

机器学习（西瓜书）

⌚ type	Post
⌚ status	Published
📅 date	@2021/07/02
≡ slug	learn-4
≡ summary	机器学习笔记、作业（以西瓜书为参考教材）
≡ tags	AI 机器学习
⌚ category	ML

Reference link

1. 绪论

1.1 基本术语：

1.2 假设空间

1.3 归纳偏好

2. 模型的评估与选择

2.1 误差与过拟合

误差：

过拟合：

欠拟合

2.2 评估方法

常见的训练集划分方式

调参

2.3 性能度量

1. MSE (mean squared error) : 均方误差

2. 错误率与精度

3. 查准率(precision)/查全率(recall)/F1

4. ROC AUC

5 代价敏感错误率与代价曲线

2.4 比较检验

1 假设检验

2 交叉验证检验

3 McNemar 检验

4 Friedman 检验与 neny1 后续检验

2.5 偏差与方差

3. 线性模型(linear model)

3.1 基本形式

3.2 线性回归

用线性去拟合非线性：

联系函数g(.)

3.3 对数几率回归

3.4 线性判别分析(LDA)

3.5 多分类学习策略

3.6 类别不平衡问题

4. 决策树

4.1 基本流程

4.2 划分属性的选择

4.2.1 信息增益(ID3)

4.2.2 信息增益率(C4.5)

4.2.3 基尼指数(CART算法)

4.3 剪枝

4.4 连续与缺失的处理

4.4.1 连续变量处理：离散化

4.4.2 缺失处理

4.5 多变量决策树

5. 神经网络

6. 支持向量机

7. 贝叶斯分类器
7.1 贝叶斯定理
7.2 贝叶斯决策论
判别式模型、生成式模型
7.3 极大似然估计
7.4 朴素贝叶斯分类器
 $P(c)$ 与 $P(x_i|c)$ 的估计
拉普拉斯修正
Homework4

8 集成学习

9 聚类
原型聚类-学习向量量化

10 降维与度量学习
10.1 K 临近算法
10.2 低维嵌入
10.3 主成分分析
10.4 核化线性降维(Kernlized PCA、非线性降维)
10.5 流行学习
10.6 度量学习

基础知识
矩阵分解

Reference link

| 机器学习 归档 - 产品经理的人工智能学习库 (easyai.tech)

1. 絮论

1.1 基本术语：

1.2 假设空间

- 可以把机器学习的过程看作是在所有的假设空间中进行搜索，目标是找到与训练集最匹配的假设。

1.3 归纳偏好

- 机器学习算法在学习的过程中对于某种类型的假设的偏好
 - 奥卡姆剃刀：多个假设与观察一致，选择最简单的一个
- NFL(No Free Lunch theorem)：在f均匀分布的情况下，即所有“问题”出现机会相同的情况下，任何算法的期望误差都是相同的。也说明只有针对具体问题去讨论什么学习算法更好才有意义。

2. 模型的评估与选择

2.1 误差与过拟合

误差：

- 在训练集上的误差称为训练误差 (training error) 或经验误差 (empirical error) 。
- 在测试集上的误差称为测试误差 (test error) 。
- 学习器在所有新样本上的误差称为泛化误差 (generalization error) 。

过拟合：

过拟合：模型的学习能力过强，学习到了训练集中不具有一般性的特征，导致泛化能力反而下降

解决方式：

- L1和L2正则化：在代价函数中加入w的惩罚项

$$\tilde{J}(w; X, y) = J(w; X, y) + \alpha \Omega(w)$$

$\Omega(w)$ 常见的有两种方式：

- L1 : $l_1 : \Omega(w) = \|w\|_1 = \sum i|w_i|$
- L2 : $l_2 : \Omega(w) = \|w\|_2^2 = \sum i w_i^2$

另外L1函数一般L1正则化往往会产生很多0项，得到稀疏解，可以做特征选择；而L2则相对平缓。

| 深入理解L1、L2正则化 - 知乎 (zhihu.com)

- **数据增强**
- **Early stopping**

Early stopping便是一种迭代次数截断的方法来防止过拟合的方法，即在模型对训练数据集迭代收敛之前停止迭代来防止过拟合。因为在初始化网络的时候一般都是初始为较小的权值，训练时间越长，部分网络权值可能越大。如果我们在合适时间停止训练，就可以将网络的能力限制在一定范围内。

- **Dropout**

深度学习网络的训练过程中，对于神经网络单元，按照一定的概率将其暂时从网络中丢弃。

- 采用**交叉验证等数据处理方式**
- **Batch Normalization :**

一种非常有用的正则化方法，可以让大型的卷积网络训练速度加快很多倍，同时收敛后分类的准确率也可以大幅度的提高。

BN在训练某层时，会对每一个mini-batch数据进行标准化(normalization)处理，使输出规范到 $N(0, 1)$ 的正态分布，减少了Internal covariate shift(内部神经元分布的改变)。

欠拟合

学习能太差，训练样本的一般性质尚未学好。

- 通过增加训练时间、添加特征项、减小正则项，或者选择学习能力更强的模型解决。

2.2 评估方法

现实中无法直接获得泛化误差，所以要选择一部分测试集计算误差作为测试误差，从而对模型进行评估。

常见的训练集划分方式

1. 留出法 (**hold-out**)：直接将数据集划分为两个互斥的部分，作为训练集和测试集
2. 交叉验证(**cross validation**)：

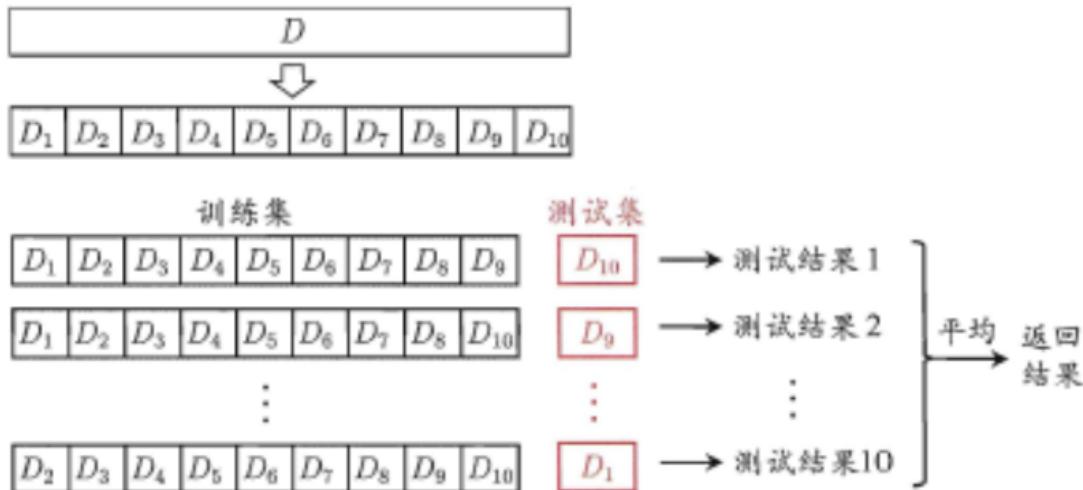


图 2.2 10 折交叉验证示意图

3. 自助法(**bootstrapping**):等概率放回随机采样

自助法的基本思想是：给定包含 m 个样本的数据集 D ，每次随机从 D 中挑选一个样本，将其拷贝放入 D' ，然后再将该样本放回初始数据集 D 中，使得该样本在下次采样时仍有可能被采到。重复执行 m 次，就可以得到了包含 m 个样本的数据集 D' 。可以得知在 m 次采样中，样本始终不被采到的概率取极限为：

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

这样，通过自助采样，初始样本集 D 中大约有36.8%的样本没有出现在 D' 中，于是可以将 D' 作为训练集， $D - D'$ 作为测试集。

调参

- 常用的做法是：对每个参数选定一个范围和步长 λ ，然后进行测试。一般来说即使是在这样的折中之后工作量依然很大。
- 当选定好模型和调参完成后，我们需要使用初始的数据集 D 重新训练模型，即让最初划分出来用于评估的测试集也被模型学习，增强模型的学习效果。

2.3 性能度量

1. MSE(mean squared error) : 均方误差

$$E(f; D) = \frac{1}{m} \sum_{i=1}^m (f(x_i) - y_i)^2 .$$

更一般的，对于数据分布 D 和概率密度函数 $p(\cdot)$ ，均方误差可描述为

$$E(f; D) = \int_{x \sim D} (f(x) - y)^2 p(x) dx .$$

2. 错误率与精度

3. 查准率(precision)/查全率(recall)/F1

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)

