

# 自然语言处理

杨海钦  
2025-2026-1 学期  
基于Yulia, Graham, Diyi, Jixing等课件更新

# 面试问题

- 假设你需要为公司训练一个“情感分类器”（输入一段文本，输出“积极”、“消极”或“中性”），请回答以下问题：
  - 你计划如何评估分类器的性能，用什么指标？
  - 你若选择逻辑回归分类器作为基线模型，为什么？
  - 你计划如何提取特征？
  - 如何调参(包括哪些参数)，怎么快速找到合适的参数？

# 大纲

- 文本分类的评估指标
  - K折交叉验证
- 逻辑回归
  - 定义
  - 损失函数
  - 优化函数
  - 多项式逻辑回归

# 后续课程，我们将探讨

1. 如何将文本“吸收/digest”成函数可用的形式  $F$ ?

(关键词：特征、特征提取、特征选择、表征)

feature, extraction/selection, representation

2. 我们可以用什么样的策略来创建决策函数  $f$ ?

(关键词：建模 models)

3. 如何评估决策函数  $s$ ?

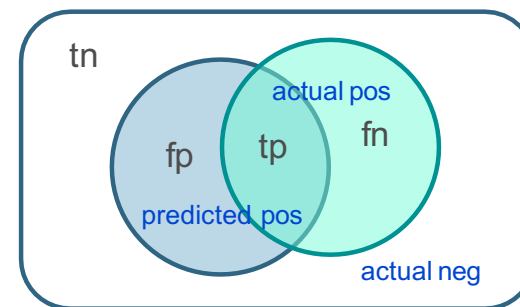
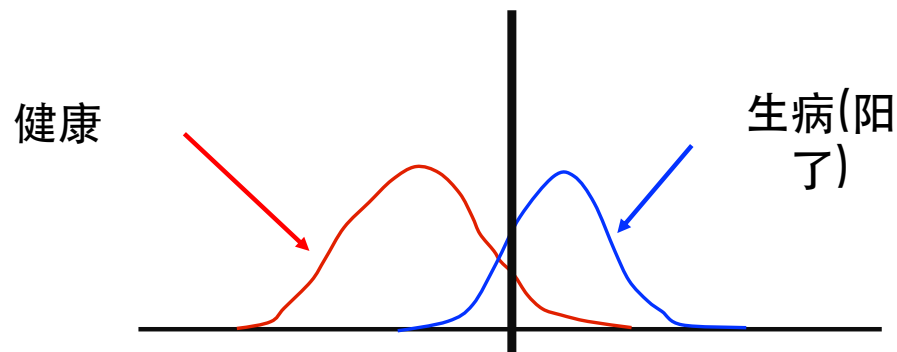
(关键词：评估 evaluation)

# 如何评估决策函数 $s$ ?

# 分类评估指标

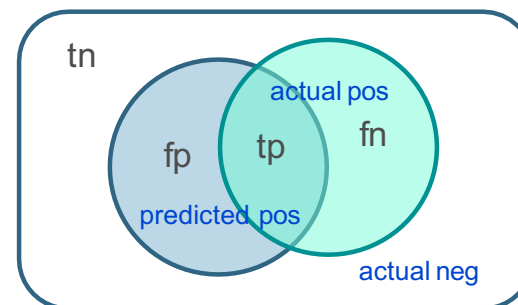
- 列联表(Contingency Table): 将模型的预测结果与正确结果进行比较
  - 亦称为: 混淆矩阵 Confusion matrix

	actual pos	actual neg
predicted pos	真阳 true positive (tp)	假阳 false positive (fp)
predicted neg	假阴 false negative (fn)	真阴 true negative (tn)



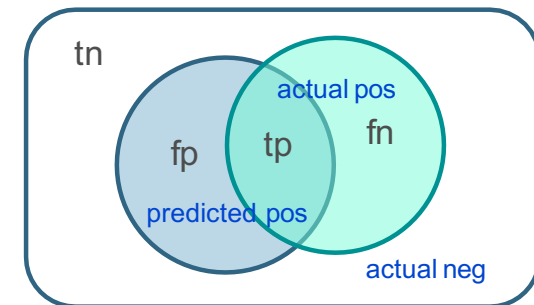
# 分类评估指标

- 信息检索评估指标: 精确率(Precision), 召回率(Recall)



# 分类评估

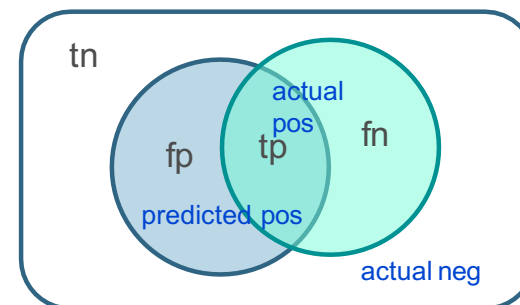
- 精度率(Precision)是在文本分类情况下，系统判定为正样本的正确比例
  - 系统判断为“正样本/positive”的文件中确实是“正样本/positive”的文件的百分比
- 每类(平均)精确度
  - $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$





# 分类评估指标

- 召回率(Recall)是在文本分类情况下，实际正样本中被判对比例
  - 它是所有正样本被系统判定为“正样本/positive”文件的百分比
- 每类(平均)精确度
  - $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$

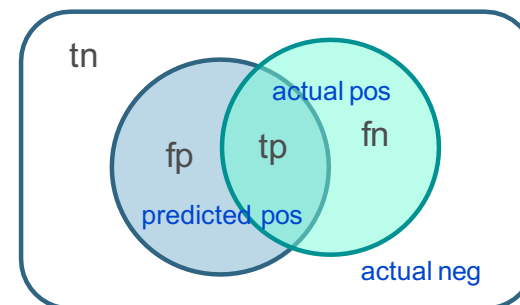


# 分类评估指标

- 我们经常想要权衡精确度P和召回率R
  - 通常：精确度越高，召回率越低
  - 可以画精确率-召回率曲线
- 把P和R合并成一个测度更方便
  - F值是一种测度

