

# 第 10 章 降维与度量学习

## 10.1 $k$ 近邻学习

$k$  近邻( $k$ -Nearest Neighbor, 简称  $k$ NN)学习是一种常用的监督学习方法, 其工作机制非常简单: 给定测试样本, 基于某种距离度量找出训练集中与其最近的  $k$  个训练样本, 然后基于这  $k$  个“邻居”的信息来进行预测. 通常, 在分类任务中可使用“投票法”, 即选择这  $k$  个样本中出现最多的类别标记作为预测结果; 在回归任务中可使用“平均法”, 即将这  $k$  个样本的实值输出标记的平均值作为预测结果; 还可基于距离远近进行加权平均或加权投票, 距离越近的样本权重越大.

参见 8.4 节.

与前面介绍的学习方法相比,  $k$  近邻学习有一个明显的不同之处: 它似乎没有显式的训练过程! 事实上, 它是“懒惰学习”(lazy learning)的著名代表, 此类学习技术在训练阶段仅仅是把样本保存起来, 训练时间开销为零, 待收到测试样本后再进行处理; 相应的, 那些在训练阶段就对样本进行学习处理的方法, 称为“急切学习”(eager learning).

图 10.1 给出了  $k$  近邻分类器的一个示意图. 显然,  $k$  是一个重要参数, 当  $k$  取不同值时, 分类结果会有显著不同. 另一方面, 若采用不同的距离计算方式, 则找出的“近邻”可能有显著差别, 从而也会导致分类结果有显著不同.

暂且假设距离计算是“恰当”的, 即能够恰当地找出  $k$  个近邻, 我们来对“最近邻分类器”(1NN, 即  $k = 1$ )在二分类问题上的性能做一个简单的讨论.

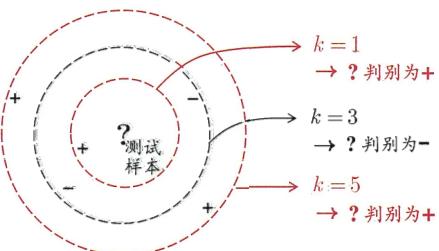


图 10.1  $k$  近邻分类器示意图. 虚线显示出等距线; 测试样本在  $k = 1$  或  $k = 5$  时被判别为正例,  $k = 3$  时被判别为反例.

给定测试样本  $x$ , 若其最近邻样本为  $z$ , 则最近邻分类器出错的概率就是  $x$  与  $z$  类别标记不同的概率, 即

$$P(\text{err}) = 1 - \sum_{c \in \mathcal{Y}} P(c | x)P(c | z). \quad (10.1)$$

假设样本独立同分布, 且对任意  $x$  和任意小正数  $\delta$ , 在  $x$  附近  $\delta$  距离范围内总能找到一个训练样本; 换言之, 对任意测试样本, 总能在任意近的范围内找到式(10.1)中的训练样本  $z$ . 令  $c^* = \arg \max_{c \in \mathcal{Y}} P(c | x)$  表示贝叶斯最优分类器的结果, 有

贝叶斯最优分类器参见  
7.1节.

$$\begin{aligned} P(\text{err}) &= 1 - \sum_{c \in \mathcal{Y}} P(c | x)P(c | z) \\ &\simeq 1 - \sum_{c \in \mathcal{Y}} P^2(c | x) \\ &\leq 1 - P^2(c^* | x) \\ &= (1 + P(c^* | x))(1 - P(c^* | x)) \\ &\leq 2 \times (1 - P(c^* | x)). \end{aligned} \quad (10.2)$$

为便于初学者理解, 本节仅做了一个简化讨论, 更严格地分析参阅 [Cover and Hart, 1967].

于是我们得到了有点令人惊讶的结论: 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

上一节的讨论是基于一个重要假设: 任意测试样本  $x$  附近任意小的  $\delta$  距离范围内总能找到一个训练样本, 即训练样本的采样密度足够大, 或称为“密采样” (dense sample). 然而, 这个假设在现实任务中通常很难满足, 例如若  $\delta = 0.001$ , 仅考虑单个属性, 则仅需 1000 个样本点平均分布在归一化后的属性取值范围内, 即可使得任意测试样本在其附近 0.001 距离范围内总能找到一个训练样本, 此时最近邻分类器的错误率不超过贝叶斯最优分类器的错误率的两倍. 然而, 这仅是属性维数为 1 的情形, 若有更多的属性, 则情况会发生显著变化. 例如假定属性维数为 20, 若要求样本满足密采样条件, 则至少需  $(10^3)^{20} = 10^{60}$  个样本. 现实应用中属性维数经常成千上万, 要满足密采样条件所需的样本数目是无法达到的天文数字. 此外, 许多学习方法都涉及距离计算, 而高维空间会给距离计算带来很大的麻烦, 例如当维数很高时甚至连计算内积

作为参照量: 宇宙间基本粒子的总数约为  $10^{80}$  (一粒灰尘中含有几十亿个基本粒子).

都不再容易.

[Bellman, 1957] 最早提出, 亦称“维数诅咒”、“维数危机”.

另一个重要途径是特征选择, 参见第 11 章.

事实上, 在高维情形下出现的数据样本稀疏、距离计算困难等问题, 是所有机器学习方法共同面临的严重障碍, 被称为“维数灾难”(curse of dimensionality).

缓解维数灾难的一个重要途径是降维(dimension reduction), 亦称“维数约简”, 即通过某种数学变换将原始高维属性空间转变为一个低维“子空间”(subspace), 在这个子空间中样本密度大幅提高, 距离计算也变得更容易. 为什么能进行降维? 这是因为在很多时候, 人们观测或收集到的数据样本虽是高维的, 但与学习任务密切相关的也许仅是某个低维分布, 即高维空间中的一个低维“嵌入”(embedding). 图 10.2 给出了一个直观的例子. 原始高维空间中的样本点, 在这个低维嵌入子空间中更容易进行学习.

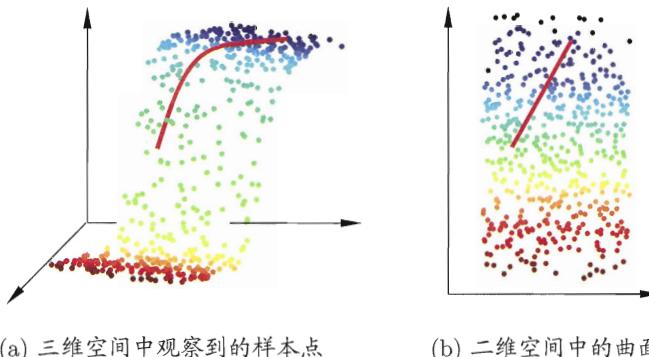


图 10.2 低维嵌入示意图

若要求原始空间中样本之间的距离在低维空间中得以保持, 如图 10.2 所示, 即得到“多维缩放”(Multiple Dimensional Scaling, 简称 MDS) [Cox and Cox, 2001] 这样一种经典的降维方法. 下面做一个简单的介绍.

假定  $m$  个样本在原始空间的距离矩阵为  $\mathbf{D} \in \mathbb{R}^{m \times m}$ , 其第  $i$  行  $j$  列的元素  $dist_{ij}$  为样本  $\mathbf{x}_i$  到  $\mathbf{x}_j$  的距离. 我们的目标是获得样本在  $d'$  维空间的表示  $\mathbf{Z} \in \mathbb{R}^{d' \times m}$ ,  $d' \leq d$ , 且任意两个样本在  $d'$  维空间中的欧氏距离等于原始空间中的距离, 即  $\|\mathbf{z}_i - \mathbf{z}_j\| = dist_{ij}$ .

令  $\mathbf{B} = \mathbf{Z}^T \mathbf{Z} \in \mathbb{R}^{m \times m}$ , 其中  $\mathbf{B}$  为降维后样本的内积矩阵,  $b_{ij} = \mathbf{z}_i^T \mathbf{z}_j$ , 有

$$\begin{aligned} dist_{ij}^2 &= \|\mathbf{z}_i\|^2 + \|\mathbf{z}_j\|^2 - 2\mathbf{z}_i^T \mathbf{z}_j \\ &= b_{ii} + b_{jj} - 2b_{ij}. \end{aligned} \quad (10.3)$$

$\mathbf{0} \in \mathbb{R}^{d'}$  为全零向量.

为便于讨论, 令降维后的样本  $\mathbf{Z}$  被中心化, 即  $\sum_{i=1}^m \mathbf{z}_i = \mathbf{0}$ . 显然, 矩阵  $\mathbf{B}$  的行与列之和均为零, 即  $\sum_{i=1}^m b_{ij} = \sum_{j=1}^m b_{ij} = 0$ . 易知

$$\sum_{i=1}^m dist_{ij}^2 = \text{tr}(\mathbf{B}) + mb_{jj}, \quad (10.4)$$

$$\sum_{j=1}^m dist_{ij}^2 = \text{tr}(\mathbf{B}) + mb_{ii}, \quad (10.5)$$

$$\sum_{i=1}^m \sum_{j=1}^m dist_{ij}^2 = 2m \text{tr}(\mathbf{B}), \quad (10.6)$$

其中  $\text{tr}(\cdot)$  表示矩阵的迹(trace),  $\text{tr}(\mathbf{B}) = \sum_{i=1}^m \|\mathbf{z}_i\|^2$ . 令

$$dist_{i\cdot}^2 = \frac{1}{m} \sum_{j=1}^m dist_{ij}^2, \quad (10.7)$$

$$dist_{\cdot j}^2 = \frac{1}{m} \sum_{i=1}^m dist_{ij}^2, \quad (10.8)$$

$$dist^2 = \frac{1}{m^2} \sum_{i=1}^m \sum_{j=1}^m dist_{ij}^2, \quad (10.9)$$

由式(10.3)和式(10.4)~(10.9)可得

$$b_{ij} = -\frac{1}{2}(dist_{ij}^2 - dist_{i\cdot}^2 - dist_{\cdot j}^2 + dist^2), \quad (10.10)$$

由此即可通过降维前后保持不变的距离矩阵  $\mathbf{D}$  求取内积矩阵  $\mathbf{B}$ .

对矩阵  $\mathbf{B}$  做特征值分解(eigenvalue decomposition),  $\mathbf{B} = \mathbf{V}\Lambda\mathbf{V}^T$ , 其中  $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_d)$  为特征值构成的对角矩阵,  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$ ,  $\mathbf{V}$  为特征向量矩阵. 假定其中有  $d^*$  个非零特征值, 它们构成对角矩阵  $\Lambda_* = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_{d^*})$ , 令  $\mathbf{V}_*$  表示相应的特征向量矩阵, 则  $\mathbf{Z}$  可表达为

$$\mathbf{Z} = \Lambda_*^{1/2} \mathbf{V}_*^T \in \mathbb{R}^{d^* \times m}. \quad (10.11)$$

在现实应用中为了有效降维, 往往仅需降维后的距离与原始空间中的距离尽可能接近, 而不必严格相等. 此时可取  $d' \ll d$  个最大特征值构成对角矩阵  $\tilde{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_{d'})$ , 令  $\tilde{\mathbf{V}}$  表示相应的特征向量矩阵, 则  $\mathbf{Z}$  可表达为

$$\mathbf{Z} = \tilde{\Lambda}^{1/2} \tilde{\mathbf{V}}^T \in \mathbb{R}^{d' \times m}. \quad (10.12)$$

图 10.3 给出了 MDS 算法的描述.

---

**输入:** 距离矩阵  $\mathbf{D} \in \mathbb{R}^{m \times m}$ , 其元素  $dist_{ij}$  为样本  $\mathbf{x}_i$  到  $\mathbf{x}_j$  的距离;  
低维空间维数  $d'$ .

**过程:**

- 1: 根据式(10.7)~(10.9)计算  $dist_{i\cdot}^2, dist_{\cdot j}^2, dist_{\cdot\cdot}^2$ ;
  - 2: 根据式(10.10)计算矩阵  $\mathbf{B}$ ;
  - 3: 对矩阵  $\mathbf{B}$  做特征值分解;
  - 4: 取  $\tilde{\Lambda}$  为  $d'$  个最大特征值所构成的对角矩阵,  $\tilde{\mathbf{V}}$  为相应的特征向量矩阵.
- 输出:** 矩阵  $\tilde{\mathbf{V}}\tilde{\Lambda}^{1/2} \in \mathbb{R}^{m \times d'}$ , 每行是一个样本的低维坐标
- 

图 10.3 MDS 算法

通常令  $d' \ll d$ .

一般来说, 欲获得低维子空间, 最简单的是对原始高维空间进行线性变换. 给定  $d$  维空间中的样本  $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m) \in \mathbb{R}^{d \times m}$ , 变换之后得到  $d' \leq d$  维空间中的样本

$$\mathbf{Z} = \mathbf{W}^T \mathbf{X}, \quad (10.13)$$

其中  $\mathbf{W} \in \mathbb{R}^{d \times d'}$  是变换矩阵,  $\mathbf{Z} \in \mathbb{R}^{d' \times m}$  是样本在新空间中的表达.

变换矩阵  $\mathbf{W}$  可视为  $d'$  个  $d$  维基向量,  $\mathbf{z}_i = \mathbf{W}^T \mathbf{x}_i$  是第  $i$  个样本与这  $d'$  个基向量分别做内积而得到的  $d'$  维属性向量. 换言之,  $\mathbf{z}_i$  是原属性向量  $\mathbf{x}_i$  在新坐标系  $\{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}\}$  中的坐标向量. 若  $\mathbf{w}_i$  与  $\mathbf{w}_j$  ( $i \neq j$ ) 正交, 则新坐标系是一个正交坐标系, 此时  $\mathbf{W}$  为正交变换. 显然, 新空间中的属性是原空间中属性的线性组合.

基于线性变换来进行降维的方法称为线性降维方法, 它们都符合式(10.13)的基本形式, 不同之处是对低维子空间的性质有不同的要求, 相当于对  $\mathbf{W}$  施加了不同的约束. 在下一节我们将会看到, 若要求低维子空间对样本具有最大可分性, 则将得到一种极为常用的线性降维方法.

对降维效果的评估, 通常是比较降维前后学习器的性能, 若性能有所提高则认为降维起到了作用. 若将维数降至二维或三维, 则可通过可视化技术来直观地判断降维效果.

### 10.3 主成分分析

亦称“主分量分析”.

主成分分析(Principal Component Analysis, 简称 PCA)是最常用的一种降维方法. 在介绍 PCA 之前, 不妨先考虑这样一个问题: 对于正交属性空间中

的样本点, 如何用一个超平面(直线的高维推广)对所有样本进行恰当的表达?

容易想到, 若存在这样的超平面, 那么它大概应具有这样的性质:

- 最近重构性: 样本点到这个超平面的距离都足够近;
- 最大可分性: 样本点在这个超平面上的投影能尽可能分开.

有趣的是, 基于最近重构性和最大可分性, 能分别得到主成分分析的两种等价推导. 我们先从最近重构性来推导.

假定数据样本进行了中心化, 即  $\sum_i \mathbf{x}_i = \mathbf{0}$ ; 再假定投影变换后得到的新坐标系为  $\{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_d\}$ , 其中  $\mathbf{w}_i$  是标准正交基向量,  $\|\mathbf{w}_i\|_2 = 1$ ,  $\mathbf{w}_i^T \mathbf{w}_j = 0$  ( $i \neq j$ ). 若丢弃新坐标系中的部分坐标, 即将维度降低到  $d' < d$ , 则样本点  $\mathbf{x}_i$  在低维坐标系中的投影是  $\mathbf{z}_i = (z_{i1}; z_{i2}; \dots; z_{id'})$ , 其中  $z_{ij} = \mathbf{w}_j^T \mathbf{x}_i$  是  $\mathbf{x}_i$  在低维坐标系下第  $j$  维的坐标. 若基于  $\mathbf{z}_i$  来重构  $\mathbf{x}_i$ , 则会得到  $\hat{\mathbf{x}}_i = \sum_{j=1}^{d'} z_{ij} \mathbf{w}_j$ .

考虑整个训练集, 原样本点  $\mathbf{x}_i$  与基于投影重构的样本点  $\hat{\mathbf{x}}_i$  之间的距离为

$$\begin{aligned} \text{const 是一个常数.} \\ \sum_{i=1}^m \left\| \sum_{j=1}^{d'} z_{ij} \mathbf{w}_j - \mathbf{x}_i \right\|_2^2 &= \sum_{i=1}^m \mathbf{z}_i^T \mathbf{z}_i - 2 \sum_{i=1}^m \mathbf{z}_i^T \mathbf{W}^T \mathbf{x}_i + \text{const} \\ &\propto -\text{tr} \left( \mathbf{W}^T \left( \sum_{i=1}^m \mathbf{x}_i \mathbf{x}_i^T \right) \mathbf{W} \right). \end{aligned} \quad (10.14)$$

根据最近重构性, 式(10.14)应被最小化, 考虑到  $\mathbf{w}_j$  是标准正交基,  $\sum_i \mathbf{x}_i \mathbf{x}_i^T$  是协方差矩阵, 有

$$\begin{aligned} \min_{\mathbf{W}} \quad &-\text{tr} (\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W}) \\ \text{s.t. } \quad &\mathbf{W}^T \mathbf{W} = \mathbf{I}. \end{aligned} \quad (10.15)$$

这就是主成分分析的优化目标.

从最大可分性出发, 能得到主成分分析的另一种解释. 我们知道, 样本点  $\mathbf{x}_i$  在新空间中超平面上的投影是  $\mathbf{W}^T \mathbf{x}_i$ , 若所有样本点的投影能尽可能分开, 则应该使投影后样本点的方差最大化, 如图 10.4 所示.

投影后样本点的方差是  $\sum_i \mathbf{W}^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{W}$ , 于是优化目标可写为

$$\begin{aligned} \max_{\mathbf{W}} \quad &\text{tr} (\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W}) \\ \text{s.t. } \quad &\mathbf{W}^T \mathbf{W} = \mathbf{I}, \end{aligned} \quad (10.16)$$

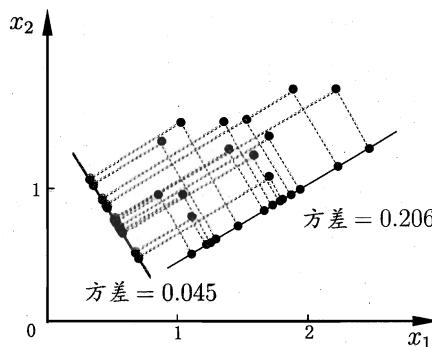


图 10.4 使所有样本的投影尽可能分开(如图中红线所示), 则需最大化投影点的方差

显然, 式(10.16)与(10.15)等价.

对式(10.15)或(10.16)使用拉格朗日乘子法可得

$$\mathbf{X}\mathbf{X}^T\mathbf{W} = \lambda\mathbf{W}, \quad (10.17)$$

实践中常通过对  $\mathbf{X}$  进行奇异值分解来代替协方差矩阵的特征值分解.

PCA 也可看作是逐一选取方差最大方向, 即先对协方差矩阵  $\sum_i \mathbf{x}_i \mathbf{x}_i^T$  做特征值分解, 取最大特征值对应的特征向量  $\mathbf{w}_1$ ; 再对  $\sum_i \mathbf{x}_i \mathbf{x}_i^T - \lambda_1 \mathbf{w}_1 \mathbf{w}_1^T$  做特征值分解, 取最大特征值对应的特征向量  $\mathbf{w}_2$ ; ……由  $\mathbf{W}$  各分量正交及

$$\sum_{i=1}^m \mathbf{x}_i \mathbf{x}_i^T = \sum_{j=1}^d \lambda_j \mathbf{w}_j \mathbf{w}_j^T$$

可知, 上述逐一选取方差最大方向的做法与直接选取最大  $d'$  个特征值等价.

---

输入: 样本集  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ;  
低维空间维数  $d'$ .

过程:

- 1: 对所有样本进行中心化:  $\mathbf{x}_i \leftarrow \mathbf{x}_i - \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i$ ;
- 2: 计算样本的协方差矩阵  $\mathbf{X}\mathbf{X}^T$ ;
- 3: 对协方差矩阵  $\mathbf{X}\mathbf{X}^T$  做特征值分解;
- 4: 取最大的  $d'$  个特征值所对应的特征向量  $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}$ .

输出: 投影矩阵  $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'})$ .

---

图 10.5 PCA 算法

降维后低维空间的维数  $d'$  通常是由用户事先指定, 或通过在  $d'$  值不同的低维空间中对  $k$  近邻分类器(或其他开销较小的学习器) 进行交叉验证来选取较好的  $d'$  值. 对 PCA, 还可从重构的角度设置一个重构阈值, 例如  $t = 95\%$ , 然后选取使下式成立的最小  $d'$  值:

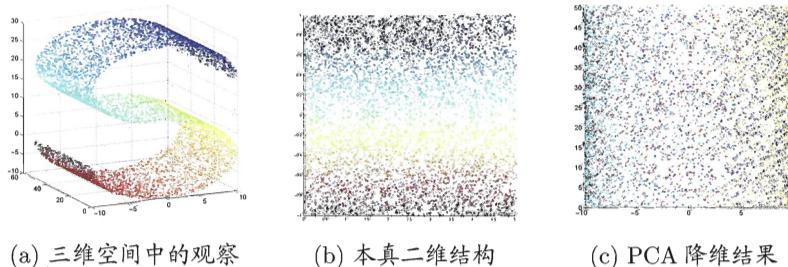
$$\frac{\sum_{i=1}^{d'} \lambda_i}{\sum_{i=1}^d \lambda_i} \geq t. \quad (10.18)$$

保存均值向量是为了通过向量减法对新样本同样进行中心化.

PCA 仅需保留  $\mathbf{W}$  与样本的均值向量即可通过简单的向量减法和矩阵-向量乘法将新样本投影至低维空间中. 显然, 低维空间与原始高维空间必有不同, 因为对应于最小的  $d - d'$  个特征值的特征向量被舍弃了, 这是降维导致的结果. 但舍弃这部分信息往往是必要的: 一方面, 舍弃这部分信息之后能使样本的采样密度增大, 这正是降维的重要动机; 另一方面, 当数据受到噪声影响时, 最小的特征值所对应的特征向量往往与噪声有关, 将它们舍弃能在一定程度上起到去噪的效果.

## 10.4 核化线性降维

线性降维方法假设从高维空间到低维空间的函数映射是线性的, 然而, 在不少现实任务中, 可能需要非线性映射才能找到恰当的低维嵌入. 图 10.6 给出了一个例子, 样本点从二维空间中的矩形区域采样后以 S 形曲面嵌入到三维空间, 若直接使用线性降维方法对三维空间观察到的样本点进行降维, 则将丢失原本的低维结构. 为了对“原本采样的”低维空间与降维后的低维空间加以区别, 我们称前者为“本真”(intrinsic)低维空间.



**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.

参见 6.6 节.

非线性降维的一种常用方法, 是基于核技巧对线性降维方法进行“核化”(kernelized). 下面我们以核主成分分析(Kernelized PCA, 简称 KPCA) [Schölkopf et al., 1998] 为例来进行演示.

假定我们将在高维特征空间中把数据投影到由  $\mathbf{W}$  确定的超平面上, 即 PCA 欲求解

$$\left( \sum_{i=1}^m z_i z_i^T \right) \mathbf{W} = \lambda \mathbf{W}, \quad (10.19)$$

其中  $\mathbf{z}_i$  是样本点  $\mathbf{x}_i$  在高维特征空间中的像. 易知

$$\begin{aligned}\mathbf{W} &= \frac{1}{\lambda} \left( \sum_{i=1}^m \mathbf{z}_i \mathbf{z}_i^T \right) \mathbf{W} = \sum_{i=1}^m \mathbf{z}_i \frac{\mathbf{z}_i^T \mathbf{W}}{\lambda} \\ &= \sum_{i=1}^m \mathbf{z}_i \boldsymbol{\alpha}_i ,\end{aligned}\quad (10.20)$$

其中  $\boldsymbol{\alpha}_i = \frac{1}{\lambda} \mathbf{z}_i^T \mathbf{W}$ . 假定  $\mathbf{z}_i$  是由原始属性空间中的样本点  $\mathbf{x}_i$  通过映射  $\phi$  产生, 即  $\mathbf{z}_i = \phi(\mathbf{x}_i)$ ,  $i = 1, 2, \dots, m$ . 若  $\phi$  能被显式表达出来, 则通过它将样本映射至高维特征空间, 再在特征空间中实施 PCA 即可. 式(10.19)变换为

$$\left( \sum_{i=1}^m \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T \right) \mathbf{W} = \lambda \mathbf{W} ,\quad (10.21)$$

式(10.20)变换为

$$\mathbf{W} = \sum_{i=1}^m \phi(\mathbf{x}_i) \boldsymbol{\alpha}_i .\quad (10.22)$$

一般情形下, 我们不清楚  $\phi$  的具体形式, 于是引入核函数

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) .\quad (10.23)$$

将式(10.22)和(10.23)代入式(10.21)后化简可得

$$\mathbf{K} \mathbf{A} = \lambda \mathbf{A} ,\quad (10.24)$$

其中  $\mathbf{K}$  为  $\kappa$  对应的核矩阵,  $(\mathbf{K})_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ ,  $\mathbf{A} = (\boldsymbol{\alpha}_1; \boldsymbol{\alpha}_2; \dots; \boldsymbol{\alpha}_m)$ . 显然, 式(10.24)是特征值分解问题, 取  $\mathbf{K}$  最大的  $d'$  个特征值对应的特征向量即可.

对新样本  $\mathbf{x}$ , 其投影后的第  $j$  ( $j = 1, 2, \dots, d'$ ) 维坐标为

$$\begin{aligned}z_j &= \mathbf{w}_j^T \phi(\mathbf{x}) = \sum_{i=1}^m \alpha_i^j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) \\ &= \sum_{i=1}^m \alpha_i^j \kappa(\mathbf{x}_i, \mathbf{x}) ,\end{aligned}\quad (10.25)$$

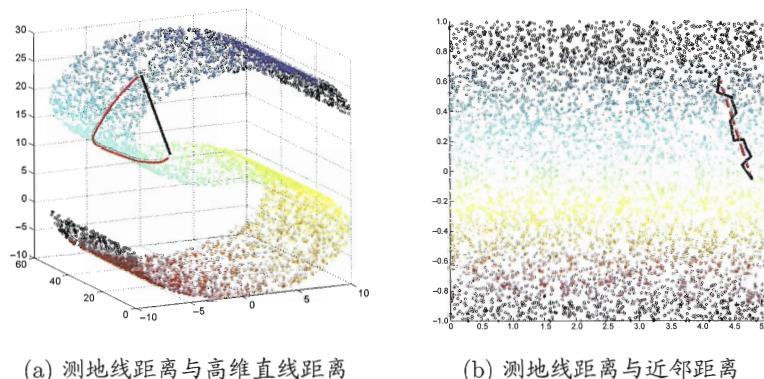
其中  $\boldsymbol{\alpha}_i$  已经过规范化,  $\alpha_i^j$  是  $\boldsymbol{\alpha}_i$  的第  $j$  个分量. 式(10.25)显示出, 为获得投影后的坐标, KPCA 需对所有样本求和, 因此它的计算开销较大.

## 10.5 流形学习

流形学习(manifold learning)是一类借鉴了拓扑流形概念的降维方法。“流形”是在局部与欧氏空间同胚的空间，换言之，它在局部具有欧氏空间的性质，能用欧氏距离来进行距离计算。这给降维方法带来了很大的启发：若低维流形嵌入到高维空间中，则数据样本在高维空间的分布虽然看上去非常复杂，但在局部上仍具有欧氏空间的性质，因此，可以容易地在局部建立降维映射关系，然后再设法将局部映射关系推广到全局。当维数被降至二维或三维时，能对数据进行可视化展示，因此流形学习也可被用于可视化。本节介绍两种著名的流形学习方法。

### 10.5.1 等度量映射

等度量映射(Isometric Mapping, 简称 Isomap) [Tenenbaum et al., 2000] 的基本出发点，是认为低维流形嵌入到高维空间之后，直接在高维空间中计算直线距离具有误导性，因为高维空间中的直线距离在低维嵌入流形上是不可达的。如图 10.7(a)所示，低维嵌入流形上两点间的距离是“测地线”(geodesic)距离：想象一只虫子从一点爬到另一点，如果它不能脱离曲面行走，那么图10.7(a)中的红色曲线是距离最短的路径，即 S 曲面上的测地线，测地线距离是两点之间的本真距离。显然，直接在高维空间中计算直线距离是不恰当的。



**图 10.7** 低维嵌入流形上的测地线距离(红色)不能用高维空间的直线距离计算，但能用近邻距离来近似

那么，如何计算测地线距离呢？这时我们可利用流形在局部上与欧氏空间同胚这个性质，对每个点基于欧氏距离找出其近邻点，然后就能建立一个近邻连接图，图中近邻点之间存在连接，而非近邻点之间不存在连接，于是，计算两

点之间测地线距离的问题, 就转变为计算近邻连接图上两点之间的最短路径问题。从图 10.7(b)可看出, 基于近邻距离逼近能获得低维流形上测地线距离很好的近似。

1972 年图灵奖得主 E. W. Dijkstra 和 1978 年图灵奖得主 R. Floyd 分别提出的著名算法, 参阅数据结构教科书。

在近邻连接图上计算两点间的最短路径, 可采用著名的 Dijkstra 算法或 Floyd 算法, 在得到任意两点的距离之后, 就可通过 10.2 节介绍的 MDS 方法来获得样本点在低维空间中的坐标。图 10.8 给出了 Isomap 算法描述。

---

**输入:** 样本集  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ;  
 近邻参数  $k$ ;  
 低维空间维数  $d'$ .

**过程:**

- 1: **for**  $i = 1, 2, \dots, m$  **do**
- 2:   确定  $\mathbf{x}_i$  的  $k$  近邻;
- 3:    $\mathbf{x}_i$  与  $k$  近邻点之间的距离设置为欧氏距离, 与其他点的距离设置为无穷大;
- 4: **end for**
- 5: 调用最短路径算法计算任意两样本点之间的距离  $\text{dist}(\mathbf{x}_i, \mathbf{x}_j)$ ;
- 6: 将  $\text{dist}(\mathbf{x}_i, \mathbf{x}_j)$  作为 MDS 算法的输入;
- 7: **return** MDS 算法的输出

**输出:** 样本集  $D$  在低维空间的投影  $Z = \{z_1, z_2, \dots, z_m\}$ .

---

MDS 参见 10.2 节。

图 10.8 Isomap 算法

需注意的是, Isomap 仅是得到了训练样本在低维空间的坐标, 对于新样本, 如何将其映射到低维空间呢? 这个问题的常用解决方案, 是将训练样本的高维空间坐标作为输入、低维空间坐标作为输出, 训练一个回归学习器来对新样本的低维空间坐标进行预测。这显然仅是一个权宜之计, 但目前似乎并没有更好的办法。

对近邻图的构建通常有两种做法, 一种是指定近邻点个数, 例如欧氏距离最近的  $k$  个点为近邻点, 这样得到的近邻图称为  $k$  近邻图; 另一种是指定距离阈值  $\epsilon$ , 距离小于  $\epsilon$  的点被认为是近邻点, 这样得到的近邻图称为  $\epsilon$  近邻图。两种方式均有不足, 例如若近邻范围指定得较大, 则距离很远的点可能被误认为近邻, 这样就出现“短路”问题; 近邻范围指定得较小, 则图中有些区域可能与其他区域不存在连接, 这样就出现“断路”问题。短路与断路都会给后续的最短路径计算造成误导。

### 10.5.2 局部线性嵌入

与 Isomap 试图保持近邻样本之间的距离不同, 局部线性嵌入(Locally Linear Embedding, 简称LLE) [Roweis and Saul, 2000] 试图保持邻域内样本之

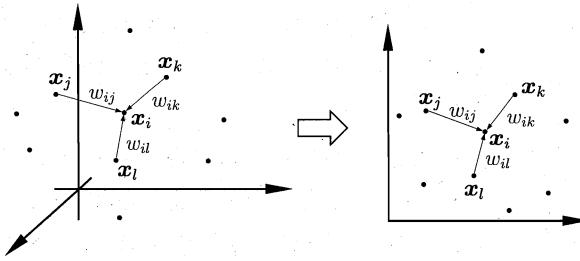


图 10.9 高维空间中的样本重构关系在低维空间中得以保持

间的线性关系。如图 10.9 所示，假定样本点  $\mathbf{x}_i$  的坐标能通过它的邻域样本  $\mathbf{x}_j$ ,  $\mathbf{x}_k$ ,  $\mathbf{x}_l$  的坐标通过线性组合而重构出来，即

$$\mathbf{x}_i = w_{ij}\mathbf{x}_j + w_{ik}\mathbf{x}_k + w_{il}\mathbf{x}_l, \quad (10.26)$$

LLE 希望式(10.26)的关系在低维空间中得以保持。

LLE 先为每个样本  $\mathbf{x}_i$  找到其近邻下标集合  $Q_i$ ，然后计算出基于  $Q_i$  中的样本点对  $\mathbf{x}_i$  进行线性重构的系数  $w_i$ :

$$\begin{aligned} & \min_{w_1, w_2, \dots, w_m} \sum_{i=1}^m \left\| \mathbf{x}_i - \sum_{j \in Q_i} w_{ij} \mathbf{x}_j \right\|_2^2 \\ & \text{s.t. } \sum_{j \in Q_i} w_{ij} = 1, \end{aligned} \quad (10.27)$$

其中  $\mathbf{x}_i$  和  $\mathbf{x}_j$  均为已知，令  $C_{jk} = (\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_k)$ ,  $w_{ij}$  有闭式解

$$w_{ij} = \frac{\sum_{k \in Q_i} C_{jk}^{-1}}{\sum_{l \in Q_i} C_{ls}^{-1}}. \quad (10.28)$$

LLE 在低维空间中保持  $w_i$  不变，于是  $\mathbf{x}_i$  对应的低维空间坐标  $\mathbf{z}_i$  可通过下式求解：

$$\min_{z_1, z_2, \dots, z_m} \sum_{i=1}^m \left\| \mathbf{z}_i - \sum_{j \in Q_i} w_{ij} \mathbf{z}_j \right\|_2^2. \quad (10.29)$$

式(10.27)与(10.29)的优化目标同形，唯一的区别是式(10.27)中需确定的是  $w_i$ ，而式(10.29)中需确定的是  $\mathbf{x}_i$  对应的低维空间坐标  $\mathbf{z}_i$ 。

令  $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m) \in \mathbb{R}^{d' \times m}$ ,  $(\mathbf{W})_{ij} = w_{ij}$ ,

$$\mathbf{M} = (\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W}), \quad (10.30)$$

则式(10.29)可重写为

$$\begin{aligned} & \min_{\mathbf{Z}} \text{tr}(\mathbf{ZMZ}^T), \\ & \text{s.t. } \mathbf{ZZ}^T = \mathbf{I}. \end{aligned} \quad (10.31)$$

式(10.31)可通过特征值分解求解:  $\mathbf{M}$  最小的  $d'$  个特征值对应的特征向量组成的矩阵即为  $\mathbf{Z}^T$ .

LLE 的算法描述如图 10.10 所示. 算法第 4 行显示出: 对于不在样本  $\mathbf{x}_i$  邻域区域的样本  $\mathbf{x}_j$ , 无论其如何变化都对  $\mathbf{x}_i$  和  $\mathbf{z}_i$  没有任何影响; 这种将变动限制在局部的思想在许多地方都有用.

---

**输入:** 样本集  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ;  
近邻参数  $k$ ;  
低维空间维数  $d'$ .  
**过程:**

- 1: **for**  $i = 1, 2, \dots, m$  **do**
- 2:   确定  $\mathbf{x}_i$  的  $k$  近邻;
- 3:   从式(10.27)求得  $w_{ij}, j \in Q_i$ ;
- 4:   对于  $j \notin Q_i$ , 令  $w_{ij} = 0$ ;
- 5: **end for**
- 6: 从式(10.30)得到  $\mathbf{M}$ ;
- 7: 对  $\mathbf{M}$  进行特征值分解;
- 8: **return**  $\mathbf{M}$  的最小  $d'$  个特征值对应的特征向量

**输出:** 样本集  $D$  在低维空间的投影  $Z = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m\}$ .

---

图 10.10 LLE 算法

## 10.6 度量学习

亦称“距离度量学习”  
(distance metric learning).

在机器学习中, 对高维数据进行降维的主要目的是希望找到一个合适的低维空间, 在此空间中进行学习能比原始空间性能更好. 事实上, 每个空间对应了在样本属性上定义的一个距离度量, 而寻找合适的空间, 实质上就是在寻找一个合适的距离度量. 那么, 为何不直接尝试“学习”出一个合适的距离度量呢? 这就是度量学习(metric learning)的基本动机.

欲对距离度量进行学习, 必须有一个便于学习的距离度量表达形式。9.3节给出了很多种距离度量的表达式, 但它们都是“固定的”、没有可调节的参数, 因此不能通过对数据样本的学习来加以改善。为此, 我们先来做一个推广。

即欧氏距离的平方, 这是为了后面推导的便利。

对两个  $d$  维样本  $\mathbf{x}_i$  和  $\mathbf{x}_j$ , 它们之间的平方欧氏距离可写为

$$\text{dist}_{\text{ed}}^2(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = \text{dist}_{ij,1}^2 + \text{dist}_{ij,2}^2 + \dots + \text{dist}_{ij,d}^2, \quad (10.32)$$

其中  $\text{dist}_{ij,k}$  表示  $\mathbf{x}_i$  与  $\mathbf{x}_j$  在第  $k$  维上的距离。若假定不同属性的重要性不同, 则可引入属性权重  $\mathbf{w}$ , 得到

$$\begin{aligned} \text{dist}_{\text{wed}}^2(\mathbf{x}_i, \mathbf{x}_j) &= \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = w_1 \cdot \text{dist}_{ij,1}^2 + w_2 \cdot \text{dist}_{ij,2}^2 + \dots + w_d \cdot \text{dist}_{ij,d}^2 \\ &= (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W} (\mathbf{x}_i - \mathbf{x}_j), \end{aligned} \quad (10.33)$$

其中  $w_i \geq 0$ ,  $\mathbf{W} = \text{diag}(\mathbf{w})$  是一个对角矩阵,  $(\mathbf{W})_{ii} = w_i$ 。

式(10.33)中的  $\mathbf{W}$  可通过学习确定, 但我们还能再往前走一步:  $\mathbf{W}$  的非对角元素均为零, 这意味着坐标轴是正交的, 即属性之间无关; 但现实问题中往往不是这样, 例如考虑西瓜的“重量”和“体积”这两个属性, 它们显然是正相关的, 其对应的坐标轴不再正交。为此, 将式(10.33)中的  $\mathbf{W}$  替换为一个普通的半正定对称矩阵  $\mathbf{M}$ , 于是就得到了马氏距离(Mahalanobis distance)

$$\text{dist}_{\text{mah}}^2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_{\mathbf{M}}^2, \quad (10.34)$$

其中  $\mathbf{M}$  亦称“度量矩阵”, 而度量学习则是对  $\mathbf{M}$  进行学习。注意到为了保持距离非负且对称,  $\mathbf{M}$  必须是(半)正定对称矩阵, 即必有正交基  $\mathbf{P}$  使得  $\mathbf{M}$  能写为  $\mathbf{M} = \mathbf{P} \mathbf{P}^T$ 。

对  $\mathbf{M}$  进行学习当然要设置一个目标。假定我们是希望提高近邻分类器的性能, 则可将  $\mathbf{M}$  直接嵌入到近邻分类器的评价指标中去, 通过优化该性能指标相应地求得  $\mathbf{M}$ 。下面我们以近邻成分分析(Neighbourhood Component Analysis, 简称 NCA) [Goldberger et al., 2005] 为例进行讨论。

近邻分类器在进行判别时通常使用多数投票法, 邻域中的每个样本投 1 票, 邻域外的样本投 0 票。不妨将其替换为概率投票法。对于任意样本  $\mathbf{x}_j$ , 它对  $\mathbf{x}_i$  分类结果影响的概率为

马氏距离以印度数学家 P. C. Mahalanobis 命名。标准马氏距离中  $\mathbf{M}$  是协方差矩阵的逆, 即  $\mathbf{M} = \Sigma^{-1}$ ; 在度量学习中  $\mathbf{M}$  被赋予更大的灵活性。

$$p_{ij} = \frac{\exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|_{\mathbf{M}}^2\right)}{\sum_l \exp\left(-\|\mathbf{x}_i - \mathbf{x}_l\|_{\mathbf{M}}^2\right)}, \quad (10.35)$$

当  $i = j$  时,  $p_{ij}$  最大. 显然,  $\mathbf{x}_j$  对  $\mathbf{x}_i$  的影响随着它们之间距离的增大而减小. 若以留一法 (LOO) 正确率的最大化为目标, 则可计算  $\mathbf{x}_i$  的留一法正确率, 即它被自身之外的所有样本正确分类的概率为

$$p_i = \sum_{j \in \Omega_i} p_{ij}, \quad (10.36)$$

其中  $\Omega_i$  表示与  $\mathbf{x}_i$  属于相同类别的样本的下标集合. 于是, 整个样本集上的留一法正确率为

$$\sum_{i=1}^m p_i = \sum_{i=1}^m \sum_{j \in \Omega_i} p_{ij}. \quad (10.37)$$

将式(10.35)代入(10.37), 再考虑到  $\mathbf{M} = \mathbf{P}\mathbf{P}^T$ , 则 NCA 的优化目标为

$$\min_{\mathbf{P}} 1 - \sum_{i=1}^m \sum_{j \in \Omega_i} \frac{\exp\left(-\|\mathbf{P}^T \mathbf{x}_i - \mathbf{P}^T \mathbf{x}_j\|_2^2\right)}{\sum_l \exp\left(-\|\mathbf{P}^T \mathbf{x}_i - \mathbf{P}^T \mathbf{x}_l\|_2^2\right)}. \quad (10.38)$$

可用随机梯度下降法求解 [Goldberger et al., 2005].

实际上, 我们不仅能把错误率这样的监督学习目标作为度量学习的优化目标, 还能在度量学习中引入领域知识. 例如, 若已知某些样本相似、某些样本不相似, 则可定义“必连”(must-link)约束集合  $\mathcal{M}$  与“勿连”(cannot-link)约束集合  $\mathcal{C}$ ,  $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}$  表示  $\mathbf{x}_i$  与  $\mathbf{x}_j$  相似,  $(\mathbf{x}_i, \mathbf{x}_k) \in \mathcal{C}$  表示  $\mathbf{x}_i$  与  $\mathbf{x}_k$  不相似. 显然, 我们希望相似的样本之间距离较小, 不相似的样本之间距离较大, 于是可通过求解下面这个凸优化问题获得适当的度量矩阵  $\mathbf{M}$  [Xing et al., 2003]:

$$\begin{aligned} \min_{\mathbf{M}} & \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}} \|\mathbf{x}_i - \mathbf{x}_j\|_{\mathbf{M}}^2 \\ \text{s.t. } & \sum_{(\mathbf{x}_i, \mathbf{x}_k) \in \mathcal{C}} \|\mathbf{x}_i - \mathbf{x}_k\|_{\mathbf{M}}^2 \geq 1, \\ & \mathbf{M} \succeq 0, \end{aligned} \quad (10.39)$$

其中约束  $\mathbf{M} \succeq 0$  表明  $\mathbf{M}$  必须是半正定的. 式(10.39)要求在不相似样本间的距离不小于 1 的前提下, 使相似样本间的距离尽可能小.

度量学习自身通常并不要求学得的  $\mathbf{M}$  是低秩的.

不同的度量学习方法针对不同目标获得“好”的半正定对称距离度量矩阵  $\mathbf{M}$ , 若  $\mathbf{M}$  是一个低秩矩阵, 则通过对  $\mathbf{M}$  进行特征值分解, 总能找到一组正交基, 其正交基数目为矩阵  $\mathbf{M}$  的秩  $\text{rank}(\mathbf{M})$ , 小于原属性数  $d$ . 于是, 度量学习学得的结果可衍生出一个降维矩阵  $\mathbf{P} \in \mathbb{R}^{d \times \text{rank}(\mathbf{M})}$ , 能用于降维之目的.

## 10.7 阅读材料

懒惰学习方法主要有  $k$  近邻学习器、懒惰决策树 [Friedman et al., 1996]; 朴素贝叶斯分类器能以懒惰学习方式使用, 也能以急切学习方式使用. 关于懒惰学习的更多内容可参阅 [Aha, 1997].

主成分分析是一种无监督的线性降维方法, 监督线性降维方法最著名的是线性判别分析(LDA) [Fisher, 1936], 参见 3.4 节, 其核化版本 KLDA [Baudat and Anouar, 2000] 参见 6.6 节. 通过最大化两个变量集合之间的相关性, 则可得到“典型相关分析” (Canonical Correlation Analysis, 简称 CCA) [Hotelling, 1936] 及其核化版本 KCCA [Harden et al., 2004], 该方法在多视图学习(multi-view learning) 中有广泛应用. 在模式识别领域人们发现, 直接对矩阵对象(例如一幅图像)进行降维操作会比将其拉伸为向量(例如把图像逐行拼接成一个向量)再进行降维操作有更好的性能, 于是产生了 2DPCA [Yang et al., 2004]、2DLDA [Ye et al., 2005]、 $(2D)^2$ PCA [Zhang and Zhou, 2005] 等方法, 以及基于张量(tensor)的方法 [Kolda and Bader, 2009].

除了 Isomap 和 LLE, 常见的流形学习方法还有拉普拉斯特征映射 (Laplacian Eigenmaps, 简称 LE) [Belkin and Niyogi, 2003]、局部切空间对齐 (Local Tangent Space Alignment, 简称 LTSA) [Zhang and Zha, 2004] 等. 局部保持投影 (Locality Preserving Projections, 简称 LPP) [He and Niyogi, 2004] 是基于 LE 的线性降维方法. 对监督学习而言, 根据类别信息扭曲后的低维空间常比本真低维空间更有利 [Geng et al., 2005]. 值得注意的是, 流形学习欲有效进行邻域保持则需样本密采样, 而这恰是高维情形下面临的重大障碍, 因此流形学习方法在实践中的降维性能往往没有预期的好; 但邻域保持的想法对机器学习的其他分支产生了重要影响, 例如半监督学习中有著名的流形假设、流形正则化 [Belkin et al., 2006]. [Yan et al., 2007] 从图嵌入的角度给出了降维方法的一个统一框架.

参见第 13 章.

半监督聚类见 13.6 节.

将必连关系、勿连关系作为学习任务优化目标的约束, 在半监督聚类的研究中使用得更早 [Wagstaff et al., 2001]. 在度量学习中, 由于这些约束是对所有样本同时发生作用 [Xing et al., 2003], 因此相应的方法被称为全局度量学习方

法。人们也尝试利用局部约束(例如邻域内的三元关系),从而产生了局部距离度量学习方法[Weinberger and Saul, 2009],甚至有一些研究试图为每个样本产生最合适距离度量[Frome et al., 2007; Zhan et al., 2009]。在具体的学习与优化求解方面,不同的度量学习方法往往采用了不同的技术,例如[Yang et al., 2006]将度量学习转化为判别式概率模型框架下基于样本对的二分类问题求解,[Davis et al., 2007]将度量学习转化为信息论框架下的Bregman优化问题,能方便地进行在线学习。

## 习题

西瓜数据集 3.0 $\alpha$  见 p.89  
的表 4.5.

- 10.1** 编程实现  $k$  近邻分类器, 在西瓜数据集 3.0 $\alpha$  上比较其分类边界与决策树分类边界之异同.

- 10.2** 令  $err$ 、 $err^*$  分别表示最近邻分类器与贝叶斯最优分类器的期望错误率, 试证明

$$err^* \leqslant err \leqslant err^* \left( 2 - \frac{|\mathcal{Y}|}{|\mathcal{Y}|-1} \times err^* \right). \quad (10.40)$$

- 10.3** 在对高维数据降维之前应先进行“中心化”, 常见的是将协方差矩阵  $\mathbf{XX}^T$  转化为  $\mathbf{XHH}^T\mathbf{X}^T$ , 其中  $\mathbf{H} = \mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^T$ , 试析其效果.

- 10.4** 在实践中, 协方差矩阵  $\mathbf{XX}^T$  的特征值分解常由中心化后的样本矩阵  $\mathbf{X}$  的奇异值分解代替, 试述其原因.

- 10.5** 降维中涉及的投影矩阵通常要求是正交的. 试述正交、非正交投影矩阵用于降维的优缺点.

princomp 函数调用.

Yale 人脸数据集见  
<http://vision.ucsd.edu/content/yale-face-database>.

- 10.6** 试使用 MATLAB 中的 PCA 函数对 Yale 人脸数据集进行降维, 并观察前 20 个特征向量所对应的图像.

- 10.7** 试述核化线性降维与流形学习之间的联系及优缺点.

- 10.8\***  $k$  近邻图和  $\epsilon$  近邻图存在的短路和断路问题会给 Isomap 造成困扰, 试设计一个方法缓解该问题.

- 10.9\*** 试设计一个方法为新样本找到 LLE 降维后的低维坐标.

参见 9.3 节.

- 10.10** 试述如何确保度量学习产生的距离能满足距离度量的四条基本性质.

## 参考文献

- Aha, D., ed. (1997). *Lazy Learning*. Kluwer, Norwell, MA.
- Baudat, G. and F. Anouar. (2000). "Generalized discriminant analysis using a kernel approach." *Neural Computation*, 12(10):2385–2404.
- Belkin, M. and P. Niyogi. (2003). "Laplacian eigenmaps for dimensionality reduction and data representation." *Neural Computation*, 15(6):1373–1396.
- Belkin, M., P. Niyogi, and V. Sindhwani. (2006). "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples." *Journal of Machine Learning Research*, 7:2399–2434.
- Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.
- Cover, T. M. and P. E. Hart. (1967). "Nearest neighbor pattern classification." *IEEE Transactions on Information Theory*, 13(1):21–27.
- Cox, T. F. and M. A. Cox. (2001). *Multidimensional Scaling*. Chapman & Hall/CRC, London, UK.
- Davis, J. V., B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. (2007). "Information-theoretic metric learning." In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 209–216, Corvalis, OR.
- Fisher, R. A. (1936). "The use of multiple measurements in taxonomic problems." *Annals of Eugenics*, 7(2):179–188.
- Friedman, J. H., R. Kohavi, and Y. Yun. (1996). "Lazy decision trees." In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI)*, 717–724, Portland, OR.
- Frome, A., Y. Singer, and J. Malik. (2007). "Image retrieval and classification using local distance functions." In *Advances in Neural Information Processing Systems 19 (NIPS)* (B. Schölkopf, J. C. Platt, and T. Hoffman, eds.), 417–424, MIT Press, Cambridge, MA.
- Geng, X., D.-C. Zhan, and Z.-H. Zhou. (2005). "Supervised nonlinear dimensionality reduction for visualization and classification." *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 35(6):1098–1107.
- Goldberger, J., G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. (2005). "Neighbourhood components analysis." In *Advances in Neural Information*

- Processing Systems 17 (NIPS)* (L. K. Saul, Y. Weiss, and L. Bottou, eds.), 513–520, MIT Press, Cambridge, MA.
- Harden, D. R., S. Szekely, and J. Shawe-Taylor. (2004). “Canonical correlation analysis: An overview with application to learning methods.” *Neural Computation*, 16(12):2639–2664.
- He, X. and P. Niyogi. (2004). “Locality preserving projections.” In *Advances in Neural Information Processing Systems 16 (NIPS)* (S. Thrun, L. K. Saul, and B. Schölkopf, eds.), 153–160, MIT Press, Cambridge, MA.
- Hotelling, H. (1936). “Relations between two sets of variates.” *Biometrika*, 28 (3-4):321–377.
- Kolda, T. G. and B. W. Bader. (2009). “Tensor decompositions and applications.” *SIAM Review*, 51(3):455–500.
- Roweis, S. T. and L. K. Saul. (2000). “Locally linear embedding.” *Science*, 290 (5500):2323–2316.
- Schölkopf, B., A. Smola, and K.-R. Müller. (1998). “Nonlinear component analysis as a kernel eigenvalue problem.” *Neural Computation*, 10(5):1299–1319.
- Tenenbaum, J. B., V. de Silva, and J. C. Langford. (2000). “A global geometric framework for nonlinear dimensionality reduction.” *Science*, 290(5500): 2319–2323.
- Wagstaff, K., C. Cardie, S. Rogers, and S. Schrödl. (2001). “Constrained  $k$ -means clustering with background knowledge.” In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, 577–584, Williamstown, MA.
- Weinberger, K. Q. and L. K. Saul. (2009). “Distance metric learning for large margin nearest neighbor classification.” *Journal of Machine Learning Research*, 10:207–244.
- Xing, E. P., A. Y. Ng, M. I. Jordan, and S. Russell. (2003). “Distance metric learning, with application to clustering with side-information.” In *Advances in Neural Information Processing Systems 15 (NIPS)* (S. Becker, S. Thrun, and K. Obermayer, eds.), 505–512, MIT Press, Cambridge, MA.
- Yan, S., D. Xu, B. Zhang, and H.-J. Zhang. (2007). “Graph embedding and extensions: A general framework for dimensionality reduction.” *IEEE Trans-*

- actions on Pattern Analysis and Machine Intelligence*, 29(1):40–51.
- Yang, J., D. Zhang, A. F. Frangi, and J.-Y. Yang. (2004). “Two-dimensional PCA: A new approach to appearance-based face representation and recognition.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):131–137.
- Yang, L., R. Jin, R. Sukthankar, and Y. Liu. (2006). “An efficient algorithm for local distance metric learning.” In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 543–548, Boston, MA.
- Ye, J., R. Janardan, and Q. Li. (2005). “Two-dimensional linear discriminant analysis.” In *Advances in Neural Information Processing Systems 17 (NIPS)* (L. K. Saul, Y. Weiss, and L. Bottou, eds.), 1569–1576, MIT Press, Cambridge, MA.
- Zhan, D.-C., Y.-F. Li, and Z.-H. Zhou. (2009). “Learning instance specific distances using metric propagation.” In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 1225–1232, Montreal, Canada.
- Zhang, D. and Z.-H. Zhou. (2005). “(2D)<sup>2</sup>PCA: 2-directional 2-dimensional PCA for efficient face representation and recognition.” *Neurocomputing*, 69 (1-3):224–231.
- Zhang, Z. and H. Zha. (2004). “Principal manifolds and nonlinear dimension reduction via local tangent space alignment.” *SIAM Journal on Scientific Computing*, 26(1):313–338.

## 休息一会儿

### 小故事：主成分分析与卡尔·皮尔逊

主成分分析 (PCA) 是迄今最常用的降维方法，它有许多名字，例如线性代数中的散度矩阵奇异值分解 (SVD)、统计学中的因子分析 (factor analysis)、信号处理中的离散 Karhünén-Loève 变换、图像分析中的 Hotelling 变换、文本分析中的潜在语义分析 (LSA)、机械工程中的本征正交分解 (POD)、气象学中的经验直交函数 (EOF)、结构动力学中的经验模分析 (EMA)、心理测量学中的 Schmidt-Mirsky 定理等。



卡尔·皮尔逊 (Karl Pearson, 1857—1936) 在 1901 年发明了 PCA。皮尔逊是一位罕见的百科全书式的学者，他是统计学家、应用数学家、哲学家、历史学家、民俗学家、宗教学家、人类学家、语言学家，还是社会活动家、教育改革家、作家。1879 年他从剑桥大学国王学院数学系毕业，此后到德国海德堡大学、柏林大学等地游学，涉猎广泛。1884 年他开始在伦敦大学学院 (University College London, 简称 UCL) 担任应用数学讲席教授，39 岁时成为英国皇家学会会士。他在 1892 年出版的科学哲学经典名著《科学的规范》，为爱因斯坦创立相对论提供了启发。皮尔逊对统计学作出了极为重要的贡献，例如他提出了相关系数、标准差、卡方检验、矩估计等，并为假设检验理论、统计决策理论奠定了基础，被尊为“统计学之父”。

Galton 是达尔文的表弟，“优生学”发明人。

皮尔逊开展统计学研究是受到了生物学家 F. Galton 和 W. Welton 的影响，希望使进化论能进行定量描述和分析。1901 年他们三人创立了著名的统计学期刊 *Biometrika*，皮尔逊担任主编直至去世。皮尔逊的独子 Egon 也是著名统计学家，是著名的“奈曼-皮尔逊定理”中的皮尔逊，他子承父业出任 UCL 的统计学教授以及 *Biometrika* 主编，后来担任了英国皇家统计学会主席。

# 第 11 章 特征选择与稀疏学习

## 11.1 子集搜索与评价

我们能用很多属性描述一个西瓜, 例如色泽、根蒂、敲声、纹理、触感等, 但有经验的人往往只需看看根蒂、听听敲声就知道是否好瓜。换言之, 对一个学习任务来说, 给定属性集, 其中有些属性可能很关键、很有用, 另一些属性则可能没什么用。我们将属性称为“特征”(feature), 对当前学习任务有用的属性称为“相关特征”(relevant feature)、没什么用的属性称为“无关特征”(irrelevant feature)。从给定的特征集合中选择出相关特征子集的过程, 称为“特征选择”(feature selection)。

特征选择是一个重要的“数据预处理”(data preprocessing)过程, 在现实机器学习任务中, 获得数据之后通常先进行特征选择, 此后再训练学习器。那么, 为什么要进行特征选择呢?

有两个很重要的原因: 首先, 我们在现实任务中经常会遇到维数灾难问题, 这是由于属性过多而造成的, 若能从中选择出重要的特征, 使得后续学习过程仅需在一部分特征上构建模型, 则维数灾难问题会大为减轻。从这个意义上说, 特征选择与第10章介绍的降维有相似的动机; 事实上, 它们是处理高维数据的两大主流技术。第二个原因是, 去除不相关特征往往会降低学习任务的难度, 这就像侦探破案一样, 若将纷繁复杂的因素抽丝剥茧, 只留下关键因素, 则真相往往更易看清。

需注意的是, 特征选择过程必须确保不丢失重要特征, 否则后续学习过程会因为重要信息的缺失而无法获得好的性能。给定数据集, 若学习任务不同, 则相关特征很可能不同, 因此, 特征选择中所谓的“无关特征”是指与当前学习任务无关。有一类特征称为“冗余特征”(redundant feature), 它们所包含的信息能从其他特征中推演出来。例如, 考虑立方体对象, 若已有特征“底面长”“底面宽”, 则“底面积”是冗余特征, 因为它能从“底面长”与“底面宽”得到。冗余特征很多时候不起作用, 去除它们会减轻学习过程的负担。但有时冗余特征会降低学习任务的难度, 例如若学习目标是估算立方体的体积, 则“底面积”这个冗余特征的存在将使得体积的估算更容易; 更确切地说, 若某个冗余特征恰好对应了完成学习任务所需的“中间概念”, 则该冗余特征是有

益的。为简化讨论，本章暂且假定数据中不涉及冗余特征，并且假定初始的特征集合包含了所有的重要信息。

欲从初始的特征集合中选取一个包含了所有重要信息的特征子集，若没有任何领域知识作为先验假设，那就只好遍历所有可能的子集了；然而这在计算上却是不可行的，因为这样做会遭遇组合爆炸，特征个数稍多就无法进行。可行的做法是产生一个“候选子集”，评价出它的好坏，基于评价结果产生下一个候选子集，再对其进行评价，……这个过程持续进行下去，直至无法找到更好的候选子集为止。显然，这里涉及两个关键环节：如何根据评价结果获取下一个候选特征子集？如何评价候选特征子集的好坏？

亦称子集“生成与搜索”

第一个环节是“子集搜索”(subset search)问题。给定特征集合  $\{a_1, a_2, \dots, a_d\}$ ，我们可将每个特征看作一个候选子集，对这  $d$  个候选单特征子集进行评价，假定  $\{a_2\}$  最优，于是将  $\{a_2\}$  作为第一轮的选定集；然后，在上一轮的选定集中加入一个特征，构成包含两个特征的候选子集，假定在这  $d - 1$  个候选两特征子集中  $\{a_2, a_4\}$  最优，且优于  $\{a_2\}$ ，于是将  $\{a_2, a_4\}$  作为本轮的选定集；……假定在第  $k + 1$  轮时，最优的候选  $(k+1)$  特征子集不如上一轮的选定集，则停止生成候选子集，并将上一轮选定的  $k$  特征集合作为特征选择结果。这样逐渐增加相关特征的策略称为“前向”(forward)搜索。类似的，若我们从完整的特征集合开始，每次尝试去掉一个无关特征，这样逐渐减少特征的策略称为“后向”(backward)搜索。还可将前向与后向搜索结合起来，每一轮逐渐增加选定相关特征(这些特征在后续轮中将确定不会被去除)、同时减少无关特征，这样的策略称为“双向”(bidirectional)搜索。

显然，上述策略都是贪心的，因为它们仅考虑了使本轮选定集最优，例如在第三轮假定选择  $a_5$  优于  $a_6$ ，于是选定集为  $\{a_2, a_4, a_5\}$ ，然而在第四轮却可能是  $\{a_2, a_4, a_6, a_8\}$  比所有的  $\{a_2, a_4, a_5, a_i\}$  都更优。遗憾的是，若不进行穷举搜索，则这样的问题无法避免。

第二个环节是“子集评价”(subset evaluation)问题。给定数据集  $D$ ，假定  $D$  中第  $i$  类样本所占的比例为  $p_i$  ( $i = 1, 2, \dots, |\mathcal{Y}|$ )。为便于讨论，假定样本属性均为离散型。对属性子集  $A$ ，假定根据其取值将  $D$  分成了  $V$  个子集  $\{D^1, D^2, \dots, D^V\}$ ，每个子集中的样本在  $A$  上取值相同，于是我们可计算属性子集  $A$  的信息增益。

$$\text{Gain}(A) = \text{Ent}(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Ent}(D^v), \quad (11.1)$$

假设每个属性有  $v$  个可取值，则  $V = v^{|A|}$ ，这可能是一个很大的值，因此实践中通常是从子集搜索过程中前一轮属性子集的评估价值出发来进行计算。

参见 4.2.1 节.