查准率 P 与查全率 R 分别定义为

$$P = \frac{TP}{TP + FP}, \quad (2.8)$$

$$R = \frac{TP}{TP + FN}. \quad (2.9)$$

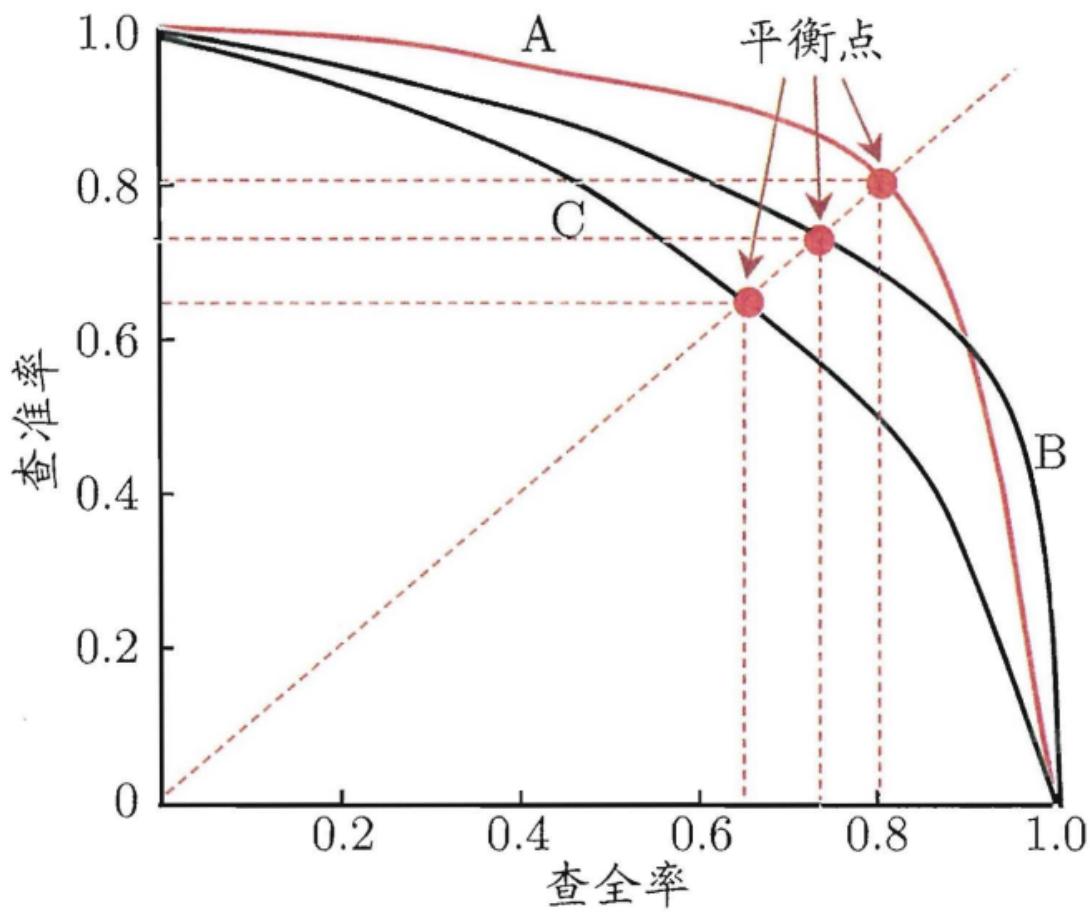


图 2.3 P-R曲线与平衡点示意图

- BEP : $P=R$ 的平衡点
- 用F-measure来对PR进行平衡

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

$\beta > 1$ 时：查全率影响更大， $\beta < 1$ 时：查准率影响更大； $=1$ 时即为F1度量，是PR的一个调和，一般来说性能较好

4. ROC AUC

| 一文看懂ROC、AUC - 知乎(zhihu.com).

		True class		
		P	N	
Hypothesized class		Y	True Positives	False Positives
		N	False Negatives	True Negatives
Column totals:		P	N	$F\text{-measure} = \frac{2TP}{2TP + FP + FN}$

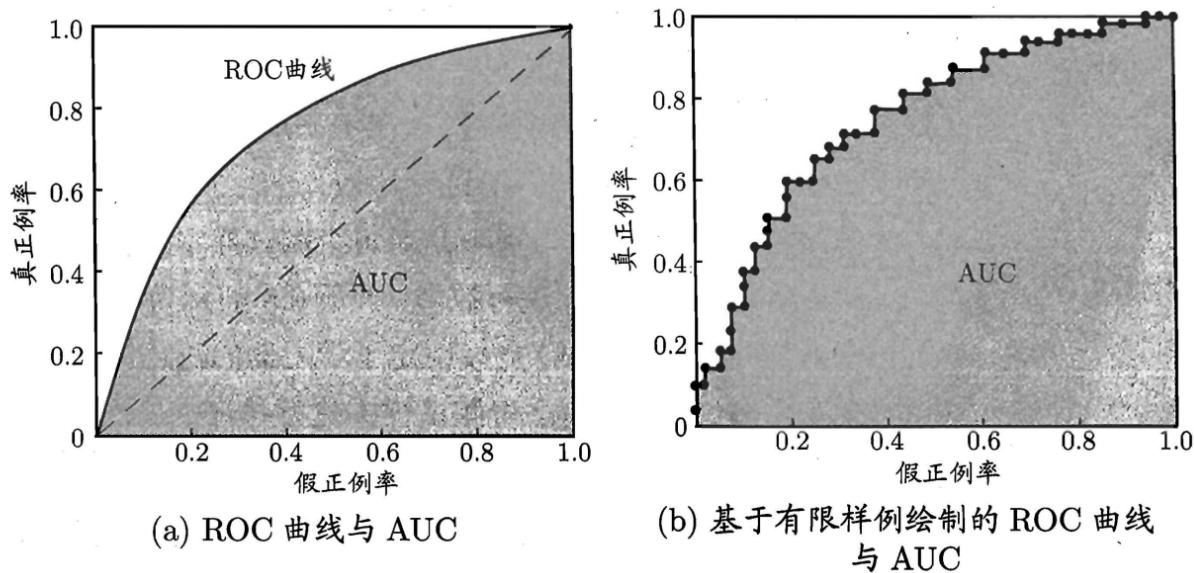
fp rate = $\frac{FP}{N}$ tp rate = $\frac{TP}{P}$
 precision = $\frac{TP}{TP+FP}$ recall = $\frac{TP}{P}$
 accuracy = $\frac{TP+TN}{P+N}$

混淆矩阵的用处

- 在有些场景下，其实准确率并没有什么实际意义，所以需要引入TPR：
 - TPR(真正例率)：预测为正，且正确的样本占总正确样本的比率
 - FPR(假正例率)：预测为正，但是错误的样本占总的错误样本的比率

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

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



例如100患者里1个患病，如果无脑预测所有患者正常，准确为99%，但是这说明不了问题，因为这种情况下对于那个负样本来说，误诊率是100%。理想情况应该是正样本预测正确的概率保持比较高，且负样本预测为正的概率降低(即不仅要把健康的都判断为健康的，还不能把太多不健康的也判断为健康)。

在这种情况下： $TPR = \frac{99}{99} = 1$ ； $FPR = \frac{1}{1} = 1$, $TPR=FPR$, 对应随机猜测

而理想情况下是 $(0, 1)$ 点，即正例全部预测为正例，且负例没有预测为正例

TPR 比 FPR 大，说明模型效果好。

- AUC为ROC曲线下的面积，其越大，说明模型效果越好

5 代价敏感错误率与代价曲线

- 实际中不同的错误造成的代价不同（如病人误诊为健康代价大，但是健康误诊为生病代价小），为不同错误类型赋予不同权值，即为代价矩阵

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，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" (cost curve) 则可达到该目的。

"代价曲线" (cost curve) 则可达到该目的。代价曲线图的横轴是取值为 [0, 1] 的正例概率代价

$$P(+)cost = \frac{p \times cost_{01}}{p \times cost_{01} + (1 - p) \times cost_{10}}, \quad (2.24)$$

其中 p 是样例为正例的概率；纵轴是取值为 [0, 1] 的归一化代价

$$cost_{norm} = \frac{FNR \times p \times cost_{01} + FPR \times (1 - p) \times cost_{10}}{p \times cost_{01} + (1 - p) \times cost_{10}}, \quad (2.25)$$

