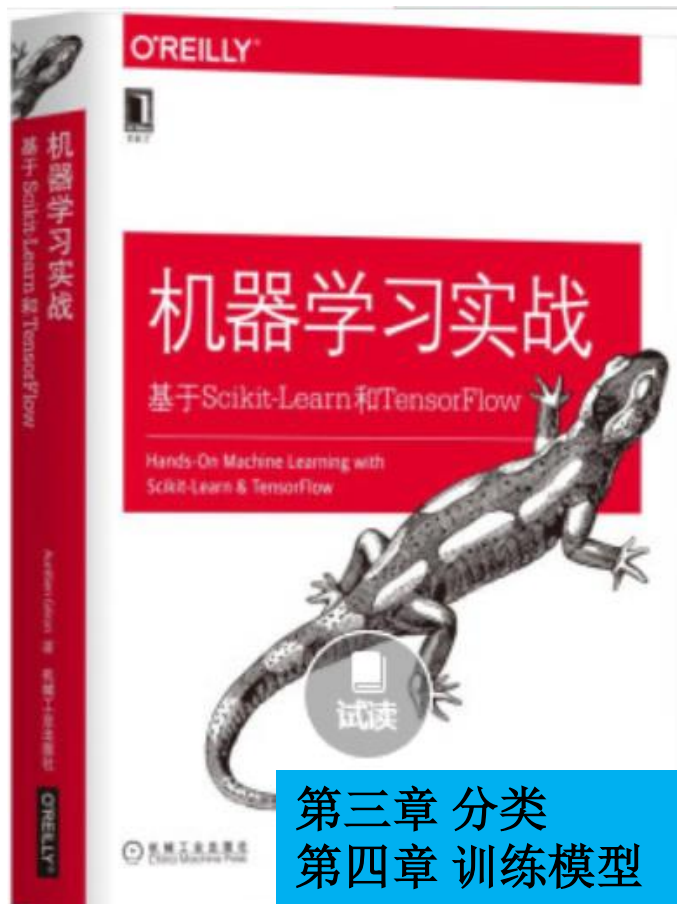


# 分类 (Classification)

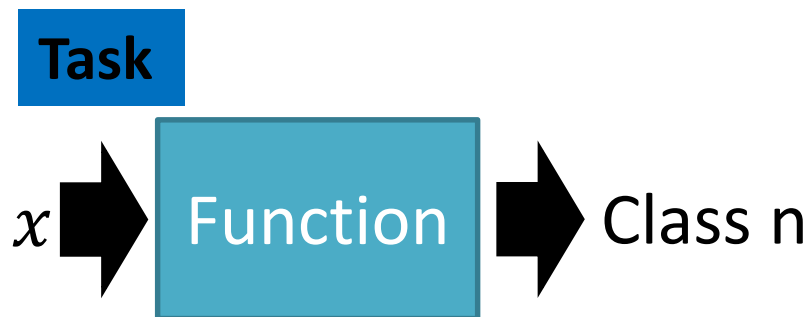
# 参考资料




教材代码: <https://github.com/ageron/handson-ml>

Scikit-learn的模型评估: <https://sklearn.apachecn.org/docs/0.21.3/32.html>

# 分类任务



- Credit Scoring
  - Input: income, savings, profession, age, past financial history .....
  - Output: accept or refuse
- Medical Diagnosis
  - Input: current symptoms, age, gender, past medical history .....
  - Output: which kind of diseases
- Handwritten character recognition    Input:     output: 金
- Face recognition
  - Input: image of a face, output: person

# 本章内容

- 为什么不直接利用线性回归模型来解决分类问题？
- 二分类问题中，逻辑回归模型训练“三步曲”
  - 分类问题中，为什么使用交叉熵（cross entropy）而不是MSE作为损失函数？
- 多分类问题中，使用Softmax回归模型
  - 对比逻辑回归模型，比较异同
- 分类器的性能评估
  - 混淆矩阵，精度和召回率，PRC，ROC
- 利用混淆矩阵进行错误分析
- 多标签分类问题
- 多输出分类问题
- 延伸过渡：
  - 层叠的逻辑回归模型 -> MLP -> DNN



重点

# 分类任务的两个经典数据集(1)

- MNIST数据集

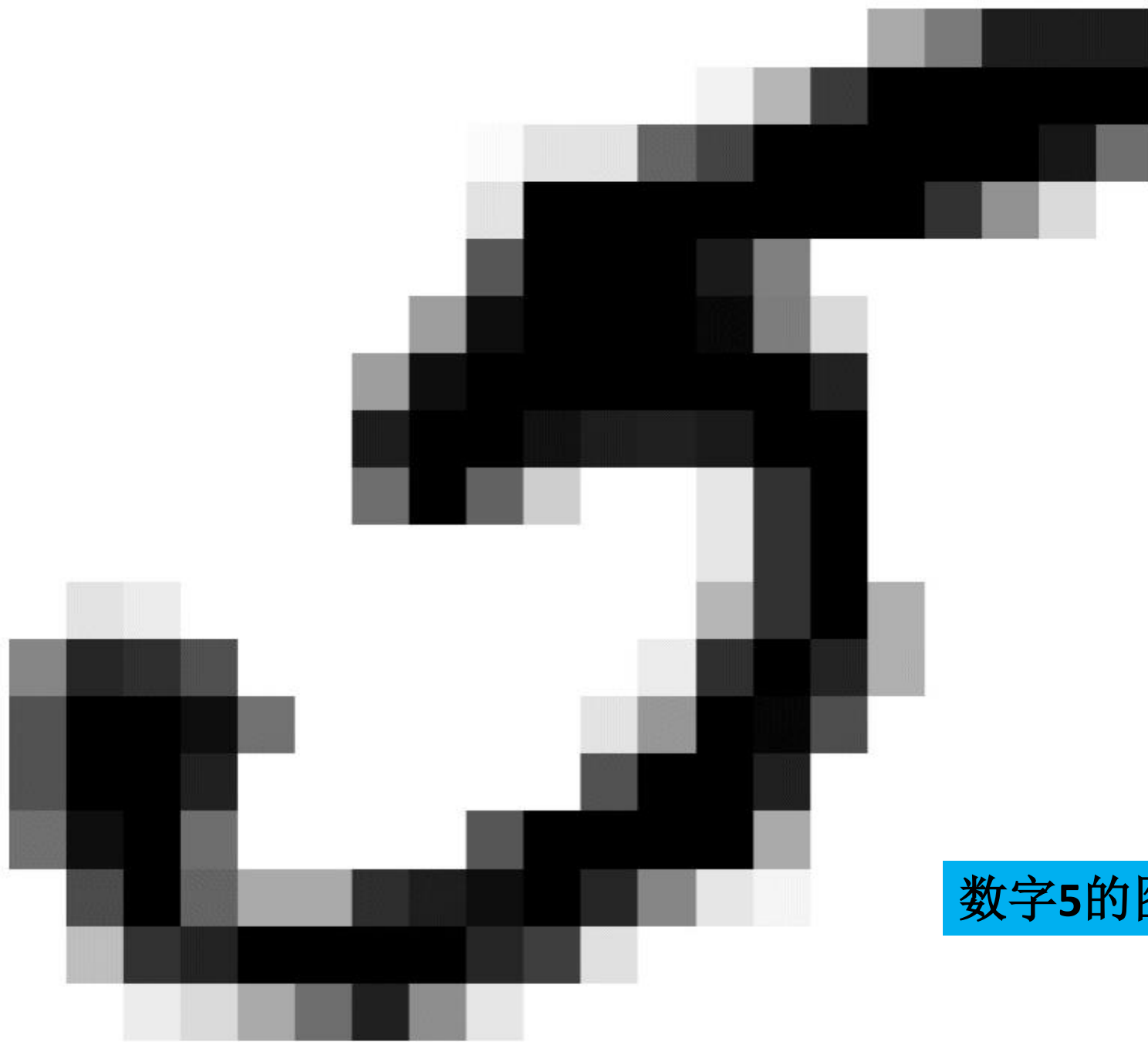
- 包含70000张尺寸较小的手写数字图片，由美国的高中生和美国人口普查局的职员手写而成。

- 每个样本，即每图片有784个特征，因为每个图片都是28\*28像素的
    - 每个特征对应一个介于0~255之间的像素值

```
>>> X, y = mnist["data"], mnist["target"]
>>> X.shape
(70000, 784)
>>> y.shape
(70000, )
```

- 机器学习中的“HelloWorld”

[illegible]



数字5的图片

# 分类任务的两个经典数据集(2)

- **Iris (鸢尾花) Data Set**

- Is perhaps the best known database to be found in the pattern recognition literature.
- Fisher's paper “The use of multiple measurements in taxonomic problems” is a classic in the field and is referenced frequently to this day.
- Contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.



# 三类Iris(鸢尾花)



**Iris Setosa**  
山鸢尾



**Iris Versicolour** 杂色鸢尾



**Iris Virginica** 维吉尼亚鸢尾



## 任务:

- f(



) = Iris Setosa

- f(



) = Iris Versicolour

- f(

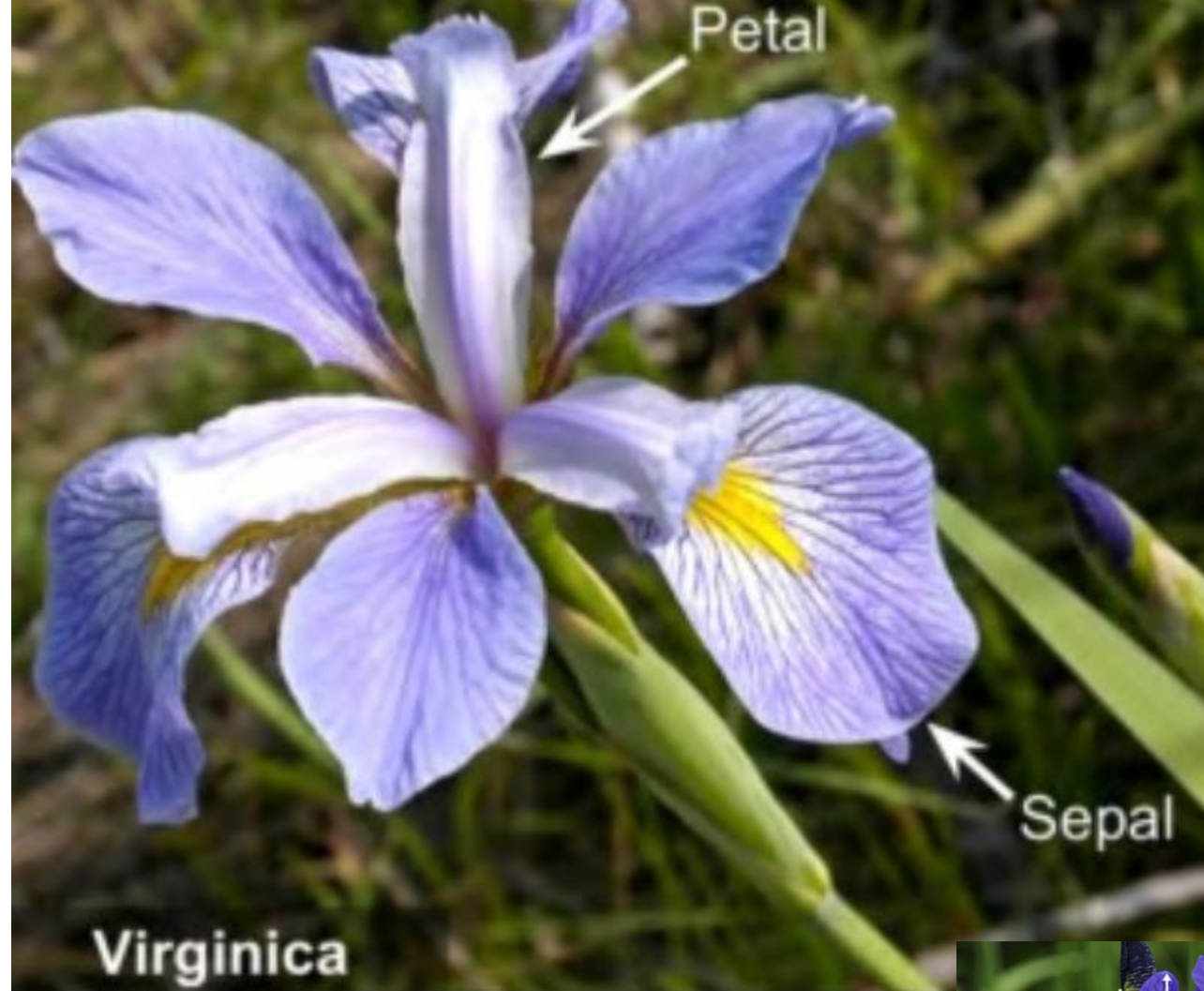


) = Iris Virginica

- **Iris Data Attribute Information:**

1. sepal(花萼) length in cm
2. sepal (花萼) width in cm
3. petal (花瓣) length in cm
4. petal (花瓣) width in cm
5. class:
  - Iris Setosa 山鸢尾
  - Iris Versicolour 杂色鸢尾
  - Iris Virginica 维吉尼亚鸢尾



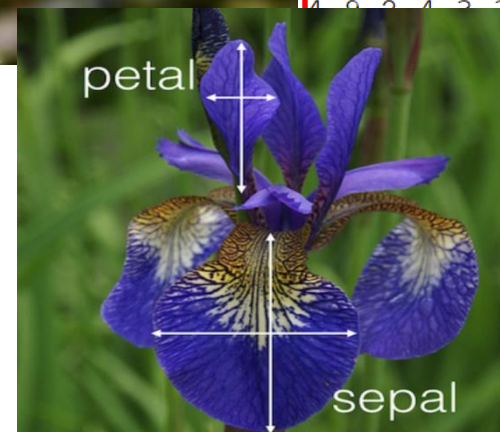


5.4,3.4,1.5,0.4	Iris-setosa
5.2,4.1,1.5,0.1	Iris-setosa
5.5,4.2,1.4,0.2	Iris-setosa
4.9,3.1,1.5,0.1	Iris-setosa
5.0,3.2,1.2,0.2	Iris-setosa
5.5,3.5,1.3,0.2	Iris-setosa
4.9,3.1,1.5,0.1	Iris-setosa
4.4,3.0,1.3,0.2	Iris-setosa
5.1,3.4,1.5,0.2	Iris-setosa
5.0,3.5,1.3,0.3	Iris-setosa
4.5,2.3,1.3,0.3	Iris-setosa
4.4,3.2,1.3,0.2	Iris-setosa
5.0,3.5,1.6,0.6	Iris-setosa
5.1,3.8,1.9,0.4	Iris-setosa
4.8,3.0,1.4,0.3	Iris-setosa
5.1,3.8,1.6,0.2	Iris-setosa
4.6,3.2,1.4,0.2	Iris-setosa
5.3,3.7,1.5,0.2	Iris-setosa
5.0,3.3,1.4,0.2	Iris-setosa
7.0,3.2,4.7,1.4	Iris-versicolor
6.4,3.2,4.5,1.5	Iris-versicolor
6.9,3.1,4.9,1.5	Iris-versicolor
5.5,2.3,4.0,1.3	Iris-versicolor
6.5,2.8,4.6,1.5	Iris-versicolor
5.7,2.8,4.5,1.3	Iris-versicolor
6.3,3.3,4.7,1.6	Iris-versicolor
4.9,2.4,3.7,1.0	Iris-versicolor
5.0,2.3,3.7,1.3	Iris-versicolor

**Task:**

**$(6.6, 2.9, 4.6, 1.3) \Rightarrow \text{Class?}$**

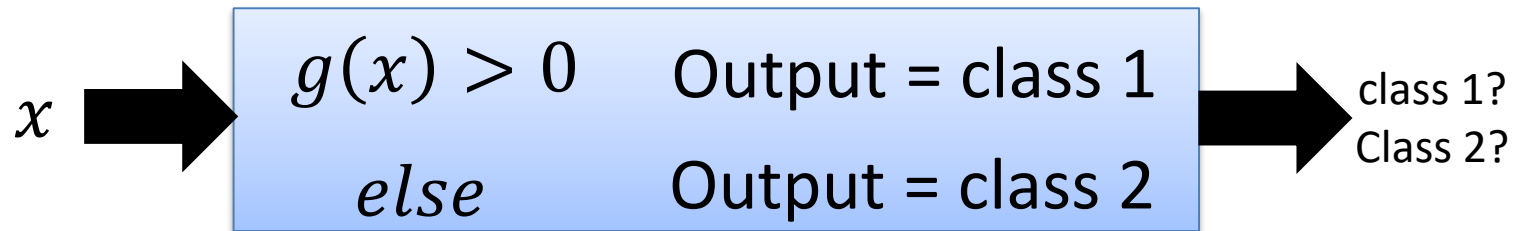
**$f(6.6, 2.9, 4.6, 1.3) = ?$**



# Ideal Alternatives

**Data:**  $(x^{(n)}, \hat{y}^{(n)})$

- Function (Model):  $f(x)$



- Loss function:

$$L(f) = \sum_n \delta(f(x^{(n)}) \neq \hat{y}^{(n)})$$

The number of times  $f$  get incorrect results on training data.

- Find the best function:
  - Example: Perceptron, SVM

Not Today

# 利用线性回归模型来解决分类问题？

- Training data for Classification

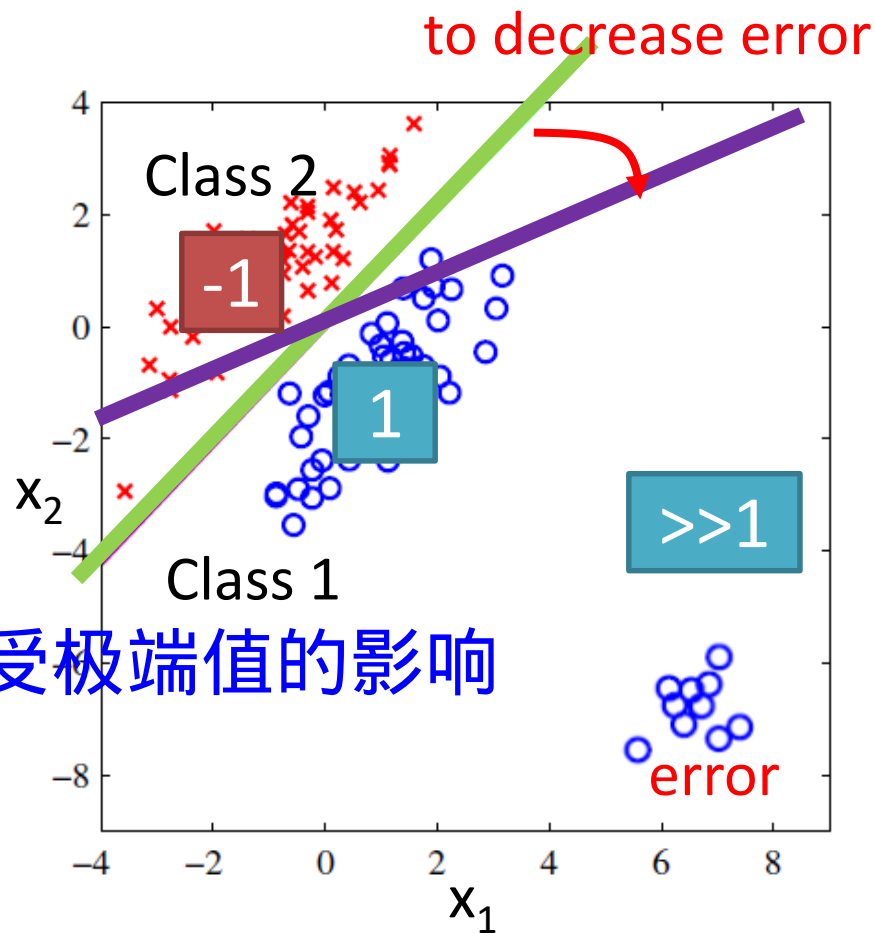
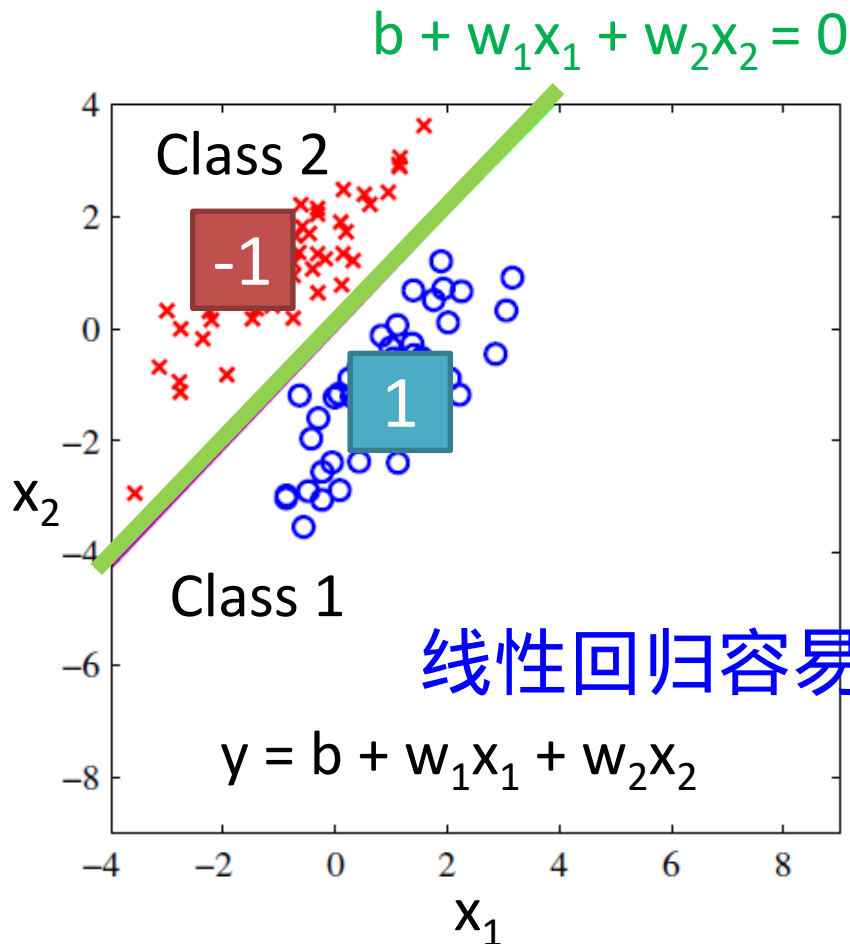
$(x^{(1)}, \hat{y}^{(1)})$	$(x^{(2)}, \hat{y}^{(2)})$	... ..	$(x^{(N)}, \hat{y}^{(N)})$
5.1, 3.4, 1.5, 0.2	6.9, 3.1, 4.9, 1.5		7.3, 2.9, 6.3, 1.8
Iris-setosa	Iris-versicolor		Iris-virginica

## Classification as Regression?

Binary classification as example

Training: Class 1 means the target is 1; Class 2 means the target is -1

Testing: closer to 1 → class 1; closer to -1 → class 2



线性回归容易受极端值的影响

Penalize to the examples that are “too correct” ...

