

機器學習

Lecture 4 Logistic Regression

Logistic Regression

- 二元分類
 - 癌症的診斷：是(1) v.s. 否(0)
 - 購買商品的意願：會購買(1) v.s. 不想購買(0)
- 屬於某類的機率
 - e.x. 根據今天的溫度 ^{X_1} 、濕度 ^{X_2} 、風向 ^{X_3} 來預測明天的降雨(1)的機率

Logistic Regression

- Logistic regression is a learning algorithm used in a supervised learning problem when the output $y \in \{0,1\}$
- Given a feature vector X , the algorithm will evaluate the probability of $y = 1$:

$$\hat{y} = P(y = 1 | x) \in [0,1]$$

複回歸

$$\begin{aligned}h_{\theta}(x_1, x_2, x_3) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= \vec{\theta} \cdot \vec{x}\end{aligned}$$

where

$$\vec{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad \vec{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Logistic Regression

$$\begin{aligned}h_{\theta}(x_1, x_2, x_3) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\&= \vec{\theta} \cdot \vec{x}\end{aligned}$$

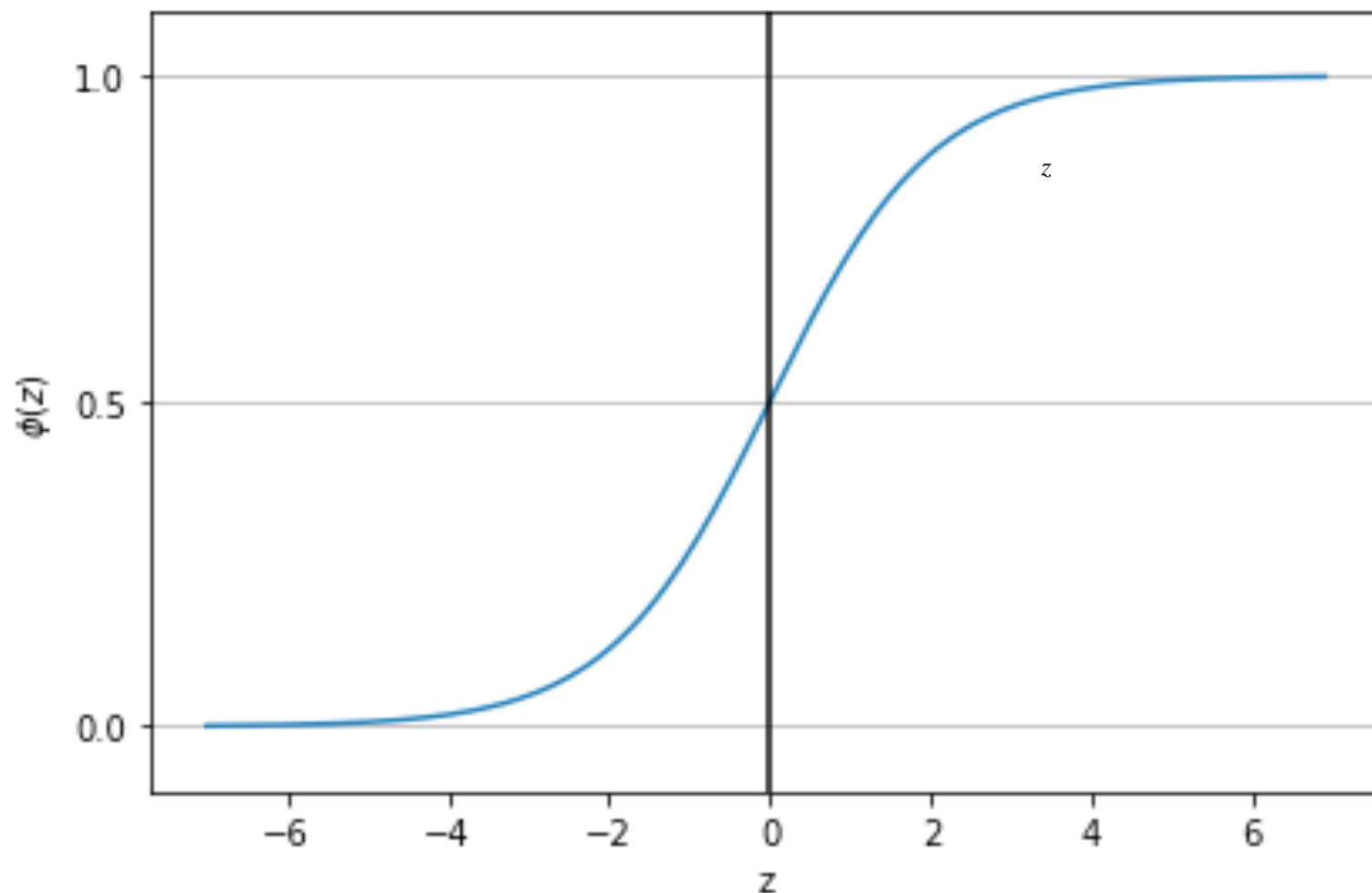
Logistic Regression:

$$\hat{y} = \frac{1}{1 + \exp\left(-\vec{\theta} \cdot \vec{x}\right)}$$

Sigmoid function

$$\phi(z) = \frac{1}{1 + \exp(-z)}$$

$$\exp(-z) = e^{-z}$$

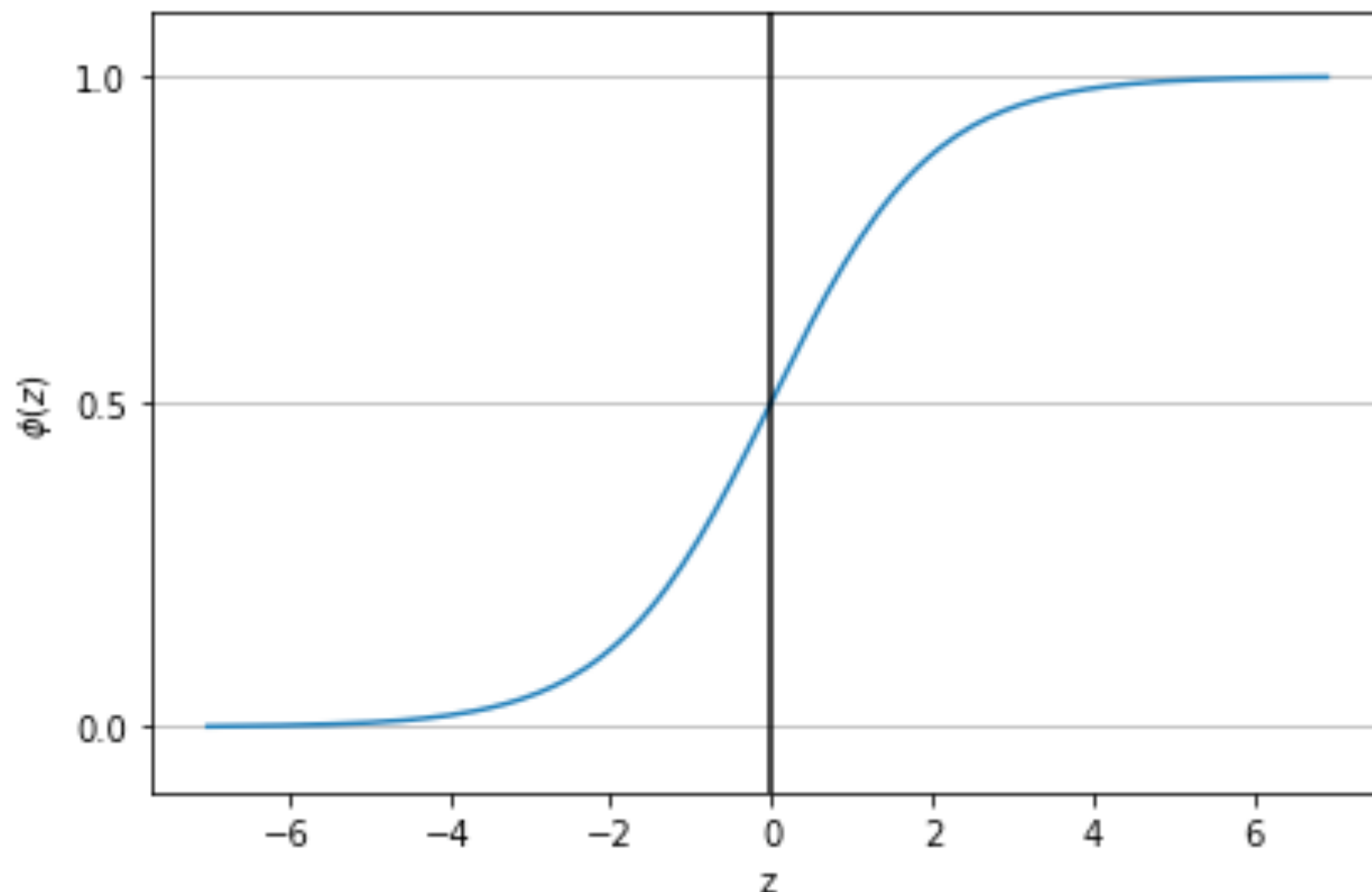


- Some observations
 - If z is a large positive number, then $\phi(z) = 1$
 - If z is small or a large negative number, then $\phi(z) = 0$
 - If $z = 0$, then $\phi(z) = 0.5$

