人　工　智　慧　概　論

113-2　week 5

大綱

1. SVM
2. KNN
3. 決策樹
4. 集成學習
    - 隨機森林
    - AdaBoost

Support Vector Machine(SVM)

支持向量機 (support vector machine, SVM)

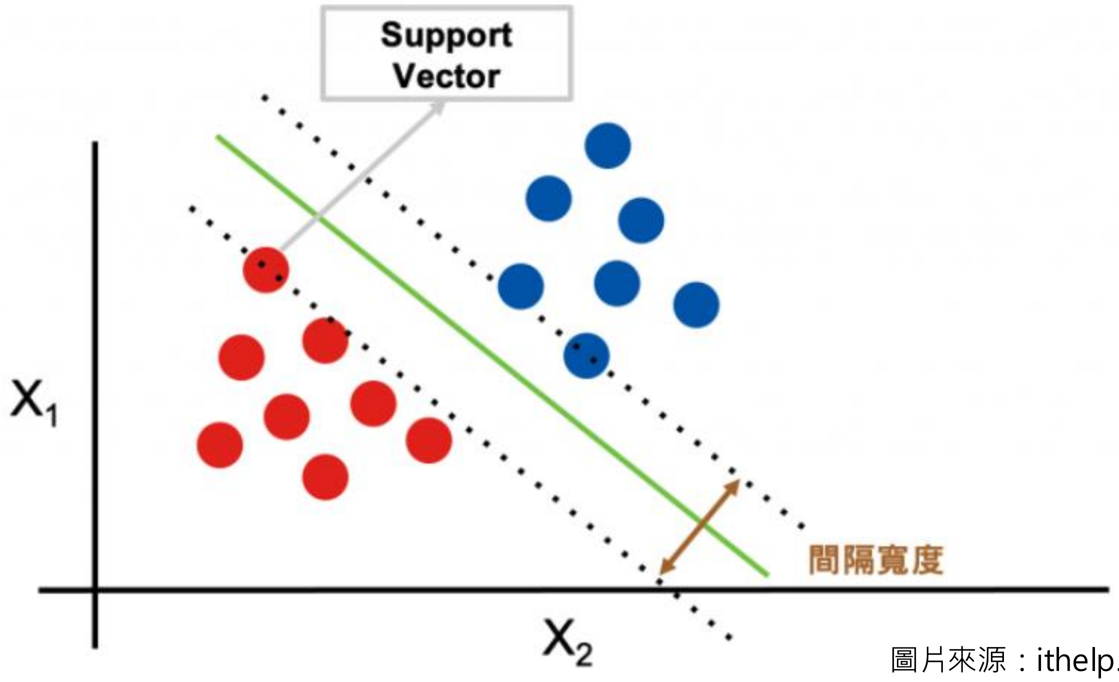
- 是一種監督式的學習方法
- 用統計風險最小化的原則來估計一個分類的超平面(hyperplane)，
- 其基礎的概念非常簡單，就是找到一個決策邊界(decision boundary)讓兩類之間的寬度(margins)最大化，使其可以完美區隔開來。
- 分成：

  線性可分支持向量機

  非線性可分支持向量機

線性可分支持向量機

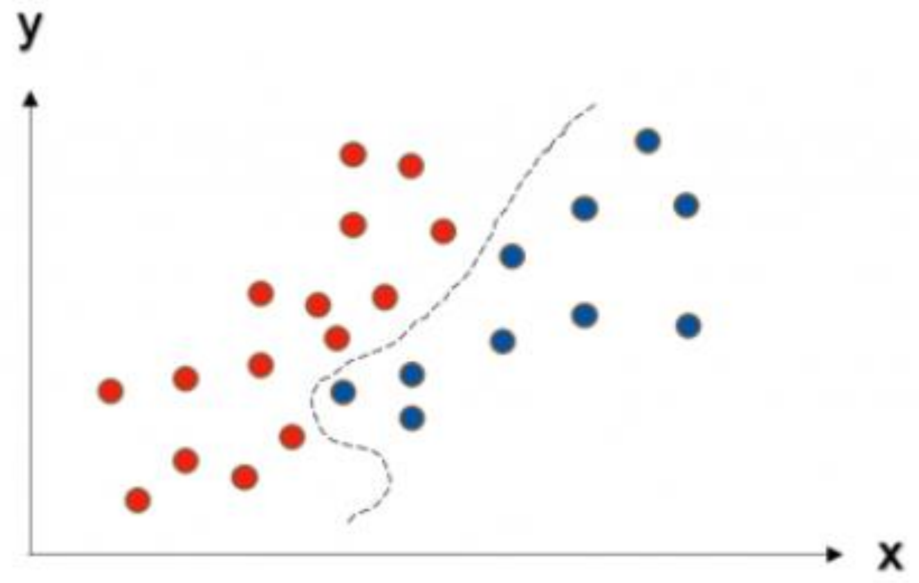
- 下圖是在二維圖形中找出一條線，目標讓這條直線與兩個類別之間的間隔寬度（margins）距離最大化。
- 其中離兩條虛線(間隔超平面)距離最近的點，就稱為支持向量 (support vector)。

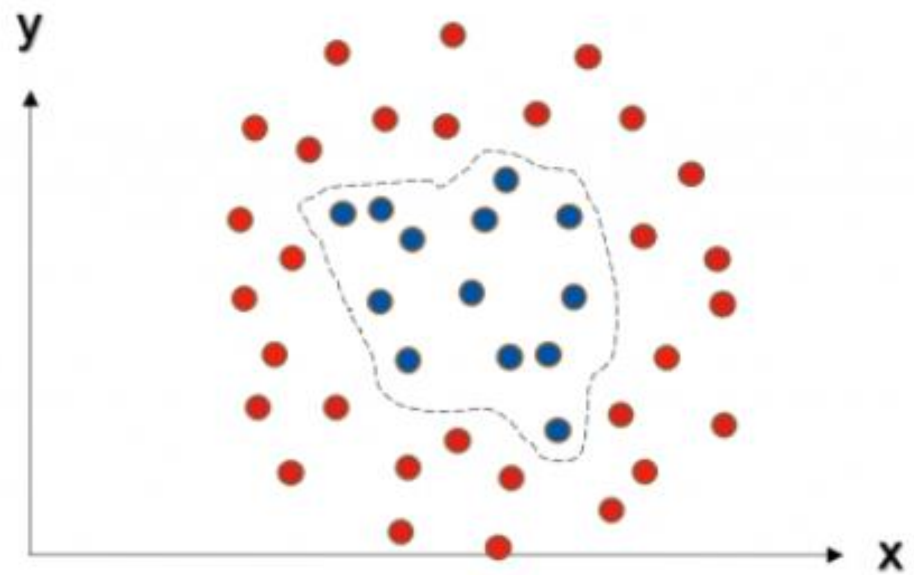


圖片來源：ithelp.ithome.com.tw/m/articles/10270447

非線性可分支持向量機

- 現實生活中的資料往往稍微複雜，那如果不是線性可分集合怎麼辦呢？
- 我們可以運用核函數(kernel function) 幫我們造出不可分的分割平面。

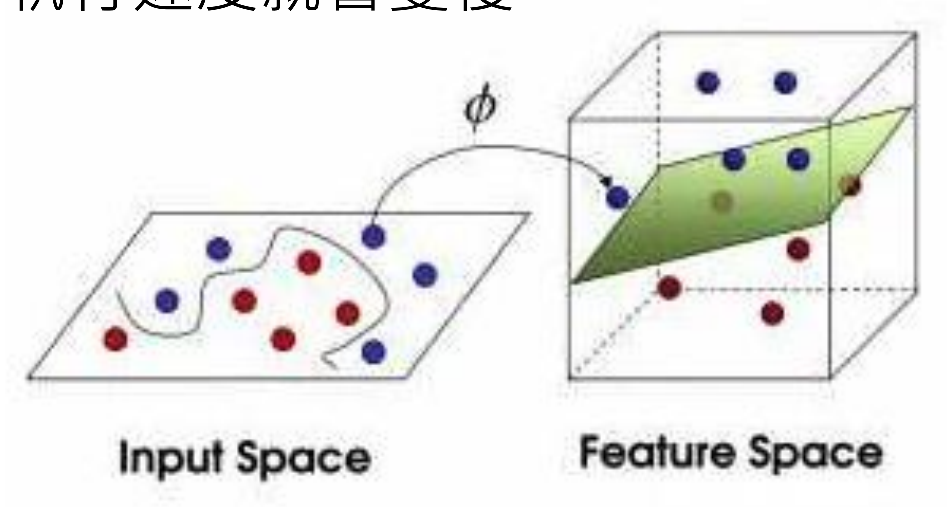




圖片來源：ithelp.ithome.com.tw/m/articles/10270447

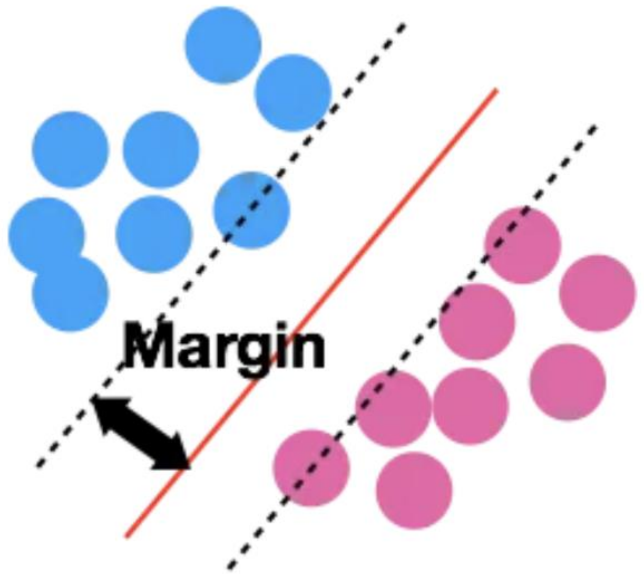
非線性可分支持向量機

- 除了進行線性分類之外 SVM 還可以使用核技巧有效地進行非線性分類
- 將其輸入的資料投到更高維度的空間，並在高維度的空間進行高維度的分類或降維。
- 透過多維度的投影技巧將原本在二維空間中不可分的點到了三維空間就可分了。
- 但是隨著資料量增加其運算也會變多，相對的執行速度就會變慢。



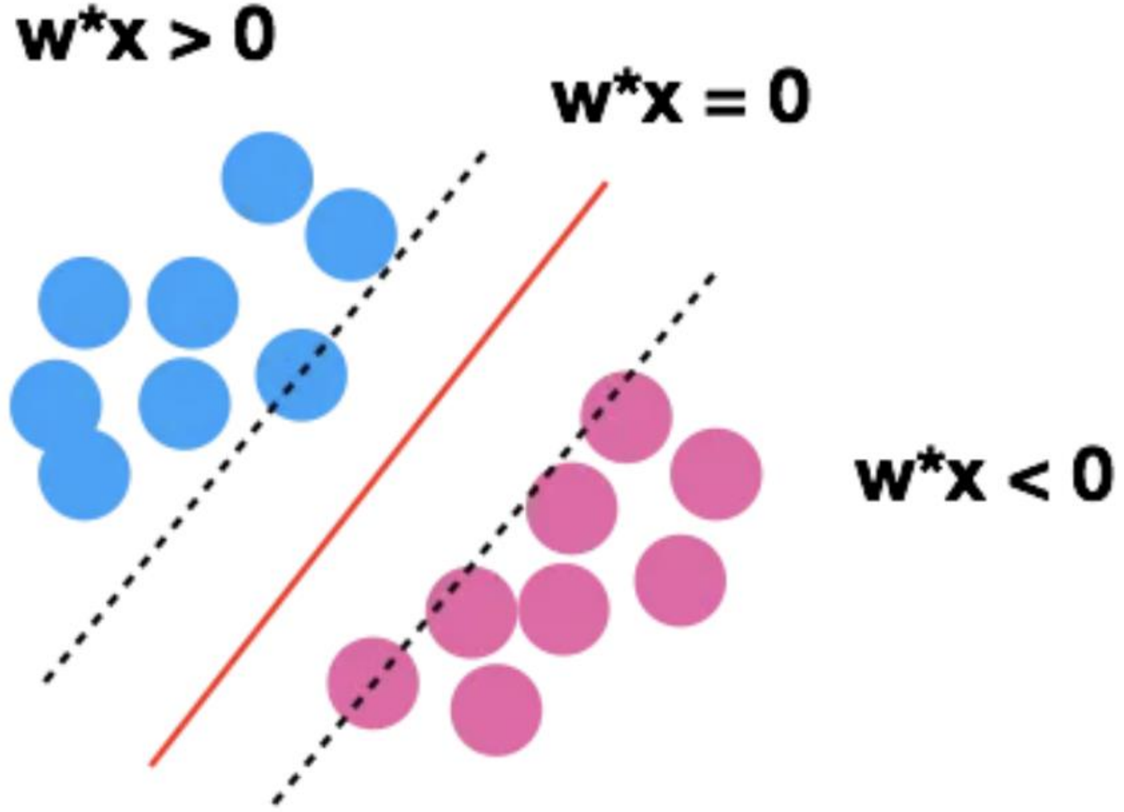
SVM是怎麼得到那條很好的線呢？

- 直線來說，首先紅色的線會創造兩條黑色平行於紅色線的虛線，並讓黑線平移碰到最近的一個點
- 紅線到黑線的距離稱為Margin，而SVM就是透過去找Margin最大的那個紅線，來找最好的線



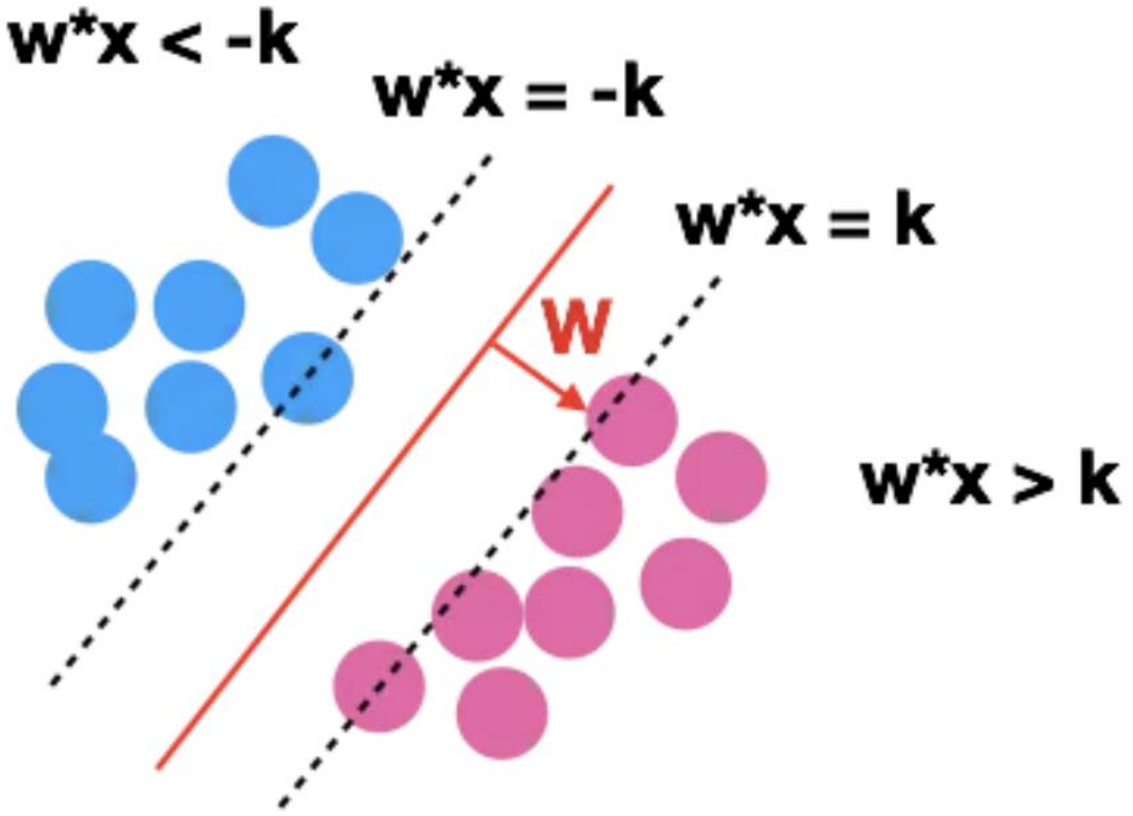
Margin最大

- 假設紅線是w*x = 0 在紅線上方的區域就是w*x >0 紅線下方的區域就是w*x <0



Margin最大

同理類推來看在左邊虛線上方的區域是w*x<-k 在右邊虛線下方的區域是w*x>k，虛線中間不會有資料點。



Margin最大化

- 虛線上的點X1,X2就是所謂的支援向量(Support vector)
- 主要是利用支援向量來算出Margin，並最大化Margin
- 那要怎麼計算margin？

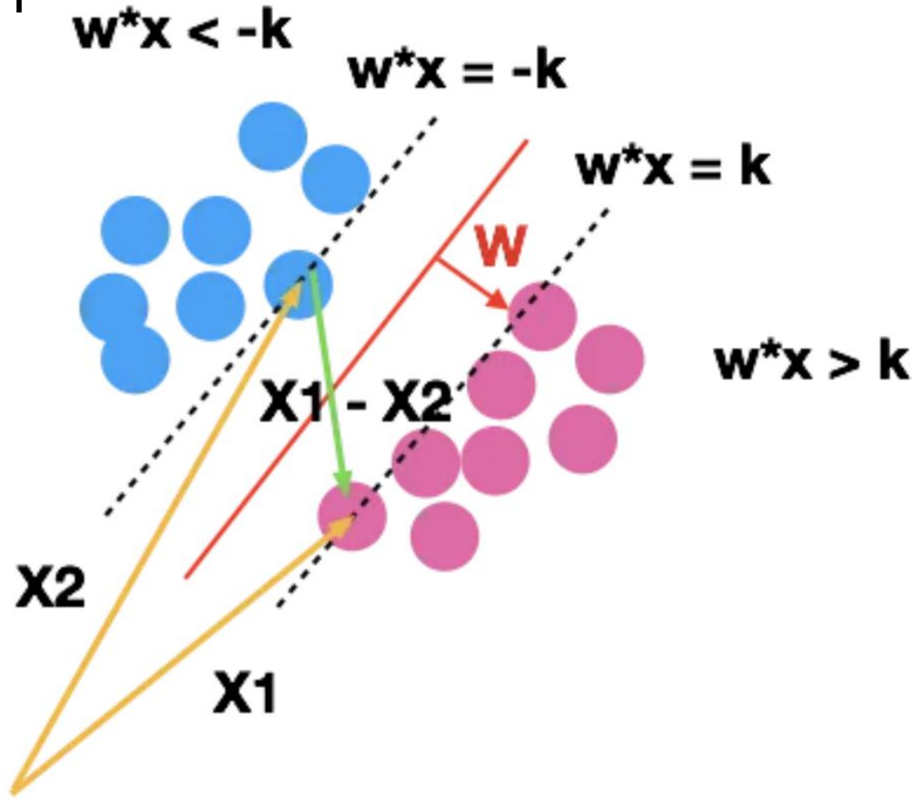
  利用X1向量-X2向量得到的向量投影到W就可以了
- 接下來在Y*(W*X) ≥k 的條件下(虛線中間沒有點)

