5강. Classification & Clustering

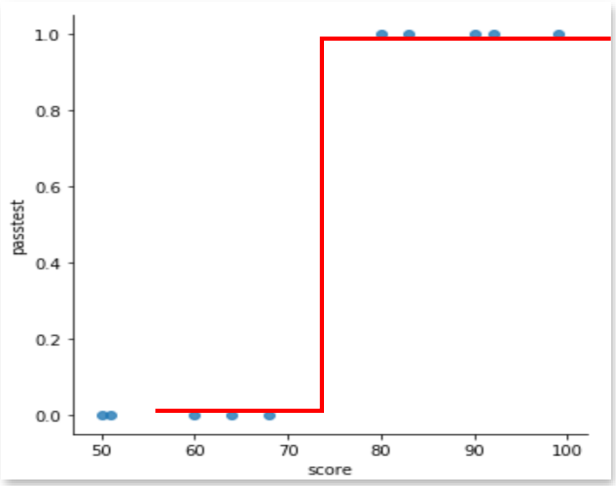
2023.01.05.

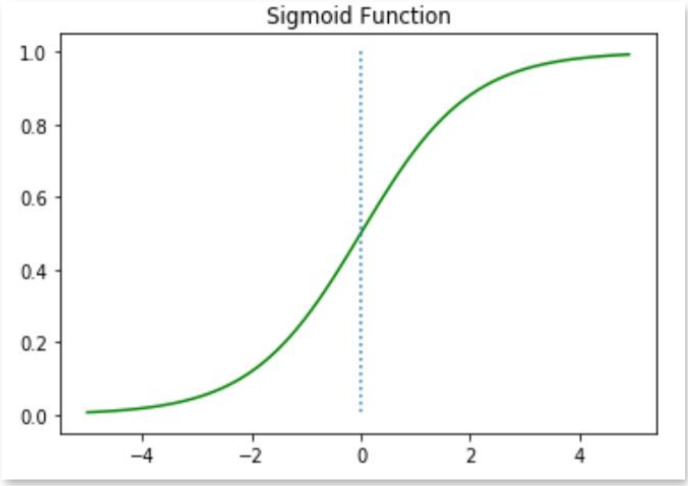
양희철
hcyang@cnu.ac.kr

Classification

Logistic regression

- Logistic regression for classification
  - 종속변수가 연속값이 아니라 비연속값이면 linear regression이 적합하지 않으며, 이 경우 logistic regression을 사용
  - 예시: 학생들의 시험 성적에 따른 합격 평가 데이터(x: 시험 성적, y: 결과[합격(1) 또는 불합격(0)])





Classification

Logistic regression

- Logistic regression for classification
  - Logistic regression models the input-output by a conditional Bernoulli distribution.
  - Binary output $y_n \in \{0,1\}$,

$$\mathbb{E}[y_n|\boldsymbol{x_n}] = \mathbb{P}(y_n = 1|\boldsymbol{x_n}) = \sigma(\boldsymbol{w}^T\boldsymbol{x_n}),$$

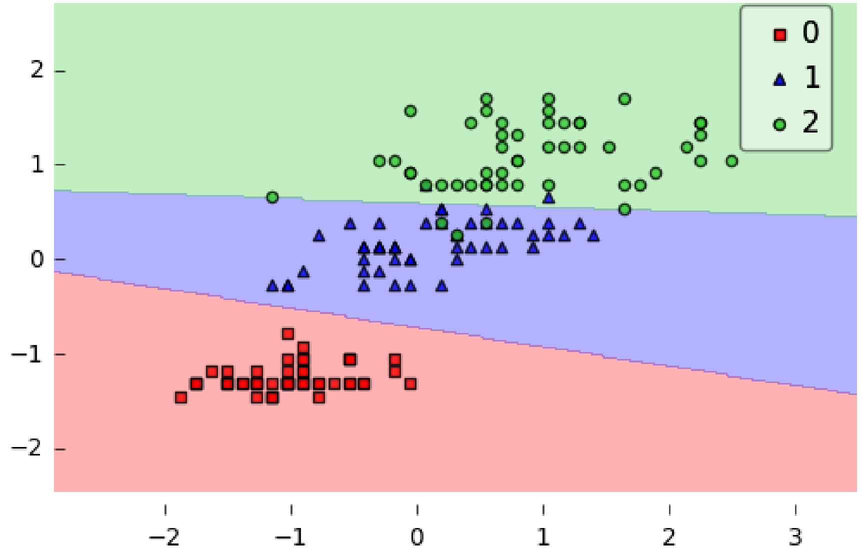
where $\sigma(\xi) = \frac{1}{1+e^{-\xi}}$

