Pattern Recognition

SVM 개념 잡기

Yukyung Choi

yk.choi@rcv.sejong.ac.kr

What is SVM?

- **S**upport **V**ector **M**achine → SVM

- Traditional Classifier

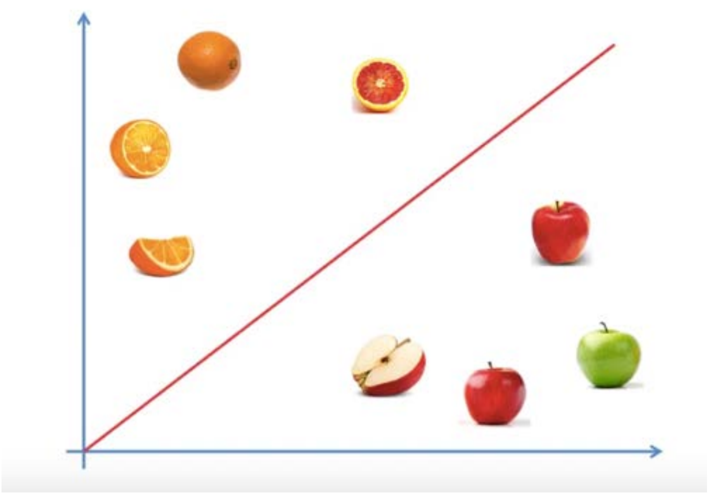
- Until now, favorite classifier to everyone
  - Wondering why? **Kernel Trick!!!**

“만약, 문제에 어떠한 알고리즘을 사용할지 모르겠다면,
SVM은 좋은 출발선이 될 수 있음”

Classifier

- **Classifier** is a hypothesis or discrete-valued function that is used to **assign (categorical) class labels to particular data points**.

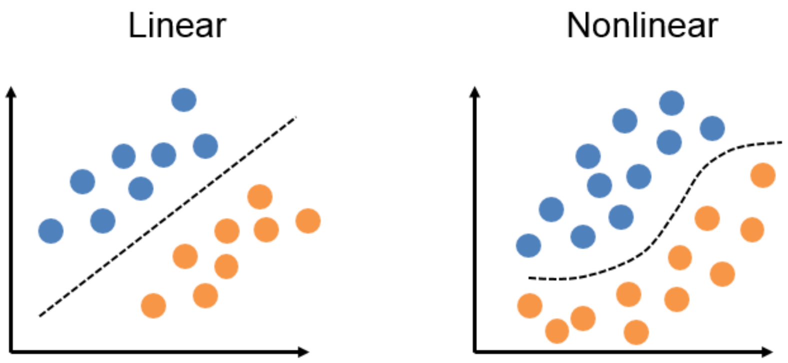
- In the email classification example, this classifier could be a hypothesis for labeling emails as **spam** or **non-spam**.



Classifier

- y = label, x = data, y = f(x), f: classifier

- If **decision function** is linear, this classifier (f) is linear classifier
- If not, this classifier (f) is non-linear classifier



y = f(x)

데이터를 구획해주는 이 **점선의 함수 (decision boundary)**를 우리는 **판별 함수 (decision function)**라 부른다.

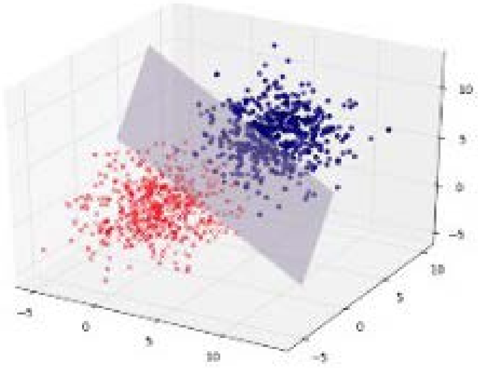
Classifier

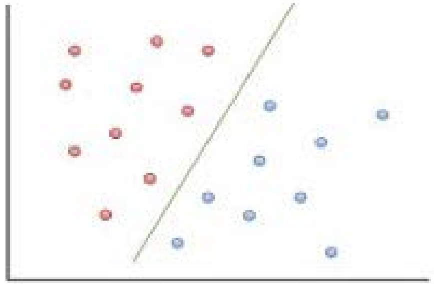
- **Hyperplane**
  - In geometry, a hyperplane is a subspace whose dimension is one less than that of its ambient space. If a space is **3-dimensional** then its hyperplanes are the **2-dimensional planes**, while if the space is **2-dimensional**, its hyperplanes are the **1-dimensional lines**.

$$\mathbf{w}^T\mathbf{x} = 0 \qquad\qquad y = ax + b$$

Hyperplane

Line

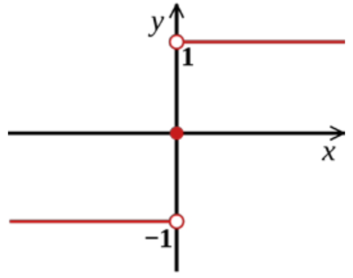




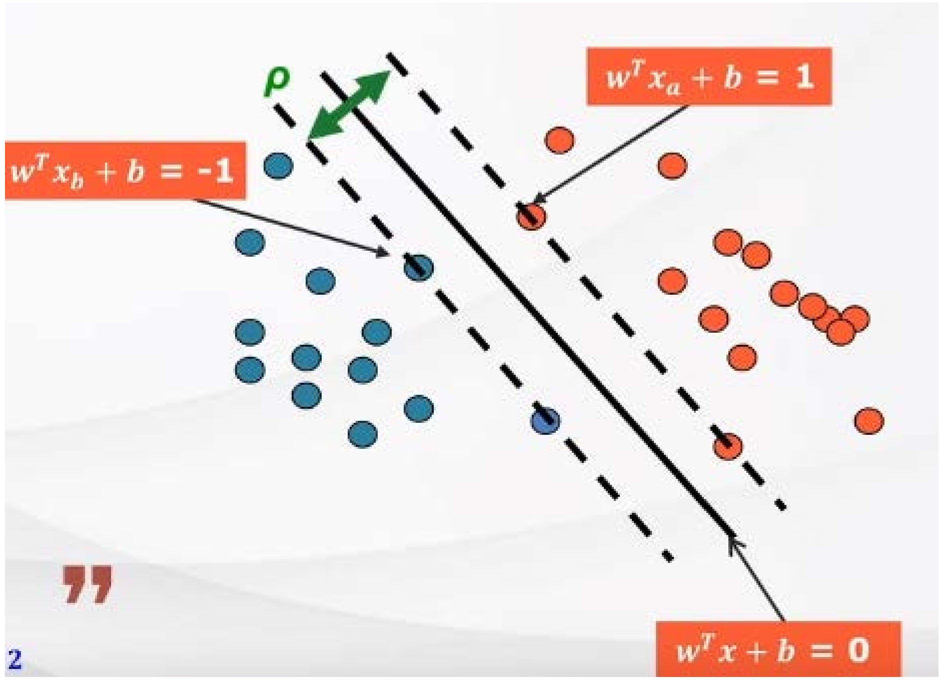
SVM Classifier

- W : vector for hyperplane
- x_i : i_{th} data, y_i : label (class) of i_{th} data

- $\mathbf{Y}$ = **sign**$(W^T\mathbf{X}+b) = f(\mathbf{X})$
  - $Y_i = +1$ when $W^T\mathbf{X}_i+b > 1$
  - $Y_i = -1$ when $W^T\mathbf{X}_i+b < -1$



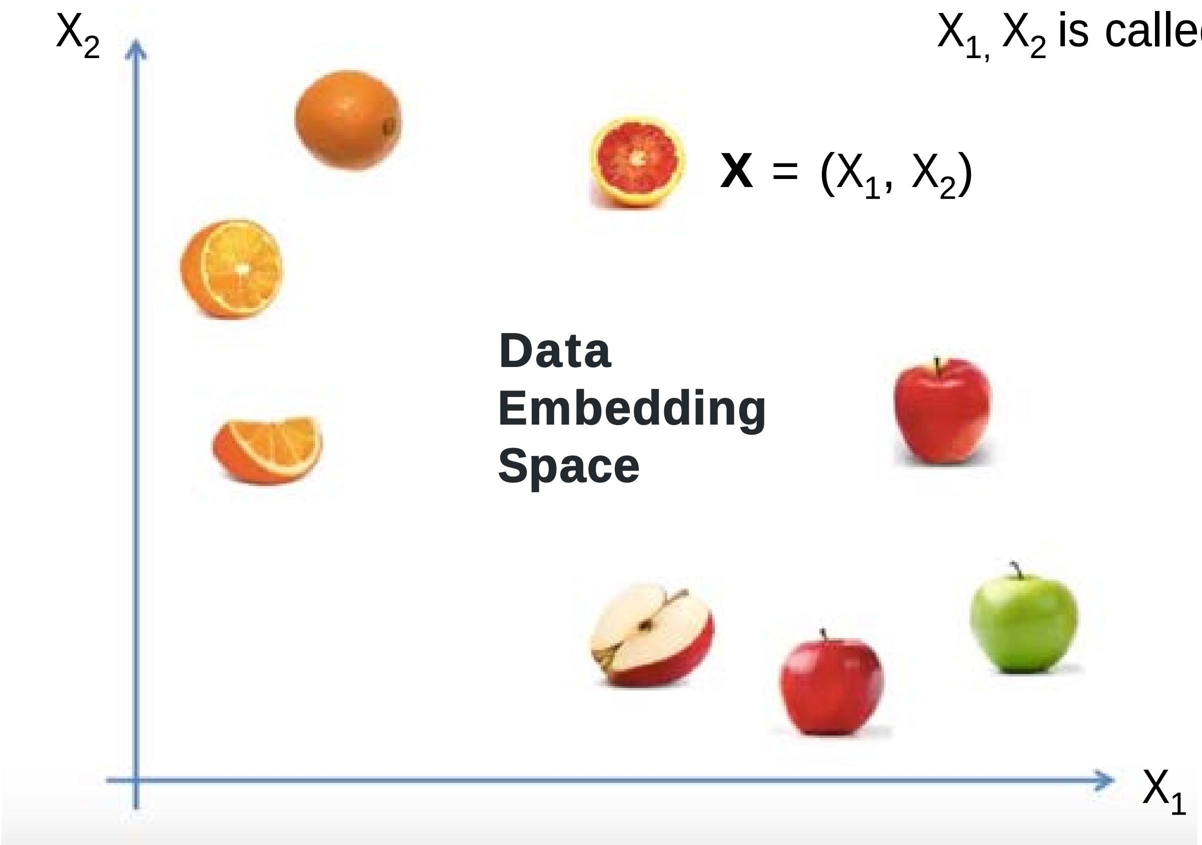
Sign function



Apple, orange classifier

▪ Data Embedding Space

X_1, X_2 is called feature or attribute.

