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

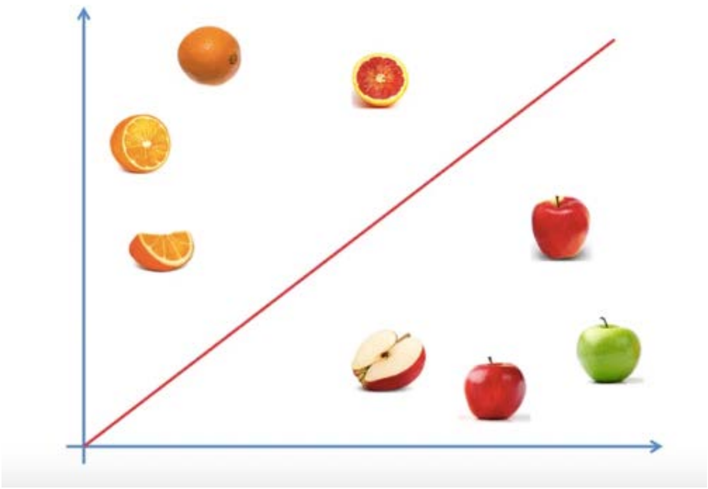
- Until now, favorite classifier to everyone
  - Wondering why? **Kernel Trick!!!**

**"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**.

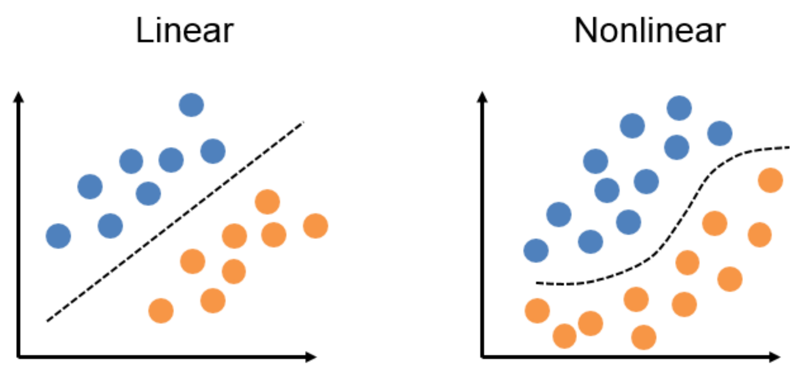
- In the email classification example, this classifier could be a hypothesis for labeling emails as **spam** or **non-spam**.



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)

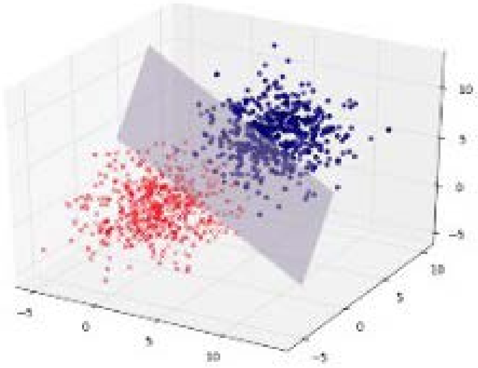
데이터를 구획해주는 이 **점선의 함수 (decision boundary)**를 우리는 **판별 함수 (decision function)**라 부른다.

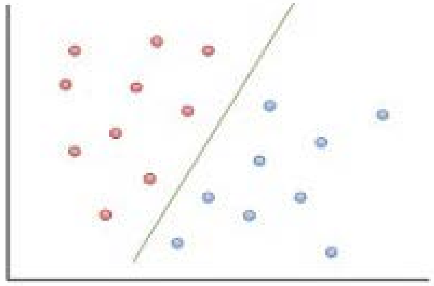
Classifier

- **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

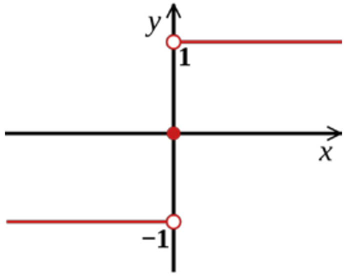




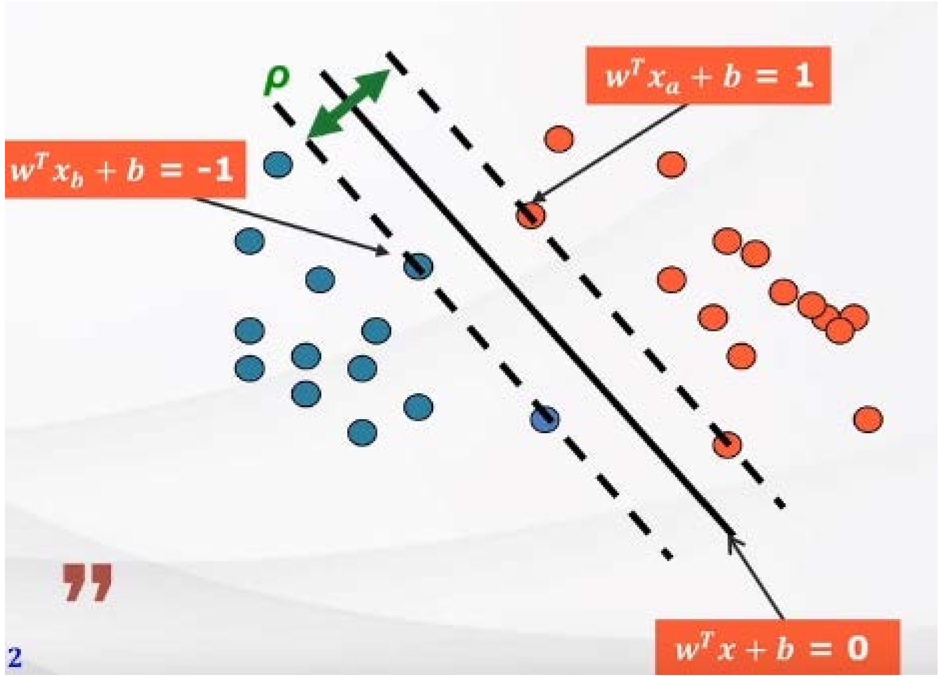
SVM Classifier

- 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 $W^T\mathbf{X}_i+b < -1$

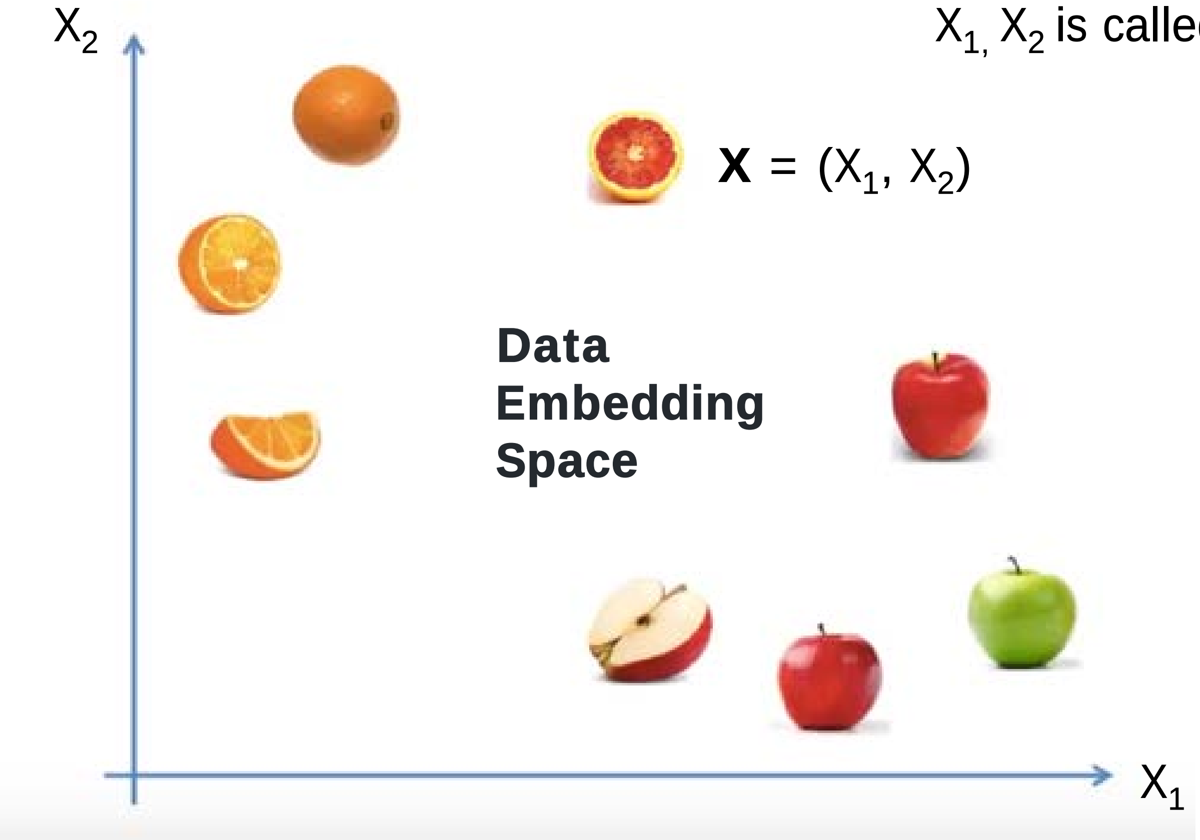


Sign function



Apple, orange classifier

- **Data Embedding Space**



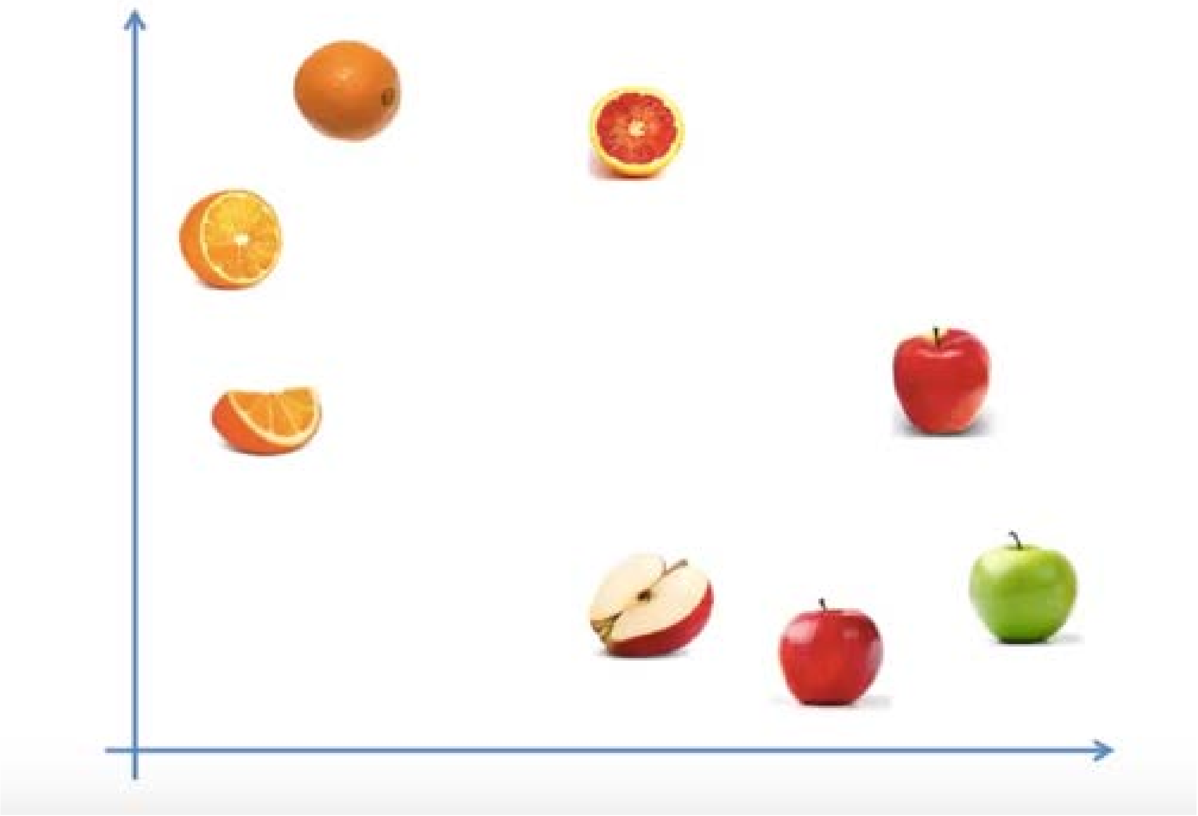
X_1, X_2 is called feature or attribute.

