

# 데이터사이언스응용 (Capstone design)

김응희

[ehkim@sunmoon.ac.kr](mailto:ehkim@sunmoon.ac.kr)

Week 11

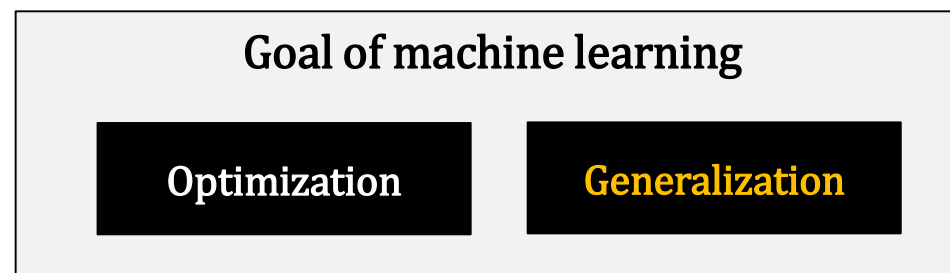
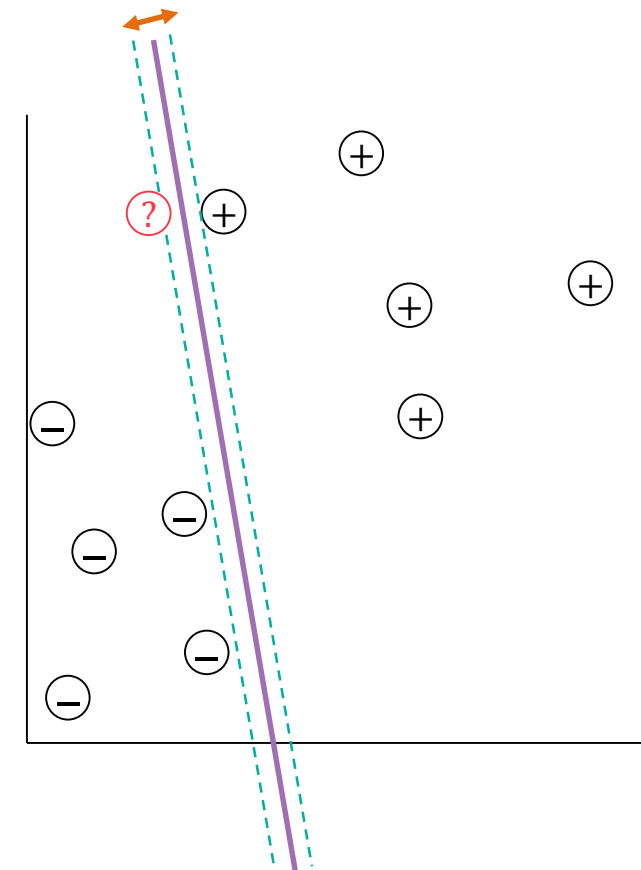
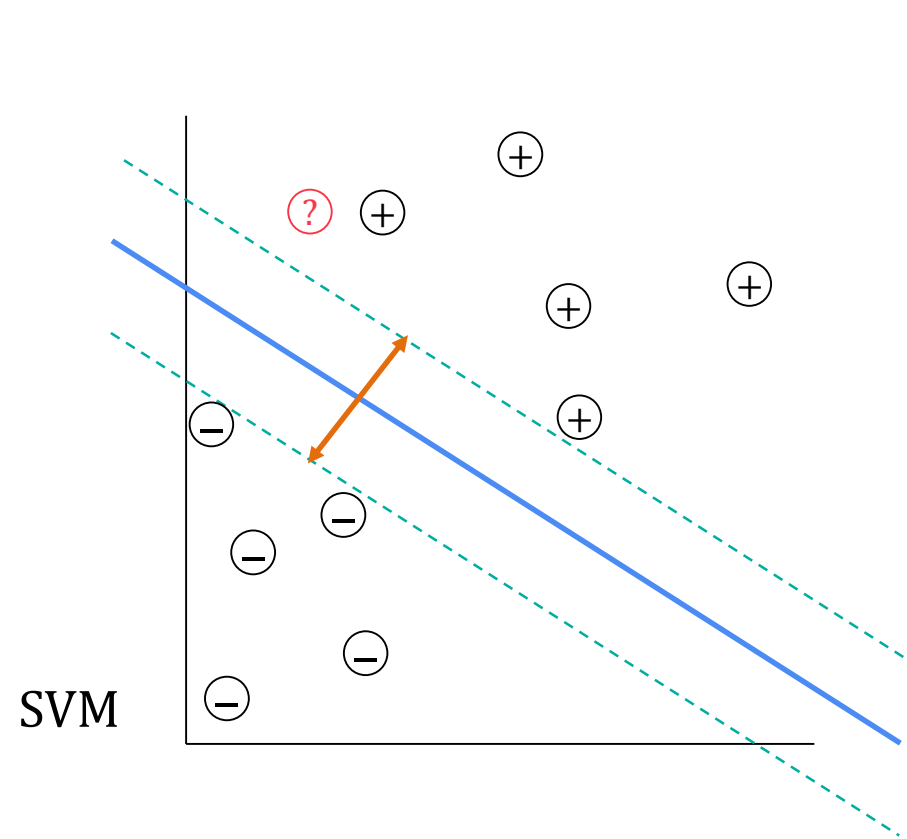
Last week,

순번	팀명	발표 날짜
1	YOLO	11.04
2	AKI	11.04
3	Harmony	11.04
4	안시성	11.04
5	H:J	11.04/11.06
6	Ajsoftware	11.06
7	제니리아	11.06

발표 시간	질의응답 시간	1팀 당 소요 시간	전체 소요 시간
15분	5분	20분	140분



3 weeks ago,



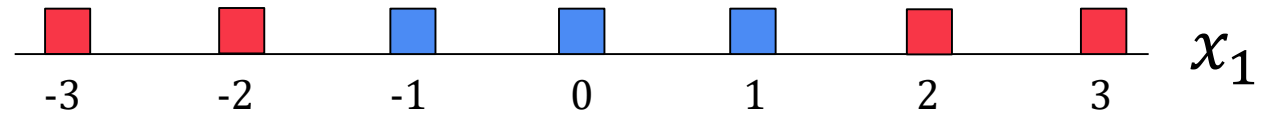
3 weeks ago,

**Main learning** is about

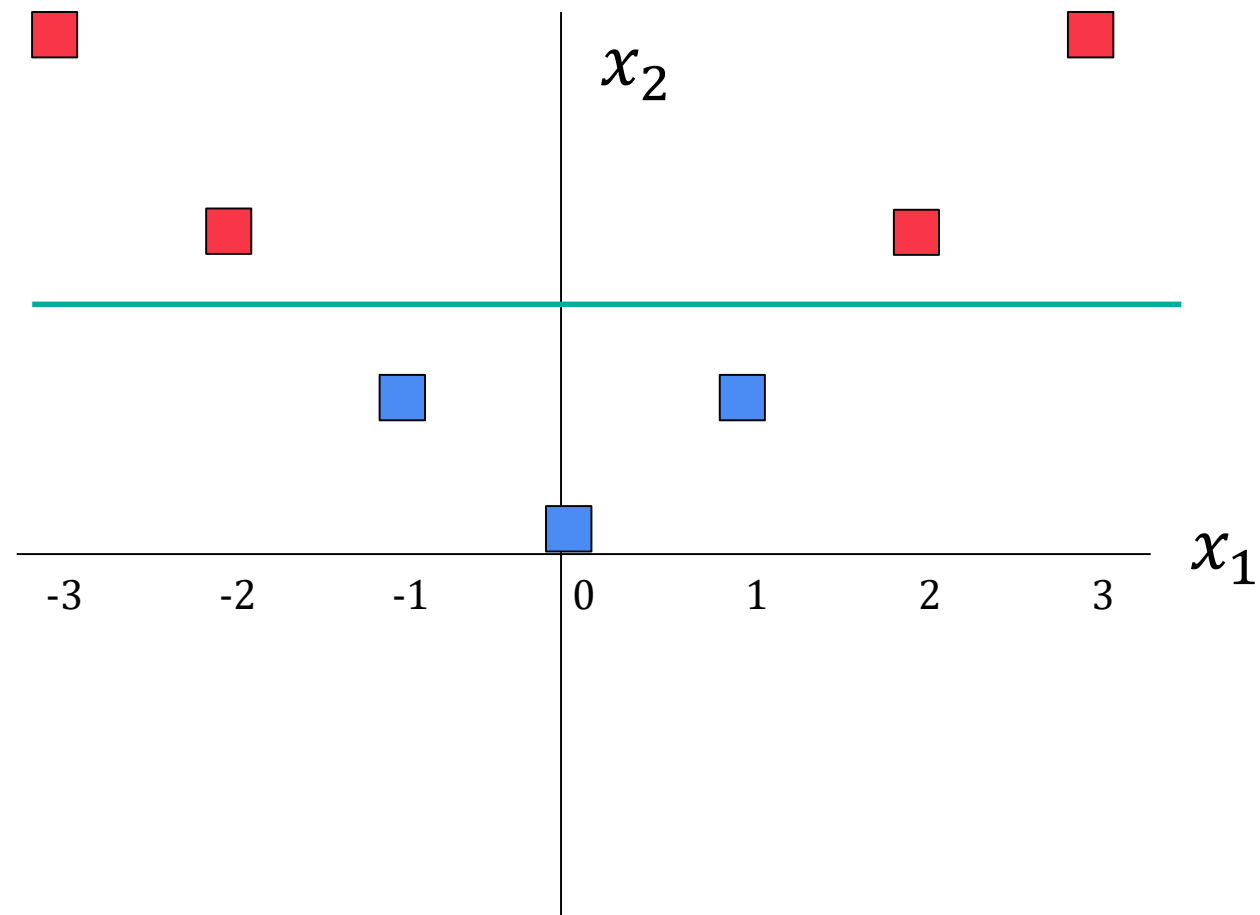
SPACE

.

2 weeks ago,



Kernel function/trick



2 weeks ago,

---

Fisher kernel

---

Graph kernels

---

Kernel smoother

Polynomial kernel

---

Radial basis function kernel (RBF)

---

String kernels

---

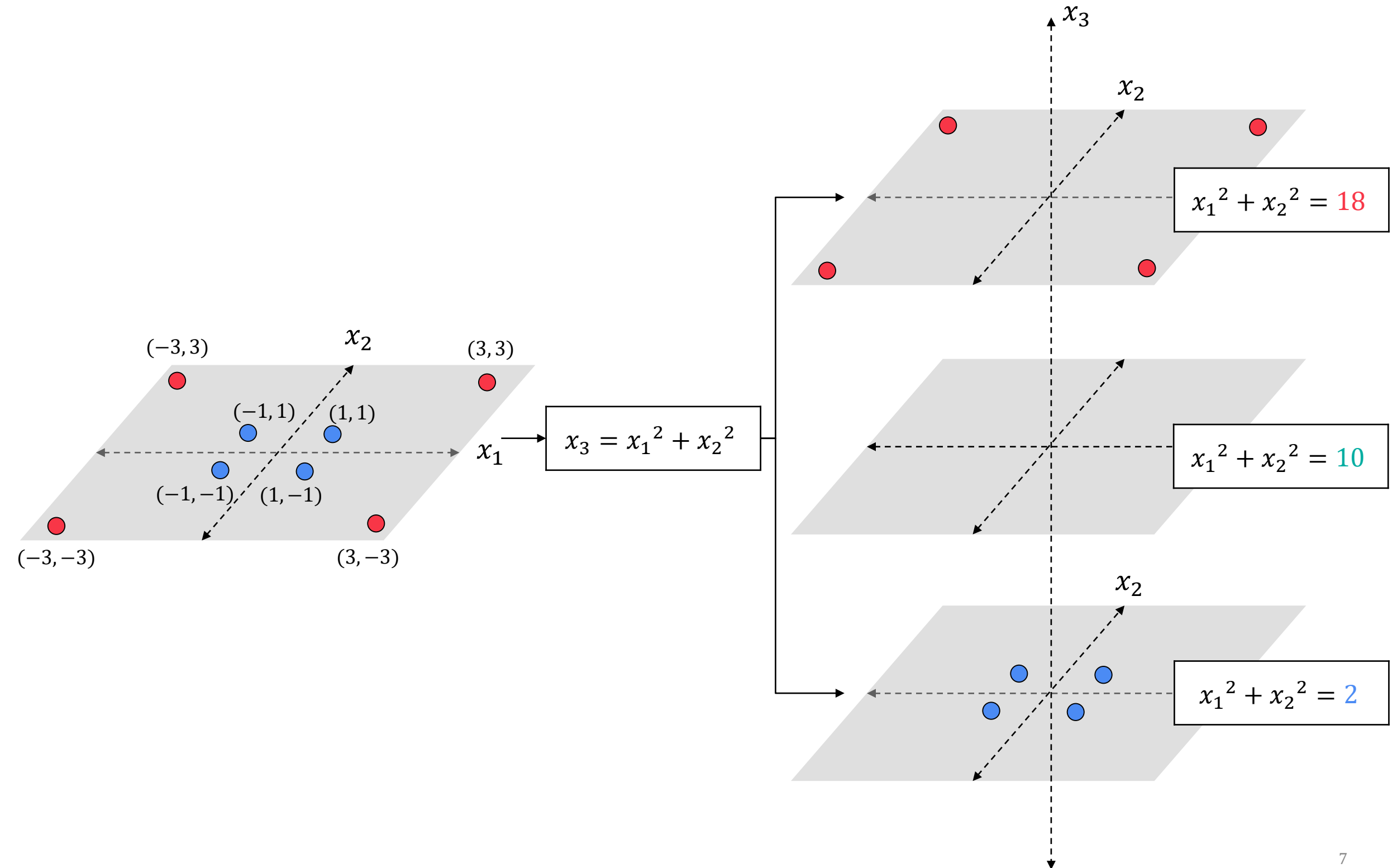
Neural tangent kernel

---

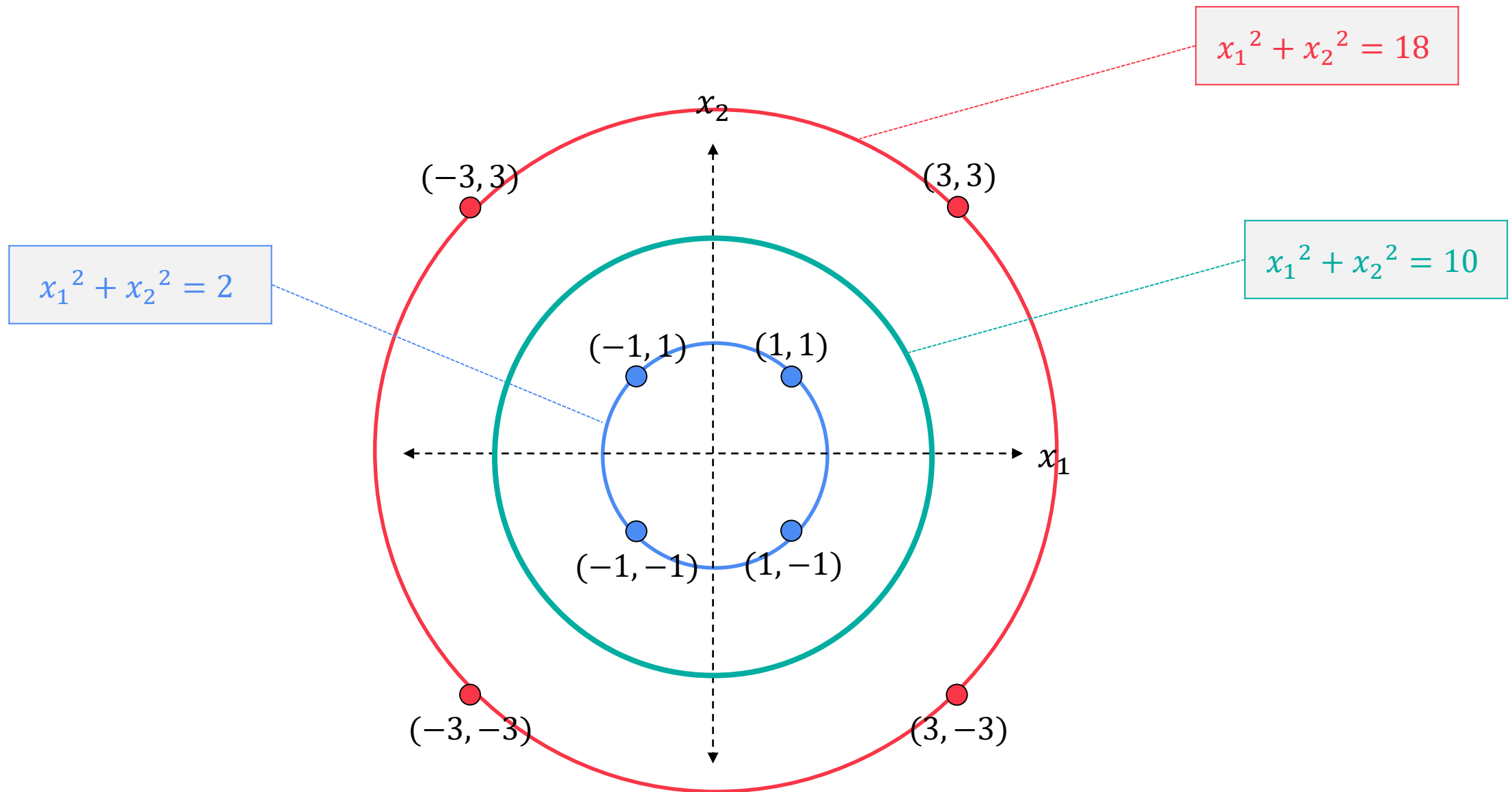
Neural network Gaussian process (NNGP) kernel

---

2 weeks ago,



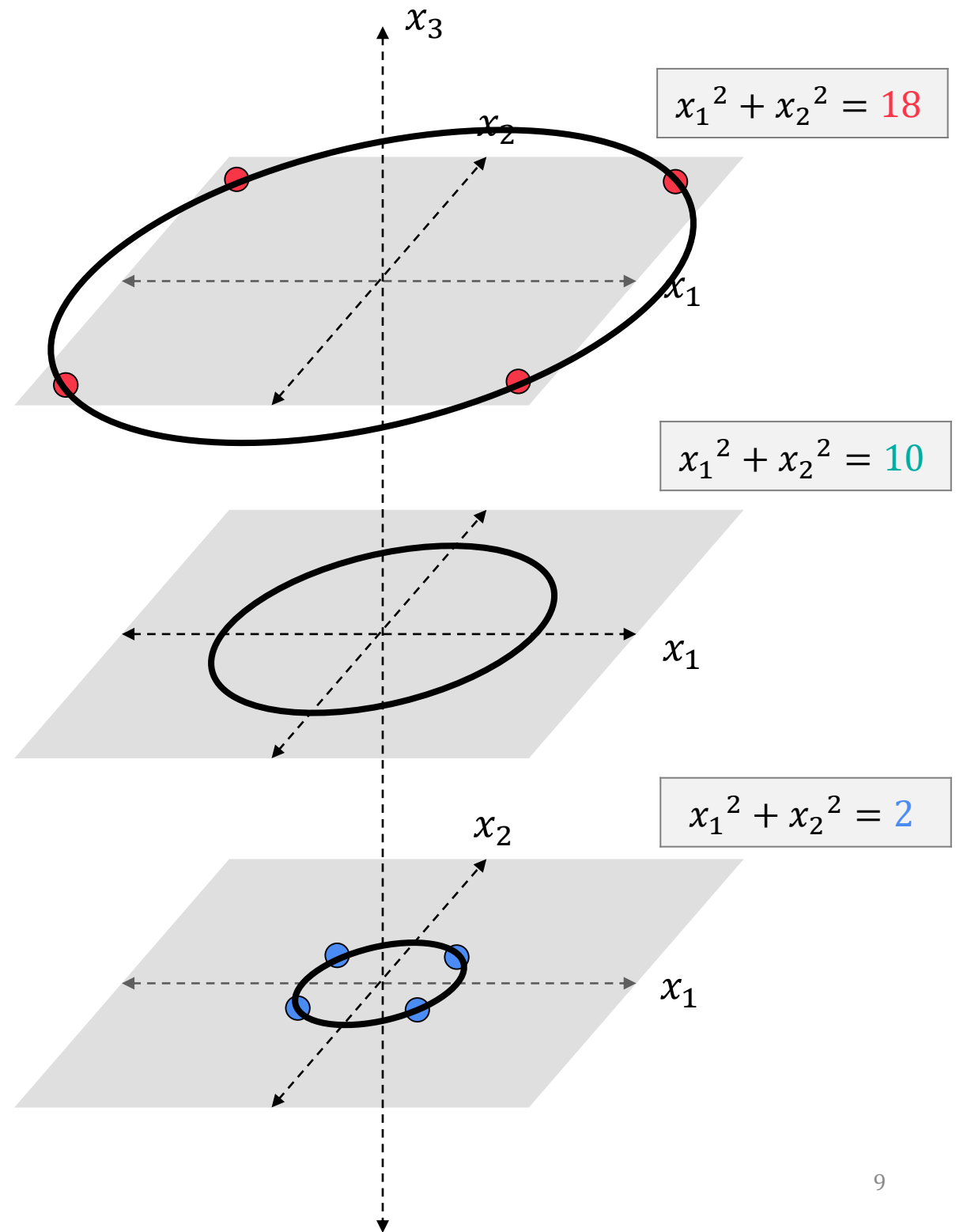
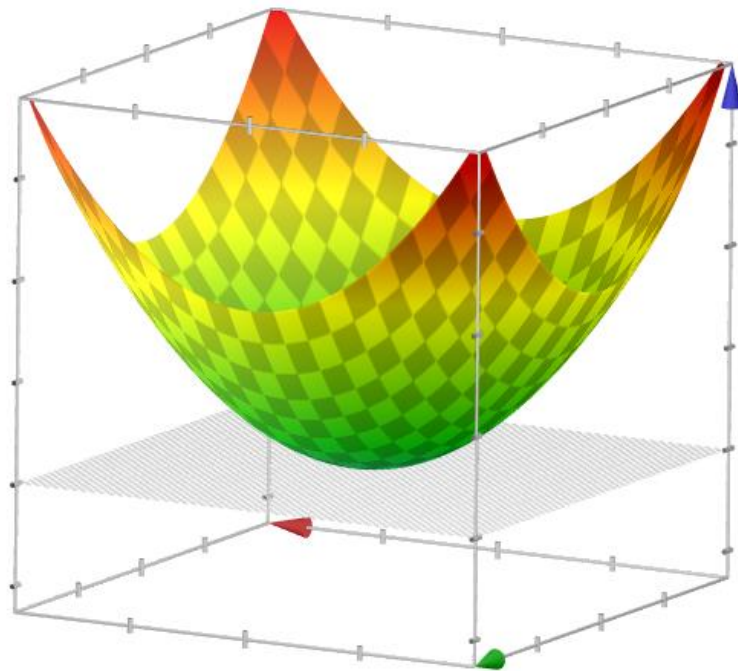
2 weeks ago,



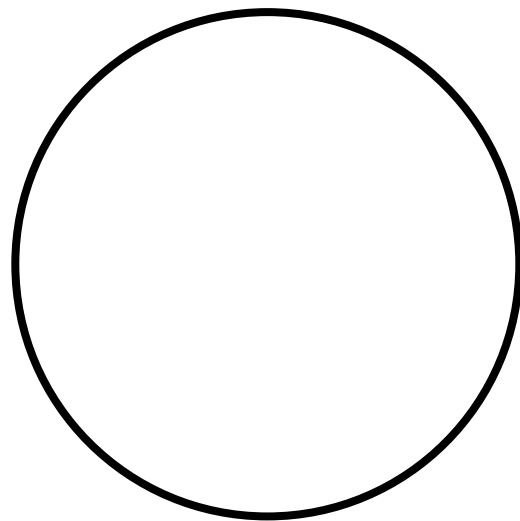


2 weeks ago,

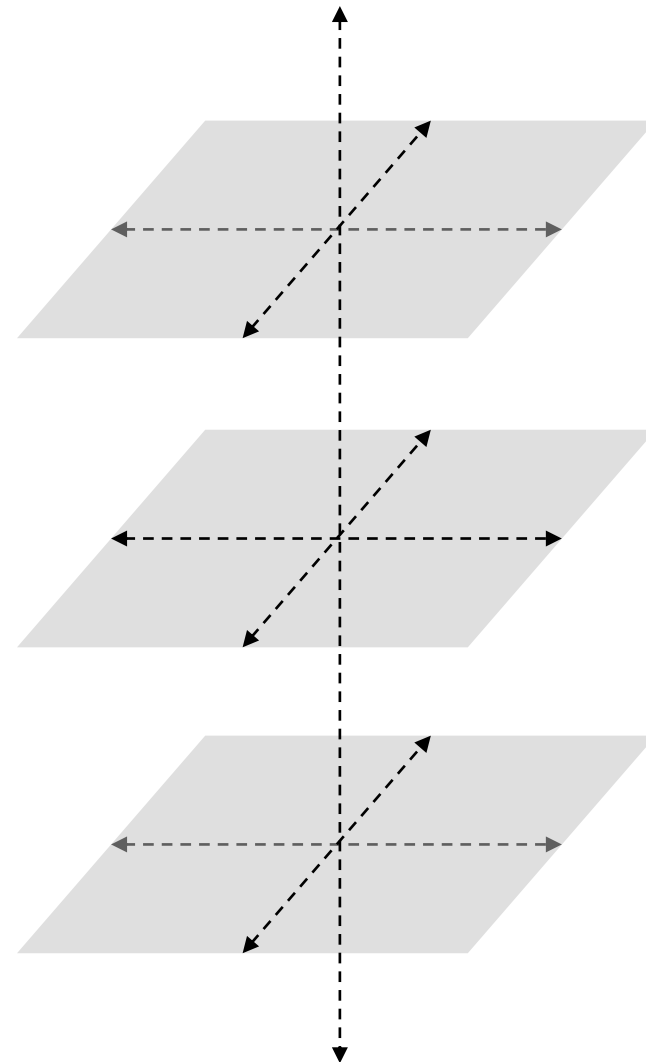
$$x_3 = x_1^2 + x_2^2$$



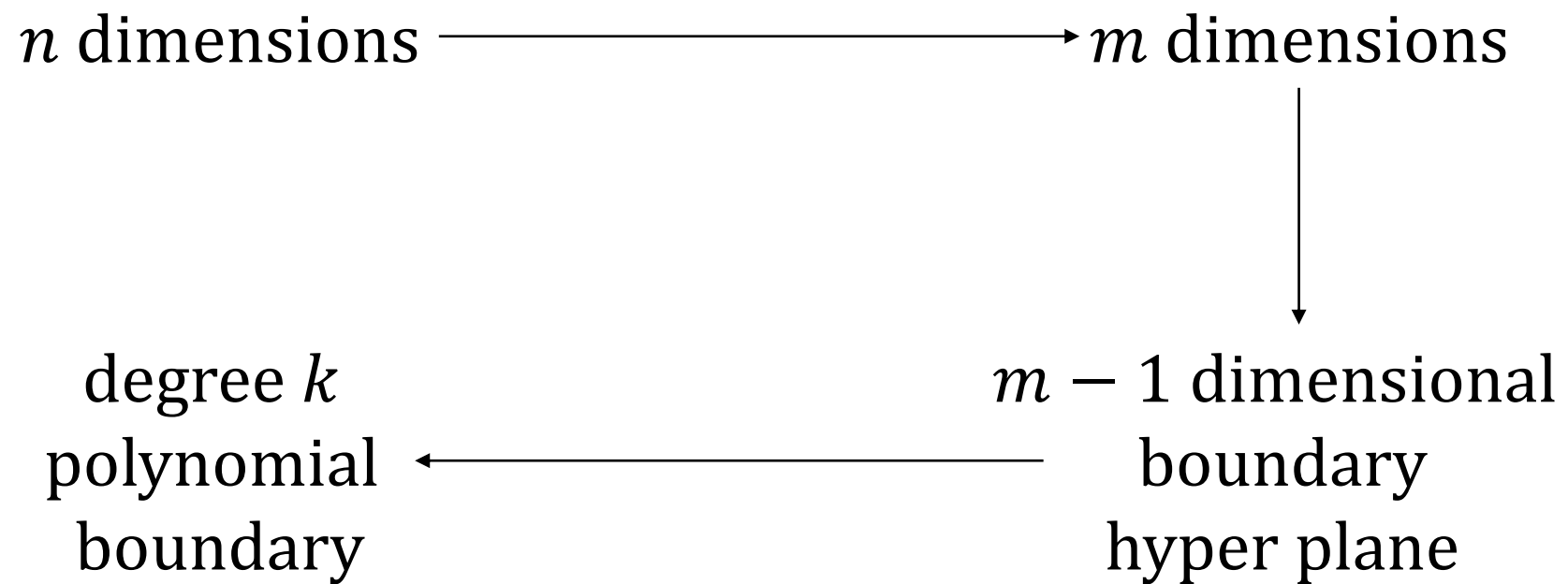
2 weeks ago,



=



2 weeks ago,



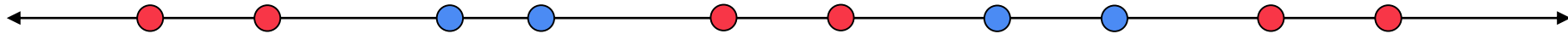
2 weeks ago,

Original dimension:  $x_1, x_2$

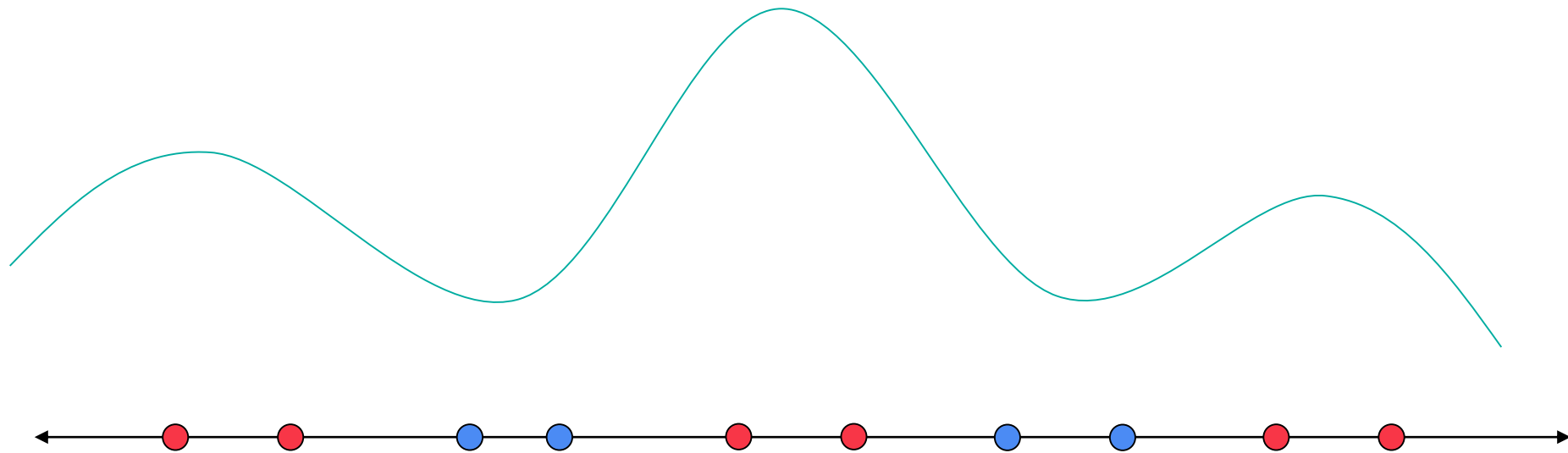
Degree	1	2	3	...
Tools	$x_1, x_2$	$x_1, x_2,$ $x_1^2, x_1x_2, x_2^2$	$x_1, x_2,$ $x_1^2, x_1x_2, x_2^2,$ $x_1^3, x_1^2x_2, x_1x_2^2, x_2^3$	...
E.g.	$2x_1 - x_2 = 1,$ $5x_1 + 4x_2 = 3$	$x_1x_2 = 1,$ $3x_1^2 - x_2^2 = 7$	$x_1^3 + 2x_1^2 - x_1 - x_2 = 2$	...

