Introduction to Data Science

DSA1101

Semester 1, 2018/2019
Week 5

k-nearest neighbor classifier in `R`

k-nearest neighbor classifier in `R`

- We have studied the *k*-nearest neighbor algorithm as an example of a classifier, and introduced some diagnostic metrics to evaluate the perfomance of a classifier.
- This week, we will learn how to implement the *k*-nearest neighbors algorithm in `R`.

Example: The Stock Market Data

Source: Yahoo Finance

- The CSV file `Smarket.csv` contains data on percentage returns for the S&P 500 stock index over 1250 days, from the beginning of 2001 until the end of 2005.

Source: Yahoo Finance

- For each date, we have recorded the percentage returns for each of the five previous trading days, `Lag1` through `Lag5`.

Example: The Stock Market Data

Source: Yahoo Finance

- The data also contains the variables `Volume` (the number of shares traded on the previous day, in billions), `Today` (the percentage return on the date in question) and `Direction` (whether the market was Up or Down on this date).

k-nearest neighbor classifier in R

- The CSV file `Smarket.csv` from the R package 'ISLR' has been uploaded to IVLE under `Files/LectureNotes/Datasets`

```
> market = read.csv("Smarket.csv")
> dim(market)
[1] 1250   10
> head(market)
  X Year   Lag1   Lag2   Lag3   Lag4   Lag5 Volume  Today Direction
1 1 2001  0.381 -0.192 -2.624 -1.055  5.010 1.1913  0.959        Up
2 2 2001  0.959  0.381 -0.192 -2.624 -1.055 1.2965  1.032        Up
3 3 2001  1.032  0.959  0.381 -0.192 -2.624 1.4112 -0.623      Down
4 4 2001 -0.623  1.032  0.959  0.381 -0.192 1.2760  0.614        Up
5 5 2001  0.614 -0.623  1.032  0.959  0.381 1.2057  0.213        Up
6 6 2001  0.213  0.614 -0.623  1.032  0.959 1.3491  1.392        Up
```

k-nearest neighbor classifier in R

- A summary of the variables in the stock market data set:

```
> summary(market[,2:10])
      Year           Lag1                Lag2
 Min.   :2001   Min.   :-4.922000   Min.   :-4.922000
 1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500
 Median :2003   Median : 0.039000   Median : 0.039000
 Mean   :2003   Mean   : 0.003834   Mean   : 0.003919
 3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750
 Max.   :2005   Max.   : 5.733000   Max.   : 5.733000
      Lag3                Lag4                Lag5
 Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.92200
 1st Qu.:-0.640000   1st Qu.:-0.640000   1st Qu.:-0.64000
 Median : 0.038500   Median : 0.038500   Median : 0.03850
 Mean   : 0.001716   Mean   : 0.001636   Mean   : 0.00561
 3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.59700
 Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.73300
     Volume            Today            Direction
 Min.   :0.3561   Min.   :-4.922000   Down:602
 1st Qu.:1.2574   1st Qu.:-0.639500   Up  :648
 Median :1.4229   Median : 0.038500
 Mean   :1.4783   Mean   : 0.003138
 3rd Qu.:1.6417   3rd Qu.: 0.596750
 Max.   :3.1525   Max.   : 5.733000
```

k-nearest neighbor classifier in R

- We will set the data entries from the years 2001-2004 as our training data for the classifier, and then test the classifier on the remaining data from the year 2005

```
> train =(market$Year <2005)
> train.data = market[train,]
> test.data  = market[!train ,]
>
> dim(test.data)
[1] 252  10
```

k-nearest neighbor classifier in R

- Our aim is to predict whether the S&P stock index will go up or down on any given day, based on its percentage returns in the preceding five days.
- We will use the `knn()` function from the 'class' package in R to perform k-nearest neighbor classification.
- `knn()` performs k-nearest neighbour classification for test set from training set. For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the k^{th} nearest vector, all candidates are included in the vote.