  來最大化margin

$$\text{Margin} = \frac{W * \overrightarrow{X1 - X2}}{2\|w\|}$$

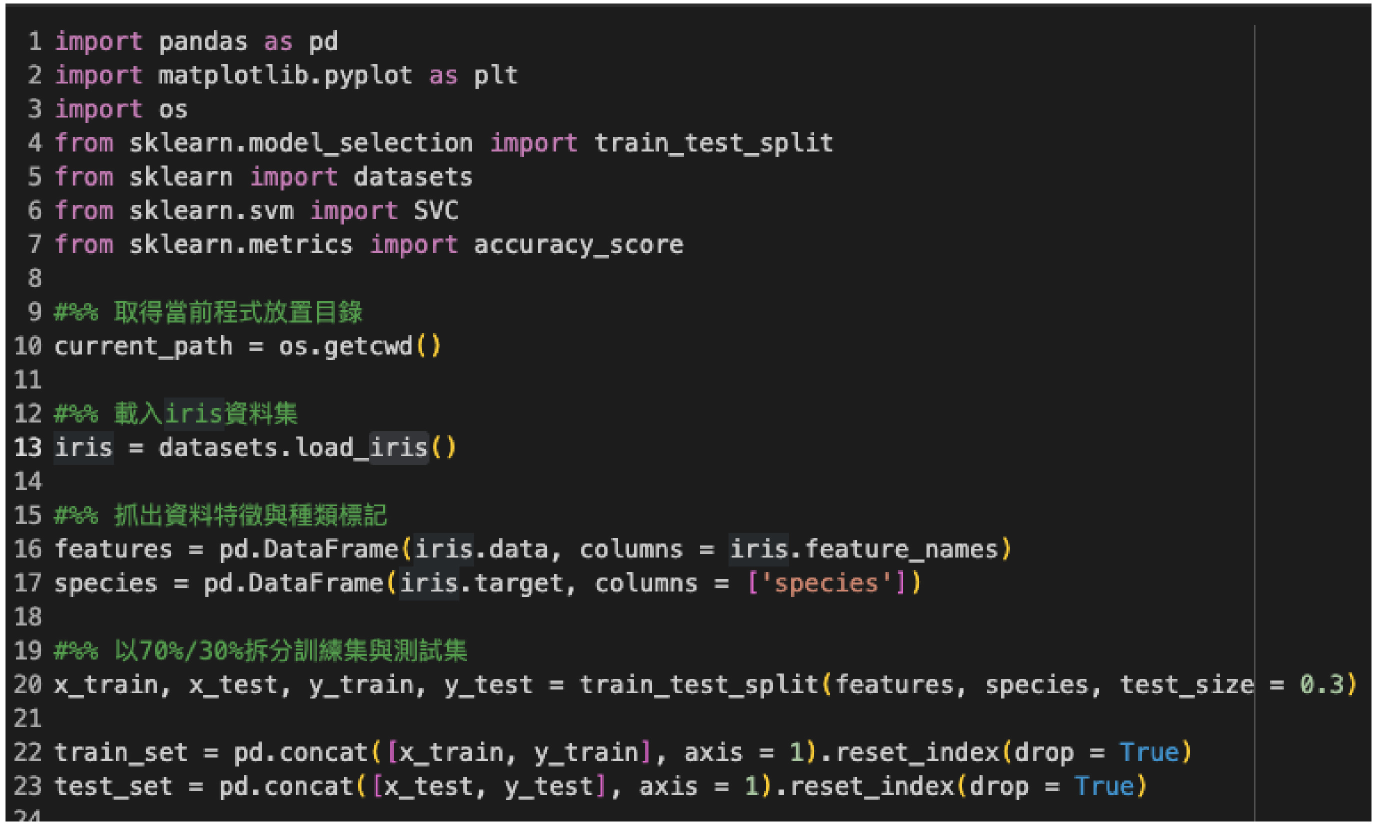
$$= \frac{2k}{2\|w\|}$$

$$\text{Condition} = y^*(w^*x) >= k$$



實作-鳶尾花分類

- 載入iris資料集
- 切分訓練集 (train set)與測試集 (test set)

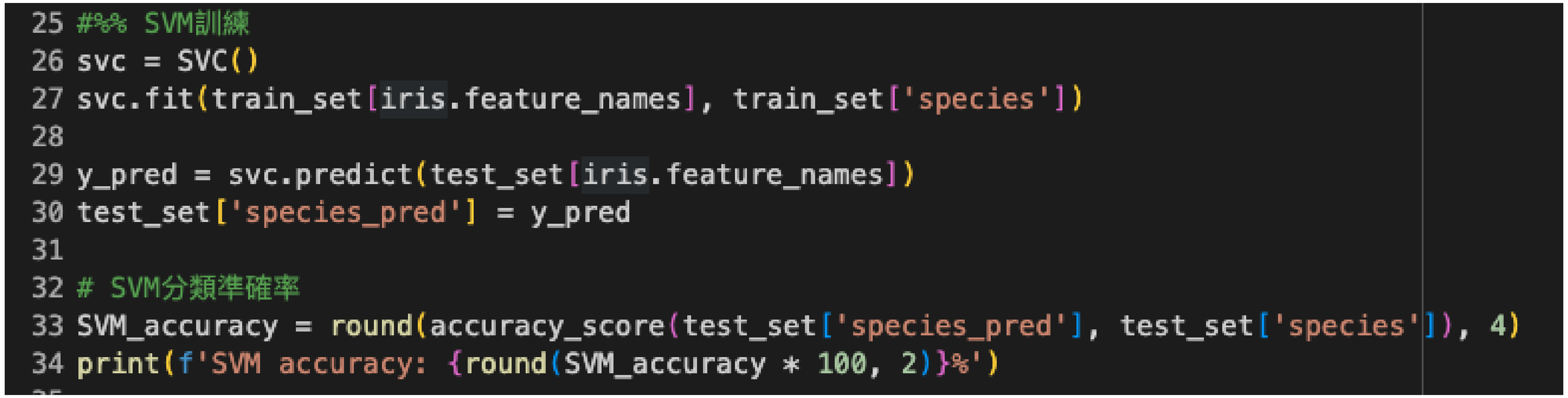


```python
import pandas as pd
import matplotlib.pyplot as plt
import os
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

#%% 取得當前程式放置目錄
current_path = os.getcwd()

#%% 載入iris資料集
iris = datasets.load_iris()

#%% 抓出資料特徵與種類標記
features = pd.DataFrame(iris.data, columns = iris.feature_names)
species = pd.DataFrame(iris.target, columns = ['species'])

#%% 以70%/30%拆分訓練集與測試集
x_train, x_test, y_train, y_test = train_test_split(features, species, test_size = 0.3)

train_set = pd.concat([x_train, y_train], axis = 1).reset_index(drop = True)
test_set = pd.concat([x_test, y_test], axis = 1).reset_index(drop = True)
```

實作-鳶尾花分類

- 以訓練集的特徵（花萼、花瓣長度與寬度）、類別（花的種類）訓練SVM模型

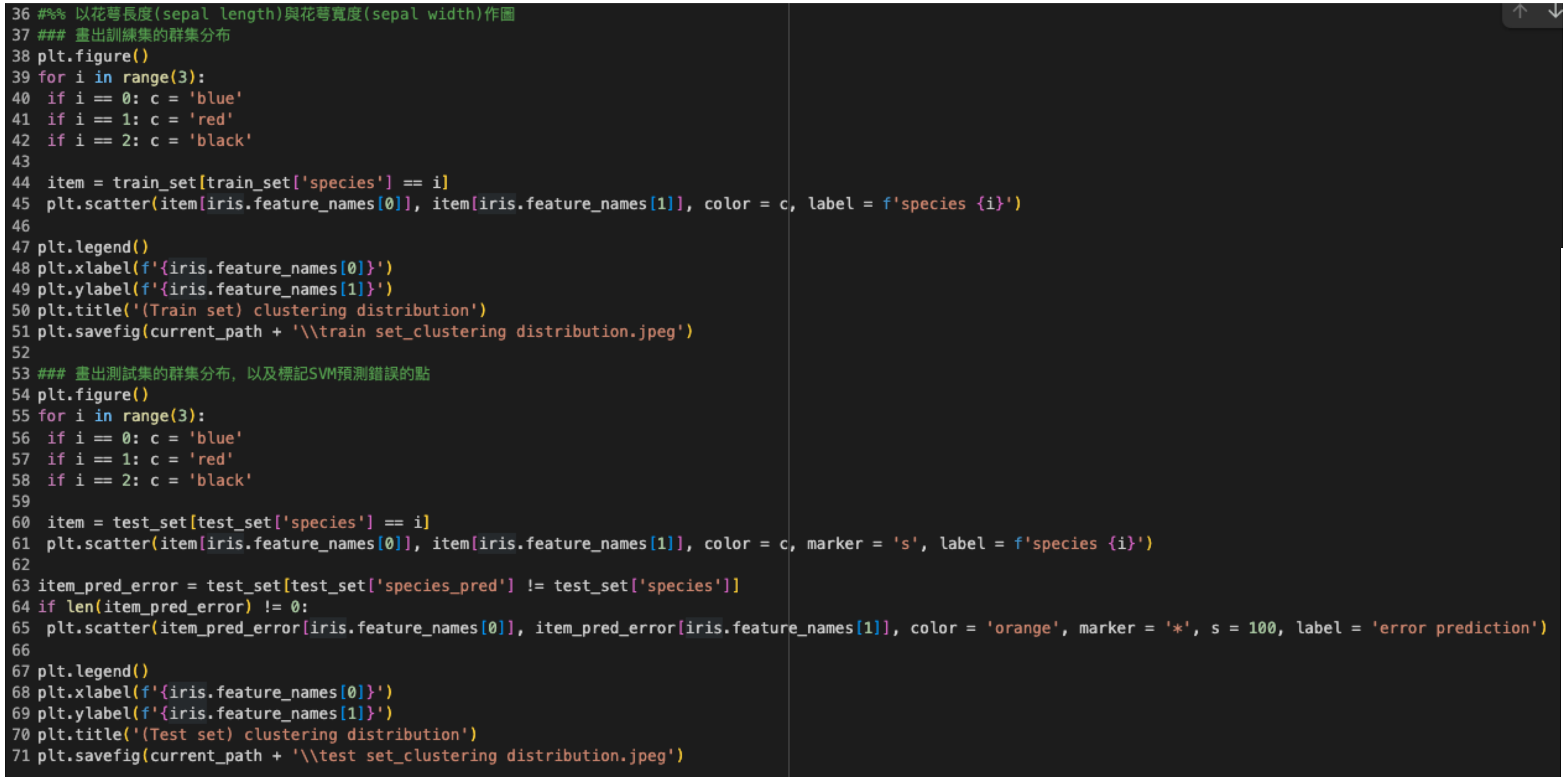
（因為是分類問題，使用Support Vector Classifier, SVC）



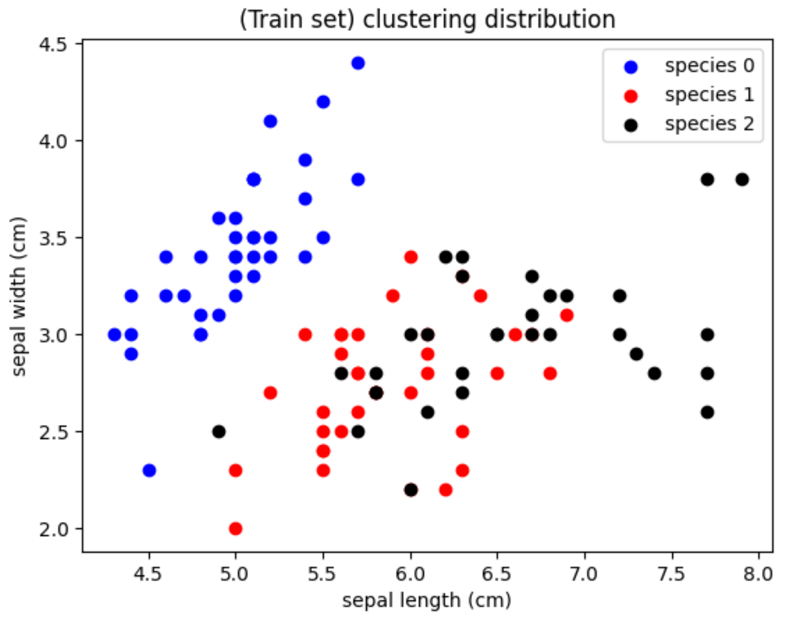
```
#%% SVM訓練
svc = SVC()
svc.fit(train_set[iris.feature_names], train_set['species'])

y_pred = svc.predict(test_set[iris.feature_names])
test_set['species_pred'] = y_pred

# SVM分類準確率
SVM_accuracy = round(accuracy_score(test_set['species_pred'], test_set['species']), 4)
print(f'SVM accuracy: {round(SVM_accuracy * 100, 2)}%')
```

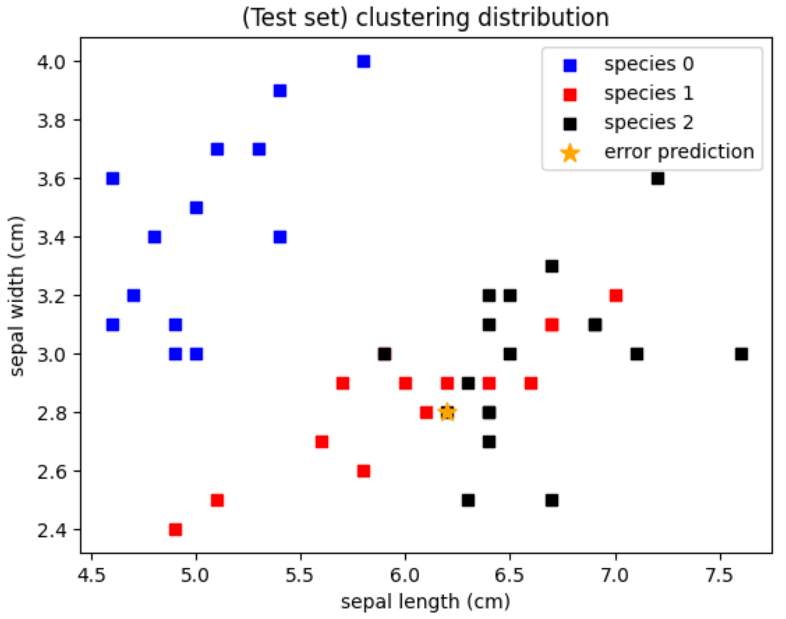
實作-鳶尾花分類

- 可視化結果



```python
#%% 以花萼長度(sepal length)與花萼寬度(sepal width)作圖
### 畫出訓練集的群集分布
plt.figure()
for i in range(3):
 if i == 0: c = 'blue'
 if i == 1: c = 'red'
 if i == 2: c = 'black'

 item = train_set[train_set['species'] == i]
 plt.scatter(item[iris.feature_names[0]], item[iris.feature_names[1]], color = c, label = f'species {i}')

plt.legend()
plt.xlabel(f'{iris.feature_names[0]}')
plt.ylabel(f'{iris.feature_names[1]}')
plt.title('(Train set) clustering distribution')
plt.savefig(current_path + '\\train set_clustering distribution.jpeg')

### 畫出測試集的群集分布，以及標記SVM預測錯誤的點
plt.figure()
for i in range(3):
 if i == 0: c = 'blue'
 if i == 1: c = 'red'
 if i == 2: c = 'black'

 item = test_set[test_set['species'] == i]
 plt.scatter(item[iris.feature_names[0]], item[iris.feature_names[1]], color = c, marker = 's', label = f'species {i}')

item_pred_error = test_set[test_set['species_pred'] != test_set['species']]
if len(item_pred_error) != 0:
 plt.scatter(item_pred_error[iris.feature_names[0]], item_pred_error[iris.feature_names[1]], color = 'orange', marker = '*', s = 100, label = 'error prediction')

plt.legend()
plt.xlabel(f'{iris.feature_names[0]}')
plt.ylabel(f'{iris.feature_names[1]}')
plt.title('(Test set) clustering distribution')
plt.savefig(current_path + '\\test set_clustering distribution.jpeg')
```





K Nearest Neighbor(KNN)

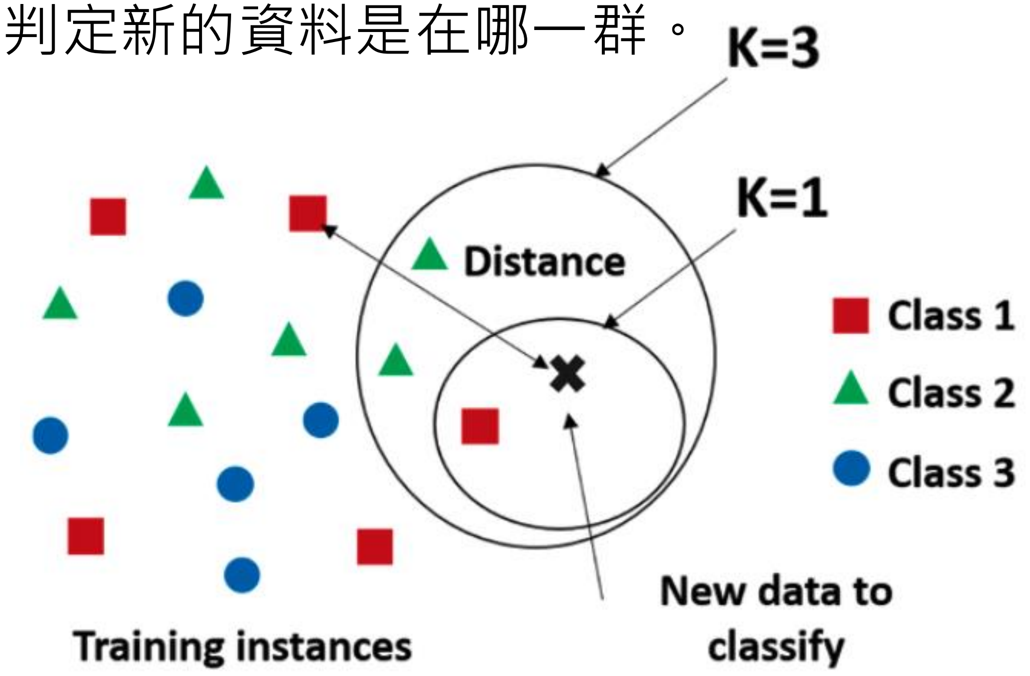
K近鄰（K Nearest Neighbors,KNN）

- 一種監督式學習的分類演算法
- KNN 演算法採多數決標準，利用 k 個最近的鄰居來判定新的資料是在哪一群。

流程：

1. 決定 k 值
2. 求每個鄰居跟自己之間的距離
3. 找出跟自己最近的 k 個鄰居，
   查看哪一組鄰居數量最多，就加入哪一組

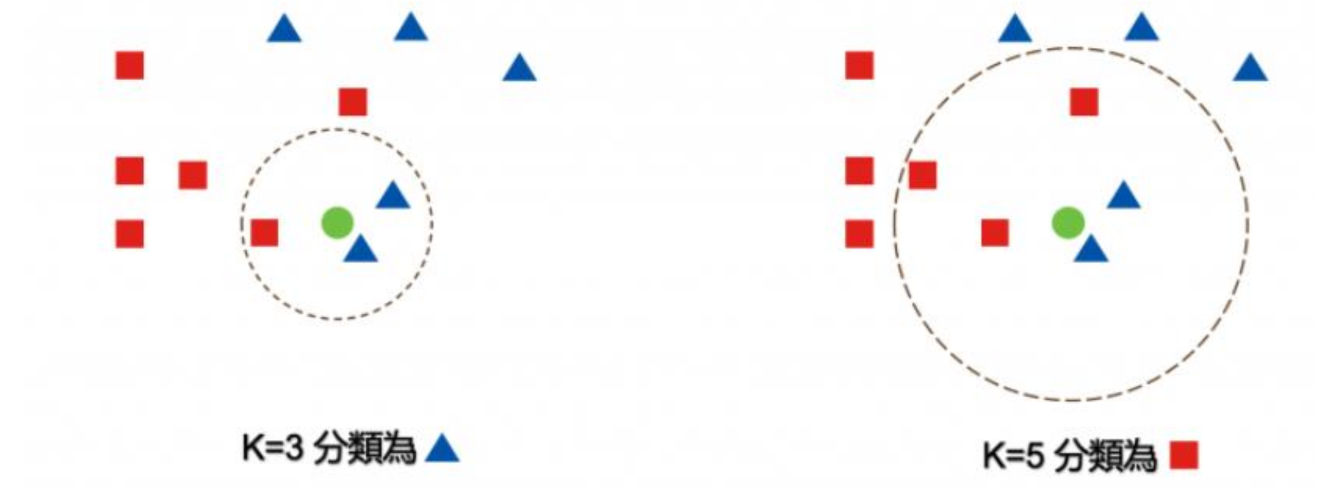
（如果還是沒辦法決定在哪一組，回到第一步調整 k 值，再繼續）



圖片來源：Human Activities and Postures Recognition: From Inertial Measurements to Quaternion-Based Approaches