## **Concept of radial basis function (RBF)**

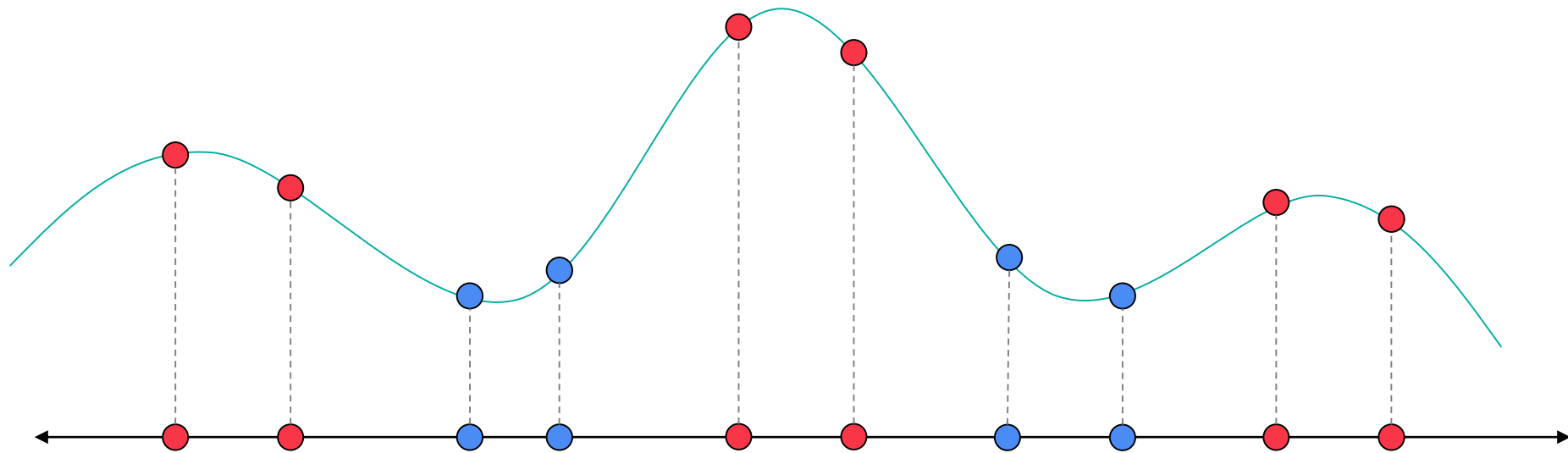
Again, nonlinearly separable data



Again, nonlinearly separable data

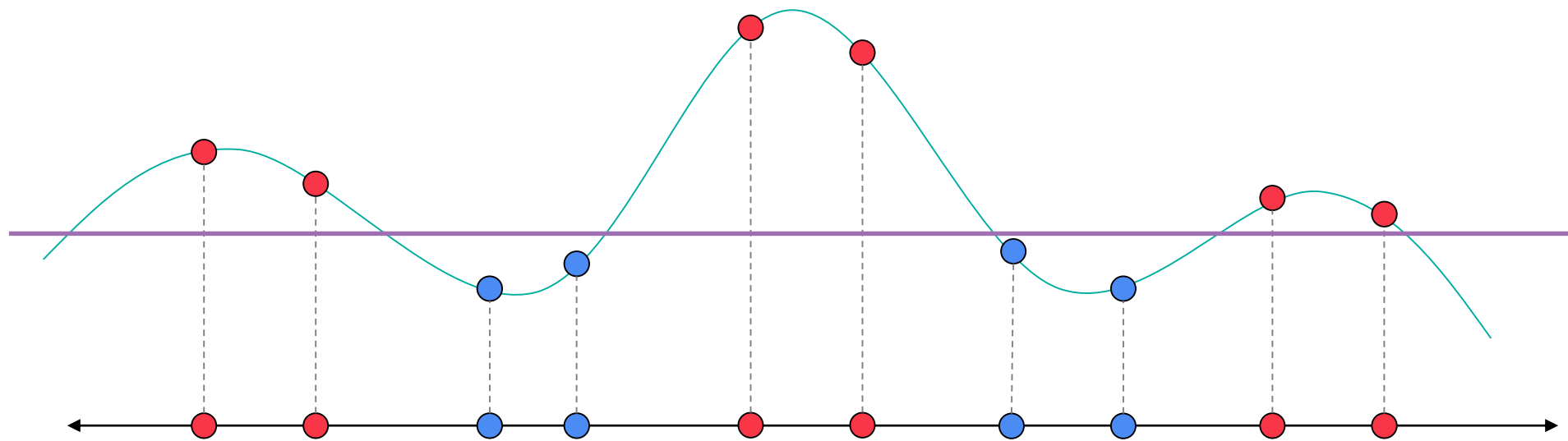


Again, nonlinearly separable data





Again, nonlinearly separable data

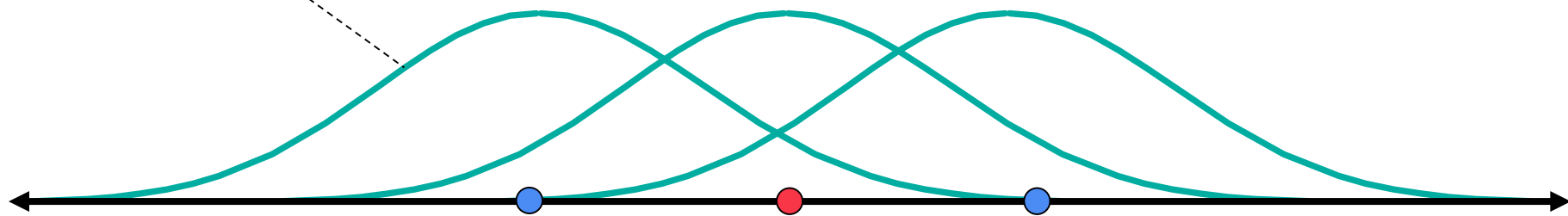


# RBF kernel

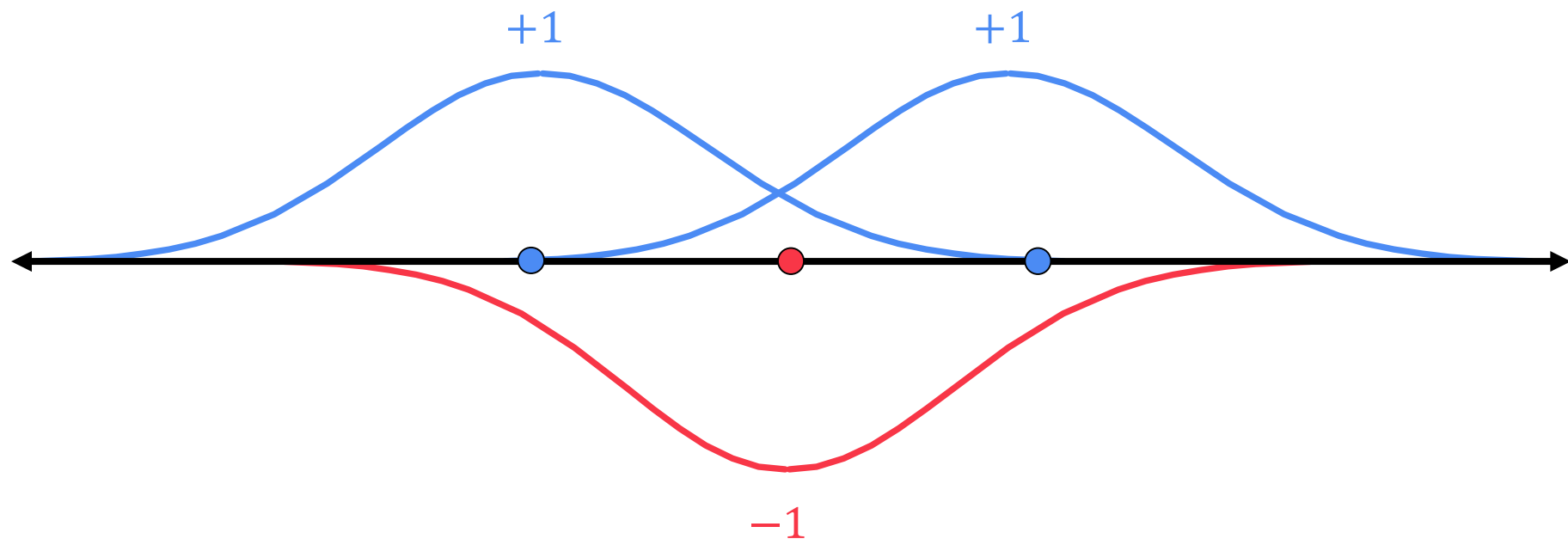


# RBF kernel

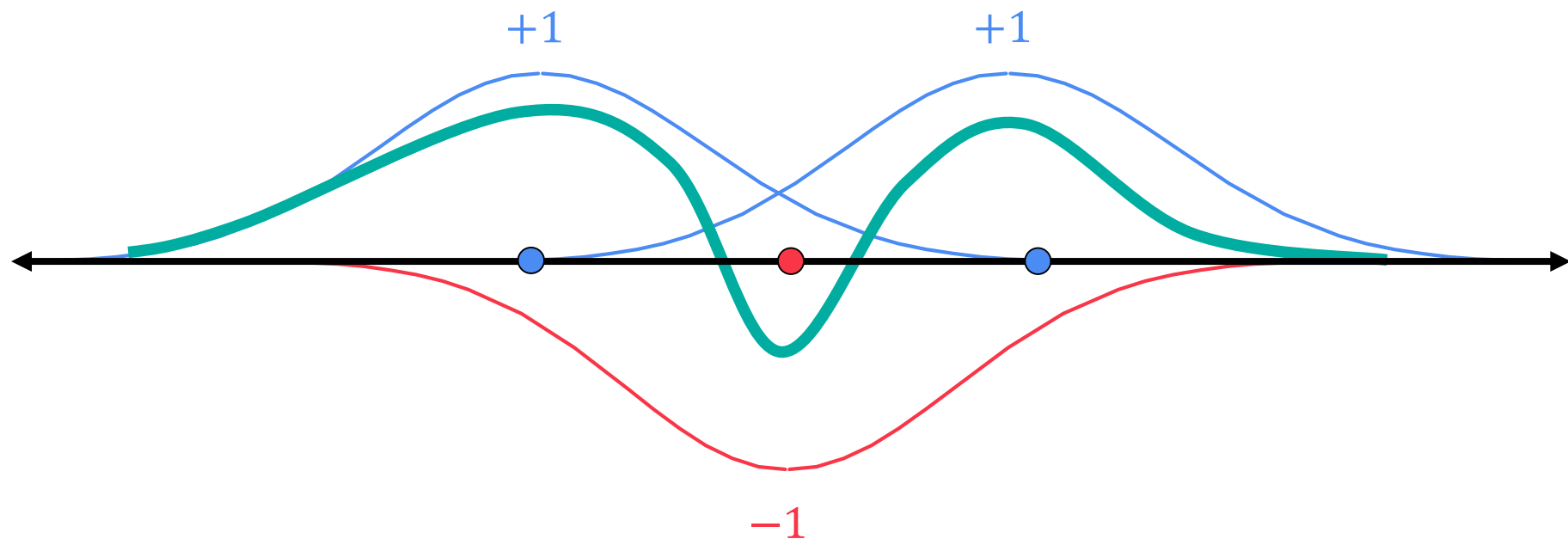
Radial basis function



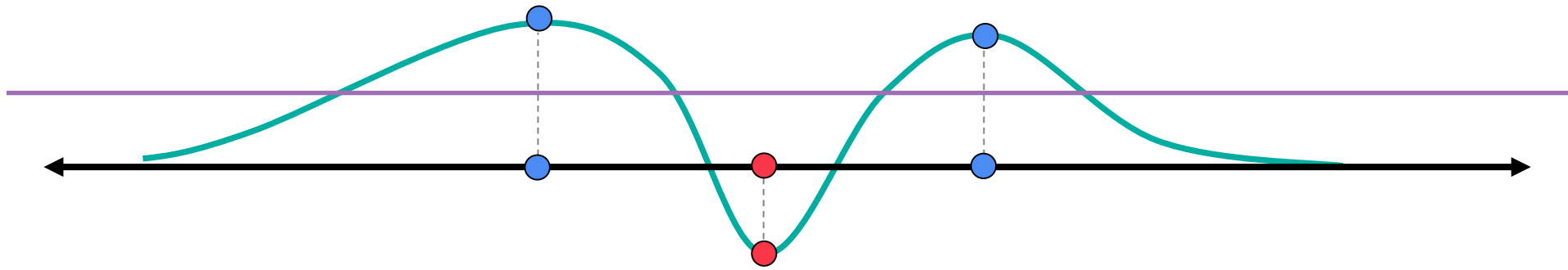
# RBF kernel



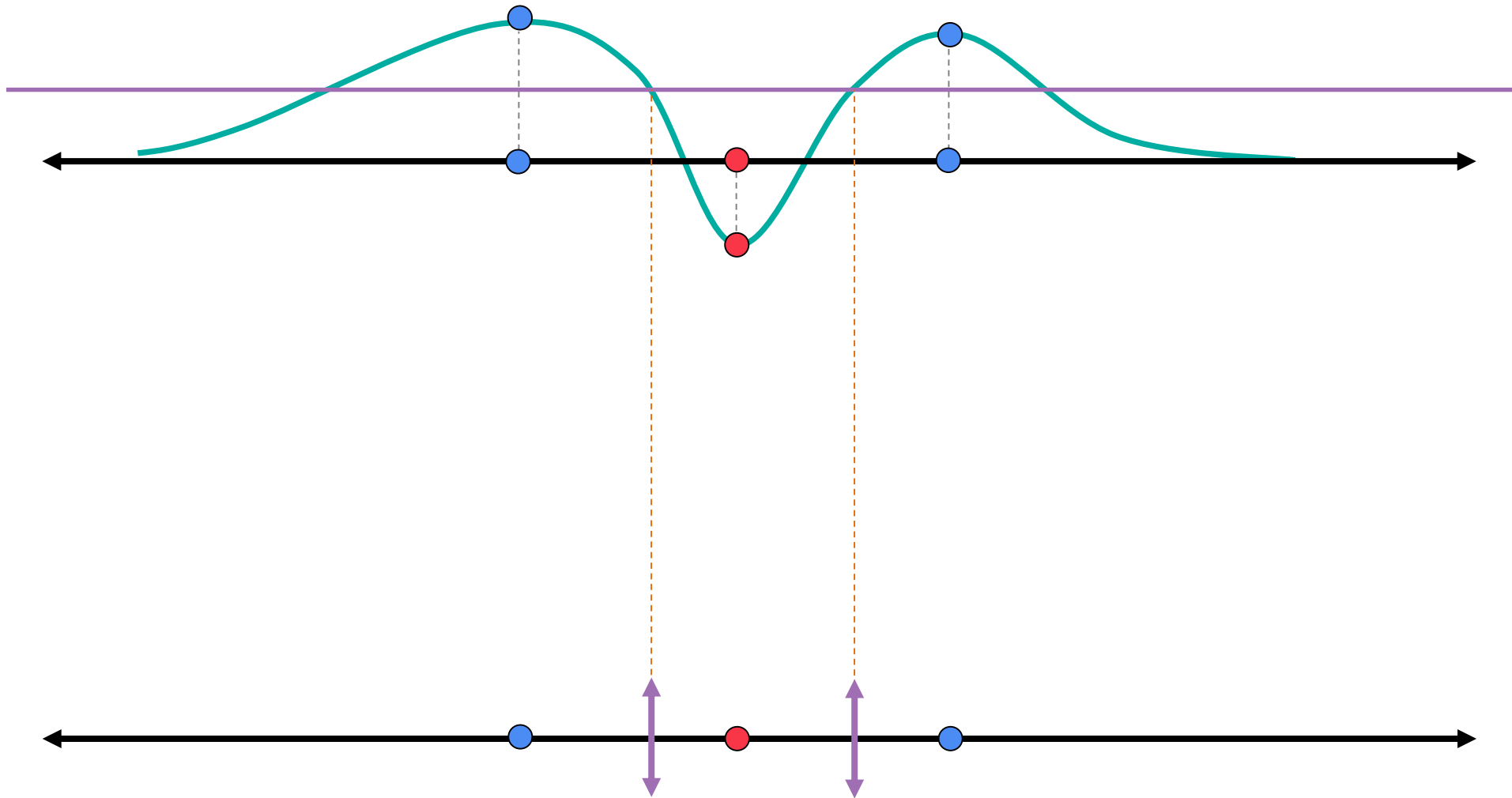
# RBF kernel



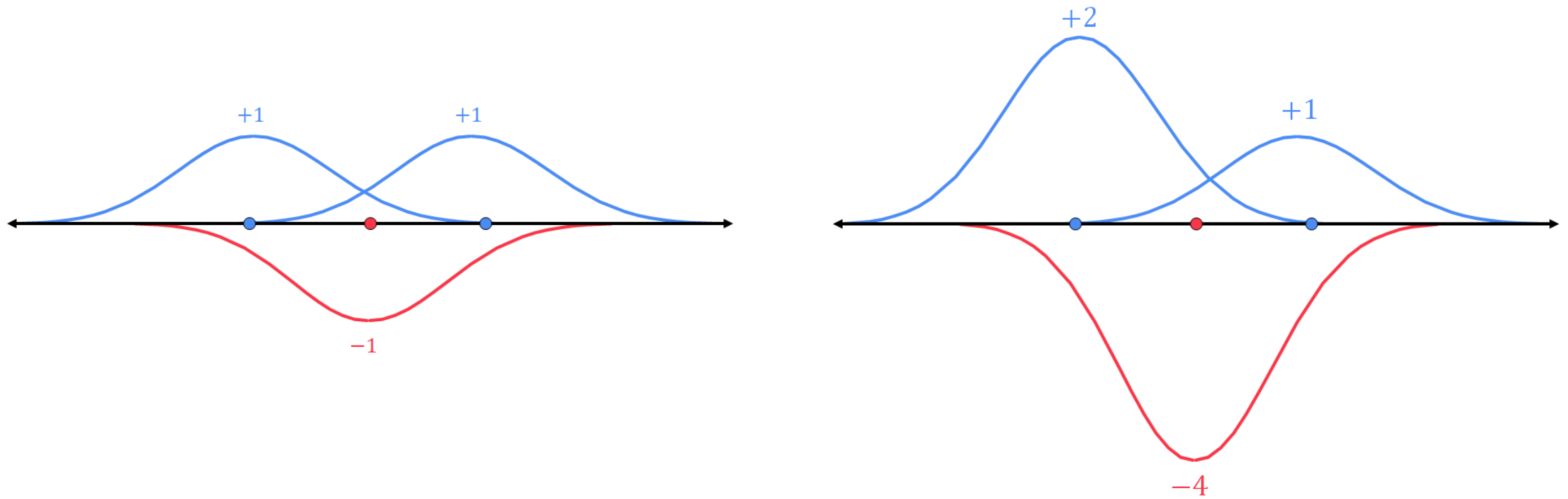
# RBF kernel



# RBF kernel

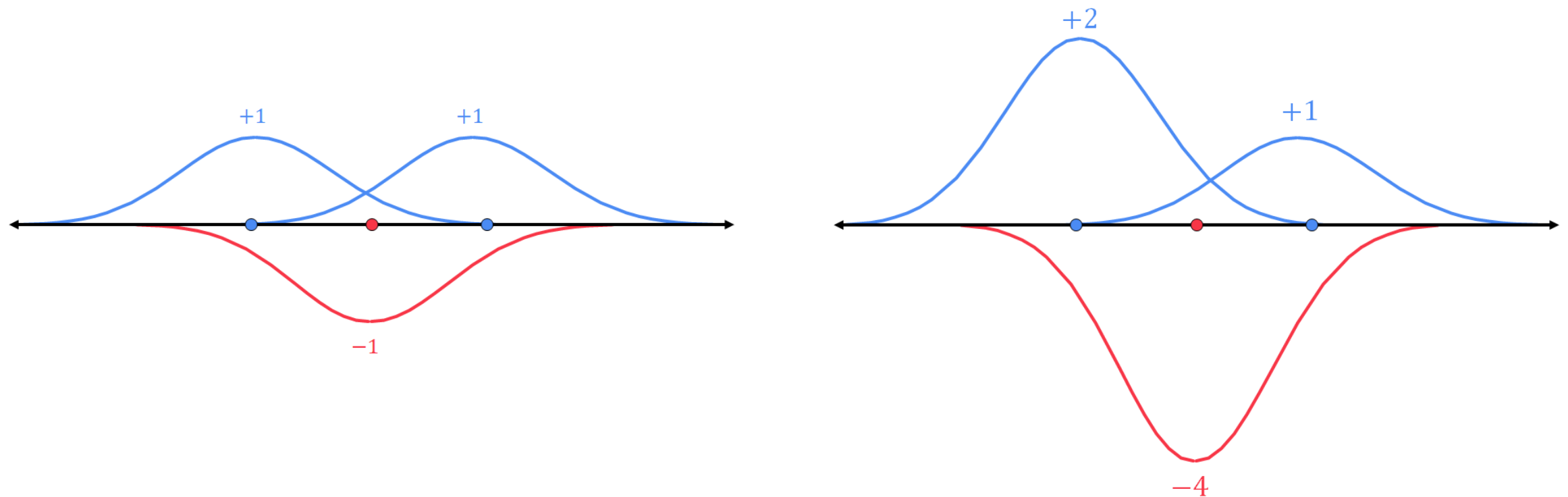


# RBF kernel



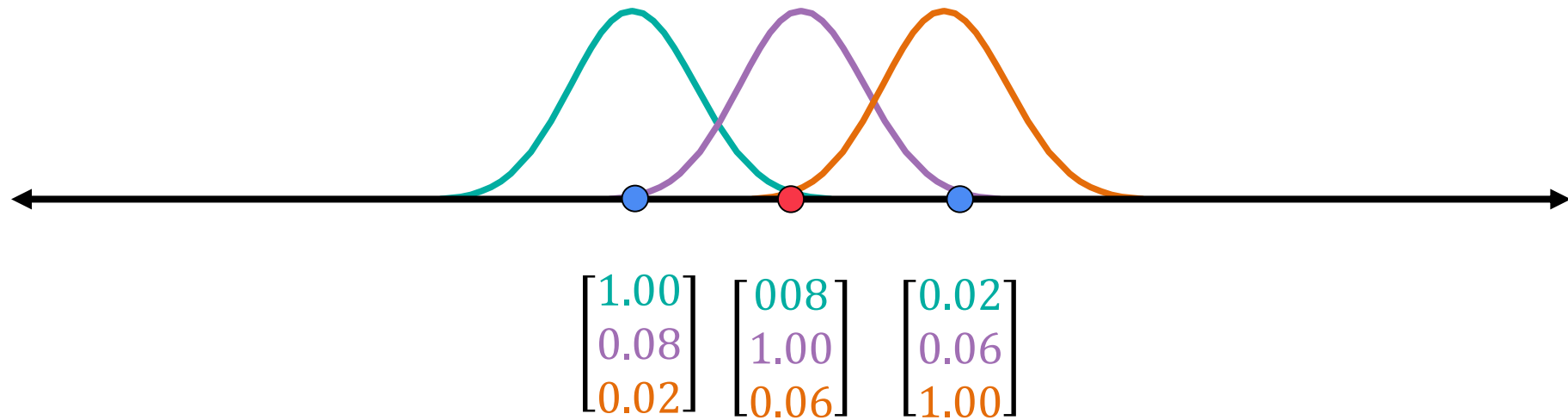


## RBF kernel

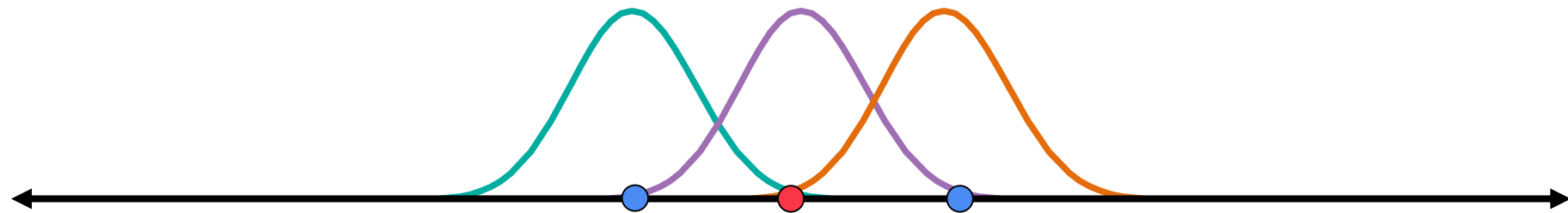


Issue: how can we find coefficient of radial basis ?

