kNN Clustering - Classification

Karen Mazidi

This example shows how to do knn clustering for classification.

The iris database comes with R. It has 150 instances and 5 columns: - Sepal.Length - Sepal.Width - Petal.Length - Petal.Width - Species: setosa, versicolor or virginica

Load and look at the data

```
attach(iris)
str(iris)    # display the structure of the object
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 ...
```

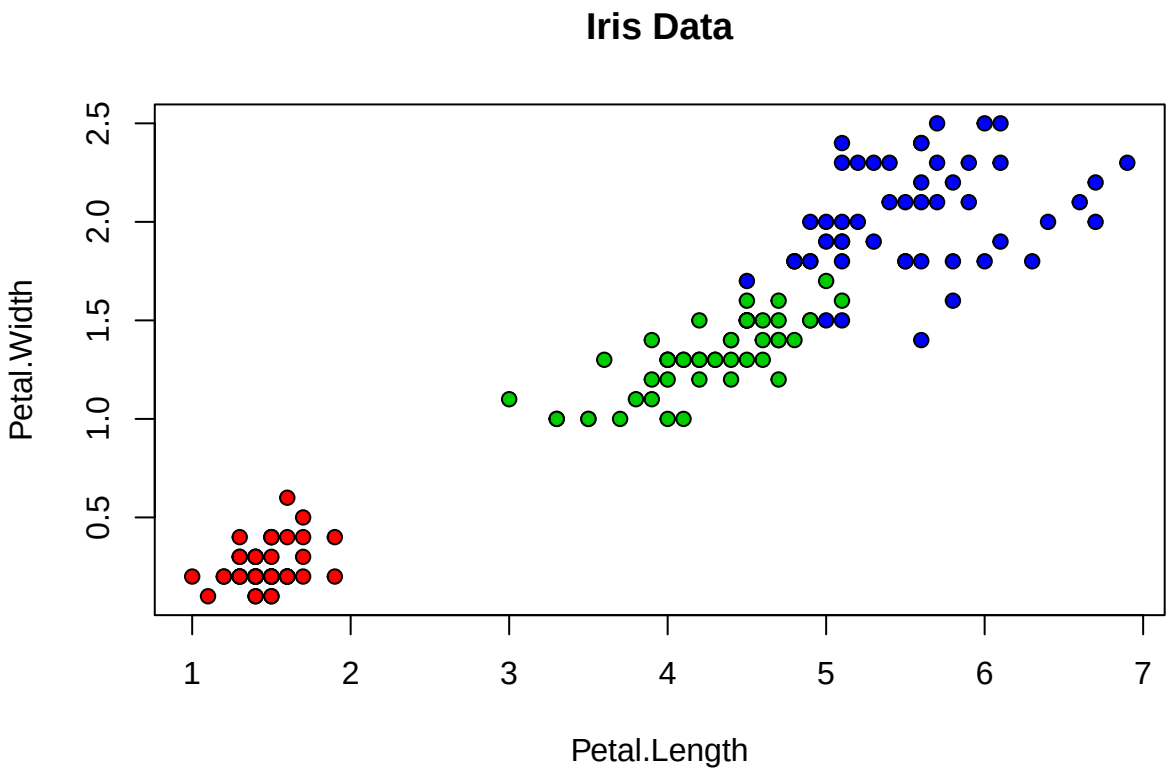
```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

Plot the data

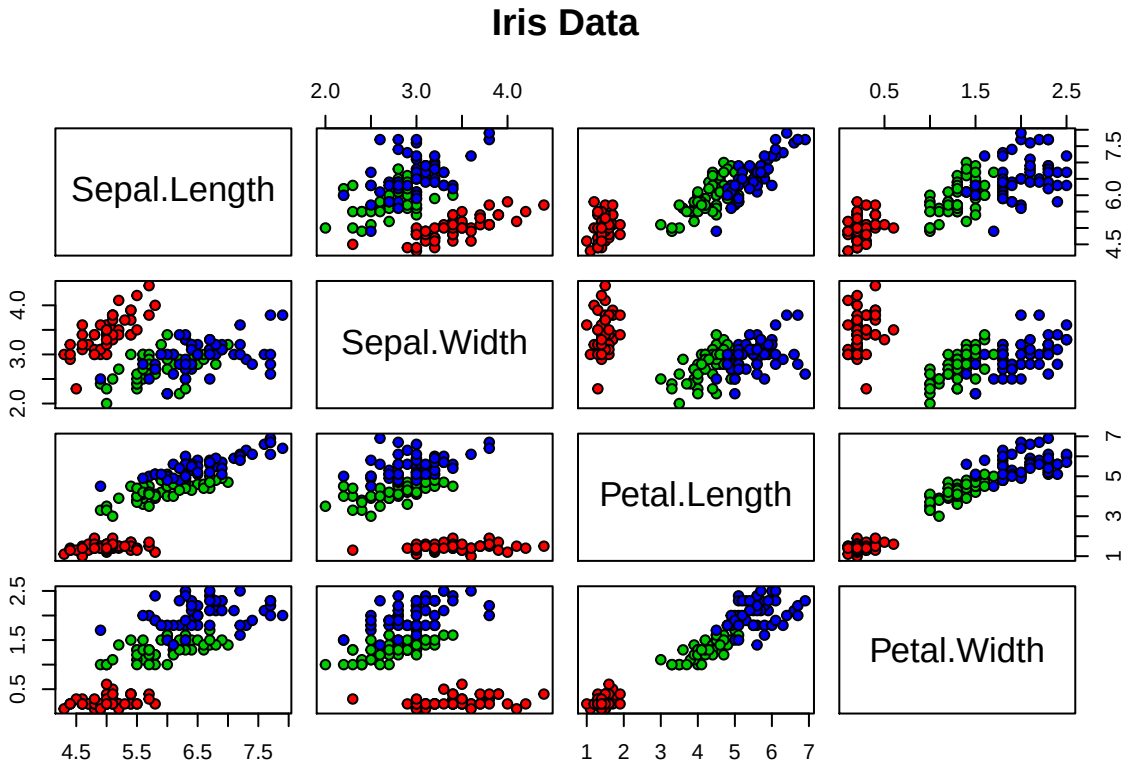
We let the 3 classes show as 3 different colors with the bg parameter and the "unclass" values 1, 2, 3 representing the 3 types of irises.