其中信息熵定义为

$$\text{Ent}(D) = - \sum_{i=1}^{|\mathcal{Y}|} p_k \log_2 p_k , \quad (11.2)$$

信息增益  $\text{Gain}(A)$  越大, 意味着特征子集  $A$  包含的有助于分类的信息越多. 于是, 对每个候选特征子集, 我们可基于训练数据集  $D$  来计算其信息增益, 以此作为评价准则.

更一般的, 特征子集  $A$  实际上确定了对数据集  $D$  的一个划分, 每个划分区域对应着  $A$  上的一个取值, 而样本标记信息  $Y$  则对应着对  $D$  的真实划分, 通过估算这两个划分的差异, 就能对  $A$  进行评价. 与  $Y$  对应的划分的差异越小, 则说明  $A$  越好. 信息熵仅是判断这个差异的一种途径, 其他能判断两个划分差异的机制都能用于特征子集评价.

许多“多样性度量”,  
如不合度量、相关系数等,  
稍加调整即可用于特征子  
集评价, 参见 8.5.2 节.

将特征子集搜索机制与子集评价机制相结合, 即可得到特征选择方法. 例如将前向搜索与信息熵相结合, 这显然与决策树算法非常相似. 事实上, 决策树可用于特征选择, 树结点的划分属性所组成的集合就是选择出的特征子集. 其他的特征选择方法未必像决策树特征选择这么明显, 但它们在本质上都是显式或隐式地结合了某种(或多种)子集搜索机制和子集评价机制.

常见的特征选择方法大致可分为三类: 过滤式(filter)、包裹式(wrapper)和嵌入式(embedding).

## 11.2 过滤式选择

过滤式方法先对数据集进行特征选择, 然后再训练学习器, 特征选择过程与后续学习器无关. 这相当于先用特征选择过程对初始特征进行“过滤”, 再用过滤后的特征来训练模型.

Relief (Relevant Features) [Kira and Rendell, 1992] 是一种著名的过滤式特征选择方法, 该方法设计了一个“相关统计量”来度量特征的重要性. 该统计量是一个向量, 其每个分量分别对应于一个初始特征, 而特征子集的重要性则是由子集中每个特征所对应的相关统计量分量之和来决定. 于是, 最终只需指定一个阈值  $\tau$ , 然后选择比  $\tau$  大的相关统计量分量所对应的特征即可; 也可指定欲选取的特征个数  $k$ , 然后选择相关统计量分量最大的  $k$  个特征.

显然, Relief 的关键是如何确定相关统计量. 给定训练集  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ , 对每个示例  $\mathbf{x}_i$ , Relief 先在  $\mathbf{x}_i$  的同类样本中寻找其最近邻  $\mathbf{x}_{i,\text{nh}}$ , 称为“猜中近邻”(near-hit), 再从  $\mathbf{x}_i$  的异类样本中寻找其最

近邻  $\mathbf{x}_{i,nm}$ , 称为“猜错近邻”(near-miss), 然后, 相关统计量对应于属性  $j$  的分量为

$$\delta^j = \sum_i -\text{diff}(x_i^j, x_{i,nh}^j)^2 + \text{diff}(x_i^j, x_{i,nm}^j)^2, \quad (11.3)$$

其中  $x_a^j$  表示样本  $\mathbf{x}_a$  在属性  $j$  上的取值,  $\text{diff}(x_a^j, x_b^j)$  取决于属性  $j$  的类型: 若属性  $j$  为离散型, 则  $x_a^j = x_b^j$  时  $\text{diff}(x_a^j, x_b^j) = 0$ , 否则为 1; 若属性  $j$  为连续型, 则  $\text{diff}(x_a^j, x_b^j) = |x_a^j - x_b^j|$ , 注意  $x_a^j, x_b^j$  已规范化到  $[0, 1]$  区间.

Relief 中相关统计量的计算已隐然具有距离度量学习的意味. 距离度量学习参见 10.6 节.

从式(11.3)可看出, 若  $\mathbf{x}_i$  与其猜中近邻  $\mathbf{x}_{i,nh}$  在属性  $j$  上的距离小于  $\mathbf{x}_i$  与其猜错近邻  $\mathbf{x}_{i,nm}$  的距离, 则说明属性  $j$  对区分同类与异类样本是有益的, 于是增大属性  $j$  所对应的统计量分量; 反之, 若  $\mathbf{x}_i$  与其猜中近邻  $\mathbf{x}_{i,nh}$  在属性  $j$  上的距离大于  $\mathbf{x}_i$  与其猜错近邻  $\mathbf{x}_{i,nm}$  的距离, 则说明属性  $j$  起负面作用, 于是减小属性  $j$  所对应的统计量分量. 最后, 对基于不同样本得到的估计结果进行平均, 就得到各属性的相关统计量分量, 分量值越大, 则对应属性的分类能力就越强.

式(11.3)中的  $i$  指出了用于平均的样本下标. 实际上 Relief 只需在数据集的采样上而不必在整个数据集上估计相关统计量 [Kira and Rendell, 1992]. 显然, Relief 的时间开销随采样次数以及原始特征数线性增长, 因此是一个运行效率很高的过滤式特征选择算法.

Relief 是为二分类问题设计的, 其扩展变体 Relief-F [Kononenko, 1994] 能处理多分类问题. 假定数据集  $D$  中的样本来自  $|\mathcal{Y}|$  个类别. 对示例  $\mathbf{x}_i$ , 若它属于第  $k$  类 ( $k \in \{1, 2, \dots, |\mathcal{Y}|\}$ ), 则 Relief-F 先在第  $k$  类的样本中寻找  $\mathbf{x}_i$  的最近邻示例  $\mathbf{x}_{i,nh}$  并将其作为猜中近邻, 然后在第  $k$  类之外的每个类中找到一个  $\mathbf{x}_i$  的最近邻示例作为猜错近邻, 记为  $\mathbf{x}_{i,l,nm}$  ( $l = 1, 2, \dots, |\mathcal{Y}|; l \neq k$ ). 于是, 相关统计量对应于属性  $j$  的分量为

$$\delta^j = \sum_i -\text{diff}(x_i^j, x_{i,nh}^j)^2 + \sum_{l \neq k} \left( p_l \times \text{diff}(x_i^j, x_{i,l,nm}^j)^2 \right), \quad (11.4)$$

其中  $p_l$  为第  $l$  类样本在数据集  $D$  中所占的比例.

### 11.3 包裹式选择

与过滤式特征选择不考虑后续学习器不同, 包裹式特征选择直接把最终将要使用的学习器的性能作为特征子集的评价准则. 换言之, 包裹式特征选择的目的就是为给定学习器选择最有利于其性能、“量身定做”的特征子集.

一般而言, 由于包裹式特征选择方法直接针对给定学习器进行优化, 因此

从最终学习器性能来看，包裹式特征选择比过滤式特征选择更好，但另一方面，由于在特征选择过程中需多次训练学习器，因此包裹式特征选择的计算开销通常比过滤式特征选择大得多。

拉斯维加斯方法和蒙特卡罗方法是两个以著名赌城名字命名的随机化方法。两者的主要区别是：若有时间限制，则拉斯维加斯方法或者给出满足要求的解，或者不给出解，而蒙特卡罗方法一定会给出解，虽然给出的解未必满足要求；若无时间限制，则两者都能给出满足要求的解。

初始化。

在特征子集  $A'$  上通过交叉验证估计学习器误差。

若连续  $T$  轮未更新则算法停止。

---

**输入：**数据集  $D$ ;  
特征集  $A$ ;  
学习算法  $\mathfrak{L}$ ;  
停止条件控制参数  $T$ .

**过程：**

```

1:  $E = \infty$ ;
2:  $d = |A|$ ;
3:  $A^* = A$ ;
4:  $t = 0$ ;
5: while  $t < T$  do
6:   随机产生特征子集  $A'$ ;
7:    $d' = |A'|$ ;
8:    $E' = \text{CrossValidation}(\mathfrak{L}(D^{A'}))$ ;
9:   if  $(E' < E) \vee ((E' = E) \wedge (d' < d))$  then
10:     $t = 0$ ;
11:     $E = E'$ ;
12:     $d = d'$ ;
13:     $A^* = A'$ 
14:   else
15:     $t = t + 1$ 
16:   end if
17: end while

```

---

**输出：**特征子集  $A^*$ .

图 11.1 LVW 算法描述

图 11.1 算法第 8 行是通过在数据集  $D$  上，使用交叉验证法来估计学习器  $\mathfrak{L}$  的误差，注意这个误差是在仅考虑特征子集  $A'$  时得到的，即特征子集  $A'$  上的误差，若它比当前特征子集  $A$  上的误差更小，或误差相当但  $A'$  中包含的特征数更少，则将  $A'$  保留下。

需注意的是，由于 LVW 算法中特征子集搜索采用了随机策略，而每次特征子集评价都需训练学习器，计算开销很大，因此算法设置了停止条件控制参数  $T$ 。然而，整个 LVW 算法是基于拉斯维加斯方法框架，若初始特征数很多（即  $|A|$  很大）、 $T$  设置较大，则算法可能运行很长时间都达不到停止条件。换言之，

若有运行时间限制，则有可能给不出解。

#### 11.4 嵌入式选择与 L<sub>1</sub> 正则化

在过滤式和包裹式特征选择方法中，特征选择过程与学习器训练过程有明显的分别；与此不同，嵌入式特征选择是将特征选择过程与学习器训练过程融为一体，两者在同一个优化过程中完成，即在学习器训练过程中自动地进行了特征选择。

给定数据集  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ，其中  $\mathbf{x} \in \mathbb{R}^d, y \in \mathbb{R}$ 。我们考虑最简单的线性回归模型，以平方误差为损失函数，则优化目标为

$$\min_{\mathbf{w}} \sum_{i=1}^m (y_i - \mathbf{w}^\top \mathbf{x}_i)^2. \quad (11.5)$$

正则化参见 6.4 节。

当样本特征很多，而样本数相对较少时，式(11.5)很容易陷入过拟合。为了缓解过拟合问题，可对式(11.5)引入正则化项。若使用 L<sub>2</sub> 范数正则化，则有

$$\min_{\mathbf{w}} \sum_{i=1}^m (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_2^2. \quad (11.6)$$

岭回归最初由 A. Tikhonov 在 1943 年发表于《苏联科学院院刊》，因此亦称“Tikhonov 回归”，而 L<sub>2</sub> 正则化亦称“Tikhonov 正则化”。

其中正则化参数  $\lambda > 0$ 。式(11.6)称为“岭回归”(ridge regression) [Tikhonov and Arsenin, 1977]，通过引入 L<sub>2</sub> 范数正则化，确能显著降低过拟合的风险。

那么，能否将正则化项中的 L<sub>2</sub> 范数替换为 L<sub>p</sub> 范数呢？答案是肯定的。若令  $p = 1$ ，即采用 L<sub>1</sub> 范数，则有

$$\min_{\mathbf{w}} \sum_{i=1}^m (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_1. \quad (11.7)$$

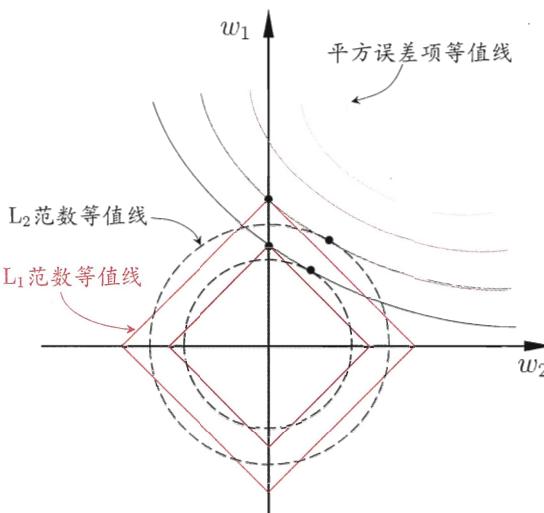
直译为“最小绝对收缩选择算子”，由于比较拗口，因此一般直接称 LASSO。

其中正则化参数  $\lambda > 0$ 。式(11.7)称为 LASSO (Least Absolute Shrinkage and Selection Operator) [Tibshirani, 1996])。

L<sub>1</sub> 范数和 L<sub>2</sub> 范数正则化都有助于降低过拟合风险，但前者还会带来一个额外的好处：它比后者更易于获得“稀疏”(sparse)解，即它求得的  $\mathbf{w}$  会有更少的非零分量。

为了理解这一点，我们来看一个直观的例子：假定  $\mathbf{x}$  仅有两个属性，于是无论式(11.6)还是(11.7)解出的  $\mathbf{w}$  都只有两个分量，即  $w_1, w_2$ ，我们将其作为两个坐标轴，然后在图中绘制出式(11.6)与(11.7)的第一项的“等值线”，即在

事实上，对  $\mathbf{w}$  施加“稀疏约束”(即希望  $\mathbf{w}$  的非零分量尽可能少)最自然的是使用 L<sub>0</sub> 范数，但 L<sub>0</sub> 范数不连续，难以优化求解，因此常使用 L<sub>1</sub> 范数来近似。

图 11.2 L<sub>1</sub> 正则化比 L<sub>2</sub> 正则化更易于得到稀疏解

$(w_1, w_2)$  空间中平方误差项取值相同的点的连线, 再分别绘制出 L<sub>1</sub> 范数与 L<sub>2</sub> 范数的等值线, 即在  $(w_1, w_2)$  空间中 L<sub>1</sub> 范数取值相同的点的连线, 以及 L<sub>2</sub> 范数取值相同的点的连线, 如图 11.2 所示. 式(11.6) 与(11.7) 的解要在平方误差项与正则化项之间折中, 即出现在图中平方误差项等值线与正则化项等值线相交处. 由图 11.2 可看出, 采用 L<sub>1</sub> 范数时平方误差项等值线与正则化项等值线的交点常出现在坐标轴上, 即  $w_1$  或  $w_2$  为 0, 而在采用 L<sub>2</sub> 范数时, 两者的交点常出现在某个象限中, 即  $w_1$  或  $w_2$  均非 0; 换言之, 采用 L<sub>1</sub> 范数比 L<sub>2</sub> 范数更易于得到稀疏解.

即选择出对应于  $\mathbf{w}$  之非零分量的特征.

注意到  $\mathbf{w}$  取得稀疏解意味着初始的  $d$  个特征中仅有对应着  $\mathbf{w}$  的非零分量的特征才会出现在最终模型中, 于是, 求解 L<sub>1</sub> 范数正则化的结果是得到了仅采用一部分初始特征的模型; 换言之, 基于 L<sub>1</sub> 正则化的学习方法就是一种嵌入式特征选择方法, 其特征选择过程与学习器训练过程融为一体, 同时完成.

L<sub>1</sub> 正则化问题的求解可使用近端梯度下降 (Proximal Gradient Descent, 简称 PGD) [Boyd and Vandenberghe, 2004]. 具体来说, 令  $\nabla$  表示微分算子, 对优化目标

$$\min_{\mathbf{x}} f(\mathbf{x}) + \lambda \|\mathbf{x}\|_1, \quad (11.8)$$

若  $f(\mathbf{x})$  可导, 且  $\nabla f$  满足 L-Lipschitz 条件, 即存在常数  $L > 0$  使得

$$\|\nabla f(\mathbf{x}') - \nabla f(\mathbf{x})\|_2^2 \leq L \|\mathbf{x}' - \mathbf{x}\|_2^2 \quad (\forall \mathbf{x}, \mathbf{x}'), \quad (11.9)$$

则在  $\mathbf{x}_k$  附近可将  $f(\mathbf{x})$  通过二阶泰勒展式近似为

$$\begin{aligned}\hat{f}(\mathbf{x}) &\simeq f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle + \frac{L}{2} \|\mathbf{x} - \mathbf{x}_k\|^2 \\ &= \frac{L}{2} \left\| \mathbf{x} - \left( \mathbf{x}_k - \frac{1}{L} \nabla f(\mathbf{x}_k) \right) \right\|_2^2 + \text{const},\end{aligned}\quad (11.10)$$

其中 const 是与  $\mathbf{x}$  无关的常数,  $\langle \cdot, \cdot \rangle$  表示内积. 显然, 式(11.10)的最小值在如下  $\mathbf{x}_{k+1}$  获得:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{1}{L} \nabla f(\mathbf{x}_k). \quad (11.11)$$

于是, 若通过梯度下降法对  $f(\mathbf{x})$  进行最小化, 则每一步梯度下降迭代实际上等价于最小化二次函数  $\hat{f}(\mathbf{x})$ . 将这个思想推广到式(11.8), 则能类似地得到其每一步迭代应为

$$\mathbf{x}_{k+1} = \arg \min_{\mathbf{x}} \frac{L}{2} \left\| \mathbf{x} - \left( \mathbf{x}_k - \frac{1}{L} \nabla f(\mathbf{x}_k) \right) \right\|_2^2 + \lambda \|\mathbf{x}\|_1, \quad (11.12)$$

即在每一步对  $f(\mathbf{x})$  进行梯度下降迭代的同时考虑  $L_1$  范数最小化.

对于式(11.12), 可先计算  $\mathbf{z} = \mathbf{x}_k - \frac{1}{L} \nabla f(\mathbf{x}_k)$ , 然后求解

$$\mathbf{x}_{k+1} = \arg \min_{\mathbf{x}} \frac{L}{2} \|\mathbf{x} - \mathbf{z}\|_2^2 + \lambda \|\mathbf{x}\|_1. \quad (11.13)$$

令  $x^i$  表示  $\mathbf{x}$  的第  $i$  个分量, 将式(11.13)按分量展开可看出, 其中不存在  $x^i x^j$  ( $i \neq j$ ) 这样的项, 即  $\mathbf{x}$  的各分量互不影响, 于是式(11.13)有闭式解

$$x_{k+1}^i = \begin{cases} z^i - \lambda/L, & \lambda/L < z^i; \\ 0, & |z^i| \leq \lambda/L; \\ z^i + \lambda/L, & z^i < -\lambda/L,\end{cases} \quad (11.14)$$

其中  $x_{k+1}^i$  与  $z^i$  分别是  $\mathbf{x}_{k+1}$  与  $\mathbf{z}$  的第  $i$  个分量. 因此, 通过 PGD 能使 LASSO 和其他基于  $L_1$  范数最小化的方法得以快速求解.

## 11.5 稀疏表示与字典学习

不妨把数据集  $D$  考虑成一个矩阵, 其每行对应于一个样本, 每列对应于一个特征. 特征选择所考虑的问题是特征具有“稀疏性”, 即矩阵中的许多列与当前学习任务无关, 通过特征选择去除这些列, 则学习器训练过程仅需在较小

模型涉及的输入因素减少了, 模型所建立的“输入-输出”关系会更清晰.

这里为了用汉语来举例说明, 我们回避了分词问题, 仅谈论汉字.

参见 6.3 节和 12.4 节.

字典亦称“码书”  
(codebook).

字典学习亦称“码书学习”  
(codebook learning).

的矩阵上进行, 学习任务的难度可能有所降低, 涉及的计算和存储开销会减少, 学得模型的可解释性也会提高.

现在我们来考虑另一种稀疏性:  $D$  所对应的矩阵中存在很多零元素, 但这些零元素并不是以整列、整行形式存在的. 在不少现实应用中我们会遇到这样的情形, 例如在文档分类任务中, 通常将每个文档看作一个样本, 每个字(词)作为一个特征, 字(词)在文档中出现的频率或次数作为特征的取值; 换言之,  $D$  所对应的矩阵的每行是一个文档, 每列是一个字(词), 行、列交汇处就是某字(词)在某文档中出现的频率或次数. 那么, 这个矩阵有多少列呢? 以汉语为例, 《康熙字典》中有 47035 个汉字, 这意味着该矩阵可有 4 万多列, 即便仅考虑《现代汉语常用字表》中的汉字, 该矩阵也有 3500 列. 然而, 给定一个文档, 相当多的字是不出现在这个文档中的, 于是矩阵的每一行都有大量的零元素; 对不同的文档, 零元素出现的列往往很不相同.

当样本具有这样的稀疏表达形式时, 对学习任务来说会有不少好处, 例如线性支持向量机之所以能在文本数据上有很好的性能, 恰是由于文本数据在使用上述的字频表示后具有高度的稀疏性, 使大多数问题变得线性可分. 同时, 稀疏样本并不会造成存储上的巨大负担, 因为稀疏矩阵已有很多高效的存储方法.

那么, 若给定数据集  $D$  是稠密的, 即普通非稀疏数据, 能否将其转化为“稀疏表示”(sparse representation)形式, 从而享有稀疏性所带来的好处呢? 需注意的是, 我们所希望的稀疏表示是“恰当稀疏”, 而不是“过度稀疏”. 仍以汉语文档为例, 基于《现代汉语常用字表》得到的可能是恰当稀疏, 即其稀疏性足以让学习任务变得简单可行; 而基于《康熙字典》则可能是过度稀疏, 与前者相比, 也许并未给学习任务带来更多的好处.

显然, 在一般的学习任务中(例如图像分类)并没有《现代汉语常用字表》可用, 我们需学习出这样一个“字典”. 为普通稠密表达的样本找到合适的字典, 将样本转化为合适的稀疏表示形式, 从而使学习任务得以简化, 模型复杂度得以降低, 通常称为“字典学习”(dictionary learning), 亦称“稀疏编码”(sparse coding). 这两个称谓稍有差别, “字典学习”更侧重于学得字典的过程, 而“稀疏编码”则更侧重于对样本进行稀疏表达的过程. 由于两者通常是在同一个优化求解过程中完成的, 因此下面我们将不做进一步区分, 笼统地称为字典学习.

给定数据集  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ , 字典学习最简单的形式为

$$\min_{\mathbf{B}, \boldsymbol{\alpha}_i} \sum_{i=1}^m \|\mathbf{x}_i - \mathbf{B}\boldsymbol{\alpha}_i\|_2^2 + \lambda \sum_{i=1}^m \|\boldsymbol{\alpha}_i\|_1, \quad (11.15)$$

其中  $\mathbf{B} \in \mathbb{R}^{d \times k}$  为字典矩阵,  $k$  称为字典的词汇量, 通常由用户指定,  $\boldsymbol{\alpha}_i \in \mathbb{R}^k$  则是样本  $\mathbf{x}_i \in \mathbb{R}^d$  的稀疏表示. 显然, 式(11.15)的第一项是希望由  $\boldsymbol{\alpha}_i$  能很好地重构  $\mathbf{x}_i$ , 第二项则是希望  $\boldsymbol{\alpha}_i$  尽量稀疏.

与 LASSO 相比, 式(11.15)显然麻烦得多, 因为除了类似于式(11.7)中  $\mathbf{w}$  的  $\boldsymbol{\alpha}_i$ , 还需学习字典矩阵  $\mathbf{B}$ . 不过, 受 LASSO 的启发, 我们可采用变量交替优化的策略来求解式(11.15).

首先在第一步, 我们固定住字典  $\mathbf{B}$ , 若将式(11.15)按分量展开, 可看出其中不涉及  $\alpha_i^u \alpha_i^v$  ( $u \neq v$ ) 这样的交叉项, 于是可参照 LASSO 的解法求解下式, 从而为每个样本  $\mathbf{x}_i$  找到相应的  $\boldsymbol{\alpha}_i$ :

$$\min_{\boldsymbol{\alpha}_i} \|\mathbf{x}_i - \mathbf{B}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1. \quad (11.16)$$

在第二步, 我们固定住  $\boldsymbol{\alpha}_i$  来更新字典  $\mathbf{B}$ , 此时可将式(11.15)写为

$$\min_{\mathbf{B}} \|\mathbf{X} - \mathbf{BA}\|_F^2, \quad (11.17)$$

其中  $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m) \in \mathbb{R}^{d \times m}$ ,  $\mathbf{A} = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \dots, \boldsymbol{\alpha}_m) \in \mathbb{R}^{k \times m}$ ,  $\|\cdot\|_F$  是矩阵的 Frobenius 范数. 式(11.17)有多种求解方法, 常用的有基于逐列更新策略的 KSVD [Aharon et al., 2006]. 令  $\mathbf{b}_i$  表示字典矩阵  $\mathbf{B}$  的第  $i$  列,  $\boldsymbol{\alpha}^i$  表示稀疏矩阵  $\mathbf{A}$  的第  $i$  行, 式(11.17)可重写为

$$\begin{aligned} \min_{\mathbf{B}} \|\mathbf{X} - \mathbf{BA}\|_F^2 &= \min_{\mathbf{b}_i} \left\| \mathbf{X} - \sum_{j=1}^k \mathbf{b}_j \boldsymbol{\alpha}^j \right\|_F^2 \\ &= \min_{\mathbf{b}_i} \left\| \left( \mathbf{X} - \sum_{j \neq i} \mathbf{b}_j \boldsymbol{\alpha}^j \right) - \mathbf{b}_i \boldsymbol{\alpha}^i \right\|_F^2 \\ &= \min_{\mathbf{b}_i} \|\mathbf{E}_i - \mathbf{b}_i \boldsymbol{\alpha}^i\|_F^2. \end{aligned} \quad (11.18)$$

在更新字典的第  $i$  列时, 其他各列都是固定的, 因此  $\mathbf{E}_i = \sum_{j \neq i} \mathbf{b}_j \boldsymbol{\alpha}^j$  是固定的, 于是最小化式(11.18)原则上只需对  $\mathbf{E}_i$  进行奇异值分解以取得最大奇异值所对应的正交向量. 然而, 直接对  $\mathbf{E}_i$  进行奇异值分解会同时修改  $\mathbf{b}_i$  和  $\boldsymbol{\alpha}^i$ , 从而可能破坏  $\mathbf{A}$  的稀疏性. 为避免发生这种情况, KSVD 对  $\mathbf{E}_i$  和  $\boldsymbol{\alpha}^i$  进行专门处理:  $\boldsymbol{\alpha}^i$  仅保留非零元素,  $\mathbf{E}_i$  则仅保留  $\mathbf{b}_i$  与  $\boldsymbol{\alpha}^i$  的非零元素的乘积项, 然后再进行奇异值分解, 这样就保持了第一步所得到的稀疏性.

初始化字典矩阵  $\mathbf{B}$  之后反复迭代上述两步, 最终即可求得字典  $\mathbf{B}$  和样本  $\mathbf{x}_i$  的稀疏表示  $\alpha_i$ . 在上述字典学习过程中, 用户能通过设置词汇量  $k$  的大小来控制字典的规模, 从而影响到稀疏程度.

## 11.6 压缩感知

奈奎斯特采样定理提供了信号恢复的充分条件而非必要条件.

亦称 compressive sensing.

在现实任务中, 我们常希望根据部分信息来恢复全部信息. 例如在数据通讯中要将模拟信号转换为数字信号, 根据奈奎斯特 (Nyquist) 采样定理, 令采样频率达到模拟信号最高频率的两倍, 则采样后的数字信号就保留了模拟信号的全部信息; 换言之, 由此获得的数字信号能精确重构原模拟信号. 然而, 为了便于传输、存储, 在实践中人们通常对采样的数字信号进行压缩, 这有可能损失一些信息, 而在信号传输过程中, 由于信道出现丢包等问题, 又可能损失部分信息. 那么, 接收方基于收到的信号, 能否精确地重构出原信号呢? 压缩感知 (compressed sensing) [Donoho, 2006; Candès et al., 2006] 为解决此类问题提供了新的思路.

假定有长度为  $m$  的离散信号  $\mathbf{x}$ , 不妨假定我们以远小于奈奎斯特采样定理要求的采样率进行采样, 得到长度为  $n$  的采样后信号  $\mathbf{y}$ ,  $n \ll m$ , 即

$$\mathbf{y} = \Phi \mathbf{x}, \quad (11.19)$$

其中  $\Phi \in \mathbb{R}^{n \times m}$  是对信号  $\mathbf{x}$  的测量矩阵, 它确定了以什么频率进行采样以及如何将采样样本组成采样后的信号.

在已知离散信号  $\mathbf{x}$  和测量矩阵  $\Phi$  时要得到测量值  $\mathbf{y}$  很容易, 然而, 若将测量值和测量矩阵传输出去, 接收方能还原出原始信号  $\mathbf{x}$  吗?

一般来说, 答案是 “No”, 这是由于  $n \ll m$ , 因此  $\mathbf{y}$ ,  $\mathbf{x}$ ,  $\Phi$  组成的式(11.19)是一个欠定方程, 无法轻易求出数值解.

假定  $\mathbf{x}$  本身不是稀疏的.

现在不妨假设存在某个线性变换  $\Psi \in \mathbb{R}^{m \times m}$ , 使得  $\mathbf{x}$  可表示为  $\Psi \mathbf{s}$ , 于是  $\mathbf{y}$  可表示为

$$\mathbf{y} = \Phi \Psi \mathbf{s} = \mathbf{A} \mathbf{s}, \quad (11.20)$$

其中  $\mathbf{A} = \Phi \Psi \in \mathbb{R}^{n \times m}$ . 于是, 若能根据  $\mathbf{y}$  恢复出  $\mathbf{s}$ , 则可通过  $\mathbf{x} = \Psi \mathbf{s}$  来恢复出信号  $\mathbf{x}$ .

粗看起来式(11.20)没有解决任何问题, 因为式(11.20)中恢复信号  $\mathbf{s}$  这个逆问题仍是欠定的. 然而有趣的是, 若  $\mathbf{s}$  具有稀疏性, 则这个问题竟能很好地得

以解决！这是因为稀疏性使得未知因素的影响大为减少。此时式(11.20)中的  $\Psi$  称为稀疏基，而  $\mathbf{A}$  的作用则类似于字典，能将信号转换为稀疏表示。

事实上，在很多应用中均可获得具有稀疏性的  $\mathbf{s}$ ，例如图像或声音的数字信号通常在时域上不具有稀疏性，但经过傅里叶变换、余弦变换、小波变换等处理后却会转化为频域上的稀疏信号。

显然，与特征选择、稀疏表示不同，压缩感知关注的是如何利用信号本身所具有的稀疏性，从部分观测样本中恢复原信号。通常认为，压缩感知分为“感知测量”和“重构恢复”这两个阶段。“感知测量”关注如何对原始信号进行处理以获得稀疏样本表示，这方面的内容涉及傅里叶变换、小波变换以及 11.5 节介绍的字典学习、稀疏编码等，不少技术在压缩感知提出之前就已在信号处理等领域有很多研究；“重构恢复”关注的是如何基于稀疏性从少量观测中恢复原信号，这是压缩感知的精髓，当我们谈到压缩感知时，通常是指该部分。

压缩感知的相关理论比较复杂，下面仅简要介绍一下“限定等距性”(Restricted Isometry Property，简称 RIP) [Candès, 2008]。

对大小为  $n \times m$  ( $n \ll m$ ) 的矩阵  $\mathbf{A}$ ，若存在常数  $\delta_k \in (0, 1)$  使得对于任意向量  $\mathbf{s}$  和  $\mathbf{A}$  的所有子矩阵  $\mathbf{A}_k \in \mathbb{R}^{n \times k}$  有

$$(1 - \delta_k) \|\mathbf{s}\|_2^2 \leq \|\mathbf{A}_k \mathbf{s}\|_2^2 \leq (1 + \delta_k) \|\mathbf{s}\|_2^2, \quad (11.21)$$

则称  $\mathbf{A}$  满足  $k$  限定等距性 ( $k$ -RIP)。此时可通过下面的优化问题近乎完美地从  $\mathbf{y}$  中恢复出稀疏信号  $\mathbf{s}$ ，进而恢复出  $\mathbf{x}$ ：

$$\min_{\mathbf{s}} \|\mathbf{s}\|_0 \quad (11.22)$$

$$\text{s.t. } \mathbf{y} = \mathbf{A}\mathbf{s}.$$

然而，式(11.22)涉及  $L_0$  范数最小化，这是个 NP 难问题。值得庆幸的是， $L_1$  范数最小化在一定条件下与  $L_0$  范数最小化问题共解 [Candès et al., 2006]，于是实际上只需关注

$$\min_{\mathbf{s}} \|\mathbf{s}\|_1 \quad (11.23)$$

$$\text{s.t. } \mathbf{y} = \mathbf{A}\mathbf{s}.$$

这样，压缩感知问题就可通过  $L_1$  范数最小化问题求解，例如式(11.23)可转化为

LASSO 的等价形式再通过近端梯度下降法求解, 即使用“基寻踪去噪”(Basis Pursuit De-Noising) [Chen et al., 1998].

这是一个典型的“协同过滤”(collaborative filtering)任务.

基于部分信息来恢复全部信息的技术在许多现实任务中有重要应用. 例如网上书店通过收集读者在网上对书的评价, 可根据读者的读书偏好来进行新书推荐, 从而达到定向广告投放的效果. 显然, 没有哪位读者读过所有的书, 也没有哪本书被所有读者读过, 因此, 网上书店所搜集到的仅有部分信息. 例如表 11.1 给出了四位读者的网上评价信息, 这里评价信息经过处理, 形成了“喜好程度”评分(5 分最高). 由于读者仅对读过的书给出评价, 因此表中出现了很多未知项“?”.

表 11.1 客户对书的喜好程度评分

	《笑傲江湖》	《万历十五年》	《人间词话》	《云海玉弓缘》	《人类的故事》
赵大	5	?	?	3	2
钱二	?	5	3	?	5
孙三	5	3	?	?	?
李四	3	?	5	4	?

那么, 能否将表 11.1 中通过读者评价得到的数据当作部分信号, 基于压缩感知的思想恢复出完整信号呢?

我们知道, 能通过压缩感知技术恢复欠采样信号的前提条件之一是信号有稀疏表示. 读书喜好数据是否存在稀疏表示呢? 答案是肯定的. 一般情形下, 读者对书籍的评价取决于题材、作者、装帧等多种因素, 为简化讨论, 假定表 11.1 中的读者喜好评分仅与题材有关. 《笑傲江湖》和《云海玉弓缘》是武侠小说, 《万历十五年》和《人类的故事》是历史读物, 《人间词话》属于诗词文学. 一般来说, 相似题材的书籍会有相似的读者, 若能将书籍按题材归类, 则题材总数必然远远少于书籍总数, 因此从题材的角度来看, 表 11.1 中反映出的信号应该是稀疏的. 于是, 应能通过类似压缩感知的思想加以处理.

亦称“低秩矩阵恢复”.

矩阵补全(matrix completion)技术[Candès and Recht, 2009]可用于解决这个问题, 其形式为

$$\min_{\mathbf{X}} \text{rank}(\mathbf{X}) \quad (11.24)$$

$$\text{s.t. } (\mathbf{X})_{ij} = (\mathbf{A})_{ij}, \quad (i, j) \in \Omega,$$

其中,  $\mathbf{X}$  表示需恢复的稀疏信号;  $\text{rank}(\mathbf{X})$  表示矩阵  $\mathbf{X}$  的秩;  $\mathbf{A}$  是如表 11.1 的

读者评分矩阵这样的已观测信号;  $\Omega$  是  $\mathbf{A}$  中非“?”元素  $(\mathbf{A})_{ij}$  的下标  $(i, j)$  的集合. 式(11.24)的约束项明确指出, 恢复出的矩阵中  $(\mathbf{X})_{ij}$  应当与已观测到的对应元素相同.

核范数亦称“迹范数”  
(trace norm).

与式(11.22)相似, 式(11.24)也是一个 NP-难问题. 注意到  $\text{rank}(\mathbf{X})$  在集合  $\{\mathbf{X} \in \mathbb{R}^{m \times n} : \|\mathbf{X}\|_F^2 \leq 1\}$  上的凸包是  $\mathbf{X}$  的“核范数”(nuclear norm):

$$\|\mathbf{X}\|_* = \sum_{j=1}^{\min\{m,n\}} \sigma_j(\mathbf{X}), \quad (11.25)$$

其中  $\sigma_j(\mathbf{X})$  表示  $\mathbf{X}$  的奇异值, 即矩阵的核范数为矩阵的奇异值之和, 于是可通过最小化矩阵核范数来近似求解式(11.24), 即

$$\begin{aligned} & \min_{\mathbf{X}} \|\mathbf{X}\|_* \\ & \text{s.t. } (\mathbf{X})_{ij} = (\mathbf{A})_{ij}, \quad (i, j) \in \Omega. \end{aligned} \quad (11.26)$$

SDP 参见附录 B.3.

式(11.26)是一个凸优化问题, 可通过半正定规划(Semi-Definite Programming, 简称 SDP)求解. 理论研究表明, 在满足一定条件时, 若  $\mathbf{A}$  的秩为  $r$ ,  $n \ll m$ , 则只需观察到  $O(mr \log^2 m)$  个元素就能完美恢复出  $\mathbf{A}$  [Recht, 2011].

## 11.7 阅读材料

特征选择是机器学习中研究最早的分支领域之一, 早期研究主要是按特征子集“生成与搜索-评价”过程进行. 在子集生成与搜索方面引入了很多人工智能搜索技术, 如分支限界法[Narendra and Fukunaga, 1977]、浮动搜索法[Pudil et al., 1994]等; 在子集评价方面则采用了很多源于信息论的准则, 如信息熵、AIC(Akaike Information Criterion)[Akaike, 1974]等.[Blum and Langley, 1997]对子集评价准则进行了讨论,[Forman, 2003]则进行了很多实验比较.

早期特征选择方法主要是过滤式的, 包裹式方法出现稍晚[Kohavi and John, 1997], 嵌入式方法事实上更晚[Weston et al., 2003], 但由于决策树算法在构建树的同时也可看作进行了特征选择, 因此嵌入式方法也可追溯到 ID3[Quinlan, 1986]. 有很多文献对特征选择方法的性能进行了实验比较[Yang and Pederson, 1997; Jain and Zongker, 1997]. 更多关于特征选择的内容可参阅[Guyon and Elisseeff, 2003; Liu et al., 2010], 以及专门关于特征选择的书籍

[Liu and Motoda, 1998, 2007].

直译为“最小角回归”，  
通常直接称 LARS.

LARS (Least Angle RegresSion) [Efron et al., 2004] 是一种嵌入式特征选择方法，它基于线性回归平方误差最小化，每次选择一个与残差相关性最大的特征。LASSO [Tibshirani, 1996] 可通过对 LARS 稍加修改而实现。在 LASSO 基础上进一步发展出考虑特征分组结构的 Group LASSO [Yuan and Lin, 2006]、考虑特征序结构的 Fused LASSO [Tibshirani et al., 2005] 等变体。由于凸性不严格，LASSO 类方法可能产生多个解，该问题通过弹性网 (Elastic Net) 得以解决 [Zou and Hastie, 2005]。

仍以汉语文档为例，一个概念可能由多个字词来表达，这些字词就构成了一个分组；若这个概念在文档中没有出现，则这整个分组所对应的变量都将为零。

对字典学习与稀疏编码 [Aharon et al., 2006]，除了通过控制字典规模从而影响稀疏性，有时还希望控制字典的“结构”，例如假设字典具有“分组结构”，即同一个分组内的变量或同为非零，或同为零。这样的性质称为“分组稀疏性” (group sparsity)，相应的稀疏编码方法则称为分组稀疏编码(group sparse coding) [Bengio et al., 2009]。稀疏编码和分组稀疏编码在图像特征抽取方面有很多应用，可参阅 [Mairal et al., 2008; Wang et al., 2010]。

压缩感知 [Donoho, 2006; Candès et al., 2006] 直接催生了人脸识别的鲁棒主成分分析 [Candès et al., 2011] 和基于矩阵补全的协同过滤 [Recht et al., 2010]。[Baraniuk, 2007] 是关于压缩感知的一个简短介绍。将  $L_0$  范数最小化转化为  $L_1$  范数最小化后，常用求解方法除了转化为 LASSO 的基寻踪去噪，还可使用基寻踪 (Basis Pursuit) [Chen et al., 1998]、匹配寻踪 (Matching Pursuit) [Mallat and Zhang, 1993] 等。[Liu and Ye, 2009] 使用投影法快速求解稀疏学习问题，并提供了一个稀疏学习程序包 SLEP (<http://www.yelab.net/software/SLEP/>)。

## 习题

西瓜数据集 3.0 见 p.84  
表 4.3.

- 11.1** 试编程实现 Relief 算法, 并考察其在西瓜数据集 3.0 上的运行结果.
- 11.2** 试写出 Relief-F 的算法描述.
- 11.3** Relief 算法是分别考察每个属性的重要性. 试设计一个能考虑每一对属性重要性的改进算法.
- 11.4** 试为 LVW 设计一个改进算法, 即便有运行时间限制, 该算法也一定能给出解.
- 11.5** 结合图 11.2, 试举例说明  $L_1$  正则化在何种情形下不能产生稀疏解.
- 11.6** 试析岭回归与支持向量机的联系.
- 11.7** 试述直接求解  $L_0$  范数正则化会遇到的困难.
- 11.8** 试给出求解  $L_1$  范数最小化问题中的闭式解(11.14)的详细推导过程.
- 11.9** 试述字典学习与压缩感知对稀疏性利用的异同.
- 11.10\*** 试改进式(11.15), 以学习出具有分组稀疏性的字典.

## 参考文献

- Aharon, M., M. Elad, and A. Bruckstein. (2006). “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation.” *IEEE Transactions on Image Processing*, 54(11):4311–4322.
- Akaike, H. (1974). “A new look at the statistical model identification.” *IEEE Transactions on Automatic Control*, 19(6):716–723.
- Baraniuk, R. G. (2007). “Compressive sensing.” *IEEE Signal Processing Magazine*, 24(4):118–121.
- Bengio, S., F. Pereira, Y. Singer, and D. Strelow. (2009). “Group sparse coding.” In *Advances in Neural Information Processing Systems 22 (NIPS)* (Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, eds.), 82–89, MIT Press, Cambridge, MA.
- Blum, A. and P. Langley. (1997). “Selection of relevant features and examples in machine learning.” *Artificial Intelligence*, 97(1-2):245–271.
- Boyd, S. and L. Vandenberghe. (2004). *Convex Optimization*. Cambridge University Press, Cambridge, UK.
- Candès, E. J. (2008). “The restricted isometry property and its implications for compressed sensing.” *Comptes Rendus Mathematique*, 346(9-10):589–592.
- Candès, E. J., X. Li, Y. Ma, and J. Wright. (2011). “Robust principal component analysis?” *Journal of the ACM*, 58(3):Article 11.
- Candès, E. J. and B. Recht. (2009). “Exact matrix completion via convex optimization.” *Foundations of Computational Mathematics*, 9(6):717–772.
- Candès, E. J., J. Romberg, and T. Tao. (2006). “Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information.” *IEEE Transactions on Information Theory*, 52(2):489–509.
- Chen, S. S., D. L. Donoho, and M. A. Saunders. (1998). “Atomic decomposition by basis pursuit.” *SIAM Journal on Scientific Computing*, 20(1):33–61.
- Donoho, D. L. (2006). “Compressed sensing.” *IEEE Transactions on Information Theory*, 52(4):1289–1306.
- Efron, B., T. Hastie, I. Johnstone, and R. Tibshirani. (2004). “Least angle regression.” *Annals of Statistics*, 32(2):407–499.

- Forman, G. (2003). "An extensive empirical study of feature selection metrics for text classification." *Journal of Machine Learning Research*, 3:1289–1305.
- Guyon, I. and A. Elisseeff. (2003). "An introduction to variable and feature selection." *Journal of Machine Learning Research*, 3:1157–1182.
- Jain, A. and D. Zongker. (1997). "Feature selection: Evaluation, application, and small sample performance." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158.
- Kira, K. and L. A. Rendell. (1992). "The feature selection problem: Traditional methods and a new algorithm." In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI)*, 129–134, San Jose, CA.
- Kohavi, R. and G. H. John. (1997). "Wrappers for feature subset selection." *Artificial Intelligence*, 97(1-2):273–324.
- Kononenko, I. (1994). "Estimating attributes: Analysis and extensions of RELIEF." In *Proceedings of the 7th European Conference on Machine Learning (ECML)*, 171–182, Catania, Italy.
- Liu, H. and H. Motoda. (1998). *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer, Boston, MA.
- Liu, H. and H. Motoda. (2007). *Computational Methods of Feature Selection*. Chapman & Hall/CRC, Boca Raton, FL.
- Liu, H., H. Motoda, R. Setiono, and Z. Zhao. (2010). "Feature selection: An ever evolving frontier in data mining." In *Proceedings of the 4th Workshop on Feature Selection in Data Mining (FSDM)*, 4–13, Hyderabad, India.
- Liu, H. and R. Setiono. (1996). "Feature selection and classification — a probabilistic wrapper approach." In *Proceedings of the 9th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE)*, 419–424, Fukuoka, Japan.
- Liu, J. and J. Ye. (2009). "Efficient Euclidean projections in linear time." In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 657–664, Montreal, Canada.
- Mairal, J., M. Elad, and G. Sapiro. (2008). "Sparse representation for color image restoration." *IEEE Transactions on Image Processing*, 17(1):53–69.
- Mallat, S. G. and Z. F. Zhang. (1993). "Matching pursuits with time-frequency

- dictionaries.” *IEEE Transactions on Signal Processing*, 41(12):3397–3415.
- Narendra, P. M. and K. Fukunaga. (1977). “A branch and bound algorithm for feature subset selection.” *IEEE Transactions on Computers*, C-26(9): 917–922.
- Pudil, P., J. Novovičová, and J. Kittler. (1994). “Floating search methods in feature selection.” *Pattern Recognition Letters*, 15(11):1119–1125.
- Quinlan, J. R. (1986). “Induction of decision trees.” *Machine Learning*, 1(1): 81–106.
- Recht, B. (2011). “A simpler approach to matrix completion.” *Journal of Machine Learning Research*, 12:3413–3430.
- Recht, B., M. Fazel, and P. Parrilo. (2010). “Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization.” *SIAM Review*, 52(3):471–501.
- Tibshirani, R. (1996). “Regression shrinkage and selection via the LASSO.” *Journal of the Royal Statistical Society - Series B*, 58(1):267–288.
- Tibshirani, R., M. Saunders, S. Rosset, J. Zhu, and K. Knight. (2005). “Sparsity and smoothness via the fused LASSO.” *Journal of the Royal Statistical Society - Series B*, 67(1):91–108.
- Tikhonov, A. N. and V. Y. Arsenin, eds. (1977). *Solution of Ill-Posed Problems*. Winston, Washington, DC.
- Wang, J., J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. (2010). “Locality-constrained linear coding for image classification.” In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 3360–3367, San Francisco, CA.
- Weston, J., A. Elisseeff, B. Schölkopf, and M. Tipping. (2003). “Use of the zero norm with linear models and kernel methods.” *Journal of Machine Learning Research*, 3:1439–1461.
- Yang, Y. and J. O. Pederson. (1997). “A comparative study on feature selection in text categorization.” In *Proceedings of the 14th International Conference on Machine Learning (ICML)*, 412–420, Nashville, TN.
- Yuan, M. and Y. Lin. (2006). “Model selection and estimation in regression with grouped variables.” *Journal of the Royal Statistical Society - Series B*,

68(1):49–67.

Zou, H. and T. Hastie. (2005). “Regularization and variable selection via the elastic net.” *Journal of the Royal Statistical Society - Series B*, 67(2):301–320.

## 休息一会儿

### 小故事：蒙特卡罗方法与斯坦尼斯拉夫·乌拉姆

斯坦尼斯拉夫·乌拉姆 (Stanisław Ulam, 1909–1984) 是著名的波兰犹太裔数学家，在遍历论、数论、集合论等方面都有重要贡献，“乌拉姆数列”就是以他的名字命名的。



乌拉姆出生于奥匈帝国利沃夫，1933年在波兰利沃夫理工学院获得数学博士学位，然后于1935年应冯·诺伊曼的邀请到普林斯顿高等研究院访问，1940年他在威斯康星大学麦迪逊分校获得教职，翌年加入美国籍。1943年起他参与“曼哈顿计划”并做出重大贡献；当前世界上绝大部分核武器所使用的设计方案“泰勒-乌拉姆方案”就是以他和“氢弹之父”爱德华·泰勒的名字命名的。

世界上最早的通用电子计算机之一——ENIAC 在发明后即被用于曼哈顿计划，乌拉姆敏锐地意识到在计算机的帮助下，可通过重复数百次模拟过程的方式来对概率变量进行统计估计。冯·诺伊曼立即认识到这个想法的重要性并给予支持。1947年乌拉姆提出这种统计方法并用于计算核裂变的连锁反应。由于乌拉姆常说他的叔叔又在蒙特卡罗赌场输钱了，因此他的同事 Nicolas Metropolis 戏称该方法为“蒙特卡罗”，不料却流传开去。

利沃夫(Lviv)在历史上先属于波兰，1867—1918年属于奥匈帝国，第一次世界大战后回归波兰，1939年划入前苏联的乌克兰，现为乌克兰利沃夫州首府。

冯·诺伊曼和爱德华·泰勒都出生在匈牙利。

蒙特卡罗方法的著名代表 Metropolis-Hastings 算法是以他的名字命名的。

# 第 12 章 计算学习理论

## 12.1 基础知识

顾名思义, 计算学习理论(computational learning theory)研究的是关于通过“计算”来进行“学习”的理论, 即关于机器学习的理论基础, 其目的是分析学习任务的困难本质, 为学习算法提供理论保证, 并根据分析结果指导算法设计.

给定样例集  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ,  $\mathbf{x}_i \in \mathcal{X}$ , 本章主要讨论二分类问题, 若无特别说明,  $y_i \in \mathcal{Y} = \{-1, +1\}$ . 假设  $\mathcal{X}$  中的所有样本服从一个隐含未知的分布  $\mathcal{D}$ ,  $D$  中所有样本都是独立地从这个分布上采样而得, 即独立同分布 (independent and identically distributed, 简称 *i.i.d.*) 样本.

令  $h$  为从  $\mathcal{X}$  到  $\mathcal{Y}$  的一个映射, 其泛化误差为

$$E(h; \mathcal{D}) = P_{\mathbf{x} \sim \mathcal{D}}(h(\mathbf{x}) \neq y) , \quad (12.1)$$

$h$  在  $D$  上的经验误差为

$$\widehat{E}(h; D) = \frac{1}{m} \sum_{i=1}^m \mathbb{I}(h(\mathbf{x}_i) \neq y_i) . \quad (12.2)$$

由于  $D$  是  $\mathcal{D}$  的独立同分布采样, 因此  $h$  的经验误差的期望等于其泛化误差. 在上下文明确时, 我们将  $E(h; \mathcal{D})$  和  $\widehat{E}(h; D)$  分别简记为  $E(h)$  和  $\widehat{E}(h)$ . 令  $\epsilon$  为  $E(h)$  的上限, 即  $E(h) \leq \epsilon$ ; 我们通常用  $\epsilon$  表示预先设定的学得模型所应满足的误差要求, 亦称“误差参数”.

本章后面部分将研究经验误差与泛化误差之间的逼近程度. 若  $h$  在数据集  $D$  上的经验误差为 0, 则称  $h$  与  $D$  一致, 否则称其与  $D$  不一致. 对任意两个映射  $h_1, h_2 \in \mathcal{X} \rightarrow \mathcal{Y}$ , 可通过其“不合”(disagreement)来度量它们之间的差别:

$$d(h_1, h_2) = P_{\mathbf{x} \sim \mathcal{D}}(h_1(\mathbf{x}) \neq h_2(\mathbf{x})) . \quad (12.3)$$

我们会用到几个常用不等式:

- Jensen 不等式: 对任意凸函数  $f(x)$ , 有

$$f(\mathbb{E}(x)) \leq \mathbb{E}(f(x)). \quad (12.4)$$

- Hoeffding 不等式 [Hoeffding, 1963]: 若  $x_1, x_2, \dots, x_m$  为  $m$  个独立随机变量, 且满足  $0 \leq x_i \leq 1$ , 则对任意  $\epsilon > 0$ , 有

$$P\left(\frac{1}{m} \sum_{i=1}^m x_i - \frac{1}{m} \sum_{i=1}^m \mathbb{E}(x_i) \geq \epsilon\right) \leq \exp(-2m\epsilon^2), \quad (12.5)$$

$$P\left(\left|\frac{1}{m} \sum_{i=1}^m x_i - \frac{1}{m} \sum_{i=1}^m \mathbb{E}(x_i)\right| \geq \epsilon\right) \leq 2 \exp(-2m\epsilon^2). \quad (12.6)$$

- McDiarmid 不等式 [McDiarmid, 1989]: 若  $x_1, x_2, \dots, x_m$  为  $m$  个独立随机变量, 且对任意  $1 \leq i \leq m$ , 函数  $f$  满足

$$\sup_{x_1, \dots, x_m, x'_i} |f(x_1, \dots, x_m) - f(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_m)| \leq c_i,$$

则对任意  $\epsilon > 0$ , 有

$$P(f(x_1, \dots, x_m) - \mathbb{E}(f(x_1, \dots, x_m)) \geq \epsilon) \leq \exp\left(\frac{-2\epsilon^2}{\sum_i c_i^2}\right), \quad (12.7)$$

$$P(|f(x_1, \dots, x_m) - \mathbb{E}(f(x_1, \dots, x_m))| \geq \epsilon) \leq 2 \exp\left(\frac{-2\epsilon^2}{\sum_i c_i^2}\right). \quad (12.8)$$

## 12.2 PAC学习

计算学习理论中最基本的是概率近似正确 (Probably Approximately Correct, 简称 PAC) 学习理论 [Valiant, 1984]. “概率近似正确”这个名字看起来有点古怪, 我们稍后再解释.

令  $c$  表示“概念” (concept), 这是从样本空间  $\mathcal{X}$  到标记空间  $\mathcal{Y}$  的映射, 它决定示例  $x$  的真实标记  $y$ , 若对任何样例  $(x, y)$  有  $c(x) = y$  成立, 则称  $c$  为目标准概念; 所有我们希望学得的目标概念所构成的集合称为“概念类” (concept class), 用符号  $\mathcal{C}$  表示.

给定学习算法  $\mathcal{L}$ , 它所考虑的所有可能概念的集合称为“假设空间” (hypothesis space), 用符号  $\mathcal{H}$  表示. 由于学习算法事先并不知道概念类的真实存在, 因此  $\mathcal{H}$  和  $\mathcal{C}$  通常是不同的, 学习算法会把自认为可能的目标概

念集中起来构成  $\mathcal{H}$ , 对  $h \in \mathcal{H}$ , 由于并不能确定它是否真是目标概念, 因此称为“假设”(hypothesis). 显然, 假设  $h$  也是从样本空间  $\mathcal{X}$  到标记空间  $\mathcal{Y}$  的映射.

若目标概念  $c \in \mathcal{H}$ , 则  $\mathcal{H}$  中存在假设能将所有示例按与真实标记一致的方式完全分开, 我们称该问题对学习算法  $\mathfrak{L}$  是“可分的”(separable), 亦称“一致的”(consistent); 若  $c \notin \mathcal{H}$ , 则  $\mathcal{H}$  中不存在任何假设能将所有示例完全正确分开, 称该问题对学习算法  $\mathfrak{L}$  是“不可分的”(non-separable), 亦称“不一致的”(non-consistent).

给定训练集  $D$ , 我们希望基于学习算法  $\mathfrak{L}$  学得的模型所对应的假设  $h$  尽可能接近目标概念  $c$ . 读者可能会问: 为什么不是希望精确地学到目标概念  $c$  呢? 这是由于机器学习过程受到很多因素的制约, 例如我们获得的训练集  $D$  往往仅包含有限数量的样例, 因此, 通常会存在一些在  $D$  上“等效”的假设, 学习算法对它们无法区别; 再如, 从分布  $\mathcal{D}$  采样得到  $D$  的过程有一定偶然性, 可以想象, 即便对同样大小的不同训练集, 学得结果也可能有所不同. 因此, 我们是希望以比较大的把握学得比较好的模型, 也就是说, 以较大的概率学得误差满足预设上限的模型; 这就是“概率”“近似正确”的含义. 形式化地说, 令  $\delta$  表示置信度, 可定义:

**定义 12.1 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  $\mathfrak{L}$ , 其输出假设  $h \in \mathcal{H}$  满足

$$P(E(h) \leq \epsilon) \geq 1 - \delta, \quad (12.9)$$

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

这样的学习算法  $\mathfrak{L}$  能以较大的概率(至少  $1 - \delta$ ) 学得目标概念  $c$  的近似(误差最多为  $\epsilon$ ). 在此基础上可定义:

**定义 12.2 PAC 可学习 (PAC Learnable):** 令  $m$  表示从分布  $\mathcal{D}$  中独立同分布采样得到的样例数目,  $0 < \epsilon, \delta < 1$ , 对所有分布  $\mathcal{D}$ , 若存在学习算法  $\mathfrak{L}$  和多项式函数  $\text{poly}(\cdot, \cdot, \cdot)$ , 使得对于任何  $m \geq \text{poly}(1/\epsilon, 1/\delta, \text{size}(\mathbf{x}), \text{size}(c))$ ,  $\mathfrak{L}$  能从假设空间  $\mathcal{H}$  中 PAC 辨识概念类  $\mathcal{C}$ , 则称概念类  $\mathcal{C}$  对假设空间  $\mathcal{H}$  而言是 PAC 可学习的, 有时也简称概念类  $\mathcal{C}$  是 PAC 可学习的.

样例数目  $m$  与误差  $\epsilon$ 、置信度  $\delta$ 、数据本身的复杂度  $\text{size}(\mathbf{x})$ 、目标概念的复杂度  $\text{size}(c)$  都有关.

对计算机算法来说, 必然要考虑时间复杂度, 于是:

**定义 12.3 PAC 学习算法 (PAC Learning Algorithm):** 若学习算法  $\mathcal{L}$  使概念类  $\mathcal{C}$  为 PAC 可学习的, 且  $\mathcal{L}$  的运行时间也是多项式函数  $\text{poly}(1/\epsilon, 1/\delta, \text{size}(\mathbf{x}), \text{size}(c))$ , 则称概念类  $\mathcal{C}$  是高效 PAC 可学习 (efficiently PAC learnable) 的, 称  $\mathcal{L}$  为概念类  $\mathcal{C}$  的 PAC 学习算法.

假定学习算法  $\mathcal{L}$  处理每个样本的时间为常数, 则  $\mathcal{L}$  的时间复杂度等价于样本复杂度. 于是, 我们对算法时间复杂度的关心就转化为对样本复杂度的关心:

**定义 12.4 样本复杂度 (Sample Complexity):** 满足 PAC 学习算法  $\mathcal{L}$  所需的  $m \geq \text{poly}(1/\epsilon, 1/\delta, \text{size}(\mathbf{x}), \text{size}(c))$  中最小的  $m$ , 称为学习算法  $\mathcal{L}$  的样本复杂度.

显然, PAC 学习给出了一个抽象地刻画机器学习能力的框架, 基于这个框架能对很多重要问题进行理论探讨, 例如研究某任务在什么样的条件下可学得较好的模型? 某算法在什么样的条件下可进行有效的学习? 需多少训练样例才能获得较好的模型?

PAC 学习中一个关键因素是假设空间  $\mathcal{H}$  的复杂度.  $\mathcal{H}$  包含了学习算法  $\mathcal{L}$  所有可能输出的假设, 若在 PAC 学习中假设空间与概念类完全相同, 即  $\mathcal{H} = \mathcal{C}$ , 这称为“恰 PAC 可学习” (properly PAC learnable); 直观地看, 这意味着学习算法的能力与学习任务“恰好匹配”. 然而, 这种让所有候选假设都来自概念类的要求看似合理, 但却并不实际, 因为在现实应用中我们对概念类  $\mathcal{C}$  通常一无所知, 更别说获得一个假设空间与概念类恰好相同的学习算法. 显然, 更重要的是研究假设空间与概念类不同的情形, 即  $\mathcal{H} \neq \mathcal{C}$ . 一般而言,  $\mathcal{H}$  越大, 其包含任意目标概念的可能性越大, 但从中找到某个具体目标概念的难度也越大.  $|\mathcal{H}|$  有时, 我们称  $\mathcal{H}$  为“有限假设空间”, 否则称为“无限假设空间”.