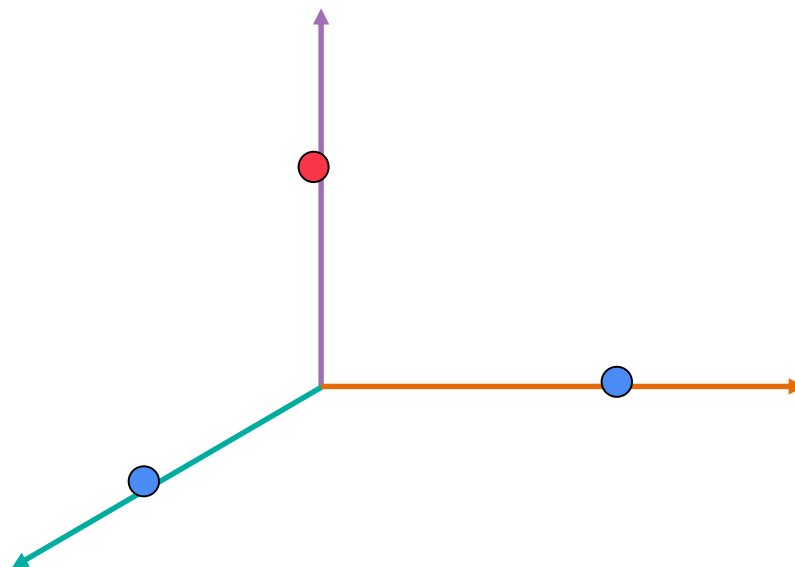
## How to find the coefficient of radial basis



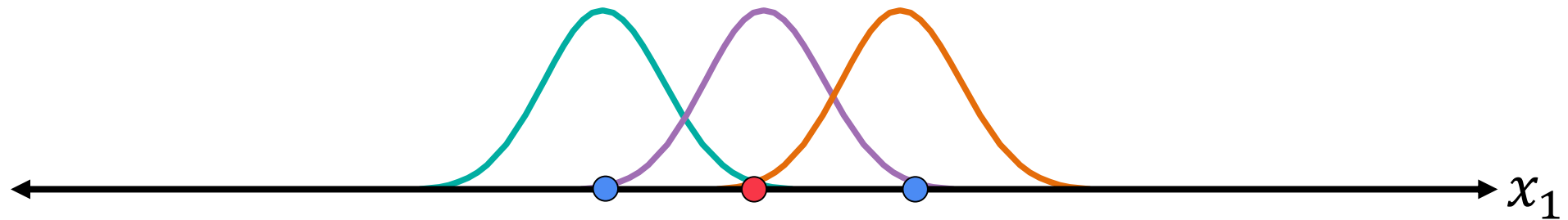
# How to find the coefficient of radial basis



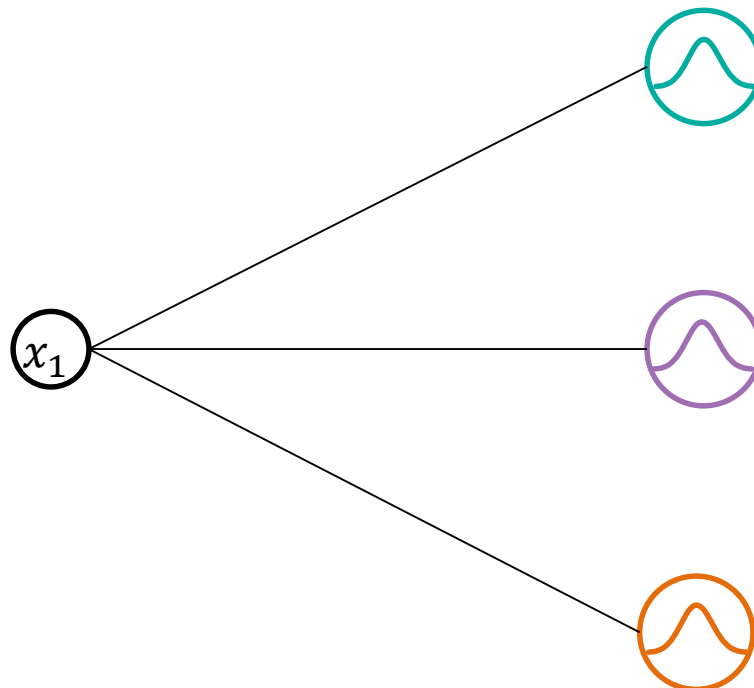
$$\begin{bmatrix} 1.00 \\ 0.08 \\ 0.02 \end{bmatrix} \quad \begin{bmatrix} 0.08 \\ 1.00 \\ 0.06 \end{bmatrix} \quad \begin{bmatrix} 0.02 \\ 0.06 \\ 1.00 \end{bmatrix}$$



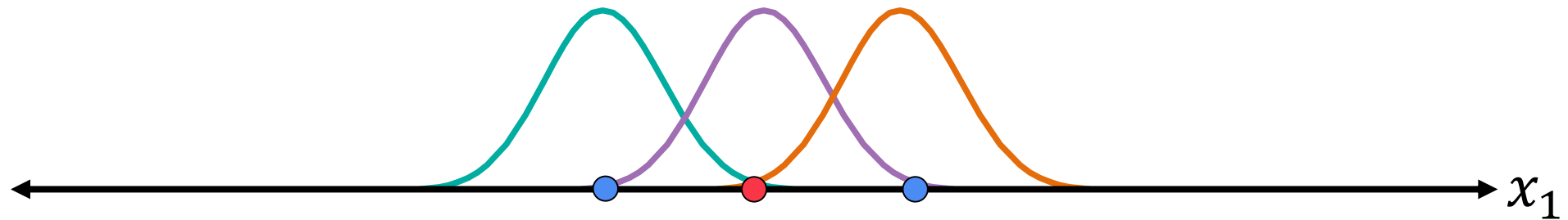
# How to find the coefficient of radial basis



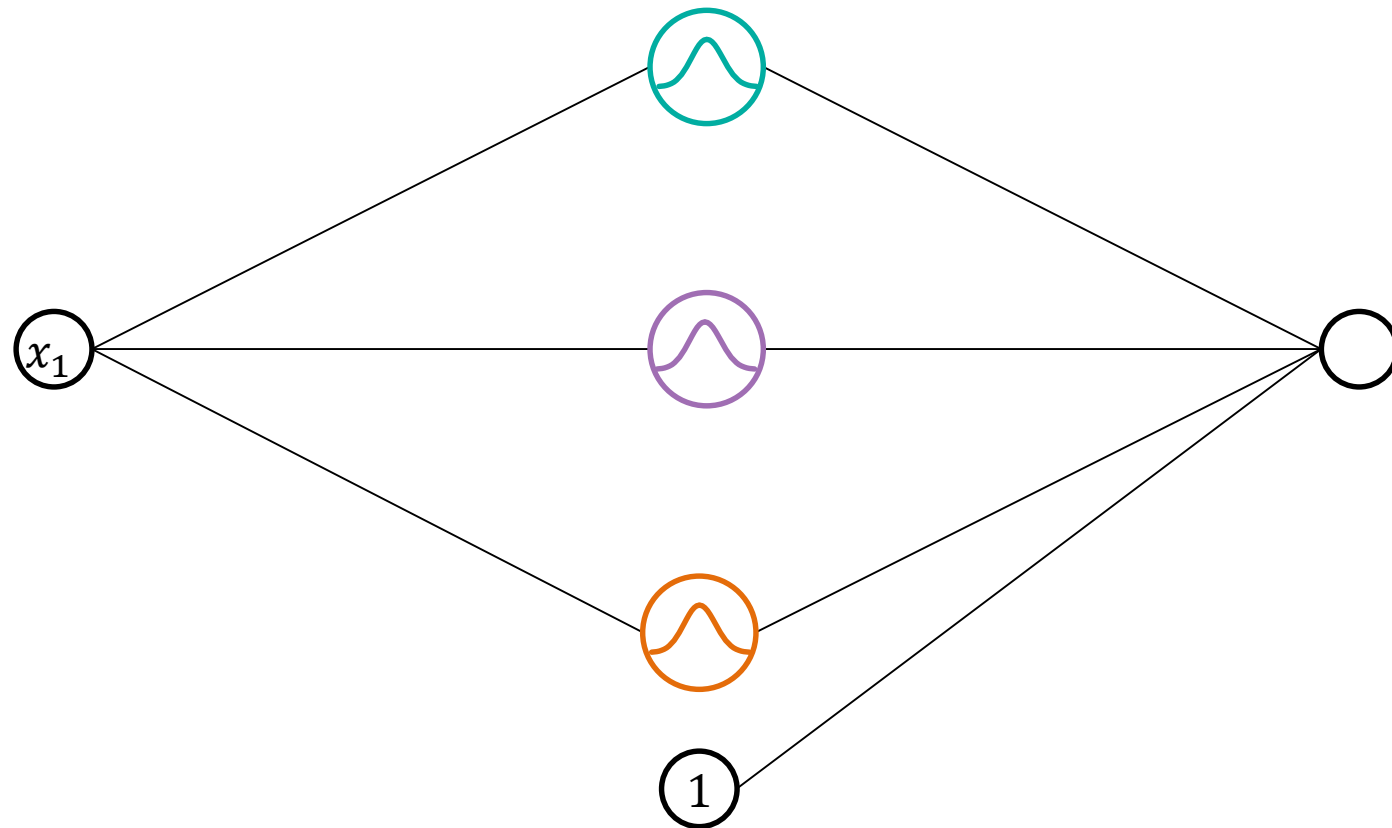
$$\begin{bmatrix} 1.00 \\ 0.08 \\ 0.02 \end{bmatrix} \quad \begin{bmatrix} 0.08 \\ 1.00 \\ 0.06 \end{bmatrix} \quad \begin{bmatrix} 0.02 \\ 0.06 \\ 1.00 \end{bmatrix}$$



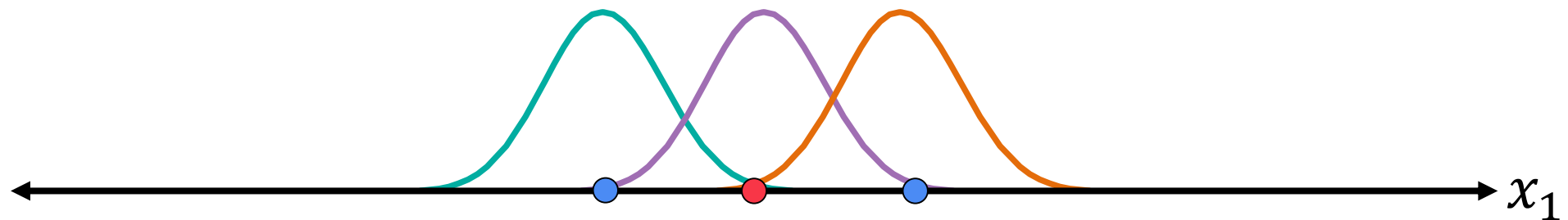
# How to find the coefficient of radial basis



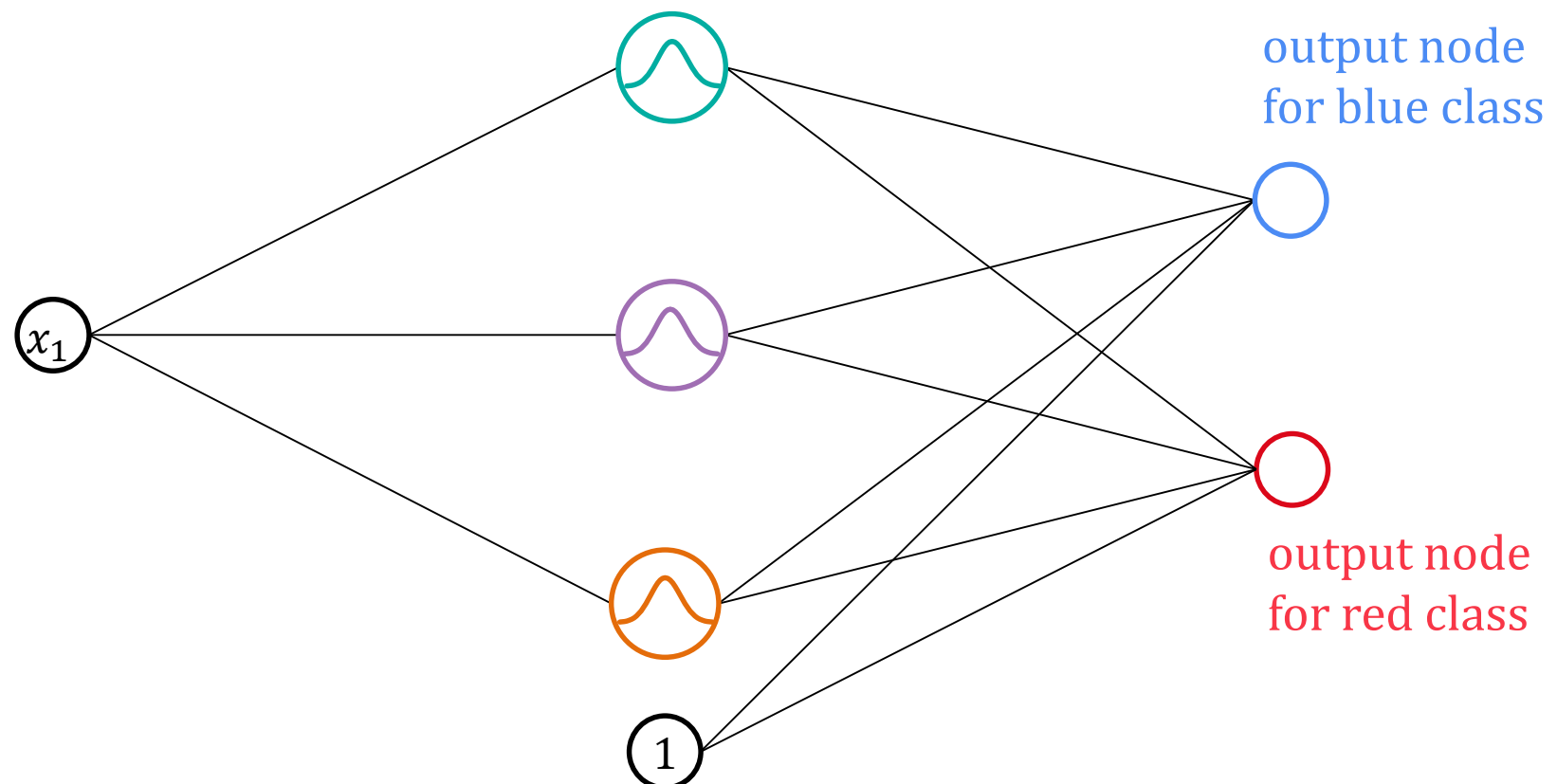
$$\begin{bmatrix} 1.00 \\ 0.08 \\ 0.02 \end{bmatrix} \quad \begin{bmatrix} 0.08 \\ 1.00 \\ 0.06 \end{bmatrix} \quad \begin{bmatrix} 0.02 \\ 0.06 \\ 1.00 \end{bmatrix}$$



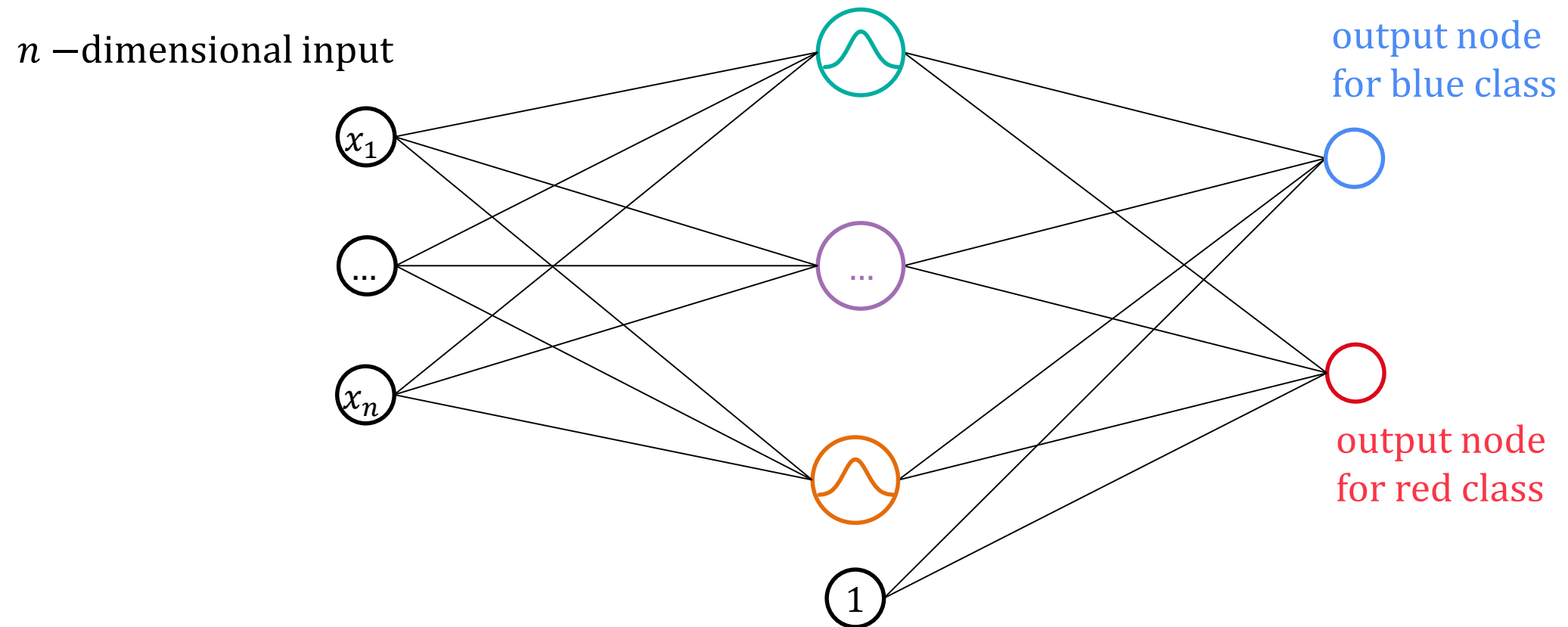
## How to find the coefficient of radial basis



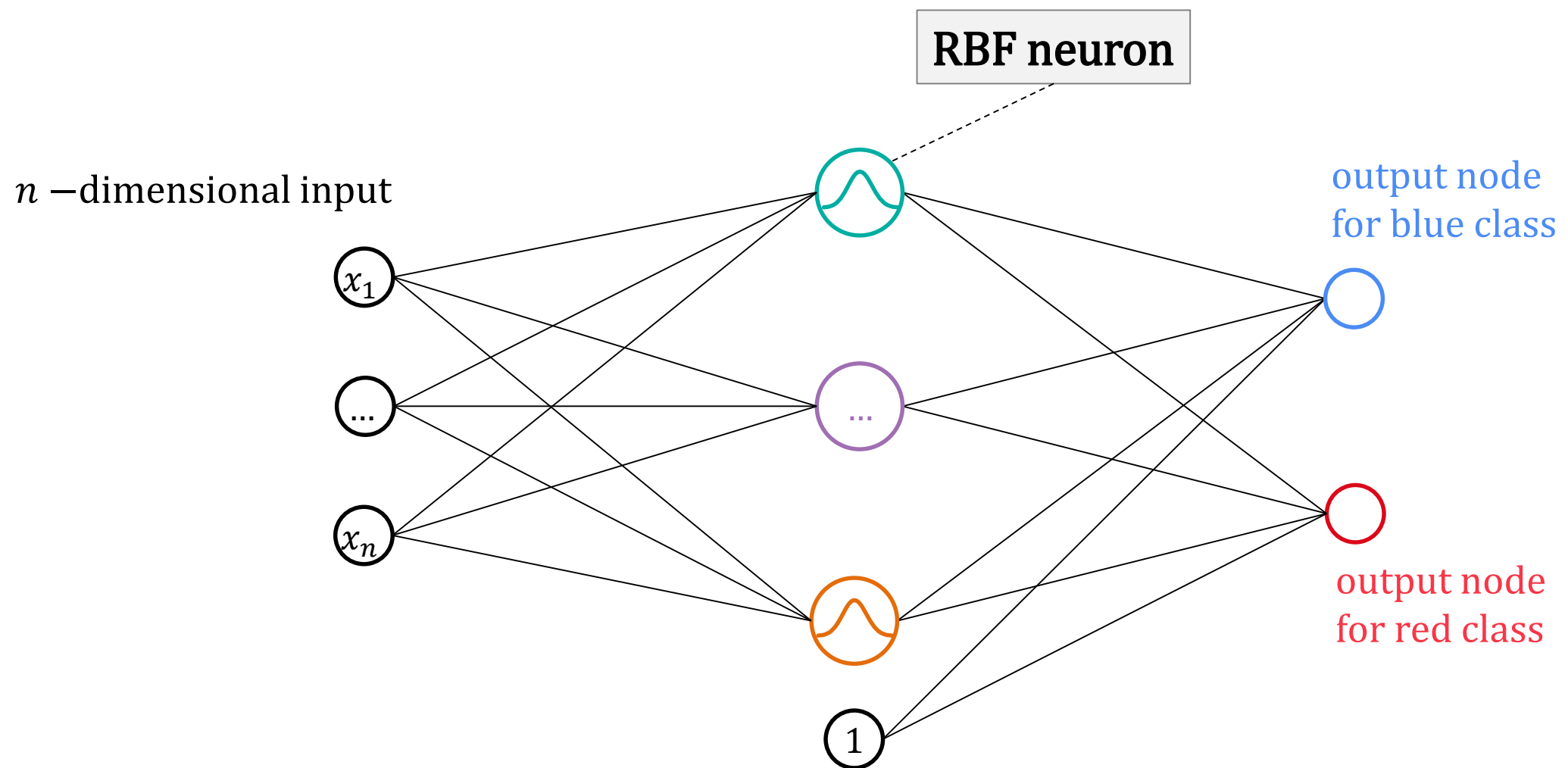
$$\begin{bmatrix} 1.00 \\ 0.08 \\ 0.02 \end{bmatrix} \quad \begin{bmatrix} 0.08 \\ 1.00 \\ 0.06 \end{bmatrix} \quad \begin{bmatrix} 0.02 \\ 0.06 \\ 1.00 \end{bmatrix}$$



## How to find the coefficient of radial basis



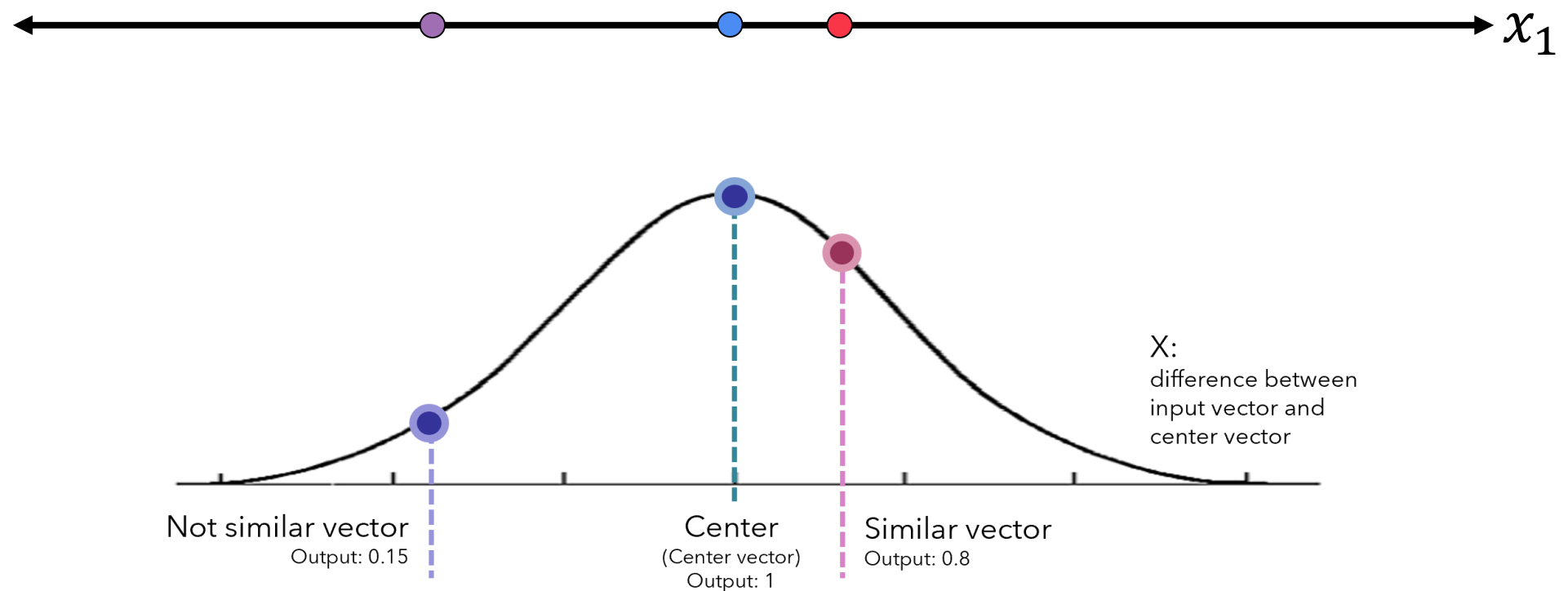
## How to find the coefficient of radial basis





# The RBF neuron

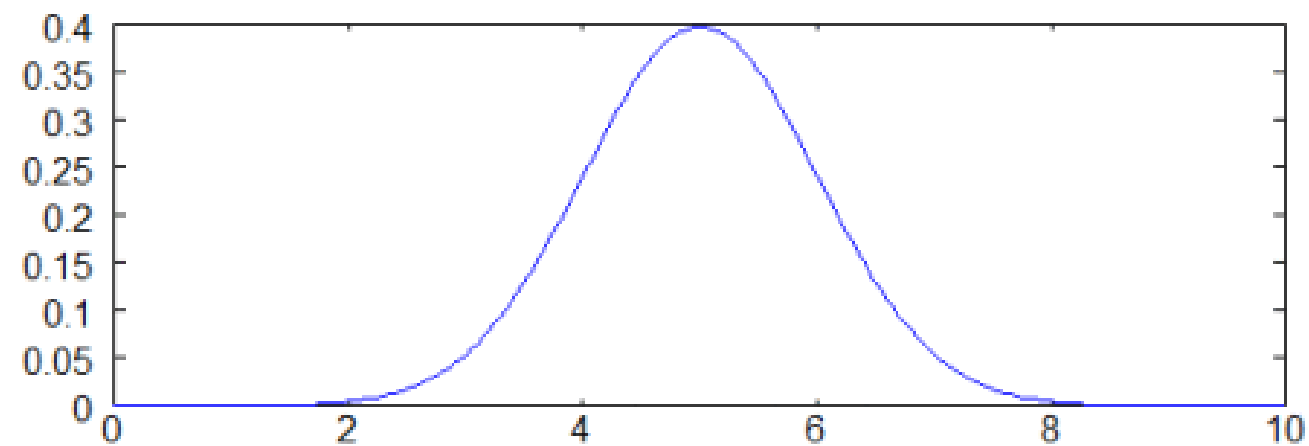
- RBF neuron stores a “**prototype vector (center)**”
- It measures of **similarity**
- It's response value (output) is also called its “**activation value**”



# RBF neural activation function

Gaussian

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



$$w/\mu = 5 \text{ \& } \sigma = 1$$

## RBF neural activation function

Gaussian

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

RBF neural activation function

$$e^{-\beta ||x-\mu||^2}$$

# RBF neural activation function

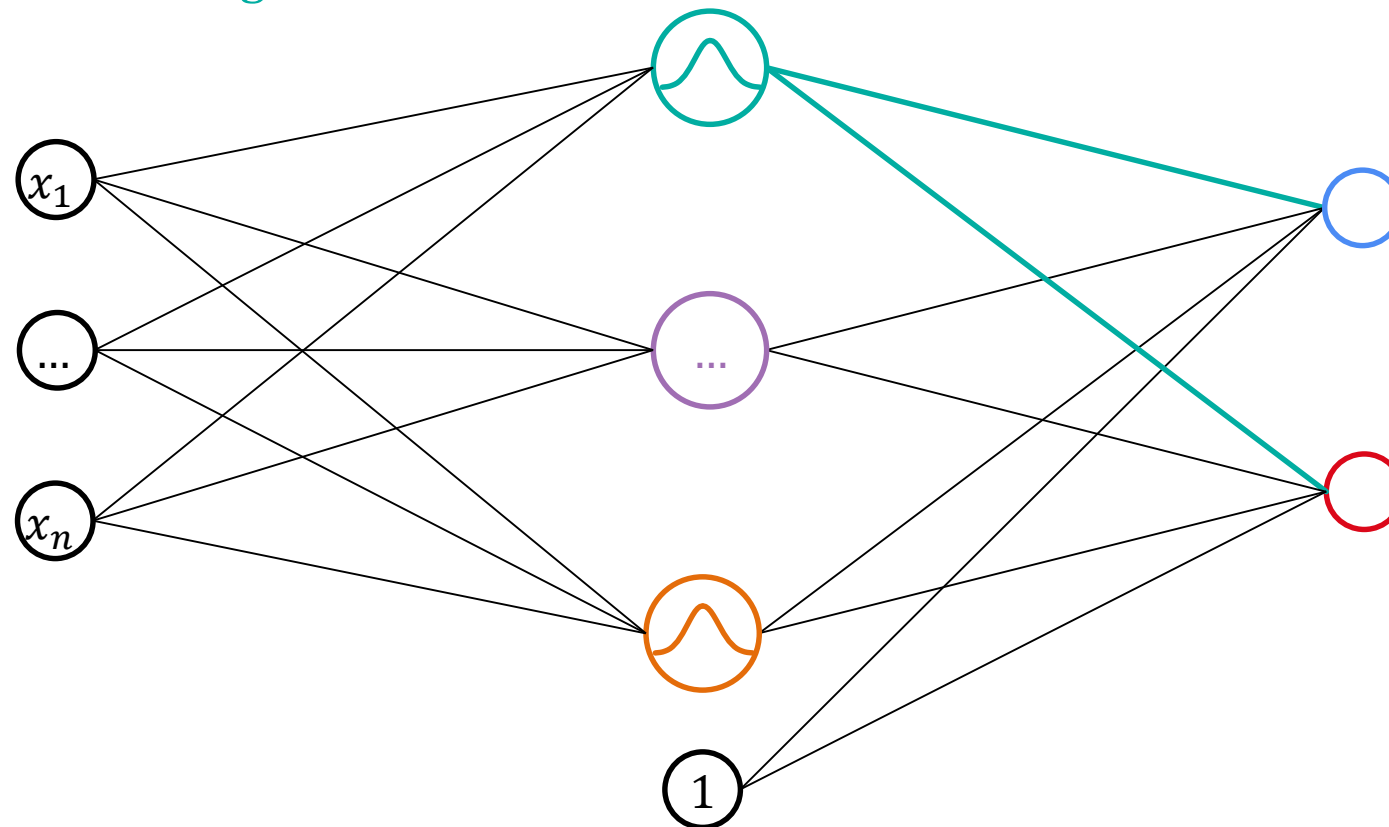
Gaussian

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

It controls the **height** of the Gaussian  
It is **redundant** with the **weights**

BRF neural activation function

$$e^{-\beta ||x-\mu||^2}$$



# RBF neural activation function

Gaussian

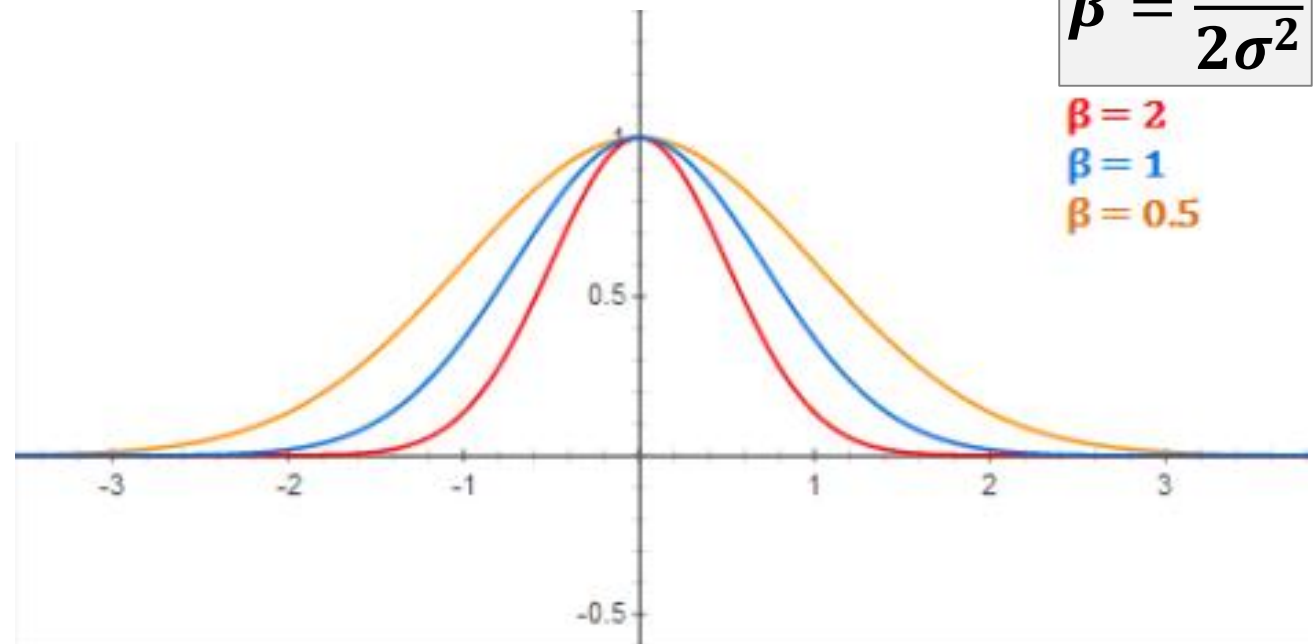
$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$
$$= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

