Simple ML Algorithms and General Principles [1]

Shaobo Li

University of Kansas

[1] Partially based on Hastie, et al. (2009) ESL, and James, et al. (2013) ISLR

Clustering – an unsupervised learning method

- Goal: find subgroups of a sample observations
  - Not based on any single variable (e.g. gender, race)
  - Based on all given variables

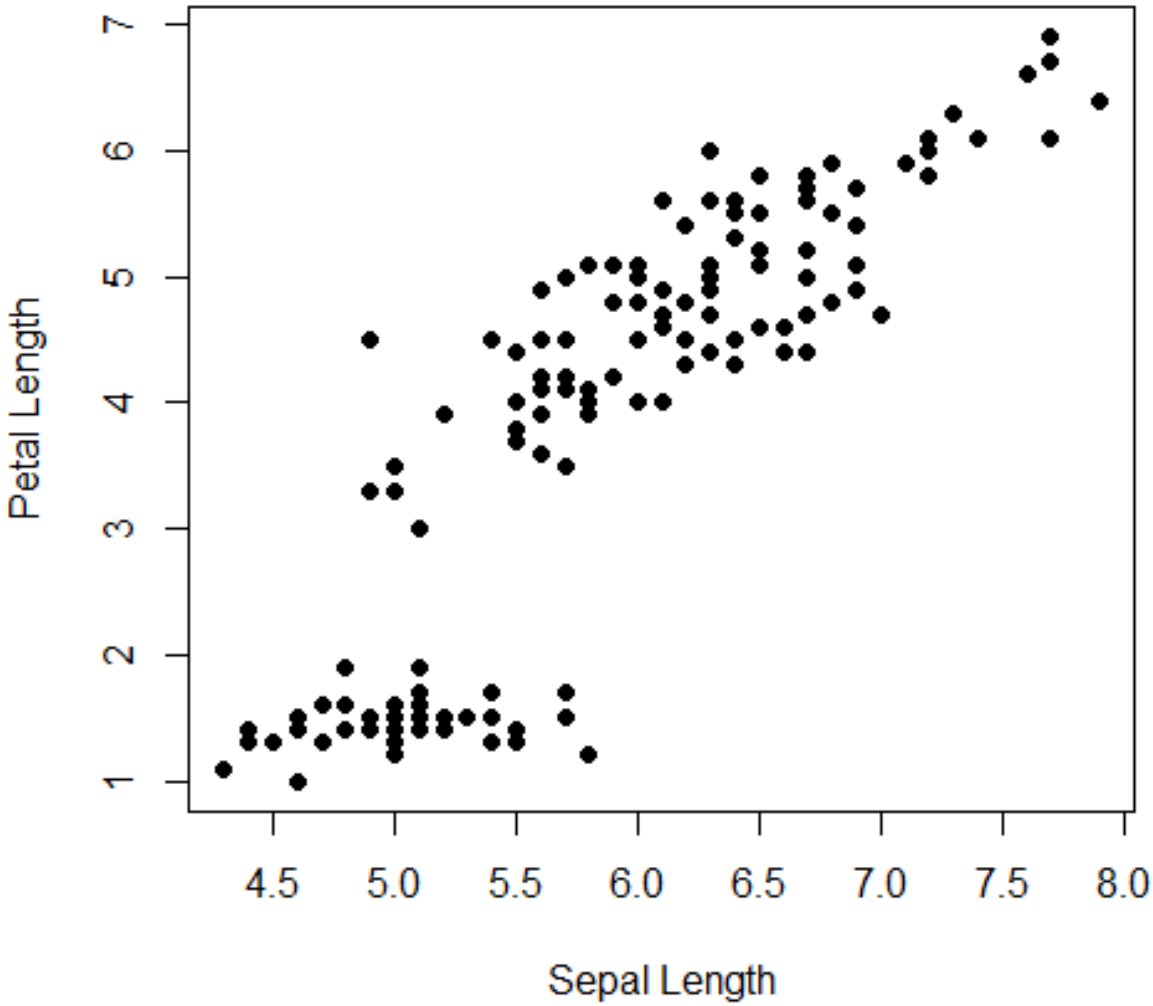
Clustering – an unsupervised learning method

- Goal: find subgroups of a sample observations
  - Not based on any single variable (e.g. gender, race)
  - Based on all given variables
- The number of subgroup is subjective

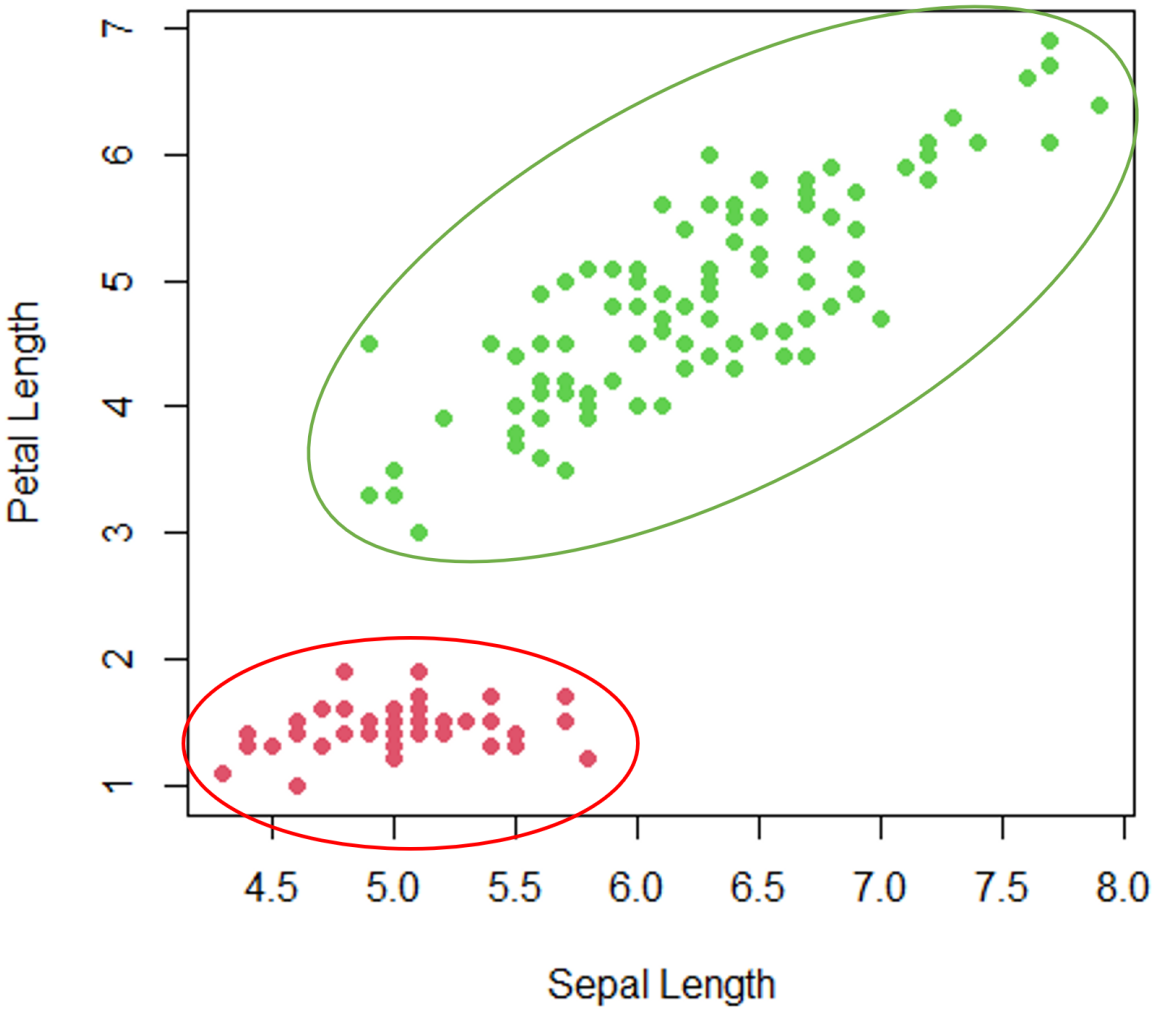
Clustering – an unsupervised learning method