## 12.3 有限假设空间

### 12.3.1 可分情形

可分情形意味着目标概念  $c$  属于假设空间  $\mathcal{H}$ , 即  $c \in \mathcal{H}$ . 给定包含  $m$  个样例的训练集  $D$ , 如何找出满足误差参数的假设呢?

容易想到一种简单的学习策略: 既然  $D$  中样例标记都是由目标概念  $c$  赋予的, 并且  $c$  存在于假设空间  $\mathcal{H}$  中, 那么, 任何在训练集  $D$  上出现标记错误的假设肯定不是目标概念  $c$ . 于是, 我们只需保留与  $D$  一致的假设, 剔除与  $D$  不一致的假设即可. 若训练集  $D$  足够大, 则可不断借助  $D$  中的样例剔除不一致的假

设, 直到  $\mathcal{H}$  中仅剩下一个假设为止, 这个假设就是目标概念  $c$ . 通常情形下, 由于训练集规模有限, 假设空间  $\mathcal{H}$  中可能存在不止一个与  $D$  一致的“等效”假设, 对这些等效假设, 无法根据  $D$  来对它们的优劣做进一步区分.

到底需多少样例才能学得目标概念  $c$  的有效近似呢? 对 PAC 学习来说, 只要训练集  $D$  的规模能使学习算法  $\mathfrak{L}$  以概率  $1 - \delta$  找到目标假设的  $\epsilon$  近似即可.

我们先估计泛化误差大于  $\epsilon$  但在训练集上仍表现完美的假设出现的概率. 假定  $h$  的泛化误差大于  $\epsilon$ , 对分布  $\mathcal{D}$  上随机采样而得的任何样例  $(\mathbf{x}, y)$ , 有

$$\begin{aligned} P(h(\mathbf{x}) = y) &= 1 - P(h(\mathbf{x}) \neq y) \\ &= 1 - E(h) \\ &< 1 - \epsilon. \end{aligned} \tag{12.10}$$

由于  $D$  包含  $m$  个从  $\mathcal{D}$  独立同分布采样而得的样例, 因此,  $h$  与  $D$  表现一致的概率为

$$\begin{aligned} P((h(\mathbf{x}_1) = y_1) \wedge \dots \wedge (h(\mathbf{x}_m) = y_m)) &= (1 - P(h(\mathbf{x}) \neq y))^m \\ &< (1 - \epsilon)^m. \end{aligned} \tag{12.11}$$

我们事先并不知道学习算法  $\mathfrak{L}$  会输出  $\mathcal{H}$  中的哪个假设, 但仅需保证泛化误差大于  $\epsilon$ , 且在训练集上表现完美的所有假设出现概率之和不大于  $\delta$  即可:

$$\begin{aligned} P(h \in \mathcal{H} : E(h) > \epsilon \wedge \hat{E}(h) = 0) &< |\mathcal{H}|(1 - \epsilon)^m \\ &< |\mathcal{H}|e^{-m\epsilon}, \end{aligned} \tag{12.12}$$

令式(12.12)不大于  $\delta$ , 即

$$|\mathcal{H}|e^{-m\epsilon} \leq \delta, \tag{12.13}$$

可得

$$m \geq \frac{1}{\epsilon} (\ln |\mathcal{H}| + \ln \frac{1}{\delta}). \tag{12.14}$$

由此可知, 有限假设空间  $\mathcal{H}$  都是 PAC 可学习的, 所需的样例数目如式(12.14)所示, 输出假设  $h$  的泛化误差随样例数目的增多而收敛到 0, 收敛速率为  $O(\frac{1}{m})$ .

### 12.3.2 不可分情形

对较为困难的学习问题, 目标概念  $c$  往往不存在于假设空间  $\mathcal{H}$  中. 假定对于任何  $h \in \mathcal{H}$ ,  $\hat{E}(h) \neq 0$ , 也就是说,  $\mathcal{H}$  中的任意一个假设都会在训练集上出现或多或少的错误. 由 Hoeffding 不等式易知:

**引理 12.1** 若训练集  $D$  包含  $m$  个从分布  $\mathcal{D}$  上独立同分布采样而得的样例,  $0 < \epsilon < 1$ , 则对任意  $h \in \mathcal{H}$ , 有

$$P(\hat{E}(h) - E(h) \geq \epsilon) \leq \exp(-2m\epsilon^2), \quad (12.15)$$

$$P(E(h) - \hat{E}(h) \geq \epsilon) \leq \exp(-2m\epsilon^2), \quad (12.16)$$

$$P(|E(h) - \hat{E}(h)| \geq \epsilon) \leq 2 \exp(-2m\epsilon^2). \quad (12.17)$$

**推论 12.1** 若训练集  $D$  包含  $m$  个从分布  $\mathcal{D}$  上独立同分布采样而得的样例,  $0 < \epsilon < 1$ , 则对任意  $h \in \mathcal{H}$ , 式(12.18)以至少  $1 - \delta$  的概率成立:

$$\hat{E}(h) - \sqrt{\frac{\ln(2/\delta)}{2m}} \leq E(h) \leq \hat{E}(h) + \sqrt{\frac{\ln(2/\delta)}{2m}}. \quad (12.18)$$

推论 12.1 表明, 样例数目  $m$  较大时,  $h$  的经验误差是其泛化误差很好的近似. 对于有限假设空间  $\mathcal{H}$ , 我们有

**定理 12.1** 若  $\mathcal{H}$  为有限假设空间,  $0 < \delta < 1$ , 则对任意  $h \in \mathcal{H}$ , 有

$$P(|E(h) - \hat{E}(h)| \leq \sqrt{\frac{\ln|\mathcal{H}| + \ln(2/\delta)}{2m}}) \geq 1 - \delta. \quad (12.19)$$

**证明** 令  $h_1, h_2, \dots, h_{|\mathcal{H}|}$  表示假设空间  $\mathcal{H}$  中的假设, 有

$$\begin{aligned} & P(\exists h \in \mathcal{H} : |E(h) - \hat{E}(h)| > \epsilon) \\ &= P((|E_{h_1} - \hat{E}_{h_1}| > \epsilon) \vee \dots \vee (|E_{h_{|\mathcal{H}|}} - \hat{E}_{h_{|\mathcal{H}|}}| > \epsilon)) \\ &\leq \sum_{h \in \mathcal{H}} P(|E(h) - \hat{E}(h)| > \epsilon), \end{aligned}$$

由式(12.17)可得

$$\sum_{h \in \mathcal{H}} P(|E(h) - \hat{E}(h)| > \epsilon) \leq 2|\mathcal{H}| \exp(-2m\epsilon^2),$$

于是, 令  $\delta = 2|\mathcal{H}| \exp(-2m\epsilon^2)$  即可得式(12.19). ■

即在  $\mathcal{H}$  的所有假设中找出最好的一个.

显然, 当  $c \notin \mathcal{H}$  时, 学习算法  $\mathfrak{L}$  无法学得目标概念  $c$  的  $\epsilon$  近似. 但是, 当假设空间  $\mathcal{H}$  给定时, 其中必存在一个泛化误差最小的假设, 找出此假设的  $\epsilon$  近似也不失为一个较好的目标.  $\mathcal{H}$  中泛化误差最小的假设是  $\arg \min_{h \in \mathcal{H}} E(h)$ , 于是, 以此为目标可将 PAC 学习推广到  $c \notin \mathcal{H}$  的情况, 这称为“不可知学习”(agnostic learning). 相应的, 我们有

**定义 12.5 不可知 PAC 可学习 (agnostic PAC learnable):** 令  $m$  表示从分布  $\mathcal{D}$  中独立同分布采样得到的样例数目,  $0 < \epsilon, \delta < 1$ , 对所有分布  $\mathcal{D}$ , 若存在学习算法  $\mathfrak{L}$  和多项式函数  $\text{poly}(\cdot, \cdot, \cdot, \cdot)$ , 使得对于任何  $m \geq \text{poly}(1/\epsilon, 1/\delta, \text{size}(\mathbf{x}), \text{size}(c))$ ,  $\mathfrak{L}$  能从假设空间  $\mathcal{H}$  中输出满足式(12.20)的假设  $h$ :

$$P\left(E(h) - \min_{h' \in \mathcal{H}} E(h') \leq \epsilon\right) \geq 1 - \delta, \quad (12.20)$$

则称假设空间  $\mathcal{H}$  是不可知 PAC 可学习的.

与 PAC 可学习类似, 若学习算法  $\mathfrak{L}$  的运行时间也是多项式函数  $\text{poly}(1/\epsilon, 1/\delta, \text{size}(\mathbf{x}), \text{size}(c))$ , 则称假设空间  $\mathcal{H}$  是高效不可知 PAC 可学习的, 学习算法  $\mathfrak{L}$  则称为假设空间  $\mathcal{H}$  的不可知 PAC 学习算法, 满足上述要求的最小  $m$  称为学习算法  $\mathfrak{L}$  的样本复杂度.

## 12.4 VC维

现实学习任务所面临的通常是无限假设空间, 例如实数域中的所有区间、 $\mathbb{R}^d$  空间中的所有线性超平面. 欲对此种情形的可学习性进行研究, 需度量假设空间的复杂度. 最常见的办法是考虑假设空间的“VC维”(Vapnik-Chervonenkis dimension) [Vapnik and Chervonenkis, 1971].

介绍 VC 维之前, 我们先引入几个概念: 增长函数 (growth function)、对分 (dichotomy) 和打散 (shattering).

给定假设空间  $\mathcal{H}$  和示例集  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ,  $\mathcal{H}$  中每个假设  $h$  都能对  $D$  中示例赋予标记, 标记结果可表示为

$$h|_D = \{(h(\mathbf{x}_1), h(\mathbf{x}_2), \dots, h(\mathbf{x}_m))\}.$$

例如, 对二分类问题, 若  $D$  中只有 2 个示例, 则赋予标记的可能结果只有 4 种; 若有 3 个示例, 则可能结果有 8 种.

$\mathbb{N}$  为自然数域.

随着  $m$  的增大,  $\mathcal{H}$  中所有假设对  $D$  中的示例所能赋予标记的可能结果数也会增大.

**定义 12.6** 对所有  $m \in \mathbb{N}$ , 假设空间  $\mathcal{H}$  的增长函数  $\Pi_{\mathcal{H}}(m)$  为

$$\Pi_{\mathcal{H}}(m) = \max_{\{\mathbf{x}_1, \dots, \mathbf{x}_m\} \subseteq \mathcal{X}} |\{(h(\mathbf{x}_1), \dots, h(\mathbf{x}_m)) \mid h \in \mathcal{H}\}|. \quad (12.21)$$

增长函数  $\Pi_{\mathcal{H}}(m)$  表示假设空间  $\mathcal{H}$  对  $m$  个示例所能赋予标记的最大可能结果数. 显然,  $\mathcal{H}$  对示例所能赋予标记的可能结果数越大,  $\mathcal{H}$  的表示能力越强, 对学习任务的适应能力也越强. 因此, 增长函数描述了假设空间  $\mathcal{H}$  的表示能力, 由此反映出假设空间的复杂度. 我们可利用增长函数来估计经验误差与泛化误差之间的关系:

**定理 12.2** 对假设空间  $\mathcal{H}$ ,  $m \in \mathbb{N}$ ,  $0 < \epsilon < 1$  和任意  $h \in \mathcal{H}$  有

证明过程参阅[Vapnik and Chervonenkis, 1971].

$$P(|E(h) - \hat{E}(h)| > \epsilon) \leq 4\Pi_{\mathcal{H}}(2m) \exp\left(-\frac{m\epsilon^2}{8}\right). \quad (12.22)$$

每个假设会把  $D$  中示例分为两类, 因此称为对分.

假设空间  $\mathcal{H}$  中不同的假设对于  $D$  中示例赋予标记的结果可能相同, 也可能不同; 尽管  $\mathcal{H}$  可能包含无穷多个假设, 但其对  $D$  中示例赋予标记的可能结果数是有限的: 对  $m$  个示例, 最多有  $2^m$  个可能结果. 对二分类问题来说,  $\mathcal{H}$  中的假设对  $D$  中示例赋予标记的每种可能结果称为对  $D$  的一种“对分”. 若假设空间  $\mathcal{H}$  能实现示例集  $D$  上的所有对分, 即  $\Pi_{\mathcal{H}}(m) = 2^m$ , 则称示例集  $D$  能被假设空间  $\mathcal{H}$  “打散”.

现在我们可以正式定义 VC 维了:

**定义 12.7** 假设空间  $\mathcal{H}$  的 VC 维是能被  $\mathcal{H}$  打散的最大示例集的大小, 即

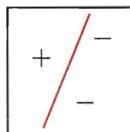
$$\text{VC}(\mathcal{H}) = \max\{m : \Pi_{\mathcal{H}}(m) = 2^m\}. \quad (12.23)$$

$\text{VC}(\mathcal{H}) = d$  表明存在大小为  $d$  的示例集能被假设空间  $\mathcal{H}$  打散. 注意: 这并不意味着所有大小为  $d$  的示例集都能被假设空间  $\mathcal{H}$  打散. 细心的读者可能已发现, VC 维的定义与数据分布  $\mathcal{D}$  无关! 因此, 在数据分布未知时仍能计算出假设空间  $\mathcal{H}$  的 VC 维.

通常这样来计算  $\mathcal{H}$  的 VC 维: 若存在大小为  $d$  的示例集能被  $\mathcal{H}$  打散, 但不存在任何大小为  $d + 1$  的示例集能被  $\mathcal{H}$  打散, 则  $\mathcal{H}$  的 VC 维是  $d$ . 下面给出两个计算 VC 维的例子:

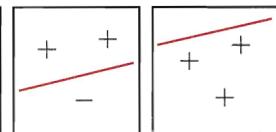
**例 12.1** 实数域中的区间  $[a, b]$ : 令  $\mathcal{H}$  表示实数域中所有闭区间构成的集合  $\{h_{[a,b]} : a, b \in \mathbb{R}, a \leq b\}$ ,  $\mathcal{X} = \mathbb{R}$ . 对  $x \in \mathcal{X}$ , 若  $x \in [a, b]$ , 则  $h_{[a,b]}(x) = +1$ , 否则  $h_{[a,b]}(x) = -1$ . 令  $x_1 = 0.5, x_2 = 1.5$ , 则假设空间  $\mathcal{H}$  中存在假设  $\{h_{[0,1]}, h_{[0,2]}, h_{[1,2]}, h_{[2,3]}\}$  将  $\{x_1, x_2\}$  打散, 所以假设空间  $\mathcal{H}$  的 VC 维至少为 2; 对任意大小为 3 的示例集  $\{x_3, x_4, x_5\}$ , 不妨设  $x_3 < x_4 < x_5$ , 则  $\mathcal{H}$  中不存在任何假设  $h_{[a,b]}$  能实现对分结果  $\{(x_3, +), (x_4, -), (x_5, +)\}$ . 于是,  $\mathcal{H}$  的 VC 维为 2.

**例 12.2** 二维实平面上的线性划分: 令  $\mathcal{H}$  表示二维实平面上所有线性划分构成的集合,  $\mathcal{X} = \mathbb{R}^2$ . 由图 12.1 可知, 存在大小为 3 的示例集可被  $\mathcal{H}$  打散, 但不存在大小为 4 的示例集可被  $\mathcal{H}$  打散. 于是, 二维实平面上所有线性划分构成的假设空间  $\mathcal{H}$  的 VC 维为 3.



存在这样的集合, 其  $2^3 = 8$  种对分均可被线性划分实现

(a) 示例集大小为 3



对任何集合, 其  $2^4 = 16$  种对分中至少有一种不能被线性划分实现

(b) 示例集大小为 4

图 12.1 二维实平面上所有线性划分构成的假设空间的 VC 维为 3

由定义 12.7 可知, VC 维与增长函数有密切联系, 引理 12.2 给出了二者之间的定量关系 [Sauer, 1972]:

亦称“Sauer引理”.

**引理 12.2** 若假设空间  $\mathcal{H}$  的 VC 维为  $d$ , 则对任意  $m \in \mathbb{N}$  有

$$\Pi_{\mathcal{H}}(m) \leq \sum_{i=0}^d \binom{m}{i}. \quad (12.24)$$

**证明** 由数学归纳法证明. 当  $m = 1, d = 0$  或  $d = 1$  时, 定理成立. 假设定理对  $(m-1, d-1)$  和  $(m-1, d)$  成立. 令  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ,  $D' = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{m-1}\}$ ,

$$\mathcal{H}|_D = \{ (h(\mathbf{x}_1), h(\mathbf{x}_2), \dots, h(\mathbf{x}_m)) \mid h \in \mathcal{H} \},$$

$$\mathcal{H}|_{D'} = \{ (h(\mathbf{x}_1), h(\mathbf{x}_2), \dots, h(\mathbf{x}_{m-1})) \mid h \in \mathcal{H} \}.$$

任何假设  $h \in \mathcal{H}$  对  $\mathbf{x}_m$  的分类结果或为  $+1$ , 或为  $-1$ , 因此任何出现在

$\mathcal{H}_{|D'|}$  中的串都会在  $\mathcal{H}_{|D|}$  中出现一次或两次. 令  $\mathcal{H}_{D'|D}$  表示在  $\mathcal{H}_{|D|}$  中出现两次的  $\mathcal{H}_{|D'|}$  中串组成的集合, 即

$$\begin{aligned}\mathcal{H}_{D'|D} = & \{(y_1, y_2, \dots, y_{m-1}) \in \mathcal{H}_{|D'|} \mid \exists h, h' \in \mathcal{H}, \\ & (h(\mathbf{x}_i) = h'(\mathbf{x}_i) = y_i) \wedge (h(\mathbf{x}_m) \neq h'(\mathbf{x}_m)), 1 \leq i \leq m-1\}.\end{aligned}$$

考虑到  $\mathcal{H}_{D'|D}$  中的串在  $\mathcal{H}_{|D|}$  中出现了两次, 但在  $\mathcal{H}_{|D'|}$  中仅出现了一次, 有

$$|\mathcal{H}_{|D|}| = |\mathcal{H}_{|D'|}| + |\mathcal{H}_{D'|D}|. \quad (12.25)$$

$D'$  的大小为  $m-1$ , 由假设可得

$$|\mathcal{H}_{|D'|}| \leq \Pi_{\mathcal{H}}(m-1) \leq \sum_{i=0}^d \binom{m-1}{i}. \quad (12.26)$$

令  $Q$  表示能被  $\mathcal{H}_{D'|D}$  打散的集合, 由  $\mathcal{H}_{D'|D}$  定义可知  $Q \cup \{\mathbf{x}_m\}$  必能被  $\mathcal{H}_{|D|}$  打散. 由于  $\mathcal{H}$  的 VC 维为  $d$ , 因此  $\mathcal{H}_{D'|D}$  的 VC 维最大为  $d-1$ , 于是有

$$|\mathcal{H}_{D'|D}| \leq \Pi_{\mathcal{H}}(m-1) \leq \sum_{i=0}^{d-1} \binom{m-1}{i}. \quad (12.27)$$

由式(12.25)~(12.27)可得

$$\begin{aligned}|\mathcal{H}_{|D|}| & \leq \sum_{i=0}^d \binom{m-1}{i} + \sum_{i=0}^{d-1} \binom{m-1}{i} \\ & = \sum_{i=0}^d \left( \binom{m-1}{i} + \binom{m-1}{i-1} \right) \\ & = \sum_{i=0}^d \binom{m}{i},\end{aligned}$$

由集合  $D$  的任意性, 引理 12.2 得证. ■

从引理 12.2 可计算出增长函数的上界:

**推论 12.2** 若假设空间  $\mathcal{H}$  的 VC 维为  $d$ , 则对任意整数  $m \geq d$  有

$e$  为自然常数.

$$\Pi_{\mathcal{H}}(m) \leq \left( \frac{e \cdot m}{d} \right)^d. \quad (12.28)$$

证明

$$\begin{aligned}
 \Pi_{\mathcal{H}}(m) &\leq \sum_{i=0}^d \binom{m}{i} \\
 &\leq \sum_{i=0}^d \binom{m}{i} \left(\frac{m}{d}\right)^{d-i} \\
 &= \left(\frac{m}{d}\right)^d \sum_{i=0}^d \binom{m}{i} \left(\frac{d}{m}\right)^i \\
 &\leq \left(\frac{m}{d}\right)^d \sum_{i=0}^m \binom{m}{i} \left(\frac{d}{m}\right)^i \\
 &= \left(\frac{m}{d}\right)^d \left(1 + \frac{d}{m}\right)^m \\
 &\leq \left(\frac{e \cdot m}{d}\right)^d
 \end{aligned}$$

■

根据推论 12.2 和定理 12.2 可得基于 VC 维的泛化误差界:

**定理 12.3** 若假设空间  $\mathcal{H}$  的 VC 维为  $d$ , 则对任意  $m > d$ ,  $0 < \delta < 1$  和  $h \in \mathcal{H}$  有

$$P\left(E(h) - \hat{E}(h) \leq \sqrt{\frac{8d \ln \frac{2em}{d} + 8 \ln \frac{4}{\delta}}{m}}\right) \geq 1 - \delta. \quad (12.29)$$

证明 令  $4\Pi_{\mathcal{H}}(2m) \exp(-\frac{m\epsilon^2}{8}) \leq 4(\frac{2em}{d})^d \exp(-\frac{m\epsilon^2}{8}) = \delta$ , 解得

$$\epsilon = \sqrt{\frac{8d \ln \frac{2em}{d} + 8 \ln \frac{4}{\delta}}{m}},$$

代入定理 12.2, 于是定理 12.3 得证. ■

由定理 12.3 可知, 式(12.29)的泛化误差界只与样例数目  $m$  有关, 收敛速率为  $O(\frac{1}{\sqrt{m}})$ , 与数据分布  $\mathcal{D}$  和样例集  $D$  无关. 因此, 基于 VC 维的泛化误差界是分布无关 (distribution-free)、数据独立 (data-independent) 的.

令  $h$  表示学习算法  $\mathcal{L}$  输出的假设, 若  $h$  满足

$$\widehat{E}(h) = \min_{h' \in \mathcal{H}} \widehat{E}(h') , \quad (12.30)$$

则称  $\mathcal{L}$  为满足经验风险最小化 (Empirical Risk Minimization, 简称 ERM) 原则的算法. 我们有下面的定理:

**定理 12.4** 任何 VC 维有限的假设空间  $\mathcal{H}$  都是(不可知) PAC 可学习的.

**证明** 假设  $\mathcal{L}$  为满足经验风险最小化原则的算法,  $h$  为学习算法  $\mathcal{L}$  输出的假设. 令  $g$  表示  $\mathcal{H}$  中具有最小泛化误差的假设, 即

$$E(g) = \min_{h \in \mathcal{H}} E(h) . \quad (12.31)$$

令

$$\delta' = \frac{\delta}{2} ,$$

$$\sqrt{\frac{(\ln 2/\delta')}{2m}} = \frac{\epsilon}{2} , \quad (12.32)$$

由推论 12.1 可知

$$\widehat{E}(g) - \frac{\epsilon}{2} \leq E(g) \leq \widehat{E}(g) + \frac{\epsilon}{2}$$

至少以  $1 - \delta/2$  的概率成立. 令

$$\sqrt{\frac{8d \ln \frac{2em}{d} + 8 \ln \frac{4}{\delta'}}{m}} = \frac{\epsilon}{2} , \quad (12.34)$$

则由定理 12.3 可知

$$P\left(E(h) - \widehat{E}(h) \leq \frac{\epsilon}{2}\right) \geq 1 - \frac{\delta}{2} .$$

从而可知

$$\begin{aligned} E(h) - E(g) &\leq \widehat{E}(h) + \frac{\epsilon}{2} - \left(\widehat{E}(g) - \frac{\epsilon}{2}\right) \\ &= \widehat{E}(h) - \widehat{E}(g) + \epsilon \\ &\leq \epsilon \end{aligned}$$

以至少  $1 - \delta$  的概率成立。由式(12.32)和(12.34)可以解出  $m$ , 再由  $\mathcal{H}$  的任意性可知定理 12.4 得证。 ■

## 12.5 Rademacher 复杂度

12.4 节提到, 基于 VC 维的泛化误差界是分布无关、数据独立的, 也就是说, 对任何数据分布都成立。这使得基于 VC 维的可学习性分析结果具有一定的“普适性”; 但从另一方面来说, 由于没有考虑数据自身, 基于 VC 维得到的泛化误差界通常比较“松”, 对那些与学习问题的典型情况相差甚远的较“坏”分布来说尤其如此。

这个名字是为了纪念  
德国数学家 H. Rademacher (1892–1969)。

Rademacher 复杂度 (Rademacher complexity) 是另一种刻画假设空间复杂度的途径, 与 VC 维不同的是, 它在一定程度上考虑了数据分布。

给定训练集  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ , 假设  $h$  的经验误差为

$$\begin{aligned}\widehat{E}(h) &= \frac{1}{m} \sum_{i=1}^m \mathbb{I}(h(\mathbf{x}_i) \neq y_i) \\ &= \frac{1}{m} \sum_{i=1}^m \frac{1 - y_i h(\mathbf{x}_i)}{2} \\ &= \frac{1}{2} - \frac{1}{2m} \sum_{i=1}^m y_i h(\mathbf{x}_i),\end{aligned}\tag{12.36}$$

其中  $\frac{1}{m} \sum_{i=1}^m y_i h(\mathbf{x}_i)$  体现了预测值  $h(\mathbf{x}_i)$  与样例真实标记  $y_i$  之间的一致性, 若对于所有  $i \in \{1, 2, \dots, m\}$  都有  $h(\mathbf{x}_i) = y_i$ , 则  $\frac{1}{m} \sum_{i=1}^m y_i h(\mathbf{x}_i)$  取最大值 1。也就是说, 经验误差最小的假设是

$$\arg \max_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m y_i h(\mathbf{x}_i).\tag{12.37}$$

然而, 现实任务中样例的标记有时会受到噪声影响, 即对某些样例  $(\mathbf{x}_i, y_i)$ , 其  $y_i$  或许已受到随机因素的影响, 不再是  $\mathbf{x}_i$  的真实标记。在此情形下, 选择假设空间  $\mathcal{H}$  中在训练集上表现最好的假设, 有时还不如选择  $\mathcal{H}$  中事先已考虑了随机噪声影响的假设。

考虑随机变量  $\sigma_i$ , 它以 0.5 的概率取值  $-1$ , 0.5 的概率取值  $+1$ , 称为

Rademacher 随机变量. 基于  $\sigma_i$ , 可将式(12.37)重写为

$\mathcal{H}$  是无限假设空间, 有可能取不到最大值, 因此使用上确界代替最大值.

$$\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m \sigma_i h(\mathbf{x}_i). \quad (12.38)$$

考虑  $\mathcal{H}$  中的所有假设, 对式(12.38)取期望可得

$$\mathbb{E}_{\sigma} \left[ \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m \sigma_i h(\mathbf{x}_i) \right], \quad (12.39)$$

其中  $\sigma = \{\sigma_1, \sigma_2, \dots, \sigma_m\}$ . 式(12.39)的取值范围是  $[0, 1]$ , 它体现了假设空间  $\mathcal{H}$  的表达能力, 例如, 当  $|\mathcal{H}| = 1$  时,  $\mathcal{H}$  中仅有一个假设, 这时可计算出式(12.39)的值为 0; 当  $|\mathcal{H}| = 2^m$  且  $\mathcal{H}$  能打散  $D$  时, 对任意  $\sigma$  总有一个假设使得  $h(\mathbf{x}_i) = \sigma_i$  ( $i = 1, 2, \dots, m$ ), 这时可计算出式(12.39)的值为 1.

考虑实值函数空间  $\mathcal{F}: \mathcal{Z} \rightarrow \mathbb{R}$ . 令  $Z = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m\}$ , 其中  $\mathbf{z}_i \in \mathcal{Z}$ , 将式(12.39)中的  $\mathcal{X}$  和  $\mathcal{H}$  替换为  $\mathcal{Z}$  和  $\mathcal{F}$  可得

**定义 12.8** 函数空间  $\mathcal{F}$  关于  $Z$  的经验 Rademacher 复杂度

$$\hat{R}_Z(\mathcal{F}) = \mathbb{E}_{\sigma} \left[ \sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \sigma_i f(\mathbf{z}_i) \right]. \quad (12.40)$$

经验 Rademacher 复杂度衡量了函数空间  $\mathcal{F}$  与随机噪声在集合  $Z$  中的相关性. 通常我们希望了解函数空间  $\mathcal{F}$  在  $\mathcal{Z}$  上关于分布  $\mathcal{D}$  的相关性, 因此, 对所有从  $\mathcal{D}$  独立同分布采样而得的大小为  $m$  的集合  $Z$  求期望可得

**定义 12.9** 函数空间  $\mathcal{F}$  关于  $Z$  上分布  $\mathcal{D}$  的 Rademacher 复杂度

$$R_m(\mathcal{F}) = \mathbb{E}_{Z \subseteq \mathcal{Z}: |Z|=m} [\hat{R}_Z(\mathcal{F})]. \quad (12.41)$$

基于 Rademacher 复杂度可得关于函数空间  $\mathcal{F}$  的泛化误差界 [Mohri et al., 2012]:

**定理 12.5** 对实值函数空间  $\mathcal{F}: \mathcal{Z} \rightarrow [0, 1]$ , 根据分布  $\mathcal{D}$  从  $\mathcal{Z}$  中独立同分布采样得到示例集  $Z = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m\}$ ,  $\mathbf{z}_i \in \mathcal{Z}$ ,  $0 < \delta < 1$ , 对任意  $f \in \mathcal{F}$ , 以

至少  $1 - \delta$  的概率有

$$\mathbb{E}[f(\mathbf{z})] \leq \frac{1}{m} \sum_{i=1}^m f(\mathbf{z}_i) + 2R_m(\mathcal{F}) + \sqrt{\frac{\ln(1/\delta)}{2m}}, \quad (12.42)$$

$$\mathbb{E}[f(\mathbf{z})] \leq \frac{1}{m} \sum_{i=1}^m f(\mathbf{z}_i) + 2\widehat{R}_Z(\mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2m}}. \quad (12.43)$$

证明 令

$$\widehat{E}_Z(f) = \frac{1}{m} \sum_{i=1}^m f(\mathbf{z}_i),$$

$$\Phi(Z) = \sup_{f \in \mathcal{F}} \mathbb{E}[f] - \widehat{E}_Z(f),$$

同时, 令  $Z'$  为只与  $Z$  有一个示例不同的训练集, 不妨设  $\mathbf{z}_m \in Z$  和  $\mathbf{z}'_m \in Z'$  为不同示例, 可得

$$\begin{aligned} \Phi(Z') - \Phi(Z) &= \left( \sup_{f \in \mathcal{F}} \mathbb{E}[f] - \widehat{E}_{Z'}(f) \right) - \left( \sup_{f \in \mathcal{F}} \mathbb{E}[f] - \widehat{E}_Z(f) \right) \\ &\leq \sup_{f \in \mathcal{F}} \widehat{E}_Z(f) - \widehat{E}_{Z'}(f) \\ &= \sup_{f \in \mathcal{F}} \frac{f(\mathbf{z}_m) - f(\mathbf{z}'_m)}{m} \\ &\leq \frac{1}{m}. \end{aligned}$$

同理可得

$$\Phi(Z) - \Phi(Z') \leq \frac{1}{m},$$

$$|\Phi(Z) - \Phi(Z')| \leq \frac{1}{m}.$$

根据 McDiarmid 不等式(12.7)可知, 对任意  $\delta \in (0, 1)$ ,

$$\Phi(Z) \leq \mathbb{E}_Z[\Phi(Z)] + \sqrt{\frac{\ln(1/\delta)}{2m}} \quad (12.44)$$

以至少  $1 - \delta$  的概率成立。下面来估计  $\mathbb{E}_Z[\Phi(Z)]$  的上界：

利用 Jensen 不等式  
(12.4) 和上确界函数的凸性。

$$\begin{aligned}
 \mathbb{E}_Z[\Phi(Z)] &= \mathbb{E}_Z \left[ \sup_{f \in \mathcal{F}} \mathbb{E}[f] - \hat{E}_Z(f) \right] \\
 &= \mathbb{E}_Z \left[ \sup_{f \in \mathcal{F}} \mathbb{E}_{Z'} [\hat{E}_{Z'}(f) - \hat{E}_Z(f)] \right] \\
 &\leq \mathbb{E}_{Z, Z'} \left[ \sup_{f \in \mathcal{F}} \hat{E}_{Z'}(f) - \hat{E}_Z(f) \right] \\
 &= \mathbb{E}_{Z, Z'} \left[ \sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m (f(\mathbf{z}'_i) - f(\mathbf{z}_i)) \right] \\
 &= \mathbb{E}_{\sigma, Z, Z'} \left[ \sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \sigma_i (f(\mathbf{z}'_i) - f(\mathbf{z}_i)) \right] \\
 &\leq \mathbb{E}_{\sigma, Z'} \left[ \sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \sigma_i f(\mathbf{z}'_i) \right] + \mathbb{E}_{\sigma, Z} \left[ \sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m -\sigma_i f(\mathbf{z}_i) \right] \\
 &= 2\mathbb{E}_{\sigma, Z} \left[ \sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \sigma_i f(\mathbf{z}_i) \right] \\
 &= 2R_m(\mathcal{F}) .
 \end{aligned}$$

$\sigma_i$  与  $-\sigma_i$  分布相同。

至此，式(12.42)得证。由定义 12.9 可知，改变  $Z$  中的一个示例对  $\hat{R}_Z(\mathcal{F})$  的值所造成的改变最多为  $1/m$ 。由 McDiarmid 不等式(12.7)可知，

$$R_m(\mathcal{F}) \leq \hat{R}_Z(\mathcal{F}) + \sqrt{\frac{\ln(2/\delta)}{2m}} \quad (12.45)$$

以至少  $1 - \delta/2$  的概率成立。再由式(12.44)可知，

$$\Phi(Z) \leq \mathbb{E}_Z[\Phi(Z)] + \sqrt{\frac{\ln(2/\delta)}{2m}}$$

以至少  $1 - \delta/2$  的概率成立。于是，

$$\Phi(Z) \leq 2\hat{R}_Z(\mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2m}} \quad (12.46)$$

以至少  $1 - \delta$  的概率成立。至此，式(12.43)得证。 ■

需注意的是，定理 12.5 中的函数空间  $\mathcal{F}$  是区间  $[0, 1]$  上的实值函数，因此定理 12.5 只适用于回归问题。对二分类问题，我们有下面的定理：

**定理 12.6** 对假设空间  $\mathcal{H}: \mathcal{X} \rightarrow \{-1, +1\}$ , 根据分布  $\mathcal{D}$  从  $\mathcal{X}$  中独立同分布采样得到示例集  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ,  $\mathbf{x}_i \in \mathcal{X}$ ,  $0 < \delta < 1$ , 对任意  $h \in \mathcal{H}$ , 以至少  $1 - \delta$  的概率有

$$E(h) \leq \widehat{E}(h) + R_m(\mathcal{H}) + \sqrt{\frac{\ln(1/\delta)}{2m}}, \quad (12.47)$$

$$E(h) \leq \widehat{E}(h) + \widehat{R}_D(\mathcal{H}) + 3\sqrt{\frac{\ln(2/\delta)}{2m}}. \quad (12.48)$$

**证明** 对二分类问题的假设空间  $\mathcal{H}$ , 令  $\mathcal{Z} = \mathcal{X} \times \{-1, +1\}$ , 则  $\mathcal{H}$  中的假设  $h$  变形为

$$f_h(\mathbf{z}) = f_h(\mathbf{x}, y) = \mathbb{I}(h(\mathbf{x}) \neq y), \quad (12.49)$$

于是就可将值域为  $\{-1, +1\}$  的假设空间  $\mathcal{H}$  转化为值域为  $[0, 1]$  的函数空间  $\mathcal{F}_{\mathcal{H}} = \{f_h : h \in \mathcal{H}\}$ . 由定义 12.8, 有

$$\begin{aligned} \widehat{R}_Z(\mathcal{F}_{\mathcal{H}}) &= \mathbb{E}_{\sigma} \left[ \sup_{f_h \in \mathcal{F}_{\mathcal{H}}} \frac{1}{m} \sum_{i=1}^m \sigma_i f_h(\mathbf{x}_i, y_i) \right] \\ &= \mathbb{E}_{\sigma} \left[ \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m \sigma_i \mathbb{I}(h(\mathbf{x}_i) \neq y_i) \right] \\ &= \mathbb{E}_{\sigma} \left[ \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m \sigma_i \frac{1 - y_i h(\mathbf{x}_i)}{2} \right] \\ &= \frac{1}{2} \mathbb{E}_{\sigma} \left[ \frac{1}{m} \sum_{i=1}^m \sigma_i + \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m (-y_i \sigma_i h(\mathbf{x}_i)) \right] \\ &= \frac{1}{2} \mathbb{E}_{\sigma} \left[ \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m (-y_i \sigma_i h(\mathbf{x}_i)) \right] \\ &= \frac{1}{2} \mathbb{E}_{\sigma} \left[ \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m (\sigma_i h(\mathbf{x}_i)) \right] \\ &= \frac{1}{2} \widehat{R}_D(\mathcal{H}). \end{aligned} \quad (12.50)$$

-  $y_i \sigma_i$  与  $\sigma_i$  分布相同.

对式(12.50)求期望后可得

$$R_m(\mathcal{F}_{\mathcal{H}}) = \frac{1}{2} R_m(\mathcal{H}). \quad (12.51)$$

由定理 12.5 和 式(12.50)~(12.51), 定理 12.6 得证. ■

定理 12.6 给出了基于 Rademacher 复杂度的泛化误差界. 与定理 12.3 对比可知, 基于 VC 维的泛化误差界是分布无关、数据独立的, 而基于 Rademacher 复杂度的泛化误差界(12.47)与分布  $\mathcal{D}$  有关, 式(12.48)与数据  $D$  有关. 换言之, 基于 Rademacher 复杂度的泛化误差界依赖于具体学习问题上的数据分布, 有点类似于为该学习问题“量身定制”的, 因此它通常比基于 VC 维的泛化误差界更紧一些.

值得一提的是, 关于 Rademacher 复杂度与增长函数, 有如下定理:

证明过程参阅 [Mohri et al., 2012].

**定理 12.7** 假设空间  $\mathcal{H}$  的 Rademacher 复杂度  $R_m(\mathcal{H})$  与增长函数  $\Pi_{\mathcal{H}}(m)$  满足

$$R_m(\mathcal{H}) \leq \sqrt{\frac{2 \ln \Pi_{\mathcal{H}}(m)}{m}}. \quad (12.52)$$

由式(12.47), (12.52)和推论 12.2 可得

$$E(h) \leq \widehat{E}(h) + \sqrt{\frac{2d \ln \frac{em}{d}}{m}} + \sqrt{\frac{\ln(1/\delta)}{2m}}, \quad (12.53)$$

也就是说, 我们从 Rademacher 复杂度和增长函数能推导出基于 VC 维的泛化误差界.

## 12.6 稳定性

无论是基于 VC 维还是 Rademacher 复杂度来推导泛化误差界, 所得到的结果均与具体学习算法无关, 对所有学习算法都适用. 这使得人们能够脱离具体学习算法的设计来考虑学习问题本身的性质, 但在另一方面, 若希望获得与算法有关的分析结果, 则需另辟蹊径. 稳定性 (stability) 分析是这方面一个值得关注的方向.

顾名思义, 算法的“稳定性”考察的是算法在输入发生变化时, 输出是否会随之发生较大的变化. 学习算法的输入是训练集, 因此下面我们先定义训练集的两种变化.

给定  $D = \{\mathbf{z}_1 = (\mathbf{x}_1, y_1), \mathbf{z}_2 = (\mathbf{x}_2, y_2), \dots, \mathbf{z}_m = (\mathbf{x}_m, y_m)\}$ ,  $\mathbf{x}_i \in \mathcal{X}$  是来自分布  $\mathcal{D}$  的独立同分布示例,  $y_i = \{-1, +1\}$ . 对假设空间  $\mathcal{H} : \mathcal{X} \rightarrow \{-1, +1\}$  和学习算法  $\mathfrak{L}$ , 令  $\mathfrak{L}_D \in \mathcal{H}$  表示基于训练集  $D$  从假设空间  $\mathcal{H}$  中学得的假设. 考虑  $D$  的以下变化:

- $D^{\setminus i}$  表示移除  $D$  中第  $i$  个样例得到的集合

$$D^{\setminus i} = \{z_1, z_2, \dots, z_{i-1}, z_{i+1}, \dots, z_m\},$$

- $D^i$  表示替换  $D$  中第  $i$  个样例得到的集合

$$D^i = \{z_1, z_2, \dots, z_{i-1}, z'_i, z_{i+1}, \dots, z_m\},$$

其中  $z'_i = (x'_i, y'_i)$ ,  $x'_i$  服从分布  $\mathcal{D}$  并独立于  $D$ .

损失函数  $\ell(\mathcal{L}_D(x), y) : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$  刻画了假设  $\mathcal{L}_D$  的预测标记  $\mathcal{L}_D(x)$  与真实标记  $y$  之间的差别, 简记为  $\ell(\mathcal{L}_D, z)$ . 下面定义关于假设  $\mathcal{L}_D$  的几种损失.

- 泛化损失

$$\ell(\mathcal{L}, \mathcal{D}) = \mathbb{E}_{x \in \mathcal{X}, z=(x,y)} [\ell(\mathcal{L}_D, z)]. \quad (12.54)$$

- 经验损失

$$\widehat{\ell}(\mathcal{L}, D) = \frac{1}{m} \sum_{i=1}^m \ell(\mathcal{L}_D, z_i). \quad (12.55)$$

- 留一(leave-one-out)损失

$$\ell_{loo}(\mathcal{L}, D) = \frac{1}{m} \sum_{i=1}^m \ell(\mathcal{L}_{D^{\setminus i}}, z_i). \quad (12.56)$$

下面定义算法的均匀稳定性 (uniform stability):

**定义 12.10** 对任何  $x \in \mathcal{X}$ ,  $z = (x, y)$ , 若学习算法  $\mathcal{L}$  满足

$$|\ell(\mathcal{L}_D, z) - \ell(\mathcal{L}_{D^{\setminus i}}, z)| \leq \beta, \quad i = 1, 2, \dots, m, \quad (12.57)$$

则称  $\mathcal{L}$  关于损失函数  $\ell$  满足  $\beta$ -均匀稳定性.

显然, 若算法  $\mathcal{L}$  关于损失函数  $\ell$  满足  $\beta$ -均匀稳定性, 则有

$$\begin{aligned} & |\ell(\mathcal{L}_D, z) - \ell(\mathcal{L}_{D^i}, z)| \\ & \leq |\ell(\mathcal{L}_D, z) - \ell(\mathcal{L}_{D^{\setminus i}}, z)| + |\ell(\mathcal{L}_{D^i}, z) - \ell(\mathcal{L}_{D^{\setminus i}}, z)| \\ & \leq 2\beta, \end{aligned}$$

也就是说，移除示例的稳定性包含替换示例的稳定性。

若损失函数  $\ell$  有界，即对所有  $D$  和  $z = (x, y)$  有  $0 \leq l(\mathcal{L}_D, z) \leq M$ ，则有 [Bousquet and Elisseeff, 2002]：

证明过程参阅 [Bousquet and Elisseeff, 2002].

**定理 12.8** 给定从分布  $\mathcal{D}$  上独立同分布采样得到的大小为  $m$  的示例集  $D$ ，若学习算法  $\mathcal{L}$  满足关于损失函数  $\ell$  的  $\beta$ -均匀稳定性，且损失函数  $\ell$  的上界为  $M$ ， $0 < \delta < 1$ ，则对任意  $m \geq 1$ ，以至少  $1 - \delta$  的概率有

$$\ell(\mathcal{L}, \mathcal{D}) \leq \hat{\ell}(\mathcal{L}, D) + 2\beta + (4m\beta + M)\sqrt{\frac{\ln(1/\delta)}{2m}}, \quad (12.58)$$

$$\ell(\mathcal{L}, \mathcal{D}) \leq \ell_{loo}(\mathcal{L}, D) + \beta + (4m\beta + M)\sqrt{\frac{\ln(1/\delta)}{2m}}. \quad (12.59)$$

定理 12.8 给出了基于稳定性分析推导出的学习算法  $\mathcal{L}$  学得假设的泛化误差界。从式(12.58)可看出，经验损失与泛化损失之间差别的收敛率为  $\beta\sqrt{m}$ ；若  $\beta = O(\frac{1}{m})$ ，则可保证收敛率为  $O(\frac{1}{\sqrt{m}})$ 。与定理 12.3 和定理 12.6 比较可知，这与基于 VC 维和 Rademacher 复杂度得到的收敛率一致。

需注意，学习算法的稳定性分析所关注的是  $|\hat{\ell}(\mathcal{L}, D) - \ell(\mathcal{L}, \mathcal{D})|$ ，而假设空间复杂度分析所关注的是  $\sup_{h \in \mathcal{H}} |\hat{E}(h) - E(h)|$ ；也就是说，稳定性分析不必考虑假设空间中所有可能的假设，只需根据算法自身的特性(稳定性)来讨论输出假设  $\mathcal{L}_D$  的泛化误差界。那么，稳定性与可学习性之间有什么关系呢？

首先，必须假设  $\beta\sqrt{m} \rightarrow 0$ ，这样才能保证稳定的学习算法  $\mathcal{L}$  具有一定的泛化能力，即经验损失收敛于泛化损失，否则可学习性无从谈起。为便于计算，我们假定  $\beta = \frac{1}{m}$ ，代入式(12.58)可得

$$\ell(\mathcal{L}, \mathcal{D}) \leq \hat{\ell}(\mathcal{L}, D) + \frac{2}{m} + (4 + M)\sqrt{\frac{\ln(1/\delta)}{2m}}. \quad (12.60)$$

最小化经验误差和最小化经验损失有时并不相同，这是由于存在某些病态的损失函数  $\ell$  使得最小化经验损失并不是最小化经验误差。为简化讨论，本章假定最小化经验损失的同时会最小化经验误差。

对损失函数  $\ell$ ，若学习算法  $\mathcal{L}$  所输出的假设满足经验损失最小化，则称算法  $\mathcal{L}$  满足经验风险最小化 (Empirical Risk Minimization) 原则，简称算法是 ERM 的。关于学习算法的稳定性和可学习性，有如下定理：

**定理 12.9** 若学习算法  $\mathcal{L}$  是 ERM 且稳定的，则假设空间  $\mathcal{H}$  可学习。

**证明** 令  $g$  表示  $\mathcal{H}$  中具有最小泛化损失的假设，即

$$\ell(g, \mathcal{D}) = \min_{h \in \mathcal{H}} \ell(h, \mathcal{D}).$$

再令

$$\begin{aligned}\epsilon' &= \frac{\epsilon}{2}, \\ \frac{\delta}{2} &= 2 \exp\left(-2m(\epsilon')^2\right),\end{aligned}$$

由 Hoeffding 不等式(12.6)可知, 当  $m \geq \frac{2}{\epsilon^2} \ln \frac{4}{\delta}$  时,

$$|\ell(g, \mathcal{D}) - \hat{\ell}(g, D)| \leq \frac{\epsilon}{2}$$

以至少  $1 - \delta/2$  的概率成立. 令式(12.60)中

$$\frac{2}{m} + (4 + M) \sqrt{\frac{\ln(2/\delta)}{2m}} = \frac{\epsilon}{2},$$

解得  $m = O\left(\frac{1}{\epsilon^2} \ln \frac{1}{\delta}\right)$  使

$$\ell(\mathfrak{L}, \mathcal{D}) \leq \hat{\ell}(\mathfrak{L}, D) + \frac{\epsilon}{2}$$

以至少  $1 - \delta/2$  的概率成立. 从而可得

$$\begin{aligned}\ell(\mathfrak{L}, \mathcal{D}) - \ell(g, \mathcal{D}) &\leq \hat{\ell}(\mathfrak{L}, D) + \frac{\epsilon}{2} - \left(\hat{\ell}(g, D) - \frac{\epsilon}{2}\right) \\ &\leq \hat{\ell}(\mathfrak{L}, D) - \hat{\ell}(g, D) + \epsilon \\ &\leq \epsilon\end{aligned}$$

以至少  $1 - \delta$  的概率成立. 定理 12.9 得证. ■

对上面这个定理读者也许会纳闷, 为什么学习算法的稳定性能导出假设空间的可学习性? 学习算法和假设空间是两码事呀. 事实上, 要注意到稳定性与假设空间并非无关, 由稳定性的定义可知两者通过损失函数  $\ell$  联系起来.

## 12.7 阅读材料

[Valiant, 1984] 提出 PAC 学习, 由此产生了“计算学习理论”这个机器学习的分支领域. [Kearns and Vazirani, 1994] 是一本很好的入门教材. 该领域最

重要的学术会议是国际计算学习理论会议 (COLT).

VC 维的名字就来自两位作者的姓氏缩写.

VC 维由 [Vapnik and Chervonenkis, 1971] 提出, 它的出现使研究无限假设空间的复杂度成为可能. Sauer 引理由于 [Sauer, 1972] 而命名, 但 [Vapnik and Chervonenkis, 1971] 和 [Shelah, 1972] 也分别独立地推导出了该结果. 本章主要讨论了二分类问题, 对多分类问题, 可将 VC 维扩展为 Natarajan 维 [Natarajan, 1989; Ben-David et al., 1995].

Rademacher 复杂度最早被 [Koltchinskii and Panchenko, 2000] 引入机器学习, 由 [Bartlett and Mendelson, 2003] 而受到重视. [Bartlett et al., 2002] 提出了局部 Rademacher 复杂度, 对噪声数据可推导出更紧的泛化误差界.

机器学习算法稳定性分析方面的研究始于 [Bousquet and Elisseeff, 2002] 的工作, 此后很多学者对稳定性与可学习性之间的关系进行了讨论, [Mukherjee et al., 2006] 和 [Shalev-Shwartz et al., 2010] 证明了 ERM 稳定性与 ERM 可学习性之间的等价关系; 但并非所有学习算法都是 ERM 的, 因此 [Shalev-Shwartz et al., 2010] 进一步研究了 AERM (Asymptotical Empirical Risk Minimization) 稳定性与可学习性之间的关系.

本章介绍的内容都是关于确定性 (deterministic) 学习问题, 即对于每个示例  $\mathbf{x}$  都有一个确定的标记  $y$  与之对应; 大多数监督学习都属于确定性学习问题. 但还有一种随机性 (stochastic) 学习问题, 其中示例的标记可认为是属性的后验概率函数, 而不再是简单确定地属于某一类. 随机性学习问题的泛化误差界分析可参见 [Devroye et al., 1996].

**习题**

- 12.1** 试证明 Jensen 不等式(12.4).
- 12.2** 试证明引理 12.1.
- 提示: 令  $\delta = 2e^{-2m\epsilon^2}$ .
- 12.3** 试证明推论 12.1.
- 12.4** 试证明:  $\mathbb{R}^d$  空间中线性超平面构成的假设空间的 VC 维是  $d + 1$ .
- 12.5** 试计算决策树桩假设空间的 VC 维.
- 12.6** 试证明: 决策树分类器的假设空间 VC 维可以为无穷大.
- 12.7** 试证明: 最近邻分类器的假设空间 VC 维为无穷大.
- 12.8** 试证明常数函数  $c$  的 Rademacher 复杂度为 0.
- 12.9** 给定函数空间  $\mathcal{F}_1, \mathcal{F}_2$ , 试证明 Rademacher 复杂度  $R_m(\mathcal{F}_1 + \mathcal{F}_2) \leq R_m(\mathcal{F}_1) + R_m(\mathcal{F}_2)$ .
- 12.10\*** 考虑定理 12.8, 试讨论通过交叉验证法来估计学习算法泛化能力的合理性.

## 参考文献

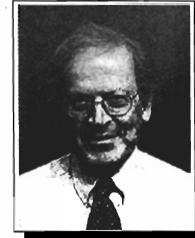
- Bartlett, P. L., O. Bousquet, and S. Mendelson. (2002). “Localized Rademacher complexities.” In *Proceedings of the 15th Annual Conference on Learning Theory (COLT)*, 44–58, Sydney, Australia.
- Bartlett, P. L. and S. Mendelson. (2003). “Rademacher and Gaussian complexities: Risk bounds and structural results.” *Journal of Machine Learning Research*, 3:463–482.
- Ben-David, S., N. Cesa-Bianchi, D. Haussler, and P. M. Long. (1995). “Characterizations of learnability for classes of  $\{0, \dots, n\}$ -valued functions.” *Journal of Computer and System Sciences*, 50(1):74–86.
- Bousquet, O. and A. Elisseeff. (2002). “Stability and generalization.” *Journal of Machine Learning Research*, 2:499–526.
- Devroye, L., L. Gyorfi, and G. Lugosi, eds. (1996). *A Probabilistic Theory of Pattern Recognition*. Springer, New York, NY.
- Hoeffding, W. (1963). “Probability inequalities for sums of bounded random variables.” *Journal of the American Statistical Association*, 58(301):13–30.
- Kearns, M. J. and U. V. Vazirani. (1994). *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA.
- Koltchinskii, V. and D. Panchenko. (2000). “Rademacher processes and bounding the risk of function learning.” In *High Dimensional Probability II* (E. Giné, D. M. Mason, and J. A. Wellner, eds.), 443–457, Birkhäuser Boston, Cambridge, MA.
- McDiarmid, C. (1989). “On the method of bounded differences.” *Surveys in Combinatorics*, 141(1):148–188.
- Mohri, M., A. Rostamizadeh, and A. Talwalkar, eds. (2012). *Foundations of Machine Learning*. MIT Press, Cambridge, MA.
- Mukherjee, S., P. Niyogi, T. Poggio, and R. M. Rifkin. (2006). “Learning theory: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization.” *Advances in Computational Mathematics*, 25(1-3):161–193.
- Natarajan, B. K. (1989). “On learning sets and functions.” *Machine Learning*, 4(1):67–97.

- Sauer, N. (1972). "On the density of families of sets." *Journal of Combinatorial Theory - Series A*, 13(1):145–147.
- Shalev-Shwartz, S., O. Shamir, N. Srebro, and K. Sridharan. (2010). "Learnability, stability and uniform convergence." *Journal of Machine Learning Research*, 11:2635–2670.
- Shelah, S. (1972). "A combinatorial problem; stability and order for models and theories in infinitary languages." *Pacific Journal of Mathematics*, 41 (1):247–261.
- Valiant, L. G. (1984). "A theory of the learnable." *Communications of the ACM*, 27(11):1134–1142.
- Vapnik, V. N. and A. Chervonenkis. (1971). "On the uniform convergence of relative frequencies of events to their probabilities." *Theory of Probability and Its Applications*, 16(2):264–280.

## 休息一会儿

### 小故事：计算学习理论之父莱斯利·维利昂特

计算机科学的绝大多数分支领域中都既有理论研究，也有应用研究，但当人们说到“理论计算机科学”时，通常是指一个特定的研究领域——TCS (Theoretical Computer Science)，它可看作计算机科学与数学的交叉，该领域中最著名的问题是“P?=NP”。



计算学习理论是机器学习的一个分支，它可认为是机器学习与理论计算机科学的交叉。提起计算学习理论，就必然要谈到英国计算机科学家莱斯利·维利昂特 (Leslie G. Valiant, 1949— )。维利昂特先后在剑桥大学国王学院、帝国理工学院学习，1974 年在华威大学获计算机科学博士学位，此后曾在卡耐基梅隆大学、利兹大学和爱丁堡大学任教，1982 年来到哈佛大学任计算机与应用数学讲席教授。1984 年他在《ACM通讯》发表了论文 “A theory of the learnable”。这篇论文首次提出了 PAC 学习，从而开创了计算学习理论的研究。2010 年 ACM 授予维利昂特图灵奖，以表彰他对 PAC 学习理论的开创性贡献，以及他对枚举和计算代数复杂性等其他一些理论计算机科学问题的重要贡献。颁奖词特别指出，维利昂特在 1984 年发表的论文创立了计算学习理论这个研究领域，使机器学习有了坚实的数学基础，扫清了学科发展的障碍。《ACM新闻》则以 “ACM Turing Award Goes to Innovator in Machine Learning” 为题对这位机器学习领域首位图灵奖得主的功绩大加褒扬。

# 第 13 章 半监督学习

## 13.1 未标记样本

我们在丰收季节来到瓜田, 满地都是西瓜, 瓜农抱来三四个瓜说这都是好瓜, 然后再指着地里的五六个瓜说这些还不好, 还需再生长若干天. 基于这些信息, 我们能否构建一个模型, 用于判别地里的哪些瓜是已该采摘的好瓜? 显然, 可将瓜农告诉我们的瓜作为正例和反例来训练一个分类器. 然而, 只用这不到十个瓜做训练样本, 有点太少了吧? 能不能把地里的那些瓜也用上呢?

形式化地看, 我们有训练样本集  $D_l = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_l, y_l)\}$ , 这  $l$  个样本的类别标记(即是否好瓜)已知, 称为“有标记”(labeled)样本; 此外, 还有  $D_u = \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \dots, \mathbf{x}_{l+u}\}$ ,  $l \ll u$ , 这  $u$  个样本的类别标记未知(即不知是否好瓜), 称为“未标记”(unlabeled)样本. 若直接使用传统监督学习技术, 则仅有  $D_l$  能用于构建模型,  $D_u$  所包含的信息被浪费了; 另一方面, 若  $D_l$  较小, 则由于训练样本不足, 学得模型的泛化能力往往不佳. 那么, 能否在构建模型的过程中将  $D_u$  利用起来呢?

一个简单的做法, 是将  $D_u$  中的示例全部标记后用于学习. 这就相当于请瓜农把地里的瓜全都检查一遍, 告诉我们哪些是好瓜, 哪些不是好瓜, 然后再用于模型训练. 显然, 这样做需耗费瓜农大量时间和精力. 有没有“便宜”一点的办法呢?

例如基于  $D_l$  训练一个 SVM, 挑选距离分类超平面最近的未标记样本来进行查询.

即尽量少向瓜农询问.

我们可以用  $D_l$  先训练一个模型, 拿这个模型去地里挑一个瓜, 询问瓜农好不好, 然后把这个新获得的有标记样本加入  $D_l$  中重新训练一个模型, 再去挑瓜, ……这样, 若每次都挑出对改善模型性能帮助大的瓜, 则只需询问瓜农比较少的瓜就能构建出比较强的模型, 从而大幅降低标记成本. 这样的学习方式称为“主动学习”(active learning), 其目标是使用尽量少的“查询”(query)来获得尽量好的性能.

显然, 主动学习引入了额外的专家知识, 通过与外界的交互来将部分未标记样本转变为有标记样本. 若不与专家交互, 没有获得额外信息, 还能利用未标记样本来提高泛化性能吗?

答案是“Yes!”, 有点匪夷所思?

事实上, 未标记样本虽未直接包含标记信息, 但若它们与有标记样本是从同样的数据源独立同分布采样而来, 则它们所包含的关于数据分布的信息对建立模型将大有裨益. 图 13.1 给出了一个直观的例示. 若仅基于图中的一个正例和一个反例, 则由于待判别样本恰位于两者正中间, 大体上只能随机猜测; 若能观察到图中的未标记样本, 则将很有把握地判别为正例.

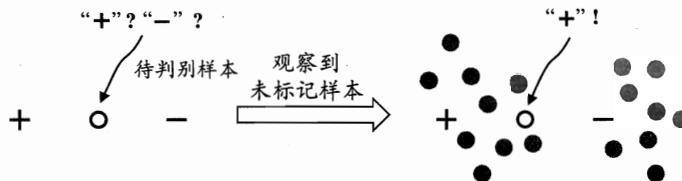


图 13.1 未标记样本效用的例示. 右边的灰色点表示未标记样本

让学习器不依赖外界交互、自动地利用未标记样本来提升学习性能, 就是半监督学习(semi-supervised learning). 半监督学习的现实需求非常强烈, 因为在现实应用中往往能容易地收集到大量未标记样本, 而获取“标记”却需耗费人力、物力. 例如, 在进行计算机辅助医学影像分析时, 可以从医院获得大量医学影像, 但若希望医学专家把影像中的病灶全都标识出来则是不现实的.“有标记数据少, 未标记数据多”这个现象在互联网应用中更明显, 例如在进行网页推荐时需请用户标记出感兴趣的网页, 但很少有用户愿花很多时间来提供标记, 因此, 有标记网页样本少, 但互联网上存在无数网页可作为未标记样本来使用. 半监督学习恰是提供了一条利用“廉价”的未标记样本的途径.

要利用未标记样本, 必然要做一些将未标记样本所揭示的数据分布信息与类别标记相联系的假设. 最常见的是“聚类假设”(cluster assumption), 即假设数据存在簇结构, 同一个簇的样本属于同一个类别. 图 13.1 就是基于聚类假设来利用未标记样本, 由于待预测样本与正例样本通过未标记样本的“撮合”聚在一起, 与相对分离的反例样本相比, 待判别样本更可能属于正类. 半监督学习中另一种常见的假设是“流形假设”(manifold assumption), 即假设数据分布在一个流形结构上, 邻近的样本拥有相似的输出值.“邻近”程度常用“相似”程度来刻画, 因此, 流形假设可看作聚类假设的推广, 但流形假设对输出值没有限制, 因此比聚类假设的适用范围更广, 可用于更多类型的学习任务. 事实上, 无论聚类假设还是流形假设, 其本质都是“相似的样本拥有相似的输出”这个基本假设.