(e.g when the two labeled points are equal in distance to the test data point.)

k-nearest neighbor classifier in `R`

- We need at least four inputs to `knn()`:

(i) A matrix containing the predictors or features x associated with the training data

(ii) A matrix containing the predictors or features x associated with the data for which we wish to make predictions

(iii) A vector containing the class labels for the training observations i.e y value for x value in (i)

(iv) A value for k, the number of nearest neighbors to be used by the classifier

k-nearest neighbor classifier in R

- We need the predictor or feature x matrices for training and testing data sets, as well as the label or outcome vector y for the training data

```
train.x = train.data[,c("Lag1","Lag2","Lag3","Lag4","Lag5")]
test.x = test.data[,c("Lag1","Lag2","Lag3","Lag4","Lag5")]
train.y = train.data[,c("Direction")]
```

k-nearest neighbor classifier in R

- Now the `knn()` function can be used to predict the market's movement for the dates in 2005.
- We set a random seed before we apply `knn()` because the classification is decided by majority vote, with ties broken at random.
- Therefore, a seed must be set in order to ensure reproducibility of results.

k-nearest neighbor classifier in R

- We perform k-nearest neighbors classification with $k = 1$:

```
> set.seed(1)
>test.y = test.data[,c("Direction")]

> knn.pred = knn(train.x,test.x,train.y,k=1)
> confusion.matrix=table(knn.pred, test.y)
> confusion.matrix
        test.y
knn.pred Down Up
    Down   55 66
    Up     56 75
```

k-nearest neighbor classifier in R

- We can evaluate the performance of the 1-nearest neighbor classifier with metrics that we have learnt last week, such as *accuracy*:

```
>confusion.matrix
        test.y
knn.pred Down Up
    Down   55 66
    Up     56 75

>(55+75)/(55+66+56+75)
[1] 0.515873
>sum(diag(confusion.matrix))/sum(confusion.matrix)
[1] 0.515873
```

- Our results indicate that only $\approx 51.6\%$ of the observations are correctly predicted.

k-nearest neighbor classifier in R

- We repeat k-nearest neighbors classification with $k = 5$:

```
> knn.pred = knn(train.x,test.x,train.y,k=5)
> confusion.matrix=table(knn.pred, test.y)
> confusion.matrix
        test.y
knn.pred Down Up
    Down   50 57
    Up     61 84
>(50+84)/(55+66+56+75)
[1] 0.531746
```

- The performance has improved, with $\approx 53.2\%$ of the observations are correctly predicted.

Example: Caravan Insurance Data

Source: Wikipedia

- The CSV file `Caravan.csv` contains data on 5822 real customer records on caravan insurance purchase

Example: Caravan Insurance Data

Source: Wikipedia

- This dataset is owned and supplied by the Dutch datamining company Sentient Machine Research, and is based on real world business data
- More on this dataset here `http://liacs.leidenuniv.nl/~puttenpwhvander/library/cc2000/data.html`

Example: Caravan Insurance Data

Source: Wikipedia

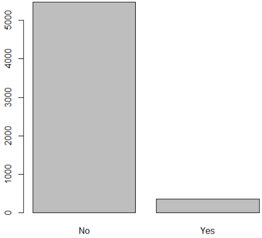
- Each record consists of 86 variables, containing sociodemographic data (variables 1-43) and product ownership (variables 44-86).
- Variable 86 (`Purchase`) indicates whether the customer purchased a caravan insurance policy

Example: Caravan Insurance Data

Source: Wikipedia

- The CSV file `Caravan.csv` has been uploaded to IVLE under `Files/LectureNotes/Datasets`

Example: Caravan Insurance Data




- ```
  > caravan = read.csv("Caravan.csv")
  > head(caravan$Purchase)
  [1] No No No No No No
  Levels: No Yes
  > summary(caravan$Purchase)
    No  Yes
  5474  348
  > plot(caravan$Purchase)
  ```
- In this data set, only $\approx 6\%$ of people purchased caravan insurance.
  ```

Example: Caravan Insurance Data

- Because the k-nearest neighbors classifier predicts the class of a given test observation by identifying the observations that are nearest to it, the scale of the variables matters
- The reason is analogous to the influence of scaling variables for k-means clustering in the unsupervised setting

Example: Caravan Insurance Data

- Any variables that are on a large scale will have a much larger effect on the Euclidean distance calculation between the observations, and hence on the k-nearest neighbors classifier, than variables that are on a small scale.
- For instance, imagine a data set that contains two variables, salary and age (measured in dollars and years, respectively).
- As far as k-nearest neighbors classification is concerned, a difference of \$1,000 in salary is enormous compared to a difference of 50 years in age. Consequently, salary will drive the k-nearest neighbors classification results, and age will have almost no effect.

Example: Caravan Insurance Data

- This is contrary to our intuition that a salary difference of \$1, 000 is quite small compared to an age difference of 50 years.
- Furthermore, the importance of scale to the k-nearest neighbors classifier leads to another issue: if we measured salary in Japanese yen, or if we measured age in minutes, then we would get quite different classification results from what we get if these two variables are measured in dollars and years.

Example: Caravan Insurance Data

- A good way to handle this problem is to standardize the data so that all standardize variables are given a mean of zero and a standard deviation of one. Then all variables will be on a comparable scale.
- The `scale()` function in R performs this.

Example: Caravan Insurance Data

- In standardizing the data, we exclude column 86, because that is the qualitative `Purchase` variable.
-