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

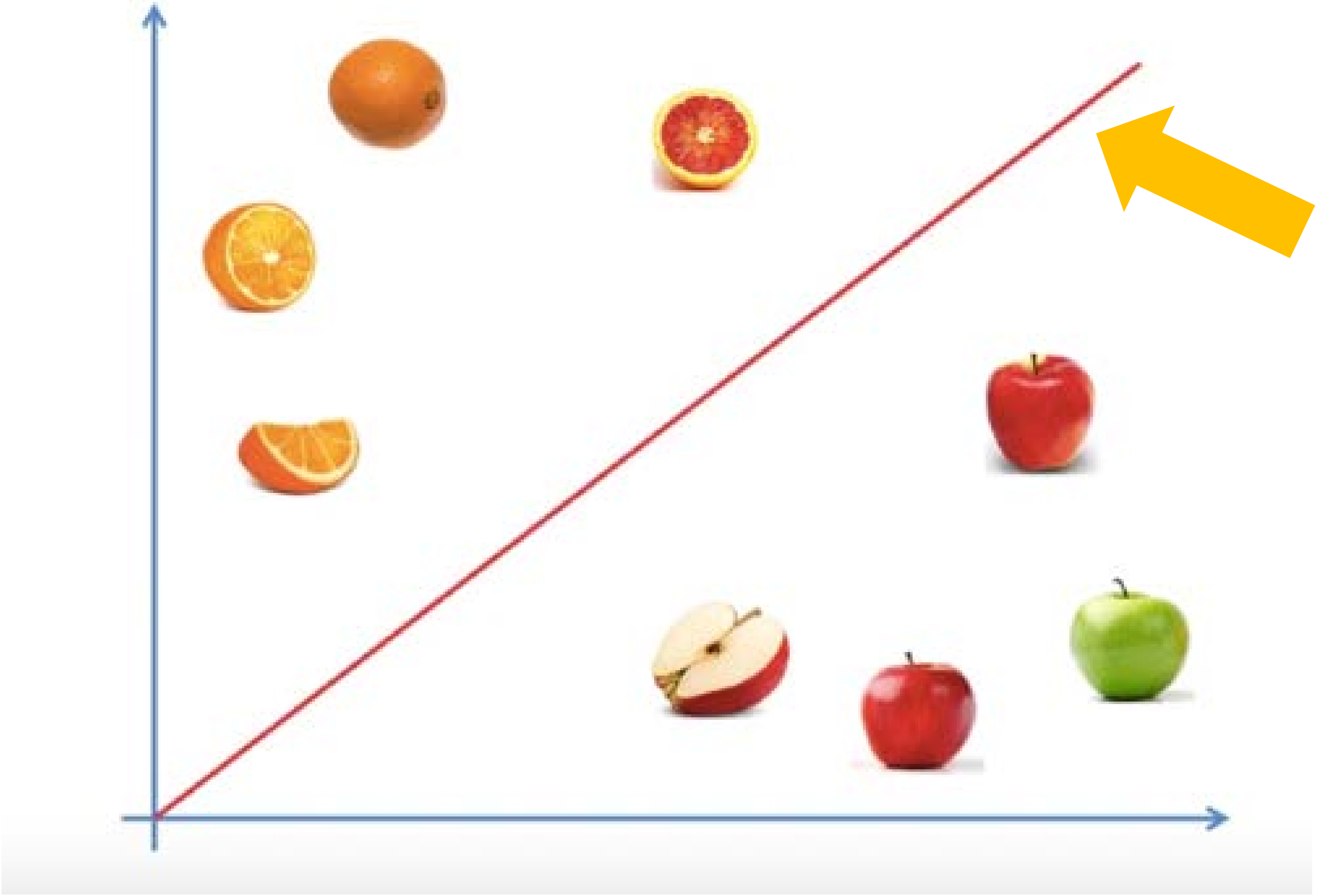
Categorical Data
범주형 데이터란 몇 개의 범주로 나누어진 데이터 예) 남/여, A/B/O/AB

Apple, orange classifier

- Which hyperplane can we choose?

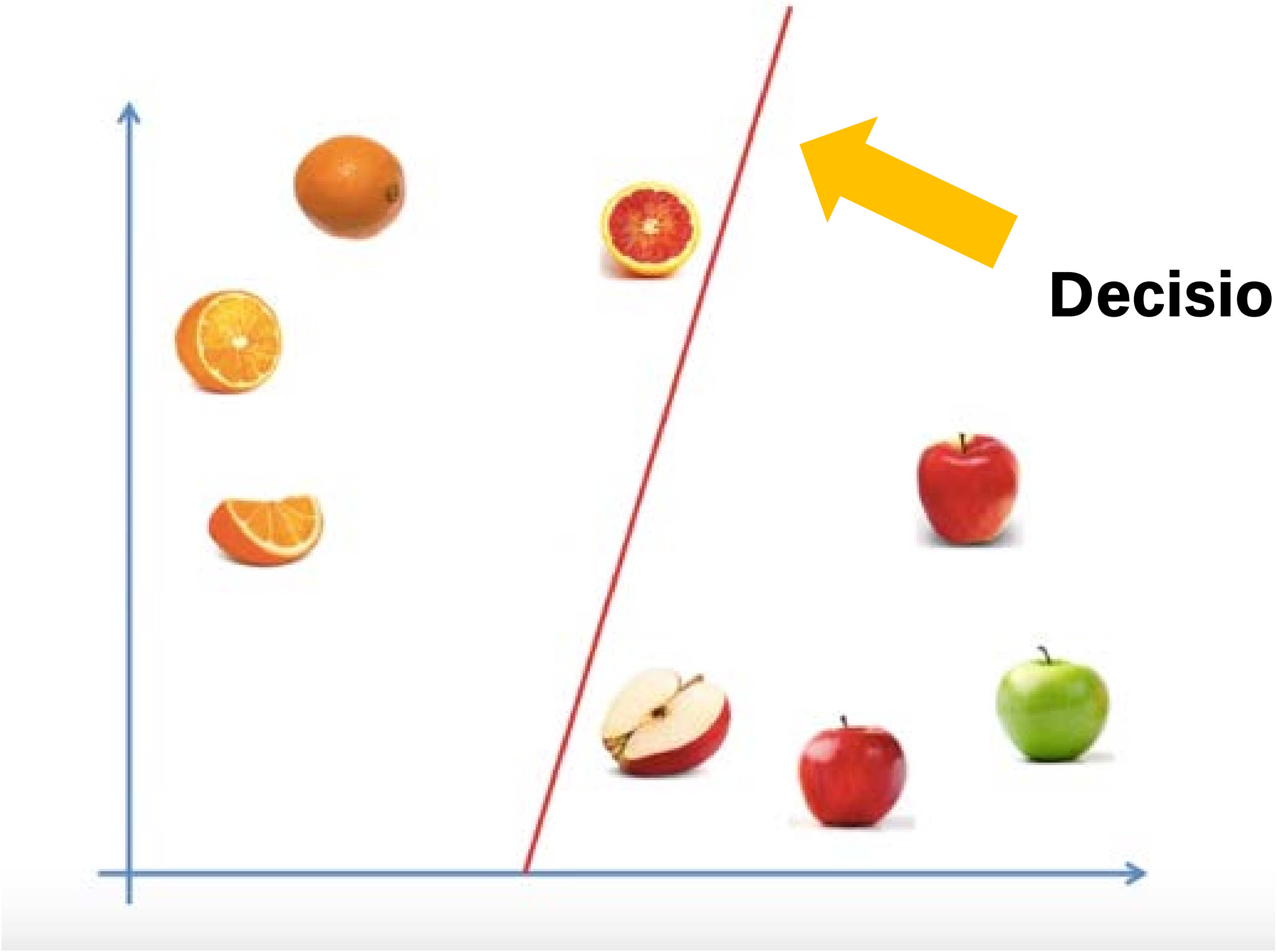


Apple, orange classifier



Decision Boundary

Apple, orange classifier

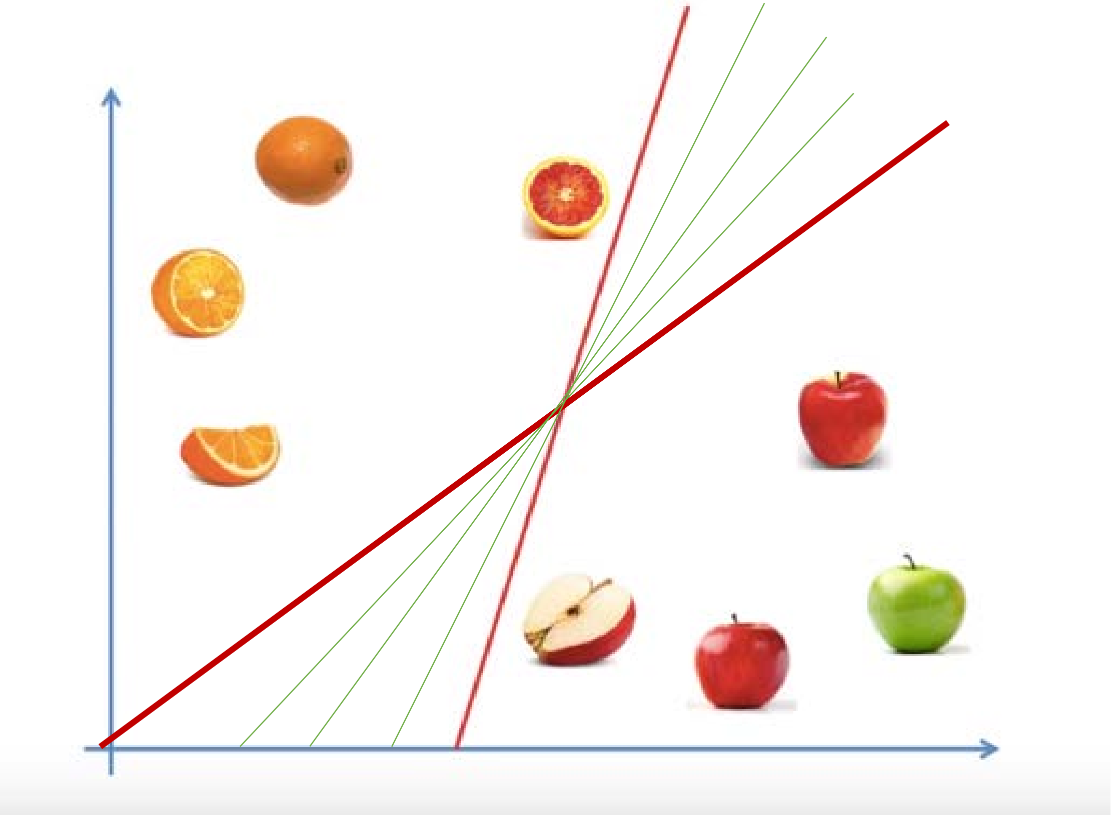


Decision Boundary

Apple, orange classifier

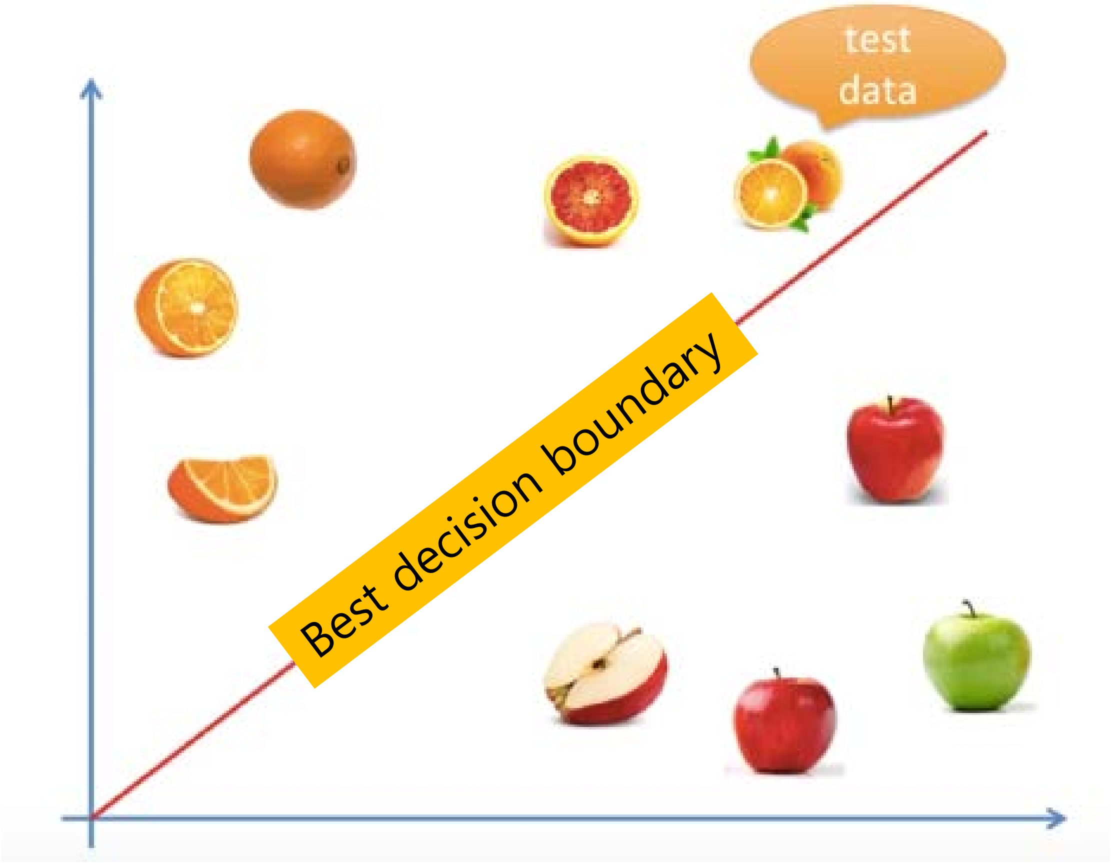
- Which one is better?
  - Classifier should have dealt with **unseen data**