K近鄰（K Nearest Neighbors,KNN）

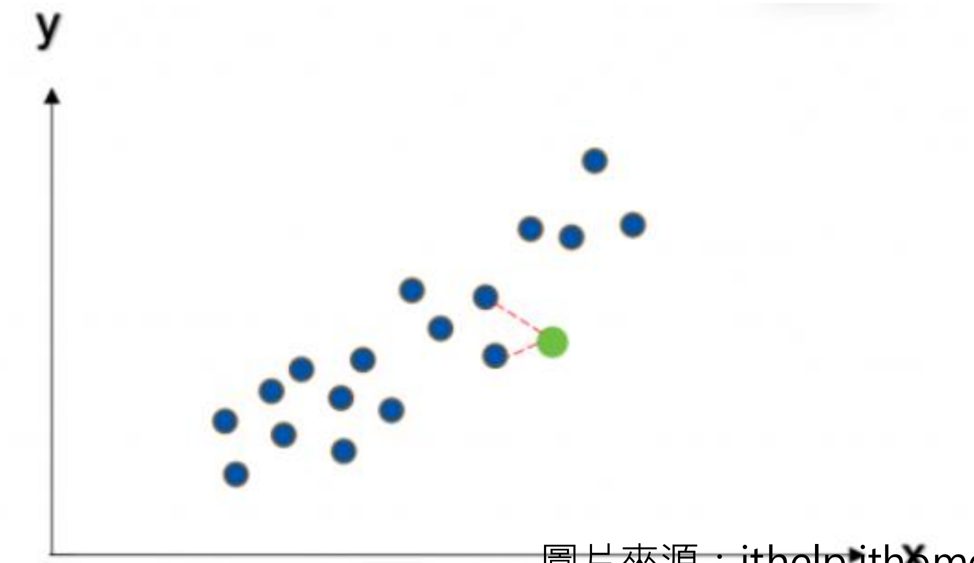
- k 的大小會影響模型最終的分類結果。
- 以下圖為例，假設綠色點是新的資料
- 當k等於3，會搜尋離綠色點最近的鄰居，我們可以發現藍色三角形為預測的結果。
- 當k設為5，結果又不一樣了，我們發現距離最近的三個鄰居為紅色正方形。



圖片來源：ithelp.ithome.com.tw/m/articles/10269826

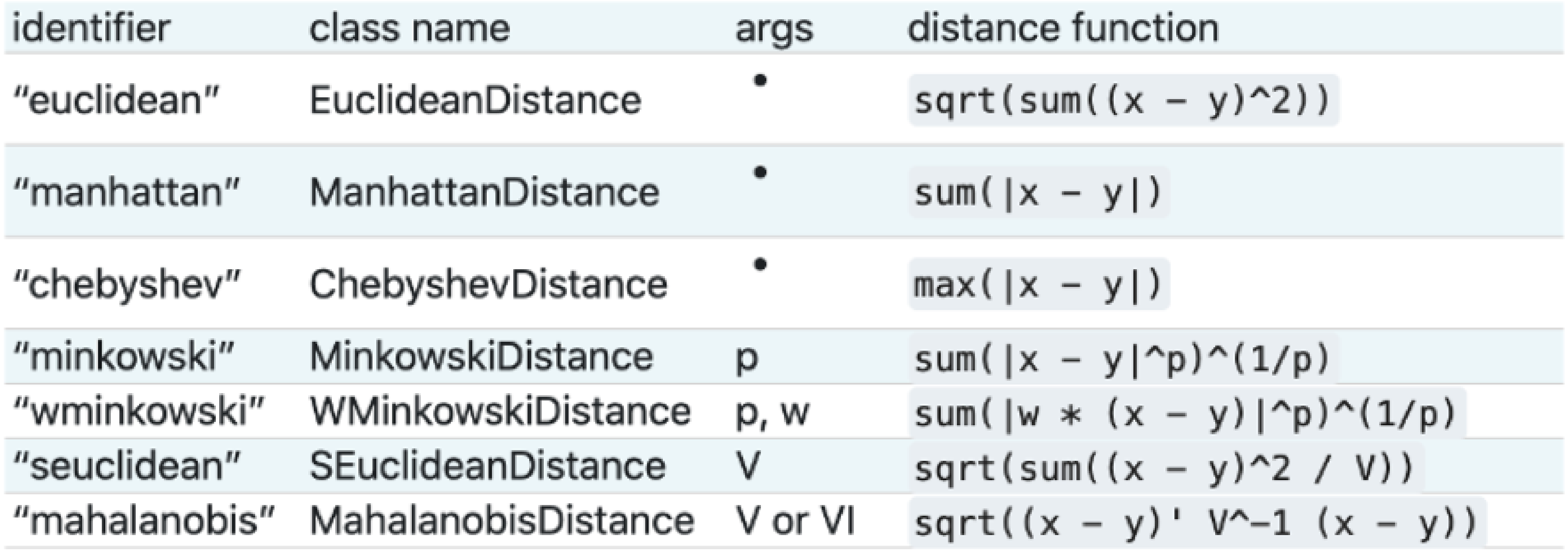
KNN 迴歸器

- KNN 同時也能運用在迴歸問題上面。
- 迴歸模型輸出的結果是一個連續性數值，預測該值是 k 個最近鄰居輸出的平均值。
- 以下圖為例當 k=2 時，假設我們有一個輸入特徵 x 要預測的輸出為 y。
- 當有一筆新的 x 進來的時候， KNN 迴歸器會尋找鄰近 2 個 x 的輸出做平均當作是該筆資料的預測結果。



圖片來源：ithelp.ithome.com.tw/m/articles/10269826

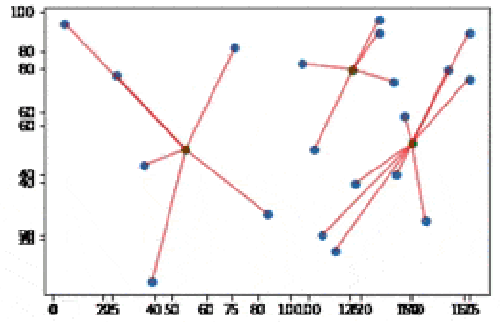
KNN 度量距離的方法

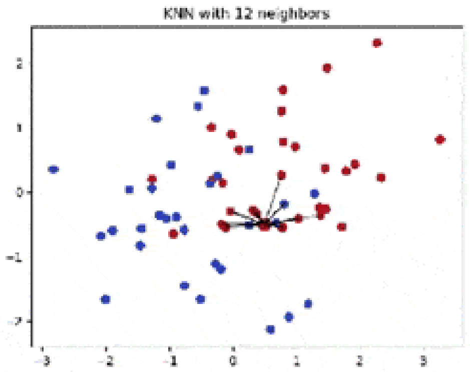


identifier	class name	args	distance function
"euclidean"	EuclideanDistance	•	`sqrt(sum((x - y)^2))`
"manhattan"	ManhattanDistance	•	`sum(|x - y|)`
"chebyshev"	ChebyshevDistance	•	`max(|x - y|)`
"minkowski"	MinkowskiDistance	p	`sum(|x - y|^p)^(1/p)`
"wminkowski"	WMinkowskiDistance	p, w	`sum(|w * (x - y)|^p)^(1/p)`
"seuclidean"	SEuclideanDistance	V	`sqrt(sum((x - y)^2 / V))`
"mahalanobis"	MahalanobisDistance	V or VI	`sqrt((x - y)' V^-1 (x - y))`

KNN 與 k-means 勿混淆

- KNN 的缺點是對資料的局部結構非常敏感，因此調整適當的 k 值極為重要
- 大家很常將 KNN 與 K-means 混淆，雖然兩者都有 k 值要設定但其實兩者無任何關聯。
- KNN 的 k 是設定鄰居的數量採多數決作為輸出的依據。
- K-means 的 k 是設定集群的類別中心點數量。


K-means



KNN

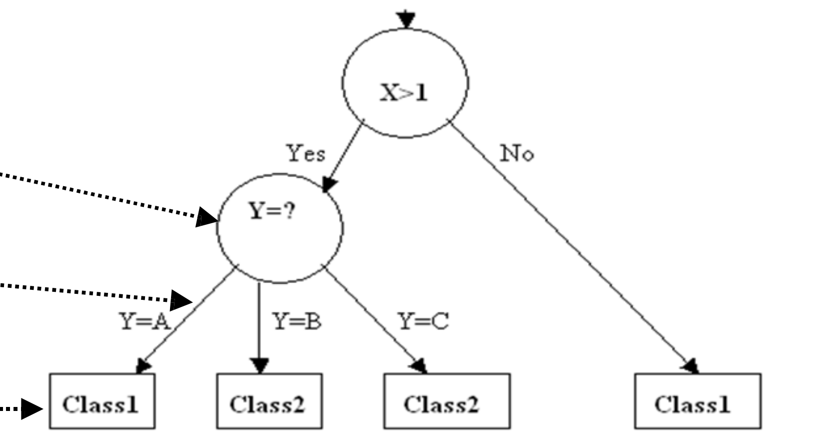
圖片來源：ithelp.ithome.com.tw/m/articles/10269826

決　策　樹

Decision Trees

- Efficient method for producing **classifiers** from data - (*logic method*).
- **Supervised learning methods** that construct decision trees from a set of input-output samples.
- It guarantees that a **simple**, but not necessary the simplest, tree is found.

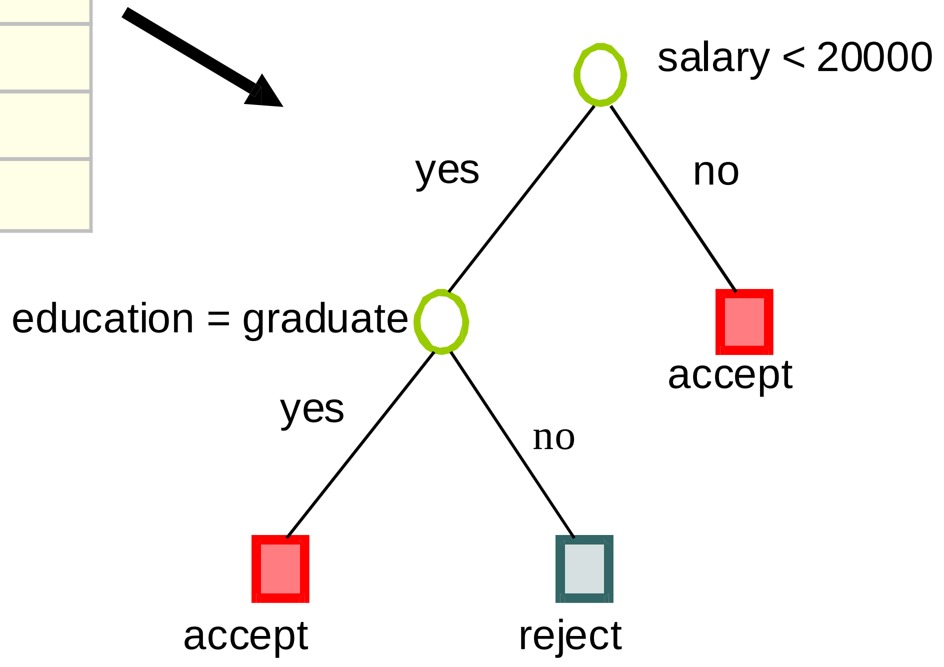
- A decision tree consists of ***nodes*** that are **tests** on the attributes.
- The outgoing ***branches*** of a node correspond to all the possible **outcomes** of the test at the node.
- ***Leafs*** are sets of samples belong to the same class



Decision Trees

Credit Analysis

salary	education	label
10000	high school	reject
40000	under graduate	accept
15000	under graduate	reject
75000	graduate	accept
18000	graduate	accept

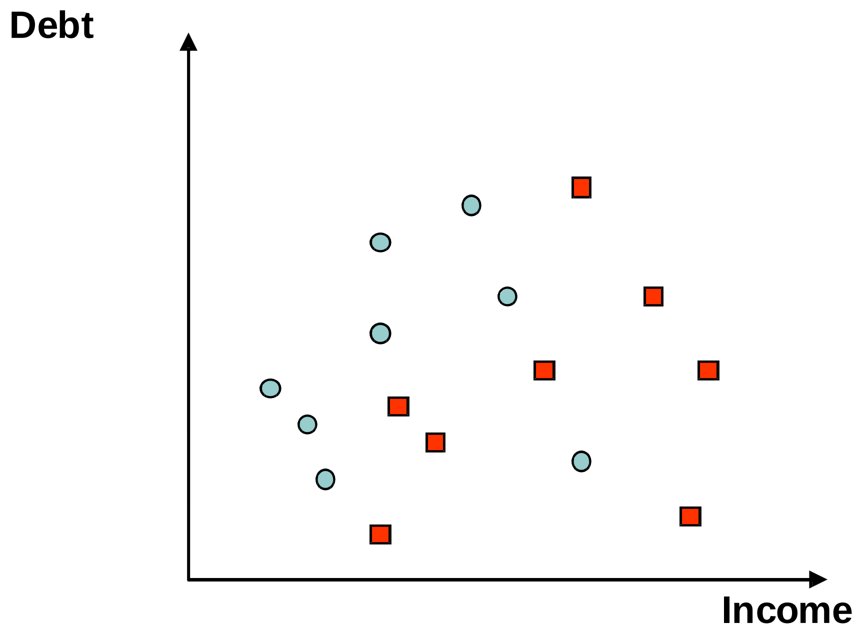


Classification by Decision Tree Induction

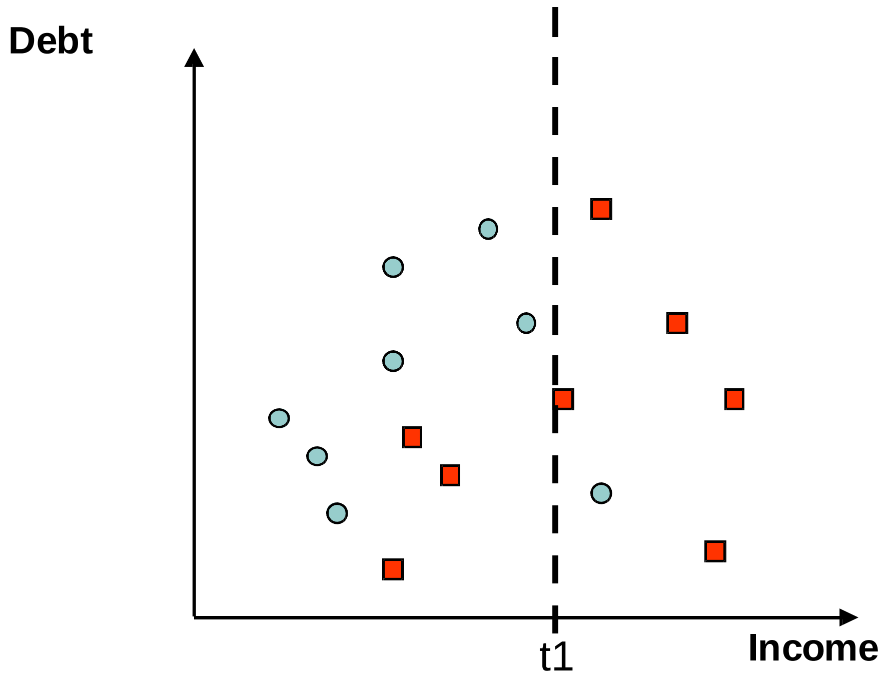
1. Decision tree **generation** consists of two phases:
   - *Tree construction*:
     - At start, all the training examples are at the root.
     - Partition examples recursively based on selected attributes.
   - *Tree pruning:*
     - Identify and remove branches that reflect noise or outliers.

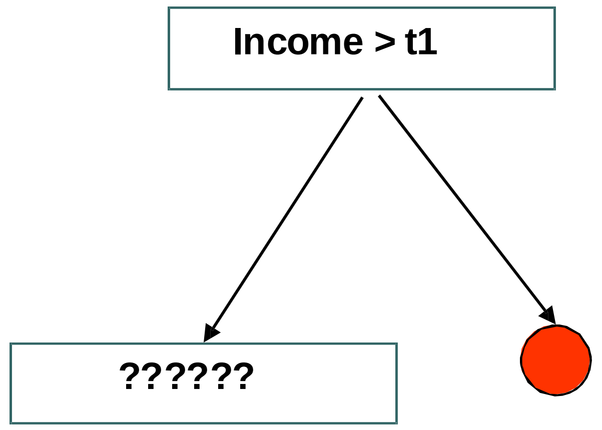
2. **Use** of decision tree: Classifying an unknown sample
   - Test the attribute values of the sample against the decision tree