Decision Boundary (決策邊界)

$\hat{y} = P(y = 1 | x) = 0.8$ 降雨機率 = 80% 雨天 (1)

$\hat{y} = P(y = 1 | x) = 0.2$ 降雨機率 = 20% 晴天 (0)



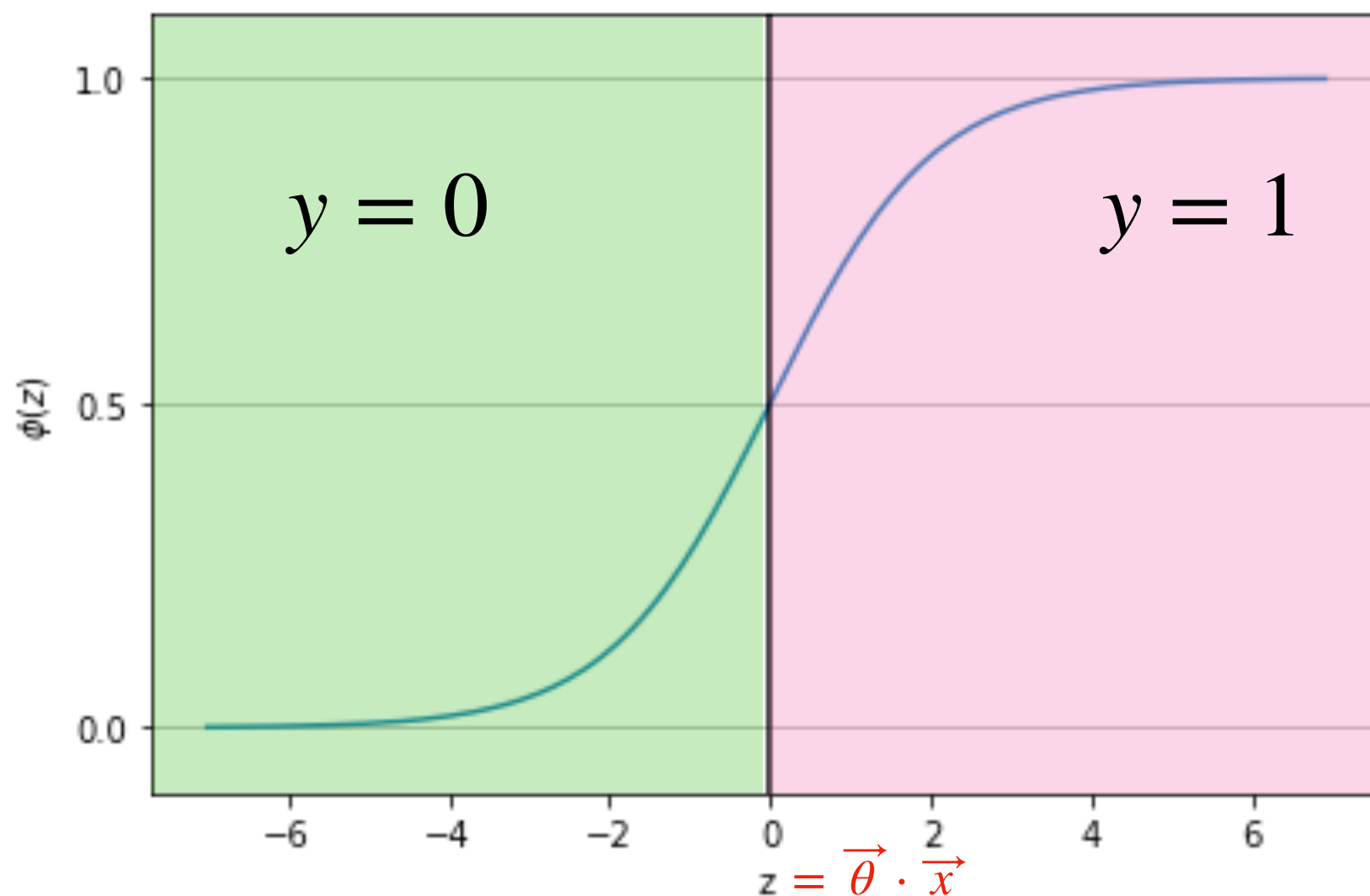
Decision Boundary (決策邊界)

$$\hat{y} = P(y = 1 | x) \geq 0.5 \Rightarrow y = 1$$

$$\hat{y} = P(y = 1 | x) < 0.5 \Rightarrow y = 0$$

$$\vec{\theta} \cdot \vec{x} < 0$$

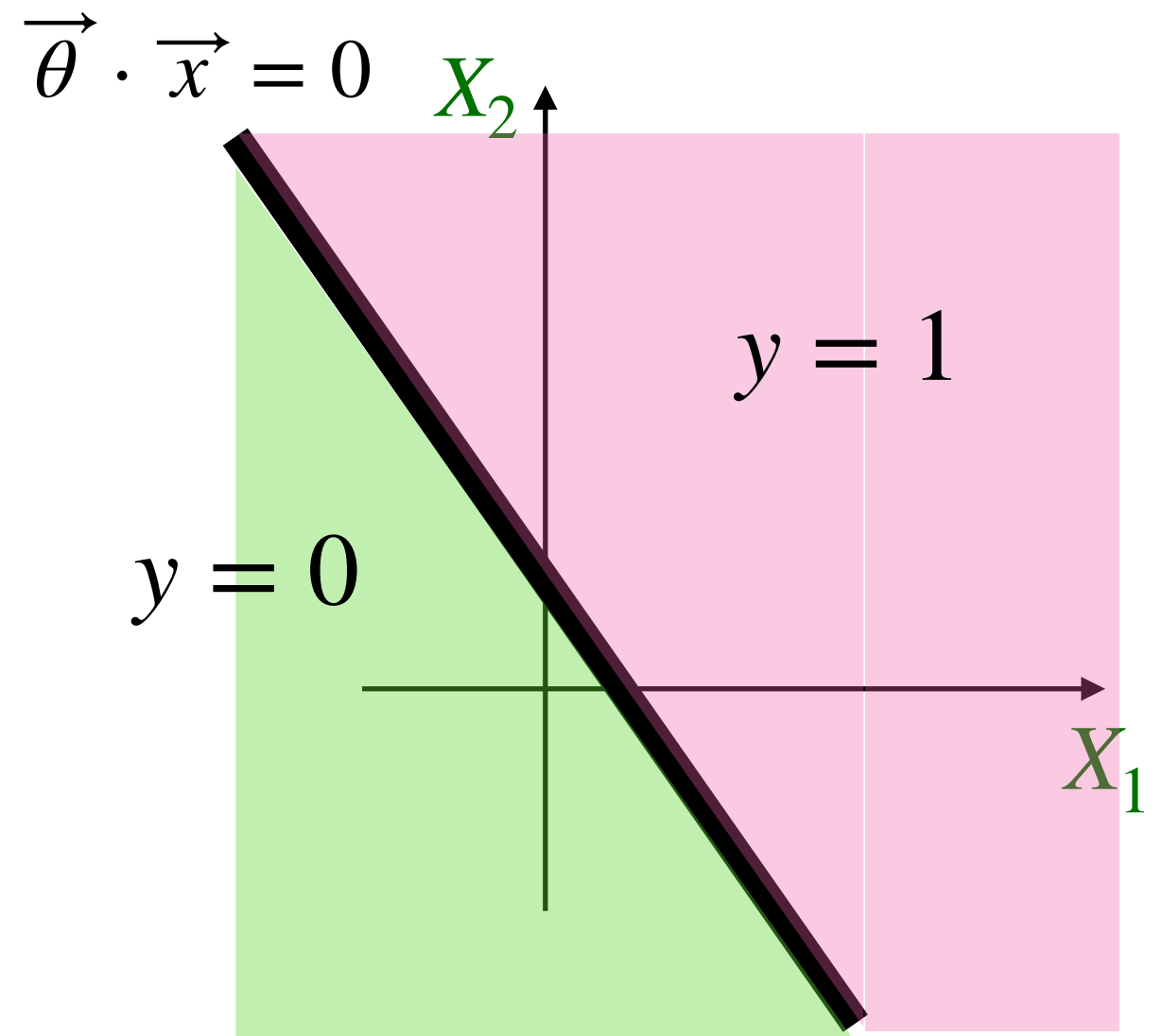
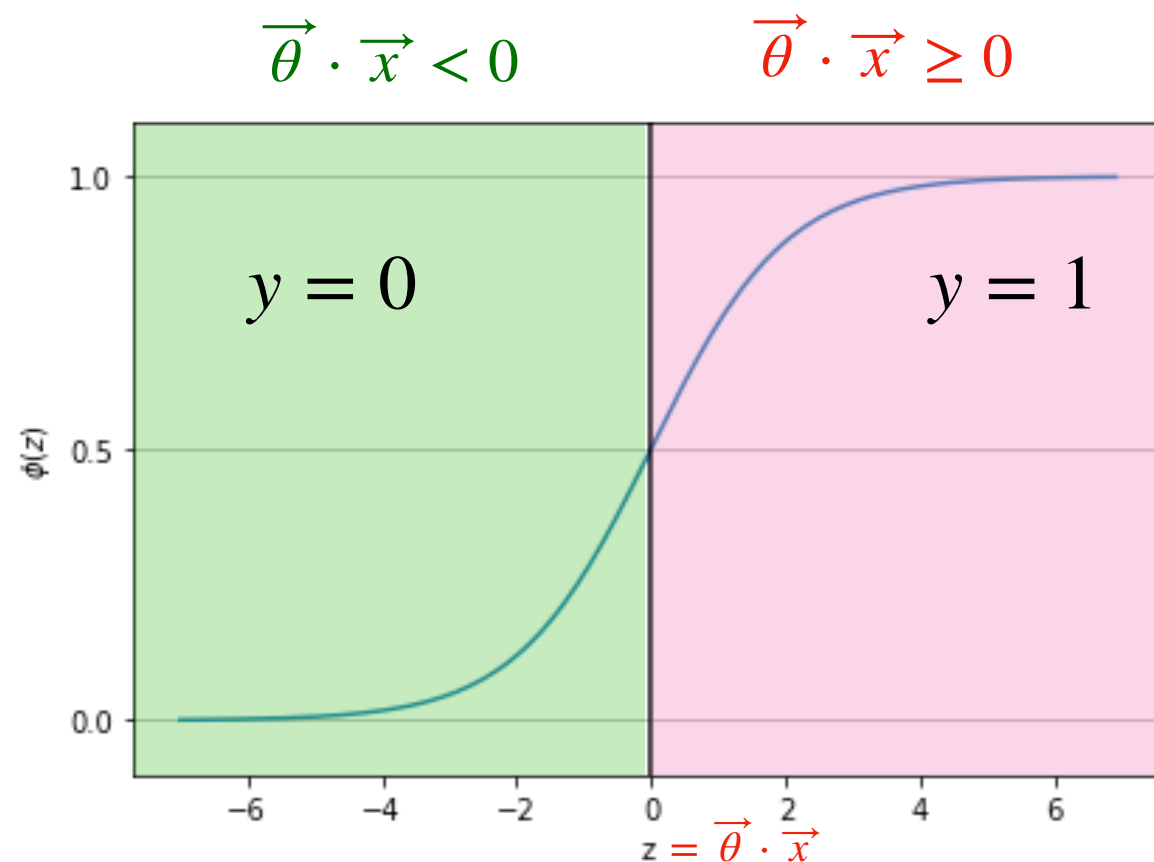
$$\vec{\theta} \cdot \vec{x} \geq 0$$



