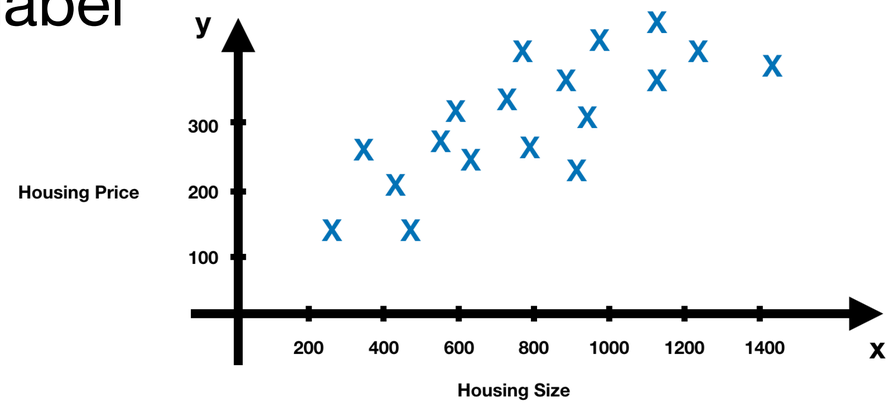
X-Village AI/ML - Classification

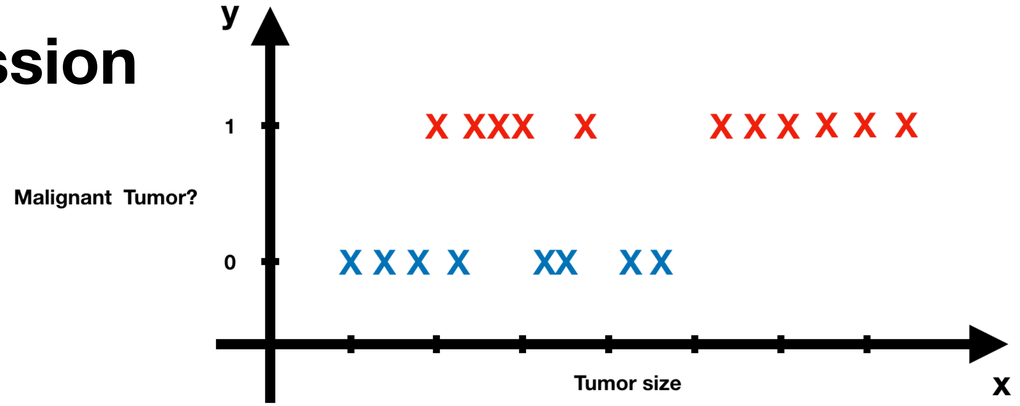
speaker: Amber (吳思嬋)

Recall - Supervised Learning

- Regression Problem: predict continuous valued label
  - Hypothesis Function - Linear Regression
  - Gradient Descent Algorithm

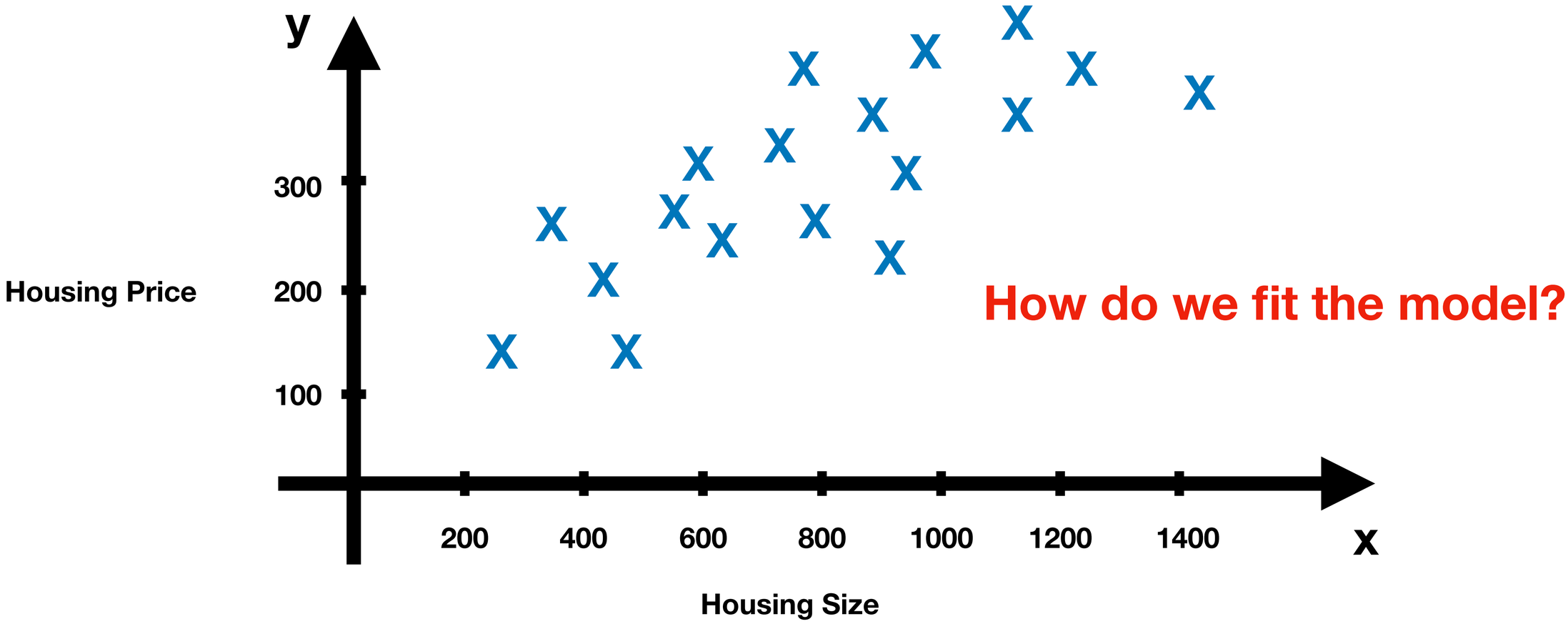


- **Classification Problem: predict discrete valued label**
  - **Hypothesis Function - Logistic Regression**
  - **Gradient Descent Algorithm**



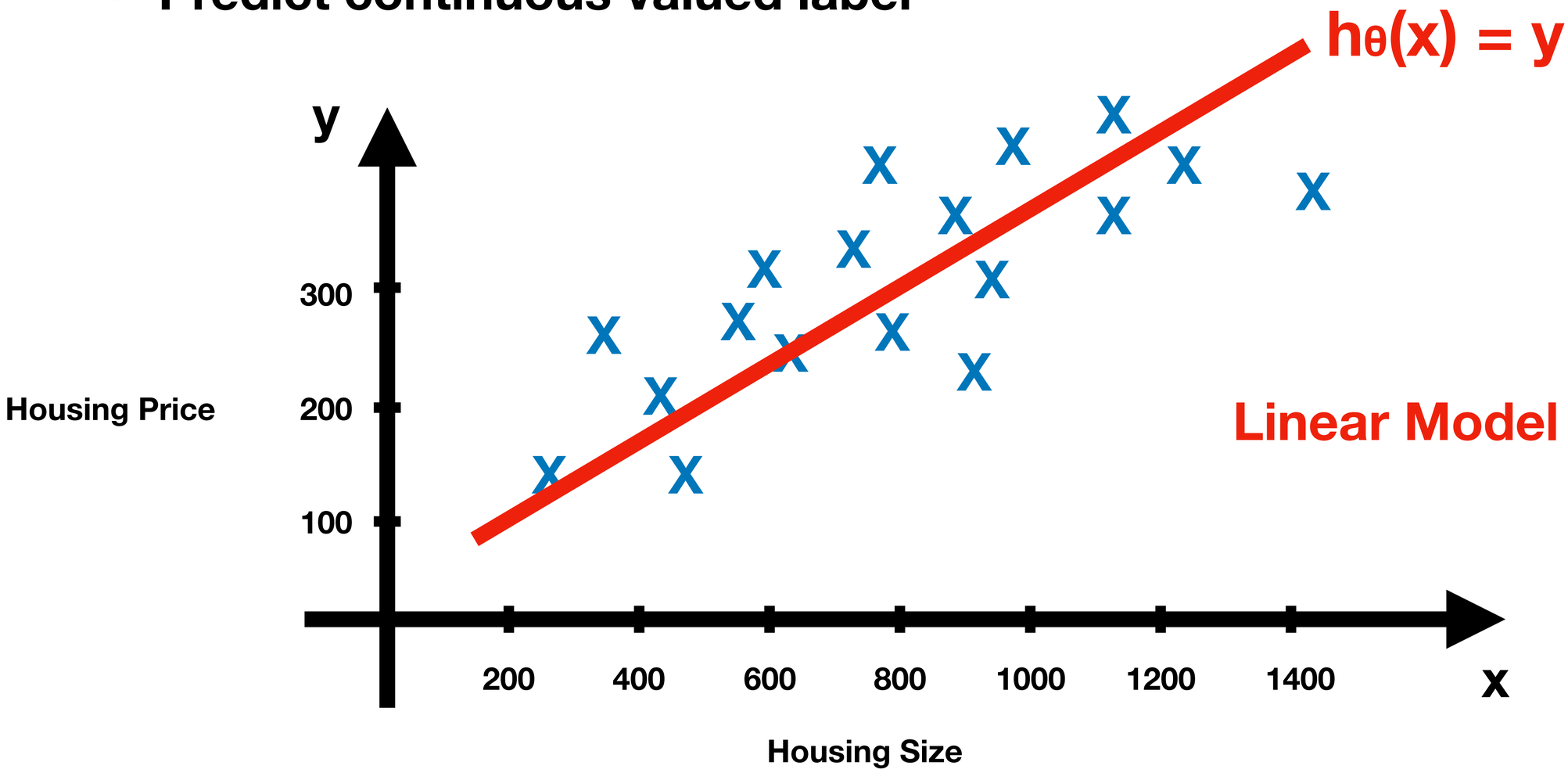
Recall - Linear Regression(1)

- **Predict continuous valued label**



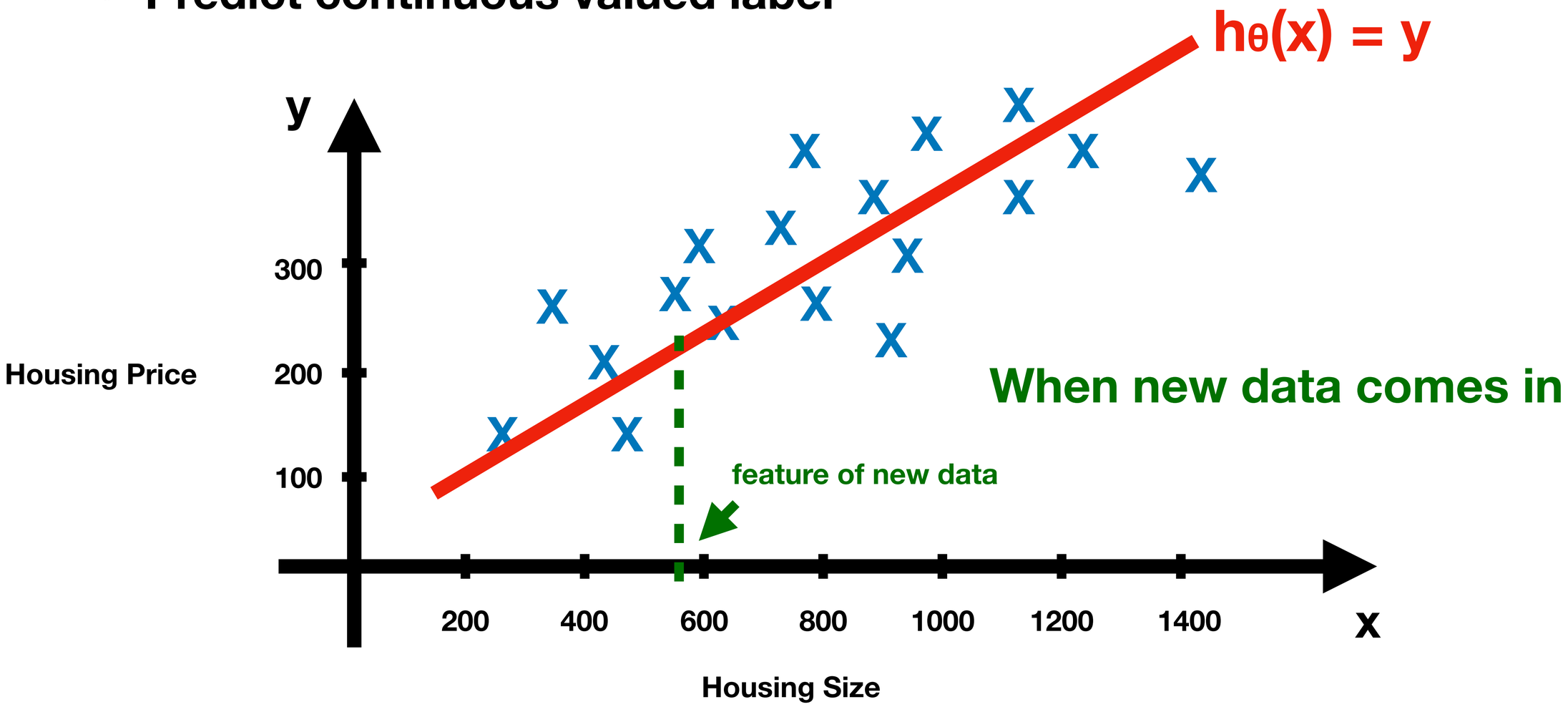
Recall - Linear Regression(2)

- **Predict continuous valued label**



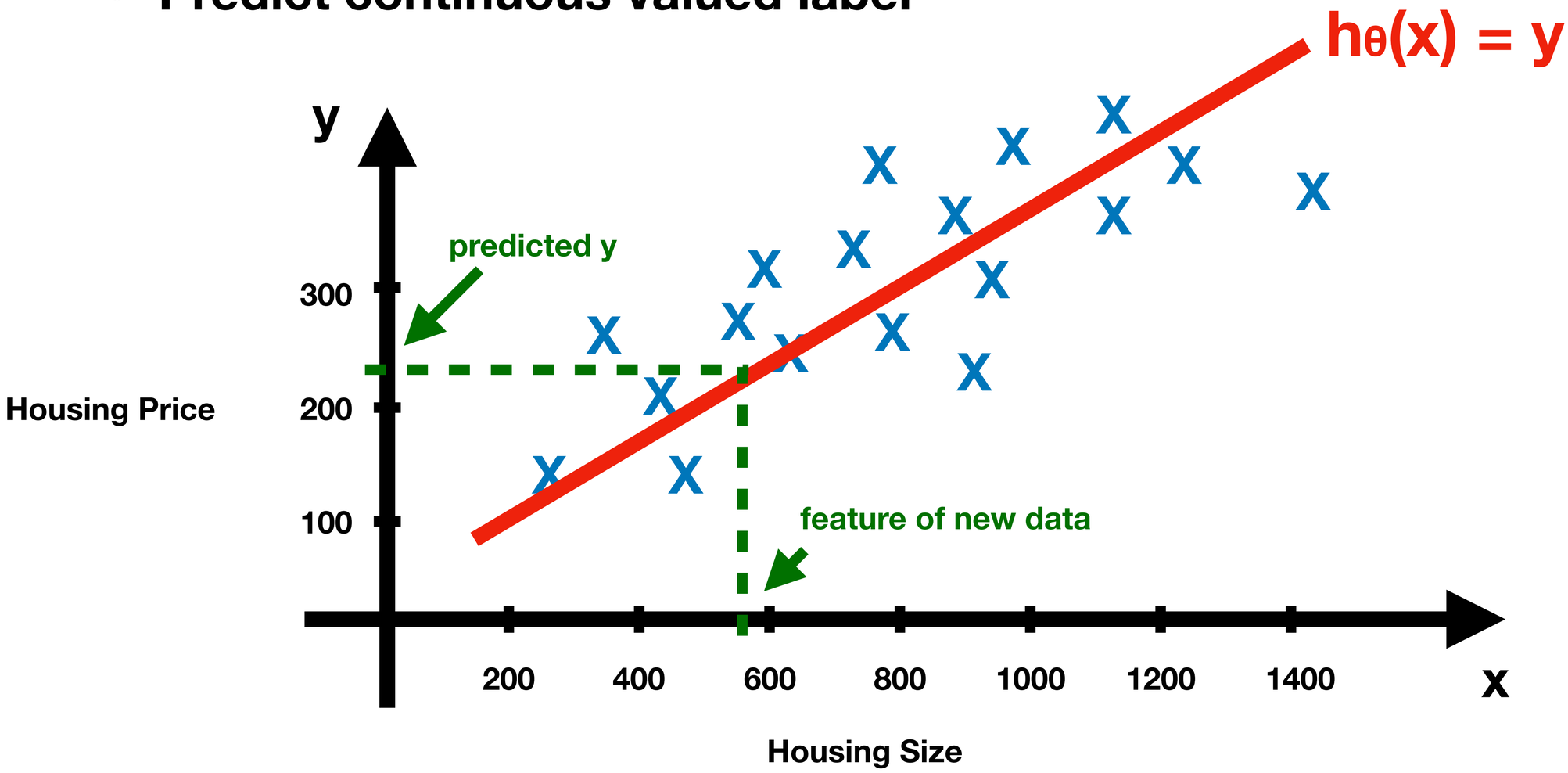
Recall - Linear Regression(3)