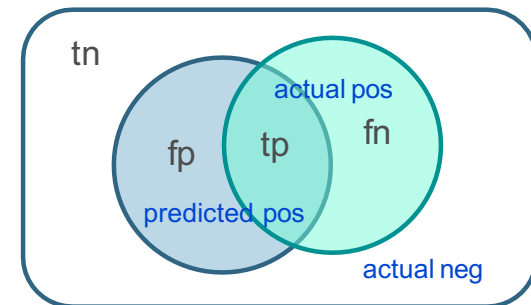
$$F_{\beta} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R},$$

$$\beta = 1 \Rightarrow F_1 = \frac{2PR}{P + R}$$



# 分类评估指标

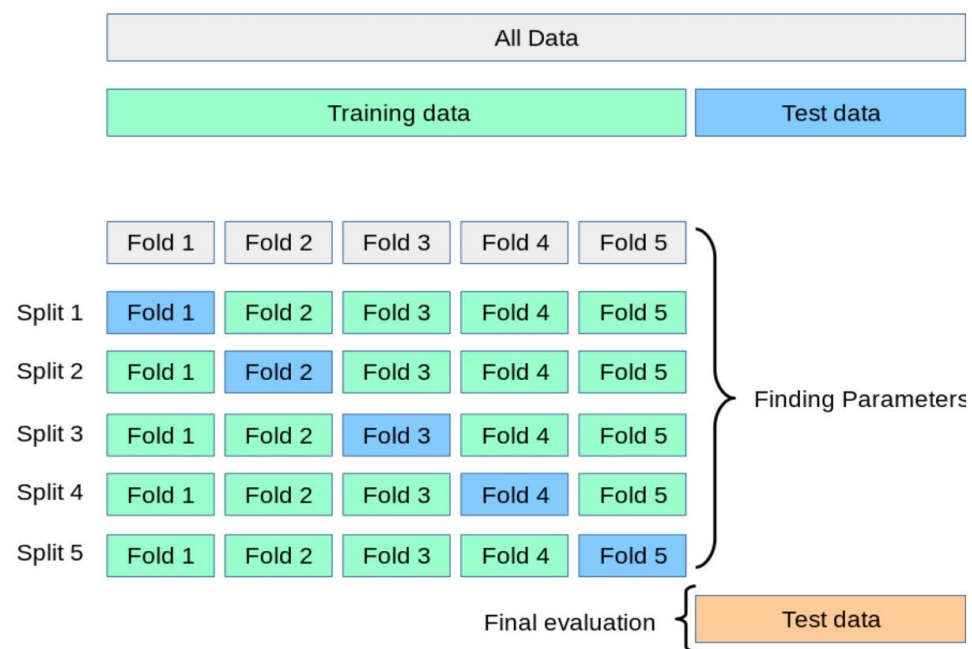
- 其他性能指标：准确率(Accuracy)和错误率(Error)
  - 准确率是系统正确预测的项的比例
  - 错误率是剩余的比例(错误预测的比例)
- 定义
  - $\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{TN} + \text{FN})$
  - $\text{Error} = 1 - \text{Accuracy}$



# 分类常用流程

- 将训练数据分成 $k$ 折（如， $k=10$ ）
- 重复 $k$ 次：在 $k-1$ 折上进行训练，并循环地在剩余折上进行测试
- 取 $k$ 折的平均值

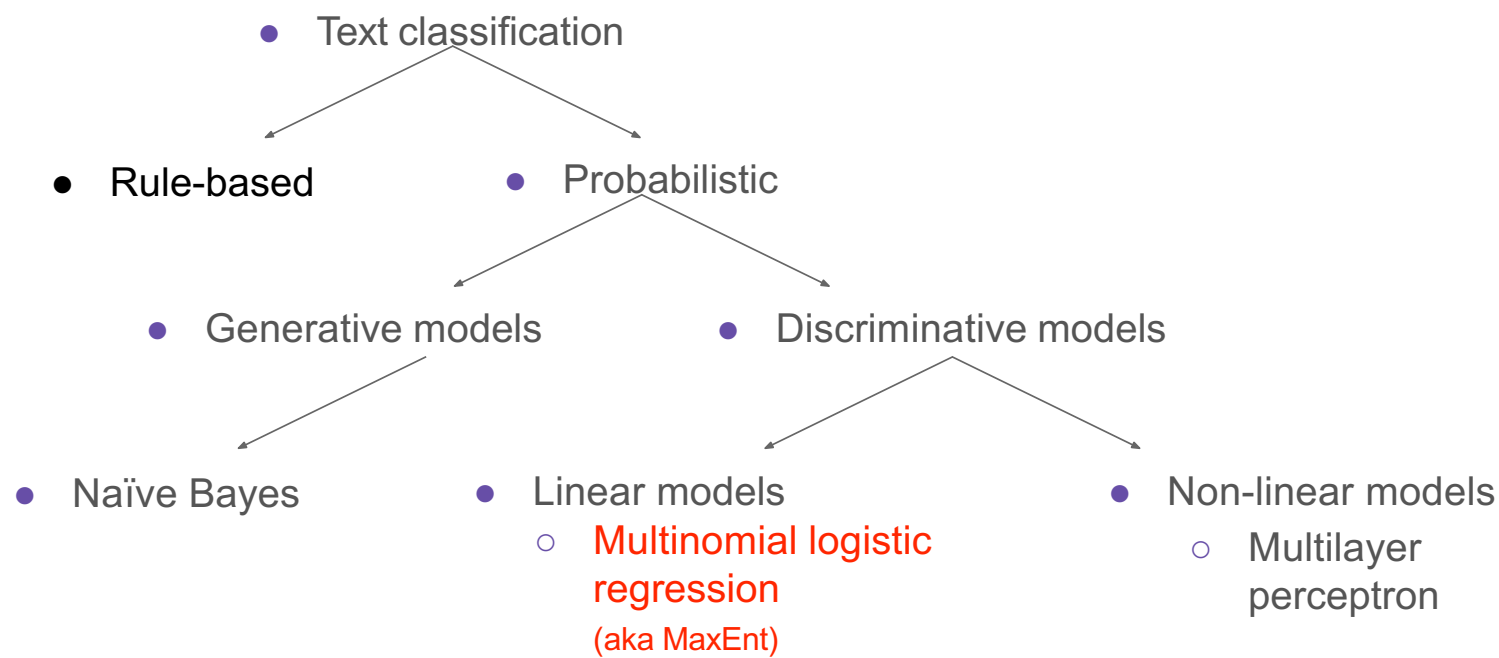
# k折交叉验证 k-fold Cross Validation



- 特殊情况: 留一交叉验证 (Leave-one-out cross validation)