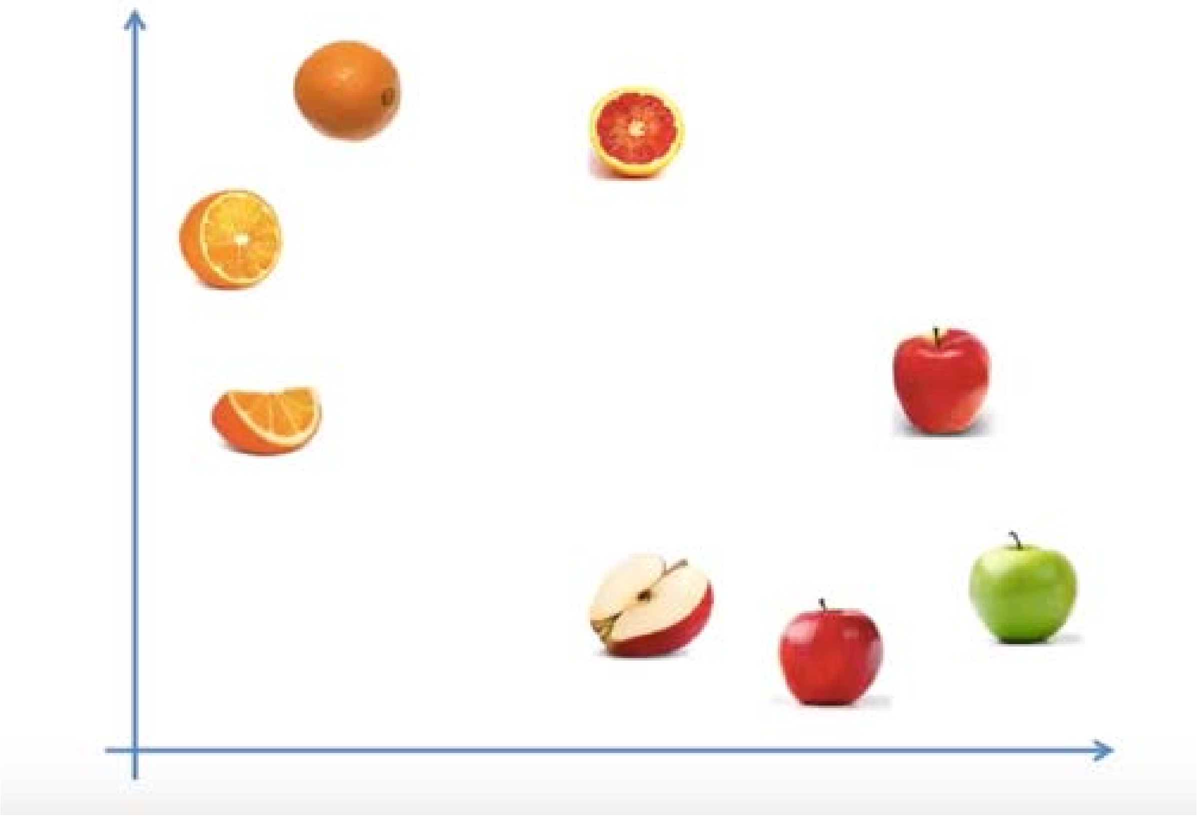


Data Embedding
범주형 자료를 **벡터 형태**로 바꾸는 것

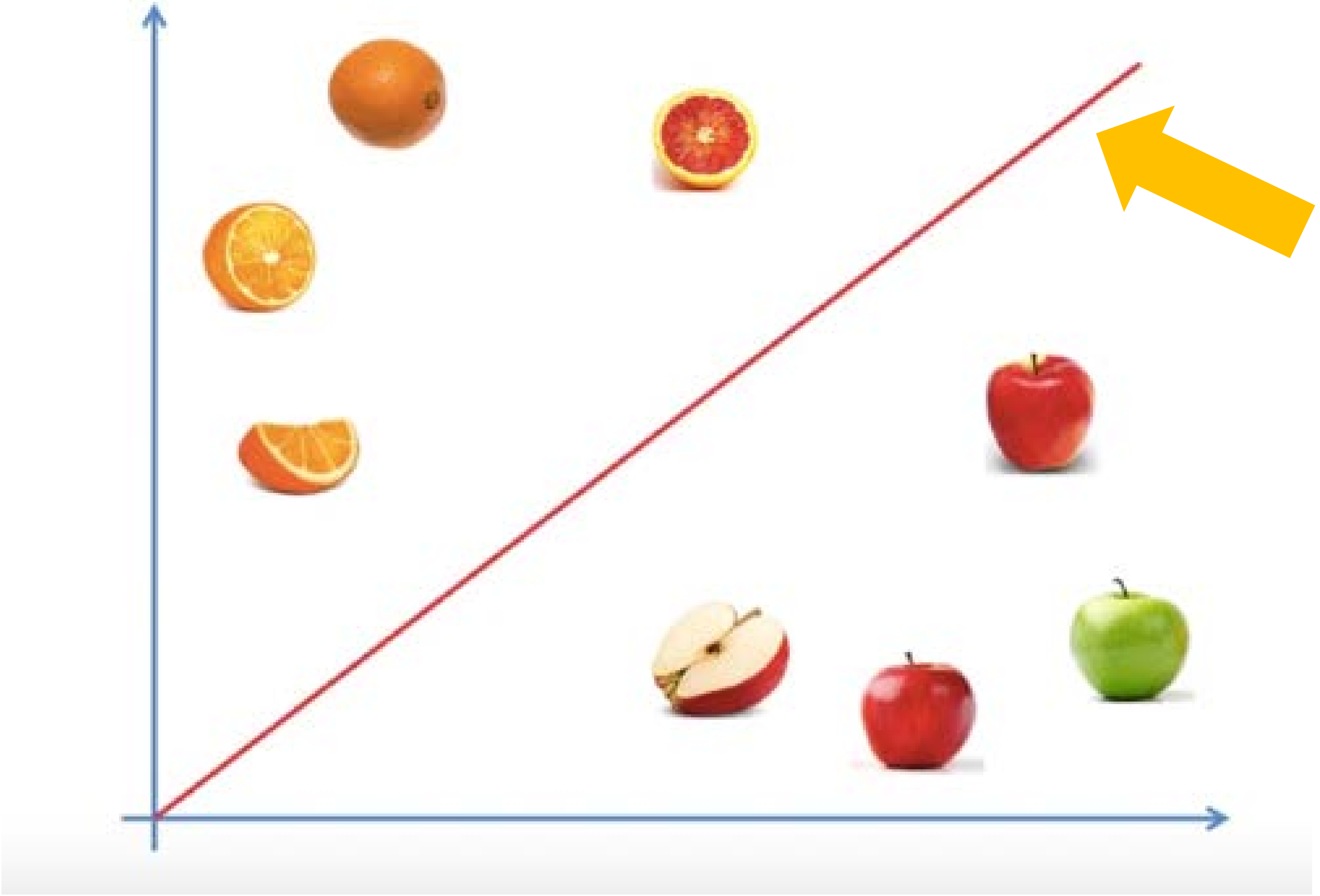
Categorical Data
범주형 데이터란 몇 개의 범주로 나누어진
데이터 예) 남/여, A/B/O/AB

Apple, orange classifier

- Which hyperplane can we choose?