- Goal: find subgroups of a sample observations
  - Not based on any single variable (e.g. gender, race)
  - Based on all given variables
- The number of subgroup is subjective
- Approaches:
  - K-means clustering
  - Hierarchical clustering
  - Model-based clustering

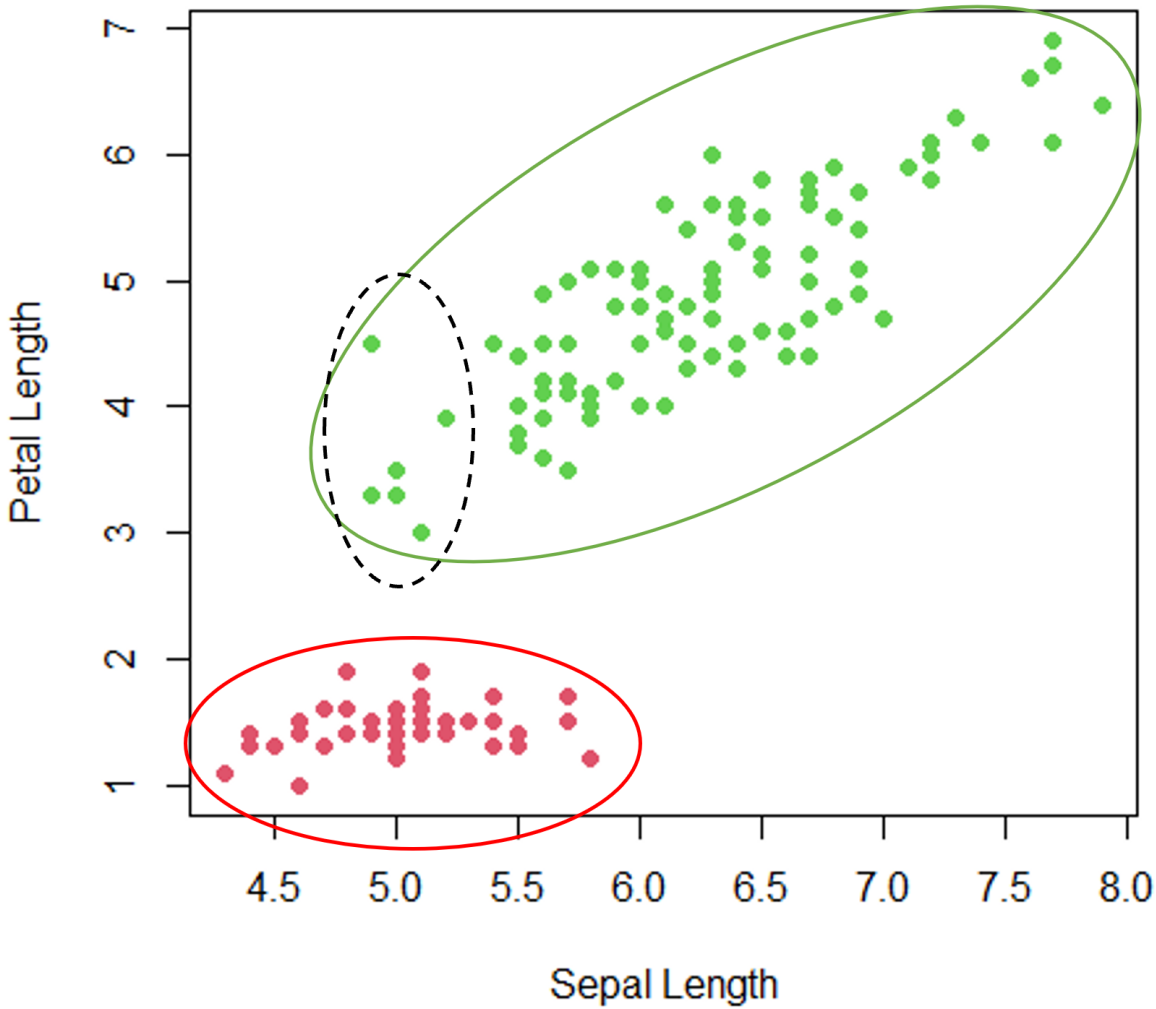
An example – Iris data



An example – Iris data



An example – Iris data

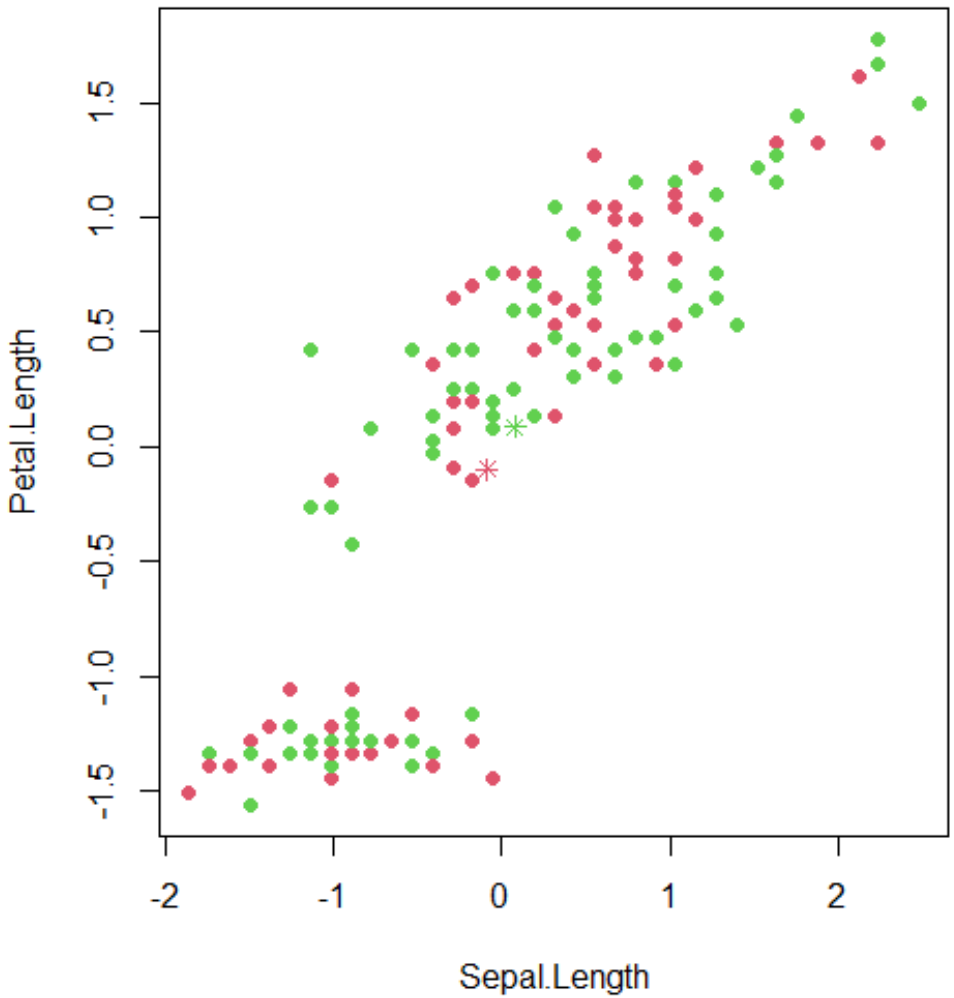


K-means clustering – step-by-step

Run the following R code, and see what it does.

```
iris1 <- scale(iris[,-c(2,4,5)])
n <- nrow(iris1)
index <- sample(2, n, replace = T)
iris.sub1 <- iris1[index==1,]
iris.sub2 <- iris1[index==2,]
mean.sub1 <- apply(iris.sub1, 2, mean)
mean.sub2 <- apply(iris.sub2, 2, mean)

plot(iris1, col=index+1, pch=16)
