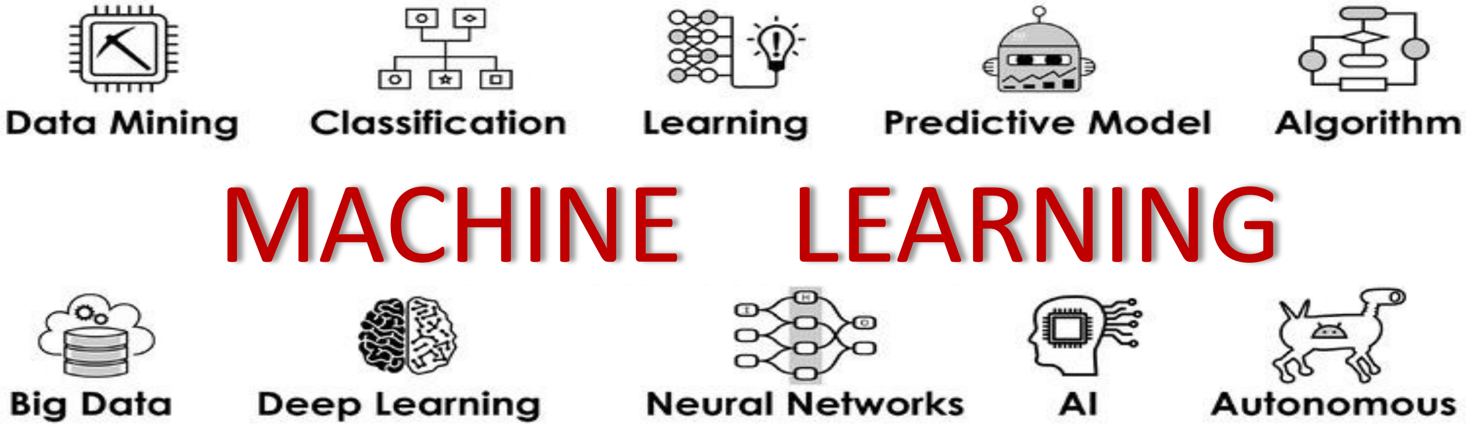
KASETSART UNIVERSITY
FACULTY OF SCIENCE AT SRIRACHA



Support Vector Machine

01418496 Selected Topic in Computer Science

Chalothon Chootong (Ph.D.)

Department of Computer Science and Information, Faculty of Science at Sriracha, Kasetsart University Sriracha Campus

Supervised Learning

Classification

Classification is about predicting a class or discrete values
Eg: Male or Female; True or False

- Logistic Regression
- Decision Tree
- Random Forest
- K-Nearest Neighbors
- Support Vector Machine Classifier
- Naïve Bayes Classifier

Regression

Regression is about predicting a quantity or continuous values
Eg: Salary; age; Price.

- Linear Regression
- Polynomial Regression
- Random Forest Regressor
- Support Vector Machine Classifier
- Bayesian Linear Regressor

Classification:

Dog

Cat

(Dog or Cat)

Regression:

Temperature

Rainfall in cm

Rainfall in cm

Unsupervised Learning

S

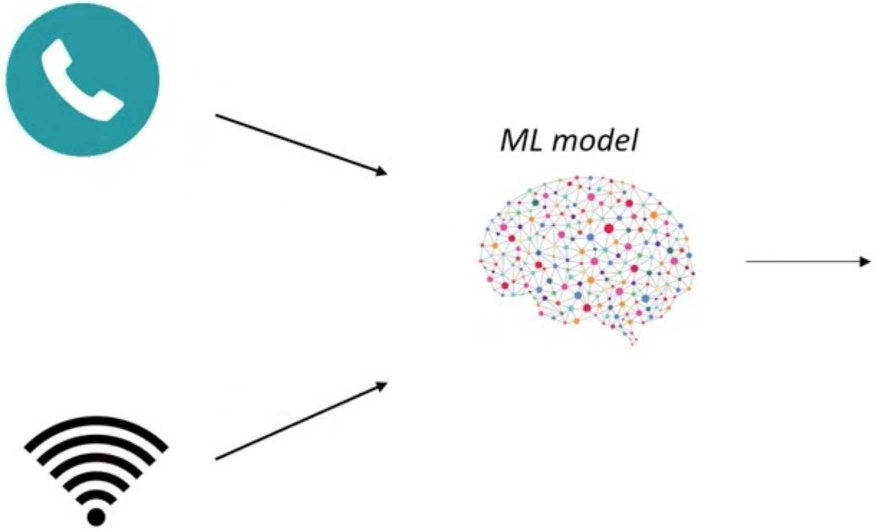
Clustering

Clustering is an unsupervised task which involves grouping the similar data points.

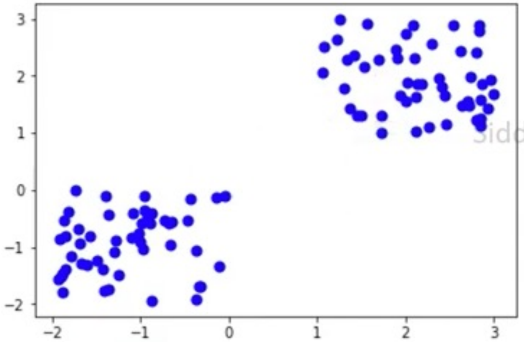
- K-Means Clustering
- Hierarchical Clustering
- Principal Component Analysis (PCA)

Association

Association is an unsupervised task that is used to find important relationship between data points



Clustering



Association

Customer 1

- *Bread*
- *Milk*
- *Fruits*
- *wheat*

Customer 2

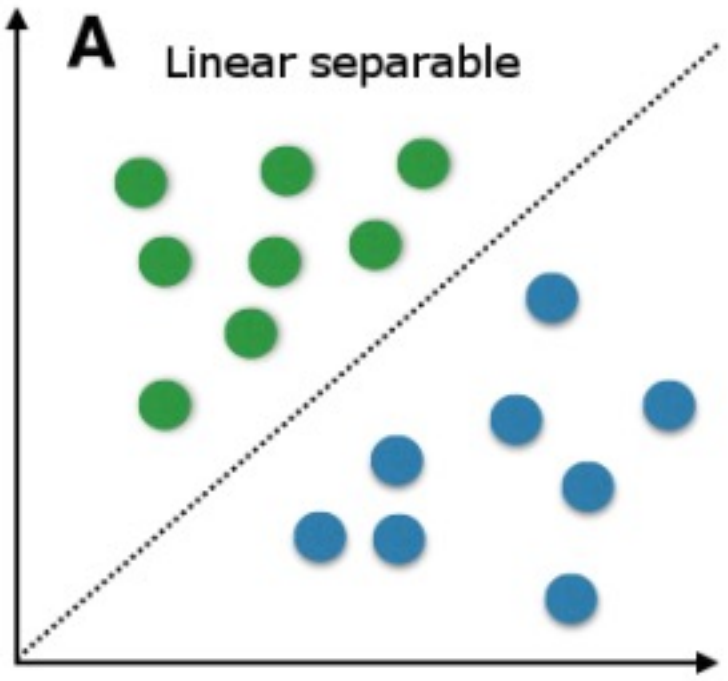
- *Bread*
- *Milk*
- *Rice*
- *Butter*

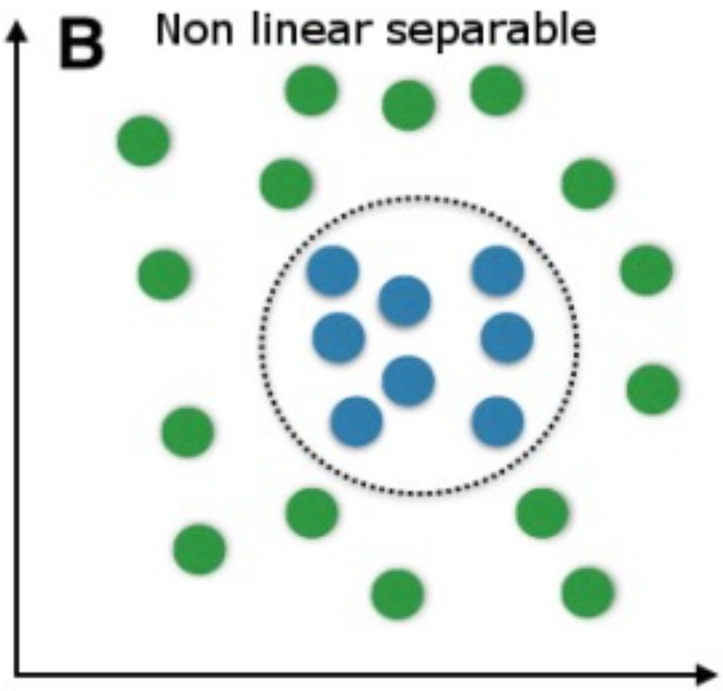
Customer 3

Now, when customer 3 goes and buys bread, it is highly likely that he will also buy milk.