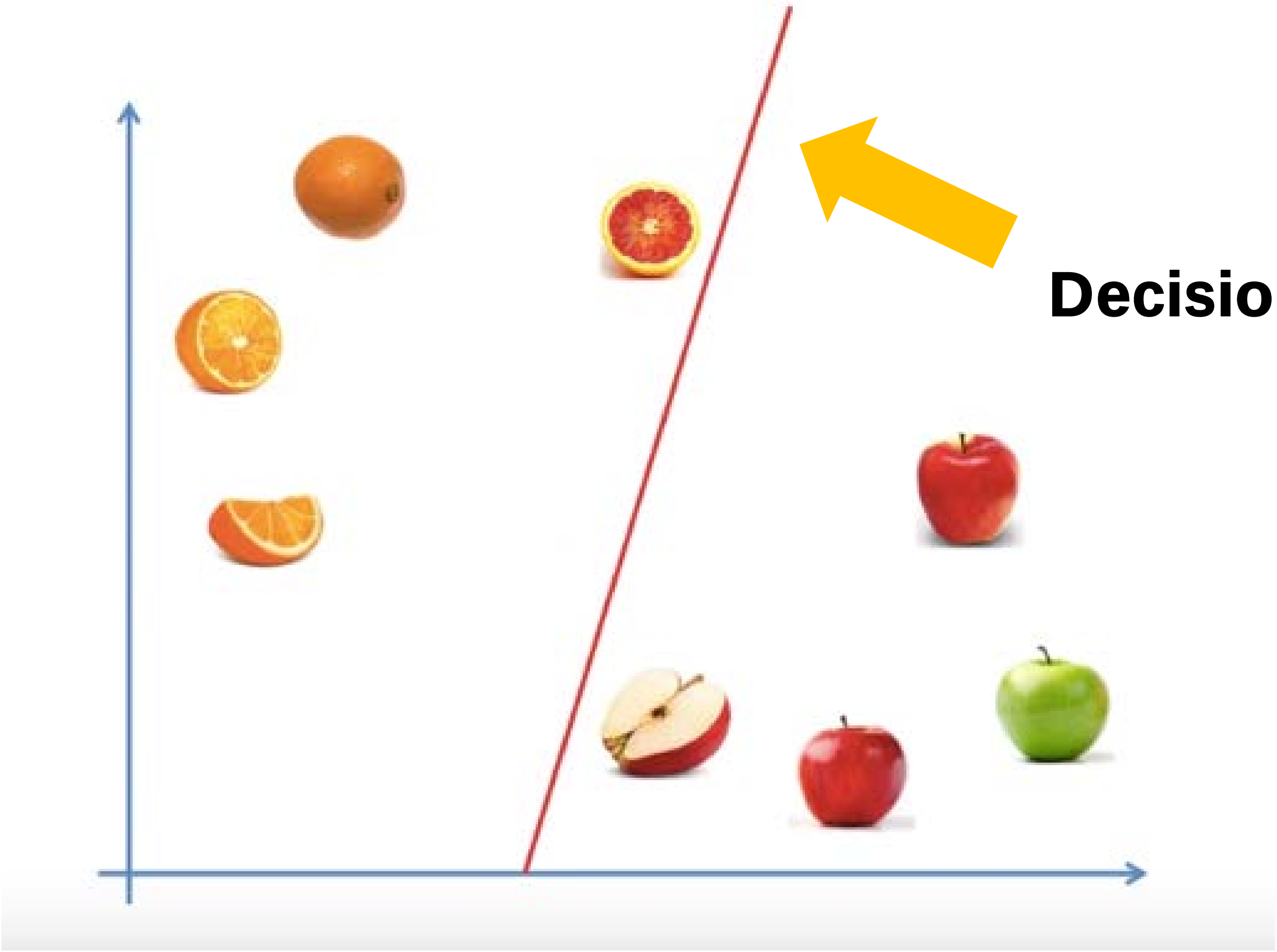


Apple, orange classifier



Decision Boundary

Apple, orange classifier

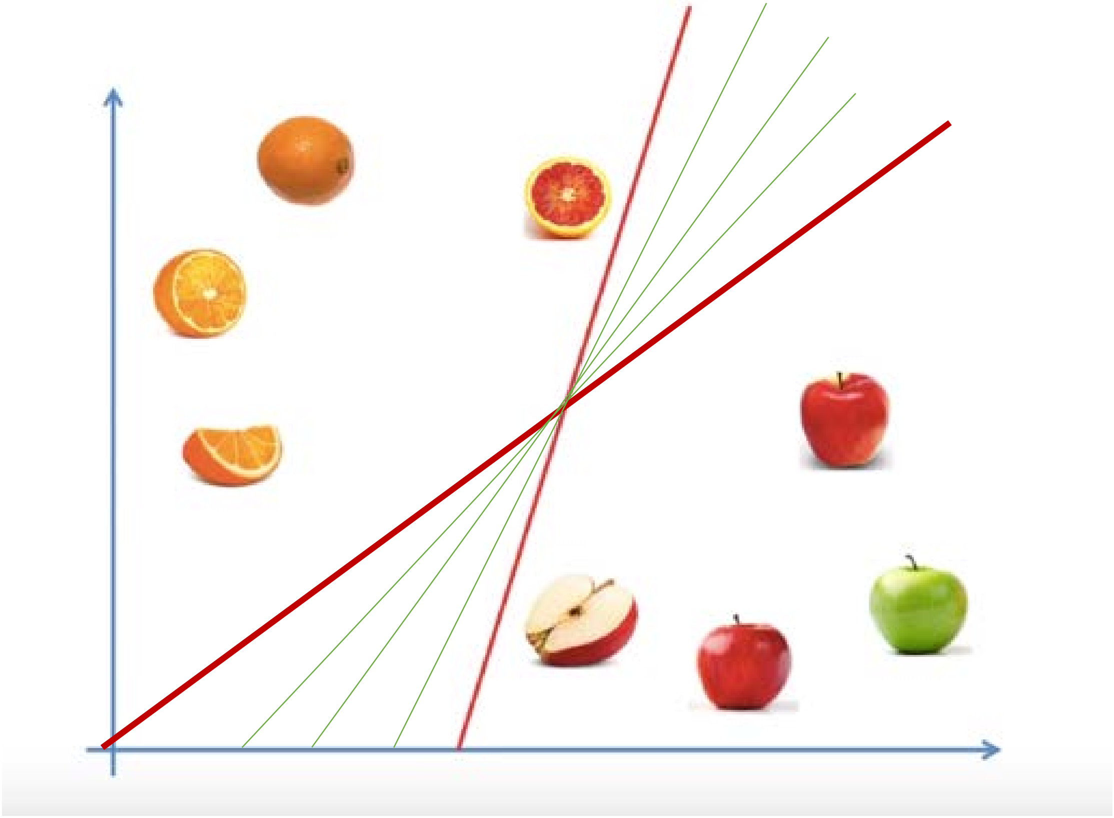


Decision Boundary

Apple, orange classifier

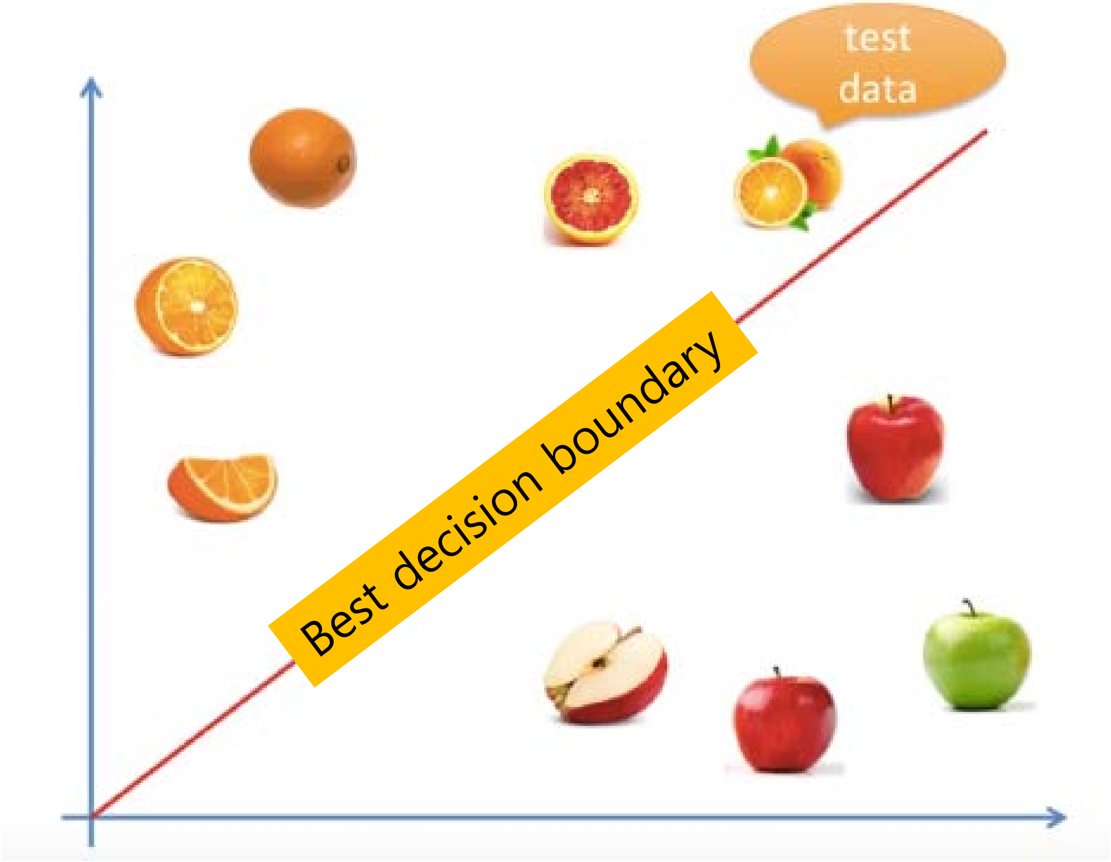
- Which one is better?
  - Classifier should have dealt with **unseen data**