points(x=mean.sub1[1], y=mean.sub1[2], col=2, pch=8)
points(x=mean.sub2[1], y=mean.sub2[2], col=3, pch=8)
```

This is a random grouping (first step)

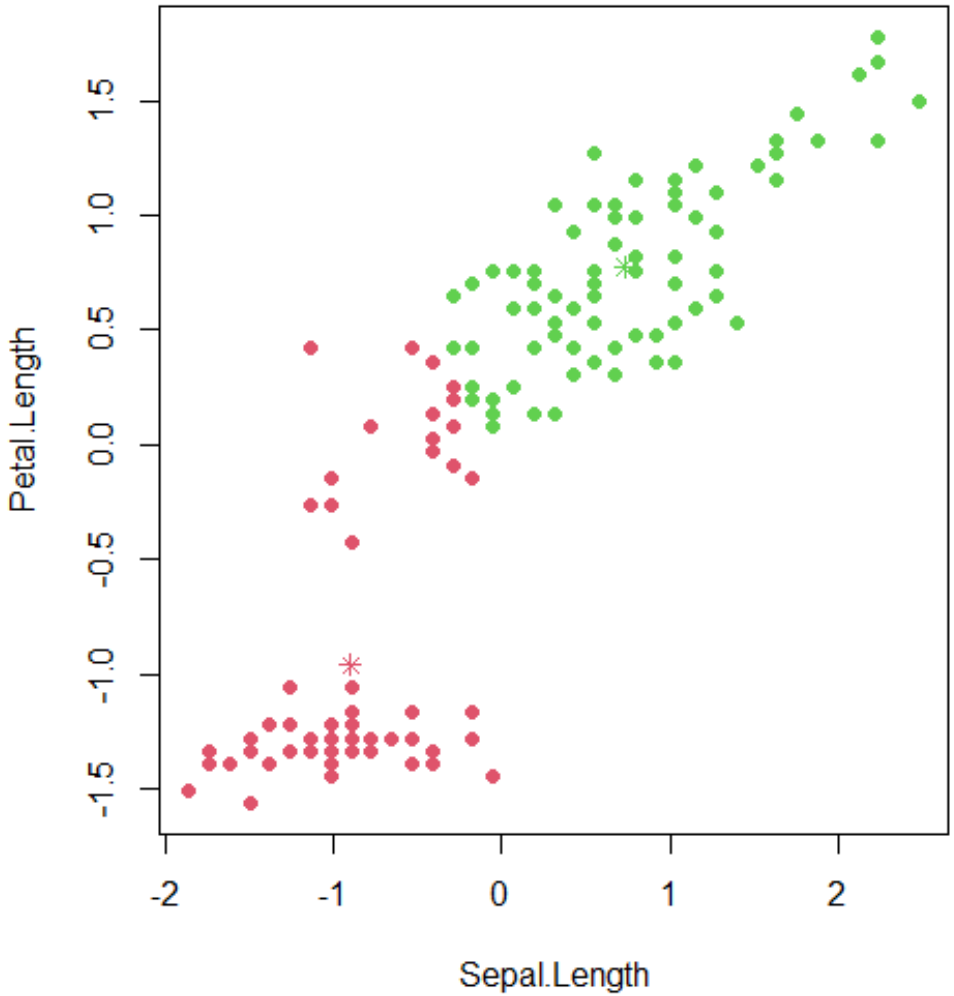


The next step

Run the following R code, and see what it does.

```
Eudist <- function(x, y) sqrt(sum((x-y)^2))

d1<-sapply(1:n,function(i) Eudist(mean.sub1,iris1[i,]))
d2<-sapply(1:n,function(i) Eudist(mean.sub2,iris1[i,]))
index.new <- apply(cbind(d1, d2), 1, which.min)
iris.sub1 <- iris1[index.new==1,]
iris.sub2 <- iris1[index.new==2,]
mean.sub1 <- apply(iris.sub1, 2, mean)
mean.sub2 <- apply(iris.sub2, 2, mean)

plot(iris1, col=index.new+1, pch=16)
points(x=mean.sub1[1], y=mean.sub1[2], col=2, pch=8)
points(x=mean.sub2[1], y=mean.sub2[2], col=3, pch=8)
```