It controls the **width** of the Gaussian

BRF neural activation function

$$e^{-\beta||x-\mu||^2}$$

$$\beta = \frac{1}{2\sigma^2}$$



## RBF neural activation function

Gaussian

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

Difference between scalar  $x$  and  $\mu$

RBF neural activation function

$$e^{-\beta ||x-\mu||^2}$$

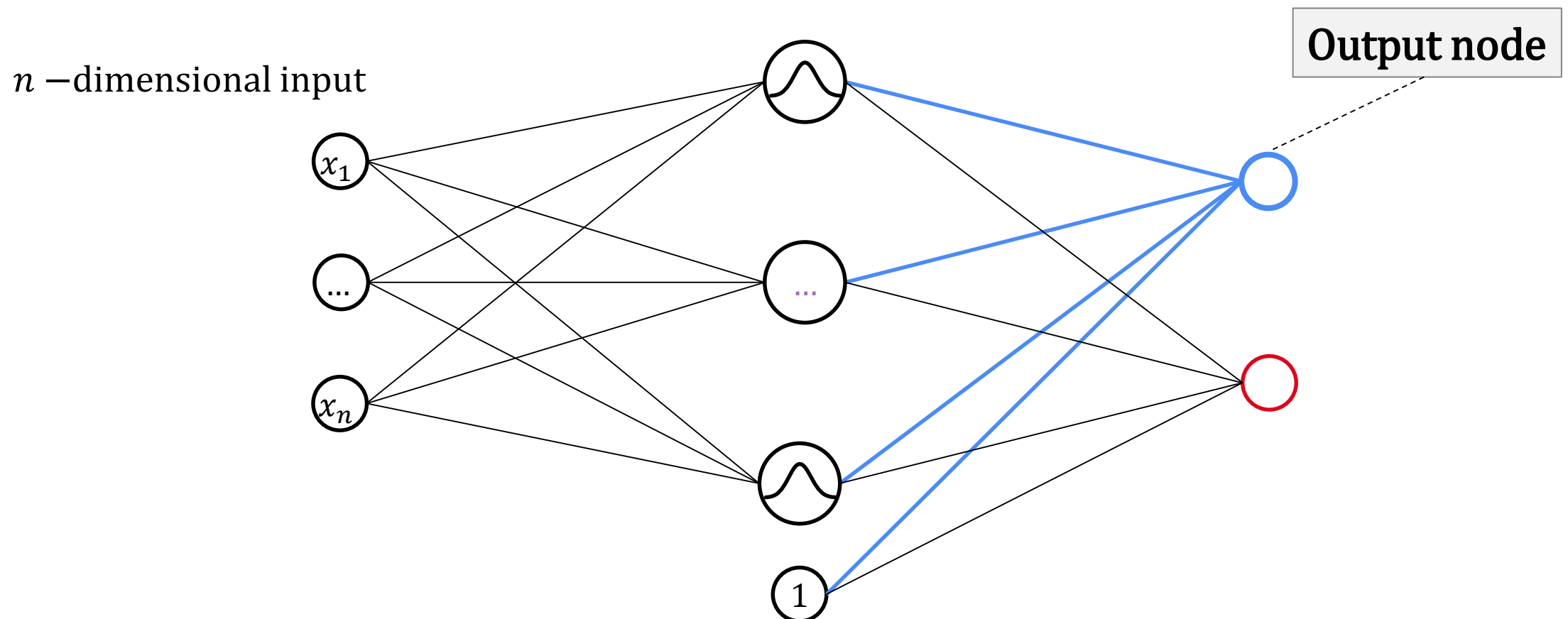
$x$ :  $n$  –dimensional vector

$\mu$ : prototype vector (center)

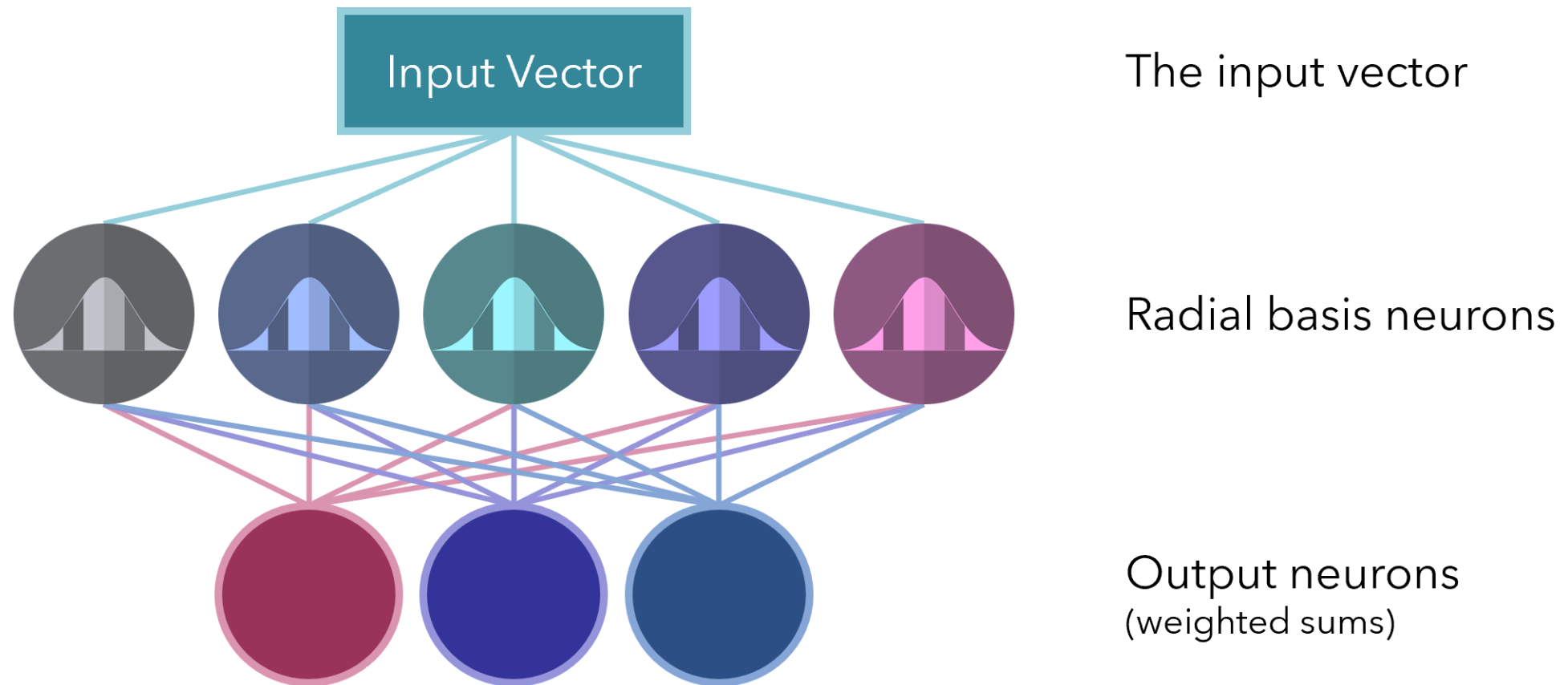
$||x - \mu||$ : Euclidean distance of  $x$  and  $\mu$

## The output node

- Generally, one per category
- By **weighted sum** we mean that an output node associates a weight value with each of the RBF neurons, and multiplies the **neuron's activation** by this **weight** before adding it to the total response.



# RBFN: Radial basis function network





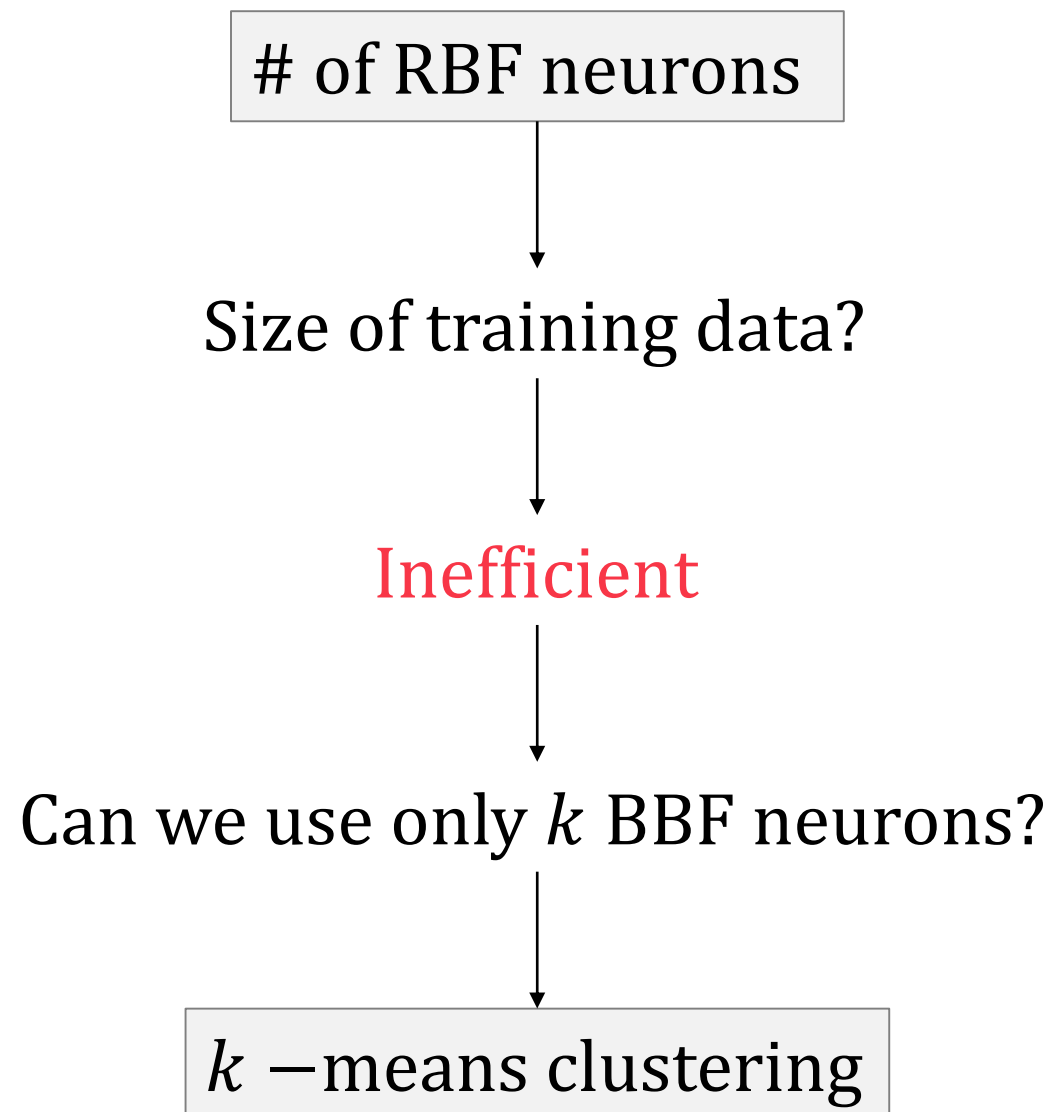
# Training the RBFN

Prototypes  
(center vectors)

Selecting  $\beta$  values

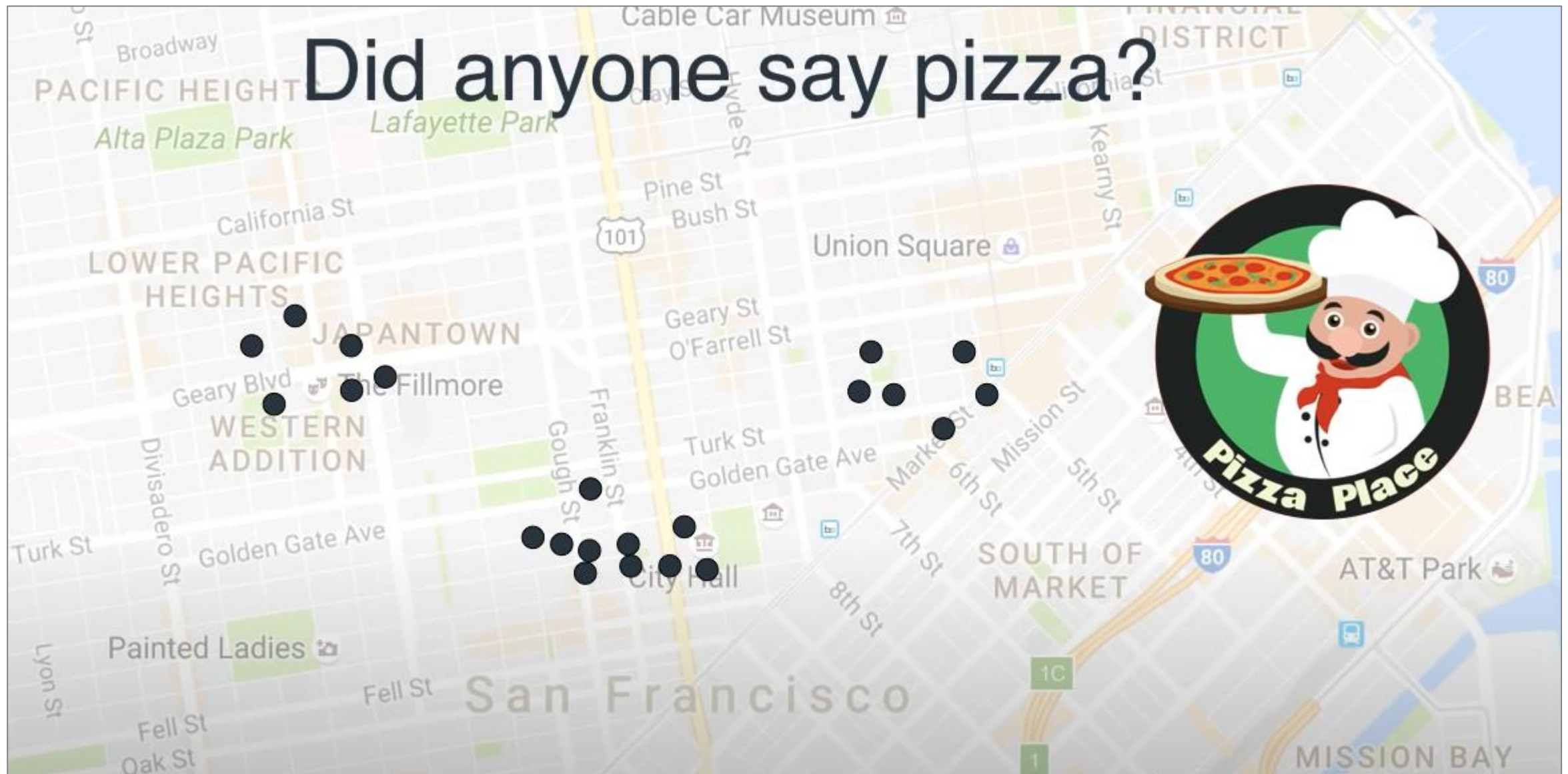
Output weights

## Prototypes: central vectors



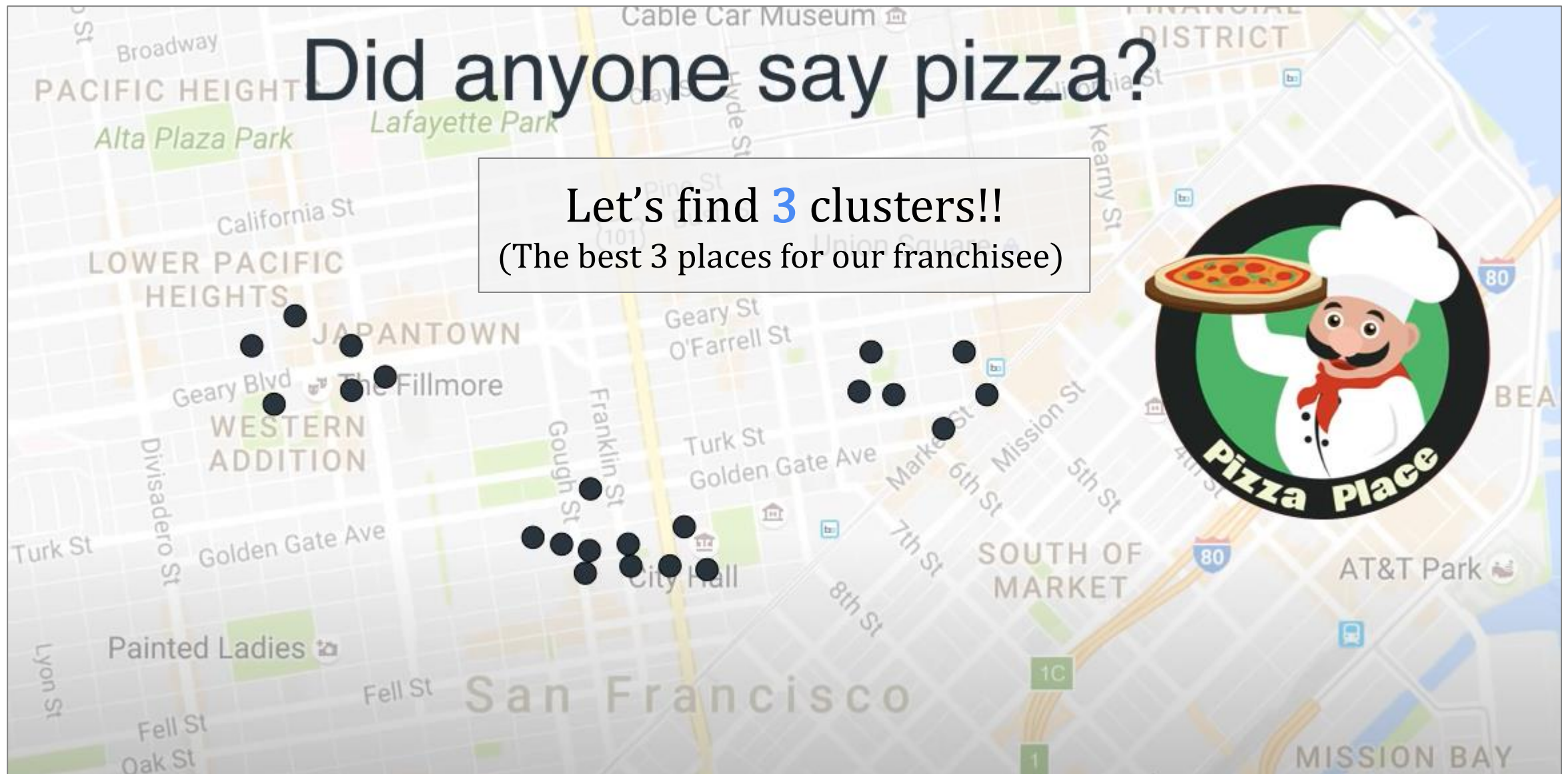
## $k$ – means clustering

<https://www.youtube.com/watch?v=IpGxLW0IZy4&t=1091s>



# $k$ – means clustering

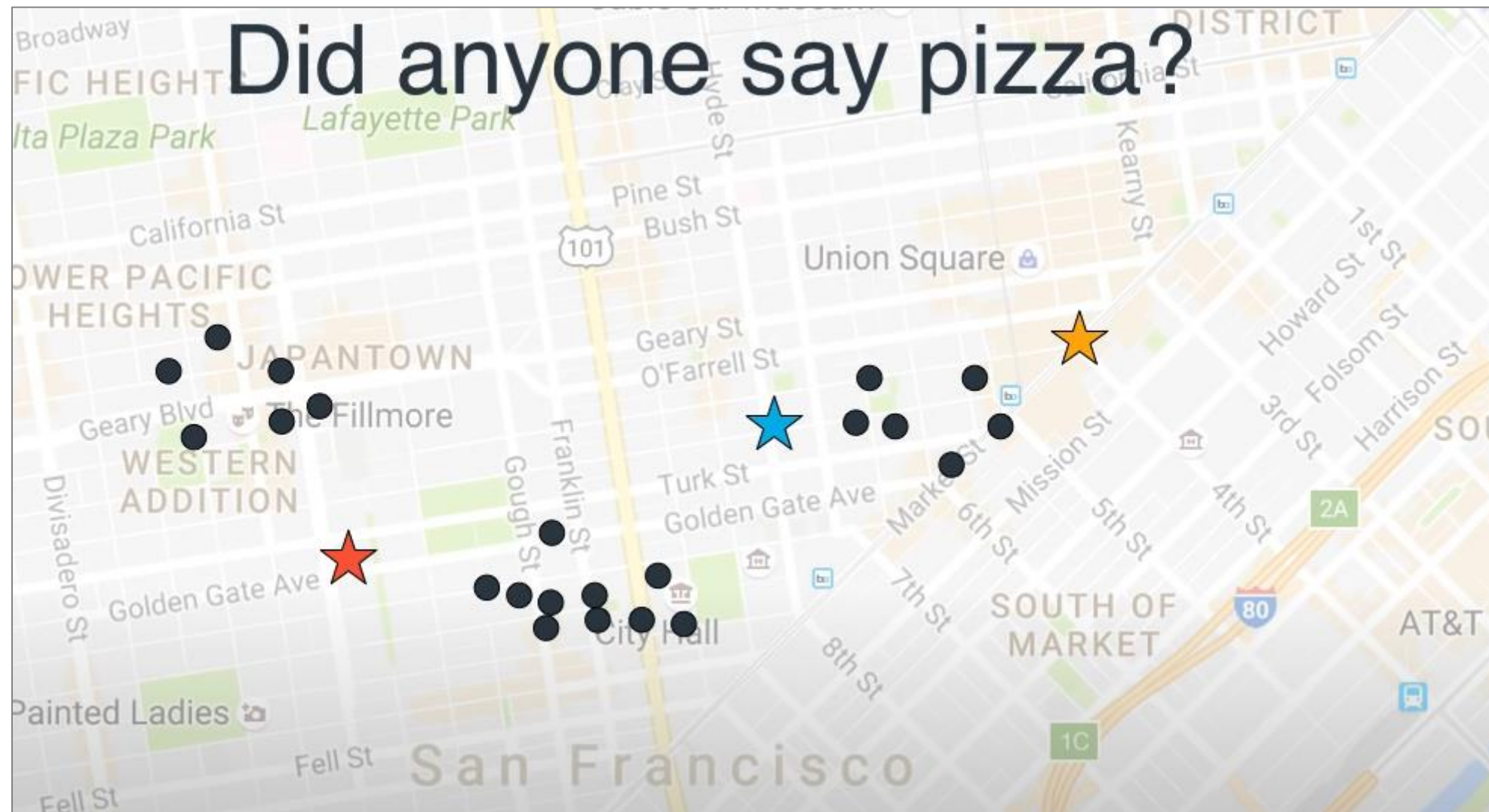
<https://www.youtube.com/watch?v=IpGxLWOIZy4&t=1091s>





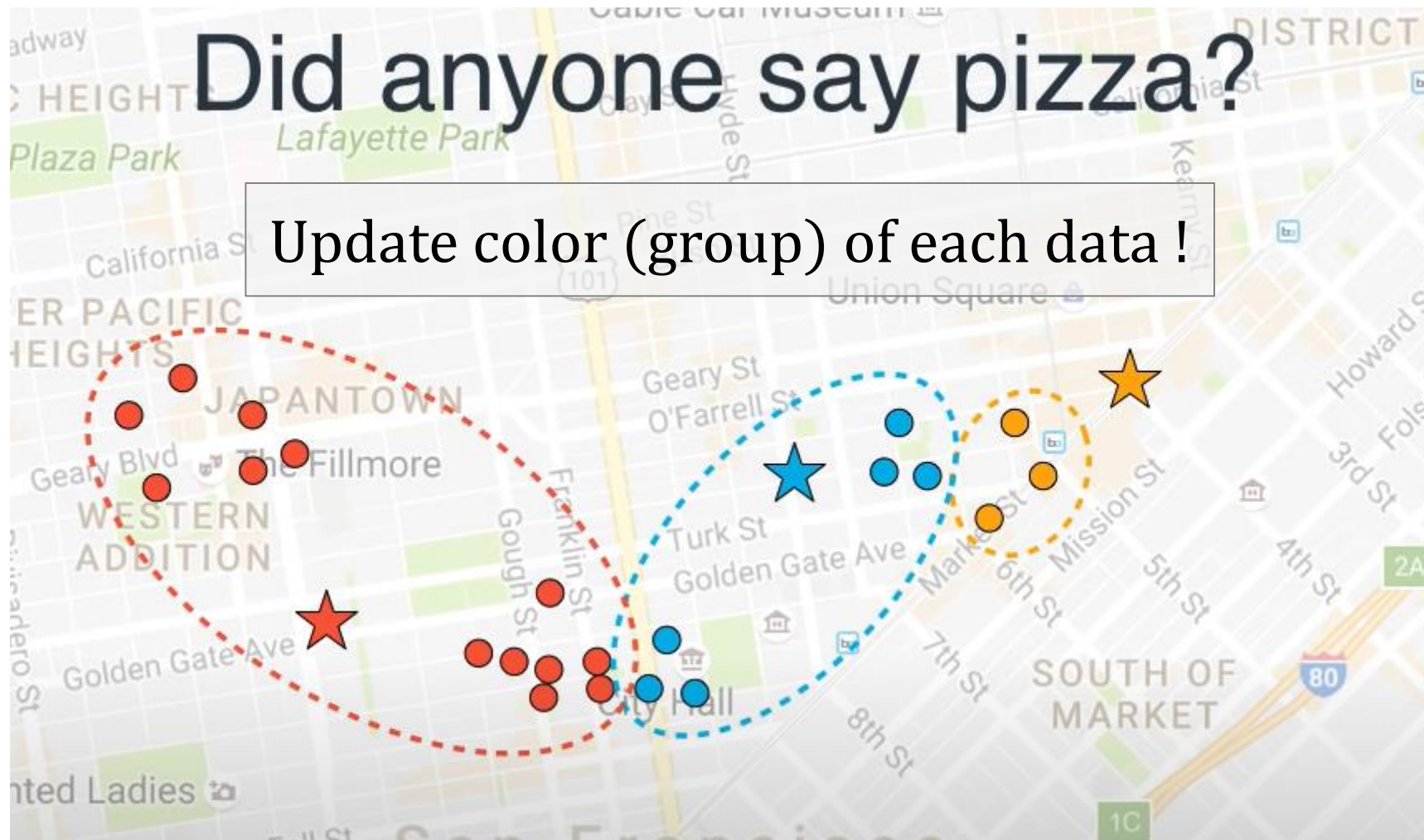
## $k$ – means clustering

<https://www.youtube.com/watch?v=IpGxLW0IZy4&t=1091s>



## $k$ –means clustering

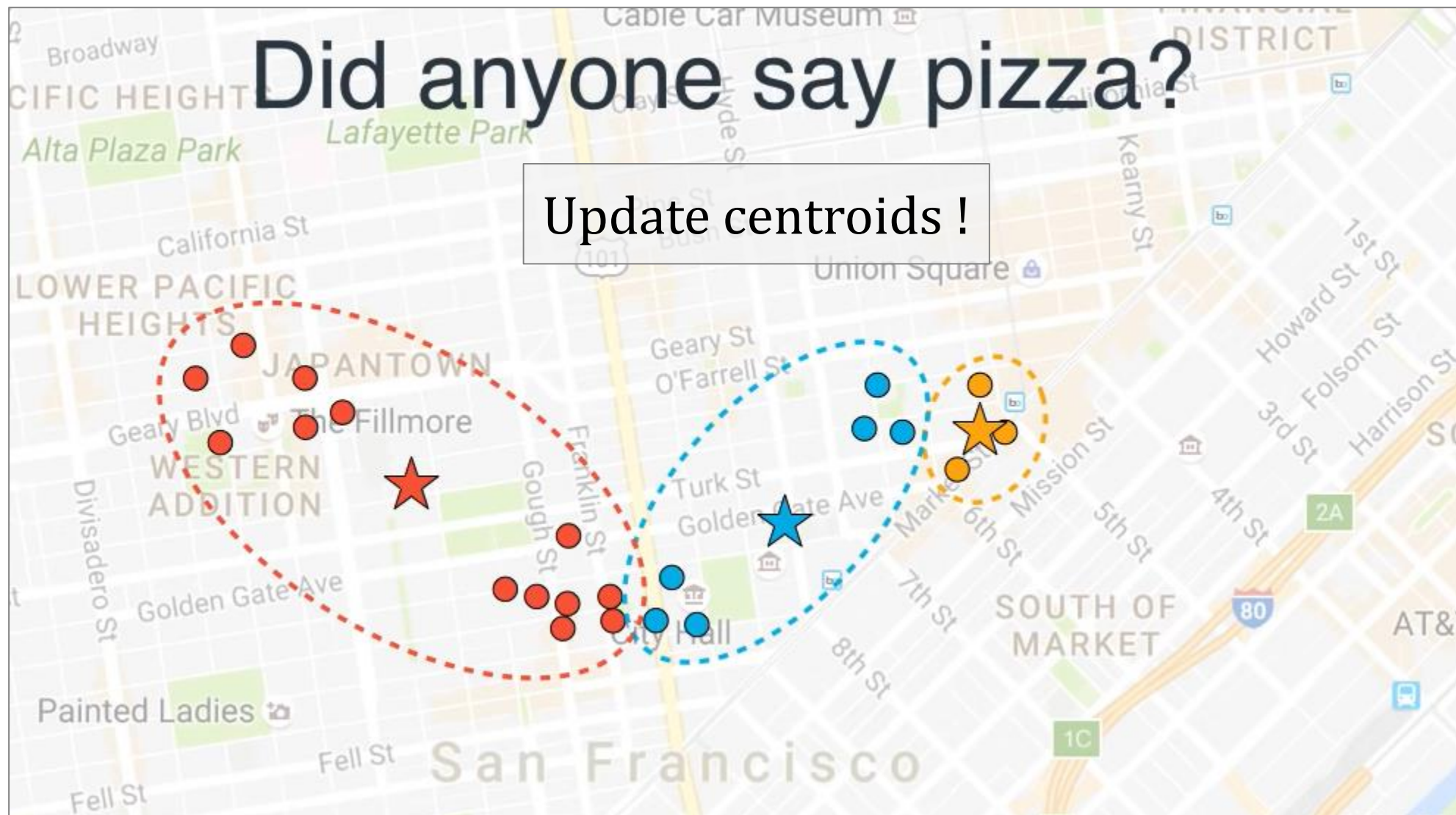
<https://www.youtube.com/watch?v=IpGxLWOIZy4&t=1091s>