“流形”概念是流形学习的基础, 参见 10.5 节.

聚类假设考虑的是类别标记, 通常用于分类任务.

半监督学习可进一步划分为纯(pure)半监督学习和直推学习(transductive learning), 前者假定训练数据中的未标记样本并非待预测的数据, 而后者则假定学习过程中所考虑的未标记样本恰是待预测数据, 学习的目的就是在这些未标记样本上获得最优泛化性能. 换言之, 纯半监督学习是基于“开放世界”假设, 希望学得模型能适用于训练过程中未观察到的数据; 而直推学习是基于“封闭世界”假设, 仅试图对学习过程中观察到的未标记数据进行预测. 图 13.2 直观地显示出主动学习、纯半监督学习、直推学习的区别. 需注意的是, 纯半监督学习和直推学习常合称为半监督学习, 本书也采取这一态度, 在需专门区分时会特别说明.

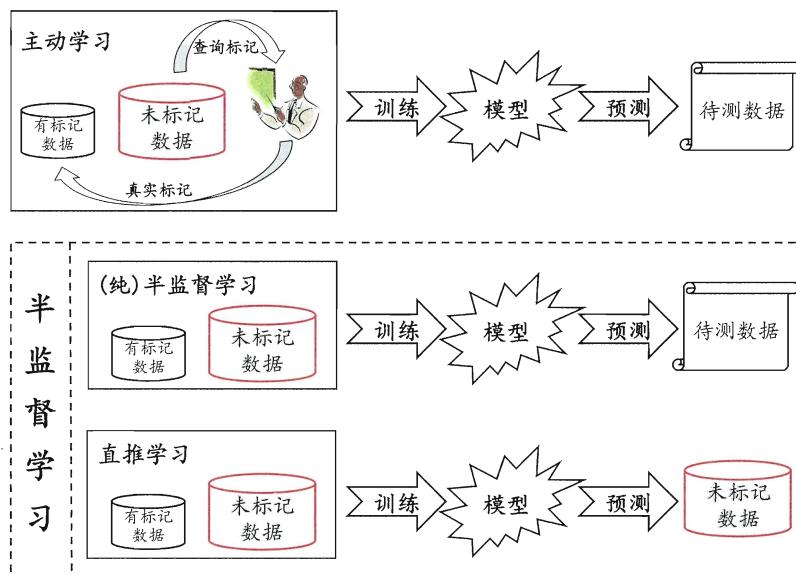


图 13.2 主动学习、(纯)半监督学习、直推学习

## 13.2 生成式方法

生成式方法(generative methods)是直接基于生成式模型的方法. 此类方法假设所有数据(无论是否有标记)都是由同一个潜在的模型“生成”的. 这个假设使得我们能通过潜在模型的参数将未标记数据与学习目标联系起来, 而未标记数据的标记则可看作模型的缺失参数, 通常可基于EM 算法进行极大似然估计求解. 此类方法的区别主要在于生成式模型的假设, 不同的模型假设将产生不同的方法.

这个假设意味着混合成分与类别之间一一对应。

给定样本  $\mathbf{x}$ , 其真实类别标记为  $y \in \mathcal{Y}$ , 其中  $\mathcal{Y} = \{1, 2, \dots, N\}$  为所有可能的类别。假设样本由高斯混合模型生成, 且每个类别对应一个高斯混合成分。换言之, 数据样本是基于如下概率密度生成:

$$p(\mathbf{x}) = \sum_{i=1}^N \alpha_i \cdot p(\mathbf{x} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \quad (13.1)$$

高斯混合模型参见 9.4 节。

其中, 混合系数  $\alpha_i \geq 0$ ,  $\sum_{i=1}^N \alpha_i = 1$ ;  $p(\mathbf{x} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$  是样本  $\mathbf{x}$  属于第  $i$  个高斯混合成分的概率;  $\boldsymbol{\mu}_i$  和  $\boldsymbol{\Sigma}_i$  为该高斯混合成分的参数。

令  $f(\mathbf{x}) \in \mathcal{Y}$  表示模型  $f$  对  $\mathbf{x}$  的预测标记,  $\Theta \in \{1, 2, \dots, N\}$  表示样本  $\mathbf{x}$  隶属的高斯混合成分。由最大化后验概率可知

$$\begin{aligned} f(\mathbf{x}) &= \arg \max_{j \in \mathcal{Y}} p(y = j | \mathbf{x}) \\ &= \arg \max_{j \in \mathcal{Y}} \sum_{i=1}^N p(y = j, \Theta = i | \mathbf{x}) \\ &= \arg \max_{j \in \mathcal{Y}} \sum_{i=1}^N p(y = j | \Theta = i, \mathbf{x}) \cdot p(\Theta = i | \mathbf{x}), \end{aligned} \quad (13.2)$$

其中

$$p(\Theta = i | \mathbf{x}) = \frac{\alpha_i \cdot p(\mathbf{x} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{i=1}^N \alpha_i \cdot p(\mathbf{x} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)} \quad (13.3)$$

为样本  $\mathbf{x}$  由第  $i$  个高斯混合成分生成的后验概率,  $p(y = j | \Theta = i, \mathbf{x})$  为  $\mathbf{x}$  由第  $i$  个高斯混合成分生成且其类别为  $j$  的概率。由于假设每个类别对应一个高斯混合成分, 因此  $p(y = j | \Theta = i, \mathbf{x})$  仅与样本  $\mathbf{x}$  所属的高斯混合成分  $\Theta$  有关, 可用  $p(y = j | \Theta = i)$  代替。不失一般性, 假定第  $i$  个类别对应于第  $i$  个高斯混合成分, 即  $p(y = j | \Theta = i) = 1$  当且仅当  $i = j$ , 否则  $p(y = j | \Theta = i) = 0$ 。

不难发现, 式(13.2)中估计  $p(y = j | \Theta = i, \mathbf{x})$  需知道样本的标记, 因此仅能使用有标记数据; 而  $p(\Theta = i | \mathbf{x})$  不涉及样本标记, 因此有标记和未标记数据均可利用, 通过引入大量的未标记数据, 对这一项的估计可望由于数据量的增长而更为准确, 于是式(13.2)整体的估计可能会更准确。由此可清楚地看出未标记数据何以能辅助提高分类模型的性能。

给定有标记样本集  $D_l = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_l, y_l)\}$  和未标记样本集

半监督学习中通常假设未标记样本数远大于有标记样本数, 虽然此假设实际并非必须。 $D_u = \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \dots, \mathbf{x}_{l+u}\}$ ,  $l \ll u$ ,  $l + u = m$ . 假设所有样本独立同分布, 且都是由同一个高斯混合模型生成的. 用极大似然法来估计高斯混合模型的参数  $\{(\alpha_i, \mu_i, \Sigma_i) \mid 1 \leq i \leq N\}$ ,  $D_l \cup D_u$  的对数似然是

$$\begin{aligned} LL(D_l \cup D_u) &= \sum_{(\mathbf{x}_j, y_j) \in D_l} \ln \left( \sum_{i=1}^N \alpha_i \cdot p(\mathbf{x}_j \mid \mu_i, \Sigma_i) \cdot p(y_j \mid \Theta = i, \mathbf{x}_j) \right) \\ &\quad + \sum_{\mathbf{x}_j \in D_u} \ln \left( \sum_{i=1}^N \alpha_i \cdot p(\mathbf{x}_j \mid \mu_i, \Sigma_i) \right). \end{aligned} \quad (13.4)$$

高斯混合模型聚类的  
EM 算法参见 9.4 节。

可通过有标记数据对模  
型参数进行初始化。

式(13.4)由两项组成: 基于有标记数据  $D_l$  的有监督项和基于未标记数据  $D_u$  的无监督项. 显然, 高斯混合模型参数估计可用 EM 算法求解, 迭代更新式如下:

- E 步: 根据当前模型参数计算未标记样本  $\mathbf{x}_j$  属于各高斯混合成分的概率

$$\gamma_{ji} = \frac{\alpha_i \cdot p(\mathbf{x}_j \mid \mu_i, \Sigma_i)}{\sum_{i=1}^N \alpha_i \cdot p(\mathbf{x}_j \mid \mu_i, \Sigma_i)}; \quad (13.5)$$

- M 步: 基于  $\gamma_{ji}$  更新模型参数, 其中  $l_i$  表示第  $i$  类的有标记样本数目

$$\mu_i = \frac{1}{\sum_{\mathbf{x}_j \in D_u} \gamma_{ji} + l_i} \left( \sum_{\mathbf{x}_j \in D_u} \gamma_{ji} \mathbf{x}_j + \sum_{(\mathbf{x}_j, y_j) \in D_l \wedge y_j=i} \mathbf{x}_j \right), \quad (13.6)$$

$$\begin{aligned} \Sigma_i &= \frac{1}{\sum_{\mathbf{x}_j \in D_u} \gamma_{ji} + l_i} \left( \sum_{\mathbf{x}_j \in D_u} \gamma_{ji} (\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \mu_i)^T \right. \\ &\quad \left. + \sum_{(\mathbf{x}_j, y_j) \in D_l \wedge y_j=i} (\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \mu_i)^T \right), \end{aligned} \quad (13.7)$$

$$\alpha_i = \frac{1}{m} \left( \sum_{\mathbf{x}_j \in D_u} \gamma_{ji} + l_i \right). \quad (13.8)$$

以上过程不断迭代直至收敛, 即可获得模型参数. 然后由式(13.3)和(13.2)就能对样本进行分类.

将上述过程中的高斯混合模型换成混合专家模型 [Miller and Uyar, 1997]、朴素贝叶斯模型 [Nigam et al., 2000] 等即可推导出其他的生成式半

监督学习方法。此类方法简单，易于实现，在有标记数据极少的情形下往往比其他方法性能更好。然而，此类方法有一个关键：模型假设必须准确，即假设的生成式模型必须与真实数据分布吻合；否则利用未标记数据反倒会降低泛化性能 [Cozman and Cohen, 2002]。遗憾的是，在现实任务中往往很难事先做出准确的模型假设，除非拥有充分可靠的领域知识。

### 13.3 半监督SVM

半监督支持向量机 (Semi-Supervised Support Vector Machine, 简称 S3VM) 是支持向量机在半监督学习上的推广。在不考虑未标记样本时，支持向量机试图找到最大间隔划分超平面，而在考虑未标记样本后，S3VM 试图找到能将两类有标记样本分开，且穿过数据低密度区域的划分超平面，如图 13.3 所示，这里的基本假设是“低密度分隔” (low-density separation)，显然，这是聚类假设在考虑了线性超平面划分后的推广。

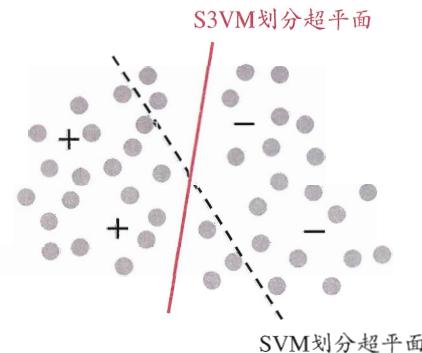


图 13.3 半监督支持向量机与低密度分隔（“+” “-” 分别表示有标记的正、反例，灰色点表示未标记样本）

半监督支持向量机中最著名的是 TSVM (Transductive Support Vector Machine) [Joachims, 1999]。与标准 SVM 一样，TSVM 也是针对二分类问题的学习方法。TSVM 试图考虑对未标记样本进行各种可能的标记指派(label assignment)，即尝试将每个未标记样本分别作为正例或反例，然后在所有这些结果中，寻求一个在所有样本(包括有标记样本和进行了标记指派的未标记样本)上间隔最大化的划分超平面。一旦划分超平面得以确定，未标记样本的最终标记指派就是其预测结果。

形式化地说, 给定  $D_l = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_l, y_l)\}$  和  $D_u = \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \dots, \mathbf{x}_{l+u}\}$ , 其中  $y_i \in \{-1, +1\}$ ,  $l \ll u$ ,  $l + u = m$ . TSVM 的学习目标是为  $D_u$  中的样本给出预测标记  $\hat{\mathbf{y}} = (\hat{y}_{l+1}, \hat{y}_{l+2}, \dots, \hat{y}_{l+u})$ ,  $\hat{y}_i \in \{-1, +1\}$ , 使得

$$\begin{aligned} \min_{\mathbf{w}, b, \hat{\mathbf{y}}, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + C_l \sum_{i=1}^l \xi_i + C_u \sum_{i=l+1}^m \xi_i \\ \text{s.t.} \quad & y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, 2, \dots, l, \\ & \hat{y}_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = l+1, l+2, \dots, m, \\ & \xi_i \geq 0, \quad i = 1, 2, \dots, m, \end{aligned} \quad (13.9)$$

其中,  $(\mathbf{w}, b)$  确定了一个划分超平面;  $\xi$  为松弛向量,  $\xi_i (i = 1, 2, \dots, l)$  对应于有标记样本,  $\xi_i (i = l+1, l+2, \dots, m)$  对应于未标记样本;  $C_l$  与  $C_u$  是由用户指定的用于平衡模型复杂度、有标记样本与未标记样本重要程度的折中参数.

显然, 尝试未标记样本的各种标记指派是一个穷举过程, 仅当未标记样本很少时才有可能直接求解. 在一般情形下, 必须考虑更高效的优化策略.

TSVM 采用局部搜索来迭代地寻找式(13.9)的近似解. 具体来说, 它先利用有标记样本学得一个 SVM, 即忽略式(13.9)中关于  $D_u$  与  $\hat{\mathbf{y}}$  的项及约束. 然后, 利用这个 SVM 对未标记数据进行标记指派(label assignment), 即将 SVM 预测的结果作为“伪标记”(pseudo-label)赋予未标记样本. 此时  $\hat{\mathbf{y}}$  成为已知, 将其代入式(13.9)即得到一个标准 SVM 问题, 于是可求解出新的划分超平面和松弛向量; 注意到此时未标记样本的伪标记很可能不准确, 因此  $C_u$  要设置为比  $C_l$  小的值, 使有标记样本所起作用更大. 接下来, TSVM 找出两个标记指派为异类且很可能发生错误的未标记样本, 交换它们的标记, 再重新基于式(13.9)求解出更新后的划分超平面和松弛向量, 然后再找出两个标记指派为异类且很可能发生错误的未标记样本, ……标记指派调整完成后, 逐渐增大  $C_u$  以提高未标记样本对优化目标的影响, 进行下一轮标记指派调整, 直至  $C_u = C_l$  为止. 此时求解得到的 SVM 不仅给未标记样本提供了标记, 还能对训练过程中未见的示例进行预测. TSVM 的算法描述如图 13.4 所示.

在对未标记样本进行标记指派及调整的过程中, 有可能出现类别不平衡问题, 即某类的样本远多于另一类, 这将对 SVM 的训练造成困扰. 为了减轻类别不平衡性所造成的不利影响, 可对图 13.4 的算法稍加改进: 将优化目标中的  $C_u$  项拆分为  $C_u^+$  与  $C_u^-$  两项, 分别对应基于伪标记而当作正、反例使用的未标记样本, 并在初始化时令

类别不平衡问题及式(13.10)的缘由见 3.6 节.

---

输入: 有标记样本集  $D_l = \{(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)\}$ ;  
 未标记样本集  $D_u = \{x_{l+1}, x_{l+2}, \dots, x_{l+u}\}$ ;  
 折中参数  $C_l, C_u$ .

过程:

- 1: 用  $D_l$  训练一个 SVM<sub>l</sub>;
- 2: 用 SVM<sub>l</sub> 对  $D_u$  中样本进行预测, 得到  $\hat{y} = (\hat{y}_{l+1}, \hat{y}_{l+2}, \dots, \hat{y}_{l+u})$ ;
- 3: 初始化  $C_u \ll C_l$ ;
- 4: **while**  $C_u < C_l$  **do**
- 5: 基于  $D_l, D_u, \hat{y}, C_l, C_u$  求解式(13.9), 得到  $(w, b), \xi$ ;
- 6: **while**  $\exists\{i, j \mid (\hat{y}_i \hat{y}_j < 0) \wedge (\xi_i > 0) \wedge (\xi_j > 0) \wedge (\xi_i + \xi_j > 2)\}$  **do**
- 7:  $\hat{y}_i = -\hat{y}_j$ ;
- 8:  $\hat{y}_j = -\hat{y}_i$ ;
- 9: 基于  $D_l, D_u, \hat{y}, C_l, C_u$  重新求解式(13.9), 得到  $(w, b), \xi$
- 10: **end while**
- 11:  $C_u = \min\{2C_u, C_l\}$
- 12: **end while**

输出: 未标记样本的预测结果:  $\hat{y} = (\hat{y}_{l+1}, \hat{y}_{l+2}, \dots, \hat{y}_{l+u})$

---

图 13.4 TSVVM 算法

$$C_u^+ = \frac{u_-}{u_+} C_u^- , \quad (13.10)$$

其中  $u_+$  与  $u_-$  为基于伪标记而当作正、反例使用的未标记样本数.

在图 13.4 算法的第 6–10 行中, 若存在一对未标记样本  $x_i$  与  $x_j$ , 其标记指派  $\hat{y}_i$  与  $\hat{y}_j$  不同, 且对应的松弛变量满足  $\xi_i + \xi_j > 2$ , 则意味着  $\hat{y}_i$  与  $\hat{y}_j$  很可能是错误的, 需对二者进行交换后重新求解式(13.9), 这样每轮迭代后均可使式(13.9)的目标函数值下降.

收敛性证明参阅  
[Joachims, 1999].

显然, 搜寻标记指派可能出错的每一对未标记样本进行调整, 是一个涉及巨大计算开销的大规模优化问题. 因此, 半监督 SVM 研究的一个重点是如何设计出高效的优化求解策略, 由此发展出很多方法, 如基于图核(graph kernel)函数梯度下降的 LDS [Chapelle and Zien, 2005]、基于标记均值估计的 meanS3VM [Li et al., 2009] 等.

### 13.4 图半监督学习

给定一个数据集, 我们可将其映射为一个图, 数据集中每个样本对应于图中一个结点, 若两个样本之间的相似度很高(或相关性很强), 则对应的结点之间存在一条边, 边的“强度”(strength)正比于样本之间的相似度(或相关性). 我们可将有标记样本所对应的结点想象为染过色, 而未标记样本所对应的结点尚

未染色。于是，半监督学习就对应于“颜色”在图上扩散或传播的过程。由于一个图对应了一个矩阵，这就使得我们能基于矩阵运算来进行半监督学习算法的推导与分析。

给定  $D_l = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_l, y_l)\}$  和  $D_u = \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \dots, \mathbf{x}_{l+u}\}$ ,  $l \ll u$ ,  $l + u = m$ . 我们先基于  $D_l \cup D_u$  构建一个图  $G = (V, E)$ , 其中结点集  $V = \{\mathbf{x}_1, \dots, \mathbf{x}_l, \mathbf{x}_{l+1}, \dots, \mathbf{x}_{l+u}\}$ , 边集  $E$  可表示为一个亲和矩阵 (affinity matrix), 常基于高斯函数定义为

$$(\mathbf{W})_{ij} = \begin{cases} \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right), & \text{if } i \neq j; \\ 0, & \text{otherwise,} \end{cases} \quad (13.11)$$

其中  $i, j \in \{1, 2, \dots, m\}$ ,  $\sigma > 0$  是用户指定的高斯函数带宽参数。

假定从图  $G = (V, E)$  将学得一个实值函数  $f : V \rightarrow \mathbb{R}$ , 其对应的分类规则为:  $y_i = \text{sign}(f(\mathbf{x}_i))$ ,  $y_i \in \{-1, +1\}$ . 直观上看, 相似的样本应具有相似的标记, 于是可定义关于  $f$  的“能量函数”(energy function) [Zhu et al., 2003]:

能量函数最小化时即得到最优结果。

$$\begin{aligned} E(f) &= \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m (\mathbf{W})_{ij} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 \\ &= \frac{1}{2} \left( \sum_{i=1}^m d_i f^2(\mathbf{x}_i) + \sum_{j=1}^m d_j f^2(\mathbf{x}_j) - 2 \sum_{i=1}^m \sum_{j=1}^m (\mathbf{W})_{ij} f(\mathbf{x}_i) f(\mathbf{x}_j) \right) \\ &= \sum_{i=1}^m d_i f^2(\mathbf{x}_i) - \sum_{i=1}^m \sum_{j=1}^m (\mathbf{W})_{ij} f(\mathbf{x}_i) f(\mathbf{x}_j) \\ &= \mathbf{f}^T (\mathbf{D} - \mathbf{W}) \mathbf{f}, \end{aligned} \quad (13.12)$$

其中  $\mathbf{f} = (f_l^T f_u^T)^T$ ,  $\mathbf{f}_l = (f(\mathbf{x}_1); f(\mathbf{x}_2); \dots; f(\mathbf{x}_l))$ ,  $\mathbf{f}_u = (f(\mathbf{x}_{l+1}); f(\mathbf{x}_{l+2}); \dots; f(\mathbf{x}_{l+u}))$  分别为函数  $f$  在有标记样本与未标记样本上的预测结果,  $\mathbf{D} = \text{diag}(d_1, d_2, \dots, d_{l+u})$  是一个对角矩阵, 其对角元素  $d_i = \sum_{j=1}^{l+u} (\mathbf{W})_{ij}$  为矩阵  $\mathbf{W}$  的第  $i$  行元素之和。

$\mathbf{W}$  为对称矩阵, 因此  $d_i$  亦为  $\mathbf{W}$  第  $i$  列元素之和。

具有最小能量的函数  $f$  在有标记样本上满足  $f(\mathbf{x}_i) = y_i$  ( $i = 1, 2, \dots, l$ ), 在未标记样本上满足  $\Delta \mathbf{f} = \mathbf{0}$ , 其中  $\Delta = \mathbf{D} - \mathbf{W}$  为拉普拉斯矩阵 (Laplacian matrix). 以第  $l$  行与第  $l$  列为界, 采用分块矩阵表示方式:  $\mathbf{W} = \begin{bmatrix} \mathbf{W}_{ll} & \mathbf{W}_{lu} \\ \mathbf{W}_{ul} & \mathbf{W}_{uu} \end{bmatrix}$ ,

$\mathbf{D} = \begin{bmatrix} \mathbf{D}_{ll} & \mathbf{0}_{lu} \\ \mathbf{0}_{ul} & \mathbf{D}_{uu} \end{bmatrix}$ , 则式(13.12)可重写为

$$E(f) = (\mathbf{f}_l^T \mathbf{f}_u^T) \left( \begin{bmatrix} \mathbf{D}_{ll} & \mathbf{0}_{lu} \\ \mathbf{0}_{ul} & \mathbf{D}_{uu} \end{bmatrix} - \begin{bmatrix} \mathbf{W}_{ll} & \mathbf{W}_{lu} \\ \mathbf{W}_{ul} & \mathbf{W}_{uu} \end{bmatrix} \right) \begin{bmatrix} \mathbf{f}_l \\ \mathbf{f}_u \end{bmatrix} \quad (13.13)$$

$$= \mathbf{f}_l^T (\mathbf{D}_{ll} - \mathbf{W}_{ll}) \mathbf{f}_l - 2 \mathbf{f}_u^T \mathbf{W}_{ul} \mathbf{f}_l + \mathbf{f}_u^T (\mathbf{D}_{uu} - \mathbf{W}_{uu}) \mathbf{f}_u. \quad (13.14)$$

由  $\frac{\partial E(f)}{\partial \mathbf{f}_u} = \mathbf{0}$  可得

$$\mathbf{f}_u = (\mathbf{D}_{uu} - \mathbf{W}_{uu})^{-1} \mathbf{W}_{ul} \mathbf{f}_l. \quad (13.15)$$

令

$$\begin{aligned} \mathbf{P} &= \mathbf{D}^{-1} \mathbf{W} = \begin{bmatrix} \mathbf{D}_{ll}^{-1} & \mathbf{0}_{lu} \\ \mathbf{0}_{ul} & \mathbf{D}_{uu}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{W}_{ll} & \mathbf{W}_{lu} \\ \mathbf{W}_{ul} & \mathbf{W}_{uu} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{D}_{ll}^{-1} \mathbf{W}_{ll} & \mathbf{D}_{ll}^{-1} \mathbf{W}_{lu} \\ \mathbf{D}_{uu}^{-1} \mathbf{W}_{ul} & \mathbf{D}_{uu}^{-1} \mathbf{W}_{uu} \end{bmatrix}, \end{aligned} \quad (13.16)$$

即  $\mathbf{P}_{uu} = \mathbf{D}_{uu}^{-1} \mathbf{W}_{uu}$ ,  $\mathbf{P}_{ul} = \mathbf{D}_{uu}^{-1} \mathbf{W}_{ul}$ , 则式(13.15)可重写为

$$\begin{aligned} \mathbf{f}_u &= (\mathbf{D}_{uu}(\mathbf{I} - \mathbf{D}_{uu}^{-1} \mathbf{W}_{uu}))^{-1} \mathbf{W}_{ul} \mathbf{f}_l \\ &= (\mathbf{I} - \mathbf{D}_{uu}^{-1} \mathbf{W}_{uu})^{-1} \mathbf{D}_{uu}^{-1} \mathbf{W}_{ul} \mathbf{f}_l \\ &= (\mathbf{I} - \mathbf{P}_{uu})^{-1} \mathbf{P}_{ul} \mathbf{f}_l. \end{aligned} \quad (13.17)$$

于是, 将  $D_l$  上的标记信息作为  $\mathbf{f}_l = (y_1; y_2; \dots; y_l)$  代入式(13.17), 即可利用求得的  $\mathbf{f}_u$  对未标记样本进行预测.

上面描述的是一个针对二分类问题的标记传播(label propagation)方法, 下面来看一个适用于多分类问题的标记传播方法 [Zhou et al., 2004].

假定  $y_i \in \mathcal{Y}$ , 仍基于  $D_l \cup D_u$  构建一个图  $G = (V, E)$ , 其中结点集  $V = \{\mathbf{x}_1, \dots, \mathbf{x}_l, \dots, \mathbf{x}_{l+u}\}$ , 边集  $E$  所对应的  $\mathbf{W}$  仍使用式(13.11), 对角矩阵  $\mathbf{D} = \text{diag}(d_1, d_2, \dots, d_{l+u})$  的对角元素  $d_i = \sum_{j=1}^{l+u} (\mathbf{W})_{ij}$ . 定义一个  $(l+u) \times |\mathcal{Y}|$  的非负标记矩阵  $\mathbf{F} = (\mathbf{F}_1^T, \mathbf{F}_2^T, \dots, \mathbf{F}_{l+u}^T)^T$ , 其第  $i$  行元素  $\mathbf{F}_i = ((\mathbf{F})_{i1}, (\mathbf{F})_{i2}, \dots, (\mathbf{F})_{i|\mathcal{Y}|})$  为示例  $\mathbf{x}_i$  的标记向量, 相应的分类规则为:  $y_i = \arg \max_{1 \leq j \leq |\mathcal{Y}|} (\mathbf{F})_{ij}$ .

对  $i = 1, 2, \dots, m$ ,  $j = 1, 2, \dots, |\mathcal{Y}|$ , 将  $\mathbf{F}$  初始化为

$$\mathbf{F}(0) = (\mathbf{Y})_{ij} = \begin{cases} 1, & \text{if } (1 \leq i \leq l) \wedge (y_i = j); \\ 0, & \text{otherwise.} \end{cases} \quad (13.18)$$

显然,  $\mathbf{Y}$  的前  $l$  行就是  $l$  个有标记样本的标记向量.

基于  $\mathbf{W}$  构造一个标记传播矩阵  $\mathbf{S} = \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}$ , 其中  $\mathbf{D}^{-\frac{1}{2}} = \text{diag}\left(\frac{1}{\sqrt{d_1}}, \frac{1}{\sqrt{d_2}}, \dots, \frac{1}{\sqrt{d_{l+u}}}\right)$ , 于是有迭代计算式

$$\mathbf{F}(t+1) = \alpha \mathbf{SF}(t) + (1 - \alpha) \mathbf{Y}, \quad (13.19)$$

其中  $\alpha \in (0, 1)$  为用户指定的参数, 用于对标记传播项  $\mathbf{SF}(t)$  与初始化项  $\mathbf{Y}$  的重要性进行折中. 基于式(13.19)迭代至收敛可得

$$\mathbf{F}^* = \lim_{t \rightarrow \infty} \mathbf{F}(t) = (1 - \alpha)(\mathbf{I} - \alpha \mathbf{S})^{-1} \mathbf{Y}, \quad (13.20)$$

由  $\mathbf{F}^*$  可获得  $D_u$  中样本的标记  $(\hat{y}_{l+1}, \hat{y}_{l+2}, \dots, \hat{y}_{l+u})$ . 算法描述如图 13.5 所示.

---

**输入:** 有标记样本集  $D_l = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_l, y_l)\}$ ;  
未标记样本集  $D_u = \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \dots, \mathbf{x}_{l+u}\}$ ;  
构图参数  $\sigma$ ;  
折中参数  $\alpha$ .

**过程:**

- 1: 基于式(13.11)和参数  $\sigma$  得到  $\mathbf{W}$ ;
- 2: 基于  $\mathbf{W}$  构造标记传播矩阵  $\mathbf{S} = \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}$ ;
- 3: 根据式(13.18)初始化  $\mathbf{F}(0)$ ;
- 4:  $t = 0$ ;
- 5: **repeat**
- 6:    $\mathbf{F}(t+1) = \alpha \mathbf{SF}(t) + (1 - \alpha) \mathbf{Y}$ ;
- 7:    $t = t + 1$
- 8: **until** 迭代收敛至  $\mathbf{F}^*$
- 9: **for**  $i = l+1, l+2, \dots, l+u$  **do**
- 10:    $y_i = \arg \max_{1 \leq j \leq |\mathcal{Y}|} (\mathbf{F}^*)_{ij}$
- 11: **end for**

---

**输出:** 未标记样本的预测结果:  $\hat{\mathbf{y}} = (\hat{y}_{l+1}, \hat{y}_{l+2}, \dots, \hat{y}_{l+u})$

---

图 13.5 迭代式标记传播算法

事实上, 图 13.5 的算法对应于正则化框架 [Zhou et al., 2004]

$$\min_{\mathbf{F}} \frac{1}{2} \left( \sum_{i,j=1}^{l+u} (\mathbf{W})_{ij} \left\| \frac{1}{\sqrt{d_i}} \mathbf{F}_i - \frac{1}{\sqrt{d_j}} \mathbf{F}_j \right\|^2 \right) + \mu \sum_{i=1}^l \|\mathbf{F}_i - \mathbf{Y}_i\|^2, \quad (13.21)$$

其中  $\mu > 0$  为正则化参数. 当  $\mu = \frac{1-\alpha}{\alpha}$  时, 式(13.21)的最优解恰为图 13.5 算法的迭代收敛解  $\mathbf{F}^*$ .

式(13.21)右边第二项是迫使学得结果在有标记样本上的预测与真实标记尽可能相同, 而第一项则迫使相近样本具有相似的标记, 显然, 它与式(13.12)都是基于半监督学习的基本假设, 不同的是式(13.21)考虑离散的类别标记, 而式(13.12)则是考虑输出连续值.

图半监督学习方法在概念上相当清晰, 且易于通过对所涉矩阵运算的分析来探索算法性质. 但此类算法的缺陷也相当明显. 首先是在存储开销上, 若样本数为  $O(m)$ , 则算法中所涉及的矩阵规模为  $O(m^2)$ , 这使得此类算法很难直接处理大规模数据; 另一方面, 由于构图过程仅能考虑训练样本集, 难以判知新样本在图中的位置, 因此, 在接收到新样本时, 或是将其加入原数据集对图进行重构并重新进行标记传播, 或是需引入额外的预测机制, 例如将  $D_l$  和经标记传播后得到标记的  $D_u$  合并作为训练集, 另外训练一个学习器例如支持向量机来对新样本进行预测.

### 13.5 基于分歧的方法

与生成式方法、半监督 SVM、图半监督学习等基于单学习器利用未标记数据不同, 基于分歧的方法(disagreement-based methods) 使用多学习器, 而学习器之间的“分歧”(disagreement)对未标记数据的利用至关重要.

disagreement 亦称 diversity.

“协同训练”(co-training) [Blum and Mitchell, 1998] 是此类方法的重要代表, 它最初是针对“多视图”(multi-view)数据设计的, 因此也被看作“多视图学习”(multi-view learning)的代表. 在介绍协同训练之前, 我们先看看什么是多视图数据.

在不少现实应用中, 一个数据对象往往同时拥有多个“属性集”(attribute set), 每个属性集就构成了一个“视图”(view). 例如对一部电影来说, 它拥有多个属性集: 图像画面信息所对应的属性集、声音信息所对应的属性集、字幕信息所对应的属性集、甚至网上的宣传讨论所对应的属性集等. 每个属性集都可看作一个视图. 为简化讨论, 暂且仅考虑图像画面属性集所构成的视图和声音属性集所构成的视图. 于是, 一个电影片段可表示为样本  $(\langle \mathbf{x}^1, \mathbf{x}^2 \rangle, y)$ , 其中  $\mathbf{x}^i$  是样本在视图  $i$  中的示例, 即基于该视图属性描述而得的属性向量, 不妨假定  $\mathbf{x}^1$  为图像视图中的属性向量,  $\mathbf{x}^2$  为声音视图中的属性向量;  $y$  是标记, 假定是电影的类型, 例如“动作片”、“爱情片”等.  $(\langle \mathbf{x}^1, \mathbf{x}^2 \rangle, y)$  这样的数据就是多视图数据.

假设不同视图具有“相容性”(compatibility), 即其所包含的关于输出空间 $\mathcal{Y}$ 的信息是一致的: 令 $\mathcal{Y}^1$ 表示从图像画面信息判别的标记空间,  $\mathcal{Y}^2$ 表示从声音信息判别的标记空间, 则有 $\mathcal{Y} = \mathcal{Y}^1 = \mathcal{Y}^2$ , 例如两者都是{爱情片, 动作片}, 而不能是 $\mathcal{Y}^1 = \{\text{爱情片, 动作片}\}$ 而 $\mathcal{Y}^2 = \{\text{文艺片, 惊悚片}\}$ . 在此假设下, 显式地考虑多视图有很多好处. 仍以电影为例, 某个片段上有两人对视, 仅凭图像画面信息难以分辨其类型, 但此时若从声音信息听到“我爱你”, 则可判断出该片段很可能属于“爱情片”; 另一方面, 若仅凭图像画面信息认为“可能是动作片”, 仅凭声音信息也认为“可能是动作片”, 则当两者一起考虑时就有很大的把握判别为“动作片”. 显然, 在“相容性”基础上, 不同视图信息的“互补性”会给学习器的构建带来很多便利.

协同训练正是很好地利用了多视图的“相容互补性”. 假设数据拥有两个充分(sufficient)且条件独立视图, “充分”是指每个视图都包含足以产生最优学习器的信息, “条件独立”则是指在给定类别标记条件下两个视图独立. 在此情形下, 可用一个简单的办法来利用未标记数据: 首先在每个视图上基于有标记样本分别训练出一个分类器, 然后让每个分类器分别去挑选自己“最有把握的”未标记样本赋予伪标记, 并将伪标记样本提供给另一个分类器作为新增的有标记样本用于训练更新……这个“互相学习、共同进步”的过程不断迭代进行, 直到两个分类器都不再发生变化, 或达到预先设定的迭代轮数为止. 算法描述如图 13.6 所示. 若在每轮学习中都考察分类器在所有未标记样本上的分类置信度, 会有很大的计算开销, 因此在算法中使用了未标记样本缓冲池 [Blum and Mitchell, 1998]. 分类置信度的估计则因基学习算法 $\mathcal{L}$ 而异, 例如若使用朴素贝叶斯分类器, 则可将后验概率转化为分类置信度; 若使用支持向量机, 则可将间隔大小转化为分类置信度.

协同训练过程虽简单, 但令人惊讶的是, 理论证明显示出, 若两个视图充分且条件独立, 则可利用未标记样本通过协同训练将弱分类器的泛化性能提升到任意高[Blum and Mitchell, 1998]. 不过, 视图的条件独立性在现实任务中通常很难满足, 因此性能提升幅度不会那么大, 但研究表明, 即便在更弱的条件下, 协同训练仍可有效地提升弱分类器的性能 [周志华, 2013].

弱分类器参见第 8 章.

例如电影画面与声音显然不会是条件独立的.

单视图数据即仅有一个属性集合的常见数据.

协同训练算法本身是为多视图数据而设计的, 但此后出现了一些能在单视图数据上使用的变体算法, 它们或是使用不同的学习算法 [Goldman and Zhou, 2000], 或使用不同的数据采样 [Zhou and Li, 2005b], 甚至使用不同的参数设置 [Zhou and Li, 2005a] 来产生不同的学习器, 也能有效地利用未标记数据来提升性能. 后续理论研究发现, 此类算法事实上无需数据拥有多视图, 仅需弱学习器

$\mathbf{x}_i$  的上标仅用于指代两个视图, 不表示序关系, 即  $\langle \mathbf{x}_i^1, \mathbf{x}_i^2 \rangle$  与  $\langle \mathbf{x}_i^2, \mathbf{x}_i^1 \rangle$  表示的是同一个样本.

令  $p, n \ll s$ .

初始化每个视图上的有标记训练集.

在视图  $j$  上用有标记样本训练  $h_j$ .

扩充有标记数据集.

---

**输入:** 有标记样本集  $D_l = \{(\langle \mathbf{x}_1^1, \mathbf{x}_1^2 \rangle, y_1), \dots, (\langle \mathbf{x}_l^1, \mathbf{x}_l^2 \rangle, y_l)\}$ ;  
 未标记样本集  $D_u = \{\langle \mathbf{x}_{l+1}^1, \mathbf{x}_{l+1}^2 \rangle, \dots, \langle \mathbf{x}_{l+u}^1, \mathbf{x}_{l+u}^2 \rangle\}$ ;  
 缓冲池大小  $s$ ;  
 每轮挑选的正例数  $p$ ;  
 每轮挑选的反例数  $n$ ;  
 基学习算法  $\mathcal{L}$ ;  
 学习轮数  $T$ .

**过程:**

- 1: 从  $D_u$  中随机抽取  $s$  个样本构成缓冲池  $D_s$ ;
- 2:  $D_u = D_u \setminus D_s$ ;
- 3: **for**  $j = 1, 2$  **do**
- 4:      $D_l^j = \{(\mathbf{x}_i^j, y_i) \mid (\langle \mathbf{x}_i^j, \mathbf{x}_i^{3-j} \rangle, y_i) \in D_l\}$ ;
- 5: **end for**
- 6: **for**  $t = 1, 2, \dots, T$  **do**
- 7:     **for**  $j = 1, 2$  **do**
- 8:          $h_j \leftarrow \mathcal{L}(D_l^j)$ ;
- 9:         考察  $h_j$  在  $D_s^j = \{\mathbf{x}_i^j \mid \langle \mathbf{x}_i^j, \mathbf{x}_i^{3-j} \rangle \in D_s\}$  上的分类置信度, 挑选  $p$  个正例置信度最高的样本  $D_p \subset D_s$ 、 $n$  个反例置信度最高的样本  $D_n \subset D_s$ ;
- 10:       由  $D_p^j$  生成伪标记正例  $\tilde{D}_p^{3-j} = \{(\mathbf{x}_i^{3-j}, +1) \mid \mathbf{x}_i^j \in D_p^j\}$ ;
- 11:       由  $D_n^j$  生成伪标记反例  $\tilde{D}_n^{3-j} = \{(\mathbf{x}_i^{3-j}, -1) \mid \mathbf{x}_i^j \in D_n^j\}$ ;
- 12:        $D_s = D_s \setminus (D_p \cup D_n)$ ;
- 13:     **end for**
- 14:     **if**  $h_1, h_2$  均未发生改变 **then**
- 15:         **break**
- 16:     **else**
- 17:         **for**  $j = 1, 2$  **do**
- 18:              $D_l^j = D_l^j \cup (\tilde{D}_p^j \cup \tilde{D}_n^j)$ ;
- 19:         **end for**
- 20:         从  $D_u$  中随机抽取  $2p + 2n$  个样本加入  $D_s$
- 21:     **end if**
- 22: **end for**

**输出:** 分类器  $h_1, h_2$

---

图 13.6 协同训练算法

因此, 该类方法被称为“基于分歧的方法”.

之间具有显著的分歧(或差异), 即可通过相互提供伪标记样本的方式来提升泛化性能 [周志华, 2013]; 不同视图、不同算法、不同数据采样、不同参数设置等, 都仅是产生差异的渠道, 而非必备条件.

基于分歧的方法只需采用合适的基学习器, 就能较少受到模型假设、损失函数非凸性和数据规模问题的影响, 学习方法简单有效、理论基础相对坚实、适用范围较为广泛. 为了使用此类方法, 需能生成具有显著分歧、性能尚可的多个学习器, 但当有标记样本很少, 尤其是数据不具有多视图时, 要做到这一点并不容易, 需有巧妙的设计.

### 13.6 半监督聚类

聚类是一种典型的无监督学习任务，然而在现实聚类任务中我们往往能获得一些额外的监督信息，于是可通过半监督聚类(semi-supervised clustering)来利用监督信息以获得更好的聚类效果。

参见 10.6 节。

聚类任务中获得的监督信息大致有两种类型。第一种类型是“必连”(must-link)与“勿连”(cannot-link)约束，前者是指样本必属于同一个簇，后者是指样本必不属于同一个簇；第二种类型的监督信息则是少量的有标记样本。

约束  $k$  均值(Constrained  $k$ -means)算法 [Wagstaff et al., 2001] 是利用第一类监督信息的代表。给定样本集  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$  以及“必连”关系

初始化  $k$  个空簇。

更新均值向量。

---

**输入:** 样本集  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ;  
 必连约束集合  $\mathcal{M}$ ;  
 勿连约束集合  $\mathcal{C}$ ;  
 聚类簇数  $k$ .

**过程:**

- 1: 从  $D$  中随机选取  $k$  个样本作为初始均值向量  $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_k\}$ ;
- 2: **repeat**
- 3:      $C_j = \emptyset (1 \leq j \leq k)$ ;
- 4:     **for**  $i = 1, 2, \dots, m$  **do**
- 5:         计算样本  $\mathbf{x}_i$  与各均值向量  $\boldsymbol{\mu}_j (1 \leq j \leq k)$  的距离:  $d_{ij} = \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_2$ ;
- 6:          $\mathcal{K} = \{1, 2, \dots, k\}$ ;
- 7:         is\_merged=false;
- 8:         **while**  $\neg$  is\_merged **do**
- 9:             基于  $\mathcal{K}$  找出与样本  $\mathbf{x}_i$  距离最近的簇:  $r = \arg \min_{j \in \mathcal{K}} d_{ij}$ ;
- 10:             检测将  $\mathbf{x}_i$  划入聚类簇  $C_r$  是否会违背  $\mathcal{M}$  与  $\mathcal{C}$  中的约束;
- 11:             **if**  $\neg$  is\_violated **then**
- 12:                  $C_r = C_r \cup \{\mathbf{x}_i\}$ ;
- 13:                 is\_merged=true
- 14:             **else**
- 15:                  $\mathcal{K} = \mathcal{K} \setminus \{r\}$ ;
- 16:                 **if**  $\mathcal{K} = \emptyset$  **then**
- 17:                     **break** 并返回错误提示
- 18:                 **end if**
- 19:             **end if**
- 20:         **end while**
- 21:     **end for**
- 22:     **for**  $j = 1, 2, \dots, k$  **do**
- 23:          $\boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$ ;
- 24:     **end for**
- 25: **until** 均值向量均未更新

**输出:** 簇划分  $\{C_1, C_2, \dots, C_k\}$

---

图 13.7 约束  $k$  均值算法

集合  $\mathcal{M}$  和“勿连”关系集合  $\mathcal{C}$ ,  $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}$  表示  $\mathbf{x}_i$  与  $\mathbf{x}_j$  必属于同簇,  $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}$  表示  $\mathbf{x}_i$  与  $\mathbf{x}_j$  必不属于同簇。该算法是  $k$  均值算法的扩展, 它在聚类过程中要确保  $\mathcal{M}$  与  $\mathcal{C}$  中的约束得以满足, 否则将返回错误提示, 算法如图 13.7 所示。

见 p.202 表 9.1.

以西瓜数据集 4.0 为例, 令样本  $\mathbf{x}_4$  与  $\mathbf{x}_{25}$ ,  $\mathbf{x}_{12}$  与  $\mathbf{x}_{20}$ ,  $\mathbf{x}_{14}$  与  $\mathbf{x}_{17}$  之间存在必连约束,  $\mathbf{x}_2$  与  $\mathbf{x}_{21}$ ,  $\mathbf{x}_{13}$  与  $\mathbf{x}_{23}$ ,  $\mathbf{x}_{19}$  与  $\mathbf{x}_{23}$  之间存在勿连约束, 即

$$\mathcal{M} = \{(\mathbf{x}_4, \mathbf{x}_{25}), (\mathbf{x}_{25}, \mathbf{x}_4), (\mathbf{x}_{12}, \mathbf{x}_{20}), (\mathbf{x}_{20}, \mathbf{x}_{12}), (\mathbf{x}_{14}, \mathbf{x}_{17}), (\mathbf{x}_{17}, \mathbf{x}_{14})\},$$

$$\mathcal{C} = \{(\mathbf{x}_2, \mathbf{x}_{21}), (\mathbf{x}_{21}, \mathbf{x}_2), (\mathbf{x}_{13}, \mathbf{x}_{23}), (\mathbf{x}_{23}, \mathbf{x}_{13}), (\mathbf{x}_{19}, \mathbf{x}_{23}), (\mathbf{x}_{23}, \mathbf{x}_{19})\}.$$

设聚类簇数  $k = 3$ , 随机选取样本  $\mathbf{x}_6$ ,  $\mathbf{x}_{12}$ ,  $\mathbf{x}_{27}$  作为初始均值向量, 图 13.8

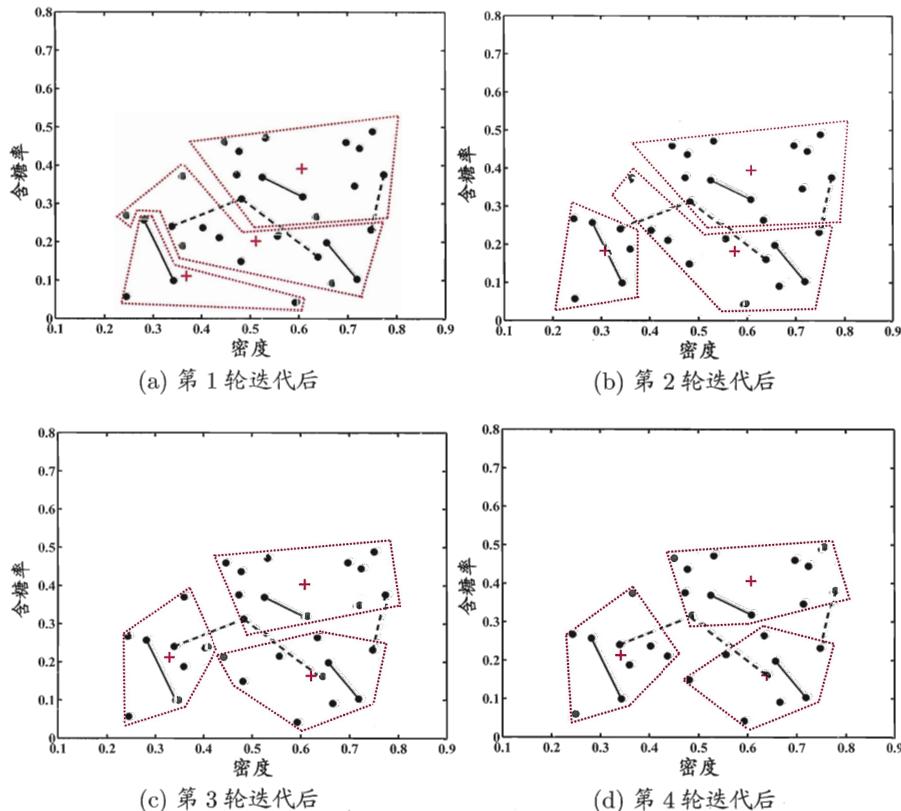


图 13.8 西瓜数据集 4.0 上约束  $k$  均值算法 ( $k = 3$ ) 在各轮迭代后的结果。样本点与均值向量分别用“●”与“+”表示, 必连约束和勿连约束分别用实线段与虚线段表示, 红色虚线显示出簇划分。

显示出约束  $k$  均值算法在不同迭代轮数后的聚类结果. 经 5 轮迭代后均值向量不再发生变化(与第 4 轮迭代相同), 于是得到最终聚类结果

$$C_1 = \{\mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_7, \mathbf{x}_9, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{16}, \mathbf{x}_{17}, \mathbf{x}_{21}\};$$

$$C_2 = \{\mathbf{x}_6, \mathbf{x}_8, \mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{15}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{20}\};$$

$$C_3 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_{22}, \mathbf{x}_{23}, \mathbf{x}_{24}, \mathbf{x}_{25}, \mathbf{x}_{26}, \mathbf{x}_{27}, \mathbf{x}_{28}, \mathbf{x}_{29}, \mathbf{x}_{30}\}.$$

此处样本标记指簇标记(cluster label), 不是类别标记(class label).

第二种监督信息是少量有标记样本. 给定样本集  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ , 假定少量的有标记样本为  $S = \bigcup_{j=1}^k S_j \subset D$ , 其中  $S_j \neq \emptyset$  为隶属于第  $j$  个聚类簇的样本. 这样的监督信息利用起来很容易: 直接将它们作为“种子”, 用它们初始化  $k$  均值算法的  $k$  个聚类中心, 并且在聚类簇迭代更新过程中不改变种子样本的簇隶属关系. 这样就得到了约束种子  $k$  均值(Constrained Seed  $k$ -means)算法 [Basu et al., 2002], 其算法描述如图 13.9 所示.

$S \subset D$ ,  $|S| \ll |D|$ .

用有标记样本初始化簇中心.

用有标记样本初始化  $k$  个簇.

更新均值向量.

---

输入: 样本集  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ;  
 少量有标记样本  $S = \bigcup_{j=1}^k S_j$  ;  
 聚类簇数  $k$ .  
 过程:  
 1: **for**  $j = 1, 2, \dots, k$  **do**  
 2:    $\mu_j = \frac{1}{|S_j|} \sum_{x \in S_j} x$   
 3:   **end for**  
 4: **repeat**  
 5:    $C_j = \emptyset$  ( $1 \leq j \leq k$ );  
 6:   **for**  $j = 1, 2, \dots, k$  **do**  
 7:     **for all**  $x \in S_j$  **do**  
 8:        $C_j = C_j \cup \{x\}$   
 9:     **end for**  
 10:   **end for**  
 11:   **for all**  $x_i \in D \setminus S$  **do**  
 12:     计算样本  $x_i$  与各均值向量  $\mu_j$  ( $1 \leq j \leq k$ ) 的距离:  $d_{ij} = \|x_i - \mu_j\|_2$  ;  
 13:     找出与样本  $x_i$  距离最近的簇:  $r = \arg \min_{j \in \{1, 2, \dots, k\}} d_{ij}$  ;  
 14:     将样本  $x_i$  划入相应的簇:  $C_r = C_r \cup \{x_i\}$   
 15:   **end for**  
 16:   **for**  $j = 1, 2, \dots, k$  **do**  
 17:      $\mu_j = \frac{1}{|C_j|} \sum_{x \in C_j} x$ ;  
 18:   **end for**  
 19: **until** 均值向量均未更新  
 输出: 簇划分  $\{C_1, C_2, \dots, C_k\}$

---

图 13.9 约束种子  $k$  均值算法

仍以西瓜数据集 4.0 为例, 假定作为种子的有标记样本为

$$S_1 = \{\mathbf{x}_4, \mathbf{x}_{25}\}, S_2 = \{\mathbf{x}_{12}, \mathbf{x}_{20}\}, S_3 = \{\mathbf{x}_{14}, \mathbf{x}_{17}\}.$$

以这三组种子样本的平均向量作为初始均值向量, 图 13.10 显示出约束种子  $k$  均值算法在不同迭代轮数后的聚类结果。经 4 轮迭代后均值向量不再发生变化(与第 3 轮迭代相同), 于是得到最终聚类结果

$$C_1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_{22}, \mathbf{x}_{23}, \mathbf{x}_{24}, \mathbf{x}_{25}, \mathbf{x}_{26}, \mathbf{x}_{27}, \mathbf{x}_{28}, \mathbf{x}_{29}, \mathbf{x}_{30}\};$$

$$C_2 = \{\mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{15}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{20}\};$$

$$C_3 = \{\mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_9, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{16}, \mathbf{x}_{17}, \mathbf{x}_{21}\}.$$

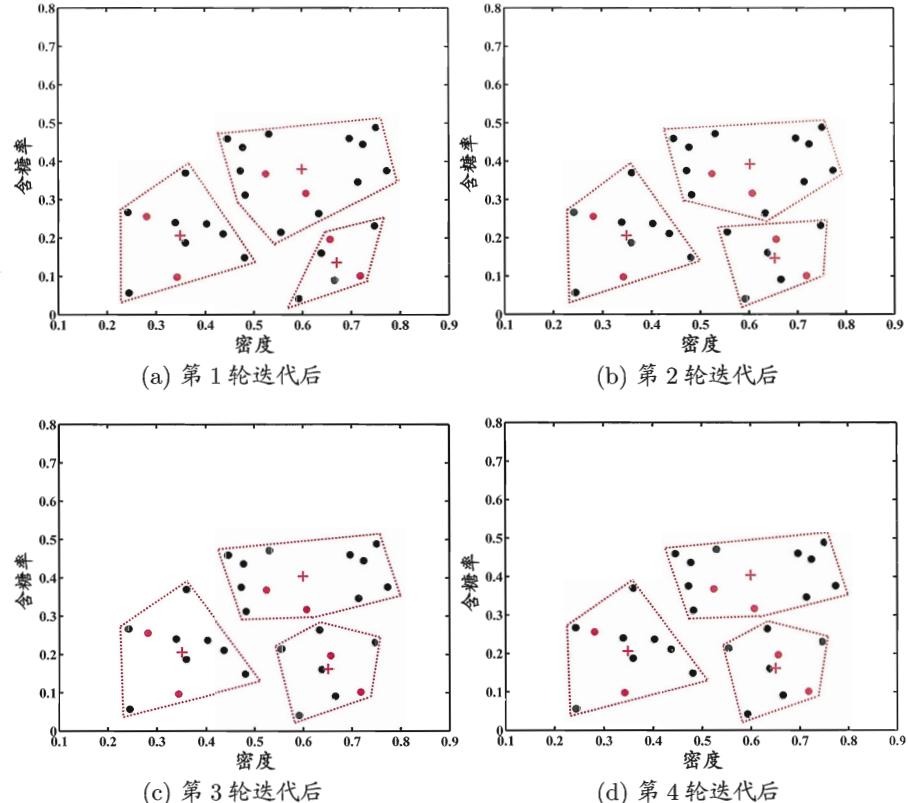


图 13.10 西瓜数据集 4.0 上约束种子  $k$  均值算法 ( $k=3$ ) 在各轮迭代后的结果。样本点与均值向量分别用“•”与“+”表示, 种子样本点为红色, 红色虚线显示出簇划分。

## 13.7 阅读材料

半监督学习的研究一般认为始于 [Shahshahani and Landgrebe, 1994], 该领域在二十世纪末、二十一世纪初随着现实应用中利用未标记数据的巨大需求涌现而蓬勃发展。国际机器学习大会(ICML)从 2008 年开始评选“十年最佳论文”, 在短短 6 年中, 半监督学习四大范型(paradigm)中基于分歧的方法、半监督 SVM、图半监督学习的代表性工作先后于 2008 年 [Blum and Mitchell, 1998]、2009 年 [Joachims, 1999]、2013 年 [Zhu et al., 2003] 获奖。

生成式半监督学习方法出现最早 [Shahshahani and Landgrebe, 1994]。由于需有充分可靠的领域知识才能确保模型假设不至于太坏, 因此该范型后来主要是在具体的应用领域加以研究。

半监督 SVM 的目标函数非凸, 有不少工作致力于减轻非凸性造成的不利影响, 例如使用连续统(continuation)方法, 从优化一个简单的凸目标函数开始, 逐步变形为非凸的 S3VM 目标函数 [Chapelle et al., 2006a]; 使用确定性退火(deterministic annealing)过程, 将非凸问题转化为一系列凸优化问题, 然后由易到难地顺序求解 [Sindhwani et al., 2006]; 利用 CCCP 方法优化非凸函数 [Collobert et al., 2006] 等。

最早的图半监督学习方法 [Blum and Chawla, 2001] 直接基于聚类假设, 将学习目标看作找出图的最小割(mincut)。对此类方法来说, 图的质量极为重要, 13.4 节的高斯距离图以及  $k$  近邻图、 $\epsilon$  近邻图都较为常用, 此外已有一些关于构图的研究 [Wang and Zhang, 2006; Jebara et al., 2009], 基于图核(graph kernel)的方法也与此有密切联系 [Chapelle et al., 2003]。

基于分歧的方法起源于协同训练, 最初设计是仅选取一个学习器用于预测 [Blum and Mitchell, 1998]。三体训练(tri-training)使用三个学习器, 通过“少数服从多数”来产生伪标记样本, 并将学习器进行集成 [Zhou and Li, 2005b]。后续研究进一步显示出将学习器集成起来更有助于性能提升, 并出现了使用更多学习器的方法。更为重要的是, 这将集成学习与半监督学习这两个长期独立发展的领域联系起来 [Zhou, 2009]。此外, 这些方法能容易地用于多视图数据, 并可自然地与主动学习进行结合 [周志华, 2013]。

[Belkin et al., 2006] 在半监督学习中提出了流形正则化(manifold regularization)框架, 直接基于局部光滑性假设对定义在有标记样本上的损失函数进行正则化, 使学得的预测函数具有局部光滑性。

半监督学习在利用未标记样本后并非必然提升泛化性能, 在有些情形下甚

$k$  近邻图和  $\epsilon$  近邻图参见 10.5.1 节。

许多集成学习研究者认为: 只要能使用多个学习器即可将弱学习器性能提升到极高, 无须使用未标记样本; 许多半监督学习研究者认为: 只要能使用未标记样本即可将弱学习器性能提升到极高, 无须使用多学习器。但这两种看法都有其局限。

至会导致性能下降。对生成式方法，其成因被认为是模型假设不准确 [Cozman and Cohen, 2002]，因此需依赖充分可靠的领域知识来设计模型。对半监督 SVM，其成因被认为是训练数据中存在多个“低密度划分”，而学习算法有可能做出不利的选择；S4VM [Li and Zhou, 2015] 通过优化最坏情形性能来综合利用多个低密度划分，提升了此类技术的安全性。更一般的“安全”(safe)半监督学习仍是一个未决问题。

这里的“安全”是指利用未标记样本后，能确保泛化性能至少不差于仅利用有标记样本。

本章主要介绍了半监督分类和聚类，但半监督学习已普遍用于各类机器学习任务，例如在半监督回归 [Zhou and Li, 2005a]、降维 [Zhang et al., 2007] 等方面都有相关研究。更多关于半监督学习的内容可参见 [Chapelle et al., 2006b; Zhu, 2006], [Zhou and Li, 2010; 周志华, 2013] 专门介绍了基于分歧的方法。[Settles, 2009] 是一个关于主动学习的介绍。

## 习题

- 13.1** 试推导出式(13.5)~(13.8).
- 13.2** 试基于朴素贝叶斯模型推导出生成式半监督学习算法.
- 13.3** 假设数据由混合专家(mixture of experts)模型生成, 即数据是基于  $k$  个成分混合而得的概率密度生成:

$$p(\mathbf{x} | \boldsymbol{\theta}) = \sum_{i=1}^k \alpha_i \cdot p(\mathbf{x} | \boldsymbol{\theta}_i), \quad (13.22)$$

其中  $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_k\}$  是模型参数,  $p(\mathbf{x} | \boldsymbol{\theta}_i)$  是第  $i$  个混合成分的概率密度, 混合系数  $\alpha_i \geq 0$ ,  $\sum_{i=1}^k \alpha_i = 1$ . 假设每个混合成分对应一个类别, 但每个类别可包含多个混合成分. 试推导相应的生成式半监督学习算法.

UCI 数据集见  
<http://archive.ics.uci.edu/ml/>.