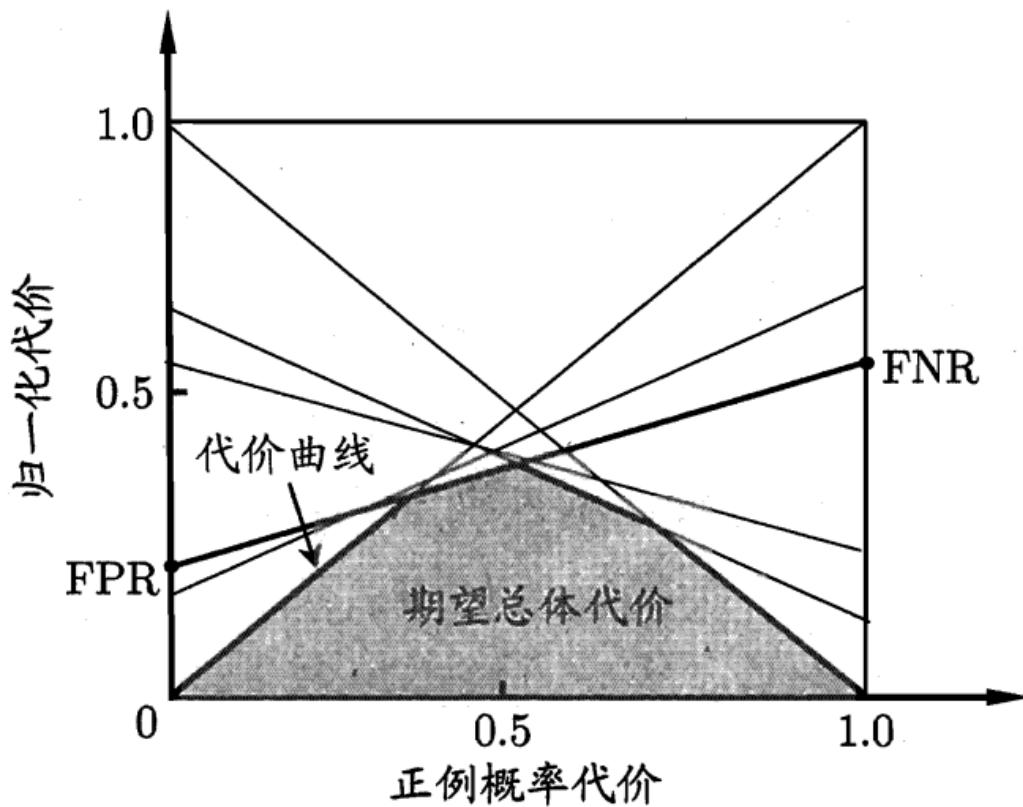


图 2.5 代价曲线与期望总体代价

- ROC 由线上每一点(相当于针对每一个点都是TPR , FPR固定了斜率 , P不断变化)对应了代价平面上的一条线段
- 代价曲线的绘制：设ROC曲线上一点的坐标为(TPR, FPR)，则可相应计算出 FNR ，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，围成的面积即为在所有条件下学习器的期望总体代价。

2.4 比较检验

虽然有了性能评价指标，但是实际比较中：

- 测试集未必能代表实际情况
- 测试集大的大小、测试集中的数据选择都会对评价指标有不同影响

- 模型本身也有一定的随机性，同样条件下多次测试结果可能不同

所以比较检验就是解决如何在相同或者不同的测试集上对不同的学习器进行比较，简单的说就是如何比较学习器的优劣。

| 此处用到了大量的概率统计中的检验方法，很多都是通用的检验方法

1 假设检验

假设检验中的"假设"是对学习器泛化错误率分布的某种判断或猜想，例 $\epsilon = \epsilon_0$. 现实任务中我们并不知道学习器的泛化错误率，只能获知其测试错误率，泛化错误率与测试错误率未必相同，但直观上两者接近的可能性应比较大，相差很远的可能性比较小. 因此可根据测试错误率估推出泛化错误率的分布.

2 交叉验证检验

- 核心思想就是用交叉验证，得到多组成对的错误率，然后用这多对错误率去分析两个模型
 - 一般先求错误率的差，假设两模型相差较小，做t检验
 - 假设检验一个重要前提是测试错误率均为泛化错误率的独立采样，但是实际采样由于不同部分训练集的重叠，并不符合独立，所以采用"5 × 2交叉验证"(做5次2折交叉验证，每次2折验证前随机打乱数据)

对两个学习器 A 和 B，若我们使用 k 折交叉验证法得到的测试错误率分别为 $\epsilon_1^A, \epsilon_2^A, \dots, \epsilon_k^A$ 和 $\epsilon_1^B, \epsilon_2^B, \dots, \epsilon_k^B$ ，其中 ϵ_i^A 和 ϵ_i^B 是在相同的第 i 折训练/测试集上得到的结果，则可用 k 折交叉验证“成对 t 检验”(paired t-tests)来进行比较检验。这里的基本思想是若两个学习器的性能相同，则它们使用相同的训练/测试集得到的测试错误率应相同，即 $\epsilon_i^A = \epsilon_i^B$.

具体来说，对 k 折交叉验证产生的 k 对测试错误率：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，则差值均值应为零。因此，可根据差值 $\Delta_1, \Delta_2, \dots, \Delta_k$ 来对“学习器 A 与 B 性能相同”这个假设做 t 检验，计算出差值的均值 μ 和方差 σ^2 ，在显著度 α 下，若变量

$$\tau_t = \left| \frac{\sqrt{k}\mu}{\sigma} \right| \quad (2.31)$$

小于临界值 $t_{\alpha/2, k-1}$ ，则假设不能被拒绝，即认为两个学习器的性能没有显著差别；否则可认为两个学习器的性能有显著差别，且平均错误率较小的那个学习器性能较优。这里 $t_{\alpha/2, k-1}$ 是自由度为 $k-1$ 的 t 分布上尾部累积分布为 $\alpha/2$ 的临界值。

3 McNemar 检验

- 针对二分类，主要思想是：若两学习器的性能相同，则 **A预测正确B预测错误数** 应等于 **B预测错误A预测正确数**，即 $e_{01}=e_{10}$ ，且 $|e_{01}-e_{10}|$ 服从 $N(1, e_{01}+e_{10})$ 分布。

$$\tau_{\chi^2} = \frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}}$$

此变量应该服从自由度为1的卡方分布，即标准正态分布变量的平方。

给定显著度 α ，当以上变量恒小于临界值时，不能拒绝假设，即认为两学习器的性能没有显著差别；否则拒绝假设，即认为两者性能有显著差别，且平均错误率较小的那个学习器性能较优。

4 Friedman 检验与 neny1 后续检验

2.5 偏差与方差

偏差与方差可以用来解释学习算法的泛化能力，即为什么学习器具有这样的性能。

- 偏差(bias)：期望输出与真实标记的差别，刻画算法的拟合能力，即准确性。
- 方差(var)：度量同样大小的训练集的变动所导致的学习性能的变化，即稳定性。
- 噪声： $\varepsilon^2 = E_D[(y_D - y)^2]$ ，表现当前任务下任何学习算法所能达到的泛化误差的下界，即误差再小也不可能比噪声小了。

泛化误差可以写作三者之和，说明了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。

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

- 其中偏差与方差是有冲突的，这称为偏差一方差窘境：

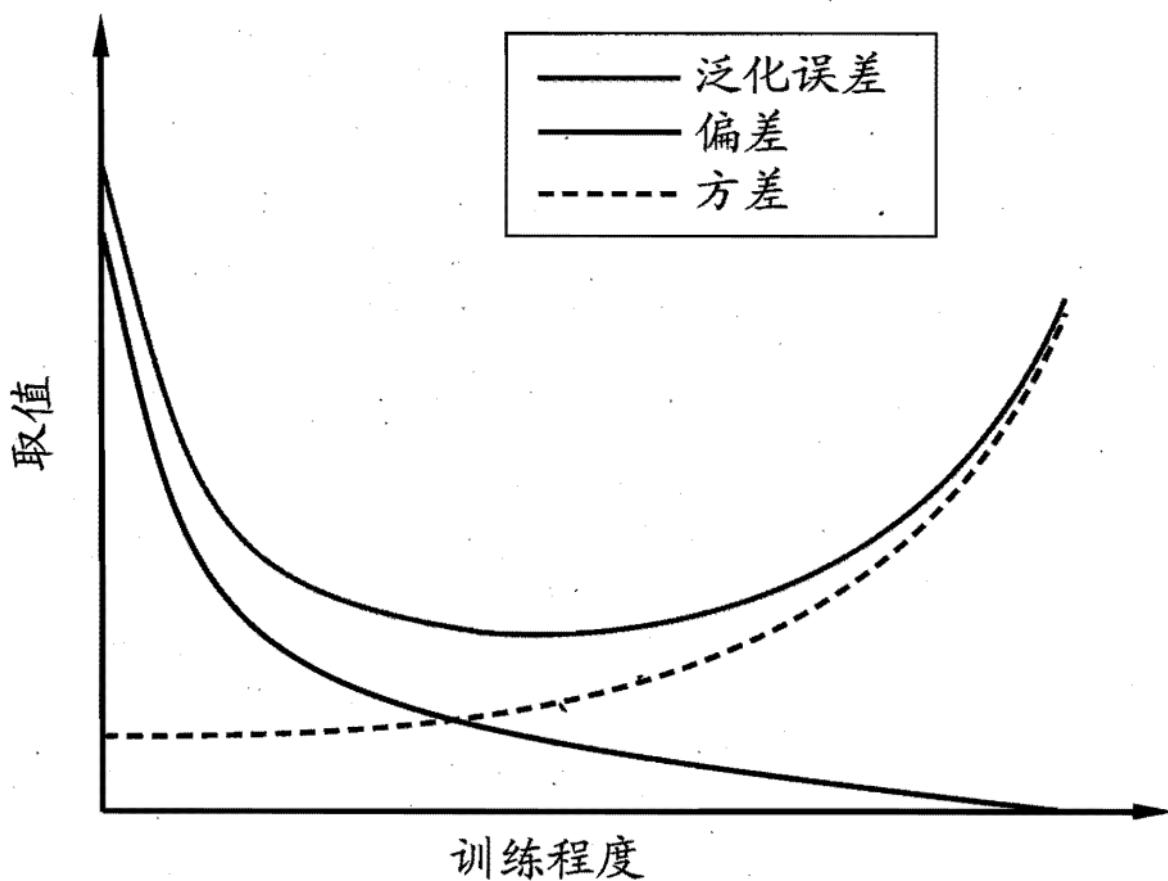


图 2.9 泛化误差与偏差、方差的关系示意图

前半段欠拟合，后半段过拟合

3. 线性模型 (linear model)

