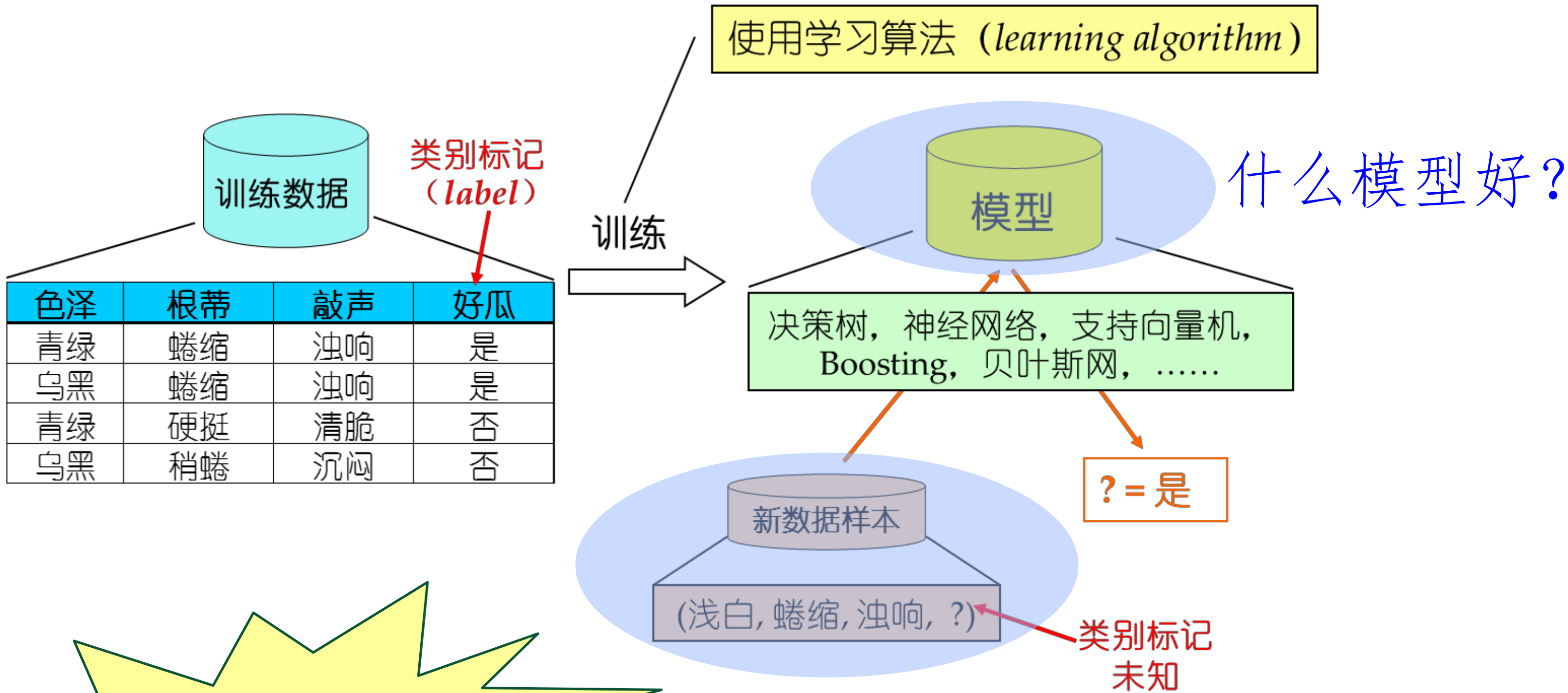
二、模型评估与选择

典型的机器学习过程



色泽	根蒂	敲声	好瓜
青绿	蜷缩	浊响	是
乌黑	蜷缩	浊响	是
青绿	硬挺	清脆	否
乌黑	稍蜷	沉闷	否

能很好地适用于 unseen instance

例如，**错误率**低、**精度**高

然而，我们手上没有 unseen instance，……

泛化误差 vs. 经验误差

错误率(error rate): e = a/m; m个样本中错误a个.

准确率(精度, accuracy): a = 1 - e.

误差(error): 实际预测输出与样本的真实输出之间的差异.

泛化误差(generalization error): 在“未来”样本上的误差;

经验误差（empirical error）: 在训练集上的误差, 亦称“训练误差”(training error)

- 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)

经验误差与过拟合

- 过拟合:

学习器把训练样本学习的“太好”，将训练样本本身的特点当做所有样本的一般性质，导致泛化性能下降

  - 优化目标加正则项
  - early stop, dropout

- 欠拟合：

对训练样本的一般性质尚未学好

  - 决策树: 拓展分支
  - 神经网络: 增加训练轮数

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

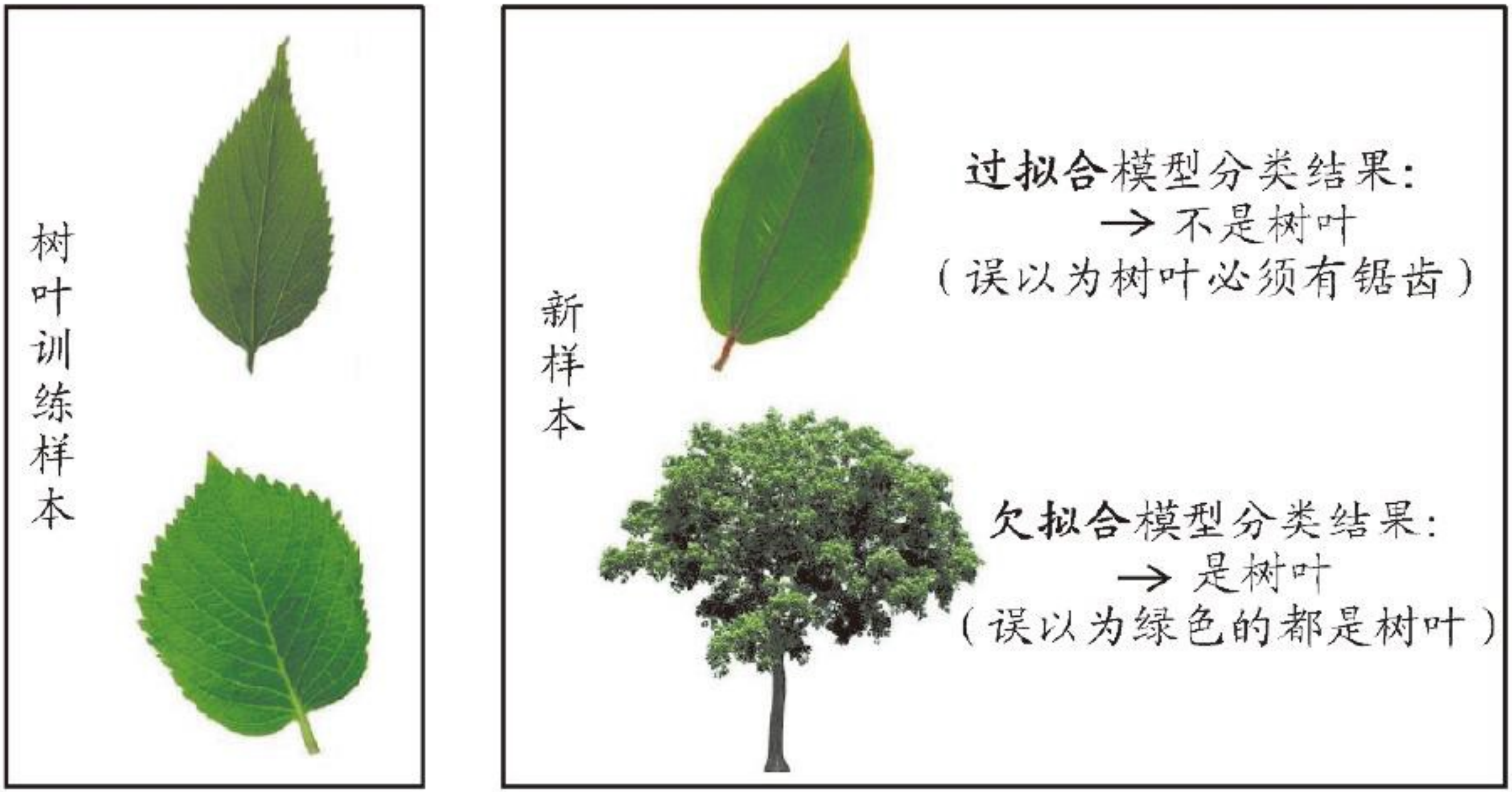


图 2.1 过拟合、欠拟合的直观类比

模型选择 (model selection)

三个关键问题：

- 如何获得测试结果？ ⇨ 评估方法
- 如何评估性能优劣？ ⇨ 性能度量
- 如何判断实质差别？ ⇨ 比较检验

评估方法

关键：怎么获得“测试集”(test set)？

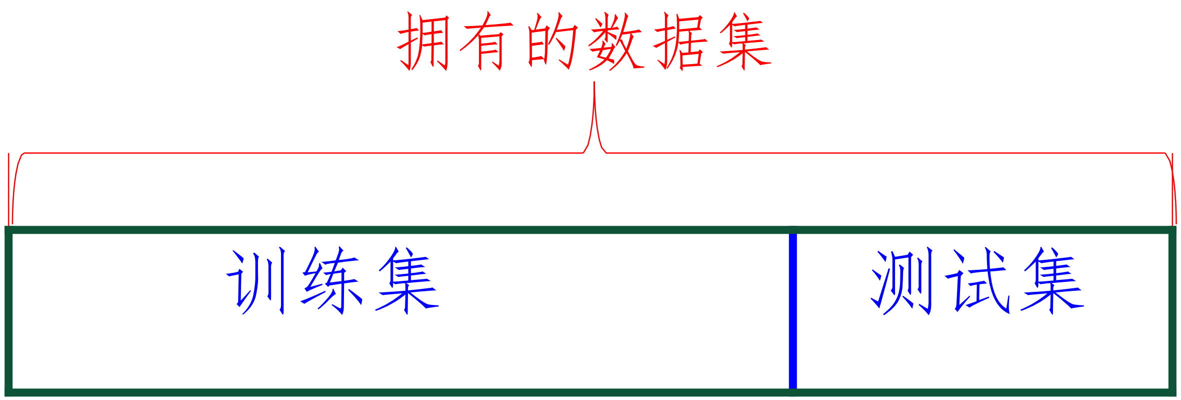
测试集应该与训练集“互斥”

测试集不应出现在训练集中！

常见方法：

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

留出法



通常将包含 m 个样本的数据集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$
拆分成训练集 S 和测试集 T(互斥)：

注意:

- 保持数据分布一致性, 例如: 分层采样stratified sampling
- 多次重复划分 (例如: 100次随机划分)--单次结果不稳定!
- 测试集(比例)不能太大（why？）、不能太小 (例如：1/5~1/3)

k-折交叉验证法(k-fold cross validation)

若 k = m，则得到“留一法”
(leave-one-out, LOO)