(Bishop, P186)

- Multiple class: Class 1 means the target is 1; Class 2 means the target is 2; Class 3 means the target is 3 ..... problematic

结论：不能利用线性回归模型来解决分类问题

# 本章内容

- 为什么不直接利用线性回归模型来解决分类问题？
- 二分类问题中，逻辑回归模型训练“三步曲”
  - 分类问题中，为什么使用交叉熵（cross entropy）而不是MSE作为损失函数？
- 多分类问题中，使用Softmax回归模型
  - 对比逻辑回归模型，比较异同
- 分类器的性能评估
  - 混淆矩阵，精度和召回率，PRC，ROC
- 利用混淆矩阵进行错误分析
- 多标签分类问题
- 多输出分类问题
- 延伸过渡：
  - 层叠的逻辑回归模型 -> MLP -> DNN



# 两个基本的分类模型

- 逻辑回归(Logistic Regression)
  - 解决“二分类”问题
  - 训练“三步曲”
  - 为什么使用交叉熵 (cross entropy) 而不是MSE作为损失函数?
- Softmax回归
  - 解决“多分类”问题
  - 训练“三步曲”
  - 逻辑回归可看做是Softmax回归的特例

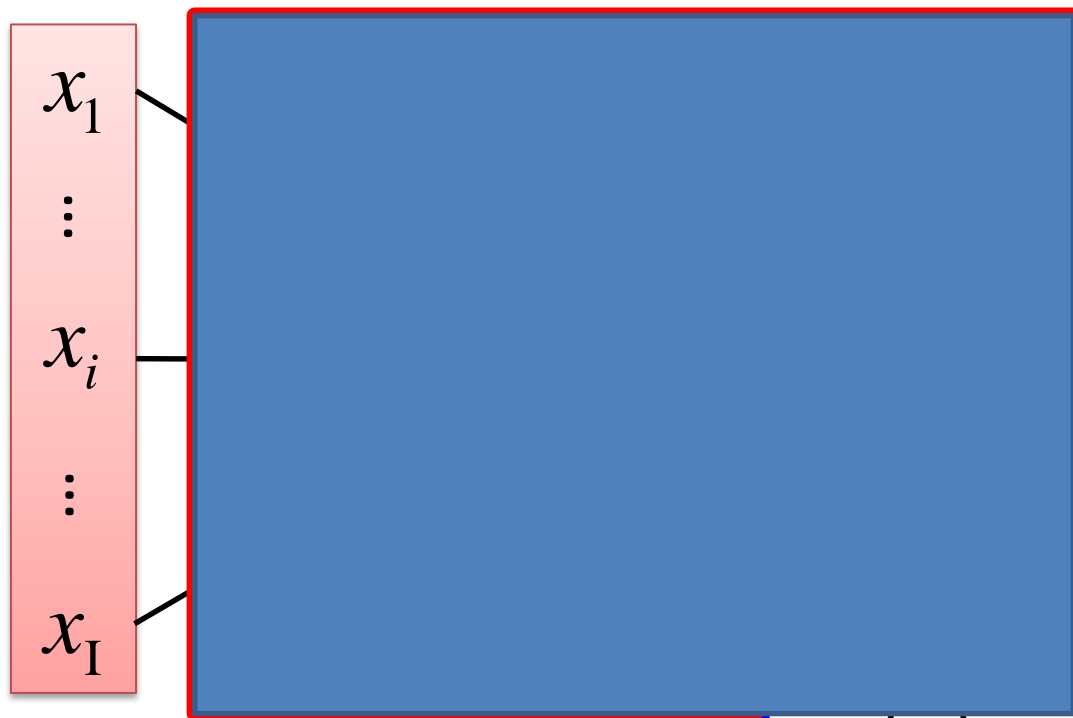
# 逻辑回归(Logistic Regression)

- Logistic回归（也称为Logit回归）通常用于二元分类器
  1. 估计一个实例属于两类中某个特定类别的概率（例如，这电子邮件是垃圾邮件的概率是多少？）（核心）
  2. 利用估计的概率与阈值的相对关系来判定其类别。比如，若估计的概率大于50%，那么模型预测这个实例属于当前类（称为正类，标记为“1”），反之预测它不属于当前类（即它属于负类，标记为“0”）。

# 逻辑回归(Logistic Regression)

- Step 1:  $\hat{p} = h_{\theta}(\mathbf{x}) = \sigma(\theta^T \cdot \mathbf{x}) = \sigma(z)$

$$\hat{y} = \begin{cases} 0, & \hat{p} < 0.5 \\ 1, & \hat{p} \geq 0.5 \end{cases}$$



计算输入特征的加权和  
(加上偏差项)，把结果  
输入Sigmoid函数进行二次  
加工后进行输出。

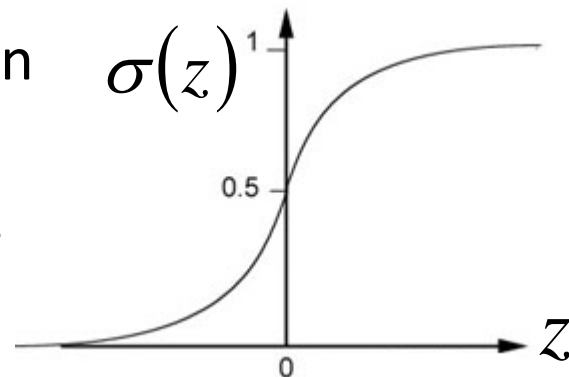
输入:  $\mathbf{x}$

输出: 为正类别的概率值

$$\hat{p} \in [0, 1]$$

function

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



# 逻辑回归(Logistic Regression)

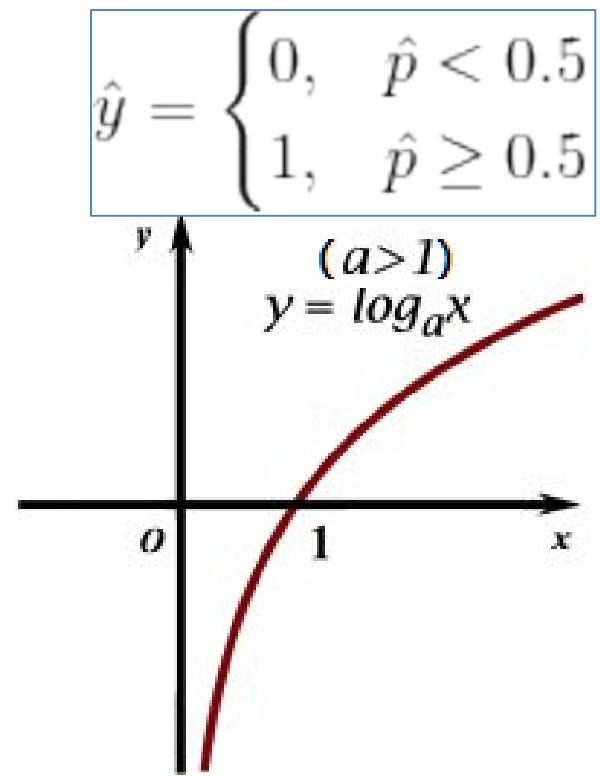
- Step2: 损失函数定义为: (对数损失函数)

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})]$$

公式 4-16: 单个样本的损失函数

$$c(\theta) = \begin{cases} -\log(\hat{p}), & y = 1 \\ -\log(1 - \hat{p}), & y = 0 \end{cases}$$

- 对于数据  $(x^{(i)}, y^{(i)})$ ,
  - 若  $y^{(i)}=1$ , 则预估的概率  $\hat{p}$  为 1 时, 损失函数取得最小值 0
  - 若  $y^{(i)}=0$ , 则预估的概率  $\hat{p}$  为 0 时, 损失函数取得最小值 0



标签  $y^{(i)}$  与预估的概率相等时, 损失函数值最小

# 逻辑回归(Logistic Regression)

- Step3: 利用梯度来更新参数

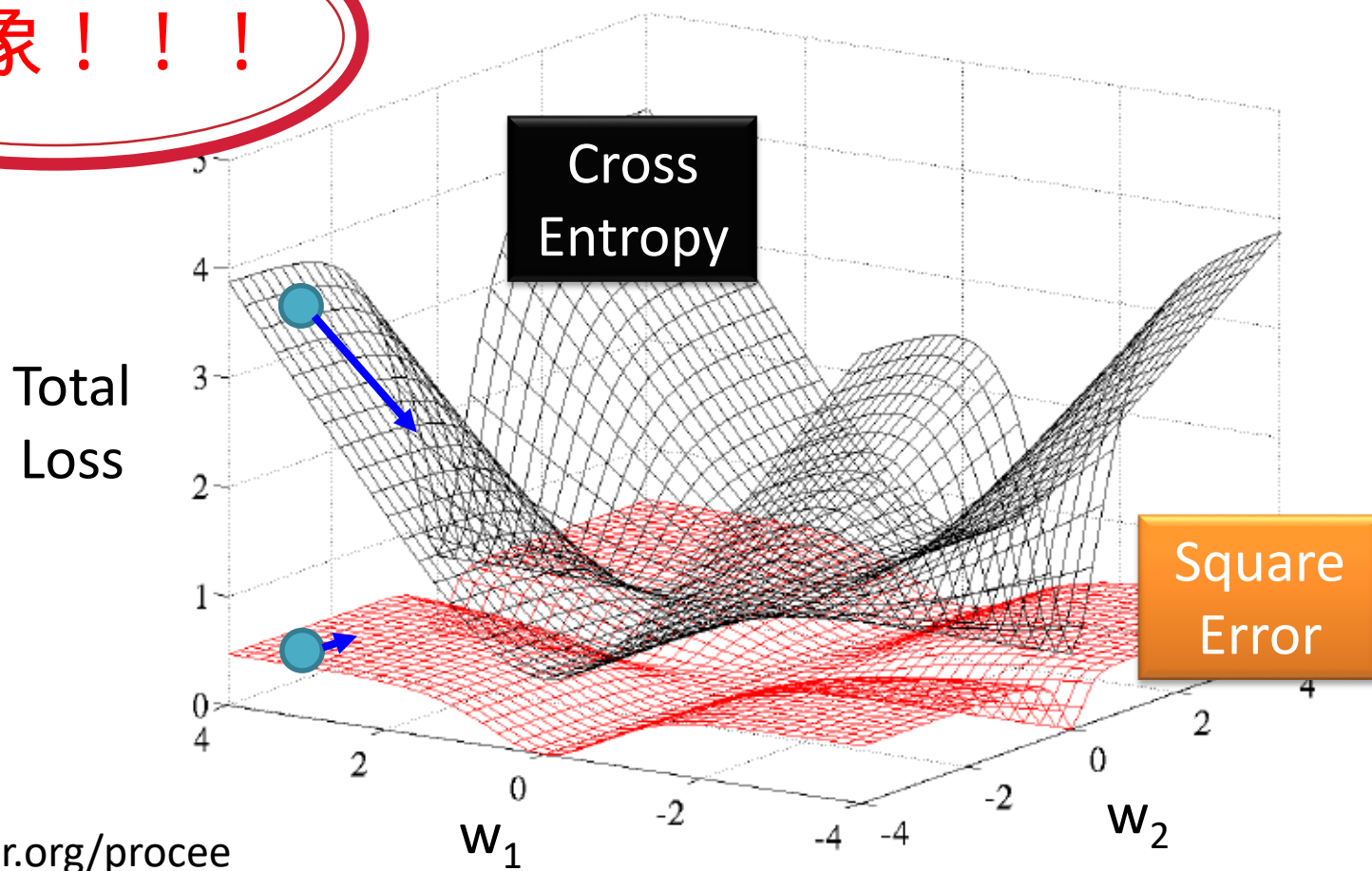
$$\theta_j \leftarrow \theta_j - \eta \frac{\partial}{\partial \theta_j} J(\theta_j)$$

$$\frac{\partial}{\partial \theta_j} J(\theta_j) = \frac{1}{m} \sum_{i=1}^m \underbrace{(\sigma(\theta^T \cdot \mathbf{x}^{(i)}) - y^{(i)})}_{\text{Larger difference, larger update}} x_j^{(i)}$$

Larger difference, larger update

# Cross Entropy vs. Square Error

形象!!!



<http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf>

不能使用均方误差作为  
分类模型的损失函数

## Logistic Regression + Square Error


Step 1:  $f_{w,b}(x) = \sigma \left( \sum_i w_i x_i + b \right)$

Step 2: Training data:  $(x^n, \hat{y}^n)$ ,  $\hat{y}^n$ : 1 for class 1, 0 for class 2

$$L(f) = \frac{1}{2} \sum_n (f_{w,b}(x^{(n)}) - \hat{y}^{(n)})^2$$

Step 3:

$$\begin{aligned} \frac{\partial (f_{w,b}(x) - \hat{y})^2}{\partial w_i} &= 2(f_{w,b}(x) - \hat{y}) \frac{\partial f_{w,b}(x)}{\partial z} \frac{\partial z}{\partial w_i} \\ &= 2(f_{w,b}(x) - \hat{y}) f_{w,b}(x) (1 - f_{w,b}(x)) x_i \end{aligned}$$

$\hat{y}^{(n)} = 1$  If  $f_{w,b}(x^{(n)}) = 1$  (close to target)   $\partial L / \partial w_i = 0$

If  $f_{w,b}(x^{(n)}) = 0$  (far from target)   $\partial L / \partial w_i = 0$

## Logistic Regression + Square Error

Step 1:  $f_{w,b}(x) = \sigma \left( \sum_i w_i x_i + b \right)$

Step 2: Training data:  $(x^{(n)}, \hat{y}^{(n)})$ ,  $\hat{y}^{(n)}$ : 1 for class 1, 0 for class 2

$$L(f) = \frac{1}{2} \sum_n (f_{w,b}(x^{(n)}) - \hat{y}^{(n)})^2$$

Step 3:

$$\frac{\partial (f_{w,b}(x) - \hat{y})^2}{\partial w_i} = 2(f_{w,b}(x) - \hat{y}) \frac{\partial f_{w,b}(x)}{\partial z} \frac{\partial z}{\partial w_i}$$
$$= 2(f_{w,b}(x) - \hat{y}) f_{w,b}(x) (1 - f_{w,b}(x)) x_i$$

$\hat{y}^{(n)} = 0$  If  $f_{w,b}(x^{(n)}) = 1$  (far from target)   $\partial L / \partial w_i = 0$

If  $f_{w,b}(x^{(n)}) = 0$  (close to target)   $\partial L / \partial w_i = 0$



## **Logistic Regression**

Step 1:  $f_{w,b}(x) = \sigma \left( \sum_i w_i x_i + b \right)$

Output: between 0 and 1

## **Linear Regression**

$$f_{w,b}(x) = \sum_i w_i x_i + b$$

Output: any value

Step 2:

Step 3:

## Logistic Regression

Step 1:  $f_{w,b}(x) = \sigma \left( \sum_i w_i x_i + b \right)$

Output: between 0 and 1

Training data:  $(x^{(n)}, \hat{y}^{(n)})$

Step 2:  $\hat{y}^{(n)}$ : 1 for class 1, 0 for class 2

$$L(f) = \sum_n l(f(x^{(n)}), \hat{y}^{(n)})$$

## Linear Regression

$$f_{w,b}(x) = \sum_i w_i x_i + b$$

Output: any value

Training data:  $(x^{(n)}, \hat{y}^{(n)})$

$\hat{y}^{(n)}$ : a real number

$$L(f) = \frac{1}{2} \sum_n (f(x^{(n)}) - \hat{y}^{(n)})^2$$

**Cross entropy:**

$$l(f(x^{(n)}), \hat{y}^{(n)}) = - \left[ \hat{y}^{(n)} \ln f(x^{(n)}) + (1 - \hat{y}^{(n)}) \ln (1 - f(x^{(n)})) \right]$$

## Logistic Regression

Step 1:  $f_{w,b}(x) = \sigma \left( \sum_i w_i x_i + b \right)$

Output: between 0 and 1

Training data:  $(x^{(n)}, \hat{y}^{(n)})$

Step 2:  $\hat{y}^{(n)}$ : 1 for class 1, 0 for class 2

$$L(f) = \sum_n l(f(x^{(n)}), \hat{y}^{(n)})$$

## Linear Regression

$$f_{w,b}(x) = \sum_i w_i x_i + b$$

Output: any value

Training data:  $(x^{(n)}, \hat{y}^{(n)})$

$\hat{y}^n$ : a real number

$$L(f) = \frac{1}{2} \sum_n (f(x^{(n)}) - \hat{y}^{(n)})^2$$

Logistic regression:  $w_i \leftarrow w_i - \eta \sum_n - \left( \hat{y}^{(n)} - f_{w,b}(x^{(n)}) \right) x_i^{(n)}$

Step 3:

Linear regression:  $w_i \leftarrow w_i - \eta \sum_n - \left( \hat{y}^{(n)} - f_{w,b}(x^{(n)}) \right) x_i^{(n)}$

# 本章内容

- 为什么不直接利用线性回归模型来解决分类问题？
- 二分类问题中，逻辑回归模型训练“三步曲”
  - 分类问题中，为什么使用交叉熵（cross entropy）而不是MSE作为损失函数？
- 多分类问题中，使用Softmax回归模型
  - 对比逻辑回归模型，比较异同
- 分类器的性能评估
  - 混淆矩阵，精度和召回率，PRC，ROC
- 利用混淆矩阵进行错误分析
- 多标签分类问题
- 多输出分类问题
- 延伸过渡：
  - 层叠的逻辑回归模型 -> MLP -> DNN

# Multi-class Classification (3 classes as example)

$$C_1: w_1, b_1 \quad z_1 = w_1 \cdot x + b_1$$

$$C_2: w_2, b_2 \quad z_2 = w_2 \cdot x + b_2$$

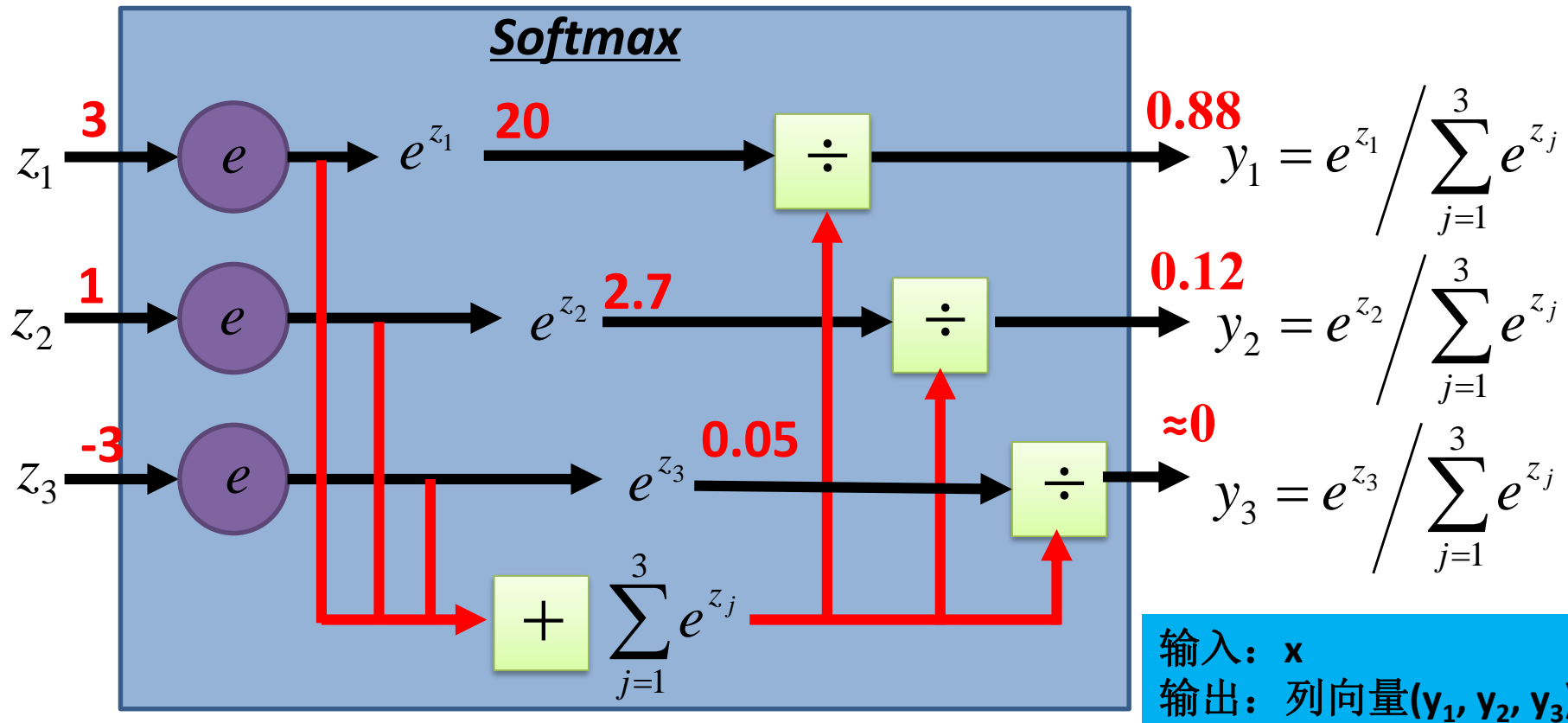
$$C_3: w_3, b_3 \quad z_3 = w_3 \cdot x + b_3$$

Probability:

