, LDA has no issue fitting separable data. The decision boundaries of the classifiers Gaussian Naive Bayes (GNB), LDA and QDA are shown in the figure below.

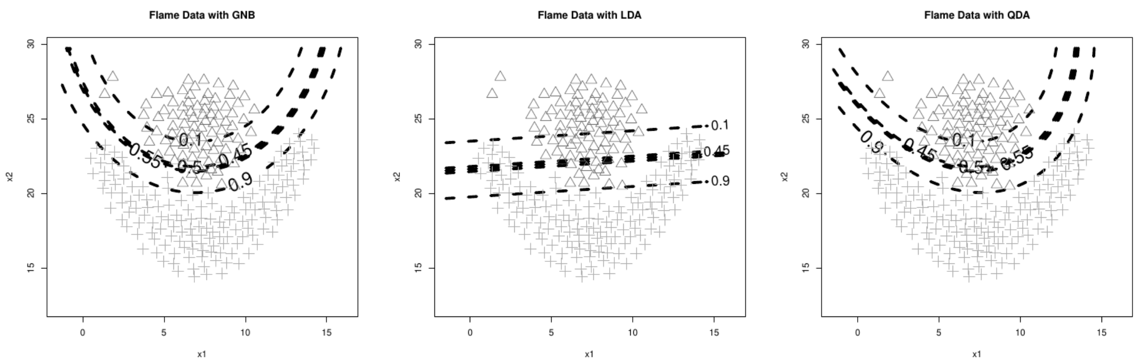


□

Example 4.3.7 (Decision Boundaries for (Milely Nonlinear) Flame Data)**.** For the "flame" data in Example 3.1.14, the script to estimate the decision boundary is shown below.

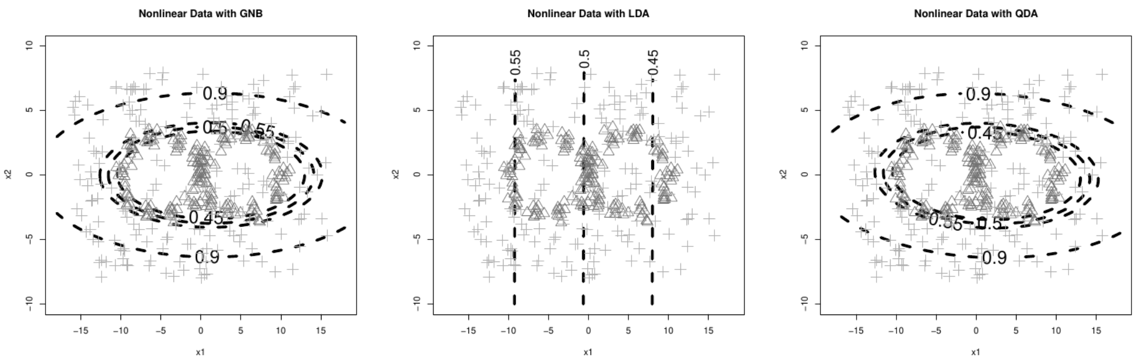
```
d.f = read.table("flame.txt", header=FALSE)
names(d.f) = c("x1", "x2", "y")
d.f$y = factor(d.f$y)

x1min = min(d.f$x1)
x1max = max(d.f$x1)
x2min = min(d.f$x2)
x2max = max(d.f$x2)

### Contour plot
g.x1 = seq(x1min-2, x1max+2, by=0.1)
g.x2 = seq(x2min-2, x2max+2, by=0.1)
d.grid = expand.grid(x1=g.x1, x2=g.x2)

library(naivebayes)
model = naive_bayes(y ~ ., d.f)

prob = predict(model, newdata=d.grid, type="prob")[,2]
prob = matrix(prob, length(g.x1), length(g.x2))
### https://stackoverflow.com/questions/24109956/how-to-increase-font-size-of-contour-labels
contour(g.x1, g.x2, prob, levels=c(0.1,0.45,0.5,0.55,0.9), col="black",
  labcex=2, xlab="x1",ylab="x2",main="Flame Data with GNB",
  lty=2,lw=5, xlim=range(g.x1), ylim=range(g.x2))
points(d.f$x1, d.f$x2, col=1+as.integer(d.f$y), pch=1+as.integer(d.f$y),
  cex=2, xlab="x1",ylab="x2")

library(MASS)
model = lda(y ~ ., d.f)

# Pick the posterior probability for Y=1 at second column
prob = predict(model, newdata=d.grid)$posterior[,2]
prob = matrix(prob, length(g.x1), length(g.x2))
contour(g.x1, g.x2, prob, levels=c(0.1,0.45,0.5,0.55,0.9), col="black",
  labcex=1.5, xlab="x1",ylab="x2",main="Flame Data with LDA",
  method="edge", lty=2,lw=5, xlim=range(g.x1), ylim=range(g.x2))
points(d.f$x1, d.f$x2, col=1+as.integer(d.f$y), pch=1+as.integer(d.f$y),
  cex=2, xlab="x1",ylab="x2")

model = qda(y ~ ., d.f)

prob = predict(model, newdata=d.grid)$posterior[,2]
prob = matrix(prob, length(g.x1), length(g.x2))
contour(g.x1, g.x2, prob, levels=c(0.1,0.45,0.5,0.55,0.9), col="black",
  labcex=2, xlab="x1",ylab="x2",main="Flame Data with QDA",
  lty=2,lw=5, xlim=range(g.x1), ylim=range(g.x2))
points(d.f$x1, d.f$x2, col=1+as.integer(d.f$y), pch=1+as.integer(d.f$y),
  cex=2, xlab="x1",ylab="x2")
```