# In-class Practice I

# 判定式线性模型：逻辑回归



# 逻辑回归分类器

## Logistic Regression Classifier

- 自然科学和社会科学中的重要分析工具
- 监督机器学习分类工具的基线/baseline
- 也是神经网络的基础
- 阅读材料：
  - [SLP3]Ch.5, <https://web.stanford.edu/~jurafsky/slp3/5.pdf>



# 文本分类

- 输入

- 输出

# 二分类的逻辑回归

- 训练样本
- 关键步骤: 对每个  $x^{(i)}$  ,

# 逻辑回归的特征

- 每个特征  $x_i \in \{x_1, x_2, \dots, x_n\}$  的权重值  $w_i \in \{w_1, w_2, \dots, w_n\}$  说明  $x_i$  的重要性
- 例如
  - $x_i = \text{"review contains 'awesome'"}: w_i = +10$
  - $x_j = \text{"review contains horrible"}: w_j = -10$
  - $x_k = \text{"review contains 'mediocre'"}: w_k = -2$

# 一个样本的逻辑回归

- 输入样本:
  - $x^{(i)} : \{x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}\}$
- 权重: 每个特征对应一个值,  $W = [w_1, w_2, \dots, w_n]$ 
  - 其他表示:  $\theta = [\theta_1, \theta_2, \dots, \theta_n]$
- 输出: 预测类别  $\hat{y}^{(i)} \in \{0, 1\}$
- 多项式逻辑回归:  $\hat{y}^{(i)} \in \{0, 1, 3, 4, 5\}$

# 如何分类？

- 每个特征 $x_i \in \{x_1, x_2, \dots, x_n\}$ 的权重值 $w_i \in \{w_1, w_2, \dots, w_n\}$ 说明 $x_i$ 的重要性
  - 考虑带偏置项 $b$
  - 将所有带权值的特征和偏移项相加

$$z = \left( \sum_{i=1}^n w_i x_i \right) + b$$

$$z = w \cdot x + b$$

- 如果 $z$ 值(总和值)高，则 $y = 1$ ；否则， $y = 0$

# 如何将 $z$ 值概率化？

$$z = w \cdot x + b$$

- $z$ 的取值范围：

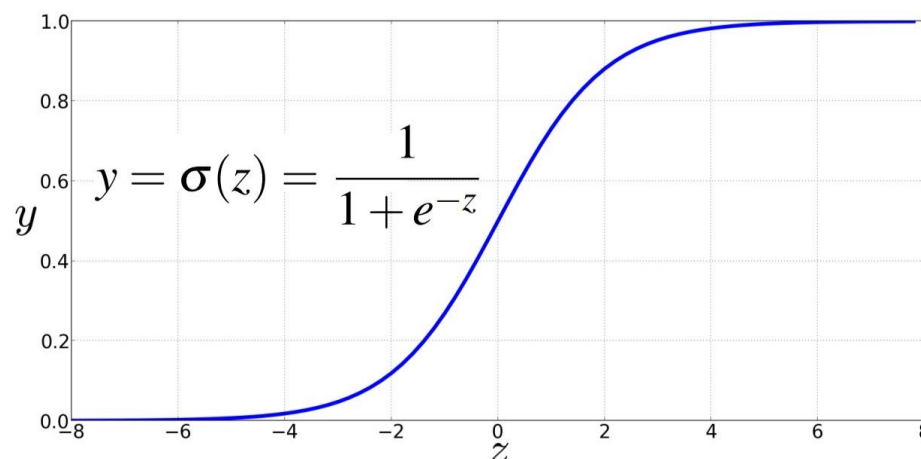
- $[-\infty, +\infty]$

- 如何概率化？

- 取值范围变成0到1

- sigmoid/logistic函数

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



## 通过sigmoid函数转换成概率值

$$\begin{aligned} \bullet P(y = 1) &= \sigma_1(w \cdot x + b) \\ &= \frac{1}{1 + \exp(-(w \cdot x + b))} \end{aligned}$$

$$\begin{aligned} \bullet P(y = 0) &= 1 - \sigma_1(w \cdot x + b) \\ &= 1 - \frac{1}{1 + \exp(-(w \cdot x + b))} \\ &= \frac{\exp(-(w \cdot x + b))}{1 + \exp(-(w \cdot x + b))} \\ &= \frac{1}{1 + \exp(w \cdot x + b)} \\ &= \sigma(-(w \cdot x + b)) \end{aligned}$$

# 概率值转化成分类器

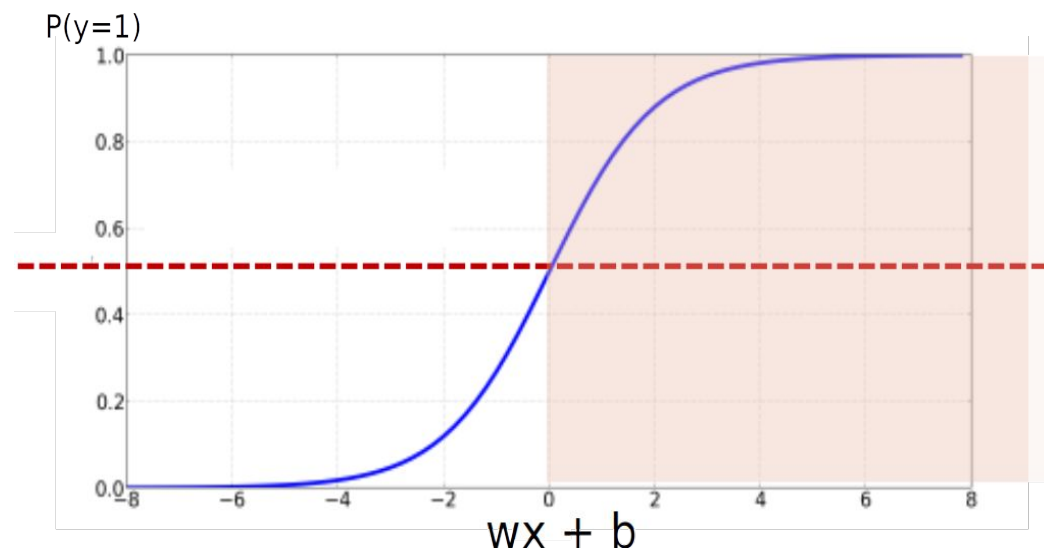
$$y = \begin{cases} 1 & \text{if } P(y = 1|x) > 0.5 \\ 0 & \text{其他} \end{cases}$$