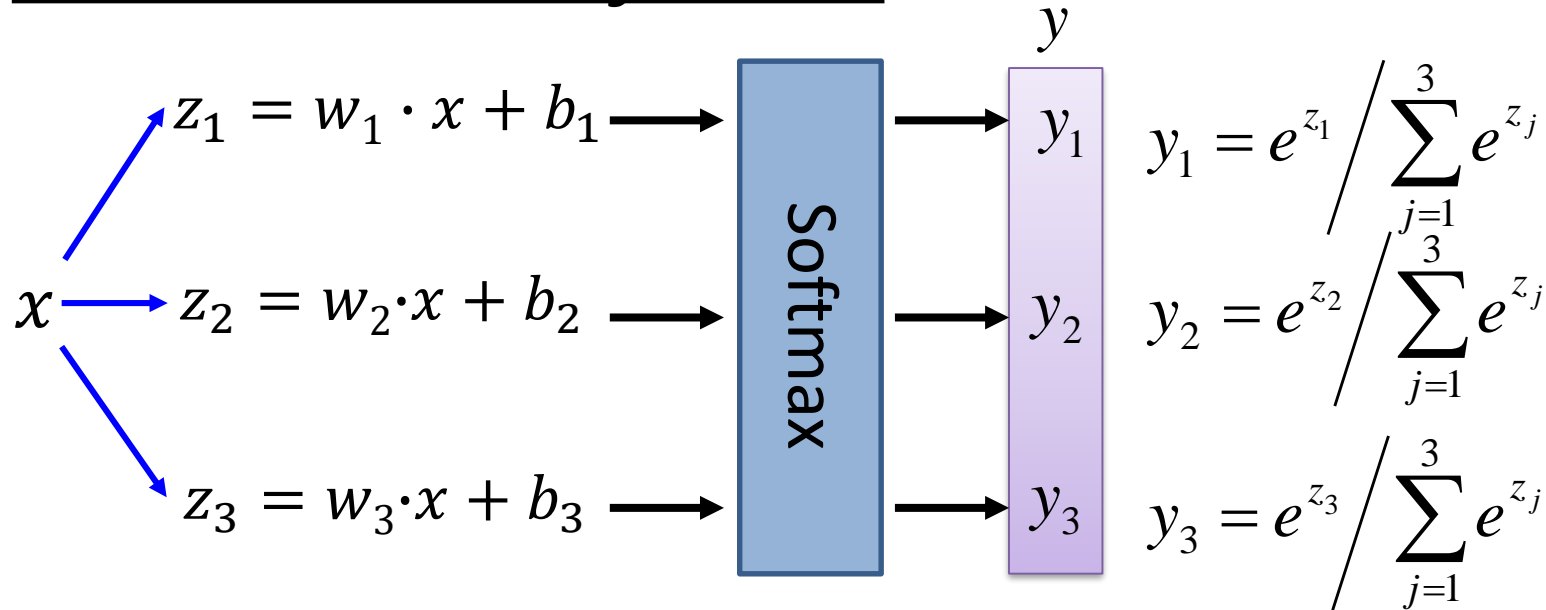
$$\blacksquare 1 > y_i > 0$$

$$\blacksquare \sum_i y_i = 1$$

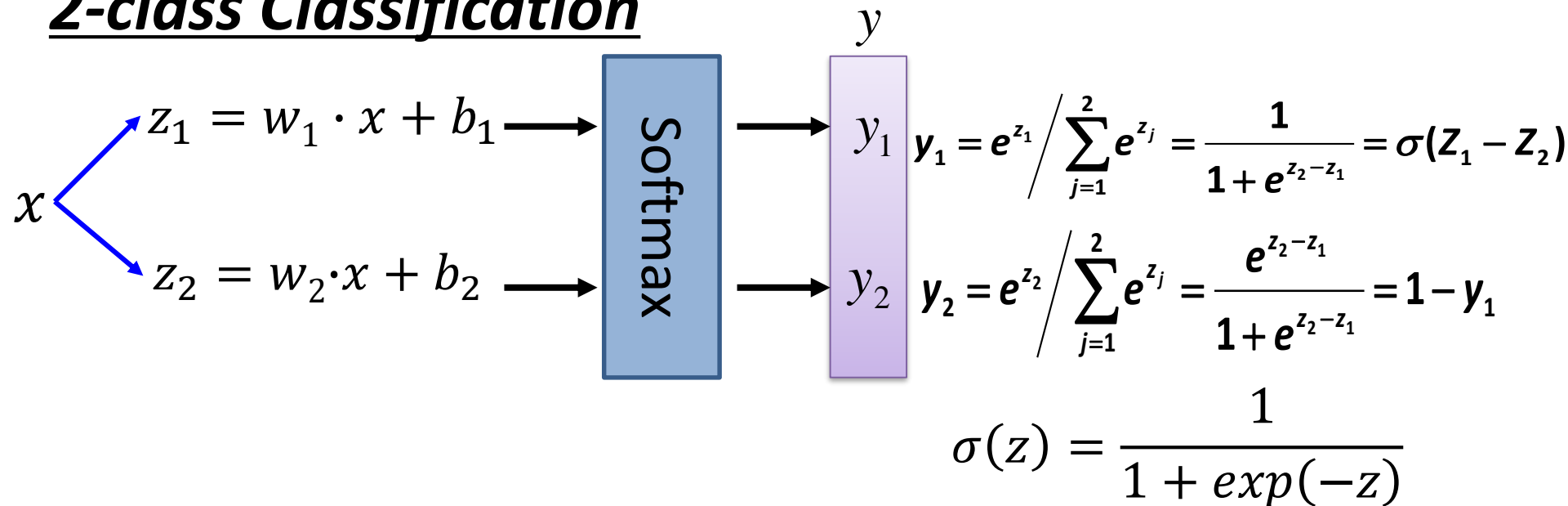
$$y_i = P(C_i | x)$$



## Multi-class Classification (3 classes as example)



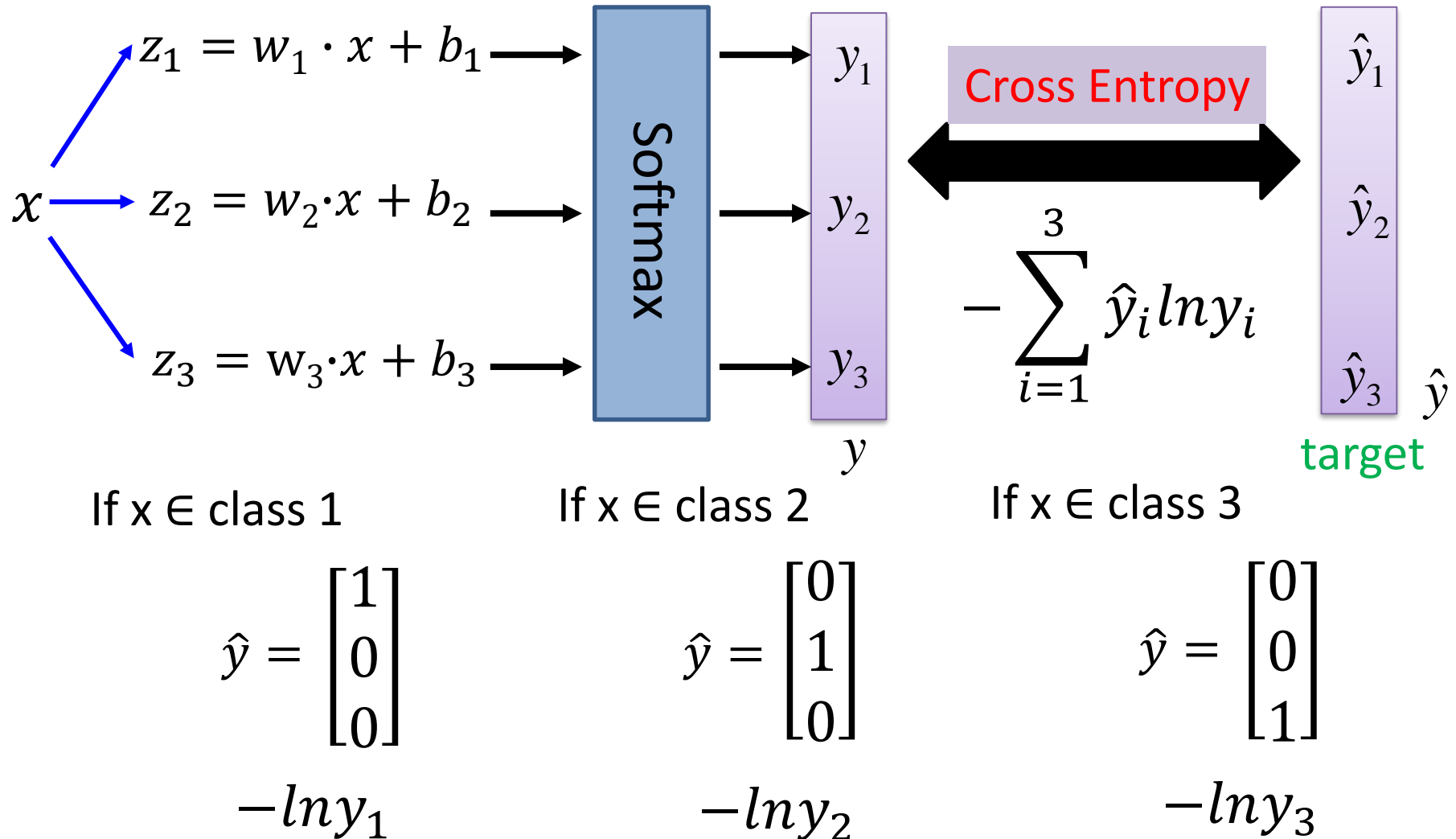
## 2-class Classification



# Multi-class Classification (3 classes as example)

## Loss function for Binary classifier:

$$l(f(x^{(n)}), \hat{y}^{(n)}) = - \left[ \hat{y}^{(n)} \ln f(x^{(n)}) + (1 - \hat{y}^{(n)}) \ln (1 - f(x^{(n)})) \right]$$



# 本章内容

- 为什么不直接利用线性回归模型来解决分类问题？
- 二分类问题中，逻辑回归模型训练“三步曲”
  - 分类问题中，为什么使用交叉熵（cross entropy）而不是MSE作为损失函数？
- 多分类问题中，使用Softmax回归模型
  - 对比逻辑回归模型，比较异同
- 分类器的性能评估
  - 混淆矩阵，精度和召回率，PRC，ROC
- 利用混淆矩阵进行错误分析
- 多标签分类问题
- 多输出分类问题
- 延伸过渡：
  - 层叠的逻辑回归模型 -> MLP -> DNN



# 分类器的性能评估(Classifier Metrics)

- 分类器的性能评估指标有哪些？
- 如何选择合适的指标来评估分类器性能？
- 如何权衡选择合适的阈值以满足需求的精确率和召回率？
- 如何使用PRC曲线和ROC曲线来比较多个分类模型？

# 分类器的性能评估指标

评估 已训练好的模型的性能

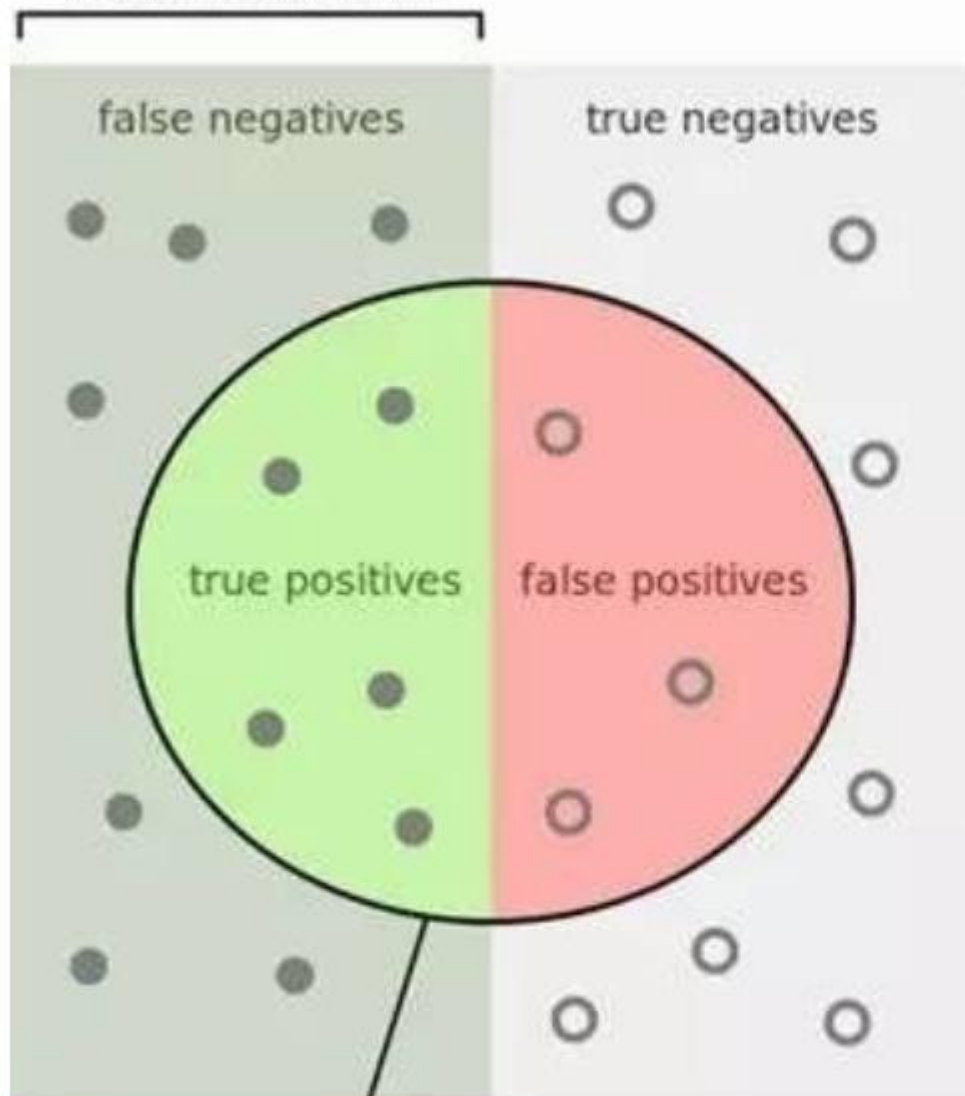
- 单一指标：
  - **Precision**: 精确率/查准率
  - **Recall**: 召回率/查全率
  - **Accuracy**: 准确率/正确率
  - **F1-score**: F1分数
- 综合指标：
  - **PRC**(Precision Recall Curve) & **AP** (Average Precision) :是综合评价整体结果的评估指标
    - 对样本不均衡敏感：哪种类型（正或者负）样本多，权重就大
  - **ROC** “受试者工作特征”（Receiver Operating Characteristic）曲线 & **AUC**（Area Under Curve）

# 分类器的性能评估指标——Precision和Recall

- 精确率/查准率Precision  $P = \frac{TP}{TP + FP}$ 
  - 是指在所有系统判定的“真”的样本中，确实是真的占比
- 召回率/查全率Recall  $R = \frac{TP}{TP + FN}$ 
  - 是指在所有确实为真的样本中，被判为的“真”的占比
- 混淆矩阵（Confusion matrix）
  - True Positive（真正，TP）被模型预测为正的正样本
  - True Negative（真负，TN）被模型预测为负的负样本
  - False Positive（假正，FP）被模型预测为正的负样本
  - False Negative（假负，FN）被模型预测为负的正样本

	actual positive	actual negative
predicted positive	$TP$	$FP$
predicted negative	$FN$	$TN$

relevant elements (实际为正类)



selected elements (预估为正类)

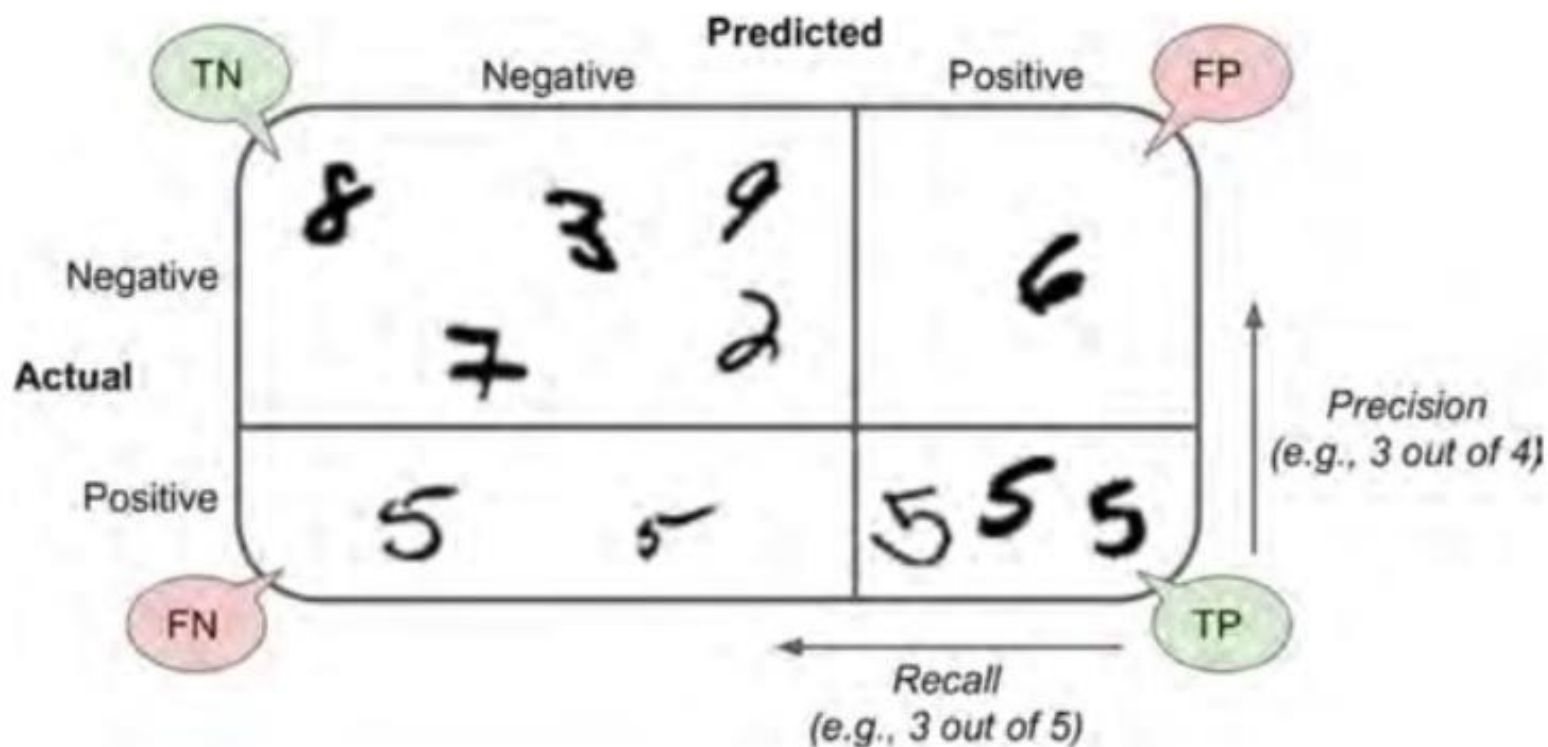
How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

# 示例：计算Precision和Recall



数字“5”探测器的结果

# 示例：计算Precision和Recall

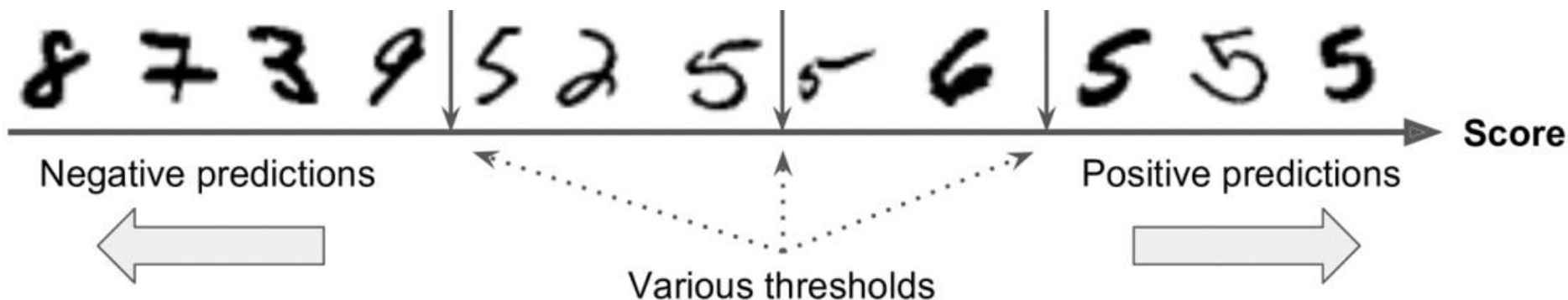


Figure 3-3. Decision threshold and precision/recall tradeoff

- 对于每个样本，按照 判断其为正类的决策分数Score进行排序
  - Score（即为：输入特征的加权和，再加上偏差项）越大，表示其为正类的预估概率越大
- 对于不同的阈值(threshold)，分别计算其Precision和Recall
  - 对于某样本，若其为正类的预估概率超过阈值，则判定为正类；否则，为负类。
  - 观察发现：阈值不同，计算得到的(Precision, Recall)值对 不同。

# 示例：计算Precision和Recall

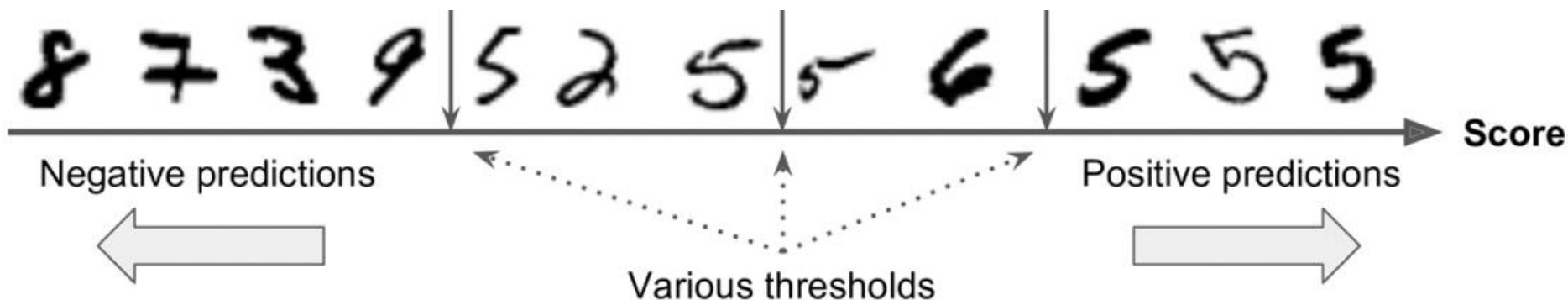


Figure 3-3. Decision threshold and precision/recall tradeoff

- 对于每个样本，按照 判断其为正类的决策分数Score进行排序
  - Score（即为：输入特征的加权和，再加上偏差项）越大，表示其为正类的预估概率越大
- 对于不同的阈值(threshold)，分别计算其Precision和Recall
  - 对于某样本，若其为正类的预估概率超过阈值，则判定为正类；否则，为负类。
  - 观察发现：阈值不同，计算得到的(Precision, Recall)值对 不同。

# 分类器的性能评估指标——F1-score

- **F1-score:** 是准确率和召回率的调和平均。

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

- 支持那些有着相近准确率和召回率的分类器。
  - 普通的平均值平等地看待所有的值，而调和平均会给小的值更大的权重。所以，要想分类器得到一个高的F1值，需要召回率和准确率同时高
- 是综合考虑precision和recall的指标，比BEP更为常用



# 分类器的性能评估指标——Accuracy

- **Accuracy:** 准确率/正确率
  - 正确的预测/总的样本数

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

- **注意区分:** Precision和Accuracy（正确率）
  - accuracy针对所有样本
  - precision针对正样本（是部分样本），即正确的正样本数/总的正样本数： $P = \frac{TP}{TP + FP}$

# Scikit-learn中的实现

- Accuracy:

```
>>> from sklearn.model_selection import cross_val_score
>>> cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
array([ 0.9502 , 0.96565, 0.96495])
```

- Precision and Recall:

```
>>> from sklearn.metrics import precision_score, recall_score
>>> precision_score(y_train_5, y_pred) # == 4344 / (4344 + 1307)
0.76871350203503808
>>> recall_score(y_train_5, y_train_pred) # == 4344 / (4344 + 1077)
0.79136690647482011
```

- F1 Score:

```
>>> from sklearn.metrics import f1_score
>>> f1_score(y_train_5, y_pred)
0.78468208092485547
```