```
> # exclude ID column
> caravan=caravan[,-1]
> standardized.X= scale(caravan[,-86])
> var(Caravan [,1])
[1] 165.0378
> var(Caravan [,2])
[1] 0.1647078
> var( standardized.X[,1])
[1] 1
> var( standardized.X[,2])
[1] 1
```

- Now every column of `standardized.X` has a standard deviation of one and a mean of zero.

Example: Caravan Insurance Data

- We now split the observations into a test set, containing the first 1,000 observations, and a training set, containing the remaining observations.
- ```
  > test=1:1000
  > train.X=standardized.X[-test ,]
  > test.X =standardized.X[test ,]
  > train.Y=caravan$Purchase[-test]
  > test.Y =caravan$Purchase[test]
  ```
  ```

Example: Caravan Insurance Data

- We fit a k-nearest neighbors model on the training data using $k = 1$, and evaluate its performance on the test data.
- ```
  > set.seed (1)
  > knn.pred = knn(train.X,test.X,train.Y,k=1)
  > confusion.matrix=table(test.Y,knn.pred)
  > confusion.matrix
        knn.pred
  test.Y  No Yes
     No  873  68
     Yes  50   9
  ```
- Instead of looking at *accuracy* of the classifier, in this example we look at another metric, *precision*
  ```

Example: Caravan Insurance Data

- Suppose that there is some non-trivial cost to trying to sell insurance to a given individual. For instance, perhaps a salesperson must visit each potential customer.
- If the company tries to sell insurance to a random selection of customers, then the success rate will be only $\approx 6\%$, which may be far too low given the costs involved.

Example: Caravan Insurance Data

- Suppose that there is some non-trivial cost to trying to sell insurance to a given individual. For instance, perhaps a salesperson must visit each potential customer.
- If the company tries to sell insurance to a random selection of customers, then the success rate will be only $\approx 6\%$, which may be far too low given the costs involved.

Example: Caravan Insurance Data

- Instead, the company would like to try to sell insurance only to customers who are likely to buy it.
- So the overall error rate (or *accuracy*) is not of interest.

Example: Caravan Insurance Data

- Instead, the company may want to use the classifier to predict who are the potential customers likely to purchase insurance
- Then the metric *precision* will be important, since it relates the proportion of individuals who will actually purchase the insurance, among the group of individuals who are predicted to purchase insurance
- This is a useful metric for targeted marketing

Example: Caravan Insurance Data

- For the k-nearest neighbors classifier with $k = 1$, the precision is $\approx 11.7\%$
- ```
  > set.seed (1)
  > knn.pred = knn(train.X,test.X,train.Y,k=1)
  > confusion.matrix=table(test.Y,knn.pred)
  > confusion.matrix
          knn.pred
  test.Y   No Yes
     No   873  68
     Yes   50   9
  > 9/(68+9)
  [1] 0.1168831
  ```
- This is double the success rate that one would obtain from randomly selecting potential customers.
  ```