- **Predict continuous valued label**



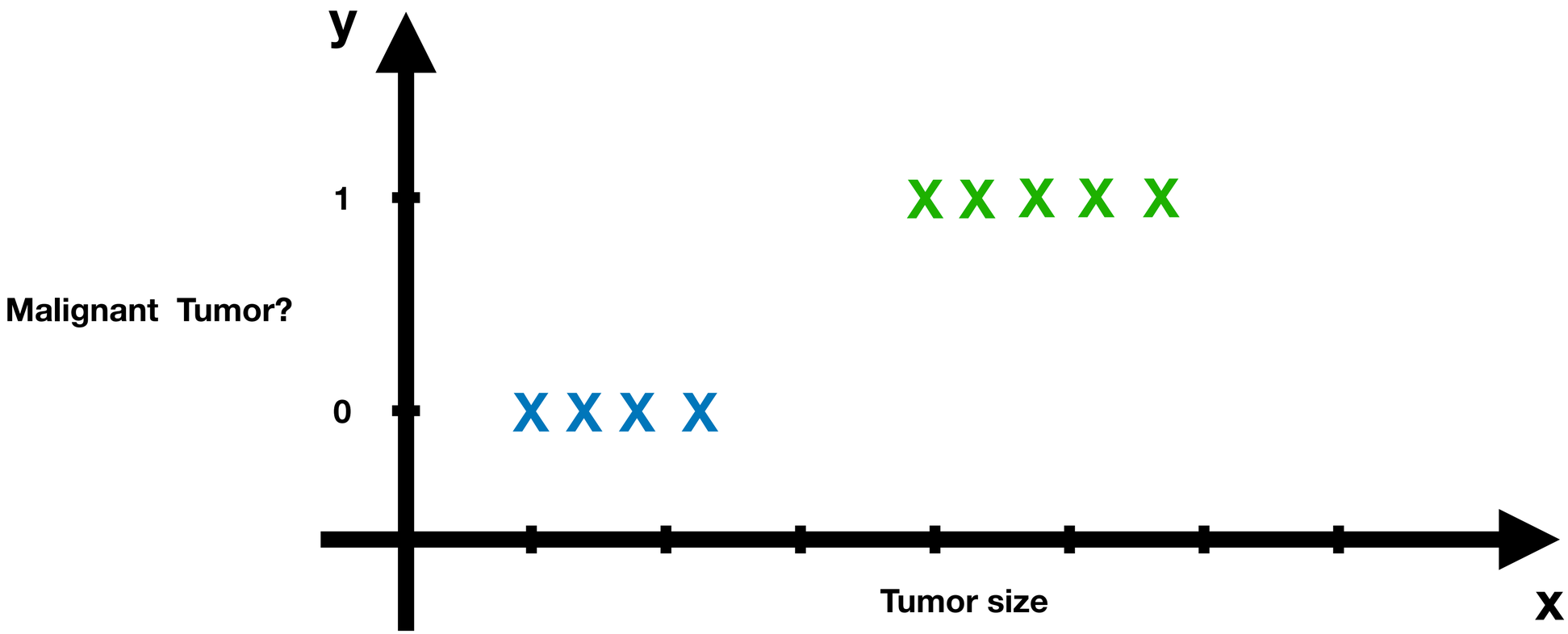
Recall - Linear Regression(4)

- **Predict continuous valued label**



Classification Problem (1)

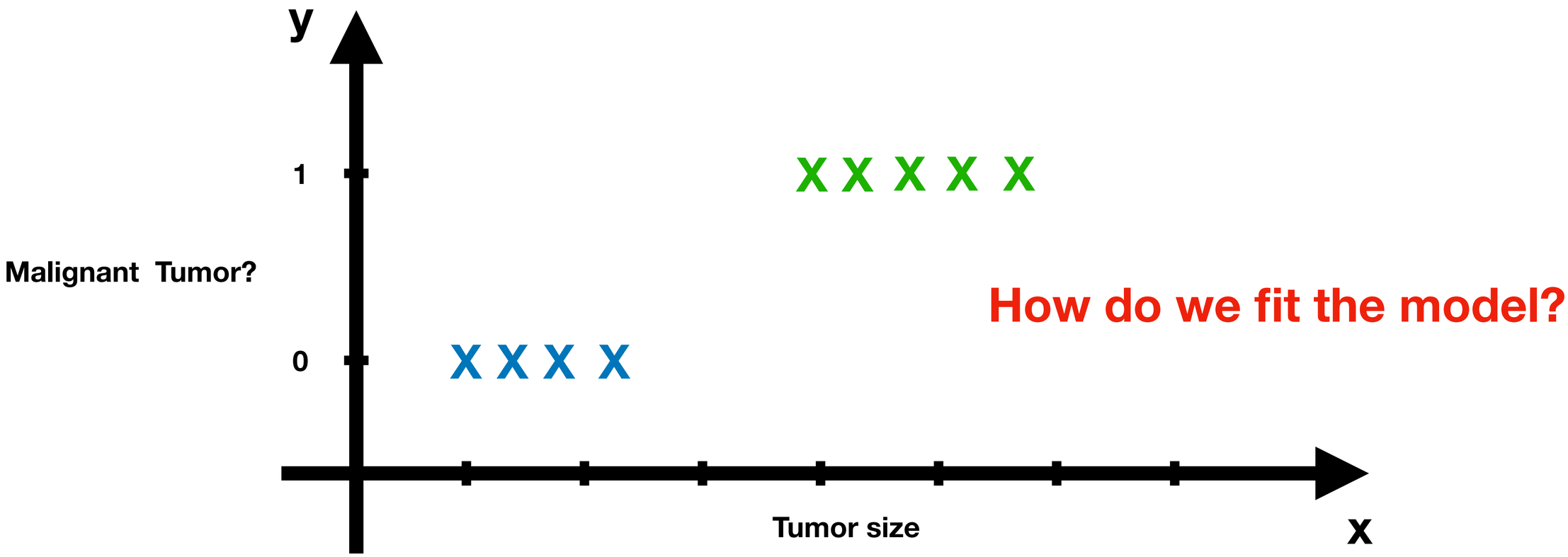
- **Predict discrete valued label**



Can we apply Linear Regression to Classification Problem ?

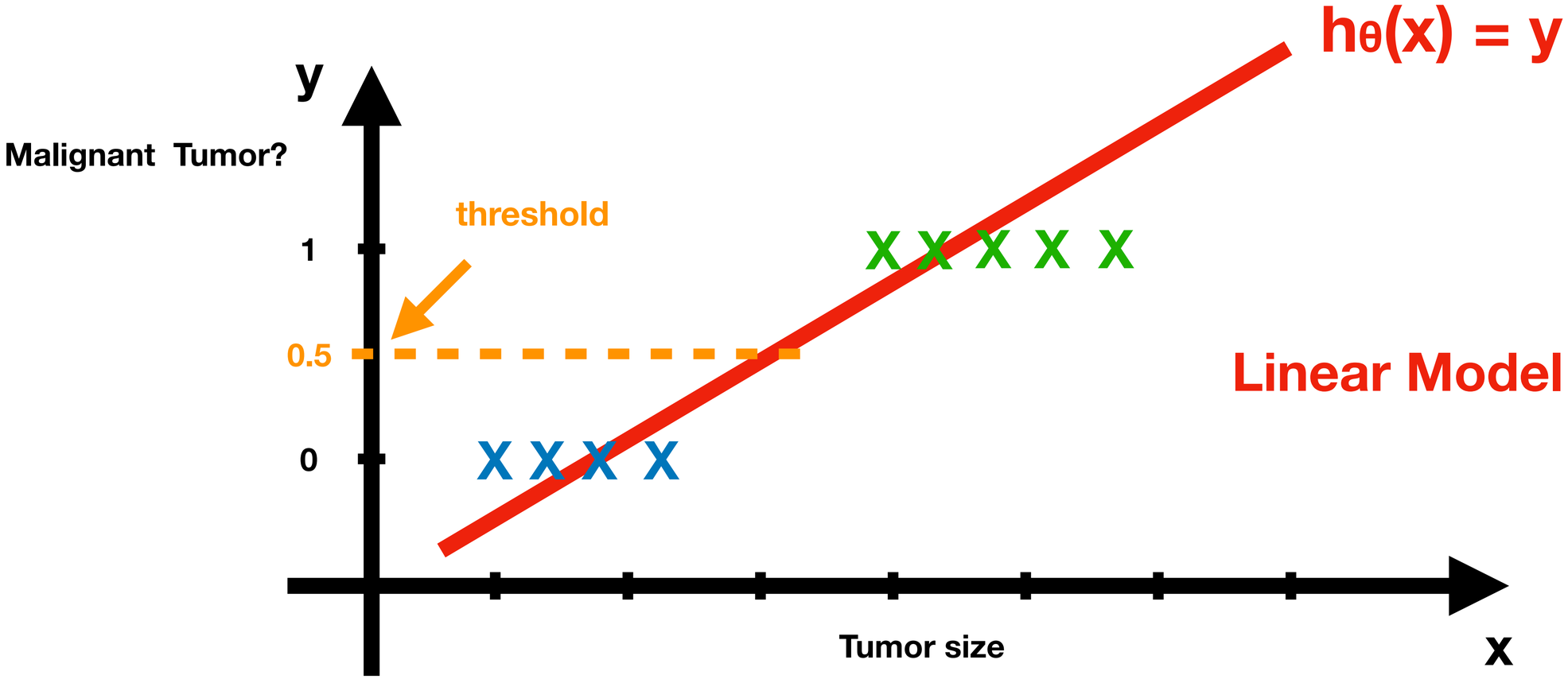
Classification Problem (2)

- **Predict discrete valued label**



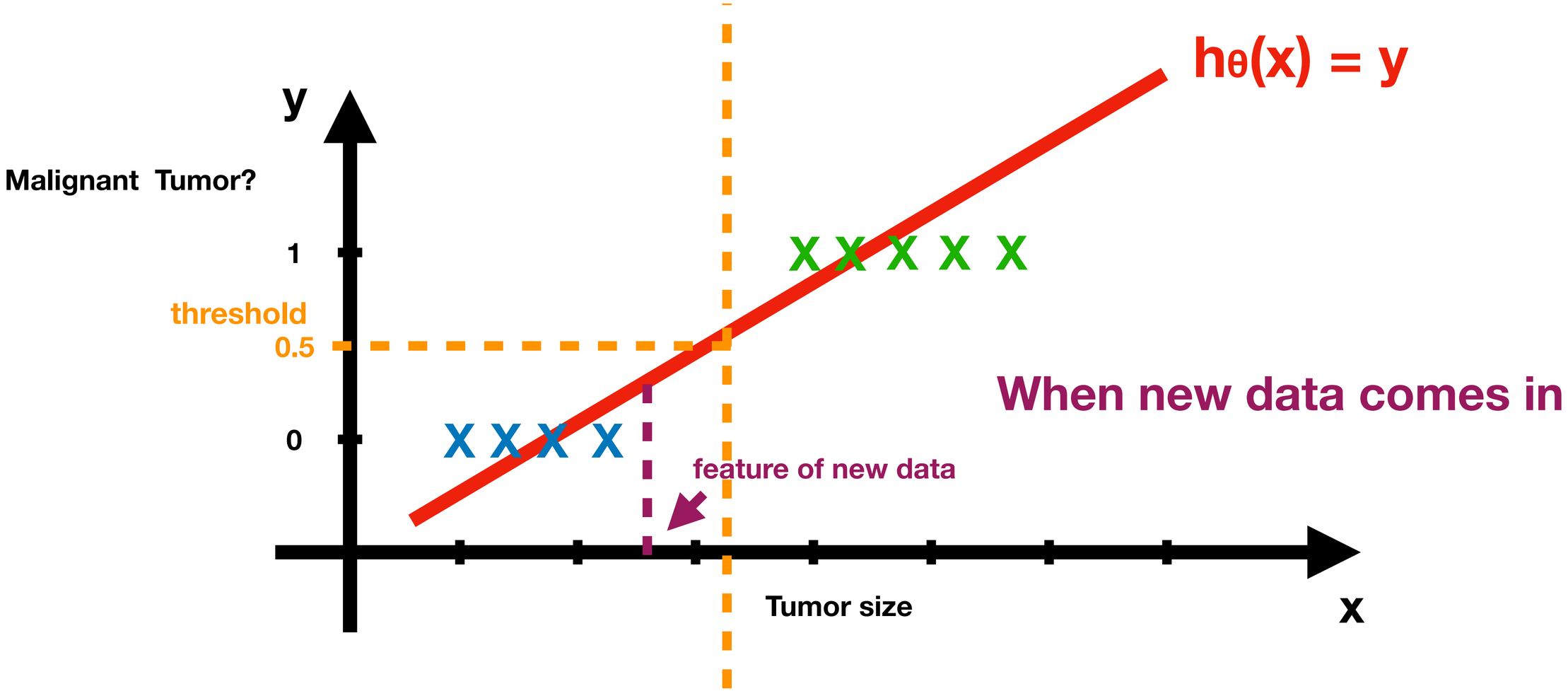
Classification Problem (3)

- **Predict discrete valued label**



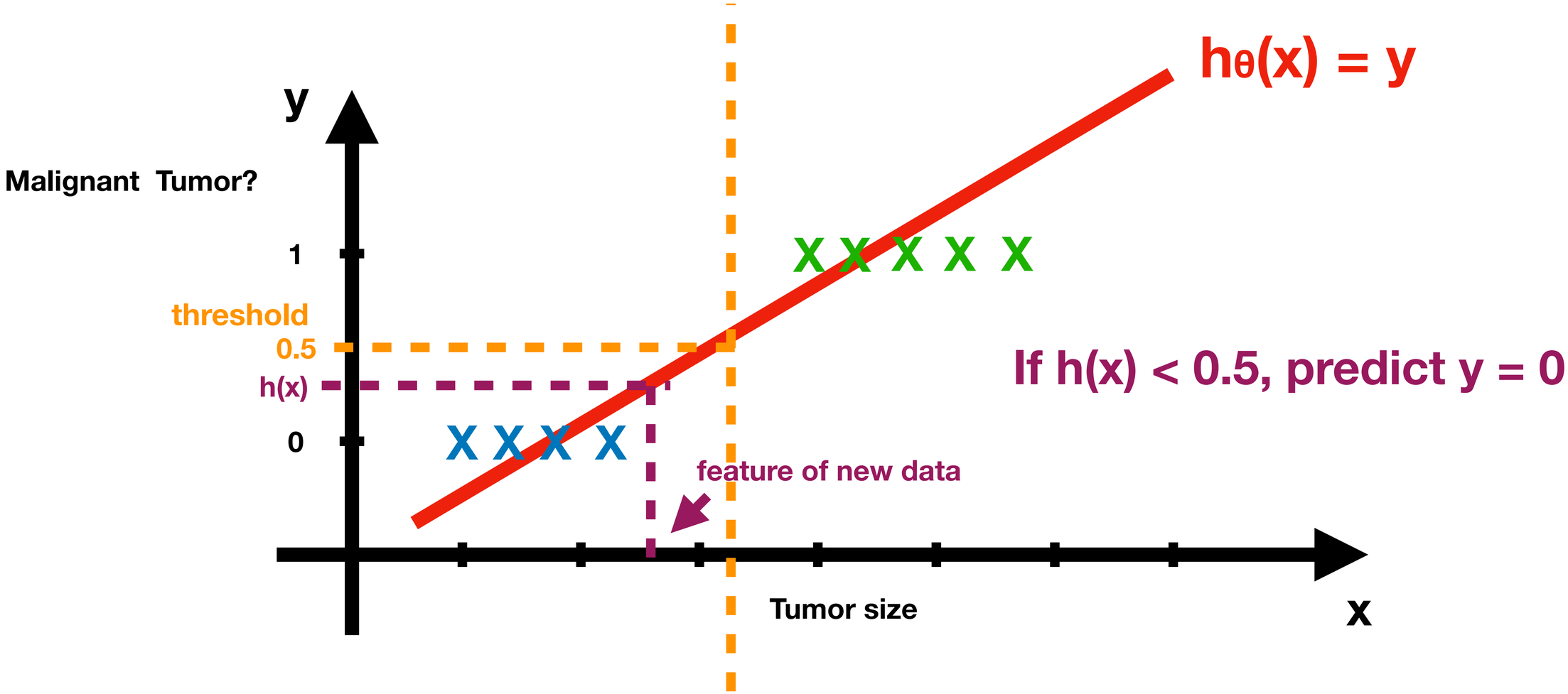
Classification Problem (4)