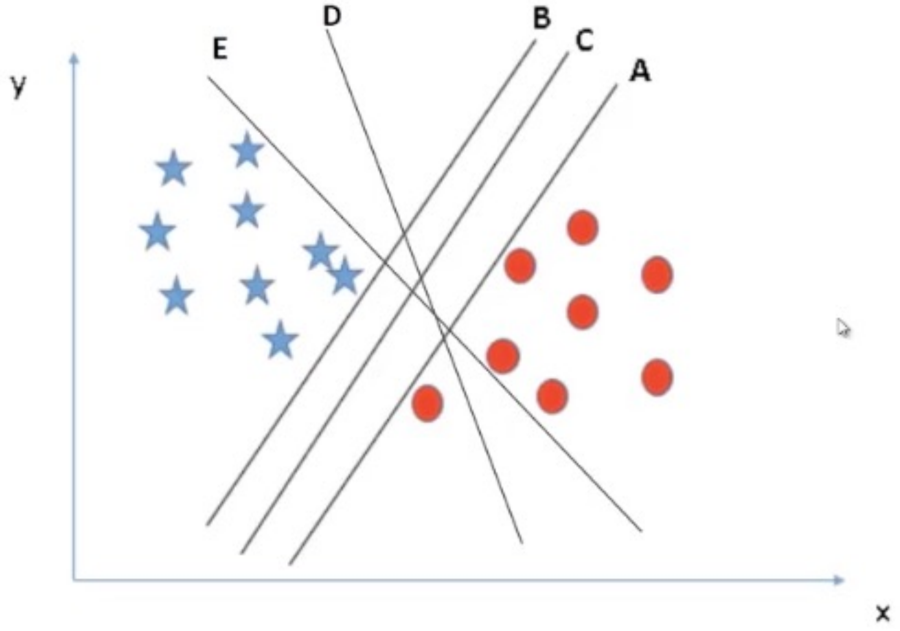
Support Vector Machines (SVM) ?

- Support Vector Machine จะเป็นการจัดกลุ่มข้อมูล Classification โดยการแบ่ง Class ของข้อมูลออกจากกัน ซึ่งสามารถใช้การแบ่งด้วยสมการเชิงเส้นได้ทั้ง Linear และ Non Linear



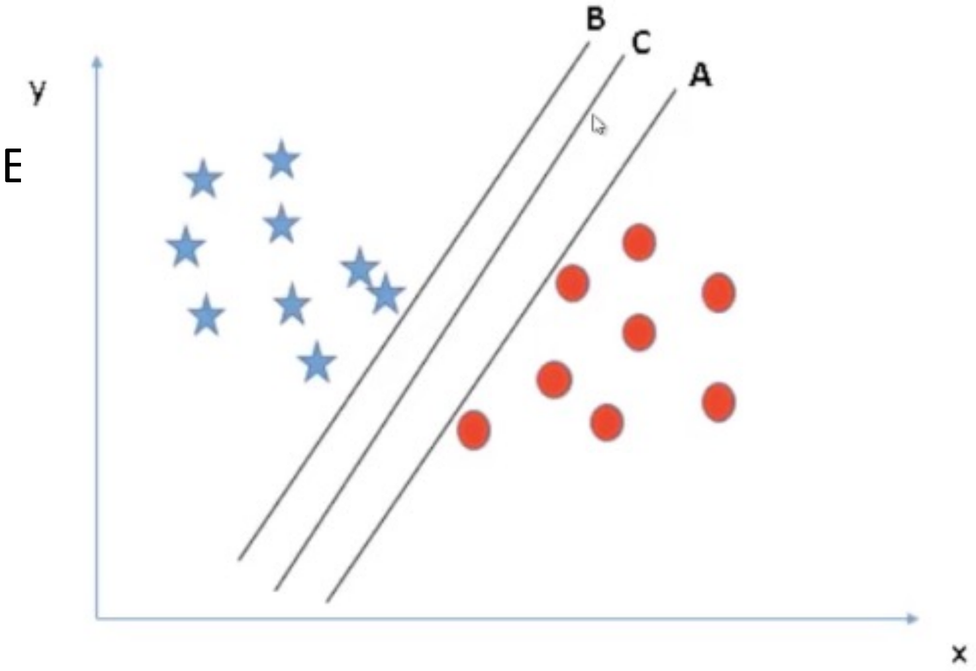


Support Vector Machines (SVM) ?

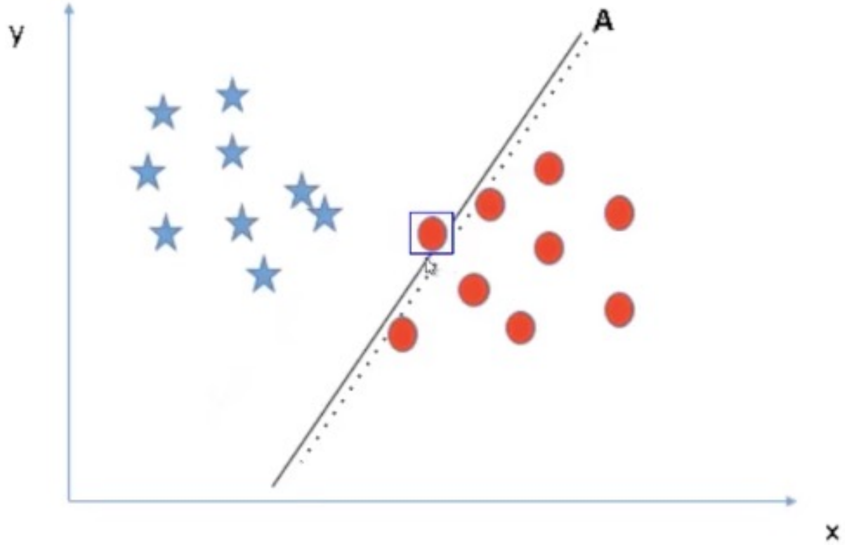


100% classification (line E and D will be removed)

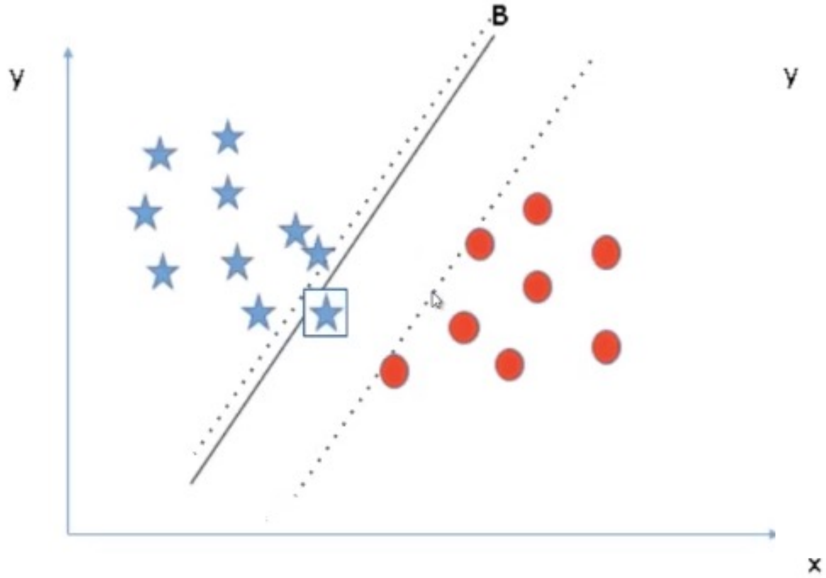
Which line is Best Separation?