Decision Tree: Example

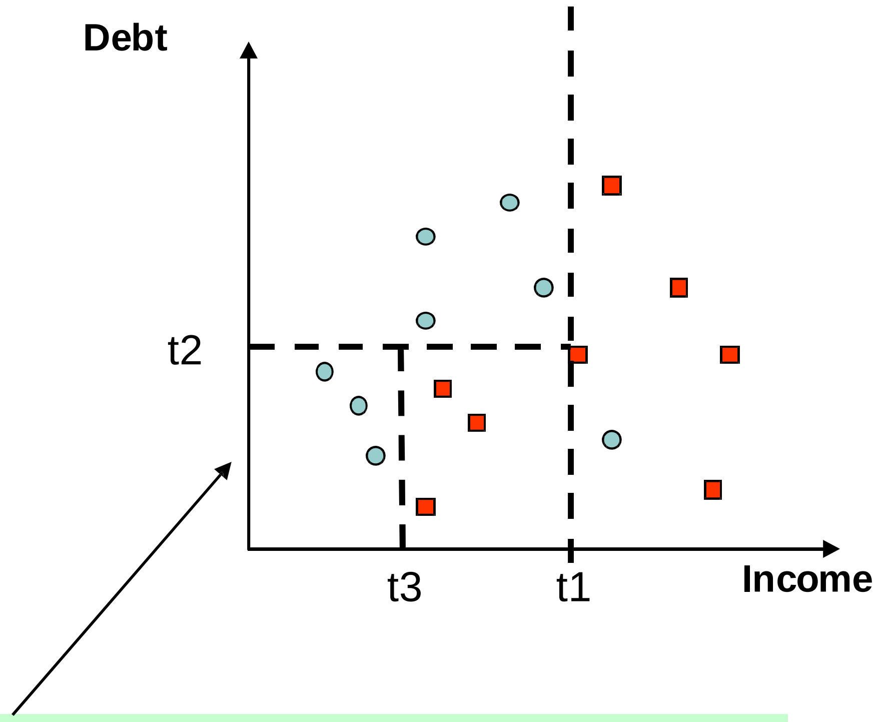


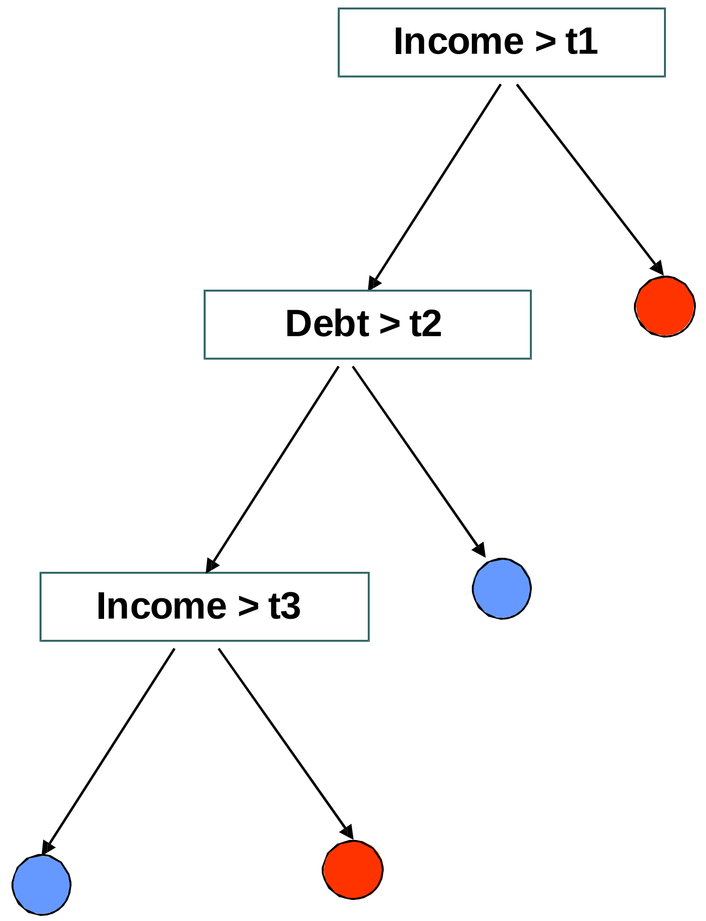
Decision Tree: Example





Decision Tree: Example



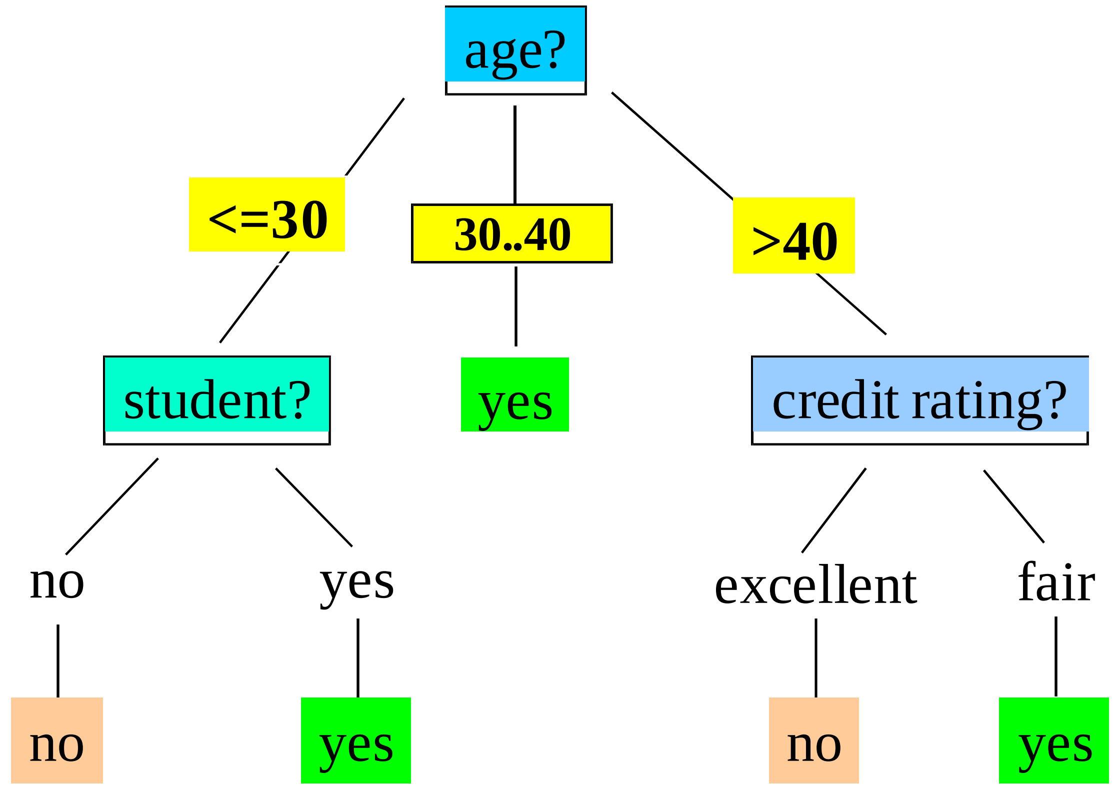


Note: tree boundaries are piecewise linear and axis-parallel

Training Dataset

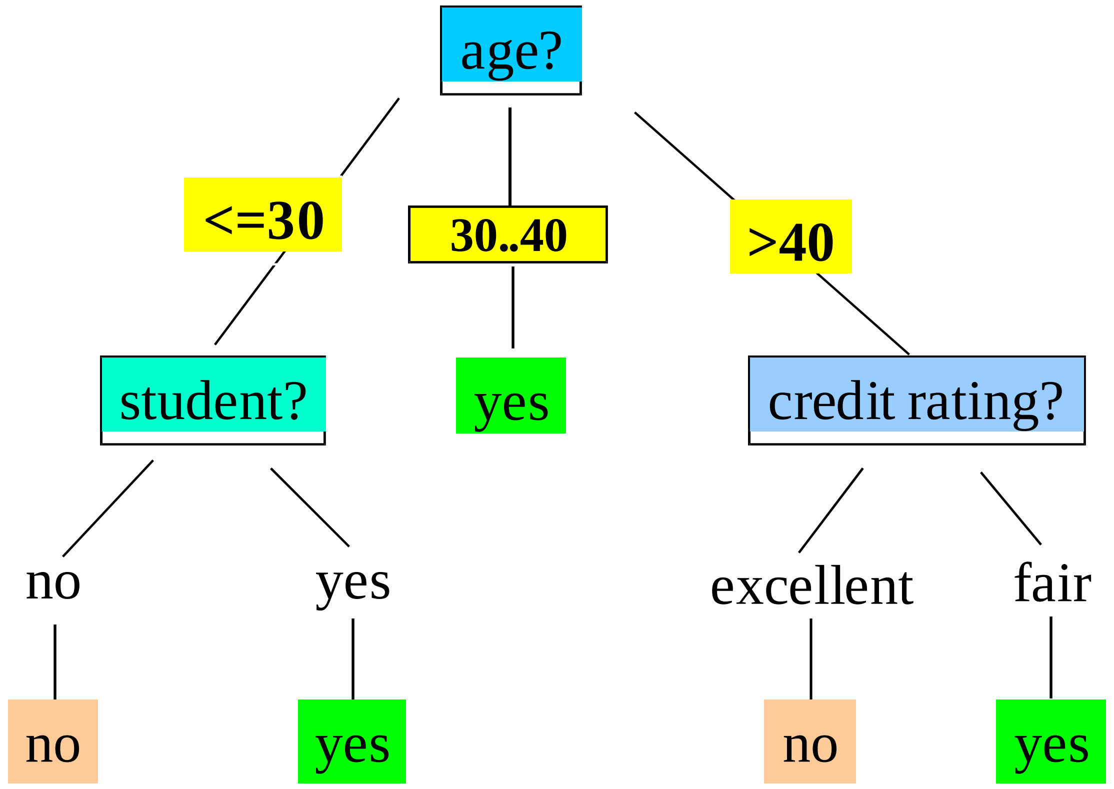
age	income	student	credit_rating	Buy-computer
<=30	high	no	fair	No
<=30	high	no	excellent	No
31…40	high	no	fair	Yes
>40	medium	no	fair	Yes
>40	low	yes	fair	Yes
>40	low	yes	excellent	No
31…40	low	yes	excellent	Yes
<=30	medium	no	fair	No
<=30	low	yes	fair	Yes
>40	medium	yes	fair	Yes
<=30	medium	yes	excellent	Yes
31…40	medium	no	excellent	Yes
31…40	high	yes	fair	Yes
>40	medium	no	excellent	No

Training Output: A Decision Tree for “buys_computer”



age	income	student	credit_rating	
<=30	high	no	fair	No
<=30	high	no	excellent	No
31…40	high	no	fair	Yes
>40	medium	no	fair	Yes
>40	low	yes	fair	Yes
>40	low	yes	excellent	No
31…40	low	yes	excellent	Yes
<=30	medium	no	fair	No
<=30	low	yes	fair	Yes
>40	medium	yes	fair	Yes
<=30	medium	yes	excellent	Yes
31…40	medium	no	excellent	Yes
31…40	high	yes	fair	Yes
>40	medium	no	excellent	No

Testing A Decision Tree for "buys_computer"



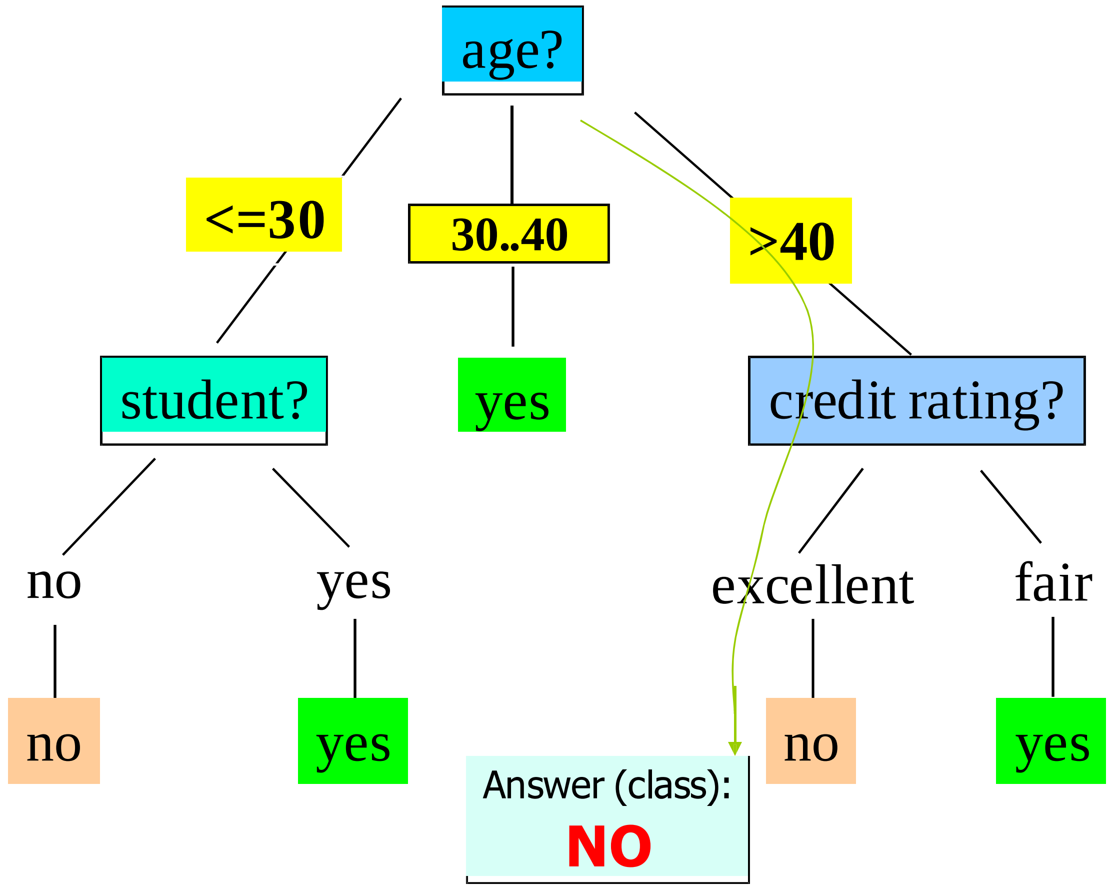
Classify new sample:

{ **Age = 45** and **income = low** and **student = no** and **credit = excellent** }

==================

Buy computer = **?**

Output: A Decision Tree for “buys_computer”



Classify new sample:

{ **Age = 45** and **income = low** and **student = no** and **credit = excellent** }

==================

Buy computer = **?**

Decision Tree Algorithms

Requirements:

1. Attribute-value description
2. Predefined classes
3. Discrete classes
4. Sufficient data
5. “Logical” classification models

Decision Trees

Pros

- Fast execution time
- Generated trees (rules) are easy to interpret by humans
- Scale well for large data sets
- Can handle high dimensional data

Cons

- Cannot capture correlations among attributes
- Consider only axis-parallel cuts

Decision Tree Algorithms

- Classifiers from machine learning and statistical community:
  - ID3
  - C4.5[Quinlan 93] $\longrightarrow$ C5.0
  - CART

Algorithm for Decision Tree Induction

- **Basic algorithm** (*a greedy algorithm*):
  - Tree is constructed in a **top-down** recursive divide-and-conquer manner.
  - At start, all the training examples are at the root.
  - Attributes are categorical (if continuous-valued, they are discretized in advance).
  - Examples are partitioned recursively based on selected attributes.
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain).

- ***Conditions for stopping partitioning:***
  - All samples for a given node belong to the same class.
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf.
  - There are no samples left.

Attribute Selection Measure

- **Information gain** (ID3/**C4.5**)
  - All attributes are assumed to be categorical.
  - Can be modified for continuous-valued attributes.

- **Gini index** (IBM IntelligentMiner, CART, …)
  - All attributes are assumed continuous-valued.
  - Assume there exist several possible split values for each attribute.
  - Can be modified for categorical attributes.

ID3

資訊理論 (information theory)

- 假設一個事件有*n*種結果，發生的機率分別為*P(v1), …, P(v n)*，這些機率都是已知的，則定義這個事件發生後所得到的資訊量為 ：

$$I(P(v_1),\ldots,P(v_n)) = \sum_{i=1}^{n} -P(v_i)\log_2 P(v_i)$$

- 各種結果發生機率愈平均，所求資訊量也愈大
- 資訊量可以當作亂度 (Entropy) 的指標，資訊量愈大，表示亂度愈大
- 解決屬性選擇的問題

亂度(Entropy)

- 昆蘭 (Quinlan)(1979)提出，以雪南 (Shannon) (1949)的資訊理論 (Information theory)為依據。
- 資訊理論:若一事件有k種結果,對應的機率為P_i。則此事件發生後所得到的資訊量I(視為Entropy)為：

  $$I=-(p_1*\log_2(p_1)+ p_2*\log_2(p_2)+\ldots+ p_k*\log_2(p_k))$$

  - Example 1: 設 k=4→ $p_1=0.25,p_2=0.25,p_3=0.25,p_4=0.25$

    $I=-(.25*\log_2(.25)*4)=2$
  - Example 2: 設 k=4→ $p_1=0$, $p_2=0.5$, $p_3=0$, $p_4=0.5$

    $I=-(.5*\log_2(.5)*2)=1$
  - Example 3: 設 k=4→ $p_1=1$, $p_2=0$, $p_3=0$, $p_4=0$

    $I=-(1*\log_2(1))=0$
- 用熵(Entropy) 衡量資料的一致性

ID3 演算法(C4.5,C5.0)

- 資訊獲利(Information Gain)

若分類標記(Y)分為(成功、失敗)兩種，X為預測變項(類別屬性;k類)，n為總樣本數(n_1為總樣本數中具成功標記的個數)，經由X變項將樣本分類後m_i為X=i類中的總樣本個數(m_{i1}為X=i類中具成功標記的個數)。根據變項X將n個樣本分為$m_1,m_2,\ldots,m_k$的資訊獲利為:

$$\mathit{Gain(X)} = I(n,n_1)-E(X),$$

其中

$$I(n,n_1)=-((n_1/n)\log_2(n_1/n)+(1-n_1/n)\log_2(1-n_1/n))$$

$$E(X)=(m_1/n)*I(m_1,m_{11})+(m_2/n)*I(m_2,m_{21})+\ldots(m_k/n)*I(m_k,m_{k1})$$

利用資訊獲利做屬性選取

- 資訊獲利=
- 測試前的資訊量 - 測試後的資訊量

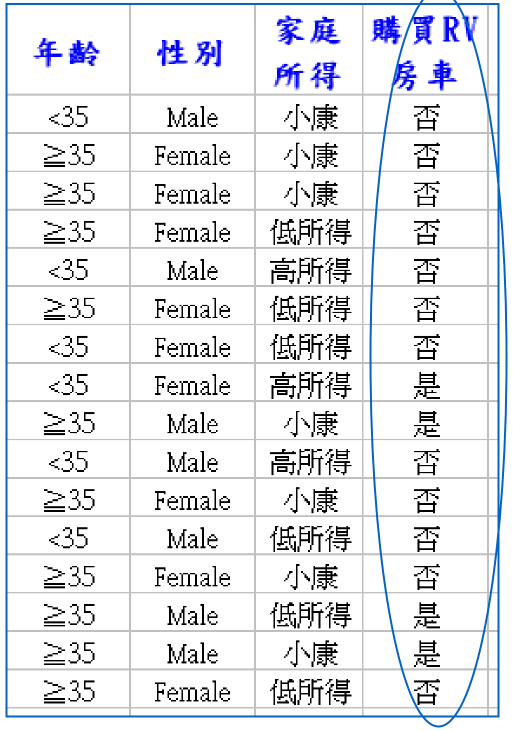
- 決策樹分類的目的
  - 訓練樣本分成亂度最小的子集合
  - 所有樣本都屬於同一分類標記的子集合

- ID3
  - 以測試後資訊量最小的屬性為優先選取
  - 也就是選擇資訊獲利最大的屬性

Example(Information Gain)

n=16

n_1=4

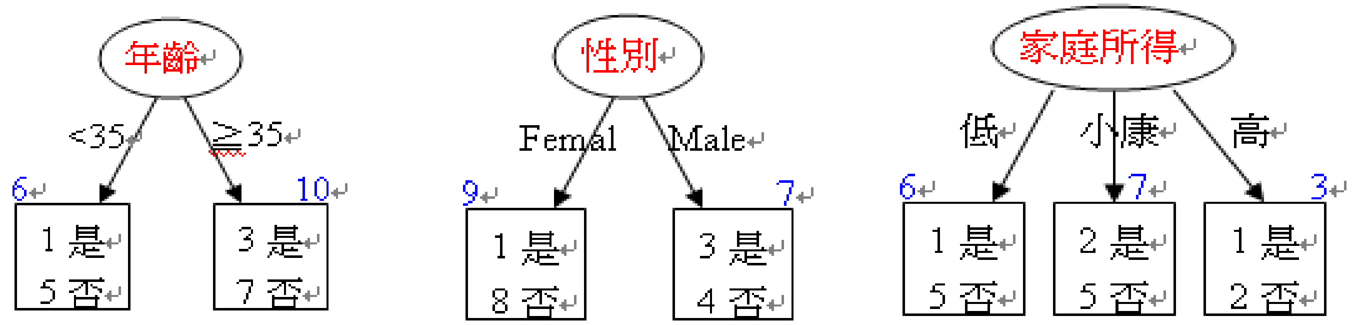


年齡	性別	家庭所得	購買RV房車
<35	Male	小康	否
≧35	Female	小康	否
≧35	Female	小康	否
≧35	Female	低所得	否
<35	Male	高所得	否
≧35	Female	低所得	否
<35	Female	低所得	否
<35	Female	高所得	是
≧35	Male	小康	是
<35	Male	高所得	否
≧35	Female	小康	否
<35	Male	低所得	否
≧35	Female	小康	否
≧35	Male	低所得	是
≧35	Male	小康	是
≧35	Female	低所得	否

$I(16,4)=-((4/16)*\log_2(4/16)+(12/16)*\log_2(12/16))=0.8113$

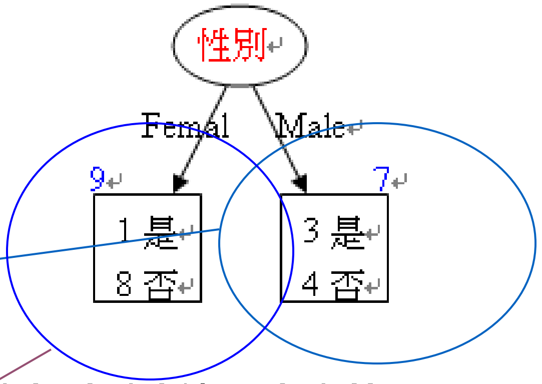
E(年齡)=(6/16)*I(6,1)+(10/16)*I(10,3)=0.7946

Gain(年齡)=I(16,4)-E(年齡)=0.0167



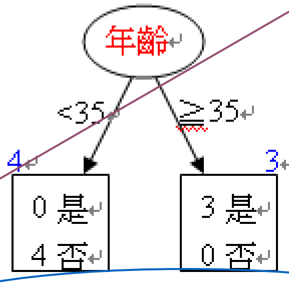
Gain(年齡)=0.0167　Gain(性別)=0.0972　Gain(家庭所得)=0.0177