线性回归 (linear regression) 是通过对提取的特征属性进行线性组合来进行最终的目标值的预测的方法。其优点在于计算速度

快、解释性强。

但是实际上很多预测的属性是非线性的，这个时候就需要将线性模型推广到**广义线性模型**，将原来的线性模型作为自变量放入到一个非线性函数中实现对非线性规律的回归。（或者也可以理解为对将目标函数通过某种非线性规律映射回线性空间再使用简单的线性回归进行预测）

$$(y = \ln(ax_1 + bx_2 + cx_3 + \dots)) \text{ 或 } e^y = ax_1 + bx_2 + cx_3 + \dots$$

此外介绍了一个简单的**线性判别模型LDA模型**，通过将2维度样本的点投影到一条直线上，使得同类之间尽可能近，不同类别之间的距离尽可能更远。本质上是寻找一个低维空间，将高维空间的样本投影到这个低维的空间中后能够很好的分类。

而对于多类别分类，采取划归的思想，及通过不同的划分方式将多分类问题划分为多个二分类问题的组合。常见的划分方式有OvO, OvR, MvM三种方式。

对于样本分类不平衡，可以采用降采样、超采样、或者对判断的阈值进行再缩放解决。

3.1 基本形式

给定d个属性描述的样本 $x = (x_1, x_2 \dots x_d)$ ，线性模型(linear model)试图学得一个通过属性的线性组合来进行预测的函数：

$$f(\mathbf{x}) = w_1x_1 + w_2x_2 + \dots + w_dx_d + b,$$

一般用向量形式写成

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b,$$

$w = (w_1, w_2 \dots w_d)$, w和b确定后模型就确定

3.2 线性回归

- 假设样本x的属性只有一个，即 $D = (x_i, y_i)_{i=1}^m$

首先进行离散化、数字化：

- 连续属性一般可以直接使用
- 对于离散属性：
 - 若属性值之间存在“序关系”，则可以将其转化为连续值，例如：身高属性分为“高”“中等”“矮”，可转化为数值：{1, 0.5, 0}。
 - 若属性值之间不存在“序关系”，则通常将其转化为向量的形式，例如：性别属性分为“男”“女”，可转化为二维向量：{(1, 0), (0, 1)}。
- 如果输入样本只有一个属性：最小二乘法(least square method)

$$E(w, b) = \sum_{i=1}^m (y_i - wx_i - b)^2$$

对E求w, b的偏导，使其等于零，得到最小值解：

$$w = \frac{\sum_{i=1}^m y_i(x_i - \bar{x})}{\sum_{i=1}^m x_i^2 - \frac{1}{m} \left(\sum_{i=1}^m x_i\right)^2} \quad b = \frac{1}{m} \sum_{i=1}^m (y_i - wx_i)$$

- 如果输入样本有多个(d个)属性：多元线性回归分析((multivariate linear regression))
 - 首先写为矩阵表示：

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1d} & 1 \\ x_{21} & x_{22} & \dots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{md} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1^T & 1 \\ \mathbf{x}_2^T & 1 \\ \vdots & \vdots \\ \mathbf{x}_m^T & 1 \end{pmatrix}$$

and $\hat{\mathbf{w}} = (w; b)$

- 求E:

所以有：

$$\hat{\mathbf{w}}^* = \arg \min_{\hat{\mathbf{w}}} (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})^T (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})$$

同样思路，对其进行求导并使其等于0：

令 $E_{\hat{\mathbf{w}}} = (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})^T (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})$, 对 $\hat{\mathbf{w}}$ 求导得到

$$\frac{\partial E_{\hat{\mathbf{w}}}}{\partial \hat{\mathbf{w}}} = 2 \mathbf{X}^T (\mathbf{X}\hat{\mathbf{w}} - \mathbf{y})$$

- 如果 $X^T X$ 可逆，即为满秩或者正定

$$\hat{\mathbf{w}}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- 否则要对其进行正则化，这里不详细介绍

用线性去拟合非线性：

$$\ln y = \mathbf{w}^T \mathbf{x} + b$$

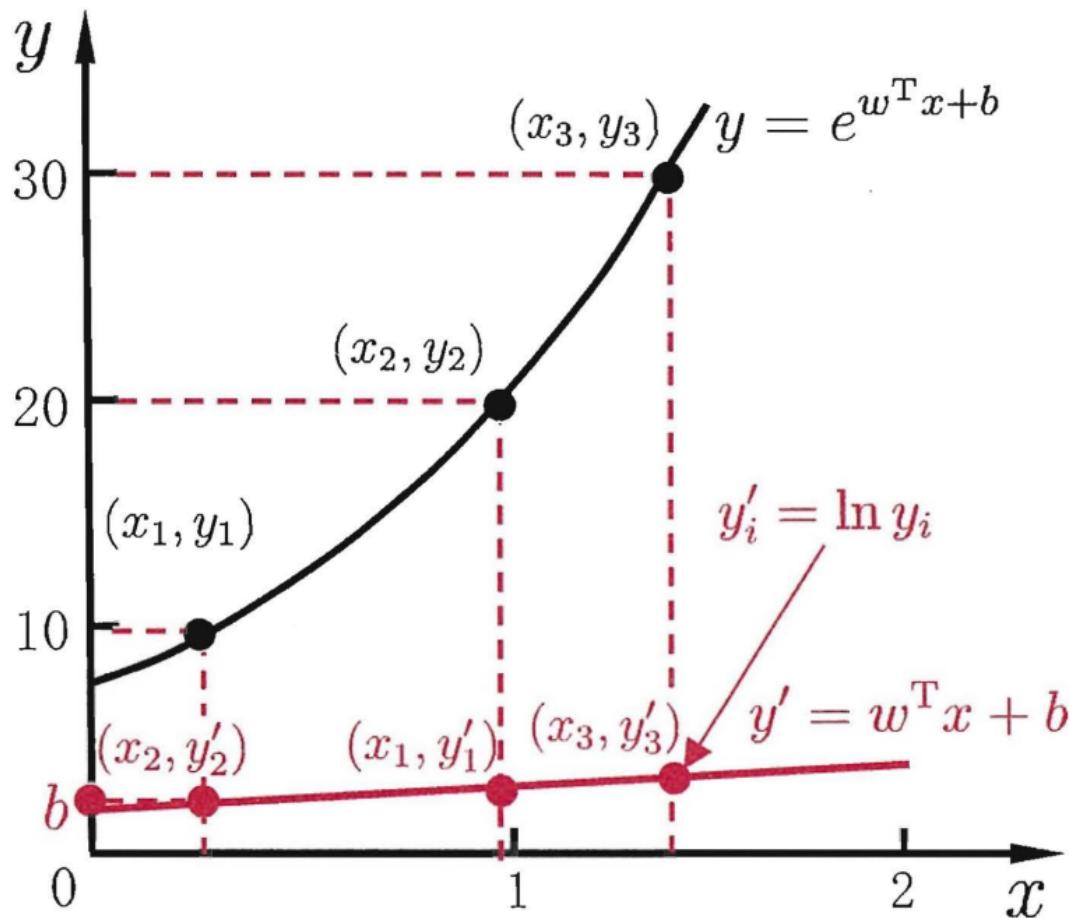


图 3.1 对数线性回归示意图

联系函数 $g(\cdot)$

更具一般性的，考虑一个单调可微的函数 $g(\cdot)$ （称为联系函数）：

$$y = g^{-1}(\mathbf{w}^T \mathbf{x} + b)$$

这样即通过 g 去将线性拟合与非线性函数联系起来，实现了对非线性函数的拟合。这样的模型叫做广义线性模型。

3.3 对数几率回归

- 主要是针对分类问题，使用对数几率函数作为联系函数，使得线性回归可以输出分类的预测值
- 以二分类为例：最理想的是单位阶跃函数：

$$y = \begin{cases} 0, & z < 0; \\ 0.5, & z = 0; \\ 1, & z > 0, \end{cases}$$

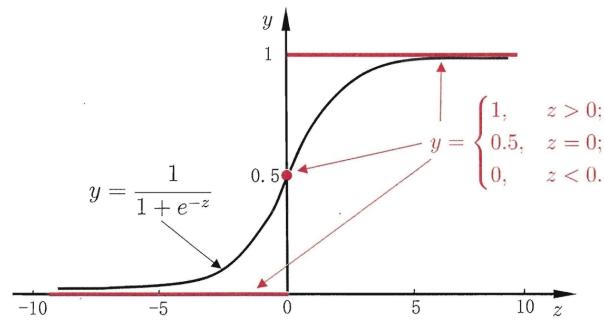


图 3.2 单位阶跃函数与对数几率函数

但是单位阶跃函数并不单调可微，所以使用对数几率函数替代(logistic function)

$$y = \frac{1}{1 + e^{-z}} \rightarrow y = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + b)}}$$

最终可写为： $\ln \frac{y}{1-y} = \mathbf{w}^T \mathbf{x} + b$

- 广义非线性模型可以使用极大似然估计来去确定w和b的值：
y视为样本为正例的可能性：

$$\ln \frac{p(y=1 | \mathbf{x})}{p(y=0 | \mathbf{x})} = \mathbf{w}^T \mathbf{x} + b .$$