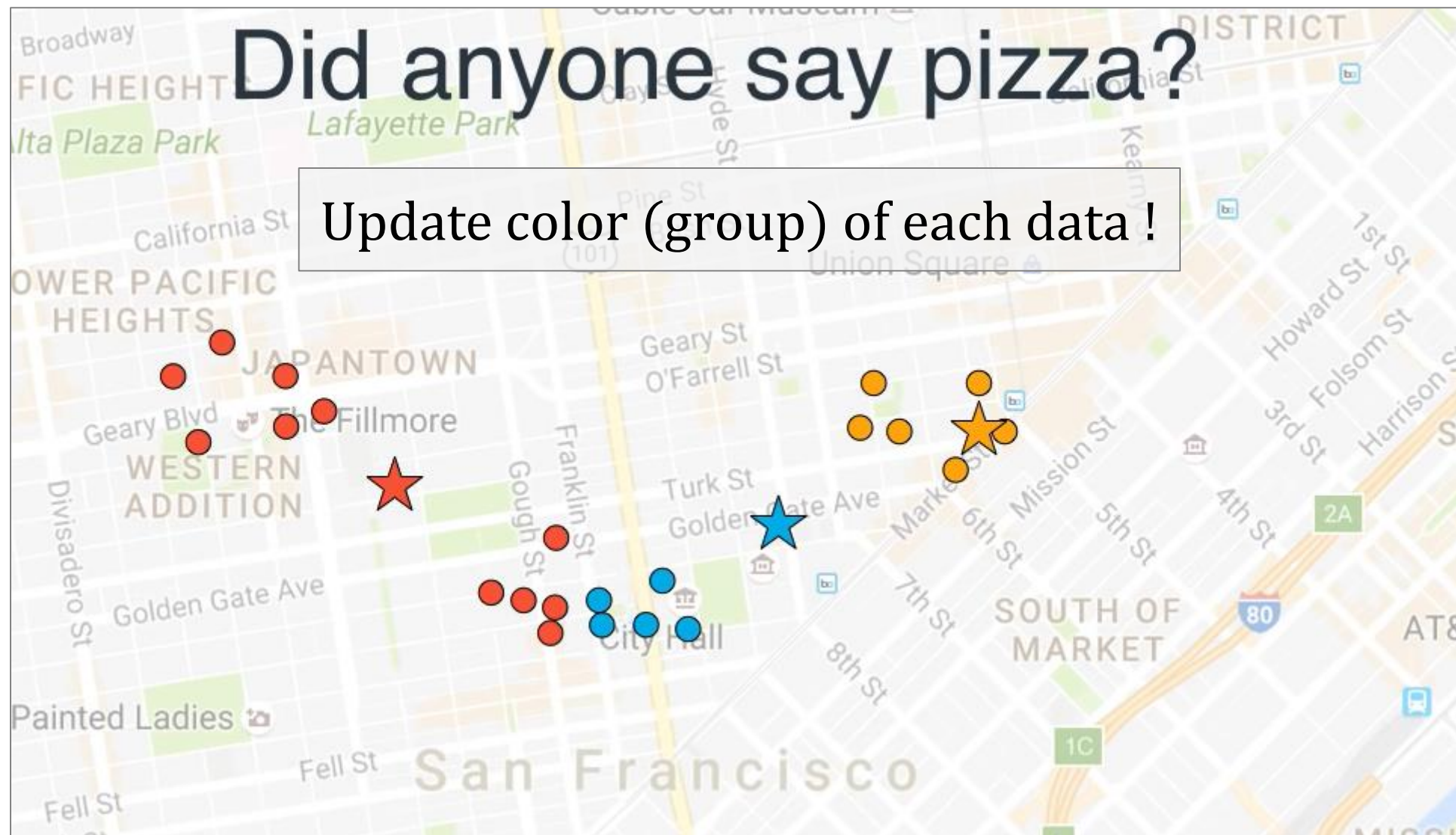
## $k$ –means clustering

<https://www.youtube.com/watch?v=IpGxLW0IZy4&t=1091s>



## $k$ – means clustering

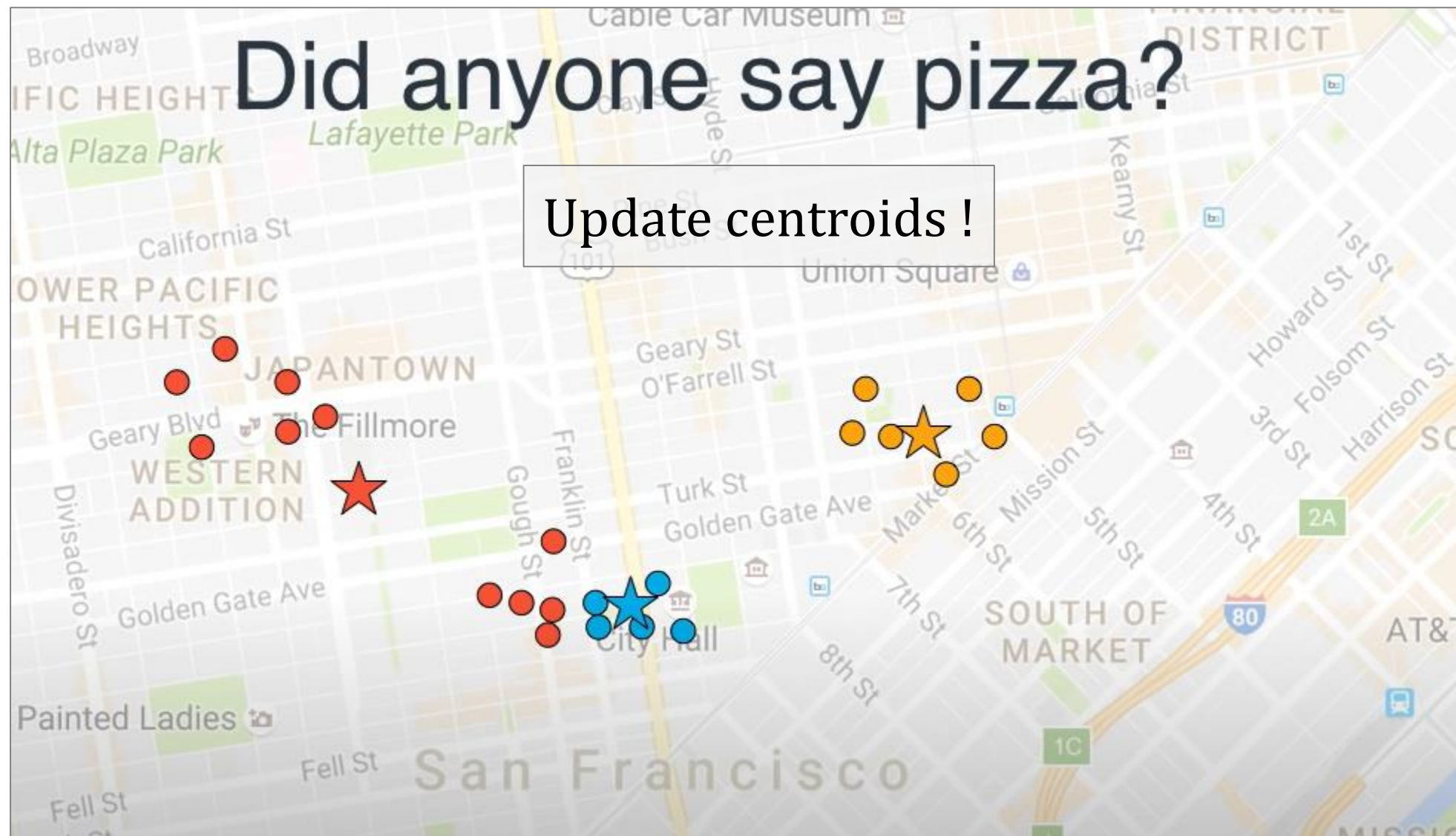
<https://www.youtube.com/watch?v=IpGxLW0IZy4&t=1091s>





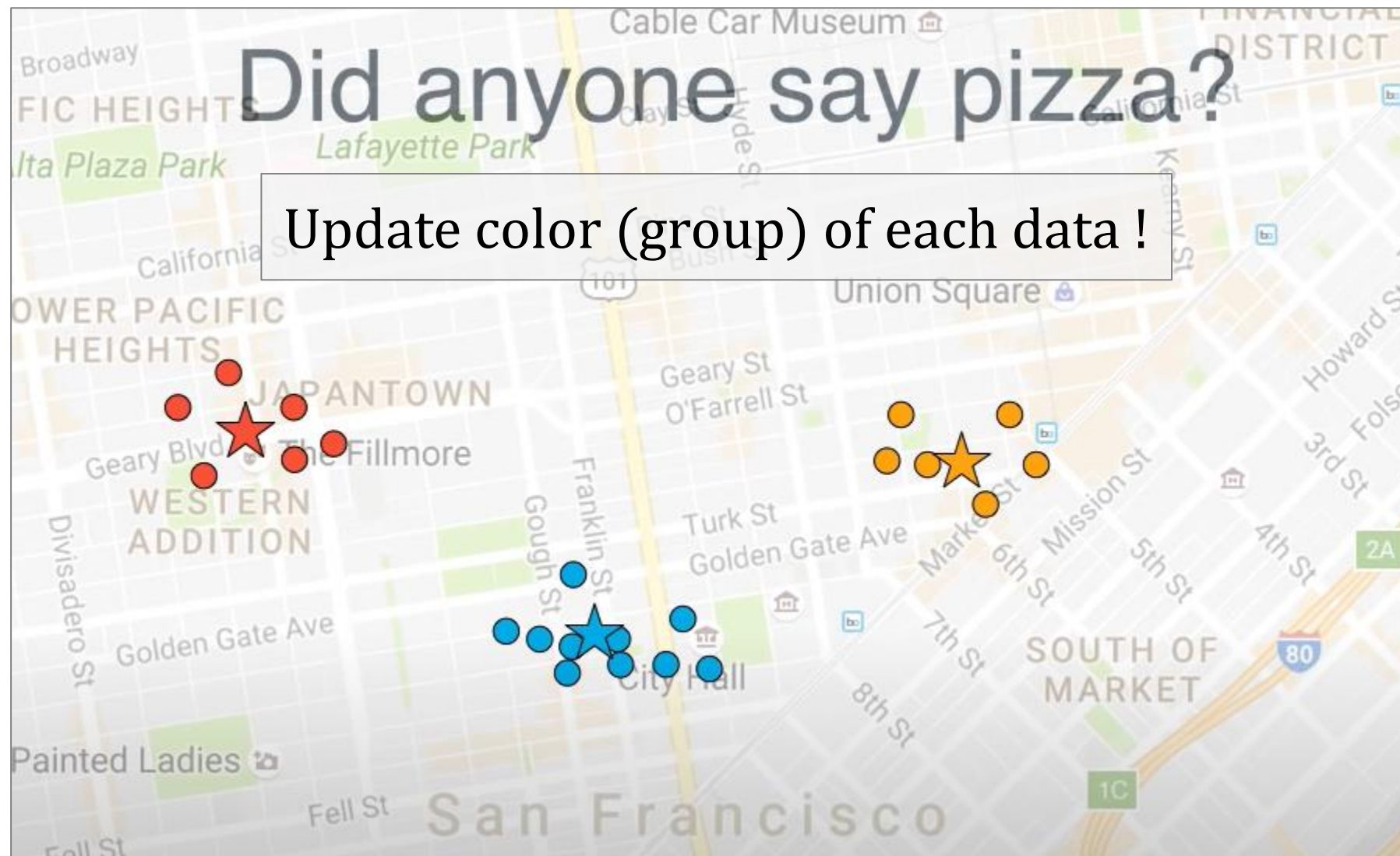
## $k$ –means clustering

<https://www.youtube.com/watch?v=IpGxLW0IZy4&t=1091s>

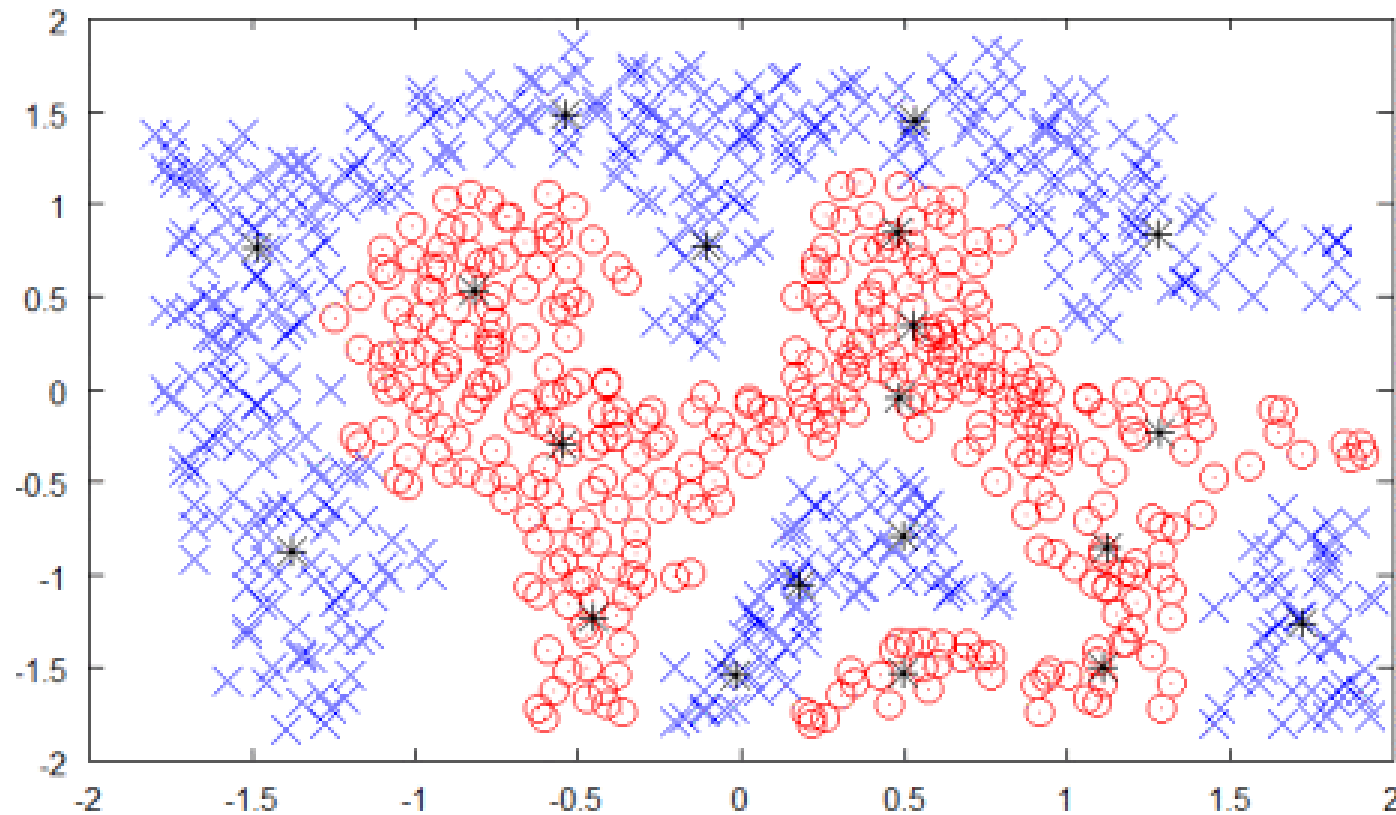


## $k$ –means clustering

<https://www.youtube.com/watch?v=IpGxLW0IZy4&t=1091s>



## Prototypes: central vectors



20 prototypes using 20-means clustering  
for two dimensional training data

10 –means clustering  
for instances belong to **blue group**

10 –means clustering  
for instances belong to **red group**

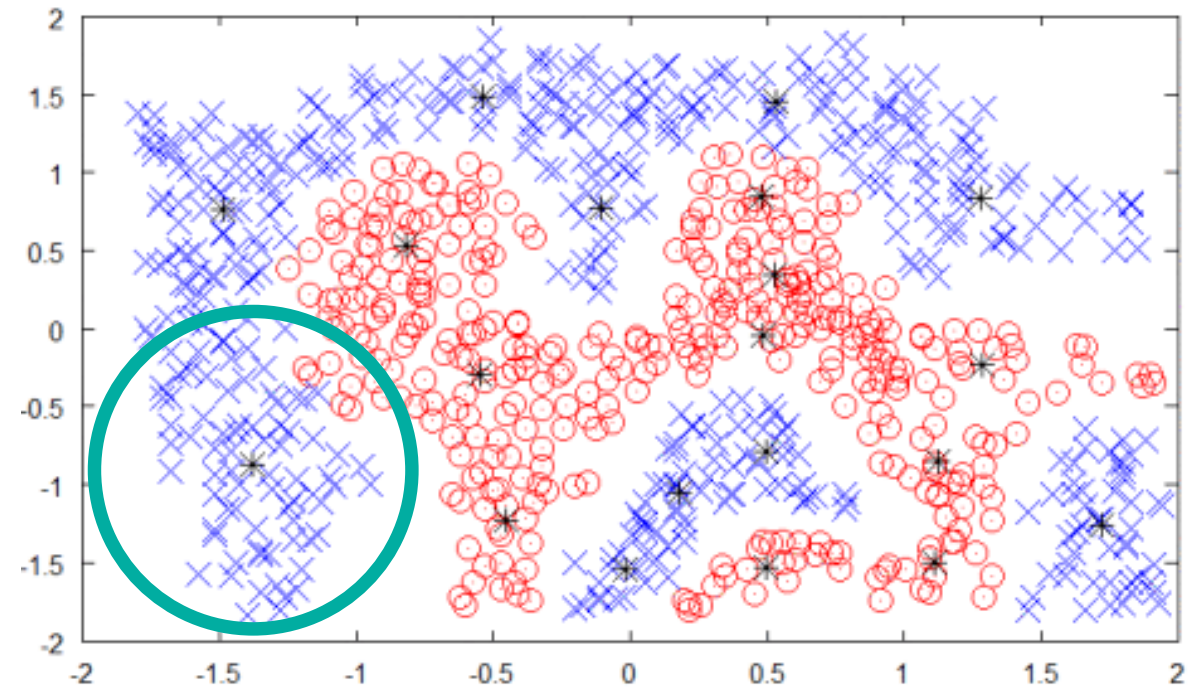
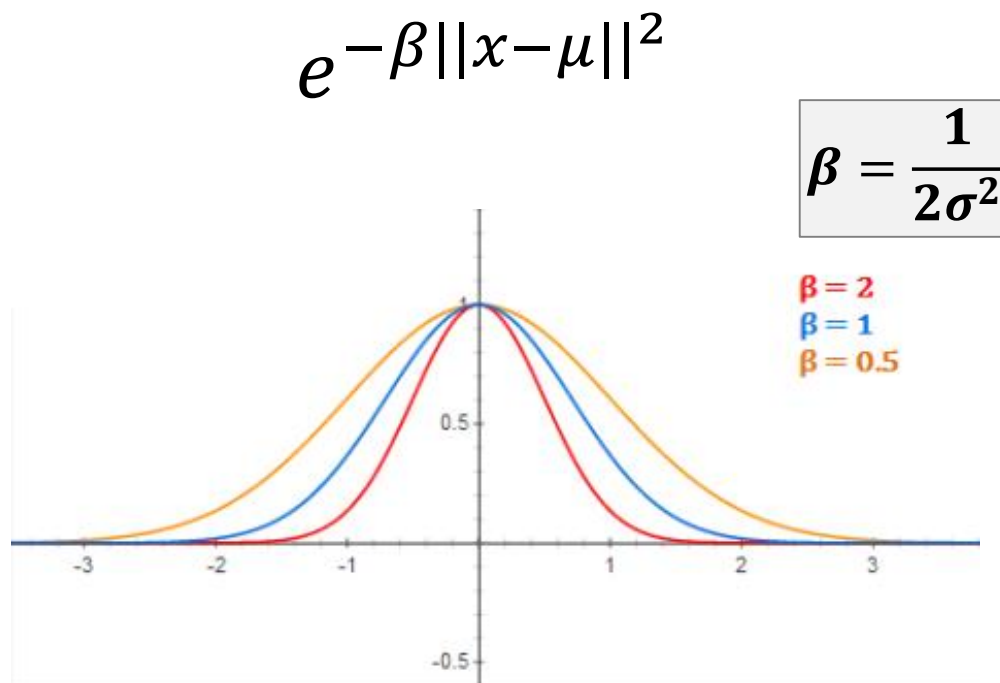
# Training the RBFN

Prototypes  
(center vectors)

Selecting  $\beta$  values

Output weights

$\beta$  values



$$\sigma = \frac{1}{m} \sum_{i=1}^m ||x_i - \mu||$$

where  $m$  is the # of blue instances  
in the green cluster

# Training the RBFN

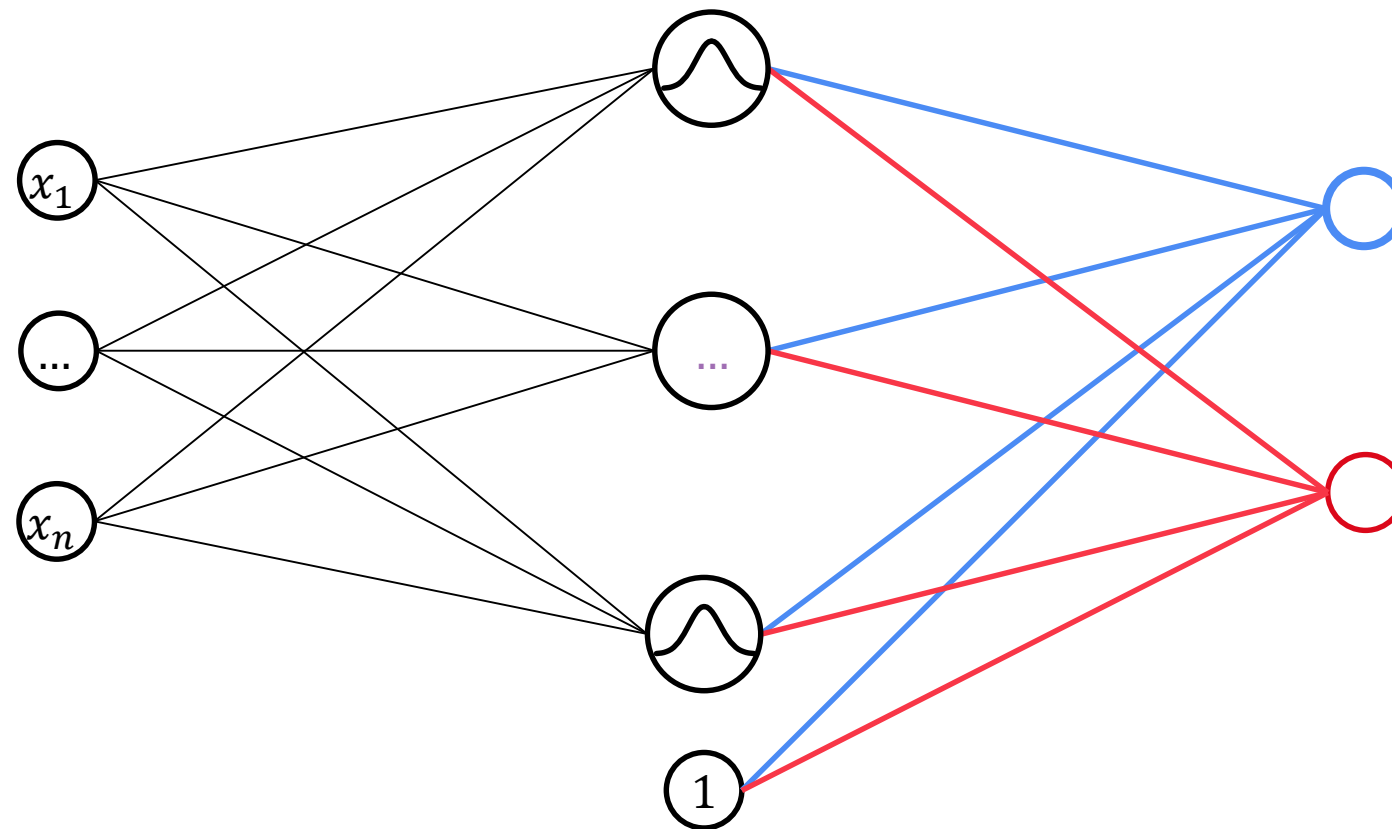
Prototypes  
(center vectors)

Selecting  $\beta$  values

**Output weights**

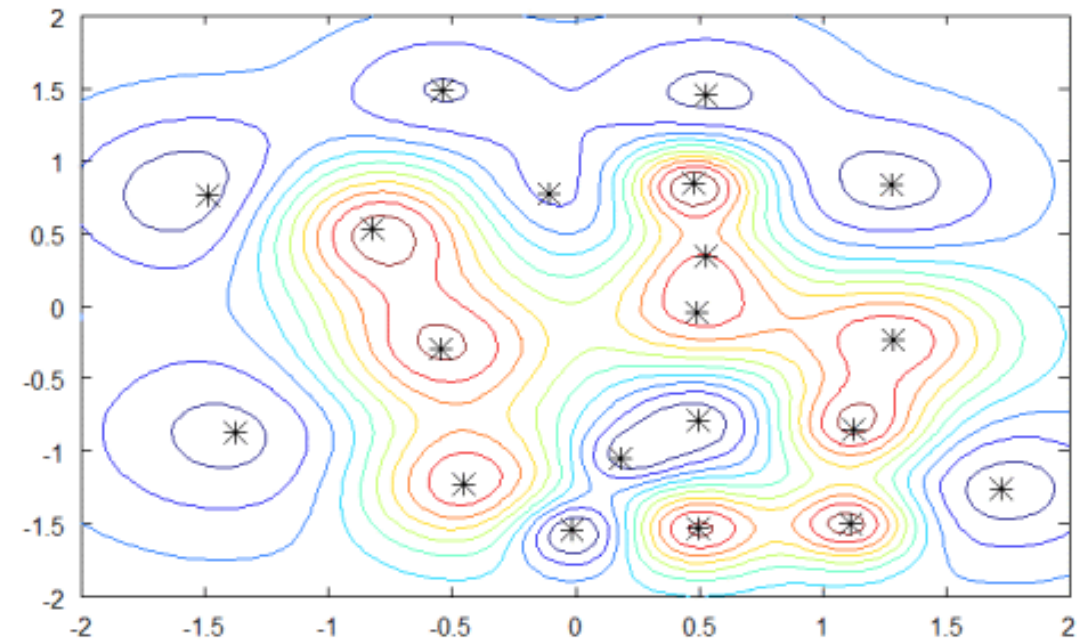
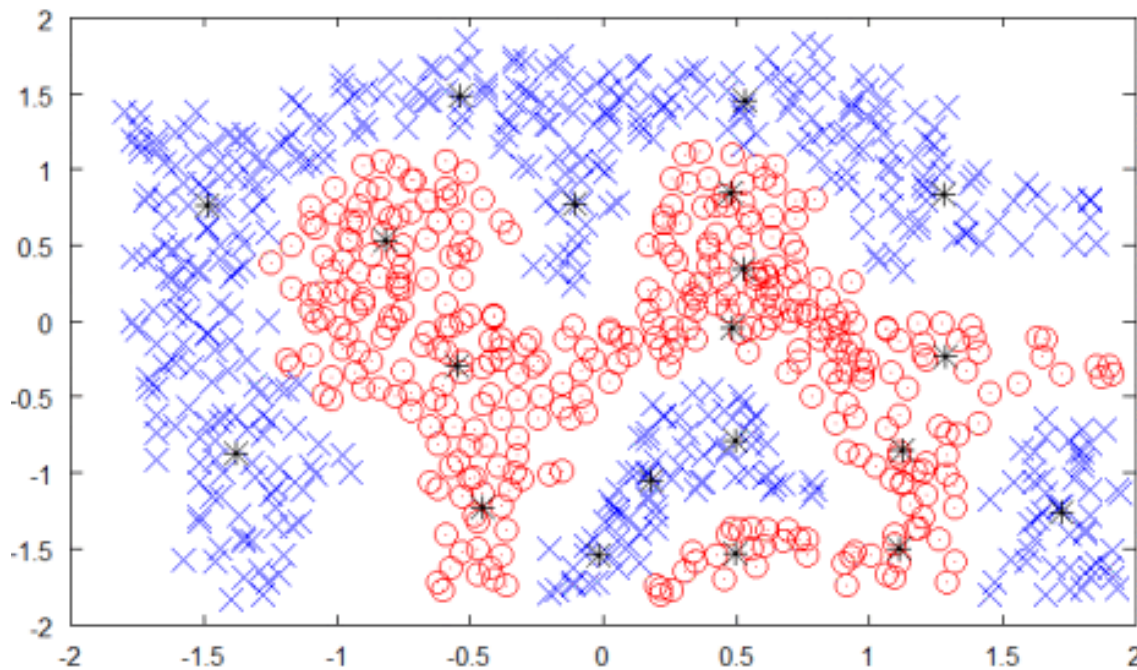
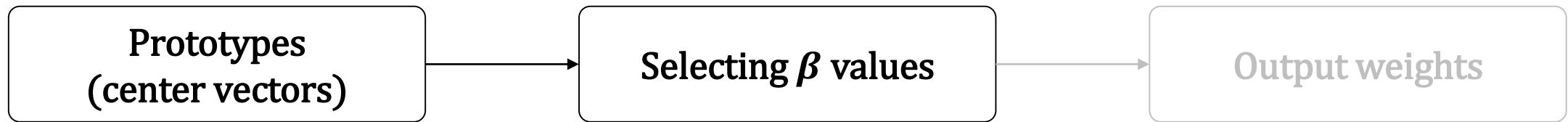
## Output weights

- Using **gradient decent**
- Training input = **activation values** of the RBF neurons
- Don't forget the **bias term**
- Gradient descent must be run **separately** for **each output node**





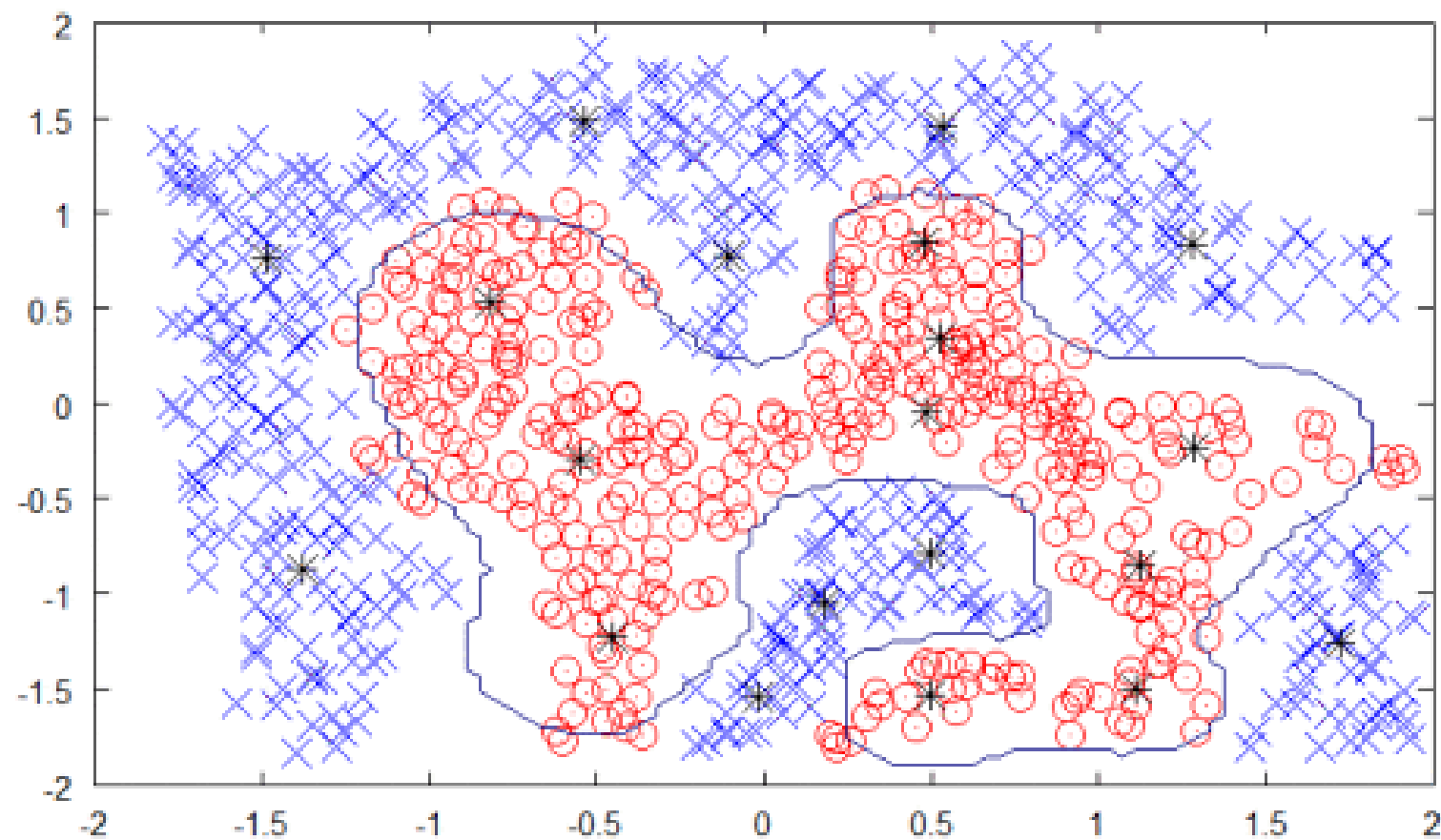
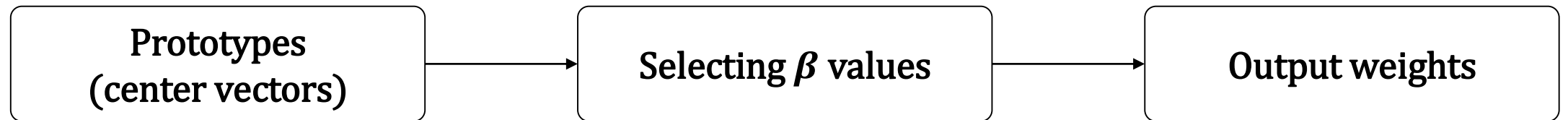
## Result with sample data



Contour plot for **red group**

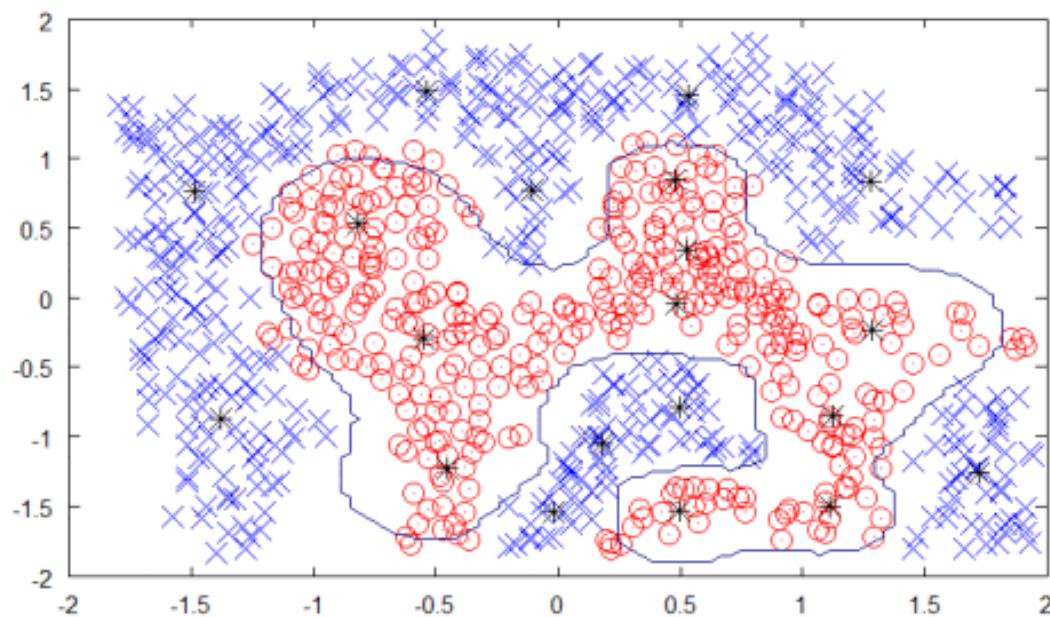


## Result with sample data



Decision boundary of RBFN

Time for SVM to play its role!



