回顾：降维

1.什么是降维？为什么要降维？
2.主成分分析（PCA）的目标函数/优化目标是什么？具体步骤？
3.降维是必须的吗？你觉得什么时候需要降维？

第四章 分类

第四章 分类

4.1 模型评估和性能度量
4.2 k最近邻分类
4.3 决策树
4.4 贝叶斯分类
4.5 组合分类
4.6 案例：信用违约预测

分类问题

分类：把未标签样本分配到预先定义好的类别； 未标签样本又叫新样本或测试样本。

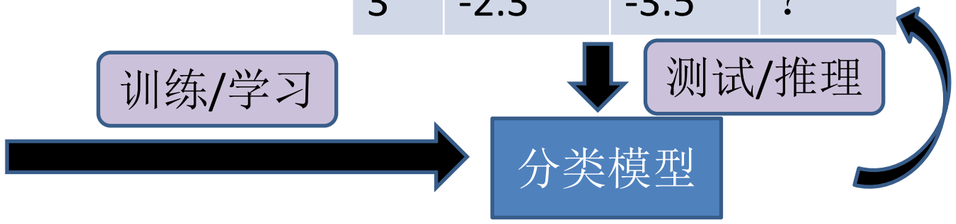
模型训练：基于标签样本（训练样本）建立用于预测新样本类别的模型的过程。

训练集：带标签的数据集

id	属性1	属性2	标签
1	2.3	4.5	1
2	1.2	5.6	1
3	2.3	4.5	1
4	-1.2	-3.4	2
5	-2.3	-5.6	2
6	-4.5	-2.3	3
7	-3.4	-4.2	3

测试集：需要预测标签的数据集

id	属性1	属性2	标签
1	3.4	2.5	?
2	1.2	4.5	?
3	-2.3	-3.5	?



模型是什么？它可以是一系列规则、一个决策树、一个带参数的线性/非线性函数。

主要分类算法：决策树、贝叶斯分类、 k最近邻、 SVM, 逻辑回归、神经网络,…

分类问题

训练集：带标签的数据集

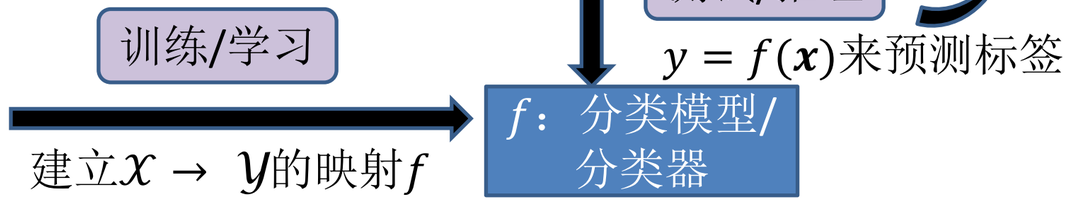
id	属性1	属性2	标签
1	2.3	4.5	1
2	1.2	5.6	1
3	2.3	4.5	1
4	-1.2	-3.4	2
5	-2.3	-5.6	2
6	-4.5	-2.3	3
7	-3.4	-4.2	3

(x_1, y_1)

$x \in \mathcal{X}$　　$y \in \mathcal{Y}$

测试集：需要预测标签的数据集

id	属性1	属性2	标签
1	3.4	2.5	?
2	1.2	4.5	?
3	-2.3	-3.5	?



$$\chi_{train} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

- 二分类 vs 多分类

  若$|\mathcal{Y}| = 2$，如$\mathcal{Y} = \{-1, 1\}$ 或者 $\{1, 0\}$，则为二分类问题。若$|\mathcal{Y}| > 2$，则为多分类。