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



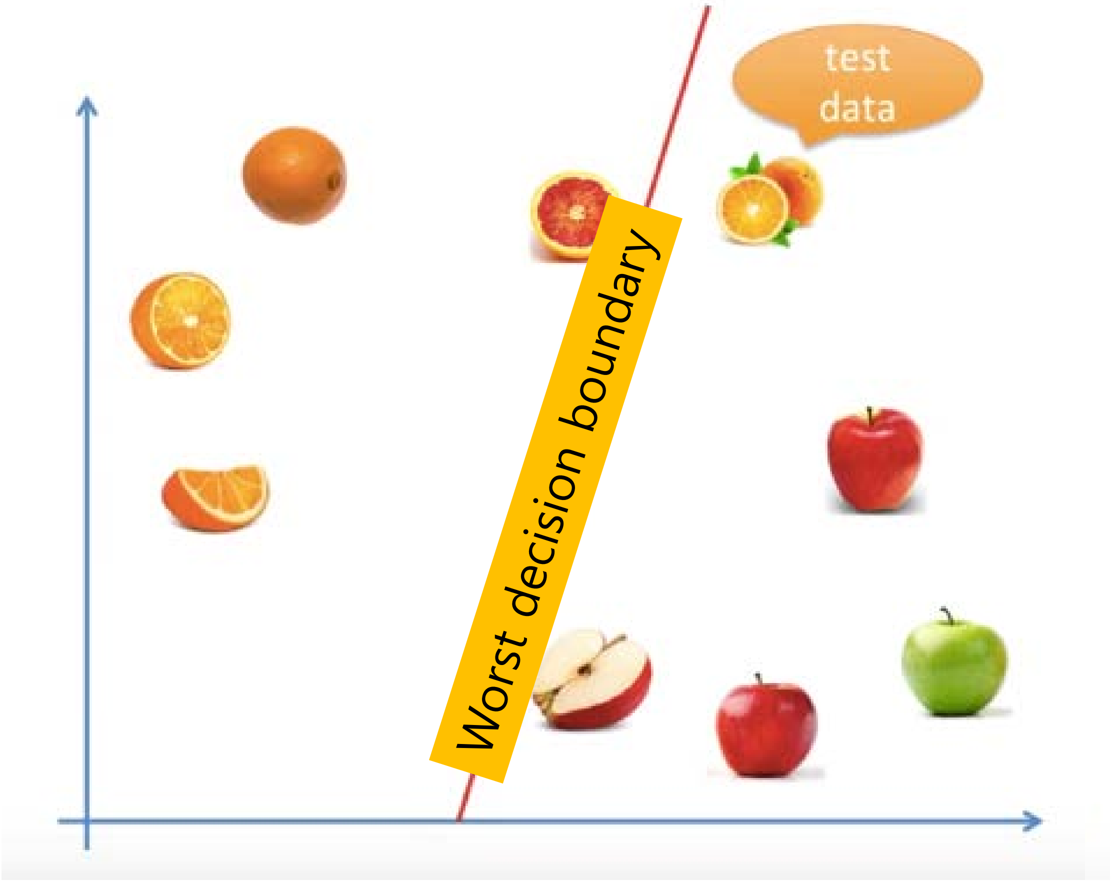
How can we decide decision boundary?

- Test data predicted well (O)



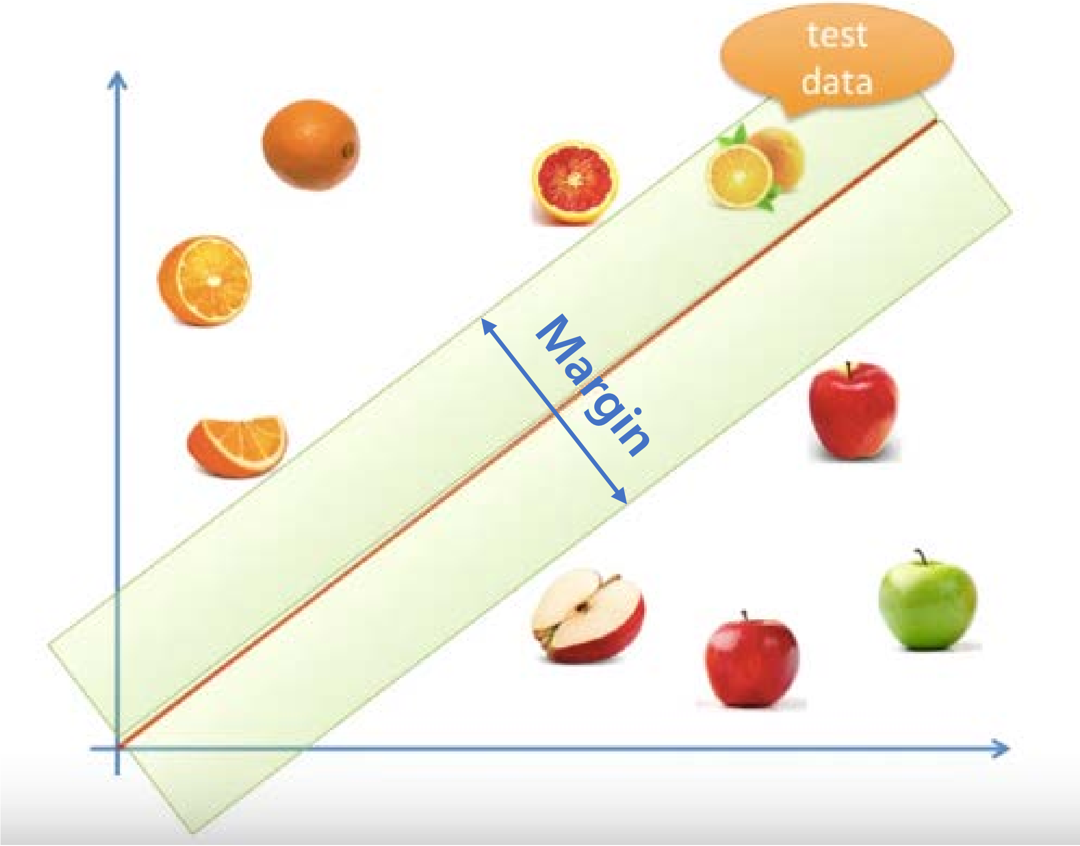
How can we decide decision boundary?

- Test data predicted well (X)



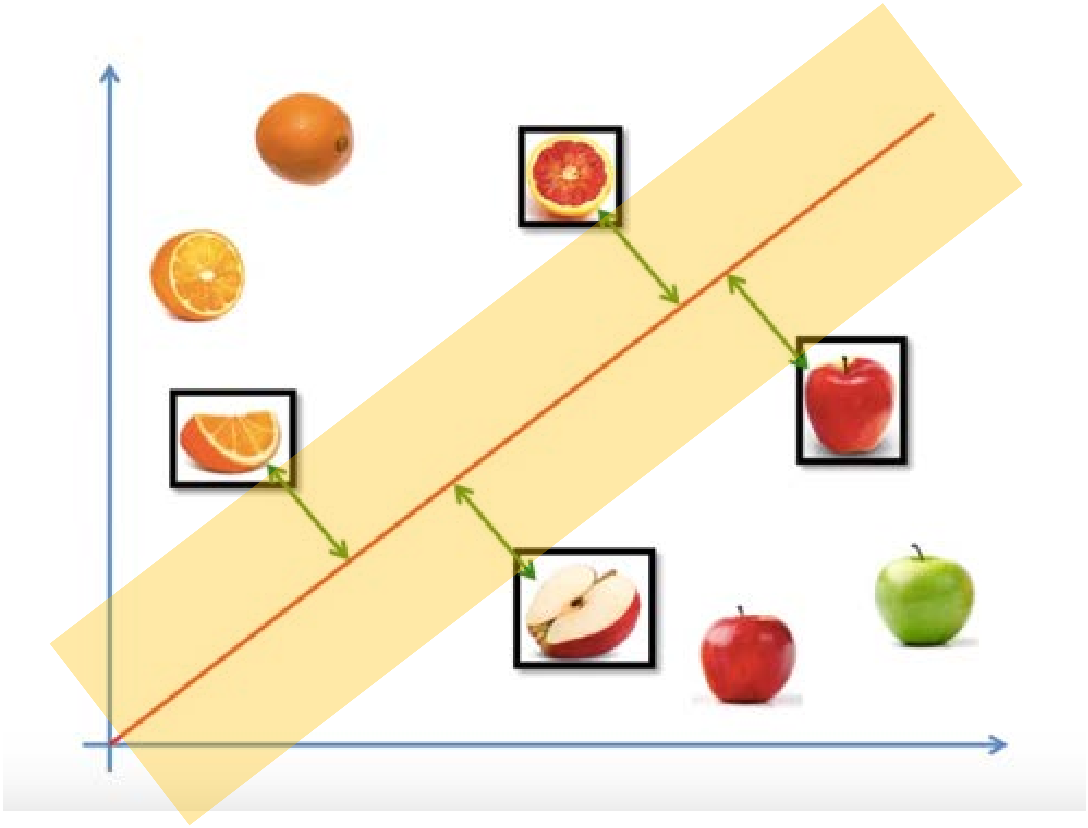
How can we decide decision boundary?

- The answer is "Large Margin"!!