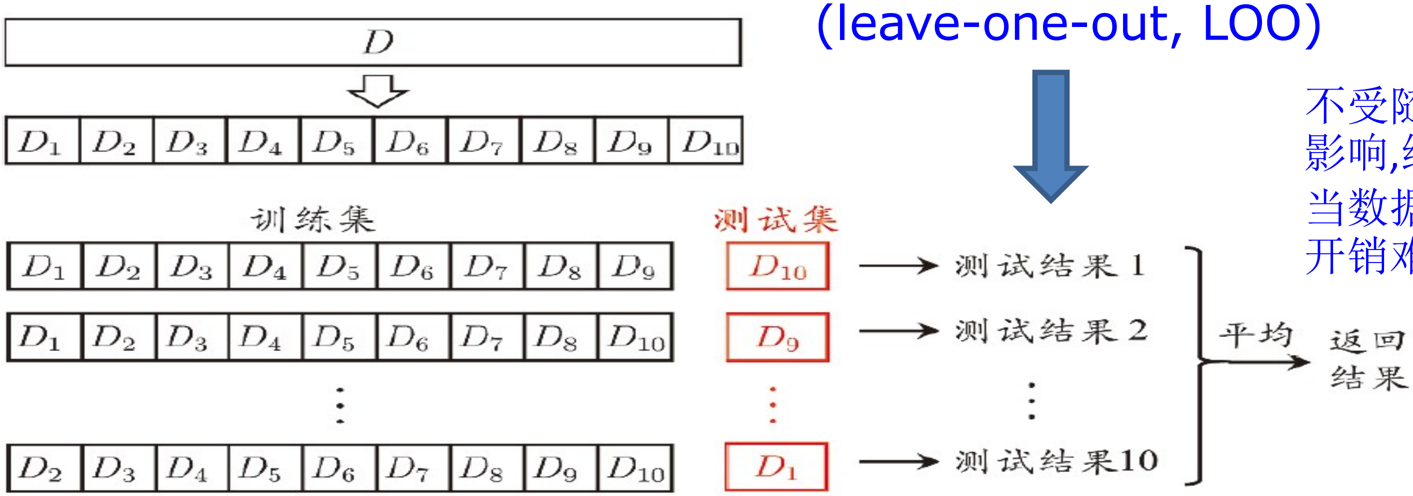


图 2.2 10 折交叉验证示意图

不受随机样本划分方式的影响,结果往往比较准确；当数据集比较大时，计算开销难以忍受。

- 将数据集分层采样划分为 k 个大小相似的互斥子集，每次用 k-1 个子集的并集作为训练集，余下的子集作为测试集，最终返回 k 个测试结果的均值，k 最常用的取值是10.
- k折交叉验证通常随机使用不同的划分重复p次，最终的评估结果是这p次k折交叉验证结果的均值，例如常见的“10次10折交叉验证”

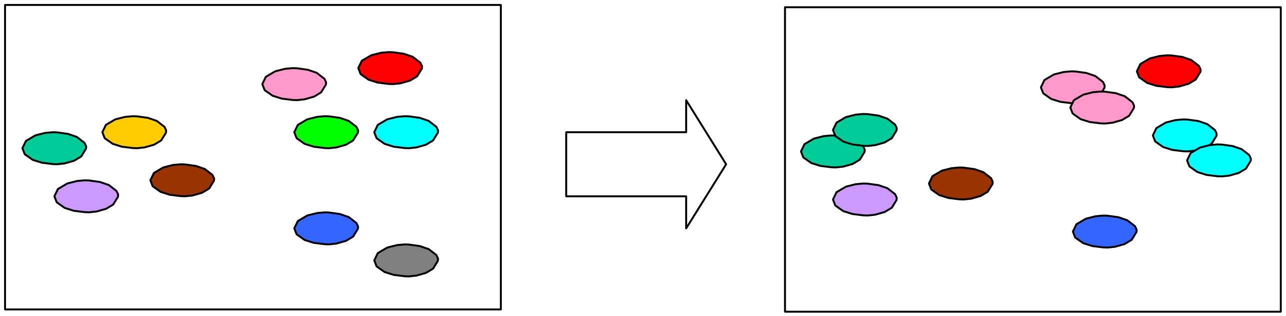
自助法

- 从初始数据集中产生多个不同的训练集，对集成学习有很大的好处
- 自助法在数据集较小、难以有效划分训练/测试集时很有用；
- 由于改变了数据集分布可能引入估计偏差，在数据量足够时，留出法和交叉验证法更常用。

基于“自助采样”(boot

亦称“有放回采样”、“可重复采样”

D



D’

约有 36.8% 的样本不出现

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

- ➢ 训练集与原样本集同规模
- ➢ 数据分布有所改变

“ **包外估计** ”(out-of-bag estimation)

初始数据集D中约有36.8%的样本未出现在采样数据集D’中。

D和D’都有m个样本。

D’用作训练集，D\D’用作测试集 – 未出现在D’的36.8%数据在D\D’中

“调参”与最终模型

算法的参数：一般由人工设定，亦称“超参数”

模型的参数：一般由学习确定

调参过程相似：先产生若干模型，然后基于某种评估方法进行选择

参数调得好不好对最终性能有关键影响

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

算法参数选定后，要用“训练集+验证集”重新训练最终模型

模型选择 (model selection)

三个关键问题:

- 如何获得测试结果? ⇨ 评估方法
- 如何评估性能优劣? ⇨ 性能度量
- 如何判断实质差别? ⇨ 比较检验

性能度量

性能度量(performance measure)是衡量模型泛化能力的评价标准，反映了任务需求.

使用不同的性能度量往往会导致不同的评判结果.

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

性能度量

预测任务中：给定样例集 $D = \{x_1, y_1\} \cdots \{x_m, y_m\}$，$y_i$ 是实例 x_i 的真实标记，要评估学习器 f 的性能，就要把学习器预测结果 f(x) 与真实标记 y 进行比较。

- 回归(regression) 任务常用均方误差：

$$E(f; D) = \frac{1}{m} \sum_{i=1}^{m} \left(f\left(\boldsymbol{x}_i\right) - y_i\right)^2$$

- 更一般的，对于数据分布 D 和概率密度函数 p(.)，均方误差可以描述为：

$$E(f; \mathcal{D}) = \int_{\boldsymbol{x} \sim \mathcal{D}} \left(f\left(\boldsymbol{x}\right) - y\right)^2 p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$$

错误率 vs. 精度

- 错误率：

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}\left(f\left(\boldsymbol{x}_i\right)\neq y_i\right)$$

- 精度：

$$\begin{aligned}\operatorname{acc}(f;D) &= \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}\left(f\left(\boldsymbol{x}_i\right)=y_i\right)\\ &= 1-E(f;D)\ .\end{aligned}$$

- 更一般的，对于数据分布 D 和概率密度函数 p(.)，错误率与精度可以描述为：

$$E(f;\mathcal{D})=\int_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{I}\left(f\left(\boldsymbol{x}\right)\neq y\right)p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$$

$$\begin{aligned}\operatorname{acc}(f;\mathcal{D}) &= \int_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{I}\left(f\left(\boldsymbol{x}\right)=y\right)p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\\ &= 1-E(f;\mathcal{D})\ .\end{aligned}$$