- 0.5成为决策边界/decision boundary

if  $w \cdot x + b > 0$

if  $w \cdot x + b \leq 0$

$$\begin{aligned} \bullet P(y = 1) &= \sigma(w \cdot x + b) \\ &= \frac{1}{1 + \exp(-(w \cdot x + b))} \end{aligned}$$





# In-class Practice II

# 常用操作:特征值归一化

- z分数(z-score)标准化

- $\mu_i = \frac{1}{N} \sum_{j=1}^N x_i^{(j)}$ ,
- $\sigma_i = \sqrt{\frac{1}{N} \sum_{j=1}^N \left( x_i^{(j)} - \mu_i \right)^2}$
- $\tilde{x}_i = \frac{x_i - \mu_i}{\sigma_i}$

- 最小-最大归一(min-max normalization)

- $\tilde{x}_i = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}$

# 如何获得W?

- 监督学习
  - 训练阶段: 我们知道每个样本 $x$ 的正确标签 $y$  (0或1)
  - 推理阶段: 预测样本 $x$ 的标签 $\hat{y}$
- 目标: 求W和b使我们预测的 $\hat{y}^{(i)}$ 和真实的 $y^{(i)}$ 之间的距离最小
  - 需要距离估计器: 一个损失函数/loss function或成本函数/cost function
  - 需要优化算法更新w和b使其损失最小化

# 概率机器学习分类器的组成部分

- 给定 $N$ 个样本(输入/输出对):  $(x^{(i)}, y^{(i)})$ 
  1. 计算输入的特征表示: 对于每个输入观测值 $x^{(i)}$ ，获得其对应的 $n$ 维特征表示 $[x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}]$ ，通常其特征 $j$ 亦可记为 $F_j(x)$
  2. 通过分类函数计算其预测值 $\hat{y} = P(y|x)$ ，分类函数类似于sigmoid或softmax函数
  3. 学习的目标函数，比如交叉熵损失
  4. 优化目标函数的算法：如随机梯度下降

# 逻辑回归的学习组件

- 损失函数
  - 交叉熵损失/cross-entropy loss
- 优化算法
  - 随机梯度下降/stochastic gradient descent

# 损失函数：预测标签 $\hat{y}$ 和实际标签 $y$ 的距离

- 目标：我们想知道分类器输出的预测标签 $\hat{y}$ 与实际标签 $y$ 的距离是多少
  - 预测标签： $\hat{y} = \sigma(w \cdot x + b)$
  - 真实标签： $y$  (0 或 1)
- 此距离亦记作： $L(\hat{y}, y)$ ，用于评估 $\hat{y}$ 和 $y$ 的距离

负对数似然损失=交叉熵损失

Negative Log Likelihood Loss (NLL)=Cross-entropy Loss

- 一种条件极大似然估计的情形
- 选择参数 $w$   $b$ ，使其最大化
  - 对数条件概率
    - 给定观测值 $x$ ，训练样本中真实标签 $y$ 出现的概率

# 交叉熵损失推导

- 目标：使正确标签 $P(y|x)$ 的概率最大化
- 因为只有两个离散值(0或1)，我们可以将概率 $P(y|x)$ 表示为
$$P(y|x) = \hat{y}^y(1 - \hat{y})^{1-y}$$

- 检验

- $y = 1$ :  $P(y|x) = \hat{y}$
- $y = 0$ :  $P(y|x) = 1 - \hat{y}$



# 交叉熵损失推导

- 目标：使正确标签 $P(y|x)$ 的概率最大化
  - 最大化： $P(y|x) = \hat{y}^y(1 - \hat{y})^{1-y}$
- 两边取对数(log):
  - 最大化： $\log P(y|x) = \log[\hat{y}^y(1 - \hat{y})^{1-y}]$   
 $= y \log \hat{y} + (1 - y) \log (1 - \hat{y})$
- 两边乘以负号，得到交叉熵损失函数
  - 最小化： $L_{\text{CE}}(\hat{y}, y) = -\log P(y|x) = -[y \log \hat{y} + (1 - y) \log (1 - \hat{y})]$
- 代入定义 $\hat{y} = \sigma(w \cdot x + b)$

$$L_{\text{CE}}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))]$$

## In-class Practice III: 检验损失函数有效性

- 我们期望损失函数
  - 值较小(当模型预测值 $\hat{y}$ 接近其正确值)
  - 值较大(当模型困扰时)

# 逻辑回归的学习组件

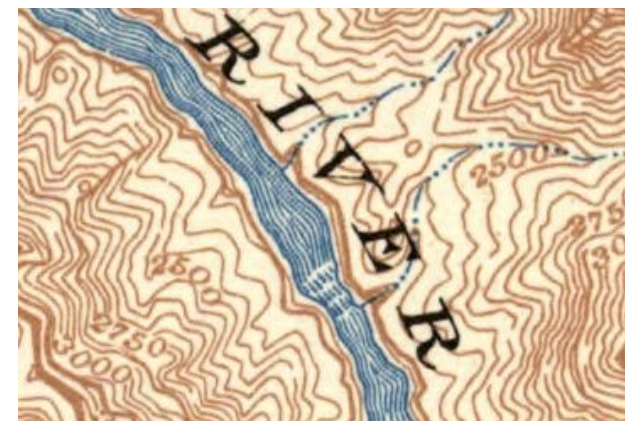
- 损失函数
  - 交叉熵损失/cross-entropy loss
- 优化算法
  - 随机梯度下降/stochastic gradient descent

# 随机梯度下降

## Stochastic Gradient Descent

- 随机梯度下降算法用于优化模型的权重
  - 模型: 逻辑回归、神经网络
- 设置
  - 模型参数:  $\theta = (w, b)$
  - 模型预测值:  $\hat{y} = f(x; \theta)$
- 目标: 找到权重, 使得总体平均损失最小

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^N L_{CE}(f(x^{(i)}; \theta), y^{(i)})$$



