MSBA 7027 Machine Learning
K-Nearest Neighbors

Zhengli Wang

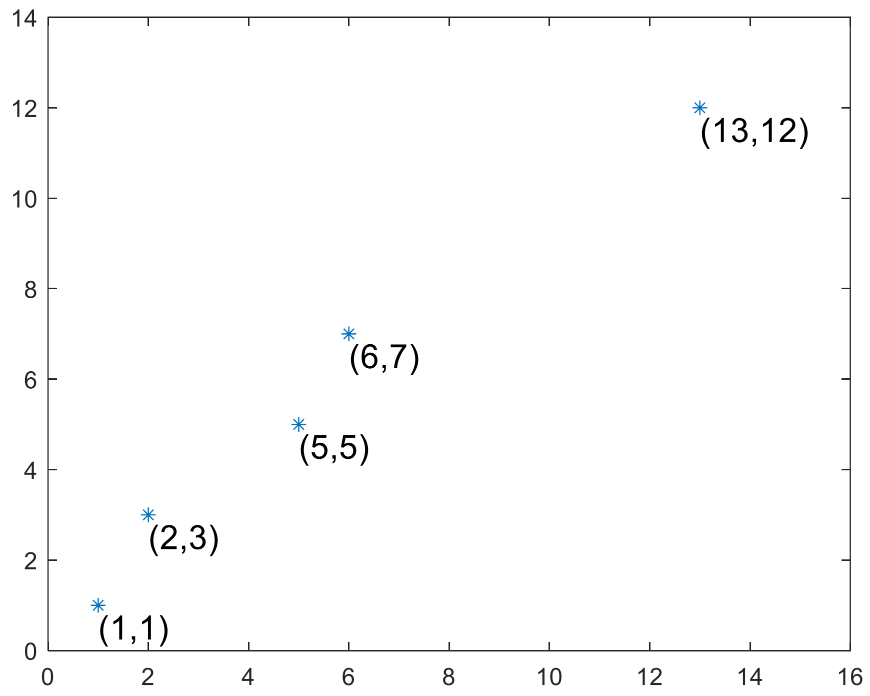
Faculty of Business and Economics
The University of Hong Kong
2023

K-Nearest Neighbors (KNN)

- Non-parametric method
  - Can be used for both regression & classification problems
- Given K and a prediction point $x^{(n+1)}$,
  - KNN identifies the closest K training observations
  - Then estimate $f(x^{(n+1)})$ using their averages

K-Nearest Neighbors (KNN)

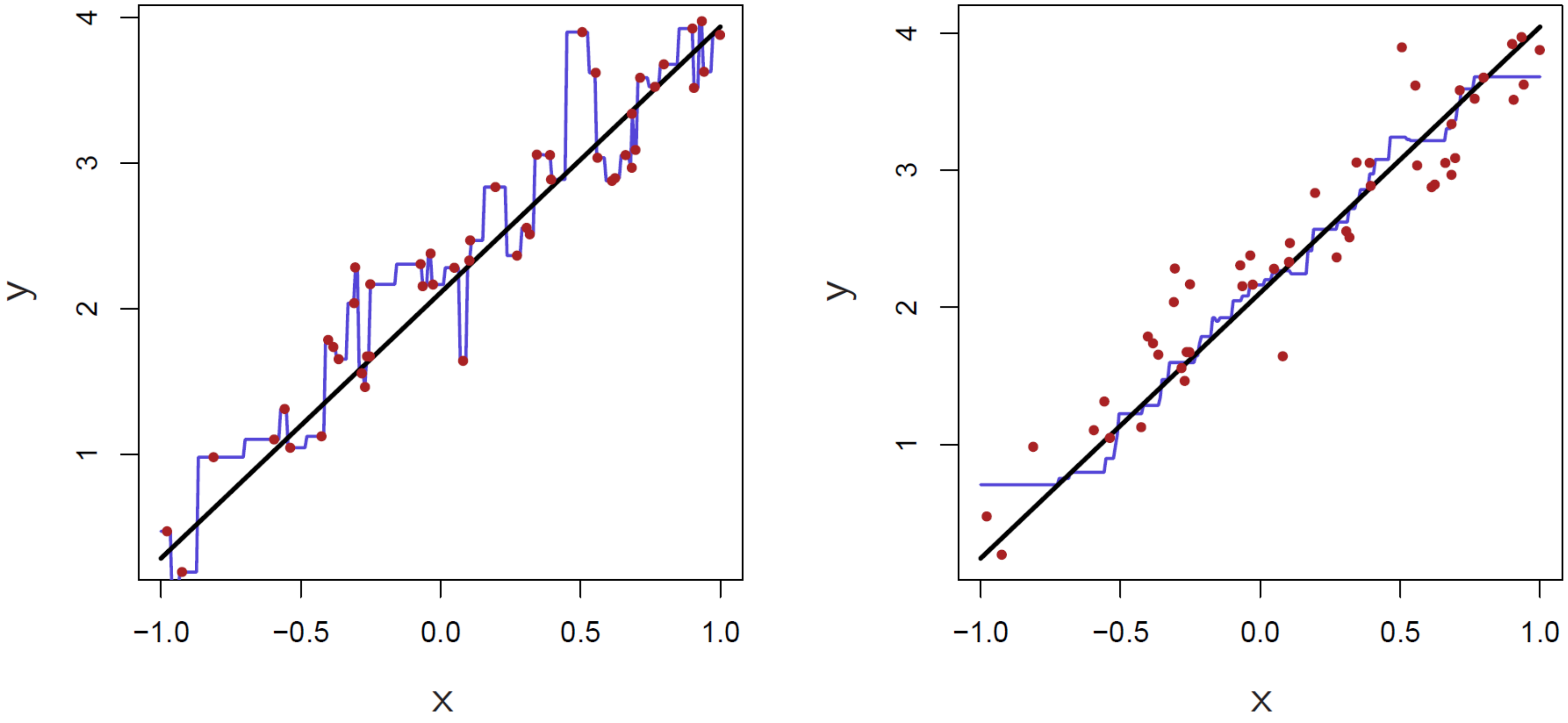
- One-dimensional Example
  - $(x^{(i)}, y^{(i)}) = (1,1), (2,3), (5,5), (6,7), (13,12)$
  - K = 3