Classification

Softmax regression

- Softmax regression for multi-class classification

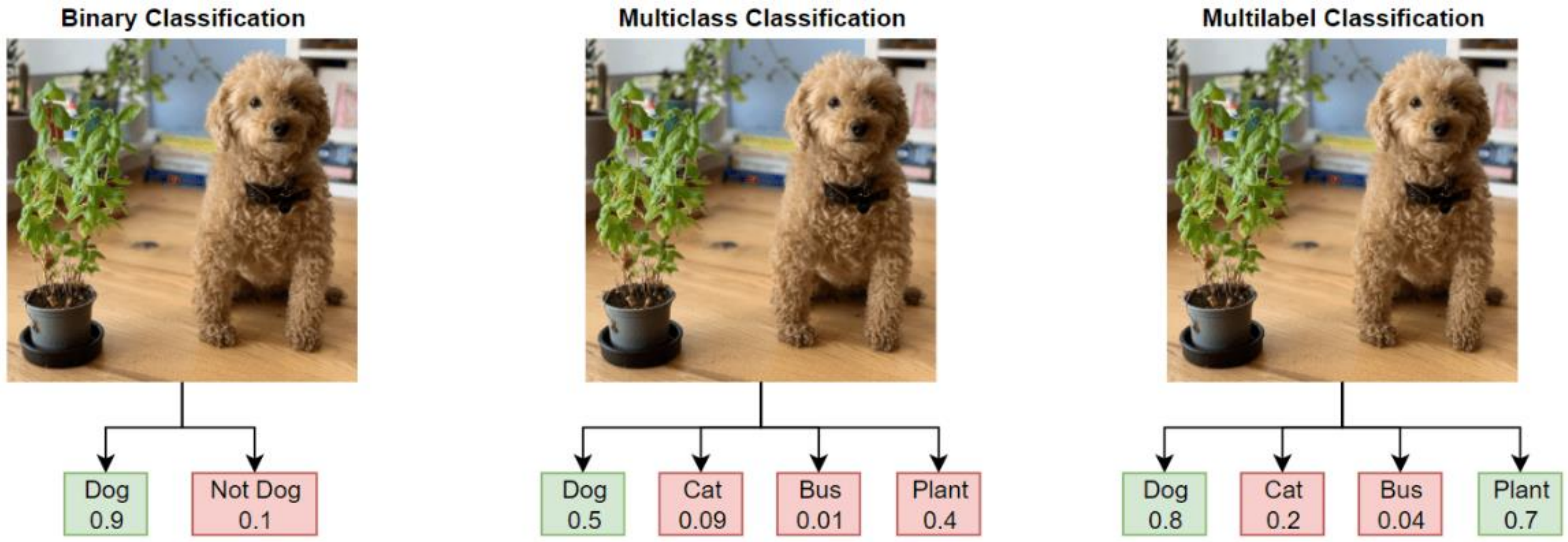
$$p(y_n = k|x_n) = \mathrm{softmax}(\boldsymbol{w}_{\boldsymbol{k}}^T \boldsymbol{x}_{\boldsymbol{n}}) = \frac{\exp(\boldsymbol{w}_{\boldsymbol{k}}^T \boldsymbol{x}_{\boldsymbol{n}})}{\sum_{j=1}^{K} \exp(\boldsymbol{w}_{\boldsymbol{j}}^T \boldsymbol{x}_{\boldsymbol{n}})}$$



Classification

Softmax regression

- Multi-label classification



Classification

Cross entropy

- Cross entropy between two probability distribution P and Q over the same underlying set of events measures the average number of bits needed to identify an event drawn from the set, if a coding scheme is used that is optimized for an unnatural probability distribution Q, rather than the true distribution P.

