Section 6: \( k \)-Nearest Neighbours.

Ensure you have completed all previous worksheets before advancing to this one.

1 Classification

To perform a \( k \)-nearest neighbour classification in R we will make use of the `knn` function in the package `class`. Load this package into R and have a look at the function’s help file.

**Task:** Load package `MASS` and use the function `mvrnorm` to generate a data set called `simA` that consists of 900 simulations from a Multivariate Normal distribution with mean vector \( \mu_A = (0,0) \) and covariance matrix:

\[
\Sigma_A = \begin{pmatrix} 10 & 3 \\ 3 & 2 \end{pmatrix}
\]

Note you should see the previous worksheet for details of how to do this.

> 

> 

> 

**Task:** Generate a data set called `simB` that consists of 900 simulations from a Multivariate Normal distribution with mean vector \( \mu_B = (5,3) \) and covariance matrix:

\[
\Sigma_B = \begin{pmatrix} 12 & 2 \\ 2 & 15 \end{pmatrix}
\]

> 

> 

> 

We will also make use of the total data set, and to do this we will create a set `simT` that consists of `simA` and `simB`:

\[
\text{simT=rbind(simA,simB)}
\]
The \texttt{rbind} function simply combines its arguments row-wise, \textit{i.e.}, \texttt{simT} is now a matrix with 1,800 rows and 2 columns.

Next we will add an additional column to \texttt{simT} so as to record the class membership of each observation, \textit{i.e.}, to record whether the row in \texttt{simT} came from original data set \texttt{simA}, or from data set \texttt{simB}.

\begin{verbatim}
> class=c(rep("A",900),rep("B",900))
> simT=cbind(simT,class)
\end{verbatim}

The first line creates a vector of length 1,800 which consists of 900 entries of the term "A" followed by 900 entries of the term "B". The \texttt{rep} function in this line creates a vector of length given by its second argument and which consists of repeated entries given by its first argument. The second line adds this vector \texttt{class} as a third column for \texttt{simT}.

The function \texttt{cbind} is similar to \texttt{rbind}, except that instead of combining data sets row-wise, it does so column-wise. Finally, note the 'mis-use' of the equals sign in the final line. This is because \texttt{R} is not reading it here as indicating equality, but rather as assigning the right hand side to be equal to whatever is on the left hand side. Instead of using an '=' sign we could have used the expression '<-', which may help make the command clearer.

In order to visualise the data set \texttt{simT} enter the following:

\begin{verbatim}
> plot(simT[,1],simT[,2],col=as.factor(simT[,3]))
\end{verbatim}

The above command plots the bi-variate data and colours the points according to class membership. As the class membership is indicated by a value of "A" or "B" in the third column of the set \texttt{simT}, we first need to convert these terms to numeric values before requesting \texttt{R} use them to indicate point colouring. This is what the command \texttt{as.factor} has done.

\section{Training and Test Data}

The function \texttt{knn} takes as its arguments a training data set and a testing data set, which we are now going to create. We will use a split whereby 2/3’s of the data will be used for training and 1/3 of the data will be used for testing (we won’t apply the validation step here as we won’t be seeking automatic model comparison). To create the test and training sub-sets enter the following:

\begin{verbatim}
> index=c(1:600,901:1500)
> train=simT[index,1:2]
> test=simT[-index,1:2]
\end{verbatim}
The first line creates a vector that consists of the ordered numbers 1, 2, \ldots, 599, 600, 901, 902, \ldots, 1500. This is used to select 2/3’s of the data in \texttt{simT} with an equal numbers of points coming from \texttt{simA} and from \texttt{simB}. The second line creates a matrix \texttt{train} that consists of the un-labelled training data, whilst the final line creates a matrix \texttt{test} that consists of the remainder of \texttt{simT} (again un-labelled). The use of \texttt{-index} in the final line is included to tell \texttt{R} that \texttt{test} is to compose of those row numbers in \texttt{simT} that are not included in the list \texttt{index}.

Now we have the training and the test data we are able to run the function \texttt{knn}, and we will begin with a value of \( k = 3 \):

\[
\texttt{result=knn(train, test, cl=simT[index,3], k=3)}
\]

Now we have the training and the test data we are able to run the function \texttt{knn}, and we will begin with a value of \( k = 3 \):

\[
\texttt{result=knn(train, test, cl=simT[index,3], k=3)}
\]

The argument \texttt{cl=simT[index,3]} tells \texttt{R} the true classification for the training data set \texttt{train} in order that suggestions for the classification of the test set \texttt{test} can be generated. The output from \texttt{result} are these suggested classifications.

We would like to determine the misclassification rate for a given value of \( k \). This can be found by entering the following:

\[
\texttt{(nrow(test)-sum(diag(table(result,simT[-index,3]))))/nrow(test)}
\]

The inner command \texttt{table(result,simT[-index,3])} creates a table comparing the classification given to the test data by \( k \)-nearest neighbour against the true classification of this data (enter it in separately to see). Applying the \texttt{diag} function to this table then selects the diagonal elements, \textit{i.e.}, the number of points where \( k \)-nearest neighbour classification agrees with the true classification, and the \texttt{sum} command simply adds these values up. The remainder of the code ensures that the output that is returned is the percentage misclassification.

To see how the \( k \)-nearest neighbour classification performs as a function of \( k \) we can write a \texttt{for} loop:

\[
\texttt{k=c(1:10)}
\]

\[
\texttt{p=rep(0,10)}
\]

\[
\texttt{summary=cbind(k,p)}
\]

\[
\texttt{colnames(summary)=c("k","Percent misclassified")}
\]

\[
\texttt{for(i in 1:10){}
+ result=knn(train, test, cl=simT[index,3], k=i)
+ summary[i,2]=(nrow(test)-sum(diag(table(result,simT[-index,3]))))/nrow(test)
+ }}
\]
> summary

**Task:** Plot the misclassification rate as a function of $k$:

> 

**Task:** Generate 8 new points (4 of type "A" and 4 of type "B") and use your optimal value of $k$ to classify them:

> 

> 

> 

> 

> 

**Exercise:** Attempt a $k$-nearest neighbour classification of the Iris data.