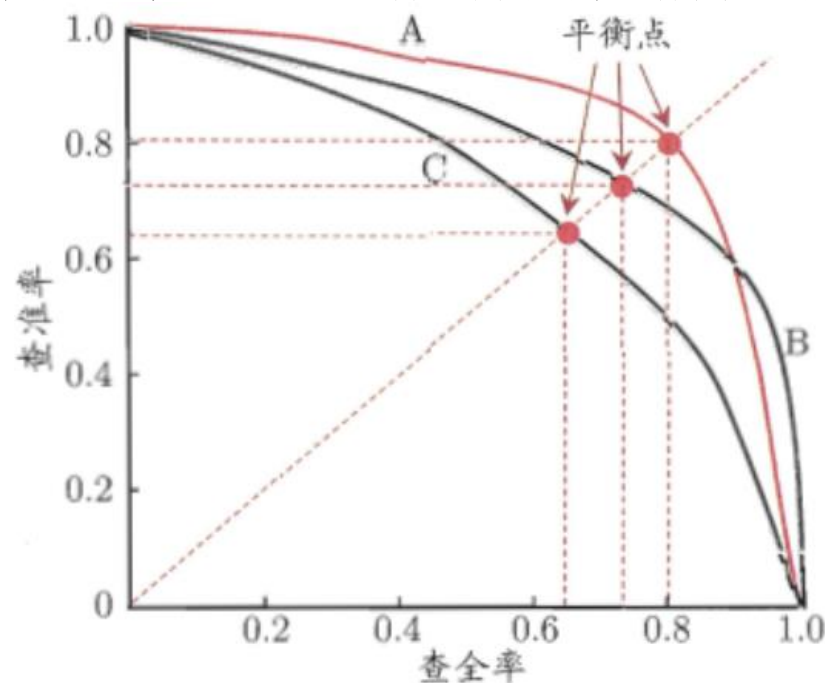
# 分类器的性能评估指标——PRC

- PRC: 是综合评价Precision和Recall整体结果的评估指标

## — 怎么画PRC曲线?

- 对于不同的决策分数阈值，分别计算查全率R和查准率P。
- 以查准率为Y轴，查全率为X轴，画出所有阈值情况下的(R, P)值对。

**PRC上的每个点都对应于一个阈值**

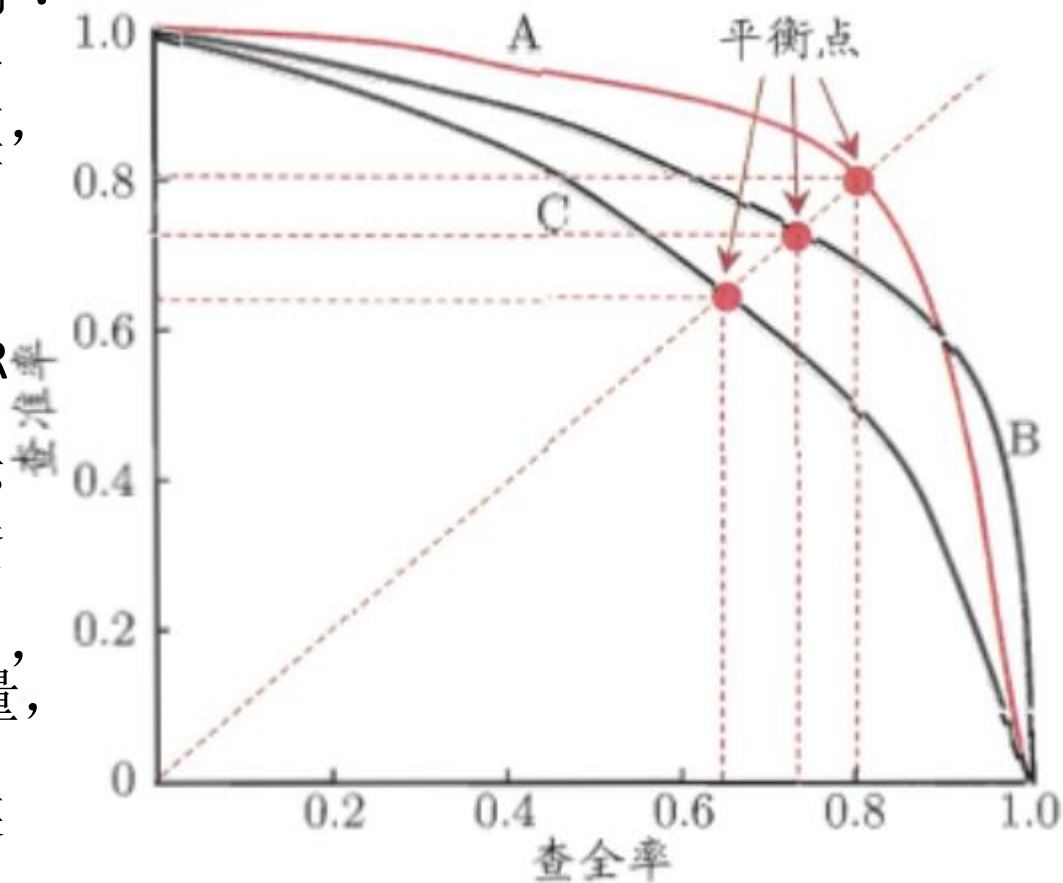


# 分类器的性能评估指标——PRC

## • 解读PRC: A, B, C, 谁最好?

1. 若一个学习器的PR曲线被另一个学习器的曲线完全包住, 则可断言后者的性能优于前者。比如右图中A优于C, B优于C

2. 若两条曲线交叉, 则比较PR曲线下的面积, 即AP (Average Precision), 该指标在一定程度上表征了学习器在查准率和查全率上取得相对“双高”的比例。实践中, 引入“平衡点”(BEP)来度量, BEP表示“查准率=查全率”时的取值, 值越大表明分类器性能越好, 比如右图中A优于B。



- 根据决策分数的阈值来预测类别:

```
>>> y_scores = sgd_clf.decision_function([some_digit])
>>> y_scores
array([ 161855.74572176])
>>> threshold = 0
>>> y_some_digit_pred = (y_scores > threshold)
array([ True], dtype=bool)
```

- 计算不同的决策分数阈值下的(P,R)值对:

```
y_scores = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3,
                             method="decision_function")
```

```
from sklearn.metrics import precision_recall_curve
precisions, recalls, thresholds = precision_recall_curve(y_train_5, y_scores)
```

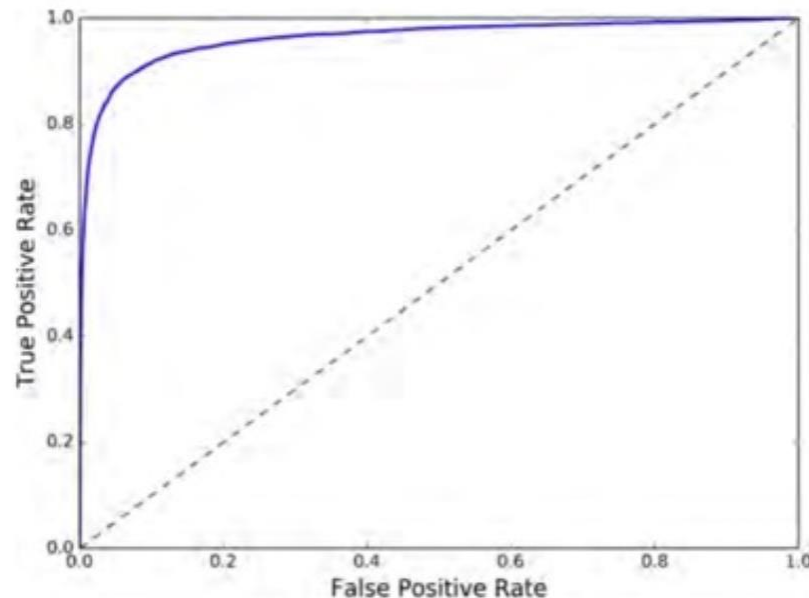
# 分类器的性能评估指标——ROC

- ROC: “受试者工作特征” (Receiver Operating Characteristic) 曲线
  - 在不同的分类阈值下, 分别计算其对应的“真正例率” TPR 和“假正例率” FPR
  - 以 TPR 为 Y 轴, 以 FPR 为 X 轴, 画出不同分类阈值下的 (FPR, TPR) 值对。

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{TN + FP}$$

**ROC 上的每个点都  
对应于一个阈值**



# 示例：如何画ROC？

Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1

左图中共有20个测试样本。

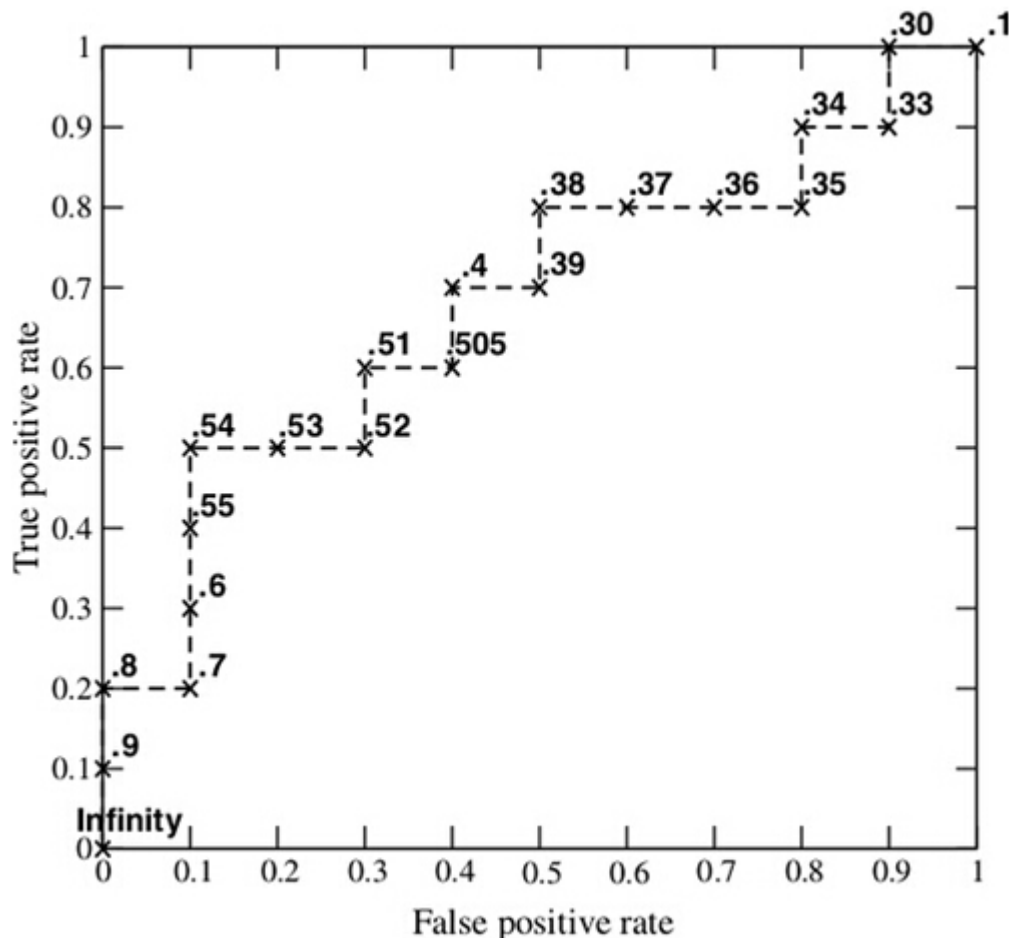
1. “Inst#”栏表示样本序号；
2. “Class”栏表示每个测试样本真正的标签（p表示正样本，n表示负样本）；
3. “Score”栏目表示每个测试样本属于正样本的预估概率，降序排列。

# 示例：如何画ROC？

Threshold = 0.1时，TPR=1, FPR=1;

Threshold = 0.3时，TPR=1, FPR=1/10;

Threshold = 0.4时，TPR=7/10, FPR=4/10;



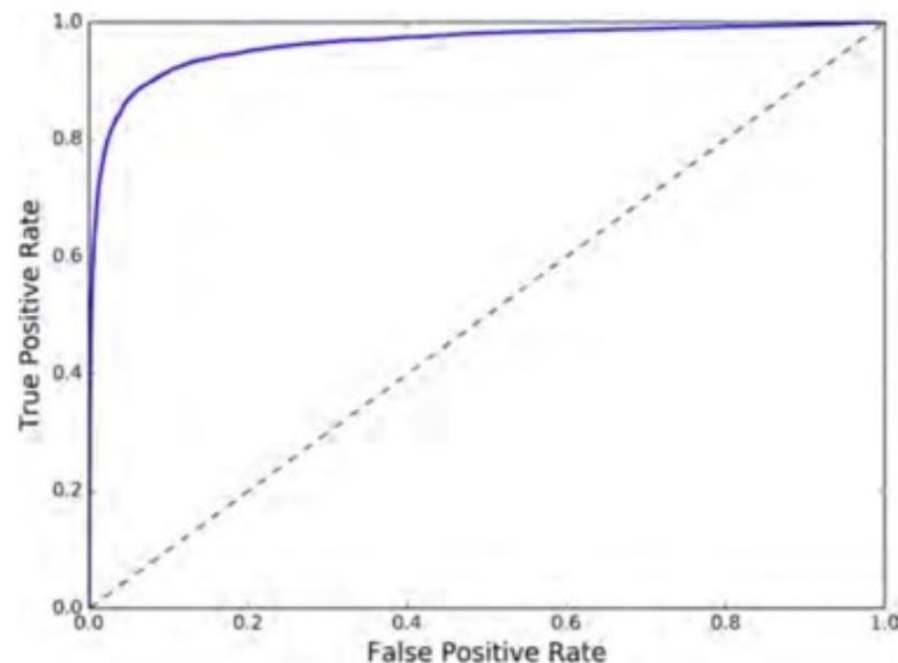
Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1

1. 从高到低，依次将“Score”值作为阈值threshold;
2. 每次选取一个不同的Score作为threshold，我们就可以得到一组FPR和TPR，即ROC曲线上的一点。  
✓ 当测试样本属于正样本的预估概率 $\geq$ threshold时，则认为它为正样本；否则为负样本。
3. 枚举所有的阈值，可得到20组FPR和TPR的值，将它们画在ROC曲线的结果如左图所示。



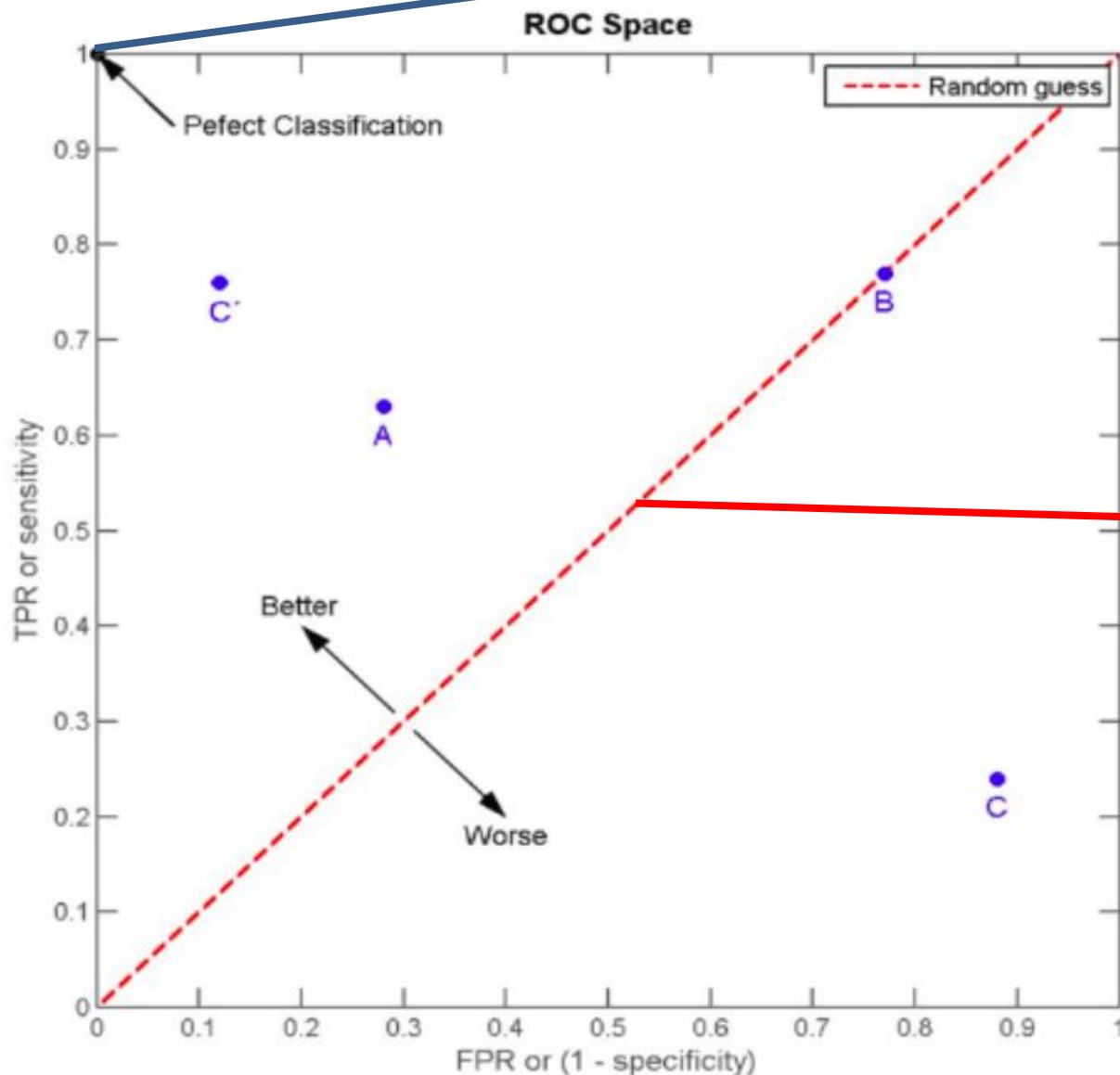
# 分类器的性能评估指标——ROC

- ROC: 优秀的分类器应该离对角虚线越远越好（向左上角）
  - 若一个学习器的ROC曲线被另一个学习器的曲线包住，那么我们可以断言后者性能优于前者；
  - 若两个学习器的ROC曲线发生交叉，则难以一般性断言两者孰优孰劣。此时若要进行比较，那么可以比较ROC曲线下的面积，即AUC（Area Under Curve），面积大的曲线对应的分类器性能更好。



# AUC (Area Under the Curve)

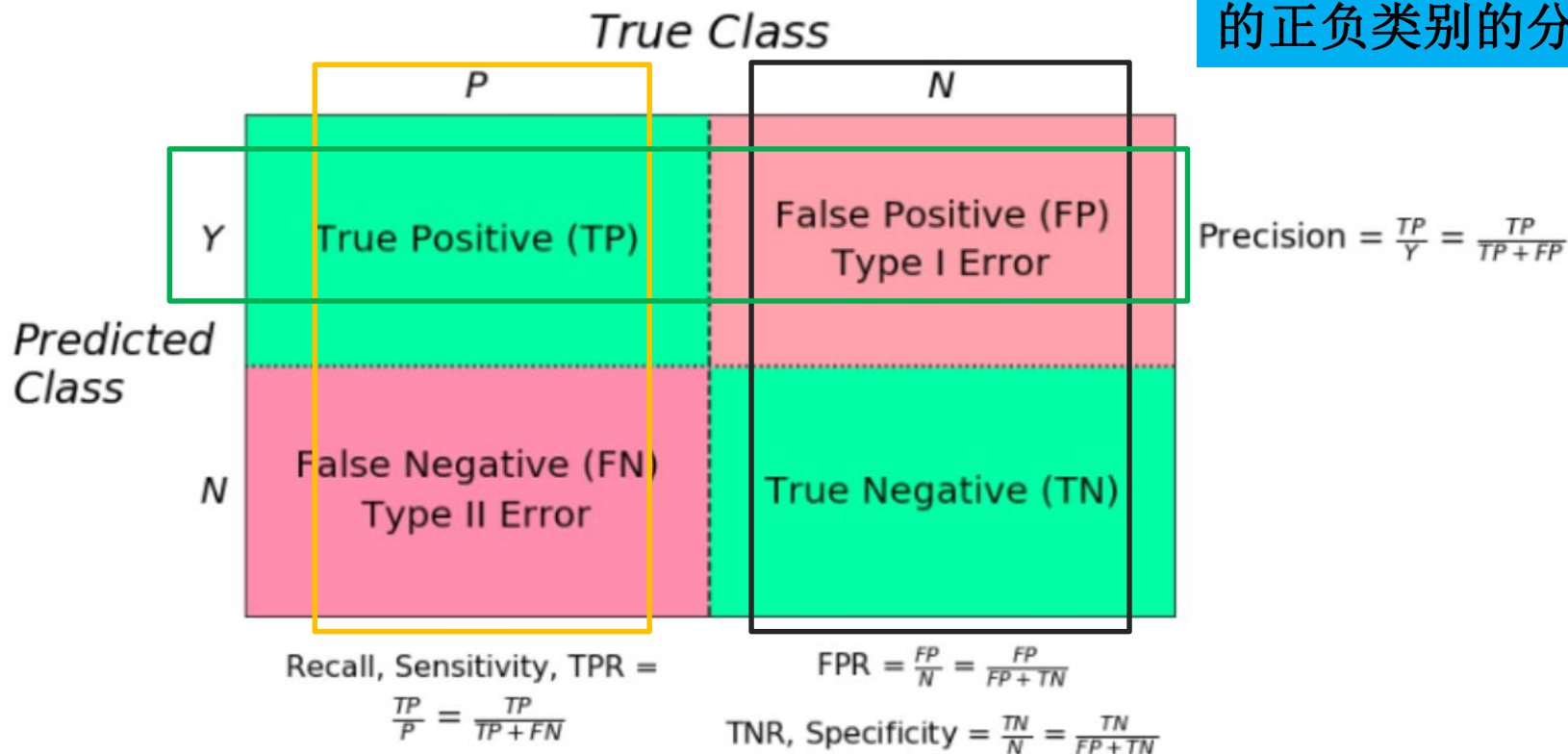
(0,1) 则对应 “理想模型”



虚线对角线对应于  
“随机猜测”模型

# PRC VS. ROC

ROC不依赖于具体的正负类别的分布



PRC:

$$P = \frac{TP}{TP+FP} \quad \text{Y轴}$$

$$R = \frac{TP}{TP+FN} \quad \text{X轴}$$

ROC:

$$TPR = \frac{TP}{TP+FN} \quad \text{Y轴}$$

$$FPR = \frac{FP}{TN+FP} \quad \text{X轴}$$

# 样本不均衡示例

该分类器实际上是很差的！

- 例如：在癌症预测的场景中，假设没有患癌症的样本为正例，患癌症样本为负例。若数据集中负例占比很少(大概0.1%)。假设现在某分类器将所有的样本都预测为正例，尝试利用不同的评估指标来评价该分类器的性能。
  - **Accuracy** = 99.9%。
  - ☺ – 使用**AUC**作为评估指标。该分类器把所有样本预测为正例，可计算得到：**TPR=1**，**FPR=1**。这种情况下该分类器的**AUC**值将等于0.5，成功规避了样本不均衡带来的问题。
  - 使用**PRC**作为评估指标。该分类器把所有样本预测为正例，可计算得到：**R=1**，**P=99.9%**。这种情况下该分类器接近完美分类器。

结论：样本不均衡时，使用**AUC**作为评估指标，评估性能非常接近其实际性能，评估很准确！

# 总结：二分类器的性能评估指标

- 单一指标：

- **Precision:** 精确率/查准率
- **Recall:** 召回率/查全率
- **F1-score:** F1分数
  - 支持那些有着相近准确率和召回率的分类器
- **Accuracy:** 准确率/正确率

- 综合指标：

- **PRC(Precision Recall Curve):**是综合评价整体结果的评估指标
  - 对样本不均衡敏感：哪种类型（正或者负）样本多，权重就大
  - AP的计算，使用`average_precision_score`方法
- **ROC “受试者工作特征” (Receiver Operating Characteristic) 曲线 & AUC (Area Under Curve)**
- 经验法则：当正例很少，或者当你关注假正例多于假反例的时候，优先使用PRC；其他情况使用ROC曲线。

## Classification metrics

See the [Classification metrics](#) section of the user guide for further details.