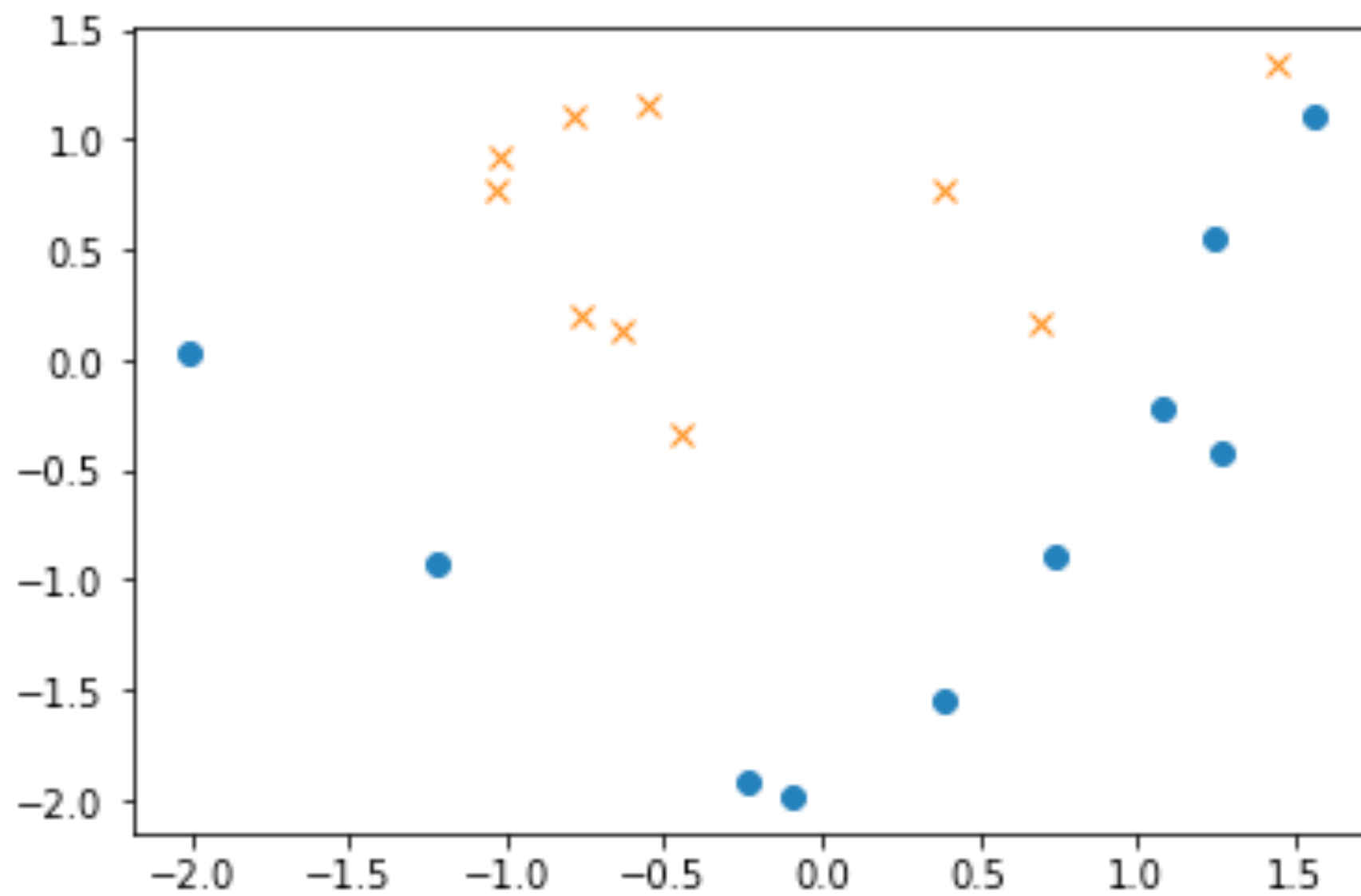
Decision Boundary (決策邊界)