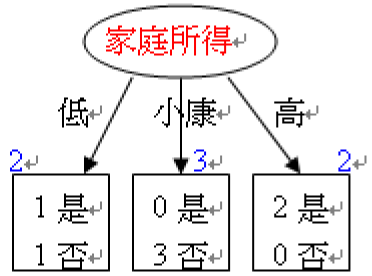
Max:作為第一個分類依據



年齡	性別	家庭所得	購買RV房車
<35	Male	小康	否
<35	Male	低所得	否
<35	Male	高所得	否
<35	Male	高所得	否
≧35	Male	小康	是
≧35	Male	小康	是
≧35	Male	低所得	是

$I(7,3)=-((3/7)*\log_2(3/7)+(4/7)*\log_2(4/7))=0.9852$

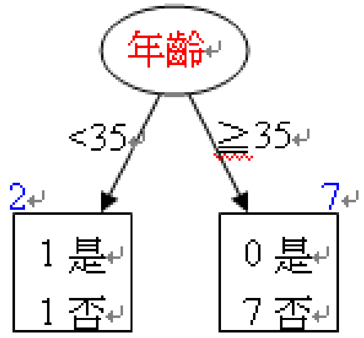


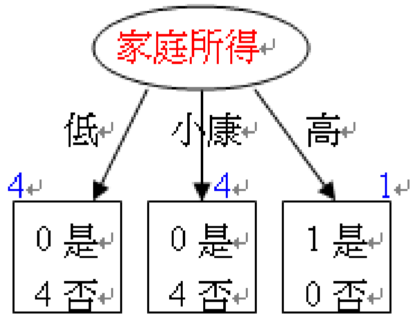


- Gain(年齡)=0.9852
- Gain(家庭所得)=0.688

年齡	性別	家庭所得	購買RV房車
<35	Female	低所得	否
<35	Female	高所得	是
≧35	Female	小康	否
≧35	Female	小康	否
≧35	Female	小康	否
≧35	Female	小康	否
≧35	Female	低所得	否
≧35	Female	低所得	否
≧35	Female	低所得	否

$I(9,1)=-((1/9)*\log_2(1/9)+(8/9)*\log_2(8/9))=0.5032$

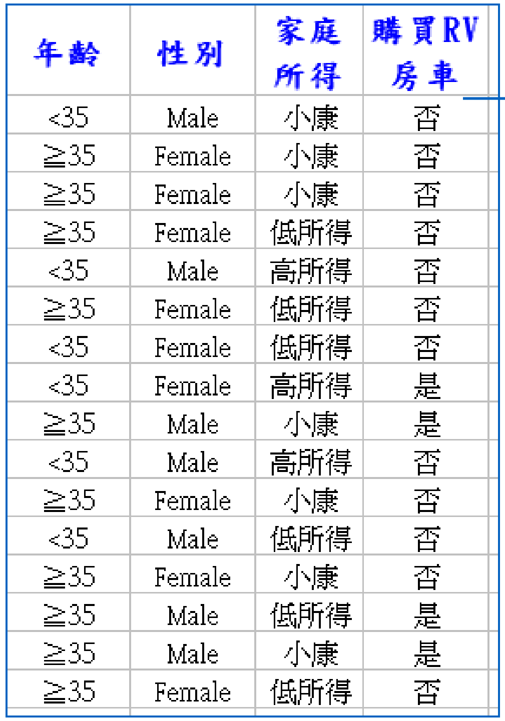




- Gain(年齡)=0.2222
- Gain(家庭所得)=0.5032

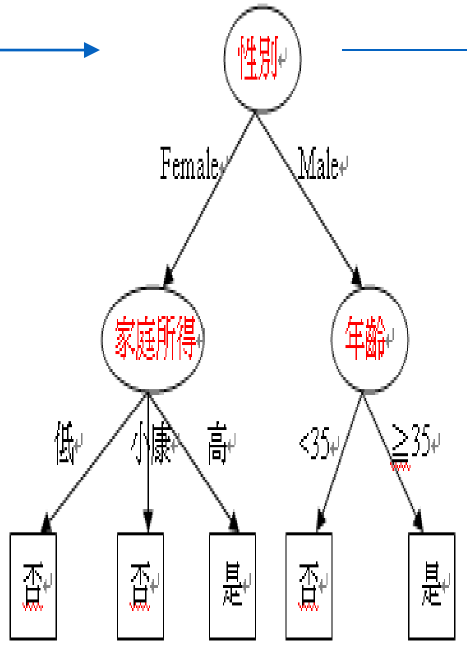
Example(end) ID3演算法

- 資料



年齡	性別	家庭所得	購買RV房車
<35	Male	小康	否
≧35	Female	小康	否
≧35	Female	小康	否
≧35	Female	低所得	否
<35	Male	高所得	否
≧35	Female	低所得	否
<35	Female	低所得	否
<35	Female	高所得	是
≧35	Male	小康	是
<35	Male	高所得	否
≧35	Female	小康	否
<35	Male	低所得	否
≧35	Female	小康	否
≧35	Male	低所得	是
≧35	Male	小康	是
≧35	Female	低所得	否

- Decision Tree



分類規則

IF 性別=Female AND 家庭所得 = 低所得 THEN 購買RV房車=否

IF 性別=Female AND 家庭所得 = 小康 THEN 購買RV房車=否

IF 性別=Female AND 家庭所得 = 高所得 THEN 購買RV房車=是

IF 性別=Male AND 年齡<35 THEN 購買RV房車=否

IF 性別=Male AND 年齡≧35 THEN 購買RV房車=是

ID3的問題

- ID3演算法所使用的資訊獲利
  - 會傾向選擇擁有許多不同數值的屬性
  - 例如：“產品編號”欄位中，每一個產品的產品編號皆不同。
  - 依產品編號進行分割，會產生出許多分支，且每一個分支都是很單一的結果，其資訊獲利會最大。但這個屬性對於建立決策樹是沒 有意義的。
- C4.5演算法
  - 利用屬性的獲利比率(Gain Ratio)
  - 資訊獲利正規化
  - 擁有最大獲利比例的屬性被設為分割屬性

Decision Tree Algorithms – C4.5

- **Recursive Building Tree Phase**:
  - *Initialize root node of tree.*
  - ***while** a node N that can be split:*

    ***for each** attribute A, evaluate splits on A, use best split to split N.*

- Use *Entropy index* to find best split.
- Separate attribute lists maintained in each node of tree.

C4.5 Algorithm

Let the classes be denoted $\{C_1, C_2, \ldots, C_k\}$. There are three possibilities for the content of the set of training samples T in the given node of decision tree:

1. *__T__ contains one or more samples, all belonging to a single class C_j:* The decision tree for T is a leaf identifying class C_j.

2. *__T__ contains no samples:*

The decision tree is again a leaf, (the class to be associated with the leaf must be determined from information other than T)

C4.5 algorithm uses as a criterion the most frequent class at the **parent** of the given node.

C4.5 Algorithm

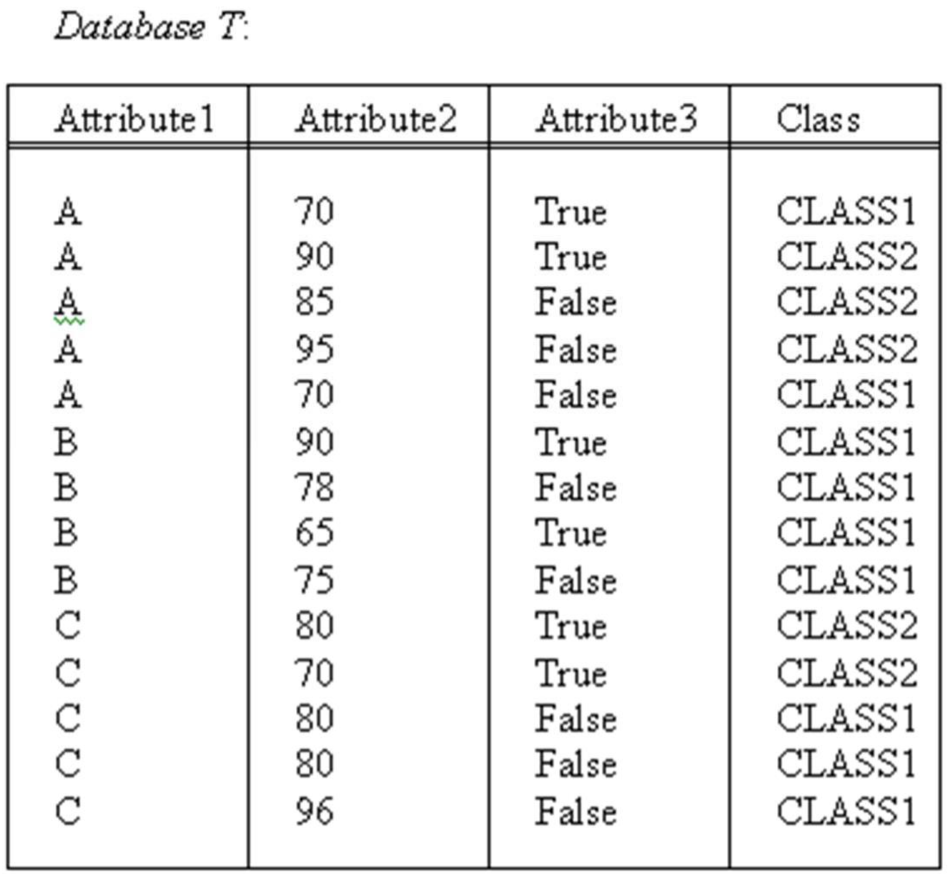
3. ***T** contains samples that belong to a mixture of classes:*
   In this situation, the idea is to refine T into subsets of samples that are heading towards single-class collections of samples. An **appropriate test** is chosen, based on single attribute, that has one or more mutually exclusive outcomes $\{O_1, O_2, \ldots, O_n\}$:

   - T is partitioned into subsets T_1, T_2, …, T_n where Ti contains all the samples in T that have outcome O_i of the chosen test.

   - The decision tree for T consists of a decision node identifying the test and one branch for each possible outcome.

C4.5 Algorithm

An Example



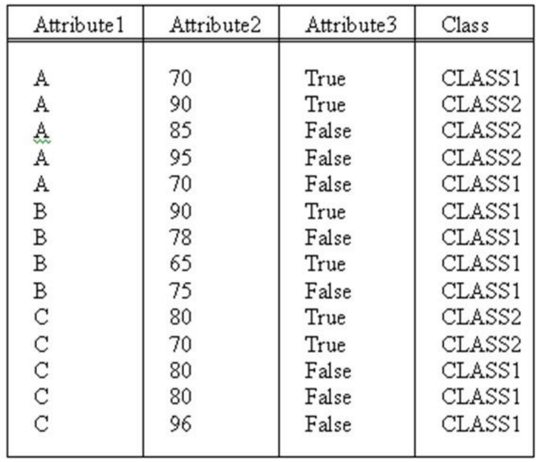
Database T:

Attribute1	Attribute2	Attribute3	Class
A	70	True	CLASS1
A	90	True	CLASS2
A	85	False	CLASS2
A	95	False	CLASS2
A	70	False	CLASS1
B	90	True	CLASS1
B	78	False	CLASS1
B	65	True	CLASS1
B	75	False	CLASS1
C	80	True	CLASS2
C	70	True	CLASS2
C	80	False	CLASS1
C	80	False	CLASS1
C	96	False	CLASS1

C4.5 Algorithm

(An Example)

Database T:



Attribute1	Attribute2	Attribute3	Class
A	70	True	CLASS1
A	90	True	CLASS2
A	85	False	CLASS2
A	95	False	CLASS2
A	70	False	CLASS1
B	90	True	CLASS1
B	78	False	CLASS1
B	65	True	CLASS1
B	75	False	CLASS1
C	80	True	CLASS2
C	70	True	CLASS2
C	80	False	CLASS1
C	80	False	CLASS1
C	96	False	CLASS1

$\text{Info}(T) = -9/14 \ \log_2(9/14) \ -5/14 \ \log_2(5/14) = 0.940$ bits

Attribute1:
========

$$\begin{aligned}\text{Info}_{x1}(T) &= 5/14\,(-2/5 \log_2 (2/5) - 3/5 \log_2 (3/5)) \\ &\quad + 4/14\,(-4/4 \log_2 (4/4) - 0/4 \log_2 (0/4)) \\ &\quad + 5/14\,(-3/5 \log_2 (3/5) - 2/5 \log_2 (2/5)) \\ &= 0.694 \text{ bits}\end{aligned}$$

$\text{Gain}(x_1) = 0.940 - 0.694 =$ **0.246 bits**

Attribute3:
========

$$\begin{aligned}\text{Info}_{x2}(T) &= 6/14\,(-3/6 \log_2 (3/6) - 3/6 \log_2 (3/6)) \\ &\quad + 8/14\,(-6/8 \log_2 (6/8) - 2/8 \log_2 (2/8)) \\ &= 0.892 \text{ bits}\end{aligned}$$

$\text{Gain}(x_2) = 0.940 - 0.892 =$ **0.048 bits**

C4.5 Algorithm

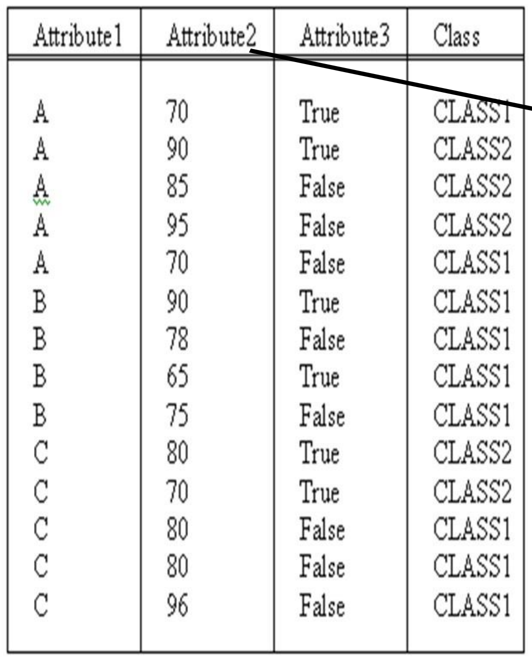
b) If attribute Y has **continuous numeric values**, a binary test with outcomes $Y \le Z$ and $Y > Z$ could be defined, based on comparing the value of attribute against a threshold value Z.

Threshold value Z:

- The training samples are first sorted on the values of the attribute Y being considered. There are only a finite number of these values, so let us denote them in sorted order as $\{v_1, v_2, \ldots, v_m\}$.
- Any threshold value lying between v_i and v_{i+1} will have the same effect of dividing the cases into those whose value of the attribute Y lies in $\{v_1, v_2, \ldots, v_i\}$ and those whose value is in $\{v_{i+1}, v_{i+2}, \ldots, v_m\}$. There are thus only **m-1 possible splits on Y**, all of which should be examined systematically to obtain an optimal split.
- It is usual to choose the midpoint of each interval: $(v_i + v_{i+1})/2$ as the representative threshold. C4.5 chooses as the threshold a **smaller value v_i** for every interval $\{v_i, v_{i+1}\}$, rather than the midpoint itself.

C4.5 Algorithm

Database T: *An Example*



Attribute1	Attribute2	Attribute3	Class
A	70	True	CLASS1
A	90	True	CLASS2
A	85	False	CLASS2
A	95	False	CLASS2
A	70	False	CLASS1
B	90	True	CLASS1
B	78	False	CLASS1
B	65	True	CLASS1
B	75	False	CLASS1
C	80	True	CLASS2
C	70	True	CLASS2
C	80	False	CLASS1
C	80	False	CLASS1
C	96	False	CLASS1

Attribute2: - After a sorting process, the set of values is:
{65, 70, 75, 78, 80, 85, 90, 95, 96},
and the set of potential threshold values Z is:

{65, 70, 75, 78, 80, 85, 90, 95}.

- The optimal Z value is Z=80, and the corresponding process of information gain computation for the test x_3 (Attribute2 $\leq$ 80 or Attribute2 > 80)

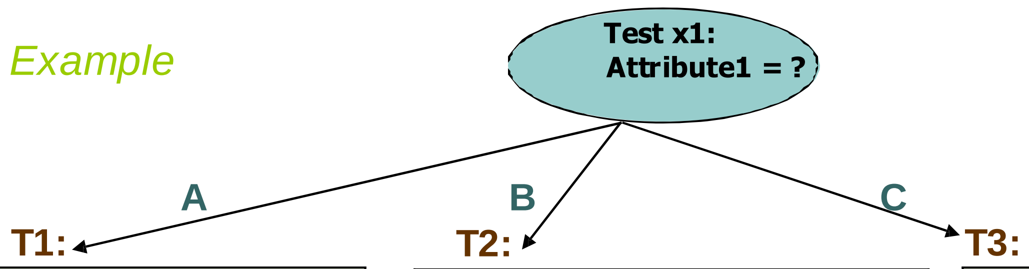
$$\text{Info}_{x3}(T) = 9/14\,(-7/9 \log_2(7/9) - 2/9 \log_2(2/9)) + 5/14\,(-2/5 \log_2(2/5) - 3/5 \log_2(3/5)) = 0.837 \text{ bits}$$

$$\text{Gain}(x_3) = 0.940 - 0.837 = \mathbf{0.103 \text{ bits}}$$

Attribute1 gives the highest gain of 0.246 bits, and therefore this attribute will be selected for the first splitting.

C4.5 Algorithm

An Example



T1:

Att.2	Att.3	Class
70	True	CLASS1
90	True	CLASS2
85	False	CLASS2
95	False	CLASS2
70	False	CLASS1

T2:

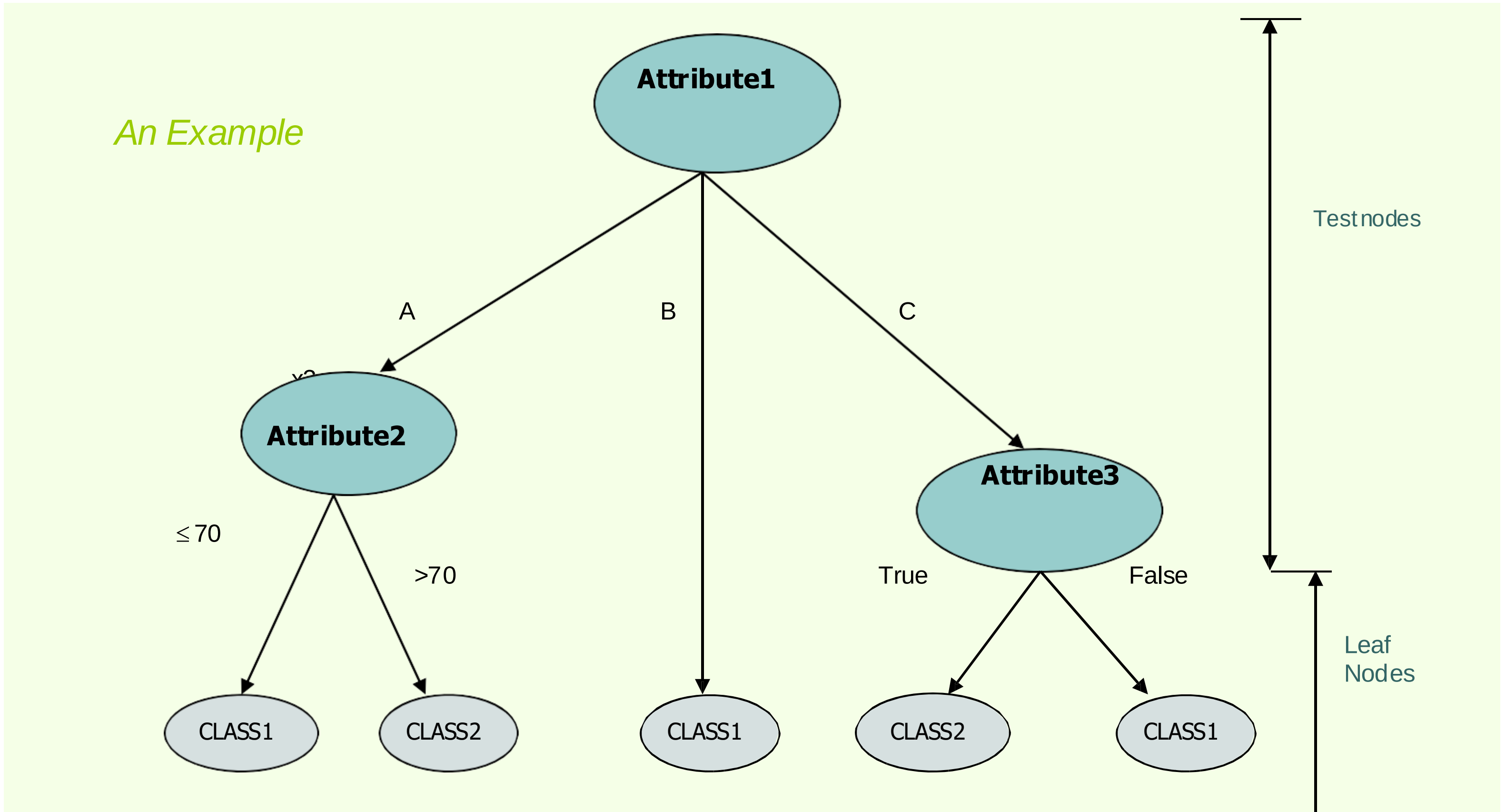
Att.2	Att.3	Class
90	True	CLASS1
78	False	CLASS1
65	True	CLASS1
75	False	CLASS1