Train sample data ➔ seen data
Test sample data ➔ unseen data



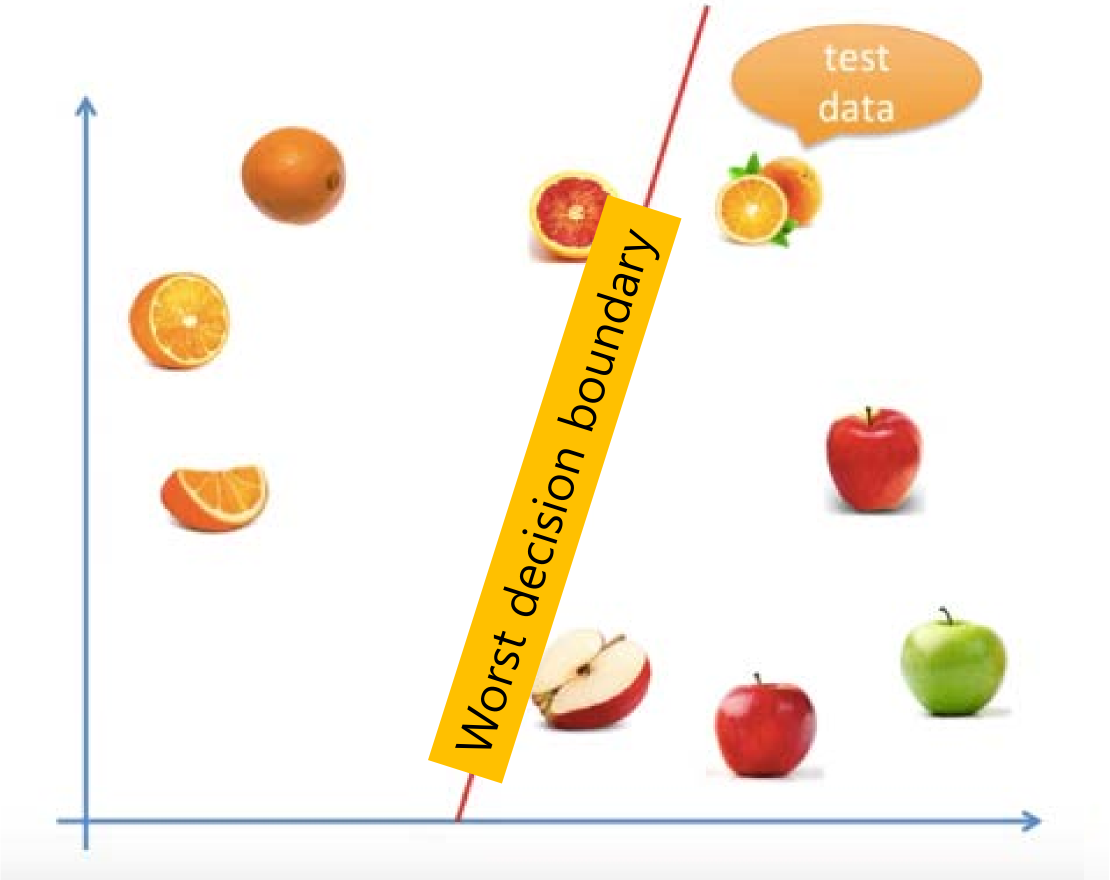
How can we decide decision boundary?

- Test data predicted well (O)



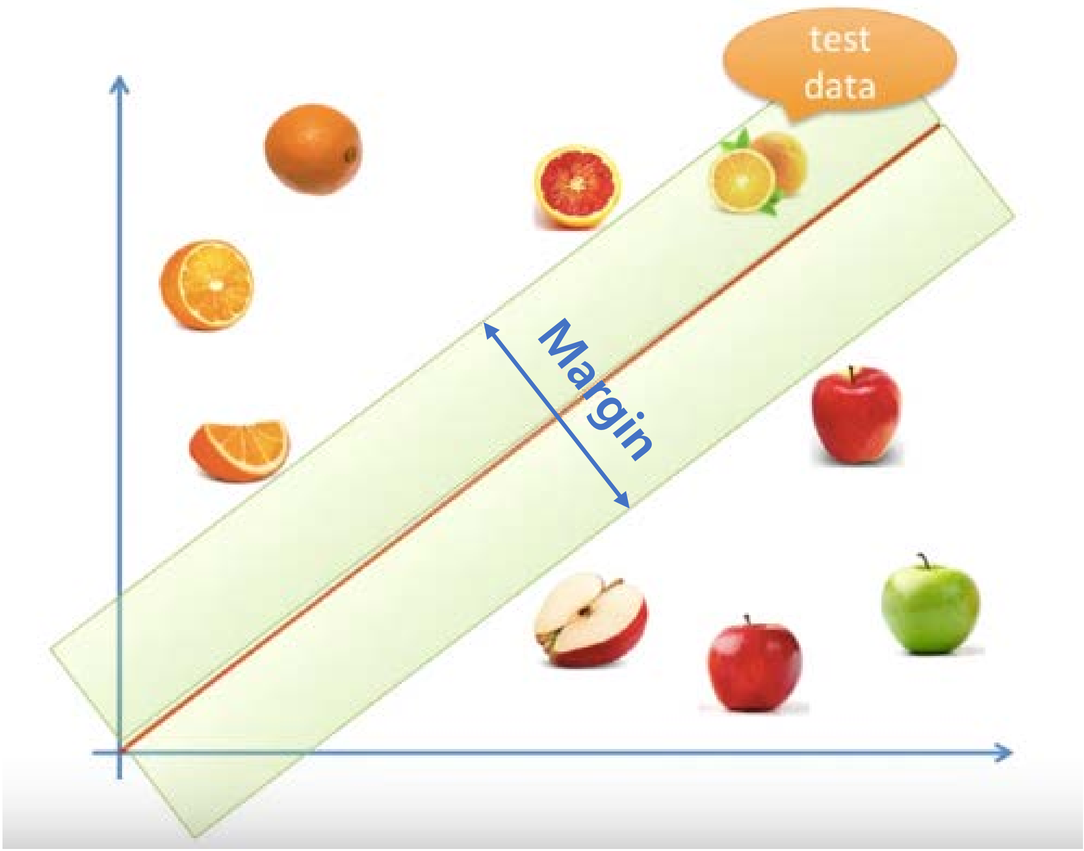
How can we decide decision boundary?

- Test data predicted well (X)



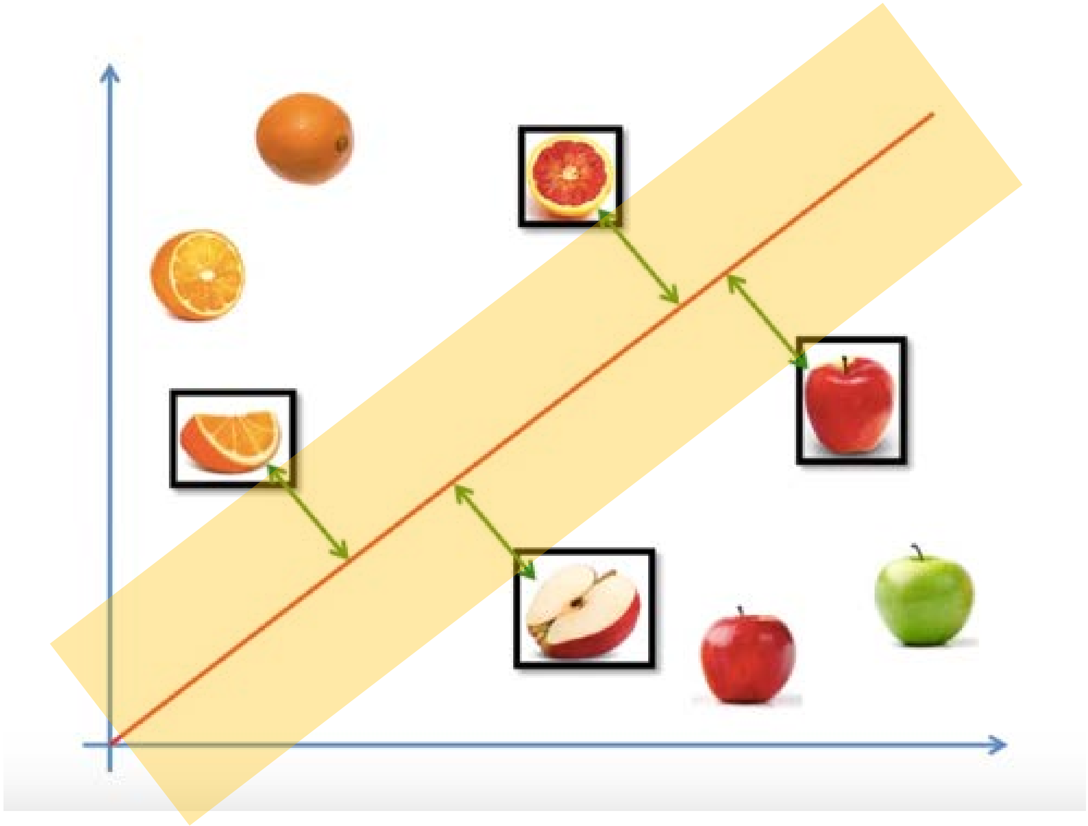
How can we decide decision boundary?

- The answer is “Large Margin”!!



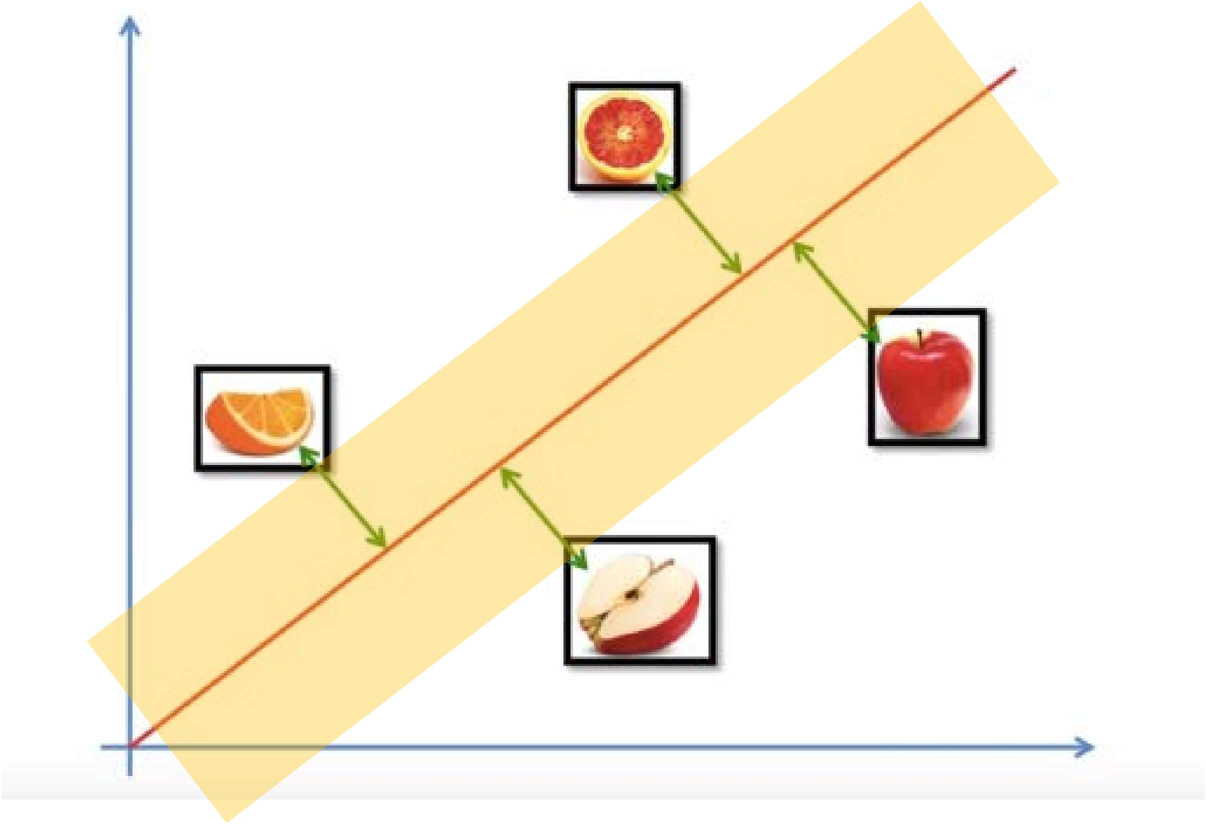
Support Vector

- **Support Vector**
  - Samples on the margin are called the support vectors.