显然有

$$p(y=1 | \mathbf{x}) = \frac{e^{\mathbf{w}^T \mathbf{x} + b}}{1 + e^{\mathbf{w}^T \mathbf{x} + b}} ,$$

$$p(y=0 | \mathbf{x}) = \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x} + b}} .$$

$$\ell(\mathbf{w}, b) = \sum_{i=1}^m \ln p(y_i | \mathbf{x}_i; \mathbf{w}, b) ,$$

对数变乘为加 → 最大化释然
即所有样本出现真实值的概率乘积最大

然后使用优化的方法去求使 $\ell(w, b)$ 值最大的 w, b 即可确定参数。

3.4 线性判别分析 (LDA)

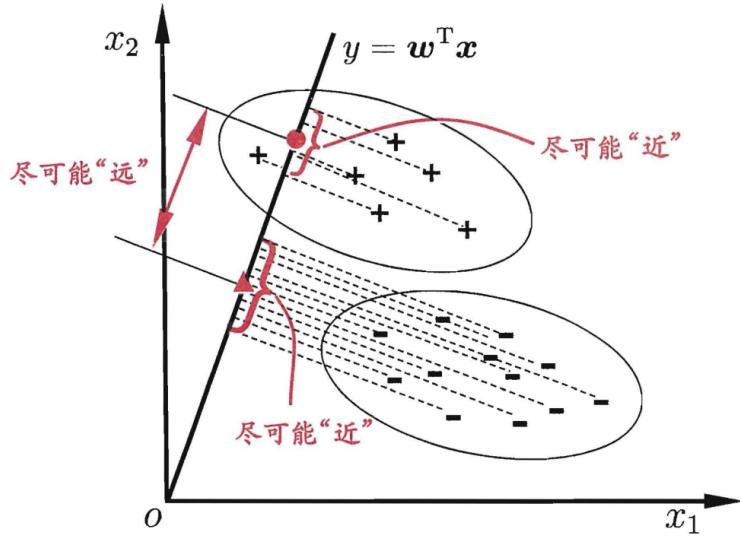


图 3.3 LDA 的二维示意图. “+”、“-” 分别代表正例和反例, 椭圆表示数据簇的外轮廓, 虚线表示投影, 红色实心圆和实心三角形分别表示两类样本投影后的中心点.

- LDA的思想是找到这样一条线，将样本垂直投影到线上之后，同类样本之间尽可能接近，不同类别的样本之间尽可能远离。

给定数据集 $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, $y_i \in \{0, 1\}$, 令 X_i 、 μ_i 、 Σ_i 分别表示第 $i \in \{0, 1\}$ 类示例的集合、均值向量、协方差矩阵. 若将数据投影到直线 w 上, 则两类样本的中心在直线上的投影分别为 $w^T \mu_0$ 和 $w^T \mu_1$; 若将所有样本点都投影到直线上, 则两类样本的协方差分别为 $w^T \Sigma_0 w$ 和 $w^T \Sigma_1 w$. 由于直线是一维空间, 因此 $w^T \mu_0$ 、 $w^T \mu_1$ 、 $w^T \Sigma_0 w$ 和 $w^T \Sigma_1 w$ 均为实数.

- 分别使用类内散度矩阵和类间散度矩阵去衡量：
 - **类内散度矩阵** : (within-class scatter matrix)

$$\begin{aligned} \mathbf{S}_w &= \Sigma_0 + \Sigma_1 \\ &= \sum_{\mathbf{x} \in X_0} (\mathbf{x} - \mu_0)(\mathbf{x} - \mu_0)^T + \sum_{\mathbf{x} \in X_1} (\mathbf{x} - \mu_1)(\mathbf{x} - \mu_1)^T \end{aligned}$$

- 类间散度矩阵：(between-class scatter matrix)

$$\mathbf{S}_b = (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^T \rightarrow \text{越大越好}$$

- 由此得到了LDA的优化目标： s_b 与 s_w 的广义瑞丽商(generalized Rayleigh quotient)

$$J = \frac{\mathbf{w}^T \mathbf{S}_b \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}}.$$

- 优化求解过程使用拉格朗日乘子法，待补充...

3.5 多分类学习策略

常见的解决的思路是将多分类任务拆分为多个二分类任务，而二分类的拆分方式有三种：

- OvO(one vs one)：
 - 给定数据集D，假定其中有N个真实类别，将这N个类别进行两两配对（一个正类/一个反类），从而产生 $N(N-1)/2$ 个二分类学习器。
 - 在测试阶段，将新样本放入所有的二分类学习器中测试，得出 $N(N-1)$ 个结果。
 - 最终通过投票将预测的最多的作为分类结果。**
- OvR(one vs rest)：
 - 给定数据集D，假定其中有N个真实类别，每次取出一个类作为正类，剩余的所有类别作为一个新的反类，从而产生N个二分类学习器。
 - 在测试阶段，得出N个结果。

3. 若仅有一个学习器预测为正类，则对应的类标作为最终分类结果；若多个分类器为正，则考虑其置信度，选择高的作为结果。

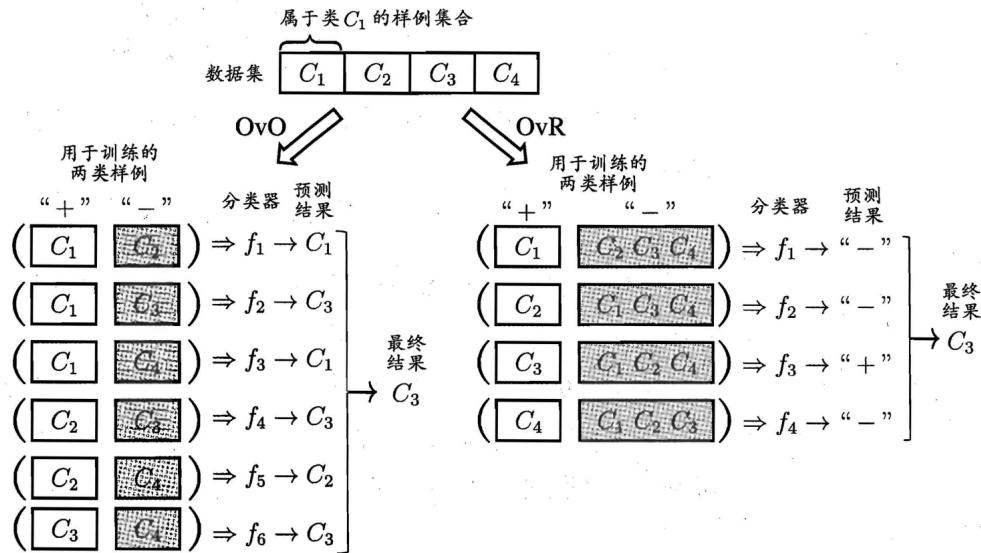


图 3.4 OvO 与 OvR 示意图

- MvM(many vs many) :

1. 给定数据集D，假定其中有N个真实类别，每次取若干个类作为正类，若干个类作为反类（通过ECOC码给出，编码），若进行了M次划分，则生成了M个二分类学习器
2. 在测试阶段（解码），得出M个结果组成一个新的码。
3. 最终通过计算海明/欧式距离选择距离最小的类别作为最终分类结果。

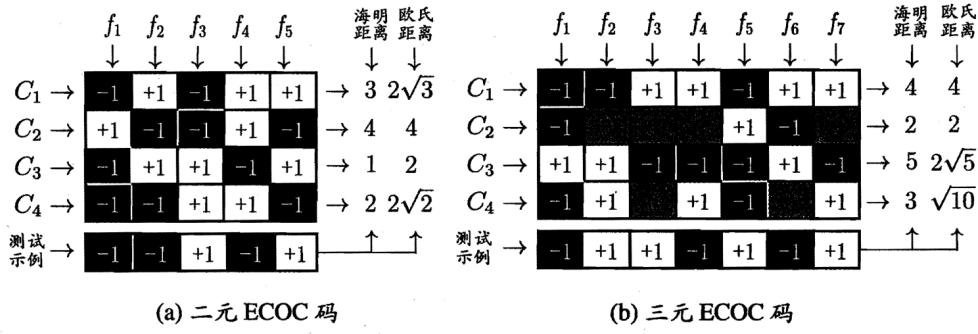


图 3.5 ECOC 编码示意图。“+1”、“-1”分别表示学习器 f_i 将该类样本作为正、反例；三元码中“0”表示 f_i 不使用该类样本

。 常用方式：ECOC：输出纠错码

1. 编码：对N个数据进行M次划分，每次把一部分划正，一部分划负，从而构成M个二分类器
 2. 解码：用这M个二分类classifier 分类得到的0, 1结果构成一个长为M的编码。每个分类 C_i 都可以得到一个对应的编码
 3. 比较：通过计算 C_i 中谁的编码与样本最接近来判断样本所属类别

ECOC编码对于分类的结果有一定的纠错和容忍能力，一般来说编码越长，纠错能力越强

3.6 类别不平衡问题

类别不平衡 (class-imbalance) 就是指分类问题中不同类别的训练样本相差悬殊的情况，例如正例有900个，而反例只有100个，这个时候我们就需要进行相应的处理来平衡这个问题。常见的做法有三种：