2D space with  $x_1$  and  $x_2$  axes

4D space with  $x_1, x_2,$   
 $x_{out}$  for blue and  $x_{out}$  for red axes

# Popular kernel functions

---

Fisher kernel

---

Graph kernels

---

Kernel smoother

Polynomial kernel

---

**Radial basis function kernel (RBF)**

---

String kernels

---

Neural tangent kernel

---

Neural network Gaussian process (NNGP) kernel

---

**SVM × RBF × Scikit-learn**

# Scikit-learn

- Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including
  - Decision tree & random forest
  - Linear regression
  - Perceptron
  - Artificial neural network
  - Support vector machine
  - Etc.

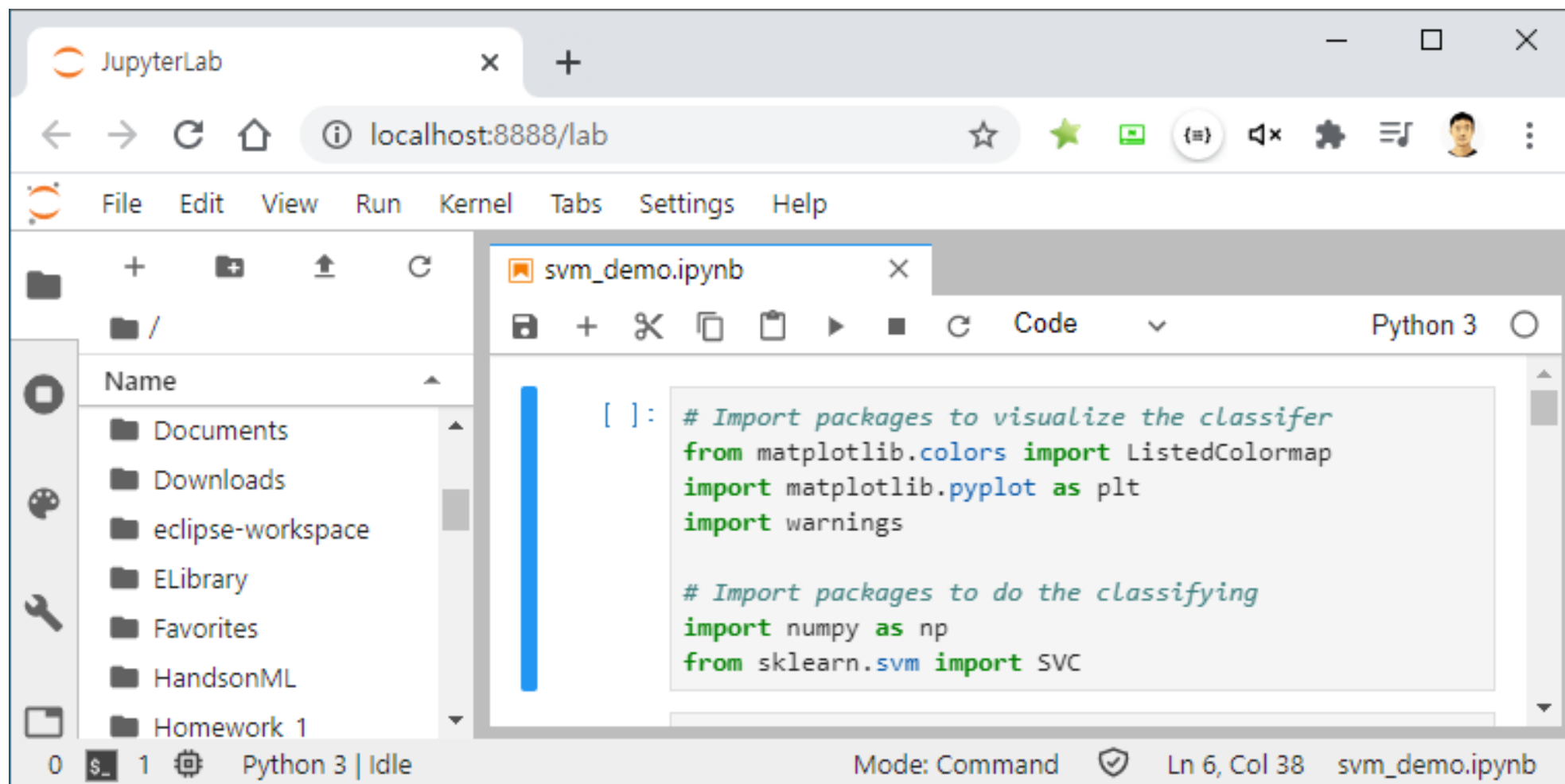
```
from sklearn.svm import SVC
```

```
svm = SVC() # Support vector machine for classification
```

```
svm.fit(X, y) # Train SVM on training data (X: a set of instances, y: a set of their labels)
```

## svm\_demo.ipynb 다운로드 & jupyter lab/notebook으로 오픈

- e-강의동 > 데이터사이언스응용 > 11주차 강의자료 > svm\_demo.ipynb



## svm\_demo.ipynb

- 필요한 package 및 library importing

```
# Import packages to visualize the classifier  
from matplotlib.colors import ListedColormap  
import matplotlib.pyplot as plt  
import warnings  
  
# Import packages to do the classifying  
import numpy as np  
from sklearn.svm import SVC
```

## svm\_demo.ipynb

- 학습 데이터(XOR problem) 생성 및 가시화(Visualization)

```
# Training data (XOR problem) generation and visualization
np.random.seed(0)
X_xor = np.random.randn(200, 2)
y_xor = np.logical_xor(X_xor[:, 0] > 0,
                       X_xor[:, 1] > 0)
y_xor = np.where(y_xor, 1, -1)

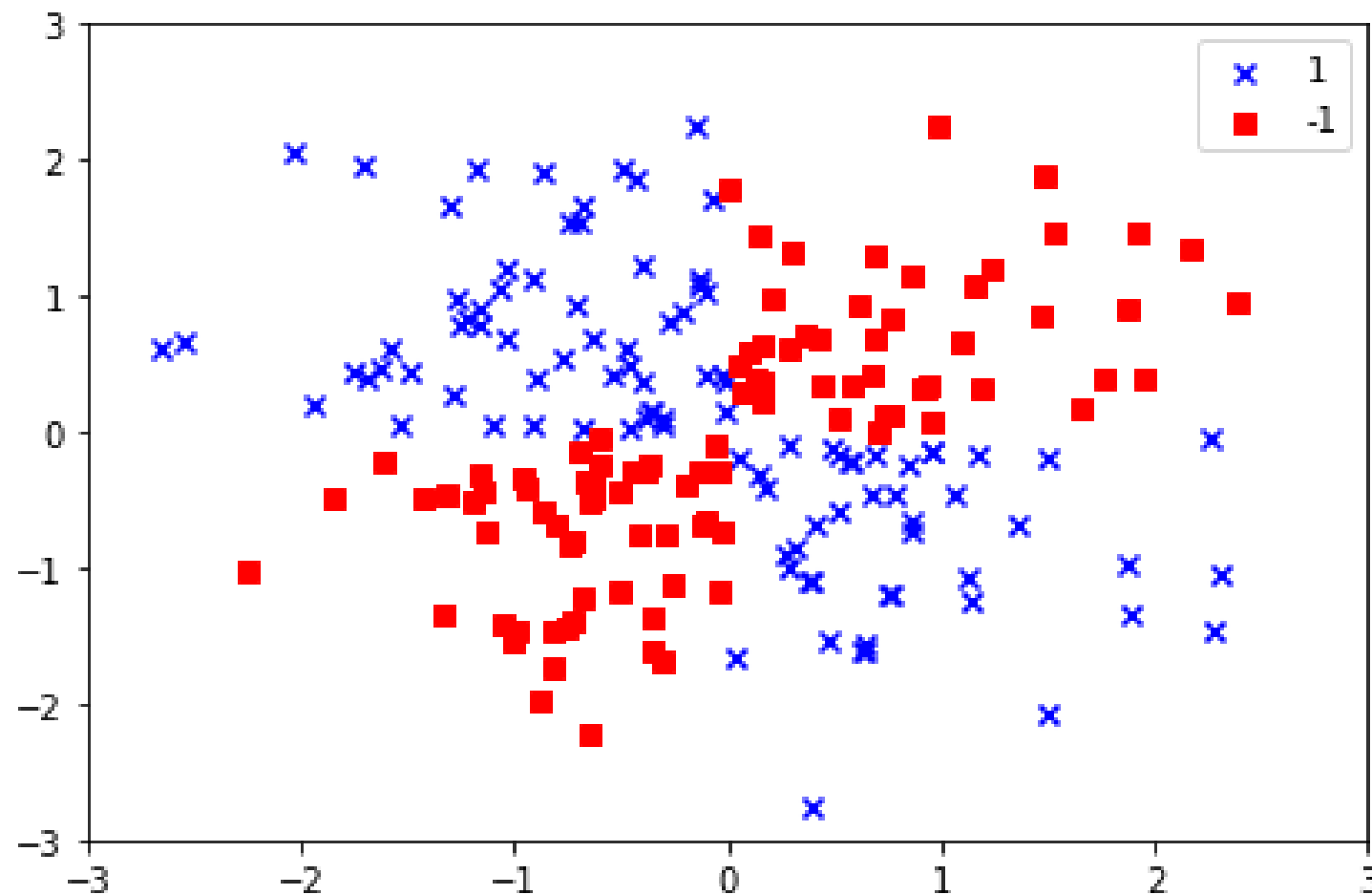
plt.scatter(X_xor[y_xor == 1, 0],
            X_xor[y_xor == 1, 1],
            c='b', marker='x',
            label='1')
plt.scatter(X_xor[y_xor == -1, 0],
            X_xor[y_xor == -1, 1],
            c='r',
            marker='s',
            label='-1')

plt.xlim([-3, 3])
plt.ylim([-3, 3])
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```



## svm\_demo.ipynb

- 학습 데이터(XOR problem) 생성 및 가시화



## svm\_demo.ipynb

- SVM의 decision boundary 가시화를 위한 코드

```
# For visualization of decision boundray of SVM
def versiontuple(v):
    return tuple(map(int, (v.split("."))))

def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):

    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
                    alpha=0.8, c=cmap(idx),
                    marker=markers[idx], label=cl)
```

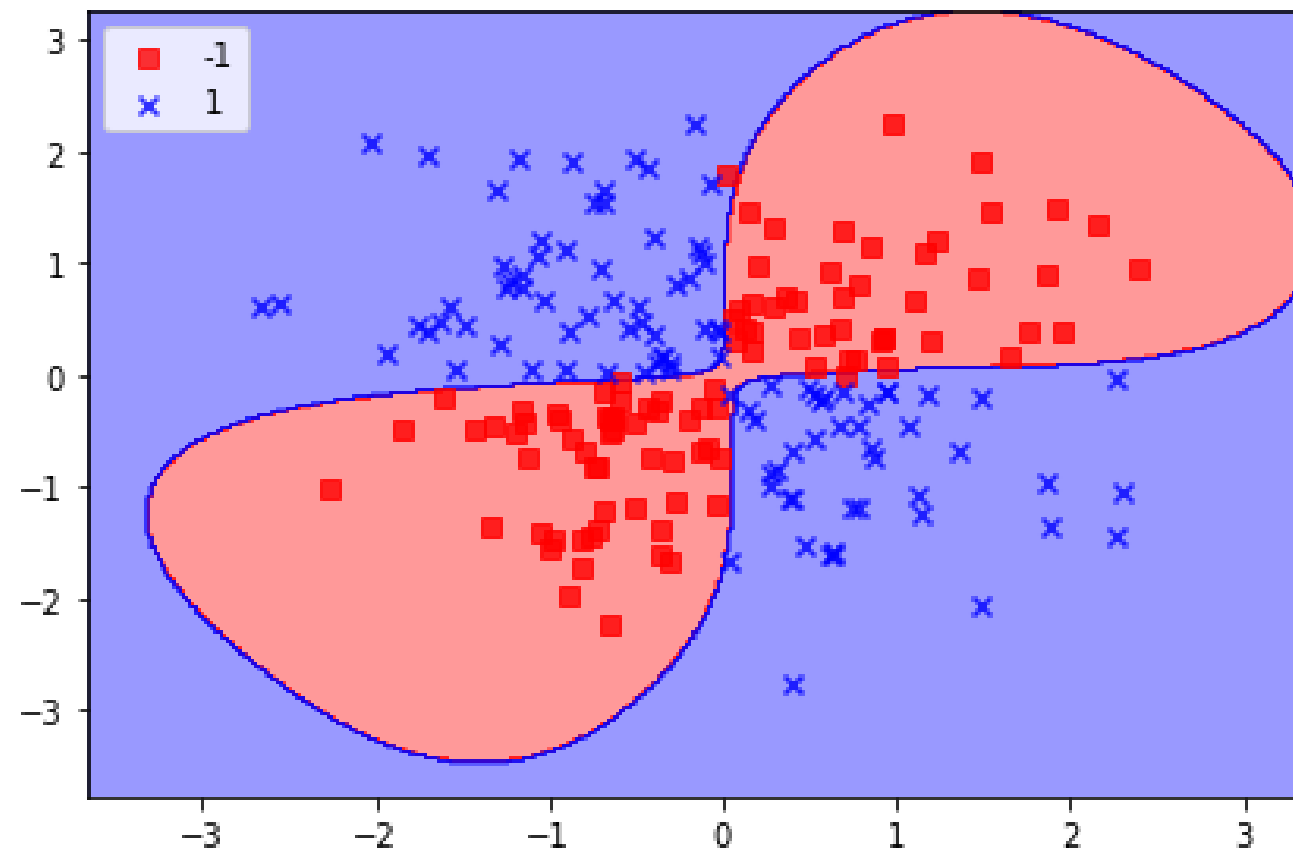
## svm\_demo.ipynb

- SVM 학습 및 decision boundary 가시화


```
# Create and train a SVC classifier
svm = SVC()
svm.fit(X_xor, y_xor)

# Visualize the decision boundaries
plot_decision_regions(X_xor, y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```

# svm\_demo.ipynb



# sklearn.svm.SVC

 [Install](#) [User Guide](#) [API](#) [Examples](#) [More ▾](#)

[Prev](#) [Up](#) [Next](#)

**scikit-learn 0.23.2**  
[Other versions](#)

Please **cite us** if you use the software.

**sklearn.svm.SVC**  
[Examples using sklearn.svm.SVC](#)

## sklearn.svm.SVC


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

[\[source\]](#)

**kernel** : {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, default='rbf'

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (n\_samples, n\_samples).

# sklearn.svm.SVC

 [Install](#) [User Guide](#) [API](#) [Examples](#) [More ▾](#)  [Go](#)

[Prev](#) [Up](#) [Next](#)

**scikit-learn 0.23.2**  
[Other versions](#)

Please **cite us** if you use the software.

**sklearn.svm.SVC**  
[Examples using sklearn.svm.SVC](#)

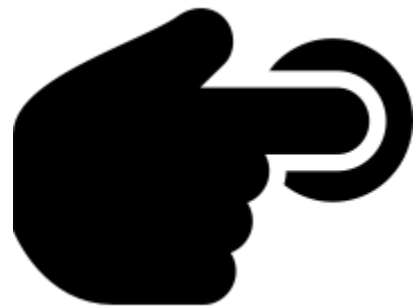
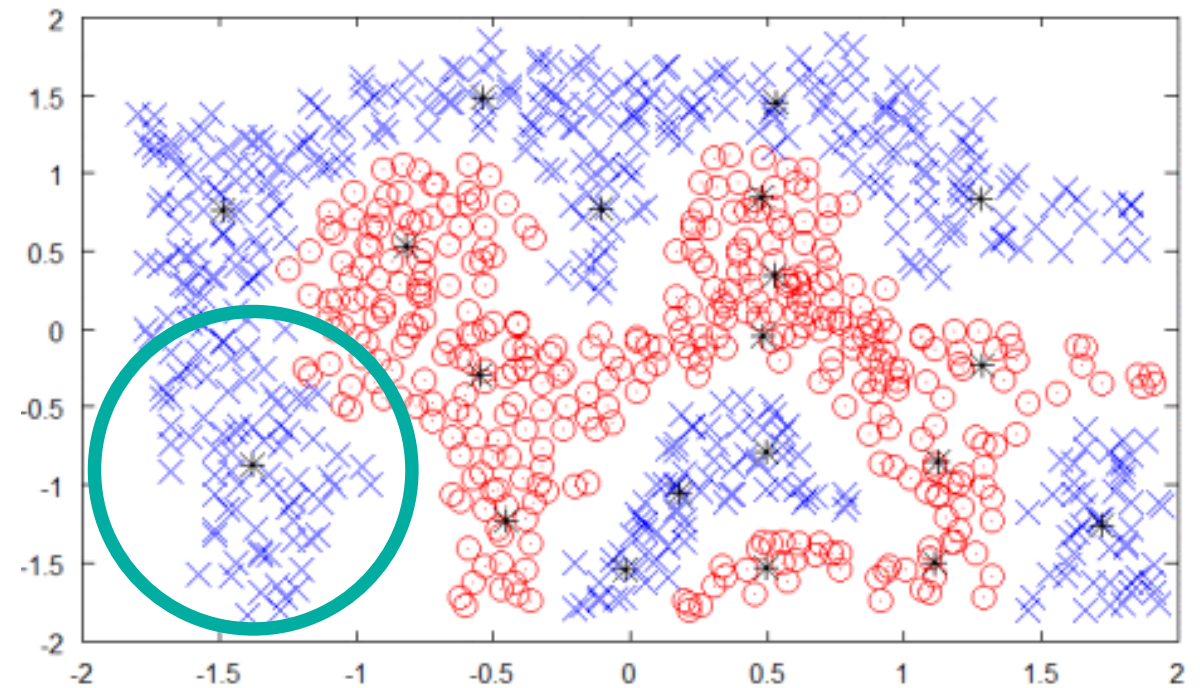
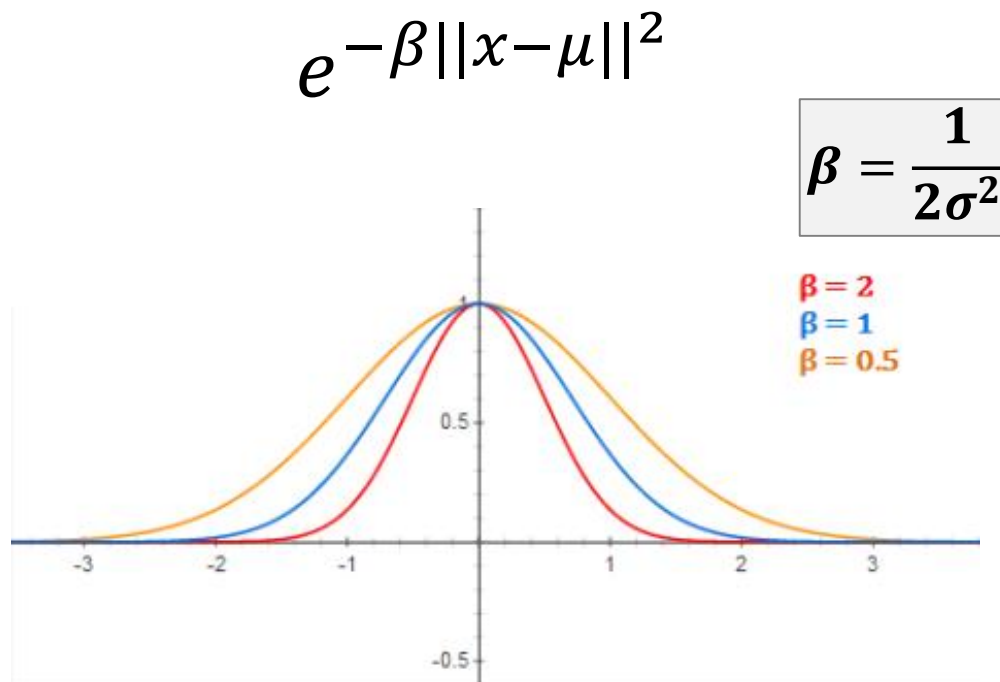
## sklearn.svm.SVC

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

[\[source\]](#)

Gamma & C

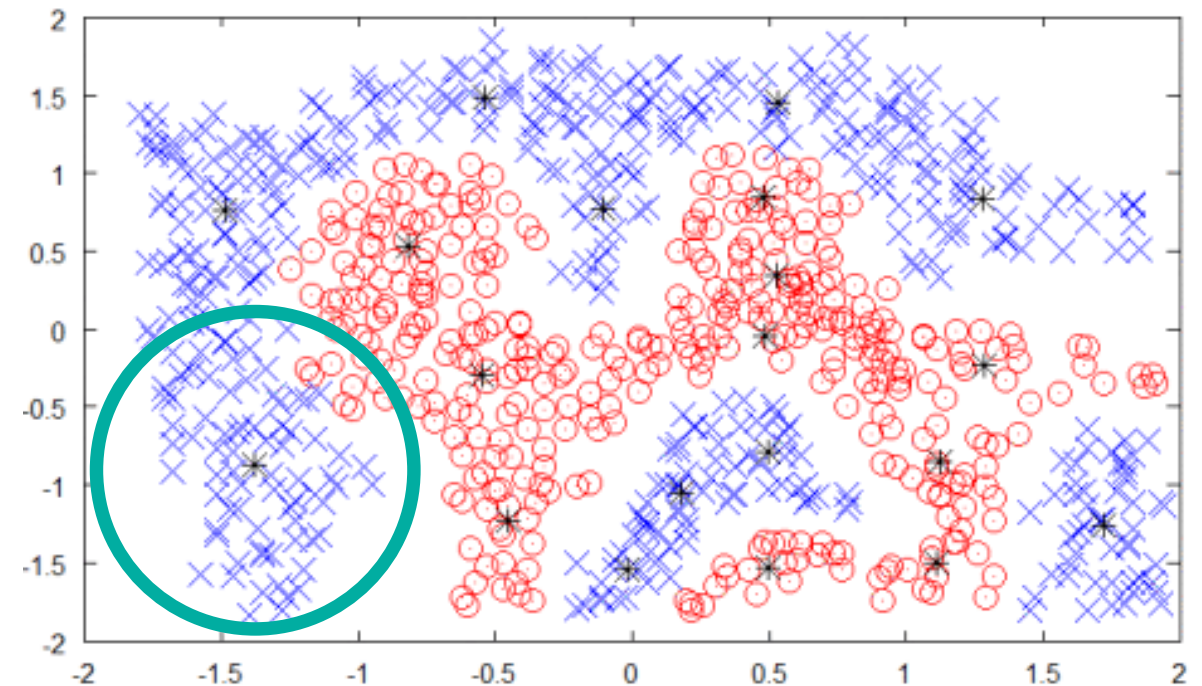
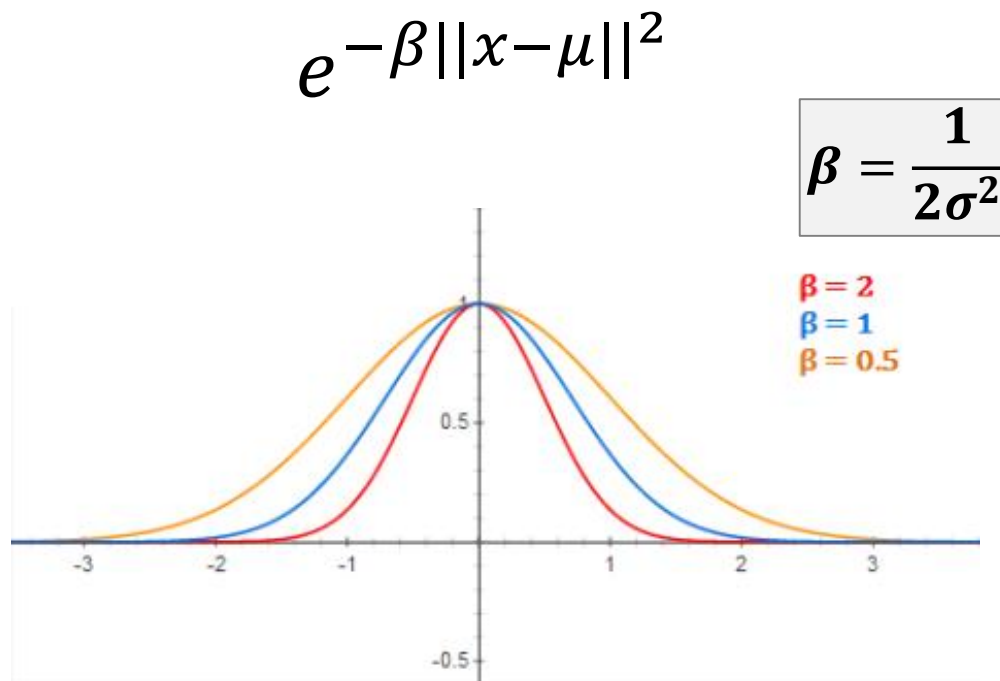
Parameter Gamma  $\gamma = \beta$



$$\sigma = \frac{1}{m} \sum_{i=1}^m ||x_i - \mu||$$

where  $m$  is the # of blue instances  
in the green cluster

Parameter Gamma  $\gamma = \beta$



With **high  $\beta$** : (relatively) few instances influence the decision boundary

With **low  $\beta$** : (relatively) many instances influence the decision boundary