- **Predict discrete valued label**



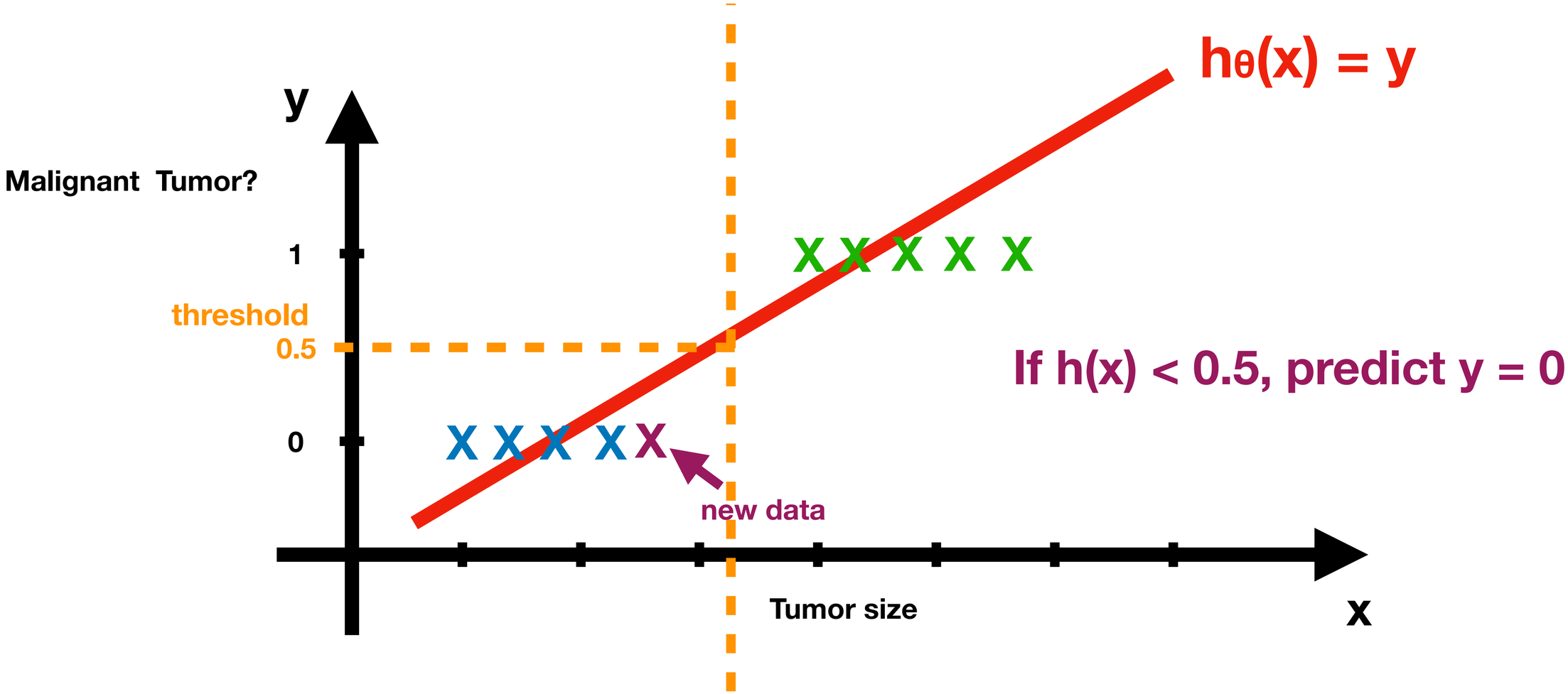
Classification Problem (5)

- **Predict discrete valued label**



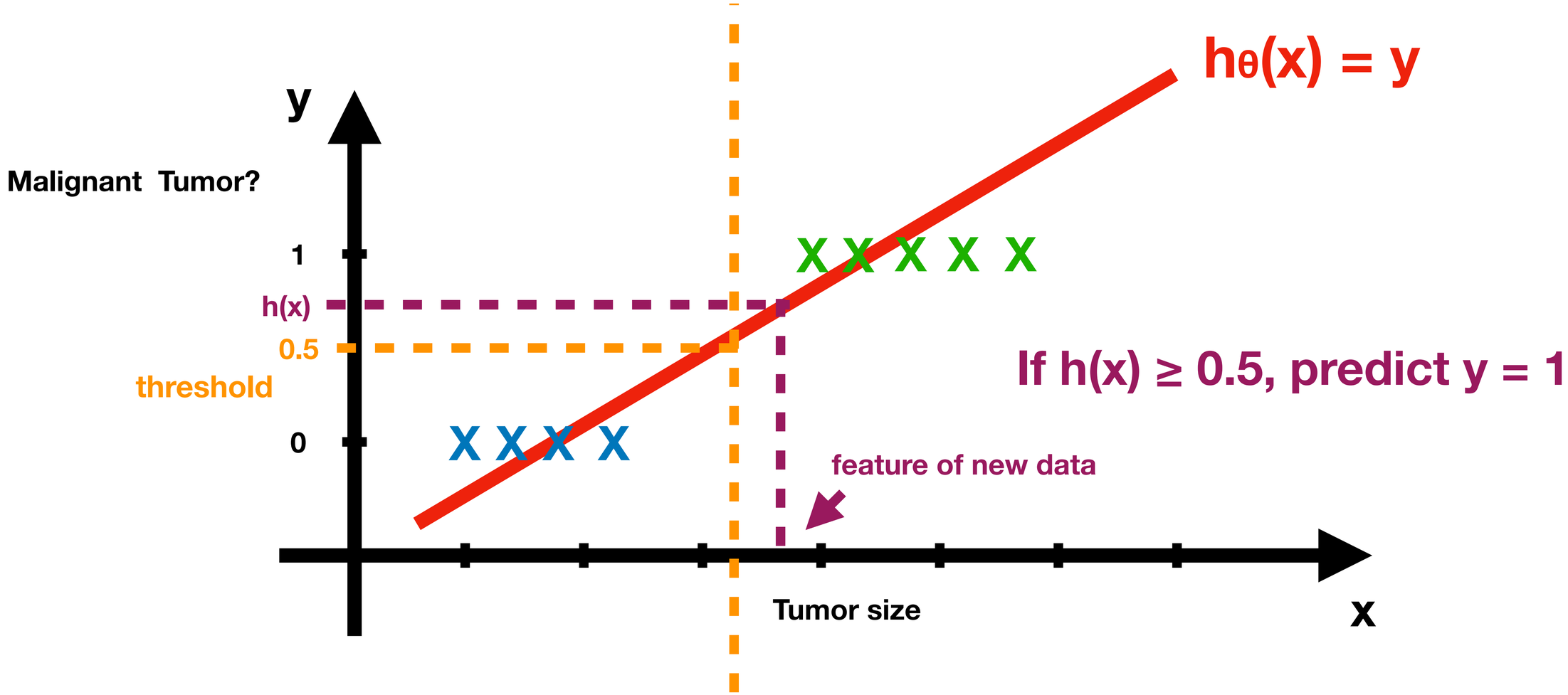
Classification Problem (6)

- **Predict discrete valued label**



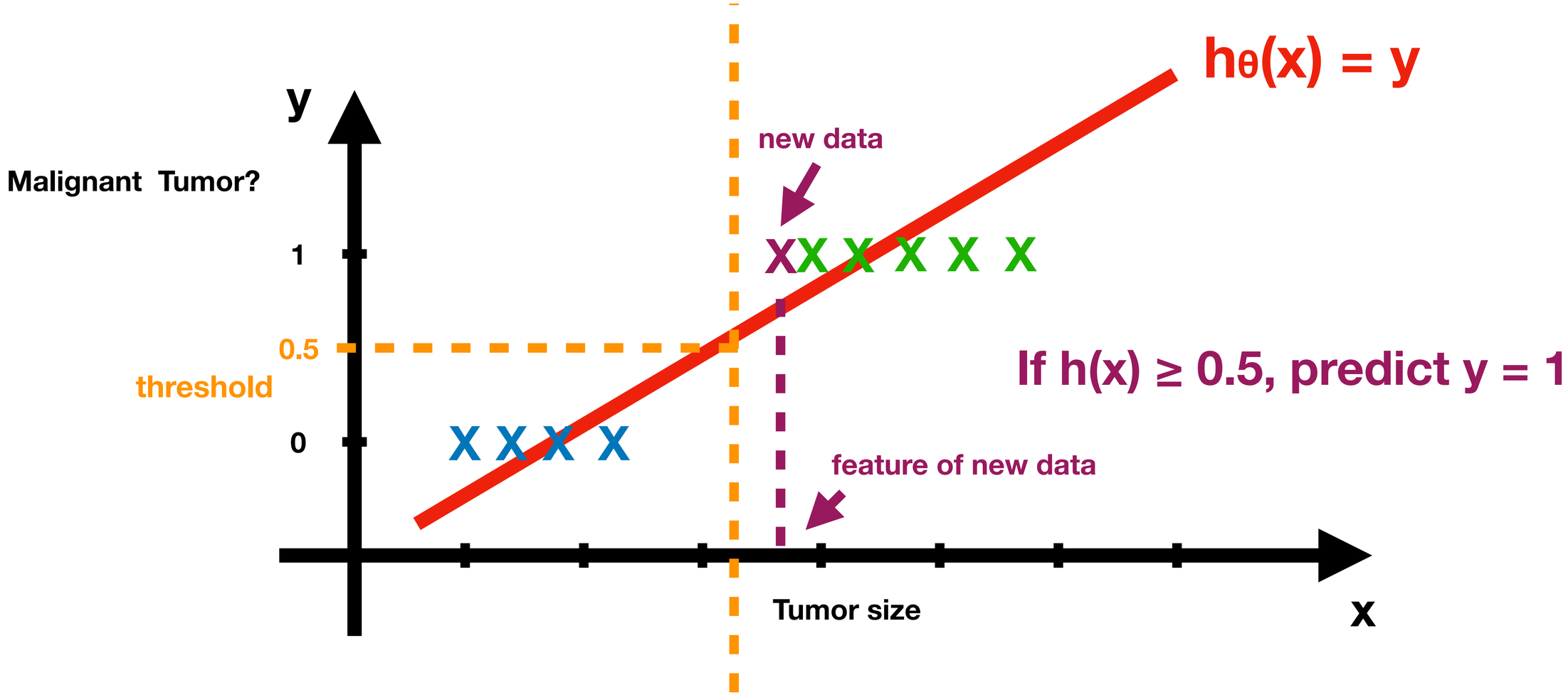
Classification Problem (7)

- **Predict discrete valued label**



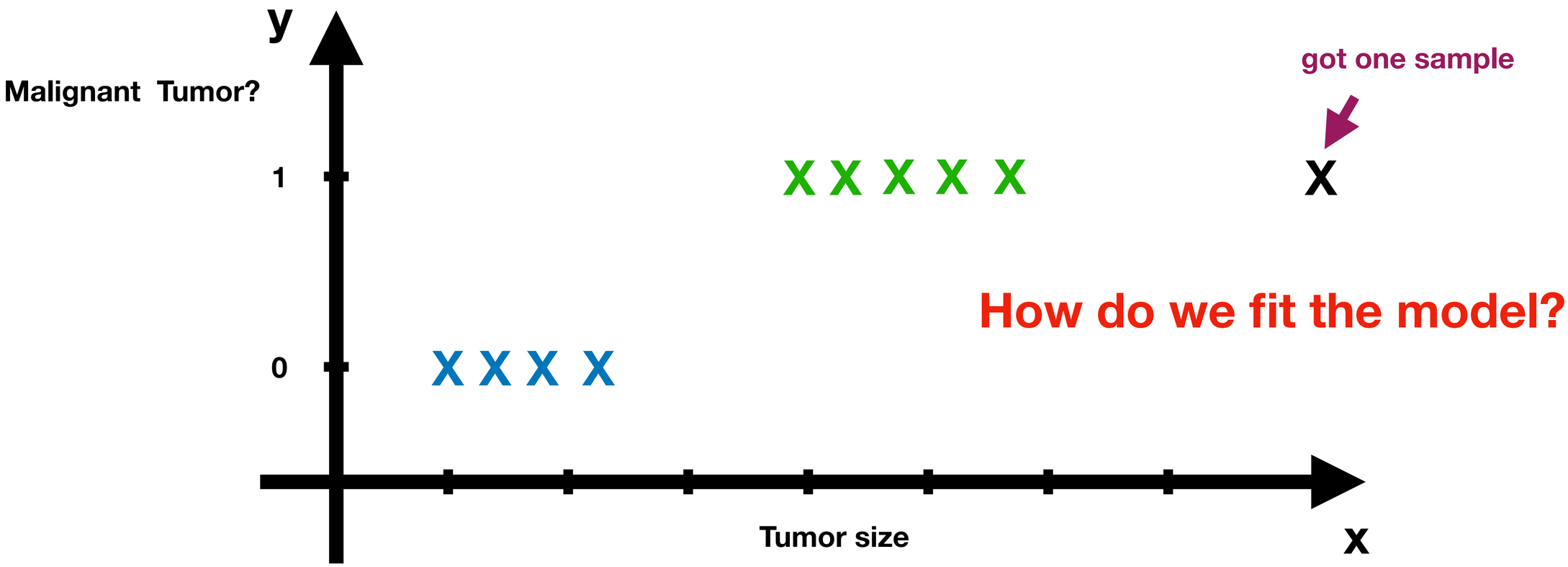
Classification Problem (8)

- **Predict discrete valued label**



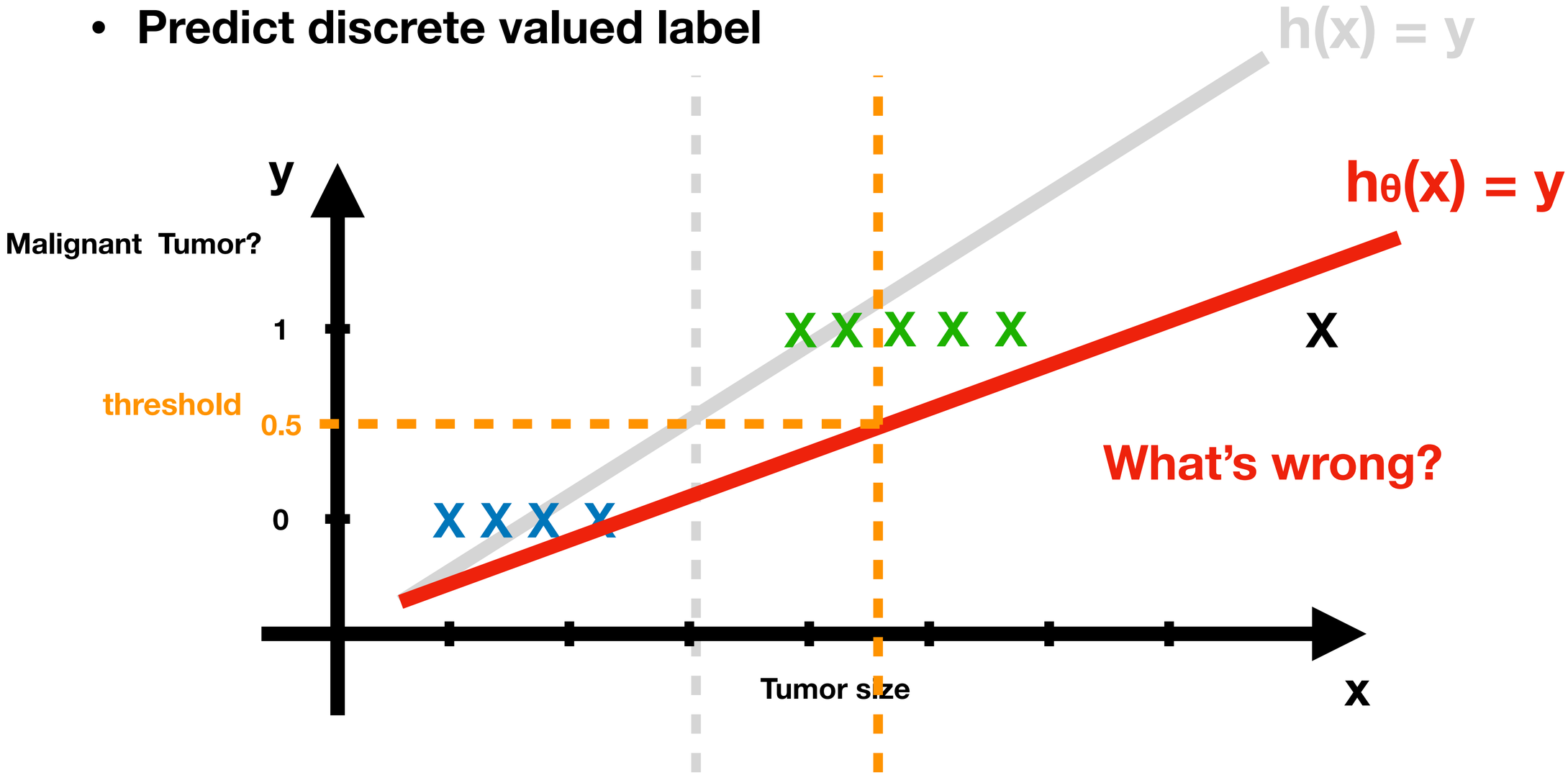
Classification Problem (9)