$$\hat{y} = P(y = 1 | x) \geq 0.5 \Rightarrow y = 1$$

$$\hat{y} = P(y = 1 | x) < 0.5 \Rightarrow y = 0$$

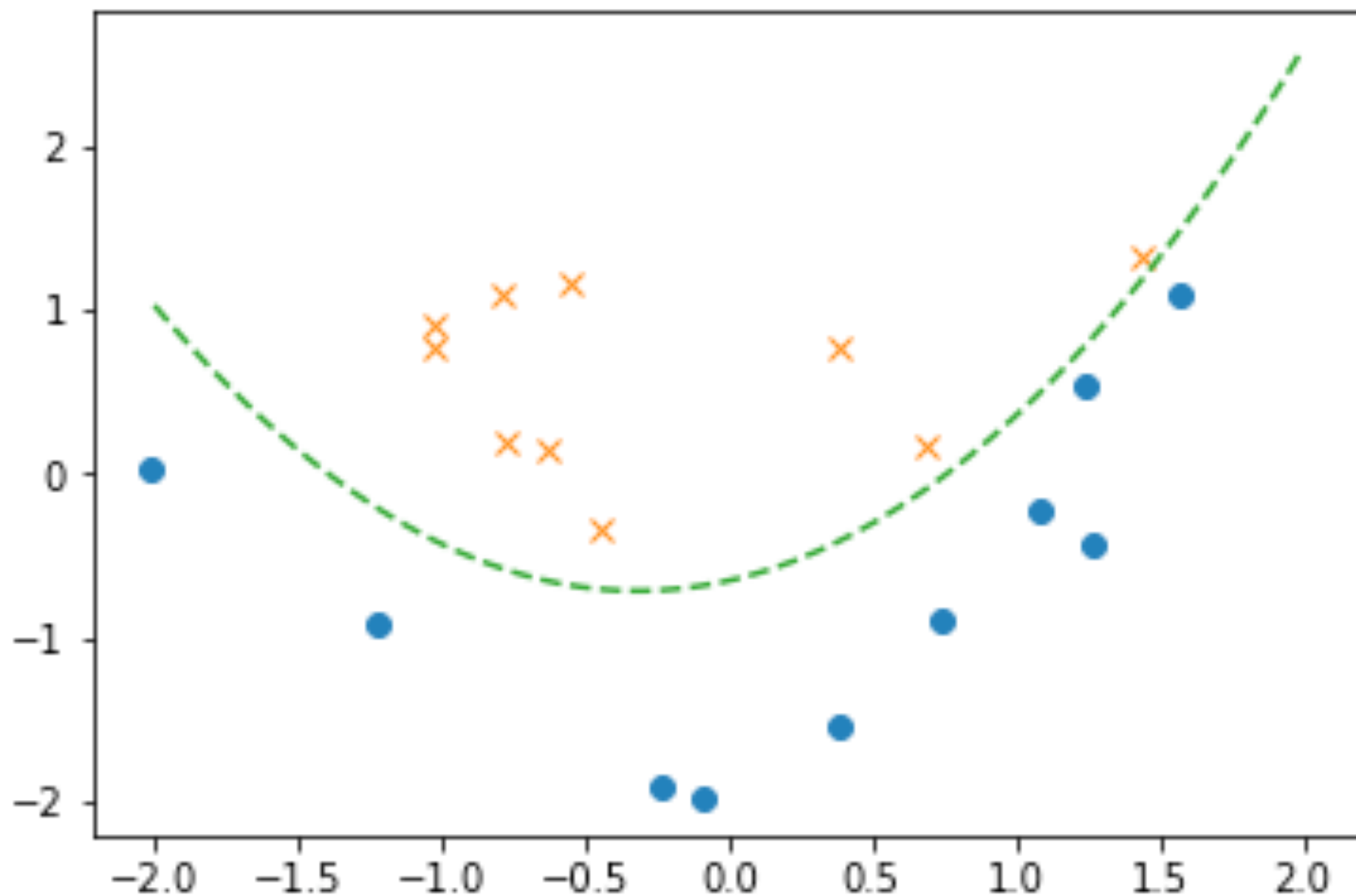


線性不可分離



Decision Boundary (決策邊界)

$$\vec{\theta} \cdot \vec{x} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2$$



Logistic Regression: Cost Function

- 觀察 $\hat{y} = P(y = 1 | x)$
 - 若 $y = 1$: 希望 $P(y = 1 | x)$ 越大越好
 - 若 $y = 0$: 希望 $P(y = 0 | x)$ 越大越好

x	y	機率
(80, 150)	0	希望 $P(y = 0 x)$ 越大越好
(35, 130)	0	希望 $P(y = 0 x)$ 越大越好
(160, 20)	1	希望 $P(y = 1 x)$ 越大越好
(125, 30)	1	希望 $P(y = 1 x)$ 越大越好

Goal: Maximize

$$P(y^{(1)} = 0 | x^{(1)})P(y^{(2)} = 0 | x^{(2)})P(y^{(3)} = 1 | x^{(3)})P(y^{(4)} = 1 | x^{(4)})$$

Logistic Regression: Cost Function

- Goal: Maximize

$$\prod_{i=1}^m P(y^{(i)} = 1 | x^{(1)})^{y^{(i)}} P(y^{(i)} = 0 | x^{(i)})^{1-y^{(i)}}$$

- 觀察

- 若 $y^{(i)} = 1$

$$P(y^{(i)} = 1 | x^{(1)})^{y^{(i)}} P(y^{(i)} = 0 | x^{(i)})^{1-y^{(i)}} = P(y^{(i)} = 1 | x^{(1)})$$

- 若 $y^{(i)} = 0$

$$P(y^{(i)} = 1 | x^{(1)})^{y^{(i)}} P(y^{(i)} = 0 | x^{(i)})^{1-y^{(i)}} = P(y^{(i)} = 0 | x^{(1)})$$

Logistic Regression: log-likelihood Function

- Goal: Maximize

$$\prod_{i=1}^m P(y^{(i)} = 1 | x^{(1)})^{y^{(i)}} P(y^{(i)} = 0 | x^{(i)})^{1-y^{(i)}}$$

- 因為機率越乘越小，所以目標改為 Maximum

$$\log \left(\prod_{i=1}^m P(y^{(i)} = 1 | x^{(1)})^{y^{(i)}} P(y^{(i)} = 0 | x^{(i)})^{1-y^{(i)}} \right)$$

$$= \sum_{i=1}^m y^{(i)} \log (P(y^{(i)} = 1 | x^{(1)})) + (1 - y^{(i)}) \log (P(y^{(i)} = 0 | x^{(i)}))$$

$$= \sum_{i=1}^m y^{(i)} \log (\hat{y}^{(i)}) + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)})$$

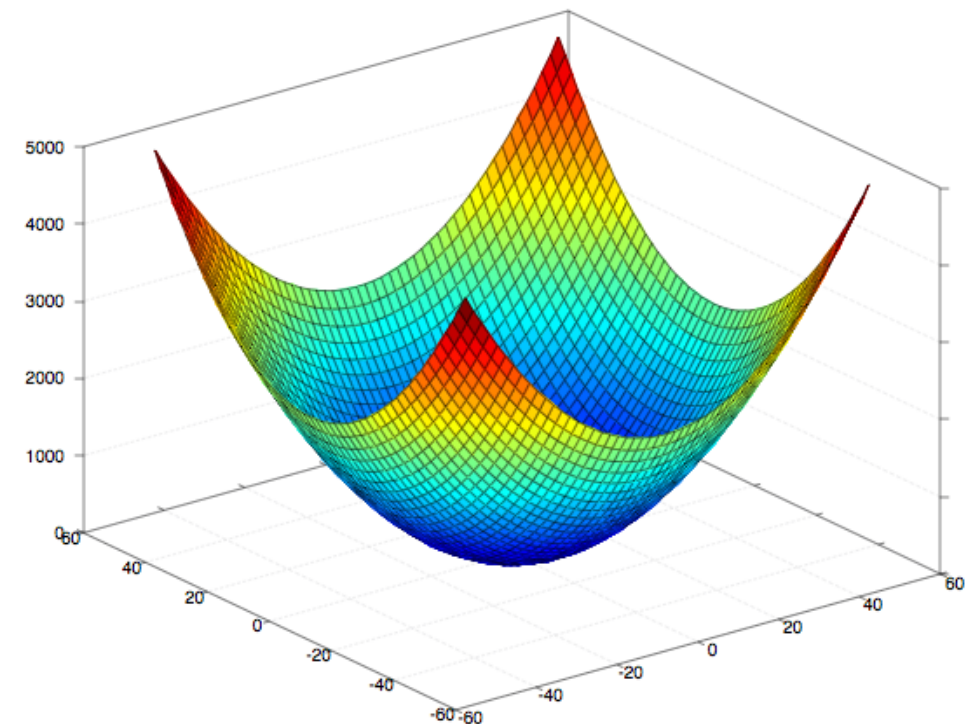
Logistic Regression: log-likelihood Function

- Goal: Maximize

$$\sum_{i=1}^m y^{(i)} \log (\hat{y}^{(i)}) + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)})$$

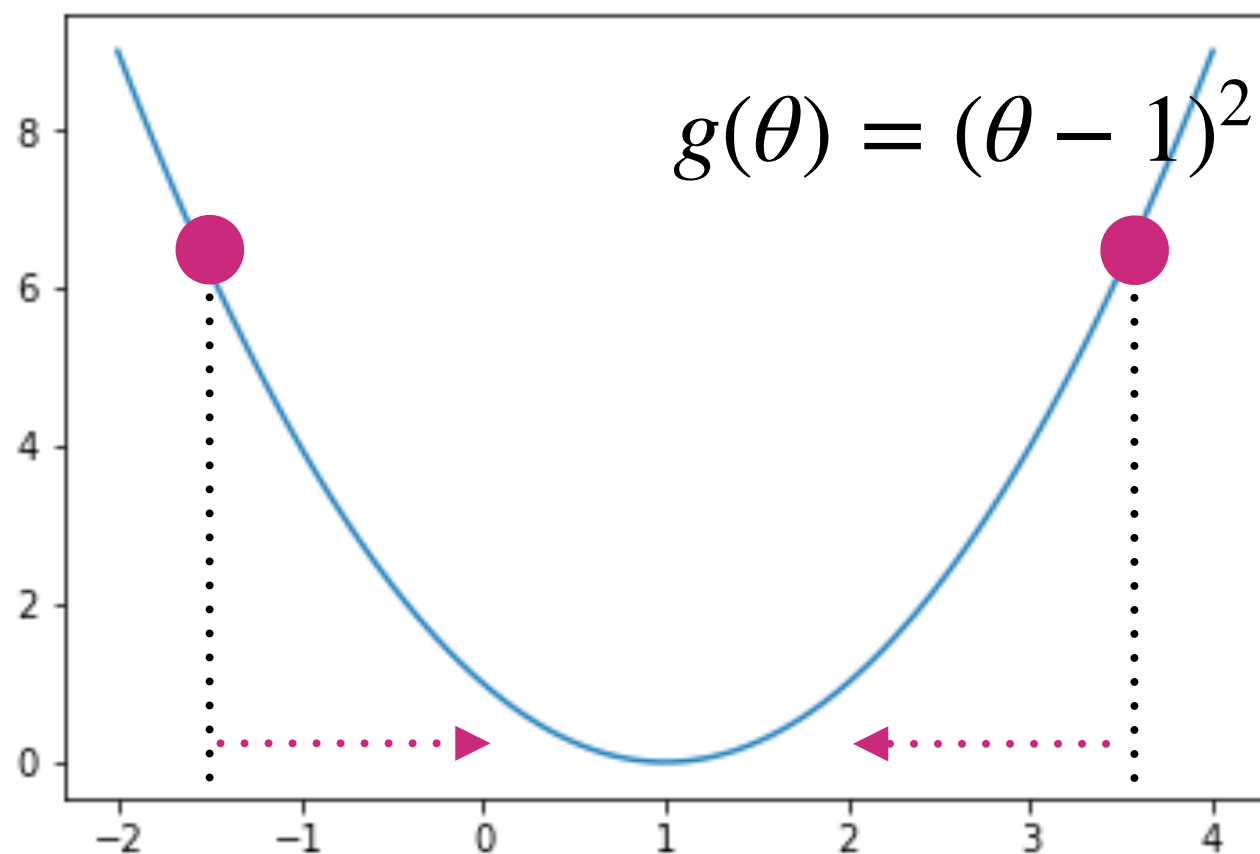
- Goal: Minimize Cost function

$$-\sum_{i=1}^m \left[y^{(i)} \log (\hat{y}^{(i)}) + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)}) \right]$$



