基本概念

1. 什么是机器学习
   机器学习就是一门研究如何通过计算的手段，利用经验来改善系统自身的性能，达到进行预测，分类等目的的学科。
   赋予机器学习的能力，通过学习来解决问题，而不是通过特定的编程来解决特定的问题。
   > [Mitchell , 1997J 给出了一个更形式化的定义假,设用 P 来评估计算机程序,在某任务类 T 上的性能，若一个程序通过利用经验E 在 T 中任务丰获得了性能改善，则我们就说关于T 和 P ， 该程序对 E 进行了学习

> 机器学习正是这样一门学科，它致力于研究如何通过计算的手段，利用经验来玫善系统自身的性能。

2. 机器学习能干什么
> 有了学习算法，我们把经验数据提供给它，它就能基于这些数据产生模型;在面对新的情况时，模型会给我们提供相应的判断

3. 什么是模型/假设？
模型是对某个对象特征的模拟和抽象。
简单的来说，就是学习算法经过对数据的学习之后得到的结果。模型是关于数据的某种潜在规律，因此也称为假设。
这条规律自身，也被称为”真相”或”真实”
所谓的”假设”，我的理解是，某个对象满足了这条规律，我们就有理由相信它就是这一类事物，或者说我们可以假设它就是这一类事物，因此称为”假设”
> 本书用"模型"泛指从数据中学得的结果有文献用"模型"指全局性结果(例如一棵决策树)，而用"模式"指局部性结呆(例如 A条规则).

4. 什么是记录/示例/样本/特征向量？
就是关于一个事件或对象的描述，反映事件或对象在某方面的表现或者性质的事项。
一条记录包括若干个属性以及对应的属性值。
从数学表示上更容易理解”特征向量”的含义，我们以一个属性作为一个维度，那么一个样本的属性就可以表示成(x1,x2,..,xn)这样的形式，为n个属性张成的空间中的一个点，或者说一个向量

5. 什么是数据集？
就是记录的集合。

6. 什么是属性空间/样本空间/输入空间？
属性张成的空间，就是属性空间。
以一个属性作为空间的一维变量，那么数据集中所有的属性组成的空间，就是属性空间。
那么数据集中的一个记录，就可以描述为空间的一个点。因此，一条记录也可以描述为“特征向量”

7. 什么是维数？
也就是属性空间的维数，简单的来说，就是数据集中属性种类的数目。一条记录有d个属性，那么d就是这条记录的维数。

8. 什么是学习/训练？
从数据中学得模型的过程称为"学习" (learning)或"训练" (training),这个过程通过执行某个学习算法来完成。
学习过程是为了找出或者逼近真相。

9. 什么是标记信息，样例，标记空间/输出空间？
样本“结果”信息称为标记信息。
有标记信息的示例成为样例。
标记信息的集合为标记空间。

> 比如给定一个瓜的特征，这个是好瓜。
> 那么”好瓜”就是标记信息。
> “一个含有某个特征的瓜是个好瓜”这是一个样例。
> 如果 给定某个瓜的特征 是输入，那么这个瓜是不是好瓜（标记信息）就是输出。
> 所有标记信息的集合就是输出空间。

10. 什么是分类，回归，聚类？
欲预测的值是离散的学习任务称为“分类”，连续的称为“回归”
通过学习，将训练集中的数据分成若干个组，称为“聚类”
> 根据训练数据是否拥有标记信息，学习任务可大致划分为两大类"监督学习" (supervised learning) 和"无监督学习" (unsupervised learning) ，分类和回归是前者的代表，而聚类则是后者的代表.

分类和回归的区分主要是看目标是离散的还是连续的。
比如，预测明天的温度变化是回归任务，预测明天是晴还是阴，这是分类任务。
分类是根据判断的结果（标记信息）进行区分。
聚类则是根据自身的特性进行区分。
或者说 分类是人为设置标记的结果 ，而聚类则是数据自身特性的结果。

11. 什么是 泛化能力？
> 学得模型适用于新样本的能力，称为"泛化"(generalization)能力

12. 模型和算法的关系
> 从数据中学得模型的过程称为"学习" (learning)或"训练" (training),这个过程通过执行某个学习算法来完成.

13. 假设空间，归纳偏好
能与训练集一致的假设的集合为假设空间。
也就是有多个假设与训练集一致，那么最终采用哪个假设（模型），就是归纳偏好。
> 有没有一般性的原则来引导算法确立"正确的"偏好呢? "奥卡姆剃刀" (Occam's razor)是一种常用的、自然科学研究中最基本的原则，即"若有多个假设与观察一致，则选最简单的那个"（但并不绝对）

14. 混淆矩阵

混淆矩阵是数据科学、数据分析和机器学习中总结分类模型预测结果的情形分析表，以矩阵形式将数据集中的记录按照真实的类别与分类模型作出的分类判断两个标准进行汇总。

分类结果的混淆矩阵：

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)

15. 指示函数

$\amalg(x)$，指示函数,如果x为真，那么值为1，否则值为0

$$\amalg(x)=\begin{cases}1 & ,x=true\\0 & ,x=false\end{cases}$$

NFL 定理

"没有免费的午餐"定理 (No Free Lunch Theorem，简称 NFL定理)：在所有问题同等重要的情况下，任意两个机器学习算法期望性能相同。
脱离具体问题来谈哪个机器学习算法更好是毫无意义的。
同时，不同的问题，在不同的情况，评判标准也可能有所不同，因此算法的好坏也不是绝对的。

在没有实际背景下，没有一种算法比随机胡猜的效果好

模型评估和选择

过拟合和欠拟合

过拟合：把训练样本的自身的特点当作所有潜在样本都会具有的一般特性
欠拟合：指对训练样本的一般特性尚未学习好。
比如：训练样本都是有锯齿的树叶，从而分类没有锯齿的都不是树叶，这是过拟合。欠拟合则是把所有绿色的东西都分类为叶子。

疑问：为什么过拟合无法避免？

疑问：为什么NP难问题是无解的？
有些问题需要考虑全体样本的情况，也就是需要考虑全集，那么这个全集可能是无穷大的，所以这个问题是无解的，我们只能去近似它而不能在多项式时间内解决它。

评估方法

基本思路：我们无法直接获得泛化误差，那么就只能通过测试集上的'误差测试'来近似泛化误差。

同时，这个测试集是从样本真实分布中独立分布采样而得。

测试集应该尽量不在训练集中出现，未在训练过程中使用过。

留出法

留出法：把数据集划分为两个互斥的集合，一个作为训练集，一个作为测试集。

为保证数据划分分布的一致性，应该采用分层采样。

不足：

1. 使用留出法，不同的划分，将导致不同的训练集，所以评估的结果也会有所差别，故应该采取多次划分，取平均值的方法。
2. 我们希望评估的是用D训练出的模型的性能，那么评估准确度
   有以下矛盾
   1. 如果S划分多，那么用于评估的测试集T就会少，那么测试结果就可能不准确
   2. 如果S划分少，那么S和D的差别较大，从而降低了结果的保真性。

一般是 $\frac{2}{3}$~$\frac{4}{5}$ 的样本用于训练

交叉验证法

交叉验证法：

1. 数据集分层采样，划分长k个互斥的子集
2. 每k-1个子集用作训练集，剩下一个用作测试集
3. 多次重复步骤1-2，取平均值

注意：如果是k个子集，那么一次交叉验证法就会得到k个模型，当数据集比较大的时候，训练k个模型的开销也会很大。更不用说还要进行多次划分。

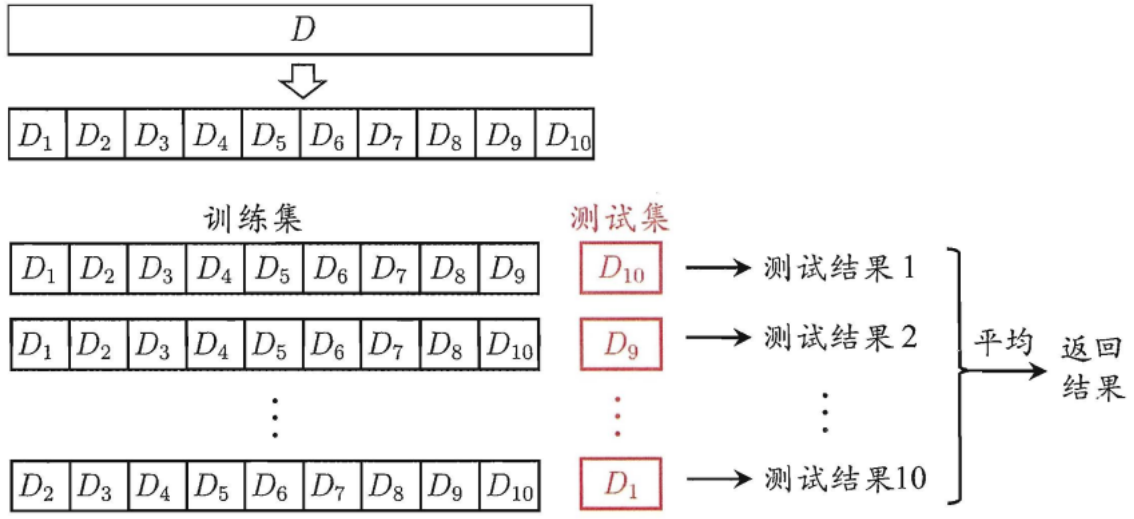


图 2.2 10 折交叉验证示意图

留一法

留一法：是交叉验证法的一种特殊情况，m个样本的数据集，划分m个子集，也就是只有一个样本用作测试。

不足：
当数据集比较大的时候，训练m个模型的计算开销可能是难以忍受的，更不用说还要进行多次划分。

自助法

自助法：从数据集D中，有放回地随机采样m次作为训练集D',以D'作为 训练集，D/D'作为测试集。

$lim_{m\to\infty}(1-\frac{1}{m})^m=\frac{1}{e}\approx 0.368$

也就是初始数据集中，有36.8%的样本从未在训练集中出现过。

自助法在数据集较小，难以有效划分训练/测试集时很有用

不足：自助法产生的数据集改变了初始数据集的分布，会引入误差。

疑问：怎么样才算数据集小?
这个看具体情况。

调参

对算法参数进行设定，参数调节。调参需要注意，常用的做法是对每个参数选定一个范围与变化步长，在计算开销与性能估计之间折中。

调参方式：产生多个模型后，基于某种评估方法进行选择。

参数的类型:

1. 算法的参数
   一般由人工设定多个候选值后产生模型，数目常在10以内
2. 模型的参数
   通过学习来产生多个候选模型。大型模型甚至数目可达上百亿。

最终模型

对于m个样本的数据集D,训练出模型并选择完成后，学习算法与参数配置都已选定，还需要用数据集D重新训练，这个模型才是我们的最终模型。

验证集 和 测试集

模型评估和选择中用于测试的数据集，为验证集。
验证集用于模型选择与调参。
测试集用于估计模型在使用时的泛化能力

性能度量

前面提到的都是模型的评估方法，也就是 “怎么做”，而性能度量则是”怎么评”。后面还有怎么比较的问题，并不能直接比较大小。
性能度量：衡量模型泛化能力的评价标准。

均方误差

均方差是"误差"的平方的期望值。
误差就是估计值与被估计量的差。
均方误差：回归任务中常用的性能度量。
f：学习器，已经训练好的模型
D：测试集
m：测试集的示例数目
$f(x_i)$:模型对x_i预测的结果
y_i:示例x_i的真实标记
$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}(f(x_i)-y_i)^2$