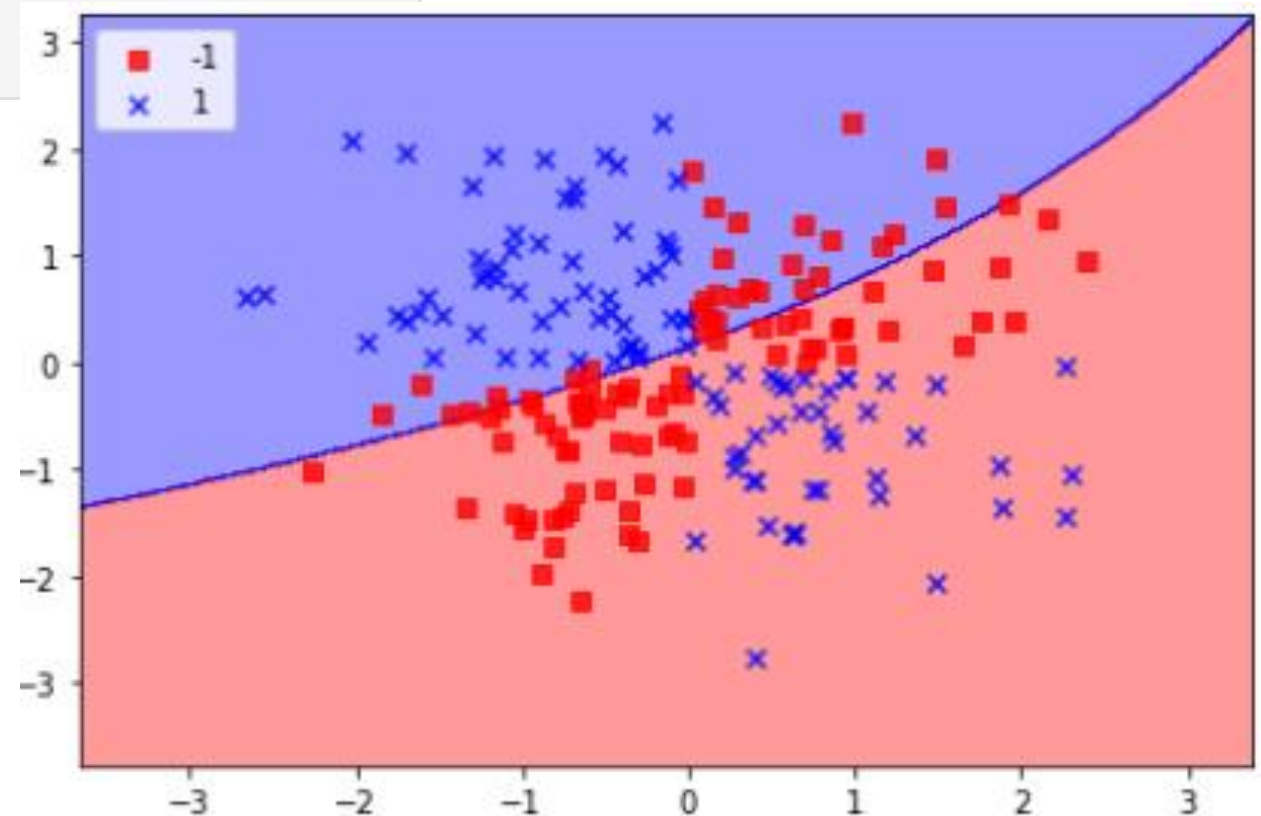


## Demonstration with various Gammas

- Gamma = 0.01

```
# Create and train a SVC classifier
svm = SVC(kernel='rbf', gamma = 0.01)
svm.fit(X_xor, y_xor)

# Visualize the decision boundaries
plot_decision_regions(X_xor, y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```

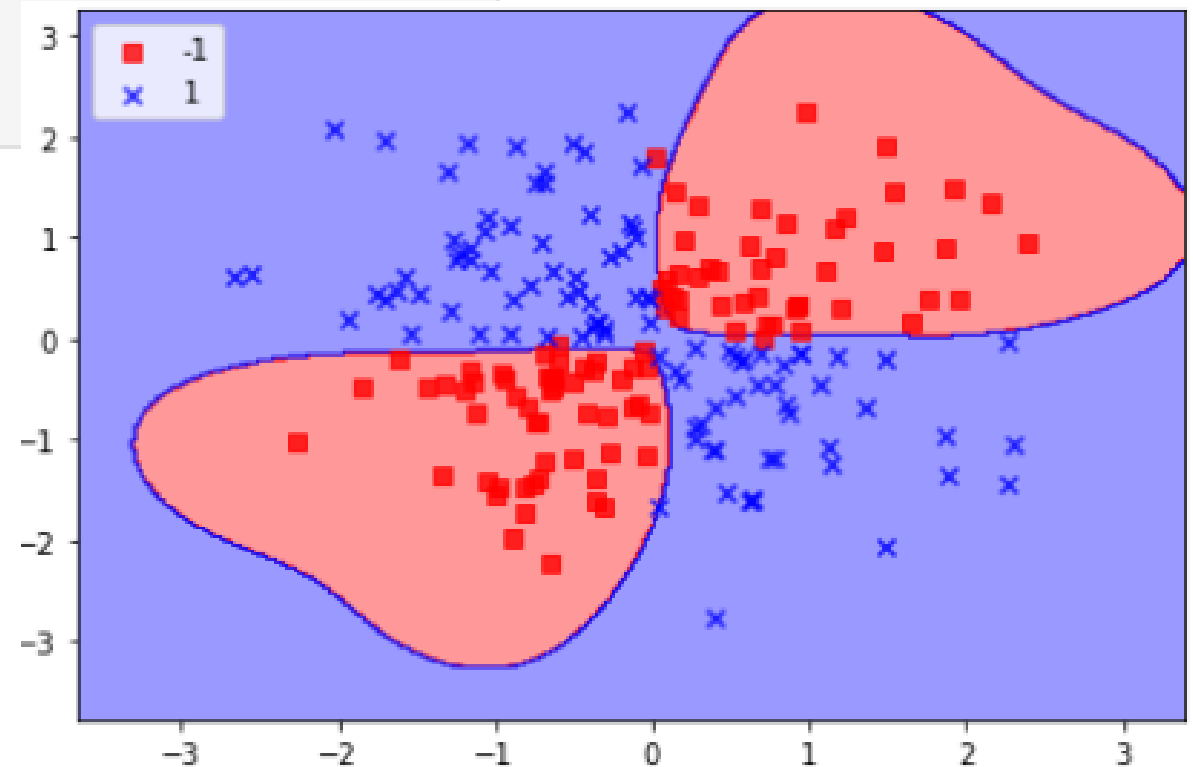


## Demonstration with various Gammas

- Gamma = 1.0

```
# Create and train a SVC classifier
svm = SVC(kernel='rbf', gamma = 1.0)
svm.fit(X_xor, y_xor)

# Visualize the decision boundaries
plot_decision_regions(X_xor, y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```

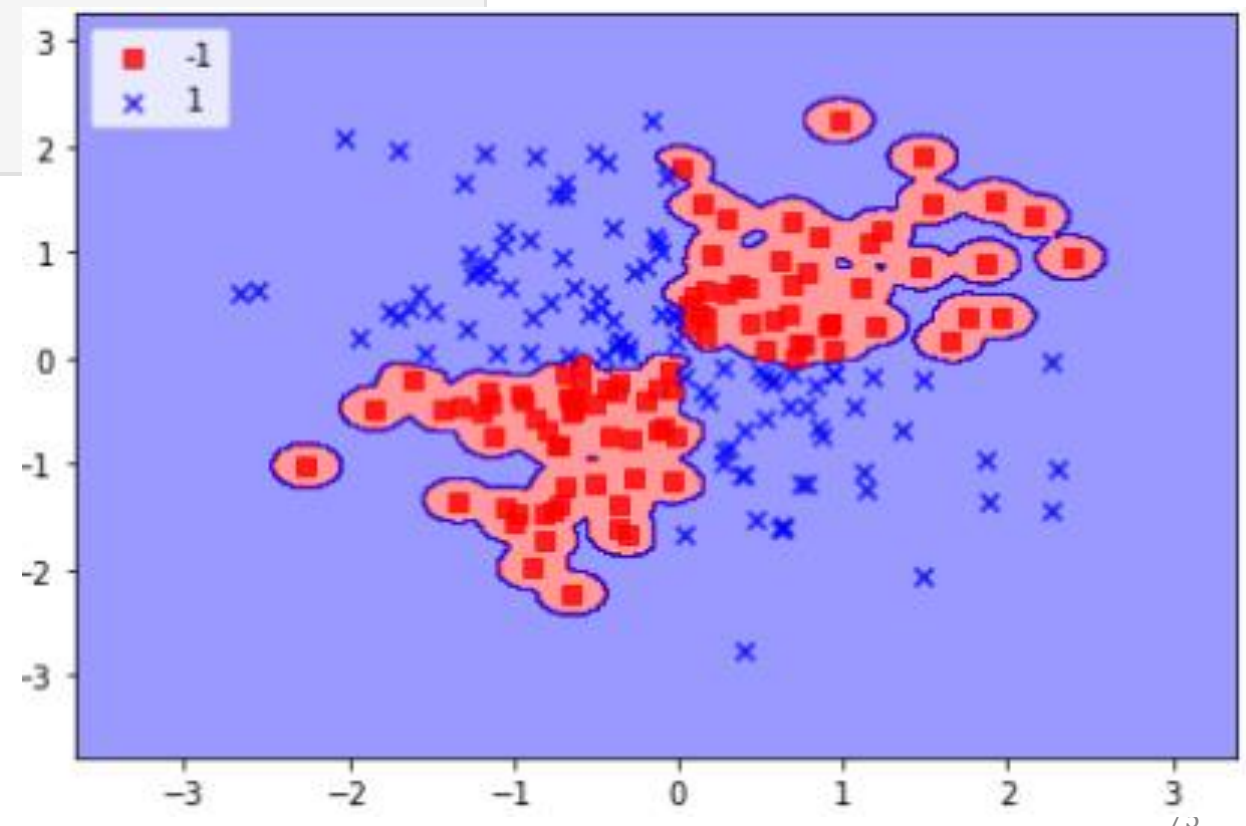


## Demonstration with various Gammas


- Gamma = 100

```
# Create and train a SVC classifier
svm = SVC(kernel='rbf', gamma = 100)
svm.fit(X_xor, y_xor)

# Visualize the decision boundaries
plot_decision_regions(X_xor, y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```



# sklearn.svm.SVC

 [Install](#) [User Guide](#) [API](#) [Examples](#) [More ▾](#)  [Go](#)

[Prev](#) [Up](#) [Next](#)

**scikit-learn 0.23.2**  
[Other versions](#)

Please **cite us** if you use the software.

**sklearn.svm.SVC**  
[Examples using sklearn.svm.SVC](#)

## sklearn.svm.SVC

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

[\[source\]](#)

Gamma & C

## Parameter $C$

- Penalty for misclassifying a data point

With **high  $C$** : heavily penalized for misclassified data (overfitting: small margin)

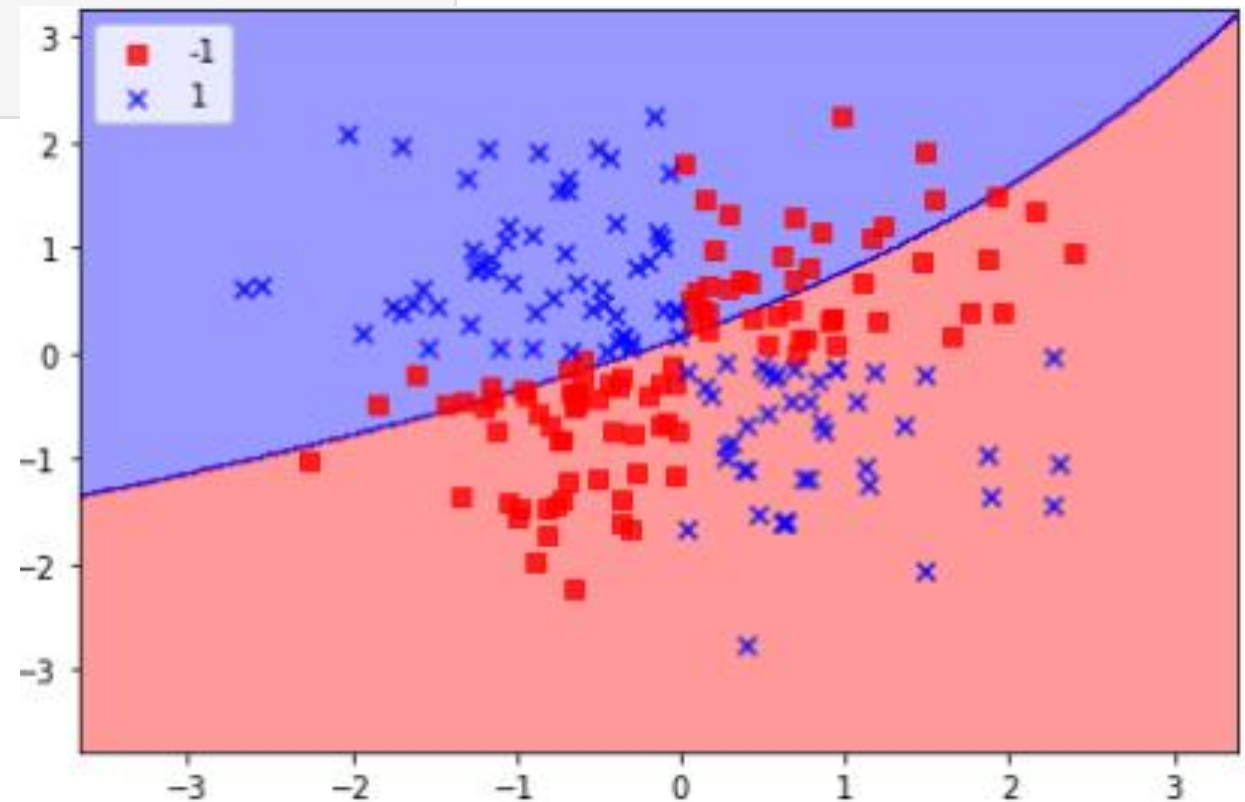
With **low  $C$** : okay with misclassified data points (underfitting: large margin)

## Demonstration with various $C$ s w/ fixed $\gamma = 0.01$

- $C = 1$

```
# Create and train a SVC classifier
svm = SVC(kernel='rbf', gamma = 0.01, C=1)
svm.fit(X_xor, y_xor)

# Visualize the decision boundaries
plot_decision_regions(X_xor, y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```

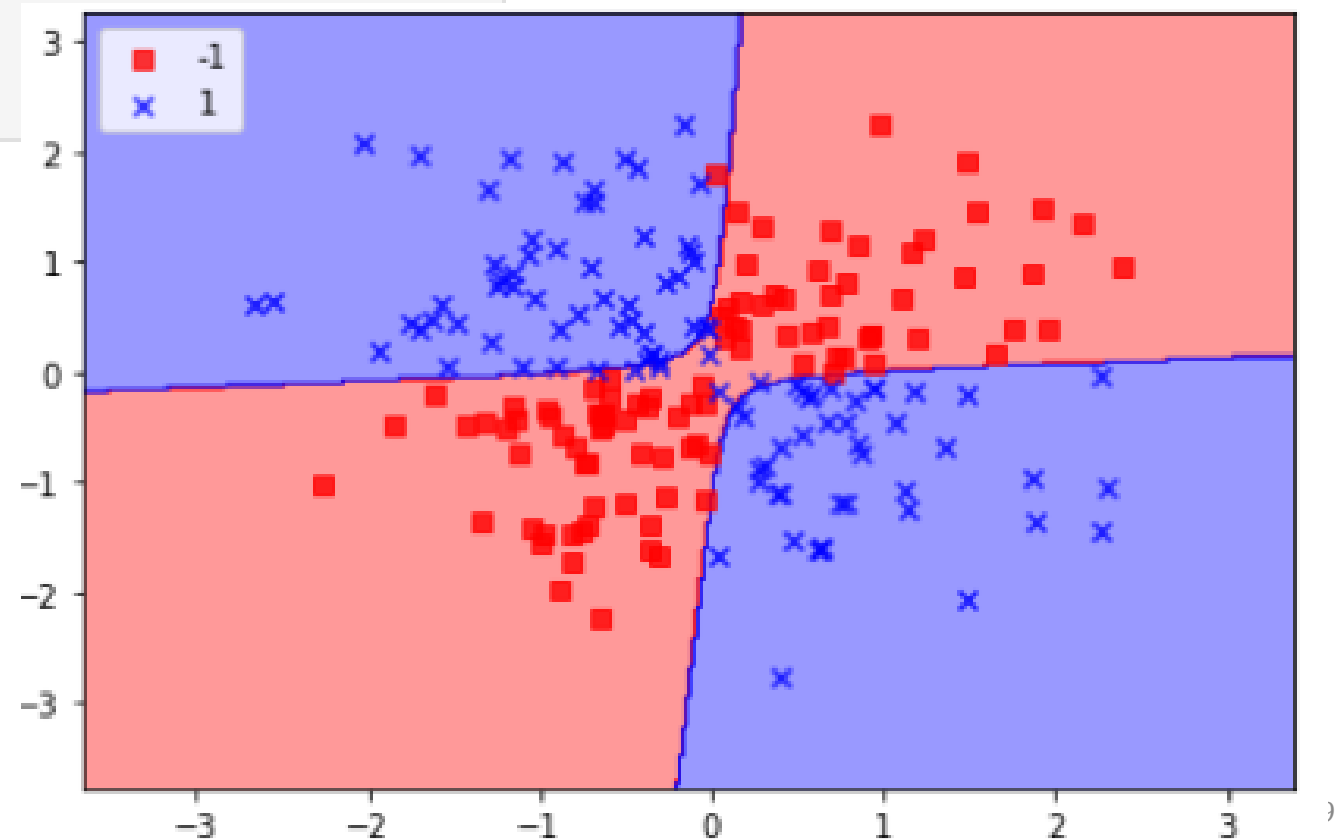


## Demonstration with various $C$ s w/ fixed $\gamma = 0.01$

- $C = 1,000$

```
# Create and train a SVC classifier
svm = SVC(kernel='rbf', gamma = 0.01, C=1000)
svm.fit(X_xor, y_xor)

# Visualize the decision boundaries
plot_decision_regions(X_xor, y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```

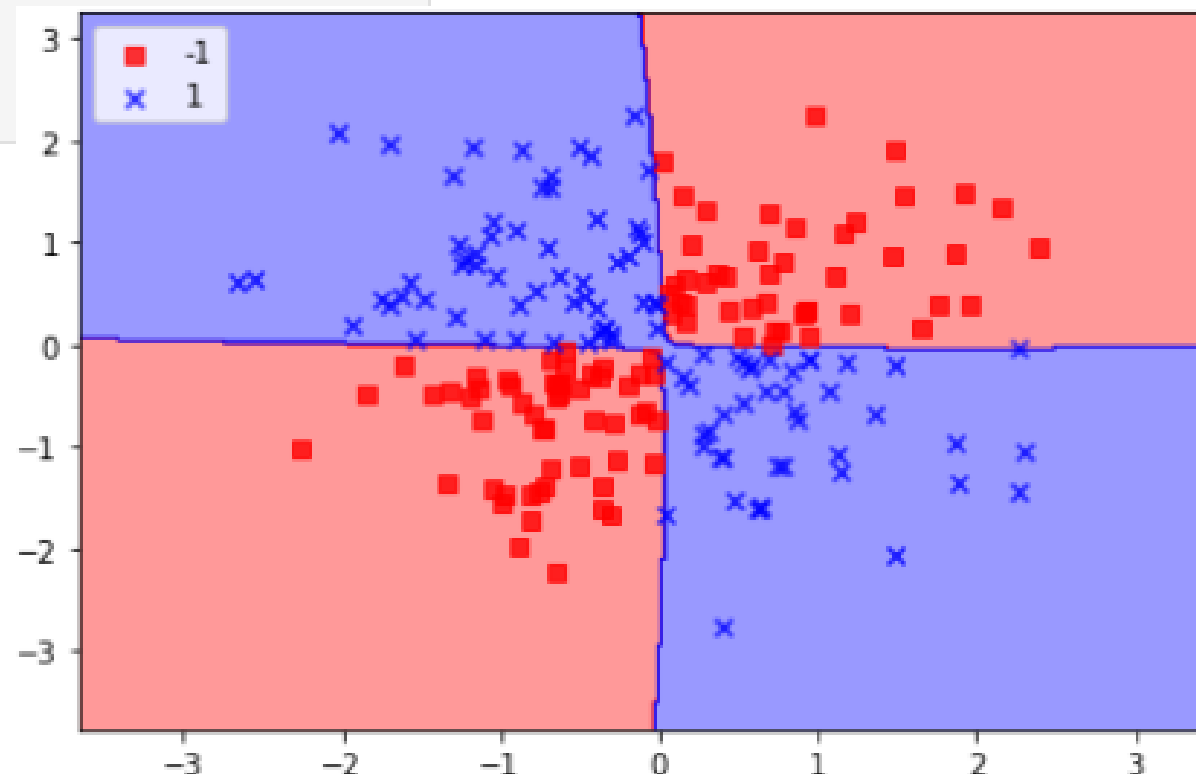


## Demonstration with various $C$ s w/ fixed $\gamma = 0.01$

- $C = 100,000$

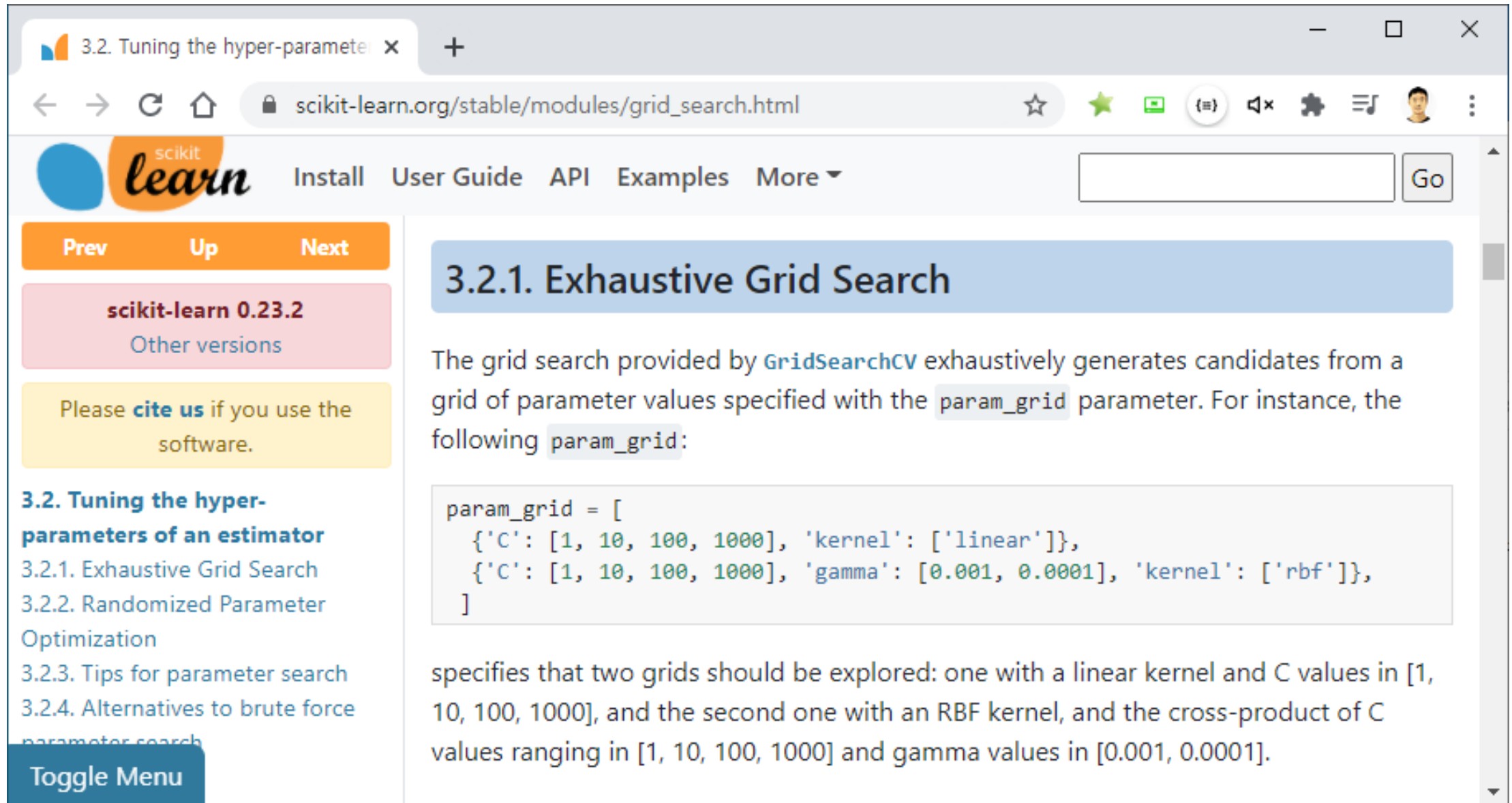
```
# Create and train a SVC classifier
svm = SVC(kernel='rbf', gamma = 0.01, C=1000000)
svm.fit(X_xor, y_xor)

# Visualize the decision boundaries
plot_decision_regions(X_xor, y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```





[https://scikit-learn.org/stable/modules/grid\\_search.html](https://scikit-learn.org/stable/modules/grid_search.html)



The screenshot shows a web browser window with the URL `scikit-learn.org/stable/modules/grid_search.html`. The page title is "3.2. Tuning the hyper-parameters of an estimator". The main content area is titled "3.2.1. Exhaustive Grid Search". It explains that the `GridSearchCV` class exhaustively generates candidates from a grid of parameter values specified with the `param_grid` parameter. An example `param_grid` is shown in a code block:

```
param_grid = [
    {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
    {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
]
```

The text continues: "specifies that two grids should be explored: one with a linear kernel and C values in [1, 10, 100, 1000], and the second one with an RBF kernel, and the cross-product of C values ranging in [1, 10, 100, 1000] and gamma values in [0.001, 0.0001]."

The left sidebar contains navigation links: "Prev", "Up", "Next", "scikit-learn 0.23.2", "Other versions", "Please cite us if you use the software.", and a list of sections: "3.2. Tuning the hyper-parameters of an estimator", "3.2.1. Exhaustive Grid Search", "3.2.2. Randomized Parameter Optimization", "3.2.3. Tips for parameter search", and "3.2.4. Alternatives to brute force parameter search". A "Toggle Menu" button is at the bottom of the sidebar.

# ODPia 소셜맵 보완 과제 최종 보고

(발표용)

---

서울대학교 의생명지식공학연구실

2015. 12. 22

## 0. 목 차

1. 문제 정의 및 모델링

2. 상세 목표 및 전략

3. 성능 평가

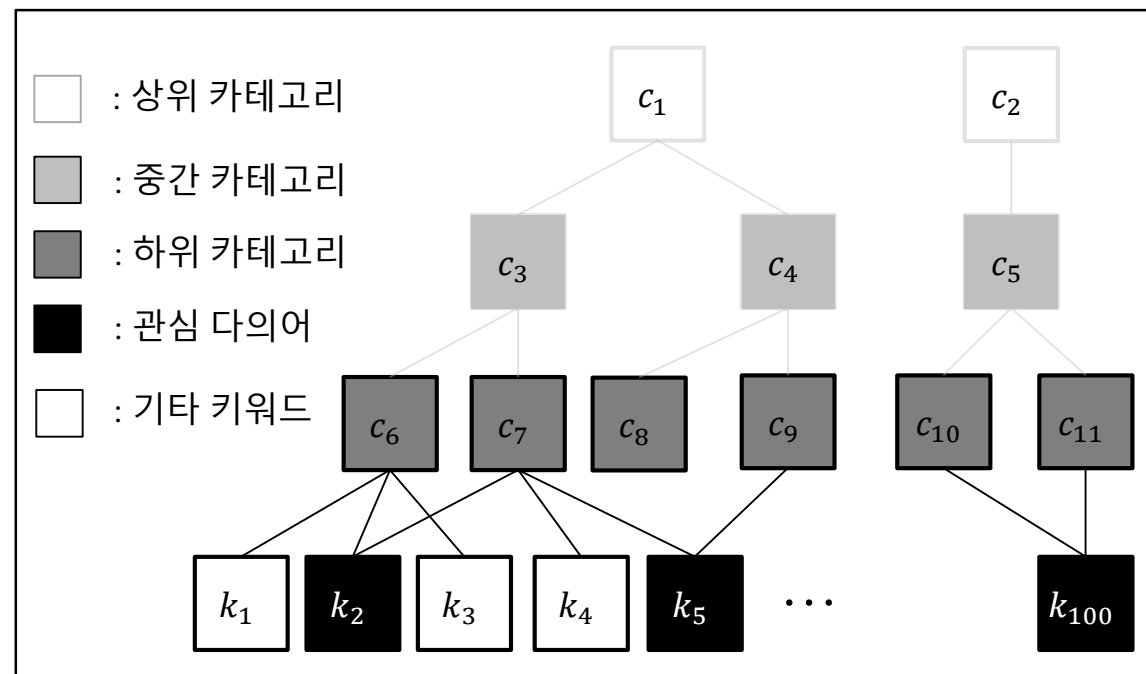
4. 제한점 및 향후 계획

# 1. 문제 정의 및 목표 모 델링

## ■ 문제 정의

- 대상: ODPia 소셜 맵 내 존재하는 다의어 집합 (이하 관심 다의어)
- 다의어: 동일한 syntax를 갖지만, 문맥에 따라 다양한 의미로 해석 가능한 literal
- 문맥: 관심 다의어가 등장하는 문장/문서
- 목표: 문맥 내, 관심 다의어 집합의 어의 증의성 (Word Sense Disambiguation, 이하 WSD) 해소
- 결과: 문맥 내, 관심 다의어 집합의 WSD 해소 모듈 및 정량적 성능 검증 결과