The decision boundaries of the classifiers Gaussian Naive Bayes (GNB), LDA and QDA are shown in the figure below.



□

Example 4.3.8 (Decision Boundaries for Strongly Nonlinear Data)**.** For the strongly nonlinear data in Example 3.1.15, the decision boundaries of the classifiers Gaussian Naive Bayes (GNB), LDA and QDA are shown in the figure below.



□

According to James et al. [2013, §4.4], the answer to choosing LDA or QDA lies in the bias-variance trade-off. When there are p predictors, then estimating a covariance matrix requires estimating $p(p+1)/2$ parameters. QDA needs to estimate a separate covariance matrix for each class, for a total of $Kp(p+1)/2$ parameters which is **large** when p is large.

Applications

It is vital for a company to closely monitor the public reception of key events, such as product launches or press releases. With its real-time access and easy accessibility of user-generated content on Facebook and Twitter, it is now possible to do "sentiment classification" or "opinion mining" [Coelho and Richert, 2015, Chapter 6]. One tool that is useful in sentiment analysis is the LDA method.

LDA is also used in special image analysis. According to `https://en.wikipedia.org/wiki/Linear_discriminant_analysis`, Otsu's method, named after Nobuyuki Otsu, was created to turn the histogram of pixels in a grayscale image to binary values by optimally picking the black/white threshold.

The algorithm assumes that the image contains two classes of pixels following bi-modal histogram (foreground pixels and background pixels), it then calculates the optimum threshold separating the two classes so that their combined spread (intra-class variance) is minimal, or equivalently, so that their inter-class variance is maximal. Consequently, Otsu's method is roughly a one-dimensional, discrete analogue of Fisher's Discriminant Analysis.

The terms Fisher's linear discriminant [Johnson and Wichern, 2007]3 and LDA are often used interchangeably, although Fisher's original article actually describes a slightly different discriminant, which does not make some of the assumptions of LDA such as normally distributed classes or equal class covariances.