<code>metrics.accuracy_score(y_true, y_pred[, ...])</code>	Accuracy classification score.
<code>metrics.auc(x, y[, reorder])</code>	Compute Area Under the Curve (AUC) using the trapezoidal rule
<code>metrics.average_precision_score(y_true, y_score)</code>	Compute average precision (AP) from prediction scores
<code>metrics.balanced_accuracy_score(y_true, y_pred)</code>	Compute the balanced accuracy
<code>metrics.brier_score_loss(y_true, y_prob[, ...])</code>	Compute the Brier score.
<code>metrics.classification_report(y_true, y_pred)</code>	Build a text report showing the main classification metrics
<code>metrics.cohen_kappa_score(y1, y2[, labels, ...])</code>	Cohen's kappa: a statistic that measures inter-annotator agreement.
<code>metrics.confusion_matrix(y_true, y_pred[, ...])</code>	Compute confusion matrix to evaluate the accuracy of a classification
<code>metrics.f1_score(y_true, y_pred[, labels, ...])</code>	Compute the F1 score, also known as balanced F-score or F-measure
<code>metrics.fbeta_score(y_true, y_pred, beta[, ...])</code>	Compute the F-beta score
<code>metrics.hamming_loss(y_true, y_pred[, ...])</code>	Compute the average Hamming loss.
<code>metrics.hinge_loss(y_true, pred_decision[, ...])</code>	Average hinge loss (non-regularized)
<code>metrics.jaccard_score(y_true, y_pred[, ...])</code>	Jaccard similarity coefficient score
<code>metrics.log_loss(y_true, y_pred[, eps, ...])</code>	Log loss, aka logistic loss or cross-entropy loss.
<code>metrics.matthews_corrcoef(y_true, y_pred[, ...])</code>	Compute the Matthews correlation coefficient (MCC)
<code>metrics.multilabel_confusion_matrix(y_true, ...)</code>	Compute a confusion matrix for each class or sample
<code>metrics.precision_recall_curve(y_true, ...)</code>	Compute precision-recall pairs for different probability thresholds
<code>metrics.precision_recall_fscore_support(...)</code>	Compute precision, recall, F-measure and support for each class
<code>metrics.precision_score(y_true, y_pred[, ...])</code>	Compute the precision
<code>metrics.recall_score(y_true, y_pred[, ...])</code>	Compute the recall
<code>metrics.roc_auc_score(y_true, y_score[, ...])</code>	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
<code>metrics.roc_curve(y_true, y_score[, ...])</code>	Compute Receiver operating characteristic (ROC)
<code>metrics.zero_one_loss(y_true, y_pred[, ...])</code>	Zero-one classification loss.

### 3.3.2. Classification metrics

The `sklearn.metrics` module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values. Most implementations allow each sample to provide a weighted contribution to the overall score, through the `sample_weight` parameter.

Some of these are restricted to the binary classification case:

<code>precision_recall_curve</code> ( <code>y_true</code> , <code>probas_pred</code> )	Compute precision-recall pairs for different probability thresholds
<code>roc_curve</code> ( <code>y_true</code> , <code>y_score</code> [, <code>pos_label</code> , ...])	Compute Receiver operating characteristic (ROC)
<code>balanced_accuracy_score</code> ( <code>y_true</code> , <code>y_pred</code> [, ...])	Compute the balanced accuracy

Others also work in the multiclass case:

<code>cohen_kappa_score</code> ( <code>y1</code> , <code>y2</code> [, <code>labels</code> , <code>weights</code> , ...])	Cohen's kappa: a statistic that measures inter-annotator agreement.
<code>confusion_matrix</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute confusion matrix to evaluate the accuracy of a classification
<code>hinge_loss</code> ( <code>y_true</code> , <code>pred_decision</code> [, <code>labels</code> , ...])	Average hinge loss (non-regularized)
<code>matthews_corrcoef</code> ( <code>y_true</code> , <code>y_pred</code> [, ...])	Compute the Matthews correlation coefficient (MCC)

Some also work in the multilabel case:

<code>accuracy_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>normalize</code> , ...])	Accuracy classification score.
<code>classification_report</code> ( <code>y_true</code> , <code>y_pred</code> [, ...])	Build a text report showing the main classification metrics
<code>f1_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute the F1 score, also known as balanced F-score or F-measure
<code>fbeta_score</code> ( <code>y_true</code> , <code>y_pred</code> , <code>beta</code> [, <code>labels</code> , ...])	Compute the F-beta score
<code>hamming_loss</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute the average Hamming loss.
<code>jaccard_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Jaccard similarity coefficient score
<code>log_loss</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>eps</code> , <code>normalize</code> , ...])	Log loss, aka logistic loss or cross-entropy loss.
<code>multilabel_confusion_matrix</code> ( <code>y_true</code> , <code>y_pred</code> )	Compute a confusion matrix for each class or sample
<code>precision_recall_fscore_support</code> ( <code>y_true</code> , <code>y_pred</code> )	Compute precision, recall, F-measure and support for each class
<code>precision_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute the precision
<code>recall_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute the recall
<code>zero_one_loss</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>normalize</code> , ...])	Zero-one classification loss.

And some work with binary and multilabel (but not multiclass) problems:

<code>average_precision_score</code> ( <code>y_true</code> , <code>y_score</code> [, ...])	Compute average precision (AP) from prediction scores
<code>roc_auc_score</code> ( <code>y_true</code> , <code>y_score</code> [, <code>average</code> , ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

[https://scikit-learn.org/stable/modules/model\\_evaluation.html#classification-metrics](https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics)



# 扩展：多分类任务的评价指标

- **mAP**(Mean Average Percision): 平均AP
  - 所有类别的AP的算数平均
- **Kappa系数**
  - kappa系数是用在统计学中评估一致性的一种方法，取值范围是 $[-1,1]$ ，实际应用中，一般是 $[0,1]$ ，与ROC曲线中一般不会出现下凸形曲线的原理类似。这个系数的值越高，则代表模型实现的分类准确度越高。
- **铰链损失**（Hinge loss）
  - 一般用来使“边缘最大化”（maximal margin）。损失取值在 $0\sim 1$ 之间，当取值为0，表示多分类模型分类完全准确，取值为1表明完全不起作用。



### 3.3.2. Classification metrics

The `sklearn.metrics` module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values. Most implementations allow each sample to provide a weighted contribution to the overall score, through the `sample_weight` parameter.

Some of these are restricted to the binary classification case:

<code>precision_recall_curve</code> (y_true, probas_pred)	Compute precision-recall pairs for different probability thresholds
<code>roc_curve</code> (y_true, y_score[, pos_label, ...])	Compute Receiver operating characteristic (ROC)
<code>balanced_accuracy_score</code> (y_true, y_pred[, ...])	Compute the balanced accuracy

Others also work in the multiclass case:

<code>cohen_kappa_score</code> (y1, y2[, labels, weights, ...])	Cohen's kappa: a statistic that measures inter-annotator agreement.
<code>confusion_matrix</code> (y_true, y_pred[, labels, ...])	Compute confusion matrix to evaluate the accuracy of a classification
<code>hinge_loss</code> (y_true, pred_decision[, labels, ...])	Average hinge loss (non-regularized)
<code>matthews_corrcoef</code> (y_true, y_pred[, ...])	Compute the Matthews correlation coefficient (MCC)

Some also work in the multilabel case:

<code>accuracy_score</code> (y_true, y_pred[, normalize, ...])	Accuracy classification score.
<code>classification_report</code> (y_true, y_pred[, ...])	Build a text report showing the main classification metrics
<code>f1_score</code> (y_true, y_pred[, labels, ...])	Compute the F1 score, also known as balanced F-score or F-measure
<code>fbeta_score</code> (y_true, y_pred, beta[, labels, ...])	Compute the F-beta score
<code>hamming_loss</code> (y_true, y_pred[, labels, ...])	Compute the average Hamming loss.
<code>jaccard_score</code> (y_true, y_pred[, labels, ...])	Jaccard similarity coefficient score
<code>log_loss</code> (y_true, y_pred[, eps, normalize, ...])	Log loss, aka logistic loss or cross-entropy loss.
<code>multilabel_confusion_matrix</code> (y_true, y_pred)	Compute a confusion matrix for each class or sample
<code>precision_recall_fscore_support</code> (y_true, y_pred)	Compute precision, recall, F-measure and support for each class
<code>precision_score</code> (y_true, y_pred[, labels, ...])	Compute the precision
<code>recall_score</code> (y_true, y_pred[, labels, ...])	Compute the recall
<code>zero_one_loss</code> (y_true, y_pred[, normalize, ...])	Zero-one classification loss.

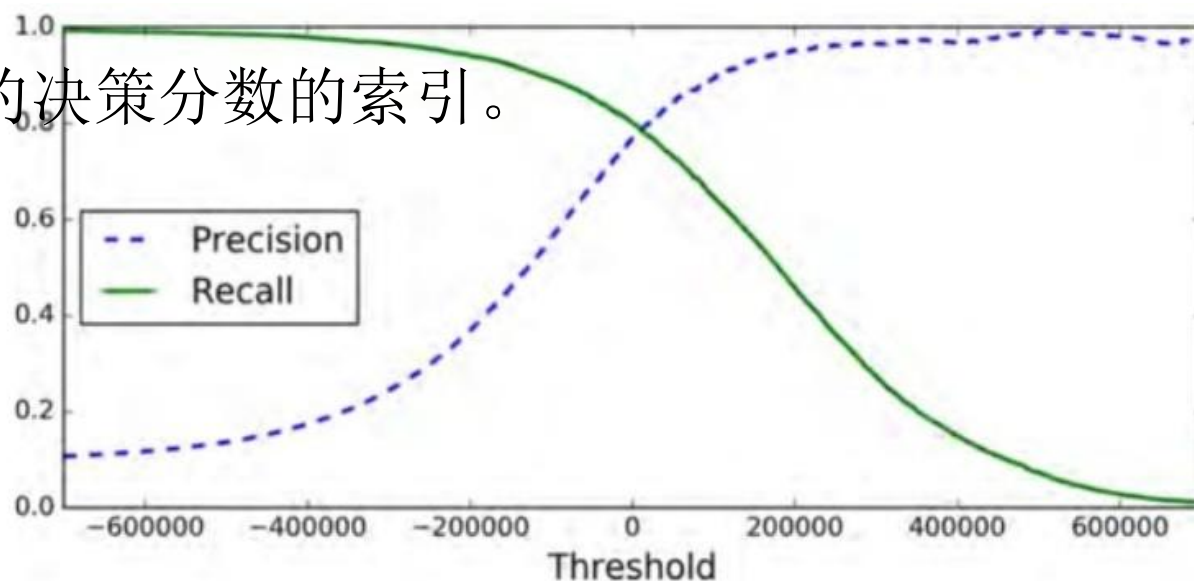
And some work with binary and multilabel (but not multiclass) problems:

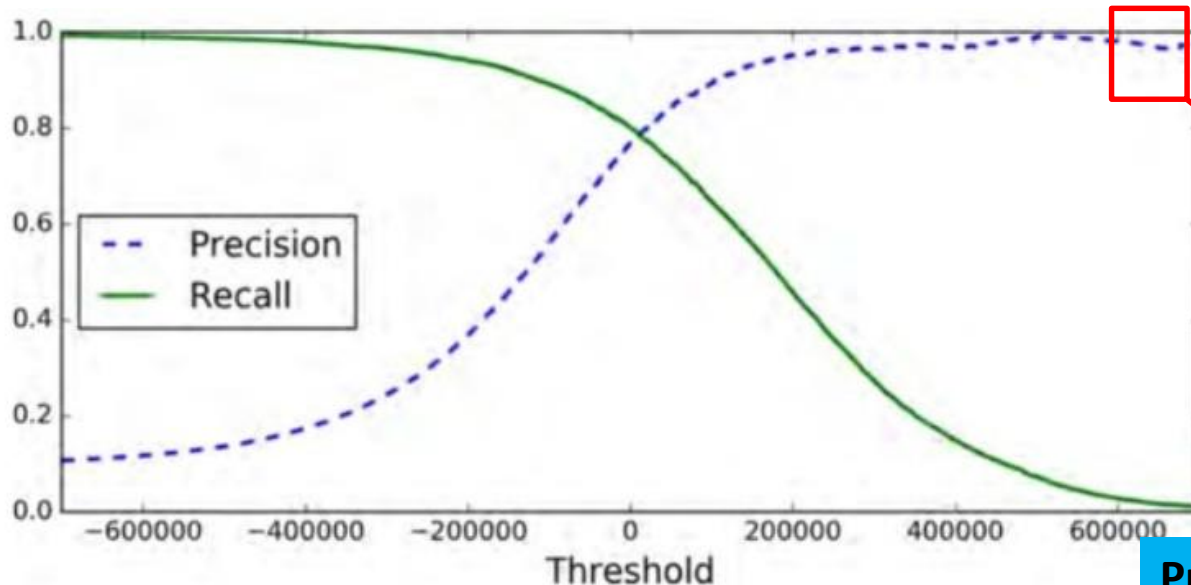
<code>average_precision_score</code> (y_true, y_score[, ...])	Compute average precision (AP) from prediction scores
<code>roc_auc_score</code> (y_true, y_score[, average, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

[https://scikit-learn.org/stable/modules/model\\_evaluation.html#classification-metrics](https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics)

# 如何权衡 满足需求的精确率和召回率？

- 查准率和查全率是一对矛盾的度量，一般而言，查准率高时，查全率往往偏低；而查全率高时，查准率往往偏低。
  - 随着**阈值变大**，Recall会变小，Precision往往会变大。
- 利用PRC或ROC来确定阈值，以满足需求的精确率和召回率
  - 找到作为阈值的决策分数的索引。





Threshold变大,  
Precision减小?

Precision: 6/8 = 75%  
Recall: 6/6 = 100%

Precision: 4/5 = 80%  
Recall: 4/6 = 67%

Precision: 3/3 = 100%  
Recall: 3/6 = 50%

Precision: 3/4 = 75%  
Recall: 3/6 = 50%

8 7 3 9 5 2 5 5 6 5 5 5

Negative predictions

Positive predictions

Score

Various thresholds

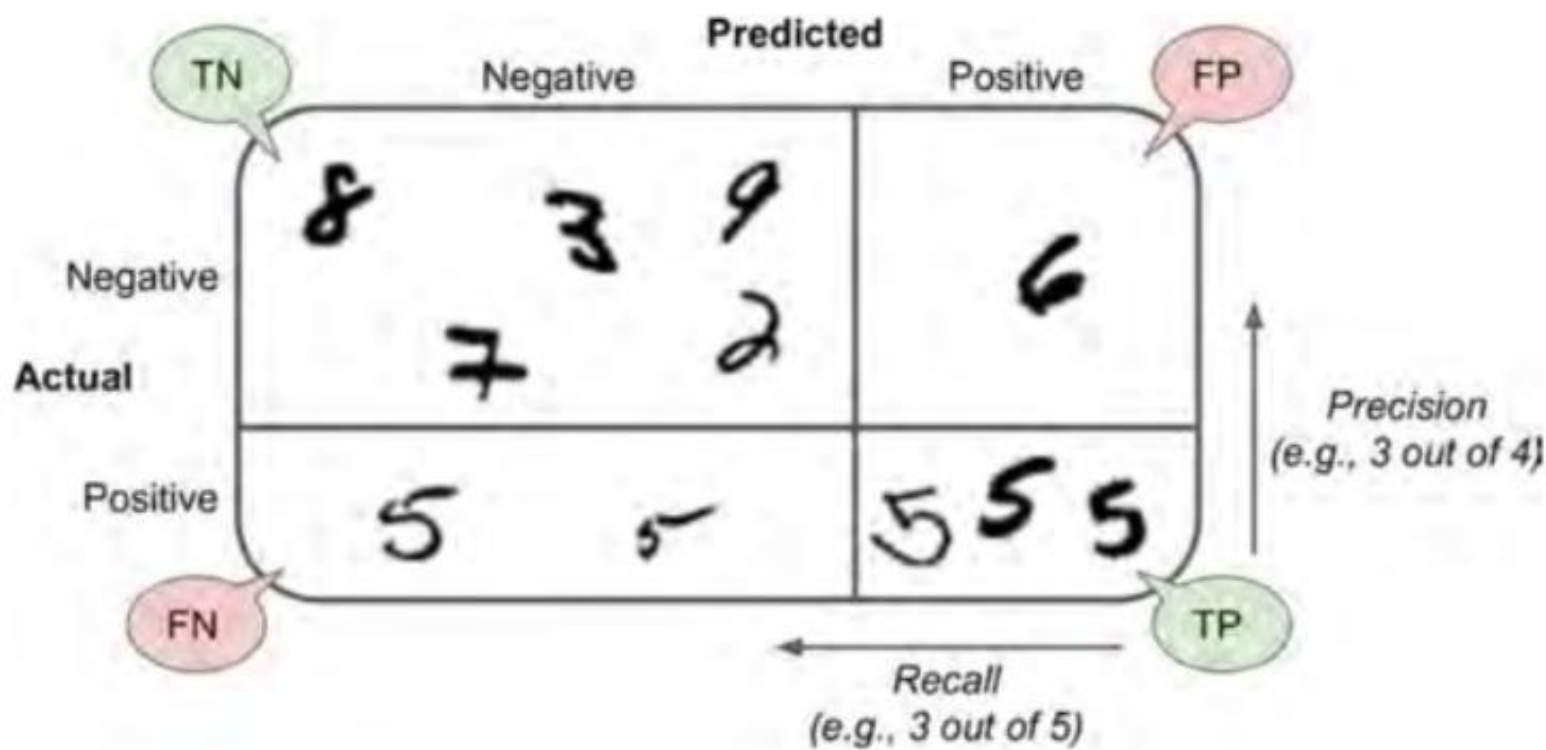
Figure 3-3. Decision threshold and precision/recall tradeoff

# 本章内容

- 为什么不直接利用线性回归模型来解决分类问题？
- 二分类问题中，逻辑回归模型训练“三步曲”
  - 分类问题中，为什么使用交叉熵（cross entropy）而不是MSE作为损失函数？
- 多分类问题中，使用Softmax回归模型
  - 对比逻辑回归模型，比较异同
- 分类器的性能评估
  - 混淆矩阵，精度和召回率，PRC， ROC
- 利用混淆矩阵进行错误分析
- 多标签分类问题
- 多输出分类问题
- 延伸过渡：
  - 层叠的逻辑回归模型 -> MLP -> DNN

# 利用混淆矩阵进行错误分析

- 回顾：二分类器的混淆矩阵 `array([[5, 1], [2, 3]])`



数字“5”探测器的结果

# 利用混淆矩阵进行错误分析

- 多分类器的混淆矩阵
  - 示例：对手写数字识别器进行错误分析

```
>>> y_train_pred = cross_val_predict(sgd_clf, X_train_scaled, y_train, cv=3)
>>> conf_mx = confusion_matrix(y_train, y_train_pred)
>>> conf_mx
array([[5725,  3, 24,  9, 10, 49, 50, 10, 39,  4],
       [ 2, 6493, 43, 25,  7, 40,  5, 10, 109,  8],
       [51, 41, 5321, 104, 89, 26, 87, 60, 166, 13],
       [47, 46, 141, 5342,  1, 231, 40, 50, 141, 92],
       [19, 29, 41, 10, 5366,  9, 56, 37, 86, 189],
       [73, 45, 36, 193, 64, 4582, 111, 30, 193, 94],
       [29, 34, 44,  2, 42, 85, 5627, 10, 45,  0],
       [25, 24, 74, 32, 54, 12,  6, 5787, 15, 236],
       [52, 161, 73, 156, 10, 163, 61, 25, 5027, 123],
       [43, 35, 26, 92, 178, 28,  2, 223, 82, 5240]])
```

行代表实际类别，列代表预测的类别

# 如何计算多分类器的混淆矩阵？

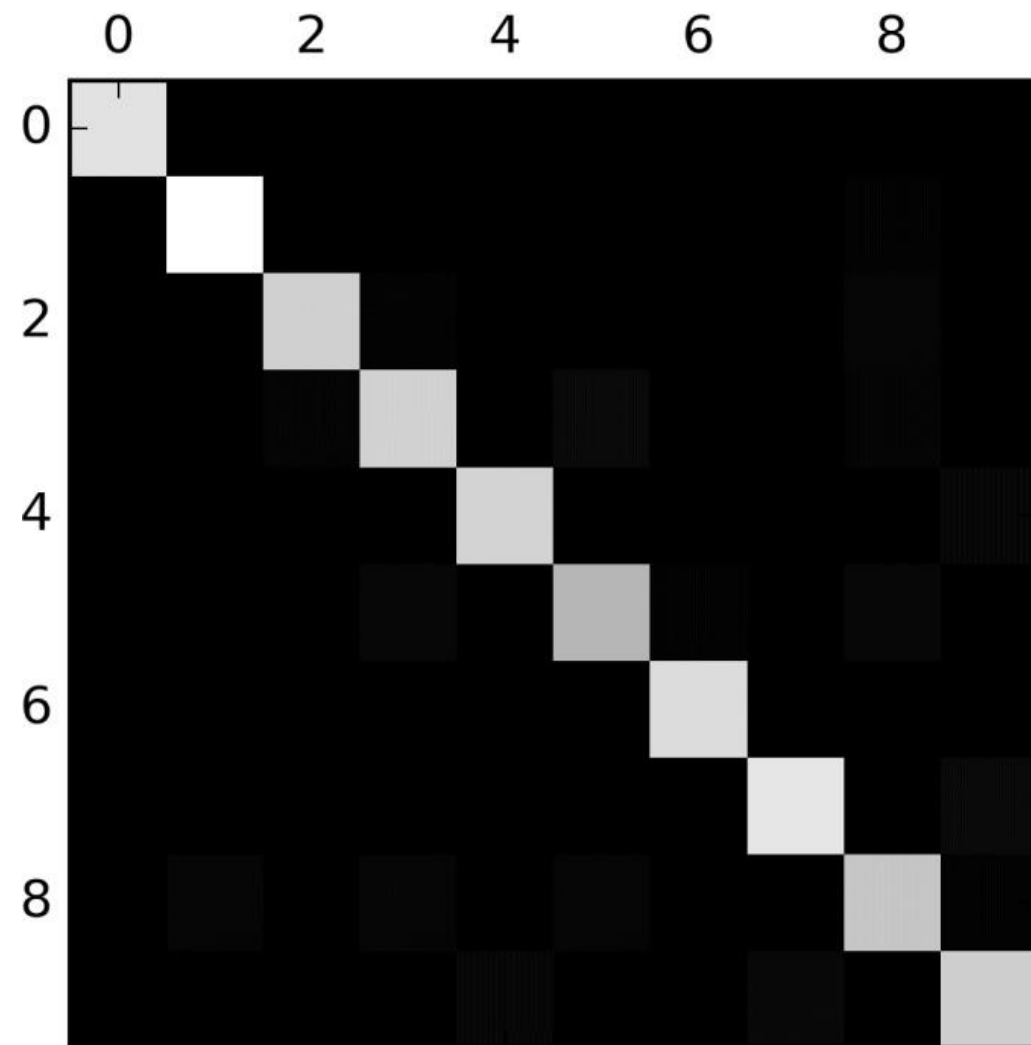
```
>>> from sklearn.metrics import confusion_matrix
>>> y_true = [2, 0, 2, 2, 0, 1]
>>> y_pred = [0, 0, 2, 2, 0, 2]
>>> confusion_matrix(y_true, y_pred)
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]])
```

```
>>> y_true = ["cat", "ant", "cat", "cat", "ant", "bird"]
>>> y_pred = ["ant", "ant", "cat", "cat", "ant", "cat"]
>>> confusion_matrix(y_true, y_pred, labels=["ant", "bird", "cat"])
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]])
```