- 13.4** 从网上下载或自己编程实现 TSVM 算法, 选择两个 UCI 数据集, 将其中 30% 的样例用作测试样本, 10% 的样例用作有标记样本, 60% 的样例用作无标记样本, 分别训练出利用无标记样本的 TSVM 以及仅利用有标记样本的 SVM, 并比较其性能.
- 13.5** 对未标记样本进行标记指派与调整的过程中有可能出现类别不平衡问题, 试给出考虑该问题后的改进 TSVM 算法.
- 13.6\*** TSVM 对未标记样本进行标记指派与调整的过程涉及很大的计算开销, 试设计一个高效的改进算法.
- 13.7\*** 试设计一个能对新样本进行分类的图半监督学习方法.
- 13.8** 自训练(self-training)是一种比较原始的半监督学习方法: 它先在有标记样本上学习, 然后用学得分类器对未标记样本进行判别以获得其伪标记, 再在有标记与伪标记样本的合集上重新训练, 如此反复. 试析该方法有何缺陷.
- 13.9\*** 给定一个数据集, 假设其属性集包含两个视图, 但事先并不知道哪些属性属于哪个视图, 试设计一个算法将这两个视图分离出来.
- 13.10** 试为图 13.7 算法的第 10 行写出违约检测算法(用于检测是否有约束未被满足).

## 参考文献

- 周志华. (2013). “基于分歧的半监督学习.” *自动化学报*, 39(11):1871–1878.
- Basu, S., A. Banerjee, and R. J. Mooney. (2002). “Semi-supervised clustering by seeding.” In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, 19–26, Sydney, Australia.
- Belkin, M., P. Niyogi, and V. Sindhwani. (2006). “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples.” *Journal of Machine Learning Research*, 7:2399–2434.
- Blum, A. and S. Chawla. (2001). “Learning from labeled and unlabeled data using graph mincuts.” In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, 19–26, Williamston, MA.
- Blum, A. and T. Mitchell. (1998). “Combining labeled and unlabeled data with co-training.” In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT)*, 92–100, Madison, WI.
- Chapelle, O., M. Chi, and A. Zien. (2006a). “A continuation method for semi-supervised SVMs.” In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 185–192, Pittsburgh, PA.
- Chapelle, O., B. Schölkopf, and A. Zien, eds. (2006b). *Semi-Supervised Learning*. MIT Press, Cambridge, MA.
- Chapelle, O., J. Weston, and B. Schölkopf. (2003). “Cluster kernels for semi-supervised learning.” In *Advances in Neural Information Processing Systems 15 (NIPS)* (S. Becker, S. Thrun, and K. Obermayer, eds.), 585–592, MIT Press, Cambridge, MA.
- Chapelle, O. and A. Zien. (2005). “Semi-supervised learning by low density separation.” In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 57–64, Savannah Hotel, Barbados.
- Collobert, R., F. Sinz, J. Weston, and L. Bottou. (2006). “Trading convexity for scalability.” In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 201–208, Pittsburgh, PA.
- Cozman, F. G. and I. Cohen. (2002). “Unlabeled data can degrade classification performance of generative classifiers.” In *Proceedings of the 15th International Conference of the Florida Artificial Intelligence Research Society*

- (*FLAIRS*), 327–331, Pensacola, FL.
- Goldman, S. and Y. Zhou. (2000). “Enhancing supervised learning with unlabeled data.” In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, 327–334, San Francisco, CA.
- Jebara, T., J. Wang, and S. F. Chang. (2009). “Graph construction and b-matching for semi-supervised learning.” In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 441–448, Montreal, Canada.
- Joachims, T. (1999). “Transductive inference for text classification using support vector machines.” In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, 200–209, Bled, Slovenia.
- Li, Y.-F., J. T. Kwok, and Z.-H. Zhou. (2009). “Semi-supervised learning using label mean.” In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 633–640, Montreal, Canada.
- Li, Y.-F. and Z.-H. Zhou. (2015). “Towards making unlabeled data never hurt.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1): 175–188.
- Miller, D. J. and H. S. Uyar. (1997). “A mixture of experts classifier with learning based on both labelled and unlabelled data.” In *Advances in Neural Information Processing Systems 9 (NIPS)* (M. Mozer, M. I. Jordan, and T. Petsche, eds.), 571–577, MIT Press, Cambridge, MA.
- Nigam, K., A. McCallum, S. Thrun, and T. Mitchell. (2000). “Text classification from labeled and unlabeled documents using EM.” *Machine Learning*, 39(2-3):103–134.
- Settles, B. (2009). “Active learning literature survey.” Technical Report 1648, Department of Computer Sciences, University of Wisconsin at Madison, Wisconsin, WI. <http://pages.cs.wisc.edu/~bsettles/pub/settles.activelearning.pdf>.
- Shahshahani, B. and D. Landgrebe. (1994). “The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon.” *IEEE Transactions on Geoscience and Remote Sensing*, 32(5): 1087–1095.

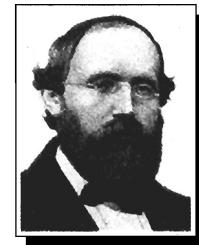
- Sindhwani, V., S. S. Keerthi, and O. Chapelle. (2006). "Deterministic annealing for semi-supervised kernel machines." In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 123–130, Pittsburgh, PA.
- Wagstaff, K., C. Cardie, S. Rogers, and S. Schrödl. (2001). "Constrained k-means clustering with background knowledge." In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, 577–584, Williamstown, MA.
- Wang, F. and C. Zhang. (2006). "Label propagation through linear neighborhoods." In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 985–992, Pittsburgh, PA.
- Zhang, D., Z.-H. Zhou, and S. Chen. (2007). "Semi-supervised dimensionality reduction." In *Proceedings of the 7th SIAM International Conference on Data Mining (SDM)*, 629–634, Minneapolis, MN.
- Zhou, D., O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. (2004). "Learning with local and global consistency." In *Advances in Neural Information Processing Systems 16 (NIPS)* (S. Thrun, L. Saul, and B. Schölkopf, eds.), 284–291, MIT Press, Cambridge, MA.
- Zhou, Z.-H. (2009). "When semi-supervised learning meets ensemble learning." In *Proceedings of the 8th International Workshop on Multiple Classifier Systems*, 529–538, Reykjavik, Iceland.
- Zhou, Z.-H. and M. Li. (2005a). "Semi-supervised regression with co-training." In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 908–913, Edinburgh, Scotland.
- Zhou, Z.-H. and M. Li. (2005b). "Tri-training: Exploiting unlabeled data using three classifiers." *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.
- Zhou, Z.-H. and M. Li. (2010). "Semi-supervised learning by disagreement." *Knowledge and Information Systems*, 24(3):415–439.
- Zhu, X. (2006). "Semi-supervised learning literature survey." Technical Report 1530, Department of Computer Sciences, University of Wisconsin at Madison, Madison, WI. [http://www.cs.wisc.edu/~jerryzhu/pub/ssl\\_survey.pdf](http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf).
- Zhu, X., Z. Ghahramani, and J. Lafferty. (2003). "Semi-supervised learning

using Gaussian fields and harmonic functions.” In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 912–919, Washington, DC.

## 休息一会儿

### 小故事：流形与伯恩哈德·黎曼

“流形”(manifold)这个名字源于德语 *Mannigfaltigkeit*, 是伟大的德国数学家伯恩哈德·黎曼 (Bernhard Riemann, 1826—1866) 提出的, 其译名则是我国拓扑学奠基人江泽涵先生借鉴文天祥《正气歌》“天地有正气, 杂然赋流形”而来, 可能是由于光滑流形恰与“气”相似, 整体上看可流动、变形。



黎曼出生于德国汉诺威的布列斯伦茨(Breselenz), 幼年时就展现出惊人的数学天赋。1846年父亲送他到哥廷根大学攻读神学, 在旁听了高斯关于最小二乘法的讲座后, 他决定转攻数学, 并在高斯指导下于1851年获博士学位。期间有两年他在柏林大学学习, 受到了雅可比、狄利克雷等大数学家的影响。1853年, 高斯让黎曼在几何学基础方面准备一个报告, 以便取得哥廷根大学的教职; 1854年, 黎曼做了“论作为几何基础的假设”的著名演讲, 这个报告开创了黎曼几何, 提出了黎曼积分, 并首次使用了 *Mannigfaltigkeit* 这个词。此后黎曼一直在哥廷根大学任教, 并在1859年接替去世的狄利克雷担任数学教授。

传统的德国大学中一个系只有一位“教授”, 相当于系主任。高斯长期担任哥廷根大学数学教授, 1855年他去世后由狄利克雷接任。

7个千禧年数学难题中, 已被证明的“庞加莱猜想”直接与流形有关: 任何一个单连通、闭的三维流形一定同胚于一个三维球面。

黎曼是黎曼几何的创立者、复变函数论的奠基人, 并对微积分、解析数论、组合拓扑、代数几何、数学物理方法均做出了开创性贡献, 他的工作直接影响了近百年数学的发展, 许多杰出的数学家前赴后继地努力论证黎曼断言过的定理。1900年希尔伯特列出的23个世纪数学问题与2000年美国克雷数学研究所列出的7个千禧年数学难题中, 有一个问题是相同的, 这就是黎曼1859年因当选院士而提交给柏林科学院的文章中提出的“黎曼猜想”。这是关于黎曼 $\zeta$ 函数非平凡零点的猜想。目前已有不同数学分支的千余个数学命题以黎曼猜想为前提, 若黎曼猜想正确, 它们将全部升格为定理。一个猜想联系了如此多不同数学分支、如此多命题, 在数学史上是极为罕见的, 因此它被公认为当前最重要的数学难题。



# 第 14 章 概率图模型

## 14.1 隐马尔可夫模型

机器学习最重要的任务, 是根据一些已观察到的证据(例如训练样本)来对感兴趣的未知变量(例如类别标记)进行估计和推测。概率模型(probabilistic model)提供了一种描述框架, 将学习任务归结于计算变量的概率分布。在概率模型中, 利用已知变量推测未知变量的分布称为“推断”(inference), 其核心是如何基于可观测变量推测出未知变量的条件分布。具体来说, 假定所关心的变量集合为  $Y$ , 可观测变量集合为  $O$ , 其他变量的集合为  $R$ , “生成式”(generative)模型考虑联合分布  $P(Y, R, O)$ , “判别式”(discriminative)模型考虑条件分布  $P(Y, R | O)$ 。给定一组观测变量值, 推断就是要由  $P(Y, R, O)$  或  $P(Y, R | O)$  得到条件概率分布  $P(Y | O)$ 。

直接利用概率求和规则消去变量  $R$  显然不可行, 因为即便每个变量仅有两种取值的简单问题, 其复杂度已至少是  $O(2^{|Y|+|R|})$ 。另一方面, 属性变量之间往往存在复杂的联系, 因此概率模型的学习, 即基于训练样本来估计变量分布的参数往往相当困难。为了便于研究高效的推断和学习算法, 需有一套能简洁紧凑地表达变量间关系的工具。

概率图模型(probabilistic graphical model)是一类用图来表达变量相关关系的概率模型。它以图为表示工具, 最常见的是用一个结点表示一个或一组随机变量, 结点之间的边表示变量间的概率相关关系, 即“变量关系图”。根据边的性质不同, 概率图模型可大致分为两类: 第一类是使用有向无环图表示变量间的依赖关系, 称为有向图模型或贝叶斯网(Bayesian network); 第二类是使用无向图表示变量间的相关关系, 称为无向图模型或马尔可夫网(Markov network)。

隐马尔可夫模型(Hidden Markov Model, 简称 HMM)是结构最简单的动态贝叶斯网(dynamic Bayesian network), 这是一种著名的有向图模型, 主要用于时序数据建模, 在语音识别、自然语言处理等领域有广泛应用。

如图 14.1 所示, 隐马尔可夫模型中的变量可分为两组。第一组是状态变量  $\{y_1, y_2, \dots, y_n\}$ , 其中  $y_i \in \mathcal{Y}$  表示第  $i$  时刻的系统状态。通常假定状态变量是隐藏的、不可被观测的, 因此状态变量亦称隐变量(hidden variable)。第二组是观

若变量间存在显式的因果关系, 则常使用贝叶斯网; 若变量间存在相关性, 但难以获得显式的因果关系, 则常使用马尔可夫网。

静态贝叶斯网参见 7.5 节。

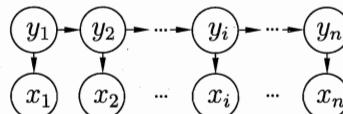


图 14.1 隐马尔可夫模型的图结构

测变量  $\{x_1, x_2, \dots, x_n\}$ , 其中  $x_i \in \mathcal{X}$  表示第  $i$  时刻的观测值. 在隐马尔可夫模型中, 系统通常在多个状态  $\{s_1, s_2, \dots, s_N\}$  之间转换, 因此状态变量  $y_t$  的取值范围  $\mathcal{Y}$  (称为状态空间) 通常是有  $N$  个可能取值的离散空间. 观测变量  $x_i$  可以是离散型也可以是连续型, 为便于讨论, 我们仅考虑离散型观测变量, 并假定其取值范围  $\mathcal{X}$  为  $\{o_1, o_2, \dots, o_M\}$ .

图 14.1 中的箭头表示了变量间的依赖关系. 在任一时刻, 观测变量的取值仅依赖于状态变量, 即  $x_t$  由  $y_t$  确定, 与其他状态变量及观测变量的取值无关. 同时,  $t$  时刻的状态  $y_t$  仅依赖于  $t-1$  时刻的状态  $y_{t-1}$ , 与其余  $n-2$  个状态无关. 这就是所谓的“马尔可夫链”(Markov chain), 即: 系统下一时刻的状态仅由当前状态决定, 不依赖于以往的任何状态. 基于这种依赖关系, 所有变量的联合概率分布为  
所谓“现在决定未来”.

$$P(x_1, y_1, \dots, x_n, y_n) = P(y_1)P(x_1 | y_1) \prod_{i=2}^n P(y_i | y_{i-1})P(x_i | y_i). \quad (14.1)$$

除了结构信息, 欲确定一个隐马尔可夫模型还需以下三组参数:

- 状态转移概率: 模型在各个状态间转换的概率, 通常记为矩阵  $\mathbf{A} = [a_{ij}]_{N \times N}$ , 其中

$$a_{ij} = P(y_{t+1} = s_j | y_t = s_i), \quad 1 \leq i, j \leq N,$$

表示在任意时刻  $t$ , 若状态为  $s_i$ , 则在下一时刻状态为  $s_j$  的概率.

- 输出观测概率: 模型根据当前状态获得各个观测值的概率, 通常记为矩阵  $\mathbf{B} = [b_{ij}]_{N \times M}$ , 其中

$$b_{ij} = P(x_t = o_j | y_t = s_i), \quad 1 \leq i \leq N, 1 \leq j \leq M$$

表示在任意时刻  $t$ , 若状态为  $s_i$ , 则观测值  $o_j$  被获取的概率.

- 初始状态概率: 模型在初始时刻各状态出现的概率, 通常记为  $\pi =$

$(\pi_1, \pi_2, \dots, \pi_N)$ , 其中

$$\pi_i = P(y_1 = s_i), \quad 1 \leq i \leq N$$

表示模型的初始状态为  $s_i$  的概率.

通过指定状态空间  $\mathcal{Y}$ 、观测空间  $\mathcal{X}$  和上述三组参数, 就能确定一个隐马尔可夫模型, 通常用其参数  $\lambda = [\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}]$  来指代. 给定隐马尔可夫模型  $\lambda$ , 它按如下过程产生观测序列  $\{x_1, x_2, \dots, x_n\}$ :

- (1) 设置  $t = 1$ , 并根据初始状态概率  $\boldsymbol{\pi}$  选择初始状态  $y_1$ ;
- (2) 根据状态  $y_t$  和输出观测概率  $\mathbf{B}$  选择观测变量取值  $x_t$ ;
- (3) 根据状态  $y_t$  和状态转移矩阵  $\mathbf{A}$  转移模型状态, 即确定  $y_{t+1}$ ;
- (4) 若  $t < n$ , 设置  $t = t + 1$ , 并转到第 (2) 步, 否则停止.

其中  $y_t \in \{s_1, s_2, \dots, s_N\}$  和  $x_t \in \{o_1, o_2, \dots, o_M\}$  分别为第  $t$  时刻的状态和观测值.

在实际应用中, 人们常关注隐马尔可夫模型的三个基本问题:

- 给定模型  $\lambda = [\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}]$ , 如何有效计算其产生观测序列  $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$  的概率  $P(\mathbf{x} | \lambda)$ ? 换言之, 如何评估模型与观测序列之间的匹配程度?
- 给定模型  $\lambda = [\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}]$  和观测序列  $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ , 如何找到与此观测序列最匹配的状态序列  $\mathbf{y} = \{y_1, y_2, \dots, y_n\}$ ? 换言之, 如何根据观测序列推断出隐藏的模型状态?
- 给定观测序列  $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ , 如何调整模型参数  $\lambda = [\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}]$  使得该序列出现的概率  $P(\mathbf{x} | \lambda)$  最大? 换言之, 如何训练模型使其能最好地描述观测数据?

上述问题在现实应用中非常重要. 例如许多任务需根据以往的观测序列  $\{x_1, x_2, \dots, x_{n-1}\}$  来推测当前时刻最有可能的观测值  $x_n$ , 这显然可转化为求取概率  $P(\mathbf{x} | \lambda)$ , 即上述第一个问题; 在语音识别等任务中, 观测值为语音信号, 隐藏状态为文字, 目标就是根据观测信号来推断最有可能的状态序列(即对应的文字), 即上述第二个问题; 在大多数现实应用中, 人工指定模型参数已变得

越来越不可行, 如何根据训练样本学得最优的模型参数, 恰是上述第三个问题。值得庆幸的是, 基于式(14.1)的条件独立性, 隐马尔可夫模型的这三个问题均能被高效求解。

## 14.2 马尔可夫随机场

马尔可夫随机场(Markov Random Field, 简称 MRF)是典型的马尔可夫网, 这是一种著名的无向图模型。图中每个结点表示一个或一组变量, 结点之间的边表示两个变量之间的依赖关系。马尔可夫随机场有一组势函数(potential functions), 亦称“因子”(factor), 这是定义在变量子集上的非负实函数, 主要用于定义概率分布函数。

图 14.2 显示出一个简单的马尔可夫随机场。对于图中结点的一个子集, 若其中任意两结点间都有边连接, 则称该结点子集为一个“团”(clique)。若在一个团中加入另外任何一个结点都不再形成团, 则称该团为“极大团”(maximal clique); 换言之, 极大团就是不能被其他团所包含的团。例如, 在图 14.2 中,  $\{x_1, x_2\}$ ,  $\{x_1, x_3\}$ ,  $\{x_2, x_4\}$ ,  $\{x_2, x_5\}$ ,  $\{x_2, x_6\}$ ,  $\{x_3, x_5\}$ ,  $\{x_5, x_6\}$  和  $\{x_2, x_5, x_6\}$  都是团, 并且除了  $\{x_2, x_5\}$ ,  $\{x_2, x_6\}$  和  $\{x_5, x_6\}$  之外都是极大团; 但是, 因为  $x_2$  和  $x_3$  之间缺乏连接,  $\{x_1, x_2, x_3\}$  并不构成团。显然, 每个结点至少出现在一个极大团中。

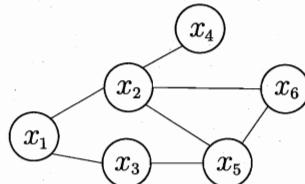


图 14.2 一个简单的马尔可夫随机场

在马尔可夫随机场中, 多个变量之间的联合概率分布能基于团分解为多个因子的乘积, 每个因子仅与一个团相关。具体来说, 对于  $n$  个变量  $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ , 所有团构成的集合为  $\mathcal{C}$ , 与团  $Q \in \mathcal{C}$  对应的变量集合记为  $\mathbf{x}_Q$ , 则联合概率  $P(\mathbf{x})$  定义为

$$P(\mathbf{x}) = \frac{1}{Z} \prod_{Q \in \mathcal{C}} \psi_Q(\mathbf{x}_Q) , \quad (14.2)$$

其中  $\psi_Q$  为与团  $Q$  对应的势函数, 用于对团  $Q$  中的变量关系进行建模,  $Z =$

$\sum_{\mathbf{x}} \prod_{Q \in \mathcal{C}} \psi_Q(\mathbf{x}_Q)$  为规范化因子, 以确保  $P(\mathbf{x})$  是被正确定义的概率. 在实际应用中, 精确计算  $Z$  通常很困难, 但许多任务往往并不需获得  $Z$  的精确值.

显然, 若变量个数较多, 则团的数目将会很多(例如, 所有相互连接的两个变量都会构成团), 这就意味着式(14.2)会有很多乘积项, 显然会给计算带来负担. 注意到若团  $Q$  不是极大团, 则它必被一个极大团  $Q^*$  所包含, 即  $\mathbf{x}_Q \subseteq \mathbf{x}_{Q^*}$ ; 这意味着变量  $\mathbf{x}_Q$  之间的关系不仅体现在势函数  $\psi_Q$  中, 还体现在  $\psi_{Q^*}$  中. 于是, 联合概率  $P(\mathbf{x})$  可基于极大团来定义. 假定所有极大团构成的集合为  $\mathcal{C}^*$ , 则有

$$P(\mathbf{x}) = \frac{1}{Z^*} \prod_{Q \in \mathcal{C}^*} \psi_Q(\mathbf{x}_Q), \quad (14.3)$$

其中  $Z^* = \sum_{\mathbf{x}} \prod_{Q \in \mathcal{C}^*} \psi_Q(\mathbf{x}_Q)$  为规范化因子. 例如图 14.2 中  $\mathbf{x} = \{x_1, x_2, \dots, x_6\}$ , 联合概率分布  $P(\mathbf{x})$  定义为

$$P(\mathbf{x}) = \frac{1}{Z} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{24}(x_2, x_4) \psi_{35}(x_3, x_5) \psi_{256}(x_2, x_5, x_6),$$

其中, 势函数  $\psi_{256}(x_2, x_5, x_6)$  定义在极大团  $\{x_2, x_5, x_6\}$  上, 由于它的存在, 使我们不再需为团  $\{x_2, x_5\}$ ,  $\{x_2, x_6\}$  和  $\{x_5, x_6\}$  构建势函数.

参见 7.5.1 节.

在马尔可夫随机场中如何得到“条件独立性”呢? 同样借助“分离”的概念, 如图 14.3 所示, 若从结点集  $A$  中的结点到  $B$  中的结点都必须经过结点集  $C$  中的结点, 则称结点集  $A$  和  $B$  被结点集  $C$  分离,  $C$  称为“分离集”(separating set). 对马尔可夫随机场, 有

- “全局马尔可夫性”(global Markov property): 给定两个变量子集的分离集, 则这两个变量子集条件独立.

也就是说, 图 14.3 中若令  $A$ ,  $B$  和  $C$  对应的变量集分别为  $\mathbf{x}_A$ ,  $\mathbf{x}_B$  和  $\mathbf{x}_C$ , 则  $\mathbf{x}_A$  和  $\mathbf{x}_B$  在给定  $\mathbf{x}_C$  的条件下独立, 记为  $\mathbf{x}_A \perp \mathbf{x}_B \mid \mathbf{x}_C$ .

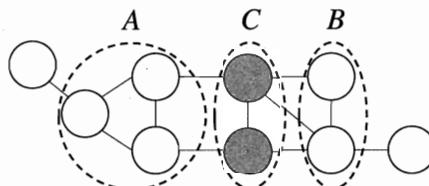


图 14.3 结点集  $A$  和  $B$  被结点集  $C$  分离

下面我们做一个简单的验证。为便于讨论，我们令图 14.3 中的  $A$ ,  $B$  和  $C$  分别对应单变量  $x_A$ ,  $x_B$  和  $x_C$ ，于是图 14.3 简化为图 14.4。



图 14.4 图 14.3 的简化版

对于图 14.4，由式(14.2)可得联合概率

$$P(x_A, x_B, x_C) = \frac{1}{Z} \psi_{AC}(x_A, x_C) \psi_{BC}(x_B, x_C). \quad (14.4)$$

基于条件概率的定义可得

$$\begin{aligned} P(x_A, x_B | x_C) &= \frac{P(x_A, x_B, x_C)}{P(x_C)} = \frac{P(x_A, x_B, x_C)}{\sum_{x'_A} \sum_{x'_B} P(x'_A, x'_B, x_C)} \\ &= \frac{\frac{1}{Z} \psi_{AC}(x_A, x_C) \psi_{BC}(x_B, x_C)}{\sum_{x'_A} \sum_{x'_B} \frac{1}{Z} \psi_{AC}(x'_A, x_C) \psi_{BC}(x'_B, x_C)} \\ &= \frac{\psi_{AC}(x_A, x_C)}{\sum_{x'_A} \psi_{AC}(x'_A, x_C)} \cdot \frac{\psi_{BC}(x_B, x_C)}{\sum_{x'_B} \psi_{BC}(x'_B, x_C)}. \end{aligned} \quad (14.5)$$

$$\begin{aligned} P(x_A | x_C) &= \frac{P(x_A, x_C)}{P(x_C)} = \frac{\sum_{x'_B} P(x_A, x'_B, x_C)}{\sum_{x'_A} \sum_{x'_B} P(x'_A, x'_B, x_C)} \\ &= \frac{\sum_{x'_B} \frac{1}{Z} \psi_{AC}(x_A, x_C) \psi_{BC}(x'_B, x_C)}{\sum_{x'_A} \sum_{x'_B} \frac{1}{Z} \psi_{AC}(x'_A, x_C) \psi_{BC}(x'_B, x_C)} \\ &= \frac{\psi_{AC}(x_A, x_C)}{\sum_{x'_A} \psi_{AC}(x'_A, x_C)}. \end{aligned} \quad (14.6)$$

由式(14.5)和(14.6)可知

$$P(x_A, x_B | x_C) = P(x_A | x_C) P(x_B | x_C), \quad (14.7)$$

即  $x_A$  和  $x_B$  在给定  $x_C$  时条件独立。

由全局马尔可夫性可得到两个很有用的推论：

- 局部马尔可夫性(local Markov property): 给定某变量的邻接变量，则该

某变量的所有邻接变量组成的集合称为该变量的“马尔可夫毯”(Markov blanket).

- 变量条件独立于其他变量. 形式化地说, 令  $V$  为图的结点集,  $n(v)$  为结点  $v$  在图上的邻接结点,  $n^*(v) = n(v) \cup \{v\}$ , 有  $\mathbf{x}_v \perp \mathbf{x}_{V \setminus n^*(v)} \mid \mathbf{x}_{n(v)}$ .
- 成对马尔可夫性(pairwise Markov property): 给定所有其他变量, 两个非邻接变量条件独立. 形式化地说, 令图的结点集和边集分别为  $V$  和  $E$ , 对图中的两个结点  $u$  和  $v$ , 若  $\langle u, v \rangle \notin E$ , 则  $\mathbf{x}_u \perp \mathbf{x}_v \mid \mathbf{x}_{V \setminus \langle u, v \rangle}$ .

现在我们来考察马尔可夫随机场中的势函数. 显然, 势函数  $\psi_Q(\mathbf{x}_Q)$  的作用是定量刻画变量集  $\mathbf{x}_Q$  中变量之间的相关关系, 它应该是非负函数, 且在所偏好的变量取值上有较大函数值. 例如, 假定图 14.4 中的变量均为二值变量, 若势函数为

$$\psi_{AC}(x_A, x_C) = \begin{cases} 1.5, & \text{if } x_A = x_C; \\ 0.1, & \text{otherwise,} \end{cases}$$

$$\psi_{BC}(x_B, x_C) = \begin{cases} 0.2, & \text{if } x_B = x_C; \\ 1.3, & \text{otherwise,} \end{cases}$$

则说明该模型偏好变量  $x_A$  与  $x_C$  拥有相同的取值,  $x_B$  与  $x_C$  拥有不同的取值; 换言之, 在该模型中  $x_A$  与  $x_C$  正相关,  $x_B$  与  $x_C$  负相关. 结合式(14.2)易知, 令  $x_A$  与  $x_C$  相同且  $x_B$  与  $x_C$  不同的变量值指派将取得较高的联合概率.

为了满足非负性, 指数函数常被用于定义势函数, 即

$$\psi_Q(\mathbf{x}_Q) = e^{-H_Q(\mathbf{x}_Q)}. \quad (14.8)$$

$H_Q(\mathbf{x}_Q)$  是一个定义在变量  $\mathbf{x}_Q$  上的实值函数, 常见形式为

$$H_Q(\mathbf{x}_Q) = \sum_{u, v \in Q, u \neq v} \alpha_{uv} x_u x_v + \sum_{v \in Q} \beta_v x_v, \quad (14.9)$$

其中  $\alpha_{uv}$  和  $\beta_v$  是参数. 上式中的第二项仅考虑单结点, 第一项则考虑每一对结点的关系.

### 14.3 条件随机场

条件随机场可看作给定观测值的马尔可夫随机场, 也可看作对率回归的扩展; 对率回归参见 3.3 节.

条件随机场(Conditional Random Field, 简称 CRF) 是一种判别式无向图模型. 14.1 节提到过, 生成式模型是直接对联合分布进行建模, 而判别式模型则是对条件分布进行建模. 前面介绍的隐马尔可夫模型和马尔可夫随机场都是生成式模型, 而条件随机场则是判别式模型.

条件随机场试图对多个变量在给定观测值后的条件概率进行建模。具体来说，若令  $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$  为观测序列， $\mathbf{y} = \{y_1, y_2, \dots, y_n\}$  为与之相应的标记序列，则条件随机场的目标是构建条件概率模型  $P(\mathbf{y} | \mathbf{x})$ 。需注意的是，标记变量  $\mathbf{y}$  可以是结构型变量，即其分量之间具有某种相关性。例如在自然语言处理的词性标注任务中，观测数据为语句（即单词序列），标记为相应的词性序列，具有线性序列结构，如图 14.5(a) 所示；在语法分析任务中，输出标记则是语法树，具有树形结构，如图 14.5(b) 所示。

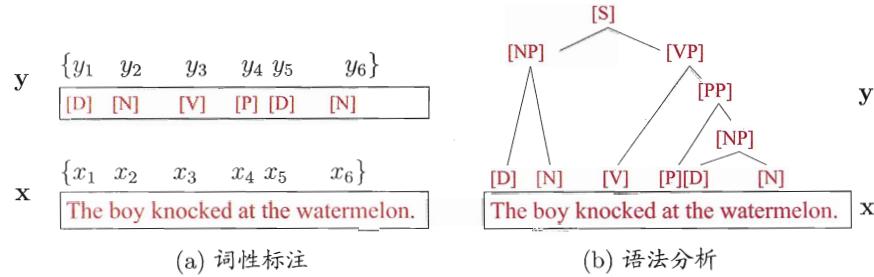


图 14.5 自然语言处理中的词性标注和语法分析任务

令  $G = \langle V, E \rangle$  表示结点与标记变量  $\mathbf{y}$  中元素一一对应的无向图， $y_v$  表示与结点  $v$  对应的标记变量， $n(v)$  表示结点  $v$  的邻接结点，若图  $G$  的每个变量  $y_v$  都满足马尔可夫性，即

$$P(y_v | \mathbf{x}, \mathbf{y}_{V \setminus \{v\}}) = P(y_v | \mathbf{x}, \mathbf{y}_{n(v)}) , \quad (14.10)$$

则  $(\mathbf{y}, \mathbf{x})$  构成一个条件随机场。

理论上来说，图  $G$  可具有任意结构，只要能表示标记变量之间的条件独立性关系即可。但在现实应用中，尤其是对标记序列建模时，最常用的仍是图 14.6 所示的链式结构，即“链式条件随机场”（chain-structured CRF）。下面我们主要讨论这种条件随机场。

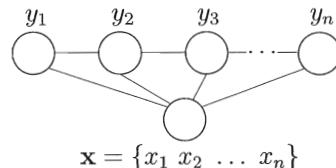


图 14.6 链式条件随机场的图结构

与马尔可夫随机场定义联合概率的方式类似, 条件随机场使用势函数和图结构上的团来定义条件概率  $P(\mathbf{y} | \mathbf{x})$ . 给定观测序列  $\mathbf{x}$ , 图 14.6 所示的链式条件随机场主要包含两种关于标记变量的团, 即单个标记变量  $\{y_i\}$  以及相邻的标记变量  $\{y_{i-1}, y_i\}$ . 选择合适的势函数, 即可得到形如式(14.2)的条件概率定义. 在条件随机场中, 通过选用指数势函数并引入特征函数(feature function), 条件概率被定义为

$$P(\mathbf{y} | \mathbf{x}) = \frac{1}{Z} \exp \left( \sum_j \sum_{i=1}^{n-1} \lambda_j t_j(y_{i+1}, y_i, \mathbf{x}, i) + \sum_k \sum_{i=1}^n \mu_k s_k(y_i, \mathbf{x}, i) \right), \quad (14.11)$$

其中  $t_j(y_{i+1}, y_i, \mathbf{x}, i)$  是定义在观测序列的两个相邻标记位置上的转移特征函数(transition feature function), 用于刻画相邻标记变量之间的相关关系以及观测序列对它们的影响,  $s_k(y_i, \mathbf{x}, i)$  是定义在观测序列的标记位置  $i$  上的状态特征函数(status feature function), 用于刻画观测序列对标记变量的影响,  $\lambda_j$  和  $\mu_k$  为参数,  $Z$  为规范化因子, 用于确保式(14.11)是正确定义的概率.

显然, 要使用条件随机场, 还需定义合适的特征函数. 特征函数通常是实值函数, 以刻画数据的一些很可能成立或期望成立的经验特性. 以图 14.5(a) 的词性标注任务为例, 若采用转移特征函数

$$t_j(y_{i+1}, y_i, \mathbf{x}, i) = \begin{cases} 1, & \text{if } y_{i+1} = [P], y_i = [V] \text{ and } x_i = \text{"knock"}; \\ 0, & \text{otherwise,} \end{cases}$$

则表示第  $i$  个观测值  $x_i$  为单词“knock”时, 相应的标记  $y_i$  和  $y_{i+1}$  很可能分别为  $[V]$  和  $[P]$ . 若采用状态特征函数

$$s_k(y_i, \mathbf{x}, i) = \begin{cases} 1, & \text{if } y_i = [V] \text{ and } x_i = \text{"knock"}; \\ 0, & \text{otherwise,} \end{cases}$$

则表示观测值  $x_i$  为单词“knock”时, 它所对应的标记很可能为  $[V]$ .

对比式(14.11)和(14.2)可看出, 条件随机场和马尔可夫随机场均使用团上的势函数定义概率, 两者在形式上没有显著区别; 但条件随机场处理的是条件概率, 而马尔可夫随机场处理的是联合概率.

## 14.4 学习与推断

基于概率图模型定义的联合概率分布, 我们能对目标变量的边际分布(marginal distribution)或以某些可观测变量为条件的条件分布进行推断。条件分布我们已经接触过很多, 例如在隐马尔可夫模型中要估算观测序列  $\mathbf{x}$  在给定参数  $\lambda$  下的条件概率分布。边际分布则是指对无关变量求和或积分后得到结果, 例如在马尔可夫网中, 变量的联合分布被表示成极大团的势函数乘积, 于是, 给定参数  $\Theta$  求解某个变量  $x$  的分布, 就变成对联合分布中其他无关变量进行积分的过程, 这称为“边际化”(marginalization)。

对概率图模型, 还需确定具体分布的参数, 这称为参数估计或参数学习问题, 通常使用极大似然估计或最大后验概率估计求解。但若将参数视为待推断的变量, 则参数估计过程和推断十分相似, 可以“吸收”到推断问题中。因此, 下面我们只讨论概率图模型的推断方法。

具体来说, 假设图模型所对应的变量集  $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$  能分为  $\mathbf{x}_E$  和  $\mathbf{x}_F$  两个不相交的变量集, 推断问题的目标就是计算边际概率  $P(\mathbf{x}_F)$  或条件概率  $P(\mathbf{x}_F | \mathbf{x}_E)$ 。由条件概率定义有

$$P(\mathbf{x}_F | \mathbf{x}_E) = \frac{P(\mathbf{x}_E, \mathbf{x}_F)}{P(\mathbf{x}_E)} = \frac{P(\mathbf{x}_E, \mathbf{x}_F)}{\sum_{\mathbf{x}_F} P(\mathbf{x}_E, \mathbf{x}_F)}, \quad (14.12)$$

其中联合概率  $P(\mathbf{x}_E, \mathbf{x}_F)$  可基于概率图模型获得, 因此, 推断问题的关键就是如何高效地计算边际分布, 即

$$P(\mathbf{x}_E) = \sum_{\mathbf{x}_F} P(\mathbf{x}_E, \mathbf{x}_F). \quad (14.13)$$

概率图模型的推断方法大致可分为两类。第一类是精确推断方法, 希望能计算出目标变量的边际分布或条件分布的精确值; 遗憾的是, 一般情形下, 此类算法的计算复杂度随着极大团规模的增长呈指数增长, 适用范围有限。第二类是近似推断方法, 希望在较低的时间复杂度下获得原问题的近似解; 此类方法在现实任务中更常用。本节介绍两种代表性的精确推断方法, 下一节介绍近似推断方法。

### 14.4.1 变量消去

精确推断的实质是一类动态规划算法, 它利用图模型所描述的条件独立性来削减计算目标概率值所需的计算量。变量消去法是最直观的精确推断算法,

也是构建其他精确推断算法的基础.

我们先以图 14.7(a) 中的有向图模型为例来介绍其工作流程.

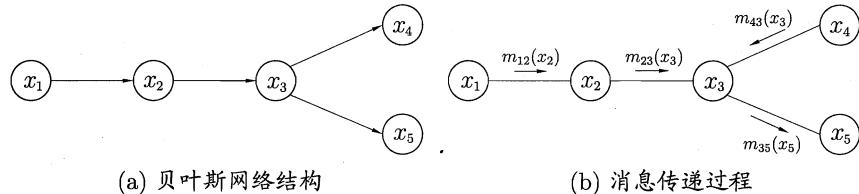


图 14.7 变量消去法及其对应的消息传递过程

假定推断目标是计算边际概率  $P(x_5)$ . 显然, 为了完成此目标, 只需通过加法消去变量  $\{x_1, x_2, x_3, x_4\}$ , 即

$$\begin{aligned} P(x_5) &= \sum_{x_4} \sum_{x_3} \sum_{x_2} \sum_{x_1} P(x_1, x_2, x_3, x_4, x_5) \\ &= \sum_{x_4} \sum_{x_3} \sum_{x_2} \sum_{x_1} P(x_1)P(x_2 | x_1)P(x_3 | x_2)P(x_4 | x_3)P(x_5 | x_3). \end{aligned} \quad (14.14)$$

基于有向图模型所描述的条件独立性.

不难发现, 若采用  $\{x_1, x_2, x_4, x_3\}$  的顺序计算加法, 则有

$$\begin{aligned} P(x_5) &= \sum_{x_3} P(x_5 | x_3) \sum_{x_4} P(x_4 | x_3) \sum_{x_2} P(x_3 | x_2) \sum_{x_1} P(x_1)P(x_2 | x_1) \\ &= \sum_{x_3} P(x_5 | x_3) \sum_{x_4} P(x_4 | x_3) \sum_{x_2} P(x_3 | x_2) m_{12}(x_2), \end{aligned} \quad (14.15)$$

其中  $m_{ij}(x_j)$  是求加过程的中间结果, 下标  $i$  表示此项是对  $x_i$  求加的结果, 下标  $j$  表示此项中剩下的其他变量. 显然,  $m_{ij}(x_j)$  是关于  $x_j$  的函数. 不断执行此过程可得

$$\begin{aligned} P(x_5) &= \sum_{x_3} P(x_5 | x_3) \sum_{x_4} P(x_4 | x_3) m_{23}(x_3) \\ &= \sum_{x_3} P(x_5 | x_3) m_{23}(x_3) \sum_{x_4} P(x_4 | x_3) \\ &= \sum_{x_3} P(x_5 | x_3) m_{23}(x_3) m_{43}(x_3) \\ &= m_{35}(x_5). \end{aligned} \quad (14.16)$$

显然, 最后的  $m_{35}(x_5)$  是关于  $x_5$  的函数, 仅与变量  $x_5$  的取值有关.

事实上, 上述方法对无向图模型同样适用. 不妨忽略图 14.7(a) 中的箭头, 将其看作一个无向图模型, 有

$$P(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z} \psi_{12}(x_1, x_2) \psi_{23}(x_2, x_3) \psi_{34}(x_3, x_4) \psi_{35}(x_3, x_5), \quad (14.17)$$

其中  $Z$  为规范化因子. 边际分布  $P(x_5)$  可这样计算:

$$\begin{aligned} P(x_5) &= \frac{1}{Z} \sum_{x_3} \psi_{35}(x_3, x_5) \sum_{x_4} \psi_{34}(x_3, x_4) \sum_{x_2} \psi_{23}(x_2, x_3) \sum_{x_1} \psi_{12}(x_1, x_2) \\ &= \frac{1}{Z} \sum_{x_3} \psi_{35}(x_3, x_5) \sum_{x_4} \psi_{34}(x_3, x_4) \sum_{x_2} \psi_{23}(x_2, x_3) m_{12}(x_2) \\ &= \dots \\ &= \frac{1}{Z} m_{35}(x_5). \end{aligned} \quad (14.18)$$

显然, 通过利用乘法对加法的分配律, 变量消去法把多个变量的积的求和问题, 转化为对部分变量交替进行求积与求和的问题. 这种转化使得每次的求和与求积运算限制在局部, 仅与部分变量有关, 从而简化了计算.

变量消去法有一个明显的缺点: 若需计算多个边际分布, 重复使用变量消去法将会造成大量的冗余计算. 例如在图 14.7(a) 的贝叶斯网上, 假定在计算  $P(x_5)$  之外还希望计算  $P(x_4)$ , 若采用  $\{x_1, x_2, x_5, x_3\}$  的顺序, 则  $m_{12}(x_2)$  和  $m_{23}(x_3)$  的计算是重复的.

#### 14.4.2 信念传播

亦称 Sum-Product 算法.

信念传播(Belief Propagation)算法将变量消去法中的求和操作看作一个消息传递过程, 较好地解决了求解多个边际分布时的重复计算问题. 具体来说, 变量消去法通过求和操作

$$m_{ij}(x_j) = \sum_{x_i} \psi(x_i, x_j) \prod_{k \in n(i) \setminus j} m_{ki}(x_i) \quad (14.19)$$

消去变量  $x_i$ , 其中  $n(i)$  表示结点  $x_i$  的邻接结点. 在信念传播算法中, 这个操作被看作从  $x_i$  向  $x_j$  传递了一个消息  $m_{ij}(x_j)$ . 这样, 式(14.15)和(14.16)所描述的变量消去过程就能描述为图 14.7(b) 所示的消息传递过程. 不难发现, 每次消息传递操作仅与变量  $x_i$  及其邻接结点直接相关, 换言之, 消息传递相关的计算被

限制在图的局部进行.

在信念传播算法中, 一个结点仅在接收到来自其他所有结点的消息后才能向另一个结点发送消息, 且结点的边际分布正比于它所接收的消息的乘积, 即

$$P(x_i) \propto \prod_{k \in n(i)} m_{ki}(x_i). \quad (14.20)$$

例如在图 14.7(b) 中, 结点  $x_3$  要向  $x_5$  发送消息, 必须事先收到来自结点  $x_2$  和  $x_4$  的消息, 且传递到  $x_5$  的消息  $m_{35}(x_5)$  恰为概率  $P(x_5)$ .

若图结构中没有环, 则信念传播算法经过两个步骤即可完成所有消息传递, 进而能计算所有变量上的边际分布:

- 指定一个根结点, 从所有叶结点开始向根结点传递消息, 直到根结点收到所有邻接结点的消息;
- 从根结点开始向叶结点传递消息, 直到所有叶结点均收到消息.

例如在图 14.7(a)中, 令  $x_1$  为根结点, 则  $x_4$  和  $x_5$  为叶结点. 以上两步消息传递的过程如图 14.8 所示. 此时图的每条边上都有方向不同的两条消息, 基于这些消息和式(14.20)即可获得所有变量的边际概率.

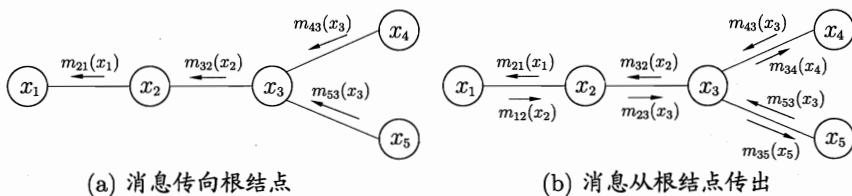


图 14.8 信念传播算法图示

## 14.5 近似推断

精确推断方法通常需要很大的计算开销, 因此在现实应用中近似推断方法更为常用. 近似推断方法大致可分为两大类: 第一类是采样(sampling), 通过使用随机化方法完成近似; 第二类是使用确定性近似完成近似推断, 典型代表为变分推断(variational inference).

### 14.5.1 MCMC采样

在很多任务中, 我们关心某些概率分布并非因为对这些概率分布本身感兴

趣，而是要基于它们计算某些期望，并且还可能进一步基于这些期望做出决策。例如对图 14.7(a) 的贝叶斯网，进行推断的目的可能是为了计算变量  $x_5$  的期望。若直接计算或逼近这个期望比推断概率分布更容易，则直接操作无疑将使推断问题的求解更为高效。

采样法正是基于这个思路。具体来说，假定我们的目标是计算函数  $f(x)$  在概率密度函数  $p(x)$  下的期望

若  $x$  是离散变量，则把积分换做求和即可。

或  $p(x)$  的相关分布。

$$\mathbb{E}_p[f] = \int f(x)p(x)dx, \quad (14.21)$$

则可根据  $p(x)$  抽取一组样本  $\{x_1, x_2, \dots, x_N\}$ ，然后计算  $f(x)$  在这些样本上的均值

$$\hat{f} = \frac{1}{N} \sum_{i=1}^N f(x_i), \quad (14.22)$$

以此来近似目标期望  $\mathbb{E}[f]$ 。若样本  $\{x_1, x_2, \dots, x_N\}$  独立，基于大数定律，这种通过大量采样的办法就能获得较高的近似精度。问题的关键是如何采样。对概率图模型来说，就是如何高效地基于图模型所描述的概率分布来获取样本。

概率图模型中最常用的采样技术是马尔可夫链蒙特卡罗(Markov Chain Monte Carlo, 简称 MCMC)方法。给定连续变量  $x \in X$  的概率密度函数  $p(x)$ ， $x$  在区间  $A$  中的概率可计算为

$$P(A) = \int_A p(x)dx. \quad (14.23)$$

若有函数  $f : X \mapsto \mathbb{R}$ ，则可计算  $f(x)$  的期望

$$p(f) = \mathbb{E}_p[f(X)] = \int_x f(x)p(x)dx. \quad (14.24)$$

若  $x$  不是单变量而是一个高维多元变量  $\mathbf{x}$ ，且服从一个非常复杂的分布，则对式(14.24)求积分通常很困难。为此，MCMC 先构造出服从  $p$  分布的独立同分布随机变量  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ ，再得到式(14.24)的无偏估计

$$\tilde{p}(f) = \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}_i). \quad (14.25)$$

然而，若概率密度函数  $p(\mathbf{x})$  很复杂，则构造服从  $p$  分布的独立同分布样本

也很困难. MCMC 方法的关键就在于通过构造“平稳分布为  $p$  的马尔可夫链”来产生样本: 若马尔可夫链运行时间足够长(即收敛到平稳状态), 则此时产出的样本  $\mathbf{x}$  近似服从于分布  $p$ . 如何判断马尔可夫链到达平稳状态呢? 假定平稳马尔可夫链  $T$  的状态转移概率(即从状态  $\mathbf{x}$  转移到状态  $\mathbf{x}'$  的概率)为  $T(\mathbf{x}' | \mathbf{x})$ ,  $t$  时刻状态的分布为  $p(\mathbf{x}^t)$ , 则若在某个时刻马尔可夫链满足平稳条件

$$p(\mathbf{x}^t)T(\mathbf{x}^{t-1} | \mathbf{x}^t) = p(\mathbf{x}^{t-1})T(\mathbf{x}^t | \mathbf{x}^{t-1}), \quad (14.26)$$

则  $p(\mathbf{x})$  是该马尔可夫链的平稳分布, 且马尔可夫链在满足该条件时已收敛到平稳状态.

也就是说, MCMC 方法先设法构造一条马尔可夫链, 使其收敛至平稳分布恰为待估计参数的后验分布, 然后通过这条马尔可夫链来产生符合后验分布的样本, 并基于这些样本来进行估计. 这里马尔可夫链转移概率的构造至关重要, 不同的构造方法将产生不同的 MCMC 算法.

Metropolis-Hastings 算法是由 N. Metropolis 等人 1953 年提出 [Metropolis et al., 1953], 此后 W. K. Hastings 将其推广到一般形式 [Hastings, 1970], 因此而得名.

Metropolis-Hastings (简称 MH) 算法是 MCMC 的重要代表. 它基于“拒绝采样”(reject sampling) 来逼近平稳分布  $p$ . 如图 14.9 所示, 算法每次根据上一轮采样结果  $\mathbf{x}^{t-1}$  来采样获得候选状态样本  $\mathbf{x}^*$ , 但这个候选样本会以一定的概率被“拒绝”掉. 假定从状态  $\mathbf{x}^{t-1}$  到状态  $\mathbf{x}^*$  的转移概率为  $Q(\mathbf{x}^* | \mathbf{x}^{t-1})A(\mathbf{x}^* | \mathbf{x}^{t-1})$ , 其中  $Q(\mathbf{x}^* | \mathbf{x}^{t-1})$  是用户给定的先验概率,  $A(\mathbf{x}^* | \mathbf{x}^{t-1})$  是  $\mathbf{x}^*$  被接受的概率. 若  $\mathbf{x}^*$  最终收敛到平稳状态, 则根据式(14.26)有

$$p(\mathbf{x}^{t-1})Q(\mathbf{x}^* | \mathbf{x}^{t-1})A(\mathbf{x}^* | \mathbf{x}^{t-1}) = p(\mathbf{x}^*)Q(\mathbf{x}^{t-1} | \mathbf{x}^*)A(\mathbf{x}^{t-1} | \mathbf{x}^*), \quad (14.27)$$

---

输入: 先验概率  $Q(\mathbf{x}^* | \mathbf{x}^{t-1})$ .

过程:

- 1: 初始化  $\mathbf{x}^0$ ;
- 2: **for**  $t = 1, 2, \dots$  **do**
- 3:   根据  $Q(\mathbf{x}^* | \mathbf{x}^{t-1})$  采样出候选样本  $\mathbf{x}^*$ ;
- 4:   根据均匀分布从  $(0, 1)$  范围内采样出阈值  $u$ ;
- 5:   **if**  $u \leq A(\mathbf{x}^* | \mathbf{x}^{t-1})$  **then**
- 6:      $\mathbf{x}^t = \mathbf{x}^*$
- 7:   **else**
- 8:      $\mathbf{x}^t = \mathbf{x}^{t-1}$
- 9:   **end if**
- 10: **end for**
- 11: **return**  $\mathbf{x}^1, \mathbf{x}^2, \dots$

---

输出: 采样出的一个样本序列  $\mathbf{x}^1, \mathbf{x}^2, \dots$

---

重复足够多次以达到平稳分布.

根据式(14.28).

实践中常会丢弃前面若干个样本, 因为达到平稳分布后产生的才是希望得到的样本.

图 14.9 Metropolis-Hastings 算法

于是, 为了达到平稳状态, 只需将接受率设置为

$$A(\mathbf{x}^* \mid \mathbf{x}^{t-1}) = \min \left( 1, \frac{p(\mathbf{x}^*)Q(\mathbf{x}^{t-1} \mid \mathbf{x}^*)}{p(\mathbf{x}^{t-1})Q(\mathbf{x}^* \mid \mathbf{x}^{t-1})} \right). \quad (14.28)$$

参见 7.5.3 节.

吉布斯采样(Gibbs sampling)有时被视为 MH 算法的特例, 它也使用马尔可夫链获取样本, 而该马尔可夫链的平稳分布也是采样的目标分布  $p(\mathbf{x})$ . 具体来说, 假定  $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$ , 目标分布为  $p(\mathbf{x})$ , 在初始化  $\mathbf{x}$  的取值后, 通过循环执行以下步骤来完成采样:

- (1) 随机或以某个次序选取某变量  $x_i$ ;
- (2) 根据  $\mathbf{x}$  中除  $x_i$  外的变量的现有取值, 计算条件概率  $p(x_i \mid \mathbf{x}_{\bar{i}})$ , 其中  $\mathbf{x}_{\bar{i}} = \{x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_N\}$ ;
- (3) 根据  $p(x_i \mid \mathbf{x}_{\bar{i}})$  对变量  $x_i$  采样, 用采样值代替原值.

#### 14.5.2 变分推断

变分推断通过使用已知简单分布来逼近需推断的复杂分布, 并通过限制近似分布的类型, 从而得到一种局部最优、但具有确定解的近似后验分布.

在学习变分推断之前, 我们先介绍概率图模型一种简洁的表示方法——盘式记法(plate notation) [Buntine, 1994]. 图 14.10 给出了一个简单的例子. 图 14.10(a)表示  $N$  个变量  $\{x_1, x_2, \dots, x_N\}$  均依赖于其他变量  $\mathbf{z}$ . 在图 14.10(b)中, 相互独立的、由相同机制生成的多个变量被放在一个方框(盘)内, 并在方框中标出类似变量重复出现的个数  $N$ ; 方框可以嵌套. 通常用阴影标注出已知的、能观察到的变量, 如图 14.10 中的变量  $x$ . 在很多学习任务中, 对属性变量使用盘式记法将使得图表示非常简洁.

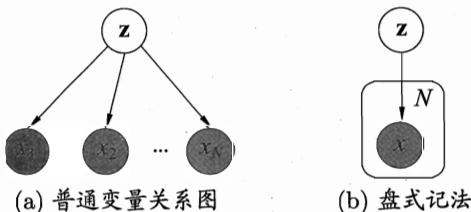


图 14.10 盘式记法的示例

变分推断使用的近似分布需具有良好的数值性质，通常是基于连续型变量的概率密度函数来刻画的。

在图 14.10(b)中，所有能观察到的变量  $x$  的联合分布的概率密度函数是

$$p(\mathbf{x} | \Theta) = \prod_{i=1}^N \sum_{\mathbf{z}} p(x_i, \mathbf{z} | \Theta), \quad (14.29)$$

所对应的对数似然函数为

$$\ln p(\mathbf{x} | \Theta) = \sum_{i=1}^N \ln \left\{ \sum_{\mathbf{z}} p(x_i, \mathbf{z} | \Theta) \right\}, \quad (14.30)$$

其中  $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$ ,  $\Theta$  是  $\mathbf{x}$  与  $\mathbf{z}$  服从的分布参数。

一般来说，图 14.10 所对应的推断和学习任务主要是由观察到的变量  $\mathbf{x}$  来估计隐变量  $\mathbf{z}$  和分布参数变量  $\Theta$ ，即求解  $p(\mathbf{z} | \mathbf{x}, \Theta)$  和  $\Theta$ 。

概率模型的参数估计通常以最大化对数似然函数为手段。对式(14.30)可使用 EM 算法：在 E 步，根据  $t$  时刻的参数  $\Theta^t$  对  $p(\mathbf{z} | \mathbf{x}, \Theta^t)$  进行推断，并计算联合似然函数  $p(\mathbf{x}, \mathbf{z} | \Theta)$ ；在 M 步，基于 E 步的结果进行最大化寻优，即对关于变量  $\Theta$  的函数  $Q(\Theta; \Theta^t)$  进行最大化从而求取

$$\begin{aligned} \Theta^{t+1} &= \arg \max_{\Theta} Q(\Theta; \Theta^t) \\ &= \arg \max_{\Theta} \sum_{\mathbf{z}} p(\mathbf{z} | \mathbf{x}, \Theta^t) \ln p(\mathbf{x}, \mathbf{z} | \Theta). \end{aligned} \quad (14.31)$$

式(14.31)中的  $Q(\Theta; \Theta^t)$  实际上是对数联合似然函数  $\ln p(\mathbf{x}, \mathbf{z} | \Theta)$  在分布  $p(\mathbf{z} | \mathbf{x}, \Theta^t)$  下的期望，当分布  $p(\mathbf{z} | \mathbf{x}, \Theta^t)$  与变量  $\mathbf{z}$  的真实后验分布相等时， $Q(\Theta; \Theta^t)$  近似于对数似然函数。于是，EM 算法最终可获得稳定的参数  $\Theta$ ，而隐变量  $\mathbf{z}$  的分布也能通过该参数获得。

需注意的是， $p(\mathbf{z} | \mathbf{x}, \Theta^t)$  未必是隐变量  $\mathbf{z}$  服从的真实分布，而只是一个近似分布。若将这个近似分布用  $q(\mathbf{z})$  表示，则不难验证

$$\ln p(\mathbf{x}) = \mathcal{L}(q) + \text{KL}(q \| p), \quad (14.32)$$

其中

$$\mathcal{L}(q) = \int q(\mathbf{z}) \ln \left\{ \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right\} d\mathbf{z}, \quad (14.33)$$

KL 散度，参见附录 C.3。

$$\text{KL}(q \| p) = - \int q(\mathbf{z}) \ln \frac{p(\mathbf{z} | \mathbf{x})}{q(\mathbf{z})} d\mathbf{z}. \quad (14.34)$$

然而在现实任务中, E步对  $p(\mathbf{z} | \mathbf{x}, \Theta^t)$  的推断很可能因  $\mathbf{z}$  模型复杂而难以进行, 此时可借助变分推断。通常假设  $\mathbf{z}$  服从分布

$$q(\mathbf{z}) = \prod_{i=1}^M q_i(\mathbf{z}_i), \quad (14.35)$$

即假设复杂的多变量  $\mathbf{z}$  可拆解为一系列相互独立的多变量  $\mathbf{z}_i$ 。更重要的是, 为简化表述, 这里将  $q_i(\mathbf{z}_i)$  简写为  $q_i$ 。可以令  $q_i$  分布相对简单或有很好的结构, 例如假设  $q_i$  为指数族(exponential family)分布, 此时有

$$\begin{aligned} \mathcal{L}(q) &= \int \prod_i q_i \left\{ \ln p(\mathbf{x}, \mathbf{z}) - \sum_i \ln q_i \right\} d\mathbf{z} \\ &= \int q_j \left\{ \int \ln p(\mathbf{x}, \mathbf{z}) \prod_{i \neq j} q_i d\mathbf{z}_i \right\} d\mathbf{z}_j - \int q_j \ln q_j d\mathbf{z}_j + \text{const} \\ &= \int q_j \ln \tilde{p}(\mathbf{x}, \mathbf{z}_j) d\mathbf{z}_j - \int q_j \ln q_j d\mathbf{z}_j + \text{const}, \end{aligned} \quad (14.36)$$

const 是一个常数。

其中

$$\ln \tilde{p}(\mathbf{x}, \mathbf{z}_j) = \mathbb{E}_{i \neq j} [\ln p(\mathbf{x}, \mathbf{z})] + \text{const}, \quad (14.37)$$

$$\mathbb{E}_{i \neq j} [\ln p(\mathbf{x}, \mathbf{z})] = \int \ln p(\mathbf{x}, \mathbf{z}) \prod_{i \neq j} q_i d\mathbf{z}_i. \quad (14.38)$$

我们关心的是  $q_j$ , 因此可固定  $q_{i \neq j}$  再对  $\mathcal{L}(q)$  进行最大化, 可发现式(14.36)等于  $-\text{KL}(q_j \parallel \tilde{p}(\mathbf{x}, \mathbf{z}_j))$ , 即当  $q_j = \tilde{p}(\mathbf{x}, \mathbf{z}_j)$  时  $\mathcal{L}(q)$  最大。于是可知变量子集  $\mathbf{z}_j$  所服从的最优分布  $q_j^*$  应满足

$$\ln q_j^*(\mathbf{z}_j) = \mathbb{E}_{i \neq j} [\ln p(\mathbf{x}, \mathbf{z})] + \text{const}, \quad (14.39)$$

即

$$q_j^*(\mathbf{z}_j) = \frac{\exp(\mathbb{E}_{i \neq j} [\ln p(\mathbf{x}, \mathbf{z})])}{\int \exp(\mathbb{E}_{i \neq j} [\ln p(\mathbf{x}, \mathbf{z})]) d\mathbf{z}_j}. \quad (14.40)$$

换言之, 在式(14.35)这个假设下, 变量子集  $\mathbf{z}_j$  最接近真实情形的分布由式(14.40)给出。

显然, 基于式(14.35)的假设, 通过恰当地分割独立变量子集  $\mathbf{z}_j$  并选择  $q_i$  服从的分布,  $\mathbb{E}_{i \neq j} [\ln p(\mathbf{x}, \mathbf{z})]$  往往有闭式解, 这使得基于式(14.40)能高效地对隐变量  $\mathbf{z}$  进行推断。事实上, 由式(14.38)可看出, 对变量子集  $\mathbf{z}_j$  分布  $q_j^*$  进行估计时融合

mean 指期望, field 则是指分布.