K-Nearest Neighbors (KNN)

- In practice, choose K by cross-validation (minimize SSE in test set)
  - Why not minimize SSE in training set (like in OLS)
- Value of K: Bias-variance tradeoff
- Small value of K vs large value of K

K-Nearest Neighbors (KNN)



Which one has a smaller K?

K-Nearest Neighbors (KNN)

- Features multi-dim: Euclidean distance

- Classification problem: Majority vote

K-Nearest Neighbors (KNN)

- Sample final exam question
  - $x^{(i)} = (x_1^{(i)}, x_2^{(i)})$
  - $(x^{(i)}, y^{(i)}) = \ (0, 3, 5), (4, 0, 1), (8, 3, 7)$
  - K = 2
  - Given $x^{(4)} = (2, 2)$, what is $\hat{y}$?
  - Derive $\hat{y}$ over the whole plane of (x_1, x_2).

KNN – implementation in R

Getting started

Load relevant packages

```
# Example of installing a package
install.packages('FNN')

library(FNN) # For func knn
library(gmodels) # For func CrossTable
```

Getting started

Load Data

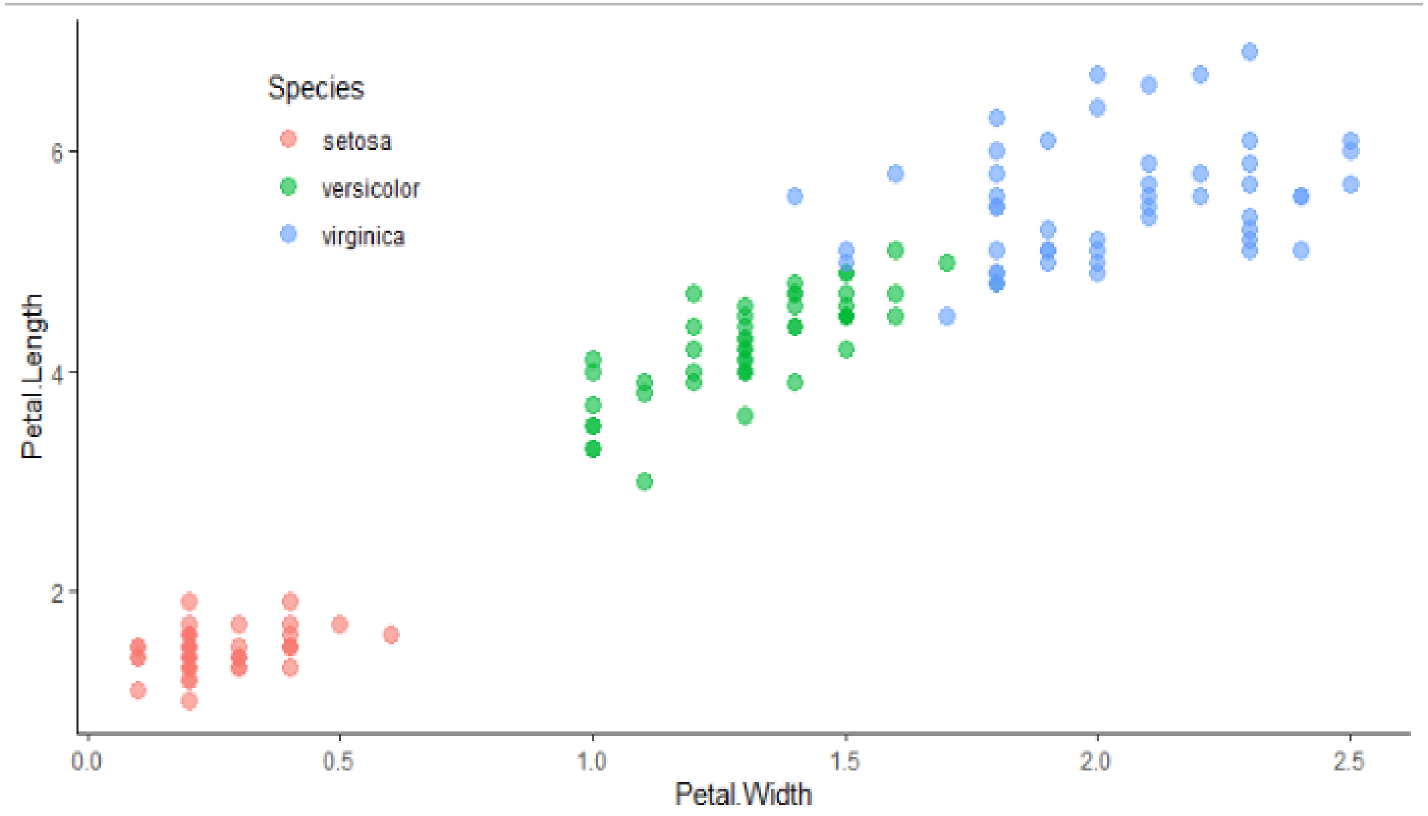
```
# Loading iris dataset
iris.rawData <- iris

# Viewing iris dataset structure and attributes
summary(iris.rawData)
```

```
> summary(iris.data)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species  
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50  
 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                  
```