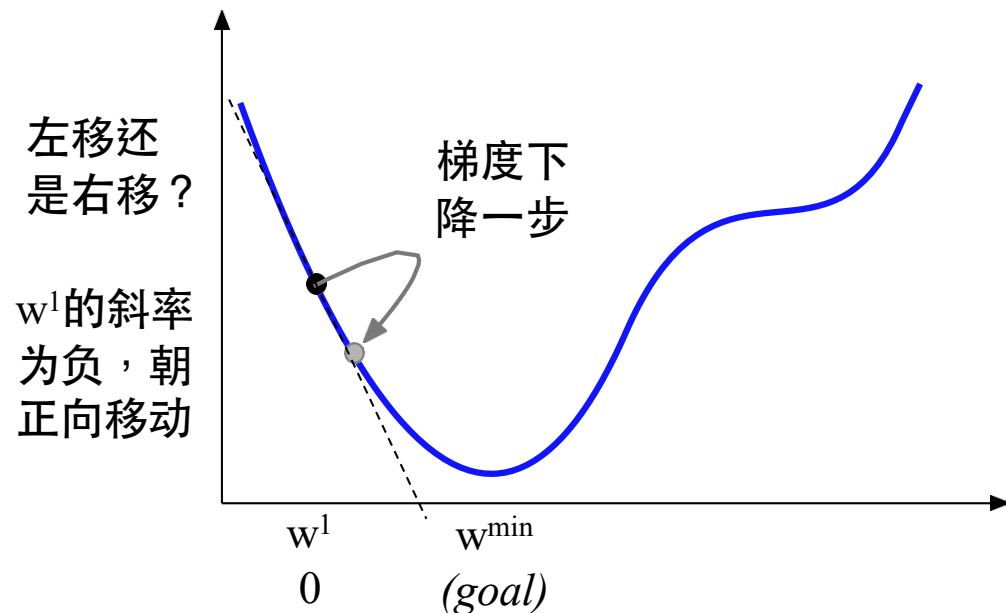
如何最快走到河谷?

# 目标：最小化损失函数

- 逻辑回归的损失函数是凸函数
  - 凸函数只有一个最小值
  - 从任意点开始的梯度下降保证能找到最小值
    - （神经网络的损失函数是非凸的）

# 标量示例

- 问题: 给定当前 $w$ ，我们该增大还是减小其目标值？如何做？
- 答案: 将 $w$ 沿损失函数的反方向移动



多变量函数的**梯度**  
/gradient是指向函数  
最大**增量**方向的矢量

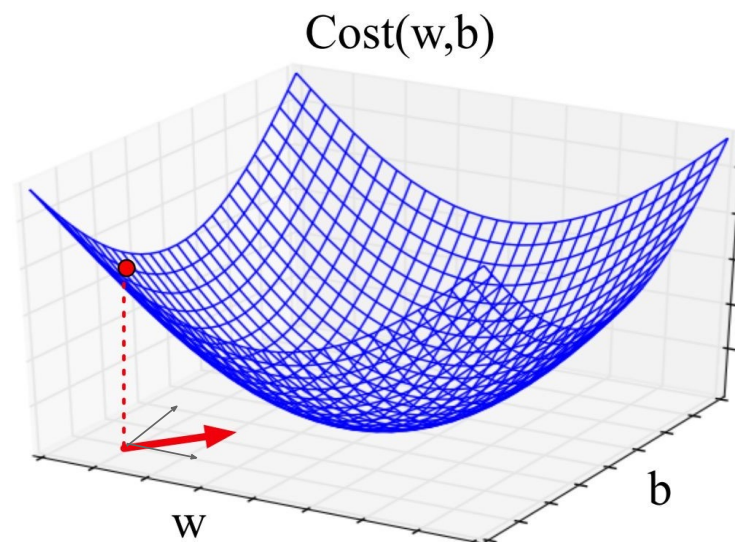
**梯度下降/Gradient descent**: 找到损失函数在当前点处的梯度，朝梯度反方向移动

# 梯度下降

- 新权值 $w^{t+1}$ 是旧权值 $w^t$ 减去由学习率 $\eta$ 加权的梯度值

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x; w), y)$$

- $\eta$ : 学习率
  - 大的学习率，移动更远



# 梯度下降：n-维空间

- 设  $\hat{y} = f(x; \theta)$ , 参数  $\theta$  在 n-维空间

$$\nabla_{\theta} L(f(x; \theta), y) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x; \theta), y) \\ \frac{\partial}{\partial w_2} L(f(x; \theta), y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x; \theta), y) \end{bmatrix}$$

- 基于梯度更新的  $\theta$  的最终方程为：

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} L(f(x; \theta), y)$$



# 逻辑回归：梯度

- 损失函数

$$L_{\text{CE}}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x + b))]$$

- 交叉熵损失函数的梯度 ([SLP3] Sec.5.10)

$$\begin{aligned} \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial w_j} &= [\sigma(w \cdot x + b) - y]x_j && \text{最小二乘法} \\ &= (\hat{y} - y)x_j \end{aligned}$$

# 随机梯度下降

**function** STOCHASTIC GRADIENT DESCENT( $L()$ ,  $f()$ ,  $x$ ,  $y$ ) **returns**  $\theta$

# where:  $L$  is the loss function

#  $f$  is a function parameterized by  $\theta$

#  $x$  is the set of training inputs  $x^{(1)}, x^{(2)}, \dots, x^{(m)}$

#  $y$  is the set of training outputs (labels)  $y^{(1)}, y^{(2)}, \dots, y^{(m)}$

$\theta \leftarrow 0$

**repeat** til done

For each training tuple  $(x^{(i)}, y^{(i)})$  (in random order)

1. Optional (for reporting): # How are we doing on this tuple?

    Compute  $\hat{y}^{(i)} = f(x^{(i)}; \theta)$  # What is our estimated output  $\hat{y}$ ?

    Compute the loss  $L(\hat{y}^{(i)}, y^{(i)})$  # How far off is  $\hat{y}^{(i)}$  from the true output  $y^{(i)}$ ?

2.  $g \leftarrow \nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)})$  # How should we move  $\theta$  to maximize loss?