- **Predict discrete valued label**



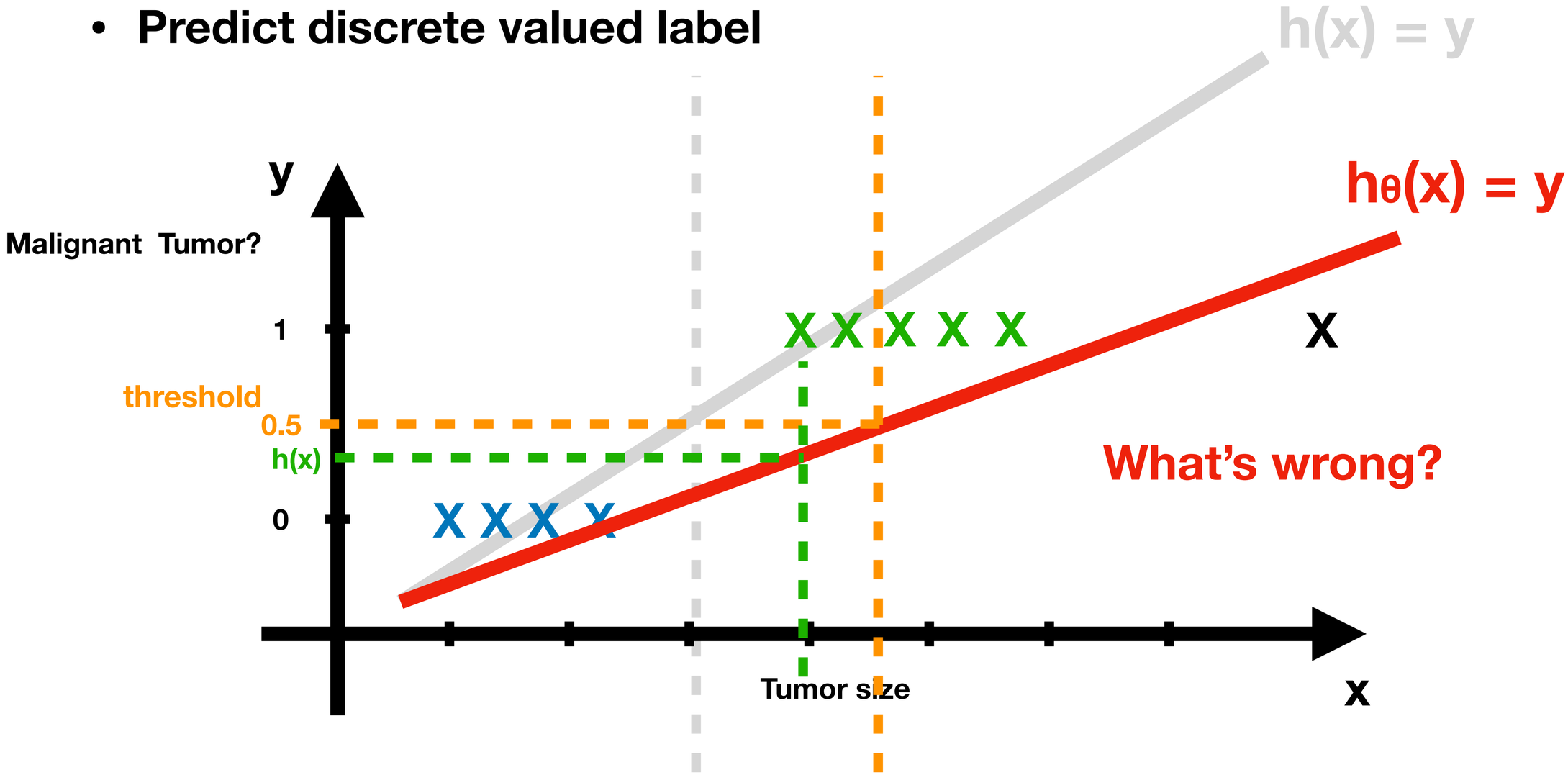
Classification Problem (10)

- **Predict discrete valued label**



Classification Problem (11)

- **Predict discrete valued label**



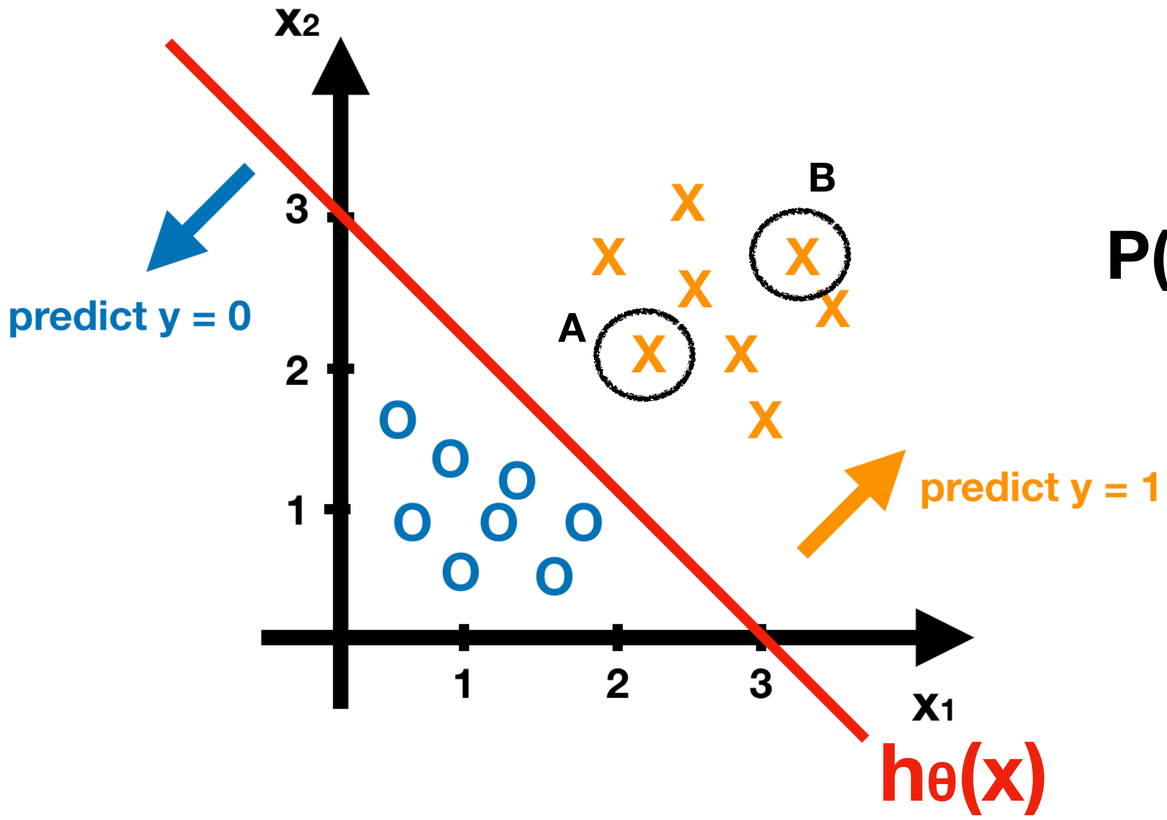
Thus, we WON’T apply Linear Regression to Classification Problem

Use Logistic Regression Instead

Logistic Regression

- Sigmoid Function $h_\theta(x)$: represent the **probability** of $y = 1$

hθ(x) = P(x∈1)



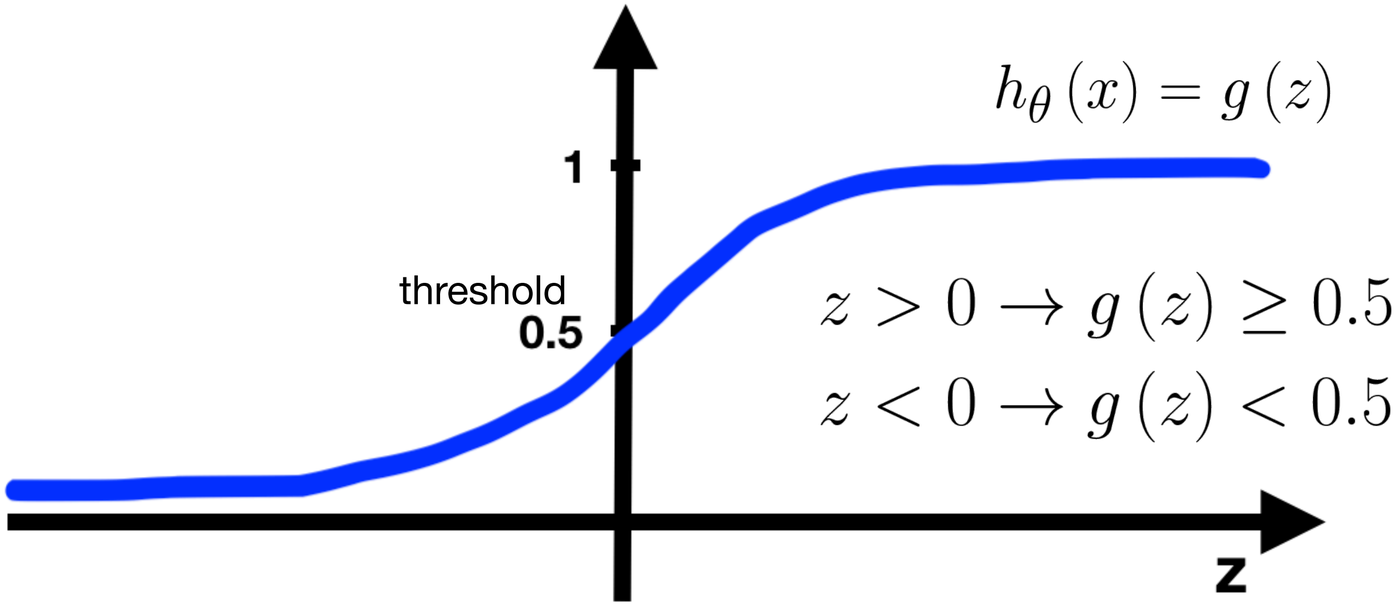
P(A∈1) < P(B∈1)

Logistic Regression

- Sigmoid Function $h_\theta(x)$: represent the **probability** of $y = 1$

$$h_\theta(x) = g(z) = \frac{1}{1 + e^{-z}}$$

$$(z = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n = \theta^T x)$$



$$z > 0 \rightarrow g(z) \geq 0.5$$

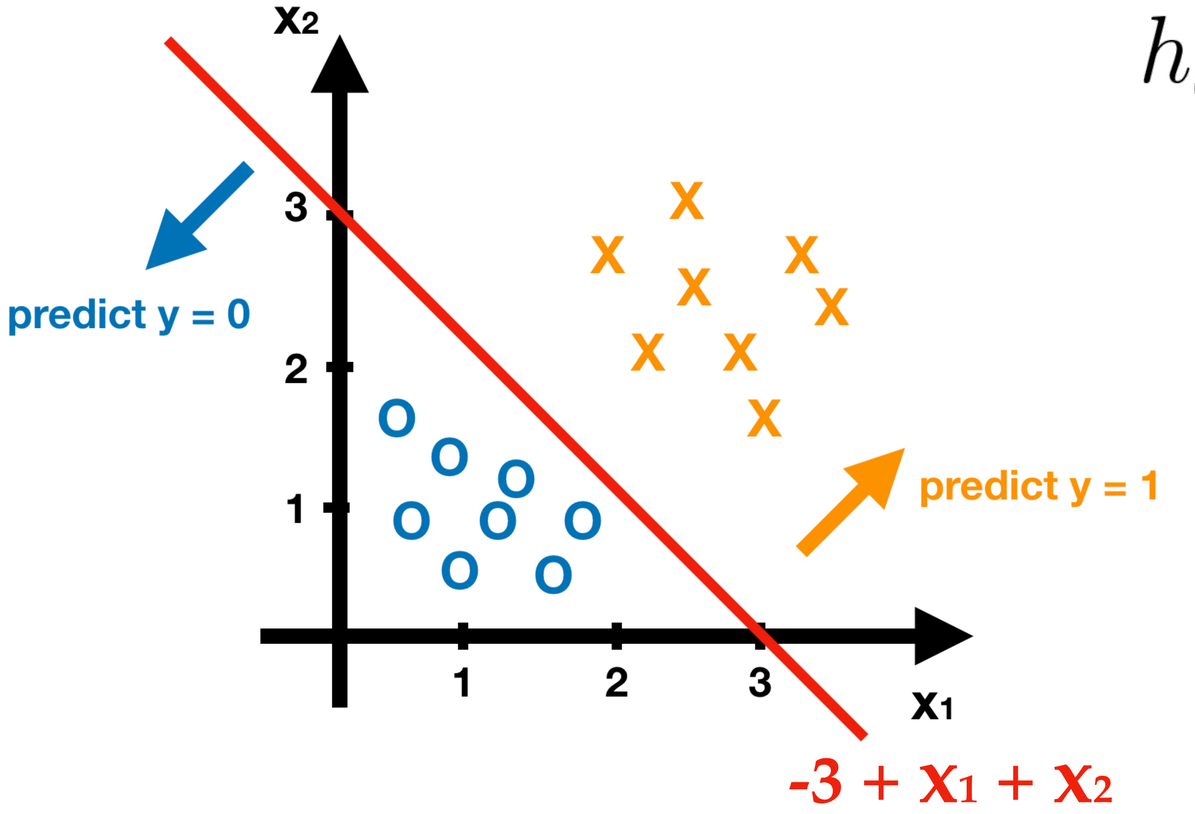
$$z < 0 \rightarrow g(z) < 0.5$$

$$h_\theta(x) \geq 0.5 \rightarrow y = 1$$

$$h_\theta(x) < 0.5 \rightarrow y = 0$$

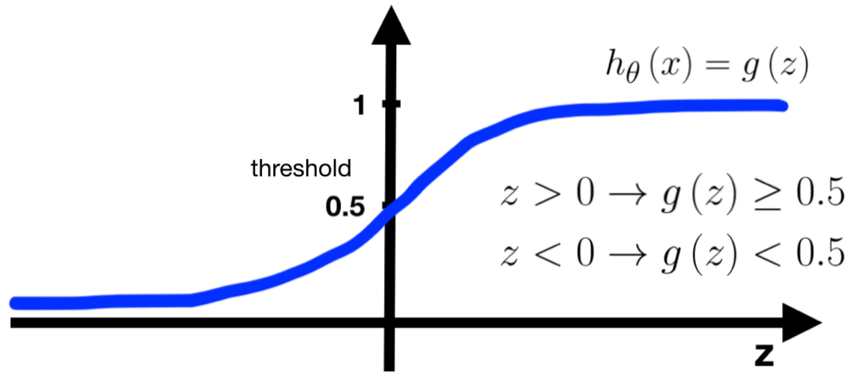
$$0 \leq h_\theta(x) \leq 1$$

Logistic Regression - Example

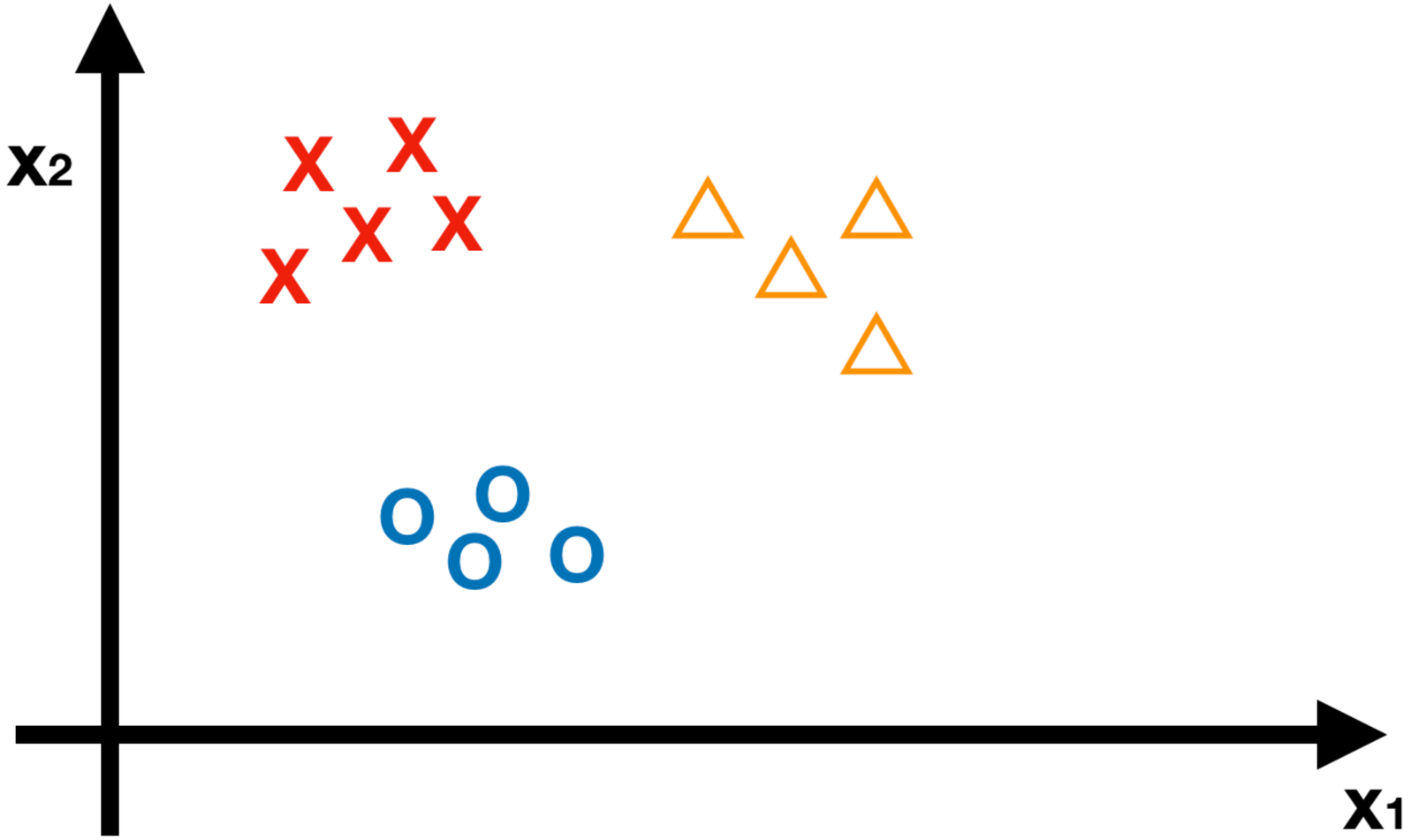


$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$$if\ -3 + x_1 + x_2 \geq 0 \rightarrow y = 1$$
$$if\ -3 + x_1 + x_2 < 0 \rightarrow y = 0$$