Gradient descent (梯度下降法)

往與導函數相反的方向移動，就會往最小值的方向移動



Gradient descent (梯度下降法)

$$\theta := \theta - \eta \frac{d}{d\theta} g(\theta)$$

η : learning rate

Logistic Regression: Algorithm

- Gradient descent (梯度下降法)

$$\theta := \theta - \eta \frac{d}{d\theta} g(\theta)$$

- Algorithm

$$\theta_j := \theta_j + \eta \sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)}) x_j^{(i)}$$

Logistic Regression: regularization

- Goal: Minimize Cost function

$$\sum_{i=1}^m \left[-y^{(i)} \log(\hat{y}^{(i)}) - (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right] + \lambda \sum_{i=1}^n \theta_i^2$$

λ : regularization parameter (正規化參數)

Iris Data Set (鳶尾花卉數據集)

■ UCI

<https://archive.ics.uci.edu/ml/datasets/Iris>

<https://www.kaggle.com/uciml/iris>

- 150個樣本，都屬於鳶尾屬下的三個亞屬，分別是山鳶尾、變色鳶尾和維吉尼亞鳶尾。

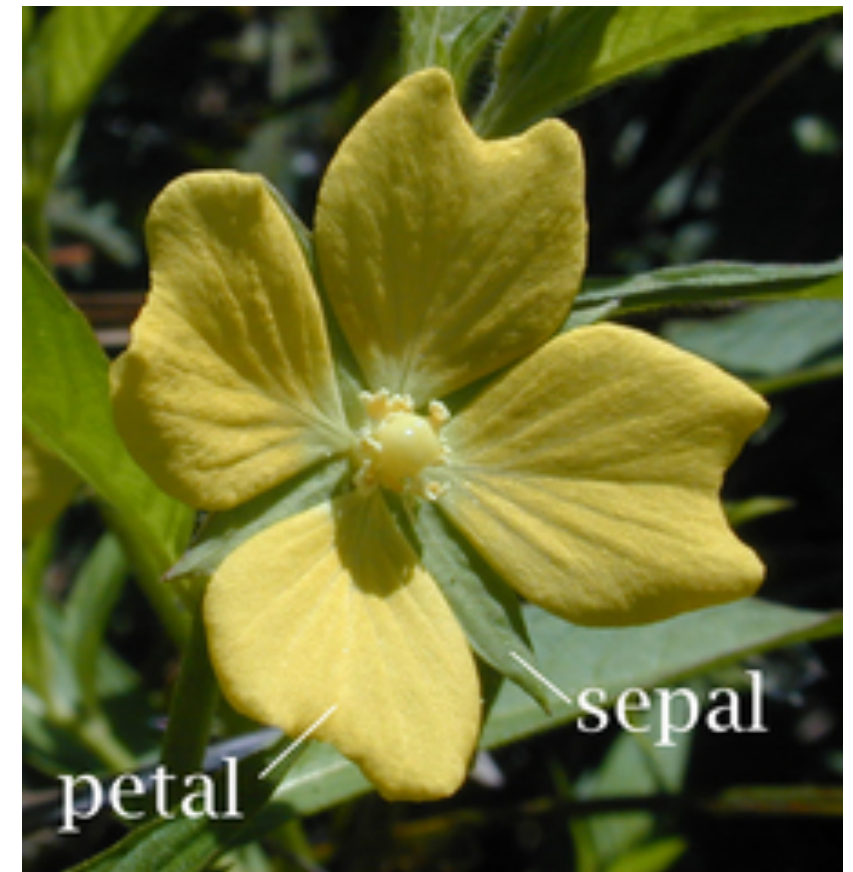
versicolor

virginica

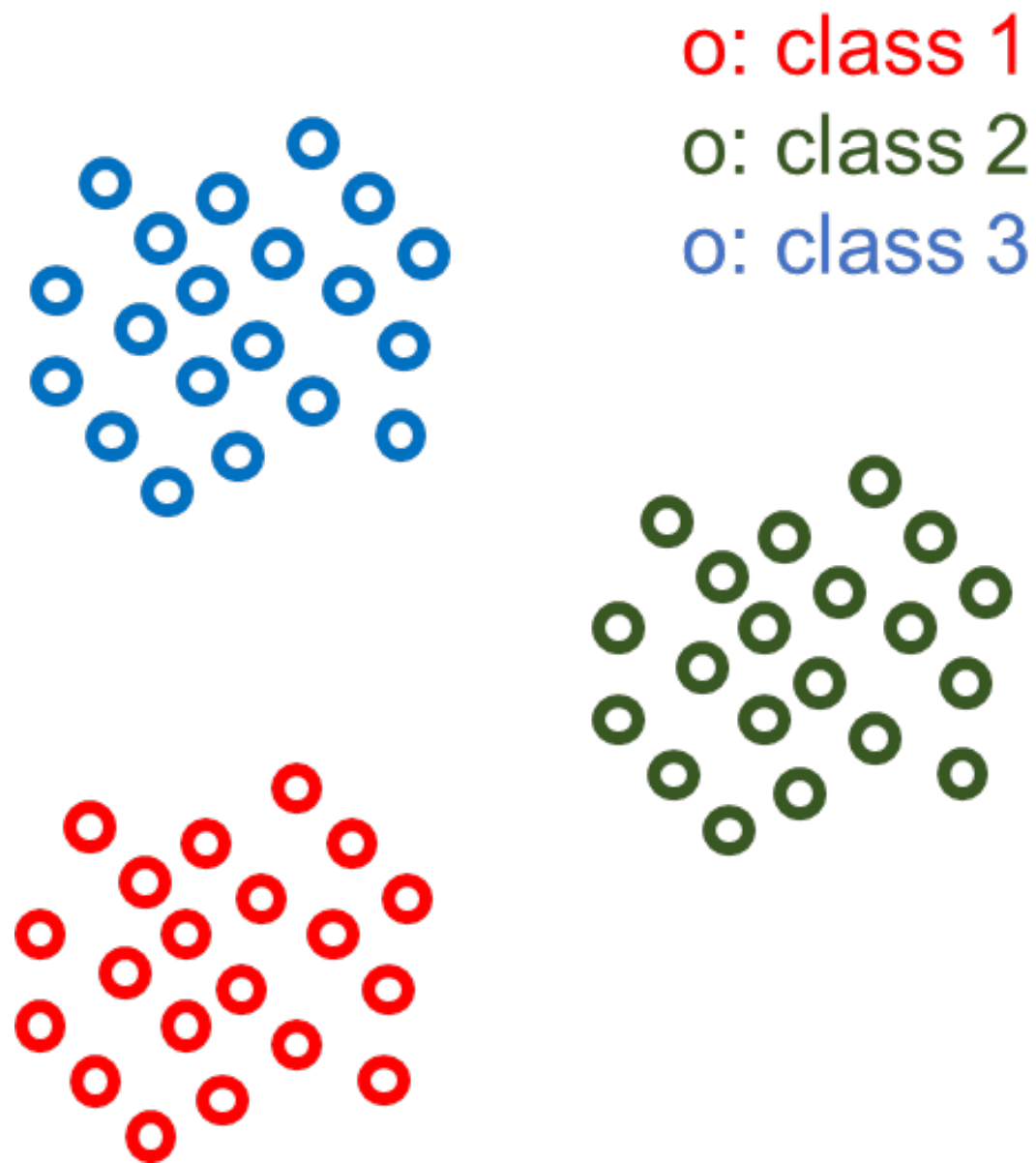
- 四個特徵：花萼和花瓣的長度和寬度

Sepal Petal

setosa

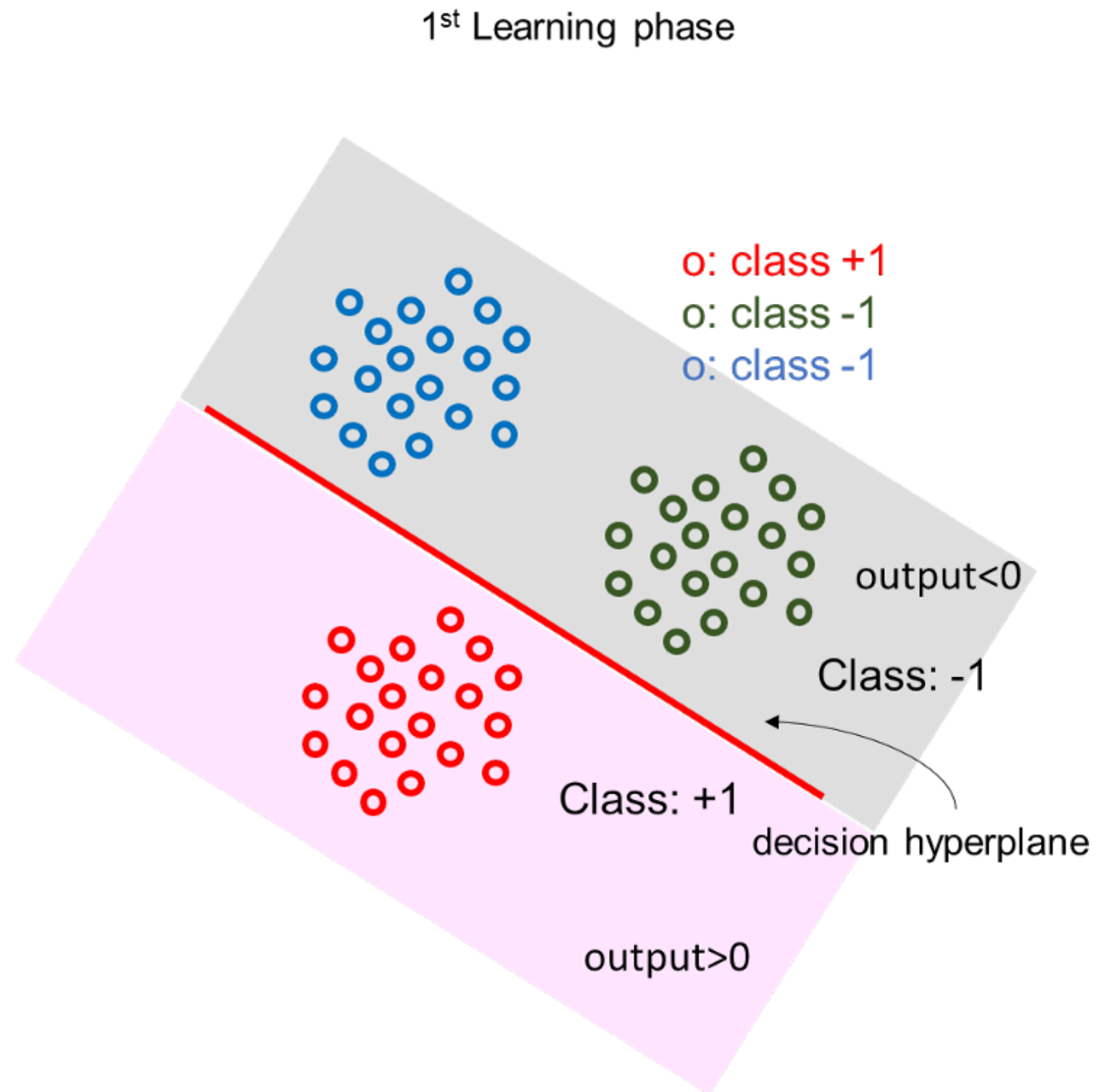


Multiple Classification



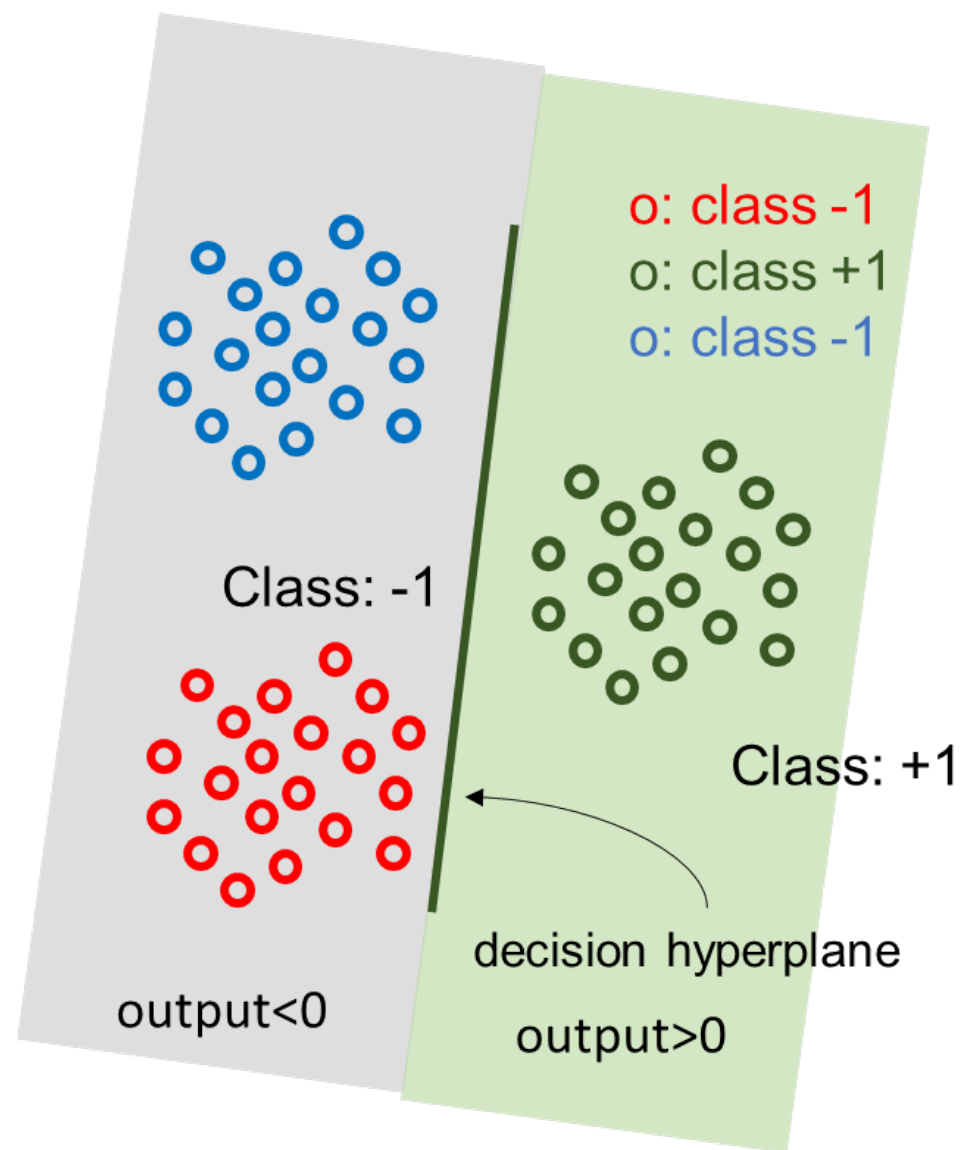
Multiple Classification

OvR (One-vs.-Rest)