4.4 General Generative Models and Neural Network Architectures

According to Wikipedia, general classes of generative models include:

- General mixture model (e.g. GMM used in clustering)
- Hidden Markov model

- Probabilistic context-free grammar and Latent Dirichlet Allocation
- Bayesian Network (a generalisation of Naive Bayes)
- (Restricted) Boltzmann Machine
- Autoregressive model (https://www.georgeho.org/deep-autoregressive-models/)
- https://en.wikipedia.org/wiki/Energy-based_model
- https://en.wikipedia.org/wiki/Generative_adversarial_network (GAN): Two ANNs contest with each other in the form of a zero-sum game, where one agent's gain is another agent's loss.
- https://en.wikipedia.org/wiki/Variational_autoencoder
- https://en.wikipedia.org/wiki/Flow-based_generative_model
  - NICE: Nonlinear Independent Components Estimation
  - Real Non-Volumne Preserving (Real NVP)
  - Generative Flow (Glow) Network
  - Masked Autoregressive Flow (MAF)
  - Continuous Normalising Flow (CNF)
- https://en.wikipedia.org/wiki/Diffusion_model : Constructing a sequence of denoising decoders to generate data from noise.
  - Denoising Diffusion Probabilistic Model (DDPM)
  - Score-Based Generative Model (SGM)
  - Score Stochastic Differential Equations
  - Diffusion Normalising Flow : "Diffusion + Flow"
  - DiffFlow : "Diffusion Model + GAN"

4.4.1 Encoder-Decoder Architecture

Encoder-Decoder Architecture is a kind of unsupervised learning which is to "compress images", to obtain "super-resolution of images" or to "highlight regions of images" (e.g. identify tumour), to encode machine translation of a source language (e.g. Malay language) to a target language (e.g. Chinese language).

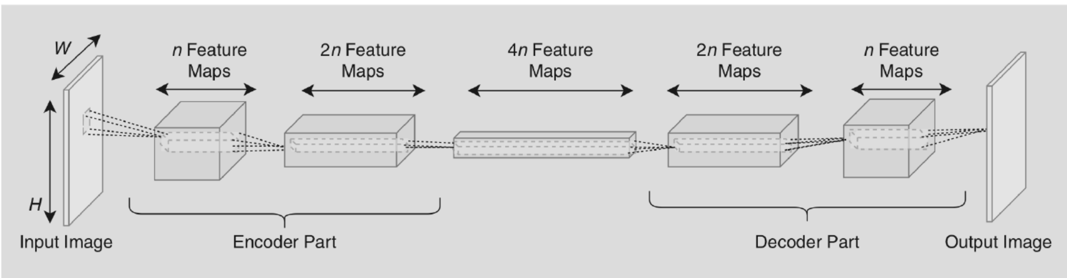
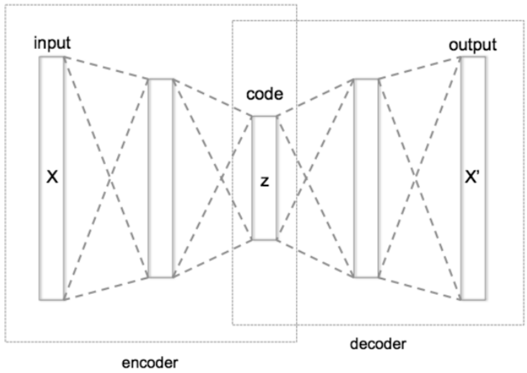


FIGURE 5. In an encoder-decoder CNN, the feature maps are spatially compressed by an encoder network, then increased back to the size of the output image by a decoder network.

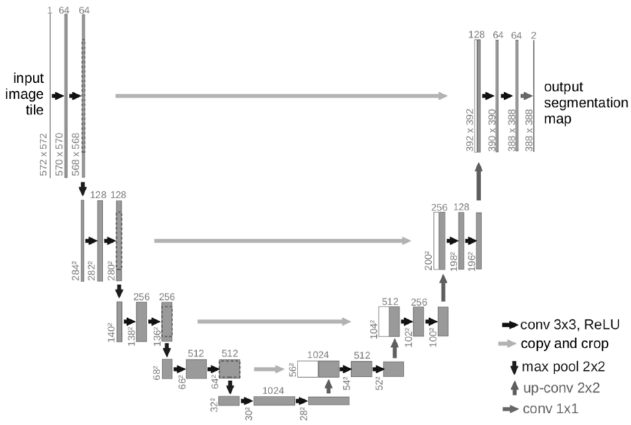
Source: https://www.khoury.northeastern.edu/home/hand/teaching/cs6140-fall-2021/Neural_Network_Architectures_for_Images.pdf

Autoencoder

`https://en.wikipedia.org/wiki/Autoencoder` is an unsupervised learning model such that $h(\boldsymbol{x}) \approx \boldsymbol{x}$. It is used for dimensionality reduction and for learning generative models of data.