$\mathcal{D}$:数据分布
$p(x)$:概率密度函数
$E(f;\mathcal{D})=\int_{x\sim\mathcal{D}}(f(x)-y)^2p(x)dx$

错误率与精度

错误率：模型分类错误的样本数占总样本数的比例。
精度：模型分类正确的样本数占总样本数的比例。

$\mathrm{II}(x)$，指示函数,如果x为真，那么值为1，否则值为0
$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\mathrm{II}(f(x_i)\neq y_i)$

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

$E(f;\mathcal{D})=\int_{x\sim\mathcal{D}}\mathrm{II}(f(x)\neq y)p(x)dx$

$acc(f;\mathcal{D}) = \int_{x\sim\mathcal{D}} \mathbb{I}\left(f(x)=y\right)p(x)dx$

查准率，查全率

FP(true positive):真正例
FP(false positive):假正例
TN(true negative):真反例
FN(true negati ve):假反例

查准率R:所有预测为正例的结果中，真正例占的比重。
比如:挑出的西瓜中有多少比例是好瓜
$P = \frac{TF}{TP+FP}$

查全率R:所有真实情况为正例中，被预测正确（真正例）占的比重。
比如:所有好瓜中有多少比例被挑了出来
$R = \frac{TP}{TP+FN}$

注意：这里挑出的西瓜的多少，指的应该是预测为正例（好瓜）的数目而不是被预测的瓜的总数

查全率和查准率的矛盾

查全率和查准率是一对相互矛盾的度量，查准率高时，查全率往往偏低，查全率高时，查准率往往偏低。

比如:要尽可能多地挑出好瓜（提高查全率），那么尽可能将更多的瓜挑上（比如全都挑走），那么自然好瓜占的比例（查准率）就低了

很多学习器是为样本产生一个实值或概率预测，然后将这个预测值与一个分类阈值进行比较，若大于阈值则为正类，否则为反类。

希望查准率高，则阈值设置大一些，意味着更看重决策的准确性，例如在商品推荐系统，尽量减少错误推荐；

希望查全率高，则阈值设置小一些，意味着“有杀错，冇放过”，例如在罪犯检测过程中。

以上性能度量都是基于某个评判条件（分类阈值）
也就是定义一个边界，
如果预测的值超过这个边界就预测为正例
否则预测为反例
可以理解成 先对结果进行排序，然后在某个地方划分，一边为正例，一边为反例。

P-R曲线

P-R曲线就是以P（查准率）-R（查全率）为坐标得到的曲线。
具体步骤为：

1. 根据预测值（可能性）对所有样本进行排序
2. 按顺序逐个把样本作为正例进行预测，得到一个R,P的值

~~P-R曲线有什么意义？~~

比较

1. 若一个学习器的P-R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。
2. 若两个学习器的P-R曲线发生交叉
    1. P-R曲线包围的面积
       它在一定程度上表征了学习器在查准率和查全率上取得双高的比例。
       但是这个值不容易估算。
    2. 平衡点 BEP
       BEP是P=R 查全率等于查准率时候的取值。
       但是这个还是过于简单。
    3. F1
       F1是基于查准率和查全率的调和平均。
       $F1 = \frac{2PR}{P+R} = \frac{2\times TP}{2\times TP+FN+FP} = \frac{2\times TP}{\text{样例总数}+TP-TN}$
    4. F_β
       更一般的F1，能表达出对查准率和查全率的不同偏好
       $F\beta = \frac{(1+\beta)PR}{\beta^2P+R}$
       $\beta > 1$:查全率影响更大
       $\beta = 1$:退化为F1
       $\beta > 1$:查准率影响更大

什么是调和平均？
调和平均的定义：$\frac{1}{F1}=\frac{1}{2}(\frac{1}{P}+\frac{1}{R})$
加权调和平均的定义:
$\frac{1}{F_\beta}=\frac{1}{1+\beta^2}(\frac{1}{P}+\frac{\beta^2}{R})$

macro-P/R/F1 micro-P/R/F1

我们希望在n个二分类混淆矩阵上综合考察查全率和查准率

第一种做法是：
分别计算出查全率和查准率，再甲酸平均值,即
$marop-P=\frac{1}{n}\sum_{i=1}^{n}P_i$
$macro-R=\frac{1}{n}\sum_{i=1}^{n}R_i$
$macro-F1=\frac{2\times macro-P\times macro-R}{macro-P\times macro-R}$

第二种做法是：
将各个矩阵对应的元素平均，再基于这些平均值进行查全率/查准率/F1的计算。
$micro-P=\frac{\overline{TP}}{\overline{TP}+\overline{FP}}$
$micro-R=\frac{\overline{TP}}{\overline{TP}+\overline{FN}}$
$micro-F1=\frac{2\times micro-P\times micro-R}{micro-P\times micro-R}$

ROC曲线和AUC

疑问：在实际应用中，怎么确定分类阈值到底取多少？

也就是说上面的性能度量都是针对某一种情况而得到的。
而我们更希望评测的是在不同任务下的综合的性能好坏。也就排序本身的质量好坏。

ROC曲线就是考虑从排序本身质量好坏的角度来评估的。(前面的P-R曲线应该也是从这个角度考虑。)

ROC仅适用于二分类问题。

TPR(true positive rate):真正例率
FPR(false positive rate):假正例率
$TPR=\frac{TP}{TP+FN}$
$FPR=\frac{FP}{TN+FP}$

具体步骤为：

1. 根据学习器的预测结果对样例进行排序
2. 按顺序逐个把样本作为正例进行预测，计算得到TPR和FPR
3. 以FPR为横轴，以TPR为纵轴作图

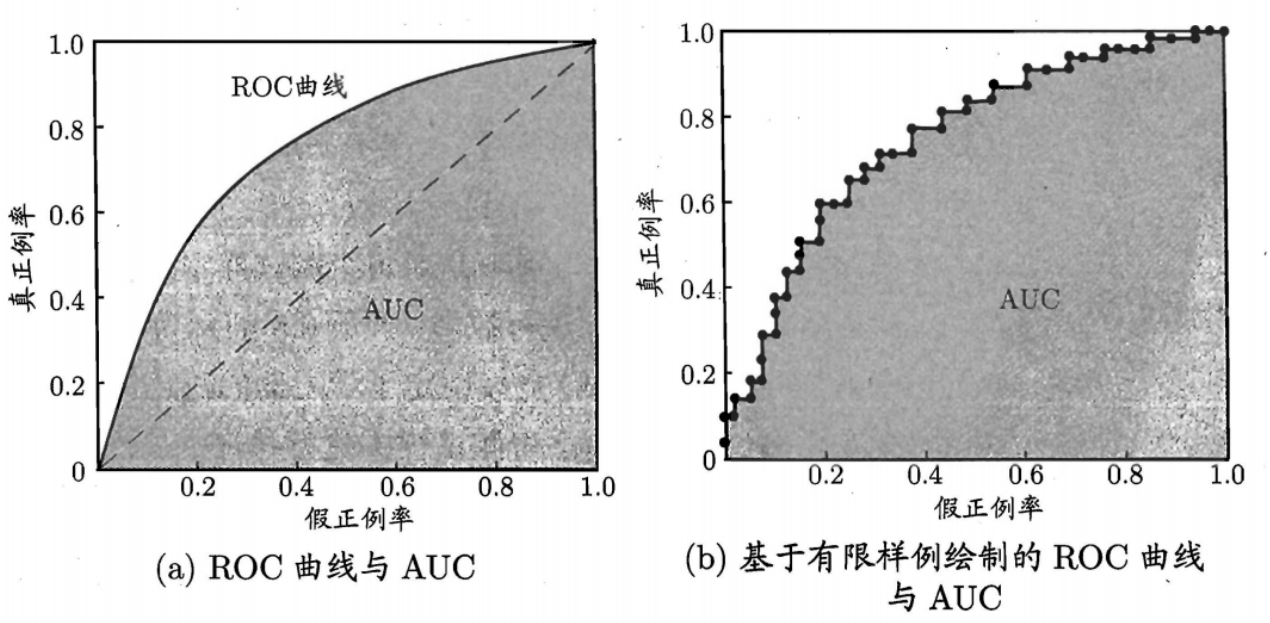


图 2.4 ROC 曲线与 AUC 示意图

设正例的数目为m^+，反例的数目为m^-，那么依次将每个样例预测为正例的时候，设前一个标记点的坐标是(x,y),如果是真正例，那么对应的标记点为$(x, y+\frac{1}{m^+})$，如果是假正例，那么对应的坐标点就是$(x+\frac{1}{m^-}, y)$。

$TPR = \frac{TP}{TP+FN} = \frac{TP}{m^+}$
$FPR = \frac{FP}{TN+FP} = \frac{FP}{m^-}$
预测为真正例:$TPR' = \frac{TP+1}{m^+}$
预测为假正例:$FPR' = \frac{FP+1}{m^-}$

1. ~~对角线对应为"随机猜想"模型，为什么？~~
   ~~个人觉得不对，如果正例的数目和反例的数目相同的情况下才是对角线吧。~~
2. (0,1)则对应于将所有正例排在所有反例之前的理想模型。
   这个应该是指AUC的面积为整个矩形的情况吧。
3. 什么是"随机猜想"模型？

比较

1. 当一个学习器的ROC曲线被另一个学习器包住时，则可以断言后者的性能优于前者。
2. 若两个学习器的ROC曲线发生交叉，较为理想的是比较ROC曲线下的面积 也就是 AUC（Area Under ROC Curve）