3.  $\theta \leftarrow \theta - \eta g$  # Go the other way instead

**return**  $\theta$

# 对比结构感知器/Structure Perceptron

```
feature_weights = {}  
for x, y in data:  
    # Make a prediction  
    features = extract_features(x)  
    predicted_y = run_classifier(features)  
    # Update the weights if the prediction is wrong  
    if predicted_y != y:  
        for feature in features:  
            feature_weights[feature] = (  
                feature_weights.get(feature, 0) +  
                y * features[feature]  
            )
```

[https://github.com/hqyang/nlp-codes/blob/main/01-simpleclassifier/trained\\_bow\\_classifier.ipynb](https://github.com/hqyang/nlp-codes/blob/main/01-simpleclassifier/trained_bow_classifier.ipynb)

# 超参/Hyperparameters

- 学习率 $\eta$ 是一个超参数
  - 太大：学习器走一大步，可能超出范围(overshoot)
  - 太小：学习器花很长时间找到最优值
- 超参
  - 简而言之，是机器学习模型的一种特殊参数
  - 不是由算法用监督的方式学习（如正则参数），而是由算法设计者选择

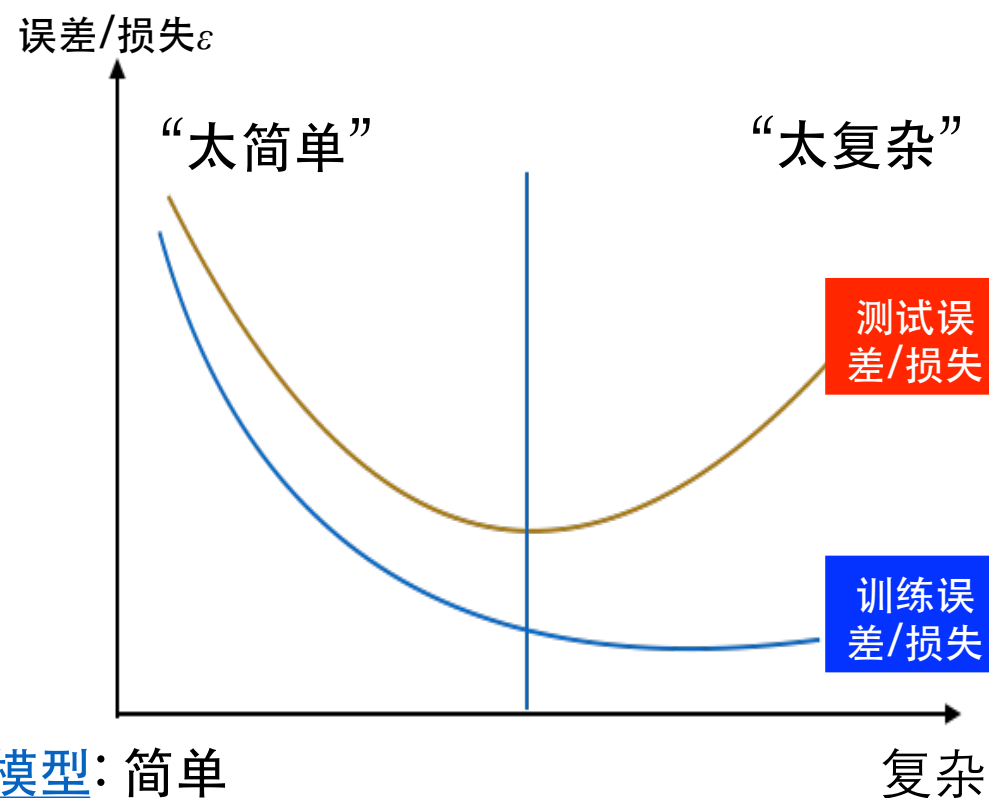
# 小批量训练/Mini-batch Training

- 随机梯度下降训练可以有三种不同的训练数据选择方式
  1. 随机选择一个样本
    - 这可能导致权值起伏不定 (choppy movements)
  2. 批处理训练/Batch Training:
    - 整个数据集
  3. 小批量训练：
    - $m$ 个样本：如512或1,024

# 欠拟合/Underfitting vs. 过拟合/Overfitting

类型	现象	原因	直观解释
欠拟合	训练误差高 测试误差高	模型太简单	学不会
过拟合	训练误差低 测试误差高	模型太复杂	学太死

- 学习目标: 泛化能力/Generalization



# 正则化/Regularization

- 为什么需要正则化？
  - 解决过拟合的问题

- 如何实现？

$$\hat{\theta} = \arg \max_{\theta} \sum_{i=1}^N \ln P(y^{(i)} | x^{(i)}; \theta) - \alpha R(\theta)$$

- 思路：选择 $R(\theta)$ 惩罚大权重
  - 惩罚大权重，让模型参数值尽可能小来拟合数据，效果更好

# L2正则 (Ridge Regression/岭回归) vs. L1正则 (Lasso Regression/套索回归)

- L2正则

- 定义:  $R(\theta) = \|\theta\|^2 = \sum_{i=1}^n \theta_i^2$

- 目标函数

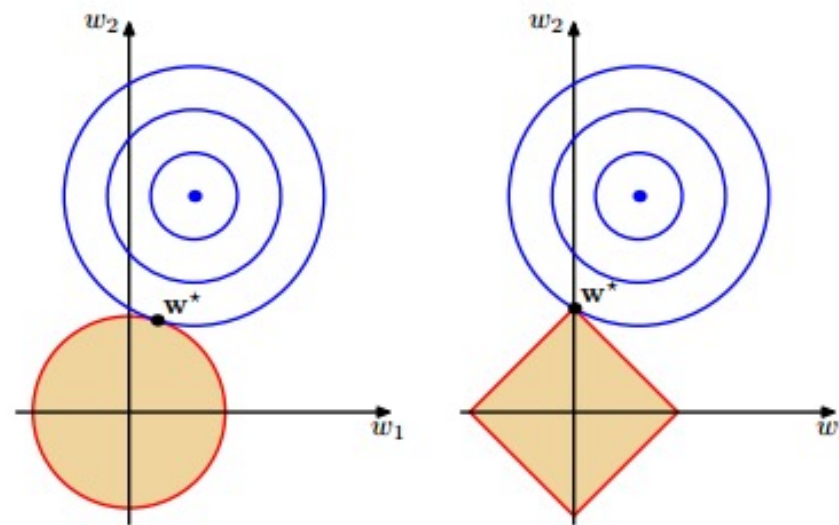
$$\hat{\theta} = \arg \max_{\theta} \sum_{i=1}^N \ln P(y^{(i)} | x^{(i)}; \theta) - \alpha \sum_{j=1}^n \theta_j^2$$

- L1正则

- 定义:  $R(\theta) = |\theta| = \sum_{i=1}^n |\theta_i|$

- 目标函数

$$\hat{\theta} = \arg \max_{\theta} \sum_{i=1}^N \ln P(y^{(i)} | x^{(i)}; \theta) - \alpha \sum_{j=1}^n |\theta_j|$$