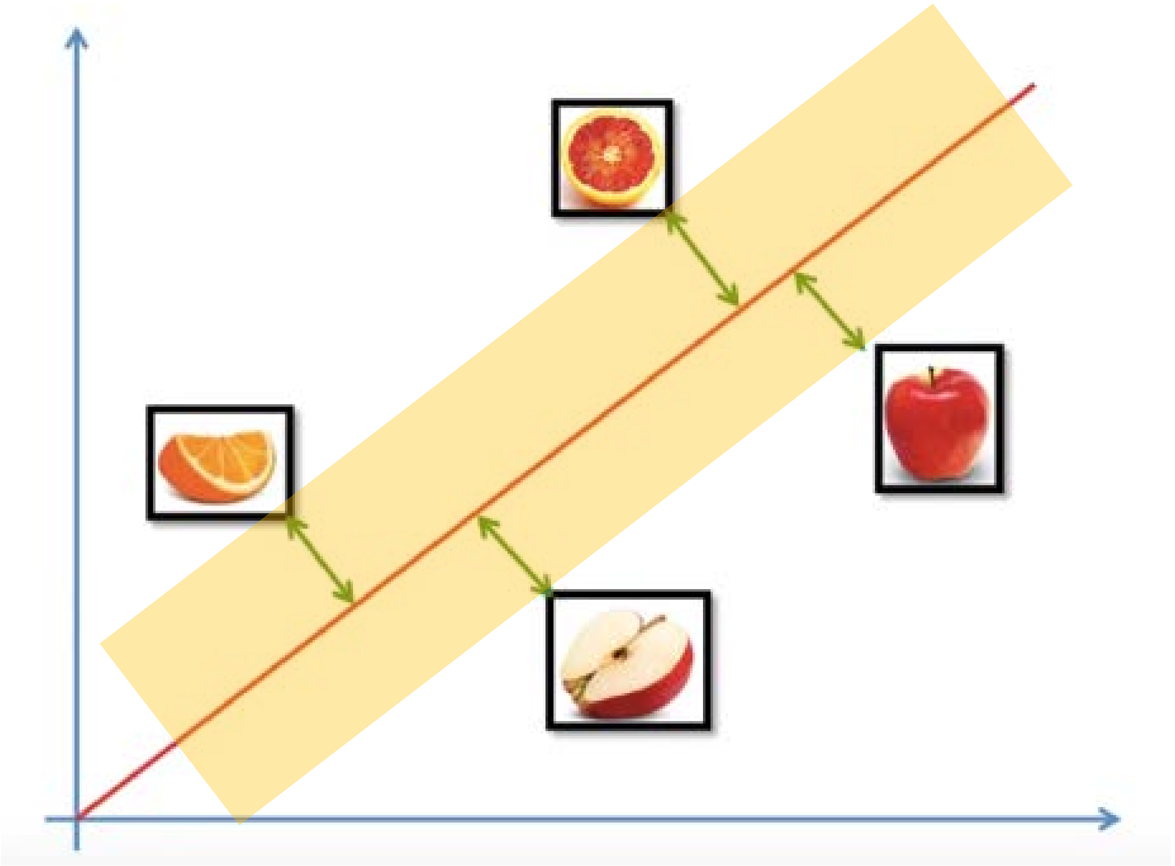


Support Vector

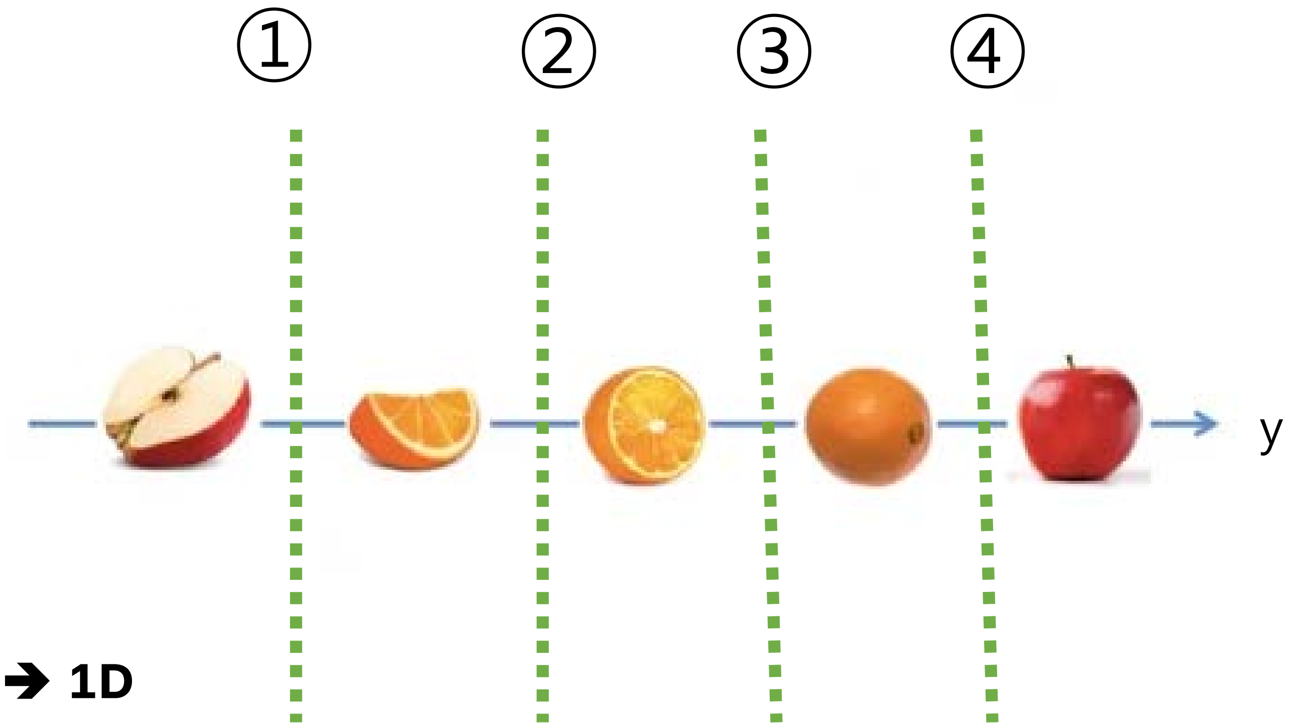
- SVM only uses support vector for prediction
  - Less computation!!!



Linearly Separable or not



What if data is not linearly separable?

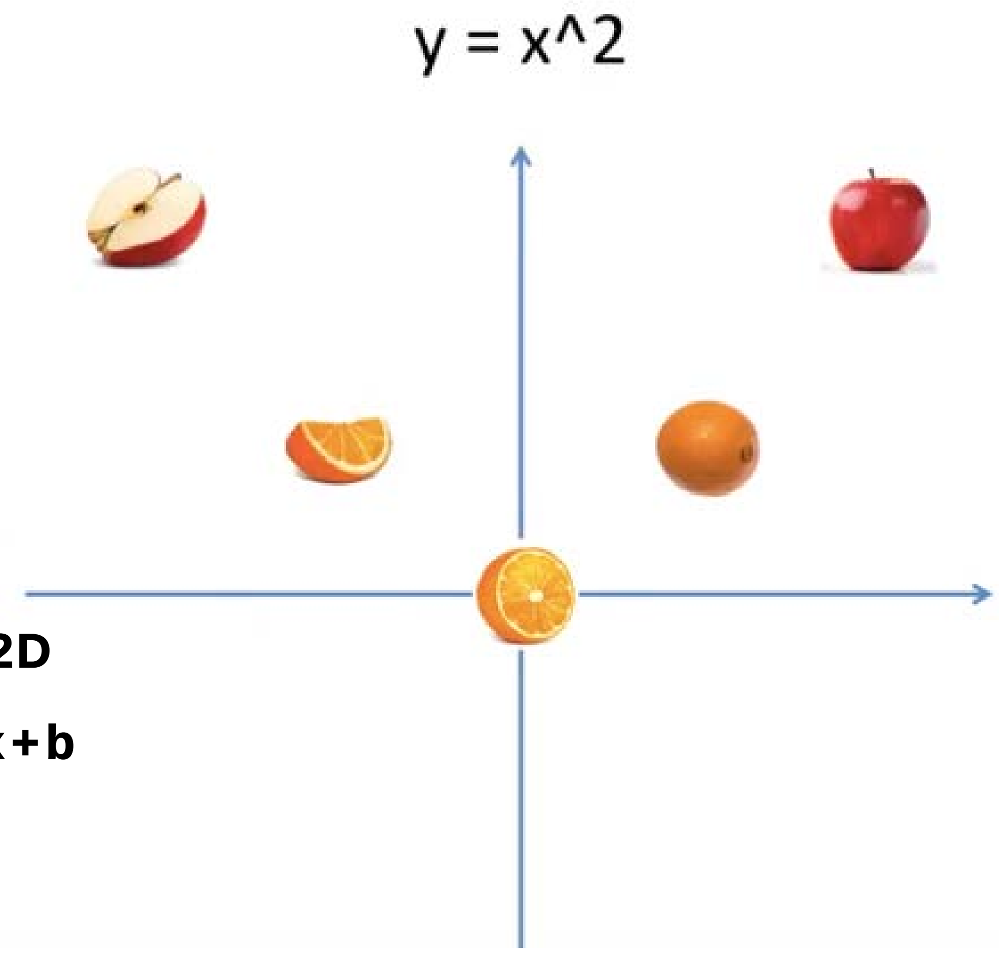


Data Embedding Space ➔ 1D

Hyperplane ➔ 0D ➔ y = b

What if data is not linearly separable?