1. 在训练样本较多的类别中进行“欠采样” (**undersampling**) , 比如从正例中采出 100个 , 常见的算法有 : EasyEnsemble。
 2. 在训练样本较少的类别中进行“过采样” (**oversampling**) , 例如通过对反例中的数据进行插值 , 来产生额外的反例 , 常见的算法有SMOTE。
 3. 直接基于原数据集进行学习 , 对预测值进行“再缩放”处理。其中再缩放也是代价敏感学习的基础。

$$\frac{y'}{1-y'} = \frac{y}{1-y} \times \frac{m^-}{m^+} \rightarrow \frac{\text{cost}(+>-)}{\text{cost}(->+)} \quad \text{即代价敏感}$$

4. 决策树

4.1 基本流程

决策树特点：

- 每个非叶节点表示一个特征属性测试。
- 每个分支代表这个特征属性在某个值域上的输出。
- 每个叶子节点存放一个类别。
- 每个节点包含的样本集合通过属性测试被划分到子节点中，根节点包含样本全集。

构造：

输入：训练集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$;
 属性集 $A = \{a_1, a_2, \dots, a_d\}$.
 过程：函数 TreeGenerate(D, A)
 1: 生成结点 node;
 2: if D 中样本全属于同一类别 C then → **终止条件1** (最好的情形)
 3: 将 node 标记为 C 类叶结点; return
 4: end if
 5: if $A = \emptyset$ OR D 中样本在 A 上取值相同 then → **终止条件2** (属性用完或分不开情形，使用后验分布)
 6: 将 node 标记为叶结点，其类别标记为 D 中样本数最多的类; return
 7: end if
 8: 从 A 中选择最优划分属性 a_* ;
 9: for a_* 的每一个值 a_*^v do → **若为连续值属性，则只有两个分支(≤与≥)**
 10: 为 node 生成一个分支; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;
 11: if D_v 为空 then → **终止条件3** (分支为空，使用先验分布)
 12: 将分支结点标记为叶结点，其类别标记为 D 中样本最多的类; return
 13: else
 14: 以 TreeGenerate($D_v, A \setminus \{a_*\}$) 为分支结点
 15: end if → **若 a_* 为连续属性，则不用去除，寻找下一个最优划分点可继续作为子节点的划分属性**
 16: end for
 输出：以 node 为根结点的一棵决策树

有三种情形会导致递归返回：

1. 当前结点包含的样本全属于同一类别，这时直接将该节点标记为叶节点，并设为相应的类别；
2. 当前属性集为空，或是所有样本在所有属性上取值相同，无法划分，这时将该节点标记为叶节点，并将其类别设为该节点所含样本最多的类别；
3. 当前结点包含的样本集合为空，不能划分，这时也将该节点标记为叶节点，并将其类别设为父节点中所含样本最多的类别。

4.2 划分属性的选择

4.2.1 信息增益 (ID3)

- 决策树中最关键的一步就是8：如何从A中选择最优的划分属性？

一般来我们希望划分后分类的纯度越来越高，这个纯度用信息熵来衡量

“信息熵” (information entropy) 是度量样本集合纯度最常用的一种指标。假定当前样本集合 D 中第 k 类样本所占的比例为 p_k ($k = 1, 2, \dots, |\mathcal{Y}|$), 则 D 的信息熵定义为

$$\text{Ent}(D) = - \sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k . \quad (4.1)$$

$\text{Ent}(D)$ 的值越小, 则 D 的纯度越高。

- 通过对每个属性 a 计算依据 a 划分的信息增益, 选择划分属性:

$$\text{Gain}(D, a) = \text{Ent}(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Ent}(D^v) .$$

Gain 越大, 说明用 a 划分信息熵减小的越多, 所以信息增益越大越好

4.2.2 信息增益率(C4.5)

- ID3 缺点: 倾向于取值数目较多的属性

例如: 如果存在一个唯一标识, 这样样本集 D 将会被划分为 $|D|$ 个分支, 每个分支只有一个样本, 这样划分后的信息熵为零, 十分纯净, 但是对分类毫无用处

- 因此 C4.5 算法在在分母中增加一项与分类数量正相关的固有值 (**intrinst value**) $IV(a)$, 以克服这一问题:

$$\text{Gain_ratio}(D, a) = \frac{\text{Gain}(D, a)}{\text{IV}(a)} ,$$

$$\text{IV}(a) = - \sum_{v=1}^V \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$$

- 但是C4.5会出现偏向于取数目较少的属性的问题，所以一般结合使用：
 - 1. 先用ID3去选择信息增益高的
 - 2. 再用c4.5去选择信息增益率高的，去进行优化、过滤

4.2.3 基尼指数(CART算法)

- 直观上，Gini反应了从里面任意选两项，他们不相同的概率，即互异率。显然Gini率越小越好。

CART 决策树 [Breiman et al., 1984] 使用“基尼指数”(Gini index)来选择划分属性。采用与式(4.1)相同的符号，数据集 D 的纯度可用基尼值来度量：

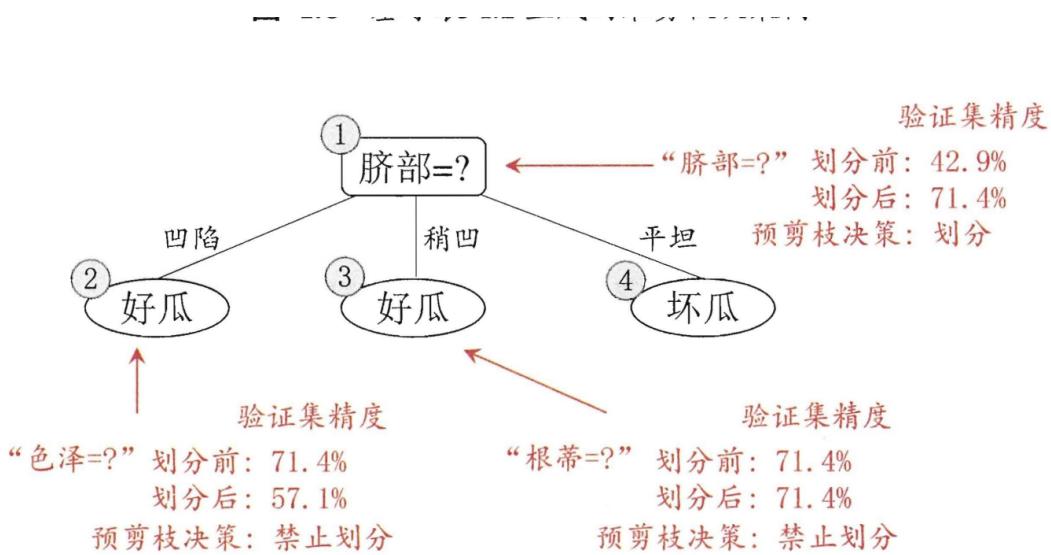
$$\begin{aligned} \text{Gini}(D) &= \sum_{k=1}^{|\mathcal{Y}|} \sum_{k' \neq k} p_k p_{k'} \\ &= 1 - \sum_{k=1}^{|\mathcal{Y}|} p_k^2 . \end{aligned} \quad (4.5)$$

4.3 剪枝

剪枝 (pruning) 则是决策树算法对付过拟合的主要手段，剪枝的策略有两种如下：

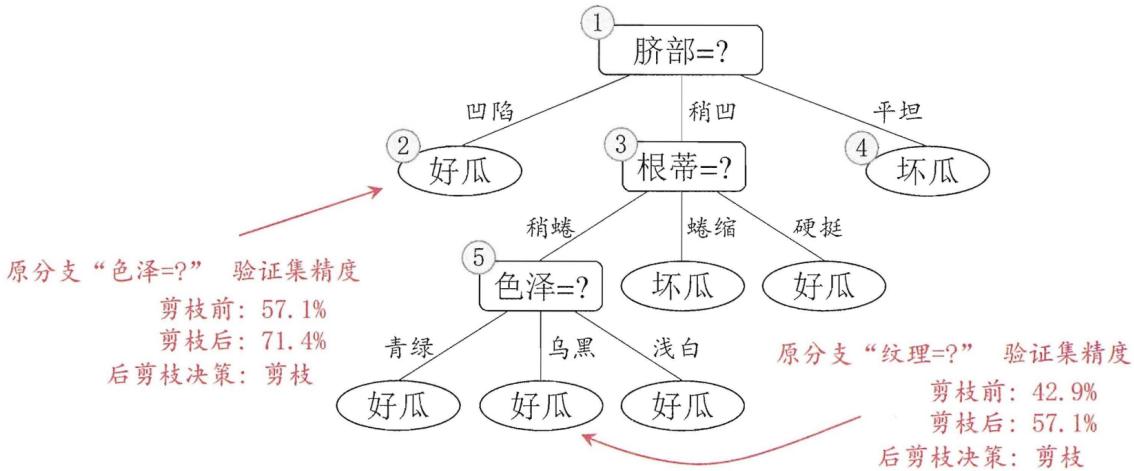
- 预剪枝 (prepruning) : 在构造的过程中先评估，再考虑是否分支。

预剪枝表示在构造树的过程中，对一个节点考虑是否分支时，首先计算决策树不分支时在测试集上的性能，再计算分支之后的性能，若分支对性能没有提升，则选择不分支（即剪枝）



- 后剪枝 (post-pruning) : 在构造好一颗完整的决策树后，自底向上，评估分支的必要性。

后剪枝则表示在构造好一颗完整的决策树后，从最下面的节点开始，考虑该节点分支对模型的性能是否有提升，若无则剪枝，即将该节点标记为叶子节点，类别标记为其包含样本最多的类别。