了  $\mathbf{z}_j$  之外的其他  $\mathbf{z}_{i \neq j}$  的信息, 这是通过联合似然函数  $\ln p(\mathbf{x}, \mathbf{z})$  在  $\mathbf{z}_j$  之外的隐变量分布上求期望得到的, 因此亦称“平均场”(mean field)方法.

在实践中使用变分法时, 最重要的是考虑如何对隐变量进行拆解, 以及假设各变量子集服从何种分布, 在此基础上套用式(14.40)的结论再结合 EM 算法即可进行概率图模型的推断和参数估计. 显然, 若隐变量的拆解或变量子集的分布假设不当, 将会导致变分法效率低、效果差.

## 14.6 话题模型

话题模型(topic model)是一族生成式有向图模型, 主要用于处理离散型的数据(如文本集合), 在信息检索、自然语言处理等领域有广泛应用. 隐狄利克雷分配模型(Latent Dirichlet Allocation, 简称 LDA)是话题模型的典型代表.

我们先来了解一下话题模型中的几个概念: 词(word)、文档(document)和话题(topic). 具体来说, “词”是待处理数据的基本离散单元, 例如在文本处理任务中, 一个词就是一个英文单词或有独立意义的中文词. “文档”是待处理的数据对象, 它由一组词组成, 这些词在文档中是不计顺序的, 例如一篇论文、一个网页都可看作一个文档; 这样的表示方式称为“词袋”(bag-of-words). 数据对象只要能用词袋描述, 就可使用话题模型. “话题”表示一个概念, 具体表示为一系列相关的词, 以及它们在该概念下出现的概率.

形象地说, 如图 14.11 所示, 一个话题就像是一个箱子, 里面装着在这个概念下出现概率较高的那些词. 不妨假定数据集中一共包含  $K$  个话题和  $T$  篇文档, 文档中的词来自一个包含  $N$  个词的词典. 我们用  $T$  个  $N$  维向量  $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_T\}$  表示数据集(即文档集合),  $K$  个  $N$  维向量  $\beta_k$  ( $k = 1, 2, \dots, K$ ) 表示话题, 其中  $\mathbf{w}_t \in \mathbb{R}^N$  的第  $n$  个分量  $w_{t,n}$  表示文档  $t$  中词  $n$  的词频,  $\beta_k \in \mathbb{R}^N$  的第  $n$  个分量  $\beta_{k,n}$  表示话题  $k$  中词  $n$  的词频.

在现实任务中可通过统计文档中出现的词来获得词频向量  $\mathbf{w}_i$  ( $i = 1, 2, \dots, T$ ), 但通常并不知道这组文档谈论了哪些话题, 也不知道每篇文档与哪些话题有关. LDA 从生成式模型的角度来看待文档和话题. 具体来说, LDA 认为每篇文档包含多个话题, 不妨用向量  $\Theta_t \in \mathbb{R}^K$  表示文档  $t$  中所包含的每个话题的比例,  $\Theta_{t,k}$  即表示文档  $t$  中包含话题  $k$  的比例, 进而通过下面的步骤由话题“生成”文档  $t$ :

通常需对词频做一些  
处理, 例如去除“停用词  
表”中的词等.

狄利克雷分布参见附录  
C.1.6.

- (1) 根据参数为  $\alpha$  的狄利克雷分布随机采样一个话题分布  $\Theta_t$ ;
- (2) 按如下步骤生成文档中的  $N$  个词:

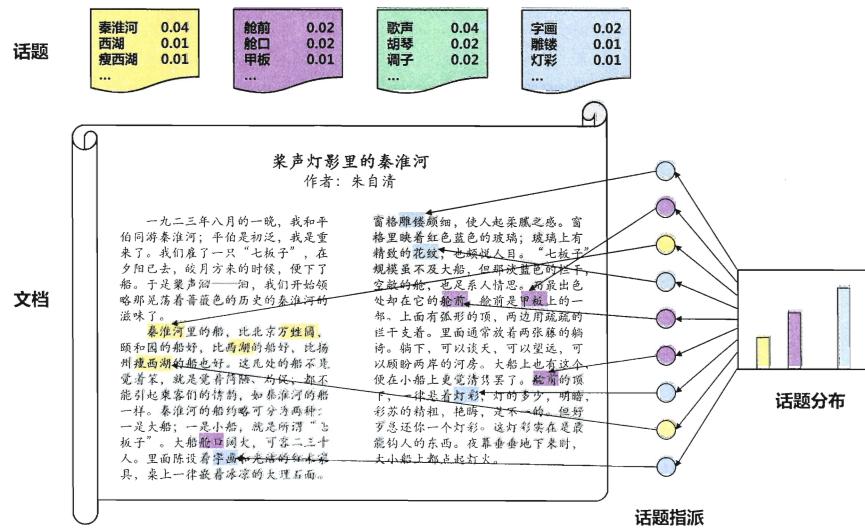


图 14.11 LDA 的文档生成过程示意图

- 根据  $\Theta_t$  进行话题指派, 得到文档  $t$  中词  $n$  的话题  $z_{t,n}$ ;
- 根据指派的话题所对应的词频分布  $\beta_k$  随机采样生成词.

图 14.11 演示出根据以上步骤生成文档的过程. 显然, 这样生成的文档自然地以不同比例包含多个话题 (步骤 1), 文档中的每个词来自一个话题 (步骤 2b), 而这个话题是依据话题比例产生的 (步骤 2a).

图 14.12 描述了 LDA 的变量关系, 其中文档中的词频  $w_{t,n}$  是唯一的已观测变量, 它依赖于对这个词进行的话题指派  $z_{t,n}$ , 以及话题所对应的词频  $\beta_k$ ; 同时, 话题指派  $z_{t,n}$  依赖于话题分布  $\Theta_t$ ,  $\Theta_t$  依赖于狄利克雷分布的参数  $\alpha$ , 而话题词频则依赖于参数  $\eta$ .

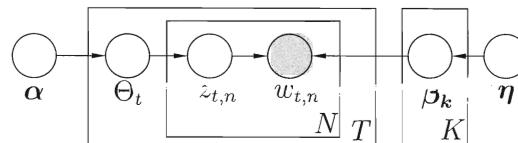


图 14.12 LDA 的盘式记法图

于是, LDA 模型对应的概率分布为

$$p(\mathbf{W}, \mathbf{z}, \boldsymbol{\beta}, \Theta | \boldsymbol{\alpha}, \boldsymbol{\eta}) = \prod_{t=1}^T p(\Theta_t | \boldsymbol{\alpha}) \prod_{i=1}^K p(\beta_k | \boldsymbol{\eta}) \left( \prod_{n=1}^N P(w_{t,n} | z_{t,n}, \beta_k) P(z_{t,n} | \Theta_t) \right), \quad (14.41)$$

其中  $p(\Theta_t | \boldsymbol{\alpha})$  和  $p(\beta_k | \boldsymbol{\eta})$  通常分别设置为以  $\boldsymbol{\alpha}$  和  $\boldsymbol{\eta}$  为参数的  $K$  维和  $N$  维狄利克雷分布, 例如

$$p(\Theta_t | \boldsymbol{\alpha}) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k \Theta_{t,k}^{\alpha_k - 1}, \quad (14.42)$$

参见附录 C.1.5.

其中  $\Gamma(\cdot)$  是 Gamma 函数. 显然,  $\boldsymbol{\alpha}$  和  $\boldsymbol{\eta}$  是模型式(14.41) 中待确定的参数.

训练文档集对应的词频.

给定训练数据  $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_T\}$ , LDA 的模型参数可通过极大似然法估计, 即寻找  $\boldsymbol{\alpha}$  和  $\boldsymbol{\eta}$  以最大化对数似然

$$LL(\boldsymbol{\alpha}, \boldsymbol{\eta}) = \sum_{t=1}^T \ln p(\mathbf{w}_t | \boldsymbol{\alpha}, \boldsymbol{\eta}). \quad (14.43)$$

但由于  $p(\mathbf{w}_t | \boldsymbol{\alpha}, \boldsymbol{\eta})$  不易计算, 式(14.43)难以直接求解, 因此实践中常采用变分法来求取近似解.

若模型已知, 即参数  $\boldsymbol{\alpha}$  和  $\boldsymbol{\eta}$  已确定, 则根据词频  $w_{t,n}$  来推断文档集所对应的话题结构(即推断  $\Theta_t$ ,  $\beta_k$  和  $z_{t,n}$ )可通过求解

$$p(\mathbf{z}, \boldsymbol{\beta}, \Theta | \mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\eta}) = \frac{p(\mathbf{W}, \mathbf{z}, \boldsymbol{\beta}, \Theta | \boldsymbol{\alpha}, \boldsymbol{\eta})}{p(\mathbf{W} | \boldsymbol{\alpha}, \boldsymbol{\eta})}. \quad (14.44)$$

然而由于分母上的  $p(\mathbf{W} | \boldsymbol{\alpha}, \boldsymbol{\eta})$  难以获取, 式(14.44)难以直接求解, 因此在实践中常采用吉布斯采样或变分法进行近似推断.

## 14.7 阅读材料

概率图模型方面已经有专门的书籍如 [Koller and Friedman, 2009].

[Pearl, 1982] 倡导了贝叶斯网的研究, [Pearl, 1988] 对这方面的早期研究工作进行了总结. 马尔可夫随机场由 [Geman and Geman, 1984] 提出. 现实应用中使用的模型经常是贝叶斯网与马尔可夫随机场的结合. 隐马尔可夫模型及其在语音识别中的应用可参阅 [Rabiner, 1989]. 条件随机场由 [Lafferty et al., 2001] 提出, 更多的内容可参阅 [Sutton and McCallum, 2012].

信念传播算法最早由 [Pearl, 1986] 作为精确推断技术提出, 后来衍生出多种近似推断算法。对一般的带环图, 信念传播算法需在初始化、消息传递等环节进行调整, 由此形成了迭代信念传播算法(Loopy Belief Propagation) [Murphy et al., 1999], 但其理论性质尚不清楚, 这方面的进展可参阅 [Mooij and Kappen, 2007; Weiss, 2000]. 有些带环图可先用“因子图” (factor graph) [Kschischang et al., 2001] 描述, 再转化为因子树(factor tree) 进行信念传播。对任意图结构的信念传播已有一些研究 [Lauritzen and Spiegelhalter, 1988]. 近来随着并行计算技术的发展, 信念传播的并行加速实现受到关注, 例如 [Gonzalez et al., 2009] 提出  $\tau_\epsilon$  近似推断的概念并设计出多核并行信念传播算法, 其时间开销随内核数的增加而线性降低。

概率图模型的建模和推断, 尤其是变分推断在 20 世纪 90 年代中期逐步发展成熟, [Jordan, 1998] 对这个阶段的主要成果进行了总结。关于变分推断的更多内容可参阅 [Wainwright and Jordan, 2008].

图模型带来的一个好处是使得人们能直观、快速地针对具体任务定义模型。LDA [Blei et al., 2003] 是这方面的重要代表, 由它产生了很多变体, 关于这方面的内容可参阅 [Blei, 2012]. 概率图模型的一个发展方向是使得模型的结构能对数据有一定的自适应能力, 即“非参数化” (non-parametric) 方法, 例如层次化狄利克雷过程模型 [Teh et al., 2006]、无限隐特征模型 [Ghahramani and Griffiths, 2006] 等。

话题模型包含了多种模型, 其中有些并不采用贝叶斯学习方法, 例如 PLSA (概率隐语义分析) [Hofmann, 2001], 它是 LSA (隐语义分析) 的概率扩展。

蒙特卡罗方法是二十世纪四十年代产生的一类基于概率统计理论、使用随机数来解决问题的数值计算方法, MCMC 是马尔可夫链与蒙特卡罗方法的结合, 最早由 [Pearl, 1987] 引入贝叶斯网推断。关于 MCMC 在概率推断中的应用可参阅 [Neal, 1993], 更多关于 MCMC 的内容可参阅 [Andrieu et al., 2003; Gilks et al., 1996].

“非参数化”指参数的数目无须事先指定, 是贝叶斯学习方法的重要发展。

贝叶斯学习参见 p.164.

LSA 是 SVD 在文本数据上的变体。

参见 p.266.

## 习题

- 14.1** 试用盘式记法表示条件随机场和朴素贝叶斯分类器.
- 14.2** 试证明图模型中的局部马尔可夫性: 给定某变量的邻接变量, 则该变量条件独立于其他变量.
- 14.3** 试证明图模型中的成对马尔可夫性: 给定其他所有变量, 则两个非邻接变量条件独立.
- 14.4** 试述在马尔可夫随机场中为何仅需对极大团定义势函数.
- 14.5** 比较条件随机场和对率回归, 试析其异同.
- 14.6** 试证明变量消去法的计算复杂度随图模型中极大团规模的增长而呈指数增长, 但随结点数的增长未必呈指数增长.
- 14.7** 吉布斯采样可看作 MH 算法的特例, 但吉布斯采样中未使用“拒绝采样”策略, 试述这样做的好处.
- 14.8** 平均场是一种近似推断方法. 考虑式(14.32), 试析平均场方法求解的近似问题与原问题的差异, 以及实践中如何选择变量服从的先验分布.
- 14.9\*** 从网上下载或自己编程实现 LDA, 试分析金庸作品《天龙八部》中每十回的话题演变情况.
- 14.10\*** 试设计一个无须事先指定话题数目的 LDA 改进算法.

## 参考文献

- Andrieu, C., N. De Freitas, A. Doucet, and M. I. Jordan. (2003). “An introduction to MCMC for machine learning.” *Machine Learning*, 50(1-2):5–43.
- Blei, D. M. (2012). “Probabilistic topic models.” *Communications of the ACM*, 55(4):77–84.
- Blei, D. M., A. Ng, and M. I. Jordan. (2003). “Latent Dirichlet allocation.” *Journal of Machine Learning Research*, 3:993–1022.
- Buntine, W. (1994). “Operations for learning with graphical models.” *Journal of Artificial Intelligence Research*, 2:159–225.
- Geman, S. and D. Geman. (1984). “Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741.
- Ghahramani, Z. and T. L. Griffiths. (2006). “Infinite latent feature models and the Indian buffet process.” In *Advances in Neural Information Processing Systems 18 (NIPS)* (Y. Weiss, B. Schölkopf, and J. C. Platt, eds.), 475–482, MIT Press, Cambridge, MA.
- Gilks, W. R., S. Richardson, and D. J. Spiegelhalter. (1996). *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC, Boca Raton, FL.
- Gonzalez, J. E., Y. Low, and C. Guestrin. (2009). “Residual splash for optimally parallelizing belief propagation.” In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 177–184, Clearwater Beach, FL.
- Hastings, W. K. (1970). “Monte Carlo sampling methods using Markov chains and their applications.” *Biometrika*, 57(1):97–109.
- Hofmann, T. (2001). “Unsupervised learning by probabilistic latent semantic analysis.” *Machine Learning*, 42(1):177–196.
- Jordan, M. I., ed. (1998). *Learning in Graphical Models*. Kluwer, Dordrecht, The Netherlands.
- Koller, D. and N. Friedman. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, MA.
- Kschischang, F. R., B. J. Frey, and H.-A. Loeliger. (2001). “Factor graphs and the sum-product algorithm.” *IEEE Transactions on Information Theory*, 47

- (2):498–519.
- Lafferty, J. D., A. McCallum, and F. C. N. Pereira. (2001). “Conditional random fields: Probabilistic models for segmenting and labeling sequence data.” In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, 282–289, Williamstown, MA.
- Lauritzen, S. L. and D. J. Spiegelhalter. (1988). “Local computations with probabilities on graphical structures and their application to expert systems.” *Journal of the Royal Statistical Society - Series B*, 50(2):157–224.
- Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. (1953). “Equations of state calculations by fast computing machines.” *Journal of Chemical Physics*, 21(6):1087–1092.
- Mooij, J. M. and H. J. Kappen. (2007). “Sufficient conditions for convergence of the sum-product algorithm.” *IEEE Transactions on Information Theory*, 53(12):4422–4437.
- Murphy, K. P., Y. Weiss, and M. I. Jordan. (1999). “Loopy belief propagation for approximate inference: An empirical study.” In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, 467–475, Stockholm, Sweden.
- Neal, R. M. (1993). “Probabilistic inference using Markov chain Monte Carlo methods.” Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto.
- Pearl, J. (1982). “Asymptotic properties of minimax trees and game-searching procedures.” In *Proceedings of the 2nd National Conference on Artificial Intelligence (AAAI)*, Pittsburgh, PA.
- Pearl, J. (1986). “Fusion, propagation and structuring in belief networks.” *Artificial Intelligence*, 29(3):241–288.
- Pearl, J. (1987). “Evidential reasoning using stochastic simulation of causal models.” *Artificial Intelligence*, 32(2):245–258.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA.
- Rabiner, L. R. (1989). “A tutorial on hidden Markov model and selected applications in speech recognition.” *Proceedings of the IEEE*, 77(2):257–286.

- Sutton, C. and A. McCallum. (2012). “An introduction to conditional random fields.” *Foundations and Trends in Machine Learning*, 4(4):267–373.
- Teh, Y. W., M. I. Jordan, M. J. Beal, and D. M. Blei. (2006). “Hierarchical Dirichlet processes.” *Journal of the American Statistical Association*, 101(476):1566–1581.
- Wainwright, M. J. and M. I. Jordan. (2008). “Graphical models, exponential families, and variational inference.” *Foundations and Trends in Machine Learning*, 1(1-2):1–305.
- Weiss, Y. (2000). “Correctness of local probability propagation in graphical models with loops.” *Neural Computation*, 12(1):1–41.

## 休息一会儿

### 小故事：概率图模型奠基人朱迪亚·珀尔

说起概率图模型，就必然要谈到犹太裔美国计算机科学家朱迪亚·珀尔 (Judea Pearl, 1936— )。珀尔出生于特拉维夫，1960 年他在以色列理工学院电子工程本科毕业后来到美国，在 Rutgers 大学和布鲁克林理工学院分别获得物理学硕士和电子工程博士学位。1965 年博士毕业后进入 RCA 研究实验室从事超导存储方面的工作，1970 年到加州大学洛杉矶分校任教至今。



参阅 1.5 节。

艾伦·纽厄尔奖是奖励那些拓宽了计算机科学，或架设了计算机科学与其他学科桥梁的卓越科学家，该奖以图灵奖得主、人工智能先驱 Allen Newell (1927–1992) 命名。机器学习界的另一位著名学者 Michael Jordan 在 2009 年获该奖。

早期的主流人工智能研究专注于以逻辑为基础来进行形式化和推理，但这样很难定量地对不确定性事件进行表达和处理。珀尔在二十世纪七十年代将概率方法引入人工智能，开创了贝叶斯网的研究，提出了信念传播算法，催生了概率图模型这一大类技术，他还以贝叶斯网为工具开创了因果推理方面的研究。由于对人工智能中概率与因果推理的重大贡献，他获得 2011 年图灵奖，此前 he 已获 ACM 与 AAAI 联合颁发的 2003 年艾伦·纽厄尔奖。ACM 评价珀尔在人工智能领域的贡献已扩展到诸多学科领域，“使统计学、心理学、医学以及社会科学中因果性的理解产生了革命性的变化”。2011 年珀尔还获得科学哲学领域最高奖拉卡托斯奖。

珀尔之子丹尼尔是《华尔街日报》驻南亚记者，“9·11”事件后他在巴基斯坦追踪报道激进武装组织时被绑架审讯并残忍地斩首，此事震惊世界。珀尔此后筹办了丹尼尔·珀尔基金会，并参与了很多致力于促进世界民族和平共处的活动。



# 第 15 章 规则学习

## 15.1 基本概念

所有预测模型在广义上都可称为一个或一组“规则”，但规则学习中的“规则”是狭义的，事实上约定俗成地省略了“逻辑”二字。

机器学习中的“规则”(rule)通常是指语义明确、能描述数据分布所隐含的客观规律或领域概念、可写成“若……，则……”形式的逻辑规则[Fürnkranz et al., 2012]。“规则学习”(rule learning)是从训练数据中学习出一组能用于对未见示例进行判别的规则。

形式化地看，一条规则形如：

$$\oplus \leftarrow f_1 \wedge f_2 \wedge \cdots \wedge f_L , \quad (15.1)$$

在数理逻辑中“文字”专指原子公式(atom)及其否定。

其中逻辑蕴含符号“ $\leftarrow$ ”右边部分称为“规则体”(body)，表示该条规则的前提，左边部分称为“规则头”(head)，表示该条规则的结果。规则体是由逻辑文字(literal)  $f_k$  组成的合取式(conjunction)，其中合取符号“ $\wedge$ ”用来表示“并且”。每个文字  $f_k$  都是对示例属性进行检验的布尔表达式，例如“(色泽=乌黑)”或“ $\neg$ (根蒂=硬挺)”。 $L$  是规则体中逻辑文字的个数，称为规则的长度。规则头的“ $\oplus$ ”同样是逻辑文字，一般用来表示规则所判定的目标类别或概念，例如“好瓜”。这样的逻辑规则也被称为“if-then 规则”。

与神经网络、支持向量机这样的“黑箱模型”相比，规则学习具有更好的可解释性，能使用户更直观地对判别过程有所了解。另一方面，数理逻辑具有极强的表达能力，绝大多数人类知识都能通过数理逻辑进行简洁的刻画和表达。例如“父亲的父亲是爷爷”这样的知识不易用函数式描述，而用一阶逻辑则可方便地写为“ $\text{爷爷}(X, Y) \leftarrow \text{父亲}(X, Z) \wedge \text{父亲}(Z, Y)$ ”，因此，规则学习能更自然地在学习过程中引入领域知识。此外，逻辑规则的抽象描述能力在处理一些高度复杂的 AI 任务时具有显著的优势，例如在问答系统中有时可能遇到非常多、甚至无穷种可能的答案，此时若能基于逻辑规则进行抽象表述或者推理，则将带来极大的便利。

假定我们从西瓜数据集学得规则集合  $\mathcal{R}$ ：

规则1：好瓜  $\leftarrow (\text{根蒂} = \text{蜷缩}) \wedge (\text{脐部} = \text{凹陷})$ ；

规则2:  $\neg \text{好瓜} \leftarrow (\text{纹理} = \text{模糊})$ .

西瓜数据集 2.0 见 p.76  
表 4.1.

规则 1 的长度为 2, 它通过判断两个逻辑文字的赋值(valuation)来对示例进行判别. 符合该规则的样本(例如西瓜数据集 2.0 中的样本 1)称为被该规则“覆盖”(cover). 需注意的是, 被规则 1 覆盖的样本是好瓜, 但没被规则 1 覆盖的未必不是好瓜; 只有被规则 2 这样以“ $\neg \text{好瓜}$ ”为头的规则覆盖的才不是好瓜.

集成学习参见第 8 章.

显然, 规则集合中的每条规则都可看作一个子模型, 规则集合是这些子模型的一个集成. 当同一个示例被判别结果不同的多条规则覆盖时, 称发生了“冲突”(conflict), 解决冲突的办法称为“冲突消解”(conflict resolution). 常用的冲突消解策略有投票法、排序法、元规则法等. 投票法是将判别相同的规则数最多的结果作为最终结果. 排序法是在规则集合上定义一个顺序, 在发生冲突时使用排序最前的规则; 相应的规则学习过程称为“带序规则”(ordered rule)学习或“优先级规则”(priority rule)学习. 元规则法是根据领域知识事先设定一些“元规则”(meta-rule), 即关于规则的规则, 例如“发生冲突时使用长度最小的规则”, 然后根据元规则的指导来使用规则集.

亦称“缺省规则”, 可认为是一种特殊的元规则.

此外, 从训练集学得的规则集合也许不能覆盖所有可能的未见示例, 例如前述规则集合  $\mathcal{R}$  无法对“根蒂=蜷缩”、“脐部=稍凹”且“纹理=清晰”的示例进行判别; 这种情况在属性数目很多时常出现. 因此, 规则学习算法通常会设置一条“默认规则”(default rule), 由它来处理规则集合未覆盖的样本; 例如为  $\mathcal{R}$  增加一条默认规则: “未被规则 1, 2 覆盖的都不是好瓜”.

从形式语言表达能力而言, 规则可分为两类: “命题规则”(propositional rule)和“一阶规则”(first-order rule). 前者是由“原子命题”(propositional atom)和逻辑连接词“与”( $\wedge$ )、“或”( $\vee$ )、“非”( $\neg$ )和“蕴含”( $\leftarrow$ )构成的简单陈述句; 例如规则集  $\mathcal{R}$  就是一个命题规则集, “根蒂=蜷缩”“脐部=凹陷”都是原子命题. 后者的基本成分是能描述事物的属性或关系的“原子公式”(atomic formula), 例如表达父子关系的谓词(predicate)“父亲( $X, Y$ )”就是原子公式, 再如表示加一操作“ $\sigma(X) = X + 1$ ”的函数“ $\sigma(X)$ ”也是原子公式. 如果进一步用谓词“自然数( $X$ )”表示  $X$  是自然数, “ $\forall X$ ”表示“对于任意  $X$  成立”, “ $\exists Y$ ”表示“存在  $Y$  使之成立”, 那么“所有自然数加 1 都是自然数”就可写作“ $\forall X \exists Y (\text{自然数}(Y) \leftarrow \text{自然数}(X) \wedge (Y = \sigma(X)))$ ”, 或更简洁的“ $\forall X (\text{自然数}(\sigma(X)) \leftarrow \text{自然数}(X))$ ”. 这样的规则就是一阶规则, 其中  $X$  和  $Y$  称为逻辑变量, “ $\forall$ ”“ $\exists$ ”分别表示“任意”和“存在”, 用于限定变量的取值范围, 称为“量词”(quantifier). 显然, 一阶规则能表达复杂的关

系, 因此也被称为“关系型规则”(relational rule). 以西瓜数据为例, 若我们简单地把属性当作谓词来定义示例与属性值之间的关系, 则命题规则集  $\mathcal{R}$  可改写为一阶规则集  $\mathcal{R}'$ :

规则 1: 好瓜( $X$ )  $\leftarrow$  根蒂( $X$ , 蟾缩)  $\wedge$  脐部( $X$ , 凹陷);

规则 2:  $\neg$  好瓜( $X$ )  $\leftarrow$  纹理( $X$ , 模糊).

显然, 从形式语言系统的角度来看, 命题规则是一阶规则的特例, 因此一阶规则的学习比命题规则要复杂得多.

## 15.2 序贯覆盖

规则学习的目标是产生一个能覆盖尽可能多的样例的规则集. 最直接的做法是“序贯覆盖”(sequential covering), 即逐条归纳: 在训练集上每学到一条规则, 就将该规则覆盖的训练样例去除, 然后以剩下的训练样例组成训练集重复上述过程. 由于每次只处理一部分数据, 因此也被称为“分治”(separate-and-conquer)策略.

我们以命题规则学习为例来考察序贯覆盖法. 命题规则的规则体是对样例属性值进行评估的布尔函数, 如“色泽=青绿”“含糖率  $\leq 0.2$ ”等, 规则头是样例类别. 序贯覆盖法的关键是如何从训练集学出单条规则. 显然, 对规则学习目标  $\oplus$ , 产生一条规则就是寻找最优的一组逻辑文字来构成规则体, 这是一个搜索问题. 形式化地说, 给定正例集合与反例集合, 学习任务是基于候选文字集合  $\mathcal{F} = \{\mathbf{f}_k\}$  来生成最优规则  $\mathbf{r}$ . 在命题规则学习中, 候选文字是形如 “ $R(\text{属性}_i, \text{属性值}_{i,j})$ ” 的布尔表达式, 其中  $\text{属性}_i$  表示样例第  $i$  个属性,  $\text{属性值}_{i,j}$  表示属性  $i$  的第  $j$  个候选值,  $R(x, y)$  则是判断  $x, y$  是否满足关系  $R$  的二元布尔函数.

最简单的做法是从空规则 “ $\oplus \leftarrow$ ” 开始, 将正例类别作为规则头, 再逐个遍历训练集中的每个属性及取值, 尝试将其作为逻辑文字增加到规则体中, 若能使当前规则体仅覆盖正例, 则由此产生一条规则, 然后去除已被覆盖的正例并基于剩余样本尝试生成下一条规则.

p.80 表 4.2 上半部分.

以西瓜数据集 2.0 训练集为例, 首先根据第 1 个样例生成文字“好瓜”和“色泽=青绿”加入规则, 得到

好瓜  $\leftarrow$  (色泽=青绿).

这条规则覆盖样例 1, 6, 10 和 17, 其中有两个正例和两个反例, 不符合“当前规则仅覆盖正例”的条件。于是, 我们尝试将该命题替换为基于属性“色泽”形成的其他原子命题, 例如“色泽=乌黑”; 然而在这个数据集上, 这样的操作不能产生符合条件的规则。于是我们回到“色泽=青绿”, 尝试增加一个基于其他属性的原子命题, 例如“根蒂=蜷缩”:

好瓜  $\leftarrow$  (色泽=青绿)  $\wedge$  (根蒂=蜷缩).

该规则仍覆盖了反例 17。于是我们将第二个命题替换为基于该属性形成的其他原子命题, 例如“根蒂=稍蜷”:

好瓜  $\leftarrow$  (色泽=青绿)  $\wedge$  (根蒂=稍蜷).

这条规则不覆盖任何反例, 虽然它仅覆盖一个正例, 但已满足“当前规则仅覆盖正例”的条件。因此我们保留这条规则并去除它覆盖的样例 6, 然后将剩下的 9 个样例用作训练集。如此继续, 我们将得到:

规则 1: 好瓜  $\leftarrow$  (色泽=青绿)  $\wedge$  (根蒂=稍蜷);

规则 2: 好瓜  $\leftarrow$  (色泽=青绿)  $\wedge$  (敲声=浊响);

规则 3: 好瓜  $\leftarrow$  (色泽=乌黑)  $\wedge$  (根蒂=蜷缩);

规则 4: 好瓜  $\leftarrow$  (色泽=乌黑)  $\wedge$  (纹理=稍糊).

这个规则集覆盖了所有正例, 未覆盖任何反例, 这就是序贯覆盖法得的结果。

上面这种基于穷尽搜索的做法在属性和候选值较多时会由于组合爆炸而不可行。现实任务中一般有两种策略来产生规则: 第一种是“自顶向下”(top-down), 即从比较一般的规则开始, 逐渐添加新文字以缩小规则覆盖范围, 直到满足预定条件为止; 亦称为“生成-测试”(generate-then-test)法, 是规则逐渐“特化”(specialization)的过程。第二种策略是“自底向上”(bottom-up), 即从比较特殊的规则开始, 逐渐删除文字以扩大规则覆盖范围, 直到满足条件为止; 亦称为“数据驱动”(data-driven)法, 是规则逐渐“泛化”(generalization)的过程。第一种策略是覆盖范围从大往小搜索规则, 第二种策略则相反; 前者通常更容易产生泛化性能较好的规则, 而后者则更适合于训练样本较少的情形, 此外, 前者对噪声的鲁棒性比后者要强得多。因此, 在命题规则学习中通常使用第一种策略, 而第二种策略在一阶规则学习这类假设空

例如不含任何属性的空规则, 它覆盖所有样例, 就是一条比较一般的规则。

例如直接以某样例的属性取值形成规则, 该规则仅覆盖此样例, 就是一条比较特殊的规则。

间非常复杂的任务上使用较多.

下面以西瓜数据集 2.0 训练集为例来展示自顶向下的规则生成方法. 首先从空规则 “好瓜  $\leftarrow$ ” 开始, 逐一将 “属性=取值” 作为原子命题加入空规则进行考察. 假定基于训练集准确率来评估规则的优劣,  $n/m$  表示加入某命题后新规则在训练集上的准确率, 其中  $m$  为覆盖的样例总数,  $n$  为覆盖的正例数. 如图 15.1 所示, 经过第一轮评估, “色泽=乌黑” 和 “脐部=凹陷” 都达到了最高准确率  $3/4$ .



西瓜数据集 2.0 训练集  
见 p.80 表 4.2 上半部分.

图 15.1 在西瓜数据集 2.0 训练集上 “自顶向下” 生成单条规则

将属性次序最靠前的逻辑文字 “色泽=乌黑” 加入空规则, 得到

好瓜  $\leftarrow$  (色泽 = 乌黑).

然后, 对上面这条规则覆盖的样例, 通过第二轮评估可发现, 将图 15.1 中的五个逻辑文字加入规则后都能达到 100% 准确率, 我们将覆盖样例最多、且属性次序最靠前的逻辑文字 “根蒂=蟠缩” 加入规则, 于是得到结果

好瓜  $\leftarrow$  (色泽 = 乌黑)  $\wedge$  (根蒂 = 蟠缩).

规则生成过程中涉及一个评估规则优劣的标准, 在上面的例子中使用的标准是: 先考虑规则准确率, 准确率相同时考虑覆盖样例数, 再相同时考虑属性次序. 现实应用中可根据具体任务情况设计适当的标准.

此外, 在上面的例子中每次仅考虑一个“最优”文字, 这通常过于贪心, 易陷入局部最优。为缓解这个问题, 可采用一些相对温和的做法, 例如采用“集束搜索”(beam search), 即每轮保留最优的  $b$  个逻辑文字, 在下一轮均用于构建候选集, 再把候选集中最优的  $b$  个留待再下一轮使用。图 15.1 中若采用  $b=2$  的集束搜索, 则第一轮将保留准确率为  $3/4$  的两个逻辑文字, 在第二轮评估后就能获得下面这条规则, 其准确率仍为 100%, 但是覆盖了 3 个正例:

$$\text{好瓜} \leftarrow (\text{脐部}=\text{凹陷}) \wedge (\text{根蒂}=\text{蜷缩}).$$

由于序贯覆盖法简单有效, 几乎所有规则学习算法都以它为基本框架。它能方便地推广到多分类问题上, 只需将每类分别处理即可: 当学习关于第  $c$  类的规则时, 将所有属于类别  $c$  的样本作为正例, 其他类别的样本作为反例。

### 15.3 剪枝优化

规则生成本质上是一个贪心搜索过程, 需有一定的机制来缓解过拟合的风险, 最常见的做法是剪枝(pruning)。与决策树相似, 剪枝可发生在规则生长过程中, 即“预剪枝”, 也可发生在规则产生后, 即“后剪枝”。通常是基于某种性能度量指标来评估增/删逻辑文字前后的规则性能, 或增/删规则前后的规则集性能, 从而判断是否要进行剪枝。

统计显著性检验参见  
决策树剪枝参见 4.3 节。  
2.4 节。

剪枝还可借助统计显著性检验来进行。例如 CN2 算法 [Clark and Niblett, 1989] 在预剪枝时, 假设用规则集进行预测必须显著优于直接基于训练样例集后验概率分布进行预测。为便于计算, CN2 使用了似然率统计量(Likelihood Ratio Statistics, 简称 LRS)。令  $m_+$ ,  $m_-$  分别表示训练样例集中的正、反例数目,  $\hat{m}_+$ ,  $\hat{m}_-$  分别表示规则(集)覆盖的正、反例数目, 则有

$$\text{LRS} = 2 \cdot \left( \hat{m}_+ \log_2 \frac{\left( \frac{\hat{m}_+}{\hat{m}_+ + \hat{m}_-} \right)}{\left( \frac{m_+}{m_+ + m_-} \right)} + \hat{m}_- \log_2 \frac{\left( \frac{\hat{m}_-}{\hat{m}_+ + \hat{m}_-} \right)}{\left( \frac{m_-}{m_+ + m_-} \right)} \right), \quad (15.2)$$

这实际上是一种信息量指标, 衡量了规则(集)覆盖样例的分布与训练集经验分布的差别: LRS 越大, 说明采用规则(集)进行预测与直接使用训练集正、反例比率进行猜测的差别越大; LRS 越小, 说明规则(集)的效果越可能仅是偶然现象。在数据量比较大的现实任务中, 通常设置为在 LRS 很大(例如 0.99)时 CN2 算法才停止规则(集)生长。

规则学习中常称为“生长集”(growing set)和“剪枝集”(pruning set).

后剪枝最常用的策略是“减错剪枝”(Reduced Error Pruning, 简称 REP) [Brunk and Pazzani, 1991], 其基本做法是: 将样例集划分为训练集和验证集, 从训练集上学得规则集  $\mathcal{R}$  后进行多轮剪枝, 在每一轮穷举所有可能的剪枝操作, 包括删除规则中某个文字、删除规则结尾文字、删除规则尾部多个文字、删除整条规则等, 然后用验证集对剪枝产生的所有候选规则集进行评估, 保留最好的那个规则集进行下一轮剪枝, 如此继续, 直到无法通过剪枝提高验证集上的性能为止.

REP 剪枝通常很有效 [Brunk and Pazzani, 1991], 但其复杂度是  $O(m^4)$ ,  $m$  为训练样例数目. IREP (Incremental REP) [Fürnkranz and Widmer, 1994] 将复杂度降到  $O(m \log^2 m)$ , 其做法是: 在生成每条规则前, 先将当前样例集划分为训练集和验证集, 在训练集上生成一条规则  $r$ , 立即在验证集上对其进行 REP 剪枝, 得到规则  $r'$ ; 将  $r'$  覆盖的样例去除, 在更新后的样例集上重复上述过程. 显然, REP 是针对规则集进行剪枝, 而 IREP 仅对单条规则进行剪枝, 因此后者比前者更高效.

若将剪枝机制与其他一些后处理手段结合起来对规则集进行优化, 则往往能获得更好的效果. 以著名的规则学习算法 RIPPER [Cohen, 1995] 为例, 其泛化性能超过很多决策树算法, 而且学习速度也比大多数决策树算法更快, 奥妙就在于将剪枝与后处理优化相结合.

RIPPER 算法描述如图 15.2 所示. 它先使用 IREP\* 剪枝机制生成规则集  $\mathcal{R}$ . IREP\* [Cohen, 1995] 是 IREP 的改进, 主要是以  $\frac{\hat{m}_+ + (m_- - \hat{m}_-)}{m_+ + m_-}$  取代了 IREP 使用的准确率作为规则性能度量指标, 在剪枝时删除规则尾部的多个文字, 并在最终得到规则集之后再进行一次 IREP 剪枝. RIPPER 中的后处理机

RIPPER 全称 Repeated Incremental Pruning to Produce Error Reduction, WEKA 中的实现称为 JRIP.

图 15.2 中重复次数取值  $k$  时亦称 RIPPER $k$ , 例如 RIPPER5 意味着  $k = 5$ .

基于 IREP\* 生成规则集.

后处理.

去除已被覆盖的样例.

输入: 训练样例集  $D$ ;

重复次数  $k$ .

过程:

- 1:  $\mathcal{R} = \text{IREP}^*(D);$
- 2:  $i = 0;$
- 3: **repeat**
- 4:    $\mathcal{R}' = \text{PostOpt}(\mathcal{R});$
- 5:    $D_i = \text{NotCovered}(\mathcal{R}', D);$
- 6:    $\mathcal{R}_i = \text{IREP}^*(D_i);$
- 7:    $\mathcal{R} = \mathcal{R}' \cup \mathcal{R}_i;$
- 8:    $i = i + 1;$
- 9: **until**  $i = k$

输出: 规则集  $\mathcal{R}$

图 15.2 RIPPER 算法

制是为了在剪枝的基础上进一步提升性能。对  $\mathcal{R}$  中的每条规则  $r_i$ , RIPPER 为它产生两个变体:

- $r'_i$ : 基于  $r_i$  覆盖的样例, 用 IREP\* 重新生成一条规则  $r'_i$ , 该规则称为替换规则(replacement rule);
- $r''_i$ : 对  $r_i$  增加文字进行特化, 然后再用 IREP\* 剪枝生成一条规则  $r''_i$ , 该规则称为修订规则(revised rule).

接下来, 把  $r'_i$  和  $r''_i$  分别与  $\mathcal{R}$  中除  $r_i$  之外的规则放在一起, 组成规则集  $\mathcal{R}'$  和  $\mathcal{R}''$ , 将它们与  $\mathcal{R}$  一起进行比较, 选择最优的规则集保留下来。这就是图 15.2 中算法第 4 行所做的操作。

为什么 RIPPER 的优化策略会有效呢? 原因很简单: 最初生成  $\mathcal{R}$  的时候, 规则是按序生成的, 每条规则都没有对其后产生的规则加以考虑, 这样的贪心算法本质常导致算法陷入局部最优; RIPPER 的后处理优化过程将  $\mathcal{R}$  中的所有规则放在一起重新加以优化, 恰是通过全局的考虑来缓解贪心算法的局部性, 从而往往能得到更好的效果 [Fürnkranz et al., 2012].

## 15.4 一阶规则学习

受限于命题逻辑表达能力, 命题规则学习难以处理对象之间的“关系”(relation), 而关系信息在很多任务中非常重要。例如, 我们在现实世界挑选西瓜时, 通常很难把水果摊上所有西瓜的特征用属性值描述出来, 因为我们很难判断: 色泽看起来多深才叫“色泽青绿”? 敲起来声音多低才叫“敲声沉闷”? 比较现实的做法是将西瓜进行相互比较, 例如, “瓜 1 的颜色比瓜 2 更深, 并且瓜 1 的根蒂比瓜 2 更蜷”, 因此“瓜 1 比瓜 2 更好”。然而, 这已超越了命题逻辑的表达能力, 需用一阶逻辑表示, 并且要使用一阶规则学习。

对西瓜数据, 我们不妨定义:

- 色泽深度: 乌黑 > 青绿 > 浅白;
- 根蒂蜷度: 蜷缩 > 稍蜷 > 硬挺;
- 敲声沉度: 沉闷 > 浊响 > 清脆;
- 纹理清晰度: 清晰 > 稍糊 > 模糊;
- 脐部凹陷度: 凹陷 > 稍凹 > 平坦;
- 触感硬度: 硬滑 > 软粘.

括号内数字对应于 p.80  
表 4.2 中的样例编号.

分隔线上半部分为背景  
知识, 下半部分为样例.

这样的规则亦称为一阶  
逻辑子句(clause).

表 15.1 西瓜数据集 5.0

色泽更深(2, 1)	色泽更深(2, 6)	色泽更深(2, 10)	色泽更深(2, 14)
色泽更深(2, 16)	色泽更深(2, 17)	色泽更深(3, 1)	色泽更深(3, 6)
...	...	...	...
色泽更深(15, 16)	色泽更深(15, 17)	色泽更深(17, 14)	色泽更深(17, 16)
根蒂更蜷(1, 6)	根蒂更蜷(1, 7)	根蒂更蜷(1, 10)	根蒂更蜷(1, 14)
...	...	...	...
根蒂更蜷(17, 7)	根蒂更蜷(17, 10)	根蒂更蜷(17, 14)	根蒂更蜷(17, 15)
敲声更沉(2, 1)	敲声更沉(2, 3)	敲声更沉(2, 6)	敲声更沉(2, 7)
...	...	...	...
敲声更沉(17, 7)	敲声更沉(17, 10)	敲声更沉(17, 15)	敲声更沉(17, 16)
纹理更清(1, 7)	纹理更清(1, 14)	纹理更清(1, 16)	纹理更清(1, 17)
...	...	...	...
纹理更清(15, 14)	纹理更清(15, 16)	纹理更清(15, 17)	纹理更清(17, 16)
脐部更凹(1, 6)	脐部更凹(1, 7)	脐部更凹(1, 10)	脐部更凹(1, 15)
...	...	...	...
脐部更凹(15, 10)	脐部更凹(15, 16)	脐部更凹(17, 10)	脐部更凹(17, 16)
触感更硬(1, 6)	触感更硬(1, 7)	触感更硬(1, 10)	触感更硬(1, 15)
...	...	...	...
触感更硬(17, 6)	触感更硬(17, 7)	触感更硬(17, 10)	触感更硬(17, 15)
更好(1, 10)	更好(1, 14)	更好(1, 15)	更好(1, 16)
...	...	...	...
更好(7, 14)	更好(7, 15)	更好(7, 16)	更好(7, 17)
¬更好(10, 1)	¬更好(10, 2)	¬更好(10, 3)	¬更好(10, 6)
...	...	...	...
¬更好(17, 2)	¬更好(17, 3)	¬更好(17, 6)	¬更好(17, 7)

于是, 西瓜数据集 2.0 训练集就转化为表 15.1 的西瓜数据集 5.0. 这样的数据直接描述了样例间的关系, 称为“关系数据”(relational data), 其中由原样本属性转化而来的“色泽更深”“根蒂更蜷”等原子公式称为“背景知识”(background knowledge), 而由样本类别转化而来的关于“更好”“¬更好”的原子公式称为关系数据样例(examples). 从西瓜数据集 5.0 可学出这样的一阶规则:

$$(\forall X, \forall Y)(\text{更好}(X, Y) \leftarrow \text{根蒂更蜷}(X, Y) \wedge \text{脐部更凹}(X, Y)).$$

显然, 一阶规则仍是式(15.1)的形式, 但其规则头、规则体都是一阶逻辑表达式, “更好(·, ·)”、“根蒂更蜷(·, ·)”、“脐部更凹(·, ·)”是关系描述所对应的谓词, 个体对象“瓜 1”、“瓜 2”被逻辑变量“X”、“Y”替换. 全称量词“ $\forall$ ”表示该规则对所有个体对象都成立; 通常, 在一阶规则中所有出现的变量都被全称量词限定, 因此下面我们在不影响理解的情况下将省略量词部分.

一阶规则有强大的表达能力, 例如它能简洁地表达递归概念, 如

$$\text{更好}(X, Y) \leftarrow \text{更好}(X, Z) \wedge \text{更好}(Z, Y).$$

统计学习一般是基于“属性-值”表示, 这与命题逻辑表示等价; 此类学习可统称为“基于命题表示的学习”。

一阶规则学习能容易地引入领域知识, 这是它相对于命题规则学习的另一大优势。在命题规则学习乃至一般的统计学习中, 若欲引入领域知识, 通常有两种做法: 在现有属性的基础上基于领域知识构造出新属性, 或基于领域知识设计某种函数机制(例如正则化)来对假设空间加以约束。然而, 现实任务中并非所有的领域知识都能容易地通过属性重构和函数约束来表达。例如, 假定获得了包含某未知元素的化合物  $X$ , 欲通过试验来发现它与已知化合物  $Y$  的反应方程式。我们可多次重复试验, 测出每次结果中化合物的组分含量。虽然我们对反应中的未知元素性质一无所知, 但知道一些普遍成立的化学原理, 例如金属原子一般产生离子键、氢原子之间一般都是共价键等, 并且也了解已知元素间可能发生的反应。有了这些领域知识, 重复几次试验后就不难学出  $X$  和  $Y$  的反应方程式, 还可能推测出  $X$  的性质、甚至发现新的分子和元素。类似这样的领域知识充斥在日常生活与各类任务中, 但在基于命题表示的学习中加以利用却非常困难。

FOIL (First-Order Inductive Learner) [Quinlan, 1990] 是著名的一阶规则学习算法, 它遵循序贯覆盖框架且采用自顶向下的规则归纳策略, 与 15.2 节中的命题规则学习过程很相似。但由于逻辑变量的存在, FOIL 在规则生成时需考虑不同的变量组合。例如在西瓜数据集 5.0 上, 对“更好( $X, Y$ )”这个概念, 最初的空规则是

$$\text{更好}(X, Y) \leftarrow .$$

接下来要考虑数据中所有其他谓词以及各种变量搭配作为候选文字。新加入的文字应包含至少一个已出现的变量, 否则没有任何实质意义。在这个例子中考虑下列候选文字:

色泽更深( $X, Y$ ), 色泽更深( $Y, X$ ), 色泽更深( $X, Z$ ), 色泽更深( $Z, X$ ),

色泽更深( $Y, Z$ ), 色泽更深( $Z, Y$ ), 色泽更深( $X, X$ ), 色泽更深( $Y, Y$ ),

根蒂更蜷( $X, Y$ ),

敲声更沉( $X, Y$ ),

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

FOIL 使用“FOIL 增益”(FOIL gain)来选择文字:

$$F\_Gain = \hat{m}_+ \times \left( \log_2 \frac{\hat{m}_+}{\hat{m}_+ + \hat{m}_-} - \log_2 \frac{m_+}{m_+ + m_-} \right), \quad (15.3)$$

决策树的信息增益参见  
4.2.1 节.

这实质上与类别不平衡性有关, 参见 3.6 节.

其中,  $\hat{m}_+$ ,  $\hat{m}_-$  分别为增加候选文字后新规则所覆盖的正、反例数;  $m_+$ ,  $m_-$  为原规则覆盖的正、反例数. FOIL 增益与决策树使用的信息增益不同, 它仅考虑正例的信息量, 并且用新规则覆盖的正例数作为权重. 这是由于关系数据中正例数往往远少于反例数, 因此通常对正例应赋予更多的关注.

在西瓜数据集 5.0 的例子中, 只需给初始的空规则体加入“色泽更深( $X, Y$ )”或“脐部更凹( $X, Y$ )”, 新规则就能覆盖 16 个正例和 2 个反例, 所对应的 FOIL 增益为候选最大值  $16 \times (\log_2 \frac{16}{18} - \log_2 \frac{25}{50}) = 13.28$ . 假定前者被选中, 则得到

$$\text{更好}(X, Y) \leftarrow \text{色泽更深}(X, Y).$$

该规则仍覆盖 2 个反例: “更好(15, 1)”与“更好(15, 6)”. 于是, FOIL 像命题规则学习那样继续增加规则体长度, 最终生成合适的单条规则加入规则集. 此后, FOIL 使用后剪枝对规则集进行优化.

若允许将目标谓词作为候选文字加入规则体, 则 FOIL 能学出递归规则; 若允许将否定形式的文字  $\neg f$  作为候选, 则往往能得到更简洁的规则集.

FOIL 可大致看作命题规则学习与归纳逻辑程序设计之间的过渡, 其自顶向下的规则生成过程不能支持函数和逻辑表达式嵌套, 因此规则表达能力仍有不足; 但它是把命题规则学习过程通过变量替换等操作直接转化为一阶规则学习, 因此比一般归纳逻辑程序设计技术更高效.

## 15.5 归纳逻辑程序设计

归纳逻辑程序设计(Inductive Logic Programming, 简称 ILP)在一阶规则学习中引入了函数和逻辑表达式嵌套. 一方面, 这使得机器学习系统具备了更为强大的表达能力; 另一方面, ILP 可看作用机器学习技术来解决基于背景知识的逻辑程序(logic program)归纳, 其学得的“规则”可被 PROLOG 等逻辑程序设计语言直接使用.

然而, 函数和逻辑表达式嵌套的引入也带来了计算上的巨大挑战. 例如, 给定一元谓词  $P$  和一元函数  $f$ , 它们能组成的文字有  $P(X)$ ,  $P(f(X))$ ,

$P(f(f(X)))$  等无穷多个, 这就使得规则学习过程中可能的候选原子公式有无穷多个。若仍采用命题逻辑规则或 FOIL 学习那样自顶向下的规则生成过程, 则在增加规则长度时将因无法列举所有候选文字而失败。实际困难还不止这些, 例如计算 FOIL 增益需对规则覆盖的全部正反例计数, 而在引入函数和逻辑表达式嵌套之后这也变得不可行。

### 15.5.1 最小一般泛化

归纳逻辑程序设计采用自底向上的规则生成策略, 直接将一个或多个正例所对应的具体事实(grounded fact)作为初始规则, 再对规则逐步进行泛化以增加其对样例的覆盖率。泛化操作可以是将规则中的常量替换为逻辑变量, 也可以是删除规则体中的某个文字。

以西瓜数据集 5.0 为例, 为简便起见, 暂且假定“更好( $X, Y$ )”仅决定于  $(X, Y)$  取值相同的关系, 正例“更好(1, 10)”和“更好(1, 15)”所对应的初始规则分别为

$$\begin{aligned} \text{更好}(1, 10) \leftarrow & \text{根蒂更蜷}(1, 10) \wedge \text{声音更沉}(1, 10) \wedge \text{脐部更凹}(1, 10) \\ & \wedge \text{触感更硬}(1, 10); \\ \text{更好}(1, 15) \leftarrow & \text{根蒂更蜷}(1, 15) \wedge \text{脐部更凹}(1, 15) \wedge \text{触感更硬}(1, 15). \end{aligned}$$

显然, 这两条规则只对应了特殊的关系数据样例, 难以具有泛化能力。因此, 我们希望把这样的“特殊”规则转变为更“一般”的规则。为达到这个目的, 最基础的技术是“最小一般泛化”(Least General Generalization, 简称 LGG) [Plotkin, 1970]。

给定一阶公式  $r_1$  和  $r_2$ , LGG 先找出涉及相同谓词的文字, 然后对文字中每个位置的常量逐一进行考察, 若常量在两个文字中相同则保持不变, 记为  $\text{LGG}(t, t) = t$ ; 否则将它们替换为同一个新变量, 并将该替换应用于公式的所有其他位置: 假定这两个不同的常量分别为  $s, t$ , 新变量为  $V$ , 则记为  $\text{LGG}(s, t) = V$ , 并在以后所有出现  $\text{LGG}(s, t)$  的位置用  $V$  来代替。例如对上面例子中的两条规则, 先比较“更好(1, 10)”和“更好(1, 15)”, 由于文字中常量“10” $\neq$ “15”, 因此将它们都替换为  $Y$ , 并在  $r_1$  和  $r_2$  中将其余位置上成对出现的“10”和“15”都替换为  $Y$ , 得到

$$\begin{aligned} \text{更好}(1, Y) \leftarrow & \text{根蒂更蜷}(1, Y) \wedge \text{声音更沉}(1, 10) \wedge \text{脐部更凹}(1, Y) \\ & \wedge \text{触感更硬}(1, Y); \end{aligned}$$

这里的数字是瓜的编号。

更好(1, Y)  $\leftarrow$  根蒂更蜷(1, Y)  $\wedge$  脐部更凹(1, Y)  $\wedge$  触感更硬(1, Y).

然后, LGG 忽略  $r_1$  和  $r_2$  中不含共同谓词的文字, 因为若 LGG 包含某条公式所没有的谓词, 则 LGG 无法特化为那条公式. 容易看出, 在这个例子中需忽略“声音更沉(1, 10)”这个文字, 于是得到的 LGG 为

更好(1, Y)  $\leftarrow$  根蒂更蜷(1, Y)  $\wedge$  脐部更凹(1, Y)  $\wedge$  触感更硬(1, Y). (15.4)

式(15.4)仅能判断瓜 1 是否比其他瓜更好. 为了提升其泛化能力, 假定另有两条关于瓜 2 的初始规则

更好(2, 10)  $\leftarrow$  颜色更深(2, 10)  $\wedge$  根蒂更蜷(2, 10)  $\wedge$  敲声更沉(2, 10)  
 $\wedge$  脐部更凹(2, 10)  $\wedge$  触感更硬(2, 10), (15.5)

于是可求取式(15.4)与(15.5)的 LGG. 注意到文字“更好(2, 10)”和“更好(1, Y)”的对应位置同时出现了常量“10”与变量“Y”, 于是可令  $LGG(10, Y) = Y_2$ , 并将所有“10”与“Y”成对出现的位置均替换为  $Y_2$ . 最后, 令  $LGG(2, 1) = X$  并删去谓词不同的文字, 就得到如下这条不包含常量的一般规则:

更好(X,  $Y_2$ )  $\leftarrow$  根蒂更蜷(X,  $Y_2$ )  $\wedge$  脐部更凹(X,  $Y_2$ )  $\wedge$  触感更硬(X,  $Y_2$ ).

参阅 [Lavrač and Dzeroski, 1993] 第 3 章.

上面的例子中仅考虑了肯定文字, 未使用“ $\neg$ ”符号. 实际上 LGG 还能进行更复杂的泛化操作. 此外, 上面还假定“更好(X, Y)”的初始规则仅包含变量同为(X, Y)的关系, 而背景知识中往往包含其他一些有用的关系, 因此许多 ILP 系统采用了不同的初始规则选择方法. 最常用的是 RLGG (Relative Least Generalization) [Plotkin, 1971], 它在计算 LGG 时考虑所有的背景知识, 将样例 e 的初始规则定义为  $e \leftarrow K$ , 其中 K 是背景知识中所有原子的合取.

容易证明, LGG 是能特化为  $r_1$  和  $r_2$  的所有一阶公式中最特殊的一个: 不存在既能特化为  $r_1$  和  $r_2$ , 也能泛化为它们的 LGG 的一阶公式  $r'$ .

在归纳逻辑程序设计中, 获得 LGG 之后, 可将其看作单条规则加入规则集, 最后再用前几节介绍的技术进一步优化, 例如对规则集进行后剪枝等.

### 15.5.2 逆归结

在逻辑学中, “演绎”(deduction)与“归纳”(induction)是人类认识世界的两种基本方式. 大致来说, 演绎是从一般性规律出发来探讨具体事物, 而归纳

十九世纪英国政治经济学家和哲学家 W. S. Jevons 通过数理方法论证, 最早明确指出归纳是演绎的逆过程.

则是从个别事物出发概括出一般性规律. 一般数学定理证明是演绎实践的代表, 而机器学习显然是属于归纳的范畴. 1965 年, 逻辑学家 J. A. Robinson 提出, 一阶谓词演算中的演绎推理能用一条十分简洁的规则描述, 这就是数理逻辑中著名的归结原理(resolution principle) [Robinson, 1965]. 二十多年后, 计算机科学家 S. Muggleton 和 W. Buntine 针对归纳推理提出了“逆归结”(inverse resolution) [Muggleton and Buntine, 1988], 这对归纳逻辑程序设计的发展起到了重要作用.

基于归结原理, 我们可将貌似复杂的逻辑规则与背景知识联系起来化繁为简; 而基于逆归结, 我们可基于背景知识来发明新的概念和关系. 下面我们先以较为简单的命题演算为例, 来看看归结、逆归结是怎么回事.

假定两个逻辑表达式  $C_1$  和  $C_2$  成立, 且分别包含了互补项  $L_1$  与  $L_2$ ; 不失一般性, 令  $L = L_1 = \neg L_2$ ,  $C_1 = A \vee L$ ,  $C_2 = B \vee \neg L$ . 归结原理告诉我们, 通过演绎推理能消去  $L$  而得到“归结项”  $C = A \vee B$ . 若定义析合范式的删除操作

$$(A \vee B) - \{B\} = A, \quad (15.6)$$

则归结过程可表述为

$$C = (C_1 - \{L\}) \vee (C_2 - \{\neg L\}), \quad (15.7)$$

简记为

$$C = C_1 \cdot C_2. \quad (15.8)$$

图 15.3 给出了归结原理的一个直观例示.

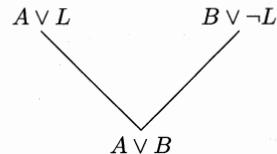


图 15.3 归结原理例示

与上面的过程相反, 逆归结研究的是在已知  $C$  和某个  $C_i$  的情况下如何得到  $C_j (i \neq j)$ . 假定已知  $C$  和  $C_1$  求  $C_2$ , 则由式(15.7), 该过程可表述为

$$C_2 = (C - (C_1 - \{L\})) \vee \{\neg L\}. \quad (15.9)$$

在逻辑推理实践中如何实现逆归结呢? [Muggleton, 1995] 定义了四种完备的逆归结操作。若以规则形式  $p \leftarrow q$  等价地表达  $p \vee \neg q$ , 并假定用小写字母表示逻辑文字、大写字母表示合取式组成的逻辑子句, 则这四种操作是:

$$\text{吸收(absorption)} : \frac{p \leftarrow A \wedge B \quad q \leftarrow A}{p \leftarrow q \wedge B \quad q \leftarrow A} \quad (15.10)$$

$$\text{辨识(identification)} : \frac{p \leftarrow A \wedge B \quad p \leftarrow A \wedge q}{q \leftarrow B \quad p \leftarrow A \wedge q} \quad (15.11)$$

$$\text{内构(intra-construction)} : \frac{p \leftarrow A \wedge B \quad p \leftarrow A \wedge C}{q \leftarrow B \quad p \leftarrow A \wedge q \quad q \leftarrow C} \quad (15.12)$$

$$\text{互构(inter-construction)} : \frac{p \leftarrow A \wedge B \quad q \leftarrow A \wedge C}{p \leftarrow r \wedge B \quad r \leftarrow A \quad q \leftarrow r \wedge C} \quad (15.13)$$

读作“ $X$  推出  $Y$ ”。

这里我们用  $\frac{X}{Y}$  表示  $X$  蕴含  $Y$ , 在数理逻辑里写作  $X \vdash Y$ . 上述规则中,  $X$  的子句或是  $Y$  的归结项, 或是  $Y$  的某个子句的等价项; 而  $Y$  中出现的新逻辑文字则可看作通过归纳学到的新命题.

归结、逆归结都能容易地扩展为一阶逻辑形式; 与命题逻辑的主要不同之处是, 一阶逻辑的归结、逆归结通常需进行合一置换操作.