Example: Caravan Insurance Data

- Using $k = 3$, the precision increases to $\approx 19\%$
- ```
  > knn.pred = knn(train.X,test.X,train.Y,k=3)
  > confusion.matrix=table(test.Y,knn.pred)
  > confusion.matrix
          knn.pred
  test.Y   No Yes
     No   920  21
     Yes   54   5
  > 5/(21+5)
  [1] 0.1923077
  ```
- This is over three times the success rate that one would obtain from randomly selecting potential customers.
  ```

Example: Caravan Insurance Data

- Using $k = 5$, the precision increases to $\approx 27\%$
-
```
> knn.pred = knn(train.X,test.X,train.Y,k=5)
> confusion.matrix=table(test.Y,knn.pred)
> confusion.matrix
        knn.pred
test.Y   No Yes
    No  930  11
    Yes  55   4
> 4/(11+4)
[1] 0.2666667
```
- This is over four times the success rate that one would obtain from randomly selecting potential customers.

Example: Customer Churn

Source: *www.kaggle.com*

- Customer churn is the loss of clients or customers
- Banks, telephone service companies, Internet service providers, pay TV companies and insurance firms often use customer churn analysis and customer churn rates as one of their key business metrics

Example: Customer Churn

Source: *www.kaggle.com*

- This is because the cost of retaining an existing customer is far less than acquiring a new one.
- Companies from these sectors often have customer service branches which attempt to win back defecting clients, because recovered long-term customers can be worth much more to a company than newly recruited clients.

Example: Customer Churn

- In our example, a wireless telecommunications company wants to predict whether a customer will churn (switch to a different company) in the next six months.
- With a reasonably accurate prediction of a person's churning, the sales and marketing groups can attempt to retain the customer by offering various incentives.

Example: Customer Churn

- Data on 8,000 current and prior customers was obtained. The variables collected for each customer follow:

(i) Age (years)

(ii) Married (true/false)

(iii) Duration as a customer (years)

(iv) Churned_contacts (count)-Number of the customer's contacts that have churned (count)

(v) Churned (true/false)-Whether the customer churned

Example: Customer Churn

- The customer churn dataset is available as the CSV file 'churn.CSV' on the course website

-
```
> churn = read.csv("churn.CSV")
> head(churn)
  ID Churned Age Married Cust_years Churned_contacts
1  1       0  61       1          3                1
2  2       0  50       1          3                2
3  3       0  47       1          2                0
4  4       0  50       1          3                3
5  5       0  29       1          1                3
6  6       0  43       1          4                3
> summary(as.factor(churn$Churned))
   0    1
6257 1743
```

- About 21.8% of the customers churned.

Example: Customer Churn

- We take the first 4000 customers' records as training data, and test the k-nearest neighbors classifier on the remaining 4000 customers (testing data).
-

```r
#Remove ID column
churn= churn[,-1]
#Standardize continuous variables
churn[,c("Age","Cust_years","Churned_contacts")]=
scale(churn[,c("Age","Cust_years","Churned_
    contacts")])

churn.X = churn[,-1]

test=1:4000
train.X=churn.X[-test,]
test.X =churn.X[test ,]
train.Y=churn$Churned[-test]
test.Y =churn$Churned[test]
```

Example: Customer Churn

- We are primarily interested in the metric *precision*, since the sales and marketing division can attempt to target the customer who are going to churn by offering various incentives.
- With $k = 1$, we get $\approx 52.4\%$ precision
- ```
  > set.seed (1)
  > knn.pred = knn(train.X,test.X,train.Y,k=1)
  > confusion.matrix=table(test.Y,knn.pred)
  > confusion.matrix
         knn.pred
  test.Y    0    1
       0 2801  354
       1  456  389
  > 389/(389+354)
  [1] 0.5235532
  ```
- We are more than two times more likely to target customers who are going to churn, if the sales and marketing division zero in on the group predicted to churn, than a random selection of customers.
  ```

Example: Customer Churn

- With $k = 3$, we get $\approx 57\%$ precision
- ```
  > set.seed (1)
  > knn.pred = knn(train.X,test.X,train.Y,k=3)
  > confusion.matrix=table(test.Y,knn.pred)
  > confusion.matrix
        knn.pred
  test.Y    0    1
       0 2852  303
       1  444  401
  > 401/(401+303)
  [1] 0.5696023
  ```
  ```

Example: Customer Churn

- With $k = 5$, we get $\approx 61\%$ precision
- ```
  > set.seed (1)
  > knn.pred = knn(train.X,test.X,train.Y,k=5)
  > confusion.matrix=table(test.Y,knn.pred)
  > confusion.matrix
        knn.pred
  test.Y    0    1
       0 2889  266
       1  425  420
  > 420/(420+266)
  [1] 0.6122449
  ```
  ```