AUC 和 ℓ_{rank}

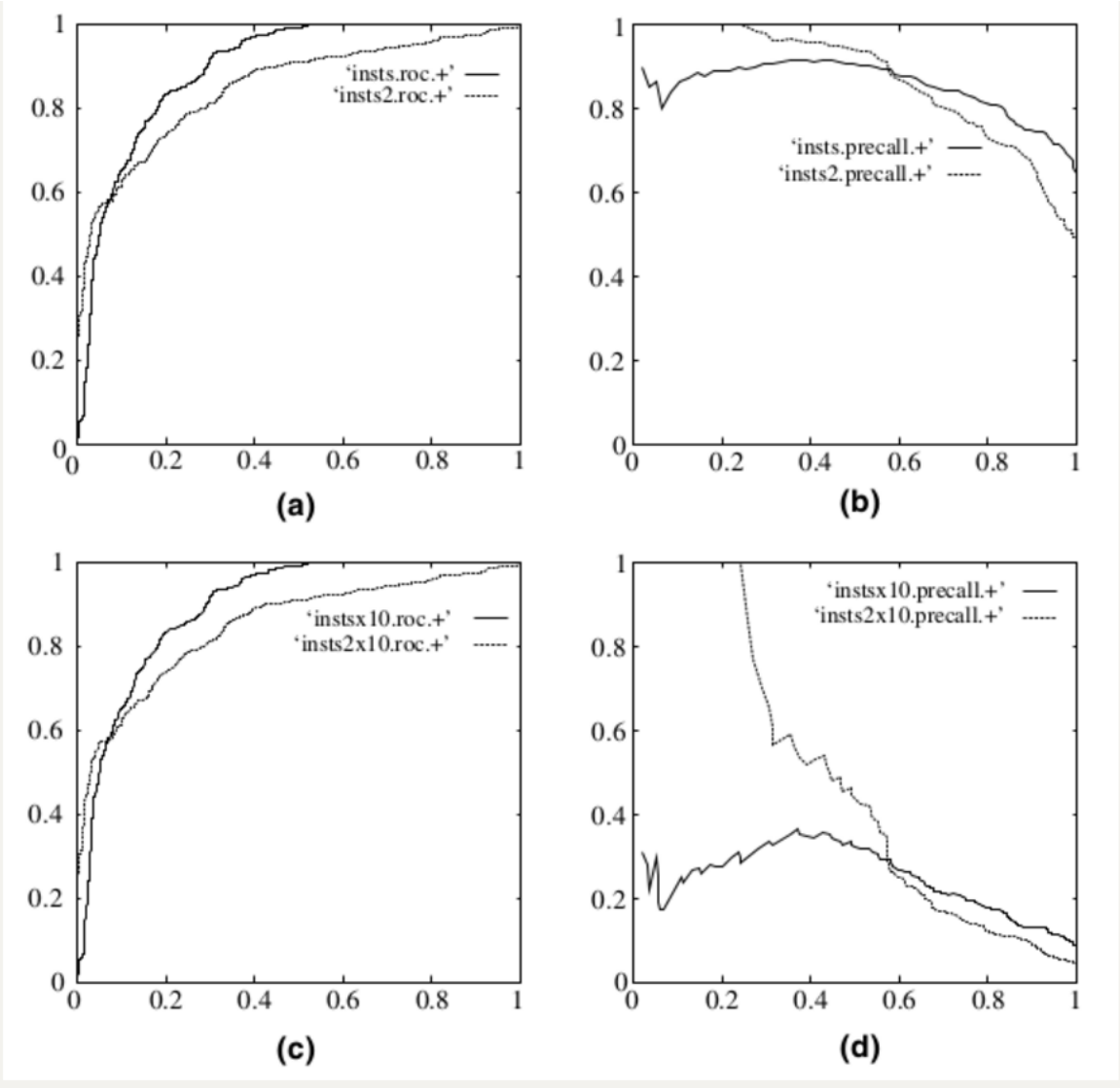
设ROC曲线由坐标$\{(x_1,y_1),(x_2,y_2),\dots,(x_m,y_m)\}$的点按序链接而形成$(x_1=0, x_m=1)$，则AUC可以估算为：
$AUC = \frac{1}{2}\sum_{i=1}^{m-1}(x_{i+1}-x_i)(y_i+y_{i+1})$

给定m^+个正例和m^-个反例，令D^+和D^-分别表示正例、反例的集合，则排序“损失”(loss)定义为:
$$\ell_{rank} = \frac{1}{m^+m^-}\sum_{x^+\in D^+}\sum_{x^-\in D^-}\left(I\big(f(x^+)<f(x^-)\big)+\frac{1}{2}I\big(f(x^+)=f(x^-)\big)\right)$$

即 考虑每一对正例、反例，若正例预测值小于反例，则记一个“罚分”，若相等则记0.5个罚分。

疑问：为什么使用ROC和AUC？

> 既然已经这么多评价标准，为什么还要使用ROC和AUC呢？因为ROC曲线有个很好的特性：当测试集中的正负样本的分布变化的时候，ROC曲线能够保持不变。在实际的数据集中经常会出现类不平衡（class imbalance）现象，即负样本比正样本多很多（或者相反），而且测试数据中的正负样本的分布也可能随着时间变化。下图是ROC曲线和Precision-Recall曲线5的对比：



在上图中，(a)和(c)为ROC曲线，(b)和(d)为Precision-Recall曲线。(a)和(b)展示的是分类其在原始测试集（正负样本分布平衡）的结果，(c)和(d)是将测试集中负样本的数量增加到原来的10倍后，分类器的结果。可以明显的看出，ROC曲线基本保持原貌，而Precision-Recall曲线则变化较大。

疑问：为什么 $AUC = 1 - \ell_{rank}$?
若一个正例在ROC曲线上对应的坐标为(x,y)，则x恰是排序在其之前的反例所占的比例，即假正例率，因此有:？？？
$AUC = 1 - \ell_{rank}$

疑问：同样是点构成的曲线，P-R曲线的面积却说不太容易估算？+

潜在问题

AUC of ROC是机器学习的社群最常使用来比较不同模型优劣的方法2 。然而近来这个做法开始受到质疑，因为有些机器学习的研究指出，AUC的噪声太多，并且很常求不出可信又有效的AUC值（此时便不能保证AUC传达本节开头所述之意义），使得AUC在模型比较时产生的问题比解释的问题更多。

非均等代价

前面的性能度量均默认错误的代价是均等的，只考虑错误的次数，但实际情况中，代价往往是不均等的，比如医生看病误诊可能是致命的，但是作业做错的代价却不是致命的。"代价敏感错误率"，"代价曲线"就是"非均等代价"下的性能度量。

代价矩阵

描述错误代价的矩阵，$cost_{ij}$表示把第i类样呗预测为第j类样本的代价。

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

一般情况下，重要的是代价的比值，而不是绝对值（只有有对比才能体现差距不是？）

代价敏感错误率

在非均等代价下，我们所希望的不再是错误的次数最少，而是希望最小化"总体代价"。

令D^+和D^-分别表示样例集D中的正例集和反例集，则"代价敏感"错误率为:

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

这个东西有什么作用？

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"则可以达到该目的。
代价曲线图的横轴是取值为[0,1]的正例概率代价：p为正例的概率

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

纵轴是取值[0,1]的归一化代价:

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

上面的FPR为假正例率，FNR为假反例率,FNR = 1 - TPR

代价曲线的绘制：

1. ROC上的每一个点对应了代价平面的一个线段
   设坐标点为(FPR,TPR)则可以计算出FNR
2. 在代价平面上绘制 从(0,FPR)到(1,FNR)的线段
   则线段下的面积表示了该条件下的期望总体代价。
3. 将ROC曲线上的每一个点转换成代价平面上的一条线段
   则所有线段的下届围成的面积为在所有条件下学习器的总体期望代价。

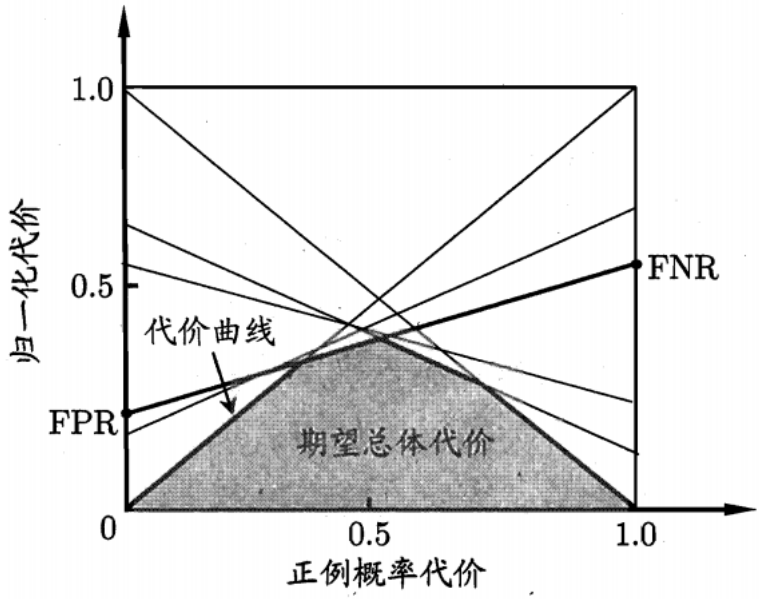


图 2.5 代价曲线与期望总体代价

为什么代价曲线能体现非均等代价的期望？

什么是归一化?
规范化是将不同变化范围的值映射到相同的固定的范围中。
归一化是其中的特例，把值映射到[0,1]的范围。

比较检验

为什么不能直接比较性能度量大大小来判断性能的优劣？

1. 我们希望比较的是泛化性能，然而通过实验评估方法，我们获得的是测试集上的性能，两者的对比结果可能未必相同。
   怎么理解？不是希望用测试集上的结果来近似总体的泛化性能吗？
2. 测试集上的性能与测试集本身的选择有很大的关系，且不论使用大小不同的测试集会得到不同的记过，即便用大小相同的测试集，若包含的测试样例不同，测试的结果也会有不同。
3. 很多机器学习算法本身有一定的随机性，即便用享有同的参数设置在同一个测试集上多次运行，其结果也会不同。
   结果不同，那么测试皆有有什么意义呢？

反正就是，上面3个条件都理解不能。
意思就是说，性能度量得到的结果也不一定是正确的，还需要进一步检验？

假设检验

在现实任务中，我们并不知道学习器的泛化错误率（以错误率为例），只能获知其测试错误率

线性回归算法

梯度下降算法:
偏导决定了参数变化的方向。
理想的情况，每次更新之后，更新的幅度会越来越小。最后收敛
如果学习速率过大，那么最终可能会导致无法收敛。
大部分情况，cost function 是凸函数，所以没有局部解，只有最优解

聚类

聚类：非监督学习
聚类就是依靠数据本身的特性对数据集进行分类。
聚类既能作为一个单独过程，用于找寻数据内在的分布结构，也可作为分类等其他学习任务的前驱过程。

性能度量

对聚类的结果，我们需要某种性能度量来评估其好坏，另一方面，若明确了最终将要使用的性能度量，则可直接将其作为聚类过程的优化目标，从而更好的得到符合要求的聚类结果。

好的聚类结果的 **簇内相似度**(**intra-cluster similarity**)高，且 **簇间相似度**(**inter-cluster similarity**)低。

其性能度量有两种，一种是外部指标，通过和参考模型对比来评估模型。另一种是内部指
标，直接考察聚类的结果而不使用任何模型

外部指标

将聚类结果和某个 **参考模型** 进行比较

定义：
数据集$D = x_1, x_2, \ldots, x_m$ ，
通过聚类给出的簇划分，$C = C_1, C_2, \ldots, C_K$
参考模型给出的簇划分，$C = C_1^*, C_2^*, \ldots, C_K^*$
簇标记：$\lambda_j \in \{1, 2, \ldots, k\}$ ，表示样本x_j的 **簇标记**，即 $x_i \in C_{\lambda_j}$

将样本两两配对有：

$a = |SS|, SS = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}$
a表示 两个模型都将 样本x_i和样本x_j 预测为同类的 样本对数目

$b = |SD|, SD = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}$
b表示，模型将样本x_i和样本$x^{(j)}$预测为同一个类，而参考模型预测为不同类的 样本对的数目

$c = |DS|, DS = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}$
c表示，模型将样本$x^{(i)}$和样本$x^{(j)}$预测为不同类，而参考模型预测为用一个类的 样本对的数目

$d = |DD|, DD = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}$
d表示 两个模型都将 样本x_i和样本x_j 预测为不同类的 样本对数目