“置换”(substitution)是用某些项来替换逻辑表达式中的变量. 例如用  $\theta = \{1/X, 2/Y\}$  置换 “ $C = \text{色泽更深}(X, Y) \wedge \text{敲声更沉}(X, Y)$ ” 可得到 “ $C' = C\theta = \text{色泽更深}(1, 2) \wedge \text{敲声更沉}(1, 2)$ ”, 其中  $\{X, Y\}$  称为  $\theta$  的作用域(domain). 与代数中的置换类似, 一阶逻辑中也有“复合置换”和“逆置换”. 例如先用  $\theta = \{Y/X\}$  将  $X$  替换为  $Y$ , 再用  $\lambda = \{1/Y\}$  将  $Y$  替换为 1, 这样的复合操作记为  $\theta \circ \lambda$ ;  $\theta$  的逆置换则记为  $\theta^{-1} = \{X/Y\}$ .

“合一”(unification)是用一种变量置换令两个或多个逻辑表达式相等. 例如对 “ $A = \text{色泽更深}(1, X)$ ” 和 “ $B = \text{色泽更深}(Y, 2)$ ”, 可用  $\theta = \{2/X, 1/Y\}$  使 “ $A\theta = B\theta = \text{色泽更深}(1, 2)$ ”; 此时称  $A$  和  $B$  是“可合一的”(unifiable), 称  $\theta$  为  $A$  和  $B$  的“合一化子”(unifier). 若  $\delta$  是一组一阶逻辑表达式  $W$  的合一化子, 且对  $W$  的任意合一化子  $\theta$  均存在相应的置换  $\lambda$  使  $\theta = \delta \circ \lambda$ , 则称  $\delta$  为  $W$  的“最一般合一置换”或“最一般合一化子”(most general unifier, 简记为 MGU), 这是归纳逻辑程序中最重要的概念之一. 例如 “色泽更深(1,  $Y$ )” 和 “色泽更深( $X$ , 2)” 能被  $\theta_1 = \{1/X\}$ ,  $\theta_2 = \{1/X, 2/Y\}$ ,  $\theta_3 = \{1/Z, Z/X\}$  合一, 但仅有  $\theta_1$  是它们的 MGU.

一阶逻辑进行归结时, 需利用合一操作来搜索互补项  $L_1$  和  $L_2$ . 对两个一阶逻辑表达式  $C_1 = A \vee L_1$  和  $C_2 = B \vee L_2$ , 若存在合一化子  $\theta$  使  $L_1\theta = \neg L_2\theta$ ,

则可对其进行归结:

$$C = (C_1 - \{L_1\})\theta \vee (C_2 - \{L_2\})\theta. \quad (15.14)$$

类似的, 可利用合一化子对式(15.9) 进行扩展得到一阶逻辑的逆归结. 基于式(15.8), 定义  $C_1 = C/C_2$  和  $C_2 = C/C_1$  为“归结商” (resolution quotient), 于是, 逆归结的目标就是在已知  $C$  和  $C_1$  时求出归结商  $C_2$ . 对某个  $L_1 \in C_1$ , 假定  $\phi_1$  是一个置换, 它能使

对  $C = A \vee B$ , 有  $A \vdash C$   
与  $\exists B (C = A \vee B)$  等价.

$$(C_1 - \{L_1\})\phi_1 \vdash C, \quad (15.15)$$

这里  $\phi_1$  的作用域是  $C_1$  中所有变量, 记为  $\text{vars}(C_1)$ , 其作用是使  $C_1 - \{L_1\}$  与  $C$  中的对应文字能合一. 令  $\phi_2$  为作用域是  $\text{vars}(L_1) - \text{vars}(C_1 - \{L_1\})$  的置换,  $L_2$  为归结商  $C_2$  中将被消去的文字,  $\theta_2$  是以  $\text{vars}(L_2)$  为作用域的置换,  $\phi_2$  与  $\phi_1$  共同作用于  $L_1$ , 使得  $\neg L_1 \phi_1 \circ \phi_2 = L_2 \theta_2$ , 于是  $\phi_1 \circ \phi_2 \circ \theta_2$  为  $\neg L_1$  与  $L_2$  的 MGU. 将前两步的复合置换  $\phi_1 \circ \phi_2$  记为  $\theta_1$ , 用  $\theta_2^{-1}$  表示  $\theta_2$  的逆置换, 则有  $(\neg L_1 \theta_1) \theta_2^{-1} = L_2$ . 于是, 类似于式(15.9), 一阶逆归结是

$$C_2 = (C - (C_1 - \{L_1\})\theta_1 \vee \{\neg L_1 \theta_1\})\theta_2^{-1}. \quad (15.16)$$

在一阶情形下  $L_1$ 、 $L_2$ 、 $\theta_1$  和  $\theta_2$  的选择通常都不唯一, 这时需通过一些其他的判断标准来取舍, 例如覆盖率、准确率、信息熵等.

以西瓜数据集 5.0 为例, 假定我们通过一些步骤已得到规则

$$C_1 = \text{更好}(1, X) \leftarrow \text{根蒂更蜷}(1, X) \wedge \text{纹理更清}(1, X);$$

$$C_2 = \text{更好}(1, Y) \leftarrow \text{根蒂更蜷}(1, Y) \wedge \text{敲声更沉}(1, Y).$$

容易看出它们是“ $p \leftarrow A \wedge B$ ”和“ $p \leftarrow A \wedge C$ ”的形式, 于是可使用内构操作式(15.12) 来进行逆归结. 由于  $C_1$ ,  $C_2$  中的谓词都是二元的, 为保持新规则描述信息的完整性, 我们创造一个新的二元谓词  $q(M, N)$ , 并根据式(15.12) 得到

$$C' = \text{更好}(1, Z) \leftarrow \text{根蒂更蜷}(1, Z) \wedge q(M, N),$$

式(15.12) 中横线下方的另两项分别是  $C_1/C'$  和  $C_2/C'$  的归结商. 对  $C_1/C'$ , 容易发现  $C'$  中通过归结消去  $L_1$  的选择可以有“ $\neg \text{根蒂更蜷}(1, Z)$ ”和

奥卡姆剃刀原则参见  
1.4 节。

“ $\neg q(M, N)$ ”。 $q$  是新发明的谓词，迟早需学习一条新规则 “ $q(M, N) \leftarrow ?$ ” 来定义它；根据奥卡姆剃刀原则，同等描述能力下学得的规则越少越好，因此我们将  $\neg q(M, N)$  作为  $L_1$ 。由式(15.16)，存在解： $L_2 = q(1, S)$ ， $\phi_1 = \{X/Z\}$ ， $\phi_2 = \{1/M, X/N\}$ ， $\theta_2 = \{X/S\}$ 。通过简单的演算即可求出归结商为 “ $q(1, S) \leftarrow \text{纹理更清}(1, S)$ ”。类似地可求出  $C_2/C'$  的归结商 “ $q(1, T) \leftarrow \text{敲声更沉}(1, T)$ ”。

逆归结的一大特点是能自动发明新谓词，这些新谓词可能对应于样例属性和背景知识中不存在的新知识，对知识发现与精化有重要意义。但自动发明的新谓词究竟对应于什么语义，例如 “ $q$ ” 意味着 “更新鲜”？“更甜”？“更多日晒”？……这只能通过使用者对任务领域的进一步理解才能明确。

上面的例子中我们只介绍了如何基于两条规则进行逆归结。在现实任务中，ILP 系统通常先自底向上生成一组规则，然后再结合最小一般泛化与逆归结做进一步学习。

## 15.6 阅读材料

规则学习是 “符号主义学习” (symbolism learning)的主要代表，是最早开始研究的机器学习技术之一 [Michalski, 1983]. [Fürnkranz et al., 2012] 对规则学习做了比较全面的总结。

AQ 是 Algorithm Quasi-optimal 的缩写。

序贯覆盖是规则学习的基本框架，最早在 [Michalski, 1969] 的 AQ 中被提出，AQ 后来发展成一个算法族，其中比较著名的有 AQ15 [Michalski et al., 1986]、AQ17-HCI [Wnek and Michalski, 1994] 等。受计算能力的制约，早期 AQ 在学习时只能随机挑选一对正反例作为种子开始训练，样例选择的随机性导致 AQ 学习效果不稳定。PRISM [Cendrowska, 1987] 解决了这个问题，该算法最早采用自顶向下搜索，并显示出规则学习与决策树学习相比的优点：决策树试图将样本空间划分为不重叠的等价类，而规则学习并不强求这一点，因此后者学得的模型能有更低的复杂度。虽然 PRISM 的性能不如 AQ，因此在当时反响不大，但今天来看，它是规则学习领域发展的重要一步。

决策树的每个叶结点对应一个等价类。

WEKA 中有 PRISM 的实现。

RIPPER 达到了比 C4.5 决策树既快又好的效果。

CN2 [Clark and Niblett, 1989] 采用集束搜索，是最早考虑过拟合问题的规则学习算法。[Fürnkranz, 1994] 显示出后剪枝在缓解规则学习过拟合中的优势。RIPPER [Cohen, 1995] 是命题规则学习技术的高峰，它融合了该领域的许多技巧，使规则学习在与决策树学习的长期竞争中首次占据上风，作者主页上的 C 语言 RIPPER 版本至今仍代表着命题规则学习的最高水平。

关系学习的研究一般认为始于 [Winston, 1970]；由于命题规则学习很难完

成此类任务,一阶规则学习开始得以发展. FOIL 通过变量替换等操作把命题规则学习转化为一阶规则学习,该技术至今仍有使用,例如 2010 年卡耐基梅隆大学开展的“永动语言学习”(Never-Ending Language Learning,简称 NELL)计划即采用 FOIL 来学习自然语言中的语义关系 [Carlson et al., 2010]. 很多文献将所有的一阶规则学习方法都划入归纳逻辑程序设计的范畴,本书则是作了更为严格的限定.

[Muggleton, 1991] 提出了“归纳逻辑程序设计”(ILP)这个术语,在 GOLEM [Muggleton and Feng, 1990] 中克服了许多从命题逻辑过渡到一阶逻辑学习的困难,并确立了自底向上归纳的 ILP 框架. 最小一般泛化(LGG)最早由 [Plotkin, 1970] 提出, GOLEM 则使用了 RLGG. PROGOL [Muggleton, 1995] 将逆归结改进为逆蕴含(inverse entailment)并取得了更好效果. 新谓词发明方面近年有一些新进展 [Muggleton and Lin, 2013]. 由于 ILP 学得的规则几乎能直接被 PROLOG 等逻辑程序解释器调用,而 PROLOG 在专家系统中常被使用,因此 ILP 成为连接机器学习与知识工程的重要桥梁. PROGOL [Muggleton, 1995] 和 ALEPH [Srinivasan, 1999] 是应用广泛的 ILP 系统,其基本思想已在本章关于 ILP 的部分有所体现. Datalog [Ceri et al., 1989] 则对数据库领域产生了很大影响,例如甚至影响了 SQL 1999 标准和 IBM DB2. ILP 方面的重要读物有 [Muggleton, 1992; Lavrač and Dzeroski, 1993],并且有专门的国际归纳逻辑程序设计会议(ILP).

ILP 复杂度很高,虽在生物数据挖掘和自然语言处理等任务中取得一些成功 [Bratko and Muggleton, 1995],但问题规模稍大就难以处理,因此,这方面的研究在统计学习兴起后受到一定抑制. 近年来随着机器学习技术进入更多应用领域,在富含结构信息和领域知识的任务中,逻辑表达的重要性逐渐凸显出来,因此出现了一些将规则学习与统计学习相结合的努力,例如试图在归纳逻辑程序设计中引入概率模型的“概率归纳逻辑程序设计”(probabilistic ILP) [De Raedt et al., 2008]、给贝叶斯网中的结点赋予逻辑意义的“关系贝叶斯网”(relational Bayesian network) [Jaeger, 2002] 等. 事实上,将关系学习与统计学习相结合是机器学习发展的一大趋势,而概率归纳逻辑程序设计是其中的重要代表,其他重要代表还有概率关系模型 [Friedman et al., 1999]、贝叶斯逻辑程序(Bayesian Logic Program) [Kersting et al., 2000]、马尔可夫逻辑网(Markov logic network) [Richardson and Domingos, 2006] 等,统称为“统计关系学习”(statistical relational learning) [Getoor and Taskar, 2007].

知识工程与专家系统参见 1.5 节.

## 习题

西瓜数据集 2.0 见 p.76  
表 4.1.

西瓜数据集 2.0 $\alpha$  见 p.86  
表 4.4.

在  $S$  无法合一时输出  
“无解”。

- 15.1** 对西瓜数据集 2.0, 允许使用否定形式的文字, 试基于自顶向下的策略学出命题规则集.
- 15.2** 对西瓜数据集 2.0, 在学习过程中可通过删去文字、将常量替换为变量来进行规则泛化, 试基于自底向上的策略学出命题规则集.
- 15.3** 从网上下载或自己编程实现 RIPPER 算法, 并在西瓜数据集 2.0 上学出规则集.
- 15.4** 规则学习也能对缺失数据进行学习. 试模仿决策树的缺失值处理方法, 基于序贯覆盖在西瓜数据集 2.0 $\alpha$  上学出命题规则集.
- 15.5** 从网上下载或自己编程实现 RIPPER 算法, 允许使用否定形式的文字, 在西瓜数据集 5.0 上学出一阶规则集.
- 15.6** 对西瓜数据集 5.0, 试利用归纳逻辑程序学习概念 “更坏( $X, Y$ )” .
- 15.7** 试证明: 对于一阶公式  $r_1$  和  $r_2$ , 不存在既能特化为  $r_1$  和  $r_2$ 、也能泛化为它们的 LGG 的一阶公式  $r'$ .
- 15.8** 试生成一个西瓜数据集 5.0 的 LGG 集合.
- 15.9\*** 一阶原子公式是一种递归定义的公式, 形如  $P(t_1, t_2, \dots, t_n)$ , 其中  $P$  是谓词或函数符号,  $t_i$  称为 “项”, 可以是逻辑常量、变量或者其他原子公式. 对一阶原子公式  $E_i$  的集合  $S = \{E_1, E_2, \dots, E_n\}$ , 试设计一个算法求解其 MGU.
- 15.10\*** 基于序贯覆盖的规则学习算法在学习下一条规则前, 会将已被当前规则集所覆盖的样例从训练集中删去. 这种贪心策略使得后续学习过程仅需关心以往未覆盖的样例, 在判定规则覆盖率时不需考虑前后规则间的相关性; 但该策略使得后续学习过程所能参考的样例越来越少. 试设计一种不删除样例的规则学习算法.

## 参考文献

- Bratko, I. and S. Muggleton. (1995). "Applications of inductive logic programming." *Communications of the ACM*, 38(11):65–70.
- Brunk, C. A. and M. J. Pazzani. (1991). "An investigation of noise-tolerant relational concept learning algorithms." In *Proceedings of the 8th International Workshop on Machine Learning (IWML)*, 389–393, Evanston, IL.
- Carlson, A., J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. (2010). "Toward an architecture for never-ending language learning." In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, 1306–1313, Atlanta, GA.
- Cendrowska, J. (1987). "PRISM: An algorithm for inducing modular rules." *International Journal of Man-Machine Studies*, 27(4):349–370.
- Ceri, S., G. Gottlob, and L. Tanca. (1989). "What you always wanted to know about Datalog (and never dared to ask)." *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166.
- Clark, P. and T. Niblett. (1989). "The CN2 induction algorithm." *Machine Learning*, 3(4):261–283.
- Cohen, W. W. (1995). "Fast effective rule induction." In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, 115–123, Tahoe, CA.
- De Raedt, L., P. Frasconi, K. Kersting, and S. Muggleton, eds. (2008). *Probabilistic Inductive Logic Programming: Theory and Applications*. Springer, Berlin.
- Friedman, N., L. Getoor, D. Koller, and A. Pfeffer. (1999). "Learning probabilistic relational models." In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, 1300–1307, Stockholm, Sweden.
- Fürnkranz, J. (1994). "Top-down pruning in relational learning." In *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI)*, 453–457, Amsterdam, The Netherlands.
- Fürnkranz, J., D. Gamberger, and N. Lavrač. (2012). *Foundations of Rule Learning*. Springer, Berlin.

- Fürnkranz, J. and G. Widmer. (1994). "Incremental reduced error pruning." In *Proceedings of the 11th International Conference on Machine Learning (ICML)*, 70–77, New Brunswick, NJ.
- Getoor, L. and B. Taskar. (2007). *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.
- Jaeger, M. (2002). "Relational Bayesian networks: A survey." *Electronic Transactions on Artificial Intelligence*, 6:Article 15.
- Kersting, K., L. De Raedt, and S. Kramer. (2000). "Interpreting Bayesian logic programs." In *Proceedings of the AAAI'2000 Workshop on Learning Statistical Models from Relational Data*, 29–35, Austin, TX.
- Lavrač, N. and S. Dzeroski. (1993). *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, NY.
- Michalski, R. S. (1969). "On the quasi-minimal solution of the general covering problem." In *Proceedings of the 5th International Symposium on Information Processing (FCIP)*, volume A3, 125–128, Bled, Yugoslavia.
- Michalski, R. S. (1983). "A theory and methodology of inductive learning." In *Machine Learning: An Artificial Intelligence Approach* (R. S. Michalski, J. Carbonell, and T. Mitchell, eds.), 111–161, Tioga, Palo Alto, CA.
- Michalski, R. S., I. Mozetic, J. Hong, and N. Lavrač. (1986). "The multi-purpose incremental learning system AQ15 and its testing application to three medical domains." In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI)*, 1041–1045, Philadelphia, PA.
- Muggleton, S. (1991). "Inductive logic programming." *New Generation Computing*, 8(4):295–318.
- Muggleton, S., ed. (1992). *Inductive Logic Programming*. Academic Press, London, UK.
- Muggleton, S. (1995). "Inverse entailment and Progol." *New Generation Computing*, 13(3-4):245–286.
- Muggleton, S. and W. Buntine. (1988). "Machine invention of first order predicates by inverting resolution." In *Proceedings of the 5th International Workshop on Machine Learning (IWML)*, 339–352, Ann Arbor, MI.
- Muggleton, S. and C. Feng. (1990). "Efficient induction of logic programs."

- In *Proceedings of the 1st International Workshop on Algorithmic Learning Theory (ALT)*, 368–381, Tokyo, Japan.
- Muggleton, S. and D. Lin. (2013). “Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited.” In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 1551–1557, Beijing, China.
- Plotkin, G. D. (1970). “A note on inductive generalization.” In *Machine Intelligence 5* (B. Meltzer and D. Mitchie, eds.), 153–165, Edinburgh University Press, Edinburgh, Scotland.
- Plotkin, G. D. (1971). “A further note on inductive generalization.” In *Machine Intelligence 6* (B. Meltzer and D. Mitchie, eds.), 107–124, Edinburgh University Press, Edinburgh, Scotland.
- Quinlan, J. R. (1990). “Learning logical definitions from relations.” *Machine Learning*, 5(3):239–266.
- Richardson, M. and P. Domingos. (2006). “Markov logic networks.” *Machine Learning*, 62(1-2):107–136.
- Robinson, J. A. (1965). “A machine-oriented logic based on the resolution principle.” *Journal of the ACM*, 12(1):23–41.
- Srinivasan,A.(1999).“The Aleph manual.” <http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph.html>.
- Winston, P. H. (1970). *Learning structural descriptions from examples*. Ph.D. thesis, Department of Electrical Engineering, MIT, Cambridge, MA.
- Wnek, J. and R. S. Michalski. (1994). “Hypothesis-driven constructive induction in AQ17-HCI: A method and experiments.” *Machine Learning*, 2(14): 139–168.

## 休息一会儿

### 小故事：机器学习先驱雷萨德·迈克尔斯基

AQ 系列算法是规则学习研究早期的重要成果，主要发明人是机器学习先驱、美籍波兰裔科学家雷萨德·迈克尔斯基 (Ryszard S. Michalski, 1937—2007)。

卡鲁兹(Kalusz)在历史上先后属于波兰、俄罗斯、德国、乌克兰等国。



迈克尔斯基出生在波兰卡鲁兹，1969 年在波兰获得计算机科学博士学位，同年在南斯拉夫布莱德 (Bled, 现属斯洛文尼亚) 举行的 FCIP 会议上发表了 AQ。1970 年他前往美国 UIUC 任教，此后在美国进一步发展了 AQ 系列算法。迈克尔斯基是机器学习领域的主要奠基人之一。1980 年他与 J. G. Carbonell、T. Mitchell 一起在卡耐基梅隆大学组织了第一次机器学习研讨会，1983、1985 年又组织了第二、三次，这个系列研讨会后来发展成国际机器学习会议 (ICML)；1983 年，迈克尔斯基作为第一主编出版了《机器学习：一种人工智能途径》这本机器学习史上里程碑性质的著作；1986 年 *Machine Learning* 创刊，迈克尔斯基是最初的三位编辑之一。1988 年他将研究组迁到乔治梅森大学，使该校成为机器学习早期发展的一个重镇。

参见 1.5 节。



# 第 16 章 强化学习

## 16.1 任务与奖赏

亦称“再励学习”。

我们考虑一下如何种西瓜。种瓜有许多步骤，从一开始的选种，到定期浇水、施肥、除草、杀虫，经过一段时间才能收获西瓜。通常要等到收获后，我们才知道种出的瓜好不好。若将得到好瓜作为辛勤种瓜劳动的奖赏，则在种瓜过程中当我们执行某个操作（例如，施肥）时，并不能立即获得这个最终奖赏，甚至难以判断当前操作对最终奖赏的影响，仅能得到一个当前反馈（例如，瓜苗看起来更健壮了）。我们需多次种瓜，在种瓜过程中不断摸索，然后才能总结出较好的种瓜策略。这个过程抽象出来，就是“强化学习”（reinforcement learning）。

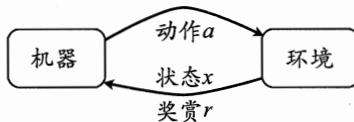


图 16.1 强化学习图示

图 16.1 给出了强化学习的一个简单图示。强化学习任务通常用马尔可夫决策过程 (Markov Decision Process, 简称 MDP) 来描述：机器处于环境  $E$  中，状态空间为  $X$ ，其中每个状态  $x \in X$  是机器感知到的环境的描述，如在种瓜任务上这就是当前瓜苗长势的描述；机器能采取的动作构成了动作空间  $A$ ，如种瓜过程中有浇水、施不同的肥、使用不同的农药等多种可供选择的动作；若某个动作  $a \in A$  作用在当前状态  $x$  上，则潜在的转移函数  $P$  将使得环境从当前状态按某种概率转移到另一个状态，如瓜苗状态为缺水，若选择动作浇水，则瓜苗长势会发生变化，瓜苗有一定的概率恢复健康，也有一定的概率无法恢复；在转移到另一个状态的同时，环境会根据潜在的“奖赏” (reward) 函数  $R$  反馈给机器一个奖赏，如保持瓜苗健康对应奖赏 +1，瓜苗凋零对应奖赏 -10，最终种出了好瓜对应奖赏 +100。综合起来，强化学习任务对应了四元组  $E = \langle X, A, P, R \rangle$ ，其中  $P : X \times A \times X \mapsto \mathbb{R}$  指定了状态转移概率， $R : X \times A \times X \mapsto \mathbb{R}$  指定了奖赏；在有的应用中，奖赏函数可能仅与状态转移有关，即  $R : X \times X \mapsto \mathbb{R}$ 。

图 16.2 给出了一个简单例子：给西瓜浇水的马尔可夫决策过程。该任务中

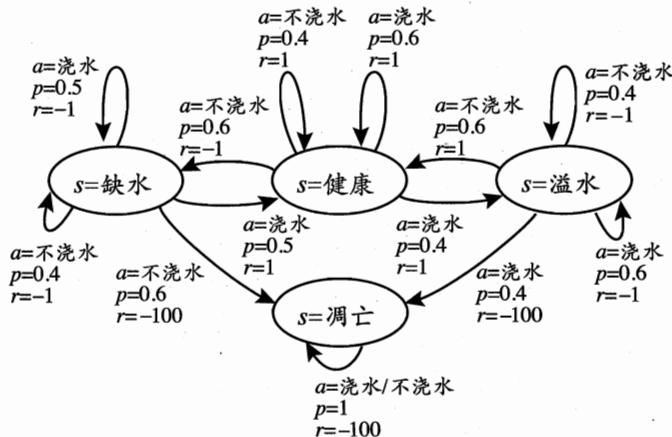


图 16.2 给西瓜浇水问题的马尔可夫决策过程

只有四个状态(健康、缺水、溢水、凋亡)和两个动作(浇水、不浇水), 在每一步转移后, 若状态是保持瓜苗健康则获得奖赏 1, 瓜苗缺水或溢水奖赏为 -1, 这时通过浇水或不浇水可以恢复健康状态, 当瓜苗凋亡时奖赏是最小值 -100 且无法恢复. 图中箭头表示状态转移, 箭头旁的  $a, p, r$  分别表示导致状态转移的动作、转移概率以及返回的奖赏. 容易看出, 最优策略在“健康”状态选择动作“浇水”、在“溢水”状态选择动作“不浇水”、在“缺水”状态选择动作“浇水”、在“凋亡”状态可选择任意动作.

需注意“机器”与“环境”的界限, 例如在种西瓜任务中, 环境是西瓜生长的自然世界; 在下棋对弈中, 环境是棋盘与对手; 在机器人控制中, 环境是机器人的躯体与物理世界. 总之, 在环境中状态的转移、奖赏的返回是不受机器控制的, 机器只能通过选择要执行的动作来影响环境, 也只能通过观察转移后的状态和返回的奖赏来感知环境.

机器要做的是通过在环境中不断地尝试而学得一个“策略”(policy)  $\pi$ , 根据这个策略, 在状态  $x$  下就能得知要执行的动作  $a = \pi(x)$ , 例如看到瓜苗状态是缺水时, 能返回动作“浇水”. 策略有两种表示方法: 一种是将策略表示为函数  $\pi : X \mapsto A$ , 确定性策略常用这种表示; 另一种是概率表示  $\pi : X \times A \mapsto \mathbb{R}$ , 随机性策略常用这种表示,  $\pi(x, a)$  为状态  $x$  下选择动作  $a$  的概率, 这里必须有  $\sum_a \pi(x, a) = 1$ .

策略的优劣取决于长期执行这一策略后得到的累积奖赏, 例如某个策略使得瓜苗枯死, 它的累积奖赏会很小, 另一个策略种出了好瓜, 它的累积奖赏会很

大。在强化学习任务中，学习的目的就是要找到能使长期累积奖赏最大化的策略。长期累积奖赏有多种计算方式，常用的有“ $T$  步累积奖赏”  $\mathbb{E}[\frac{1}{T} \sum_{t=1}^T r_t]$  和“ $\gamma$  折扣累积奖赏”  $\mathbb{E}[\sum_{t=0}^{+\infty} \gamma^t r_{t+1}]$ ，其中  $r_t$  表示第  $t$  步获得的奖赏值， $\mathbb{E}$  表示对所有随机变量求期望。

读者也许已经感觉到强化学习与监督学习的差别。若将这里的“状态”对应为监督学习中的“示例”、“动作”对应为“标记”，则可看出，强化学习中的“策略”实际上就相当于监督学习中的“分类器”（当动作是离散的）或“回归器”（当动作是连续的），模型的形式并无差别。但不同的是，在强化学习中并没有监督学习中的有标记样本（即“示例-标记”对），换言之，没有人直接告诉机器在什么状态下应该做什么动作，只有等到最终结果揭晓，才能通过“反思”之前的动作是否正确来进行学习。因此，强化学习在某种意义上可看作具有“延迟标记信息”的监督学习问题。

## 16.2 K-摇臂赌博机

### 16.2.1 探索与利用

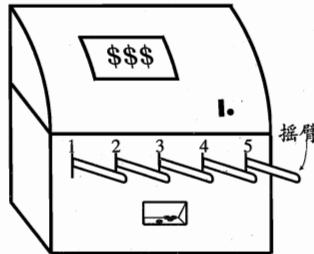
与一般监督学习不同，强化学习任务的最终奖赏是在多步动作之后才能观察到，这里我们不妨先考虑比较简单的情形：最大化单步奖赏，即仅考虑一步操作。需注意的是，即便在这样的简化情形下，强化学习仍与监督学习有显著不同，因为机器需通过尝试来发现各个动作产生的结果，而没有训练数据告诉机器应当做哪个动作。

欲最大化单步奖赏需考虑两个方面：一是需知道每个动作带来的奖赏，二是要执行奖赏最大的动作。若每个动作对应的奖赏是一个确定值，那么尝试一遍所有的动作便能找出奖赏最大的动作。然而，更一般的情形是，一个动作的奖赏值是来自于一个概率分布，仅通过一次尝试并不能确切地获得平均奖赏值。

亦称“K-摇臂老虎机”。

实际上，单步强化学习任务对应了一个理论模型，即“ $K$ -摇臂赌博机” ( $K$ -armed bandit)。如图 16.3 所示， $K$ -摇臂赌博机有  $K$  个摇臂，赌徒在投入一个硬币后可选择按下其中一个摇臂，每个摇臂以一定的概率吐出硬币，但这个概率赌徒并不知道。赌徒的目标是通过一定的策略最大化自己的奖赏，即获得最多的硬币。

若仅为获知每个摇臂的期望奖赏，则可采用“仅探索” (exploration-only) 法：将所有的尝试机会平均分配给每个摇臂（即轮流按下每个摇臂），最后以每个摇臂各自的平均吐币概率作为其奖赏期望的近似估计。若仅为执行奖赏最大的动作，则可采用“仅利用” (exploitation-only) 法：按下目前最优的（即到

图 16.3  $K$ -摇臂赌博机图示

目前为止平均奖赏最大的)摇臂, 若有多个摇臂同为最优, 则从中随机选取一个。显然, “仅探索” 法能很好地估计每个摇臂的奖赏, 却会失去很多选择最优摇臂的机会; “仅利用” 法则相反, 它没有很好地估计摇臂期望奖赏, 很可能经常选不到最优摇臂。因此, 这两种方法都难以使最终的累积奖赏最大化。

事实上, “探索” (即估计摇臂的优劣)和“利用” (即选择当前最优摇臂)这两者是矛盾的, 因为尝试次数(即总投币数)有限, 加强了一方则会自然削弱另一方, 这就是强化学习所面临的“探索-利用窘境” (Exploration-Exploitation dilemma)。显然, 欲累积奖赏最大, 则必须在探索与利用之间达成较好的折中。

### 16.2.2 $\epsilon$ -贪心

$\epsilon$ -贪心法基于一个概率来对探索和利用进行折中: 每次尝试时, 以  $\epsilon$  的概率进行探索, 即以均匀概率随机选取一个摇臂; 以  $1 - \epsilon$  的概率进行利用, 即选择当前平均奖赏最高的摇臂(若多个, 则随机选取一个)。

令  $Q(k)$  记录摇臂  $k$  的平均奖赏。若摇臂  $k$  被尝试了  $n$  次, 得到的奖赏为  $v_1, v_2, \dots, v_n$ , 则平均奖赏为

$$Q(k) = \frac{1}{n} \sum_{i=1}^n v_i . \quad (16.1)$$

若直接根据式(16.1)计算平均奖赏, 则需记录  $n$  个奖赏值。显然, 更高效的做法是对均值进行增量式计算, 即每尝试一次就立即更新  $Q(k)$ 。不妨用下标来表示尝试的次数, 初始时  $Q_0(k) = 0$ 。对于任意的  $n \geq 1$ , 若第  $n-1$  次尝试后的平均奖赏为  $Q_{n-1}(k)$ , 则在经过第  $n$  次尝试获得奖赏  $v_n$  后, 平均奖赏应更新为

$$Q_n(k) = \frac{1}{n} ((n-1) \times Q_{n-1}(k) + v_n) \quad (16.2)$$

式(16.3)会在 16.4.2 节中用到.

$$= Q_{n-1}(k) + \frac{1}{n} (v_n - Q_{n-1}(k)). \quad (16.3)$$

这样,无论摇臂被尝试多少次都仅需记录两个值:已尝试次数  $n-1$  和最近平均奖赏  $Q_{n-1}(k)$ .  $\epsilon$ -贪心算法描述如图 16.4 所示.

---

输入: 摆臂数  $K$ ;  
奖赏函数  $R$ ;  
尝试次数  $T$ ;  
探索概率  $\epsilon$ .

过程:

```

1:  $r = 0$ ;
2:  $\forall i = 1, 2, \dots, K : Q(i) = 0, \text{count}(i) = 0$ ;
3: for  $t = 1, 2, \dots, T$  do
4:   if  $\text{rand}() < \epsilon$  then
5:      $k =$  从  $1, 2, \dots, K$  中以均匀分布随机选取
6:   else
7:      $k = \arg \max_i Q(i)$ 
8:   end if
9:    $v = R(k)$ ;
10:   $r = r + v$ ;
11:   $Q(k) = \frac{Q(k) \times \text{count}(k) + v}{\text{count}(k) + 1}$ ;
12:   $\text{count}(k) = \text{count}(k) + 1$ ;
13: end for
```

输出: 累积奖赏  $r$

---

图 16.4  $\epsilon$ -贪心算法

若摇臂奖赏的不确定性较大,例如概率分布较宽时,则需更多的探索,此时需要较大的  $\epsilon$  值;若摇臂的不确定性较小,例如概率分布较集中时,则少量的尝试就能很好地近似真实奖赏,此时需要的  $\epsilon$  较小.通常令  $\epsilon$  取一个较小的常数,如 0.1 或 0.01.然而,若尝试次数非常大,那么在一段时间后,摇臂的奖赏都能很好地近似出来,不再需要探索,这种情形下可让  $\epsilon$  随着尝试次数的增加而逐渐减小,例如令  $\epsilon = 1/\sqrt{t}$ .

### 16.2.3 Softmax

Softmax 算法基于当前已知的摇臂平均奖赏来对探索和利用进行折中.若各摇臂的平均奖赏相当,则选取各摇臂的概率也相当;若某些摇臂的平均奖赏明显高于其他摇臂,则它们被选取的概率也明显更高.

Softmax 算法中摇臂概率的分配是基于 Boltzmann 分布

$$P(k) = \frac{e^{\frac{Q(k)}{\tau}}}{\sum_{i=1}^K e^{\frac{Q(i)}{\tau}}}, \quad (16.4)$$

其中,  $Q(i)$  记录当前摇臂的平均奖赏;  $\tau > 0$  称为“温度”,  $\tau$  越小则平均奖赏高的摇臂被选取的概率越高.  $\tau$  趋于 0 时 Softmax 将趋于“仅利用”,  $\tau$  趋于无穷大时 Softmax 则将趋于“仅探索”. Softmax 算法描述如图 16.5 所示.

第 4 行中式(16.4)的参数.

$Q(i)$  和  $\text{count}(i)$  分别记录摇臂  $i$  的平均奖赏和选中次数.

本次尝试的奖赏值.

式(16.2)更新平均奖赏.

---

输入: 摆臂数  $K$ ;  
奖赏函数  $R$ ;  
尝试次数  $T$ ;  
温度参数  $\tau$ .

过程:

- 1:  $r = 0$ ;
- 2:  $\forall i = 1, 2, \dots, K : Q(i) = 0, \text{count}(i) = 0$ ;
- 3: **for**  $t = 1, 2, \dots, T$  **do**
- 4:    $k$  = 从  $1, 2, \dots, K$  中根据式(16.4)随机选取
- 5:    $v = R(k)$ ;
- 6:    $r = r + v$ ;
- 7:    $Q(k) = \frac{Q(k) \times \text{count}(k) + v}{\text{count}(k) + 1}$ ;
- 8:    $\text{count}(k) = \text{count}(k) + 1$ ;
- 9: **end for**

输出: 累积奖赏  $r$

---

图 16.5 Softmax 算法

$\epsilon$ -贪心算法与 Softmax 算法孰优孰劣, 主要取决于具体应用. 为了更直观地观察它们的差别, 考虑一个简单的例子: 假定 2-摇臂赌博机的摇臂 1 以 0.4 的概率返回奖赏 1, 以 0.6 的概率返回奖赏 0; 摆臂 2 以 0.2 的概率返回奖赏 1, 以 0.8 的概率返回奖赏 0. 图 16.6 显示了不同算法在不同参数下的平均累积奖赏, 其中每条曲线对应于重复 1000 次实验的平均结果. 可以看出, Softmax ( $\tau = 0.01$ ) 的曲线与“仅利用”的曲线几乎重合.

对于离散状态空间、离散动作空间上的多步强化学习任务, 一种直接的办法是将每个状态上动作的选择看作一个  $K$ -摇臂赌博机问题, 用强化学习任务的累积奖赏来代替  $K$ -摇臂赌博机算法中的奖赏函数, 即可将赌博机算法用于每个状态: 对每个状态分别记录各动作的尝试次数、当前平均累积奖赏等信息, 基于赌博机算法选择要尝试的动作. 然而这样的做法有很多局限, 因为它没

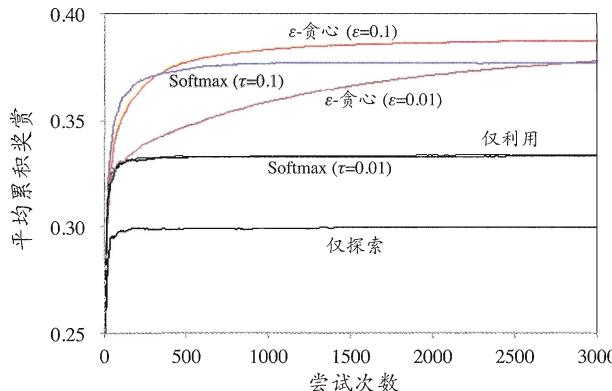


图 16.6 不同算法在 2-摇臂赌博机上的性能比较

有考虑强化学习任务马尔可夫决策过程的结构. 在 16.3 节将会看到, 若能有效考虑马尔可夫决策过程的特性, 则可有更聪明的办法.

### 16.3 有模型学习

16.4 节将讨论模型未知情形.

考虑多步强化学习任务, 暂且先假定任务对应的马尔可夫决策过程四元组  $E = \langle X, A, P, R \rangle$  均为已知, 这样的情形称为“模型已知”, 即机器已对环境进行了建模, 能在机器内部模拟出与环境相同或近似的状况. 在已知模型的环境中学习称为“有模型学习”(model-based learning). 此时, 对于任意状态  $x, x'$  和动作  $a$ , 在  $x$  状态下执行动作  $a$  转移到  $x'$  状态的概率  $P_{x \rightarrow x'}^a$  是已知的, 该转移所带来的奖赏  $R_{x \rightarrow x'}^a$  也是已知的. 为便于讨论, 不妨假设状态空间  $X$  和动作空间  $A$  均为有限.

#### 16.3.1 策略评估

在模型已知时, 对任意策略  $\pi$  能估计出该策略带来的期望累积奖赏. 令函数  $V^\pi(x)$  表示从状态  $x$  出发, 使用策略  $\pi$  所带来的累积奖赏; 函数  $Q^\pi(x, a)$  表示从状态  $x$  出发, 执行动作  $a$  后再使用策略  $\pi$  带来的累积奖赏. 这里的  $V(\cdot)$  称为“状态值函数”(state value function),  $Q(\cdot)$  称为“状态-动作值函数”(state-action value function), 分别表示指定“状态”上以及指定“状态-动作”上的累积奖赏.

由累积奖赏的定义, 有状态值函数

$$\begin{cases} V_T^\pi(x) = \mathbb{E}_\pi \left[ \frac{1}{T} \sum_{t=1}^T r_t \mid x_0 = x \right], & T \text{ 步累积奖赏;} \\ V_\gamma^\pi(x) = \mathbb{E}_\pi \left[ \sum_{t=0}^{+\infty} \gamma^t r_{t+1} \mid x_0 = x \right], & \gamma \text{ 折扣累积奖赏.} \end{cases} \quad (16.5)$$

为叙述简洁, 后面在涉及上述两种累积奖赏时, 就不再说明奖赏类别, 读者从上下文应能容易地判知. 令  $x_0$  表示起始状态,  $a_0$  表示起始状态上采取的第一个动作; 对于  $T$  步累积奖赏, 用下标  $t$  表示后续执行的步数. 我们有状态-动作值函数

$$\begin{cases} Q_T^\pi(x, a) = \mathbb{E}_\pi \left[ \frac{1}{T} \sum_{t=1}^T r_t \mid x_0 = x, a_0 = a \right]; \\ Q_\gamma^\pi(x, a) = \mathbb{E}_\pi \left[ \sum_{t=0}^{+\infty} \gamma^t r_{t+1} \mid x_0 = x, a_0 = a \right]. \end{cases} \quad (16.6)$$

这样的递归等式称为 Bellman 等式.

由于 MDP 具有马尔可夫性质, 即系统下一时刻的状态仅由当前时刻的状态决定, 不依赖于以往任何状态, 于是值函数有很简单的递归形式. 对于  $T$  步累积奖赏有

$$\begin{aligned} V_T^\pi(x) &= \mathbb{E}_\pi \left[ \frac{1}{T} \sum_{t=1}^T r_t \mid x_0 = x \right] \\ &= \mathbb{E}_\pi \left[ \frac{1}{T} r_1 + \frac{T-1}{T} \frac{1}{T-1} \sum_{t=2}^T r_t \mid x_0 = x \right] \\ &= \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} \mathbb{E}_\pi \left[ \frac{1}{T-1} \sum_{t=1}^{T-1} r_t \mid x_0 = x' \right] \right) \\ &= \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V_{T-1}^\pi(x') \right). \end{aligned} \quad (16.7)$$

类似地, 对于  $\gamma$  折扣累积奖赏有

$$V_\gamma^\pi(x) = \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a (R_{x \rightarrow x'}^a + \gamma V_\gamma^\pi(x')). \quad (16.8)$$

需注意的是, 正是由于  $P$  和  $R$  已知, 才可以进行全概率展开.

读者可能已发现, 用上面的递归等式来计算值函数, 实际上就是一种动态规划算法. 对于  $V_T^\pi$ , 可设想递归一直进行下去, 直到最初的起点; 换言之, 从值函数的初始值  $V_0^\pi$  出发, 通过一次迭代能计算出每个状态的单步奖赏  $V_1^\pi$ , 进而

---

**输入:** MDP 四元组  $E = \langle X, A, P, R \rangle$ ;  
被评估的策略  $\pi$ ;  
累积奖赏参数  $T$ .

**过程:**

- 1:  $\forall x \in X : V(x) = 0$ ;
- 2: **for**  $t = 1, 2, \dots$  **do**
- 3:    $\forall x \in X : V'(x) = \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{t} R_{x \rightarrow x'}^a + \frac{t-1}{t} V(x') \right)$ ;
- 4:   **if**  $t = T + 1$  **then**
- 5:     **break**
- 6:   **else**
- 7:      $V = V'$
- 8:   **end if**
- 9: **end for**

**输出:** 状态值函数  $V$

---

图 16.7 基于  $T$  步累积奖赏的策略评估算法

从单步奖赏出发, 通过一次迭代计算出两步累积奖赏  $V_2^\pi$ , ……图 16.7 中算法遵循了上述流程, 对于  $T$  步累积奖赏, 只需迭代  $T$  轮就能精确地求出值函数.

对于  $V_\gamma^\pi$ , 由于  $\gamma^t$  在  $t$  很大时趋于 0, 因此也能使用类似的算法, 只需将图 16.7 算法的第 3 行根据式(16.8)进行替换. 此外, 由于算法可能会迭代很多次, 因此需设置一个停止准则. 常见的是设置一个阈值  $\theta$ , 若在执行一次迭代后值函数的改变小于  $\theta$  则算法停止; 相应的, 图 16.7 算法第 4 行中的  $t = T + 1$  需替换为

$$\max_{x \in X} |V(x) - V'(x)| < \theta. \quad (16.9)$$

有了状态值函数  $V$ , 就能直接计算出状态-动作值函数

$$\begin{cases} Q_T^\pi(x, a) = \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V_{T-1}^\pi(x') \right); \\ Q_\gamma^\pi(x, a) = \sum_{x' \in X} P_{x \rightarrow x'}^a (R_{x \rightarrow x'}^a + \gamma V_\gamma^\pi(x')). \end{cases} \quad (16.10)$$

### 16.3.2 策略改进

对某个策略的累积奖赏进行评估后, 若发现它并非最优策略, 则当然希望对其进行改进. 理想的策略应能最大化累积奖赏

$$\pi^* = \arg \max_{\pi} \sum_{x \in X} V^\pi(x). \quad (16.11)$$

$V(x)$  为  $x$  的累积奖赏.

式(16.7)更新值函数.

这个写法是为了便于在同样的算法框架下考虑  $T$  步累积奖赏和  $\gamma$  折扣累积奖赏.

参见习题 16.2.

一个强化学习任务可能有多个最优策略, 最优策略所对应的值函数  $V^*$  称为最优化函数, 即

$$\forall x \in X : V^*(x) = V^{\pi^*}(x). \quad (16.12)$$

注意, 当策略空间无约束时式(16.12)的  $V^*$  才是最优策略对应的值函数, 例如对离散状态空间和离散动作空间, 策略空间是所有状态上所有动作的组合, 共有  $|A|^{|X|}$  种不同的策略. 若策略空间有约束, 则违背约束的策略是“不合法”的, 即便其值函数所取得的累积奖赏值最大, 也不能作为最优化函数.

由于最优化函数的累积奖赏值已达最大, 因此可对前面的 Bellman 等式(16.7)和(16.8)做一个改动, 即将对动作的求和改为取最优:

$$\begin{cases} V_T^*(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V_{T-1}^*(x') \right); \\ V_\gamma^*(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a (R_{x \rightarrow x'}^a + \gamma V_\gamma^*(x')). \end{cases} \quad (16.13)$$

换言之,

$$V^*(x) = \max_{a \in A} Q^{\pi^*}(x, a). \quad (16.14)$$

代入式(16.10)可得最优状态-动作值函数

$$\begin{cases} Q_T^*(x, a) = \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} \max_{a' \in A} Q_{T-1}^*(x', a') \right); \\ Q_\gamma^*(x, a) = \sum_{x' \in X} P_{x \rightarrow x'}^a (R_{x \rightarrow x'}^a + \gamma \max_{a' \in A} Q_\gamma^*(x', a')). \end{cases} \quad (16.15)$$

上述关于最优化函数的等式, 称为最优 Bellman 等式, 其唯一解是最优化函数.

最优 Bellman 等式揭示了非最优策略的改进方式: 将策略选择的动作改变为当前最优的动作. 显然, 这样的改变能使策略更好. 不妨令动作改变后对应的策略为  $\pi'$ , 改变动作的条件为  $Q^\pi(x, \pi'(x)) \geq V^\pi(x)$ , 以  $\gamma$  折扣累积奖赏为例, 由式(16.10)可计算出递推不等式

$$\begin{aligned} V^\pi(x) &\leq Q^\pi(x, \pi'(x)) \\ &= \sum_{x' \in X} P_{x \rightarrow x'}^{\pi'(x)} (R_{x \rightarrow x'}^{\pi'(x)} + \gamma V^\pi(x')) \\ &\leq \sum_{x' \in X} P_{x \rightarrow x'}^{\pi'(x)} (R_{x \rightarrow x'}^{\pi'(x)} + \gamma Q^\pi(x', \pi'(x'))) \\ &= \dots \end{aligned}$$

$$= V^{\pi'}(x). \quad (16.16)$$

值函数对于策略的每一点改进都是单调递增的, 因此对于当前策略  $\pi$ , 可放心地将其改进为

$$\pi'(x) = \arg \max_{a \in A} Q^{\pi}(x, a), \quad (16.17)$$

直到  $\pi'$  与  $\pi$  一致、不再发生变化, 此时就满足了最优 Bellman 等式, 即找到了最优策略.

### 16.3.3 策略迭代与值迭代

由前两小节我们知道了如何评估一个策略的值函数, 以及在策略评估后如何改进至获得最优策略. 显然, 将这两者结合起来即可得到求解最优解的方法: 从一个初始策略(通常是随机策略)出发, 先进行策略评估, 然后改进策略, 评估改进的策略, 再进一步改进策略, ……不断迭代进行策略评估和改进, 直到策略收敛、不再改变为止. 这样的做法称为“策略迭代”(policy iteration).

图 16.8 给出的算法描述, 就是在基于  $T$  步累积奖赏策略评估的基础上, 加

---

**输入:** MDP 四元组  $E = \langle X, A, P, R \rangle$ ;  
累积奖赏参数  $T$ .

**过程:**

1:  $\forall x \in X : V(x) = 0, \pi(x, a) = \frac{1}{|A(x)|}$ ;  
2: **loop**  
3:   **for**  $t = 1, 2, \dots$  **do**  
4:      $\forall x \in X : V'(x) = \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{t} R_{x \rightarrow x'}^a + \frac{t-1}{t} V(x') \right)$ ;  
5:     **if**  $t = T + 1$  **then**  
6:       **break**  
7:     **else**  
8:        $V = V'$   
9:     **end if**  
10:   **end for**  
11:    $\forall x \in X : \pi'(x) = \arg \max_{a \in A} Q(x, a)$ ;  
12:   **if**  $\forall x : \pi'(x) = \pi(x)$  **then**  
13:     **break**  
14:   **else**  
15:      $\pi = \pi'$   
16:   **end if**  
17: **end loop**

**输出:** 最优策略  $\pi$

---

图 16.8 基于  $T$  步累积奖赏的策略迭代算法

参见习题 16.3.

入策略改进而形成的策略迭代算法。类似的,可得到基于  $\gamma$  折扣累积奖赏的策略迭代算法。策略迭代算法在每次改进策略后都需重新进行策略评估,这通常比较耗时。

由式(16.16)可知,策略改进与值函数的改进是一致的,因此可将策略改进视为值函数的改善,即由式(16.13)可得

$$\begin{cases} V_T(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V_{T-1}(x') \right); \\ V_\gamma(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a (R_{x \rightarrow x'}^a + \gamma V_\gamma(x')) . \end{cases} \quad (16.18)$$

于是可得到值迭代(value iteration)算法,如图 16.9 所示。

---

输入: MDP 四元组  $E = \langle X, A, P, R \rangle$ ;  
累积奖赏参数  $T$ ;  
收敛阈值  $\theta$ 。

过程:

```

1:  $\forall x \in X : V(x) = 0;$ 
2: for  $t = 1, 2, \dots$  do
3:    $\forall x \in X : V'(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{t} R_{x \rightarrow x'}^a + \frac{t-1}{t} V(x') \right);$ 
4:   if  $\max_{x \in X} |V(x) - V'(x)| < \theta$  then
5:     break
6:   else
7:      $V = V'$ 
8:   end if
9: end for
```

式(16.18)更新值函数。

式(16.10)计算  $Q$  值。

输出: 策略  $\pi(x) = \arg \max_{a \in A} Q(x, a)$

---

图 16.9 基于  $T$  步累积奖赏的值迭代算法

若采用  $\gamma$  折扣累积奖赏,只需将图 16.9 算法中第 3 行替换为

$$\forall x \in X : V'(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a (R_{x \rightarrow x'}^a + \gamma V(x')). \quad (16.19)$$

从上面的算法可看出,在模型已知时强化学习任务能归结为基于动态规划的寻优问题。与监督学习不同,这里并未涉及到泛化能力,而是为每一个状态找到最好的动作。

#### 16.4 免模型学习

在现实的强化学习任务中,环境的转移概率、奖赏函数往往很难得知,甚

至很难知道环境中一共有多少状态。若学习算法不依赖于环境建模，则称为“免模型学习”(model-free learning)，这比有模型学习要困难得多。

#### 16.4.1 蒙特卡罗强化学习

蒙特卡罗方法参见 14.7 节；14.5.1 节中使用过马尔可夫链蒙特卡罗方法。

在免模型情形下，策略迭代算法首先遇到的问题是策略无法评估，这是由于模型未知而导致无法做全概率展开。此时，只能通过在环境中执行选择的动作，来观察转移的状态和得到的奖赏。受  $K$  摆臂赌博机的启发，一种直接的策略评估替代方法是多次“采样”，然后求取平均累积奖赏来作为期望累积奖赏的近似，这称为蒙特卡罗强化学习。由于采样必须为有限次数，因此该方法更适合于使用  $T$  步累积奖赏的强化学习任务。

另一方面，策略迭代算法估计的是状态值函数  $V$ ，而最终的策略是通过状态-动作值函数  $Q$  来获得。当模型已知时，从  $V$  到  $Q$  有很简单的转换方法，而当模型未知时，这也可能出现困难。于是，我们将估计对象从  $V$  转变为  $Q$ ，即估计每一对“状态-动作”的值函数。

此外，在模型未知的情形下，机器只能是从一个起始状态(或起始状态集合)开始探索环境，而策略迭代算法由于需对每个状态分别进行估计，因此在这种情形下无法实现。例如探索种瓜的过程只能从播下种子开始，而不能任意选择种植过程中的一个状态开始。因此，我们只能在探索的过程中逐渐发现各个状态并估计各状态-动作对的值函数。

综合起来，在模型未知的情形下，我们从起始状态出发，使用某种策略进行采样，执行该策略  $T$  步并获得轨迹

$$\langle x_0, a_0, r_1, x_1, a_1, r_2, \dots, x_{T-1}, a_{T-1}, r_T, x_T \rangle,$$

然后，对轨迹中出现的每一对状态-动作，记录其后的奖赏之和，作为该状态-动作对的一次累积奖赏采样值。多次采样得到多条轨迹后，将每个状态-动作对的累积奖赏采样值进行平均，即得到状态-动作值函数的估计。

可以看出，欲较好地获得值函数的估计，就需要多条不同的采样轨迹。然而，我们的策略有可能是确定性的，即对于某个状态只会输出一个动作，若使用这样的策略进行采样，则只能得到多条相同的轨迹。这与  $K$  摆臂赌博机的“仅利用”法面临相同的问题，因此可借鉴探索与利用折中的办法，例如使用  $\epsilon$ -贪心法，以  $\epsilon$  的概率从所有动作中均匀随机选取一个，以  $1 - \epsilon$  的概率选取当前最优动作。我们将确定性的策略  $\pi$  称为“原始策略”，在原始策略上使用  $\epsilon$ -贪心法的策略记为

$$\pi^\epsilon(x) = \begin{cases} \pi(x), & \text{以概率 } 1 - \epsilon; \\ A \text{ 中以均匀概率选取的动作,} & \text{以概率 } \epsilon. \end{cases} \quad (16.20)$$

假定只有一个最优动作.

对于最大化值函数的原始策略  $\pi = \arg \max_a Q(x, a)$ , 其  $\epsilon$ -贪心策略  $\pi^\epsilon$  中, 当前最优动作被选中的概率是  $1 - \epsilon + \frac{\epsilon}{|A|}$ , 而每个非最优动作被选中的概率是  $\frac{\epsilon}{|A|}$ . 于是, 每个动作都有可能被选取, 而多次采样将会产生不同的采样轨迹.

与策略迭代算法类似, 使用蒙特卡罗方法进行策略评估后, 同样要对策略进行改进. 前面在讨论策略改进时利用了式(16.16)揭示的单调性, 通过换入当前最优动作来改进策略. 对于任意原始策略  $\pi$ , 其  $\epsilon$ -贪心策略  $\pi^\epsilon$  仅是将  $\epsilon$  的概率均匀分配给所有动作, 因此对于最大化值函数的原始策略  $\pi'$ , 同样有  $Q^{\pi}(x, \pi'(x)) \geq V^{\pi}(x)$ , 于是式(16.16)仍成立, 即可以使用同样方法来进行策略改进.

图 16.10 给出了上述过程的算法描述, 这里被评估与被改进的是同一个策略, 因此称为“同策略”(on-policy)蒙特卡罗强化学习算法. 算法中奖赏均值采用增量式计算, 每采样出一条轨迹, 就根据该轨迹涉及的所有“状态-动作”对来对值函数进行更新.

---

**输入:** 环境  $E$ ;  
动作空间  $A$ ;  
起始状态  $x_0$ ;  
策略执行步数  $T$ .

**过程:**

默认均匀概率选取动作.  
采样第  $s$  条轨迹.

对每一个状态-动作对.  
计算轨迹中的累积奖赏.  
式(16.2)更新平均奖赏.

根据值函数得到策略.

- 1:  $Q(x, a) = 0, \text{count}(x, a) = 0, \pi(x, a) = \frac{1}{|A(x)|};$
- 2: **for**  $s = 1, 2, \dots$  **do**
- 3:   在  $E$  中执行策略  $\pi$  产生轨迹  
 $< x_0, a_0, r_1, x_1, a_1, r_2, \dots, x_{T-1}, a_{T-1}, r_T, x_T >;$
- 4:   **for**  $t = 0, 1, \dots, T - 1$  **do**
- 5:      $R = \frac{1}{T-t} \sum_{i=t+1}^T r_i;$
- 6:      $Q(x_t, a_t) = \frac{Q(x_t, a_t) \times \text{count}(x_t, a_t) + R}{\text{count}(x_t, a_t) + 1};$
- 7:      $\text{count}(x_t, a_t) = \text{count}(x_t, a_t) + 1$
- 8:   **end for**
- 9:   对所有已见状态  $x$ :  
 $\pi(x, a) = \begin{cases} \arg \max_{a'} Q(x, a'), & \text{以概率 } 1 - \epsilon; \\ \text{以均匀概率从 } A \text{ 中选取动作,} & \text{以概率 } \epsilon. \end{cases}$
- 10: **end for**

**输出:** 策略  $\pi$

---

图 16.10 同策略蒙特卡罗强化学习算法

同策略蒙特卡罗强化学习算法最终产生的是  $\epsilon$ -贪心策略。然而，引入  $\epsilon$ -贪心是为了便于策略评估，在使用策略时并不需要  $\epsilon$ -贪心；实际上我们希望改进的是原始(非  $\epsilon$ -贪心)策略。那么，能否仅在策略评估时引入  $\epsilon$ -贪心，而在策略改进时却改进原始策略呢？

这其实是可行的。不妨用两个不同的策略  $\pi$  和  $\pi'$  来产生采样轨迹，两者的区别在于每个“状态-动作对”被采样的概率不同。一般的，函数  $f$  在概率分布  $p$  下的期望可表达为

$$\mathbb{E}[f] = \int_x p(x)f(x)dx , \quad (16.21)$$

可通过从概率分布  $p$  上的采样  $\{x_1, x_2, \dots, x_m\}$  来估计  $f$  的期望，即

$$\hat{\mathbb{E}}[f] = \frac{1}{m} \sum_{i=1}^m f(x_i) . \quad (16.22)$$

若引入另一个分布  $q$ ，则函数  $f$  在概率分布  $p$  下的期望也可等价地写为

$$\mathbb{E}[f] = \int_x q(x) \frac{p(x)}{q(x)} f(x) dx . \quad (16.23)$$

上式可看作  $\frac{p(x)}{q(x)} f(x)$  在分布  $q$  下的期望，因此通过在  $q$  上的采样  $\{x'_1, x'_2, \dots, x'_m\}$  可估计为

$$\hat{\mathbb{E}}[f] = \frac{1}{m} \sum_{i=1}^m \frac{p(x'_i)}{q(x'_i)} f(x'_i) . \quad (16.24)$$

这样基于一个分布的采样来估计另一个分布下的期望，称为重要性采样(importance sampling)。

回到我们的问题上来，使用策略  $\pi$  的采样轨迹来评估策略  $\pi$ ，实际上就是对累积奖赏估计期望

$$Q(x, a) = \frac{1}{m} \sum_{i=1}^m r_i . \quad (16.25)$$

若改用策略  $\pi'$  的采样轨迹来评估策略  $\pi$ ，则仅需对累积奖赏加权，即

$$Q(x, a) = \frac{1}{m} \sum_{i=1}^m \frac{P_i^\pi}{P_i^{\pi'}} r_i , \quad (16.26)$$

其中  $P_i^\pi$  和  $P_i^{\pi'}$  分别表示两个策略产生第  $i$  条轨迹的概率。对于给定的一条轨迹  $\langle x_0, a_0, r_1, \dots, x_{T-1}, a_{T-1}, r_T, x_T \rangle$ ，策略  $\pi$  产生该轨迹的概率为

$$P^\pi = \prod_{i=0}^{T-1} \pi(x_i, a_i) P_{x_i \rightarrow x_{i+1}}^{a_i} . \quad (16.27)$$

虽然这里用到了环境的转移概率  $P_{x_i \rightarrow x_{i+1}}^{a_i}$ , 但式(16.24)中实际只需两个策略概率的比值

$$\frac{P^\pi}{P^{\pi'}} = \prod_{i=0}^{T-1} \frac{\pi(x_i, a_i)}{\pi'(x_i, a_i)}. \quad (16.28)$$

若  $\pi$  为确定性策略而  $\pi'$  是  $\pi$  的  $\epsilon$ -贪心策略, 则  $\pi(x_i, a_i)$  始终为 1,  $\pi'(x_i, a_i)$  为  $\frac{\epsilon}{|A|}$  或  $1 - \epsilon + \frac{\epsilon}{|A|}$ , 于是就能对策略  $\pi$  进行评估了. 图 16.11 给出了“异策略”(off-policy) 蒙特卡罗强化学习算法的描述.