T3:

Att.2	Att.3	Class
80	True	CLASS2
70	True	CLASS2
80	False	CLASS1
80	False	CLASS1
96	False	CLASS1

Initial decision tree and subset cases for a database T

C4.5 Algorithm



A final decision tree for database T

C4.5 Algorithm

Decision Tree – Pseudo-code:

An Example

```
If        Attribute1 = A
          Then
                    If        Attribute2 <= 70
                              Then
                                        Classification = CLASS1;
                              Else
                                        Classification = CLASS2;
Elseif    Attribute1 = B
          Then
                                        Classification = CLASS1;
Elseif    Attribute1 = C
          Then
                    If        Attribute3 = True
                              Then
                                        Classification = CLASS2;
                              Else
                                        Classification = CLASS1.
```

C4.5 Algorithm

- Gain criterion has **a serious deficiency**: a strong bias in favor of tests with many outcomes.
- Solution was find in some kind of normalization:

$$\text{Split-info}(X) = -\sum_{i=1}^{n} ((|T_i| / |T|) \log_2 (|T_i| / |T|))$$

$$\text{Gain-ratio}(X) = \text{Gain}(X) / \text{Split-info}(X)$$

Example:

Split-info(x_1) = - 5/14 $\log_2$ (5/14) – 4/14 $\log_2$ (4/14) - 5/14 $\log_2$ (5/14) = 1.577 bits Gain-ratio(x_1) = 0.246 / 1.557 = **0.156**

Decision Tree Algorithms

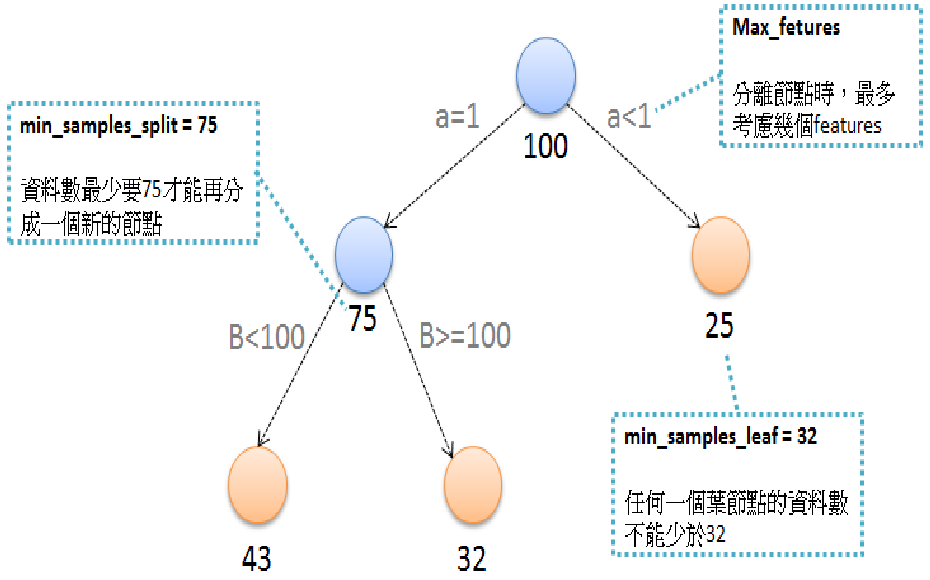
- **Building phase**
  - Recursively split nodes using best splitting attribute for node.
- **Pruning phase**
  - Smaller imperfect decision tree generally achieves better accuracy.
  - Prune leaf nodes recursively to prevent over-fitting.

Avoid Overfitting in Classification

- The generated tree may **overfit** the training data:
  - Too many branches, some may reflect anomalies due to noise or outliers.
  - Result is in poor accuracy for unseen samples.
- Two approaches to avoid **overfitting**:
  - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold.
    - Difficult to choose an appropriate threshold.
  - Postpruning: Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree".

Prepuning

- Minimum samples for a node split：資料數目不得小於多少才能再產生新節點。
- Minimum samples for a terminal node (leaf)：要成為葉節點，最少需要多少資料。
- Maximum depth of tree (vertical depth)：限制樹的高度最多幾層。
- Maximum number of terminal nodes：限制最終葉節點的數目
- Maximum features to consider for split：在分離節點時，最多考慮幾種特徵值。


min_samples_split = 75
資料數最少要75才能再分成一個新的節點
Max_fetures
分離節點時，最多考慮幾個features
100
a=1
a<1
75
25
B<100
B>=100
43
32
min_samples_leaf = 32
任何一個葉節點的資料數不能少於32

Classification And Regression Trees (CART)

Splitting Based on GINI index

- When a node p is split into k partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,

n = number of records at node p.

Attribute Selection using Gini **Index**

- If a data set *T* contains examples from *n* classes, gini index, *gini*(*T*) is defined as

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class *j* in *T.*

- If a data set *T* is split into two subsets T_1 and T_2 with sizes N_1 and N_2 respectively, the *gini* index of the split data contains examples from *n* classes, the *gini* index *gini*(*T*) is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute provides the smallest $gini_{split}(T)$ is chosen to split the node (*need to enumerate all possible splitting points for each attribute*).

Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_{j} [p(j \mid t)]^2$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

- Maximum (1 - 1/n_c) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Examples for Computing GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – P(C1)^2 – P(C2)^2 = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – (1/6)^2 – (5/6)^2 = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – (2/6)^2 – (4/6)^2 = 0.444

集　成　學　習

Ensemble Learning

- So far – learning methods that learn a single hypothesis (model), chosen form a hypothesis space that is used to make predictions.

"No Free Lunch" Theorem

- No single algorithm wins all the time!

- Ensemble learning
  - select a collection (*ensemble*) of hypotheses (models) and combine their predictions.
- Example 1 - *generate 100 different decision trees from the same or different training set and have them vote on the best classification for a new example.*

Example: Weather Forecast

	⛈	☀	☀	⛈	⛈	☀	☀
1	⛈	⛈ X	☀	☀ X	⛈	☀	⛈ X
2	☀ X	☀	☀	☀ X	⛈	☀	⛈ X
3	⛈	☀	⛈ X	⛈	☀ X	⛈ X	☀
4	⛈	☀	⛈ X	⛈	☀ X	☀	☀
5	⛈	⛈ X	☀	⛈	⛈	⛈ X	☀
Combine	⛈	☀	☀	⛈	⛈	☀	☀

Combine decisions of multiple models using voting procedure!!!

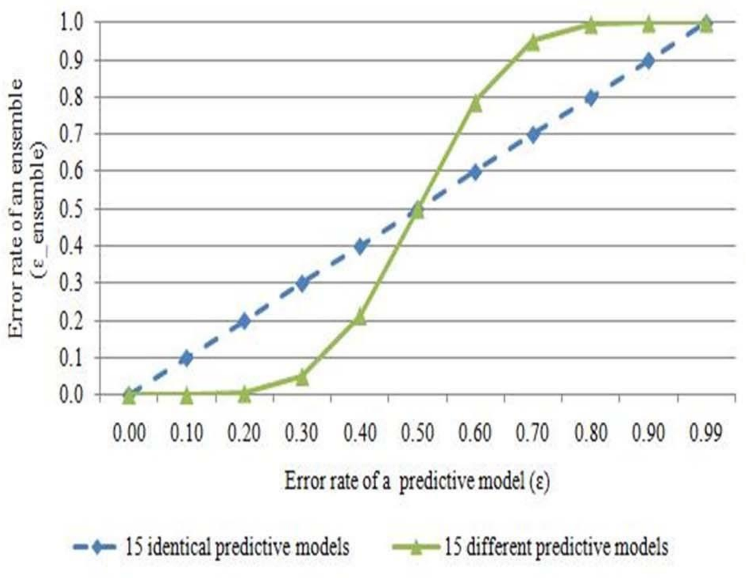
Intuition

- Majority vote: ε=0.3, n=15 => Binomial Distribution

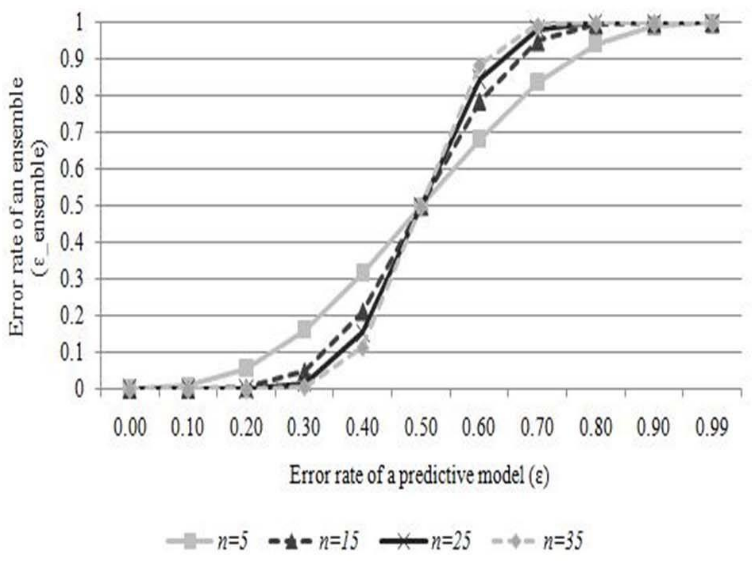
$$\varepsilon_{ensemble} = \sum_{i=8}^{15} \binom{15}{i} \varepsilon^{i}(1-\varepsilon)^{15-i} = 0.05$$

- Suppose we have 5 completely independent classifiers…
  - If accuracy is 70% for each
    - $(.7^5)+5(.7^4)(.3)+10(.7^3)(.3^2)+\ldots$
    - **83.7% majority vote accuracy**
  - 101 such classifiers
    - **99.9% majority vote accuracy**

Intuition – not any more…



(a) Identical predictive models vs. different predictive models in an ensemble



(b) The different number of predictive models in an ensemble

Value of Ensembles

- *Key motivation*: reduce the error rate!!!
  - *Hope is that it will become much more unlikely that the ensemble will misclassify an example.*
- Examples: Human ensembles are demonstrably better:
  - *How many jelly beans in the jar?: Individual estimates vs. group average.*
  - *Who Wants to be a Millionaire: Audience vote.*

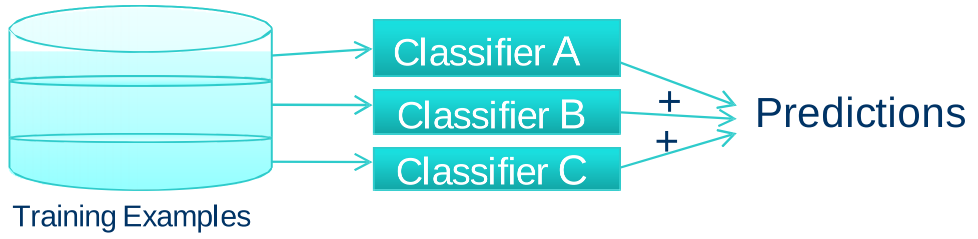
- *Theory behind:* When combing multiple **independent** and **diverse decisions** each of which is *at least more accurate than random guessing*, random errors cancel each other out, correct decisions are reinforced.

Achieving Diversity

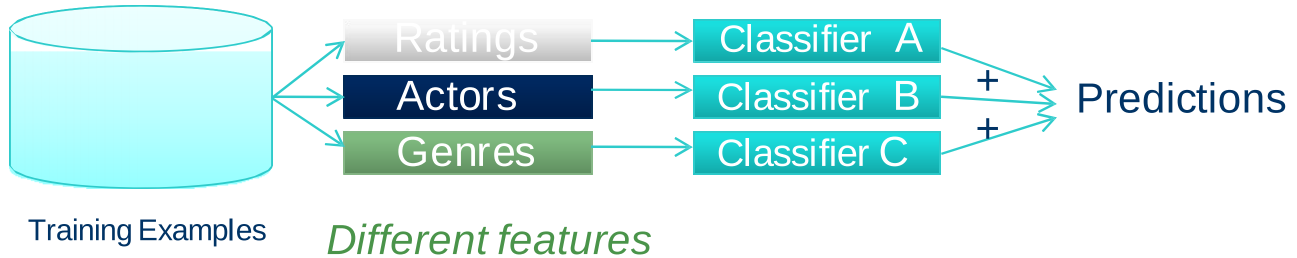
- Using different learning algorithms
- Using different hyper-parameters in the same algorithm (for example, different polynomial regression)
- Using different input representations, such as using different subsets of input features
- Using different training subsets of input data, where we have known procedures of bagging, boosting, and cascading.

Achieving Diversity

Diversity from differences in inputs



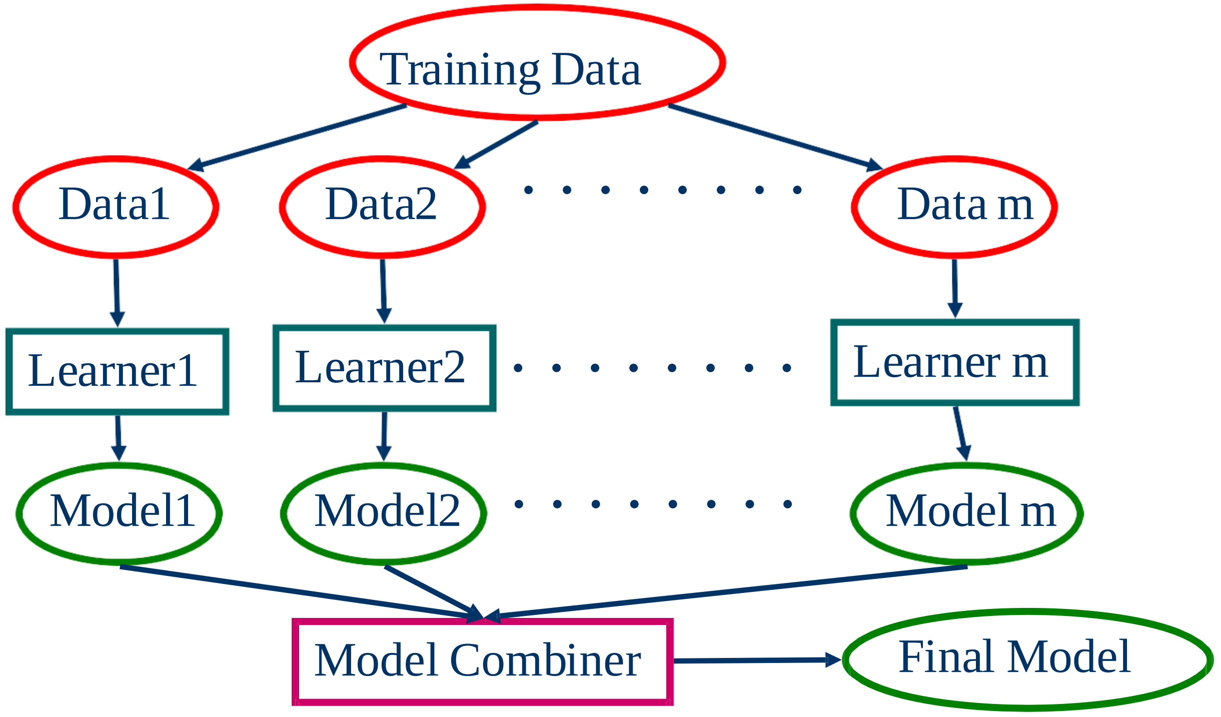
Divide up training data among models



Different features

Learning Ensembles

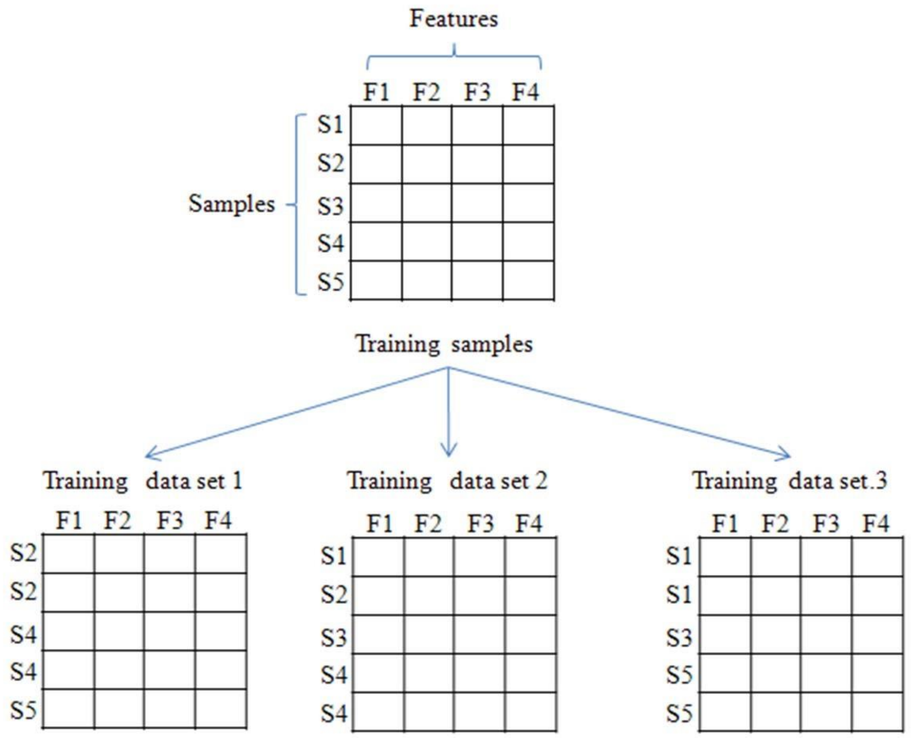
For example: Learn multiple alternative definitions of a concept using: **different training data**



Homogenous Ensembles

- Use a single, arbitrary learning algorithm but manipulate training data to make it learn multiple models.
  - Data1 ≠ Data2 ≠ … ≠ Data m
  - Learner1 = Learner2 = … = Learner m
- Different methods for changing training data:
  - Bagging: Resample training data
  - Boosting: Reweight training data

Bagging


Features
F1 F2 F3 F4
Samples
S1
S2
S3
S4
S5
Training samples
Training data set 1
F1 F2 F3 F4
S2
S2
S4
S4
S5
Training data set 2
F1 F2 F3 F4
S1
S2
S3
S4
S4
Training data set 3
F1 F2 F3 F4
S1
S1
S3
S5
S5

Bagging

- Create ensembles by "*bootstrap aggregation*", i.e., repeatedly randomly resampling the training data (Brieman, 1996).
  - Bootstrap: draw N items from X with replacement

- Bagging approach:
  - Train *M* learners on *M* bootstrap samples
  - Combine outputs by voting (e.g., majority vote)

- Bagging: Decreases error by decreasing the variance in the results due to *unstable learners*: algorithms (*like decision trees and neural networks*) whose output can change dramatically when the training data is slightly changed.

Random Forest

隨機森林演算法

- 訓練樣本數為N，特徵數為M
- 1. 自訓練樣本以(Bagging, **B**ootstrap **agg**regat**ing**)產生數目n(<N)的隨機樣本訓練集
- 2. 對隨機樣本訓練集進行決策樹學習
  - A. 針對每個節點，隨機選取m個特徵(取出不放回)
  - B. 依據不同目標函數(entropy, gini, Squared Error)，找出最佳分割
- 3. 重複k次(步驟1~2)(訓練出k棵決策樹)
- 4. 分類用majority voting；回歸用平均

Random Forest – practical consideration

- Splits are chosen according to a purity measure:
  - E.g. squared error (regression), Gini index or devinace (classification)
- How to select N?
  - Build trees until the error no longer decreases
- How to select M?
  - Try to recommend defaults, half of them and twice of them and pick the best.