因为每个样本对仅在上面的集合中出现过一次，并且 i < j，则有$a + b + c + d = \frac{m(m-1)}{2}$

Jaccard系数(简称 JC)
$JC = \frac{a}{a+b+c}$

FM指数（简称 FMI）
$FMI = \sqrt{\frac{a}{a+b} \times \frac{a}{a+c}}$

Rand指数（简称 RI）
$RI = \frac{2(a+d)}{m(m-1)}$

上述性能度量的结果值域为[0,1] ，值越大越好。

内部指标

直接考察聚类结果而不利用任何 **参考模型**
聚类的结果簇划分为：$C = \{C_1, C_2, \ldots, C_k\}$
定义：
$avg(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \leq i < j \leq |C|} dist(x_i, x_j)$
表示 簇内两个样本距离的均值。

$diam(C) = max_{1 \leq i < j \leq |C|} dist(x_i, x_j)$
表示 簇内两个样本的最远距离

$d_{min}(C_i, C_j) = min_{x_i \in C_i, x_j \in C_j} dist(x_i, x_j)$
表示 两个簇之间样本的最小距离

$d_{cen}(C_i, C_j) = dist(\mu_i, \mu_j)$
表示两个簇，中心的距离。

其中
$dist(\cdot,\cdot)$ 表示两个样本之间的距离。
$\mu = \frac{1}{|C|}\sum_{1\le i\le |C|} x_i$

DB指数（简称 DBI）
$DBI = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{avg(C_i)+avg(C_j)}{d_{cen}(C_i,C_j)}\right)$
分子：两个簇的样本平均距离之和
分母：两个簇中心的距离
注：分母书本是$d_{cen}(\mu_i,\mu_j)$应该是$d_{cen}(C_i,C_j)$

Dunn指数（简称 DI）
$DI = \min_{1\le i\le k}\left\{\min_{j\neq i}\left(\frac{d_{min}(C_i,C_j)}{max_{1\le l\le k}diam(C_l)}\right)\right\}$
分子：两个簇的最短距离
分母：簇内最短距离

DBI越小越好，DI越大越好。

距离计算

聚类的性能度量涉及了 **距离** 的概念。
这个聚类根据属性种类的不同又分为两大种类。
对于属性值有 **序** 的关系，可以使用 **闵可夫斯基距离**
对于属性值是 **无序** 的，比如{飞机，火车，轮船}等，直接给每个类定义一个数值，再使用 **闵可夫斯基距离** 没有很大的意义，对于 这种属性，应该使用 **VDM 距离**。

对函数 $dist(\cdot,\cdot)$，若它是一个 **距离度量** ，则需满足一些基本性质：

1. 非负性：$dist(x^{(i)},x^{(j)}) \ge 0$;
   两个样本的距离不能为负
2. 同一性：$dist(x^{(i)},x^{(j)}) = 0$当且仅当$x^{(i)} = x^{(j)}$
   两个样本重叠时距离才为0
3. 对称性：$dist(x^{(i)},x^{(j)}) = dist(x^{(j)},x^{(i)})$
4. 直递性：$dist(x^{(i)},x^{(j)}) \le dist(x^{(i)},x^{(k)}) + dist(x^{(k)},x^{(j)})$
   也称为三角不等式，但是 有一些 **距离度量** 不一定满足这个性质。

闵可夫斯基距离(Minkowski distance)

给定样本 $x^{(i)} = (x_1^{(i)};x_2^{(i)};\ldots;x_n^{(i)})$ 和样本 $x^{(j)} = (x_1^{(j)};x_2^{(j)};\ldots;x_n^{(j)})$
闵可夫斯基距离(**Minkowski distance**)
$dist_{mk}(x^{(i)},x^{(j)}) = \left(\sum_{u=1}^{n}|x_u^{(i)} - x_u^{(j)}|^P\right)^{\frac{1}{P}}$

也就是 $x^{(i)} - x^{(j)}$的L_p范数$||x^{(i)} - x^{(j)}||_p$

p = 2时，**闵可夫斯基距离** 即 **欧氏距离**
$dist_{ed}(x^{(i)},x^{(j)}) = ||x^{(i)} - x^{(j)}||_2 = \sqrt{\sum_{u=1}^{n}|x_u^{(i)} - x_u^{(j)}|^2}$

p = 1时，**闵可夫斯基距离** 即 **曼哈顿距离**
$dist_{man}(x^{(i)},x^{(j)}) = ||x^{(i)} - x^{(j)}||_1 = \sum_{u=1}^{n}|x_u^{(i)} - x_u^{(j)}|$

连续属性 ：又叫 **数值属性** ，在定义域上有无穷多个可能的取值。
离散属性 ：又叫 **列名属性** ，在定义域上是有限个取值。
有序属性 ：能直接上属性值上计算距离，如：{1,2,3}
无序属性 ：不能在属性值上计算距离，如：{飞机，火车，轮船}

显然，**闵可夫斯基距离** 是用于有序属性的（无序属性进行这样的运算没有具体意义）。

VMD Value Difference Metric

对于 **无序属性** 可以采用 **VMD(Value Difference Metric)**

定义：
$m_{u,a}$:表示在属性u上取值为a的样本数
$m_{u,a,i}$:表示在第 **i** 个样本簇中在属性 **u** 上取值为 **a** 的样本数
k:表示样本簇数
则属性 **u** 上两个离散值 **a** 与 **b** 之间的 **VMD距离** 为：

$$VMD_p(a,b)=\sum_{i=1}^{k}\left|\frac{m_{u,a,i}}{m_{u,a}}-\frac{m_{u,b,i}}{m_{u,b}}\right|$$

这里实际上就是通过把第i簇的属性u根据其不同取值所占的总比例来 **数值化** 或者说 **有序化** 这个属性。（以属性值的权重来数值化离散的属性值）

于是，将 **闵可夫斯基距离** 和 **VMD距离** 结合即可处理混合属性。
定义：有n_c个有序属性，$n-n_c$个无序属性，不失一般性，令有序属性排列在无序属性之前，则

$$MinkovDM_p(x^{(i)},x^{(j)})=(\sum_{u=1}^{n_c}\left|x_u^{(i)}-x_u^{(j)}\right|^p+\sum_{u=n_c+1}^{n}VMD_p(x_u^{(i)},x_u^{(j)}))^{\frac{1}{p}}$$

当样本空间的属性的重要性不同时，可使用 **加权距离(weighted distance)**，
以 **闵可夫斯基距离** 为例:

$$dist_{wmk}(x^{(i)},x^{(j)})=\left(w_1\cdot|x_1^{(i)}-x_1^{(j)}|^p+\ldots+w_n\cdot|x_n^{(i)}-x_n^{(j)}|^p\right)^{\frac{1}{P}}$$

非度量距离

不满足 **直递性** 的距离称为非度量距离。

比如 **人马** 分别和 **人** 还有 **马** 相似，
但是 **人** 和 **马** 不相似。

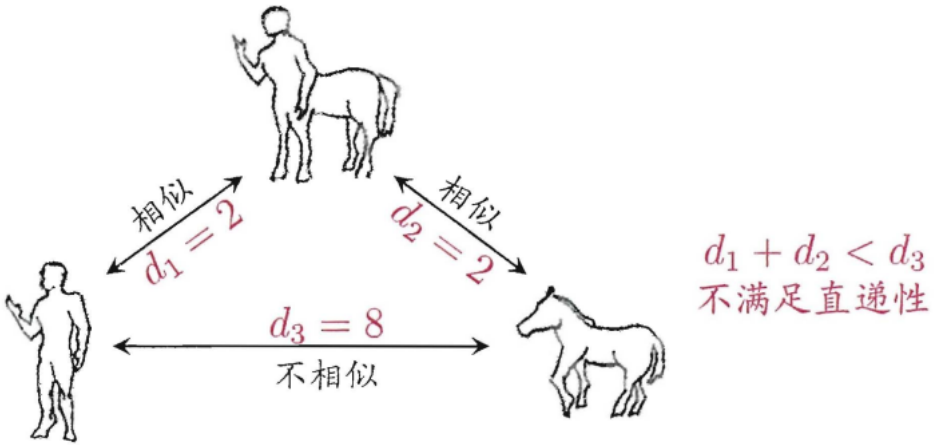


图 9.1 非度量距离的一个例子

原型聚类

原型聚类亦称 **基于原型的聚类 prototype-based clustering**。
此类算法假设类结构能通过一组原型刻画。

k-means 算法

k-means算法采用一组原型向量来刻画聚类结构。
k-means算法 通过设置随机的k个样本作为初始的原型组，每个样本划分为最近那个原型所代表的类中，而原型更新为其所代表的类中所有样本的中心。

有损压缩的思想是用一个n维的向量来代替其所表示的样本（有损压缩的原理）。
定义：距离样本最近的原型，就表示其所在的类，如果两个样本的距离最近的原型相同，那么这两个样本就是同一个类。

设有k个类，对应的原型(簇的中心)为$\mu_1,\mu_2,\ldots,\mu_k$
设第i个样本所在的对应的原型的下标是c_i，

也就是说第i个样本对应的原型就是μ_{c_i}

cost function：$J(c_1, c_2, \ldots, c_n, \mu_1, \ldots, \mu_k) = \frac{1}{m}\sum_{i=1}^{n}||x^{(i)} - \mu_{c_i}||_2^2$

西瓜书的C_i的含义是 C_i是原型μ_i对应的所有样本的集合。

则代价函数也可以描述为：$E = \sum_{i=1}^{k}\sum_{\boldsymbol{x}\in C_i}||\boldsymbol{x} - \boldsymbol{\mu}_i||_2^2$

算法的目的就是最小化上面的代价函数。

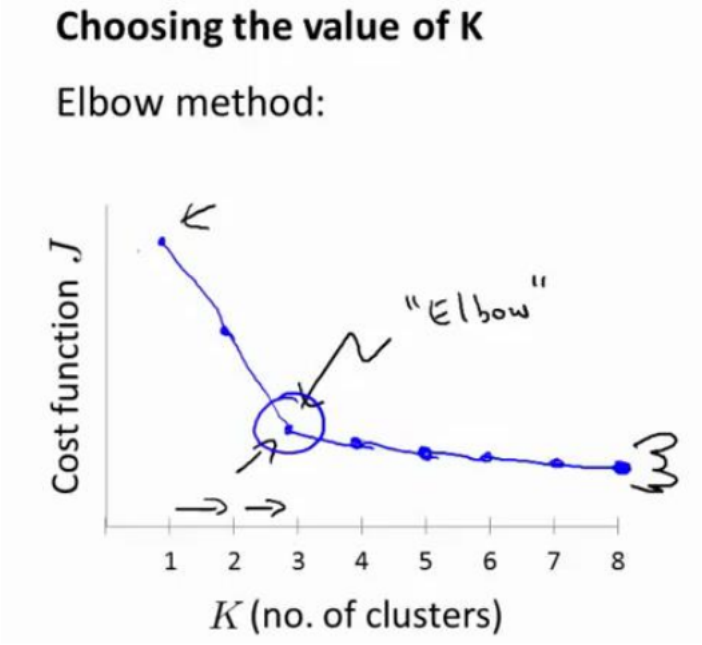
算法具体过程为：

1. 首先是初始化问题，可以随机初始化向量$\boldsymbol{\mu}$，也可以随机取k个样本作为初始的$\boldsymbol{\mu}$
2. 更新步骤
   1. 更新c_i，对于每个样本，取最近的u_j作为其所在的类，也就是$c_i = j$
   2. 更新μ_i，对于每个类，这个类所有样本的中心作为新的μ_i
3. 为避免运行时间过长，通常设置一个最大运行轮数或最小调整幅度阈值，若达到最大轮数或调整幅度小于阈值，则停止运行。