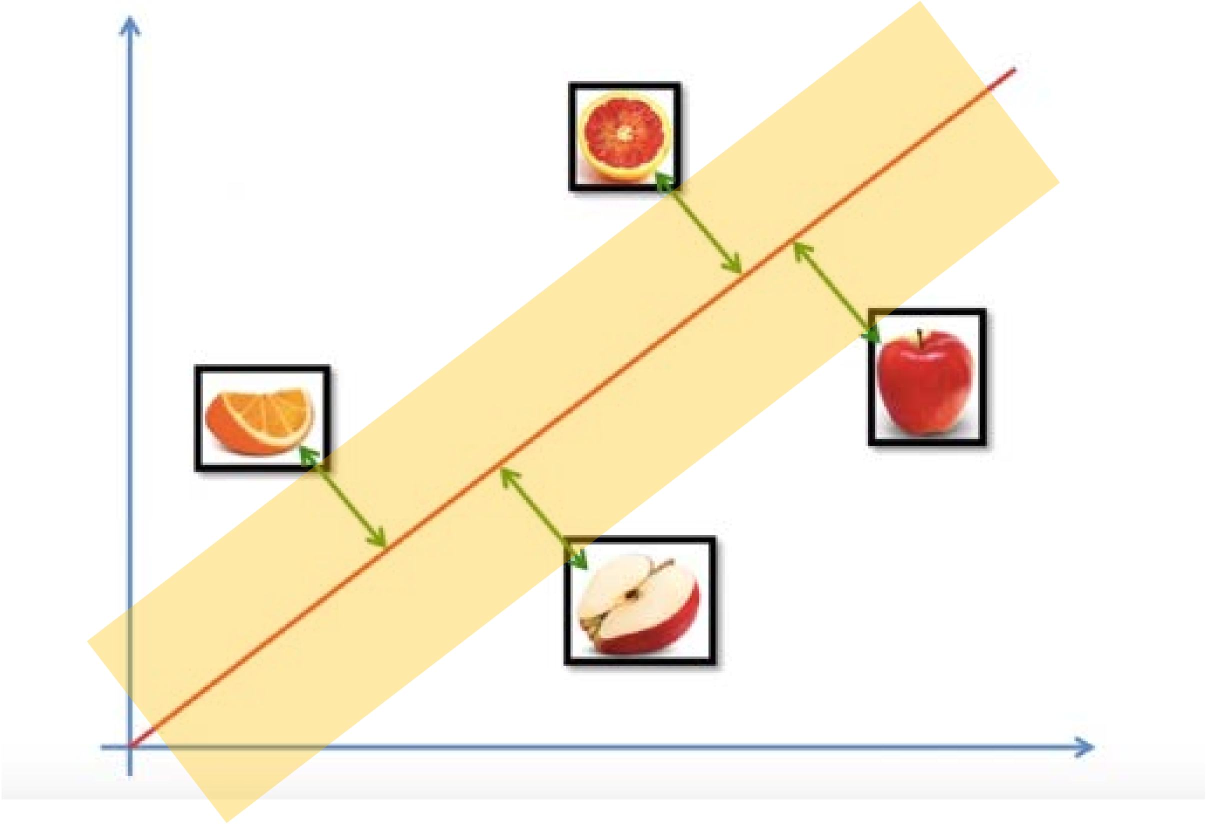
Support Vector

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

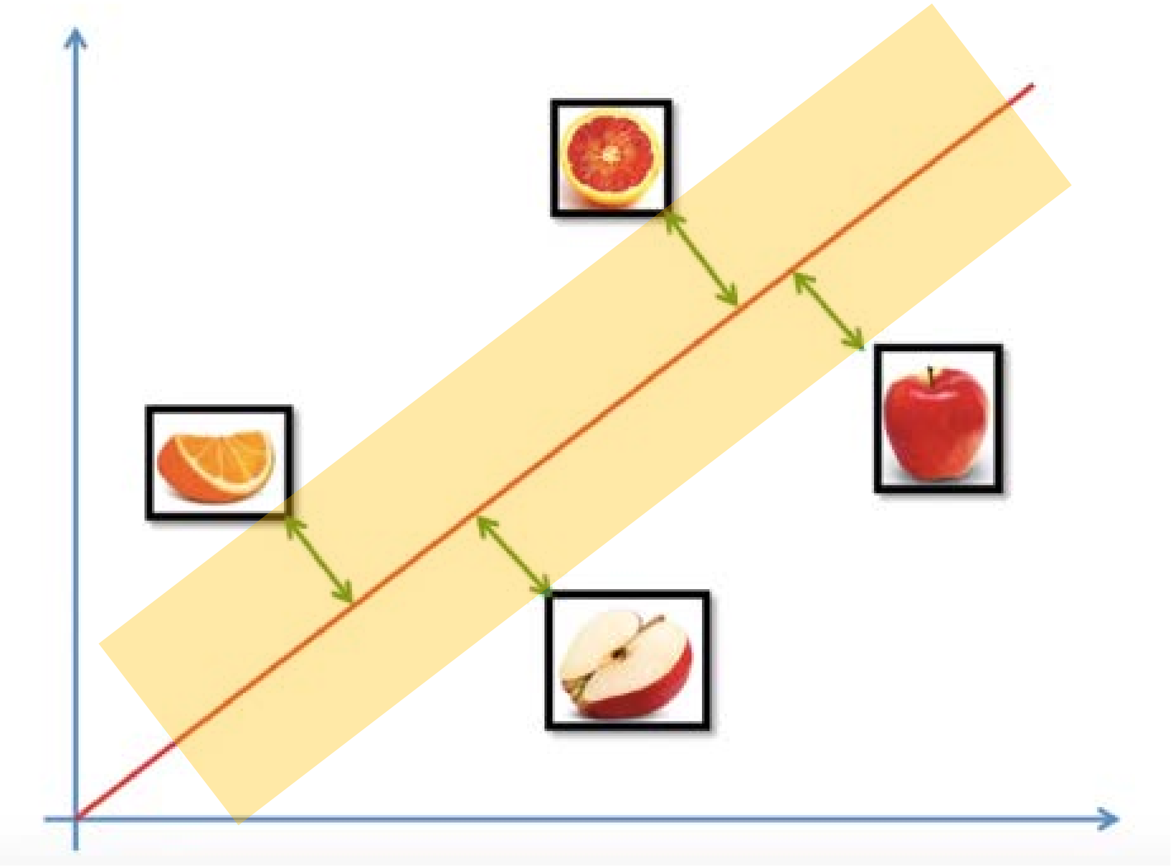


Support Vector

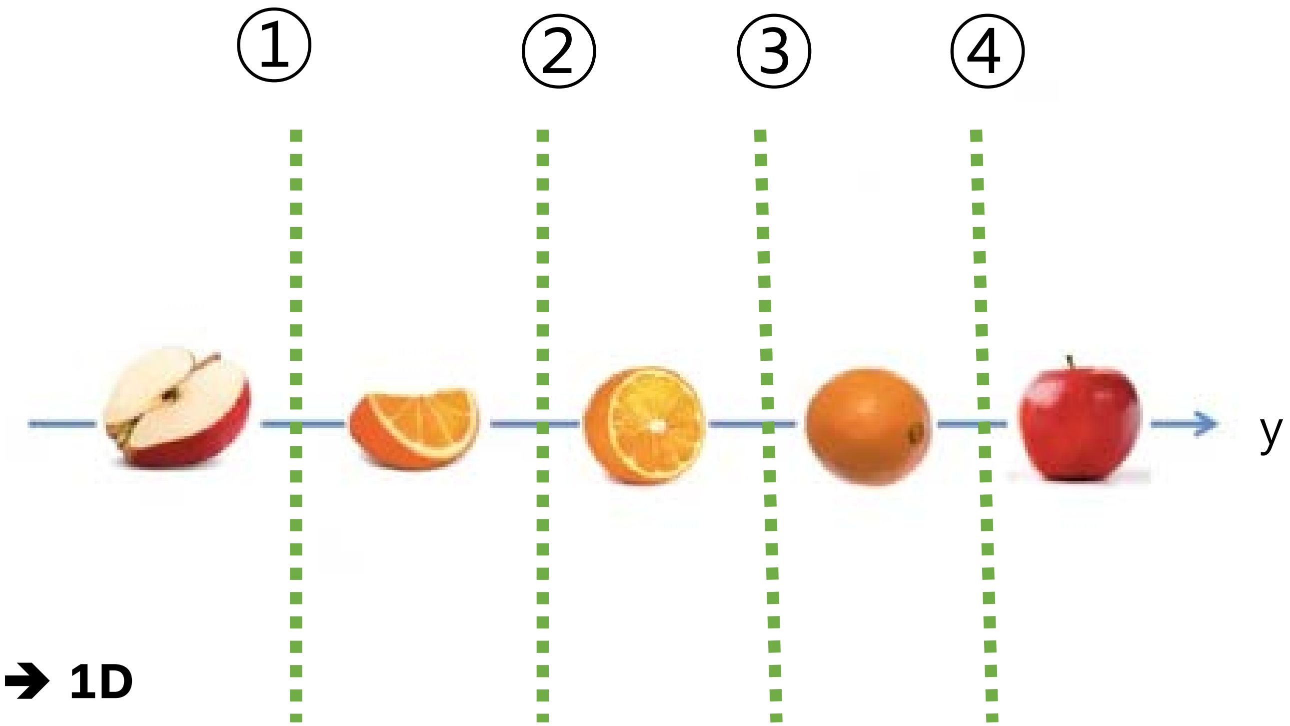
- SVM only uses support vector for prediction
  - Less computation!!!



Linearly Separable or not



What if data is not linearly separable?

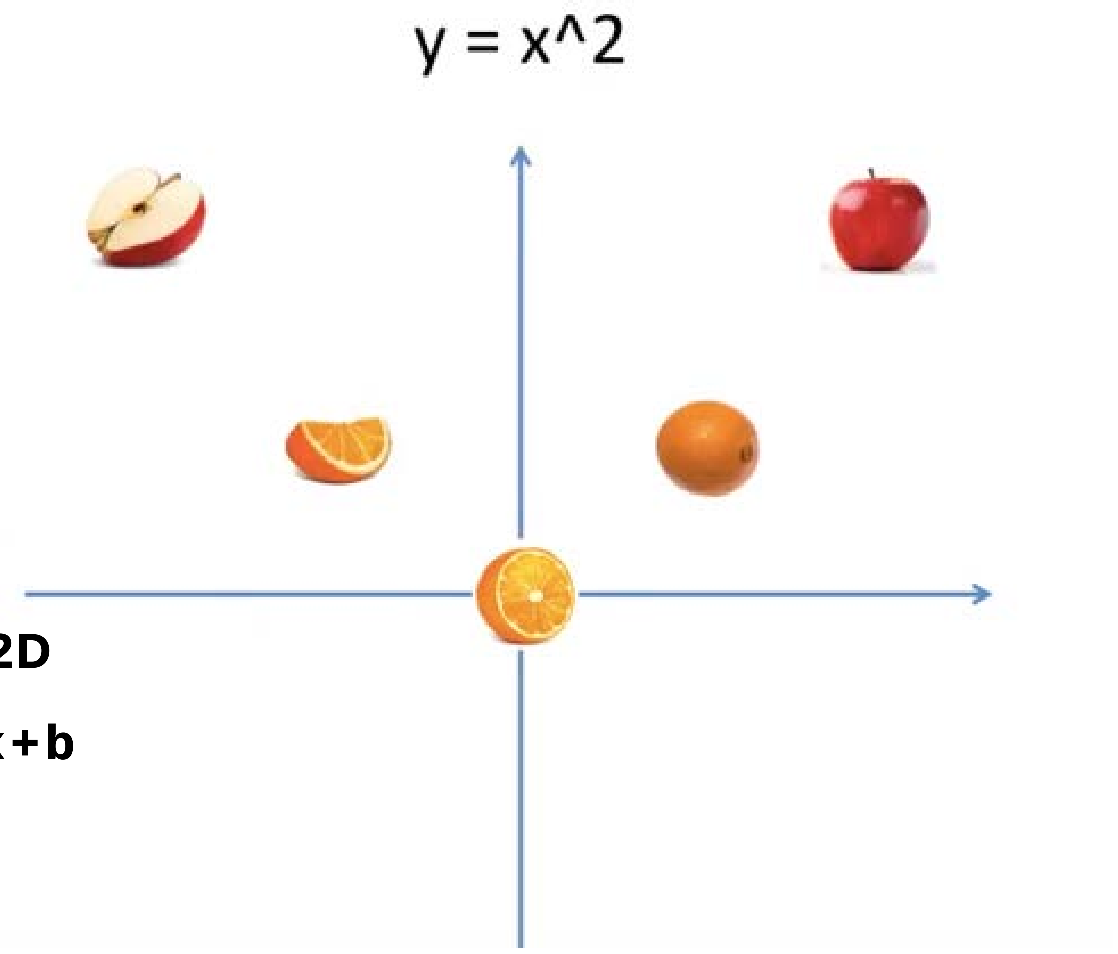


Data Embedding Space ➔ 1D

Hyperplane ➔ 0D ➔ y = b

What if data is not linearly separable?

- Mapping lower dimension to high dimension

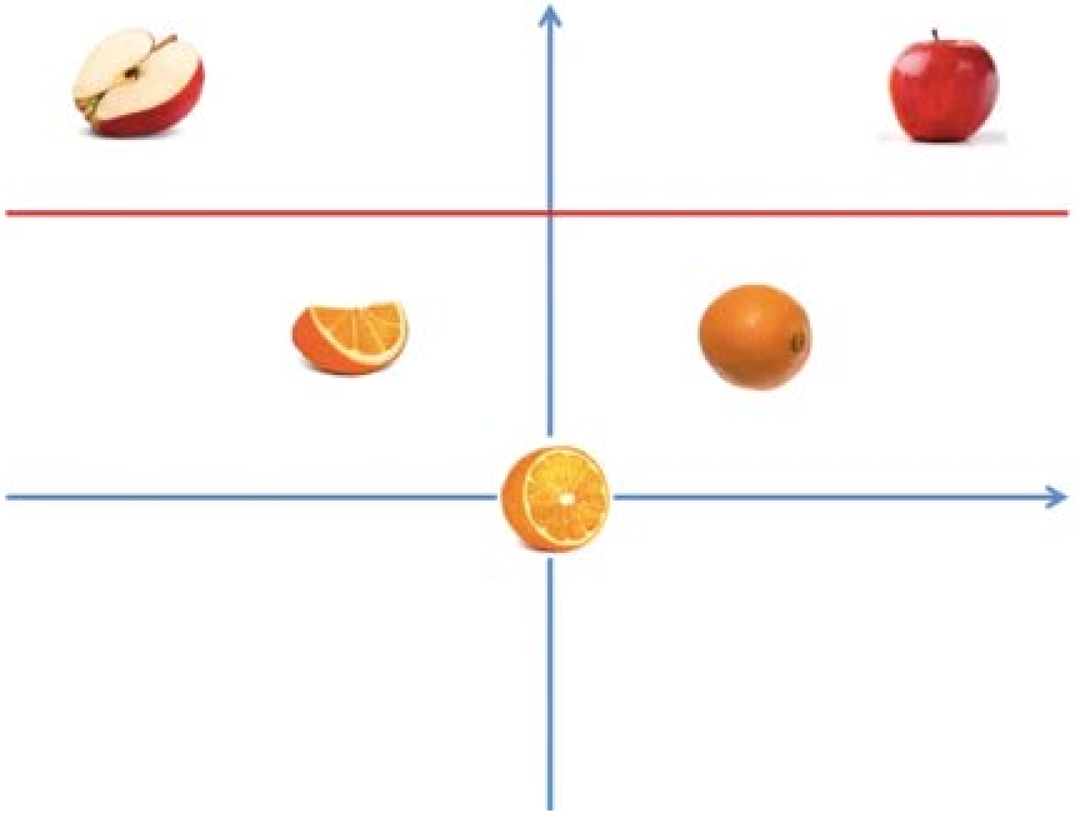


Data Embedding Space ➔ 2D

Hyperplane ➔ 1D ➔ y = ax+b

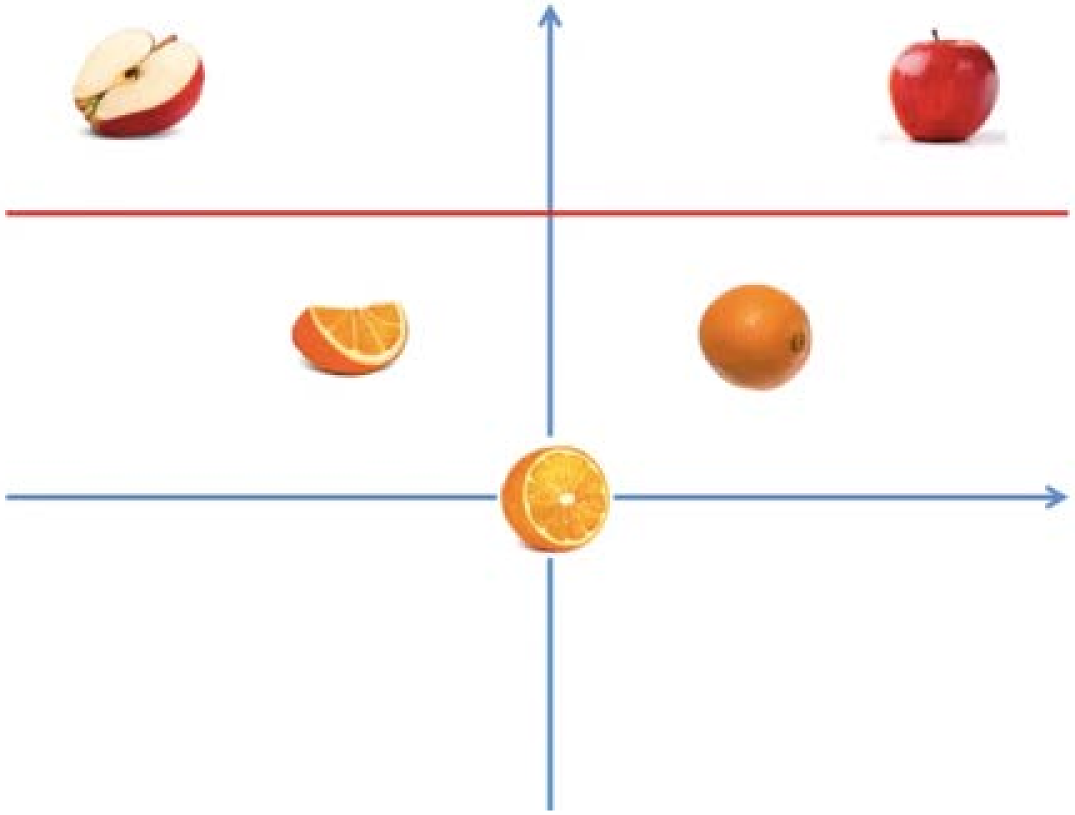
What if data is not linearly separable?