查准率 vs. 查全率

- 查准率(精度，Precision)：精度是**精确性**的度量，表示被分为正例的示例中实际为正例的比例。

- 查全率(Recall)：召回率是**覆盖面**的度量，度量有多个正例被分为正例。

查准率 vs. 查全率

- 查准率(准确率，Precision)：Precision 是精确性的度量，表示被分为正例的实例中实际为正例的比例。（预测中的正例）

$$P = \frac{TP}{TP + FP}$$

- 查全率(召回率，Recall)：Recall 是覆盖面的度量，度量有多个正例被分为正例。（真实情况中被预测中的正例）

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

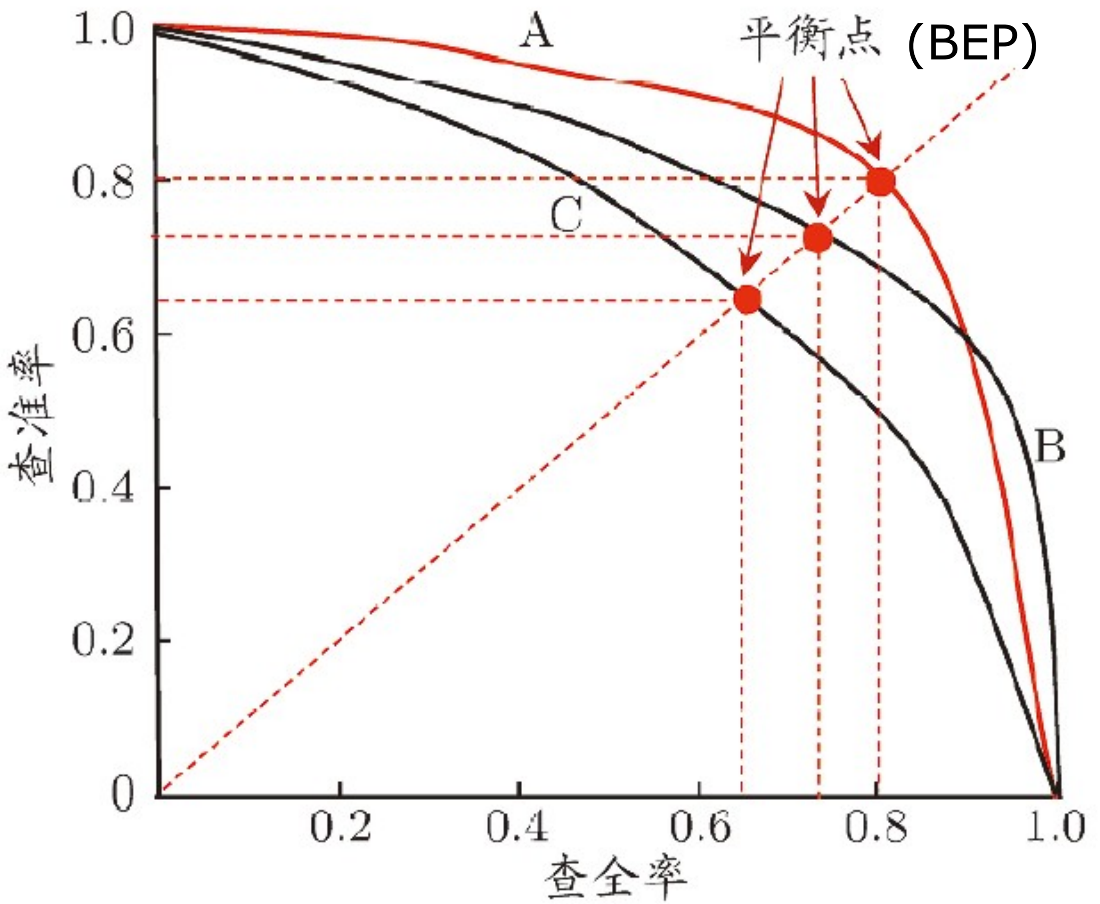
表 2.1 分类结果混淆矩阵

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)

1. 查准率高时，查全率往往偏低；
   提高precision只挑选最有把握的样本，导致Recall低。
2. 查全率高时，查准率往往偏低。
   增加样本数可以提高Recall，导致Precision变低。

PR图, BEP(Break-Even Point)

根据学习器的预测结果按正例可能性大小对样例进行排序，并逐个把样本作为正例进行预测



P-R图：

- 学习器 A 优于 学习器 C
- 学习器 B 优于 学习器 C
- 学习器 A ?? 学习器 B

BEP：平衡点(Break-Even Point)，**P = R时的取值**：

- 学习器 A 优于 学习器 B
- 学习器 A 优于 学习器 C
- 学习器 B 优于 学习器 C

F1

比 BEP 更常用的 F1 度量—P和R的调和平均(harmonic mean)：

$$\frac{1}{F1}=\frac{1}{2}\cdot\left(\frac{1}{P}+\frac{1}{R}\right) \Longrightarrow F1=\frac{2\times P\times R}{P+R}=\frac{2\times TP}{\text{样例总数}+TP-TN}$$

F1的一般形式F_β，表达对查准率/查全率有不同偏好(相对重要性)：

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

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

推荐系统中，更希望推荐内容是用户感兴趣的--P更重要；
逃犯信息检索中，更希望尽可能少漏掉罪犯--R更重要。

例子 – 捕鱼

- 不妨举这样一个例子：某池塘有1400条鲤鱼，300只虾，300只鳖。现在以捕鲤鱼为目的。撒一大网，逮着了700条鲤鱼，200只虾，100只鳖。那么，这些指标分别如下：
  1. 正确率 = 700 / (700 + 200 + 100) = 70%
  2. 召回率 = 700 / 1400 = 50%
  3. F_1 = 70%*50%*2 / (70% + 50%) = 58.3%
- 不妨看看如果把池子里的所有的鲤鱼、虾和鳖都一网打尽，这些指标又有何变化：
  1. 正确率 = 1400 / (1400 + 300 + 300) = 70%
  2. 召回率 = 1400 / 1400 = 100%
  3. F_1 = 70%*100%*2 / (70% + 100%) = 82.35%

宏xx vs. 微xx

若能得到多个混淆矩阵：

(例如多次训练/测试的结果，多分类的两两混淆矩阵)