<그림 1. ODPia 소셜 맵>

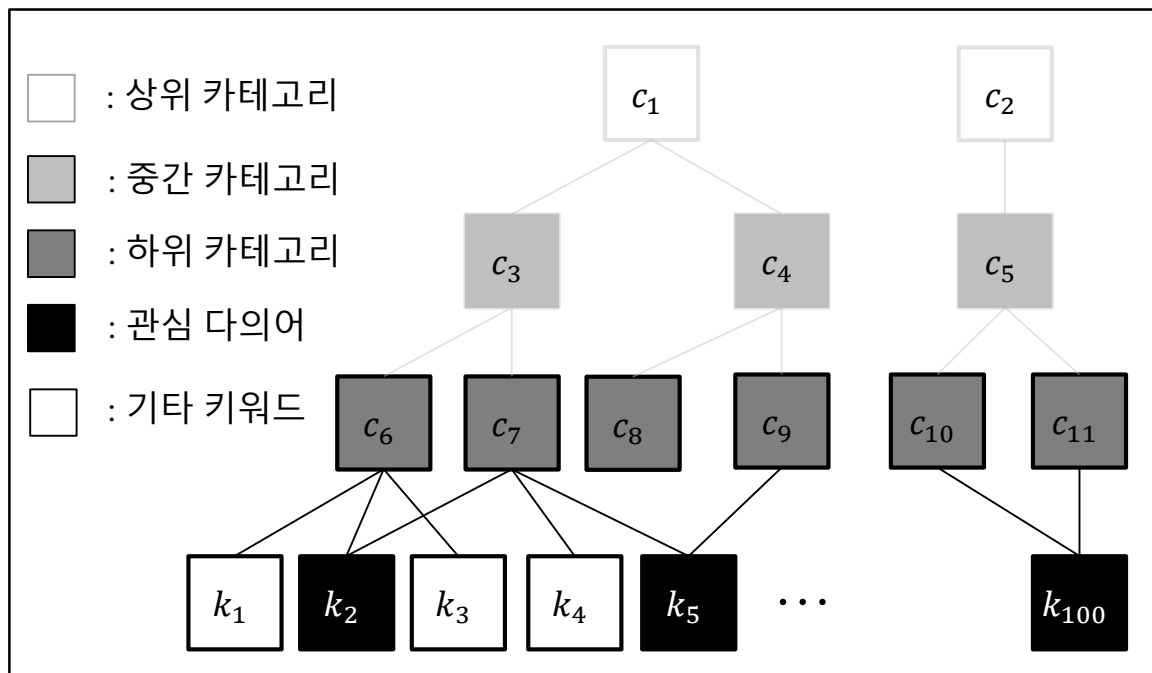


# 1. 문제 정의 및 목표 모델링

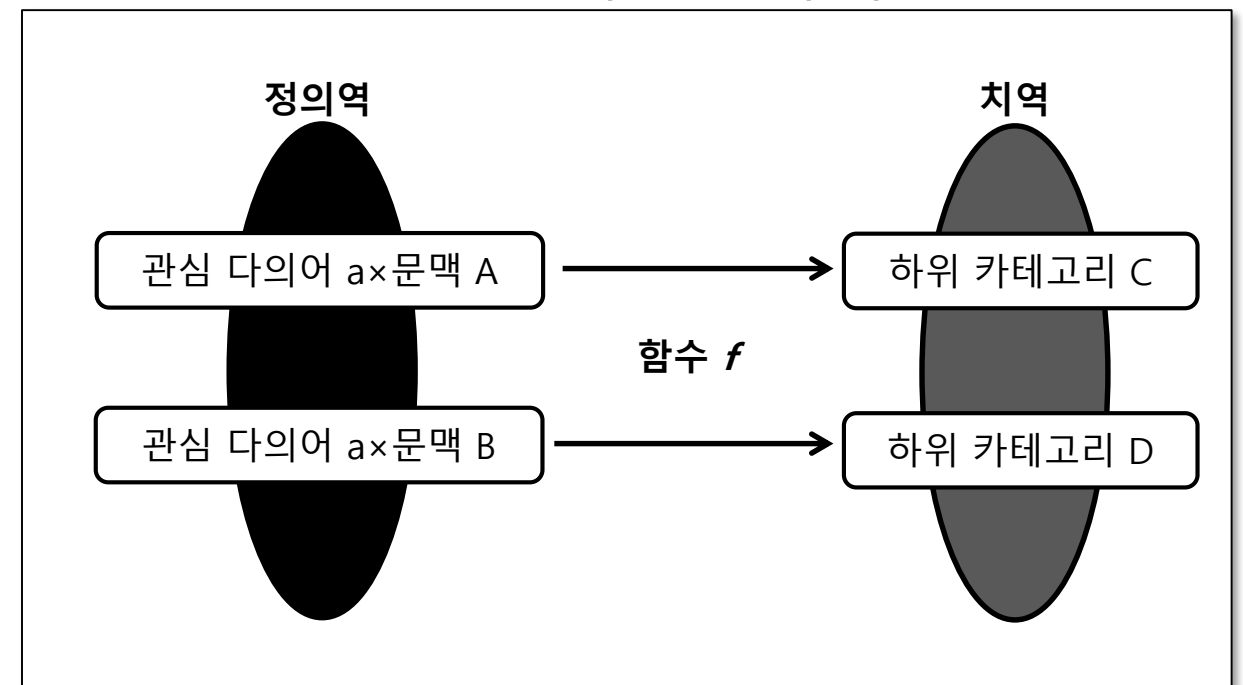
## ■ 목표 모델링

- 정의역: 관심 다의어 집합  $\times$  문맥집합
- 치역: 소셜 맵의 하위 카테고리 집합
- 함수 (분류기)  $f$ : 관심다의어 $\times$ 문맥  $\rightarrow$  하위 카테고리

<그림 1. ODPia 소셜 맵>



<그림 2. 목표 모델링 개념화>



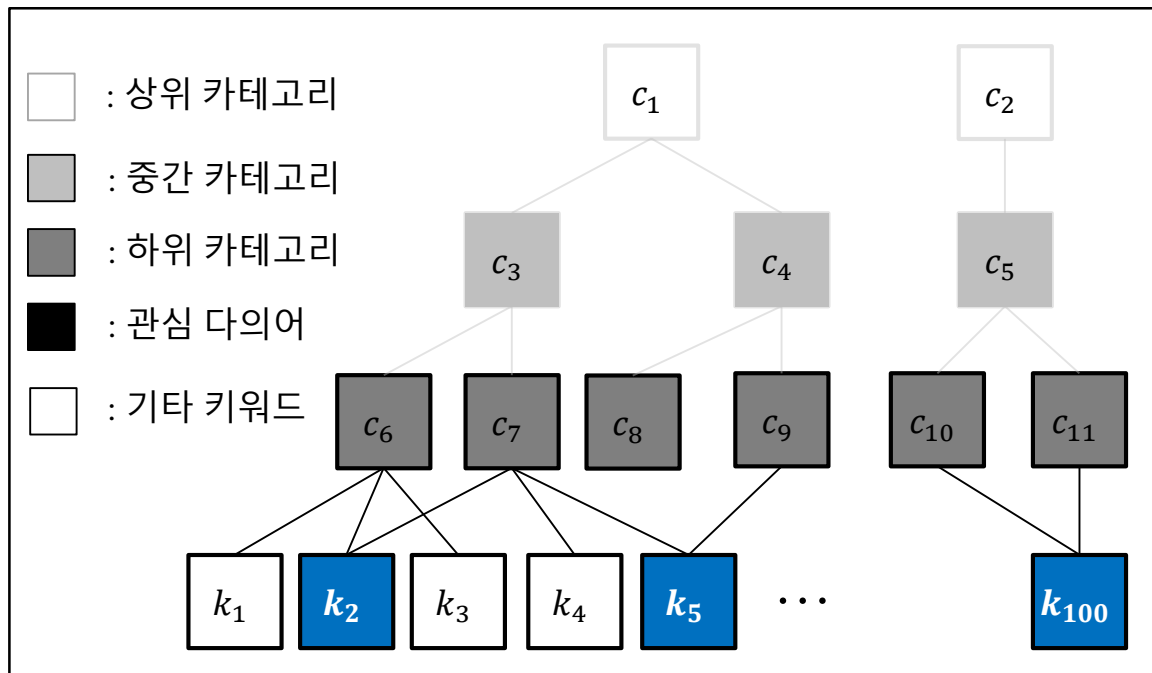
## 2. 상세 목표 및 전략

### • 상세 목표

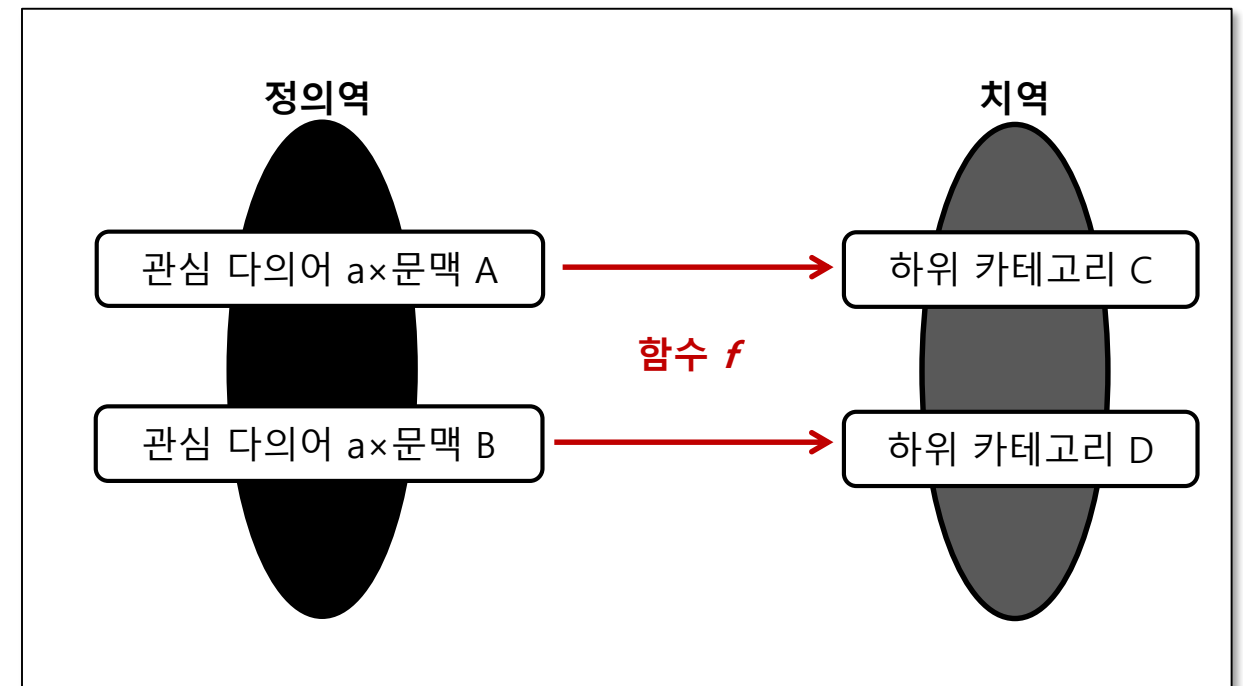
1. 높은 **분류 정확도**

2. 넓은 **적용 범위**

<그림 1. ODPia 소셜 맵>



<그림 2. 목표 모델링 개념화>



## 2. 상세 목표 및 전략

- 상세 목표 달성을 위한 전략

### 1. 높은 분류 정확도

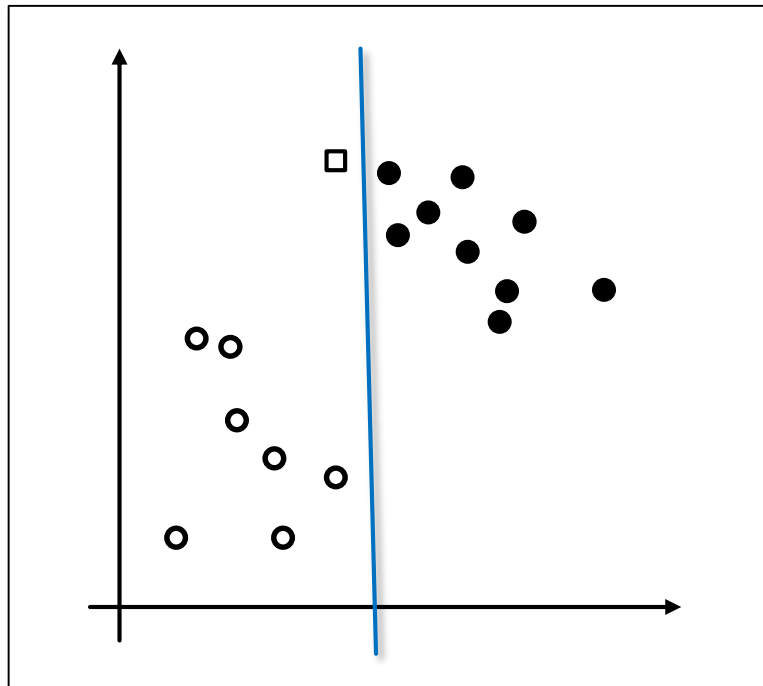
<그림 3. SVM 특징점 및 도입 사유>

#### - SVM (Support Vector Machine)

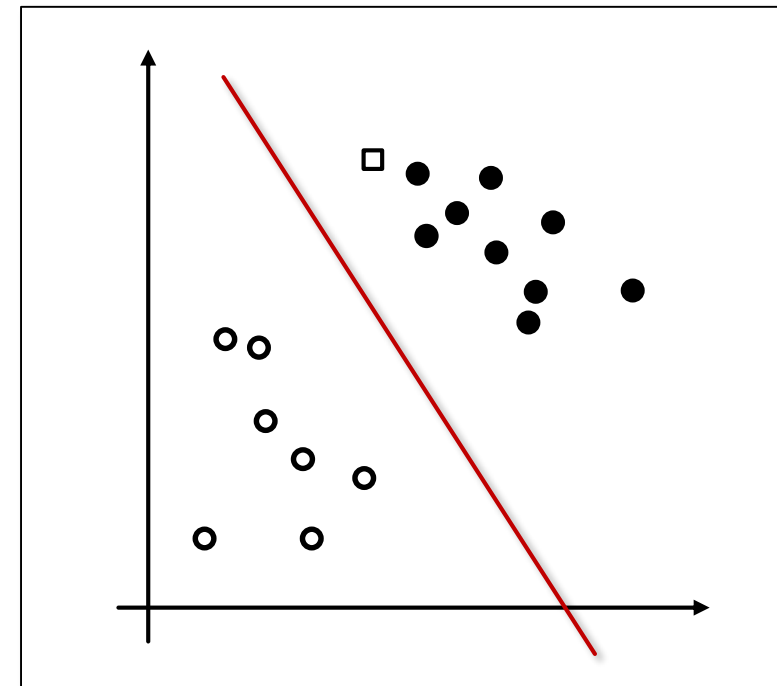
○: 학습 데이터, 부류 1

●: 학습 데이터, 부류 2

□: 테스트 데이터



SVM 이외의 분류기



SVM 기반 분류기

## 2. 상세 목표 및 전략

- 상세 목표 달성을 위한 전략

### 1. 높은 분류 정확도

<그림 3. 정의역 벡터화 도식화>

- SVM (Support Vector Machine)

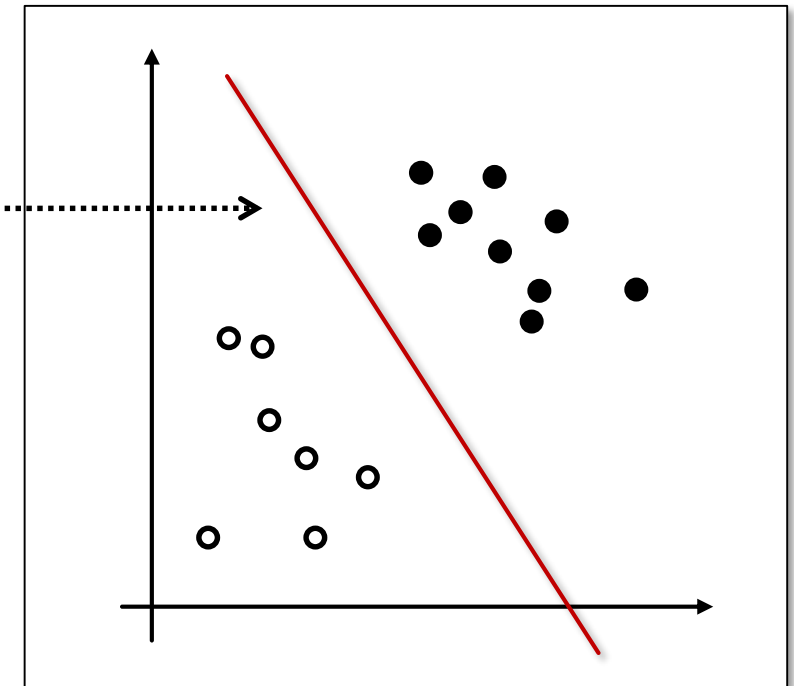
정의역  
- 정의역 벡터화

관심 다의어 a×문맥 A

관심 다의어 a×문맥 B

...

일반 문장



SVM 기반 분류기

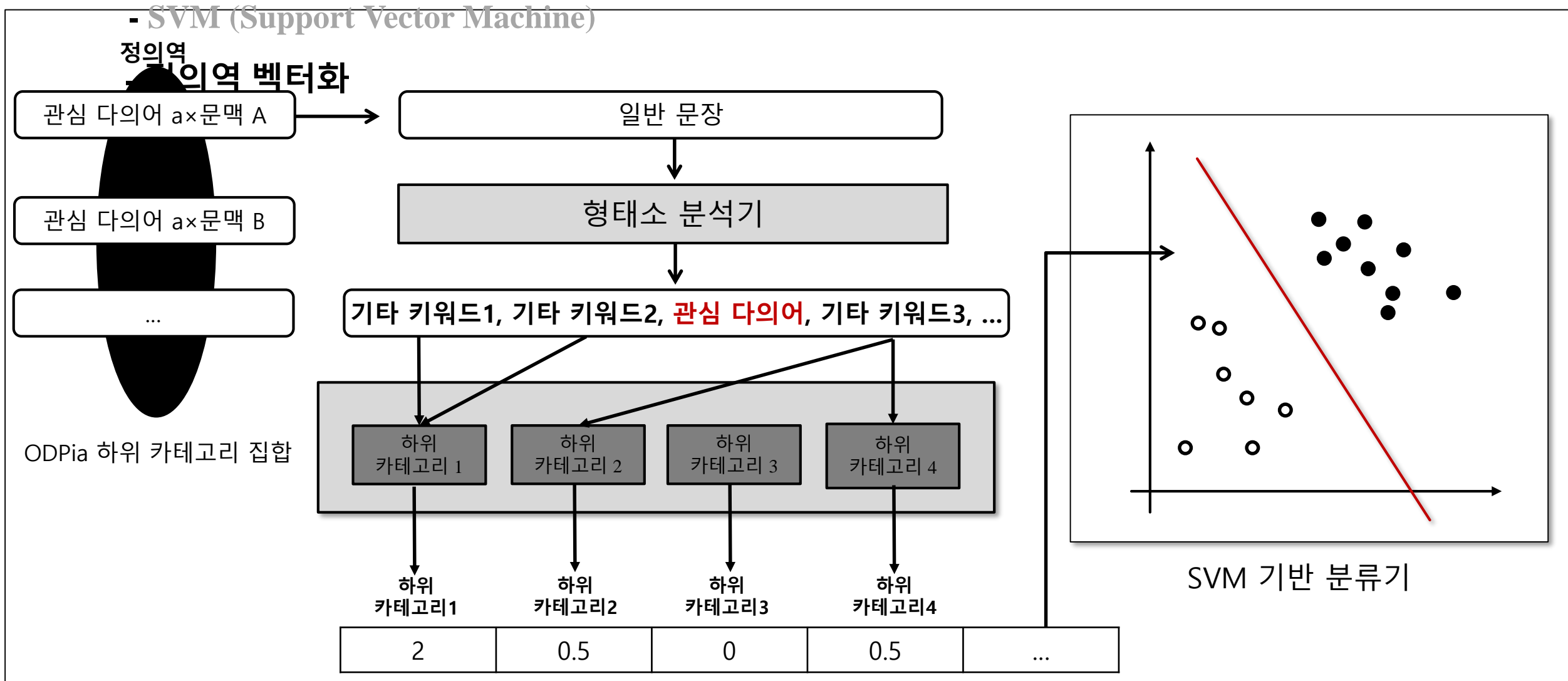


## 2. 상세 목표 및 전략

- 상세 목표 달성을 위한 전략

### 1. 높은 분류 정확도

<그림 3. 정의역 벡터화 도식화>



## 2. 상세 목표 및 전략

- 상세 목표 달성을 위한 전략
- 2. 넓은 **적용 범위**: ODPia 소셜 맵 토폴로지 정보 활용
  - 토폴로지 정보
    - ODPia 소셜 맵 내, 다의어 수: 5,610
    - ODPia 소셜 맵 내, 다의어의 평균 차수: 2.4
    - ODPia 소셜 맵 내, 다의어의 차수 범위: 2-9
- 전략: **적은 자원** 소모 → **적용 범위 확장**
  1. 차수 최대 다의어 기반 (Top-down)
    - 차수 최대 다의어: 이태영(법조인, 정치인, 학술인/교수, 문화예술인, 스포츠인, 국내인물, 공무원/교육인, 국영/민영 기업인, 대중문화연예인)
  2. 다의어,카테고리 다빈도 조합 기반 (Bottom-up)
    - 대중문화연예인, 문화예술인(1016)
    - 공무원/교육인, 학술인/교수(585)
    - 대중문화연예인, 스포츠인(250)
    - 공무원/교육인, 문화예술인, 학술인/교수(219)
    - ...

## 2. 상세 목표 및 전략

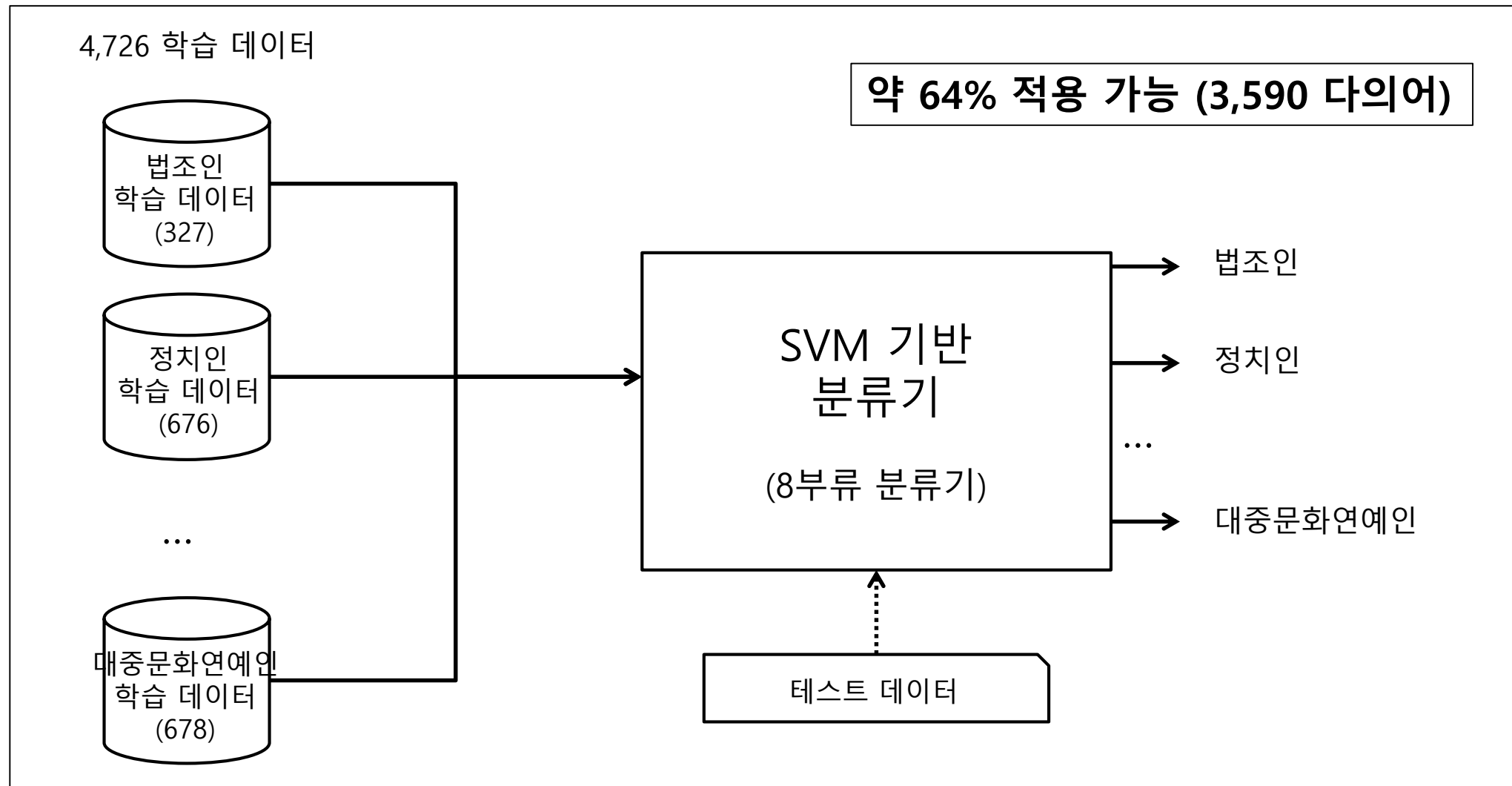
상세 목표 달성을 위한 전략

2. 넓은 적용 범위: ODPia 소셜 맵 토폴로지 정보 활용

1. 차수 최대 다의어 기반 (Top-down)

- 차수 최대 다의어: 이태영(법조인, 정치인, 예술인/교수, 문화예술인, 스포츠인, 국내인물, 공무원/교직원, 국경/민영 기업인, 대중문화연예인)

<그림 4. 차수 최대 다의어 기반 8부류 분류기>

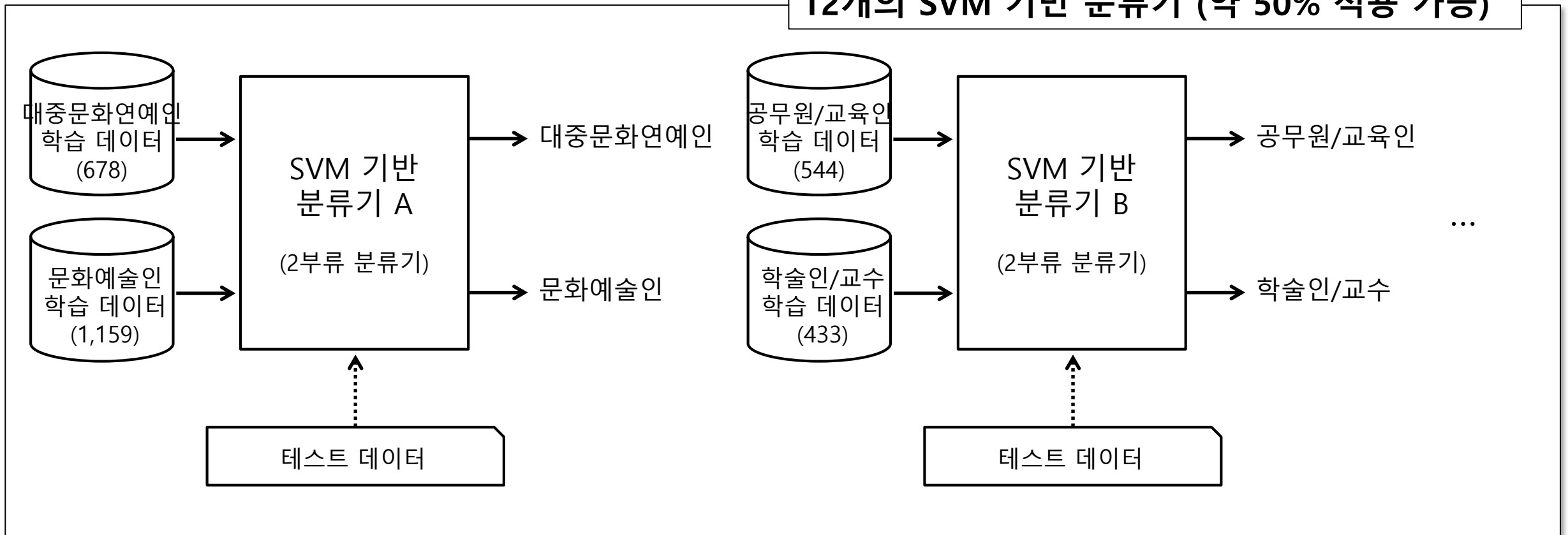


## 2. 상세 목표 및 전략

- 상세 목표 달성을 위한 전략
  - 2. 넓은 적용 범위: ODPia 소셜 맵 토폴로지 정보 활용
    - 1. 다국어, 카테고리 다빈도 조합 기반 (Bottom-up)
      - 대중문화연예인, 문화예술인(1016)
      - 공무원/교육인, 학술인/교수(585)
      - 대중문화연예인, 스포츠인(250)
      - 공무원/교육인, 문화예술인, 학술인/교수(219)
      - ...

<그림 5. 다국어-카테고리 다빈도 조합 기반, 다수 SVM 분류기>

12개의 SVM 기반 분류기 (약 50% 적용 가능)



3. 성능 평가

• Top-down 기반, 8부류 SVM 분류기 (10-fold cross validation)

<표 1. Top-down 기반 Confusion matrix>

	법조인	공무원 /교육인	학술인/교수	대중문화 연예인	문화예술인	스포츠인	정치인	국영/민영 기업인
법조인	321	0	1	0	0	0	0	5
공무원/교육인	22	441	41	16	3	0	6	15
학술인/교수	6	16	394	4	3	0	2	8
대중문화연예인	3	156	15	367	95	0	25	17
문화예술인	10	31	29	72	949	0	21	47
스포츠인	0	0	0	0	0	457	0	0
정치인	20	57	14	15	11	83	462	14
국영/민영 기업인	14	91	5	9	21	2	3	307

평균 정확도: 80.3%

3. 성능 평가

• Bottom-up 기반, 12개의 SVM 분류기 (10-fold cross validation)

<표2. Bottom-up 기반 Confusion matrix 집합>

	대중문화 연예인	문화예술인
대중문화 연예인	673	5
문화예술인	67	1092

	공무원 /교육인	학술인/교수
공무원/교육인	537	7
학술인/교수	0	433

	대중문화 연예인	스포츠인
대중문화 연예인	677	1
스포츠인	0	457

	공무원 /교육인	학술인/교수	문화예술인
공무원 /교육인	536	5	3
학술인/교수	0	433	0
문화예술인	17	33	1109

	대중문화 연예인	문화예술인	스포츠인
대중문화 연예인	671	7	0
문화예술인	73	1086	0
스포츠인	0	0	457

	공무원/교육 인	정치인
공무원/교육 인	540	4
정치인	44	632

3. 성능 평가

• Bottom-up 기반, 12개의 SVM 분류기 (10-fold cross validation)

<표2. Bottom-up 기반 Confusion matrix 집합>

	공무원 /교육인	학술인 /교수	정치인
공무원 /교육인	533	8	3
학술인 /교수	0	433	0
정치인	34	8	634

	법조인	공무원 /교육인
법조인	194	133
공무원 /교육인	46	498

	공무원 /교육인	학술인/교 수	스포츠인
공무원 /교육인	534	10	0
학술인 /교수	0	433	0
스포츠인	0	0	457

	대중문화 연예인	문화예술인	국영/민영 기업인
대중문화 연예인	675	1	2
문화예술인	43	1111	5
국영/민영 기업인	3	2	447

	공무원 /교육인	학술인 /교수	대중문화 연예인	문화예술 인
공무원 /교육인	534	5	4	1
학술인 /교수	0	433	0	0
대중문화 연예인	14	1	661	2
문화예술 인	19	36	60	1044

	아나운서/ 방송제작자	스포츠인
아나운서/ 방송제작자	562	1
스포츠인	0	457

평균 정확도: 96.5%

## 4. 제한점 및 향후 계획

- 제한점
  - - 학습 데이터
    - - 학습 데이터 볼륨 불충분
    - - 학습 데이터의 대표성 검증 미흡
  - - 적용 범위
    - - 50% ~ 60% 대의 적용 범위
- Lessons
  - - 단일 SMART 분류기 vs. 다수 단순 분류기
- 향후 계획
  - - 학습 데이터 질/양 향상
  - - 적용 범위 확대
  - - SVM 이외의 분류 방법론 도입 (ANN, 앙상블 기법 등)



**감사합니다**

Thank you