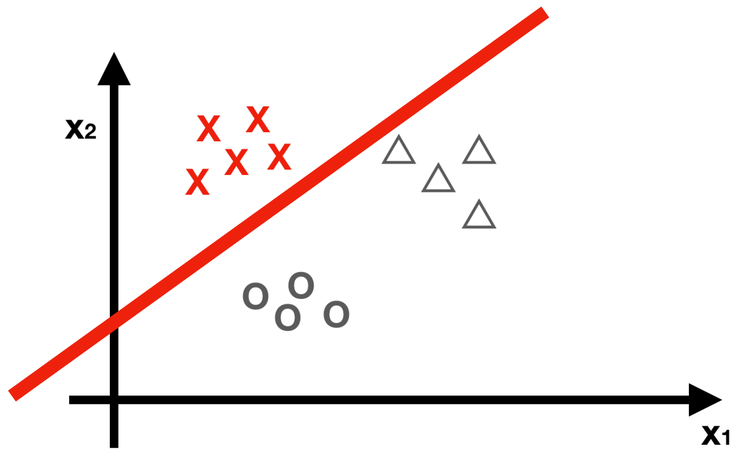


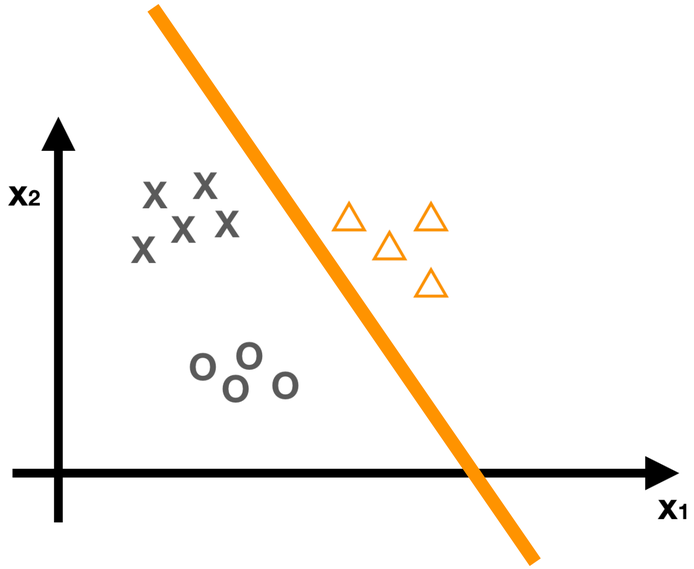
Multiclass Classification

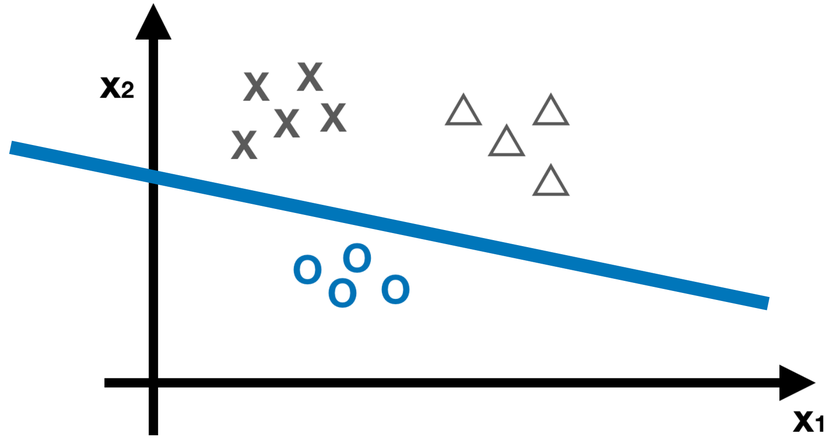


Multiclass Classification

- One-vs-all







$h_{\theta_1}(x) = P(x \in X)$ $h_{\theta_2}(x) = P(x \in \triangle)$ $h_{\theta_3}(x) = P(x \in O)$

Example - Iris(鳶尾花)

- **sklearn - load_iris, train_test_split, LogristicRegression**

Training Data

Feature: petal size, calyx size

Target: setosa, versicolor, virginica

Learning Algorithm — Logistic Regression

$$X \rightarrow h_\theta(x) \rightarrow Y$$

Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive

Example - Iris(鳶尾花)

Petal Length	Petal width	Calyx length	Calyx width	Target
5.1	3.5	1.4	0.2	0
5.8	2.7	3.9	1.2	1
6	2.7	5.1	1.6	1
6.9	3.1	5.4	2.1	2
6.7	3.1	5.6	2.4	2

target 0: setosa
target 1: versicolor
target 2: virginica

```
# 引入sklearn的內建資料集
from sklearn import datasets

# 載入鳶尾花資料集
iris = datasets.load_iris()
# iris.target_names, # target欄位名稱
# iris.target, # target欄位資料
# iris.data # 特徵資料
```

Example - Iris (鳶尾花)

- **sklearn - train_test_split package**

```
# 引入sklearn的內建資料集
from sklearn import datasets
# 引入 train_test_split 套件，用來切割資料
from sklearn.model_selection import train_test_split

# 載入鳶尾花資料集
iris = datasets.load_iris()

# train_X: 訓練集資料特徵, train_y: 訓練集 target
# test_X: 測試集資料特徵, test_y: 測試集target
train_X, test_X, train_y, test_y = train_test_split(iris.data,
                                                    iris.target,
                                                    random_state=0)
```

Example - Iris（鳶尾花）

- **sklearn - LogristicRegression**

```python
from sklearn import datasets
from sklearn.model_selection import train_test_split
# 引入 LogisticRegression 套件，用來 fit model
from sklearn.linear_model import LogisticRegression

iris = datasets.load_iris()
train_X, test_X, train_y, test_y = train_test_split(iris.data,
                                                    iris.target,
                                                    random_state=0)

clf = LogisticRegression() # 建立 LogisticRegression 方法
clf.fit(train_X, train_y) # fit model
y_predict = clf.predict(test_X) # 使用測試集資料，來預測 target

score = clf.score(test_X, test_y) # 衡量 model 準確度
print(score) # 0.86
```

Exercise - wine (酒)

- **sklearn - load_wine, train_test_split, LogristicRegression**

Training Data

Feature: alcohol, malic_acid, ash, … , proline.

Target: class0, class1, class2

Learning Algorithm — Logistic Regression

$$X \rightarrow h_\theta(x) \rightarrow Y$$

Classes	3
Samples per class	[59,71,48]
Samples total	178
Dimensionality	13
Features	real, positive

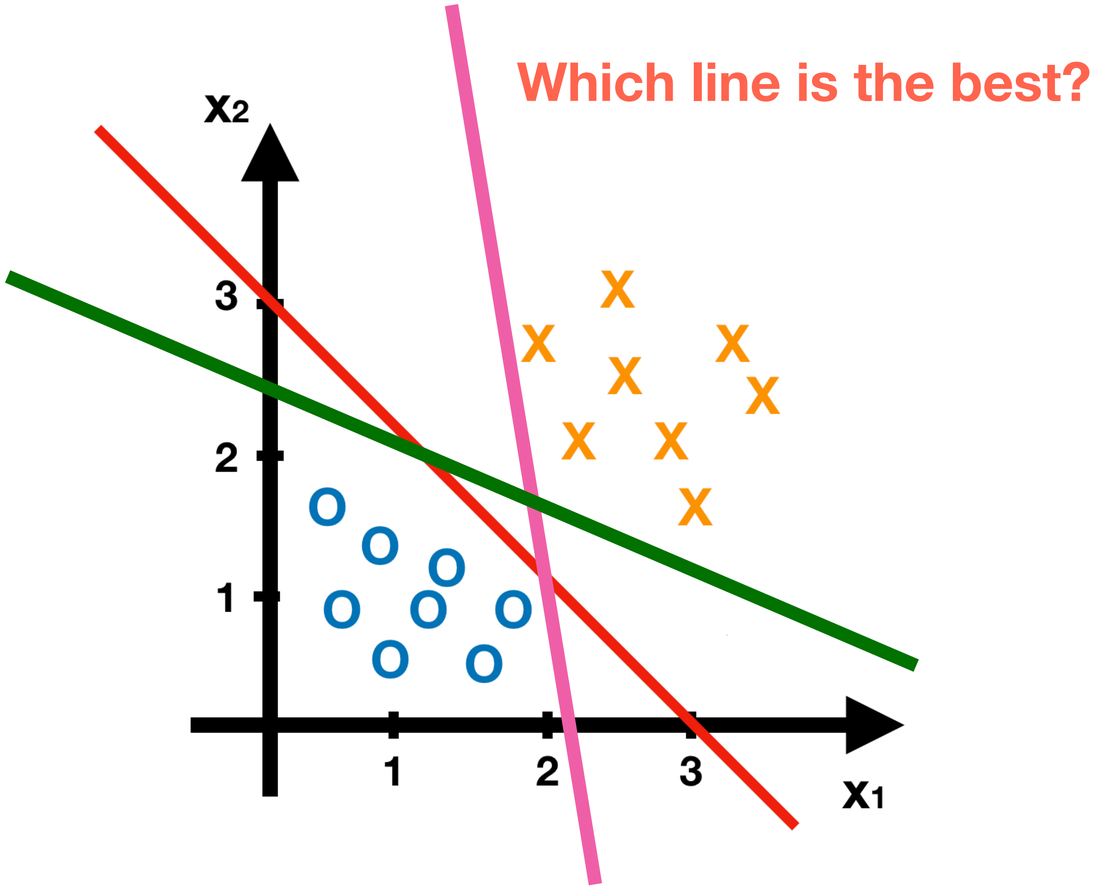
Exercise - wine(酒)

alcohol	malic_acid	….	proline	target
1.32e+01	1.78e+00	….	1.065e+03	0
1.229e+01	1.410e+00	….	4.280e+02	1
1.413e+01	4.100e+00	….	1.600e+00	2

target 0: class 0
target 1: class 1
target 2: class 2

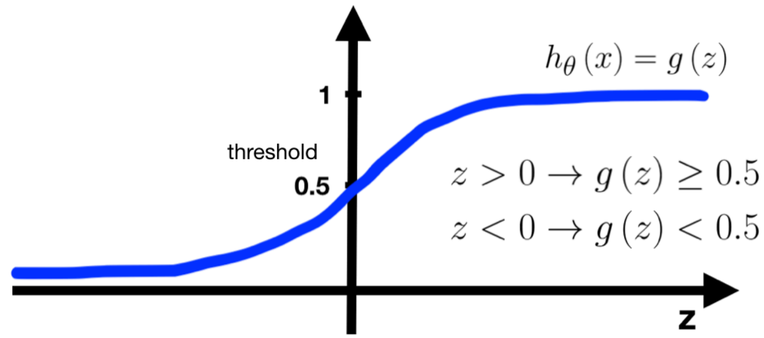
Cost Function - Logistic Regression (1)

- Measuring the accuracy of $h_\theta(x)$



$$h_\theta(x) = g(z) = \frac{1}{1+e^{-z}}$$

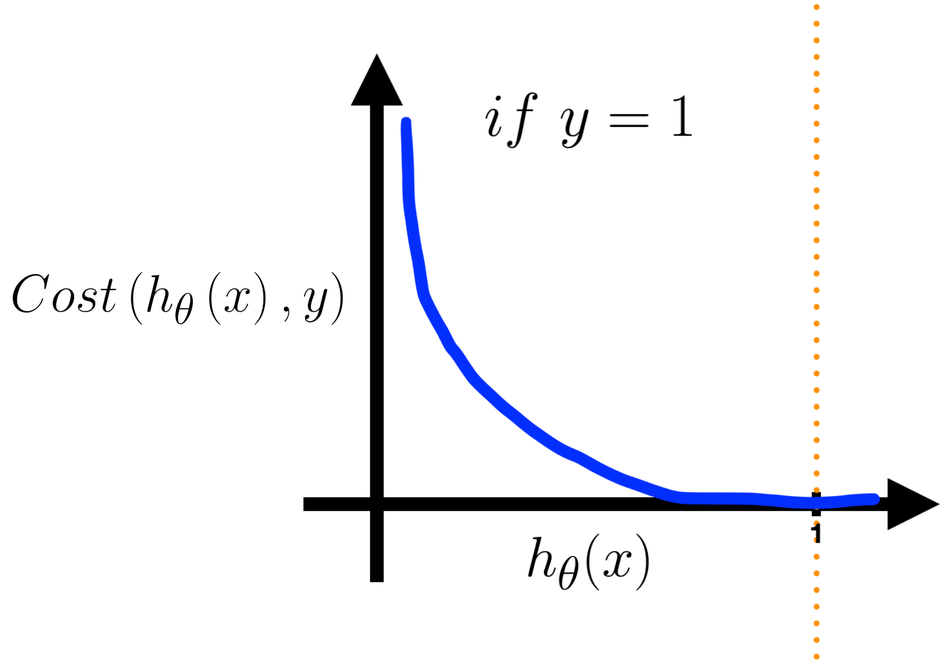
$$(z = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n = \theta^T x)$$

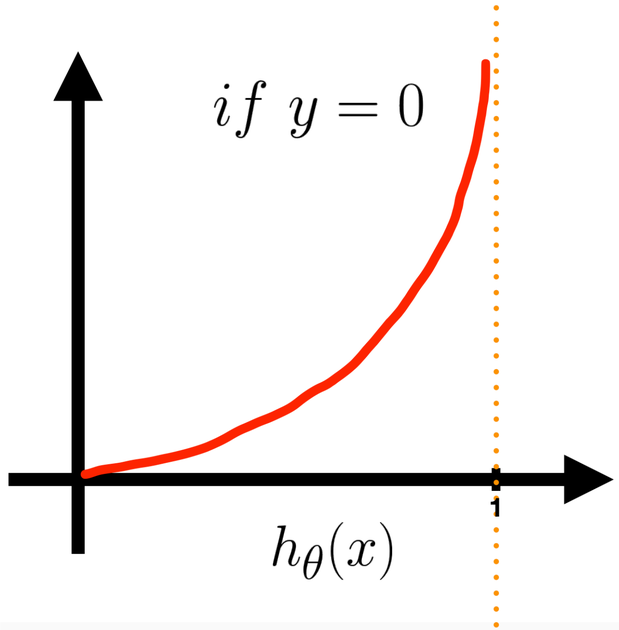


Cost Function - Logistic Regression (2)

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

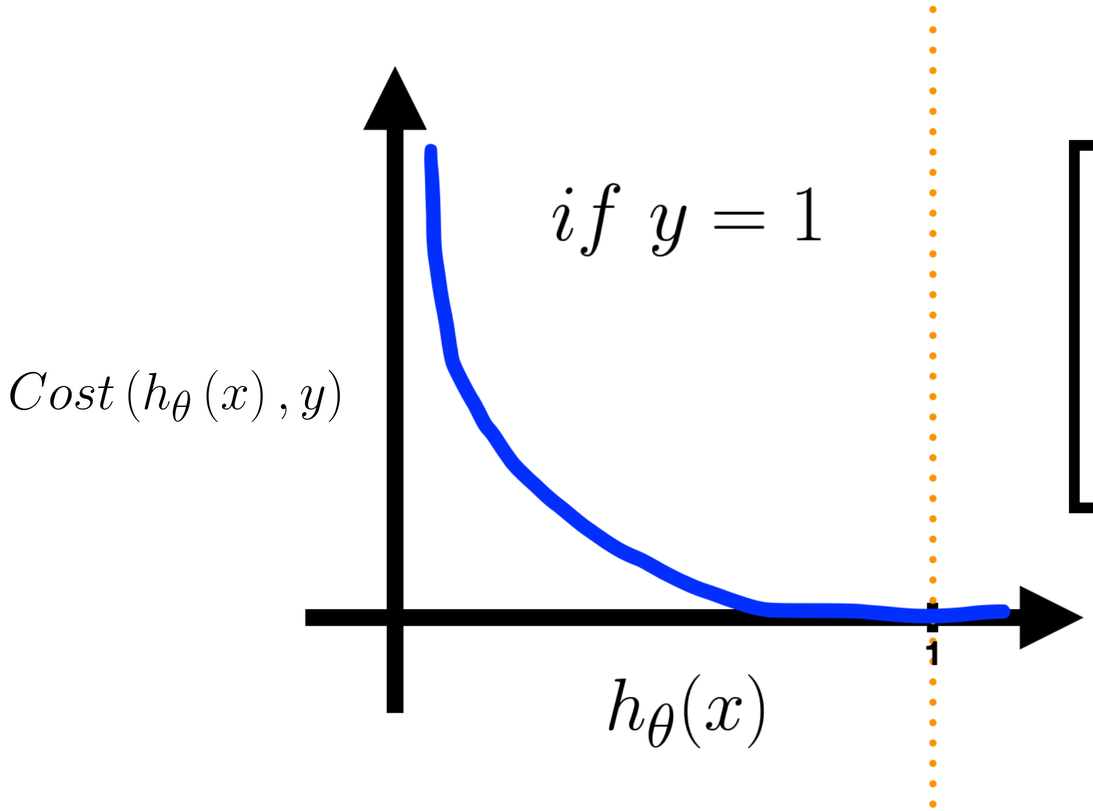
$$Cost\left(h_\theta(x), y\right) = \begin{cases} -log\left(h_\theta(x)\right) & if\ y = 1 \\ -log\left(1 - (h_\theta(x)\right) & if\ y = 0 \end{cases}$$