- 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**

The *kernel function* can be any of the following:

- linear: $\langle x, x' \rangle$.
- polynomial: $(\gamma \langle x, x' \rangle + r)^d$. d is specified by keyword `degree`, r by `coef0`.
- rbf: $\exp(-\gamma \|x - x'\|^2)$. γ is specified by keyword `gamma`, must be greater than 0.
- sigmoid $(\tanh(\gamma \langle x, x' \rangle + r))$, where r is specified by `coef0`.

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

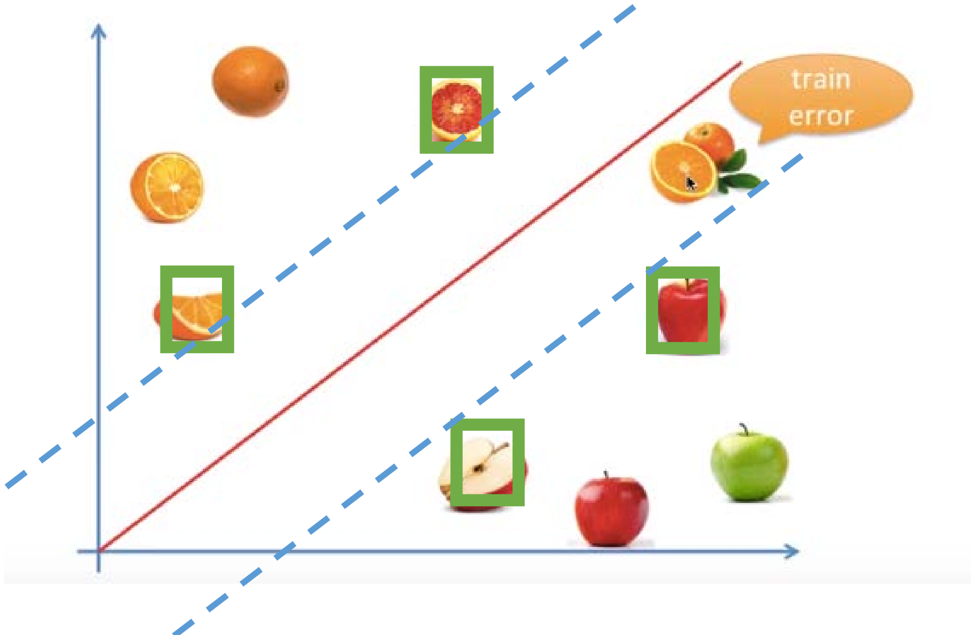
$$K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j) = x_i^T A^T A x_j$$

https://scikit-learn.org/stable/modules/svm.html

SVM Parameter - Cost

Manually hand-tuned

- Cost is small == Margin is large



$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \quad \text{subject to } y_i (w^T \phi(x_i) + b) \geq 1 - \zeta_i, \quad \zeta_i \geq 0, i = 1, \ldots, n$$

Margin width misclassification

C is small

Training error is allowed

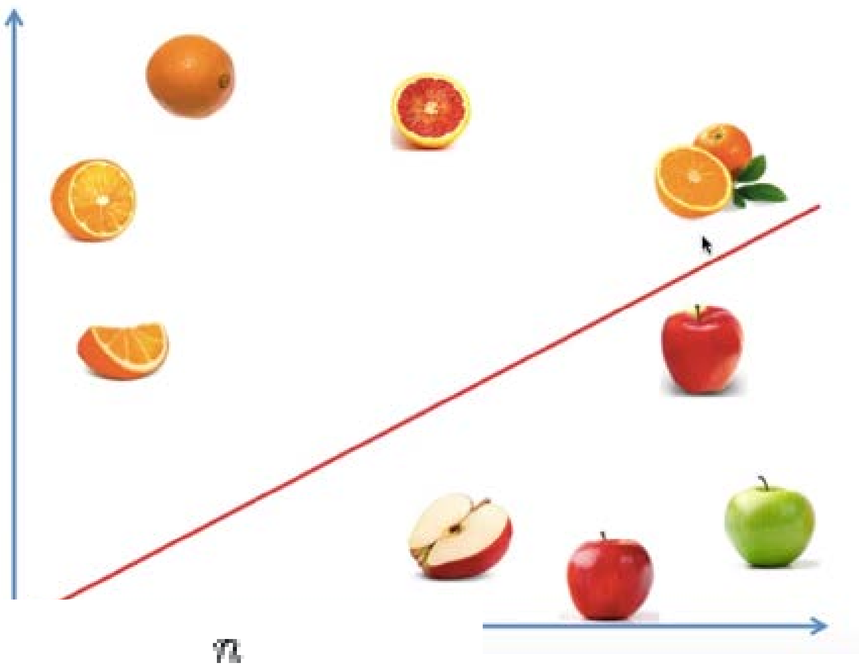
Overfitting is not allowed

Margin is large

Testing error is small

SVM Parameter - Cost

- Cost is large == Margin is small



C is large

Training error is not allowed

Overfitting is allowed

Margin is small

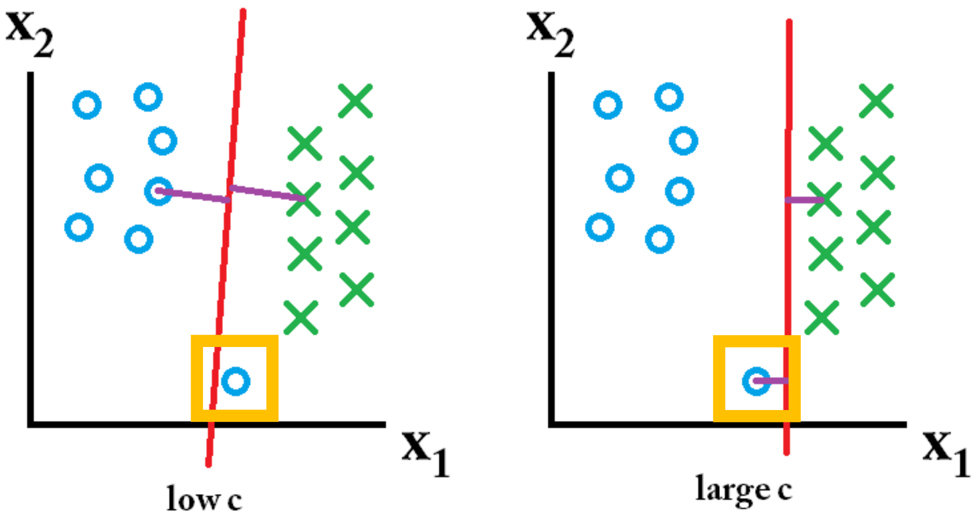
Testing error is large

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \quad \text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i, \quad \zeta_i \geq 0, i = 1, \ldots, n$$

Margin width misclassification

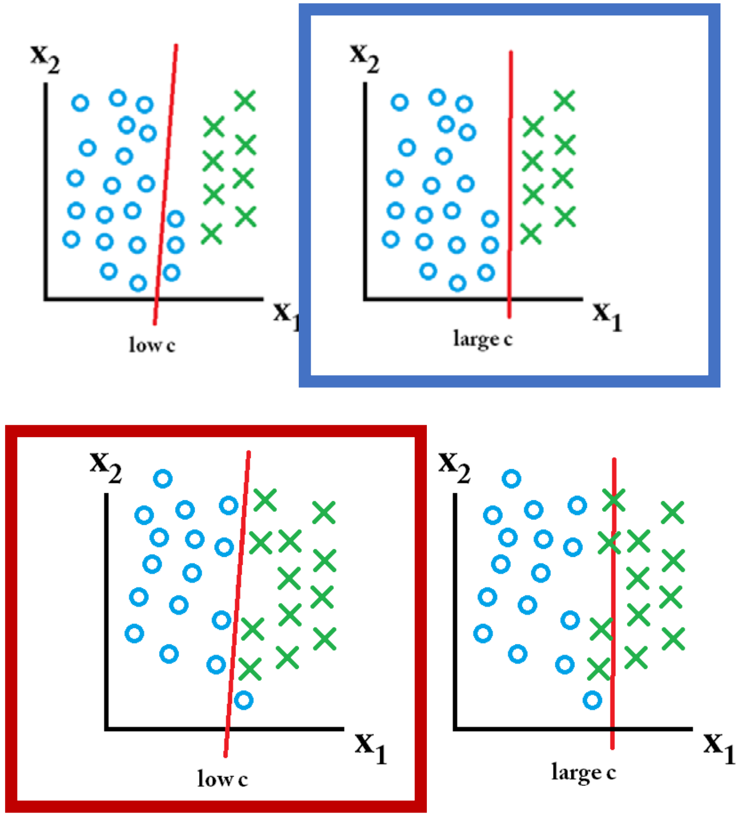
SVM Parameter - Cost

- 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?



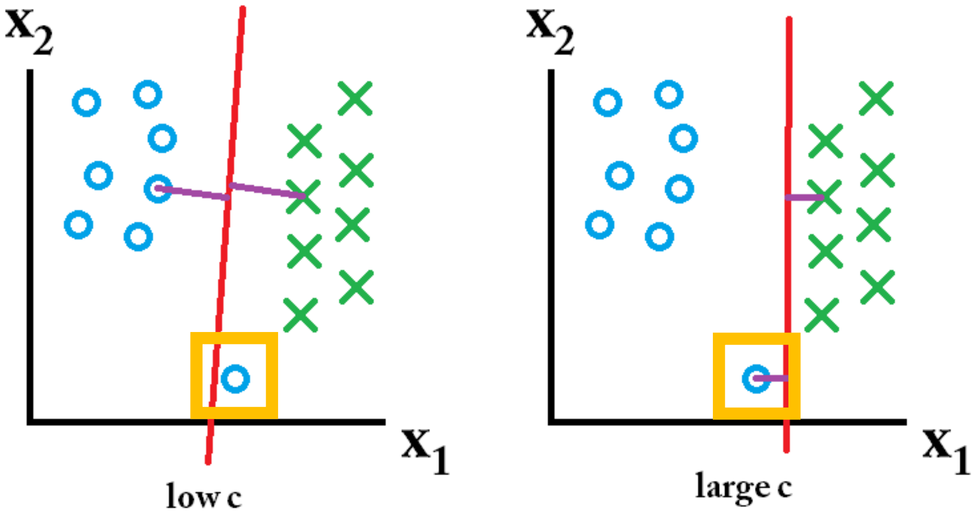
Inlier or outlier

?



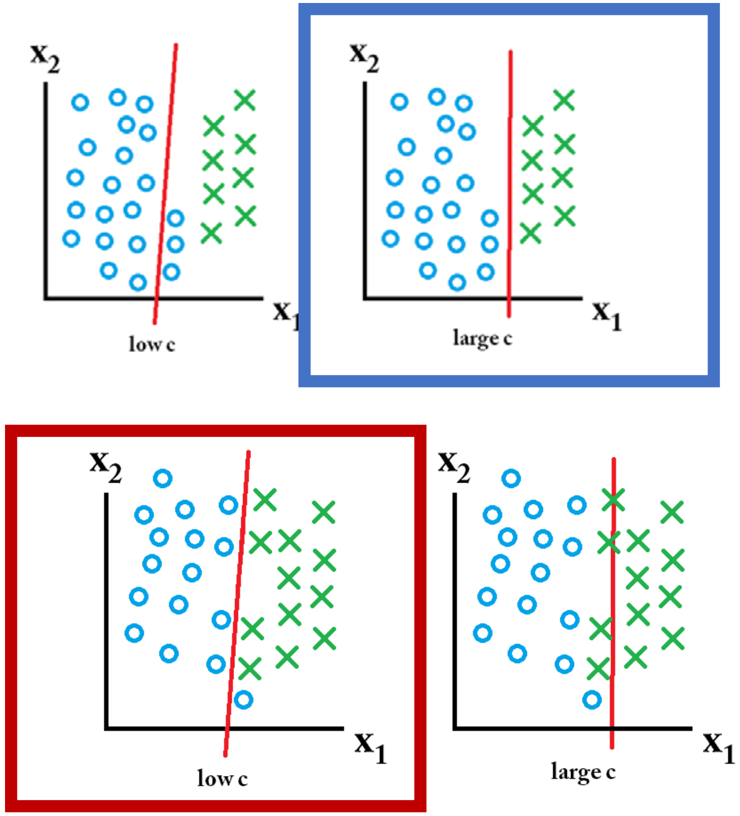
SVM Parameter - Cost

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

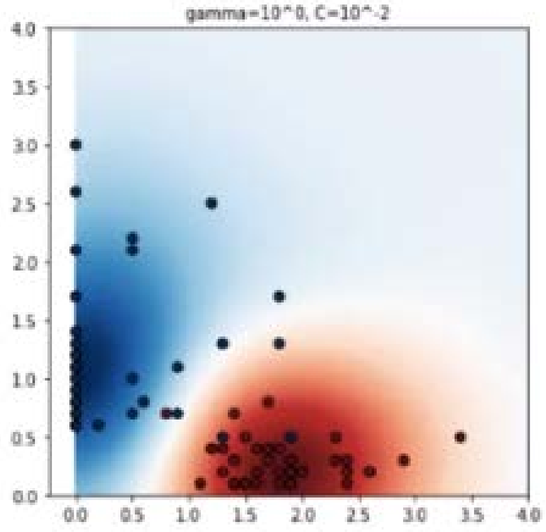


Inlier or outlier

?