$$H(P,Q) = -\sum_{i} P(x_i) \log Q(x_i)$$

Classification

Cross entropy

- Logistic regression
  - For $p \in \{y, 1-y\}, q \in \{\hat{y}, 1-\hat{y}\}$,

$$\mathcal{E} = \sum_{n=1}^{N} \left[-y_n \log \hat{y}_n - (1-y_n)\log(1-\hat{y}_n)\right]$$

- Softmax regression
  - For $p \in \{y_1, y_2, \dots, y_K\}, q \in \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_K\}$,

$$\mathcal{E} = \sum_{n=1}^{N} \sum_{k=1}^{K} \left[-y_{k,n} \log \hat{y}_{k,n}\right]$$

Classification

Cross entropy

- Example
  - For $\hat{y} = [1\ 0\ 0]$ and $\hat{y} = [0.7\ 0.2\ 0.1]$,

$$\begin{aligned}\mathcal{E} &= -1 \log 0.7 - 0 \log 0.2 - 0 \log 0.1 \\ &= 0.36\end{aligned}$$

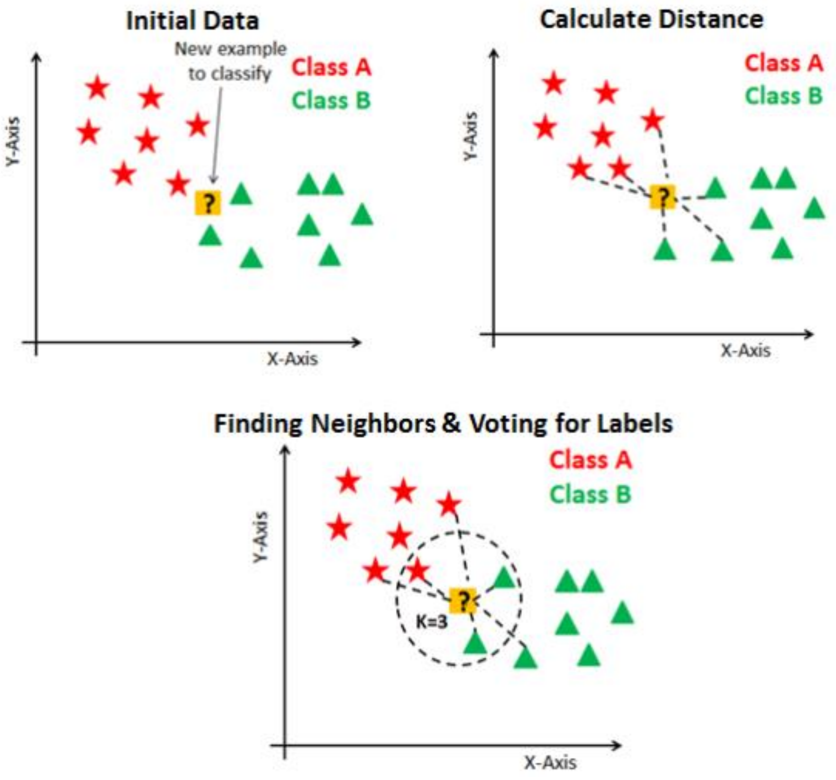
  - For $\hat{y} = [1\ 0\ 0]$ and $\hat{y} = [0.5\ 0.3\ 0.2]$,