Data points are regrouped (second step)



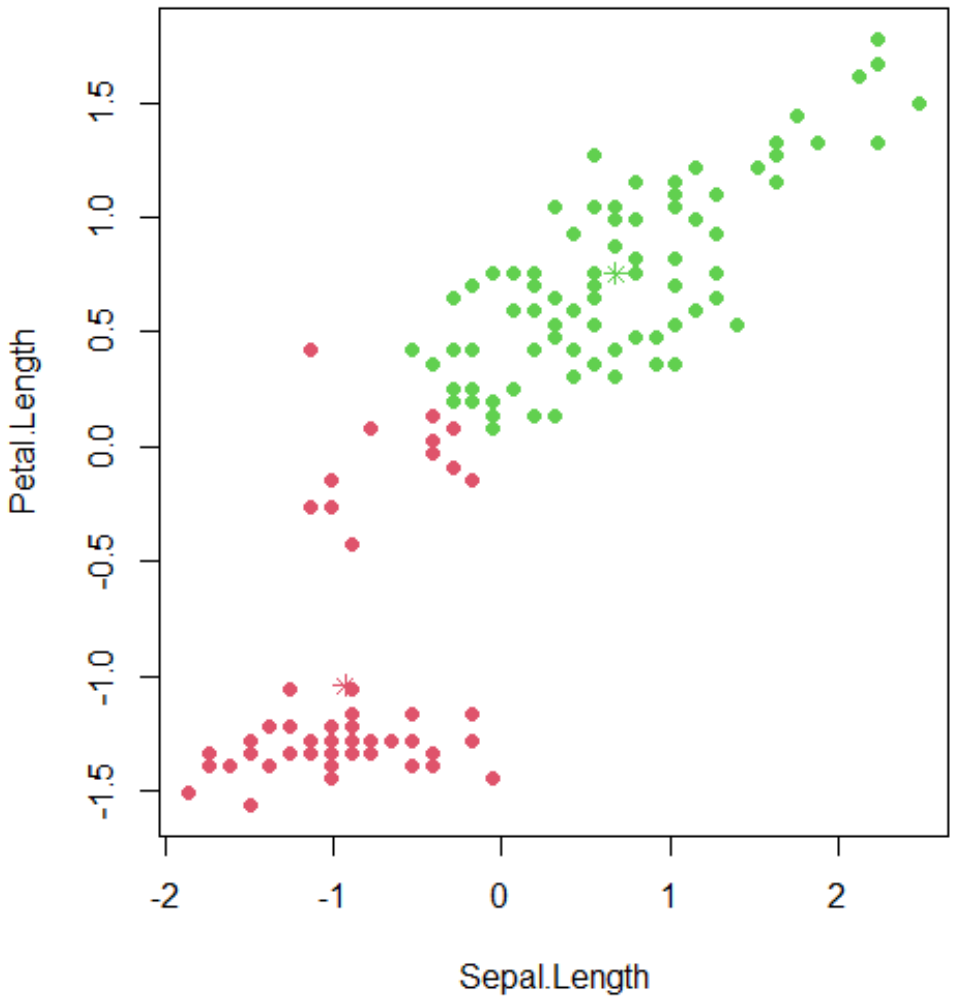
How does this happen?

Let's repeat the second step

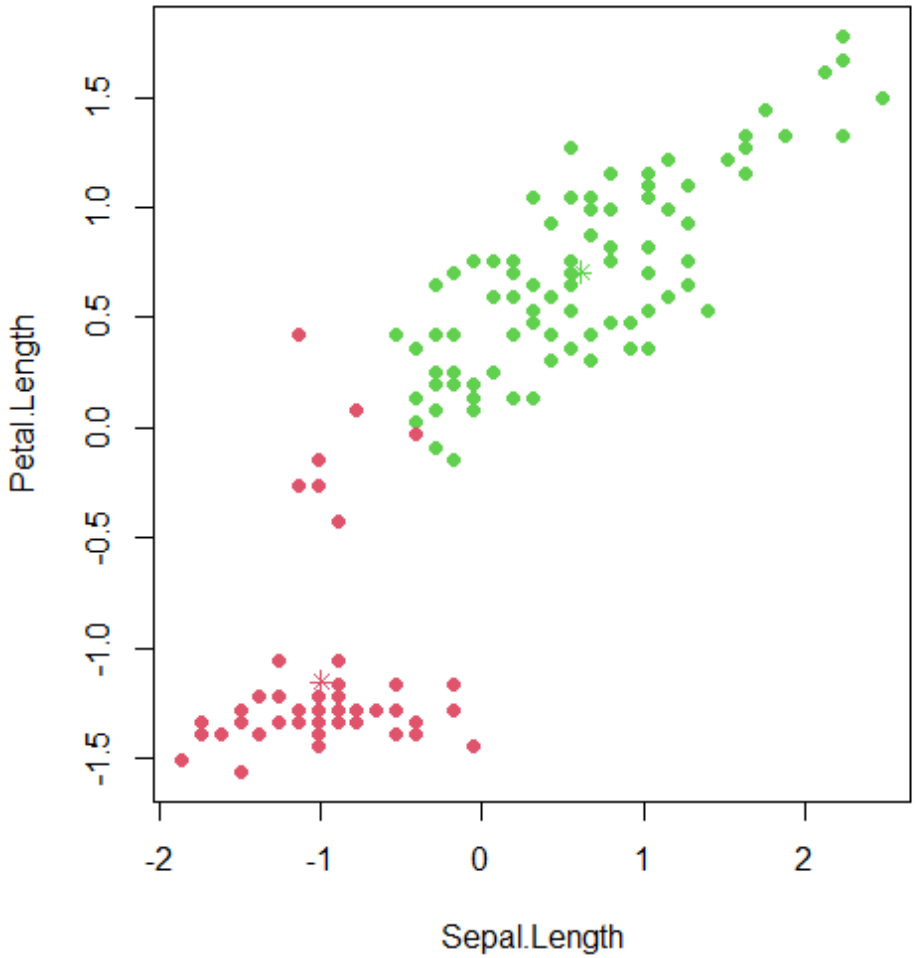
```
d1<-sapply(1:n,function(i) Eudist(mean.sub1,iris1[i,]))
d2<-sapply(1:n,function(i) Eudist(mean.sub2,iris1[i,]))
index.new <- apply(cbind(d1, d2), 1, which.min)
iris.sub1 <- iris1[index.new==1,]
iris.sub2 <- iris1[index.new==2,]
mean.sub1 <- apply(iris.sub1, 2, mean)
mean.sub2 <- apply(iris.sub2, 2, mean)

plot(iris1, col=index.new+1, pch=16)
points(x=mean.sub1[1], y=mean.sub1[2], col=2, pch=8)
points(x=mean.sub2[1], y=mean.sub2[2], col=3, pch=8)
```