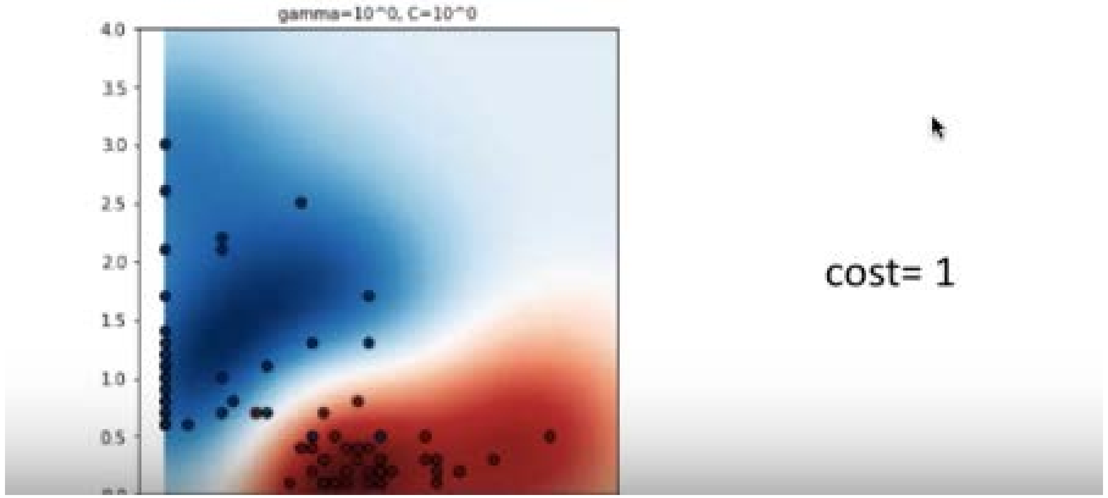


SVM Parameter - Cost



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

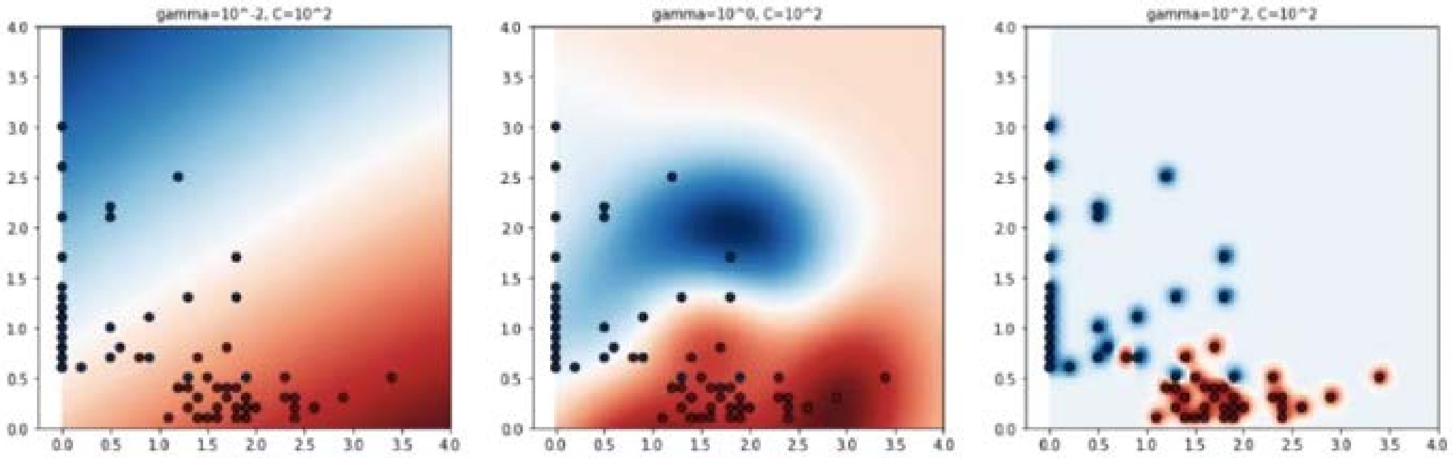
SVM parameter – Gamma in RBF kernel

- 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'.

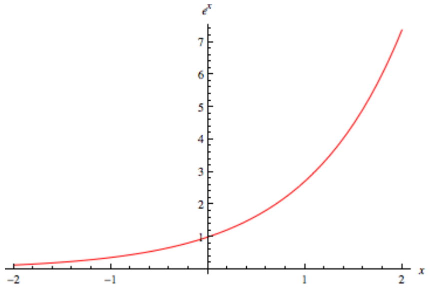
Radial Base Function (also called Gaussian Kernel)

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

Far **Close**

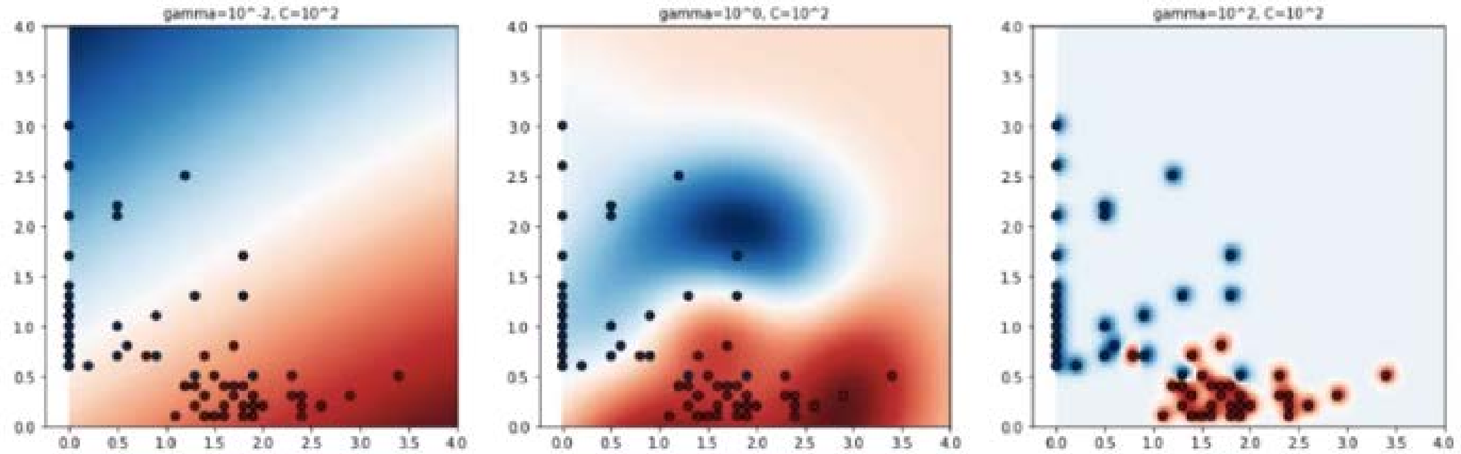


gamma = 0.01 gamma = 1 gamma = 100



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

Pattern Recognition

SVM 개념 잡기 II

Yukyung Choi

yk.choi@rcv.sejong.ac.kr

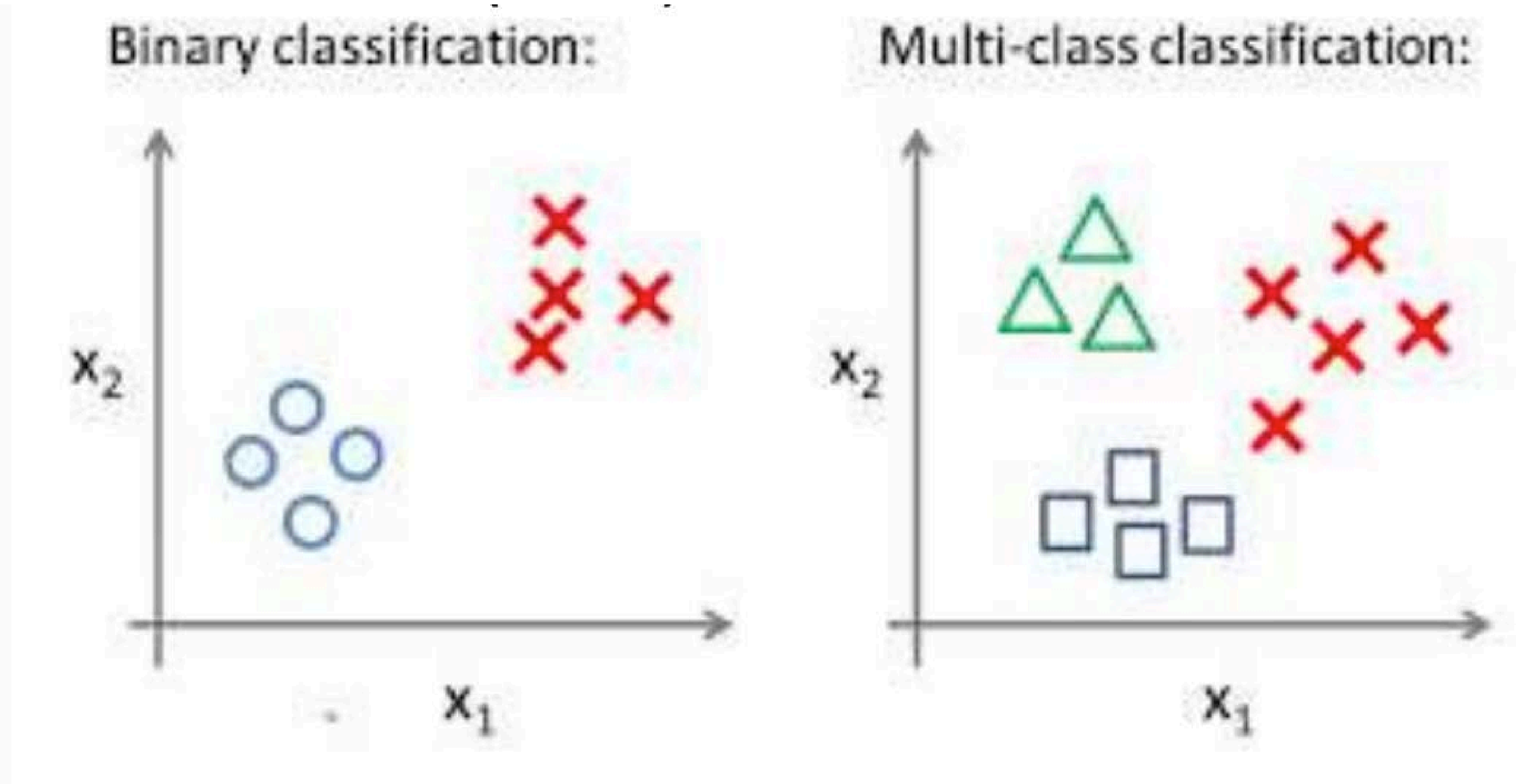
Previous Work

- What is a support vector
- What is a best decision boundary
- Hard-SVM
- Soft-SVM
- Parameter C
- Parameter gamma

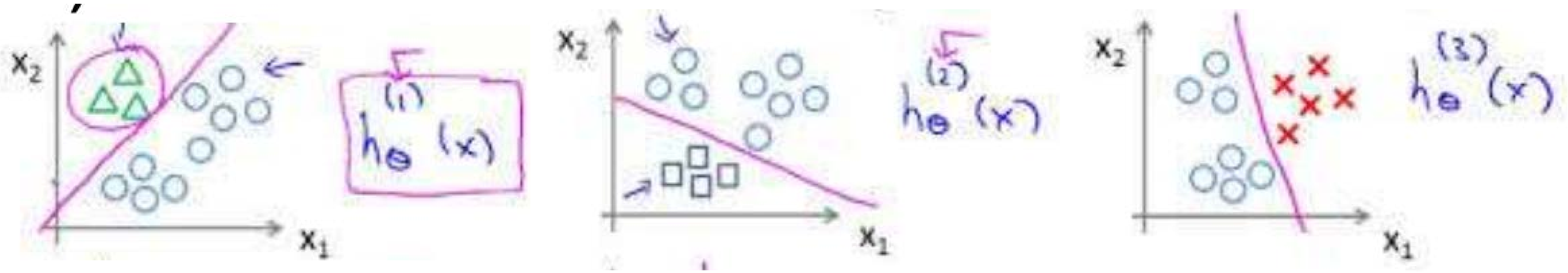
Today Lecture

- Multiclass SVM
  - One vs One
  - One vs Rest (One vs All)
- SVM with Unbalanced Data
- SVM Optimizer

Multiclass Classification



One-vs-Rest (OVR)



One-vs-Rest (OVR)