- 监督学习

  训练集中样本的类标签是一种最常用的监督信息。监督信息的数量和质量对模型学习起到关键作用。

第四章 分类

4.1 模型评估和性能度量

4.2 k最近邻分类

4.3 决策树

4.4 贝叶斯分类

4.5 组合分类

4.6 案例：信用违约预测

什么样的模型是好的？

用于模型选择（不同模型性能对比）、参数设置（同一个模型不同超参数设置）。

分两个方面：

怎么评估一个模型的好坏？--模型评估方法

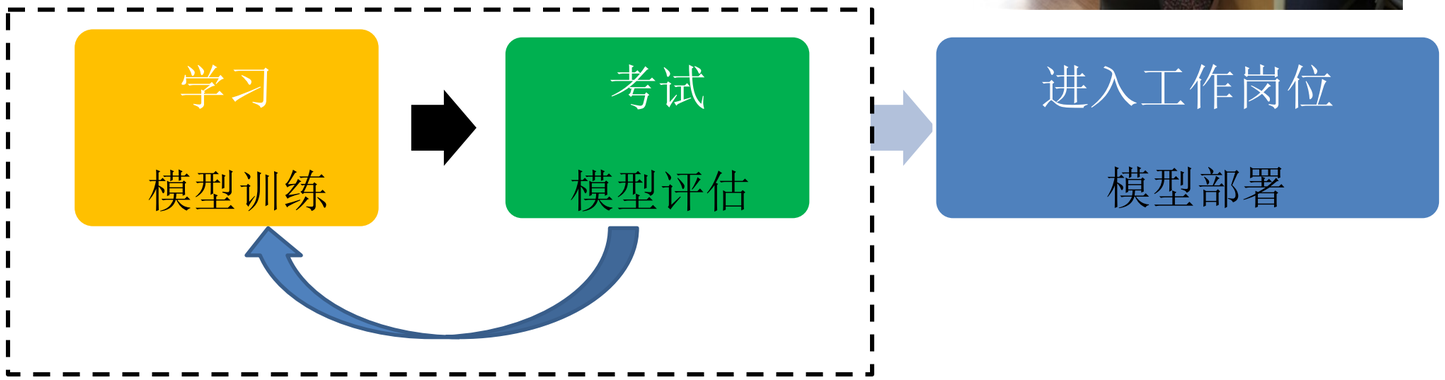
怎么衡量有多好？--性能度量

学习质量评估的重要性

- 如何评价一个学生的学习效果？
- 考试考得好一定工作能力强吗？

人才培养

合理设置考核方法直接关系教学效果



问题一：模型评估

分类任务的目标：训练好的模型对未知样本的分类尽可能准确 –泛化能力

泛化误差：在“未来”样本上的误差 ➡ 理想的评估方法 （工作能力）

然而，在训练模型的时候并不知道未知样本，不能直接评估泛化误差

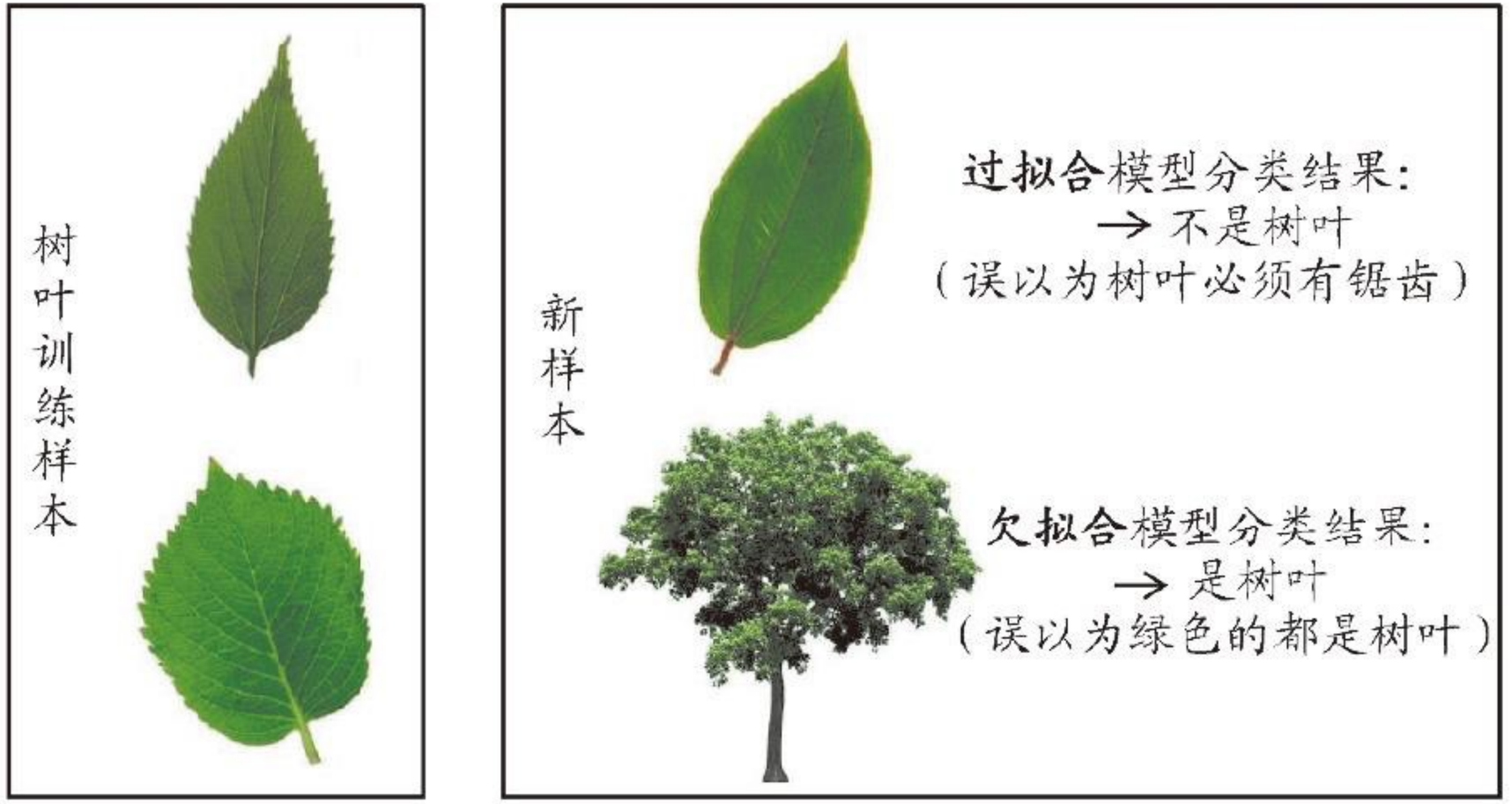
要找一个近似的方法。。。

训练误差：在训练集上的误差，亦称“经验误差”合适吗？

- 泛化误差越小越好
- 经验误差是否越小越好？

NO! 因为会出现“过拟合”(overfitting)

过拟合 (overfitting) vs. 欠拟合 (underfitting)