Adaboost

Boosting: Strong and Weak Learners

- Strong Learner Objective of machine learning
  - Take labeled data for training
  - Produce a classifier which can be *arbitrarily accurate*

- Weak Learner
  - Take labeled data for training
  - Produce a classifier which is more accurate than random guessing

Boosting: Strong and Weak Learners

- Weak Learner: only needs to generate a hypothesis with a training accuracy greater than 0.5, i.e., < 50% error over any distribution
  - *Strong learners are very difficult to construct*
  - *Constructing weaker learners is relatively easy*

- Questions: Can a set of **weak learners** create a single **strong learner** ?

YES ☺

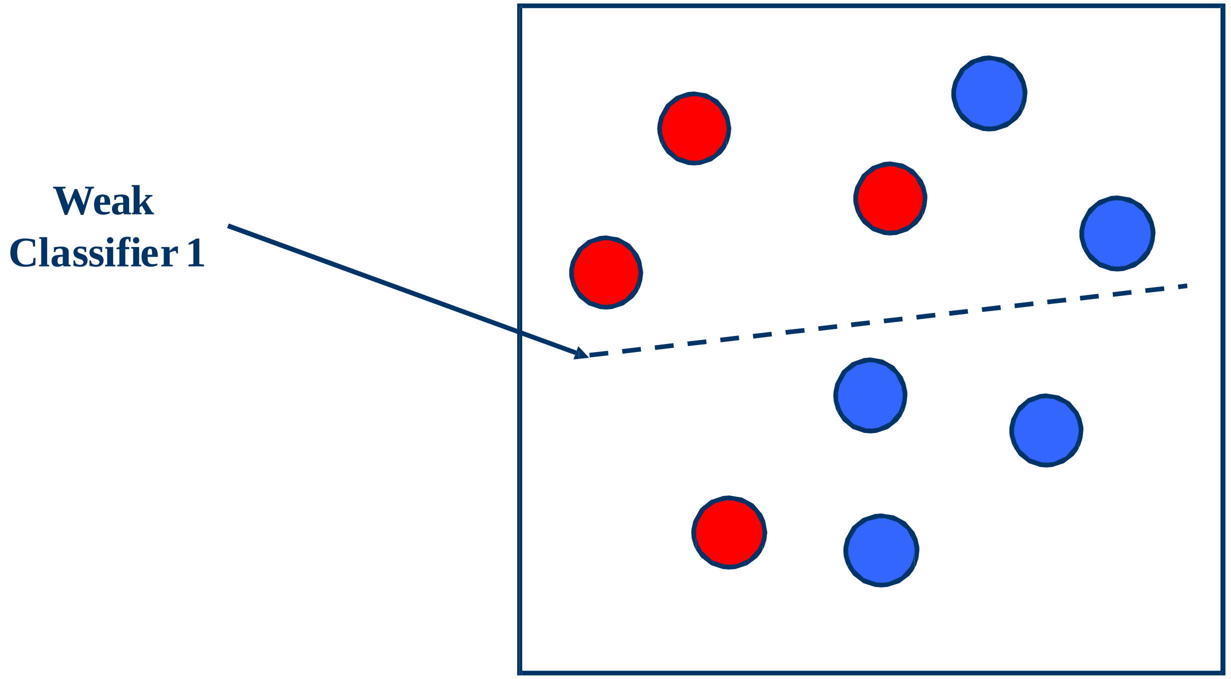
Boost weak classifiers to a strong learner!!!! Shapire (1990)

Boosting: Construct Weak Classifiers

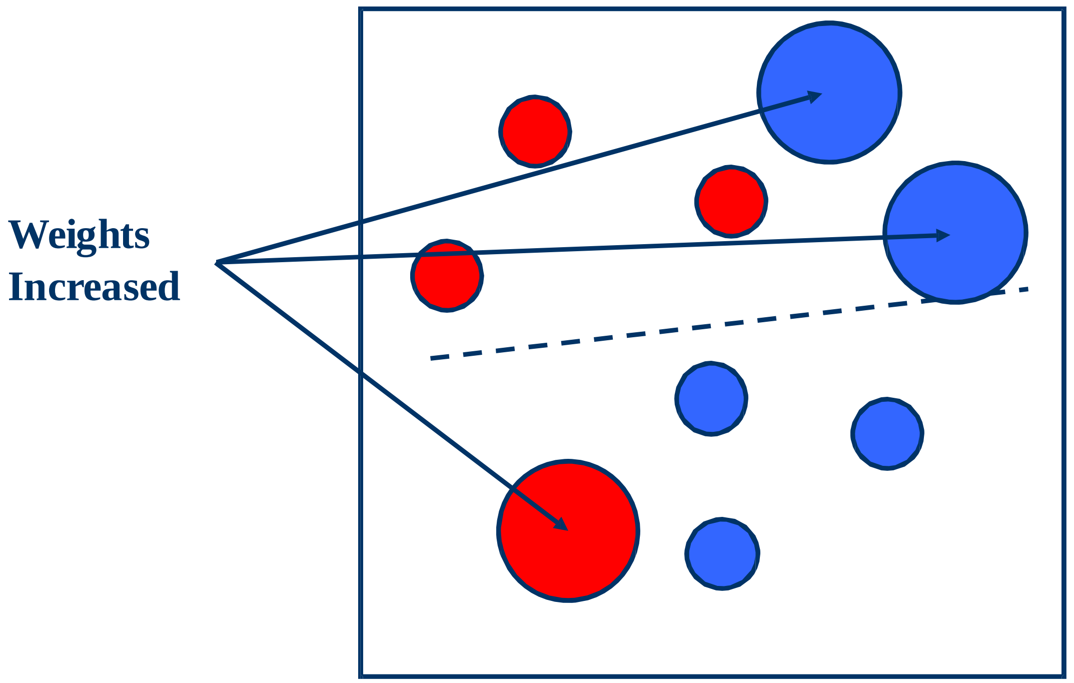
- *Idea:* Focus on difficult samples which are not correctly classified in the previous steps

- Use different data distribution:
  - Start with uniform weighting of samples
  - During each step of learning
    - Increase weights of the samples which are not correctly learned by the weak learner
    - Decrease weights of the samples which are correctly learned by the weak learner

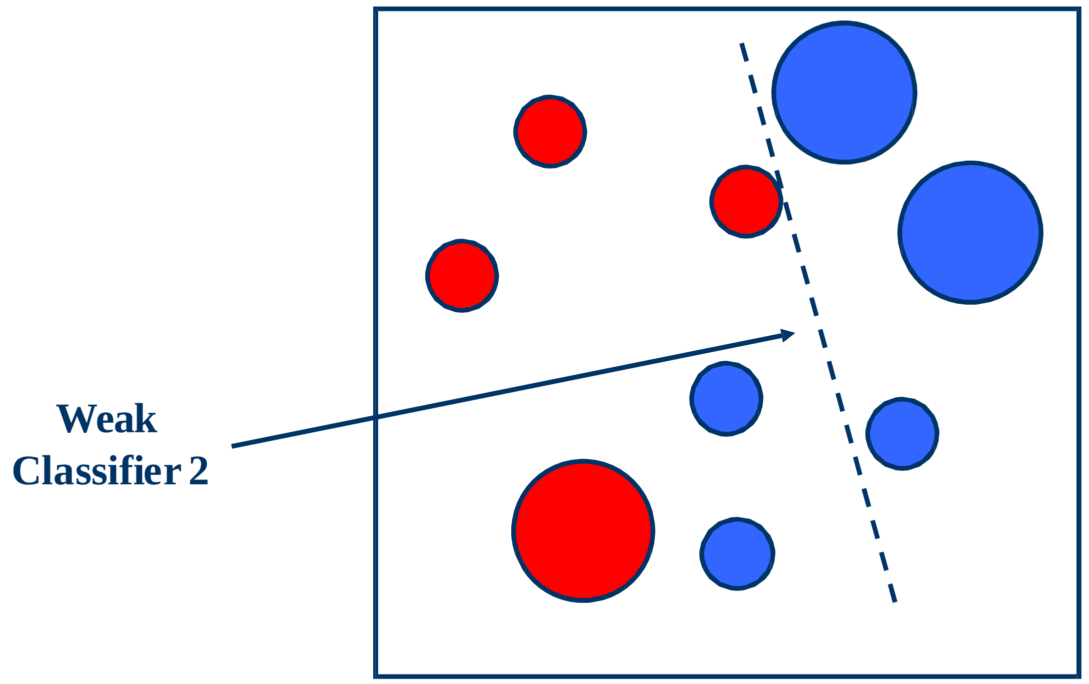
Boosting idea



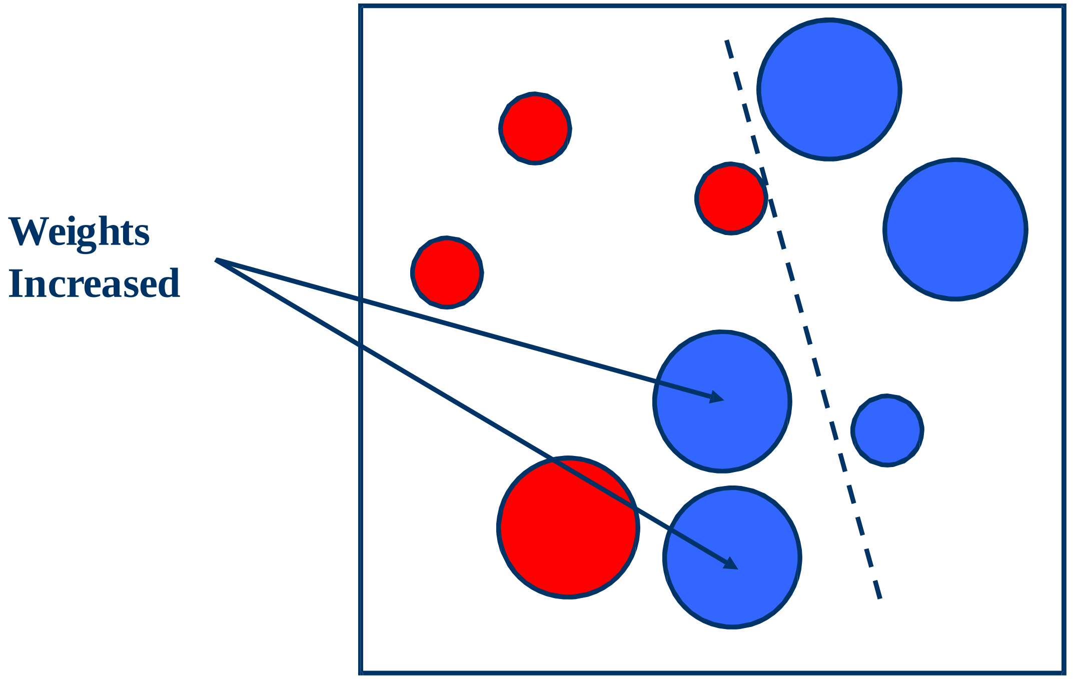
Boosting idea



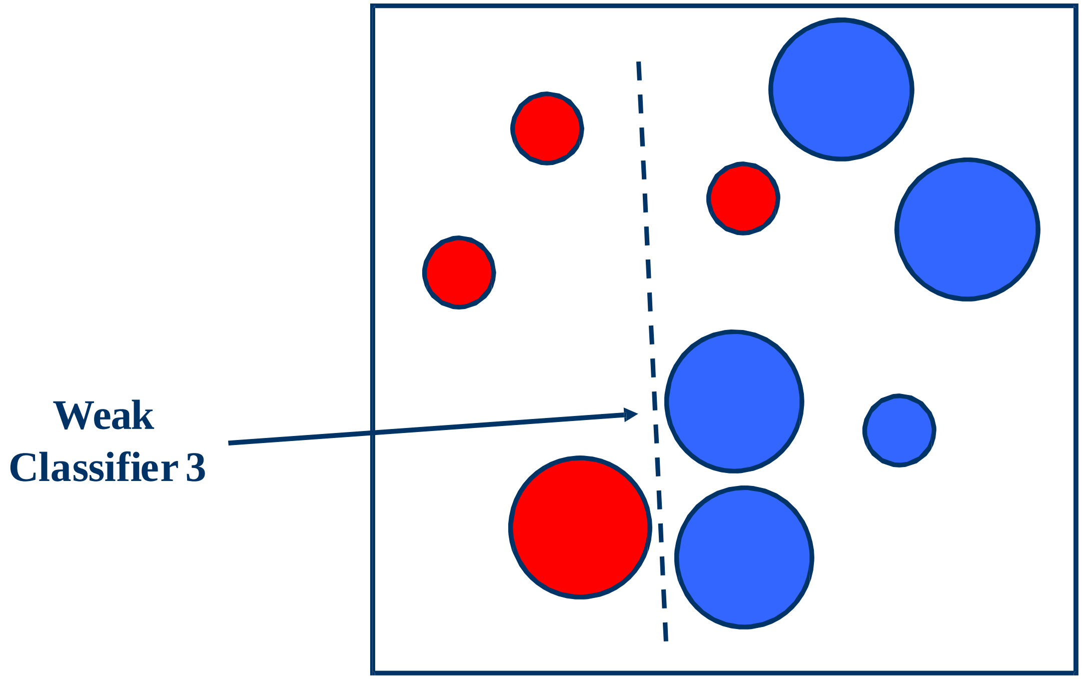
Boosting idea



Boosting idea

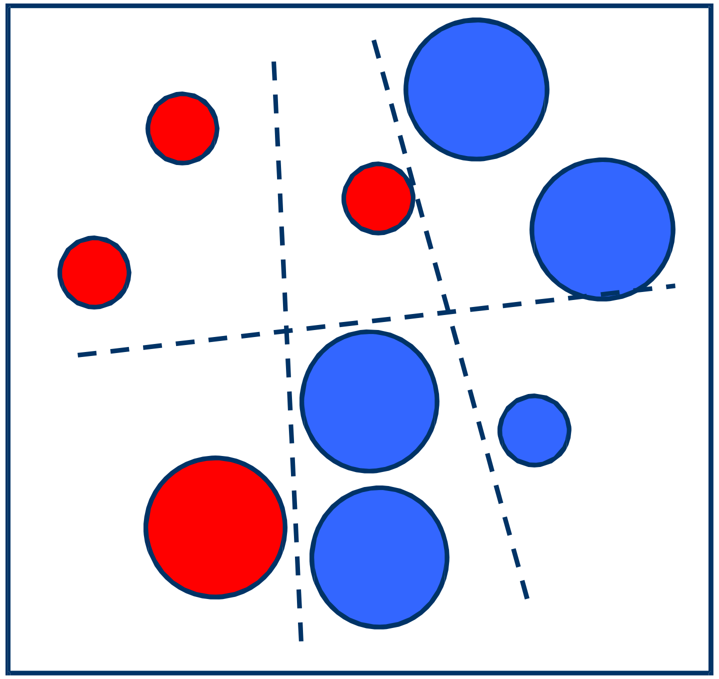


Boosting idea



Boosting idea

Final classifier is
a combination of weak
classifiers



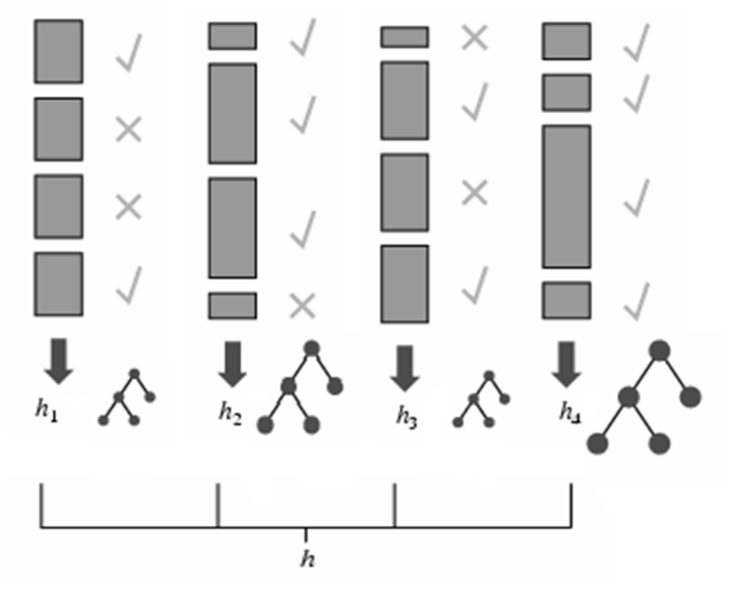
Boosting: Combine Weak Classifiers

- *Idea:* Better weak classifier gets a larger weight!!!

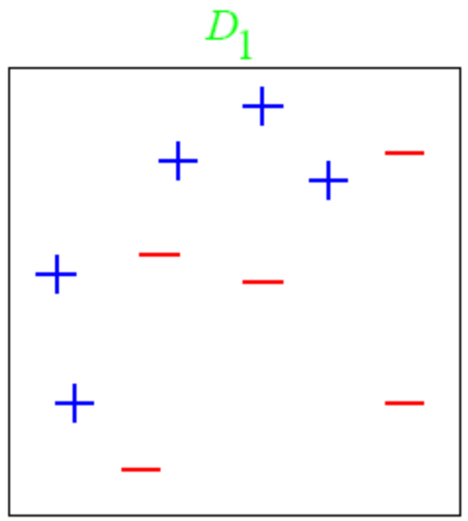
- Weighted Voting
  - Construct strong classifier by weighted voting of the weak classifiers
  - Weight of each learner is directly proportional to its accuracy

AdaBoost: Adaptive Boosting

- Each rectangle corresponds to an example, with weight proportional to its height.
- Crosses correspond to misclassified examples.
- Size of decision tree indicates the weight of that hypothesis in the final ensemble.