```
plot(Petal.Length, Petal.Width, pch=21, bg=c("red","green3","blue")
     [unclass(Species)], main="Iris Data")
```



Pairs scatter plots

```
pairs(iris[1:4], main = "Iris Data", pch = 21, bg = c("red", "green3", "blue")[unclass(Species)])
```



Divide into train/test sets

We will randomly sample the data set to let 2/3 be training and 1/3 test,

```
set.seed(1958)  # setting a seed gets the same results every time
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.67, 0.33))
iris.train <- iris[ind==1, 1:4]
iris.test <- iris[ind==2, 1:4]
iris.trainLabels <- iris[ind==1, 5]
iris.testLabels <- iris[ind==2, 5]
```

Classify

The knn() function uses Euclidean distance to find the k nearest neighbors.

Classificiation is decided by majority vote with ties broken at random.

Using an odd k can avoid some ties.

```
library(class)
iris_pred <- knn(train=iris.train, test=iris.test, cl=iris.trainLabels, k=3)
```

Compute accuracy

We built a classifier with 98% accuracy.

It's often a good idea to scale the variables for clustring to make the distance calculations better. However in this case, the 3 predictors are roughly in the same scale so it's probably not necessary.

```
results <- iris_pred == iris.testLabels
acc <- length(which(results==TRUE)) / length(results)
# or combine into one line:
#acc <- length(which(iris_pred == iris.testLabels)) / length(iris_pred)
acc
```

```
## [1] 0.98
```

Multiclass Classification with Logistic Regression

Clustering can easily divide the iris data sets into 3 clusters. Can we do the same thing with logistic regression? That is, can we classify into 3 classes instead of just doing a binary classification as we have done before?

One versus all

Yes, we can, with a technique called "one versus all". First we classify setosa versus not setosa. Then we remove setosa from the data set and classify versicolor versus not versicolor.

Copy iris to iris2, and make iris2$Species be 1 if it is setosa and 0 otherwise.

Then train a logistic regression model for setosa v. not-setosa.

```
iris2 <- iris
iris2$Species <- ifelse (iris2$Species=="setosa",1,0)
iris2$Species <- as.factor(iris2$Species)
```

```
# create train and test sets for iris 2
set.seed(1958)  # setting a seed gets the same results every time
ind2 <- sample(2, nrow(iris2), replace=TRUE, prob=c(0.67, 0.33))
iris.train2 <- iris2[ind2==1,]
iris.test2 <- iris2[ind2==2,]

# create a logistic regression model for setosa v. not-setosa
glm1 <- glm(iris2$Species~., data=iris2, family="binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
glm1.probs <- predict(glm1, newdata=iris.test2, type="response")
glm1.pred <- rep(0, nrow(iris.test2))
glm1.pred[glm1.probs>0.5] = 1
table(glm1.pred, iris.test2$Species)
```

```
##
## glm1.pred  0  1
##         0 33  0
##         1  0 17
```

```
mean(glm1.pred==iris.test2$Species)
```

```
## [1] 1
```

Wow, we got 100% accuracy.

Let's remove setosa from the data set, making a new data set, iris3. Then alter the Species to be 1 for versicolor and 0 otherwise. Finally, train a new model.

```
iris3 <- iris[51:150,]  # remove setosa
iris3$Species <- ifelse (iris3$Species=="versicolor",1,0)
iris3$Species <- as.factor(iris3$Species)
str(iris3)
```

```
## 'data.frame':    100 obs. of  5 variables:
##  $ Sepal.Length: num  7 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 ...
##  $ Sepal.Width : num  3.2 3.2 3.1 2.3 2.8 2.8 3.3 2.4 2.9 2.7 ...
##  $ Petal.Length: num  4.7 4.5 4.9 4 4.6 4.5 4.7 3.3 4.6 3.9 ...
##  $ Petal.Width : num  1.4 1.5 1.5 1.3 1.5 1.3 1.6 1 1.3 1.4 ...
##  $ Species     : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```
ind3 <- sample(2, nrow(iris3), replace=TRUE, prob=c(0.67, 0.33))
iris.train3 <- iris3[ind3==1,]
iris.test3 <- iris3[ind3==2,]

# train a logistic regression model for versicolor v. non-versicolor
glm2 <- glm(iris3$Species~., data=iris3, family="binomial")
glm2.probs <- predict(glm2, newdata=iris.test3, type="response")
glm2.pred <- rep(0, nrow(iris.test3))
glm2.pred[glm2.probs>0.5] = 1
table(glm2.pred, iris.test3$Species)
```

```
##
## glm2.pred  0  1
##         0 14  0
```

```
##          1  0 16
```

```
mean(glm2.pred==iris.test3$Species)
```

```
## [1] 1
```

100% accuracy again!