过拟合：关注太多细节，要求太苛刻—太敏感。

欠拟合：对重要信息描述不到位—太迟钝。

泛化误差的近似：测试误差

老师给小明做了10道线性代数的题目，他想知道小明是否掌握了相关知识点？
做法1：老师遮住刚才的10道题目的答案，让小明再回答一遍，他都答对了！训练误差为0。

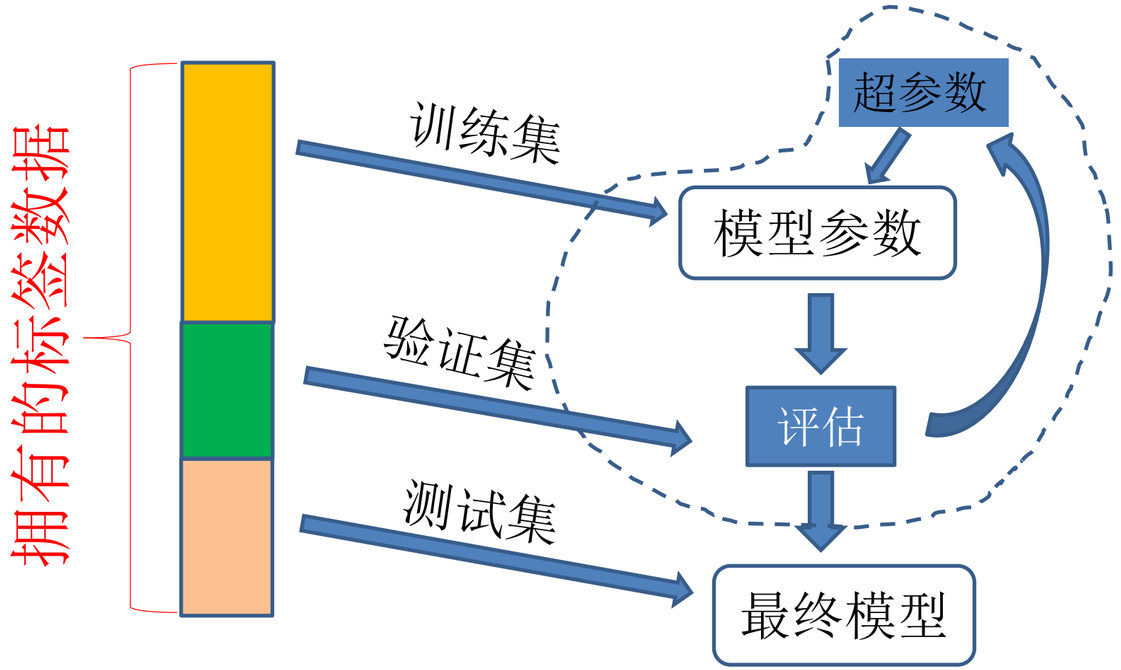
小明真的掌握这些知识点了吗？
其实，小明的记忆力超强，把这10题的答案都记住了。

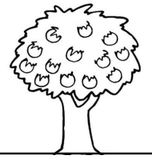
做法2：拿小明没有做过的题目考他。

如何近似地评估模型的泛化能力？

把带标签数据分成两部分，一部分用于训练，一部分用于测试。由于测试数据在训练过程中不出现，因此测试样本类似“未知样本”，测试误差近似泛化误差。

“调参”与最终模型





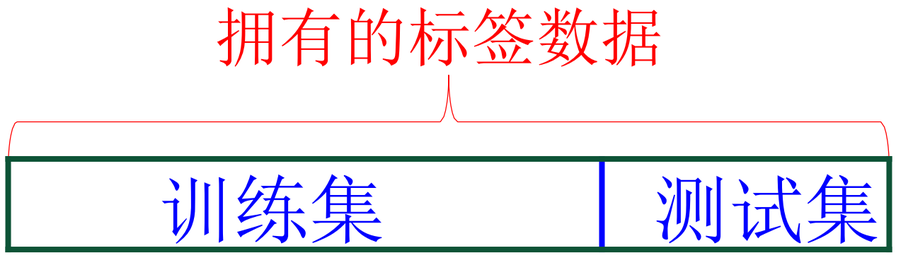
模型训练，即学习模型参数之前，除了给定训练集，一般还需要人工设定算法特有的参数，称“超参数”。