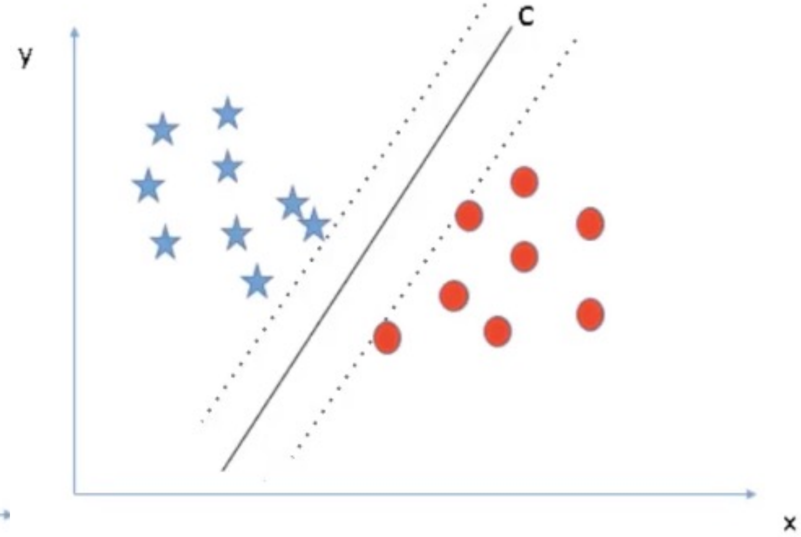
Support Vector Machines (SVM) ?



Margin of line A



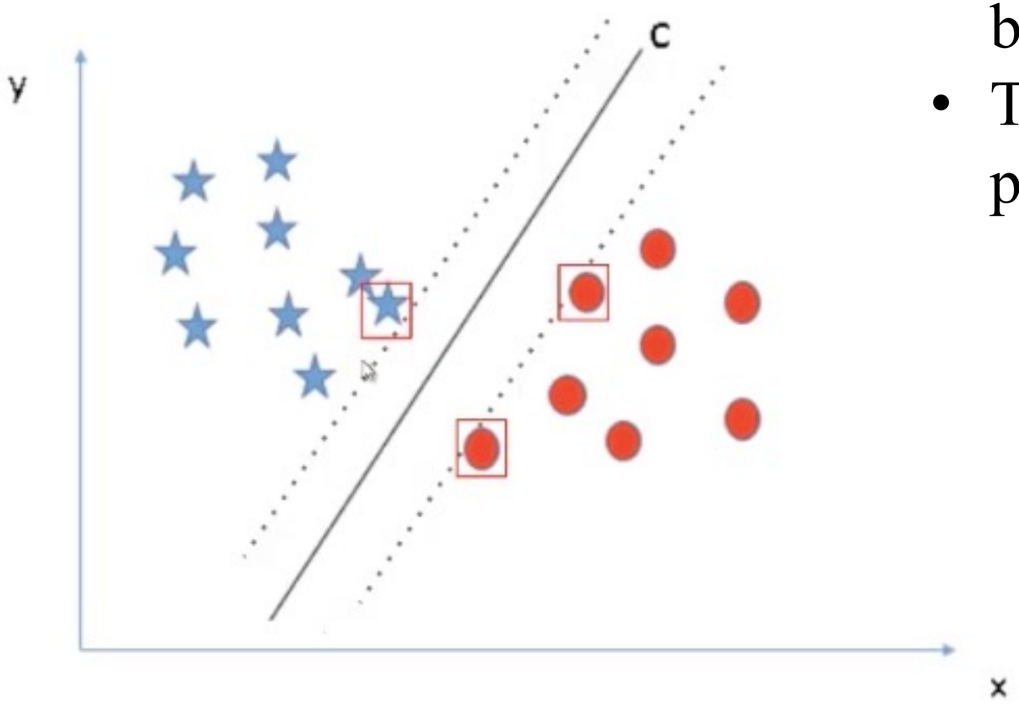
Margin of line B



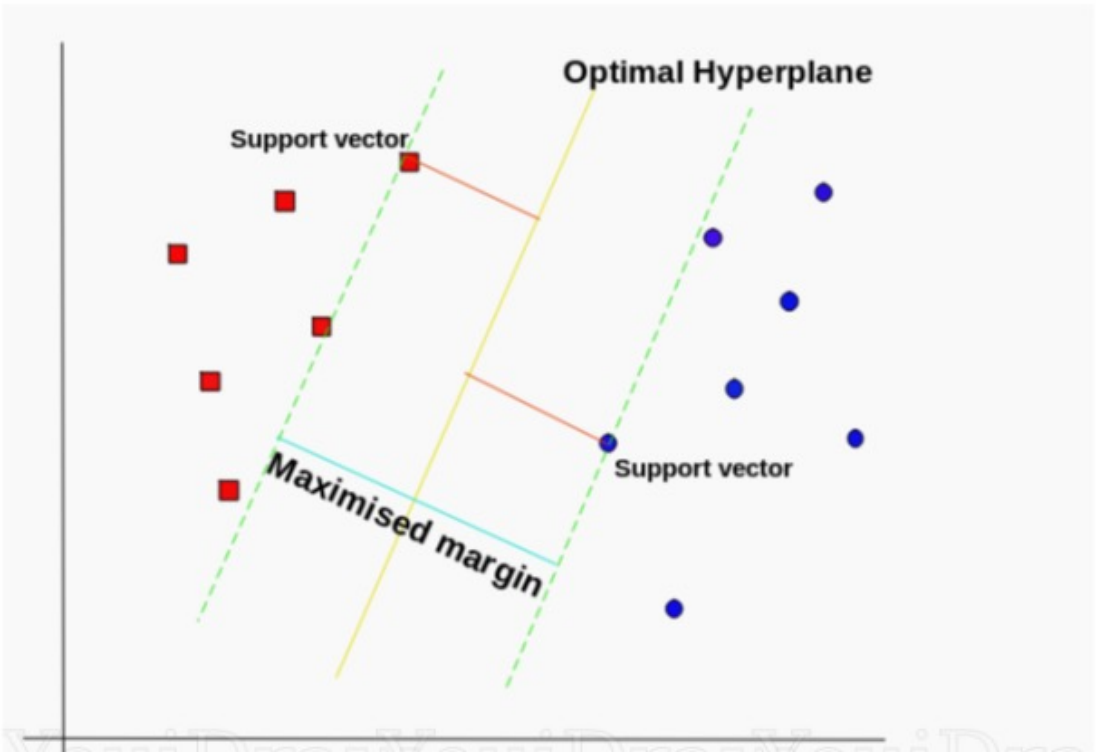
Margin of line C

So Line C present the maximum margin

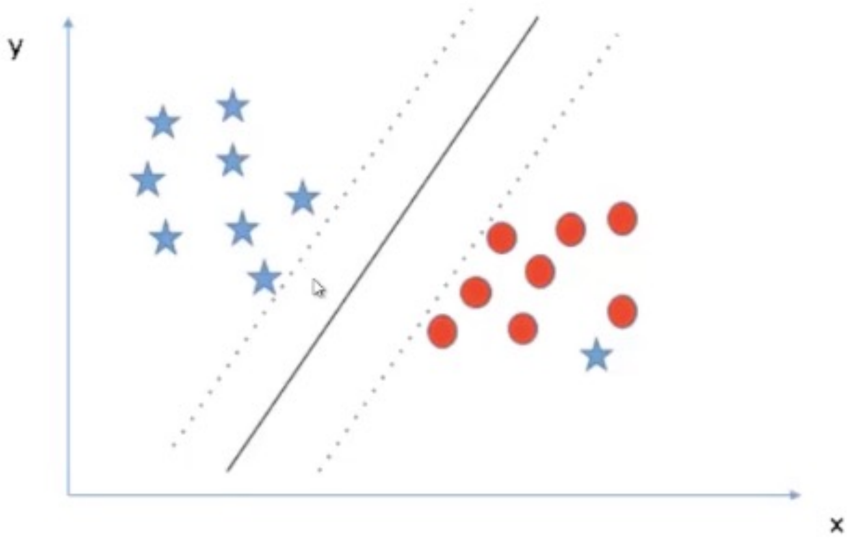
Support Vector Machines (SVM) ?