宏(macro-)查准率、查全率、F1

-各个混淆矩阵上先分别计算出P和R，再平均

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

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

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

微(micro-)查准率、查全率、F1

-各个混淆矩阵上先求出平均，再分别计算出P和R

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

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

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

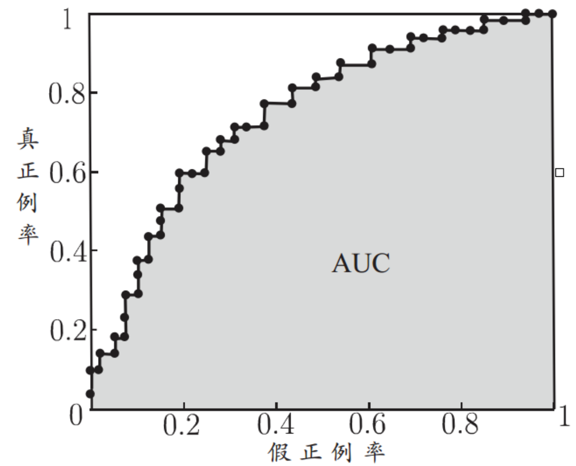
ROC, AUC

- 很多学习器是为测试样本产生一个**实值或概率预测**；
- 将预测值与一个分类阈值进 t 行比较，大于 t 分为正例，否则为反类。这个值 **t 直接决定了机器的泛化能力**。
- 如神经网络一般是对每个测试样本预测 [0.0, 1.0]之间的值，再与阈值 0.5 进行比较；
- 根据预测的实值，将测试样本进行排序：
  - 最可能是正例的排在最前；
  - 最不可能是正例的排在最后；
  - 寻找某个截断点 (cut point) 将样本分为两个部分，前正 || 后负
  - 不同的任务中，根据任务需求来采用不同的截断点：
    - 更重视P，则可选择排序靠前的位置截断；
    - 更重视R，则可选择排序靠后的位置截断。

ROC

类似P-R曲线，根据学习器的预测结果对样例排序，并逐个作为正例进行预测，以“**假正例率**”**为横轴**，“**真正例率**”**为纵轴**可得到ROC曲线,全称“**受试者工作特征**”.

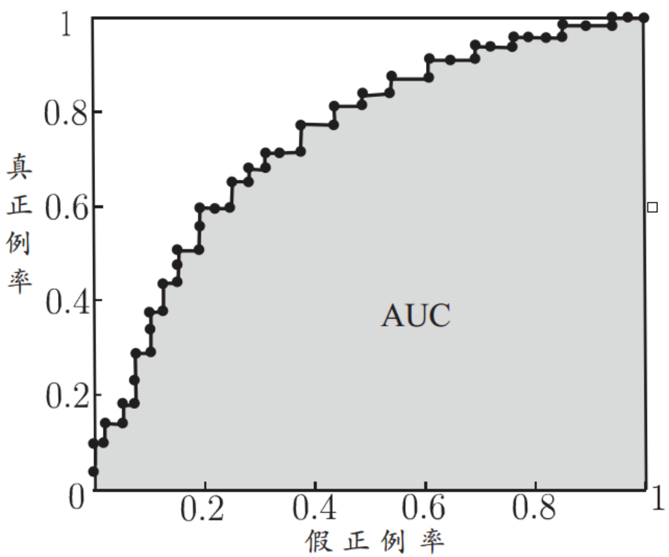
ROC图的绘制：给定 m^+ 个正例和 m^- 个负例，根据学习器预测结果对样例进行排序，将分类阈值设为每个样例的预测值，当前标记点坐标为 (x, y), 当前若为真正例，则对应标记点的坐标为 $(x, y+\frac{1}{m^+})$;当前若为假正例，则对应标记点的坐标为$(x+\frac{1}{m^-}, y)$,然后用线段连接相邻点.



基于有限样例绘制的 ROC 曲线与 AUC

AUC

若某个学习器的ROC曲线被另一个学习器的曲线“包住”，则后者性能优于前者；否则如果曲线交叉，可以根据ROC曲线下面积大小进行比较，也即AUC值.



基于有限样例绘制的 ROC 曲线与 AUC

假设ROC曲线由 $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ 的点按序连接而形成 $(x_1 = 0, x_m = 1)$，则：AUC可估算为：

$$\mathrm{AUC} = \frac{1}{2} \sum_{i=1}^{m-1} (x_{i+1} - x_i) \cdot (y_i + y_{i+1})$$

AUC衡量了样本预测的排序质量。

非均等代价

现实任务中**不同类型的错误所造成的后果很可能不同**，为了权衡不同类型错误所造成的不同损失，可为错误赋予“**非均等代价**”（unequal cost）。

表 2.2　二分类代价矩阵

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

- 在非均等代价下，不再最小化错误次数，而是最小化“总体代价”代价敏感(cost-sensitive)错误率：

$$E(f;D;cost) = \frac{1}{m}\left(\sum_{\boldsymbol{x}_i \in D^+} \mathbb{I}\left(f\left(\boldsymbol{x}_i\right) \neq y_i\right) \times cost_{01} + \sum_{\boldsymbol{x}_i \in D^-} \mathbb{I}\left(f\left(\boldsymbol{x}_i\right) \neq y_i\right) \times cost_{10}\right)$$

代价曲线

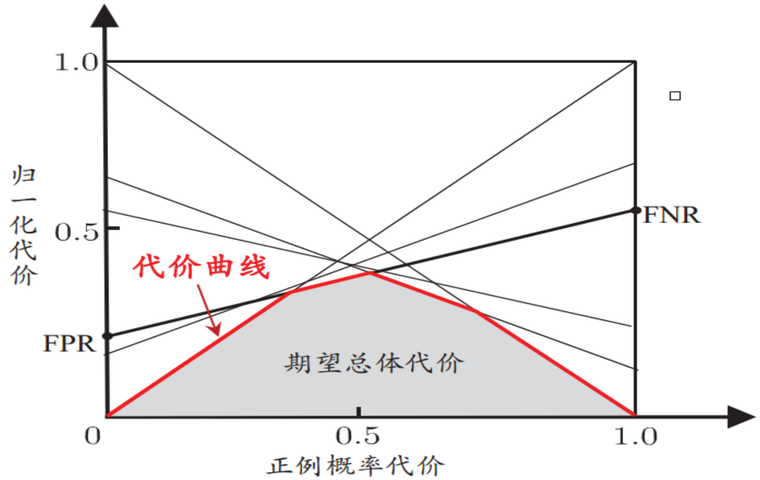
在非均等代价下，ROC曲线不能直接反映出学习器的期望总体代价，而“代价曲线”可以。