超参数怎么选择呢？

区别：训练集 vs. 验证集 (validation set) vs. 测试集

算法参数选定后，要用“训练集+验证集”重新训练最终模型

训练集(training set)和测试集(test set)的划分



我们需要考虑的问题：

- 为了减少过拟合风险，测试集应该与训练集“互斥”
- 训练集和测试集都尽可能大—矛盾

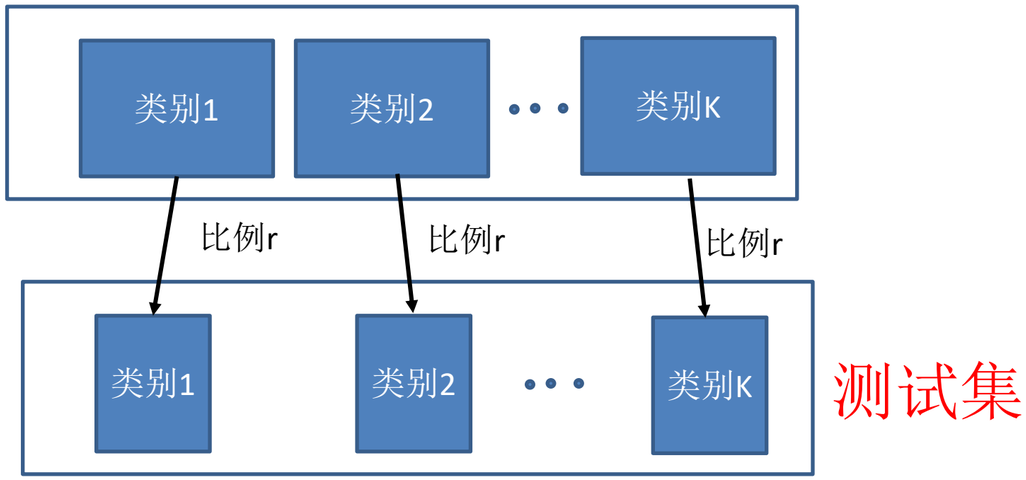
常见方法：

- 留出法 (hold-out)
- 交叉验证法 (cross validation)
- 自助法 (bootstrap)

思考：
训练集太小有什么问题？
测试集太小有什么问题？

留出法（holdout）

拥有的数据集

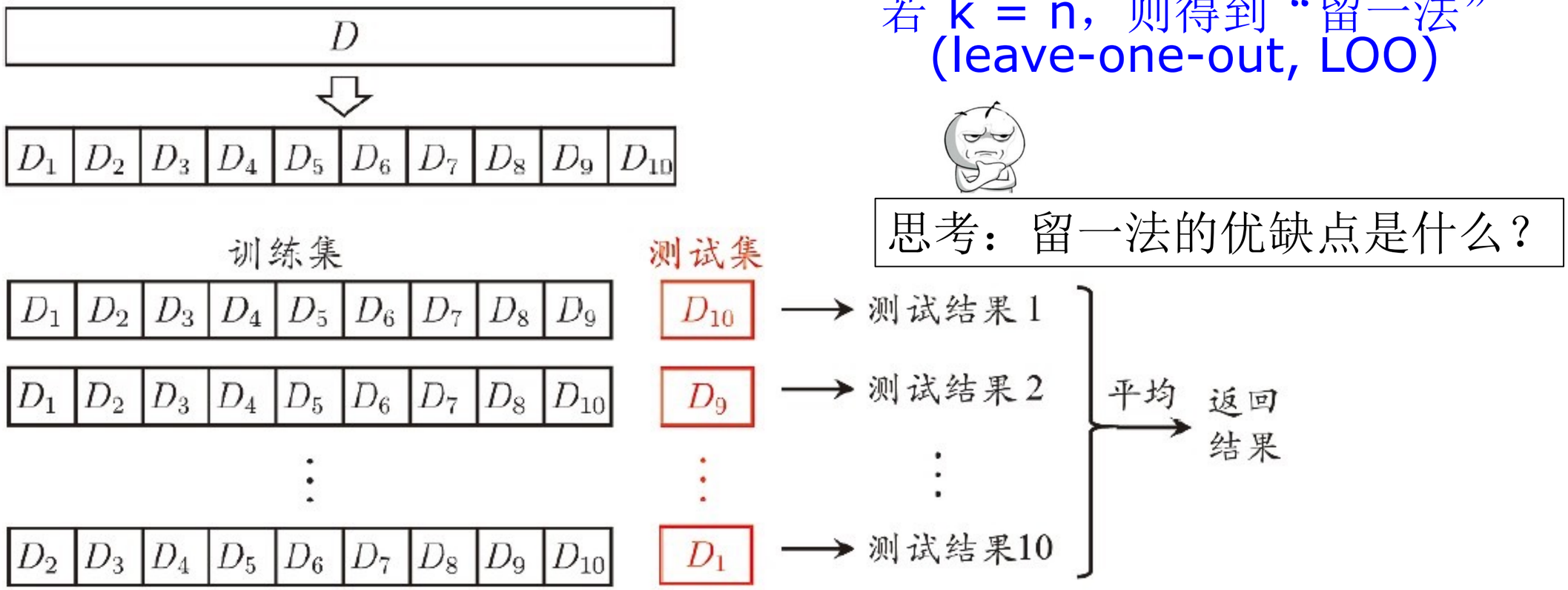


测试集

注意：

- 保持数据分布一致性 （例如：分层采样，即从每个类里面随机抽样同比例的样本)
- 多次重复划分 (例如：100次随机划分)
- 测试集不能太大、不能太小，为什么？一般在 1/5~1/3

k-折交叉验证法(k-fold cross validation)



若 k = n，则得到“留一法”
(leave-one-out, LOO)