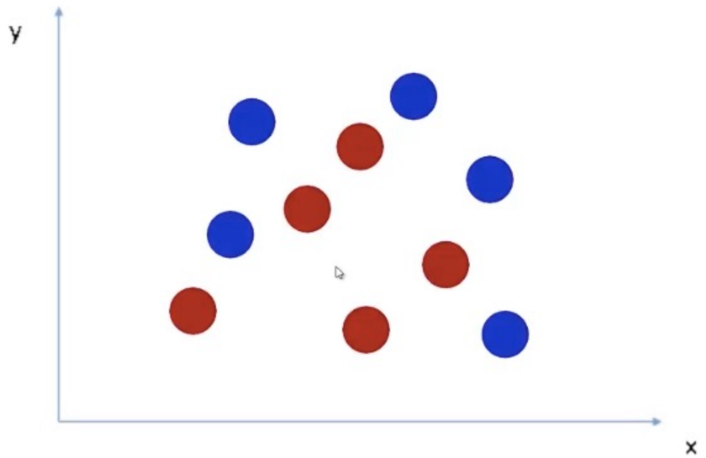


- the margin for hyper-plane C is high as compared to both A and B
- The point that help us to identify the right hyper-plane they are called "Support Vector"



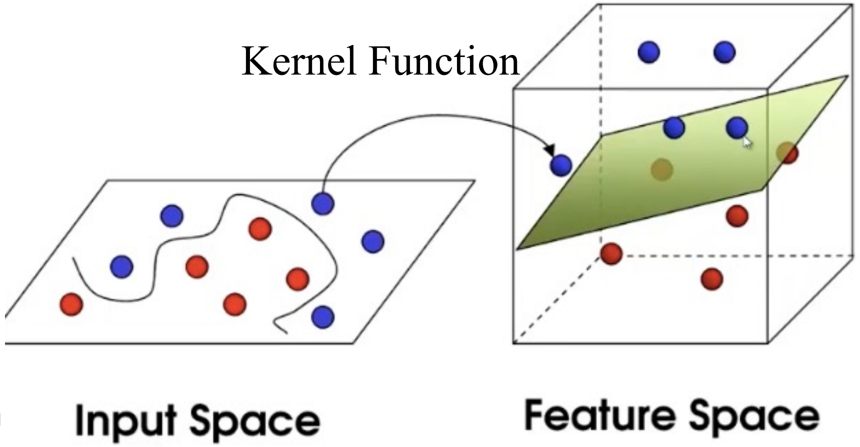
100% classification → Maximum Classification





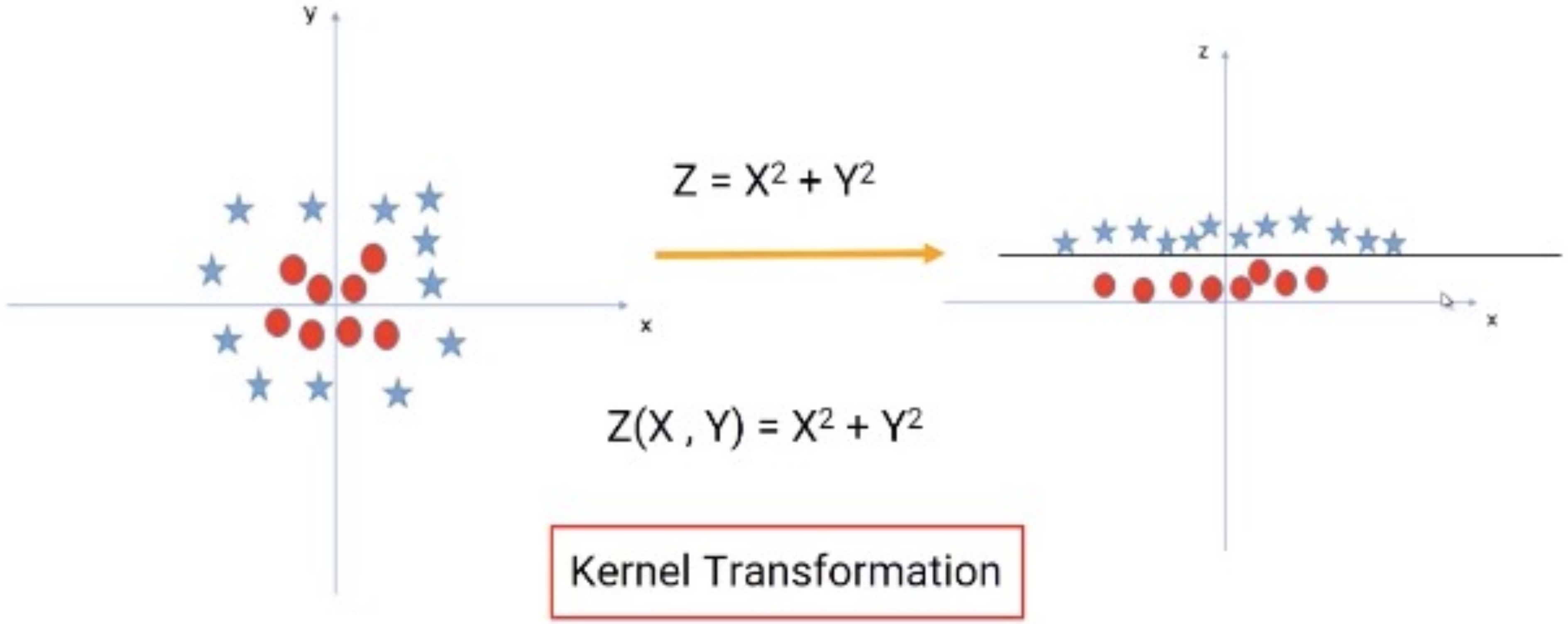
- In the scenario below, we can't have linear hyperplane between the two classes,
- How does SVM classify these two classes?

- SVM can solve this problem
- It solves this problem by introducing an additional feature.



Projecting to Higher Dimension

- An additional feature ($z = x^2+y^2$)



SVM with python

```
import pandas as pd
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
………

model = SVC( )
model.fit(train_x,train_y)
```

How to tune Parameters of SVM?

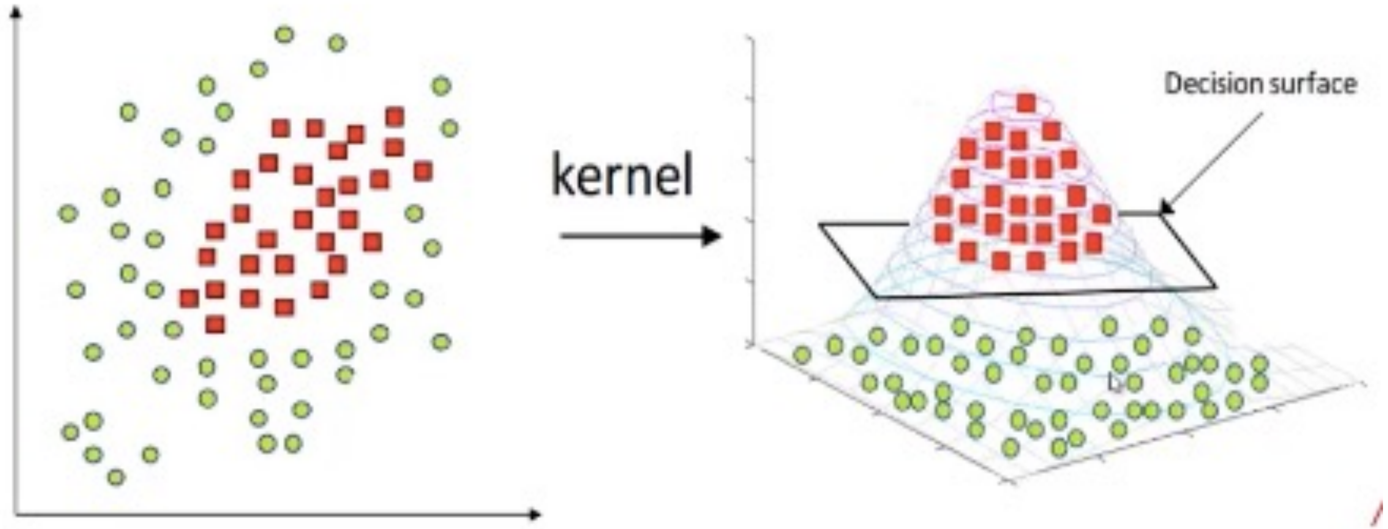
sklearn.svm.SVC(***C=1.0*, *kernel='rbf'*, *degree=3*, *gamma=0.0***, *coef0=0.0*, *shrinking=True*, *probability=False*,*tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=-1*, *random_state=None*)