行代表实际类别，列代表预测的类别

# 利用混淆矩阵进行错误分析

行代表实际类别，列代表预测的类别



```
array([[5725, 3, 24, 9, 10, 49, 50, 10, 39, 4],  
       [ 2, 6493, 43, 25, 7, 40, 5, 10, 109, 8],  
       [ 51, 41, 5321, 104, 89, 26, 87, 60, 166, 13],  
       [ 47, 46, 141, 5342, 1, 231, 40, 50, 141, 92],  
       [ 19, 29, 41, 10, 5366, 9, 56, 37, 86, 189],  
       [ 73, 45, 36, 193, 64, 4582, 111, 30, 193, 94],  
       [ 29, 34, 44, 2, 42, 85, 5627, 10, 45, 0],  
       [ 25, 24, 74, 32, 54, 12, 6, 5787, 15, 236],  
       [ 52, 161, 73, 156, 10, 163, 61, 25, 5027, 123],  
       [ 43, 35, 26, 92, 178, 28, 2, 223, 82, 5240]])
```

值越大，图片越亮

观察到：数字5对应的格子看起来比其他数字要暗淡许多。这可能是数据集当中数字5的图片比较少，又或者是分类器对于数字5的表现不如其他数字那么好。

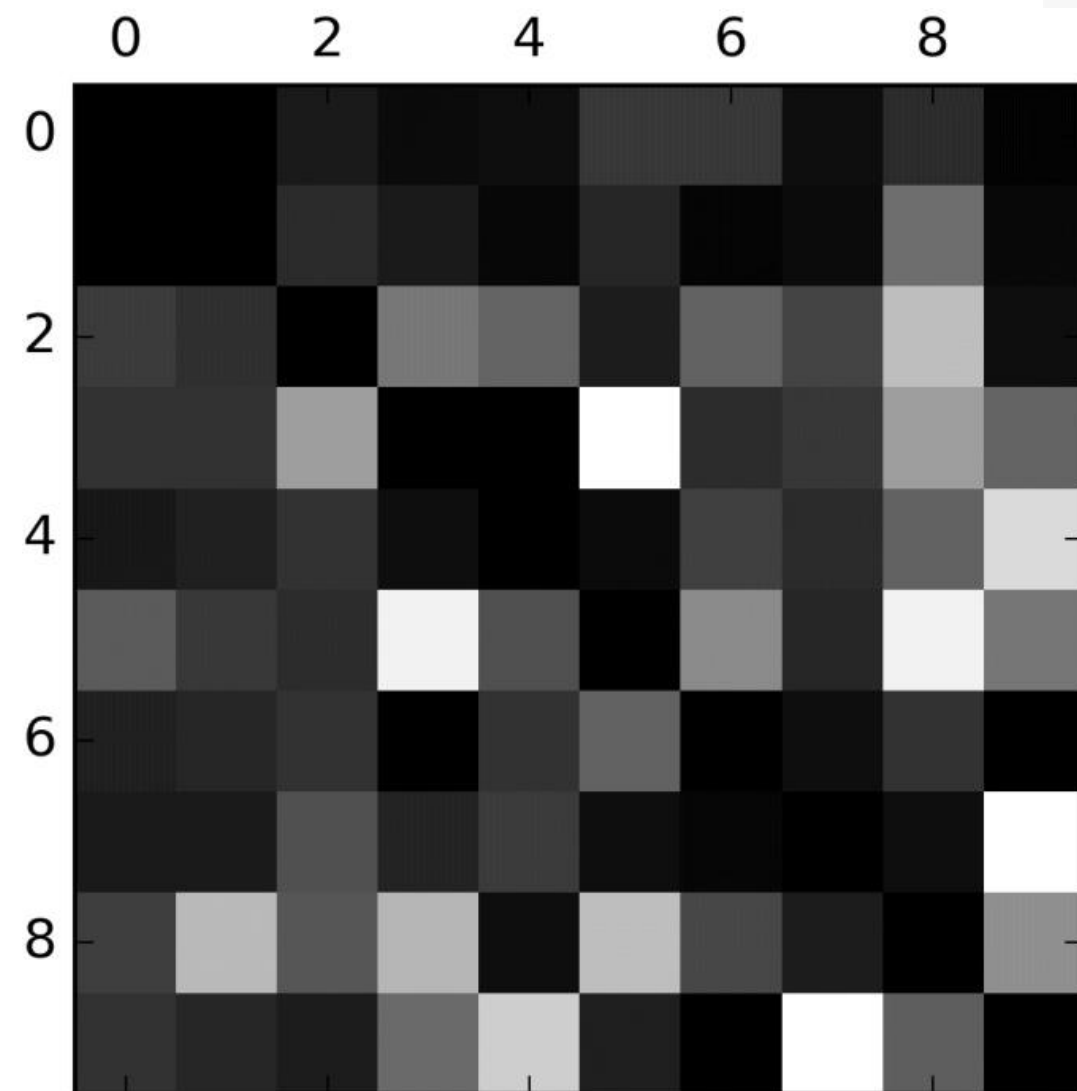
- Q1: 该分类器将5误判为3的次数为?
- Q2: 完美的分类器的混淆矩阵的特点是?



# 利用混淆矩阵进行错误分析

行代表实际类别，列代表预测的类别

```
array([[5725, 3, 24, 9, 10, 49, 50, 10, 39, 4],
       [2, 6493, 43, 25, 7, 40, 5, 10, 109, 8],
       [51, 41, 5321, 104, 89, 26, 87, 60, 166, 13],
       [47, 46, 141, 5342, 1, 231, 40, 50, 141, 92],
       [19, 29, 41, 10, 5366, 9, 56, 37, 86, 189],
       [73, 45, 36, 193, 64, 4582, 111, 30, 193, 94],
       [29, 34, 44, 2, 42, 85, 5627, 10, 45, 0],
       [25, 24, 74, 32, 54, 12, 6, 5787, 15, 236],
       [52, 161, 73, 156, 10, 163, 61, 25, 5027, 123],
       [43, 35, 26, 92, 178, 28, 2, 223, 82, 5240]])
```



1. 第8、9列相当亮，说明许多图片被误分成数字8或者数字9。
2. 第8、9行也相当亮，说明数字8、数字9经常被误以为是其他数字。
3. 有些行相当黑，比如第一行，这意味着大部分的数字1被正确分类（一些被误分类为数字8）。
4. 误差图不是严格对称的。比如，比起将数字8误分类为数字5的数量，有更多的数字5被误分类为数字8。

# 本章内容

- 为什么不直接利用线性回归模型来解决分类问题？
- 二分类问题中，逻辑回归模型训练“三步曲”
  - 分类问题中，为什么使用交叉熵（cross entropy）而不是MSE作为损失函数？
- 多分类问题中，使用Softmax回归模型
  - 对比逻辑回归模型，比较异同
- 分类器的性能评估
  - 混淆矩阵，精度和召回率，PRC， ROC
- 利用混淆矩阵进行错误分析
- 多标签分类问题
- 多输出分类问题
- 延伸过渡：
  - 层叠的逻辑回归模型 -> MLP -> DNN

# 多标签分类问题

- 多个二值标签的分类问题
  - 每个样本可以有多个类别
- 示例1：人脸识别器
- 示例2：给照片打标签



多分类问题



标签为：sea, sunset

# 多标签分类的评价指标

1. Hamming loss(汉明损失), 表示所有label中错误样本的比例, 所以该值越小则网络的分类能力越强。计算公式如下。

$$\text{HammingLoss}(x_i, y_i) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{\text{xor}(x_i, y_i)}{|L|},$$

其中:  $|D|$ 表示样本总数,  $|L|$ 表示标签总数,  $x_i$ 和 $y_i$ 分别表示预测结果和ground truth。xor表示异或运算。

2. Jaccard index(杰卡德指数), 概念挺陌生, 公式是再熟悉不过了。其中:  $T$ 表示ground truth,  $P$ 表示预测结果。再观察这个公式, 和检测算法中的IOU多么相近。

$$\frac{|T \cap P|}{|T \cup P|}$$

3. 精度、召回率和F1值。其中精度计算公式为  $\frac{|T \cap P|}{|P|}$ , 召回率计算公式为  $\frac{|T \cap P|}{|T|}$ , F1值的计算为精度和召回率的调和平均数。

4. 准确匹配。这个是最严格的标准了, 是预测结果和ground truth完全一致时的样本数与总的样本数的比值。

### 3.3.2. Classification metrics

The `sklearn.metrics` module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values. Most implementations allow each sample to provide a weighted contribution to the overall score, through the `sample_weight` parameter.

Some of these are restricted to the binary classification case:

<code>precision_recall_curve</code> ( <code>y_true</code> , <code>probas_pred</code> )	Compute precision-recall pairs for different probability thresholds
<code>roc_curve</code> ( <code>y_true</code> , <code>y_score</code> [, <code>pos_label</code> , ...])	Compute Receiver operating characteristic (ROC)
<code>balanced_accuracy_score</code> ( <code>y_true</code> , <code>y_pred</code> [, ...])	Compute the balanced accuracy

Others also work in the multiclass case:

<code>cohen_kappa_score</code> ( <code>y1</code> , <code>y2</code> [, <code>labels</code> , <code>weights</code> , ...])	Cohen's kappa: a statistic that measures inter-annotator agreement.
<code>confusion_matrix</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute confusion matrix to evaluate the accuracy of a classification
<code>hinge_loss</code> ( <code>y_true</code> , <code>pred_decision</code> [, <code>labels</code> , ...])	Average hinge loss (non-regularized)
<code>matthews_corrcoef</code> ( <code>y_true</code> , <code>y_pred</code> [, ...])	Compute the Matthews correlation coefficient (MCC)

Some also work in the multilabel case:

<code>accuracy_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>normalize</code> , ...])	Accuracy classification score.
<code>classification_report</code> ( <code>y_true</code> , <code>y_pred</code> [, ...])	Build a text report showing the main classification metrics
<code>f1_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute the F1 score, also known as balanced F-score or F-measure
<code>fbeta_score</code> ( <code>y_true</code> , <code>y_pred</code> , <code>beta</code> [, <code>labels</code> , ...])	Compute the F-beta score
<code>hamming_loss</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute the average Hamming loss.
<code>jaccard_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Jaccard similarity coefficient score
<code>log_loss</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>eps</code> , <code>normalize</code> , ...])	Log loss, aka logistic loss or cross-entropy loss.
<code>multilabel_confusion_matrix</code> ( <code>y_true</code> , <code>y_pred</code> )	Compute a confusion matrix for each class or sample
<code>precision_recall_fscore_support</code> ( <code>y_true</code> , <code>y_pred</code> )	Compute precision, recall, F-measure and support for each class
<code>precision_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute the precision
<code>recall_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute the recall
<code>zero_one_loss</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>normalize</code> , ...])	Zero-one classification loss.

And some work with binary and multilabel (but not multiclass) problems:

<code>average_precision_score</code> ( <code>y_true</code> , <code>y_score</code> [, ...])	Compute average precision (AP) from prediction scores
<code>roc_auc_score</code> ( <code>y_true</code> , <code>y_score</code> [, <code>average</code> , ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

[https://scikit-learn.org/stable/modules/model\\_evaluation.html#classification-metrics](https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics)

# Scikit-learn中的模型评估方法

## 3.3. 模型评估: 量化预测的质量

校验者: @飓风 @小瑶 @FAME @v @Loopy 译者: @小瑶 @片刻 @那伊抹微笑

有 3 种不同的 API 用于评估模型预测的质量:

- **Estimator score method (估计器得分的方法)**: Estimators (估计器) 有一个 `score` (得分) 方法, 为其解决的问题提供了默认的 evaluation criterion (评估标准)。在这个页面上没有相关讨论, 但是在每个 estimator (估计器) 的文档中会有相关的讨论。
- **Scoring parameter (评分参数)**: Model-evaluation tools (模型评估工具) 使用 `cross-validation` (如 `model_selection.cross_val_score` 和 `model_selection.GridSearchCV`) 依靠 internal scoring strategy (内部 scoring (得分) 策略)。这在 [scoring 参数: 定义模型评估规则](#) 部分讨论。
- **Metric functions (指标函数)**: `metrics` 模块实现了针对特定目的评估预测误差的函数。这些指标在以下部分部分详细介绍 [分类指标](#), [多标签排名指标](#), [回归指标](#) 和 [聚类指标](#)。

<https://sklearn.apachecn.org/docs/0.21.3/32.html>

# 本章内容

- 为什么不直接利用线性回归模型来解决分类问题？
- 二分类问题中，逻辑回归模型训练“三步曲”
  - 分类问题中，为什么使用交叉熵（cross entropy）而不是MSE作为损失函数？
- 多分类问题中，使用Softmax回归模型
  - 对比逻辑回归模型，比较异同
- 分类器的性能评估
  - 混淆矩阵，精度和召回率，PRC， ROC
- 利用混淆矩阵进行错误分析
- 多标签分类问题
- 多输出分类问题
- 延伸过渡：
  - 层叠的逻辑回归模型 -> MLP -> DNN

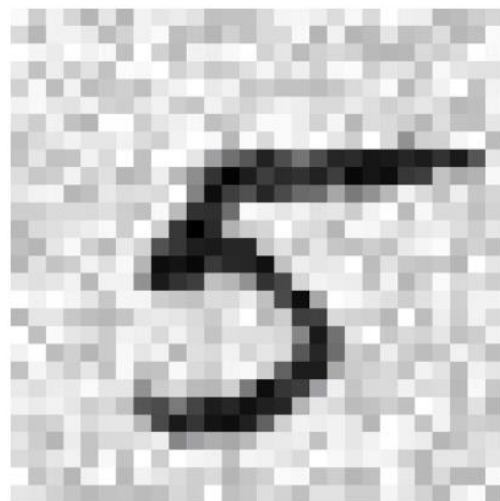
# 多输出分类问题

- 是多标签分类的简单泛化
- 多输出分类器 **VS.** 多标签分类器
  - 相同点：分类器的输出都是多标签
  - 不同点：
    - 多标签分类器中每个标签是二分类
    - 多输出分类器中每个标签可以是多分类，也可以是具体的标量。

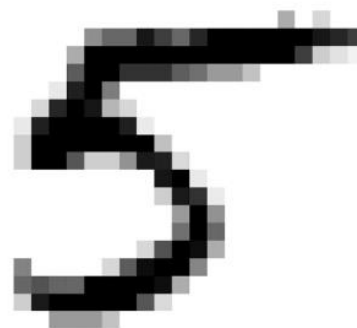


# 示例：多输出分类问题

- 建立一个系统，它可以去除图片当中的噪音。
- 它将一张混有噪音的图片作为输入，期待它输出一张干净的数字图片，用一个像素强度的数组表示，就像MNIST图片那样。



**Input**



**Output**

# 总结：回归和分类任务——关键概念

英文概念	中文概念	定义
Under/over-fitting	欠拟合/ 过拟合	使用了过于简单/复杂的模型。
Evaluation metric	评价指标	一个衡量模型效果的数值。
Mean squared error	均方差	回归模型所使用的一种评价指标。
Cross-validation	交叉验证	为了更好地估计准确率，把训练数据分成2份（或者多份）独立的训练/ 测试数据集的方法。
Holdout method	Holdout 方法	一种交叉验证的方法，保留一份测试数据集用于模型测试。
K-fold cross-validation	K折交叉 验证	一种交叉验证的方法，数据集被分为K份独立的子集，每次取出一份作 为测试集，其余数据用来训练模型。
Confusion matrix	混淆矩阵	用于比较分类结果和实际测得值的一种矩阵。
ROC - Receiver operator characteristic	ROC	一种用于记录真阳性、假阳性、真阴性、假阴性的数值。
AUC - Area under the ROC curve	ROC曲线 下面积	ROC曲线下方的面积大小。
Tuning parameter	调整参数	机器学习算法的一个内部参数，比如内核平滑回归算法的窗宽参数。
Grid search	网格搜索	优化模型参数时采用的一种暴力搜索策略。

# 回顾：机器学习项目的通用工作流程

- 1 定义问题：软件架构设计、确定评价指标
- 2 获取数据：自动化方式
- 3 研究数据：可视化方式，相关性研究等
- 4 准备数据：数据清理、特征选择及处理
- 5 研究模型：确定验证集上的评估方法、列出可能的模型并训练，选择最有希望的3~5个模型
- 6 微调模型：寻找最佳超参数，模型融合，评估泛化性能
- 7 展示解决方案：将工作进行文档化总结展示
- 8 启动、监视、维护系统：投入使用

# 总结：回归和分类任务——从超参数角度

- 影响模型复杂度的超参数
  - 模型定义中自带的超参数
  - 正则化因子：修正了损失函数的定义
    - **注意：**正则化模型的学习曲线的误差计算是基于未添加正则项的损失函数
- 影响自动化训练过程的超参数
  - 学习率
  - 训练前是否需要特征进行标准化或归一化
  - SGD/BGD/MBGD
    - BatchSize, Epoch

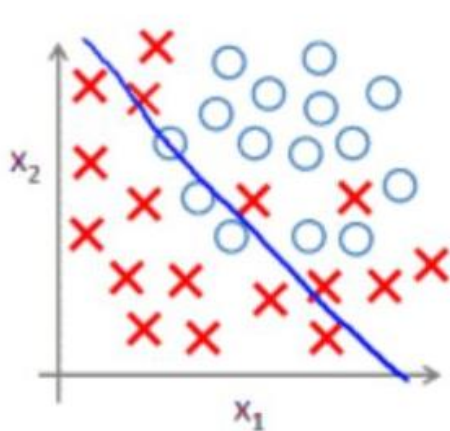
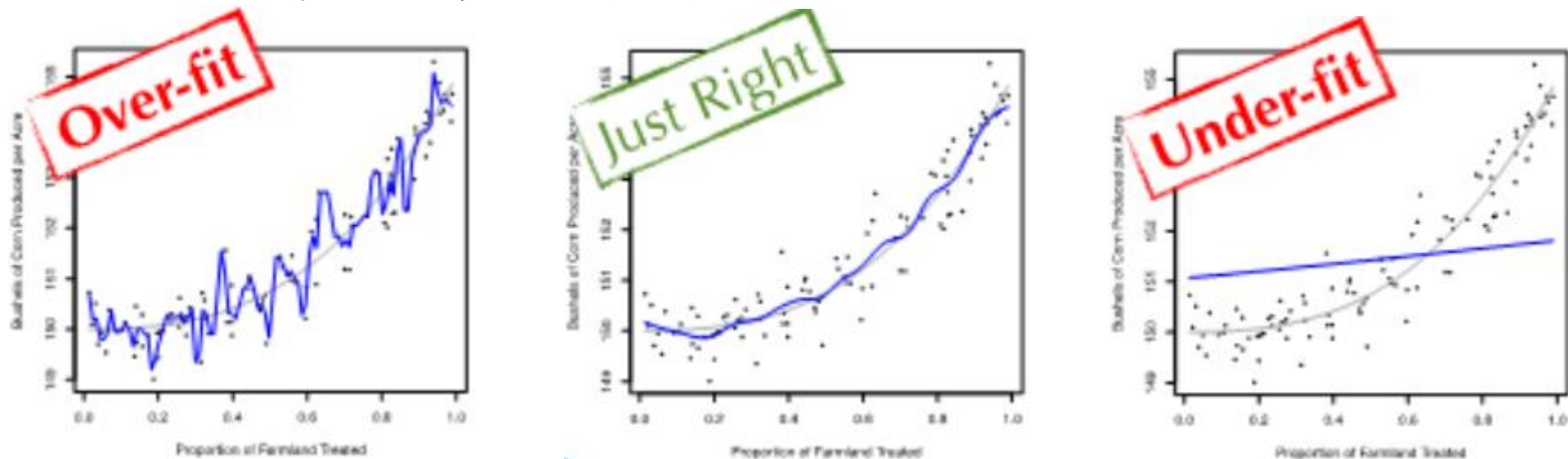
Model	Parameters to optimize	Good range of values
Linear Regression	<ul style="list-style-type: none"> <li>fit_intercept</li> <li>normalize</li> </ul>	<ul style="list-style-type: none"> <li>True / False</li> <li>True / False</li> </ul>
Ridge	<ul style="list-style-type: none"> <li>alpha</li> <li>Fit_intercept</li> <li>Normalize</li> </ul>	<ul style="list-style-type: none"> <li>0.01, 0.1, 1.0, 10, 100</li> <li>True/False</li> <li>True/False</li> </ul>
k-neighbors	<ul style="list-style-type: none"> <li>N_neighbors</li> <li>p</li> </ul>	<ul style="list-style-type: none"> <li>2, 4, 8, 16 ....</li> <li>2, 3</li> </ul>
SVM	<ul style="list-style-type: none"> <li>C</li> <li>Gamma</li> <li>class_weight</li> </ul>	<ul style="list-style-type: none"> <li>0.001, 0.01.....10...100...1000</li> <li>'Auto', RS*</li> <li>'Balanced' , None</li> </ul>
Logistic Regression	<ul style="list-style-type: none"> <li>Penalty</li> <li>C</li> </ul>	<ul style="list-style-type: none"> <li>L1 or l2</li> <li>0.001, 0.01.....10...100</li> </ul>
Naive Bayes (all variations)	NONE	NONE
Lasso	<ul style="list-style-type: none"> <li>Alpha</li> <li>Normalize</li> </ul>	<ul style="list-style-type: none"> <li>0.1, 1.0, 10</li> <li>True/False</li> </ul>
Random Forest	<ul style="list-style-type: none"> <li>N_estimators</li> <li>Max_depth</li> <li>Min_samples_split</li> <li>Min_samples_leaf</li> <li>Max features</li> </ul>	<ul style="list-style-type: none"> <li>120, 300, 500, 800, 1200</li> <li>5, 8, 15, 25, 30, None</li> <li>1, 2, 5, 10, 15, 100</li> <li>1, 2, 5, 10</li> <li>Log2, sqrt, None</li> </ul>
Xgboost	<ul style="list-style-type: none"> <li>Eta</li> <li>Gamma</li> <li>Max_depth</li> <li>Min_child_weight</li> <li>Subsample</li> <li>Colsample_bytree</li> <li>Lambda</li> <li>alpha</li> </ul>	<ul style="list-style-type: none"> <li>0.01,0.015, 0.025, 0.05, 0.1</li> <li>0.05-0.1,0.3,0.5,0.7,0.9,1.0</li> <li>3, 5, 7, 9, 12, 15, 17, 25</li> <li>1, 3, 5, 7</li> <li>0.6, 0.7, 0.8, 0.9, 1.0</li> <li>0.6, 0.7, 0.8, 0.9, 1.0</li> <li>0.01-0.1, 1.0 , RS*</li> <li>0, 0.1, 0.5, 1.0 RS*</li> </ul>

# 总结：回归和分类 任务——从模型效果角度分析

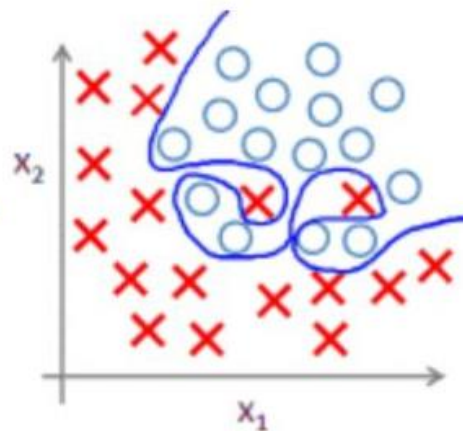
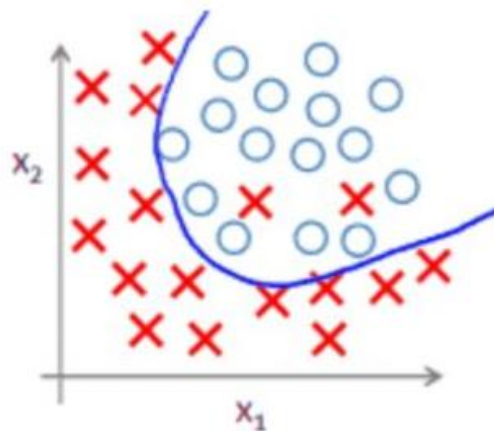
- 展示模型拟合效果图
- 利用训练误差和验证误差
- 利用训练集和验证集上的学习曲线
  - 回归任务中，学习曲线的纵轴为MSE
  - 分类任务中，学习曲线的纵轴为？