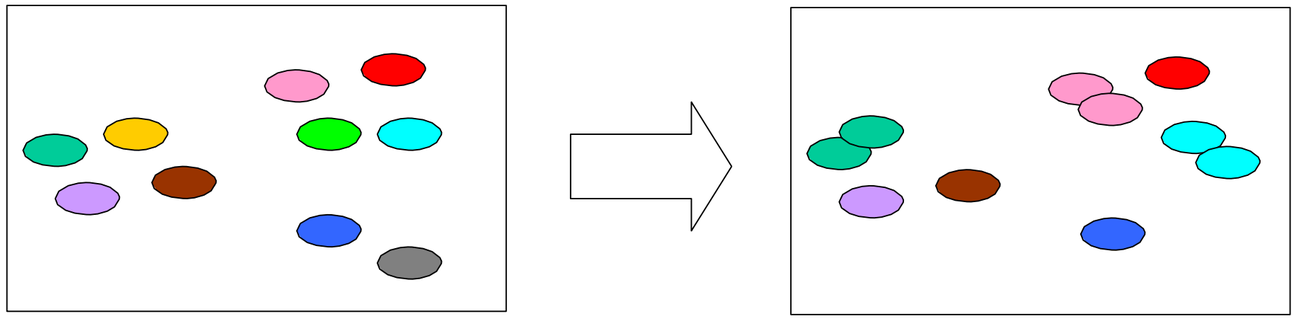
思考：留一法的优缺点是什么？

好处：在保证训练集大小的同时，所有训练样本都能用于模型测试，减小测试集随机性导致的误差；
缺点：需要进行K次模型训练。

自助法（Bootstrapping）

基于“自助采样”：从n个样本中每次随机抽一个，然后放回，重复n次。

亦称“有放回采样”、“可重复采样”



$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n}\to\frac{1}{e}\approx 0.368$$

“包外估计”(out-of-bag estimation)

优点：训练集与原样本集同规模
缺点：数据分布有改变

问题二：性能度量

性能度量是衡量模型泛化能力的评价标准

什么样的模型是“好”的，不仅取决于算法和数据， 还取决于任务需求

- 分类任务用错误率（error rate）或准确率(accuracy)

$$\mathrm{E}(f,D)=\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(f(\boldsymbol{x}_i)\neq y_i)$$

$$\begin{aligned}\mathrm{acc}(f,D)&=\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(f(\boldsymbol{x}_i)=y_i)\\&=1-\mathrm{E}(f,D)\end{aligned}$$

- 回归任务常用均方误差：

$$\mathrm{E}(f,D)=\frac{1}{n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i)-y_i)^2$$

注意：有些文献中把accuracy翻译成精度

查准率 vs. 查全率

混淆矩阵（二分类）

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

例：已知一个包含10个样本的测试集，前4个为正例，后6个为反例。现有一个算法预测出来的结果是：1 0 1 0 1 0 0 0 1 0，其中1代表正例，0代表反例。 计算查准率和查全率。

- 查准率(precision)： $P = \frac{TP}{TP + FP}$
- 查全率/ 召回率 （recall）： $R = \frac{TP}{TP + FN}$

F1度量：查准率P和查全率R的调和平均

$$F1 = \frac{2 \times P \times R}{P + R} = \frac{2 \times TP}{\text{样例总数} + TP - TN}$$

- 哪些应用更看中查准率？ 哪些应用更看中查全率？

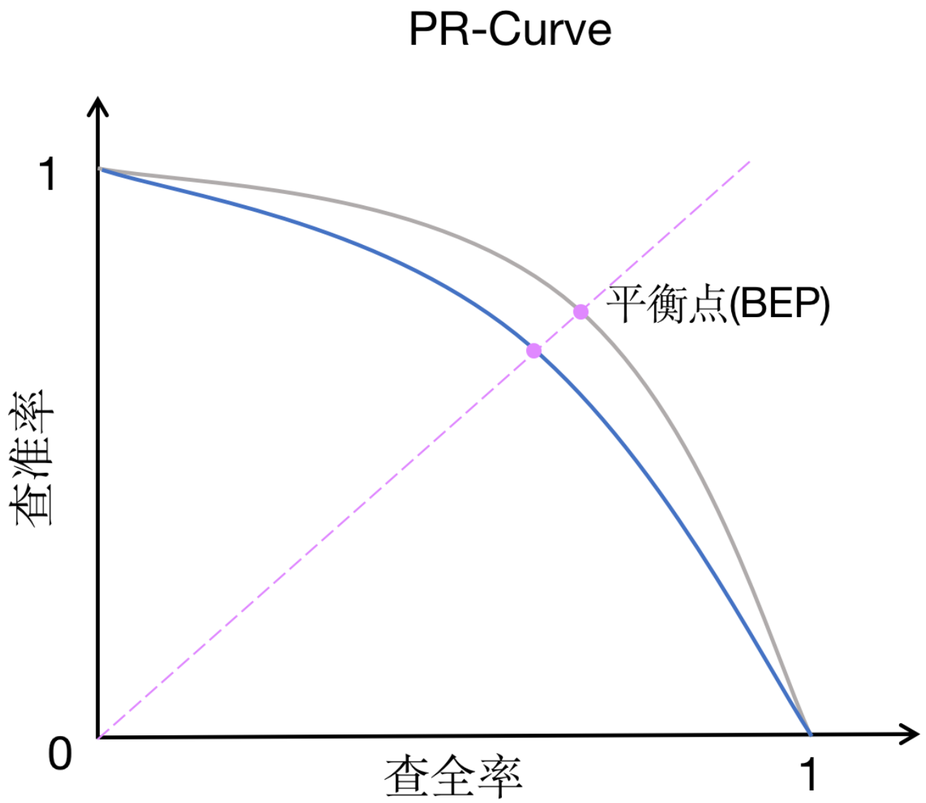
若对查准率/查全率有不同偏好：

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

$\beta > 1$ 时查全率有更大影响；$\beta < 1$ 时查准率有更大影响

PR曲线：查准率P和查全率R

分类器返回的不是直接对应类别的离散值，而是分布概率



宏xx vs. 微xx

若能得到多个混淆矩阵：例如多分类的两两混淆矩阵

宏(macro-)查准率、查全率、F1

$$\text{macro-}P = \frac{1}{n}\sum_{i=1}^{n} P_i \ ,$$

$$\text{macro-}R = \frac{1}{n}\sum_{i=1}^{n} R_i \ ,$$

$$\text{macro-}F1 = \frac{2 \times \text{macro-}P \times \text{macro-}R}{\text{macro-}P + \text{macro-}R} \ .$$