3 Classes of Iris Species: Setosa, versicolor, virginica

Data visualisation

Features: Petal.Width, Petal.Length



Data Standardization

```
# standardize data
standardize <- function(x) {
  return ( x - mean(x) )/( sd(x) )
}
# Only standardize the first 4 columns (5th column is label)
iris.standardizeData = iris.rawData
for(i in seq(1,4)){
  iris.standardizeData[,i] = standardize(iris.rawData[,i])
}
# Split into train & test set
set.seed(123)
split  <- rsample::initial_split(iris.standardizeData, prop = 0.7, strata = "Species")
iris.train  <- rsample::training(split)
iris.test   <- rsample::testing(split)
iris.trainFeatMat = iris.train[,1:4]
iris.trainLabel <- iris.train[,5]
iris.testFeatMat <- iris.test[,1:4]
iris.testLabel <- iris.test[,5]
```

Perform KNN

knn function with the following parameters:

- Train feature matrix
- Test feature matrix
- Train labels
- A value for K

Returns predicted test labels

```
# Building our knn classifier
predictTestLabel <- knn(train = iris.trainFeatMat, test = iris.testFeatMat, cl = iris.trainLabel, k = 3)
```

Model Evaluation

# Contingency Table
CrossTable(x = iris.testLabel, y = predictTestLabel, prop.chisq = FALSE)

- **Overall, KNN performs very well**
    - Able to predict correctly almost all instances
    - Only misclassifies two instances

```
Total Observations in Table:  45


               | predictTestLabel
iris.testLabel |     setosa | versicolor |  virginica |  Row Total |
---------------|------------|------------|------------|------------|
        setosa |         15 |          0 |          0 |         15 |
               |      1.000 |      0.000 |      0.000 |      0.333 |
               |      1.000 |      0.000 |      0.000 |            |
               |      0.333 |      0.000 |      0.000 |            |
---------------|------------|------------|------------|------------|
    versicolor |          0 |         15 |          0 |         15 |
               |      0.000 |      1.000 |      0.000 |      0.333 |
               |      0.000 |      0.882 |      0.000 |            |
               |      0.000 |      0.333 |      0.000 |            |
---------------|------------|------------|------------|------------|
     virginica |          0 |          2 |         13 |         15 |
               |      0.000 |      0.133 |      0.867 |      0.333 |
               |      0.000 |      0.118 |      1.000 |            |
               |      0.000 |      0.044 |      0.289 |            |
---------------|------------|------------|------------|------------|
  Column Total |         15 |         17 |         13 |         45 |
               |      0.333 |      0.378 |      0.289 |            |
---------------|------------|------------|------------|------------|
```

Perform KNN

Note: For KNN-regression, syntax is the same except changing the function name to: knn.reg

End