Note that the code does not change at all. Why?

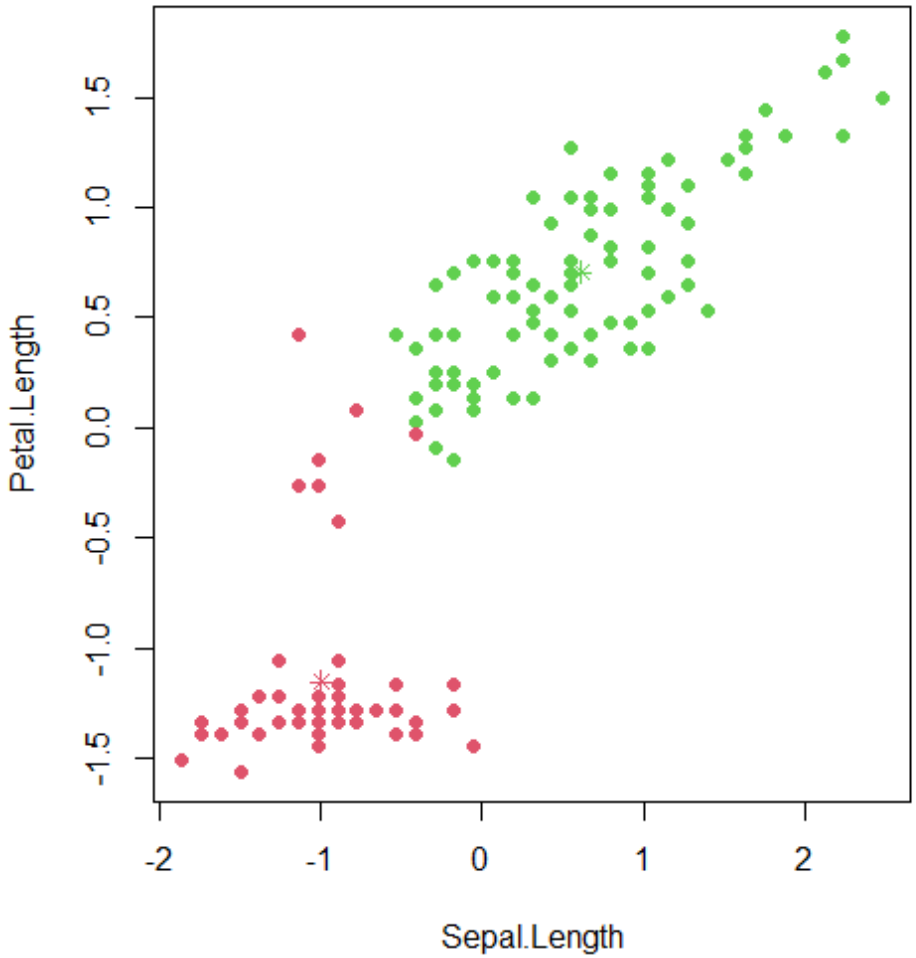
Data points are regrouped again



We can keep repeating this step, until...



We can keep repeating this step, until...



Members in each cluster do not change, which means the algorithm converges. How can we translate it into some numeric scores?

Statistics behind k-means clustering

The algorithm attempts to

- Minimize variance within clusters
- Maximize variance between clusters
- How about total variance?

Statistics behind k-means clustering

The algorithm attempts to

- Minimize variance within clusters
- Maximize variance between clusters
- How about total variance?

Instead of computing variance, we compute sum squared error (SSE).

- For a single variable X, SSE is defined as

$$SSE(X) = \sum_{i=1}^{n} (X_i - \bar{X})^2$$

Statistics behind k-means clustering

The algorithm attempts to

- Minimize variance within clusters
- Maximize variance between clusters
- How about total variance?

Instead of computing variance, we compute sum squared error (SSE).

- For a single variable X, SSE is defined as

$$SSE(X) = \sum_{i=1}^{n} (X_i - \bar{X})^2$$

- For multiple variables $\mathbf{X} = (X_1, ..., X_p)$, SSE is defined as

Statistics behind k-means clustering

The algorithm attempts to

- Minimize variance within clusters
- Maximize variance between clusters
- How about total variance?

Instead of computing variance, we compute sum squared error (SSE).

- For a single variable X, SSE is defined as

$$SSE(X) = \sum_{i=1}^{n} (X_i - \bar{X})^2$$

- For multiple variables $\mathbf{X} = (X_1, ..., X_p)$, SSE is defined as

$$SSE(\mathbf{X}) = \sum_{i=1}^{n} \sum_{j=1}^{p} (X_{ij} - \bar{X}_j)^2$$

Statistics behind k-means clustering

- Computing $SSE(\mathbf{X})$ for entire sample (all observations) gives total sum squared error (SST).

Statistics behind k-means clustering

- Computing $SSE(\mathbf{X})$ for entire sample (all observations) gives total sum squared error (SST).
- Computing $SSE(\mathbf{X})$ for each cluster gives within-group sum squared error.