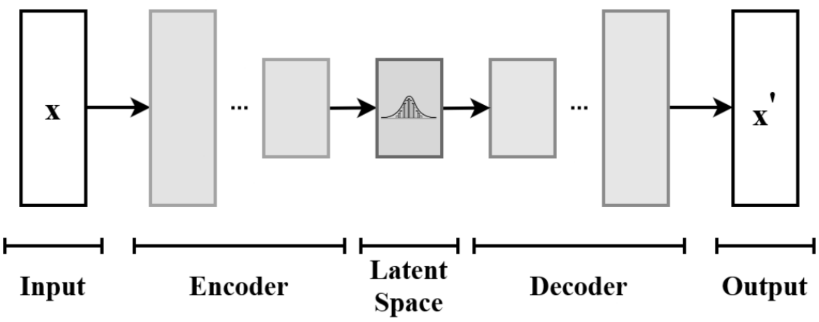
One famous autoencoder is the `https://en.wikipedia.org/wiki/U-Net` which is used in biomedical image segmentation.



Source: `https://www.khoury.northeastern.edu/home/hand/teaching/cs6140-fall-2021/Neural_Network_Architectures_for_Images.pdf`

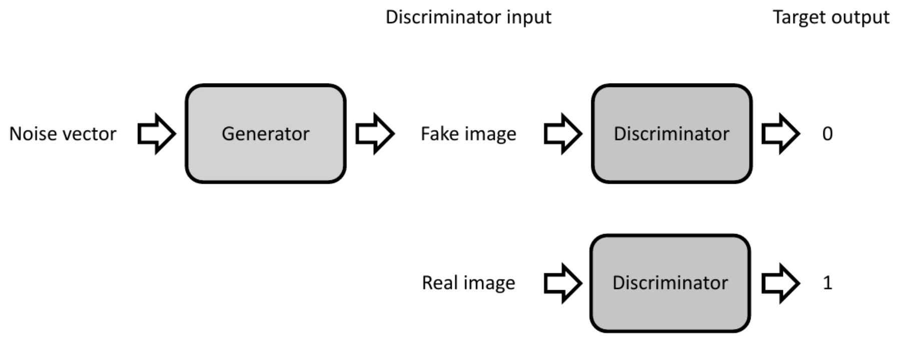
Variational Autoencoder (VAE)

`https://en.wikipedia.org/wiki/Variational_autoencoder` is an artificial neural network architecture introduced by Diederik P. Kingma and Max Welling (2013). It is part of the families of probabilistic graphical models and variational Bayesian methods.



4.4.2 Generative Adversarial Network (GAN)

https://en.wikipedia.org/wiki/Generative_adversarial_network is a class of machine learning frameworks containing two distinct neural networks working in tandem to produce an output from the input. The concept was initially developed by Ian Goodfellow and his colleagues in June 2014. In a GAN, the two neural networks contest with each other in the form of a zero-sum game, where one agent's gain is another agent's loss.



Within GANs are subtypes that have unique architectures, such as:

- Vanilla GAN: This is the basic version of a GAN that needs to be adapted for many specific real-world applications.
- CycleGAN: The cycle-consistent generative adversarial network, or https://junyanz.github.io/CycleGAN/, is useful for image-to-image translation, moving an image from one domain to another.
- DCGAN: Deep convolutional generative adversarial network, or DCGAN (https://arxiv.org/abs/1511.06434, leverages CNNs for more powerful image generation.
- Text-2-image: A text-2-image GAN can create novel images from text-based descriptions, such as adding specific eye colour to a generated face.

4.4.3 Applications

The AI based on generative models is usually called "generative AI".

