STATS 266 Handout - Basic Machine Learning with R

Qi Wang

2025-05-12

Contents

1. Introduction

This tutorial introduces three fundamental machine learning algorithms—K-Nearest Neighbors (KNN), K-means clustering, and Decision Trees—using R. We explain each method's core concept, typical use cases, and advantages or limitations. Each section concludes with an applied example using the classic `iris` dataset.

```r
library(tidyverse)
library(class)
library(caret)
library(cluster)
library(rpart)
library(rpart.plot)
```

1. Dataset Overview

```r
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

3. K-Nearest Neighbors (KNN)

3.1 What is KNN?

K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm. It assumes that similar points are near each other in feature space. Given a query point, it searches the training data for the K closest examples (usually based on Euclidean distance), and the majority class among those neighbors becomes the predicted class.

Mathematically, the predicted class $\hat{y}$ is:

$$\hat{y} = \arg\max_{c} \sum_{i=1}^{K} I(y^{(i)} = c)$$

Where:

- $y^{(i)}$ is the label of the i-th nearest neighbor
- $I(\cdot)$ is the indicator function

Characteristics

- **Lazy learner:** No training phase, computation happens at query time.
- **Distance-sensitive:** Sensitive to feature scaling and irrelevant features.
- **Curse of dimensionality:** High-dimensional data can degrade performance.

3.2 KNN Example in R

```r
set.seed(123)
index <- createDataPartition(iris$Species, p = 0.8, list = FALSE)
train <- iris[index, ]
test <- iris[-index, ]

train_x <- train[, 1:4]
train_y <- train$Species
test_x <- test[, 1:4]
test_y <- test$Species

pred_knn <- knn(train = train_x, test = test_x, cl = train_y, k = 5)
confusionMatrix(pred_knn, test_y)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         0
##   virginica       0          0        10
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.8843, 1)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 4.857e-15
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           1.0000
## Specificity                 1.0000            1.0000           1.0000
## Pos Pred Value              1.0000            1.0000           1.0000
```

```
## Neg Pred Value              1.0000          1.0000          1.0000
## Prevalence                  0.3333          0.3333          0.3333
## Detection Rate              0.3333          0.3333          0.3333
## Detection Prevalence        0.3333          0.3333          0.3333
## Balanced Accuracy           1.0000          1.0000          1.0000
```

4. K-means Clustering

4.1 What is K-means?

K-means clustering is a centroid-based algorithm that partitions n observations into K clusters, where each observation belongs to the cluster with the nearest mean.

The objective is to minimize the within-cluster sum of squares (WCSS):

$$\text{WCSS} = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - \mu_k\|^2$$

Where:

- C_k is the set of points assigned to cluster k
- μ_k is the mean (centroid) of cluster k

Algorithm Steps

1. Initialize K centroids (random or K-means++)
2. Assign each point to the nearest centroid
3. Update centroids as the mean of assigned points
4. Repeat until convergence

Considerations

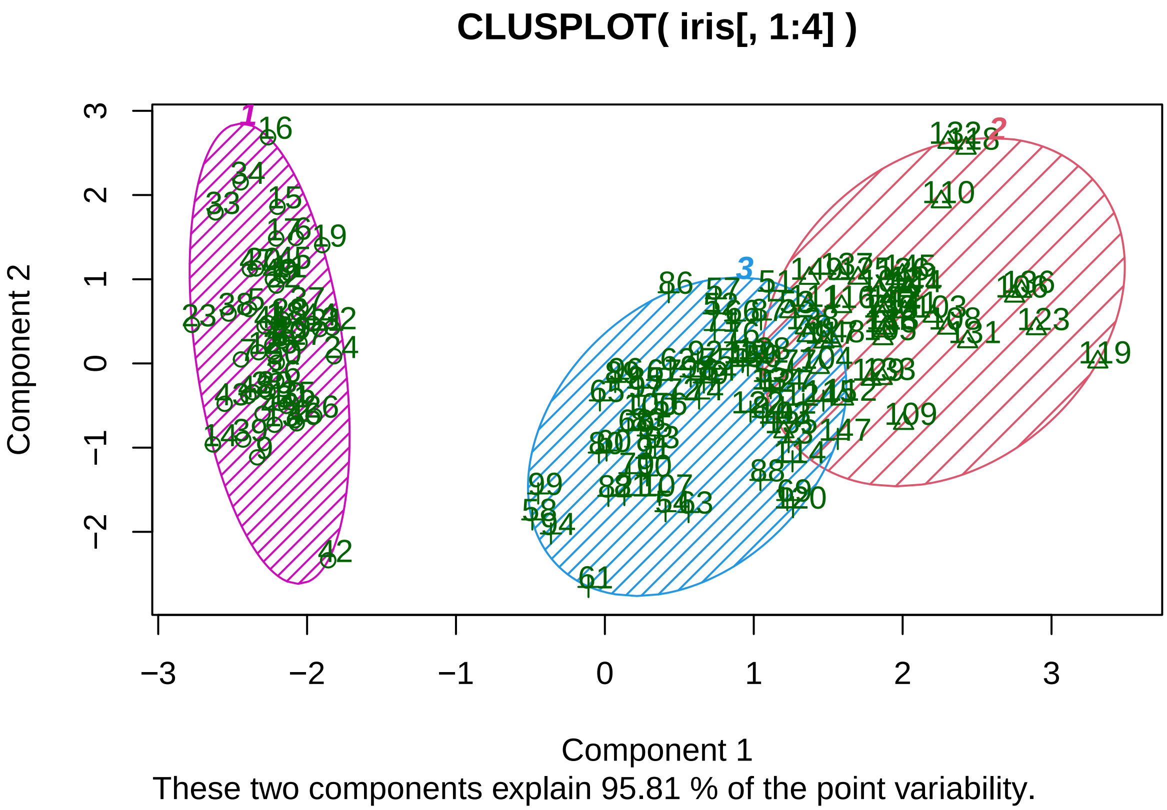
- Sensitive to initialization → use `nstart > 1`
- Distance metric is usually Euclidean
- Cannot handle non-spherical clusters or outliers well

4.2 K-means Example in R