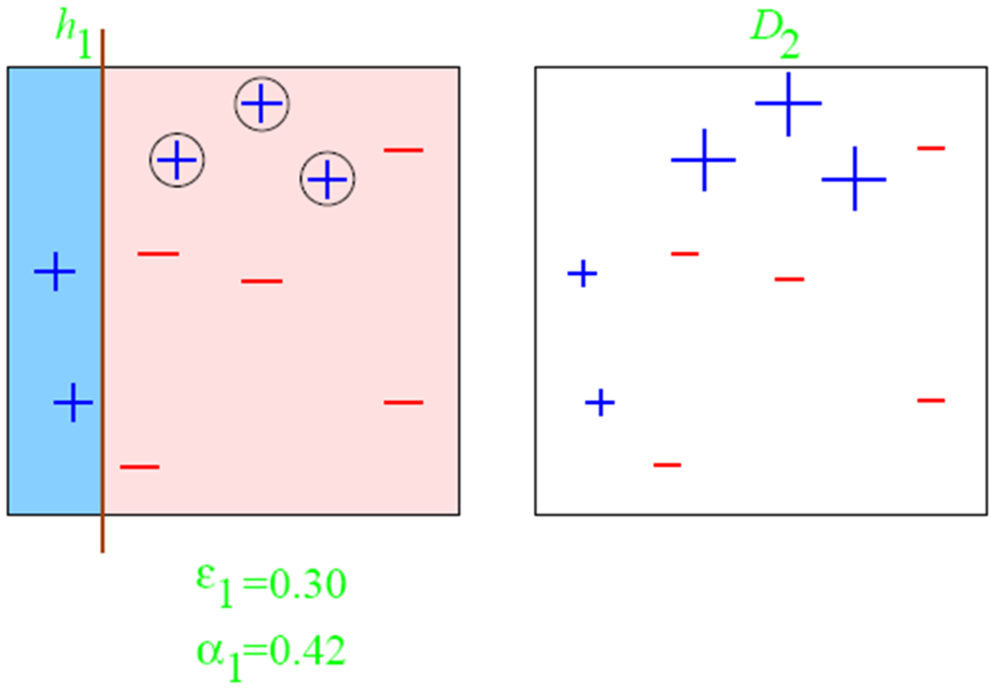


Toy Example

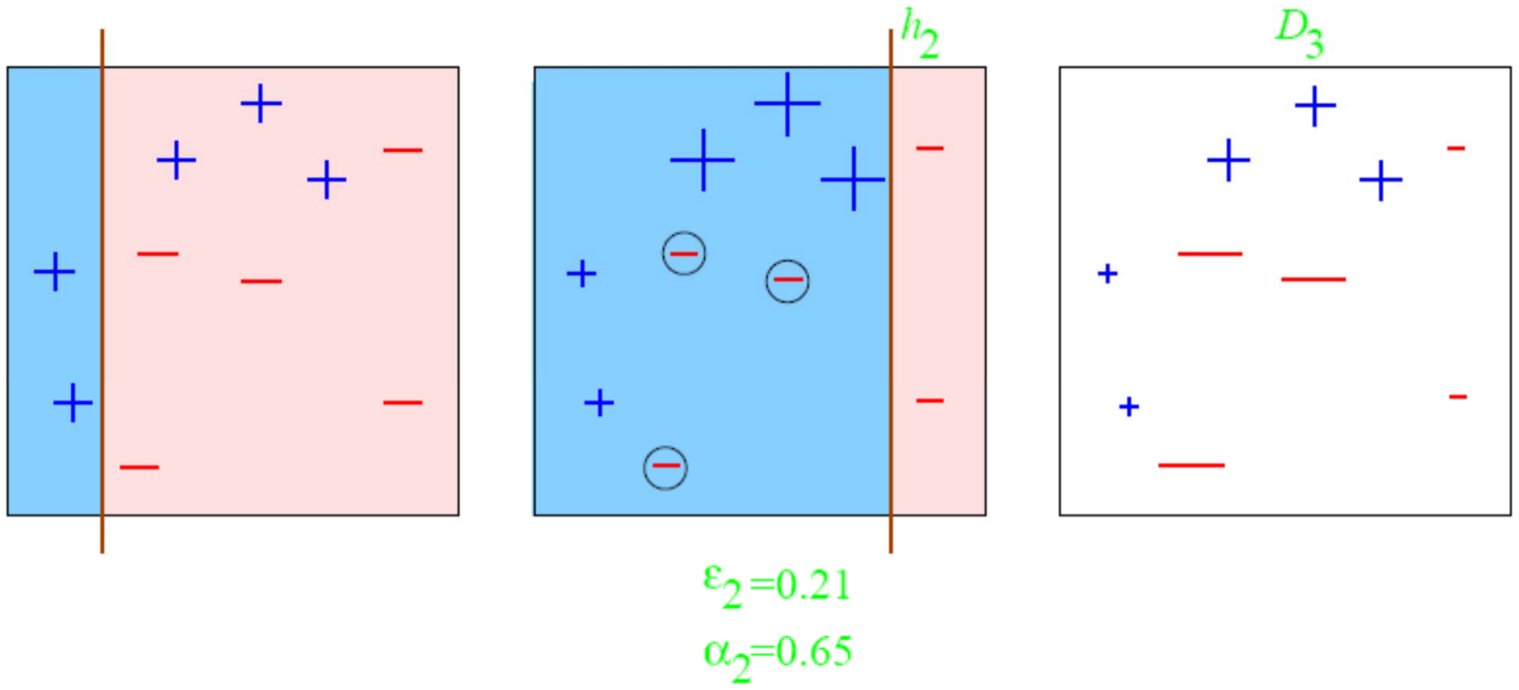


weak classifiers = vertical or horizontal half-planes

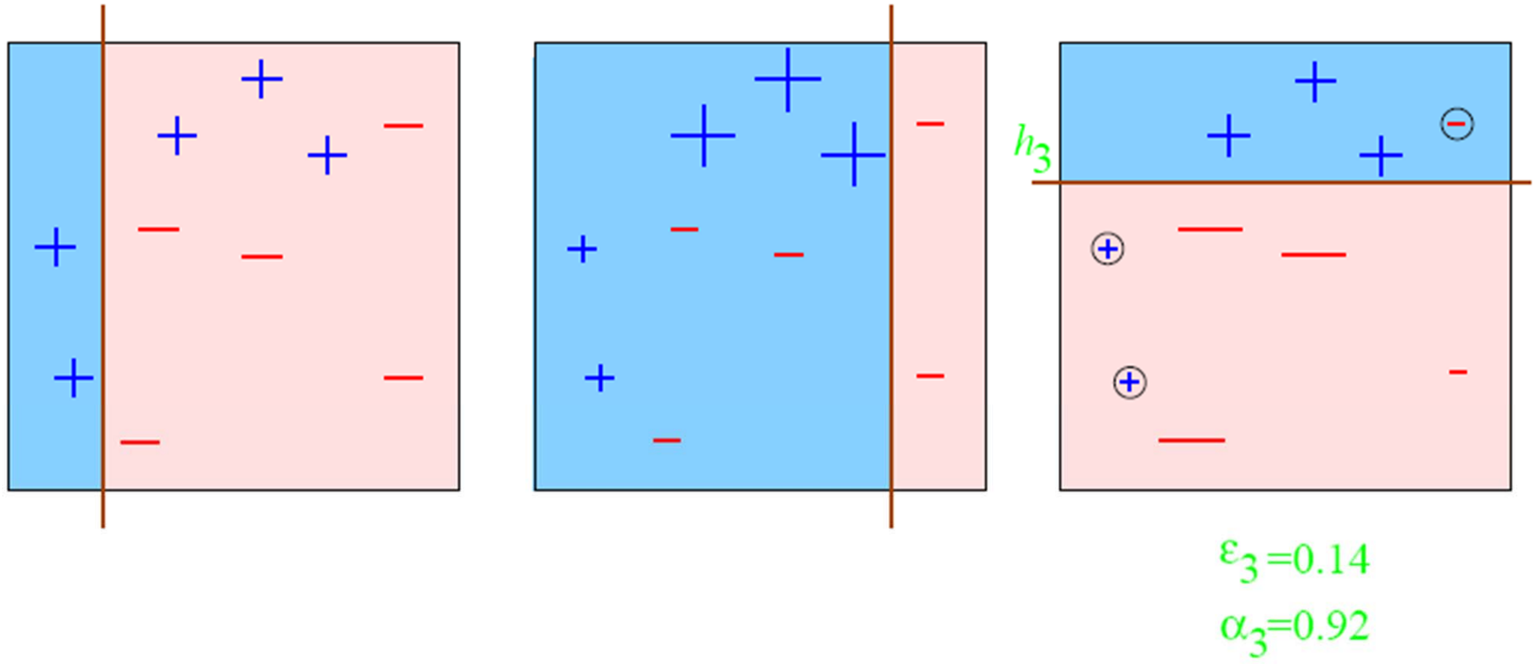
Round 1



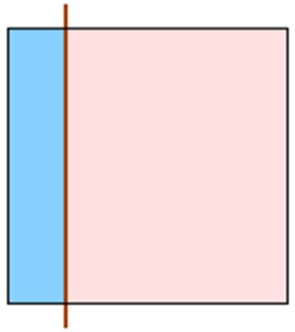
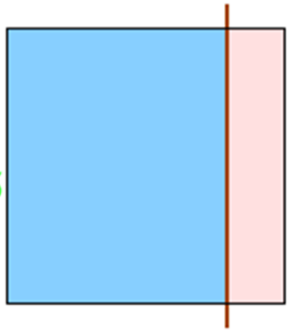
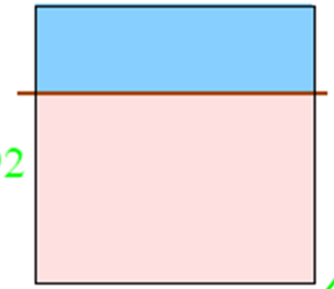
Round 2

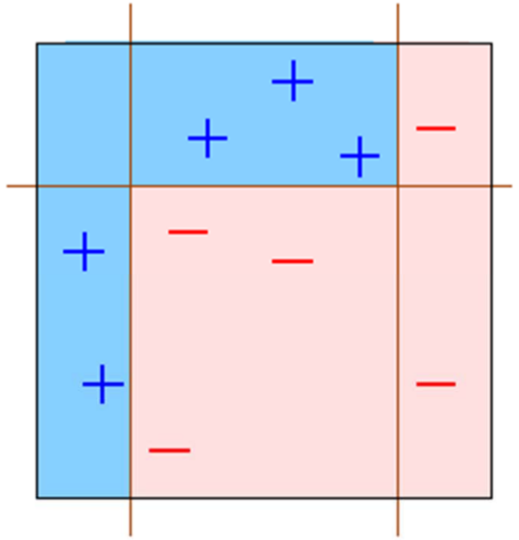


Round 3

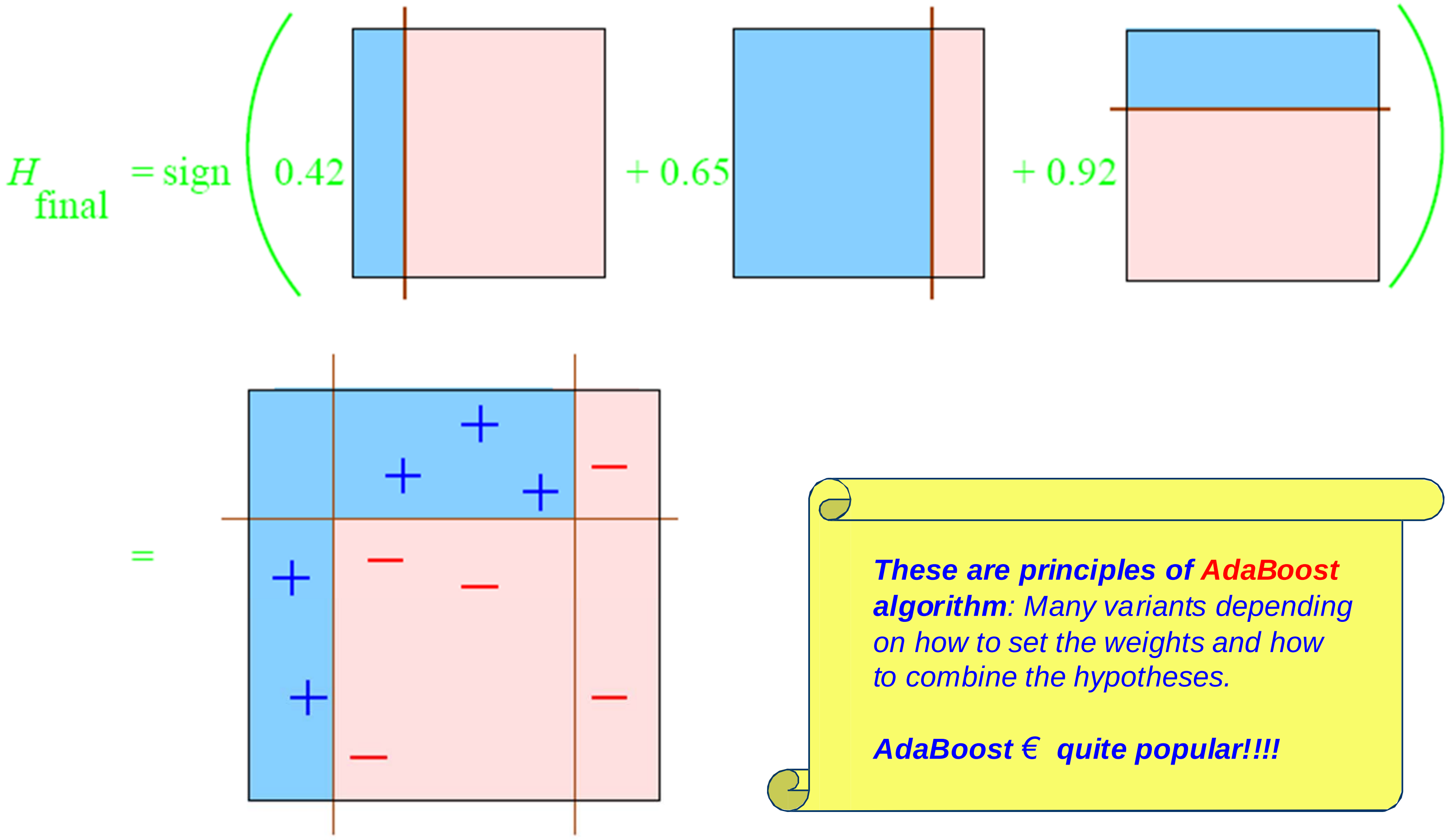


Final Classifier

$H_{\text{final}} = \text{sign}\Big(0.42$  $+ 0.65$  $+ 0.92$  $\Big)$

$=$ 

Final Classifier



***These are principles of AdaBoost algorithm**: Many variants depending on how to set the weights and how to combine the hypotheses.*

AdaBoost € quite popular!!!!

AdaBoost演算法

輸入：訓練集$D = \{(x_1;y_1),(x_2;y_2),...,(x_m;y_m)\}$；

基礎學習器L；訓練迴圈輪數T；

初始化$D_1(i) = 1/N$,第一次反覆運算時每個樣本的權重都為$1/N$

For$t$= 1 to$T$

在D_t下訓練：得到弱假設h_t，即通過訓練集和當前權值分佈得到第t次的弱學習器$h_t=L(D,D_t)$

計算h_t的錯誤率：$\varepsilon_t=\Sigma D_t(i)[h_t(x_i)\neq y_i]$

令$\alpha_t=1/2\ln((1-\varepsilon_t)/\varepsilon_t)$此即為$h_t$的權重

更新樣本權值：

$$D_{t+1}(i) = \frac{D_t(t)}{Z_t} * \begin{cases} e^{-a_t} & if\ h_t(x_i) = y_i \\ e^{a_t} & if\ h_t(x_i) \neq y_i \end{cases}$$

Z_t為歸一化因數

迴圈結束

輸出：

$$H(X) = sign(\sum_{t=1}^{T} a_t h_t(x))$$

Gradient Boosting

- Gradient Boosting是一種迭代的boosting算法
  - 由Freidman於1999年首次提出
  - 連續訓練多個弱學習器
  - 每次優化前一個弱學習器的損失函數負梯度方向
  - 逐步構建一個強大的集成模型
  - 在每一輪訓練中，Gradient Boosting會計算樣本的損失函數負梯度，用這些負梯度作為新的樣本標籤
  - 並用前一個弱學習器的預測結果作為輸入特徵，進行下一個弱學習器的訓練
  - 每一輪訓練後，Gradient Boosting將新的弱學習器的預測結果與前面所有弱學習器的預測結果相加，從而得到更好的集成模型

根據一個人的身高來預測他的體重

1. 初始化模型
   1. 初始化一個弱學習器（例如一個決策樹）
   2. 將其擬合到訓練數據集上。
   3. 這是第一個弱學習器，預測結果可能與真實體重存在誤差
2. 計算殘差
   1. 計算每個樣本的預測誤差，即預測體重和真實體重之間的差異（殘差）
   2. 預測過高的樣本，其殘差為負數，預測過低的樣本，其殘差為正數
3. 訓練下一個弱學習器
   1. 初始化另一個弱學習器，並將其擬合到殘差上
   2. 讓第二個弱學習器學習如何根據第一個弱學習器的預測結果來進行修正
4. 加總預測結果
   1. 連續訓練更多的弱學習器
   2. 每個弱學習器都在前一個弱學習器的殘差上進行優化
   3. 最終將所有弱學習器的預測結果加總起來，得到最終的集成模型

在Gradient Boosting中
每個弱學習器的預測結果並不是直接相加
通過乘以學習率（learning rate）的值來控制其貢獻

- 預測房價訓練數據：
- 第一個弱學習器的預測結果：1.0,2.0,3.0,4.0,5.0
- 第二個弱學習器的預測結果：0.5,1.5,2.5,3.5,4.5
- 假設學習率（learning rate）為0.1
- 將每個弱學習器的預測結果乘以學習率，得到如下：
- 第一個弱學習器加權預測結果：
- 0.1,0.2,0.3,0.4,0.5
- 第二個弱學習器的加權預測結果：
- 0.05,0.15,0.25,0.35,0.45
- 將這兩個弱學習器的加權預測結果相加，得到最終的預測結果：
- 0.15,0.35,0.55,0.75,0.95
- 最終的預測結果 0.15,0.35,0.55,0.75,0.95 就是Gradient Boosting模型對於這五個樣本的預測結果。

學習速率（learning rate）

- 在Gradient Boosting中
  - 每個弱學習器的學習速率（learning rate）是一樣的
  - 一個全局的超參數，用來控制每個弱學習器的貢獻程度
  - 學習速率是在算法開始訓練之前由使用者指定在整個訓練過程保持不變。
- 當訓練Gradient Boosting模型時
  - 每個弱學習器的預測結果都會乘以學習速率
  - 然後將所有弱學習器的結果相加得到最終的預測結果
  - 學習速率的值通常是一個介於0和1之間的小數
  - 較小的學習速率意味著每個弱學習器的貢獻變小，模型訓練過程會更加穩定
  - 較大的學習速率可能使模型更快地收斂，但也容易導致過度擬合。
- 訓練Gradient Boosting模型時
  - 使用者需要根據數據集的大小和複雜程度、問題的性質以及計算資源的情況等因素，選擇合適的學習速率值
  - 選擇學習速率是一個重要的過程，通常通過交叉驗證等技術來選擇最佳的學習速率參數，以獲得最好的模型性能

Gradient Boosting

- Gradient Boosting使用梯度提升演算法
- 邏輯斯回歸或神經網路的訓練採用梯度下降
  - 兩者都是在每一輪反覆運算中利用損失函數相對於模型的負梯度方向的資訊來對當前模型進行更新
  - 在梯度下降中，模型是以參數化形式表示，模型更新等價于參數更新
  - 在梯度提升中，模型不需要進行參數化表示，直接定義在函數空間中擴展了可以使用的模型種類

Boosting: Summary

- Originally developed by computational learning theorists – Schapire, 1990 (*weak learner*).
- Revised to be a practical algorithm, *AdaBoost*, for building ensembles that empirically improves generalization performance - Freund & Shapire, 1996.

AdaBoost - key insights:

- Instead of sampling (as in bagging) re-weigh examples!
- Needs smaller number of training samples than bagging
- Final classification based on weighted vote of weak classifiers

Boosting

- Advantages of boosting
  - Flexibility in the choice of weak learners
  - Testing is fast
  - Easy to implement
  - Integrates classification with feature selection
  - Complexity of training may be linear instead of quadratic in the number of training samples

- Disadvantages
  - Often doesn’t work for many-class problems

Ensemble Learning: A Reflection

- Combining models adds complexity
  - More difficult to characterize, anticipate predictions, explain predictions, etc.
  - But accuracy may increase
- Violation of Ockham's Razor
  - "simplicity leads to greater accuracy"
  - Identifying the best model requires identifying the proper "model complexity"

scikit-learn中的boosting方法

- AdaBoost (Adaptive Boosting)：
  - 通過連續訓練多個弱學習器，每次調整樣本權重，專注於前一個弱學習器分類錯誤的樣本
  - 並結合多個弱學習器的預測結果，使得最終預測更準確
- Gradient Boosting：
  - 通過連續訓練多個弱學習器，每次優化前一個弱學習器的損失函數負梯度方向
  - sklearn中的GradientBoostingClassifier用於分類問題，GradientBoostingRegressor用於回歸問題
- XGBoost (Extreme Gradient Boosting)：
  - XGBoost是一種基於Gradient Boosting算法的高效實現
  - 通過樹的並行計算和選擇性的列存儲來加快訓練速度，並具有較好的化性能
  - sklearn中的XGBClassifier用於分類問題，XGBRegressor用於回歸問題
- LightGBM：
  - LightGBM是另一種基於Gradient Boosting算法的高效實現
  - 使用基於直方圖的算法來進行分桶（binning）和加速訓練過程
  - sklearn中的LGBMClassifier用於分類問題，LGBMRegressor用於回歸問題
- CatBoost：
  - CatBoost是俄羅斯Yandex公司開發的一種基於梯度提升算法的高性能機器學習庫
  - sklearn中的CatBoostClassifier用於分類問題，CatBoostRegressor用於回歸問題