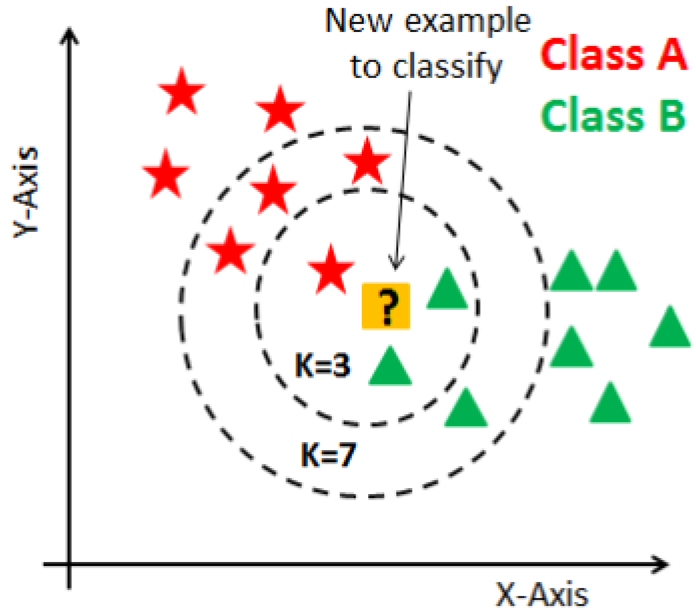
$$\begin{aligned}\mathcal{E} &= -1 \log 0.5 - 0 \log 0.3 - 0 \log 0.2 \\ &= 0.69\end{aligned}$$

Classification

K-nearest neighbor classification

- KNN classification algorithm

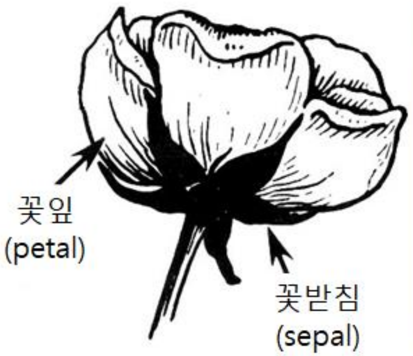




Classification

K-nearest neighbor classification

- Practice



붓꽃 데이터 꽃잎, 꽃받침의 너비와 높이 정보가 있는 데이터가 제공된다.

```python
from sklearn.datasets import load_iris
iris = load_iris()
print(iris.data)
```

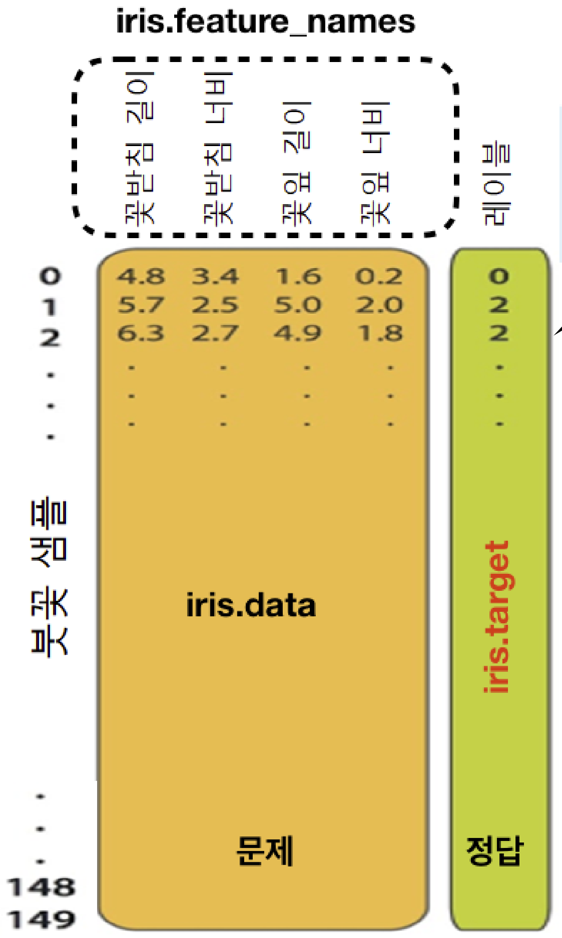
```
array([[5.1, 3.5, 1.4, 0.2],
[4.9, 3. , 1.4, 0.2],
[4.7, 3.2, 1.3, 0.2],
...
```

Classification

K-nearest neighbor classification

- Practice

```
>>> iris.data.shape
(150, 4)
```



Setosa : 0
Versicolor : 1
Virginica : 2 로 레이블 되어 있다.