Statistics behind k-means clustering

- Computing $SSE(\mathbf{X})$ for entire sample (all observations) gives total sum squared error (SST).
- Computing $SSE(\mathbf{X})$ for each cluster gives within-group sum squared error.
- Between-group sum square:

Statistics behind k-means clustering

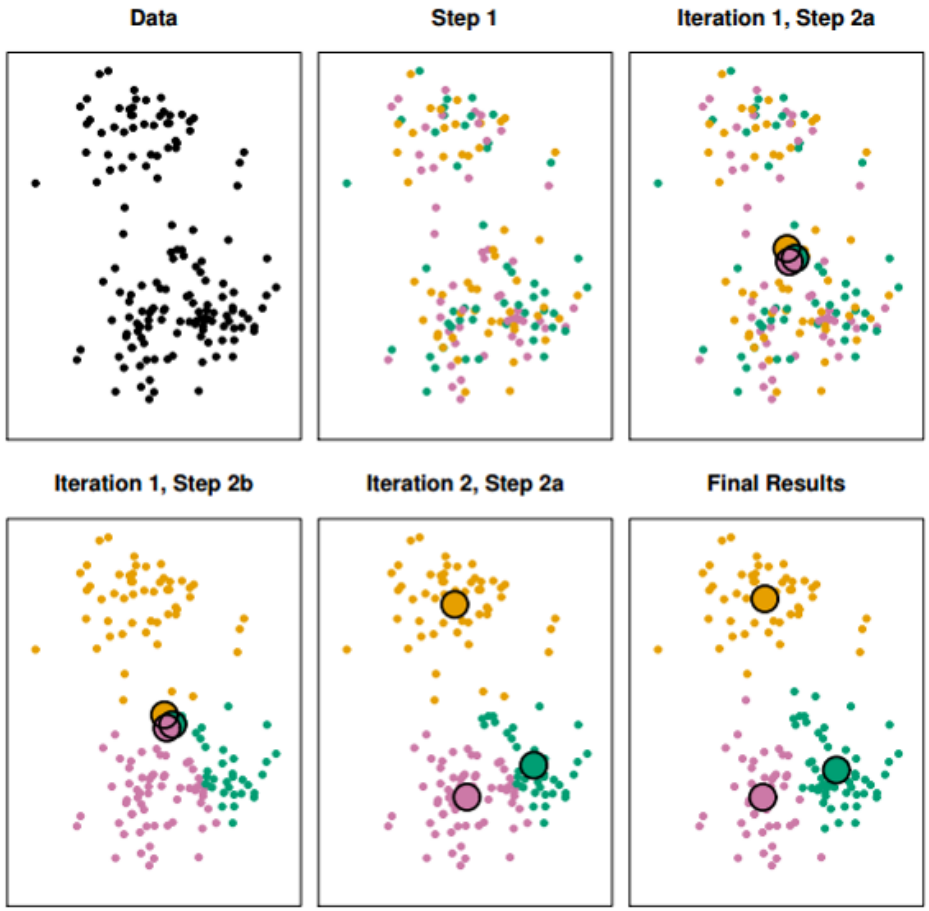
- Computing $SSE(\mathbf{X})$ for entire sample (all observations) gives total sum squared error (SST).
- Computing $SSE(\mathbf{X})$ for each cluster gives within-group sum squared error.
- Between-group sum square:
  the difference between total SS and sum of within SS.

Statistics behind k-means clustering

- Computing $SSE(\mathbf{X})$ for entire sample (all observations) gives total sum squared error (SST).
- Computing $SSE(\mathbf{X})$ for each cluster gives within-group sum squared error.
- Between-group sum square:
  the difference between total SS and sum of within SS.

- Exercise:
  Revisit the algorithm we just performed. Compute the above three measures at the end of each step. How are they changing over iterations?

Recap



- Here is a very good animation to illustrate k-means clustering algorithm. [link]

K-means algorithm

1. Randomly find k data points (observations) as the initial centers
2. For each data point, find the closest center and label it (e.g., using different colors). Now you have k clusters
3. Re-calculate the centers of current clusters
4. Repeat step 2 and 3 until the centers do not change

Supervised Learning

- Labeled data
- The goal is to predict or explain certain outcome
- Type of problem:
  - Regression: outcome is continuous
  - Classification: outcome is categorical
- Popular ML algorithms:
  - Least square, nearest neighbor, CART, gradient boosting, neural network, deep learning

Supervised Learning – Least Squares

- Linear regression model

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \epsilon$$

- The estimated model is

$$\hat{Y} = \hat{f}(\mathbf{x}) = \hat{\beta}^T \mathbf{x}$$

- Solve $\hat{\beta}$ using least square

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - \beta^T \mathbf{x}_i)^2$$

Supervised Learning – Nearest Neighbor

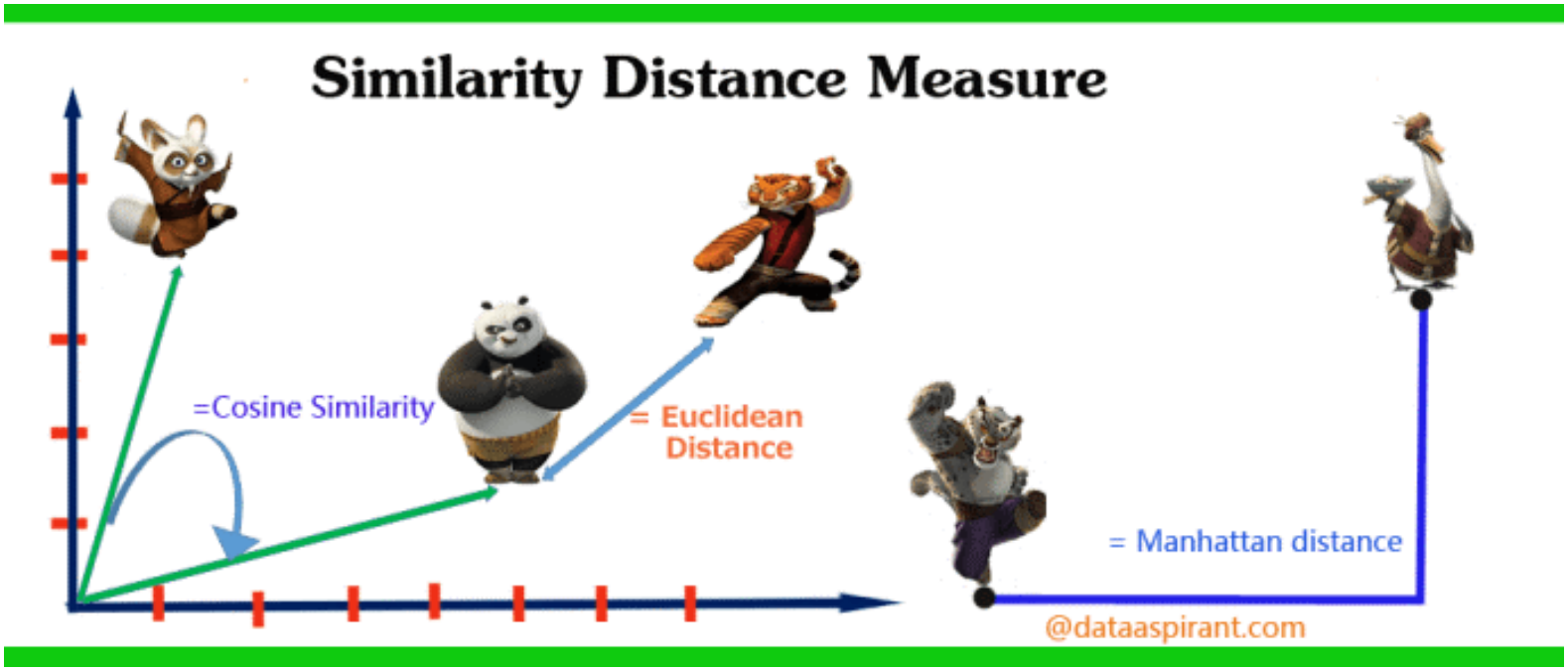
- Model:

$$\hat{Y} = \hat{f}(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} y_i$$

  where $N_k(\mathbf{x})$ is the neighborhood of $\mathbf{x}$ defined by the k closest points $\mathbf{x}_i$.

- k determines the flexibility of the model (should larger k or smaller k results in more flexible model?)
- How to define the neighbor? (How to find closest points to $\mathbf{x}_i$?) - Similarity measures

Distance-based Similarity Measures



An example[2]

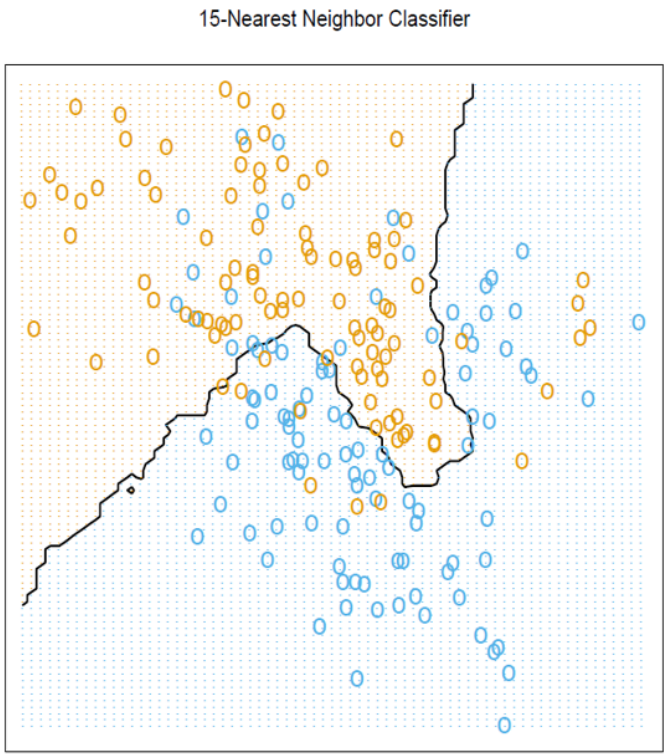


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

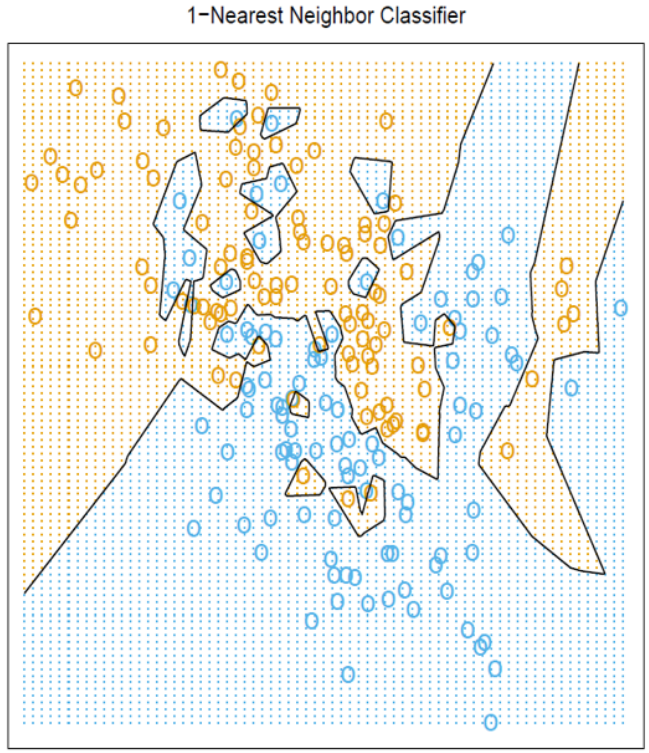


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$), *and then predicted by 1-nearest-neighbor classification.*