SVM with python

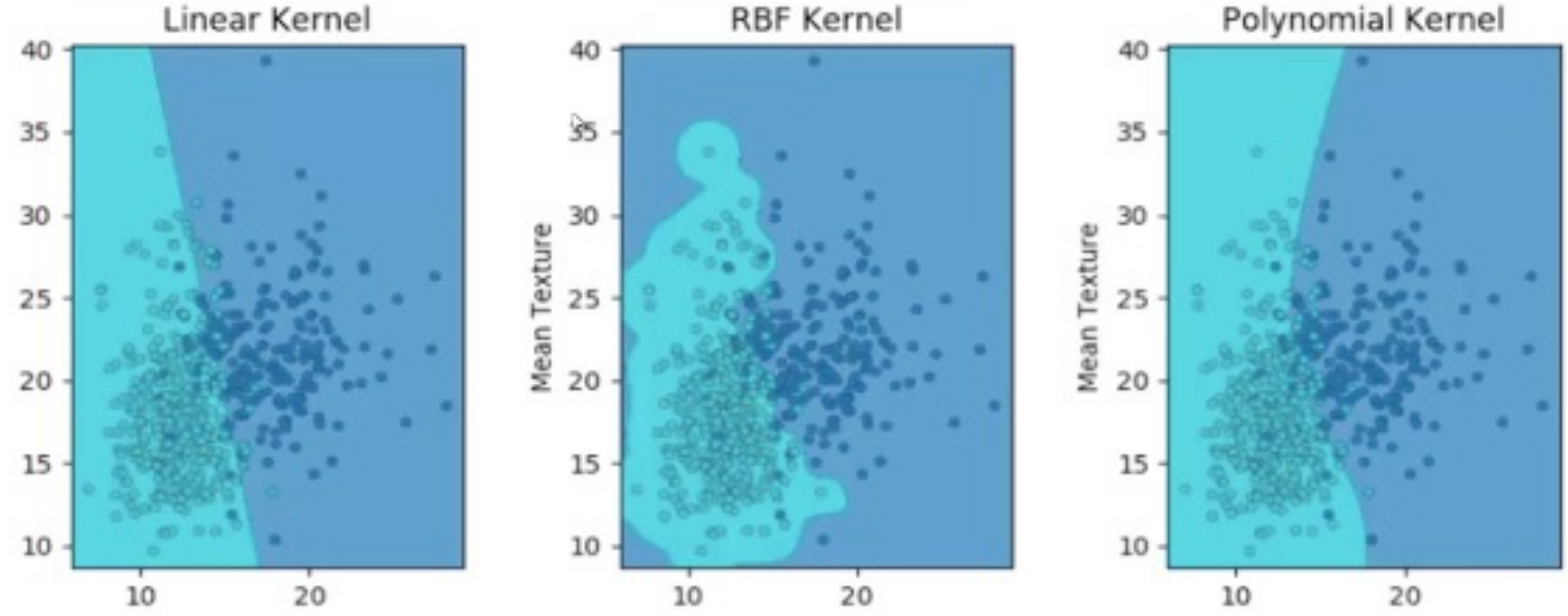
- *Kernel*
  - There are various options available with kernel: “linear”, “rbf”,”poly” and others (default value is “rbf”).
  - “rbf” and “poly” are useful for non-linear hyper-plane.

- **Radial Basic Function (rbf):**

$$Z(X, Y) = \exp\left(-\frac{\| X - Y \|^2}{2\sigma^2}\right)$$



The example shape of each kernel type

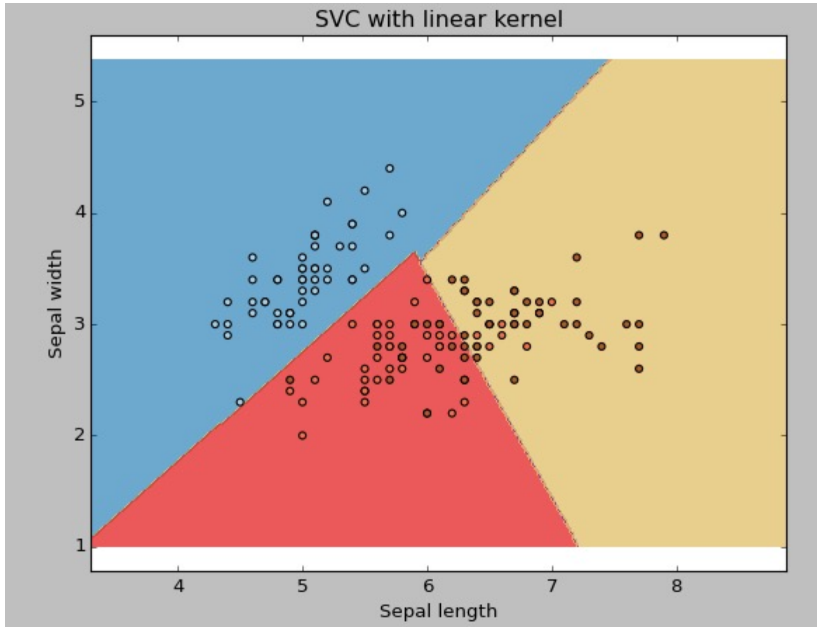


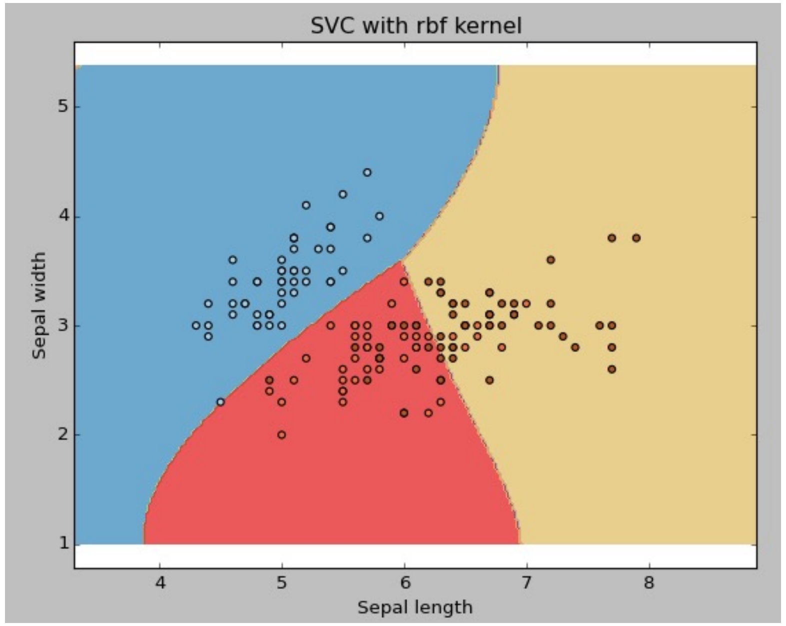
https://youtu.be/3liCbRZPrZA

SVM with python

```
svc = svm.SVC(kernel='linear', C=1,gamma=0)
scv.fit(X, y)
```

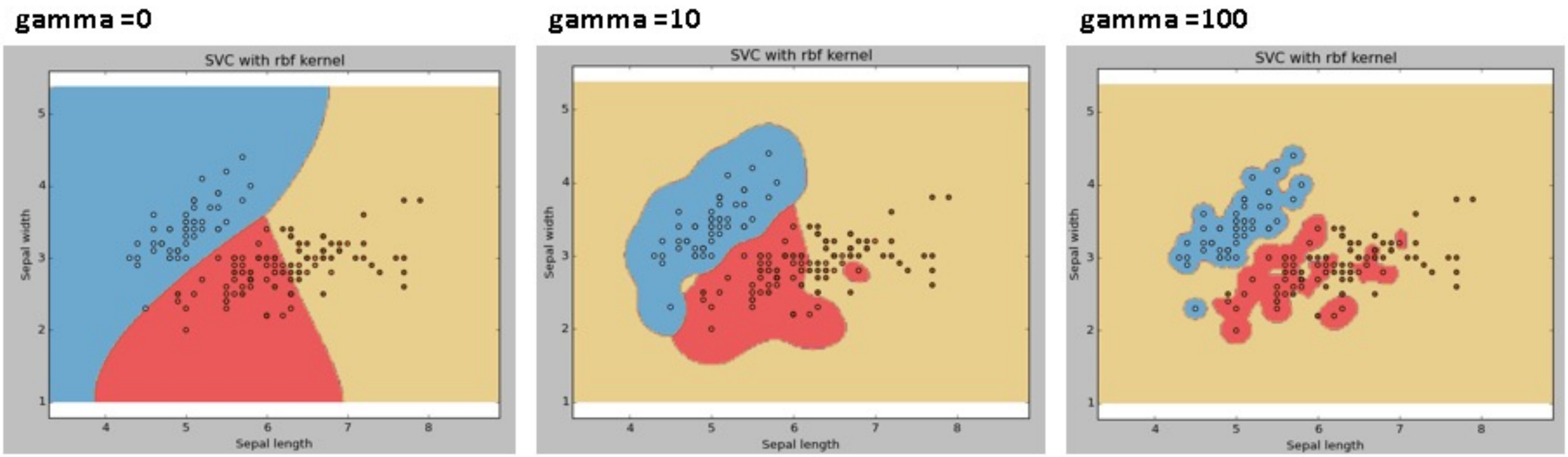
```
svc = svm.SVC(kernel='rbf', C=1,gamma=0)
Svc.fit(X, y)
```