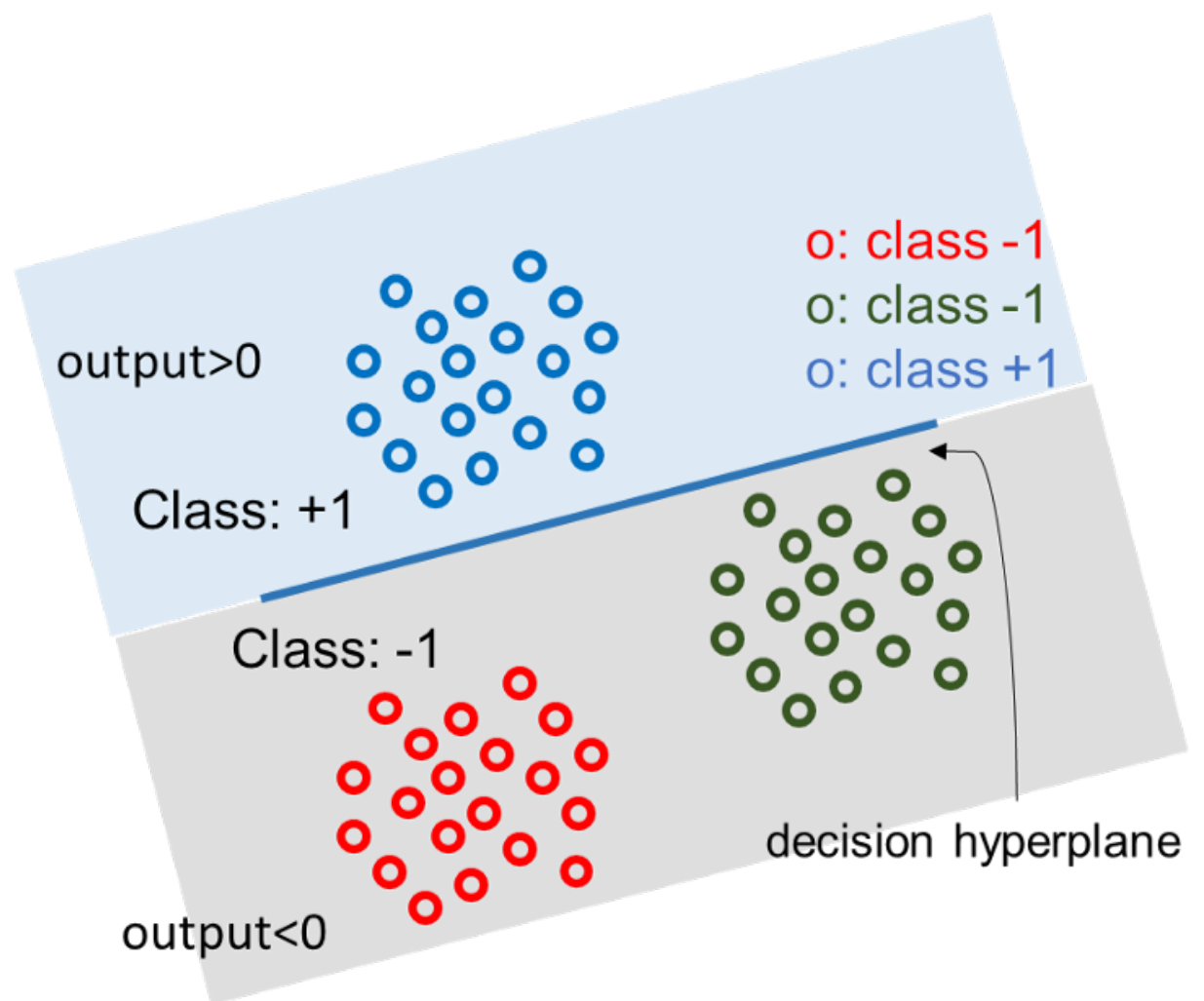


Multiple Classification

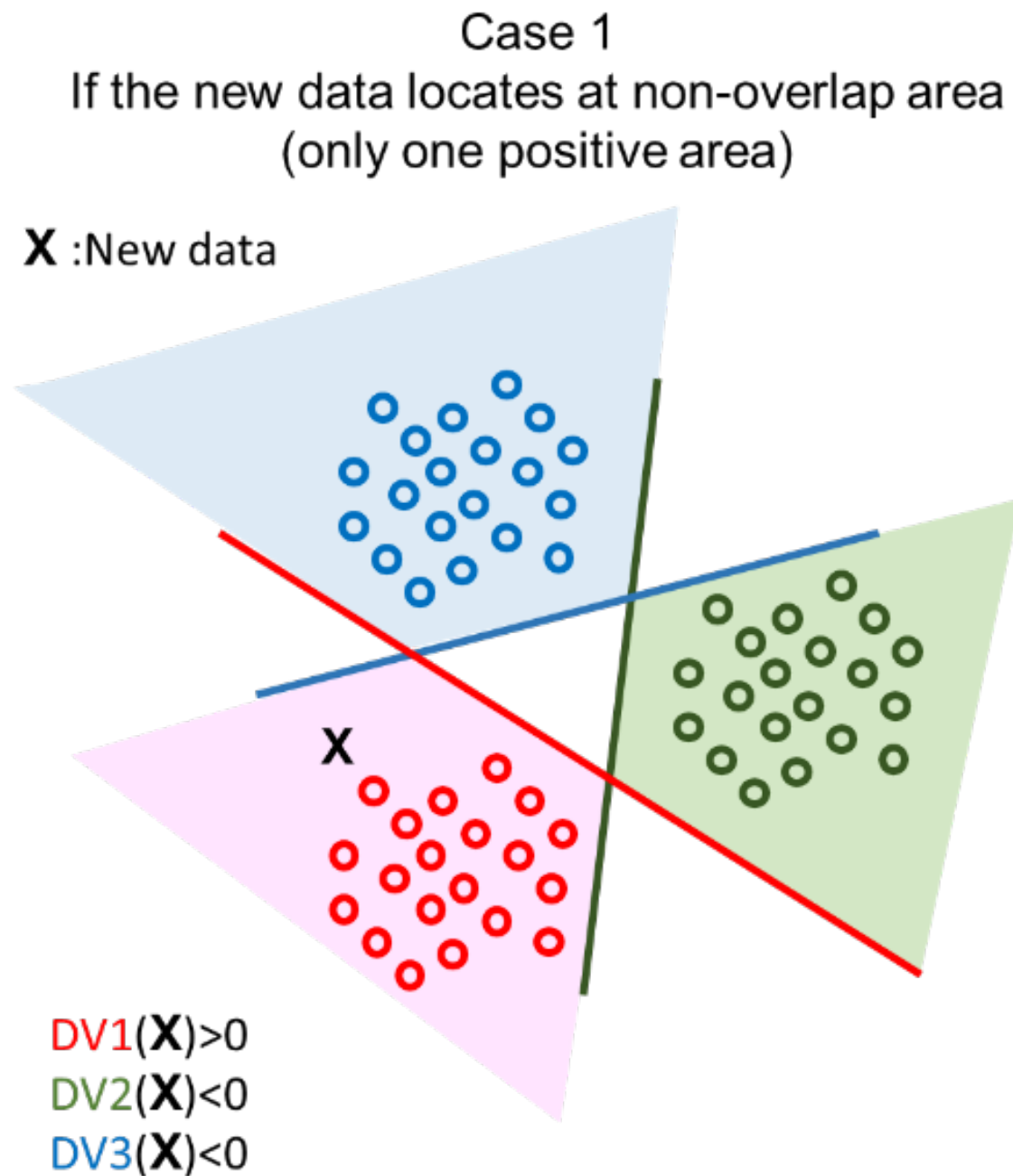
2nd Learning phase



3th Learning phase

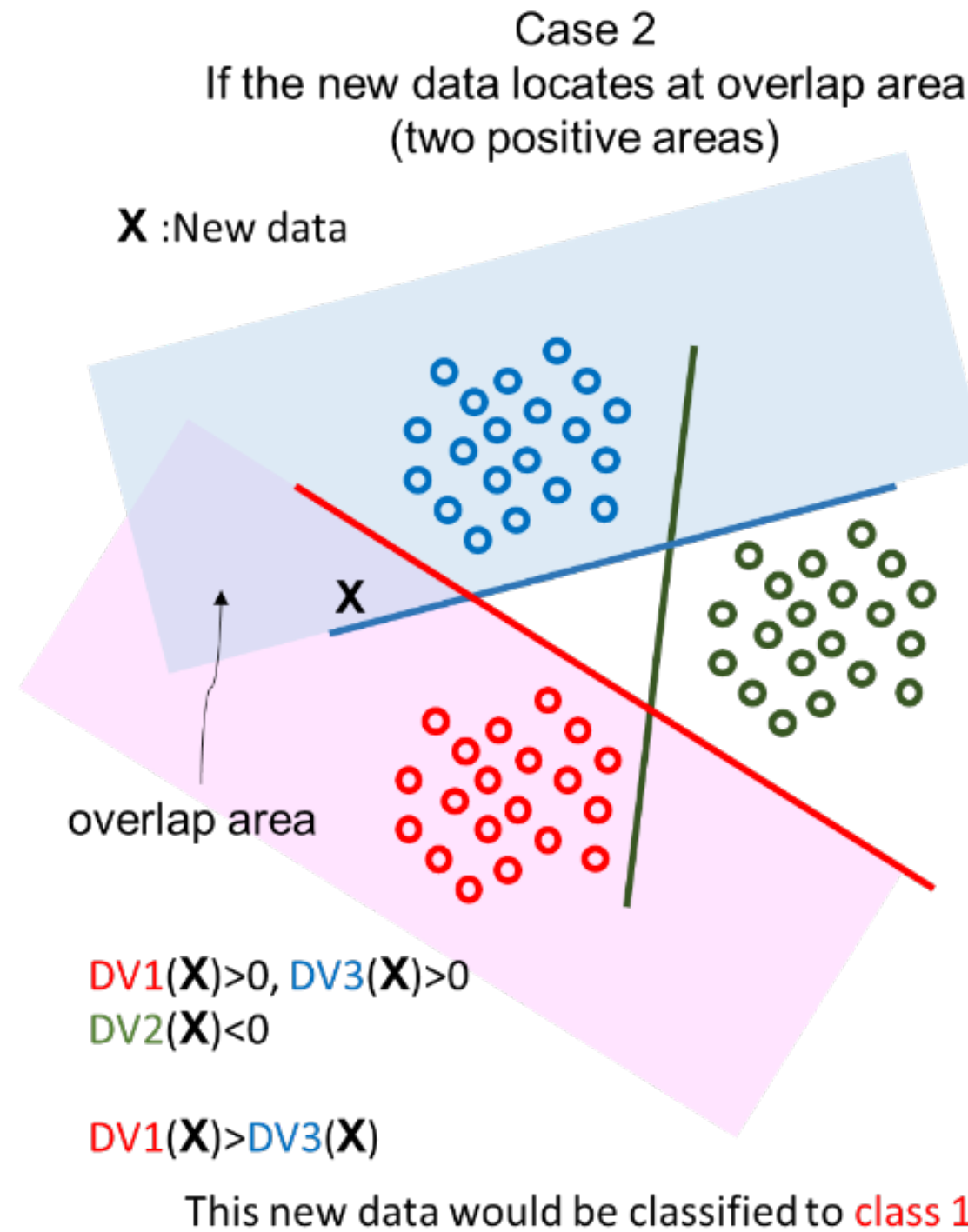


Multiple Classification



This new data would be classified to **class 1**.