微(micro-)查准率、查全率、F1

$$\text{micro-}P = \frac{\overline{TP}}{\overline{TP} + \overline{FP}} \ ,$$

$$\text{micro-}R = \frac{\overline{TP}}{\overline{TP} + \overline{FN}} \ ,$$

$$\text{micro-}F1 = \frac{2 \times \text{micro-}P \times \text{micro-}R}{\text{micro-}P + \text{micro-}R} \ .$$

非均等代价

犯不同的错误往往会造成不同的损失

此时需考虑“非均等代价”(unequal cost)

真实类别	预测类别	
	第 0 类	第 1 类
第 0 类	0	$cost_{01}$
第 1 类	$cost_{10}$	0

- 代价敏感(cost-sensitive)错误率：

$$\mathrm{E}(f, D, cost) = \frac{1}{n}\left(\sum_{\boldsymbol{x}_i \in D^0} \mathbb{I}(f(x_i) \neq y_i) \times cost_{01} + \sum_{\boldsymbol{x}_i \in D^1} \mathbb{I}(f(x_i) \neq y_i) \times cost_{10}\right)$$

偏差-方差分解 (bias-variance decomposition)

对回归任务，泛化误差可通过“偏差-方差分解”拆解为：

$$E(f;D) = bias^2(\boldsymbol{x}) + var(\boldsymbol{x}) + \varepsilon^2$$

期望输出与真实输出的差别

$$bias^2(\boldsymbol{x}) = \left(\bar{f}(\boldsymbol{x}) - y\right)^2$$

同样大小的训练集的变动，所导致的性能变化

$$var(\boldsymbol{x}) = \mathbb{E}_D\left[\left(f(\boldsymbol{x};D) - \bar{f}(\boldsymbol{x})\right)^2\right]$$

训练样本的标记与真实标记有区别

$$\varepsilon^2 = \mathbb{E}_D\left[(y_D - y)^2\right]$$

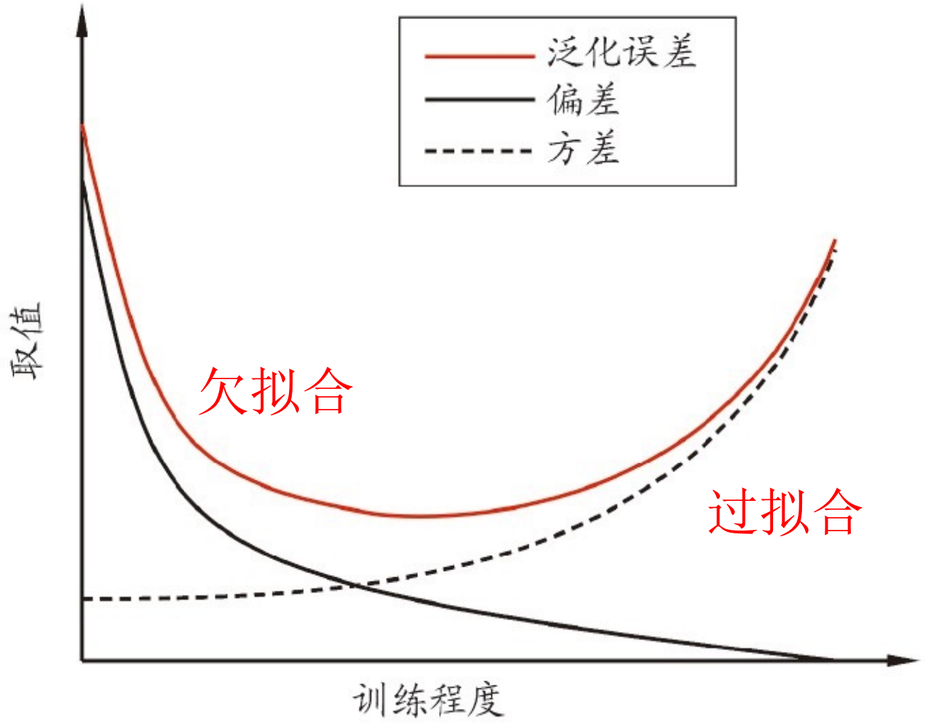
表达了当前任务上任何学习算法所能达到的期望泛化误差下界

泛化性能是由学习算法的能力、数据的充分性以及学习任务本身的难度共同决定

偏差-方差窘境 (bias-variance dillema)