SVM with python

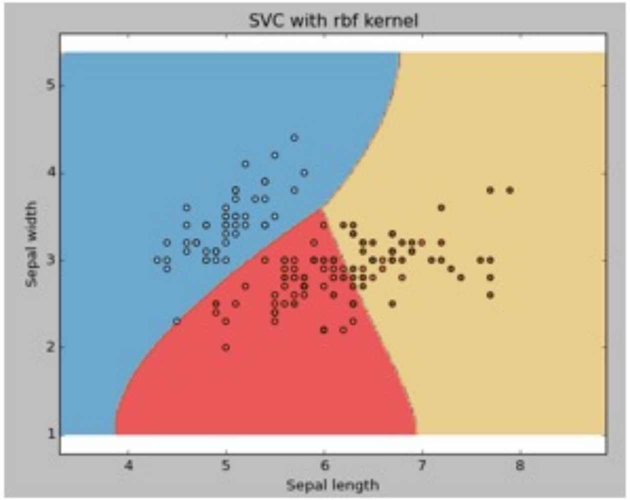
- **gamma:** Higher the value of gamma, will try to exact fit the as per training data set.



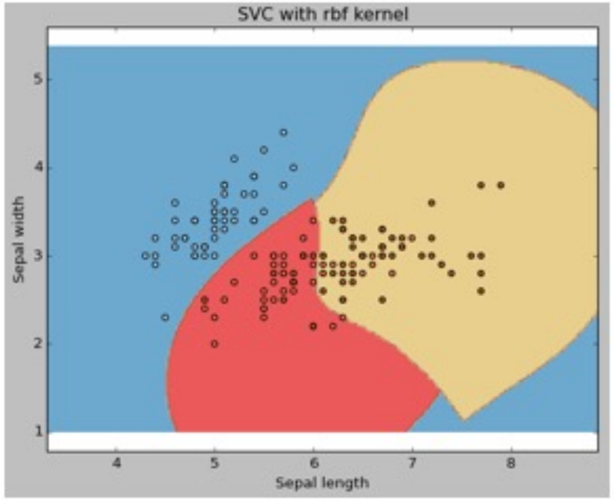
SVM with python

- **C:** Penalty parameter C of the error term. It also controls the trade-off between smooth decision boundaries and classifying the training points correctly.

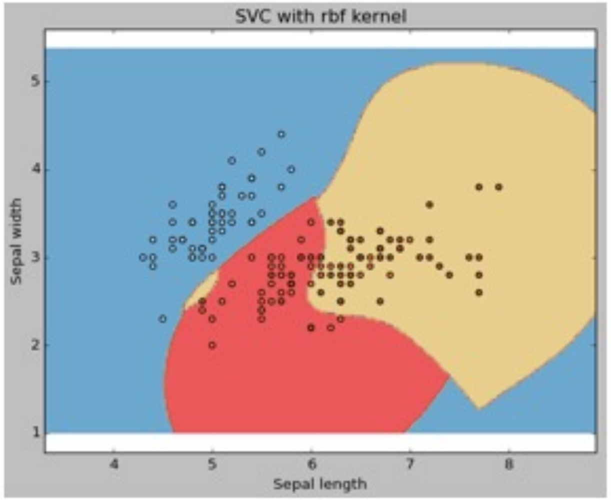
c =1



C =100

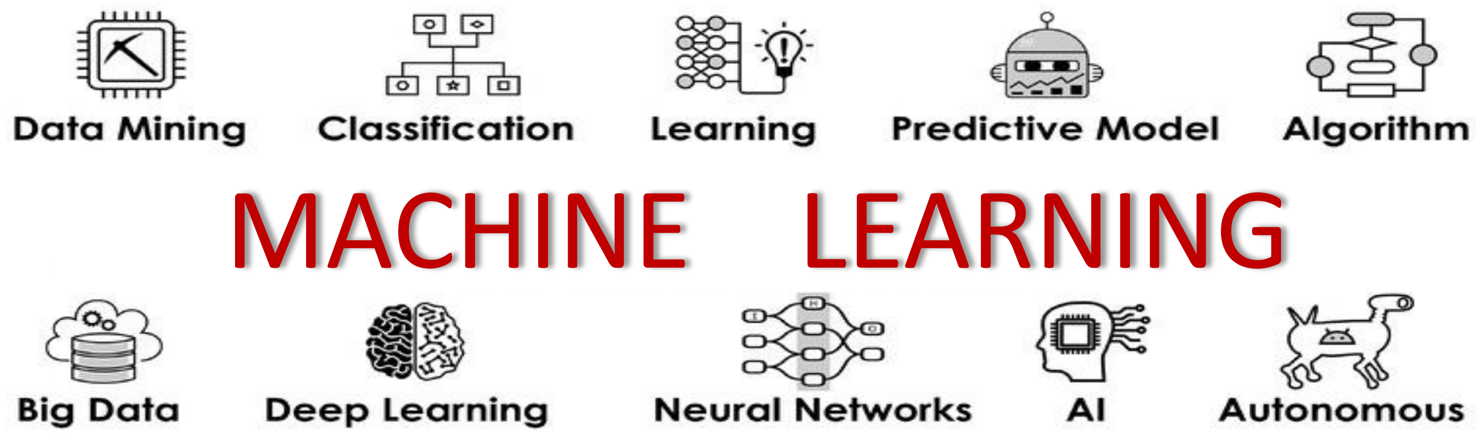


c =1000



KASETSART UNIVERSITY

FACULTY OF SCIENCE AT SRIRACHA



Naïve Bay / Neural Network

01418496 Selected Topic in Computer Science

Chalothon Chootong (Ph.D.)

Department of Computer Science and Information, Faculty of Science at Sriracha, Kasetsart University Sriracha Campus

Naive Bayes

- Naive Bayes is a simple but surprisingly powerful algorithm for predictive modelling.
- Selecting the best hypothesis (h) given data (d).
- The easiest ways of selecting the most probable hypothesis given the data is using Bayes' Theorem.

$$P\left(\frac{H}{D}\right) = \frac{P(D|H) * P(H)}{P(D)}$$

P(H): คือค่าความน่าจะเป็นที่สมมุติฐาน H จะเป็นจริง

P(D): คือค่าความน่าจะเป็นของข้อมูล D

P(D|H): คือค่าความน่าจะเป็นของข้อมูล D ที่จะทำให้สมมุติฐาน H เป็นจริง

P(H|D): คือค่าความน่าจะเป็นของสมมุติฐาน ที่ทำให้ข้อมูล D เป็น จริง

Naive Bayes Classifier

- Prediction of membership probabilities is made for every class.
- Calculate the probability of data points that associate to a particular class.
- The class having maximum probability is appraised as the most suitable class.
- This is also referred as Maximum A Posteriori (MAP).
  - This can be written as:

★ $$\text{MAP}(h) = \max(P(h|d))$$

Or $$\text{MAP}(h) = \max((P(d|h) * P(h)) / P(d))$$

Or $$\text{MAP}(h) = \max(P(d|h) * P(h))$$

P(d) ใช้สำหรับการ Normalization โดยเราสามารถตัดทิ้งได้ ถ้าเราให้ความสนใจที่ สมมุติฐาน H อย่างเดียว

Representation Used By Naive Bayes Models

- A list of probabilities are stored to file for a learned Naive Bayes model:
  - Class Probabilities: The probabilities of each class in the training dataset.
  - Conditional Probabilities: The conditional probabilities of each input value given each class value.

- **Calculating Class Probabilities**, For example in a binary classification, the probability of class 1 can find by

$$P_{class(1)} = \frac{Count_{class(1)}}{Count_{calss(0)} + Count_{class(1)}}$$

Representation Used By Naive Bayes Models

- **Calculating Conditional Probabilities**
  - If a "*weather*" attribute had the values "*sunny*" and "*rainy*" and the class attribute had the class values "*go-out*" and "*stay-home*",
  - then the conditional probabilities of each weather value for each class value could be calculated as:

$$P_{(weather=sunny|class=go-out)} = \frac{Count_{weather=sunny\ and\ class=go-out}}{Count_{class=go-out}}$$

$$P_{(weather=sunny|class=stay-home)} = \frac{Count_{weather=sunny\ and\ class=stay-home}}{Count_{class=stay-home}}$$

Make Predictions With a Naive Bayes Model

- If we had a new instance with the *weather of sunny*, we could make predictions for new data using Bayes theorem.