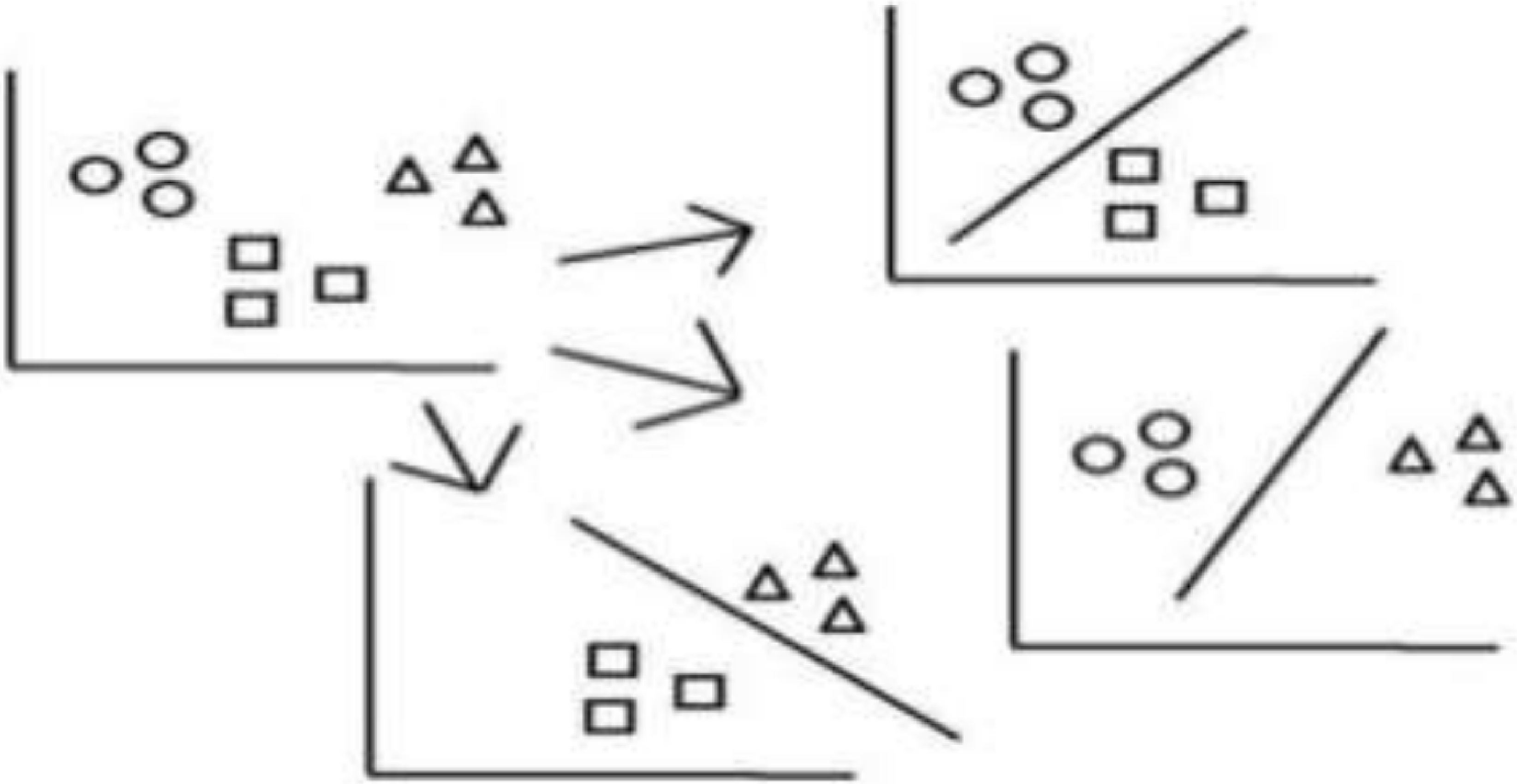
Training: Fits one classifier per class against all other data as a negative class. In total K classifiers.

Prediction: applies K classifiers to a new data point. Selects the one that got a positive class. In case of ties, selects the class with highest confidence.

Pros:

- Efficient
- Interpretable

One vs One (OvO)



One vs One (OvO)

<u>Training:</u> Fits (K-1) classifier per class against each other class. In total K*(K-1)/2 classifiers.

<u>Prediction:</u> applies K*(K-1)/2 classifiers to a new data point. Selects the class that got the majority of votes ("+1"). In case of ties, selects the class with highest confidence.

Pros:

- Used for Kernel algorithms (e.g. "SVM").

Cons:

- Not as fast as OVR

Sklearn SVM: OvO vs OvR

```
>>> X = [[0], [1], [2], [3]]
>>> Y = [0, 1, 2, 3]
>>> clf = svm.SVC(gamma='scale', decision_function_shape='ovo')
>>> clf.fit(X, Y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovo', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
>>> dec = clf.decision_function([[1]])
>>> dec.shape[1] # 4 classes: 4*3/2 = 6
6
>>> clf.decision_function_shape = "ovr"
>>> dec = clf.decision_function([[1]])
>>> dec.shape[1] # 4 classes
4
```

```
>>> lin_clf = svm.LinearSVC()
>>> lin_clf.fit(X, Y)
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0)
>>> dec = lin_clf.decision_function([[1]])
>>> dec.shape[1]
4
```

Similar to SVC with parameter kernel=’linear’, but implemented in terms of liblinear rather than libsvm

Sklearn SVM: OvO vs OvR

Below is an example of multiclass learning using OvO:

```
>>> from sklearn import datasets
>>> from sklearn.multiclass import OneVsOneClassifier
>>> from sklearn.svm import LinearSVC
>>> iris = datasets.load_iris()
>>> X, y = iris.data, iris.target
>>> OneVsOneClassifier(LinearSVC(random_state=0)).fit(X, y).predict(X)
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

«

1.12.2.1. Multiclass learning

Below is an example of multiclass learning using OvR:

```
>>> from sklearn import datasets
>>> from sklearn.multiclass import OneVsRestClassifier
>>> from sklearn.svm import LinearSVC
>>> iris = datasets.load_iris()
>>> X, y = iris.data, iris.target
>>> OneVsRestClassifier(LinearSVC(random_state=0)).fit(X, y).predict(X)
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Colab 실습

- 지난 시간 SVM 열어 확인

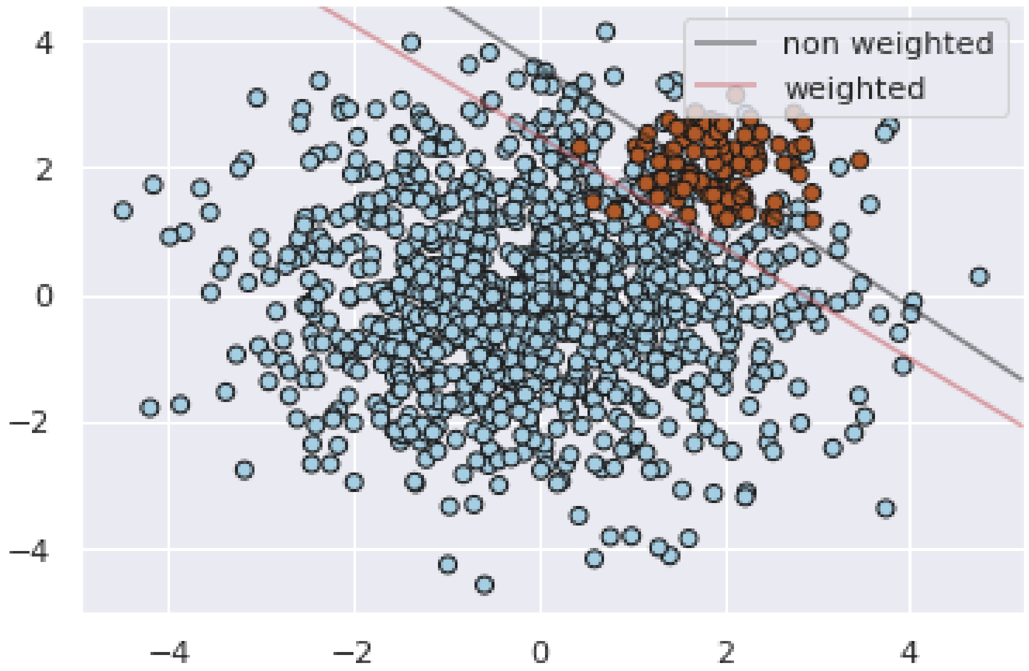
Unbalanced problems

- Sklearn: class_weight

```
n_samples_1 = 1000
n_samples_2 = 100

# 1번 클래스에 10배 가중치;;
# 샘플 비율 차이만큼 가중치

wclf = svm.SVC(kernel='linear', class_weight={1: 10})
wclf.fit(X, y)
```



Skearn SVM OvR 에서는 default로 Unbalanced Problem을 해결해주지 않음

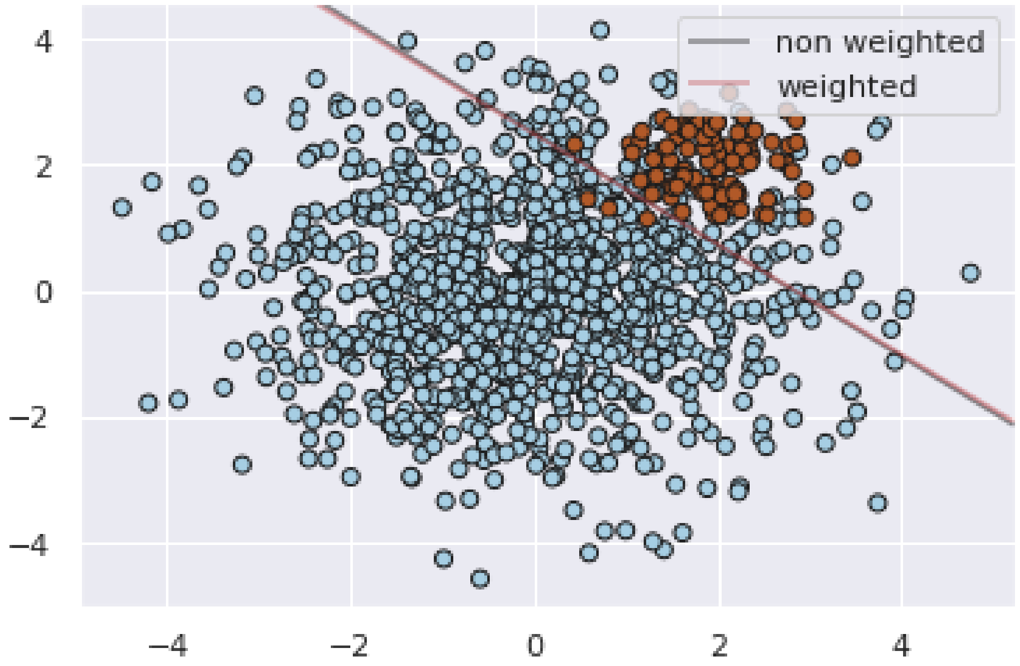
Unbalanced problems

- Sklearn: class_weight

n_samples_1 = 1000
n_samples_2 = 100

1번 클래스에 10배 가중치;;
샘플 비율 차이만큼 가중치

wclf = svm.SVC(kernel='linear', class_weight='balanced')
wclf.fit(X, y)



Skearn SVM 옵션으로 class_weight='balanced'

Colab 실습

- 지난 시간 SVM 열어 확인

Pattern Recognition

SVM 개념 잡기 III

Yukyung Choi

yk.choi@rcv.sejong.ac.kr

SVM Optimization

❖ 여백은 아래와 같이 공식화

$$\text{여백} = 2h = \frac{2|d(\mathbf{x})|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \qquad (5.4)$$

❖ 이제 문제를 **조건부 최적화 문제**로 공식화

• 조건부 최적화 문제

$$\left.\begin{array}{l} \text{아래 조건 하에,} \\ \quad \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \geq 1, \forall \mathbf{x}_i \in \omega_1 \\ \quad \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \leq -1, \forall \mathbf{x}_i \in \omega_2 \\ \dfrac{2}{\|\mathbf{w}\|} \text{를 최대화하라.} \end{array}\right\} \qquad (5.5)$$

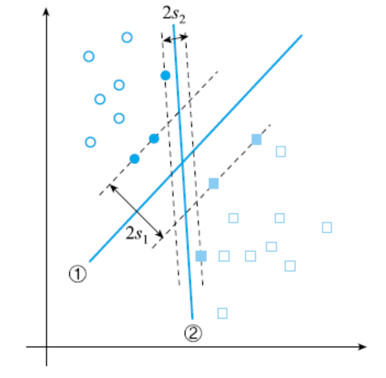


그림 5.3 선형 분리 가능한 상황

❖ 훈련 집합 $X=\{(\mathbf{x}_1, t_1),\ldots,(\mathbf{x}_N, t_N)\}$

SVM Optimization

❖ (5.5)를 간단한 형태로 변형하면,

- 조건부 최적화 문제

$$\left.\begin{array}{l}\text{아래 조건 하에,}\\ \quad t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-1\geq 0,\ i=1,\cdots,N\\ J(\mathbf{w})=\frac{1}{2}\|\mathbf{w}\|^2 \text{ 을 최소화하라.}\end{array}\right\} \qquad (5.6)$$

$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b\geq 1, \forall \mathbf{x}_i \in \omega_1$
$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b\leq -1, \forall \mathbf{x}_i \in \omega_2$
t_i 를 통해 합쳐 쓰고

최대화 문제 ➔ 최소화 문제로 변경

❖ 문제의 특성

- ❖ 해의 유일성
  - ❖ (5.6)은 볼록이므로 해는 유일하다.
  - ❖ 따라서 구한 해는 전역 최적 점을 보장한다. (Convex Optimization!!)
- ❖ 문제의 난이도
  - ❖ *N*개의 선형 부등식을 조건으로 가진 2차 함수의 최적화 문제
- ❖ **조건부 최적화 문제는 "라그랑제 승수"**로 푼다. (11.2.3 절)

SVM Optimization

- ❖ 문제의 볼록 성질을 이용하여 풀기 쉬운 형태로 변환 (prime ➔ dual)

  - ❖ 볼록 성질을 만족하는 조건부 최적화 문제는 **Wolfe 듀얼**로 변형할 수 있다.