> input:$D = \{x_1, x_2, \ldots, x_n\}, k$
> init $\mu_1, \mu_2, \ldots, \mu_k$
> Repeat {
> for i = 1 to m
> c(i) := index (form 1 to K) of cluster centroid closest to x(i)
> for k = 1 to K
> μk := average (mean) of points assigned to cluster k
> }

聚类数 k的 值怎么确定？

肘部法则 依次取k，若代价函数关于k的图 出现明显的拐点，则k取拐点那个点



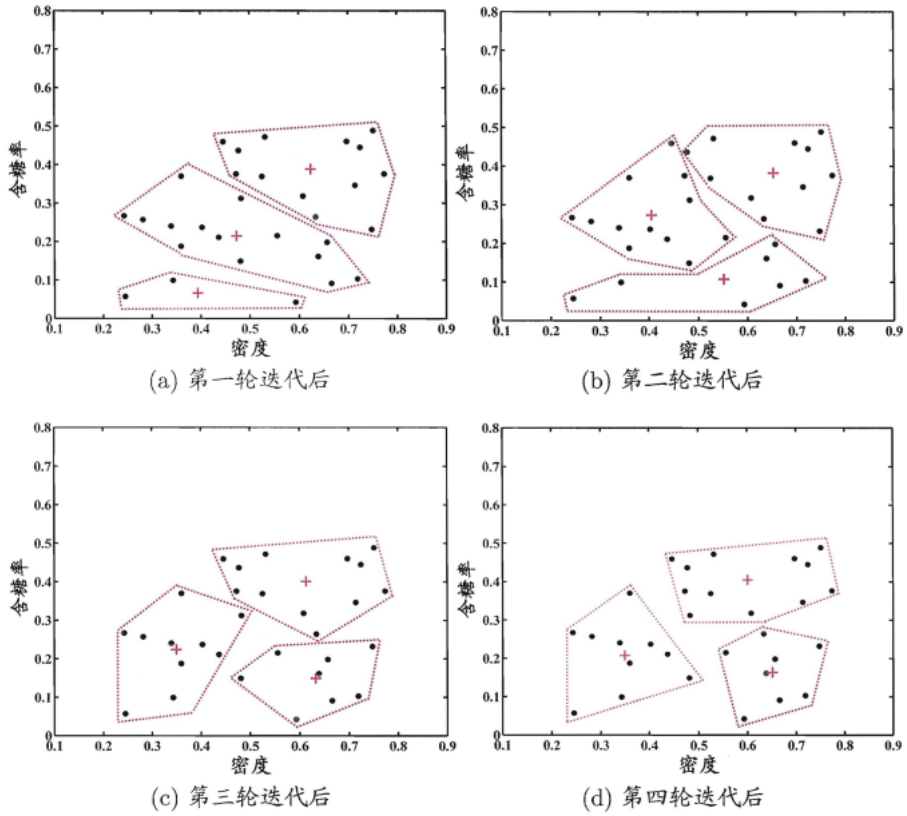


图 9.3 西瓜数据集 4.0 上 k 均值算法($k=3$)在各轮迭代后的结果. 样本点与均值向量分别用"•"与"+"表示, 红色虚线显示出簇划分.

k-means算法仅在凸形簇结果上效果好。

学习向量量化 算法

学习向量量化 算法 和 k-means算法一样采用一组原型向量来刻画聚类结构。
思想和k-means差不多，只不过，这个算法不是使用样本的中心作为原型的更新，而是通过比较样本的标记信息来决定远离还是靠近样本，同时也没有遍历所有样本。

这个算法要求样本含有标记信息，它的思想就是通过标记信息来指导 μ的更新。

算法过程：

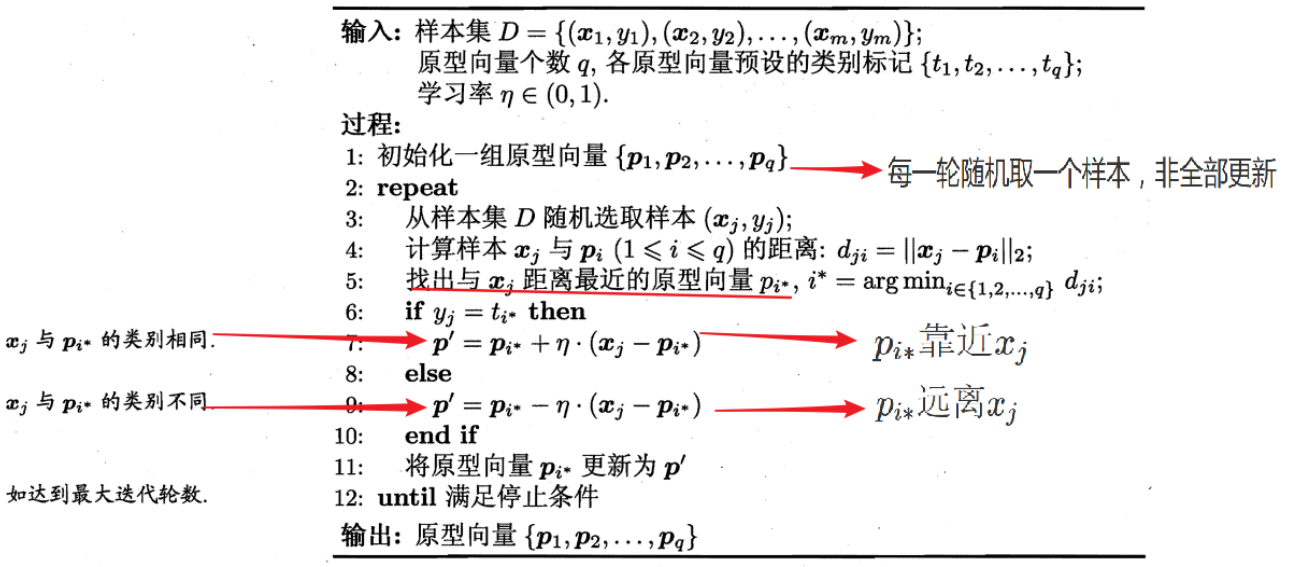


图 9.4 学习向量量化算法

结束条件：当达到最大迭代轮数或者原型向量更新很小甚至不更新。
对于任意的样本x，它将被划分到与其距离最近的原型向量所代表的簇中。

为什么这样更新距离是更近？
如第7行所示，$p' = p_{i*} + \eta \cdot (x_j - p_{i*})$
则p' 和 x_j 之间的距离为：$||p' - x_j||_2 = ||p_{i*} + \eta \cdot (x_j - p_{i*}) - x_j||_2 = (1-\eta) \cdot ||p_{i*} - x_j||_2$

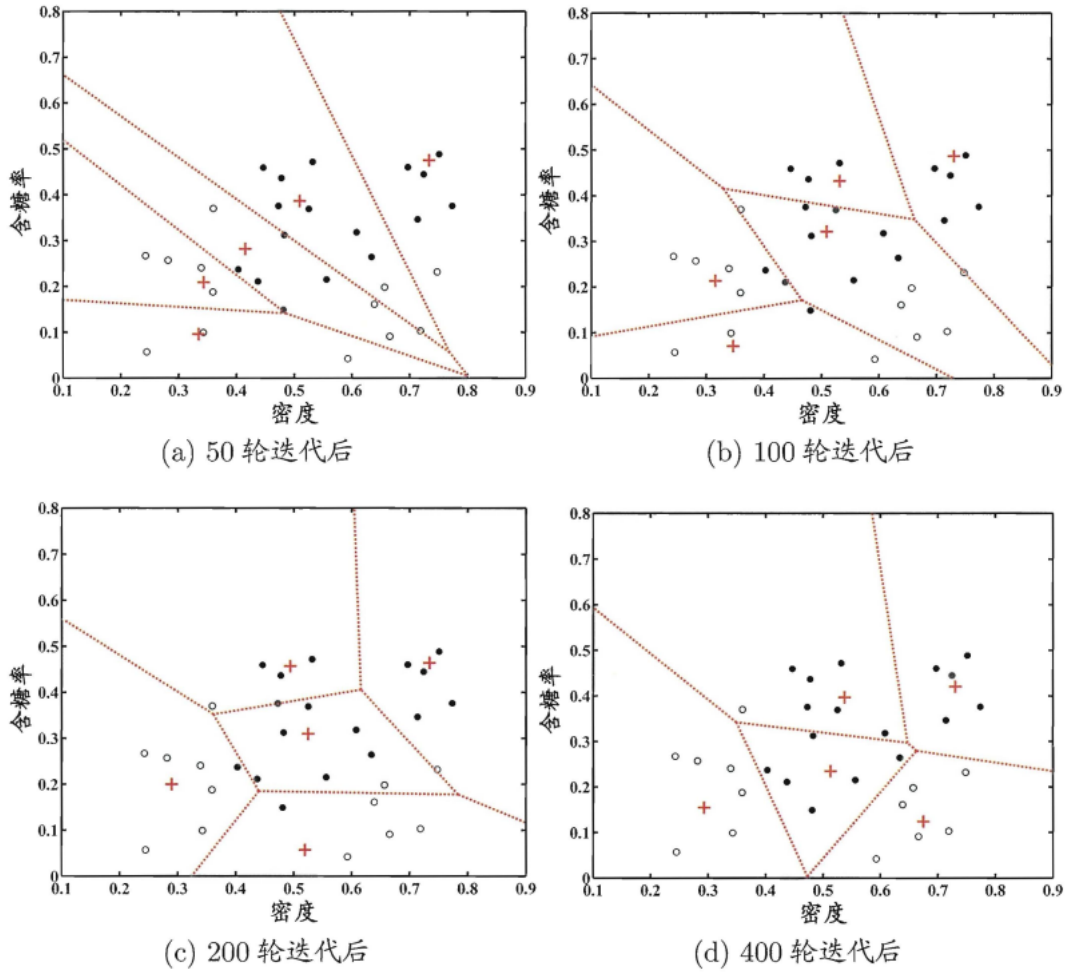


图 9.5 西瓜数据集 4.0 上 LVQ 算法($q = 5$)在不同轮数迭代后的聚类结果. c_1, c_2 类样本点与原型向量分别用"•", "○"与"+"表示, 红色虚线显示出聚类形成的 Voronoi 剖分.