MAP(h) = max(P(d|h) * P(h))

$$Class_{(go\_out)} = \frac{P(weather = sunny|class = go\_out)}{P(class = go - out)}$$

$$Class_{(stay\_home)} = \frac{P(weather = sunny|class = stay\_home)}{P(class = stay\_home)}$$

$$P(go\_out|weather = sunny) = \frac{go\_out}{(go\_out + stay\_home)}$$

$$P(stay\_home|weather = sunny) = \frac{stay\_home}{(go\_out + stay\_home)}$$

We can choose the class that has the largest calculated value.

The example

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

- Out of 1000 records in training data, you have 500 Bananas, 300 Oranges and 200 Others.
  - P(Y=Banana) = 500 / 1000 = 0.50
  - P(Y=Orange) = 300 / 1000 = 0.30
  - P(Y=Other) = 200 / 1000 = 0.20
- Compute the probability of evidence
  - P(x1=Long) = 500 / 1000 = 0.50
  - P(x2=Sweet) = 650 / 1000 = 0.65
  - P(x3=Yellow) = 800 / 1000 = 0.80

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

- Compute the probability of likelihood of evidences
  - P(x1=Long | Y=Banana) = 400 / 500 = 0.80
  - P(x2=Sweet | Y=Banana) = 350 / 500 = 0.70
  - P(x3=Yellow | Y=Banana) = 450 / 500 = 0.90

So, the overall probability of Likelihood of evidence for Banana = 0.8 * 0.7 * 0.9 = 0.504

- If fruit is ‘long’, ‘sweet’, and ‘yellow’, what fruit is it?

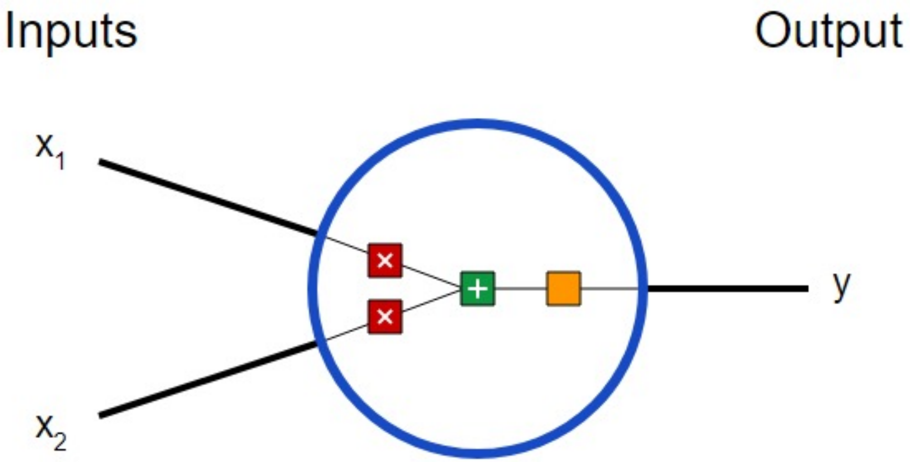
$$P(Banana|Long, Sweet\ and\ Yellow) = \frac{P(Long|Banana) * P(Sweet|Banana) * P(Yellow|Banana) * P(Banana)}{P(Long) * P(Sweet) * P(Yellow)}$$

$$= \frac{0.8*0.7*0.9*0.5}{P(Evidence)} = \frac{0.252}{P(Evidence)}$$

$$P(Orenge|Long, Sweet\ and\ Yellow) = 0\ , because\ P(Long|Orange) = 0$$

$$P(Other\ Fruit|Long, Sweet\ and\ Yellow) = \frac{0.01875}{P(Evidence)}$$

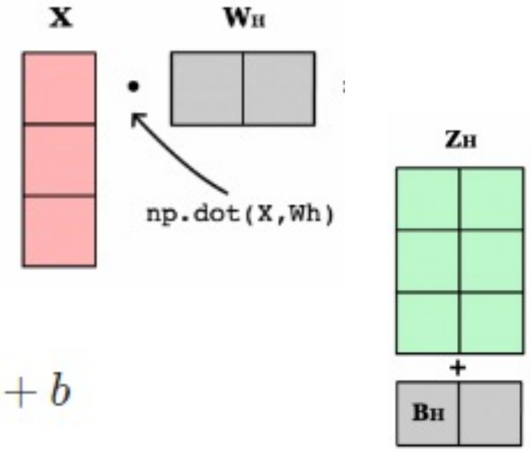
Basic Neural Networks



- Neural Network หรือ Artificial Neural Network คือ โครงข่ายประสาทเทียม เป็นแนวคิดที่ออกแบบระบบโครงข่ายคอมพิวเตอร์ ให้เลียนแบบการทำงานของสมองมนุษย์
- ถ้า เรามี 2 input, neural network ก็จะมีลักษณะดังภาพ
  - แต่ละ input จะถูกคูณกับ weight

$$x_1 \rightarrow x_1 * w_1$$

$$x_2 \rightarrow x_2 * w_2$$



  - จากนั้นจะถูกบวกกับ bias

$$(x_1 * w_1) + (x_2 * w_2) + b$$

  - จากนั้นหาผลรวมและส่งไปยัง Activation Function

$$y = f(x_1 * w_1 + x_2 * w_2 + b)$$

Activation Function เป็นฟังก์ชันที่รับผลการประมวลผลจากทุก input ภานใน 1 neural แล้วคำนวณว่าจะส่งเป็น Output เท่าไหร่ Activation Function นิยม ได้แก่ ReLU และ Sigmoid

Basic Neural Networks

- Example:
  - กำหนดให้ input x = [2,3] Network จะทำการ คำนวณค่าอย่างไร ถ้า weight เท่ากับ w = [0,1] bias = 0

$$\begin{aligned} h_1 = h_2 &= f(w \cdot x + b) \\ &= f((0 * 2) + (1 * 3) + 0) \\ &= f(3) \\ &= 0.9526 \end{aligned}$$

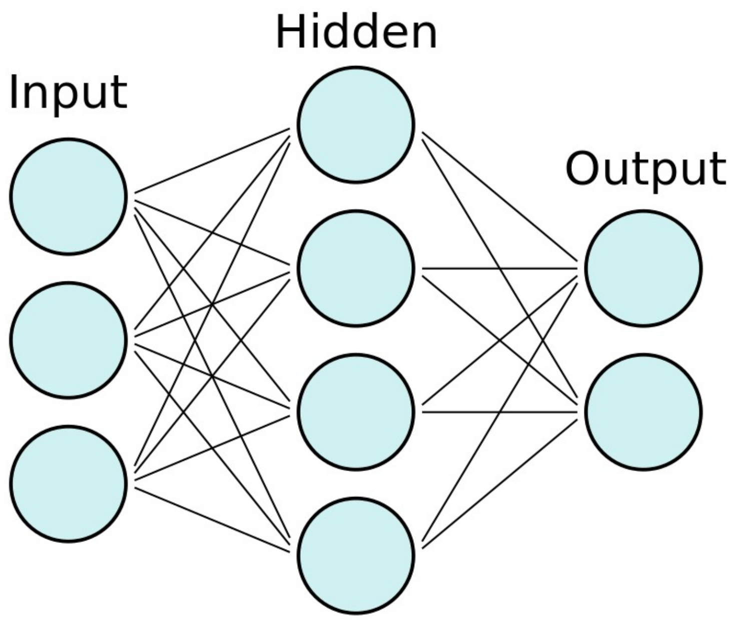
$$\begin{aligned} o_1 &= f(w \cdot [h_1, h_2] + b) \\ &= f((0 * h_1) + (1 * h_2) + 0) \\ &= f(0.9526) \\ &= \boxed{0.7216} \end{aligned}$$

Sentiment Analysis

- **Basic Neural Networks**

$$o_j = f\left(\sum_i w_{i,j} a_i + b_i\right)$$

Neural network formula




```python
#For Neural Network
from keras.models import Sequential
from keras import layers
```

```python
model = Sequential()
model.add(layers.Dense(10, input_dim=input_dim, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='Adam', metrics=['accuracy'])
model.summary()
```