Classification

K-nearest neighbor classification

- Practice

sepal : 꽃받침
petal : 꽃잎

```
print(iris.feature_names) # 4개의 특징 이름을 출력한다.
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

레이블이 0, 1, 2로 인코딩되어 있는 것도 확인할 수 있다

```
# 정수는 꽃의 종류를 나타낸다: 0 = setosa, 1=versicolor, 2=virginica
print(iris.target)
```

```
[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 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 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]
```

Classification

K-nearest neighbor classification

- Practice

```
# (80:20)으로 분할한다.
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

iris = load_iris()
X_train,X_test,y_train,y_test = train_test_split(iris.data, iris.target,test_size=0.2)
```

Classification

K-nearest neighbor classification

- Practice

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

num_neigh = 1
knn = KNeighborsClassifier(n_neighbors = num_neigh)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
scores = metrics.accuracy_score(y_test, y_pred)
print('n_neighbors가 {0:d}일때 정확도: {1:.3f}'.format(num_neigh, scores))
```

```
n_neighbors가 1일때 정확도: 0.933
```

Classification

K-nearest neighbor classification

- Practice
  - 새로운 데이터에 대한 예측

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

iris = load_iris()
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(iris.data, iris.target)
```

Classification

K-nearest neighbor classification

- Practice
  - 새로운 데이터에 대한 예측

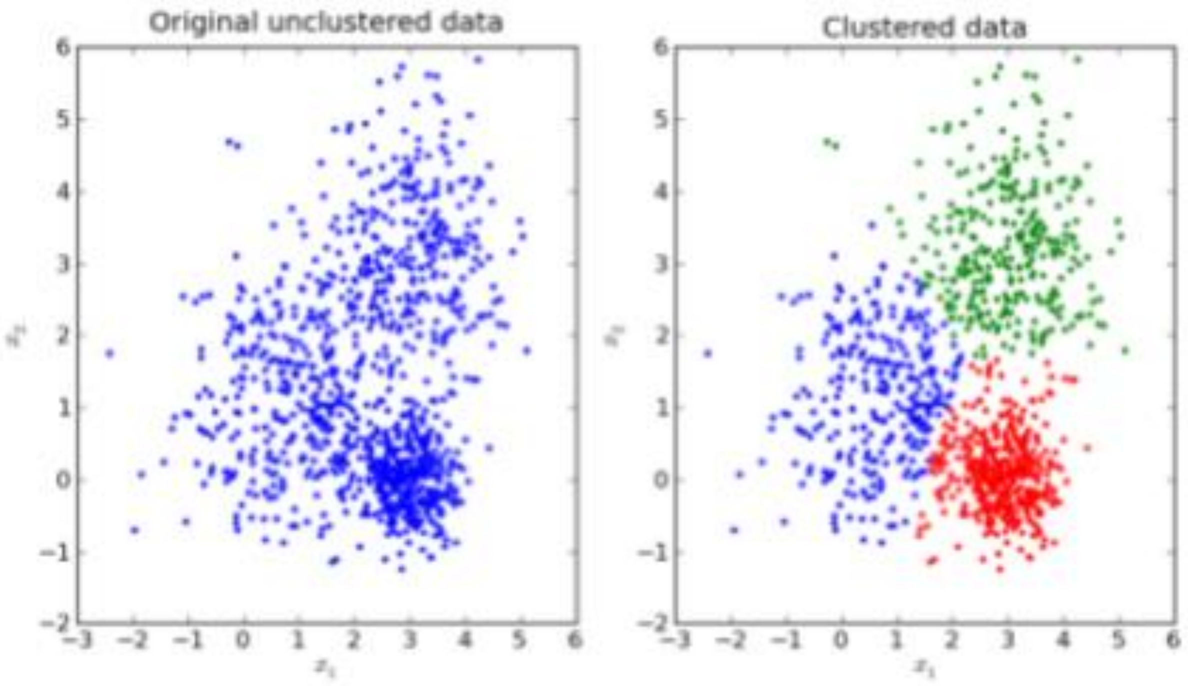
```
classes = {0:'setosa', 1:'versicolor', 2:'virginica'}