一般而言，偏差与方差存在冲突：

- 训练不足时，学习器拟合能力不强，偏差主导
- 随着训练程度加深，学习器拟合能力逐渐增强，方差逐渐主导
- 训练充足后，学习器的拟合能力很强，方差主导



怎么判断过拟合，如何解决？

- 判断过拟合：训练误差小，测试误差却很大，即方差大
- 原因：相对于目标数据集，模型过于复杂，一个复杂的模型需要更多的样本来支持。

- 可以从以下几个方面寻求解决方法：

进行特征删选，去掉不重要的特征，或用PCA等算法降维
增加正则项权重
增加训练数据集大小

第四章 分类

4.1 模型评估和性能度量

4.2 k最近邻分类

4.3 决策树

4.4 贝叶斯分类

4.5 组合分类

4.6 案例：信用违约预测

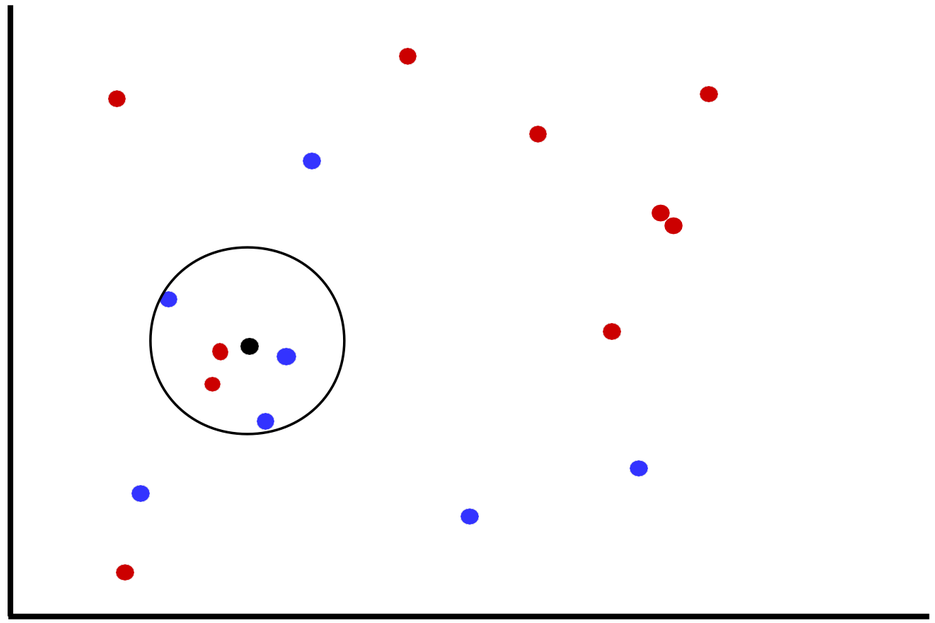
k近邻分类

k-nearest neighbor(k-NN)