Examples of Generative AI include:

- DeepSeek3 and DeepSeekR1: open source large language model (LLM) and reasoning model;
- ChaptGPT (and GPT-4o): closed source large language models (and reasoning model) created by OpenAI, backed by Microsoft;
- DALL-E 3 (https://openai.com/dall-e-3): image generation tool created by OpenAI;
- Google Gemini (former Bard): large language model created by Google.
- Stable Diffusion 3 (https://stability.ai/stable-image): text to image generation tool created by stability.ai (they also offer video, audio, 3D and language generation models);

Generative models for real-world **applications** include

- Health care and pharmaceuticals:
  - Enhancing medical images: Generative AI can augment medical images like X-rays or MRIs, synthesise images, reconstruct images, or create reports about images. This technology can even generate new images to demonstrate how a disease may progress in time.

  - Discovering new medicines: Researchers can use generative AI called generative design to develop new medicines.
  - Simplify tasks with patient notes and information: Healthcare professionals keep and take notes about patient medical care. Generational AI can build patient information summaries, create transcripts of verbally recorded notes, or find essential details in medical records more effectively than human efforts.
  - Personalised treatment: Generative AI can consider a large amount of patient information, including medical images and genetic testing, to deliver a customised treatment plan tailored to the patient's needs.

- Media and entertainment (relevant to Content Creators / YouTubers): Generative models can be used in generating images, summary of reports and subtitles (speech to text) for online social media.
  - Create audio and visual content: Generative AI can create new video content from scratch, create visual content faster by creating visual effects, adding graphics, or streamlining editing.
  - Manage tags for better content management: Generative AI can tag and index extensive media libraries, making locating the files you need at any time easier. Similar to our manufacturing example above, generative AI allows using conversational language to find the information or media you're looking for in a complex media library.

- Advertising and marketing:
  - Generate marketing text and images: Generative AI can help marketing professionals create consistent, on-brand text and images to use in marketing campaigns in many languages.
  - Generate personalised recommendations: Generative AI helps create powerful recommendation engines to help customers discover new products they might like.
  - Create product descriptions: Beyond flashy advertising campaigns, generative AI can help with tedious or time-consuming content requirements like creating product descriptions.
  - Enhance **search engine optimisation (SEO)**: SEO professionals can use generative AI for tasks like image tags or page titles or to create content drafts. Tools like ChatGPT or Bard can be used to provide recommendations that could make content that improve SEO ranking.

- Manufacturing
  - Accelerating the design process: Using generative AI, engineers and project managers can work through the design process much faster by generating design ideas and asking the AI to assess ideas based on the constraints of the project.
  - Provide smart maintenance solutions for equipment: Maintenance professionals can use generative AI to track the performance of heavy equipment based on historical data, potentially alerting them to trouble before the machine malfunctions. Generative AI can also recommend routine maintenance schedules.
  - Improve supply chain: Generative AI can help tracking down the cause of problems in the supply chain by speaking conversationally with the technology to sort through a vast amount of transactional or product data. Generative AI can also help generate delivery schedules or recommendations for suppliers.

- Software development

  - Generating code: Software developers can create, optimise, and auto-complete code with generative AI. Generative AI can create code blocks by comparing them to a library of similar information.
  - Translate programming languages: Generative AI can be a tool for developers to interact with software without needing a programming language. The generative AI would act as a translator.
  - Automate testing: Developers can improve their automated testing processes using generative AI to highlight potential problems and execute testing sequences faster than other AI methods. Generative AI can learn the logic of the software and how users will interact with it and create test cases to demonstrate various user scenarios.

- Financial services
  - Create investment strategies: Generative AI can recommend the best investments according to client's goals. This technology can find and execute trades much faster than human investors and can do so within the parameters you set for the kind of transaction you want.
  - Communicate and educate clients and investors: Financial services professionals sometimes need to communicate complex information to clients and colleagues. Generational AI can provide personalised customer service without adding more customer service professionals.
  - Quickly draft documentation and monitor regulation: Generative AI can monitor regulatory activity, keep you informed of any changes, and create drafts of documents such as investment research or insurance policies.