高斯混合聚类

高斯混合聚类(**Mixture-of-Gaussian**)采用 **概率模型** 来表达聚类原型,簇划分由对应原型对应后验概率确定。

高斯分布：
对n维样本空间X中的随机向量x，若x服从高斯分布，其概率密度函数为：
$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$
其中$\boldsymbol{\mu}, \boldsymbol{x}$为n维均值向量，$\Sigma$是$n \times n$的协方差矩阵,$|\Sigma|$为$\Sigma$的行列式
为明确显示高斯分布与相应参数的依赖关系，将概率密度函数记为$p(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma) = p(\boldsymbol{x})$

高斯混合分布：$p_{\mathcal{M}}(x) = \sum_{i-1}^{k} \alpha_i \cdot p(\boldsymbol{x}|\boldsymbol{\mu}_i, \Sigma_i)$
该分布由k个混合成分组成，每个混合成分对应一个高斯分布。其中μ_i和Σ_i为第i个高斯混合成分的参数，而$\alpha_i > 0$为相应的 **混合系数 (mixture coefficient)** ,且$\sum_{i-1}^{k} \alpha_i = 1$

高斯混合聚类的思想是：
假设样本生成过程由高斯混合分布生成，则一个高斯混合成分就表示一个类，在已知的样本（也就是数据集）的情况下，使用极大似然法来估算这个混合高斯分布的各个参数。

PS:可以通过增大分类的数目k来使得数据的分布更加近似高斯混合分布。

训练集：$D = x_1, x_2, \ldots, x_m$
$z_j \in \{1, 2, \ldots, k\}$，表示生成样本$x_j$对应的高斯混合成分。

贝叶斯公式：$P(AB) = P(A)P(B|A) = P(B)P(A|B)$

那么有：
$P(z_j = j \wedge x_j) = P_{\mathcal{M}}(x_j)P(z_j = i|x_j) = P(z_j = i)P(x_j|z_j = i)$

$P_{\mathcal{M}}(x_j)$:高斯混合分布生成样本x_j的概率
$P(z_j = i|x_j)$:已知样本x_j，样本x_j为第i个高斯成分生成的概率
$P(z_j = i)$:样本为第i个高斯分布成分生成的概率，则$P(z_j = i) = \alpha_i$
$P(x_j|z_j = i)$:第i个高斯成分生成x_j的概率，则$P(x_j|z_j = i) = p(x_j|\mu_i, \Sigma_i)$

得：已知样本x_j，样本x_j为第i个高斯成分生成的概率为：
$$P(z_j = i|x_j) = \frac{P(z_j=i)P(x_j|z_j=i)}{P_{\mathcal{M}}(x_j)} = \frac{\alpha_i \cdot p(x_j|\mu_i, \Sigma_i)}{\sum_{l=1}^{k} \alpha_l \cdot p(\boldsymbol{x}_j|\boldsymbol{\mu}_l, \Sigma_l)}$$

为方便，简记为$P(z_j = i|x_j) = \gamma_{ji}$

则x_j分类的结果为其概率最大的一类中：$\lambda_j = \underset{i \in \{1,2,\ldots,k\}}{argmax} P(z_j = i|x_j) = \underset{i \in \{1,2,\ldots,k\}}{argmax} \gamma_{ji}$

那么问题来了，模型的参数怎么确定？

已知样本$D = x_1, x_2, \ldots, x_m$，求 $(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \alpha_i)$
由极大似然法：
m个样本的联合分布为：
$$\begin{aligned} L(D) &= \prod_{j=1}^{m} P_{\mathcal{M}}(x_j) \\ &= \prod_{j=1}^{m} \sum_{l=1}^{k} \alpha_l \cdot p(\boldsymbol{x}_j|\boldsymbol{\mu}_l, \Sigma_l) \end{aligned}$$

则 求 $\boldsymbol{\mu}_i$，由极大似然法：$\boldsymbol{\mu}_i = \underset{\mu_i}{argmax} = L(d)$
也就是要求 $\frac{\partial L(D)}{\boldsymbol{\mu}_i} = 0$
为简化运算，取对数
$$\begin{aligned} LL(D) &= \ln L(D) \\ &= \sum_{j=1}^{m} \ln \sum_{l=1}^{k} \alpha_l \cdot p(\boldsymbol{x}_j|\boldsymbol{\mu}_l, \Sigma_l) \end{aligned}$$

$$\frac{\partial LL(D)}{\partial \boldsymbol{\mu}_i} = \frac{\partial \sum_{j=1}^{m} \ln \sum_{l=1}^{k} \alpha_l \cdot p(\boldsymbol{x}_j|\boldsymbol{\mu}_l, \Sigma_l)}{\partial \boldsymbol{\mu}_i}$$

$$= \sum_{j=1}^{m} \frac{1}{\sum_{l=1}^{k} \alpha_l \cdot p(\boldsymbol{x}_j|\boldsymbol{\mu}_l, \Sigma_l)} \cdot \frac{\partial \sum_{l=1}^{k} \alpha_l \cdot p(\boldsymbol{x}_j|\boldsymbol{\mu}_l, \Sigma_l)}{\partial \mu_i} \quad ,$$

$$= \sum_{j=1}^{m} \frac{\alpha_i}{\sum_{l=1}^{k} \alpha_l \cdot p(\boldsymbol{x}_j|\boldsymbol{\mu}_l, \Sigma_l)} \cdot \frac{\partial p(\boldsymbol{x}_j|\boldsymbol{\mu}_i, \Sigma_i)}{\partial \mu_i} \quad ,\text{当}l = i\text{的时候，求导的值}$$

$$= \sum_{j=1}^{m} \frac{\alpha_i}{\sum_{l=1}^{k} \alpha_l \cdot p(\boldsymbol{x}_j|\boldsymbol{\mu}_l, \Sigma_l)} \cdot p(\boldsymbol{x}_j|\boldsymbol{\mu}_i, \Sigma_i) \cdot \Sigma_i^{-1}(\boldsymbol{x}_j - \boldsymbol{\mu}_i) \quad ,\text{这一步求}$$

$$= \sum_{j=1}^{m} p(z_j = i|\boldsymbol{x}_j) \cdot \Sigma_i^{-1}(\boldsymbol{x}_j - \boldsymbol{\mu}_i) \quad ,$$

$$= \Sigma_i^{-1} \sum_{j=1}^{m} p(z_j = i|\boldsymbol{x}_j) \cdot (\boldsymbol{x}_j - \boldsymbol{\mu}_i) \quad ,\text{常数提到求和符号前面，}\Sigma_i\text{的值能由}\mu_i\text{计算，但不是它的}$$

$$= \Sigma_i^{-1} \sum_{j=1}^{m} \gamma_{ji} \cdot (\boldsymbol{x}_j - \boldsymbol{\mu}_i) \quad ,$$

$$= 0 \quad ,\text{等号两边左乘一个}\Sigma_i$$

解得$\mu_i = \dfrac{\sum_{j=1}^{m}\gamma_{ji}\boldsymbol{x}_j}{\sum_{j=1}^{m}\gamma_{ji}}$

求 $\frac{\partial p(\boldsymbol{x}_j|\boldsymbol{\mu}_i,\Sigma_i)}{\partial \mu_i}$：

$$\begin{aligned}\frac{\partial p(\boldsymbol{x}_j|\boldsymbol{\mu}_i,\Sigma_i)}{\partial \mu_i} &= \frac{\partial \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}}{\partial \mu_i}\\ &= \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}\frac{\partial\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)}{\partial \mu_i}\\ &= p(\boldsymbol{x}_j|\boldsymbol{\mu}_i,\Sigma_i)\cdot\Sigma_i^{-1}(\boldsymbol{x}_j-\boldsymbol{\mu}_i)\end{aligned}$$

$$\begin{aligned}\frac{\partial(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{\partial \mu_i} &= \frac{\partial(\boldsymbol{x}_j-\boldsymbol{\mu}_i)}{\partial \mu_i}\Sigma_i(\boldsymbol{x}_j-\boldsymbol{\mu}_i)+\frac{\partial(\boldsymbol{x}_j-\boldsymbol{\mu}_i)}{\partial \mu_i}(\Sigma_i^{-1})^T(\boldsymbol{x}_j-\boldsymbol{\mu}_i)\\ &= -\left(\Sigma_i^{-1}+(\Sigma_i^{-1})^T\right)(\boldsymbol{x}_j-\boldsymbol{\mu}_i)\\ &= -2\Sigma_i^{-1}(\boldsymbol{x}_j-\boldsymbol{\mu}_i)\end{aligned}$$

> $\begin{cases}\frac{\partial u^T v}{\partial} = \frac{\partial u}{\partial x}v+\frac{\partial v}{\partial x}u\\ \frac{\partial a^T x}{\partial x} = \frac{\partial x^T a}{\partial x} = a\\ (A^T)^{-1} = (A^{-1})^T\end{cases}$

同理：

$$\Sigma_i = \frac{\sum_{j=1}^{m}\gamma_{ji}(\boldsymbol{x}_j-\boldsymbol{\mu}_i)(\boldsymbol{x}_j-\boldsymbol{\mu}_i)^T}{\sum_{j=1}^{m}\gamma_{ji}}$$

对于α_i，除了最大化$LL(D)$，还有约束条件:$\alpha_i \geq 0, \sum_{i=1}^{k}=1$，因此考虑$LL(D)$的拉格朗日形式：$LL(D)F+\lambda\left(\sum_{i=1}^{k}-1\right)$

对α_i的导数为0，得到 $\sum_{j=1}^{m}\frac{p(\boldsymbol{x}_j|\boldsymbol{\mu}_i,\Sigma_i)}{\sum_{l=1}^{k}\alpha_l p(\boldsymbol{x}_j|\boldsymbol{\mu}_l,\Sigma_l)}+\lambda=0$

两边同乘以α_i

对所有的i求和，可以得到 $\lambda=-m$（见后面）

，得$\sum_{j=1}^{m}\gamma_{ji}+\lambda\alpha_i=0$

则$\alpha_i = \frac{1}{m}\sum_{j=1}^{m}\gamma_{ji}$

对所有的i求和：

西瓜书上原文是 对所有的样本求和 ，实际上是对所有的 i 进行求和：

$$\sum_{j=1}^{m}\frac{\alpha_i p(\boldsymbol{x}_j|\boldsymbol{\mu}_i,\Sigma_i)}{\sum_{l=1}^{k}\alpha_l p(\boldsymbol{x}_j|\boldsymbol{\mu}_l,\Sigma_l)}+\alpha_i\lambda=0 \quad ,\text{原式}$$

$$\sum_{i=1}^{k}\sum_{j=1}^{m}\frac{\alpha_i p(\boldsymbol{x}_j|\boldsymbol{\mu}_i,\Sigma_i)}{\sum_{l=1}^{k}\alpha_l p(\boldsymbol{x}_j|\boldsymbol{\mu}_l,\Sigma_l)}+\sum_{i=1}^{k}\alpha_i\lambda=0 \quad ,\text{求和}$$

$$\sum_{j=1}^{m}\frac{\sum_{i=1}^{k}\alpha_i p(\boldsymbol{x}_j|\boldsymbol{\mu}_i,\Sigma_i)}{\sum_{l=1}^{k}\alpha_l p(\boldsymbol{x}_j|\boldsymbol{\mu}_l,\Sigma_l)}+\lambda=0 \quad ,\text{求和符号放分子上，且}\sum_{i=1}^{k}\alpha_i=1$$

$$\sum_{j=1}^{m}1+\lambda=0 \quad ,\text{分子，分母相同约分}$$

$$\lambda=-m$$

由上述的推导即可获得高斯混合模型的EM算法：在每步迭代中，先根据当前参数来计算每一个样本属于每个高斯成分的后验概率γ_{ji}(E步),再根据上面推导的公式来更新模型参数 $\{(\mu_i,\Sigma_i,\alpha_i)|1\leq i\leq k\}$(M步)

具体算法为：

输入： 样本集 $D=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m\}$;
高斯混合成分个数 k.