Multiple Classification

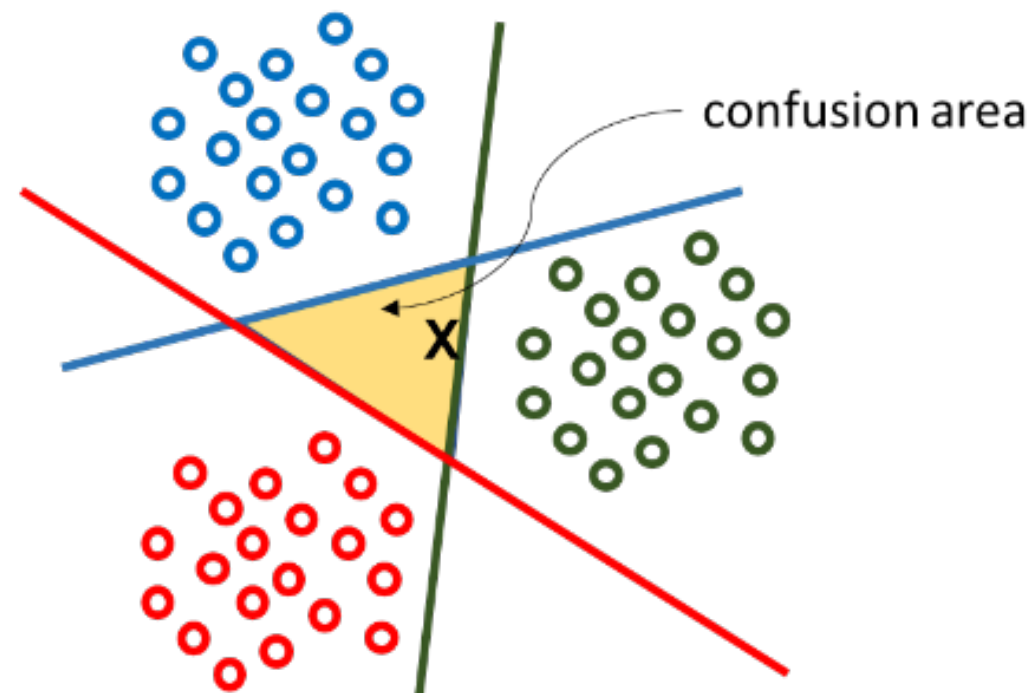


Multiple Classification

Case 3

If the new data locates at confusion area
(three negative areas)

X :New data



$$DV1(X) < 0, DV2(X) < 0, DV3(X) < 0$$

$$DV2(X) > DV3(X) > DV1(X)$$

This new data would be classified to **class 2**.

Python

■ 載入 Iris data set

(1/6)

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris

iris = load_iris()

feature = pd.DataFrame(iris['data'], columns = iris['feature_names'])
target = pd.DataFrame(iris['target'], columns = ['class'])

data = pd.concat([feature, target], axis = 1)
df = data[data['class'] != 2]
```

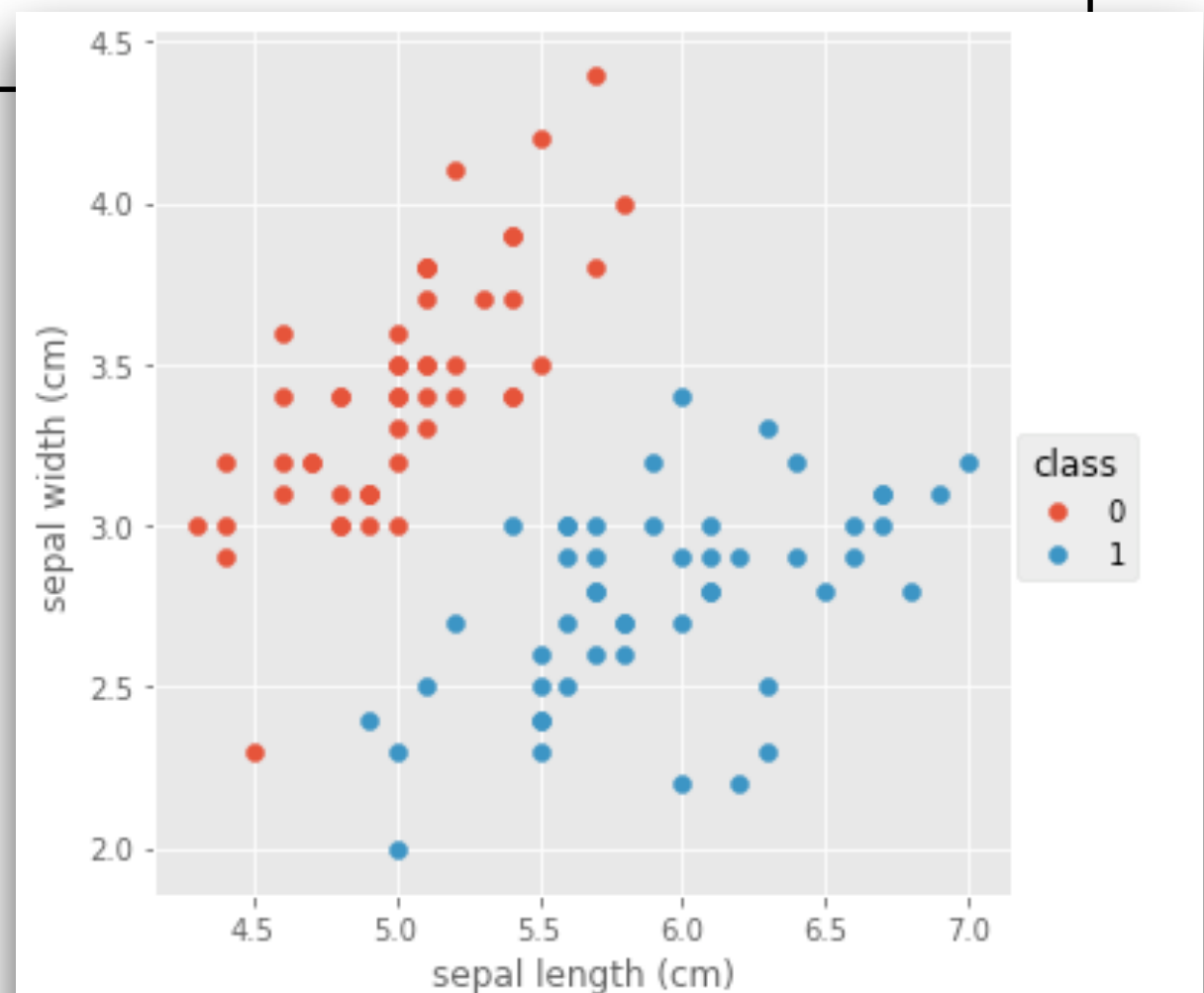
只考慮類別為 0 和 1 的資料

Python

■ 畫散佈圖

(2/6)

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('ggplot') # make plots look better
g = sns.FacetGrid(df, hue="class", size=5)
g.map(plt.scatter, "sepal length (cm)", "sepal width (cm)")
g.add_legend()
```



Python

■ 標準化

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X = df.iloc[:, :2].values
y = df.iloc[:, 4].values

sc = StandardScaler()
sc.fit(X)
X_std = sc.transform(X)
```

(3/6)

Python

■ 訓練 logistic regression 模型

(4/6)

```
from sklearn.linear_model import LogisticRegression
from matplotlib.colors import ListedColormap
```

```
lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_std, y)
```

```
print(lr.coef_)
print(lr.intercept_)
```

Output $\vec{\theta}$

$$C = \frac{1}{\lambda}$$

Python

■ 定義決策區域 (decision region)

```
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=0.3, 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=colors[idx],
                    marker=markers[idx],
                    label=cl,
                    edgecolor='black')

    # highlight test samples
    if test_idx:
        # plot all samples
        X_test, y_test = X[test_idx, :], y[test_idx]

        plt.scatter(X_test[:, 0],
                    X_test[:, 1],
                    c='',
                    edgecolor='black',
                    alpha=1.0,
                    linewidth=1,
                    marker='o',
                    s=100,
                    label='test set')
```


Python

■ 畫出目前模型的決策邊界

(6/6)

```
plot_decision_regions(X_std, y, classifier=lr)
plt.xlabel('sepal length [standardized]')
plt.ylabel('sepal width [standardized]')
plt.legend(loc='upper left')
plt.tight_layout()
#plt.savefig('images.png', dpi=300)
plt.show()
```