```r
iris_kmeans <- kmeans(iris[, 1:4], centers = 3, nstart = 25)
table(iris_kmeans$cluster, iris$Species)
```

```
##
##      setosa versicolor virginica
##    1     50          0         0
##    2      0          2        36
##    3      0         48        14
```

```
clusplot(iris[, 1:4], iris_kmeans$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```



5. Decision Tree

5.1 What is a Decision Tree?

A decision tree is a recursive, hierarchical structure used for classification or regression. It splits the dataset based on input variables using binary decision rules such as:

$$\text{if } x_j \leq t \Rightarrow \text{go left, else go right}$$

The tree grows by selecting the best feature and threshold at each node using impurity measures such as **Gini impurity** or **Entropy**.

Splitting Criteria for Classification

- **Gini Impurity**:

$$G = 1 - \sum_{k=1}^{K} p_k^2$$

- **Entropy**:

$$H = -\sum_{k=1}^{K} p_k \log_2(p_k)$$

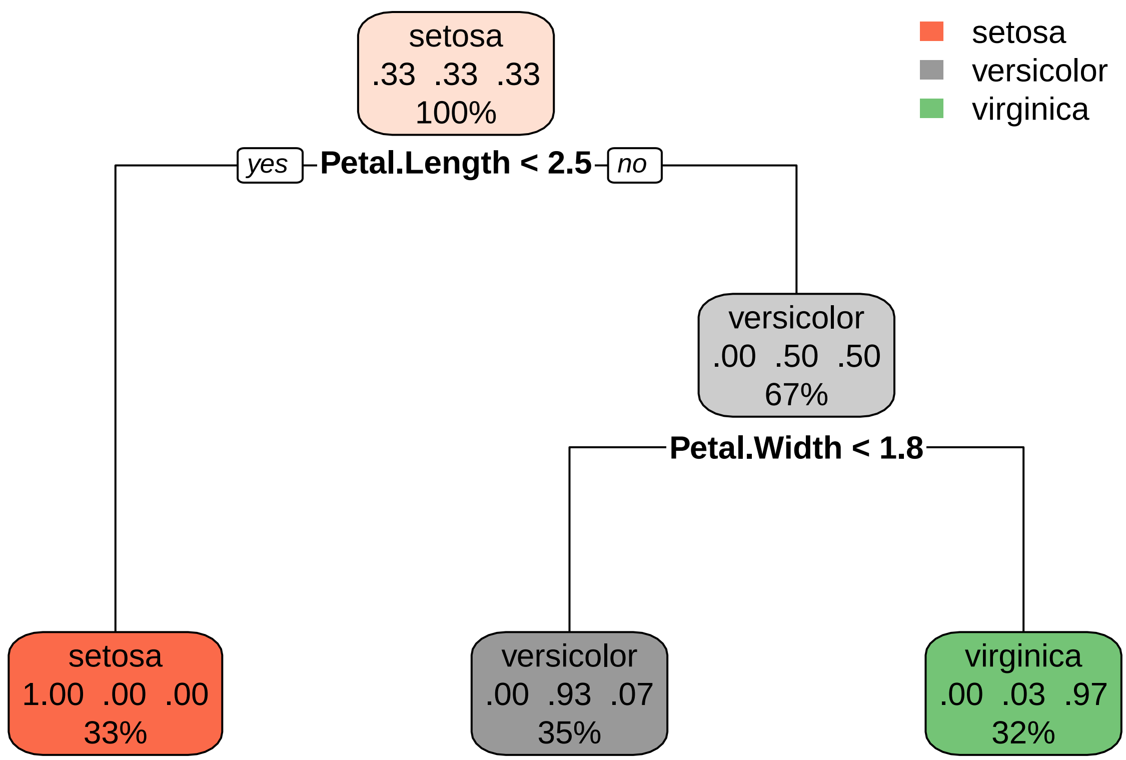
Where p_k is the proportion of class k in the node.

Characteristics

- **Interpretable:** Easy to visualize and explain
- **Greedy algorithm:** Splits are chosen locally at each node
- **Overfitting risk:** Often needs pruning or ensemble methods

5.2 Decision Tree Example in R

```
tree_model <- rpart(Species ~ ., data = train, method = "class")
rpart.plot(tree_model, type = 2, extra = 104)
```



```r
tree_pred <- predict(tree_model, newdata = test, type = "class")
confusionMatrix(tree_pred, test$Species)
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction   setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         2
##   virginica       0          0         8
##
## Overall Statistics
##
##                Accuracy : 0.9333
##                  95% CI : (0.7793, 0.9918)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 8.747e-12
##
##                   Kappa : 0.9
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.8000
## Specificity                 1.0000            0.9000           1.0000
## Pos Pred Value              1.0000            0.8333           1.0000
## Neg Pred Value              1.0000            1.0000           0.9091
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3333           0.2667
## Detection Prevalence        0.3333            0.4000           0.2667
## Balanced Accuracy           1.0000            0.9500           0.9000
```

6. Summary

We introduced three fundamental machine learning algorithms:

- **KNN** for classification using proximity in feature space.
- **K-means** for clustering based on similarity.
- **Decision Trees** for interpretable classification with rule-based splits.

Each was demonstrated on the iris dataset using R code examples.