4.4 连续与缺失的处理

4.4.1 连续变量处理：离散化

- 二分法：

常用的方法为二分法，基本思想为：给定样本集D与连续属性 α ，二分法试图找到一个划分点 t 将样本集D在属性 α 上分为 $\leq t$ 与 $> t$ 。

- 划分点选择：

1. 首先将 α 的所有取值按升序排列，所有相邻属性的均值作为候选划分点（ $n-1$ 个， n 为 α 所有的取值数目）。
2. 计算每一个划分点划分集合D（即划分为两个分支）后的信息增益。

$$\begin{aligned} \text{Gain}(D, a) &= \max_{t \in T_a} \text{Gain}(D, a, t) \\ &= \max_{t \in T_a} \text{Ent}(D) - \sum_{\lambda \in \{-, +\}} \frac{|D_t^\lambda|}{|D|} \text{Ent}(D_t^\lambda), \end{aligned}$$

3. 选择最大信息增益的划分点作为最优划分点。

4.4.2 缺失处理

- 样本属性缺失带来的两个主要问题：
 - 1如何选择划分属性
 - 2给定划分属性，若某样本在该属性上缺失值，如何划分到具体的分支上
- 1如何选择划分属性：

在样本集D中选取在属性 α 上没有缺失值的样本子集，计算在该样本子集上的信息增益，最终的信息增益等于该样本子集划分后信息增益乘以样本子集占样本集的比重

简单的说就是计算信息熵时候，只用无缺失的样本，但是最终要在信息熵上乘一个这些样本所占的比例表示权值

$$\text{Gain}(D, a) = \rho \times \text{Gain}(\tilde{D}, a) \rightarrow \text{表示在属性}\alpha\text{上无缺失的样本子集}$$

无缺失样本子集
所占比重

$$= \rho \times \left(\text{Ent}(\tilde{D}) - \sum_{v=1}^V \tilde{r}_v \text{Ent}(\tilde{D}^v) \right)$$

- 2给定划分属性，若某样本在该属性上缺失值，如何划分到具体的分支上：

定义：

$$\rho = \frac{\sum_{x \in \tilde{D}} w_x}{\sum_{x \in D} w_x}, \quad \text{样本子集所占比例}$$

$$\tilde{p}_k = \frac{\sum_{x \in \tilde{D}_k} w_x}{\sum_{x \in \tilde{D}} w_x} \quad (1 \leq k \leq |\mathcal{Y}|), \quad \text{样本子集每个类别的比例}$$

$$\tilde{r}_v = \frac{\sum_{x \in \tilde{D}^v} w_x}{\sum_{x \in \tilde{D}} w_x} \quad (1 \leq v \leq V). \quad \text{每个分支所含样本比例}$$

方法：根据其他无缺样本所占的比例，将这些有缺样本加权后划分到每一个类中。

对问题(2), 若样本 \mathbf{x} 在划分属性 a 上的取值已知, 则将 \mathbf{x} 划入与其取值对应的子结点, 且样本权值在子结点中保持为 $w_{\mathbf{x}}$. 若样本 \mathbf{x} 在划分属性 a 上的取值未知, 则将 \mathbf{x} 同时划入所有子结点, 且样本权值在与属性值 a^v 对应的子结点中调整为 $\tilde{r}_v \cdot w_{\mathbf{x}}$; 直观地看, 这就是让同一个样本以不同的概率划入到不同的子结点中去.

4.5 多变量决策树

- 每个决策节点不再是通过单属性进行划分, 而是有可能使用多个属性变量决策划分
- 优点: 有些复杂情况使用单变量决策过程往往很复杂(左图), 使用多变量可能是更好的选择

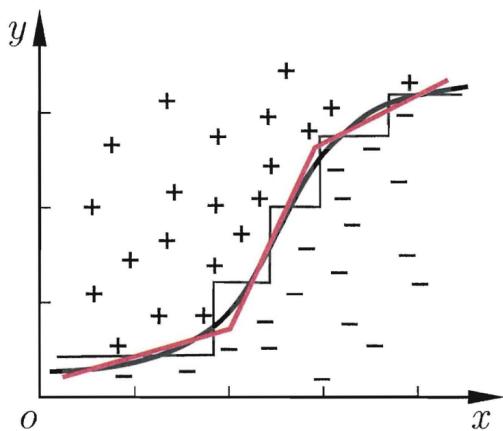


图 4.12 决策树对复杂分类边界的分段近似

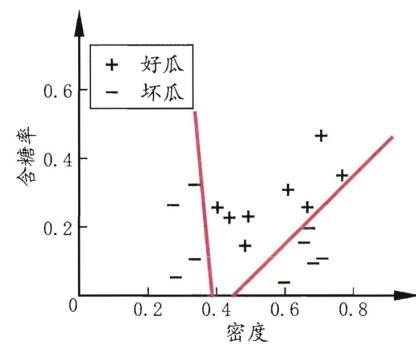


图 4.14 图 4.13 多变量决策树对应的分类边界

5. 神经网络

6. 支持向量机

7. 贝叶斯分类器

7.1 贝叶斯定理

“Posterior”

$$P(\text{H|E}) = \frac{P(\text{H})P(\text{E|H})}{P(\text{E})} = \frac{P(\text{H})P(\text{E|H})}{P(\text{H})P(\text{E|H}) + P(\neg\text{H})P(\text{E|\neg H})}$$

“Prior” $\rightarrow P(\text{H}) = 1/21$

“Likelihood”

$$P(\text{E|H}) = 0.4 \quad \left\{ \begin{array}{l} \text{A grid of green dots representing data points.} \\ \text{A vertical line on the left side of the grid.} \end{array} \right\} \Rightarrow P(\text{E|\neg H}) = 0.1$$



(from 3Blue1Brown)

贝叶斯定理的公式如下所示：

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

其中， $P(A)$ 和 $P(B)$ 分别表示事件 A 和事件 B 的先验概率， $P(A|B)$ 表示在事件 B 发生的条件下，事件 A 的后验概率， $P(B|A)$ 表示在事件 A 发生的条件下，事件 B 的条件概率。

贝叶斯定理中的后验概率更像是一种逆概率，是根据已知新信息后对条件的重新评估，也可以理解为一种根据新信息对概率的修正。

- 3Blue1Brown的视频解释的很清楚

| 【官方双语】贝叶斯定理，使概率论直觉化 哔哩哔哩 bilibili

- **先验概率**

先验概率是指在考虑任何证据之前，我们对事件发生的概率的初始信念。在贝叶斯定理中，它表示为 $P(A)$ 。例如，在一个二分类问题中，如果我们知道有 70% 的患者患有某种疾病，那么我们可以说该疾病的先验概率为 0.7。

- **后验概率**

后验概率是指在考虑了某些证据或信息之后，我们对事件发生的概率的信念。在贝叶斯定理中，它表示为 $P(A|B)$ 。例如，在一个二分类问题中，如果我们知道一个患者的某些症状，那么我们可以根据这些症状计算患者患上该疾病的后验概率。

- **全概率**

全概率是指一个事件的概率可以被分解为多个条件下的概率之和。在贝叶斯定理中，全概率表示为 $P(B)$ ，可以通过以下公式计算：

$$P(B) = \sum_{i=1}^n P(B|A_i)P(A_i)$$

其中， A_i 是样本空间的一组互斥事件，n 是 A 的数量。在分类问题中，全概率公式可以用来计算一个样本属于每个类别的概率，从而计算后验概率。

7.2 贝叶斯决策论

对分类任务来说，**贝叶斯决策理论是在所有相关概率都已知的理想情况下，借助 Bayesian 公式的思想来实施决策。**

简单的说，就是对于每一个样本，选择能够使得后验概率 $P(c|x)$ 最大的类别标记。

- 期望损失(expected loss)：

$$E(L(\alpha|x)) = \sum_{i=1}^n L(\alpha_i|x)p(\omega_i|x)$$

其中， $L(\alpha_i|x)$ 是在给定样本 x 的条件下选择操作 α_i 造成的损失， $P(\omega_i|x)$ 是在给定样本 x 的条件下类别 ω_i 的后验概率。

(一种简单的 $L(\alpha_i|x)$ 取法仅当 w_i 为x的类别时取0，其他均取1。显然选择正确的概率越大，损失函数E的值越小)

贝叶斯决策的目标就是选择一个判定准则 h 使得总体的期望损失最小：

贝叶斯判顶准则：为最小化总体风险，只需要在每个样本上选择那个能使得条件风险最小的类别标记。

此时称 h 为贝叶斯最优分类器， $R(h)$ 为贝叶斯风险， $1 - R(h)$ 反映了分类器可以达到的最好性能。

判别式模型、生成式模型

使用贝叶斯模型来最小化决策风险，首先要获得后验概率，但是实际任务中后验概率比较难获得，从这个角度来看机器学习的任务是基于训练集尽可能准确的估计出后验概率 $P(c|x)$ 。