代价曲线的横轴是取值为[0,1]的正例概率代价

$$P(+)cost = \frac{p \times cost_{01}}{p \times cost_{01} + (1-p) \times cost_{10}}$$

纵轴是取值为[0,1]的归一化代价

$$cost_{norm} = \frac{\text{FNR} \times p \times cost_{01} + \text{FPR} \times (1-p) \times cost_{10}}{p \times cost_{01} + (1-p) \times cost_{10}}$$

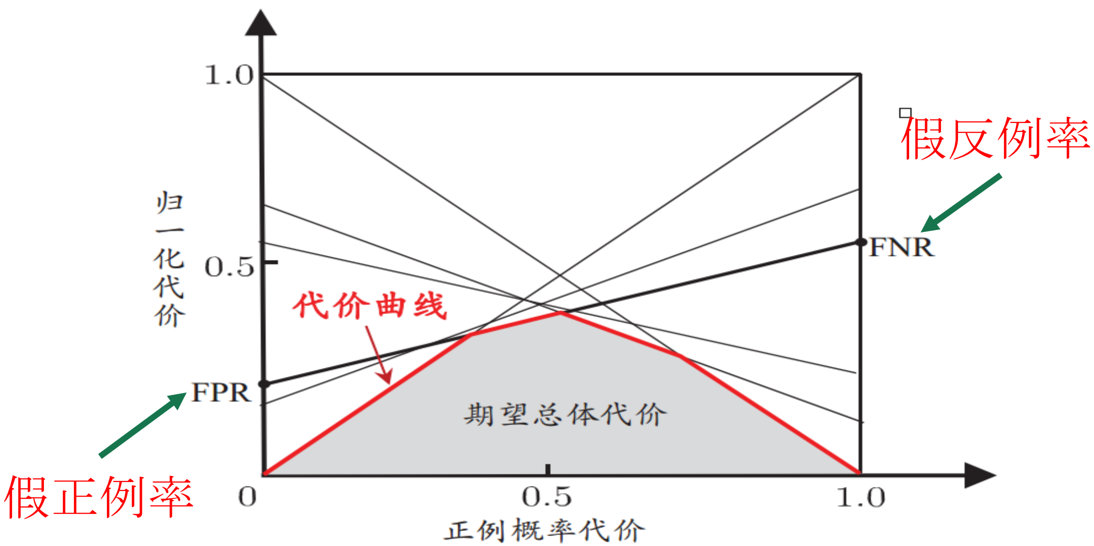


代价曲线与期望总体代价

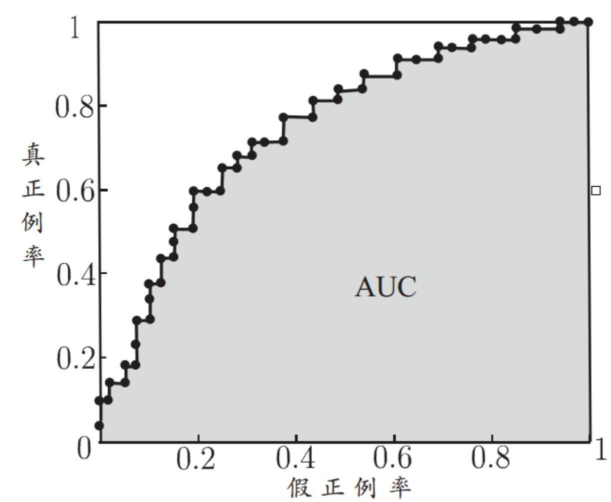
代价曲线

代价曲线图的绘制：ROC曲线上每个点对应了代价曲线上的一条线段，设ROC曲线上点的坐标为 (FPR, TPR) (假正例率，真正例率)，则可相应计算出 假反例率FNR(1 - TPR)，然后在代价平面上绘制一条从 (0,FPR) 到 (1,FNR) 的线段，线段下的面积即表示了该条件下的期望总体代价；

如此将ROC曲线上的每个点转化为代价平面上的一条线段，然后取所有线段的下界，围成的面积即为所有条件下学习器的期望总体代价。



代价曲线与期望总体代价



基于有限样例绘制的 ROC 曲线与 AUC

模型选择 (model selection)

三个关键问题:

- 如何获得测试结果? ⇨ 评估方法
- 如何评估性能优劣? ⇨ 性能度量
- 如何判断实质差别? ⇨ 比较检验

比较检验

在某种度量下取得评估结果后，是否可以直接比较以评判优劣？

NO！

因为：

- 测试性能不等于泛化性能（我们只在测试集上进行了评估）
- 测试性能随着测试集的变化而变化
- 很多机器学习算法本身有一定的随机性

机器学习 ⇨ “概率近似正确”

假设检验为学习器性能比较提供了重要依据，基于其结果我们可以推断出若在测试集上观察到学习器A比B好，则**A的泛化性能是否在统计意义上优于B，以及这个结论的把握有多大**。

计算学习理论

Computational learning theory

Leslie Valiant（莱斯利 维利昂特）
(1949-)
2010年图灵奖

PAC (Probably Approximately Correct)

learning model [Valiant, 1984]

$$P(|f(\boldsymbol{x}) - y| \leq \epsilon) \geq 1 - \delta$$

什么是“可学习的”

- 概率近似正确(Probably Approximately Correct, PAC)

    我们希望以比较大的把握学得比较好的模型，即**以较大概率学得误差满足预设上限的**模型.

    令δ表示置信度，上述要求形式化为：

    定义 PAC辨识(PAC Identify)

    对$0 < \epsilon, \delta < 1$,所有$c \in \mathcal{C}$和分布$\mathcal{D}$,若存在学习算法$\mathcal{L}$, 其输出假设$h \in \mathcal{H}$ 满足

$$P(E(h) \leq \epsilon) \geq 1 - \delta,$$

则称学习算法$\mathcal{L}$能从假设空间$\mathcal{H}$中PAC辨识概念类$\mathcal{C}$.