Sentiment Analysis

- **Basic Neural Networks**
  - **Training model**

```
history = model.fit(X_train, y_train,
                    epochs=10,
                    verbose=1,
                    validation_data=(X_test, y_test),
                    batch_size=10)
```

```
Train on 800 samples, validate on 200 samples
Epoch 1/10
800/800 [==============================] - 1s 2ms/step - loss: 0.6863 - accuracy: 0.5725 - val_loss: 0.6716 - val_accuracy:
0.6850
Epoch 2/10
800/800 [==============================] - 0s 606us/step - loss: 0.6187 - accuracy: 0.8200 - val_loss: 0.6259 - val_accurac
y: 0.7550
Epoch 3/10
800/800 [==============================] - 0s 601us/step - loss: 0.5141 - accuracy: 0.8963 - val_loss: 0.5668 - val_accurac
y: 0.7750
Epoch 4/10
800/800 [==============================] - 0s 610us/step - loss: 0.4020 - accuracy: 0.9337 - val_loss: 0.5195 - val_accurac
y: 0.8000
Epoch 5/10
800/800 [==============================] - 0s 610us/step - loss: 0.3091 - accuracy: 0.9563 - val_loss: 0.4875 - val_accurac
y: 0.8050
```

```python
#plot graph
import matplotlib.pyplot as plt
plt.style.use('ggplot')

def plot_history(history):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    x = range(1, len(acc) + 1)

    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.plot(x, acc, 'b', label='Training acc')
    plt.plot(x, val_acc, 'r', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.legend()
    plt.subplot(1, 2, 2)
    plt.plot(x, loss, 'b', label='Training loss')
    plt.plot(x, val_loss, 'r', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend()

plot_history(history)
```

Save, Load and Use model

Save

Load

Use

```
from keras import models

model.save('NN_Sentiment_model')

trained_model = models.load_model("NN_Sentiment_model")

predicted_class = trained_model.predict(X_test[10])
actual_class = y_test[10]
print(predicted_class, actual_class)
```

Neural Networks with Embedded Layer

- Create Model

```
from keras.models import Sequential
from keras import layers
```

```
embedding_dim = 50
model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           input_length=maxlen,
                           output_dim=embedding_dim
                           ))
model.add(layers.Flatten())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

Neural Networks with Embedded Layer

- Train Model

```
history = model.fit(X_train, y_train,
                    epochs=20,
                    verbose=1,
                    validation_data=(X_test, y_test),
                    batch_size=10)
```

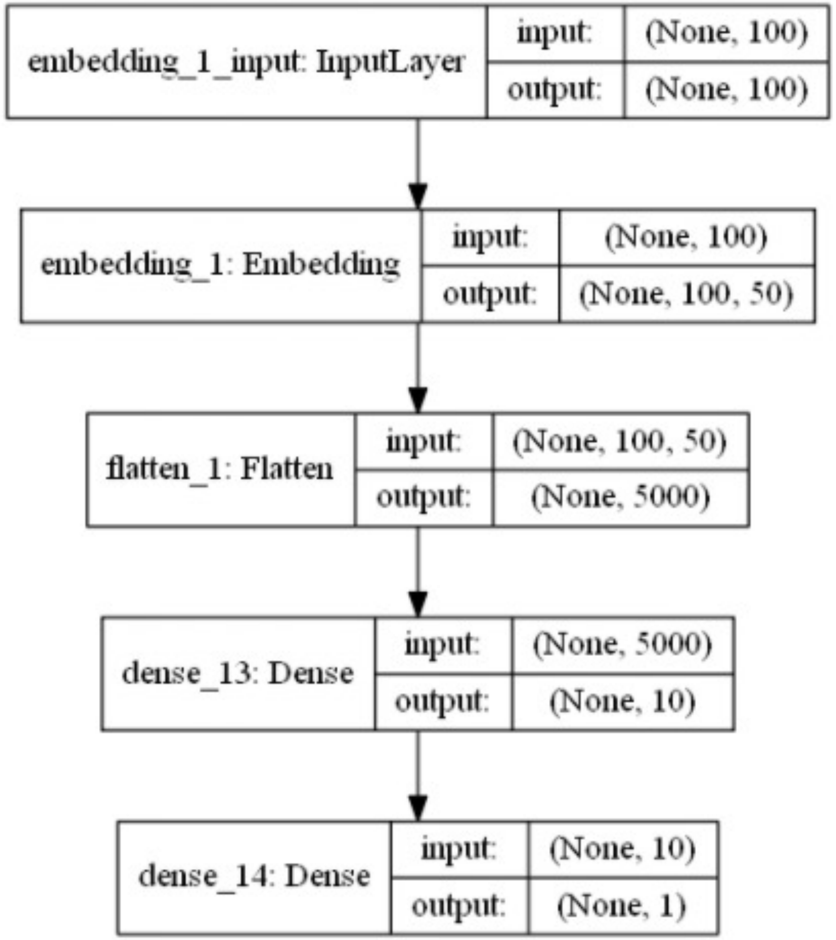
- Print Accuracy and Plot Graph

```
loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
print("Training Accuracy: {:.4f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

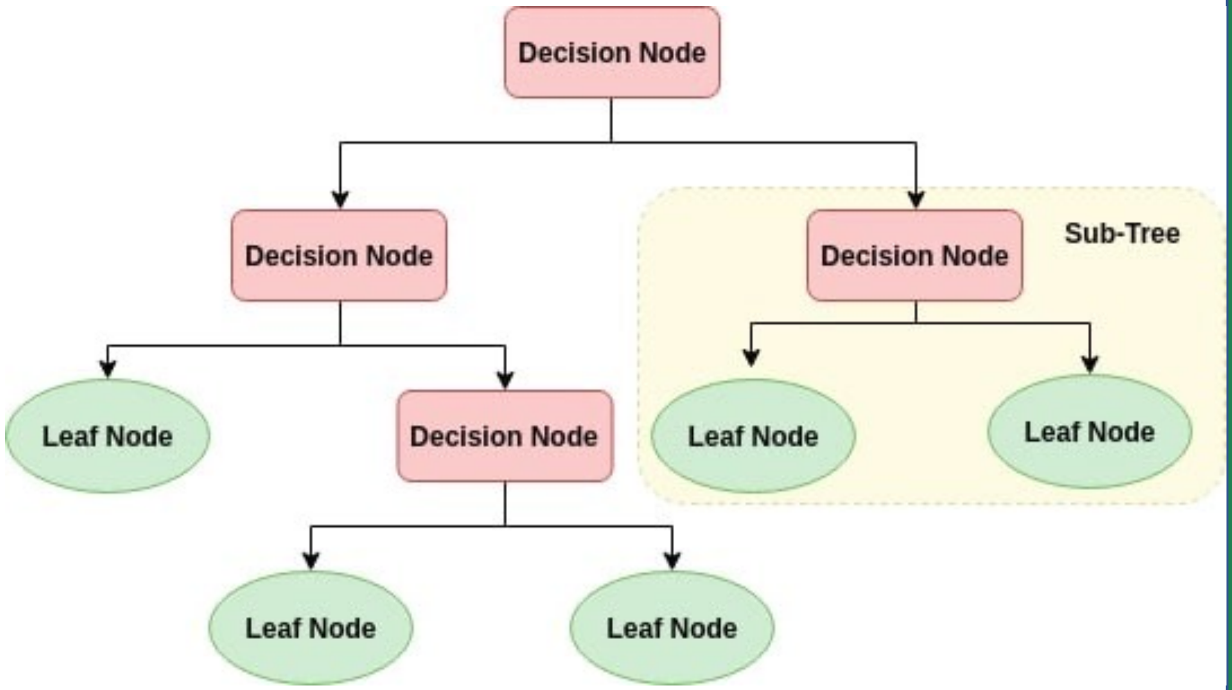
```
plot_history(history)
```

Model Structure

```
from keras.utils import plot_model
plot_model(model, show_shapes=True, to_file='model_structure.png')
```



Decision Tree Algorithm



- A decision tree is a flowchart-like tree structure
  - An internal node represents feature(or attribute)
  - The branch represents a decision rule
  - Each leaf node represents the outcome.
- It learns to partition on the basis of the attribute value.

Decision Tree Algorithm

- Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic.
- Its training time is faster compared to the neural network algorithm.
- The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions.
- There are two types of Decision Tree:
  - Regression Tree
  - Classification Tree