# 多项式逻辑回归

## Multinomial Logistic Regression

- 通常我们需要分更多的类别
  - 情感识别: 正向/负向/中性的
  - 词类: 名词、动词、形容词、副词、介词等
  - 文本分类: 新闻、体育、政治等
- 如果类别 $>2$ , 使用多项式逻辑回归/Multinomial Logistic Regression
  - = Softmax regression/归一化指数函数回归
  - = Multinomial logit 多项式逻辑
  - = (曾用名: 最大熵模型/Maximum entropy modeling, MaxEnt)
- 逻辑回归仅表示二分类

# 多项式逻辑回归

## Multinomial Logistic Regression

- 所有事件的概率总和必须为1
  - $P(\text{正向}|d) + P(\text{负向}|d) + P(\text{中性}|d) = 1$
- 需要一个S型曲线的泛化: softmax
  - 给定 $K$ 维向量 $\mathbf{z} = [z_1, z_2, \dots, z_K]$ ,  $z_i$ 为任意值
  - 输出一个概率分布
    - 每个值在 $[0,1]$ 范围内
    - 所有的值加起来等于1

# 柔性最大值函数Softmax: 泛化的Sigmoid函数

- 将任意值的 $K$ 维向量 $\mathbf{z} = [z_1, z_2, \dots, z_K]$ 转换成概率值

$$\text{softmax}(z_j) = \frac{\exp(z_j)}{\sum_{i=1}^K z_i}, 1 \leq j \leq K$$

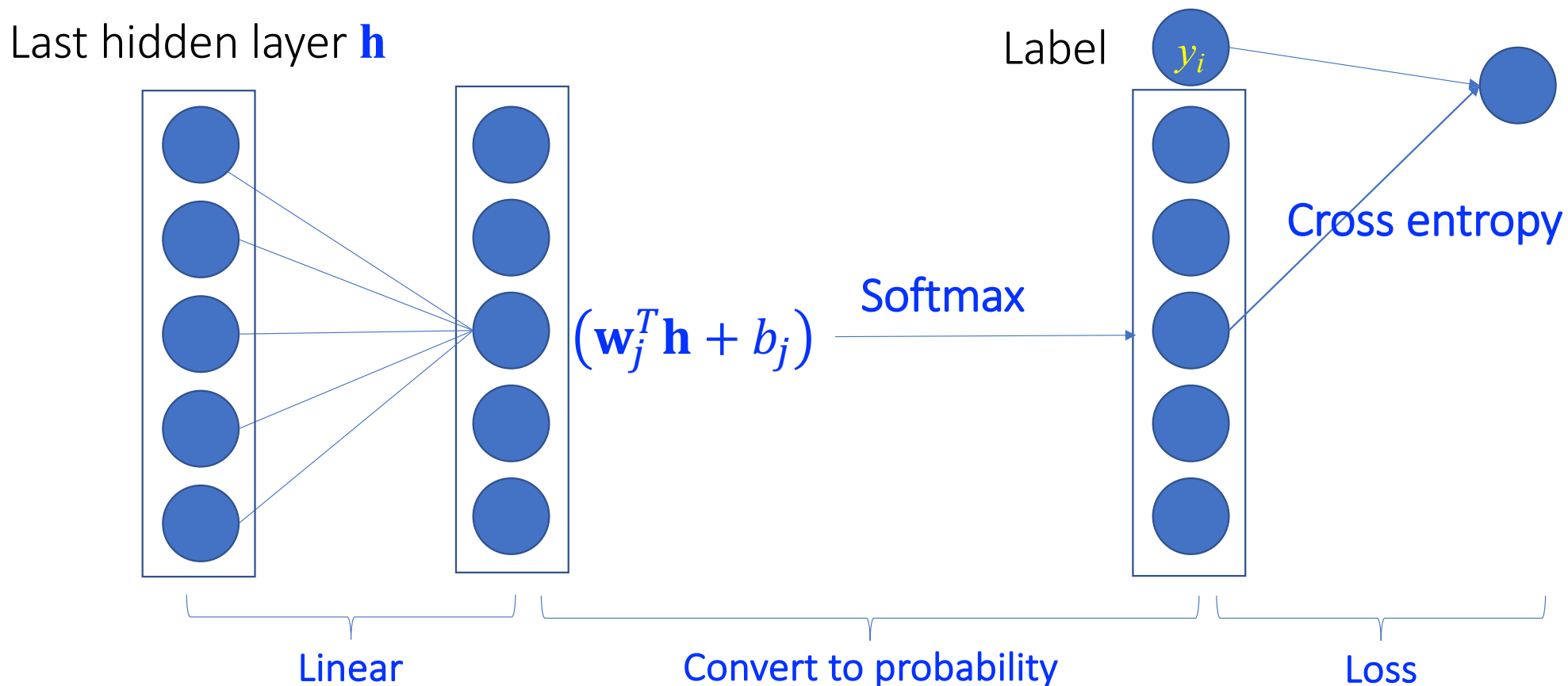
- 分母 $\sum_{i=1}^K z_i$ 将所有值转成概率值

$$\text{softmax}(\mathbf{z}) = \left[ \frac{\exp(z_1)}{\sum_{i=1}^K z_i}, \frac{\exp(z_2)}{\sum_{i=1}^K z_i}, \dots, \frac{\exp(z_K)}{\sum_{i=1}^K z_i} \right]$$

- 例子

- $\mathbf{z} = [0.01, 5, 0.5, -0.5, -5]$
- $\text{softmax}(\mathbf{z}) = [0.006659, 0.978429, 0.010869, 0.003997, 0.000044]$

# 多项式逻辑回归：小结



# 概率机器学习分类器的组成部分

- 给定 $N$ 个样本(输入/输出对):  $(x^{(i)}, y^{(i)})$ 
  1. 计算输入的特征表示: 对于每个输入观测值 $x^{(i)}$ ，获得其对应的 $n$ 维特征表示 $[x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}]$ ，通常其特征 $j$ 亦可记为 $F_j(x)$
  2. 通过分类函数计算其预测值 $\hat{y} = P(y|x)$ ，分类函数类似于sigmoid或softmax函数
  3. 学习的目标函数，比如交叉熵损失
  4. 优化目标函数的算法：如随机梯度下降