这样的学习算法$\mathcal{L}$能以较大概率(至少$1-\delta$)学得目标概念c 的近似(误差最多为ϵ).

常用方法

假设检验为学习器性能比较提供了重要依据，基于其结果我们可以推断出若在测试集上观察到学习器A比B好，则A的泛化性能是否在统计意义上优于B，以及这个结论的把握有多大。

统计假设检验 (hypothesis test) 为学习器性能比较提供了重要依据。

统计显著性

- 两学习器比较
  - 交叉验证 t 检验 (基于成对 t 检验)

    k 折交叉验证； 5x2交叉验证
  - McNemar 检验 (基于列联表，卡方检验)
- 多学习器比较
  - Friedman + Nemenyi
    - Friedman检验 (基于序值，F检验; 判断”是否都相同”)
    - Nemenyi 后续检验 (基于序值，进一步判断两两差别)

二项检验

记泛化错误率为 ϵ 测试错误率为 $\hat{\epsilon}$，假定测试样本从样本总体分布中独立采样而来，我们可以使用“二项检验”对 $\epsilon \leq \epsilon_0$ 进行假设检验。

- 泛化错误率为 ϵ 的学习器在一个样本上犯错误的概率为 ϵ ，
- 测试错误率 $\hat{\epsilon}$ 意味着在 m 个测试样本中恰有 $\hat{\epsilon} \times m$ 个被误分类,
- 假定测试样本从样本总体分布中独立采样而来，那么泛化错误率为 ϵ 的学习器将其中 m’ 个样本误分类、其余样本正确分类的概率是： $\epsilon^{m'}(1-\epsilon)^{m-m'}$
- 由此可估算出其恰将 $\hat{\epsilon} \times m$ 个样本误分类的概率如下：

$$P(\hat{\epsilon};\epsilon) = \binom{m}{\hat{\epsilon} \times m} \epsilon^{\hat{\epsilon} \times m}(1-\epsilon)^{m-\hat{\epsilon} \times m}$$

- 上式表达了在包含 m 个样本的测试集上，泛化错误率为 ϵ 的学习器被测得测试错误率为 $\hat{\epsilon}$ 的概率。（**二项分布**）

二项检验

假设 $\epsilon \leq \epsilon_0$，若测试错误率小于

$$\bar{\epsilon} = \max \epsilon \quad \text{s.t.} \quad \sum_{i=\epsilon_0 \times m+1}^{m} \binom{m}{i} \epsilon^i (1-\epsilon)^{m-i} < \alpha$$

此时若测试错误率 $\hat{\epsilon}$ 小于临界值 $\bar{\epsilon}$，根据二项检验有：

在 α 的显著度下，假设不能被拒绝，也即能以 $1-\alpha$ 的置信度认为，模型的泛化错误率不大于 ϵ_0。**(以 α 的置信度认为学习器的泛化错误率大于 ϵ_0)**

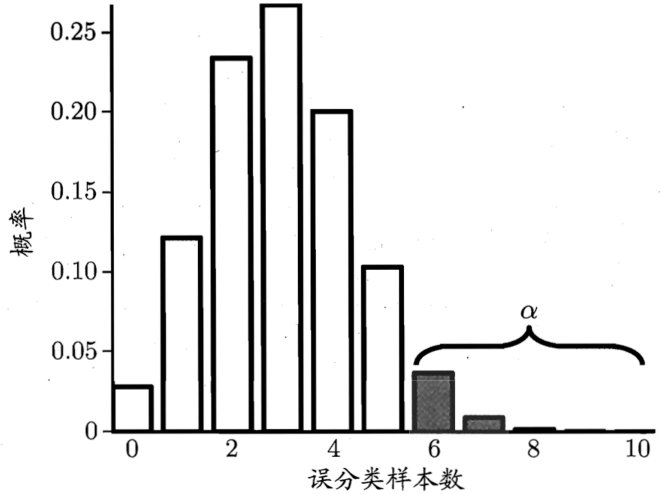


图 2.6 二项分布示意图($m = 10, \epsilon = 0.3$)

t检验

对应的，面对多次重复留出法或者交叉验证法进行多次训练/测试时可使用“t检验”。

假定得到了k个测试错误率，$\hat{\epsilon}_1, \hat{\epsilon}_2, \ldots, \hat{\epsilon}_k$,假设，$\epsilon = \epsilon_0$ 对于显著度α,若 $[t_{-\alpha/2}, t_{\alpha/2}]$ 位于临界范围$|\mu - \epsilon_0|$内，则假设不能被拒绝，即可认为泛化错误率$\epsilon = \epsilon_0$ ，其置信度为 $1 - \alpha$.

交叉验证t检验

现实任务中，更多时候需要对不同学习器的性能进行比较

对两个学习器A和B, 若 k 折交叉验证得到的测试错误率分别为 $\epsilon_1^A, \ldots, \epsilon_k^A$ 和 $\epsilon_1^B, \ldots, \epsilon_k^B$ ，可用 k 折交叉验证“成对t检验”进行比较检验。若两个学习器的性能相同，则他们使用相同的训练/测试集得到的测试错误率应相同，即 $\epsilon_i^A = \epsilon_i^B$.

交叉验证t检验

先对每对结果求差，$\Delta_i = \epsilon_i^A - \epsilon_i^B$,若两个学习器性能相同，则差值应该为0，继而用 $\Delta_1, ..., \Delta_k$ 来对“学习器A与B性能相同”这个假设做t检验。

假设检验的前提是测试错误率为泛化错误率的独立采样，然而由于样本有限，使用交叉验证导致训练集重叠，测试错误率并不独立，从而过高估计假设成立的概率，为缓解这一问题，可采用“5*2交叉验证”法.