# 总结：回归和分类 任务——从模型效果角度分析

- 展示模型拟合效果图



欠拟合



过拟合

# 总结：回归和分类 任务——从模型效果角度分析

- 利用训练误差和验证误差

- 假设你的算法表现如下：

- 训练错误率= 1%
    - 开发错误率= 11%

过拟合

- 假设你的算法表现如下：

- 训练错误率= 15%
    - 开发错误率= 16%

欠拟合

- 假设你的算法表现如下：

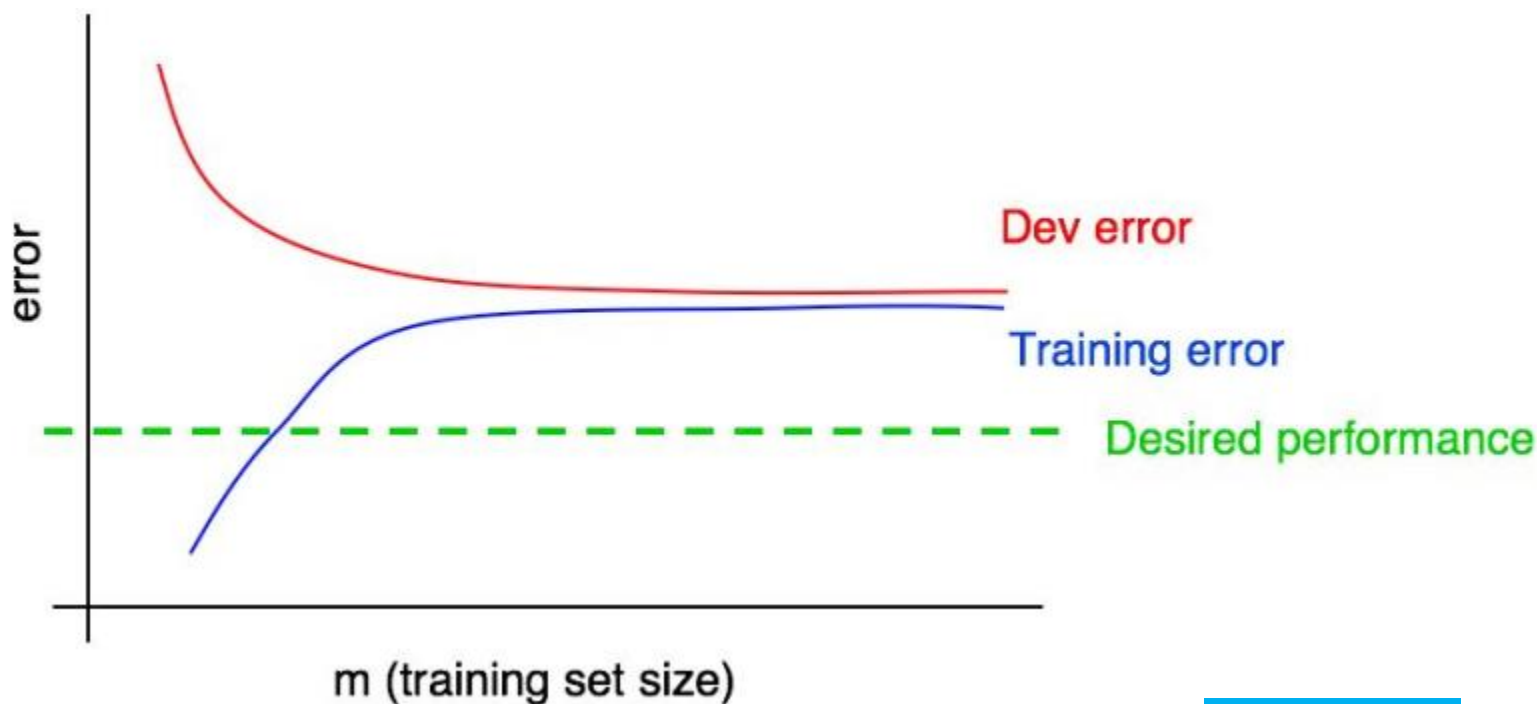
- 训练错误率= 0.5%
    - 开发错误率= 1%





# 总结：回归和分类 任务——从模型效果角度分析

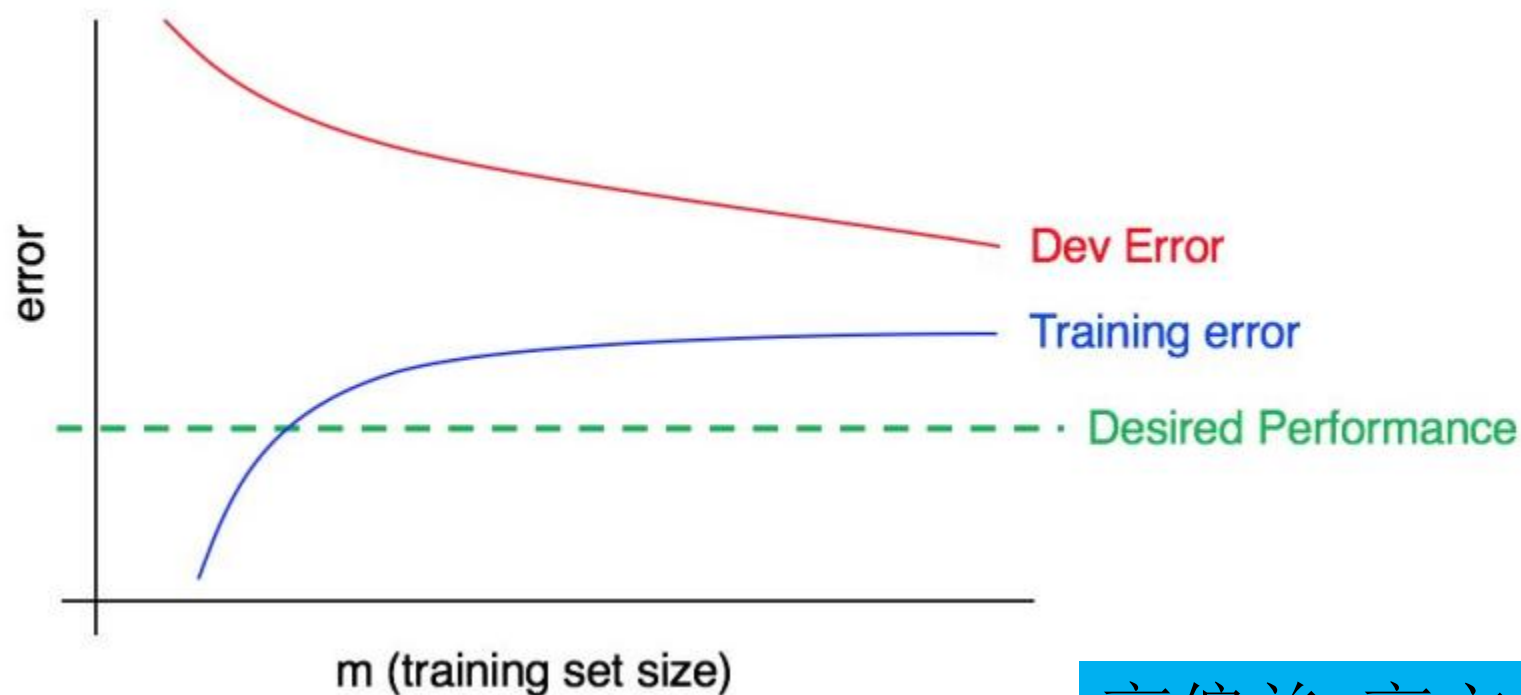
- 利用训练集和验证集上的学习曲线



欠拟合

# 总结：回归和分类 任务——从模型效果角度分析

- 利用训练集和验证集上的学习曲线



高偏差+高方差

# 总结：回归和分类 任务——从模型效果角度分析

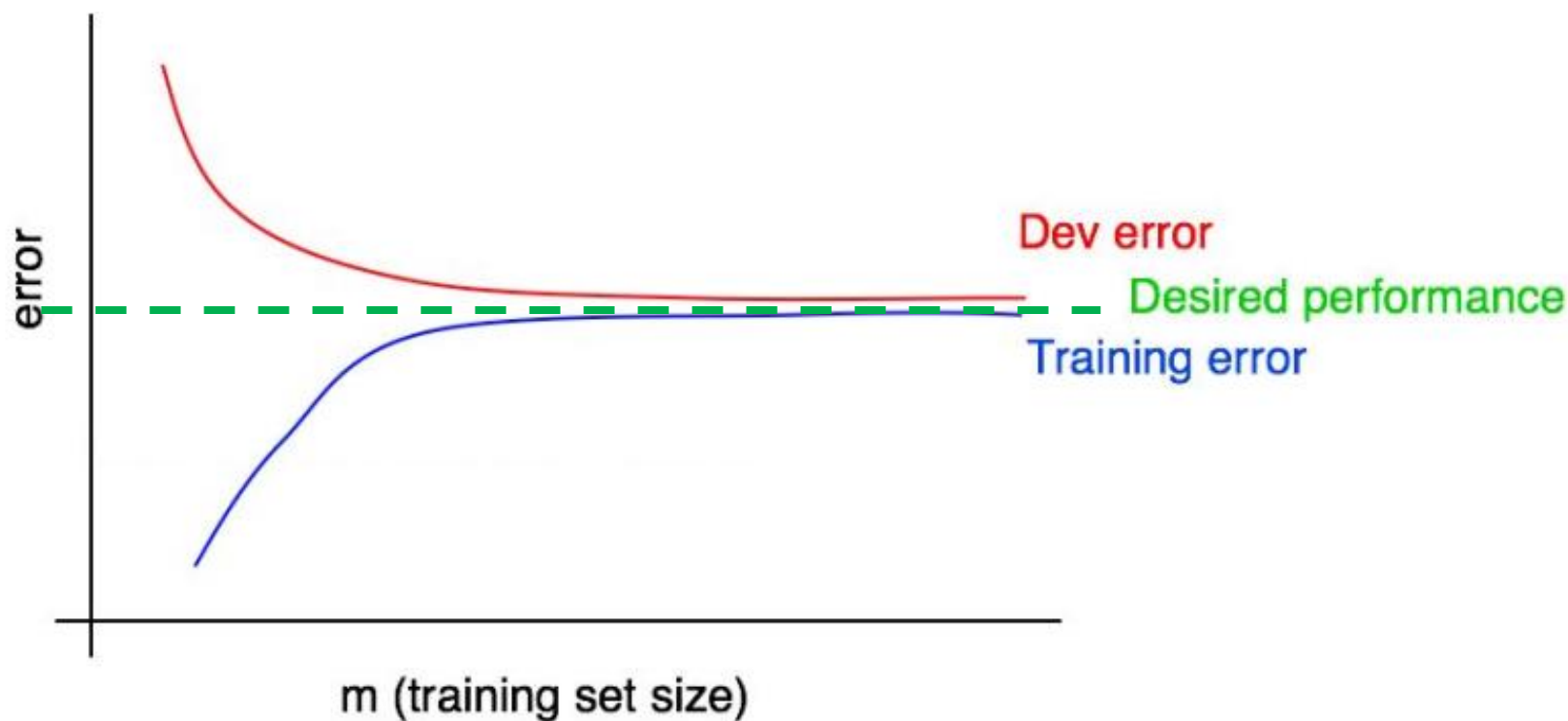
- 利用训练集和验证集上的学习曲线



过拟合

# 总结：回归和分类 任务——从模型效果角度分析

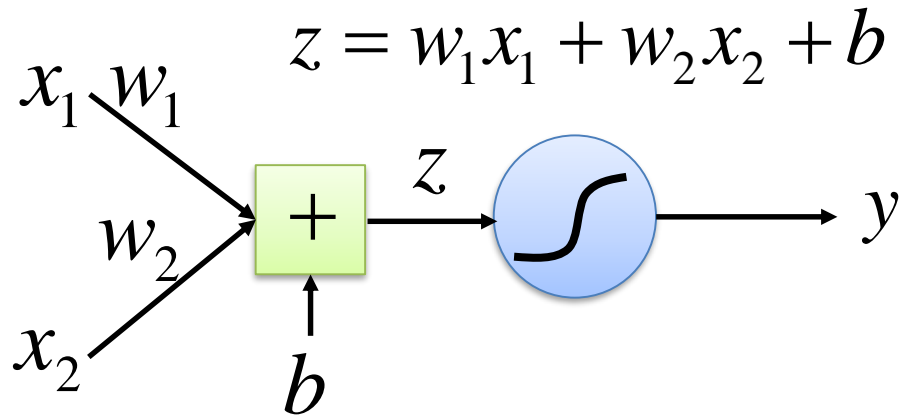
- 利用训练集和验证集上的学习曲线



# 本章内容

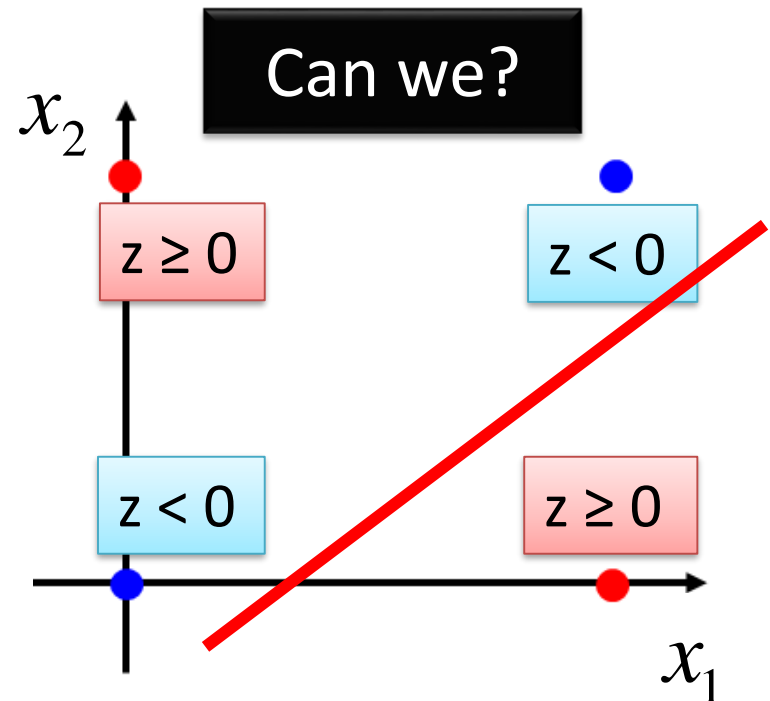
- 为什么不直接利用线性回归模型来解决分类问题？
- 二分类问题中，逻辑回归模型训练“三步曲”
  - 分类问题中，为什么使用交叉熵（cross entropy）而不是MSE作为损失函数？
- 多分类问题中，使用Softmax回归模型
  - 对比逻辑回归模型，比较异同
- 分类器的性能评估
  - 混淆矩阵，精度和召回率，PRC， ROC
- 利用混淆矩阵进行错误分析
- 多标签分类问题
- 多输出分类问题
- 延伸过渡：
  - 层叠的逻辑回归模型 -> MLP -> DNN

# Limitation of Logistic Regression



$$\begin{cases} \text{Class1} & y \geq 0.5 \quad (z \geq 0) \\ \text{Class2} & y < 0.5 \quad (z < 0) \end{cases}$$

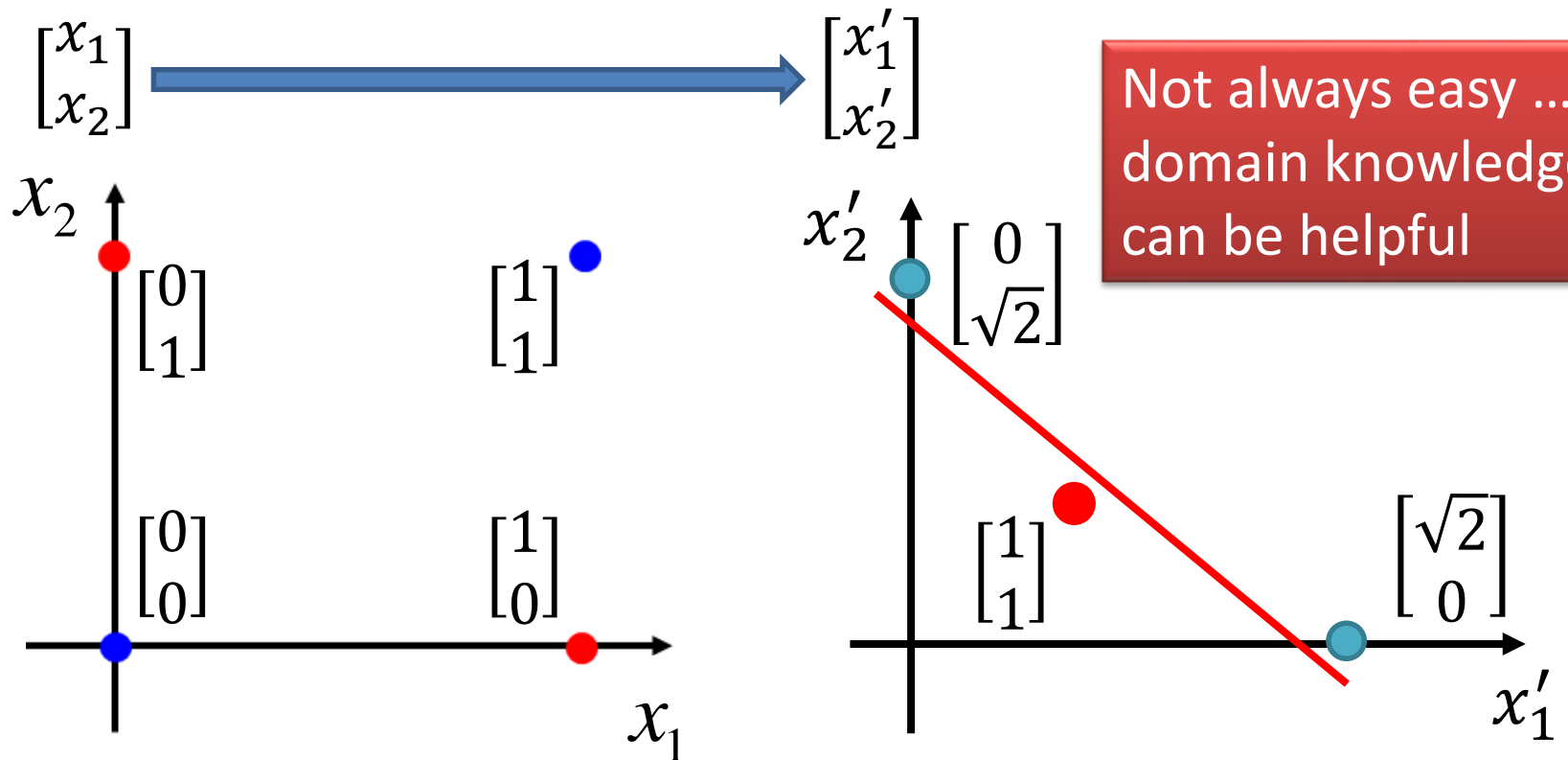
Input Feature		Label
$x_1$	$x_2$	
0	0	Class 2
0	1	Class 1
1	0	Class 1
1	1	Class 2



# Limitation of Logistic Regression

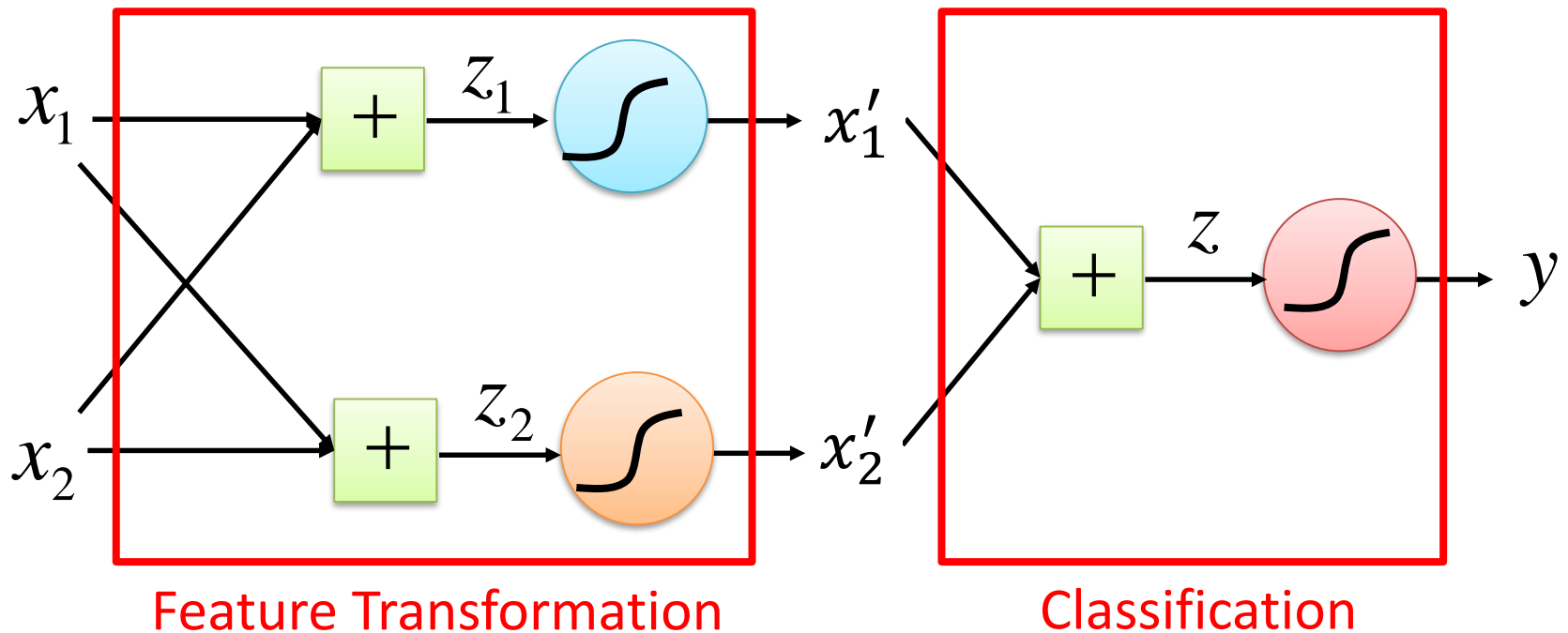
- **Feature transformation**

$x'_1$ : distance to  $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$   
 $x'_2$ : distance to  $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$