# 아직 보지 못한 새로운 데이터를 제시해 보자.
X = [[3,4,5,2],
[5,4,2,2]]
y = knn.predict(X)

print(classes[y[0]])
print(classes[y[1]])
```

```
versicolor
setosa
```

Clustering

Clustering

- 데이터는 존재하지만 데이터의 label이나 category가 주어지지 않은 경우 classification이 아닌 clustering을 통해 데이터들을 설명할 수 있음



Clustering

K-means clustering

- 클러스터 내부에 속한 데이터들이 서로 가장 가까운 내부 거리를 가지도록 분류

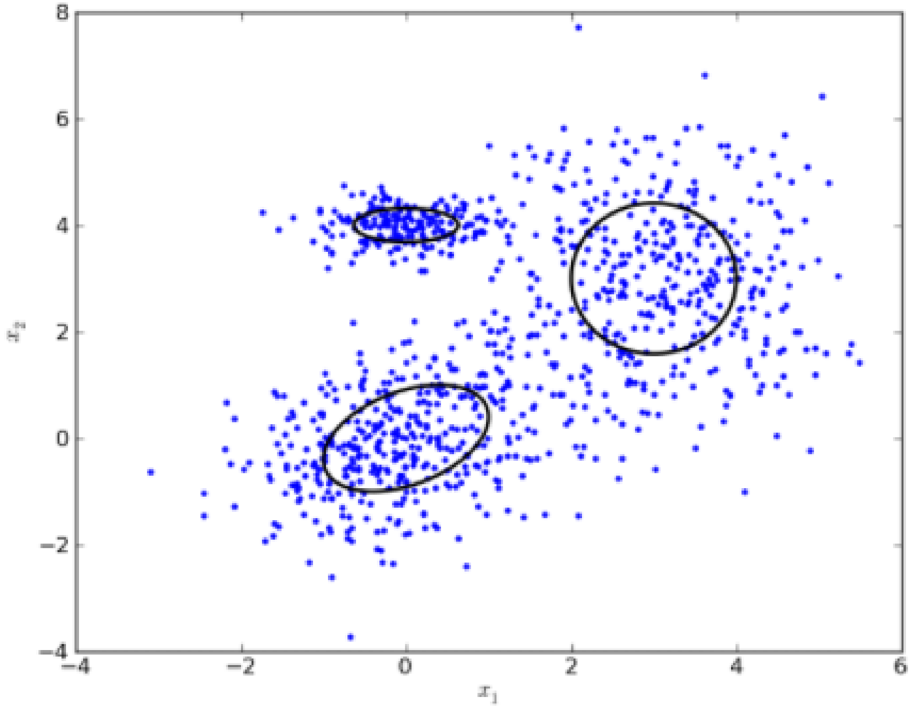
$$\min_{b,w} \sum_{i}^{n} \sum_{j}^{k} w_{ij} \|x_i - b_j\|_2^2 \text{ s.t. } \sum_{j} w_{ij} = 1, \forall j$$

  - n개의 데이터
  - k개의 클러스터
  - b_j: j번째 클러스터의 중심
  - $w_{i,j}$: 데이터 i가 j번째 클러스터에 속하는지 나타내는 변수

Clustering

Gaussian mixture clustering

- Gaussian mixture model: 데이터가 k개의 Gaussian으로 구성되어있다고 할 때, 가장 데이터를 잘 설명하는 k개의 평균과 covariance를 찾는 알고리즘



Clustering

Gaussian mixture clustering

- Gaussian mixture model: 데이터가 k개의 Gaussian으로 구성되어있다고 할 때, 가장 데이터를 잘 설명하는 k개의 평균과 covariance를 찾는 알고리즘
  - Log likelihood function을 performance measure로 가짐

$$\ln p(X|\pi, \mu, \Sigma) = \sum_{i}^{n} \ln \sum_{j}^{k} \pi_j N(x_i|\mu_j, \Sigma_j)$$

  - n개의 데이터
  - k개의 Gaussian
  - μ_j, Σ_j: j번째 Gaussian의 평균과 covariance
  - π_j: 각각의 데이터가 j번째 Gaussian에 속할 확률

Clustering

K-means clustering

- Practice