Cost Function - Logistic Regression (3)

$$Cost\left(h_\theta\left(x\right), y\right) = \begin{cases} -log\left(h_\theta\left(x\right)\right) & if\ y = 1 \\ -log\left(1 - \left(h_\theta(x)\right)\right) & if\ y = 0 \end{cases}$$

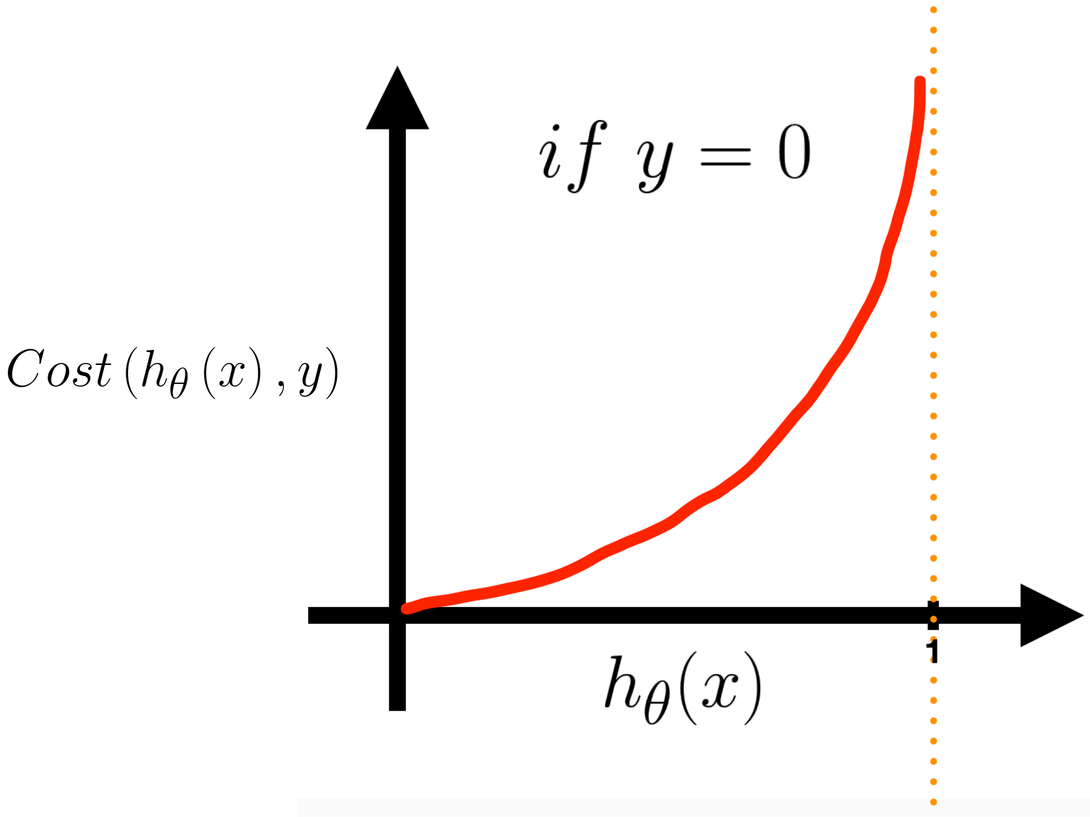


$$cost \to \infty\ when\ h_\theta\left(x\right) = 0$$

$$cost = 0\ \ if\ y = 1, h_\theta\left(x\right) = 1$$

Cost Function - Logistic Regression (4)

$$Cost\left(h_\theta\left(x\right),y\right) = \begin{cases} -log\left(h_\theta\left(x\right)\right) & if\ y=1 \\ -log\left(1-\left(h_\theta(x)\right)\right) & if\ y=0 \end{cases}$$



$$cost \to \infty\ when\ h_\theta\left(x\right)=1$$

$$cost=0\ \ if\ y=0, h_\theta\left(x\right)=0$$

MINIMIZE The Cost Function by using Gradient Descent

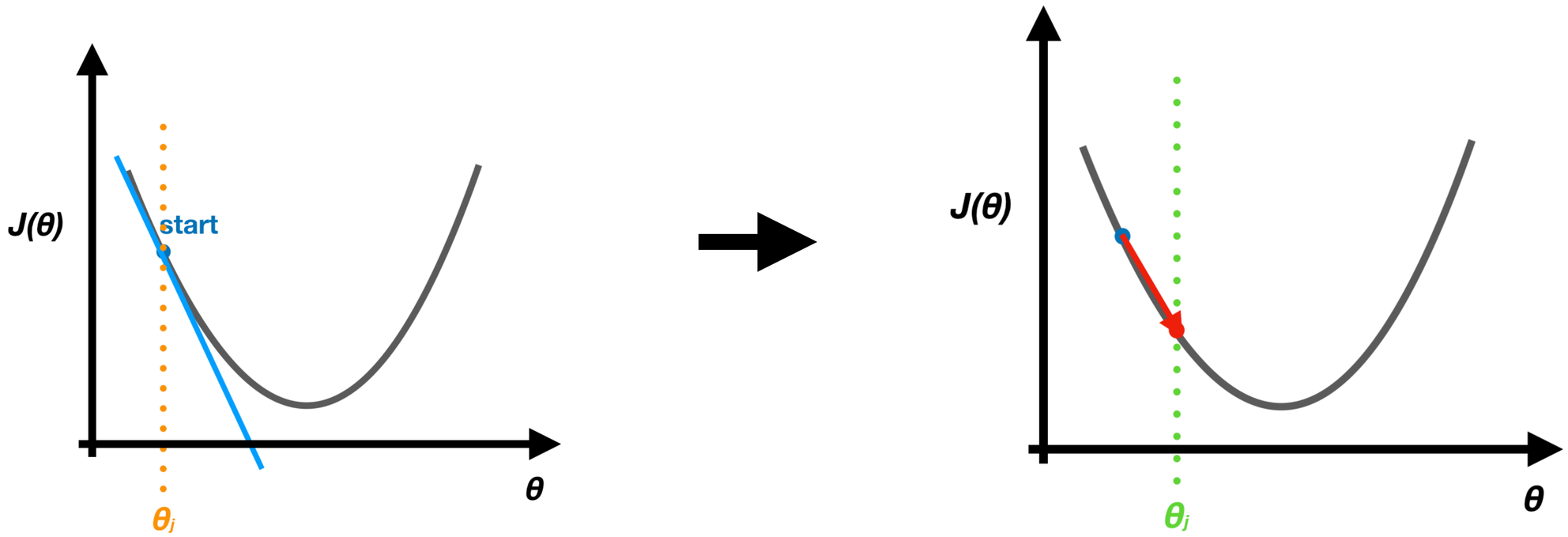
Gradient Descent

Repeat until converge {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

negative in following figure

} where j represents the feature index number.



Gradient Descent

- **Simultaneously** updating parameters θ at each iteration.

Repeat until converge {

$$tmp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$tmp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := tmp0$$

$$\theta_1 := tmp1$$

}

Repeat until converge {

$$tmp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\theta_0 := tmp0$$

$$tmp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_1 := tmp1$$

}

Gradient Descent

- **Simultaneously** updating parameters θ at each iteration.

Repeat until converge {

$$tmp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$tmp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := tmp0$$

$$\theta_1 := tmp1$$

}

Correct !!

Repeat until converge {

$$tmp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\theta_0 := tmp0$$

$$tmp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_1 := tmp1$$

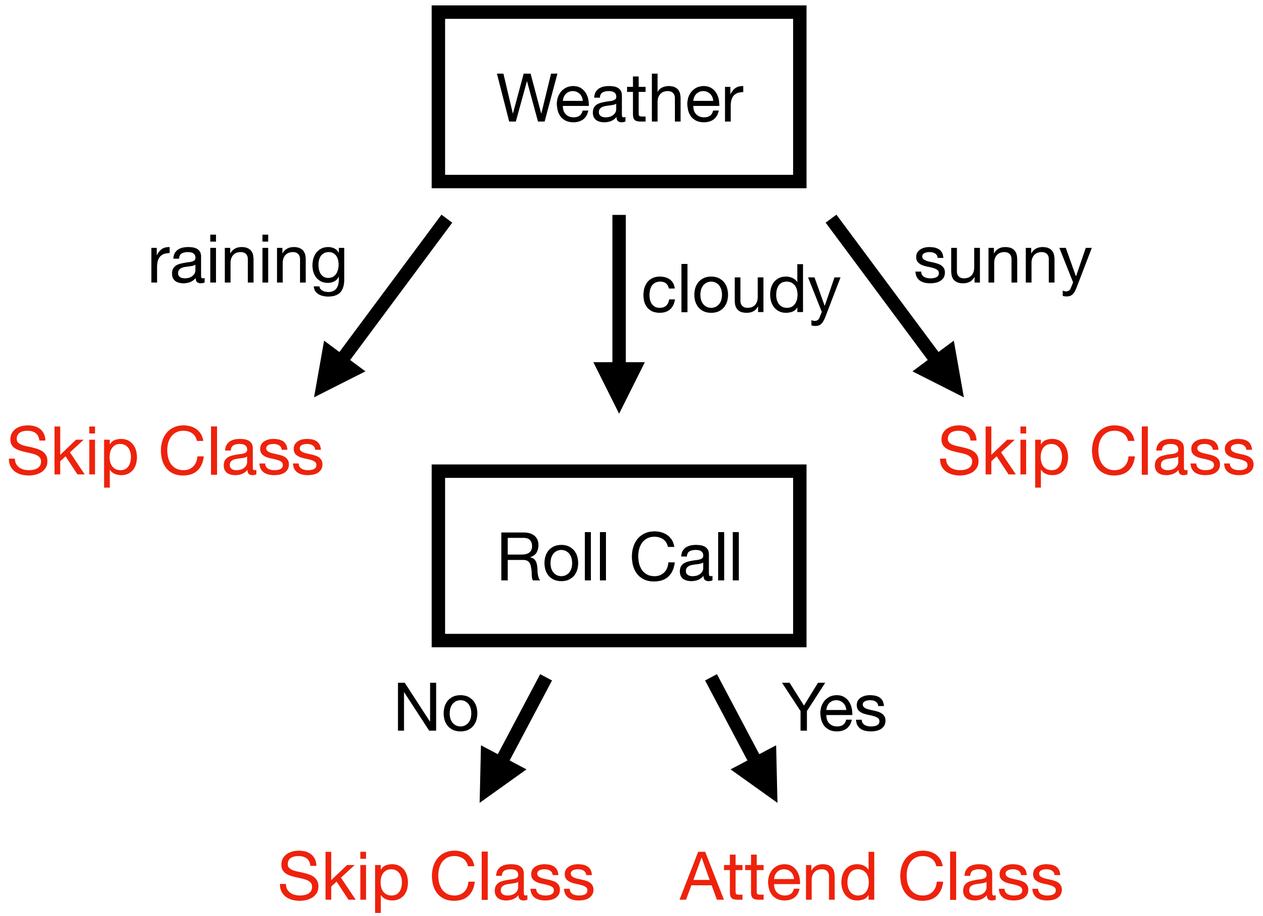
}

Other Classification Method

- Decision Tree
- Random Decision Forests
- K-NearestNeighbor (KNN)
- Support Vector Machine (SVM)

Decision Tree

- Building classification model in the form of tree structure



Feature: Weather, Roll Call

Target: Skip Class, Attend Class

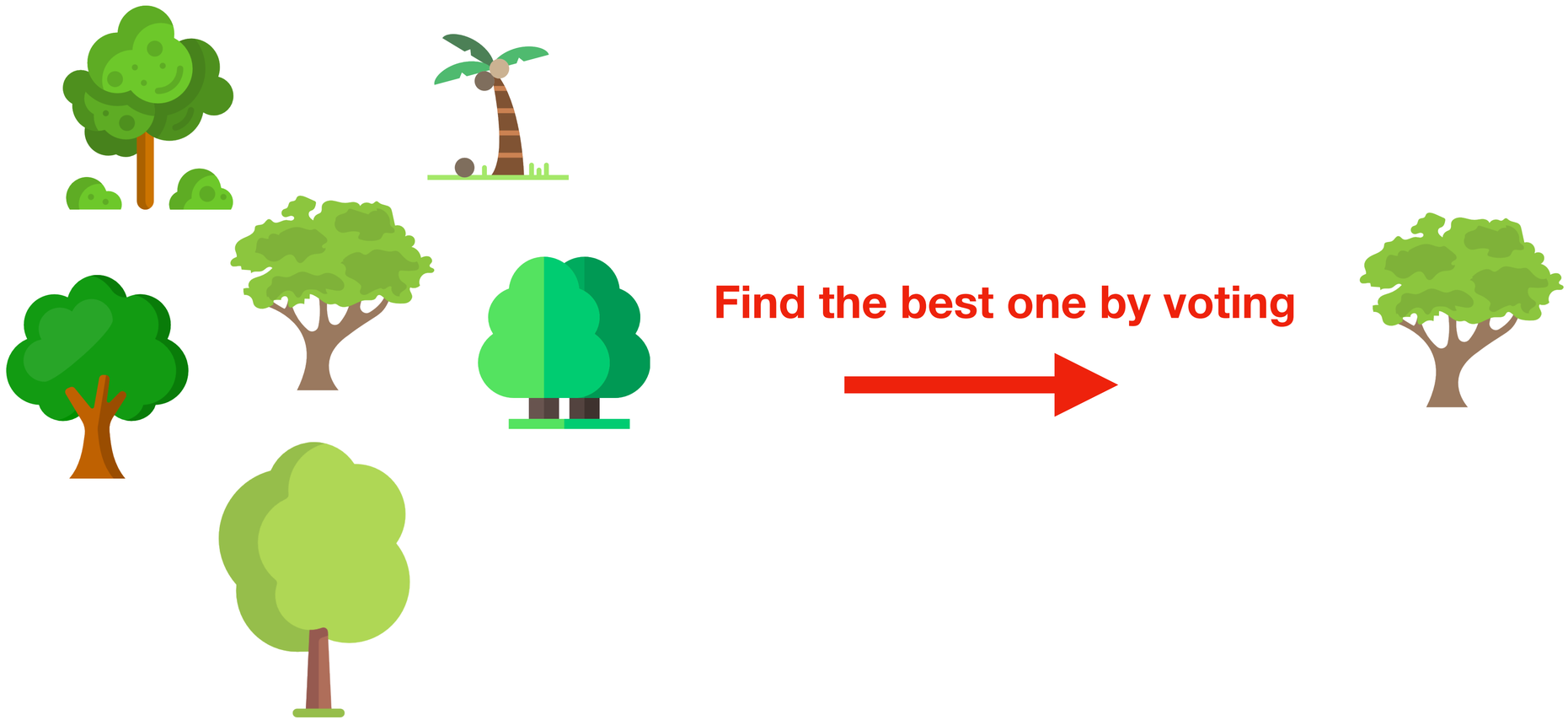
Decision Tree Example - Iris (鳶尾花)

- **sklearn - DecisionTreeClassifier**

```
1  from sklearn import datasets
2  from sklearn.model_selection import train_test_split
3  # 引入 DecisionTreeClassifier 套件，用來 fit model
4  from sklearn.tree import DecisionTreeClassifier
5
6  iris = datasets.load_iris()
7  train_X, test_X, train_y, test_y = train_test_split(iris.data,
8                                                      iris.target,
9                                                      random_state=0)
10
11 clf = DecisionTreeClassifier()
12 clf.fit(train_X, train_y) # fit model
13 y_predict = clf.predict(test_X) # 使用測試集資料，來預測 target
14
15 score = clf.score(test_X, test_y) # 衡量 model 準確度
16 print(score) # 0.86
```