볼록 성질을 만족하는 조건부 최적화 문제는 Wolfe 듀얼 문제로 변형할 수 있다. 원래 문제가 $f_i(\boldsymbol{\theta}) \geq 0,\ i = 1, \cdots, N$이라는 조건 하에 $J(\boldsymbol{\theta})$를 최소화하는 것이라 하자. 이때 Wolfe 듀얼 문제는 $\partial L(\boldsymbol{\theta}, \boldsymbol{\alpha}) / \partial \boldsymbol{\theta} = \mathbf{0}$과 $\alpha_i \geq 0,\ i = 1 \cdots, N$이라는 두 가지 조건 하에 $\boxed{L(\boldsymbol{\theta}, \boldsymbol{\alpha}) = J(\boldsymbol{\theta}) - \sum_{i=1,N} \alpha_i f_i(\boldsymbol{\theta})}$를 최대화하는 것이다. 부등식 조건이 등식 조건으로 바뀌었고 최소화 문제가 최대화 문제로 바뀌었다.

SVM Optimization

❖ **라그랑제 승수 방법**

① 목적 함수와 조건을 하나의 식 (즉, 라그랑제 함수 *L*)으로 만들고,

② KKT조건을 이용 (변수 제거)

③ 라그랑제 함수를을 최적화하는 해를 구한다. (α_i는 라그랑제 승수만으로 구성된 식)

① 목적 함수와 조건을 하나의 식 (즉, 라그랑제 함수 *L*)으로 만들고,

라그랑제 함수 $L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = J(\boldsymbol{\theta}) - \sum_{i=1,N} \lambda_i f_i(\boldsymbol{\theta})$

f(w,b)

$$t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1 \geq 0, \; i = 1, \cdots, N$$

$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ 을 최소화하라.

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i (t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1) \qquad (5.7)$$

SVM Optimization

② KKT조건을 이용 (변수 제거)

❖ (5.7)의 KKT 조건 →

① $\partial L(\boldsymbol{\theta}, \boldsymbol{\lambda}) / \partial \boldsymbol{\theta} = \mathbf{0}$

② $\lambda_i \geq 0,\ i = 1, \cdots, N$

③ $\lambda_i f_i(\boldsymbol{\theta}) = 0,\ i = 1, \cdots, N$

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i (t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1)$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{0} \quad \rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \mathbf{x}_i \qquad (5.8)$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = 0 \quad \rightarrow \quad \sum_{i=1}^{N} \alpha_i t_i = 0 \qquad (5.9)$$

$$\alpha_i \geq 0,\ i = 1, \cdots, N \qquad (5.10)$$

$$\alpha_i (t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1) = 0,\ i = 1, \cdots, N \qquad (5.11)$$

❖ (5.11)에 의하면 모든 샘플이 α_i=0 또는 $t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b)$-1=0이어야 함. **$\alpha_i \neq 0$인 샘플이 서포트 벡터임**

❖ (5.8)에 의하면, **라그랑제 승수 α_i 알면 w 구할 수 있음 (결정 초평면을 구한 셈)**

  ❖ 이제부터 '**w** 구하는 대신 라그랑제 승수 구하는' 문제로 관심 전환

❖ (5.11)로 ***b*** **구할 수 있음**

SVM Optimization

❖ (5.6)을 Wolfe 듀얼로 바꾸어 쓰면,

- 조건부 최적화 문제

$$\left.\begin{array}{l}\text{아래 조건 하에,}\\ \quad t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1 \ge 0,\ i = 1, \cdots, N \\ J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \text{ 을 최소화하라.}\end{array}\right\} \quad (5.6)$$

❖ 5.8~5.10

- 조건부 최적화 문제

$$\left.\begin{array}{l}\text{아래 조건 하에,}\\ \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \mathbf{x}_i \\ \quad \sum_{i=1}^{N} \alpha_i t_i = 0 \\ \quad \alpha_i \ge 0,\ i = 1, 2, \ldots, N \\ L(\mathbf{w}, b, \boldsymbol{\alpha}) \text{ 를 최대화하라.}\end{array}\right\} \quad (5.12)$$

- 조건부 최적화 문제

$$\left.\begin{array}{l}\text{아래 조건 하에,}\\ \quad t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1 \ge 0,\ i = 1, \cdots, N \\ J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \text{ 을 최소화하라.}\end{array}\right\}$$

(Due to KKT Condition)

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{0} \quad \rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \mathbf{x}_i \qquad (5.8)$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = 0 \quad \rightarrow \quad \sum_{i=1}^{N} \alpha_i t_i = 0 \qquad (5.9)$$

$$\alpha_i \ge 0,\ i = 1, \cdots, N \qquad (5.10)$$

$$\alpha_i(t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1) = 0,\ i = 1, \cdots, N \qquad (5.11)$$

SVM Optimization

최소화 문제 ➔ 최대화 문제

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i (t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1) \qquad (5.7)$$

(5.9)식으로 앞 식 삭제

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i (t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1) - \frac{1}{2}\|\mathbf{w}\|^2 \qquad (5.7)$$

(5.13)

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \mathbf{x}_i \qquad (5.8)$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = 0 \rightarrow \sum_{i=1}^{N} \alpha_i t_i = 0 \qquad (5.9)$$

$$\alpha_i \geq 0, i = 1, \cdots, N \qquad (5.10)$$

$$\alpha_i (t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1) = 0, i = 1, \cdots, N \qquad (5.11)$$

❖ 간단한 수식 정리를 하면,

- 조건부 최적화 문제

아래 조건 하에,

$$\sum_{i=1}^{N} \alpha_i t_i = 0$$

$$\alpha_i \geq 0, i = 1, \cdots, N$$

$$\tilde{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j$$ 를 최대화하라.

❖ 흥미로운 특성들

- ❖ **2차 함수의 최대화 문제임**
- ❖ **w**와 b가 사라졌다. (α를 찾는 문제가 되었다.)
- ❖ 특징 벡터 $\mathbf{x}_i$가 내적 형태로 나타난다. (비선형으로 확장하는 발판)
- ❖ 목적 함수의 두번째 Σ항은 N^2개의 항을 갖는다. (여전히 풀기 어려운 문제)

SVM Optimization

- 조건부 최적화 문제

$$\left.\begin{array}{l}\text{아래 조건 하에,}\\ \quad t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1 \geq 0,\ i = 1, \cdots, N \\ J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \text{ 을 최소화하라.}\end{array}\right\} \qquad (5.6)$$

- 조건부 최적화 문제

$$\left.\begin{array}{l}\text{아래 조건 하에,}\\ \quad \sum_{i=1}^{N} \alpha_i t_i = 0 \\ \quad \alpha_i \geq 0,\ i = 1, \cdots, N \\ \tilde{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j \text{ 를 최대화하라.}\end{array}\right\} \qquad (5.13)$$

SVM Optimization

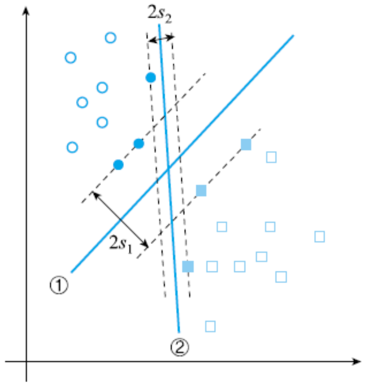


그림 5.3 선형 분리 가능한 상황

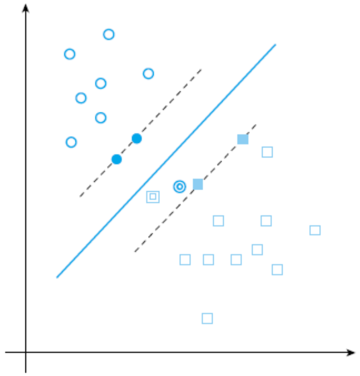


그림 5.6 선형 분리 불가능한 상황의 SVM

- 조건부 최적화 문제

아래 조건하에,

$$\left.\begin{aligned} &\sum_{i=1}^{N} \alpha_i t_i = 0 \\ &\alpha_i \ge 0,\ i = 1, \cdots, N \\ &\tilde{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j \text{ 를 최대화하라.} \end{aligned}\right\} \quad (5.13)$$

- 조건부 최적화 문제 (선형 SVM)

아래 조건하에,

$$\left.\begin{aligned} &\sum_{i=1}^{N} \alpha_i t_i = 0 \\ &\boxed{0 \le \alpha_i \le C,}\ i = 1, \cdots, N \\ &\tilde{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j \text{ 를 최대화하라.} \end{aligned}\right\} \quad (5.27)$$

쉬운 방정식으로 변경했으나, N이 크면 여전히 분석적으로 최적화 하기는 어렵다.
결국 수치적(수치해석..)으로 풀어야 한다.

Solution: Quadratic Programming(Q.P)

$$\min \frac{1}{2} x^T P x + q^T x$$
$$s.t. \quad Gx \leq h$$
$$Ax = b$$

cvxopt.solvers.qp(P, q, G, h, A, b)

$$\max_{\alpha} \sum_{i}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)} x^{(j)} >$$

$$\boxed{H_{i,j} = y^{(i)} y^{(j)} < x^{(i)} x^{(j)} >}$$

$$\max_{\alpha} \sum_{i}^{m} \alpha_i - \frac{1}{2} \alpha^T \mathbf{H} \alpha$$
$$s.t. \ \alpha_i \geq 0$$
$$\sum_{i}^{m} \alpha_i y^{(i)} = 0$$

$$\min_{\alpha} \frac{1}{2} \alpha^T \mathbf{H} \alpha - 1^T \alpha$$
$$s.t. \quad -\alpha_i \leq 0$$
$$s.t. \ y^T \alpha = 0$$

- $P := H$ a matrix of size $m \times m$
- $q := -\vec{1}$ a vector of size $m \times 1$
- $G := -diag[1]$ a diagonal matrix of -1s of size $m \times m$
- $h := \vec{0}$ a vector of zeros of size $m \times 1$
- $A := y$ the label vector of size $m \times 1$
- $b := 0$ a scalar

Computing matrix **H**

Consider the simple example with 2 input samples $\{x^{(1)}, x^{(2)}\} \in \mathbb{R}^2$ which are two dimensional vectors. i.e. $x^{(1)} = (x_1^{(1)}, x_2^{(1)})^T$

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(2)} & x_2^{(2)} \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix}$$

We now proceed to creating a new matrix X' where each input sample x is multiplied by the corresponding output label y. This can be done easily in Numpy using vectorization and padding.

$$X' = \begin{bmatrix} x_1^{(1)} y^{(1)} & x_2^{(1)} y^{(1)} \\ x_1^{(2)} y^{(2)} & x_2^{(2)} y^{(2)} \end{bmatrix}$$

$\boxed{\Rightarrow \mathbf{X' = y \cdot x}}$

Finally we take the **matrix multiplication** of X' and its transpose giving $\boxed{H = X'X'^T}$

$$H = X' @ X'^T = \begin{bmatrix} x_1^{(1)} y^{(1)} & x_2^{(1)} y^{(1)} \\ x_1^{(2)} y^{(2)} & x_2^{(2)} y^{(2)} \end{bmatrix} \begin{bmatrix} x_1^{(1)} y^{(1)} & x_1^{(2)} y^{(2)} \\ x_2^{(1)} y^{(1)} & x_2^{(2)} y^{(2)} \end{bmatrix}$$

$$H = \begin{bmatrix} x_1^{(1)} x_1^{(1)} y^{(1)} y^{(1)} + x_2^{(1)} x_2^{(1)} y^{(1)} y^{(1)} & x_1^{(1)} x_1^{(2)} y^{(1)} y^{(2)} + x_2^{(1)} x_2^{(2)} y^{(1)} y^{(2)} \\ x_1^{(2)} x_1^{(1)} y^{(2)} y^{(1)} + x_2^{(2)} x_2^{(1)} y^{(2)} y^{(1)} & x_1^{(2)} x_1^{(2)} y^{(2)} y^{(2)} + x_2^{(2)} x_2^{(2)} y^{(2)} y^{(2)} \end{bmatrix}$$

SVM Implementation from scratch

```
#Initializing values and computing H. Note the 1. to force to float type
m,n = X.shape
y = y.reshape(-1,1) * 1.
X_dash = y * X
H = np.dot(X_dash , X_dash.T) * 1.

#Converting into cvxopt format
P = cvxopt_matrix(H)
q = cvxopt_matrix(-np.ones((m, 1)))
G = cvxopt_matrix(-np.eye(m))
h = cvxopt_matrix(np.zeros(m))
A = cvxopt_matrix(y.reshape(1, -1))
b = cvxopt_matrix(np.zeros(1))

#Setting solver parameters (change default to decrease tolerance)
cvxopt_solvers.options['show_progress'] = False
cvxopt_solvers.options['abstol'] = 1e-10
cvxopt_solvers.options['reltol'] = 1e-10
cvxopt_solvers.options['feastol'] = 1e-10

#Run solver
sol = cvxopt_solvers.qp(P, q, G, h, A, b)
alphas = np.array(sol['x'])
```

$$\min_{\alpha} \frac{1}{2}\alpha^T \mathbf{H} \alpha - 1^T \alpha$$

$$s.t. \quad -\alpha_i \leq 0$$

$$s.t. \quad y^T \alpha = 0$$

- $P := H$ a matrix of size $m \times m$
- $q := -\vec{1}$ a vector of size $m \times 1$
- $G := -diag[1]$ a diagonal matrix of -1s of size $m \times m$
- $h := \vec{0}$ a vector of zeros of size $m \times 1$
- $A := y$ the label vector of size $m \times 1$
- $b := 0$ a scalar

Colab 실습

- SVM_CVXOPT