过程：

1: 初始化高斯混合分布的模型参数 $\{(\alpha_i,\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)\mid 1\leqslant i\leqslant k\}$
2: **repeat**
EM 算法的 E步.
3: **for** $j=1,2,\ldots,m$ **do**
4: 根据式(9.30)计算 $\boldsymbol{x}_j$ 由各混合成分生成的后验概率, 即 $\gamma_{ji}=p_{\mathcal{M}}(z_j=i\mid\boldsymbol{x}_j)\ (1\leqslant i\leqslant k)$
5: **end for**
EM 算法的 M步.
6: **for** $i=1,2,\ldots,k$ **do**
7: 计算新均值向量: $\boldsymbol{\mu}_i'=\frac{\sum_{j=1}^{m}\gamma_{ji}\boldsymbol{x}_j}{\sum_{j=1}^{m}\gamma_{ji}}$;
8: 计算新协方差矩阵: $\boldsymbol{\Sigma}_i'=\frac{\sum_{j=1}^{m}\gamma_{ji}(\boldsymbol{x}_j-\boldsymbol{\mu}_i')(\boldsymbol{x}_j-\boldsymbol{\mu}_i')^{\mathrm{T}}}{\sum_{j=1}^{m}\gamma_{ji}}$;
9: 计算新混合系数: $\alpha_i'=\frac{\sum_{j=1}^{m}\gamma_{ji}}{m}$;
10: **end for**
11: 将模型参数 $\{(\alpha_i,\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)\mid 1\leqslant i\leqslant k\}$ 更新为 $\{(\alpha_i',\boldsymbol{\mu}_i',\boldsymbol{\Sigma}_i')\mid 1\leqslant i\leqslant k\}$
例如达到最大迭代轮数.
12: **until** 满足停止条件
13: $C_i=\varnothing\ (1\leqslant i\leqslant k)$
14: **for** $j=1,2,\ldots,m$ **do**
15: 根据式(9.31)确定 $\boldsymbol{x}_j$ 的簇标记 λ_j;
16: 将 $\boldsymbol{x}_j$ 划入相应的簇: $C_{\lambda_j}=C_{\lambda_j}\bigcup\{\boldsymbol{x}_j\}$
17: **end for**
输出： 簇划分 $\mathcal{C}=\{C_1,C_2,\ldots,C_k\}$

图 9.6 高斯混合聚类算法

k-means 算法可以看作是高斯聚类在混合成分方差相等，且每个样本仅指派给一个混合成分时的特例。

疑问：协方差矩阵怎么计算？

密度聚类

密度聚类：基于密度的聚类，此类算法假设聚类结构能通过样本的分布的紧密成都确定。
通常情形下，密度聚类算法从样本密度的角度来考察样本之间的可连接性，并基于可连接样本不断扩展聚类簇以获得最终的聚类结果，

DBSCAN 算法

把每个样本看作一个点，如果两个点可达（距离在一定范围内），则这两个点可以看作属于同一个类。
DBSCAN算法的过程 简单的来说就是 建图和寻找"联通分支"的过程。

DBSCAN算法有两个参数$(\epsilon, MinPts)$
ϵ:描述类内点的紧密程度。
$MinPts$:样本称为核心对象的条件。

一堆堆定义：
x_i的ϵ邻域：表示距离这个样本ϵ的范围内的样本的集合

核心对象:如果一个样本的ϵ邻域内样本的数目大于$MinPts$，那么就说这个样本为核心对象。

密度直达:样本和其邻域内的样本相互直达

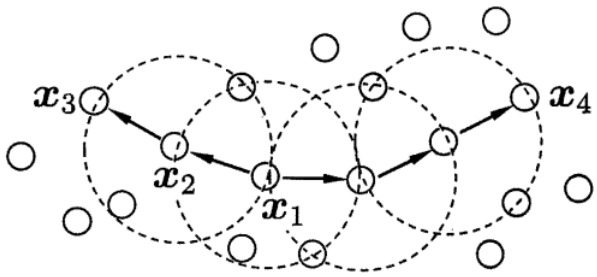


图 9.8 DBSCAN 定义的基本概念$(MinPts = 3)$: 虚线显示出 ϵ-邻域, x_1 是核心对象, x_2 由 x_1 密度直达, x_3 由 x_1 密度可达, x_3 与 x_4 密度相连.

算法的思想：把相互可达的核心对象及其可达的非核心对象划分为一个类。
对于非核心对象，则根据先来后到的原则来进行瓜分。

感觉就像在求联通分支：每个样本表示一个节点，距离小于ϵ则连边，然后求各个连通分支。
（注：核心对象之间是双向边，核心对象和非核心对象为单向边）

优化建议：上面的算法是根据先来后到的原则来瓜分非核心对象，这些非核心对象分配给其最近的类应该更为合适。

算法过程：

1. 随机取一个核心对象，加入队列q中
2. 循环处理q，直到q为空
    1. 从q取出一个对象（这个对象是非核心对象，因为下面）
    2. 如果这个核心对象，就把它邻域内的所有样本都加入到队列中（其中就包括了非核心对象）
3. 上述过程所有取出的样本作为一类，
4. 剩下的样本重复上面的过程，知道划分完毕

输入: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
邻域参数 $(\epsilon, MinPts)$.

过程:

初始化核心对象集合: $\Omega = \varnothing$
for $j = 1, 2, \ldots, m$ **do**
  确定样本 $\boldsymbol{x}_j$ 的 ϵ-邻域 $N_\epsilon(\boldsymbol{x}_j)$;
  if $|N_\epsilon(\boldsymbol{x}_j)| \geqslant MinPts$ **then**
    将样本 $\boldsymbol{x}_j$ 加入核心对象集合: $\Omega = \Omega \bigcup \{\boldsymbol{x}_j\}$
  end if
end for
初始化聚类簇数: $k = 0$
初始化未访问样本集合: $\Gamma = D$
while $\Omega \neq \varnothing$ **do**
  记录当前未访问样本集合: $\Gamma_{\text{old}} = \Gamma$;
  随机选取一个核心对象 $\boldsymbol{o} \in \Omega$, 初始化队列 $Q =< \boldsymbol{o} >$;
  $\Gamma = \Gamma \setminus \{\boldsymbol{o}\}$;
  while $Q \neq \varnothing$ **do**
    取出队列 Q 中的首个样本 $\boldsymbol{q}$;
    if $|N_\epsilon(\boldsymbol{q})| \geqslant MinPts$ **then**
      令 $\Delta = N_\epsilon(\boldsymbol{q}) \bigcap \Gamma$;
      将 Δ 中的样本加入队列 Q;
      $\Gamma = \Gamma \setminus \Delta$;
    end if
  end while
  $k = k + 1$, 生成聚类簇 $C_k = \Gamma_{\text{old}} \setminus \Gamma$;
  $\Omega = \Omega \setminus C_k$
end while

输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$

图 9.9 DBSCAN 算法

2-7行：找出所有的核心对象
14行：while循环内只处理核心对象
18行：非核心对象也加入到队列中

简单从图的角度看，
建图：核心对象之间建立双向边，核心对象和非核心对象建立单项边。
找强联通分支：直接根据上述建立的图搜索即可。

输出结果：密度可达的核心对象及其邻域划分为一个簇。

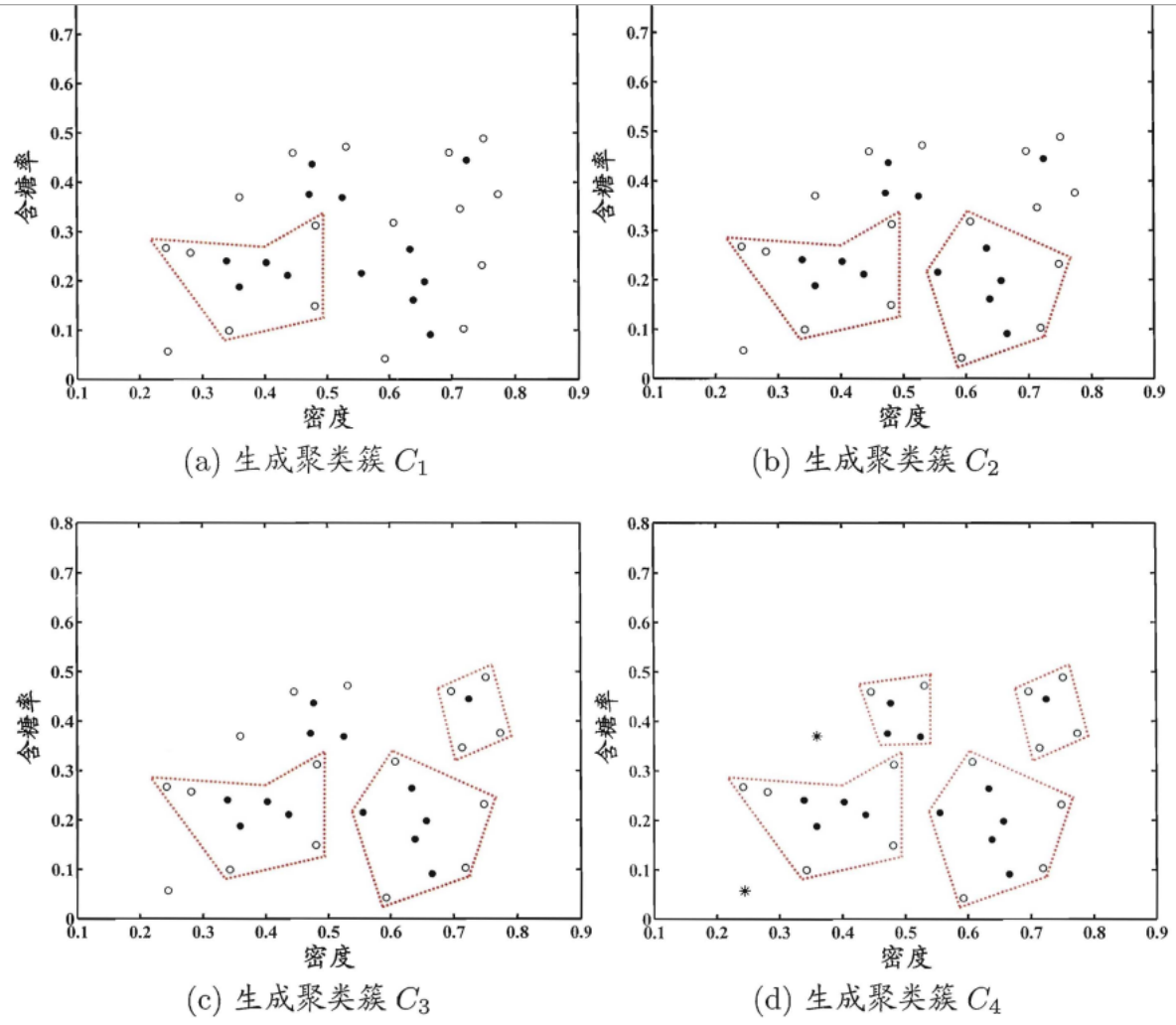


图 9.10 DBSCAN 算法($\epsilon = 0.11$, $MinPts = 5$)生成聚类簇的先后情况. 核心对象、非核心对象、噪声样本分别用"●" "○" "*"表示, 红色虚线显示出簇划分.

层次聚类

层次聚类试图在不同层次对数据集进行划分，从而形成树形的聚类结构。

AGNES 算法

AGNES 算法的过程类似 kruskal算法
算法过程：

1. 初始化每个样本作为一个类
2. 取最近的两个类合并成一个类
3. 不断重复这个步骤，直到达到预设的聚类簇个数，或者类间距离大于某个阈值

输入: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
聚类簇距离度量函数 d;
聚类簇数 k.