# 多项式逻辑回归实现

- 规则

Train accuracy: 0.4345739700374532  
Dev/test accuracy: 0.4214350590372389

- 词袋特征和结构感知器

Train accuracy: 0.7332631086142322  
Dev/test accuracy: 0.5676657584014533

- 朴素贝叶斯

Train accuracy: 0.858497191011236  
Dev/test accuracy: 0.631244323342416

- 逻辑回归

Train accuracy: 0.521184456928839  
Dev/test accuracy: 0.5277020890099909

- 代码:

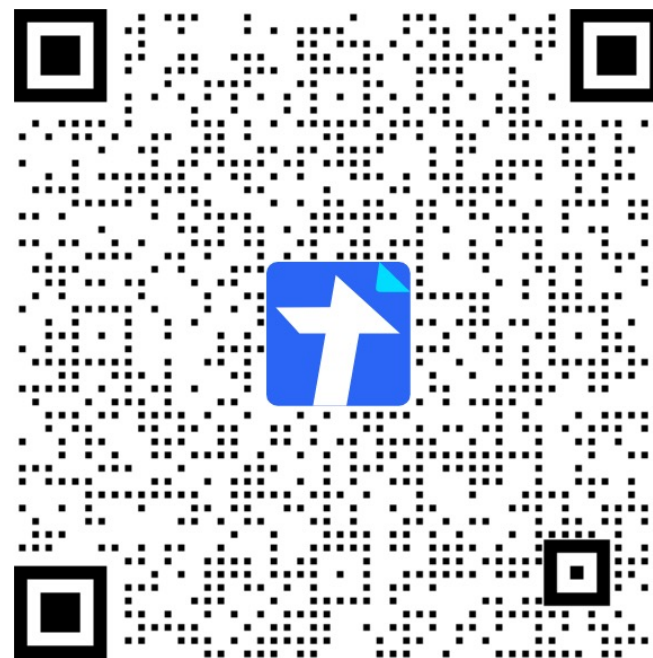
- <https://github.com/hqyang/nlp-codes/blob/main/01-simpleclassifier>

- 如何实现？

- 如何优化？

# 一句话总结

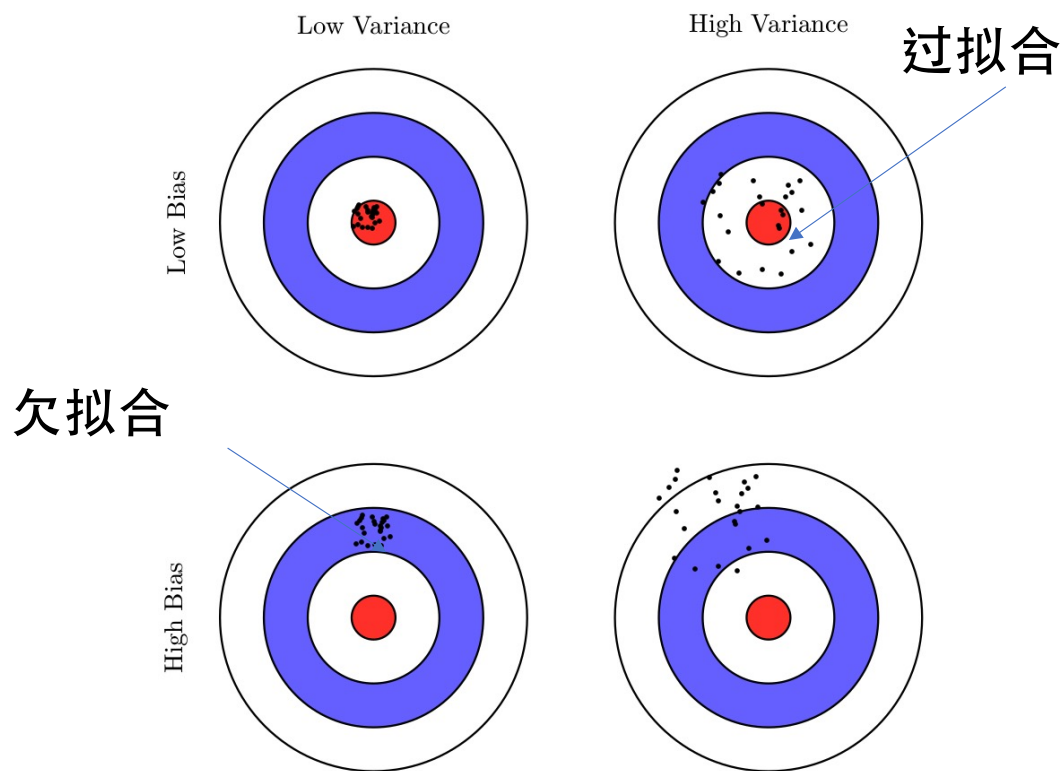
- 课外阅读
  - [SLP3]Ch.4-5 , Ch. 7



# Appendix



# 偏见-方差平衡 Bias-Variance Tradeoff



- **偏差Bias**：这类模型不能很好地拟合数据

- 修正：一个**更具表现力/more expressive**的模型类

- **方差Variance**：这类模型可以拟合数据，但因为模型太复杂以致拟合不好

- 修正：选择**表现力较差/less expressive**的模型类

$$E_{D, \epsilon} \{(f - \hat{y})^2\} = E\{[f - h]^2 + E[h - \hat{y}]^2\} = \text{Bias}^2 + \text{Variance}$$

- $f$ : 真实函数;  $\hat{y}$ : 预测函数;  $h$ : 预测的期望

# In-class Practice I

- 右边是一个二分类器的混淆矩阵，计算其对应的分类指标
  - 准确率
  - 错误率
  - 精确率、召回率、F1值

类别 \ 预测	正向	负向	总数
正向	7,800	200	8,000
负向	300	1,700	2,000
总数	8,100	1,900	10,000

## In-class Practice II

- 某情感分析任务中，用逻辑回归判断一句话是“正面评价”（标签 1）还是“负面评价”（标签 0），已训练好的模型参数如下：
  - 特征 1：关键词“好”的出现次数（记为 $x_1$ ），权重 $w_1 = 2$
  - 特征 2：关键词“差”的出现次数（记为 $x_2$ ），权重 $w_2 = -3$
  - 偏置项： $b = -1$
- 判断下面句子的情感标签(正面/负面): 假设有分词
  1. 这部电影很好，剧情不差
  2. 这家餐厅菜品差，服务也不好

## In-class Practice III

- 已知下面句子的情感标签判定如下，计算其对应不同真实值的交叉熵损失值，对比其不同

$x$	预测概率值
这部电影很好，剧情不差	$P(+1 x)=0.731$