Random Decision Forests

- Building multiple trees in randomly selected subspaces of the feature space



Classification Method - Random Decision Forests

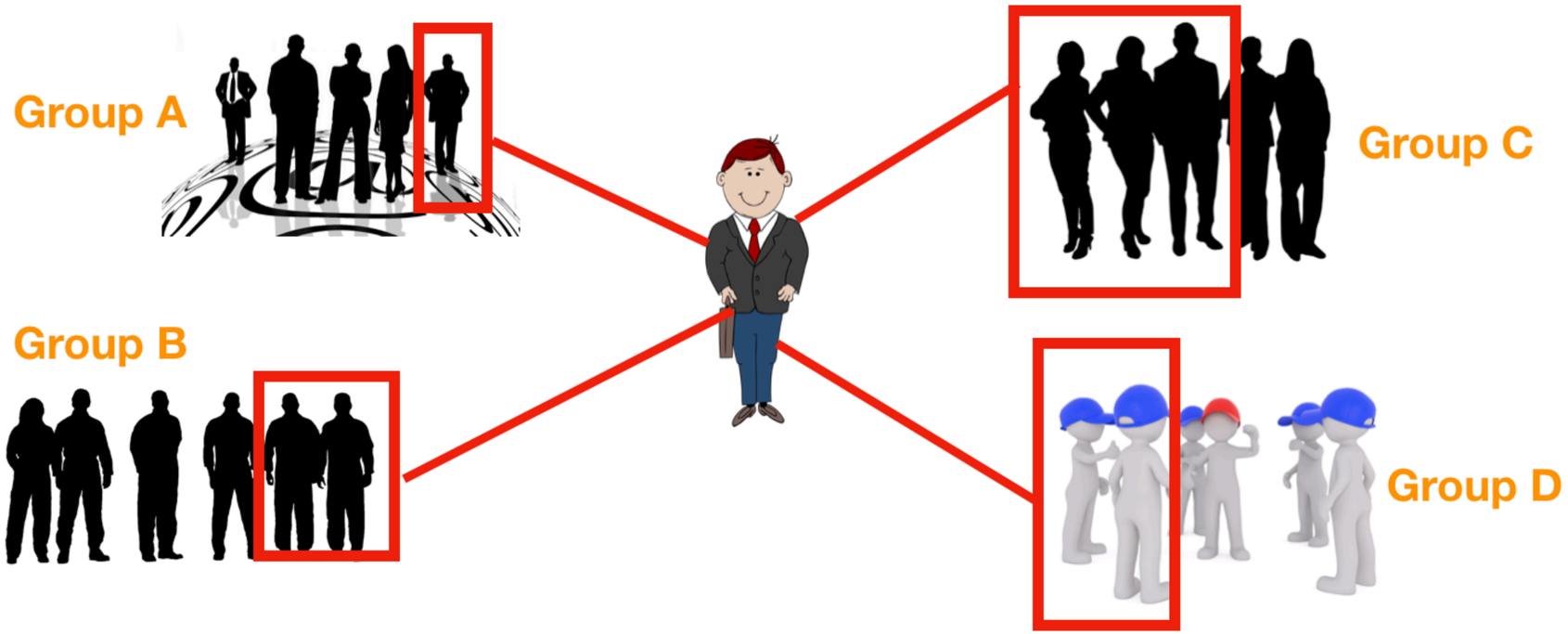
- **sklearn - RandomForestClassifier**

```
1  from sklearn import datasets
2  from sklearn.model_selection import train_test_split
3  # 引入 RandomForestClassifier 套件，用來 fit model
4  from sklearn.ensemble import RandomForestClassifier
5
6  iris = datasets.load_iris()
7  train_X, test_X, train_y, test_y = train_test_split(iris.data,
8                                                      iris.target,
9                                                      random_state=0)
10
11 clf = RandomForestClassifier()
12 clf.fit(train_X, train_y) # fit model
13 predict_y = clf.predict(test_X) # 使用測試集資料，來預測 target
14
15 score = clf.score(test_X, test_y) # 衡量 model 準確度
16 print(score) # 0.97
```

K-NearestNeighbor (KNN)

- Classifying a data point into one group by its k-nearest neighbors

Example: suppose k=8, which group does this man belong to?

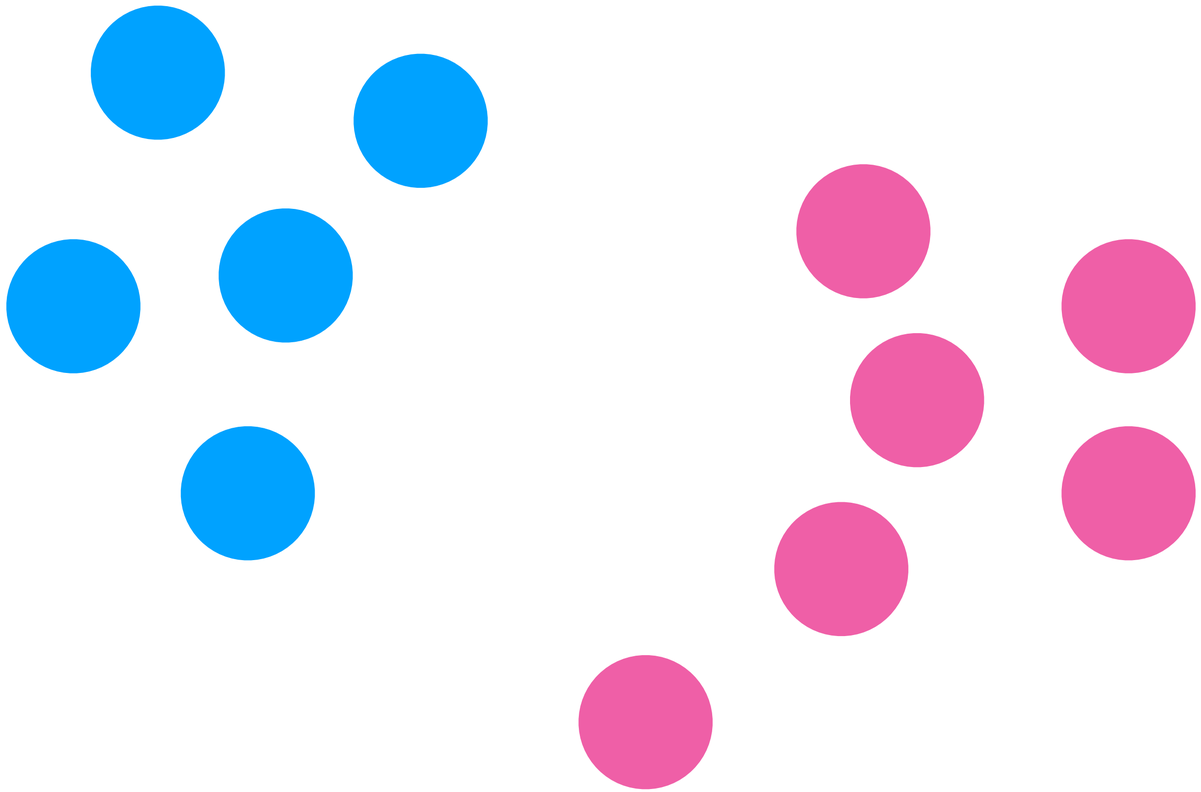


Classification Method - KNN

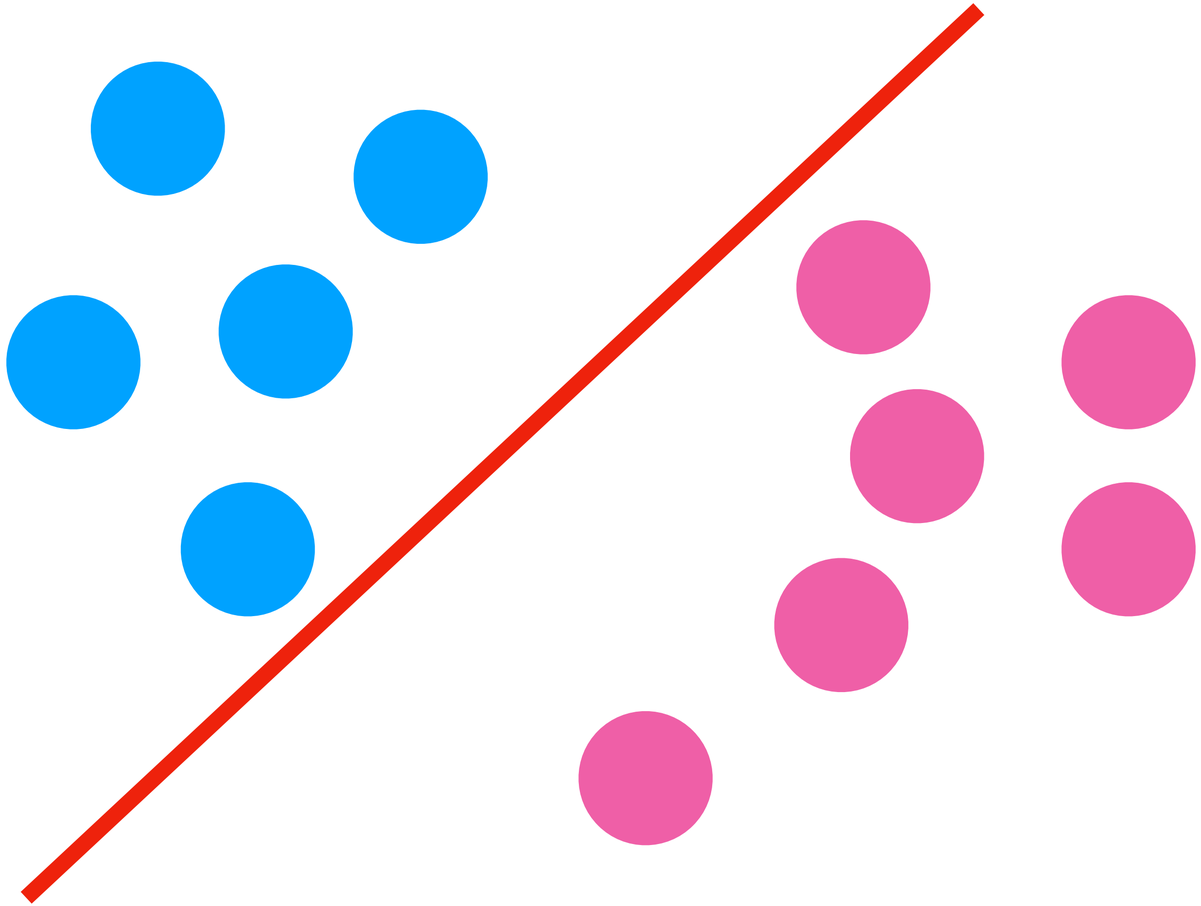
- **sklearn - KNeighborsClassifier**

```
1  from sklearn import datasets
2  from sklearn.model_selection import train_test_split
3  # 引入 KNeighborsClassifier 套件，用來 fit model
4  from sklearn.neighbors import KNeighborsClassifier
5
6  iris = datasets.load_iris()
7  train_X, test_X, train_y, test_y = train_test_split(iris.data,
8                                                      iris.target,
9                                                      random_state=0)
10
11 clf = KNeighborsClassifier()
12 clf.fit(train_X, train_y) # fit model
13 y_predict = clf.predict(test_X) # 使用測試集資料，來預測 target
14
15 score = clf.score(test_X, test_y) # 衡量 model 準確度
16 print(score) # 0.97
```

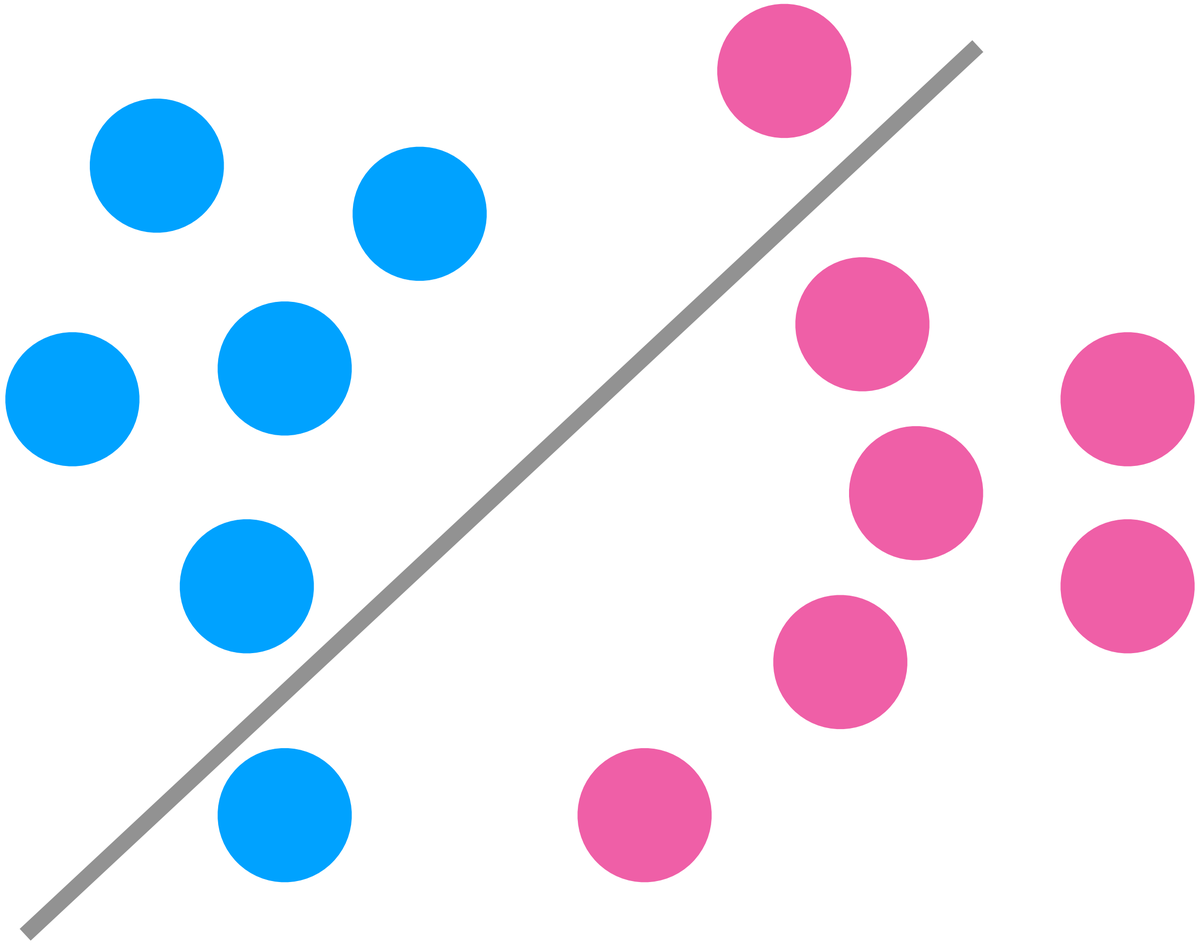
Support Vector Machine (SVM)



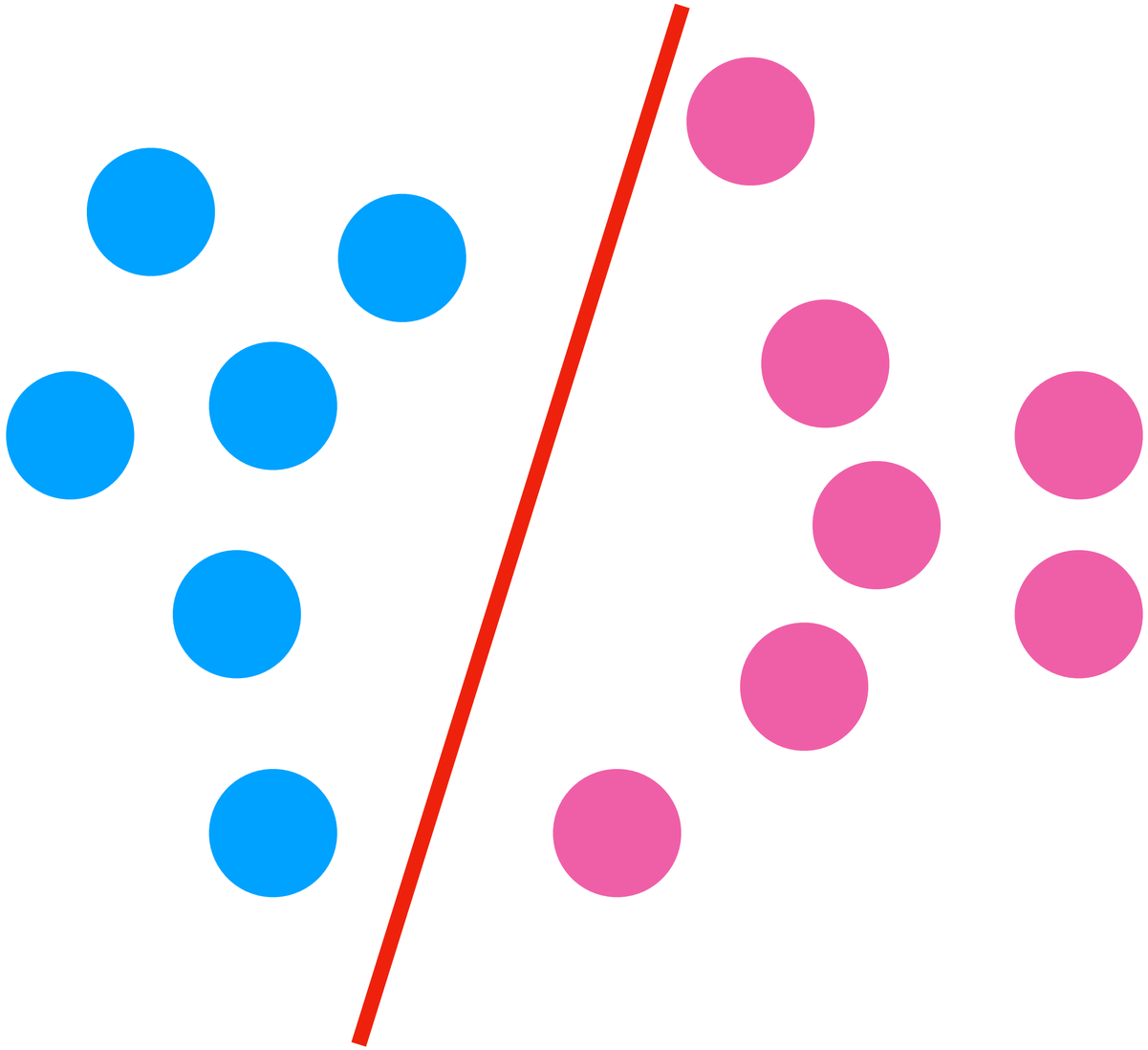
Support Vector Machine (SVM)