```python
import numpy as np

# 닥스 훈트의 몸 길이와 몸 높이
dachshund_length = [77, 78, 85, 83, 73, 77, 73, 80]
dachshund_height = [25, 28, 19, 30, 21, 22, 17, 35]

# 사모예드의 몸 길이와 몸 높이
samoyed_length = [75, 77, 86, 86, 79, 83, 83, 88]
samoyed_height = [56, 57, 50, 53, 60, 53, 49, 61]

# 말티즈의 몸 길이와 몸 높이
maltese_length = [34, 38, 38, 41, 30, 37, 41, 35]
maltese_height = [22, 25, 19, 30, 21, 24, 28, 18]

#새로운 데이터
x = [45, 70, 59, 70]
y = [44, 39, 30, 64]

dog_length = dachshund_length + samoyed_length + maltese_length + x
dog_height = dachshund_height + samoyed_height + maltese_height + y

dog_data = np.column_stack((dog_length, dog_height))

print(dog_data)
```

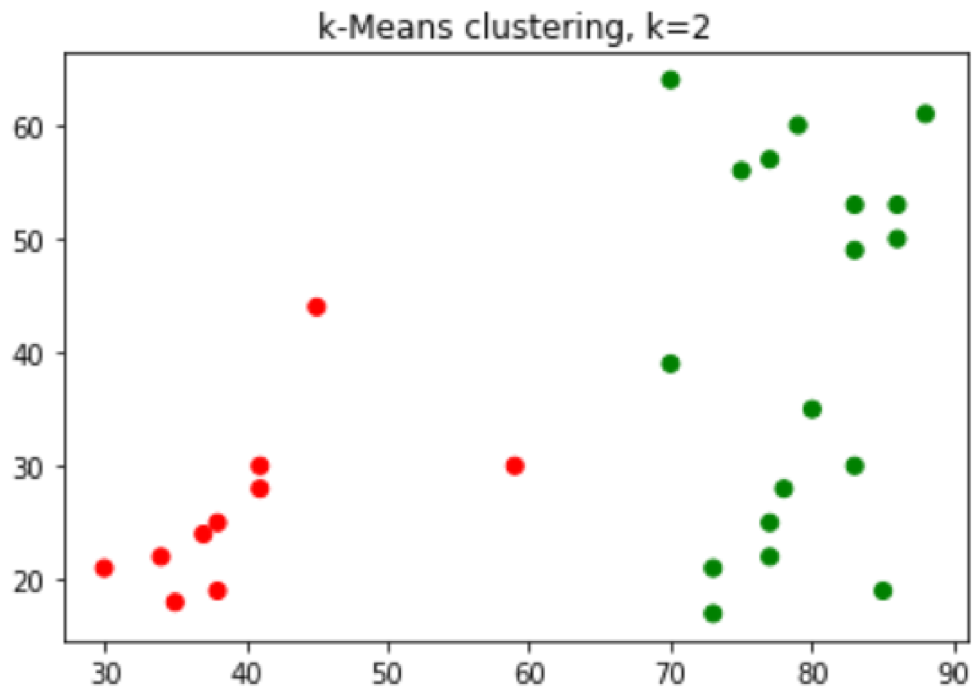
```
[[77 25]
 [78 28]
 [85 19]
 [83 30]
 [73 21]
 [77 22]
 [73 17]
 [80 35]
 [75 56]
 [77 57]
 [86 50]
 [86 53]
 [79 60]
 [83 53]
 [83 49]
 [88 61]
 [34 22]
 [38 25]
 [38 19]
 [41 30]
 [30 21]
 [37 24]
 [41 28]
 [35 18]
 [45 44]
 [70 39]
 [59 30]
 [70 64]]
```

Clustering

K-means clustering

- Practice

```python
from sklearn import cluster
import matplotlib.pyplot as plt

model = cluster.KMeans(n_clusters=2)
model.fit(dog_data)
labels = model.predict(dog_data)
colors = np.array(['red', 'green', 'blue', 'magenta'])
plt.title('k-Means clustering, k=2')
plt.scatter(dog_data[:, 0], dog_data[:, 1], color=colors[labels])
```



Clustering

K-means clustering

- Practice