- Mapping lower dimension to high dimension

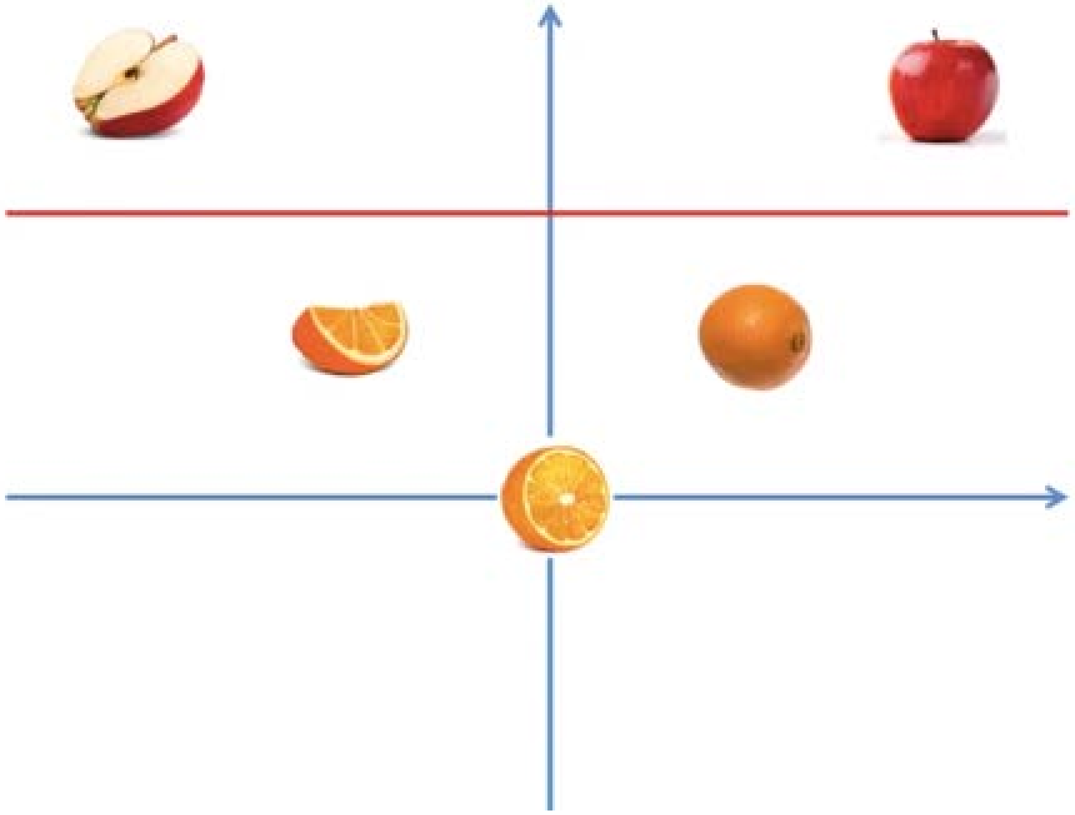


Data Embedding Space ➔ 2D

Hyperplane ➔ 1D ➔ y = ax+b

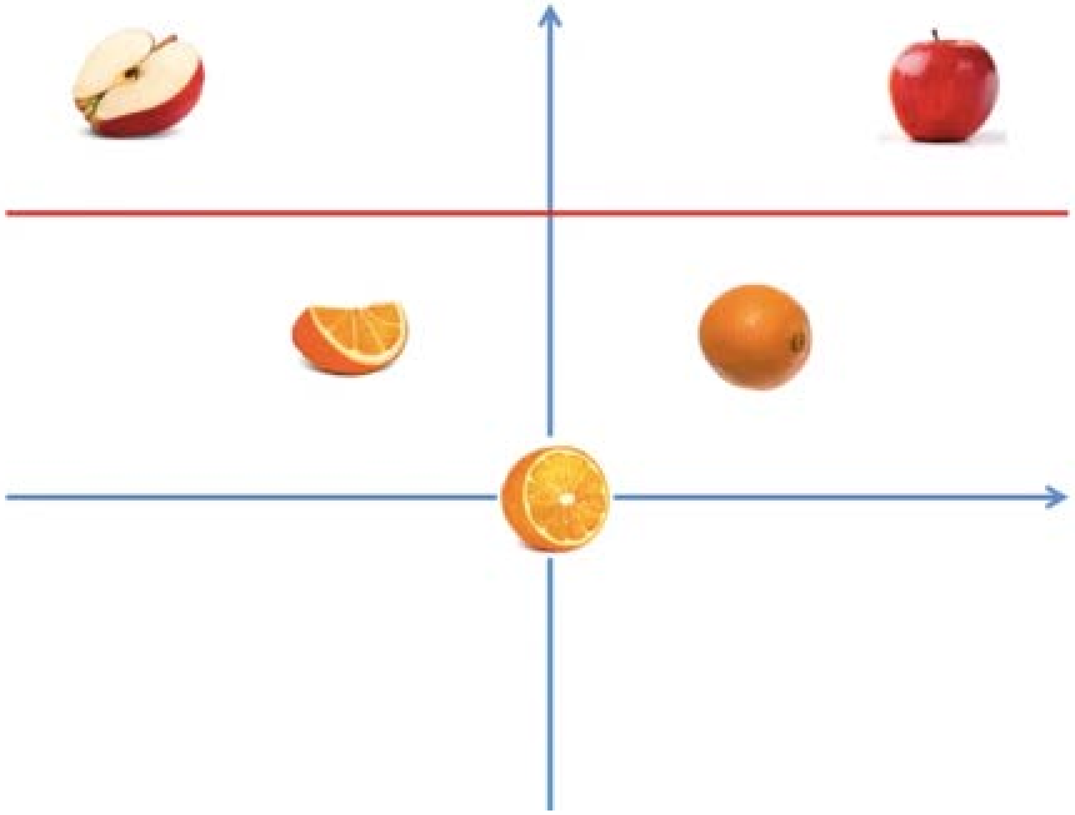
What if data is not linearly separable?

- Now it is linearly separable in higher dimension
  - Mapping to high dimension requires **much computation!**



What if data is not linearly separable?

- **Kernel trick** in SVM do this without explicitly
  - Move data point to higher dimension with **low computation!**



Kernel Trick

- The **kernel trick** avoids the explicit mapping that is needed to get linear learning algorithms.

- **Kernel methods** owe their name to the use of kernel functions, which enable them to operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space, but rather by **simply computing the inner products** between the images of all pairs of data in the feature space

Kernel Trick

- Kernel Function ➔ **simply computing the inner products**

$$
\begin{aligned}
linear &: \quad K(x_1, x_2) = x_1^T x_2 \\
polynomial &: \quad K(x_1, x_2) = (x_1^T x_2 + c)^d, \quad c > 0 \\
sigmoid &: \quad K(x_1, x_2) = \tanh\{a(x_1^T x_2) + b\}, \quad a, b \geq 0 \\
gaussian &: \quad K(x_1, x_2) = exp\left\{-\frac{\|x_1 - x_2\|_2^2}{2\sigma^2}\right\}, \quad \sigma \neq 0
\end{aligned}
$$

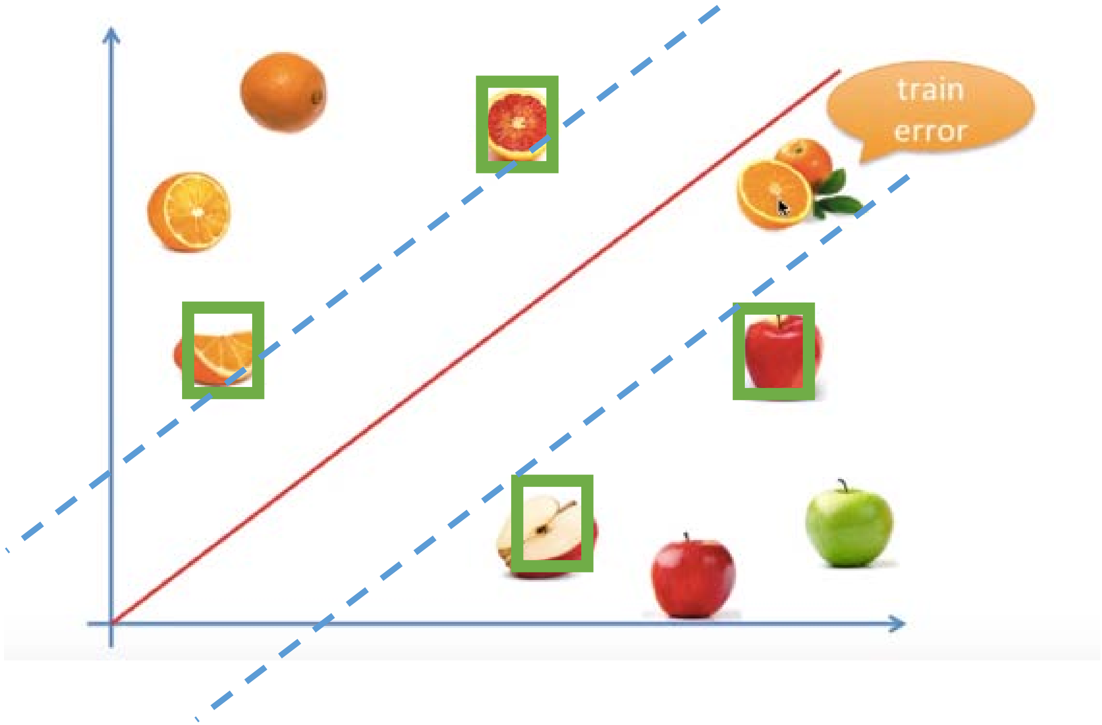
Mapping 함수의 inner-product.. Mapping (m➔n)