- 判别式模型：一种方式是直接对 $P(c|x)$ 进行建模，直接得到概率模型。如决策树、BP网络、支持向量机都属于判别式模型的范畴

- 生成式模型：

基于贝叶斯模型：

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

- 对于给定的样本 x , $p(x)$ 与类别标记无关，所以估计 $P(c|x)$ 的问题就转化为如果基于已知的数据集D对 $P(c)$ 和 $P(x|c)$ 进行估计。
- **$P(c)$ 是先验概率**：表达各个类所占样本的比例。根据大数定律，当训练集足够且独立同分布时，可以直接使用样本出现频率来近似估计。
- **$P(x|c)$ 是类条件概率**，它涉及到关于x的所有的属性的联合概率，直接根据样本估计会出现很多问题。（例如样本总数都没有样本空间的取值可能多）

7.3 极大似然估计

- 统计学中的频率派和贝叶斯派

在统计学中，频率派和贝叶斯派是两种不同的哲学观点。

- 频率派

频率派认为参数虽然是未知的，但是频率却是客观存在的固定值。他们使用大量的数据来推断未知参数，并使用置信区间来描述这些参数的不确定性。

- 贝叶斯派

贝叶斯派认为，概率是表示我们对事件的不确定性的度量。参数是未观察到的随机变量，其本身也有分布，因此可以假定参数服从一个先验分布，然后基于观测到的数据来计算参数的后验分布。他们使用贝叶斯定理将新信息合并到已知信息中，以更新我们对参数的估计值和不确定性。

在机器学习中，频率派和贝叶斯派都有自己的应用。例如，频率派经常用于基于极大似然估计的模型拟合，而贝叶斯派则经常用于贝叶斯模型拟合和贝叶斯优化等问题。

- D_c 表示数据集 D 中 c 样本集合，假设独立同分布，则参数 θ_c 对于 D_c 的似然：

$$P(D_c|\theta_c) = \prod_{x \in D_c} P(x|\theta_c)$$

极大似然估计就是寻找一个参数 θ_c 使得 $P(D_c|\theta_c)$ 最大。即当时事件已发送了，那么去找一个最有可能导致现在已发生事件的概率模型的参数。

为了避免连乘，一般对 $P(D_c|\theta_c)$ 取对数

即 $\hat{\theta}_c = \operatorname{argmax}_{\theta_c} \operatorname{Log}(P(D_c|\theta_c)) = \operatorname{argmax} LL(\theta_c)$

- 需要注意的是极大似然的准确性严重依赖于假设的概率分布是否符合潜在的真实数据分布，往往需要在一定程度上利用关于任务本身的相关知识经验，不能凭空猜测概率分布形式。

7.4 朴素贝叶斯分类器

- 基于贝叶斯公式进行后验概率分布的估计的主要困难在于类条件概率 $P(x|c)$ 是 x 所有的属性的联合分布，这个分布很有可能很复杂。而朴素贝叶斯分布采用了一个很简单的假设解决这个问题：

属性条件独立性假设：对已知类别，假设所有的属性相互独立。

换而言之，每个属性对于分类的结果的影响是独立的。

- 基于这个假设：

$$P(c|x) = \frac{P(c)P(x|c)}{P(x)} = \frac{P(c)}{P(x)} \prod_{i=1}^d P(x_i|c)$$

即 $P(c|x)$ 等于每个属性 x_i 对于的 $P(x_i|c)$ 的连乘。

而 $P(x)$ 对于每个类别是一样的，所以此时**朴素贝叶斯判定准则**可以写作：

$$h_{nb}(x) = \operatorname{argmax}_c P(c) \prod_{i=1}^d P(x_i|c)$$

- 所以朴素贝叶斯的训练过程其实就是根据训练集D来估计类先验概率 $P(c)$ ，然后为每个属性估计一个条件概率 $P(x_i|c)$ 。这样即可根据上式进行类别判定。

$P(c)$ 与 $P(x_i|c)$ 的估计

- $P(c)$ 估计： $P(c) = \frac{|D_c|}{|D|}$

- $P(x_i|c)$ 的估计：

- 离散属性：

$$P(x_i|c) = \frac{|D_{c,i}|}{|D_c|}$$

- 连续属性：

先估计 $\mu_{c,i}$, $\sigma_{c,i}$ (分别是是c类样本在第i个属性上的取值的均值和方差)：

$$\text{然后带入正态分布} : p(x_i|c) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

拉普拉斯修正

为了避免某项属性在训练集没有出现，从而导致对于概率为0，从而连乘后抹去其他属性信息，估算概率时使用拉普拉斯平滑 (Laplacian correction)

- 具体来说，计算概率时在分子+1，分母加此类别数N or 第i个属性的种类数

$$\hat{P}(c) = \frac{|D_c| + 1}{|D| + N}$$

$$\hat{P}(x_i|c) = \frac{|D_{c,i}| + 1}{|D_c| + N_i}$$

Homework4

Homework4 朴素贝叶斯分类

8 集成学习

AdaBoost

9 聚类

原型聚类-学习向量量化

算法伪代码：

输入：样本集 $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$;
原型向量个数 q , 各原型向量预设的类别标记 $\{t_1, t_2, \dots, t_q\}$;
学习率 $\eta \in (0, 1)$.

过程：

- 1: 初始化一组原型向量 $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_q\}$
- 2: **repeat**
- 3: 从样本集 D 随机选取样本 (\mathbf{x}_j, y_j) ;
- 4: 计算样本 \mathbf{x}_j 与 \mathbf{p}_i ($1 \leq i \leq q$) 的距离: $d_{ji} = \|\mathbf{x}_j - \mathbf{p}_i\|_2$;
- 5: 找出与 \mathbf{x}_j 距离最近的原型向量; $i^* = \arg \min_{i \in \{1, 2, \dots, q\}} d_{ji}$;
- 6: **if** $y_j = t_{i^*}$ **then**
- 7: $\mathbf{p}' = \mathbf{p}_{i^*} + \eta \cdot (\mathbf{x}_j - \mathbf{p}_{i^*})$
- 8: **else**
- 9: $\mathbf{p}' = \mathbf{p}_{i^*} - \eta \cdot (\mathbf{x}_j - \mathbf{p}_{i^*})$
- 10: **end if**
- 11: 将原型向量 \mathbf{p}_{i^*} 更新为 \mathbf{p}'
- 12: **until** 满足停止条件
- 13: **return** 当前原型向量

输出：原型向量 $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_q\}$

10 降维与度量学习

10.1 K 临近算法

简单的lazy learning 算法, 不需要进行显示的训练：

给定测试样本，基于某种距离度量找出训练集中与其最靠近的k训练样本，然后基于这k个"邻居"的信息来进行预测.

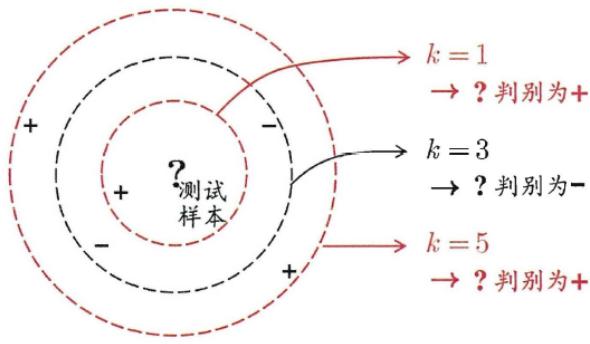


图 10.1 k 近邻分类器示意图. 虚线显示出等距线; 测试样本在 $k = 1$ 或 $k = 5$ 时被判别为正例, $k = 3$ 时被判别为反例.

- 一般对于分类任务可以采取投票，回归任务则可以取平均或者加权平均
- 另外：最近邻分类器虽简单，但它的泛化错误率不超过贝叶斯最优分类器的错误率的两倍。

10.2 低维嵌入

10.3 主成分分析

10.4 核化线性降维(Kernlized PCA、非线性降维)

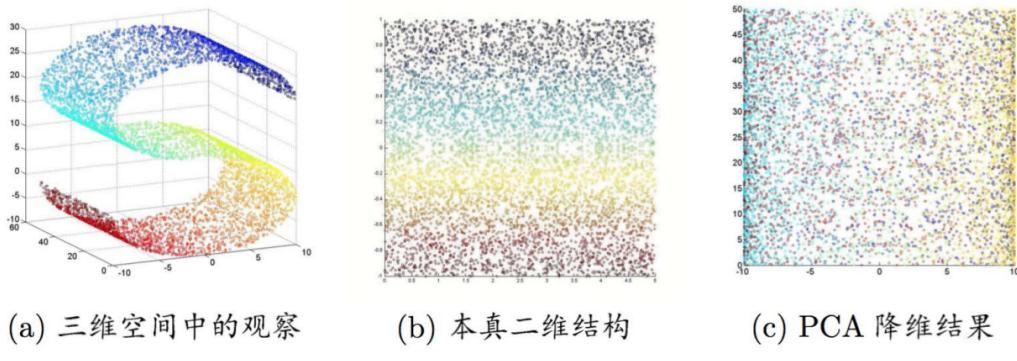


图 10.6 三维空间中观察到的 3000 个样本点, 是从本真二维空间中矩形区域采样后以 S 形曲面嵌入, 此情形下线性降维会丢失低维结构. 图中数据点的染色显示出低维空间的结构.

10.5 流行学习

10.6 度量学习

基础知识

矩阵分解