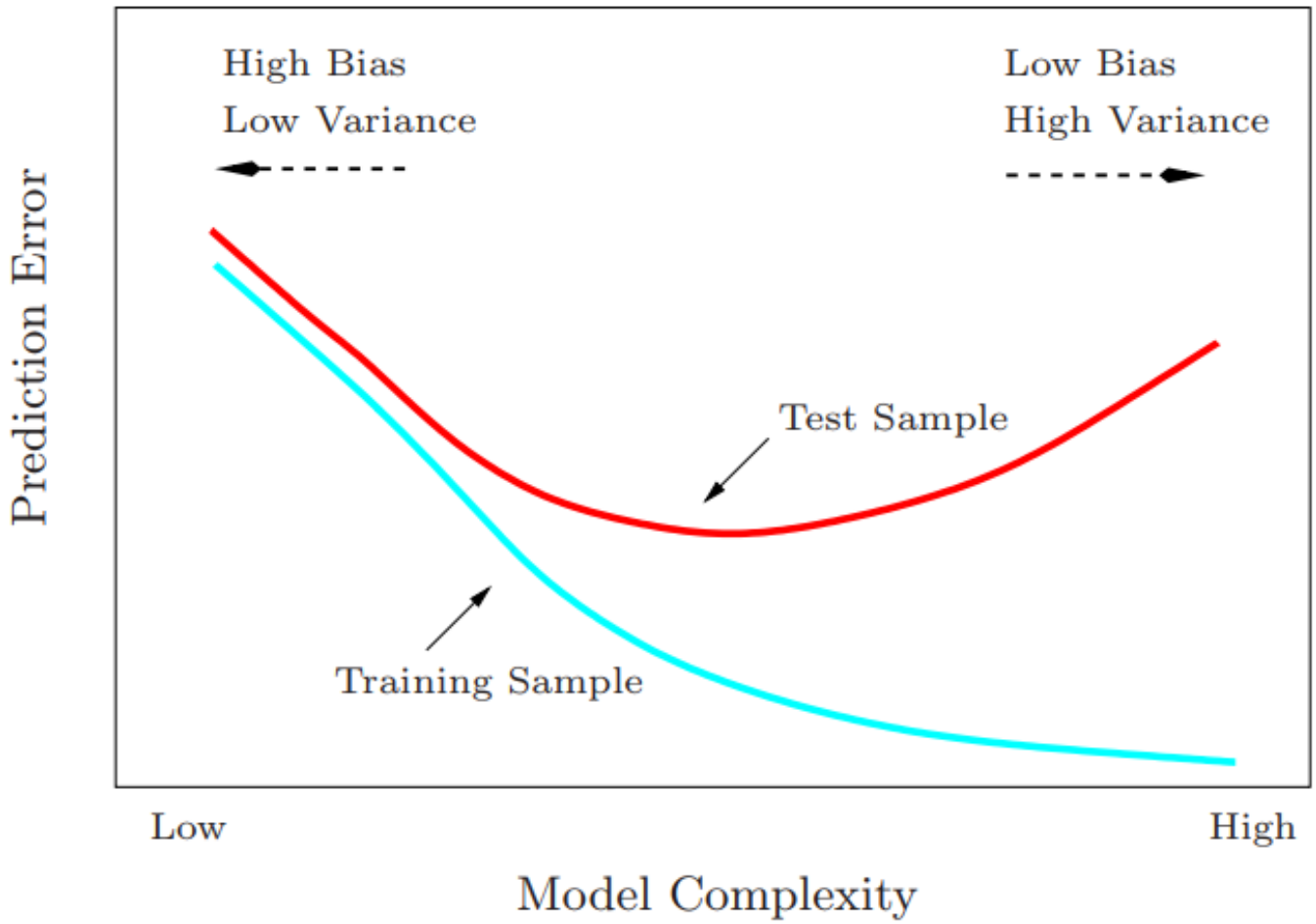
[2] ESL pg.15-16

Model Assessment (for supervised learning)

How do we know if the estimated model $\hat{f}(x)$ is **useful**?

- We never know the true $f(x)$
- Split sample to training and testing sets
- Train the model (learning algorithm) based on training sample by minimizing *training error*
- Apply the estimated model on the testing sample to calculate the *prediction (testing) error*
- We care more about testing error rather than training error

Prediction Error



Prediction Error

One of the most common metrics is the *mean squared error*

- Denote training set $Tr = \{x_i, y_i\}_1^N$, and testing set $Te = \{x_i, y_i\}_1^M$

$$\text{MSE}_{Tr} = \frac{1}{N} \sum_{i \in Tr} (y_i - \hat{f}(x_i))^2$$

$$\text{MSE}_{Te} = \frac{1}{M} \sum_{i \in Te} (y_i - \hat{f}(x_i))^2$$

- Training error, MSE_{Tr}, may be biased due to overfitting
- In this course, we denote *MSE* as training error, and *MSPE* as testing error


HOW MUCH ACCURACY DID YOU GET ?
I GOT 99%
AND WHAT WAS THE PROPORTION OF YOUR TEST SET ?
I TESTED WITH THE SAME TRAINING SET
imgflip.com

Classification Problems

- Response variable is **qualitative**
- Train a classifier $\hat{C}(x)$ that can label any new input data x
- It usually involves certain decision rule
- Prediction (testing) error: Misclassification rate (MR)

$$MR_{Te} = \frac{1}{M} \sum_{i \in Te} \mathbb{I}[y_i \neq \hat{C}(x_i)]$$

K-Fold Cross-validation

- Instead of doing training vs. testing once, we do it K times

Folder 1	Folder 2	Folder 3	Folder 4	Folder 5
Test	Train	Train	Train	Train
Train	Test	Train	Train	Train
Train	Train	Test	Train	Train
Train	Train	Train	Test	Train
Train	Train	Train	Train	Test

- Use 2,3,4,5 as training and 1 as testing
- Use 1,3,4,5 as training and 2 as testing
- Keep doing this loop...
- Average 5 testing errors, that is CV score

Leave-one-out Cross-validation

- By the name, it requires to repeat training-testing procedure n times
- However, for least square linear model, there is a short cut that makes LOOCV the same that of a single model fit

$$CV_n = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2$$

  where h_i is the diagonal element of "hat" matrix.
- In general, the estimates from LOOCV are highly correlated hence their average can have high variance
- In practice, $K = 5$ or 10 is recommended
- Exercise: write your own code to perform 10-fold CV for knn (try different k) model on the "iris" dataset.

Bias-Variance Tradeoff

- This is a very important tradeoff that governs the choice of statistical learning methods.
- **Bias**: how far the estimated model $\hat{f}(x)$ is to the true model $f(x)$.
  - Unbiased estimate is defined as: $\mathbb{E}\hat{f}(x) = f(x)$
  - Usually, we calculate the squared bias: $(\mathbb{E}\hat{f}(x) - f(x))^2$
- **Variance**: the variation of estimated model $\hat{f}(x)$ based on different training set.

Bias-Variance Tradeoff

Source: link

Bias-Variance Tradeoff

- Suppose the data arise from a model $Y = f(x) + \epsilon$, with $\mathbb{E}(\epsilon) = 0$ and $Var(\epsilon) = \sigma^2$.
- Suppose $\hat{f}(x)$ is trained based on some training data, and let (x_0, y_0) be a test observation from the same population.
- The *expected prediction error* can be decomposed to:

$$\mathbb{E}[y_0 - \hat{f}(x_0)]^2 = \sigma^2 + Bias^2(\hat{f}(x_0)) + Var(\hat{f}(x_0))$$

(Challenge yourself: Show it.)

- Typically as the **flexibility** of $\hat{f}$ increases, its variance increases, and its bias decreases.