5*2交叉验证法

所谓5*2折交叉验证就是做5次二折交叉验证，每次二折交叉验证之前将数据打乱，使得5次交叉验证中的数据划分不重复。为缓解测试数据错误率的非独立性，仅计算第一次2折交叉验证结果的平均值 $\mu = 0.5(\Delta_1^1 + \Delta_1^2)$ 和每次二折实验计算得到的方差 $\sigma_i^2 = \left(\Delta_i^1 - \frac{\Delta_i^1+\Delta_i^2}{2}\right)^2 + \left(\Delta_i^2 - \frac{\Delta_i^1+\Delta_i^2}{2}\right)^2$,则变量

$$\tau_t = \frac{\mu}{\sqrt{0.2\sum_{i=1}^{5}\sigma_i^2}}$$

服从自由度为5的t分布。

“误差”包含了哪些因素？

换言之，从机器学习的角度看，

“误差”从何而来?

偏差与方差

通过实验可以估计学习算法的泛化性能，而“**偏差-方差分解**”可以用来帮助**解释泛化性能**。偏差-方差分解试图对学习算法期望的泛化错误率进行拆解。

对测试样本 x,令 y_D为 x 在数据集中的标记, y 为 x 的真实标记, $f(\boldsymbol{x};D)$ 为训练集 D上学得模型 f 在 x 上的预测输出。以回归任务为例：学习算法的期望预期为：

$$\bar{f}(\boldsymbol{x}) = \mathbb{E}_D[f(\boldsymbol{x};D)]$$

使用样本数目相同的不同训练集产生的方差为

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

噪声(标记错误)为

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

偏差与方差

期望输出与真实标记的差别称为偏差，即 $bias^2(\boldsymbol{x}) = \left(\bar{f}(\boldsymbol{x}) - y\right)^2$

为便与讨论，假定噪声期望为0，也即 $\mathbb{E}_D[y_D - y] = 0$ ，对泛化误差分解：

$$
\begin{aligned}
E(f;D) &= \mathbb{E}_D\left[\left(f(\boldsymbol{x};D) - y_D\right)^2\right] \\
&= \mathbb{E}_D\left[\left(f(\boldsymbol{x};D)\boxed{- \bar{f}(\boldsymbol{x}) + \bar{f}(\boldsymbol{x})} - y_D\right)^2\right] \\
&= \mathbb{E}_D\left[\left(f(\boldsymbol{x};D) - \bar{f}(\boldsymbol{x})\right)^2\right] + \mathbb{E}_D\left[\left(\bar{f}(\boldsymbol{x}) - y_D\right)^2\right] \\
&\quad + \mathbb{E}_D\left[2\left(f(\boldsymbol{x};D) - \bar{f}(\boldsymbol{x})\right)\boxed{\left(\bar{f}(\boldsymbol{x}) - y_D\right)}\right]
\end{aligned}
$$

期望预测与$\mathbf{y_D}$一致，最后项为0

$$
\begin{aligned}
&= \mathbb{E}_D\left[\left(f(\boldsymbol{x};D) - \bar{f}(\boldsymbol{x})\right)^2\right] + \mathbb{E}_D\left[\left(\bar{f}(\boldsymbol{x}) - y_D\right)^2\right] \\
&= \mathbb{E}_D\left[\left(f(\boldsymbol{x};D) - \bar{f}(\boldsymbol{x})\right)^2\right] + \mathbb{E}_D\left[\left(\bar{f}(\boldsymbol{x})\boxed{- y + y} - y_D\right)^2\right] \\
&= \mathbb{E}_D\left[\left(f(\boldsymbol{x};D) - \bar{f}(\boldsymbol{x})\right)^2\right] + \mathbb{E}_D\left[\left(\bar{f}(\boldsymbol{x}) - y\right)^2\right] + \mathbb{E}_D\left[\left(y - y_D\right)^2\right] \\
&\quad + 2\mathbb{E}_D\left[\left(\bar{f}(\boldsymbol{x}) - y\right)\boxed{\left(y - y_D\right)}\right]
\end{aligned}
$$

又由假设中噪声期望为0，可得

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

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：

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

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

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

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

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

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

偏差与方差

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

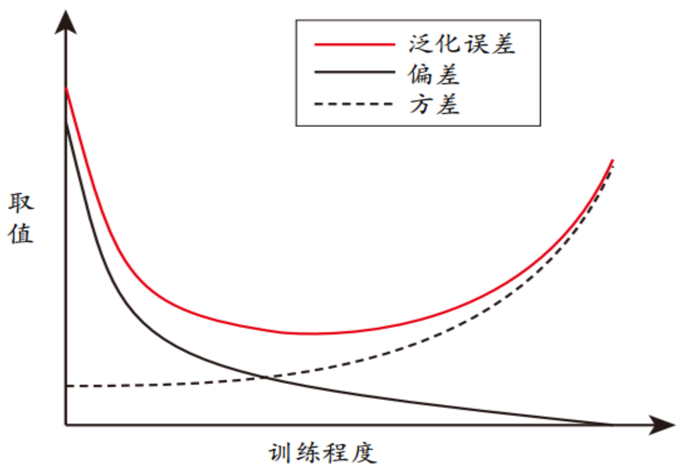
- 偏差度量了学习**算法期望预测与真实结果的偏离程度**；即刻画了学习**算法本身的拟合能力**；
- 方差度量了**同样大小训练集的变动所导致的学习性能的变化**；即刻画了**数据扰动所造成的影响**；
- 噪声表达了在当前任务上**任何学习算法所能达到的期望泛化误差的下界**；即刻画了学习**问题本身的难度**。

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所共同决定的。给定学习任务为了取得好的泛化性能，需要**使偏差小(充分拟合数据)**而且**方差较小(减少数据扰动产生的影响)**。

偏差与方差

一般来说，偏差与方差是有冲突的，称为偏差-方差窘境。
如右图所示，假如我们能控制算法的训练程度：

- 在训练不足时，学习器拟合能力不强，训练数据的扰动不足以使学习器的拟合能力产生显著变化，此时**偏差主导泛化错误率**；
- 随着训练程度加深，学习器拟合能力逐渐增强，**方差逐渐主导泛化错误率**；
- **训练充足后**，学习器的拟合能力非常强，训练数据的轻微扰动都会导致学习器的显著变化，若训练数据自身非全局特性被学到则会发生**过拟合**。



泛化误差与偏差、方差的关系示意图

前往第三站……