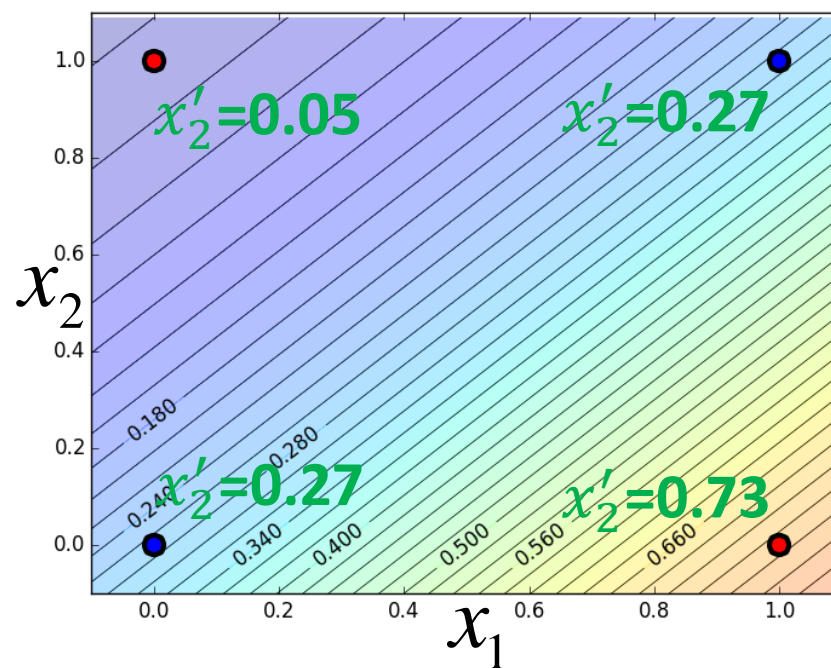
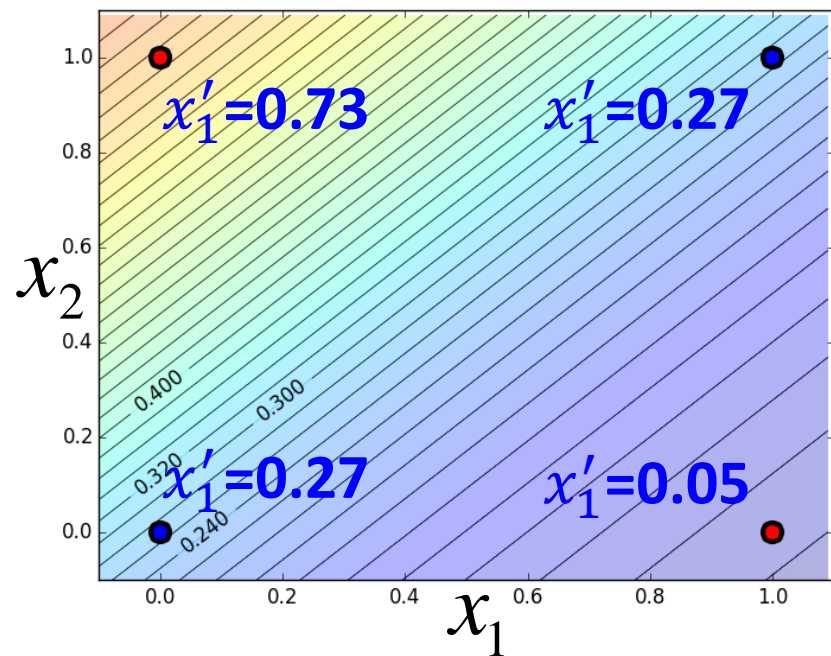
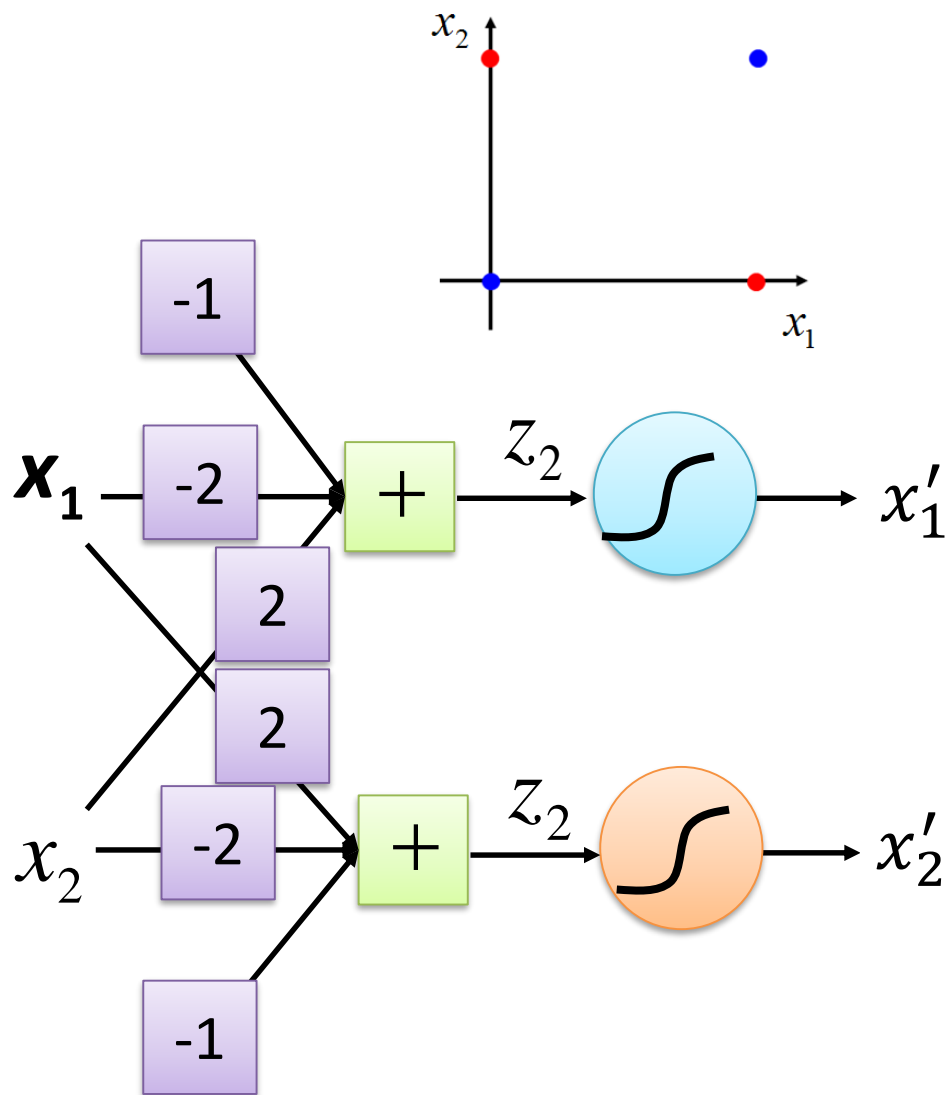
# Limitation of Logistic Regression

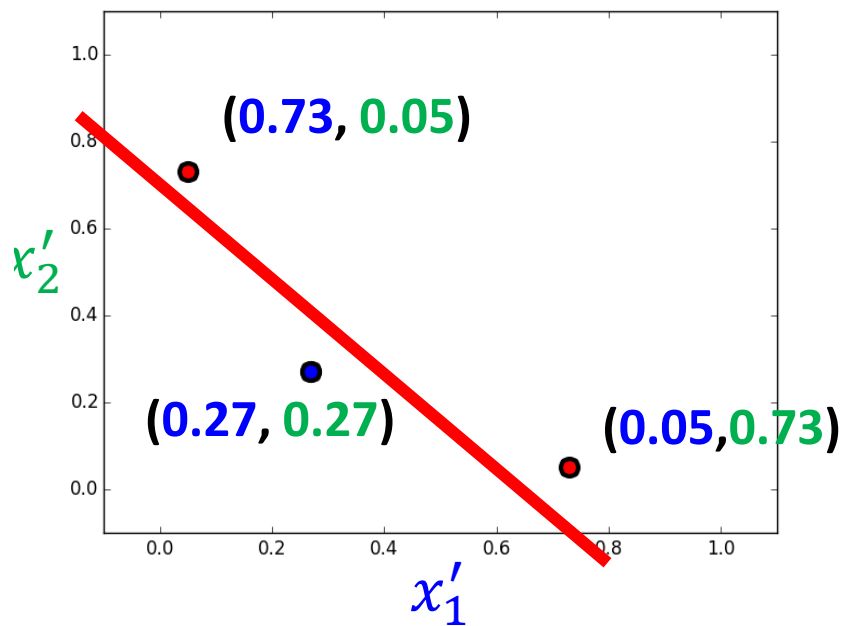
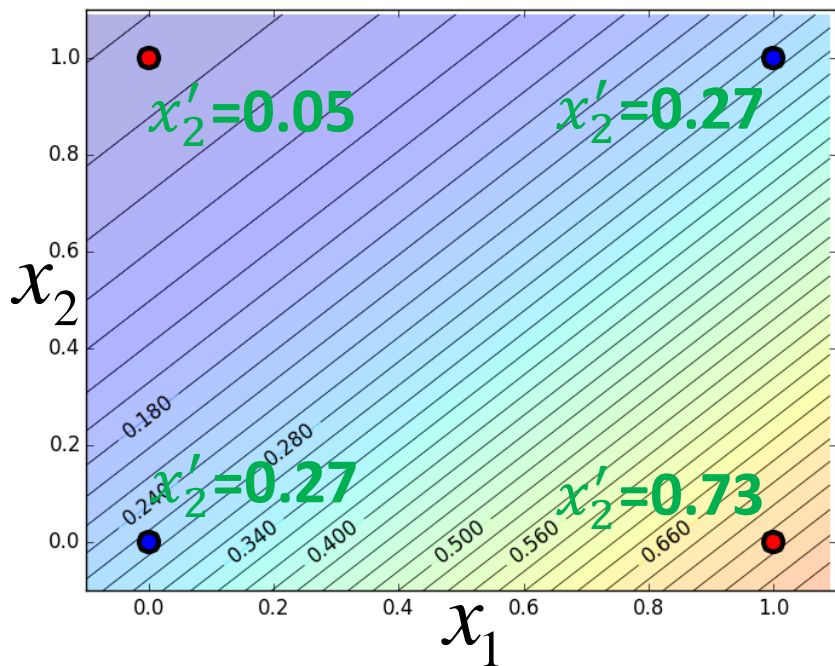
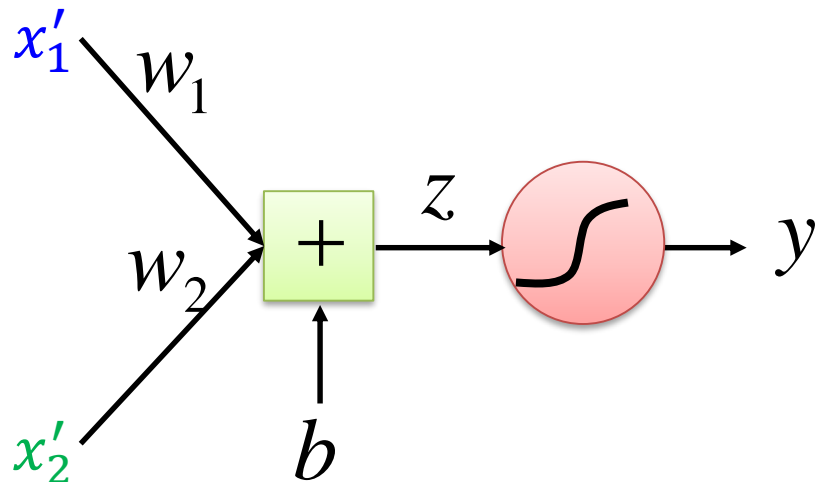
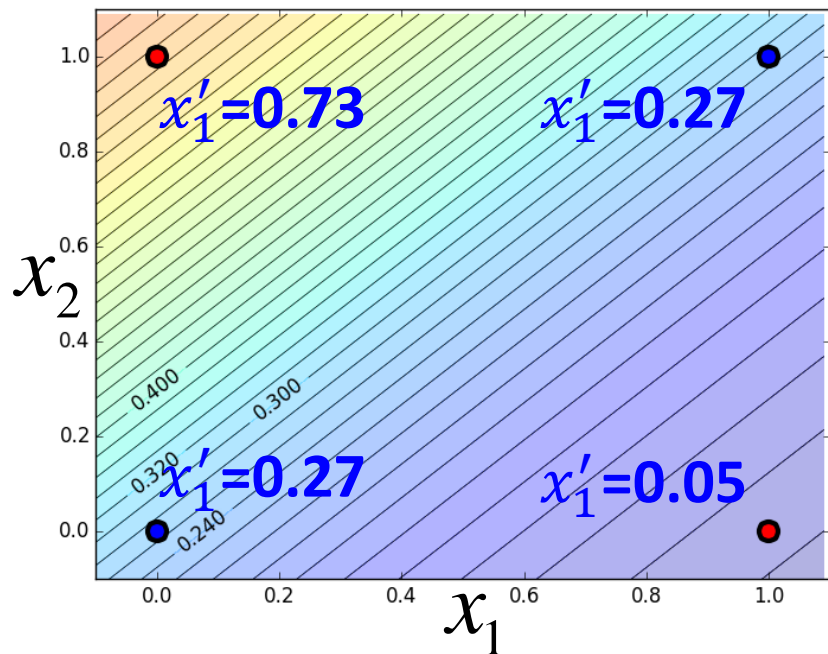
- Cascading logistic regression models



(ignore bias in this figure)

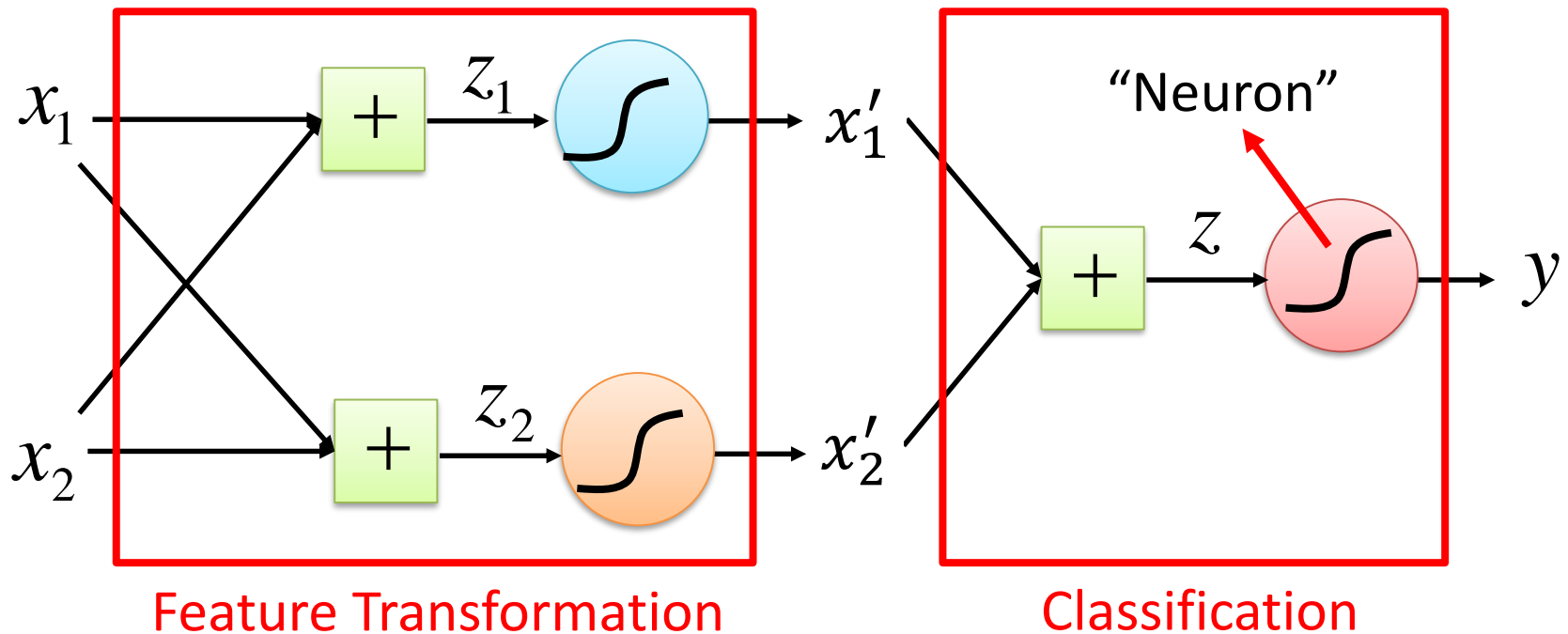
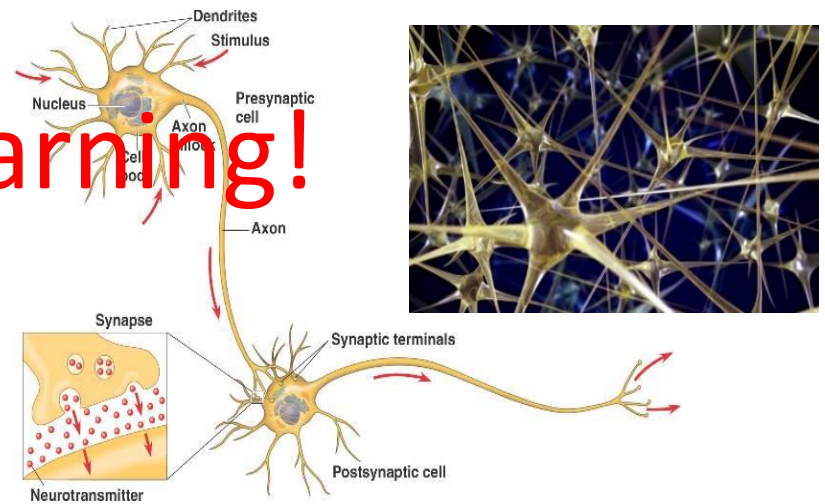






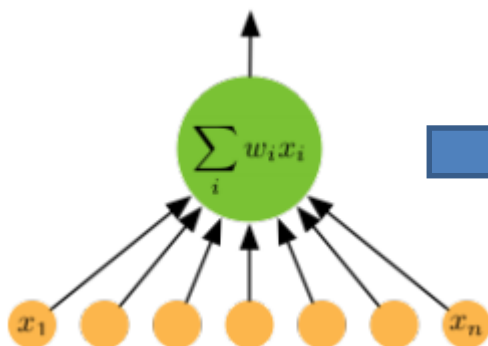
# Deep Learning!

All the parameters of the logistic regressions are jointly learned.

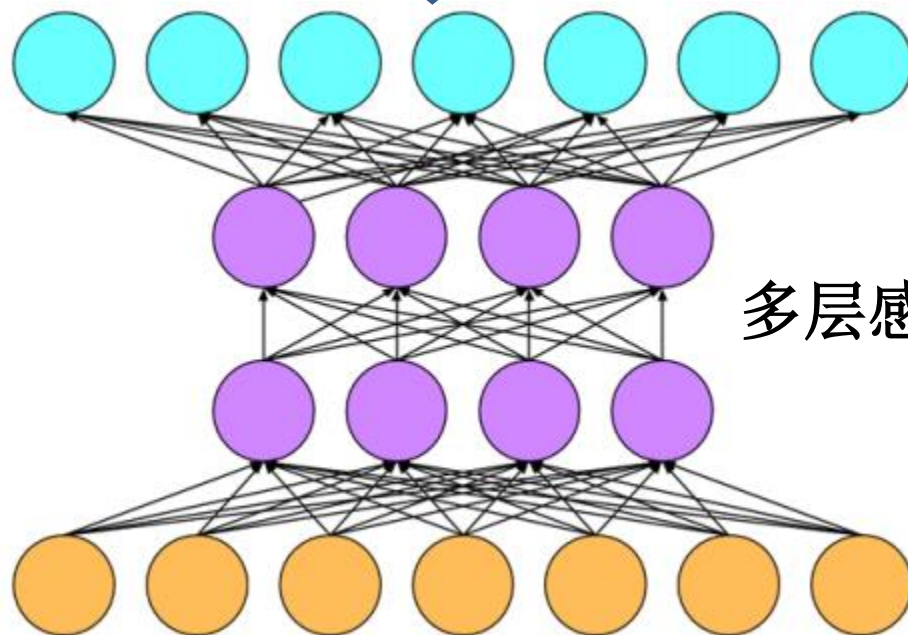
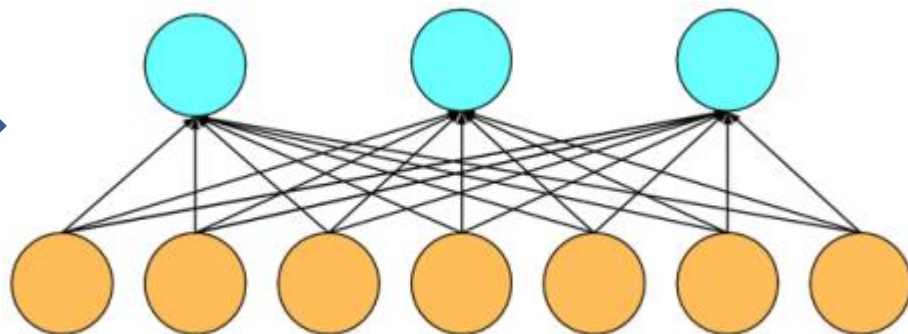


**Neural Network**

线性回归



多类逻辑回归



多层感知机

多个范例驱动

机器学习项目的通用工作流程

- 1 定义问题
- 2 获取数据
- 3 研究数据
- 4 准备数据
- 5 研究模型
- 6 微调模型
- 7 展示解决方案
- 8 启动、监视、维护系统

机器学习 -> 深度学习  
**Scikit-Learn keras**

1 定义问题

2 获取数据

3 研究数据

4 准备数据

5 研究模型

6 微调模型

7 展示解决方案

8 启动、监视、维护系统

- ◆ 从机器学习到深度学习：
  - 回归问题，分类问题
  - CNN、RNN、GAN
  - 迁移学习
- ◆ 如何训练模型？（“三步曲”）
- ◆ 如何选择最佳超参数？
- ◆ 如何评估和提升模型泛化性能
- ◆ 欠拟合？过拟合？
  - 怎么判断？怎么解决

**Scikit-Learn keras**

# 回顾：机器学习中的关键术语

- 机器学习中模型的种类：(从应用场景的角度划分)
  - 监督学习、非监督学习、半监督学习、强化学习、迁移学习
- 数据：
  - 3类数据集：
    - 训练集：利用训练集上的误差来训练模型
    - 验证集：利用验证集上的误差 来挑出备选模型，确定各个备选模型的最佳超参数，并从中挑选出最佳模型
    - 测试集：用于评估模型的表现
  - 样本：（输入，输出），一条数据为一个样本
  - 标签/标注：样本数据中的输出项
- 2个计算过程：
  - 学习/训练/优化：目标是找到最佳函数来表达输入输出之间的关系
    - 理想的最佳函数，应在优化和泛化的平衡点处，应刚好在欠拟合和过拟合的界线上
  - 推理/预测/泛化：将找到的最佳函数应用到测试数据/真实数据上，做出推理/预测。

# 回顾：机器学习中的关键术语

- 两类函数：
  - 最佳函数：（模型训练的终极目标是找到这个函数）
  - 成本/代价/损失函数：用来定义训练过程中找到的函数的好坏程度
- 两类参数：
  - 参数：模型训练过程中自动找到
  - 超参数：程序员来选择/指定
    - 半自动：网格搜索、随机搜索
    - 手动：程序员设定
- 训练过程中，模型的状态：
  - 欠拟合
  - 过拟合
  - 正好



# 回顾：机器学习中的关键术语

- 两个评估指标：
  - 模型评估标准（在训练数据集和验证数据集上）：优化指标，即损失函数。通常用于模型的训练中。
    - 损失函数的取值 越小，则越好。
  - 模型测试标准（在测试集上）：验收指标。用于测试模型的泛化性能
    - 比如：样本平衡的分类问题中常用的验收指标 ROC AUC
    - 但是，ROC AUC不能直接被优化，故不能作为损失函数，而是将ROC AUC的替代指标 交叉熵作为优化指标，即损失函数。
- 评估方法：
  - 留出验证集(holdout)
  - 交叉验证(Cross Validation)