k-NN分类算法的两个基本步骤：

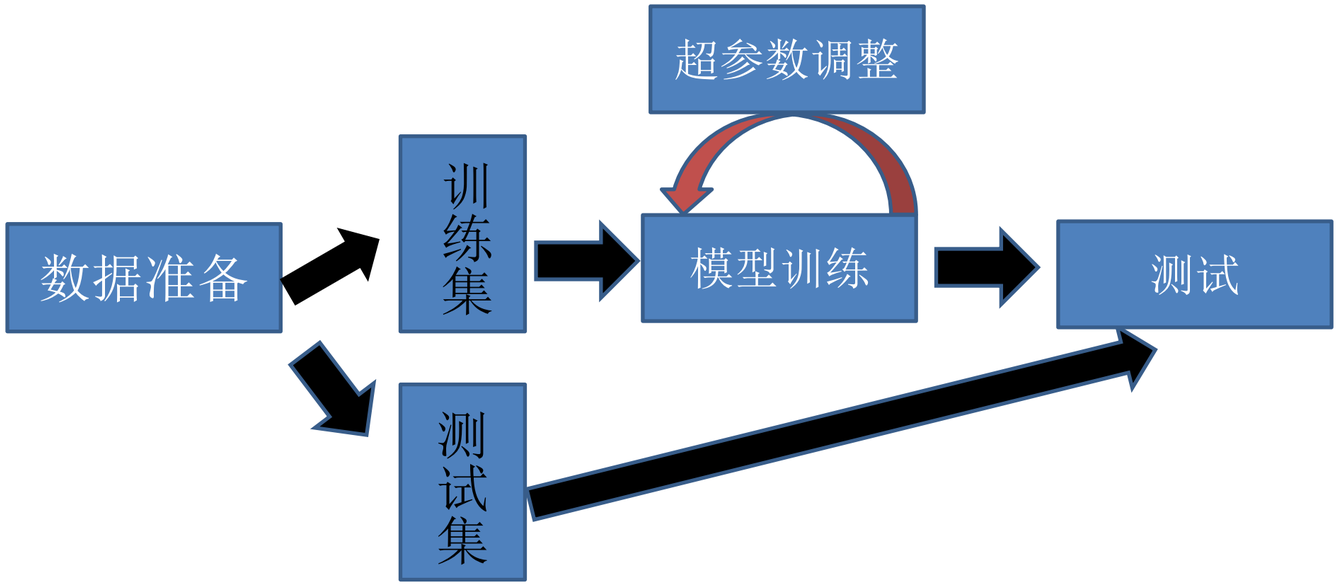
1. 对某个测试样本，在训练集中找到k个离其最近的训练样本，即k个近邻；

2. 找出这k个近邻中出现次数最多的那个类标签作为该测试样本标签的预测结果。



给定测试点（黑色的点），当k=5时，k-NN对该测试点的预测结果为：蓝色。

典型的分类过程

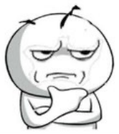


所有分类方法都是先建模，后预测的吗？

“惰性”学习法(lazy learning)：存储训练集，等测试样本来了才建模和预测，没有独立的模型训练过程！

k-NN算法的特点

- k-NN算法仍然为有监督的学习算法；
- 它属于“惰性”学习算法，即不会预训练一个分类器或预测模型，而是将模型的构建与未知数据类别的预测同时进行。



K-NN算法不仅可以对离散因变量（y对应类别）预测来进行分类，也可以对连续因变量做预测（y是连续值，为回归问题），怎么做？

决定k-NN分类算法的关键

关键1：哪些才是近邻，即如何衡量相似度？

常用的相似度/距离度量包括：欧式距离、 余弦相似度等。

计算相似度之前可能需要进行特征规范化。

算法实现需要考虑的问题

一般需要考虑算法的训练和测试阶段需要的计算量（时间复杂度）、内存需求等。

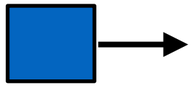
- k-NN不需要训练，所以没有训练时间；
- 测试阶段，需要计算测试样本与n个训练样本之间的相似度，当n很大时这可能会很慢；

总结：当应用于规模大、响应快、以及计算资源有限的应用时，需要寻求近邻的快速实现算法，比如对稀疏数据可以先建立索引。

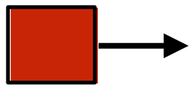
对稀疏数据建立索引*

假设测试样本只有两个特征的值大于0，左图中的红色和蓝色。

要从训练集中找到与该测试样本相近的训练样本，只要从这两个特征至少有一个的值是大于0的训练样本中找。



$[x_1, x_3, x_9]$



$[x_2, x_4, x_5]$

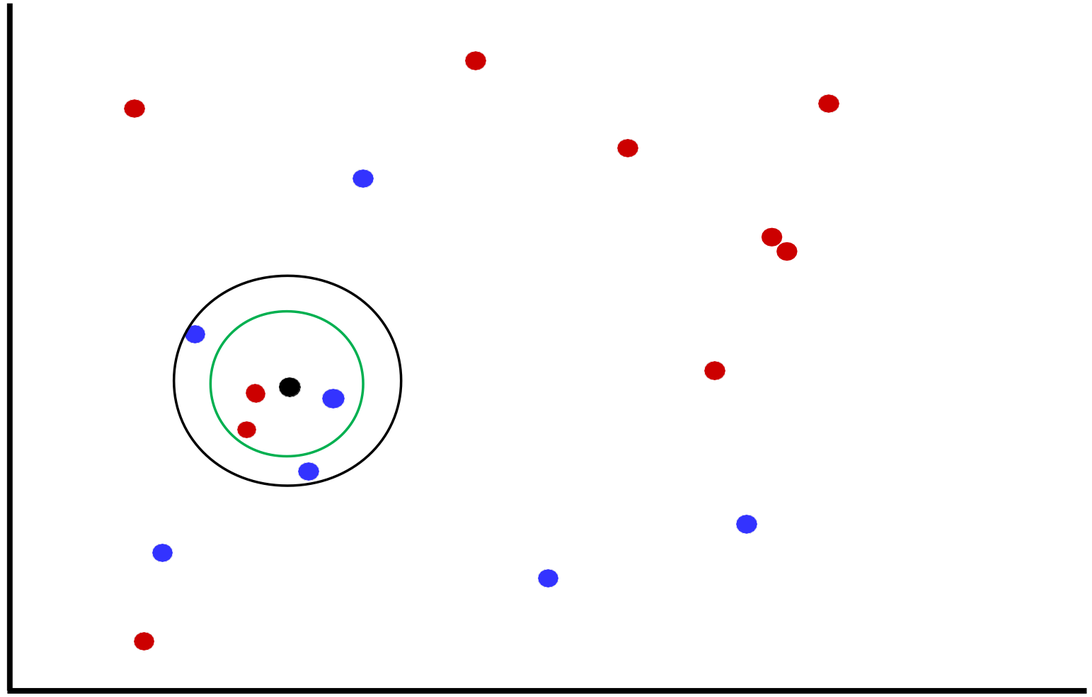
我们对每个特征建立一个对应训练样本的索引。如左图表示蓝色和红色特征分别不为0的训练样本。

通过以上特征-样本的索引，只要计算这6个训练样本与测试样本之间的相似度，而不是整个训练集中的样本。

以上方法利用了数据的稀疏性，比如文本数据

决定k-NN分类算法的关键

关键2：怎么设置近邻的个数，即k取多少？



给定测试点（黑色的点），

当k=5时， 预测结果为：蓝色。

当k=3时， 预测结果为：红色。

k取值对k-NN算法的影响

- **k太小**：容易出现过拟合，使得结果对训练集中的噪声很敏感；

思考：如果k=1时，测试样本的标签预测值取决于什么？

- **k太大**：模型太简单（欠拟合），过于平滑，不能有效反映数据集中类别之间的差别。

思考：如果k=n（训练样本个数），测试样本的标签被预测成什么？

k的取值

一般规则：

- 为了避免（二分类）出现类别一样多的情况，k一般取奇数；
- 对于数据规模大、结构复杂的情况，k一般取得大一点；对较小的数据集，k的取值得相对小。

具体操作：

从k=1开始，逐渐增大k的值，一般不超过20。最后采用在验证集上得到准确率最高的那个k值。