过程:

1: **for** $j = 1, 2, \ldots, m$ **do**
2: $C_j = \{\boldsymbol{x}_j\}$
3: **end for**
4: **for** $i = 1, 2, \ldots, m$ **do**
5: **for** $j = 1, 2, \ldots, m$ **do**
6: $M(i, j) = d(C_i, C_j)$;
7: $M(j, i) = M(i, j)$
8: **end for**
9: **end for**
10: 设置当前聚类簇个数: $q = m$
11: **while** $q > k$ **do**
12: 找出距离最近的两个聚类簇 C_{i^*} 和 C_{j^*};
13: 合并 C_{i^*} 和 C_{j^*}: $C_{i^*} = C_{i^*} \bigcup C_{j^*}$;
14: **for** $j = j^* + 1, j^* + 2, \ldots, q$ **do**
15: 将聚类簇 C_j 重编号为 C_{j-1}
16: **end for**
17: 删除距离矩阵 M 的第 j^* 行与第 j^* 列;
18: **for** $j = 1, 2, \ldots, q - 1$ **do**
19: $M(i^*, j) = d(C_{i^*}, C_j)$;
20: $M(j, i^*) = M(i^*, j)$
21: **end for**
22: $q = q - 1$
23: **end while**

输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$

通常使用 $d_{\min}$, $d_{\max}$ 或 d_{avg}.

初始化单样本聚类簇.

初始化聚类簇距离矩阵.

$i^* < j^*$.

图 9.11 AGNES 算法

1-3行：初始化每一个类为一个簇
4-9行：计算类间距离
10-23：两辆合并最近的类

关于层次的直观理解：从某一层划分（虚线），则划分出来的每一棵子树代表一个簇。

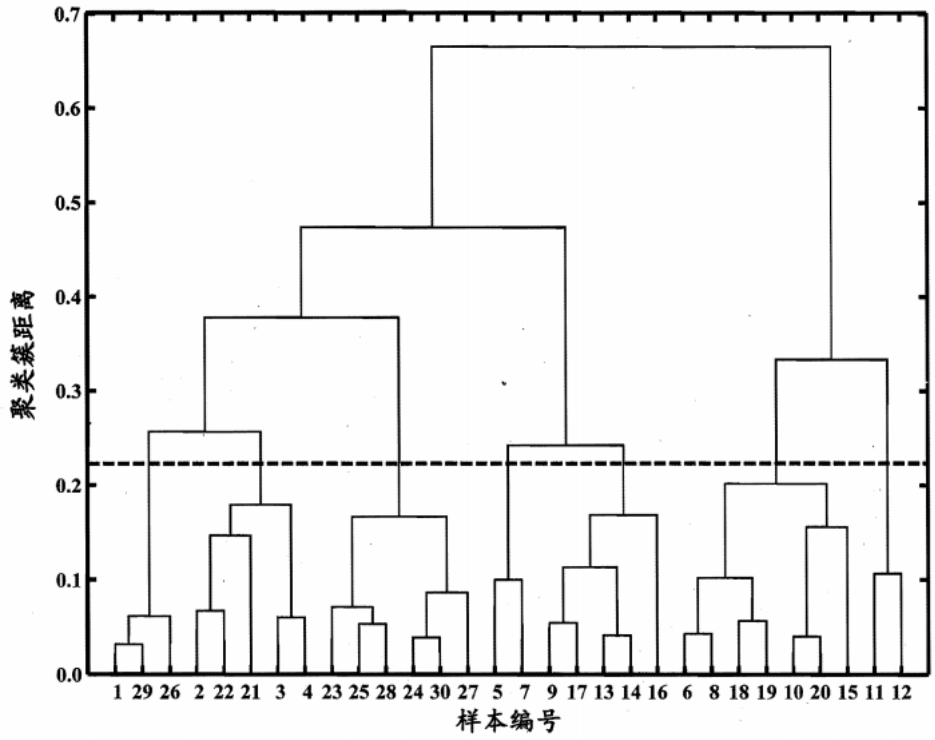


图 9.12 西瓜数据集 4.0 上 AGNES 算法生成的树状图(采用 $d_{\max}$). 横轴对应于样本编号, 纵轴对应于聚类簇距离.

每一个中间节点，对应的纵轴表示，它子树所代表的两个簇类的距离。
注：上面那个图之所以那么好看，是因为样本的编号不是顺序排列的。

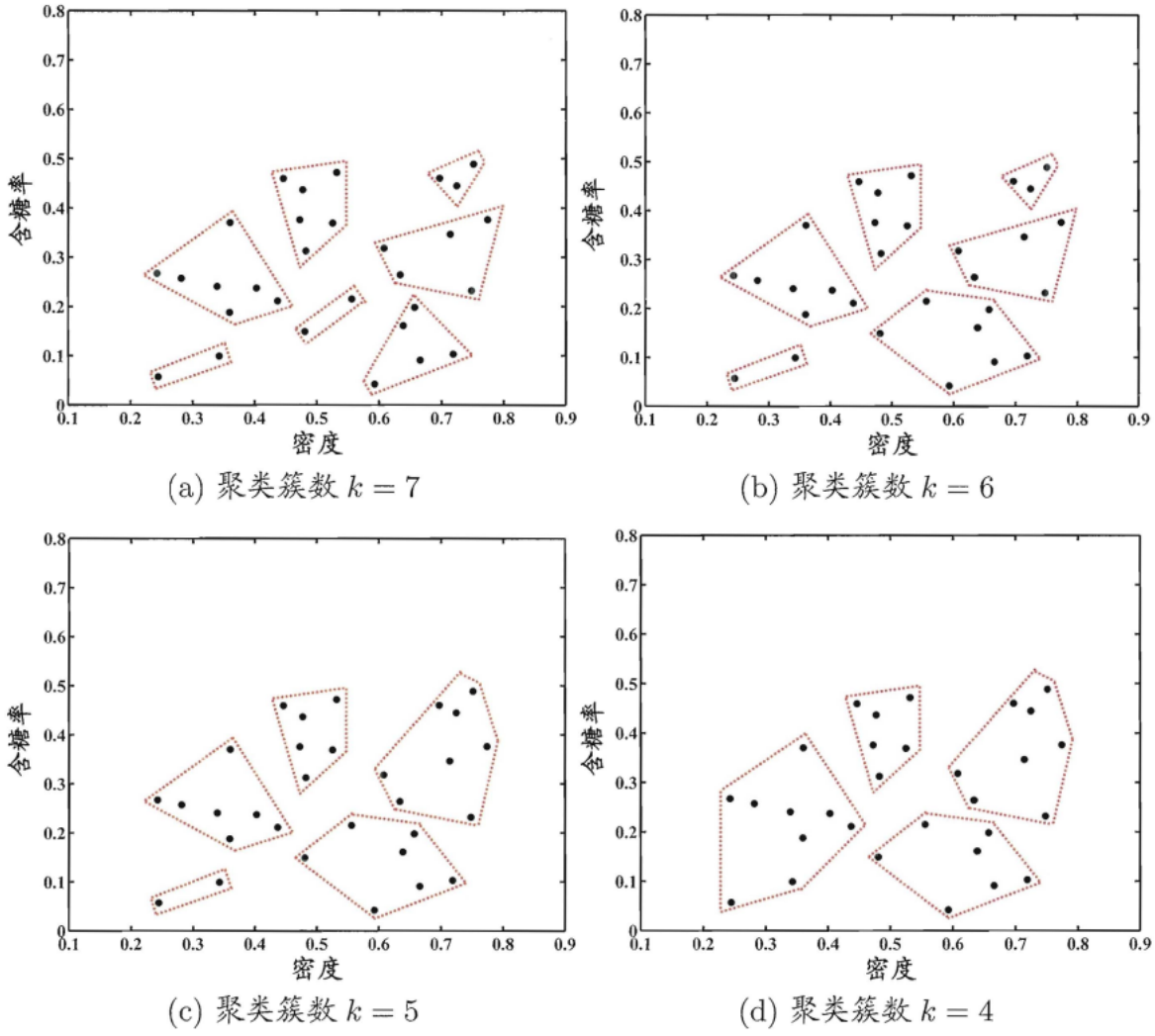


图 9.13 西瓜数据集 4.0 上 AGNES 算法(采用 $d_{\max}$)在不同聚类簇数($k=7, 6, 5, 4$)时的簇划分结果. 样本点用"•"表示, 红色虚线显示出簇划分.

最大似然法

极大似然法是点估计方法的一种。
所谓参数的点估计，就是找一个合适的统计量，将样本观测值代入到统计量得到的值就作为该参数的估计。
具体来说，就是已知每个样本的分布，根据样本估计这个分布的参数。
极大似然法的思想主要基于"小概率事件在一次实验中几乎不可能发生"。
也就是说，一次实验发生这个已经发生的概率（样本）的概率是最大的。

具体到计算的话，就是

1. 假设概率分布函数
2. 求样本的联合分布函数（似然函数）
3. 令似然函数的求导结果为0，得到若干个方程
   导数为0的地方，往往是极值点
4. 根据上面的方程求解参数

已知样本X1，X2,X3,...,XN,其混合分布为L(X1,X2,X3,...,XN)，则L的参数 应该满足在这个参数下，取得上面的样本的概率为最大。则可以使用偏导 $=0$ 来解得参数

朗格朗日乘数法

当在求某个函数的极值的时候，题目往往还带有约束条件，比如上面的极大似然法的参数可能还有其他的约束条件。
那么可以考虑使用拉格朗日乘数法。
具体的做法是：把约束条件也加入到目标函数中去，使得约束条件成为目标函数的一部分，进而把问题转换成了无约束条件的最优化问题

矩阵求导/微分

根据wiki上的说法，矩阵求导至今为止还没有公认的一种定义（约定），甚至有时候一篇文章也可能同时使用了不同的矩阵求导的定义（约定）。
PS:这里说的矩阵求导，指的是矩阵对矩阵求导，而标量对标量的求导可以看成是1×1矩阵的求导。

矩阵求导的有两种比较普遍的约定：
对于 $\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{X}}$，主要有两种处理方法：

1. 先展开X，转置，再展开Y（展开的意思见下面）
2. 先展开Y，转置，再展开X
   这里所谓的展开，指的是把求导运算代入到被展开矩阵中的每一个元素上。

上面的两种约定得到的结果，刚好互为转置。
以两个列向量$\boldsymbol{y} \in \mathbb{R}^m$ 和 $\boldsymbol{x} \in \mathbb{R}^n$为例,采用第二种约定:

$$\begin{aligned}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} &= \begin{bmatrix} \frac{\partial y_1}{\partial \boldsymbol{x}} \\ \frac{\partial y_2}{\partial \boldsymbol{x}} \\ \vdots \\ \frac{\partial y_m}{\partial \boldsymbol{x}} \end{bmatrix}^T && \text{,展开}\boldsymbol{y}\text{，并转置} \\ &= \begin{bmatrix} \frac{\partial y_1}{\partial \boldsymbol{x}} \frac{\partial y_2}{\partial \boldsymbol{x}} \cdots \frac{\partial y_m}{\partial \boldsymbol{x}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \frac{\partial y_2}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix} && \text{,展开}\boldsymbol{x}\end{aligned}$$

以上是矩阵求导的基本定义，然后还有一些定理和公式，比如链式法则，两个矩阵相乘再求导等，目前还没有进行证明。