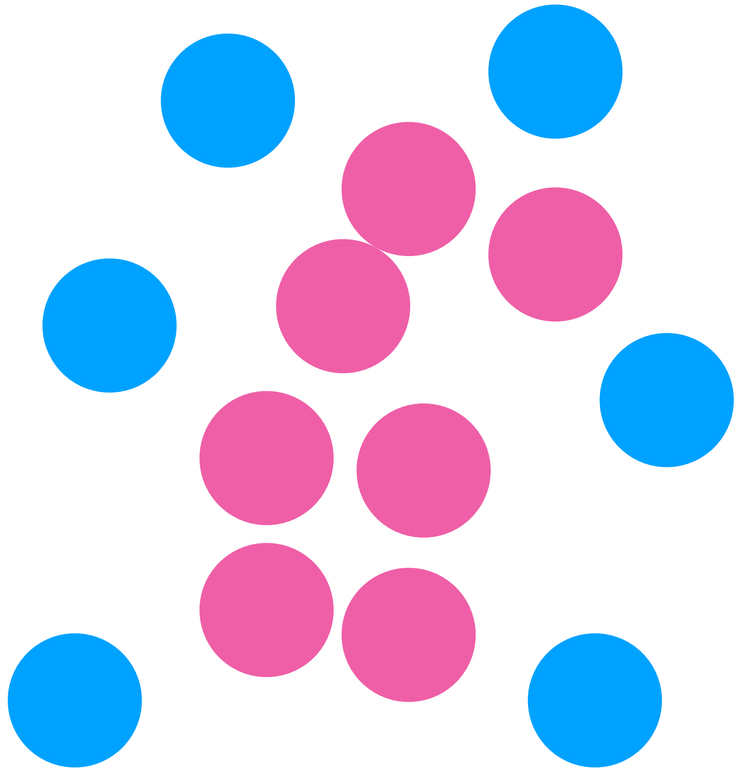
Support Vector Machine (SVM)

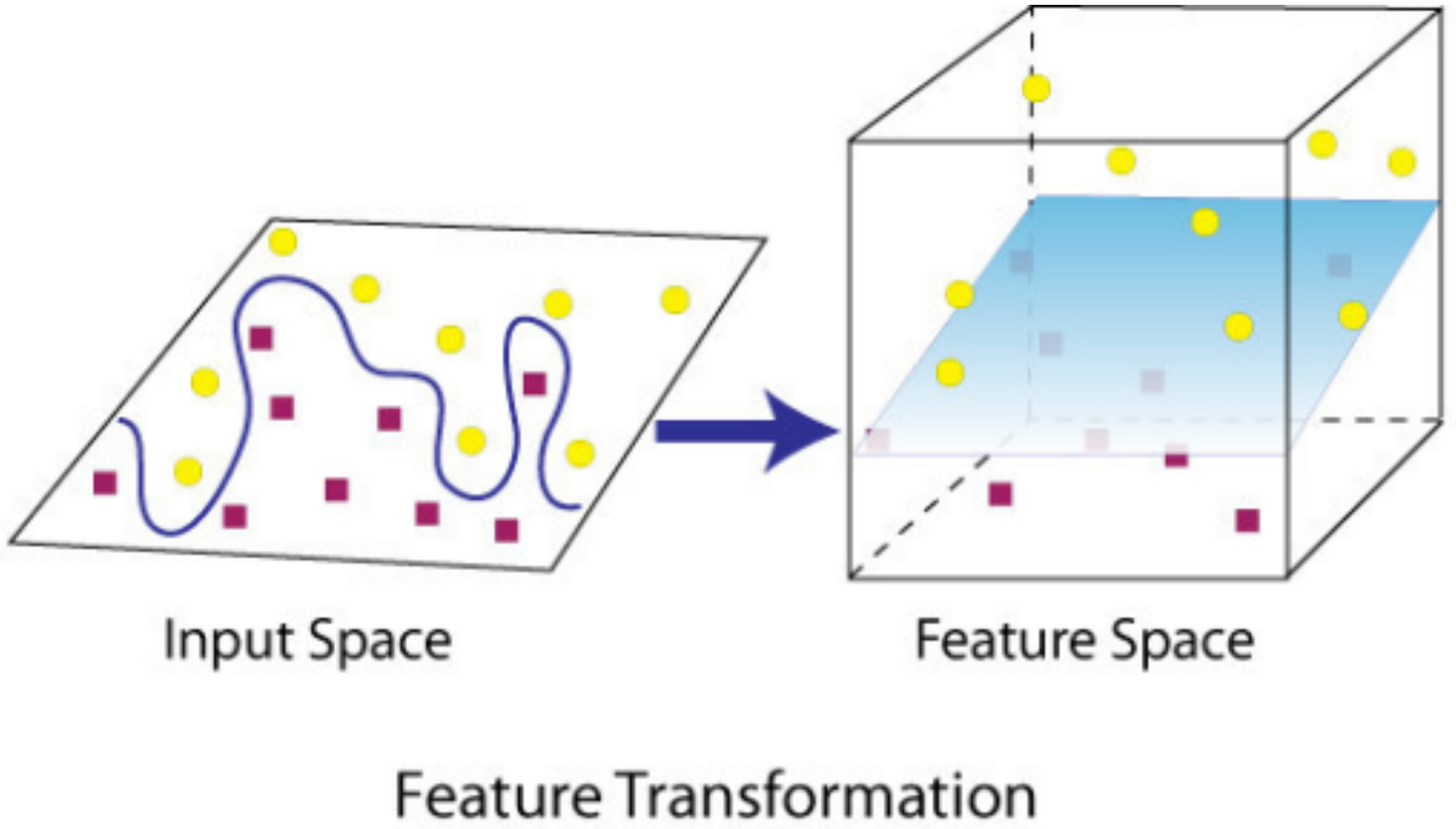


Support Vector Machine (SVM)



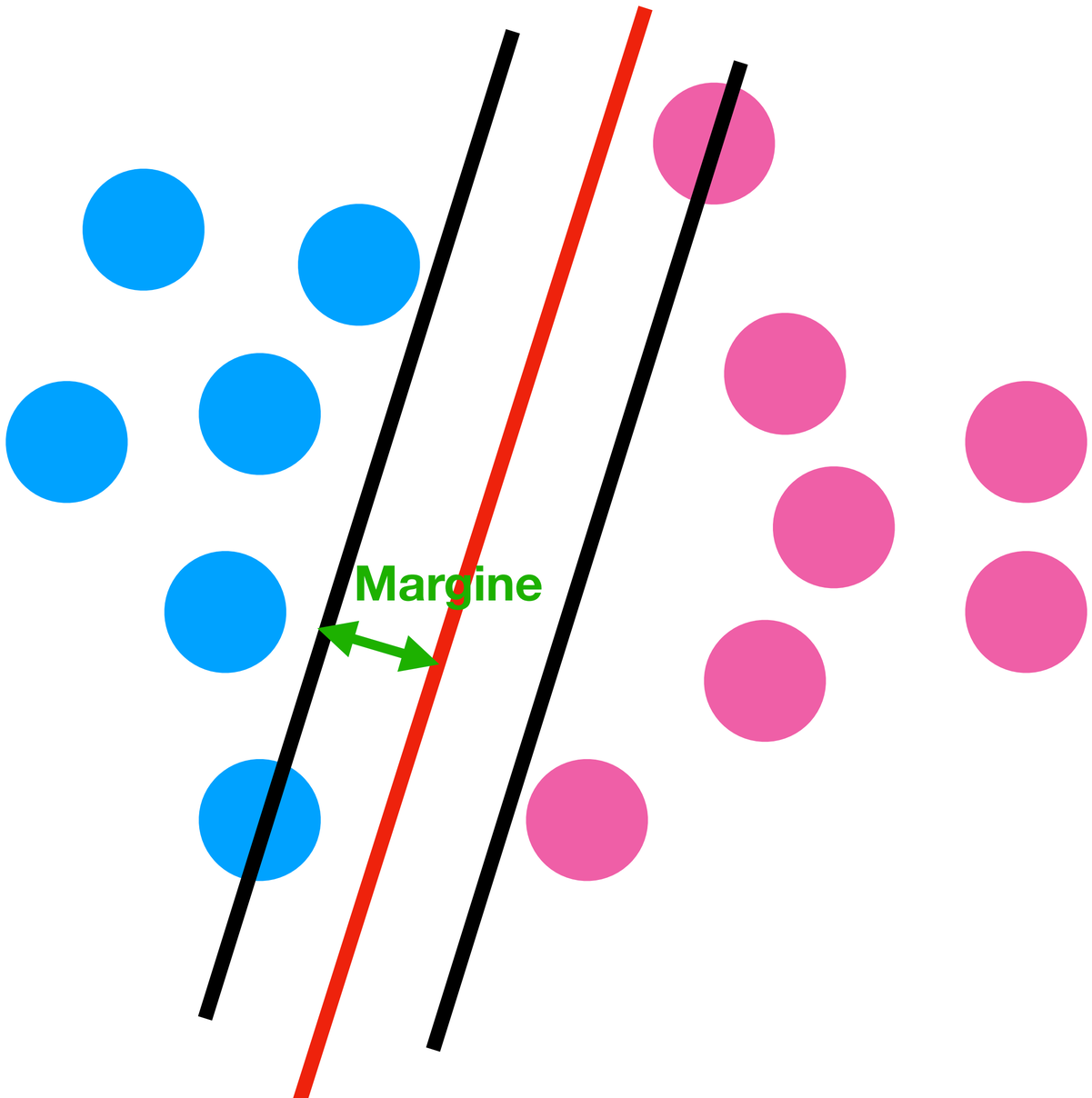
Support Vector Machine (SVM)





https://openi.nlm.nih.gov/detailedresult.php?img=PMC2850872_1472-6947-10-16-1&req=4

Support Vector Machine (SVM)



Classification Method - SVM

- **sklearn - SVM**

```python
from sklearn import datasets
from sklearn.model_selection import train_test_split
# 引入 SVM 套件，用來 fit model
from sklearn import svm

iris = datasets.load_iris()
train_X, test_X, train_y, test_y = train_test_split(iris.data,
                                                    iris.target,
                                                    random_state=0)

clf = svm.SVC()
clf.fit(train_X, train_y) # fit model
predict_y = clf.predict(test_X) # 使用測試集資料，來預測 target

score = clf.score(test_X, test_y) # 衡量 model 準確度
print(score) # 0.97
```

實務練習 - 自然語言處理

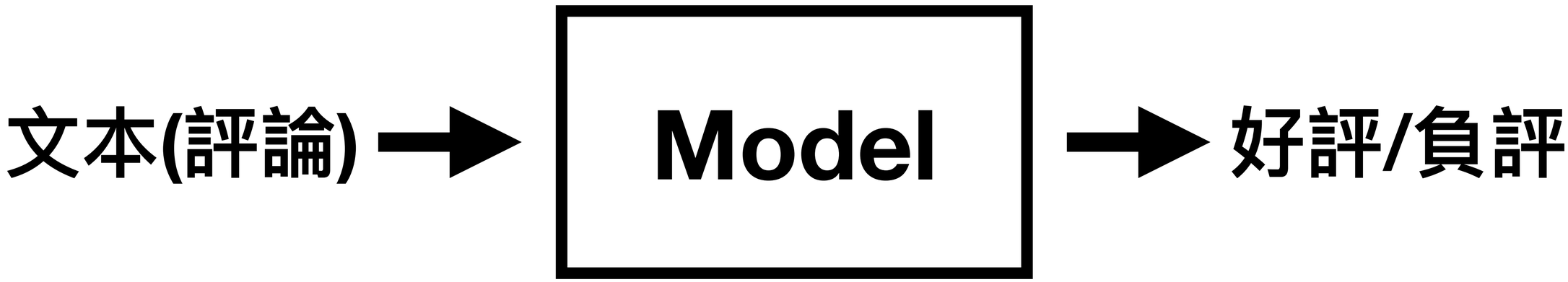
資料格式

- 資料下載

- 資料標題: sentence(評論), category(1: 好評， -1: 差評)

sentence	category
This is a very cool product. It was extremely easy to install and took only seconds…	1
I was pissed when I received the collar. I searched high and low trying to determine how many collars I would be getting in my purchase and I was never able to find that…	-1

目標



Logistic Regression

第一步 - 資料處理

- 移除文本中無意義的單字
- 移除不必要的符號，將文本分離成一個個的單字 - Tokenization
- 移除文本中不相關的字 - tag, url
- 英文字母大小寫轉換成一致 - hello, Hello, HELLO
- 考量詞性還原 - am, is, are 都是 be 動詞

第二步- 文本編碼

- 詞袋模型
- D1: 'Dog is black’
- D2: 'Sky is blue'
- D3: 'Dog is dancing'

提示 - CountVectorizer

詞彙字典

語句	black	blue	dancing	dog	is	sky
D1	1	0	0	1	1	0
D2	0	1	0	0	1	1
D3	0	0	1	1	1	0

第二步 - 文本編碼 (補充)

- TF-IDF
- CNN

第三步，訓練model

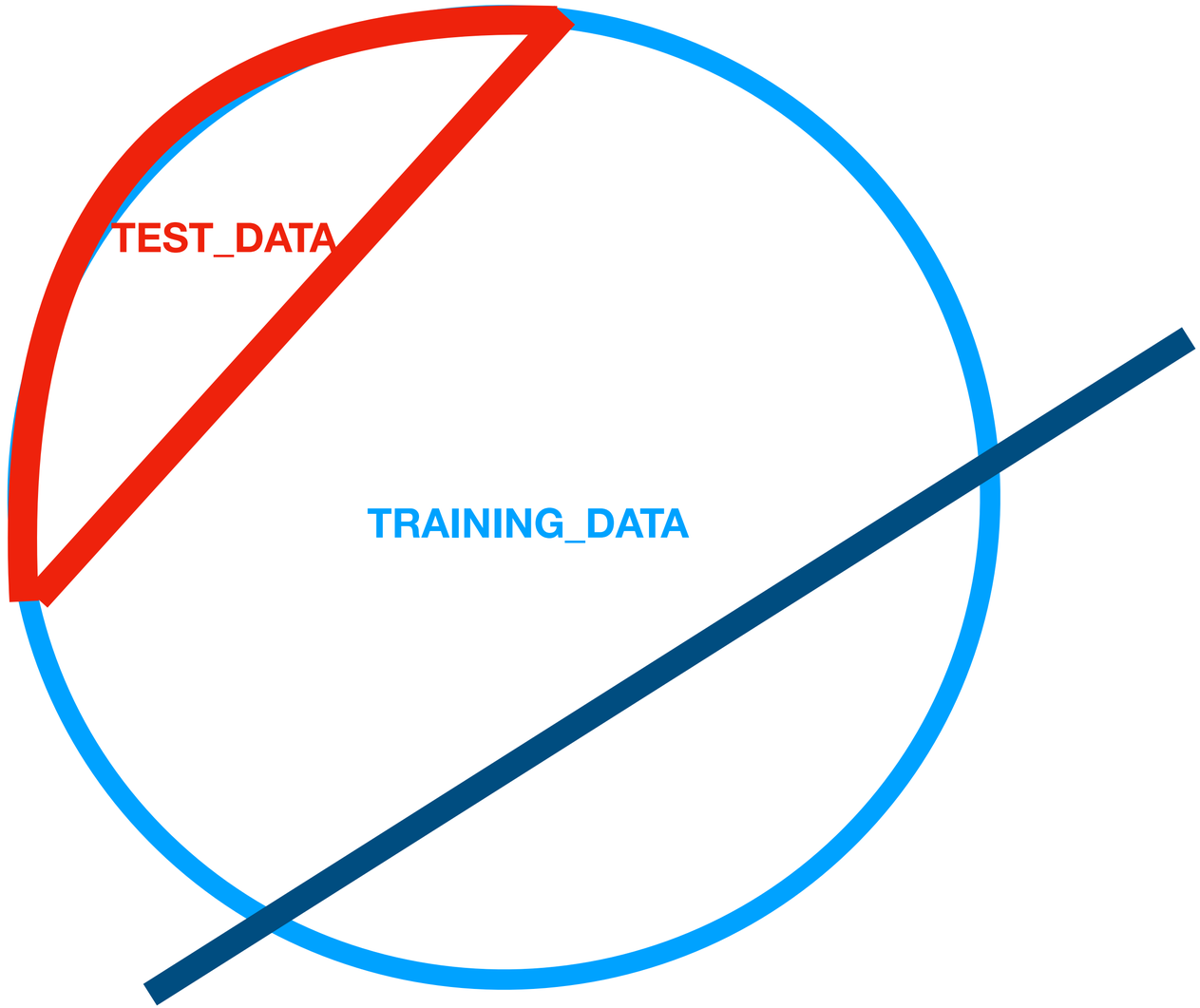
Recall: example-鳶尾花

	black	blue	dog	is	category
D1	1	0	0	0	1
D2	1	0	0	1	-1
D3	1	1	1	1	1
D4	0	0	1	0	1
D5	1	1	0	1	?

train_data

test_data

提示 - train_test_split



程式要求

- 程式語言 - python3
- 資料處理/編碼 - 任何你需要的方法
- 使用 train_data 來訓練 model, 應用在 test_data 上
- 需使用 Linear Regression 來訓練 model
- test_data 的答案及參考範例程式之後會公佈