$$K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j) = x_i^T A^T A x_j$$

SVM Parameter - Cost

Manually hand-tuned

- Cost is small == Margin is large



C is small

Training error is allowed

Overfitting is not allowed

Margin is large

Testing error is small

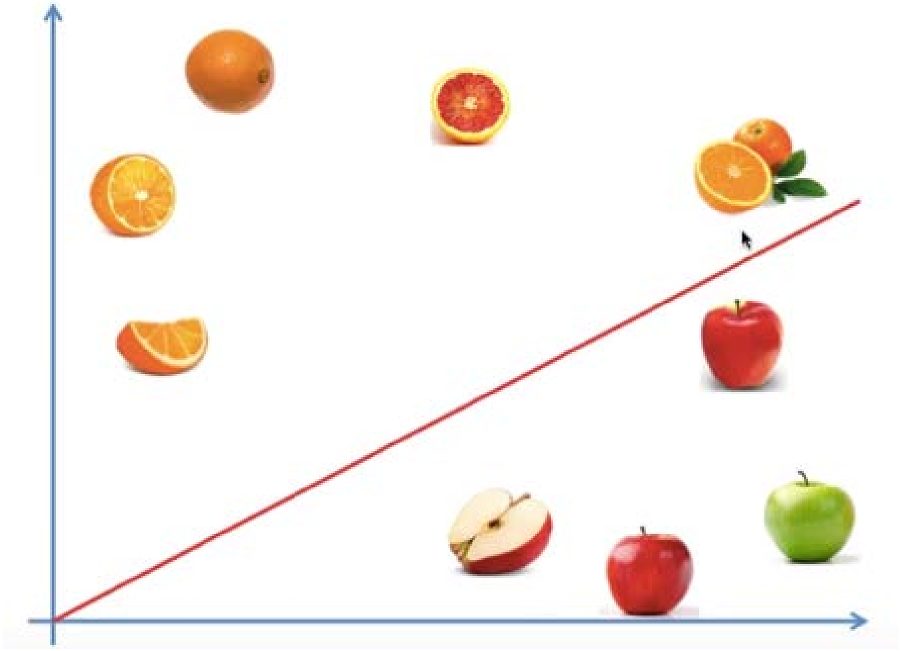
$$(\theta) = C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_i^2$$

misclassification

Margin width

SVM Parameter - Cost

- Cost is large == Margin is small



$$(\theta) = C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_i^2$$

misclassification

Margin width

C is large

Training error is not allowed

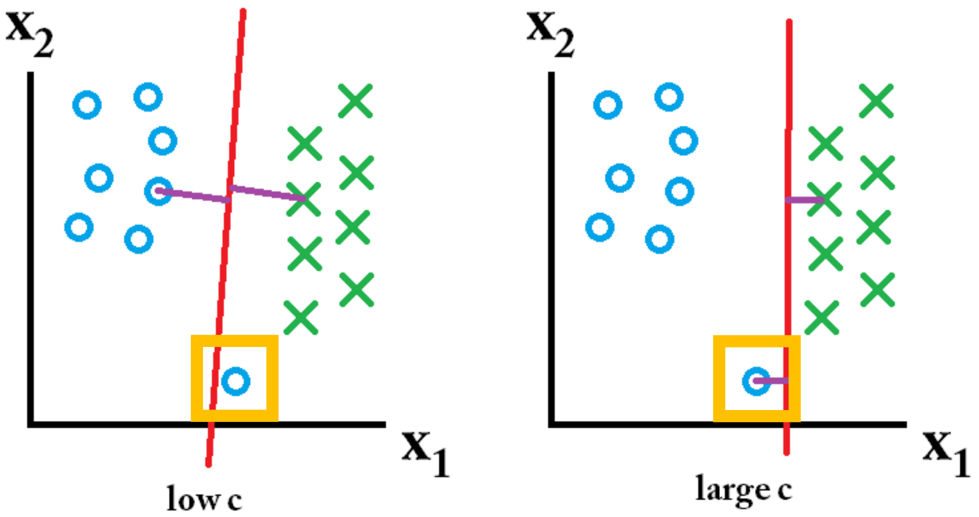
Overfitting is allowed

Margin is small

Testing error is large

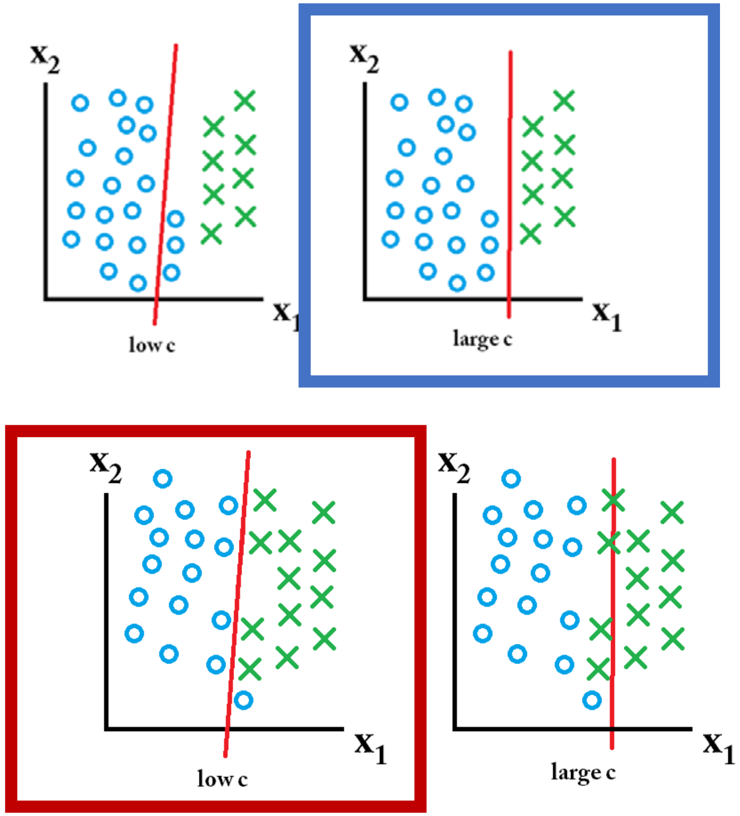
SVM Parameter - Cost

- We **assume** that some samples caused by train error are the **outlier**.
- Therefore, we generally select a **large margin** for decision boundary.
- But, if not?



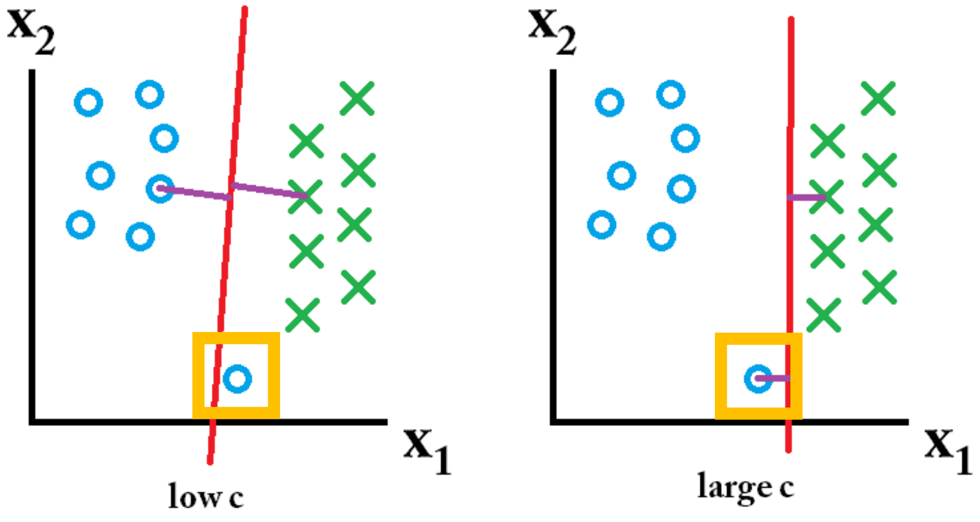
Inlier or outlier

?



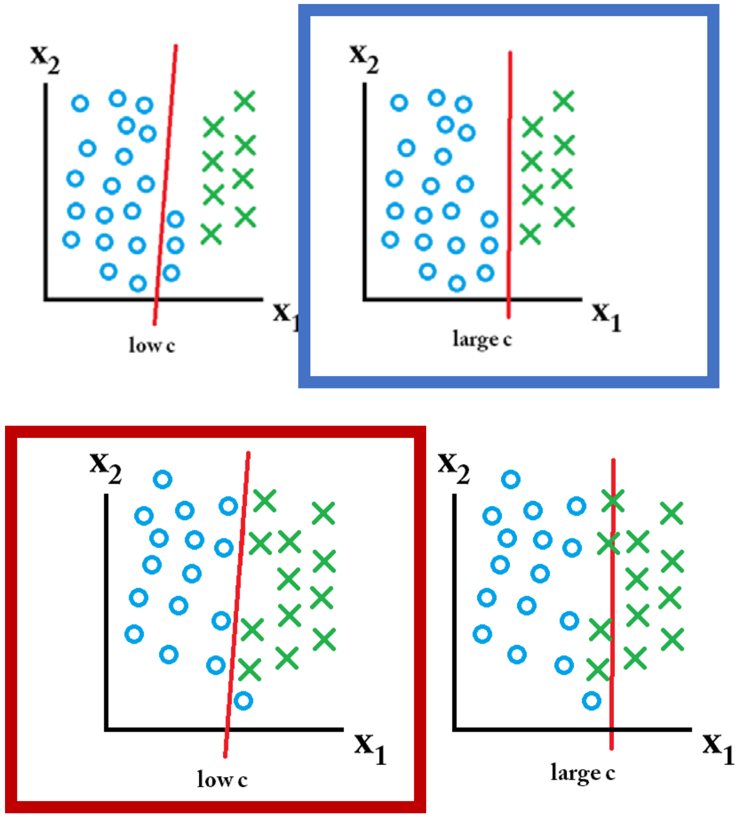
SVM Parameter - Cost

- Therefore, we cannot argue that we should choose large C, but we must make a decision through **data analysis**.

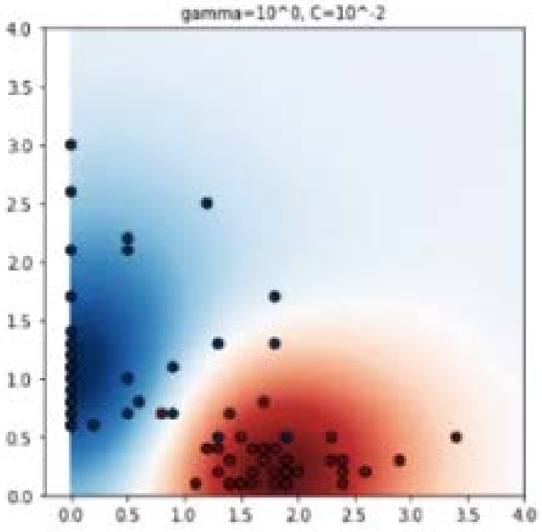


Inlier or outlier

?

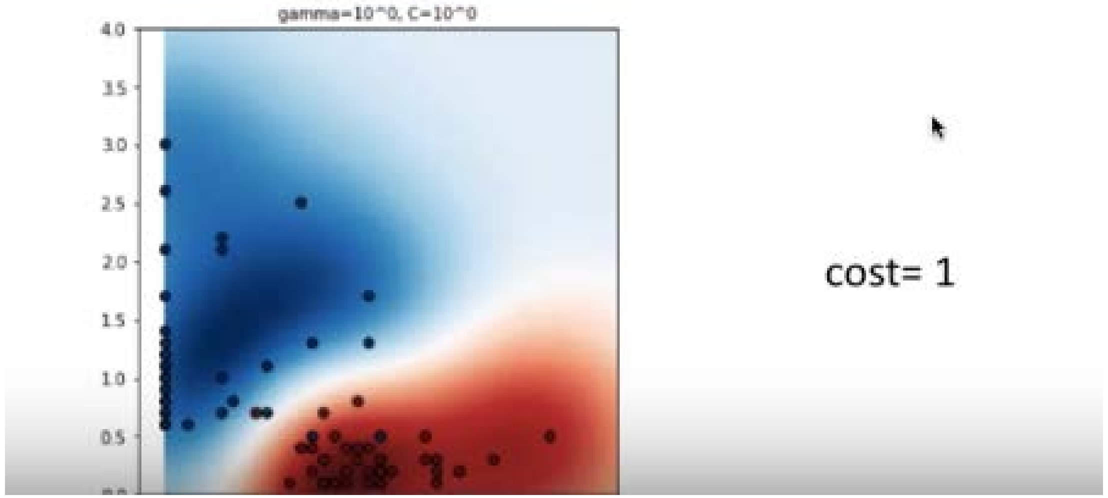


SVM Parameter - Cost



cost= 0.01

Cost is small
Training error is allowed
Overfitting is not allowed
Decision boundary is simple



cost= 1

Cost is large
Training error is allowed
Overfitting is allowed
Decision boundary is complex

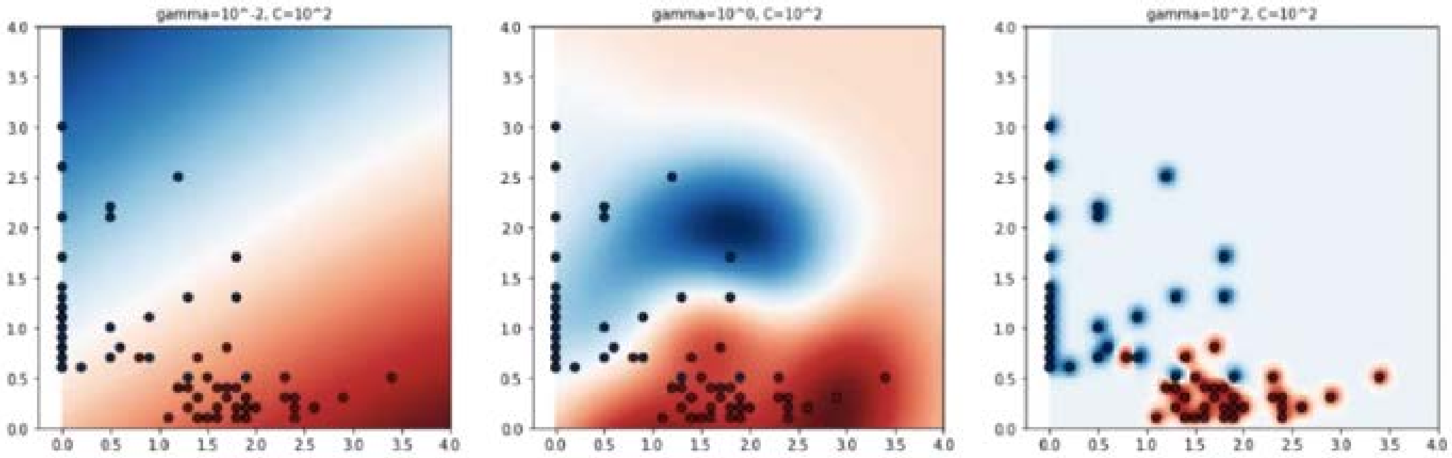
SVM parameter – Gamma in RBF kernel

- Intuitively, the **gamma parameter** defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'.

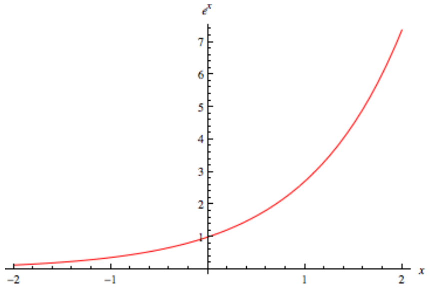
Radial Base Function (also called Gaussian Kernel)

```
K(x,x') = exp(-gamma * ||x-x'||^2)
```

Far **Close**

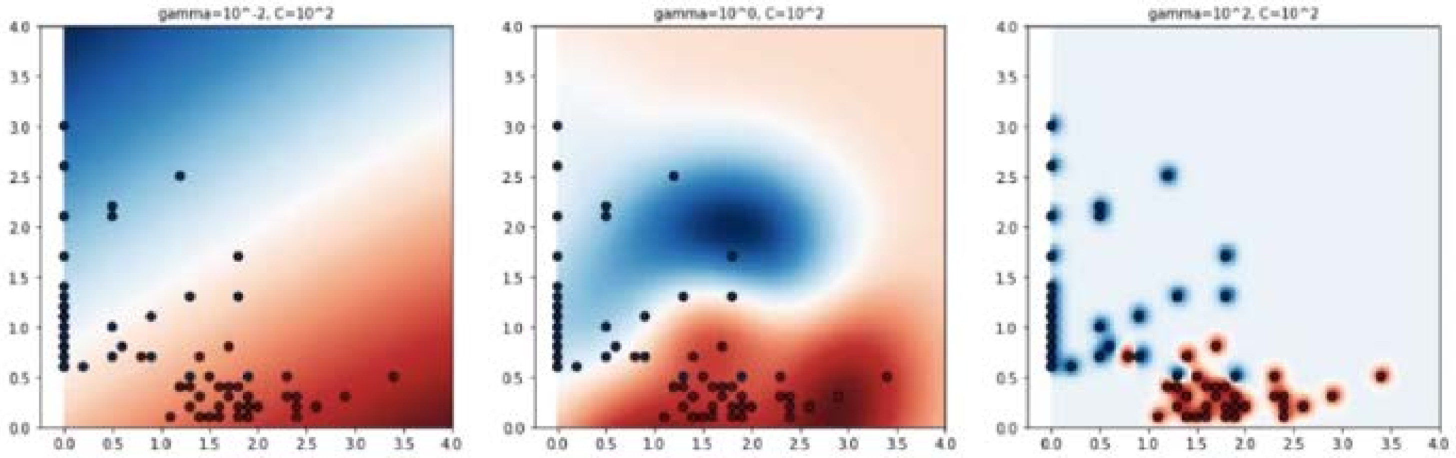


gamma = 0.01 gamma = 1 gamma = 100



SVM parameter – Gamma in RBF kernel

Far **Close**



gamma = 0.01 gamma = 1 gamma = 100

Gamma is small
Influence is large
Margin is large
Similarly to a linear model

Gamma is large
Influence is small
Margin is small
Overfitting is allowed

Find optical parameter – **data analysis**

- **Grid Search**
  - **Grid search** builds a model for **every combination** of hyper-parameters specified and evaluates each model.

Gamma

cost

	1	10	100
1	0.7	0.8	0.7
10	0.8	0.8	0.9
100	0.6	0.8	0.8