---

输入: 环境 $E$ ;	<b>过程:</b>
动作空间 $A$ ;	1: $Q(x, a) = 0$ , $\text{count}(x, a) = 0$ , $\pi(x, a) = \frac{1}{ A(x) }$ ;
起始状态 $x_0$ ;	2: <b>for</b> $s = 1, 2, \dots$ <b>do</b>
策略执行步数 $T$ .	3: 在 $E$ 中执行 $\pi$ 的 $\epsilon$ -贪心策略产生轨迹
	$< x_0, a_0, r_1, x_1, a_1, r_2, \dots, x_{T-1}, a_{T-1}, r_T, x_T >$ ;
重要性采样系数.	4: $p_i = \begin{cases} 1 - \epsilon + \epsilon/ A , & a_i = \pi(x); \\ \epsilon/ A , & a_i \neq \pi(x), \end{cases}$
计算修正的累积奖赏.	5: <b>for</b> $t = 0, 1, \dots, T-1$ <b>do</b>
式(16.2) 更新平均奖赏.	6: $R = \frac{1}{T-t} \sum_{i=t+1}^T (r_i \times \prod_{j=i}^{T-1} \frac{1}{p_j})$ ;
根据值函数得到策略.	7: $Q(x_t, a_t) = \frac{Q(x_t, a_t) \times \text{count}(x_t, a_t) + R}{\text{count}(x_t, a_t) + 1}$ ;
	8: $\text{count}(x_t, a_t) = \text{count}(x_t, a_t) + 1$
	9: <b>end for</b>
	10: $\pi(x) = \arg \max_{a'} Q(x, a')$
	11: <b>end for</b>
	输出: 策略 $\pi$

---

图 16.11 异策略蒙特卡罗强化学习算法

#### 16.4.2 时序差分学习

蒙特卡罗强化学习算法通过考虑采样轨迹, 克服了模型未知给策略估计造成的困难. 此类算法需在完成一个采样轨迹后再更新策略的值估计, 而前面介绍的基于动态规划的策略迭代和值迭代算法在每执行一步策略后就进行值函数更新. 两者相比, 蒙特卡罗强化学习算法的效率低得多, 这里的主要问题是蒙特卡罗强化学习算法没有充分利用强化学习任务的 MDP 结构. 时序差分(Temporal Difference, 简称 TD) 学习则结合了动态规划与蒙特卡罗方法的思想, 能做到更高效的免模型学习.

蒙特卡罗强化学习算法的本质, 是通过多次尝试后求平均来作为期望累

积奖赏的近似，但它在求平均时是“批处理式”进行的，即在一个完整的采样轨迹完成后再对所有的状态-动作对进行更新。实际上这个更新过程能增量式进行。对于状态-动作对  $(x, a)$ ，不妨假定基于  $t$  个采样已估计出值函数  $Q_t^\pi(x, a) = \frac{1}{t} \sum_{i=1}^t r_i$ ，则在得到第  $t+1$  个采样  $r_{t+1}$  时，类似式(16.3)，有

$$Q_{t+1}^\pi(x, a) = Q_t^\pi(x, a) + \frac{1}{t+1} (r_{t+1} - Q_t^\pi(x, a)). \quad (16.29)$$

显然，只需给  $Q_t^\pi(x, a)$  加上增量  $\frac{1}{t+1}(r_{t+1} - Q_t^\pi(x, a))$  即可。更一般的，将  $\frac{1}{t+1}$  替换为系数  $\alpha_{t+1}$ ，则可将增量项写作  $\alpha_{t+1}(r_{t+1} - Q_t^\pi(x, a))$ 。在实践中通常令  $\alpha_t$  为一个较小的正数值  $\alpha$ ，若将  $Q_t^\pi(x, a)$  展开为每步累积奖赏之和，则可看出系数之和为 1，即令  $\alpha_t = \alpha$  不会影响  $Q_t$  是累积奖赏之和这一性质。更新步长  $\alpha$  越大，则越靠后的累积奖赏越重要。

以  $\gamma$  折扣累积奖赏为例，利用动态规划方法且考虑到模型未知时使用状态-动作值函数更方便，由式(16.10)有

$$\begin{aligned} Q^\pi(x, a) &= \sum_{x' \in X} P_{x \rightarrow x'}^a (R_{x \rightarrow x'}^a + \gamma V^\pi(x')) \\ &= \sum_{x' \in X} P_{x \rightarrow x'}^a (R_{x \rightarrow x'}^a + \gamma \sum_{a' \in A} \pi(x', a') Q^\pi(x', a')). \end{aligned} \quad (16.30)$$

通过增量求和可得

$$Q_{t+1}^\pi(x, a) = Q_t^\pi(x, a) + \alpha (R_{x \rightarrow x'}^a + \gamma Q_t^\pi(x', a') - Q_t^\pi(x, a)), \quad (16.31)$$

其中  $x'$  是前一次在状态  $x$  执行动作  $a$  后转移到的状态， $a'$  是策略  $\pi$  在  $x'$  上选择的动作。

使用式(16.31)，每执行一步策略就更新一次值函数估计，于是得到图 16.12 的算法。该算法由于每次更新值函数需知道前一步的状态(state)、前一步的动作(action)、奖赏值(reward)、当前状态(state)、将要执行的动作(action)，由此得名为 Sarsa 算法 [Rummery and Niranjan, 1994]。显然，Sarsa 是一个同策略算法，算法中评估(第 6 行)、执行(第 5 行)的均为  $\epsilon$ -贪心策略。

将这几个英文单词的首字母连起来。

将 Sarsa 修改为异策略算法，则得到图 16.13 描述的 Q-学习(Q-learning)算法 [Watkins and Dayan, 1992]，该算法评估(第 6 行)的是  $\epsilon$ -贪心策略，而执行(第 5 行)的是原始策略。

默认均匀概率选取动作.

单步执行策略.

原始策略的 $\epsilon$ -贪心策略.

式(16.31)更新值函数.

---

输入: 环境  $E$ ;  
动作空间  $A$ ;  
起始状态  $x_0$ ;  
奖赏折扣  $\gamma$ ;  
更新步长  $\alpha$ .

过程:

- 1:  $Q(x, a) = 0, \pi(x, a) = \frac{1}{|A(x)|};$
- 2:  $x = x_0, a = \pi(x);$
- 3: **for**  $t = 1, 2, \dots$  **do**
- 4:    $r, x' =$  在  $E$  中执行动作  $a$  产生的奖赏与转移的状态;
- 5:    $a' = \pi^\epsilon(x');$
- 6:    $Q(x, a) = Q(x, a) + \alpha(r + \gamma Q(x', a') - Q(x, a));$
- 7:    $\pi(x) = \arg \max_{a''} Q(x, a'');$
- 8:    $x = x', a = a'$
- 9: **end for**

输出: 策略  $\pi$

---

图 16.12 Sarsa 算法

默认均匀概率选取动作.

单步执行策略.

原始策略.

式(16.31)更新值函数.

---

输入: 环境  $E$ ;  
动作空间  $A$ ;  
起始状态  $x_0$ ;  
奖赏折扣  $\gamma$ ;  
更新步长  $\alpha$ .

过程:

- 1:  $Q(x, a) = 0, \pi(x, a) = \frac{1}{|A(x)|};$
- 2:  $x = x_0;$
- 3: **for**  $t = 1, 2, \dots$  **do**
- 4:    $r, x' =$  在  $E$  中执行动作  $\pi^\epsilon(x)$  产生的奖赏与转移的状态;
- 5:    $a' = \pi(x');$
- 6:    $Q(x, a) = Q(x, a) + \alpha(r + \gamma Q(x', a') - Q(x, a));$
- 7:    $\pi(x) = \arg \max_{a''} Q(x, a'');$
- 8:    $x = x', a = a'$
- 9: **end for**

输出: 策略  $\pi$

---

图 16.13 Q-学习算法

## 16.5 值函数近似

前面我们一直假定强化学习任务是在有限状态空间上进行, 每个状态可用一个编号来指代; 值函数则是关于有限状态的“表格值函数”(tabular value function), 即值函数能表示为一个数组, 输入  $i$  对应的函数值就是数组元素  $i$  的值, 且更改一个状态上的值不会影响其他状态上的值. 然而, 现实强化学习任务

所面临的状态空间往往是连续的, 有无穷多个状态. 这该怎么办呢?

一个直接的想法是对状态空间进行离散化, 将连续状态空间转化为有限离散状态空间, 然后就能使用前面介绍的方法求解. 遗憾的是, 如何有效地对状态空间进行离散化是一个难题, 尤其是在对状态空间进行探索之前.

实际上, 我们不妨直接对连续状态空间的值函数进行学习. 假定状态空间为  $n$  维实数空间  $X = \mathbb{R}^n$ , 此时显然无法用表格值函数来记录状态值. 先考虑简单情形, 即值函数能表达为状态的线性函数 [Busoniu et al., 2010]

$$V_{\theta}(x) = \theta^T x, \quad (16.32)$$

其中  $x$  为状态向量,  $\theta$  为参数向量. 由于此时的值函数难以像有限状态那样精确记录每个状态的值, 因此这样值函数的求解被称为值函数近似 (value function approximation).

我们希望通过式(16.32)学得的值函数尽可能近似真实值函数  $V^\pi$ , 近似程度常用最小二乘误差来度量:

$$E_{\theta} = \mathbb{E}_{x \sim \pi} \left[ (V^\pi(x) - V_{\theta}(x))^2 \right], \quad (16.33)$$

其中  $\mathbb{E}_{x \sim \pi}$  表示由策略  $\pi$  所采样而得的状态上的期望.

为了使误差最小化, 采用梯度下降法, 对误差求负导数

$$\begin{aligned} -\frac{\partial E_{\theta}}{\partial \theta} &= \mathbb{E}_{x \sim \pi} \left[ 2(V^\pi(x) - V_{\theta}(x)) \frac{\partial V_{\theta}(x)}{\partial \theta} \right] \\ &= \mathbb{E}_{x \sim \pi} [2(V^\pi(x) - V_{\theta}(x)) x], \end{aligned} \quad (16.34)$$

于是可得到对于单个样本的更新规则

$$\theta = \theta + \alpha (V^\pi(x) - V_{\theta}(x)) x. \quad (16.35)$$

我们并不知道策略的真实值函数  $V^\pi$ , 但可借助时序差分学习, 基于  $V^\pi(x) = r + \gamma V^\pi(x')$  用当前估计的值函数代替真实值函数, 即

$$\begin{aligned} \theta &= \theta + \alpha (r + \gamma V_{\theta}(x') - V_{\theta}(x)) x \\ &= \theta + \alpha (r + \gamma \theta^T x' - \theta^T x) x, \end{aligned} \quad (16.36)$$

其中  $\mathbf{x}'$  是下一时刻的状态.

需注意的是, 在时序差分学习中需要状态-动作值函数以便获取策略. 这里一种简单做法是令  $\theta$  作用于表示状态和动作的联合向量上, 例如给状态向量增加一维用于存放动作编号, 即将式(16.32)中的  $\mathbf{x}$  替换为  $(\mathbf{x}; a)$ ; 另一种做法是用 0/1 对动作选择进行编码得到向量  $\mathbf{a} = (0; \dots; 1; \dots; 0)$ , 其中“1”表示该动作被选择, 再将状态向量与其合并得到  $(\mathbf{x}; \mathbf{a})$ , 用于替换式(16.32)中的  $\mathbf{x}$ . 这样就使得线性近似的对象为状态-动作值函数.

基于线性值函数近似来替代 Sarsa 算法中的值函数, 即可得到图 16.14 的线性值函数近似 Sarsa 算法. 类似地可得到线性值函数近似 Q-学习算法. 显然, 可以容易地用其他学习方法来代替式(16.32)中的线性学习器, 例如通过引入核方法实现非线性值函数近似.

核方法参见第 6 章.

---

**输入:** 环境  $E$ ;  
 动作空间  $A$ ;  
 起始状态  $x_0$ ;  
 奖赏折扣  $\gamma$ ;  
 更新步长  $\alpha$ .

**过程:**

- 1:  $\theta = \mathbf{0}$ ;
- 2:  $\mathbf{x} = x_0$ ,  $a = \pi(\mathbf{x}) = \arg \max_{a''} \theta^T(\mathbf{x}; a'')$ ;
- 3: **for**  $t = 1, 2, \dots$  **do**
- 4:    $r, \mathbf{x}'$  = 在  $E$  中执行动作  $a$  产生的奖赏与转移的状态;
- 5:    $a' = \pi^\epsilon(\mathbf{x}')$ ;
- 6:    $\theta = \theta + \alpha(r + \gamma \theta^T(\mathbf{x}'; a') - \theta^T(\mathbf{x}; a))(\mathbf{x}; a)$ ;
- 7:    $\pi(\mathbf{x}) = \arg \max_{a''} \theta^T(\mathbf{x}; a'')$ ;
- 8:    $\mathbf{x} = \mathbf{x}', a = a'$
- 9: **end for**

**输出:** 策略  $\pi$

---

图 16.14 线性值函数近似 Sarsa 算法

原始策略的  $\epsilon$ -贪心策略.  
 式(16.36)更新参数.

## 16.6 模仿学习

亦称“学徒学习”  
 (apprenticeship learning),  
 “示范学习”(learning  
 from demonstration), “观  
 察学习”(learning by  
 watching); 与机器学习早  
 期的“示教学习”有直接  
 联系, 参见 1.5 节.

在强化学习的经典任务设置中, 机器所能获得的反馈信息仅有决策后的累积奖赏, 但在现实任务中, 往往能得到人类专家的决策过程范例, 例如在种瓜任务上能得到农业专家的种植过程范例. 从这样的范例中学习, 称为“模仿学习”(imitation learning).

### 16.6.1 直接模仿学习

强化学习任务中多步决策的搜索空间巨大, 基于累积奖赏来学习很多步之前的合适决策非常困难, 而直接模仿人类专家的“状态-动作对”可显著缓解这一困难, 我们称其为“直接模仿学习”.

假定我们获得了一批人类专家的决策轨迹数据  $\{\tau_1, \tau_2, \dots, \tau_m\}$ , 每条轨迹包含状态和动作序列

$$\tau_i = \langle s_1^i, a_1^i, s_2^i, a_2^i, \dots, s_{n_i+1}^i \rangle,$$

其中  $n_i$  为第  $i$  条轨迹中的转移次数.

有了这样的数据, 就相当于告诉机器在什么状态下应选择什么动作, 于是可利用监督学习来学得符合人类专家决策轨迹数据的策略.

我们可将所有轨迹上的所有“状态-动作对”抽取出来, 构造出一个新的数据集合

$$D = \{(s_1, a_1), (s_2, a_2), \dots, (s_{\sum_{i=1}^m n_i}, a_{\sum_{i=1}^m n_i})\},$$

即把状态作为特征, 动作作为标记; 然后, 对这个新构造出的数据集合  $D$  使用分类(对于离散动作)或回归(对于连续动作)算法即可学得策略模型. 学得的这个策略模型可作为机器进行强化学习的初始策略, 再通过强化学习方法基于环境反馈进行改进, 从而获得更好的策略.

### 16.6.2 逆强化学习

在很多任务中, 设计奖赏函数往往相当困难, 从人类专家提供的范例数据中反推出奖赏函数有助于解决该问题, 这就是逆强化学习 (inverse reinforcement learning) [Abbeel and Ng, 2004].

在逆强化学习中, 我们知道状态空间  $X$ 、动作空间  $A$ , 并且与直接模仿学习类似, 有一个决策轨迹数据集  $\{\tau_1, \tau_2, \dots, \tau_m\}$ . 逆强化学习的基本思想是: 欲使机器做出与范例一致的行为, 等价于在某个奖赏函数的环境中求解最优策略, 该最优策略所产生的轨迹与范例数据一致. 换言之, 我们要寻找某种奖赏函数使得范例数据是最优的, 然后即可使用这个奖赏函数来训练强化学习策略.

不妨假设奖赏函数能表达为状态特征的线性函数, 即  $R(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ . 于是, 策略  $\pi$  的累积奖赏可写为

$$\rho^\pi = \mathbb{E} \left[ \sum_{t=0}^{+\infty} \gamma^t R(\mathbf{x}_t) \mid \pi \right] = \mathbb{E} \left[ \sum_{t=0}^{+\infty} \gamma^t \mathbf{w}^\top \mathbf{x}_t \mid \pi \right]$$

$$= \mathbf{w}^T \mathbb{E} \left[ \sum_{t=0}^{+\infty} \gamma^t \mathbf{x}_t \mid \pi \right], \quad (16.37)$$

即状态向量加权和的期望与系数  $\mathbf{w}$  的内积.

将状态向量的期望  $\mathbb{E} [\sum_{t=0}^{+\infty} \gamma^t \mathbf{x}_t \mid \pi]$  简写为  $\bar{\mathbf{x}}^\pi$ . 注意到获得  $\bar{\mathbf{x}}^\pi$  需求取期望. 我们可使用蒙特卡罗方法通过采样来近似期望, 而范例轨迹数据集恰可看作最优策略的一个采样, 于是, 可将每条范例轨迹上的状态加权求和再平均, 记为  $\bar{\mathbf{x}}^*$ . 对于最优奖赏函数  $R(\mathbf{x}) = \mathbf{w}^{*T} \mathbf{x}$  和任意其他策略产生的  $\bar{\mathbf{x}}^\pi$ , 有

$$\mathbf{w}^{*T} \bar{\mathbf{x}}^* - \mathbf{w}^{*T} \bar{\mathbf{x}}^\pi = \mathbf{w}^{*T} (\bar{\mathbf{x}}^* - \bar{\mathbf{x}}^\pi) \geq 0. \quad (16.38)$$

若能对所有策略计算出  $(\bar{\mathbf{x}}^* - \bar{\mathbf{x}}^\pi)$ , 即可解出

$$\begin{aligned} \mathbf{w}^* &= \arg \max_{\mathbf{w}} \min_{\pi} \mathbf{w}^T (\bar{\mathbf{x}}^* - \bar{\mathbf{x}}^\pi) \\ \text{s.t. } &\|\mathbf{w}\| \leq 1 \end{aligned} \quad (16.39)$$

显然, 我们难以获得所有策略, 一个较好的办法是从随机策略开始, 迭代地求解更好的奖赏函数, 基于奖赏函数获得更好的策略, 直至最终获得最符合范例轨迹数据集的奖赏函数和策略, 如图 16.15 算法所示. 注意在求解更好的奖赏函数时, 需将式(16.39)中对所有策略求最小改为对之前学得的策略求最小.

---

**输入:** 环境  $E$ ;  
状态空间  $X$ ;  
动作空间  $A$ ;  
范例轨迹数据集  $D = \{\tau_1, \tau_2, \dots, \tau_m\}$ .

**过程:**

- 1:  $\bar{\mathbf{x}}^* =$  从范例轨迹中算出状态加权和的均值向量;
- 2:  $\pi =$  随机策略;
- 3: **for**  $t = 1, 2, \dots$  **do**
- 4:    $\bar{\mathbf{x}}_t^\pi =$  从  $\pi$  的采样轨迹算出状态加权和的均值向量;
- 5:   求解  $\mathbf{w}^* = \arg \max_{\mathbf{w}} \min_{i=1}^t \mathbf{w}^T (\bar{\mathbf{x}}^* - \bar{\mathbf{x}}_i^\pi)$  s.t.  $\|\mathbf{w}\| \leq 1$ ;
- 6:    $\pi =$  在环境  $\langle X, A, R(\mathbf{x}) = \mathbf{w}^{*T} \mathbf{x} \rangle$  中求解最优策略;
- 7: **end for**

**输出:** 奖赏函数  $R(\mathbf{x}) = \mathbf{w}^{*T} \mathbf{x}$  与策略  $\pi$

---

图 16.15 迭代式逆强化学习算法

## 16.7 阅读材料

强化学习专门书籍中最著名的是 [Sutton and Barto, 1998]. [Gosavi, 2003] 从优化的角度来讨论强化学习, [Whiteson, 2010] 则侧重于介绍基于演化算法搜索的强化学习方法. [Mausam and Kolobov, 2012] 从马尔可夫决策过程的视角介绍强化学习, [Sigaud and Buffet, 2010] 覆盖了很多内容, 包括本章未介绍的部分可观察马尔可夫决策过程 (Partially Observable MDP, 简称 POMDP)、策略梯度法等. 基于值函数近似的强化学习可参阅 [Busoniu et al., 2010].

欧洲强化学习研讨会(EWRL)是专门性的强化学习系列研讨会, 多学科强化学习与决策会议(RLDM)则是从 2013 年开始的新会议.

[Kaelbling et al., 1996] 是一个较早的强化学习综述, [Kober et al., 2013; Deisenroth et al., 2013] 则综述了强化学习在机器人领域的应用.

[Kuleshov and Precup, 2000] 和 [Vermorel and Mohri, 2005] 介绍了多种  $K$ -摇臂赌博机算法并进行了比较. 多摇臂赌博机模型在统计学领域有大量研究 [Berry and Fristedt, 1985], 近年来在“在线学习”(online learning)、“对抗学习”(adversarial learning) 等方面有广泛应用, [Bubeck and Cesa-Bianchi, 2012] 对其“悔界”(regret bound)分析方面的结果进行了综述.

时序差分(TD)学习最早是 A. Samuel 在他著名的跳棋工作中提出, [Sutton, 1988] 提出了 TD( $\lambda$ ) 算法, 由于 [Tesauro, 1995] 基于 TD( $\lambda$ ) 研制的 TD-Gammon 程序在西洋双陆棋上达到人类世界冠军水平而使 TD 学习备受关注. Q-学习算法是 [Watkins and Dayan, 1992] 提出, Sarsa 则是在 Q-学习算法基础上的改进 [Rummery and Niranjan, 1994]. TD 学习近年来仍有改进和推广, 例如广义 TD 学习 [Ueno et al., 2011]、使用资格迹(eligibility traces)的 TD 学习 [Geist and Scherrer, 2014] 等. [Dann et al., 2014] 对 TD 学习中的策略评估方法进行了比较.

模仿学习被认为是强化学习提速的重要手段 [Lin, 1992; Price and Boutiller, 2003], 在机器人领域被广泛使用 [Argall et al., 2009]. [Abbeel and Ng, 2004; Langford and Zadrozny, 2005] 提出了逆强化学习方法.

在运筹学与控制论领域, 强化学习方面的研究被称为“近似动态规划”(approximate dynamic programming), 可参阅 [Bertsekas, 2012].

“后悔”(regret)是指在不确定性条件下的决策与确定性条件下的决策所获得的奖赏间的差别.

Samuel 跳棋工作参见 p.22.

## 习题

- 16.1** 用于  $K$ -摇臂赌博机的 UCB (Upper Confidence Bound, 上置信界) 方法每次选择  $Q(k) + UC(k)$  最大的摇臂, 其中  $Q(k)$  为摇臂  $k$  当前的平均奖赏,  $UC(k)$  为置信区间. 例如

$$Q(k) + \sqrt{\frac{2 \ln n}{n_k}},$$

其中  $n$  为已执行所有摇臂的总次数,  $n_k$  为已执行摇臂  $k$  的次数. 试比较 UCB 方法与  $\epsilon$ -贪心法和 Softmax 方法的异同.

- 16.2** 借鉴图 16.7, 试写出基于  $\gamma$  折扣奖赏函数的策略评估算法.
- 16.3** 借鉴图 16.8, 试写出基于  $\gamma$  折扣奖赏函数的策略迭代算法.
- 16.4** 在没有 MDP 模型时, 可以先学习 MDP 模型(例如使用随机策略进行采样, 从样本中估计出转移函数和奖赏函数), 然后再使用有模型强化学习方法. 试述该方法与免模型强化学习方法的优缺点.
- 16.5** 试推导出 Sarsa 算法的更新公式(16.31).
- 16.6** 试借鉴图 16.14 给出线性值函数近似 Q-学习算法.
- 16.7** 线性值函数近似在实践中往往有较大误差. 试结合 BP 神经网络, 将线性值函数近似 Sarsa 算法推广为使用神经网络近似的 Sarsa 算法.
- 16.8** 试结合核方法, 将线性值函数近似 Sarsa 算法推广为使用核函数的非线性值函数近似 Sarsa 算法.
- 16.9** 对于目标驱动(goal-directed)的强化学习任务, 目标是到达某一状态, 例如将汽车驾驶到预定位置. 试为这样的任务设置奖赏函数, 并讨论不同奖赏函数的作用(例如每一步未达目标的奖赏为 0、-1 或 1).
- 16.10\*** 与传统监督学习不同, 直接模仿学习在不同时刻所面临的数据分布可能不同. 试设计一个考虑不同时刻数据分布变化的模仿学习算法.

## 参考文献

- Abbeel, P. and A. Y. Ng. (2004). "Apprenticeship learning via inverse reinforcement learning." In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, Banff, Canada.
- Argall, B. D., S. Chernova, M. Veloso, and B. Browning. (2009). "A survey of robot learning from demonstration." *Robotics and Autonomous Systems*, 57(5):469–483.
- Berry, D. and B. Fristedt. (1985). *Bandit Problems*. Chapman & Hall/CRC, London, UK.
- Bertsekas, D. P. (2012). *Dynamic Programming and Optimal Control: Approximate Dynamic Programming*, 4th edition. Athena Scientific, Nashua, NH.
- Bubeck, S. and N. Cesa-Bianchi. (2012). "Regret analysis of stochastic and nonstochastic multi-armed bandit problems." *Foundations and Trends in Machine Learning*, 5(1):1–122.
- Busoniu, L., R. Babuska, B. De Schutter, and D. Ernst. (2010). *Reinforcement Learning and Dynamic Programming Using Function Approximators*. Chapman & Hall/CRC Press, Boca Raton, FL.
- Dann, C., G. Neumann, and J. Peters. (2014). "Policy evaluation with temporal differences: A survey and comparison." *Journal of Machine Learning Research*, 15:809–883.
- Deisenroth, M. P., G. Neumann, and J. Peters. (2013). "A survey on policy search for robotics." *Foundations and Trends in Robotics*, 2(1-2):1–142.
- Geist, M. and B. Scherrer. (2014). "Off-policy learning with eligibility traces: A survey." *Journal of Machine Learning Research*, 15:289–333.
- Gosavi, A. (2003). *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*. Kluwer, Norwell, MA.
- Kaelbling, L. P., M. L. Littman, and A. W. Moore. (1996). "Reinforcement learning: A survey." *Journal of Artificial Intelligence Research*, 4:237–285.
- Kober, J., J. A. Bagnell, and J. Peters. (2013). "Reinforcement learning in robotics: A survey." *International Journal on Robotics Research*, 32(11):1238–1274.
- Kuleshov, V. and D. Precup. (2000). "Algorithms for the multi-armed bandit

- problem.” *Journal of Machine Learning Research*, 1:1–48.
- Langford, J. and B. Zadrozny. (2005). “Relating reinforcement learning performance to classification performance.” In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, 473–480, Bonn, Germany.
- Lin, L.-J. (1992). “Self-improving reactive agents based on reinforcement learning, planning and teaching.” *Machine Learning*, 8(3-4):293–321.
- Mausam and A. Kolobov. (2012). *Planning with Markov Decision Processes: An AI Perspective*. Morgan & Claypool, San Rafael, CA.
- Price, B. and C. Boutilier. (2003). “Accelerating reinforcement learning through implicit imitation.” *Journal of Artificial Intelligence Research*, 19: 569–629.
- Rummery, G. A. and M. Niranjan. (1994). “On-line Q-learning using connectionist systems.” Technical Report CUED/F-INFENG/TR 166, Engineering Department, Cambridge University, Cambridge, UK.
- Sigaud, O. and O. Buffet. (2010). *Markov Decision Processes in Artificial Intelligence*. Wiley, Hoboken, NJ.
- Sutton, R. S. (1988). “Learning to predict by the methods of temporal differences.” *Machine Learning*, 3(1):9–44.
- Sutton, R. S. and A. G. Barto. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.
- Tesauro, G. (1995). “Temporal difference learning and TD-Gammon.” *Communications of the ACM*, 38(3):58–68.
- Ueno, T., S. Maeda, M. Kawanabe, and S. Ishii. (2011). “Generalized TD learning.” *Journal of Machine Learning Research*, 12:1977–2020.
- Vermorel, J. and M. Mohri. (2005). “Multi-armed bandit algorithms and empirical evaluation.” In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, 437–448, Porto, Portugal.
- Watkins, C. J. C. H. and P. Dayan. (1992). “Q-learning.” *Machine Learning*, 8 (3-4):279–292.
- Whiteson, S. (2010). *Adaptive Representations for Reinforcement Learning*. Springer, Berlin.

## 休息一会儿

### 小故事：马尔可夫决策过程与安德烈·马尔可夫

安德烈·安德烈维奇·马尔可夫(Andrey Andreyevich Markov, 1856—1922)是著名俄罗斯数学家、圣彼得堡数学学派代表性人物，在概率论、数论、函数逼近论、微分方程等方面有重要贡献。



马尔可夫出生在莫斯科东南的梁赞(Ryazan), 17岁时独立发现了一种线性常微分方程的解法，引起了圣彼得堡大学几位数学家的注意。1874年他考入圣彼得堡大学数学系，1878年毕业并留校任教，1884年获博士学位，导师是圣彼得堡学派领袖、著名数学家切比雪夫。此后马尔可夫一直在圣彼得堡大学任教。马尔可夫在早期主要是沿着切比雪夫开创的方向，改进和完善了大数定律和中心极限定理，但他最重要的工作无疑是开辟了随机过程这个领域。他在1906—1912年间提出了马尔可夫链，开创了对马尔可夫过程的研究。现实世界里小到分子的布朗运动、大到传染病流行过程，马尔可夫过程几乎无所不在。在他的名著《概率演算》中，马尔可夫是以普希金的长诗《叶甫根尼·奥涅金》中元、辅音字母变化的规律为例来展示马尔可夫链的性质。马尔可夫决策过程是马尔可夫过程与确定性动态规划的结合，基本思想在二十世纪五十年代出现，此时马尔可夫已去世三十多年了。

切比雪夫在圣彼得堡大学培养出马尔可夫、李亚普诺夫、柯尔金、格拉维等著名数学家，还影响了圣彼得堡大学之外的很多数学家。圣彼得堡学派标志着俄罗斯数学走到了世界前沿。

马尔可夫的儿子也叫安德烈·安德烈维奇·马尔可夫(1903—1979)，也是著名数学家，数理逻辑中的“马尔可夫原则”(Markov Principle)、“马尔可夫规则”(Markov Rule)，理论计算机科学中图灵完备的“马尔可夫算法”等，是以小马尔可夫的名字命名的。马尔可夫的弟弟弗拉基米尔·安德烈维奇·马尔可夫(1871—1897)也是一位数学家，“马尔可夫兄弟不等式”就是以他和哥哥安德烈的名字命名的。



# 附录

## A 矩阵

### A.1 基本演算

记实矩阵  $\mathbf{A} \in \mathbb{R}^{m \times n}$  第  $i$  行第  $j$  列的元素为  $(\mathbf{A})_{ij} = A_{ij}$ . 矩阵  $\mathbf{A}$  的转置(transpose)记为  $\mathbf{A}^T$ ,  $(\mathbf{A}^T)_{ij} = A_{ji}$ . 显然,

$$(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T, \quad (\text{A.1})$$

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T. \quad (\text{A.2})$$

对于矩阵  $\mathbf{A} \in \mathbb{R}^{m \times n}$ , 若  $m = n$  则称为  $n$  阶方阵. 用  $\mathbf{I}_n$  表示  $n$  阶单位阵, 方阵常直接用  $\mathbf{I}$  表示单位阵.  $\mathbf{A}$  的逆矩阵  $\mathbf{A}^{-1}$  满足  $\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ . 不难发现,

$$(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T, \quad (\text{A.3})$$

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1} \mathbf{A}^{-1}. \quad (\text{A.4})$$

对于  $n$  阶方阵  $\mathbf{A}$ , 它的迹(trace)是主对角线上的元素之和, 即  $\text{tr}(\mathbf{A}) = \sum_{i=1}^n A_{ii}$ . 迹有如下性质:

$$\text{tr}(\mathbf{A}^T) = \text{tr}(\mathbf{A}), \quad (\text{A.5})$$

$$\text{tr}(\mathbf{A} + \mathbf{B}) = \text{tr}(\mathbf{A}) + \text{tr}(\mathbf{B}), \quad (\text{A.6})$$

$$\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA}), \quad (\text{A.7})$$

$$\text{tr}(\mathbf{ABC}) = \text{tr}(\mathbf{BCA}) = \text{tr}(\mathbf{CAB}). \quad (\text{A.8})$$

$n$  阶方阵  $\mathbf{A}$  的行列式(determinant)定义为

$$\det(\mathbf{A}) = \sum_{\sigma \in S_n} \text{par}(\sigma) A_{1\sigma_1} A_{2\sigma_2} \dots A_{n\sigma_n}, \quad (\text{A.9})$$

其中  $S_n$  为所有  $n$  阶排列(permuation)的集合,  $\text{par}(\sigma)$  的值为  $-1$  或  $+1$  取决于  $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$  为奇排列或偶排列, 即其中出现降序的次数为奇数或偶

数, 例如  $(1, 3, 2)$  中降序次数为 1,  $(1, 4, 3, 2)$  中降序次数为 2. 对于单位阵, 有  $\det(\mathbf{I}) = 1$ . 对于 2 阶方阵, 有

$$\det(\mathbf{A}) = \det \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = A_{11}A_{22} - A_{12}A_{21}.$$

$n$  阶方阵  $\mathbf{A}$  的行列式有如下性质:

$$\det(c\mathbf{A}) = c^n \det(\mathbf{A}), \quad (\text{A.10})$$

$$\det(\mathbf{A}^T) = \det(\mathbf{A}), \quad (\text{A.11})$$

$$\det(\mathbf{AB}) = \det(\mathbf{A}) \det(\mathbf{B}), \quad (\text{A.12})$$

$$\det(\mathbf{A}^{-1}) = \det(\mathbf{A})^{-1}, \quad (\text{A.13})$$

$$\det(\mathbf{A}^n) = \det(\mathbf{A})^n. \quad (\text{A.14})$$

矩阵  $\mathbf{A} \in \mathbb{R}^{m \times n}$  的 Frobenius 范数定义为

$$\|\mathbf{A}\|_F = (\text{tr}(\mathbf{A}^T \mathbf{A}))^{1/2} = \left( \sum_{i=1}^m \sum_{j=1}^n A_{ij}^2 \right)^{1/2}. \quad (\text{A.15})$$

容易看出, 矩阵的 Frobenius 范数就是将矩阵张成向量后的  $L_2$  范数.

## A.2 导数

向量  $\mathbf{a}$  相对于标量  $x$  的导数(derivative), 以及  $x$  相对于  $\mathbf{a}$  的导数都是向量, 其第  $i$  个分量分别为

$$\left( \frac{\partial \mathbf{a}}{\partial x} \right)_i = \frac{\partial a_i}{\partial x}, \quad (\text{A.16})$$

$$\left( \frac{\partial x}{\partial \mathbf{a}} \right)_i = \frac{\partial x}{\partial a_i}. \quad (\text{A.17})$$

类似的, 矩阵  $\mathbf{A}$  对于标量  $x$  的导数, 以及  $x$  对于  $\mathbf{A}$  的导数都是矩阵, 其第  $i$  行第  $j$  列上的元素分别为

$$\left( \frac{\partial \mathbf{A}}{\partial x} \right)_{ij} = \frac{\partial A_{ij}}{\partial x}, \quad (\text{A.18})$$

$$\left( \frac{\partial \mathbf{x}}{\partial \mathbf{A}} \right)_{ij} = \frac{\partial \mathbf{x}}{\partial A_{ij}} . \quad (\text{A.19})$$

对于函数  $f(\mathbf{x})$ , 假定其对向量的元素可导, 则  $f(\mathbf{x})$  关于  $\mathbf{x}$  的一阶导数是一个向量, 其第  $i$  个分量为

$$(\nabla f(\mathbf{x}))_i = \frac{\partial f(\mathbf{x})}{\partial x_i} , \quad (\text{A.20})$$

$f(\mathbf{x})$  关于  $\mathbf{x}$  的二阶导数是称为海森矩阵(Hessian matrix)的一个方阵, 其第  $i$  行第  $j$  列上的元素为

$$(\nabla^2 f(\mathbf{x}))_{ij} = \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} . \quad (\text{A.21})$$

向量和矩阵的导数满足乘法法则(product rule)

$$\mathbf{a} \text{ 相对于 } \mathbf{x} \text{ 为常向量.} \quad \frac{\partial \mathbf{x}^T \mathbf{a}}{\partial \mathbf{x}} = \frac{\partial \mathbf{a}^T \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a} , \quad (\text{A.22})$$

$$\frac{\partial \mathbf{AB}}{\partial \mathbf{x}} = \frac{\partial \mathbf{A}}{\partial \mathbf{x}} \mathbf{B} + \mathbf{A} \frac{\partial \mathbf{B}}{\partial \mathbf{x}} . \quad (\text{A.23})$$

由  $\mathbf{A}^{-1} \mathbf{A} = \mathbf{I}$  和式(A.23), 逆矩阵的导数可表示为

$$\frac{\partial \mathbf{A}^{-1}}{\partial \mathbf{x}} = -\mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial \mathbf{x}} \mathbf{A}^{-1} . \quad (\text{A.24})$$

若求导的标量是矩阵  $\mathbf{A}$  的元素, 则有

$$\frac{\partial \text{tr}(\mathbf{AB})}{\partial A_{ij}} = B_{ji} , \quad (\text{A.25})$$

$$\frac{\partial \text{tr}(\mathbf{AB})}{\partial \mathbf{A}} = \mathbf{B}^T . \quad (\text{A.26})$$

进而有

$$\frac{\partial \text{tr}(\mathbf{A}^T \mathbf{B})}{\partial \mathbf{A}} = \mathbf{B} , \quad (\text{A.27})$$

$$\frac{\partial \text{tr}(\mathbf{A})}{\partial \mathbf{A}} = \mathbf{I} , \quad (\text{A.28})$$

$$\frac{\partial \text{tr}(\mathbf{ABA}^T)}{\partial \mathbf{A}} = \mathbf{A}(\mathbf{B} + \mathbf{B}^T) . \quad (\text{A.29})$$

由式(A.15)和(A.29)有

$$\frac{\partial \|\mathbf{A}\|_F^2}{\partial \mathbf{A}} = \frac{\partial \text{tr}(\mathbf{A}\mathbf{A}^T)}{\partial \mathbf{A}} = 2\mathbf{A}. \quad (\text{A.30})$$

链式法则(chain rule)是计算复杂导数时的重要工具。简单地说，若函数  $f$  是  $g$  和  $h$  的复合，即  $f(x) = g(h(x))$ ，则有

$$\frac{\partial f(x)}{\partial x} = \frac{\partial g(h(x))}{\partial h(x)} \cdot \frac{\partial h(x)}{\partial x}. \quad (\text{A.31})$$

例如在计算下式时，将  $\mathbf{Ax} - \mathbf{b}$  看作一个整体可简化计算：

$$\begin{aligned} \frac{\partial}{\partial \mathbf{x}} (\mathbf{Ax} - \mathbf{b})^T \mathbf{W} (\mathbf{Ax} - \mathbf{b}) &= \frac{\partial (\mathbf{Ax} - \mathbf{b})}{\partial \mathbf{x}} \cdot 2\mathbf{W} (\mathbf{Ax} - \mathbf{b}) \\ &= 2\mathbf{AW} (\mathbf{Ax} - \mathbf{b}). \end{aligned} \quad (\text{A.32})$$

### A.3 奇异值分解

任意实矩阵  $\mathbf{A} \in \mathbb{R}^{m \times n}$  都可分解为

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T, \quad (\text{A.33})$$

其中， $\mathbf{U} \in \mathbb{R}^{m \times m}$  是满足  $\mathbf{U}^T \mathbf{U} = \mathbf{I}$  的  $m$  阶酉矩阵(unitary matrix);  $\mathbf{V} \in \mathbb{R}^{n \times n}$  是满足  $\mathbf{V}^T \mathbf{V} = \mathbf{I}$  的  $n$  阶酉矩阵； $\Sigma \in \mathbb{R}^{m \times n}$  是  $m \times n$  的矩阵，其中  $(\Sigma)_{ii} = \sigma_i$  且其他位置的元素均为 0， $\sigma_i$  为非负实数且满足  $\sigma_1 \geq \sigma_2 \geq \dots \geq 0$ 。

常将奇异值按降序排列以确保  $\Sigma$  的唯一性。

当  $\mathbf{A}$  为对称正定矩阵时，奇异值分解与特征值分解结果相同。

式(A.33)中的分解称为奇异值分解(Singular Value Decomposition, 简称 SVD)，其中  $\mathbf{U}$  的列向量  $\mathbf{u}_i \in \mathbb{R}^m$  称为  $\mathbf{A}$  的左奇异向量(left-singular vector)， $\mathbf{V}$  的列向量  $\mathbf{v}_i \in \mathbb{R}^n$  称为  $\mathbf{A}$  的右奇异向量(right-singular vector)， $\sigma_i$  称为奇异值(singular value)。矩阵  $\mathbf{A}$  的秩(rank)就等于非零奇异值的个数。

奇异值分解有广泛的用途，例如对于低秩矩阵近似(low-rank matrix approximation)问题，给定一个秩为  $r$  的矩阵  $\mathbf{A}$ ，欲求其最优  $k$  秩近似矩阵  $\tilde{\mathbf{A}}$ ， $k \leq r$ ，该问题可形式化为

$$\begin{aligned} \min_{\tilde{\mathbf{A}} \in \mathbb{R}^{m \times n}} \quad & \|\mathbf{A} - \tilde{\mathbf{A}}\|_F \\ \text{s.t.} \quad & \text{rank}(\tilde{\mathbf{A}}) = k. \end{aligned} \quad (\text{A.34})$$

奇异值分解提供了上述问题的解析解: 对矩阵  $\mathbf{A}$  进行奇异值分解后, 将矩阵  $\Sigma$  中的  $r - k$  个最小的奇异值置零获得矩阵  $\Sigma_k$ , 即仅保留最大的  $k$  个奇异值, 则

$$\mathbf{A}_k = \mathbf{U}_k \Sigma_k \mathbf{V}_k^T \quad (\text{A.35})$$

就是式(A.34)的最优解, 其中  $\mathbf{U}_k$  和  $\mathbf{V}_k$  分别是式(A.33)中的前  $k$  列组成的矩阵. 这个结果称为 Eckart-Young-Mirsky 定理.

## B 优化

### B.1 拉格朗日乘子法

拉格朗日乘子法(Lagrange multipliers)是一种寻找多元函数在一组约束下的极值的方法. 通过引入拉格朗日乘子, 可将有  $d$  个变量与  $k$  个约束条件的最优化问题转化为具有  $d + k$  个变量的无约束优化问题求解.

先考虑一个等式约束的优化问题. 假定  $\mathbf{x}$  为  $d$  维向量, 欲寻找  $\mathbf{x}$  的某个取值  $\mathbf{x}^*$ , 使目标函数  $f(\mathbf{x})$  最小且同时满足  $g(\mathbf{x}) = 0$  的约束. 从几何角度看, 该问题的目标是在由方程  $g(\mathbf{x}) = 0$  确定的  $d - 1$  维曲面上寻找能使目标函数  $f(\mathbf{x})$  最小化的点. 此时不难得到如下结论:

函数等值线与约束曲面相切.

可通过反证法证明: 若梯度  $\nabla f(\mathbf{x}^*)$  与约束曲面不正交, 则仍可在约束曲面上移动该点使函数值进一步下降.

- 对于约束曲面上的任意点  $\mathbf{x}$ , 该点的梯度  $\nabla g(\mathbf{x})$  正交于约束曲面;
- 在最优点  $\mathbf{x}^*$ , 目标函数在该点的梯度  $\nabla f(\mathbf{x}^*)$  正交于约束曲面.

由此可知, 在最优点  $\mathbf{x}^*$ , 如附图B.1 所示, 梯度  $\nabla g(\mathbf{x})$  和  $\nabla f(\mathbf{x})$  的方向必相同或相反, 即存在  $\lambda \neq 0$  使得

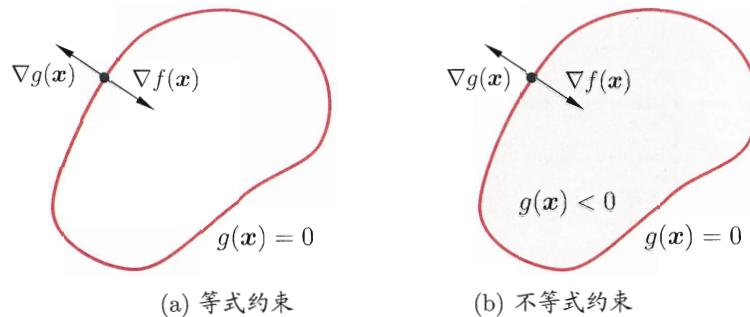
$$\nabla f(\mathbf{x}^*) + \lambda \nabla g(\mathbf{x}^*) = 0, \quad (\text{B.1})$$

对等式约束,  $\lambda$  可能为正也可能为负.

$\lambda$  称为拉格朗日乘子. 定义拉格朗日函数

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x}), \quad (\text{B.2})$$

不难发现, 将其对  $\mathbf{x}$  的偏导数  $\nabla_{\mathbf{x}} L(\mathbf{x}, \lambda)$  置零即得式(B.1), 同时, 将其对  $\lambda$  的偏导数  $\nabla_{\lambda} L(\mathbf{x}, \lambda)$  置零即得约束条件  $g(\mathbf{x}) = 0$ . 于是, 原约束优化问题可转化为对拉格朗日函数  $L(\mathbf{x}, \lambda)$  的无约束优化问题.



**附图B. 1** 拉格朗日乘子法的几何含义: 在 (a) 等式约束  $g(\mathbf{x}) = 0$  或 (b) 不等式约束  $g(\mathbf{x}) \leq 0$  下, 最小化目标函数  $f(\mathbf{x})$ . 红色曲线表示  $g(\mathbf{x}) = 0$  构成的曲面, 而其围成的阴影区域表示  $g(\mathbf{x}) < 0$ .

现在考虑不等式约束  $g(\mathbf{x}) \leq 0$ , 如附图B. 1 所示, 此时最优点  $\mathbf{x}^*$  或在  $g(\mathbf{x}) < 0$  的区域中, 或在边界  $g(\mathbf{x}) = 0$  上. 对于  $g(\mathbf{x}) < 0$  的情形, 约束  $g(\mathbf{x}) \leq 0$  不起作用, 可直接通过条件  $\nabla f(\mathbf{x}) = 0$  来获得最优点; 这等价于将  $\lambda$  置零然后对  $\nabla_{\mathbf{x}} L(\mathbf{x}, \lambda)$  置零得到最优点.  $g(\mathbf{x}) = 0$  的情形类似于上面等式约束的分析, 但需注意的是, 此时  $\nabla f(\mathbf{x}^*)$  的方向必与  $\nabla g(\mathbf{x}^*)$  相反, 即存在常数  $\lambda > 0$  使得  $\nabla f(\mathbf{x}^*) + \lambda \nabla g(\mathbf{x}^*) = 0$ . 整合这两种情形, 必满足  $\lambda g(\mathbf{x}) = 0$ . 因此, 在约束  $g(\mathbf{x}) \leq 0$  下最小化  $f(\mathbf{x})$ , 可转化为在如下约束下最小化式(B.2)的拉格朗日函数:

$$\begin{cases} g(\mathbf{x}) \leq 0; \\ \lambda \geq 0; \\ \mu_j g_j(\mathbf{x}) = 0. \end{cases} \quad (\text{B.3})$$

式(B.3)称为 Karush-Kuhn-Tucker (简称KKT)条件.

上述做法可推广到多个约束. 考虑具有  $m$  个等式约束和  $n$  个不等式约束, 且可行域  $\mathbb{D} \subset \mathbb{R}^d$  非空的优化问题

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & h_i(\mathbf{x}) = 0 \quad (i = 1, \dots, m), \\ & g_j(\mathbf{x}) \leq 0 \quad (j = 1, \dots, n). \end{aligned} \quad (\text{B.4})$$

引入拉格朗日乘子  $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \dots, \lambda_m)^T$  和  $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_n)^T$ , 相应的拉格

朗日函数为

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^m \lambda_i h_i(\boldsymbol{x}) + \sum_{j=1}^n \mu_j g_j(\boldsymbol{x}), \quad (\text{B.5})$$

由不等式约束引入的 KKT 条件 ( $j = 1, 2, \dots, n$ ) 为

$$\begin{cases} g_j(\boldsymbol{x}) \leq 0; \\ \mu_j \geq 0; \\ \mu_j g_j(\boldsymbol{x}) = 0. \end{cases} \quad (\text{B.6})$$

一个优化问题可以从两个角度来考察, 即“主问题”(primal problem)和“对偶问题”(dual problem). 对主问题(B.4), 基于式(B.5), 其拉格朗日“对偶函数”(dual function)  $\Gamma : \mathbb{R}^m \times \mathbb{R}^n \mapsto \mathbb{R}$  定义为

在推导对偶问题时, 常通过将拉格朗日乘子  $L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$  对  $\boldsymbol{x}$  求导并令导数为 0, 来获得对偶函数的表达形式.

$$\begin{aligned} \Gamma(\boldsymbol{\lambda}, \boldsymbol{\mu}) &= \inf_{\boldsymbol{x} \in \mathbb{D}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\ &= \inf_{\boldsymbol{x} \in \mathbb{D}} \left( f(\boldsymbol{x}) + \sum_{i=1}^m \lambda_i h_i(\boldsymbol{x}) + \sum_{j=1}^n \mu_j g_j(\boldsymbol{x}) \right). \end{aligned} \quad (\text{B.7})$$

$\boldsymbol{\mu} \succeq 0$  表示  $\boldsymbol{\mu}$  的分量均为非负.

若  $\tilde{\boldsymbol{x}} \in \mathbb{D}$  为主问题(B.4)可行域中的点, 则对任意  $\boldsymbol{\mu} \succeq 0$  和  $\boldsymbol{\lambda}$  都有

$$\sum_{i=1}^m \lambda_i h_i(\tilde{\boldsymbol{x}}) + \sum_{j=1}^n \mu_j g_j(\tilde{\boldsymbol{x}}) \leq 0, \quad (\text{B.8})$$

进而有

$$\Gamma(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\boldsymbol{x} \in \mathbb{D}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq L(\tilde{\boldsymbol{x}}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq f(\tilde{\boldsymbol{x}}). \quad (\text{B.9})$$

若主问题(B.4)的最优值为  $p^*$ , 则对任意  $\boldsymbol{\mu} \succeq 0$  和  $\boldsymbol{\lambda}$  都有

$$\Gamma(\boldsymbol{\lambda}, \boldsymbol{\mu}) \leq p^*, \quad (\text{B.10})$$

即对偶函数给出了主问题最优值的下界. 显然, 这个下界取决于  $\boldsymbol{\mu}$  和  $\boldsymbol{\lambda}$  的值. 于是, 一个很自然的问题是: 基于对偶函数能获得的最好下界是什么? 这就引出了优化问题

$$\max_{\lambda, \mu} \Gamma(\lambda, \mu) \text{ s.t. } \mu \succeq 0. \quad (\text{B.11})$$

式(B.11)就是主问题(B.4)的对偶问题, 其中  $\lambda$  和  $\mu$  称为“对偶变量”(dual variable). 无论主问题(B.4)的凸性如何, 对偶问题(B.11)始终是凸优化问题.

考虑式(B.11)的最优值  $d^*$ , 显然有  $d^* \leq p^*$ , 这称为“弱对偶性”(weak duality)成立; 若  $d^* = p^*$ , 则称为“强对偶性”(strong duality)成立, 此时由对偶问题能获得主问题的最优下界. 对于一般的优化问题, 强对偶性通常不成立. 但是, 若主问题为凸优化问题, 如式(B.4)中  $f(\mathbf{x})$  和  $g_j(\mathbf{x})$  均为凸函数,  $h_i(\mathbf{x})$  为仿射函数, 且其可行域中至少有一点使不等式约束严格成立, 则此时强对偶性成立. 值得注意的是, 在强对偶性成立时, 将拉格朗日函数分别对原变量和对偶变量求导, 再并令导数等于零, 即可得到原变量与对偶变量的数值关系. 于是, 对偶问题解决了, 主问题也就解决了.

这称为 Slater 条件.

## B.2 二次规划

二次规划(Quadratic Programming, 简称 QP)是一类典型的优化问题, 包括凸二次优化和非凸二次优化. 在此类问题中, 目标函数是变量的二次函数, 而约束条件是变量的线性不等式.

非标准二次规划问题中可以包含等式约束. 注意到等式约束能用两个不等式约束来代替; 不等式约束可通过增加松弛变量的方式转化为等式约束.

假定变量个数为  $d$ , 约束条件的个数为  $m$ , 则标准的二次规划问题形如

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A} \mathbf{x} \leq \mathbf{b}, \end{aligned} \quad (\text{B.12})$$

其中  $\mathbf{x}$  为  $d$  维向量,  $\mathbf{Q} \in \mathbb{R}^{d \times d}$  为实对称矩阵,  $\mathbf{A} \in \mathbb{R}^{m \times d}$  为实矩阵,  $\mathbf{b} \in \mathbb{R}^m$  和  $\mathbf{c} \in \mathbb{R}^d$  为实向量,  $\mathbf{A} \mathbf{x} \leq \mathbf{b}$  的每一行对应一个约束.

若  $\mathbf{Q}$  为半正定矩阵, 则式(B.12)目标函数是凸函数, 相应的二次规划是凸二次优化问题; 此时若约束条件  $\mathbf{A} \mathbf{x} \leq \mathbf{b}$  定义的可行域不为空, 且目标函数在此可行域有下界, 则该问题将有全局最小值. 若  $\mathbf{Q}$  为正定矩阵, 则该问题有唯一的全局最小值. 若  $\mathbf{Q}$  为非正定矩阵, 则式(B.12)是有多个平稳点和局部极小点的 NP 难问题.

常用的二次规划解法有椭球法(ellipsoid method)、内点法(interior point)、增广拉格朗日法(augmented Lagrangian)、梯度投影法(gradiant projection) 等. 若  $\mathbf{Q}$  为正定矩阵, 则相应的二次规划问题可由椭球法在多项式时间内求解.

### B.3 半正定规划

半正定规划(Semi-Definite Programming, 简称SDP)是一类凸优化问题, 其中的变量可组织成半正定对称矩阵形式, 且优化问题的目标函数和约束都是这些变量的线性函数.

给定  $d \times d$  的对称矩阵  $\mathbf{X}, \mathbf{C}$ ,

$$\mathbf{C} \cdot \mathbf{X} = \sum_{i=1}^d \sum_{j=1}^d C_{ij} X_{ij}, \quad (\text{B.13})$$

若  $\mathbf{A}_i (i = 1, 2, \dots, m)$  也是  $d \times d$  的对称矩阵,  $b_i (i = 1, 2, \dots, m)$  为  $m$  个实数, 则半正定规划问题形如

$$\begin{aligned} & \min_{\mathbf{X}} \quad \mathbf{C} \cdot \mathbf{X} \\ \text{s.t. } & \mathbf{A}_i \cdot \mathbf{X} = b_i, \quad i = 1, 2, \dots, m \\ & \mathbf{X} \succeq 0. \end{aligned} \quad (\text{B.14})$$

$\mathbf{X} \succeq 0$  表示  $\mathbf{X}$  半正定.

半正定规划与线性规划都拥有线性的目标函数和约束, 但半正定规划中的约束  $\mathbf{X} \succeq 0$  是一个非线性、非光滑约束条件. 在优化理论中, 半正定规划具有一定的一般性, 能将几种标准的优化问题(如线性规划、二次规划)统一起来.

常见的用于求解线性规划的内点法经过少许改造即可求解半正定规划问题, 但半正定规划的计算复杂度较高, 难以直接用于大规模问题.

### B.4 梯度下降法

一阶方法仅使用目标函数的一阶导数, 不利用其高阶导数.

梯度下降法(gradient descent)是一种常用的一阶(first-order)优化方法, 是求解无约束优化问题最简单、最经典的方法之一.

考虑无约束优化问题  $\min_{\mathbf{x}} f(\mathbf{x})$ , 其中  $f(\mathbf{x})$  为连续可微函数. 若能构造一个序列  $\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \dots$  满足

$$f(\mathbf{x}^{t+1}) < f(\mathbf{x}^t), \quad t = 0, 1, 2, \dots \quad (\text{B.15})$$

则不断执行该过程即可收敛到局部极小点. 欲满足式(B.15), 根据泰勒展式有

$$f(\mathbf{x} + \Delta \mathbf{x}) \simeq f(\mathbf{x}) + \Delta \mathbf{x}^T \nabla f(\mathbf{x}), \quad (\text{B.16})$$

于是, 欲满足  $f(\mathbf{x} + \Delta\mathbf{x}) < f(\mathbf{x})$ , 可选择

$$\Delta\mathbf{x} = -\gamma \nabla f(\mathbf{x}), \quad (\text{B.17})$$

每步的步长  $\gamma_t$  可不同.

*L-Lipschitz* 条件是指对于任意  $\mathbf{x}$ , 存在常数  $L$  使得  $\|\nabla f(\mathbf{x})\| \leq L$  成立.

其中步长  $\gamma$  是一个小常数. 这就是梯度下降法.

若目标函数  $f(\mathbf{x})$  满足一些条件, 则通过选取合适的步长, 就能确保通过梯度下降收敛到局部极小点. 例如若  $f(\mathbf{x})$  满足 *L-Lipschitz* 条件, 则将步长设置为  $1/(2L)$  即可确保收敛到局部极小点. 当目标函数为凸函数时, 局部极小点就对应着函数的全局最小点, 此时梯度下降法可确保收敛到全局最优解.

当目标函数  $f(\mathbf{x})$  二阶连续可微时, 可将式(B.16)替换为更精确的二阶泰勒展式, 这样就得到了牛顿法(Newton's method). 牛顿法是典型的二阶方法, 其迭代轮数远小于梯度下降法. 但牛顿法使用了二阶导数  $\nabla^2 f(\mathbf{x})$ , 其每轮迭代中涉及到海森矩阵(A.21)的求逆, 计算复杂度相当高, 尤其在高维问题中几乎不可行. 若能以较低的计算代价寻找海森矩阵的近似逆矩阵, 则可显著降低计算开销, 这就是拟牛顿法(quasi-Newton method).

## B.5 坐标下降法

求解极大值问题时亦称“坐标上升法”(coordinate ascent).

坐标下降法(coordinate descent)是一种非梯度优化方法, 它在每步迭代中沿一个坐标方向进行搜索, 通过循环使用不同的坐标方向来达到目标函数的局部极小值.

不妨假设目标是求解函数  $f(\mathbf{x})$  的极小值, 其中  $\mathbf{x} = (x_1, x_2, \dots, x_d)^T \in \mathbb{R}^d$  是一个  $d$  维向量. 从初始点  $\mathbf{x}^0$  开始, 坐标下降法通过迭代地构造序列  $\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \dots$  来求解该问题,  $\mathbf{x}^{t+1}$  的第  $i$  个分量  $x_i^{t+1}$  构造为

$$x_i^{t+1} = \arg \min_{y \in \mathbb{R}} f(x_1^{t+1}, \dots, x_{i-1}^{t+1}, y, x_{i+1}^t, \dots, x_d^t). \quad (\text{B.18})$$

通过执行此操作, 显然有

$$f(\mathbf{x}^0) \geq f(\mathbf{x}^1) \geq f(\mathbf{x}^2) \geq \dots \quad (\text{B.19})$$

与梯度下降法类似, 通过迭代执行该过程, 序列  $\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \dots$  能收敛到所期望的局部极小点或驻点(stationary point).

坐标下降法不需计算目标函数的梯度, 在每步迭代中仅需求解一维搜索问题, 对于某些复杂问题计算较为简便. 但若目标函数不光滑, 则坐标下降法有可能陷入非驻点(non-stationary point).

## C 概率分布

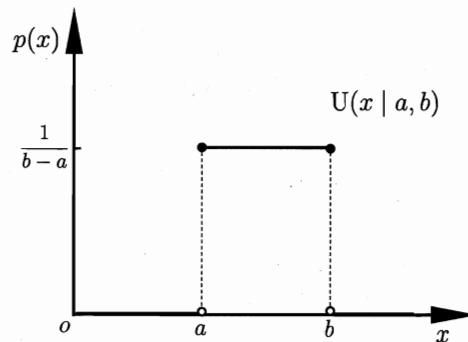
### C.1 常见概率分布

本节简要介绍几种常见概率分布. 对于每种分布, 我们将给出概率密度函数以及期望  $\mathbb{E}[\cdot]$ 、方差  $\text{var}[\cdot]$  和协方差  $\text{cov}[\cdot, \cdot]$  等几个主要的统计量.

#### C.1.1 均匀分布

这里仅介绍连续均匀分布.

均匀分布(uniform distribution)是关于定义在区间  $[a, b] (a < b)$  上连续变量的简单概率分布, 其概率密度函数如附图C.1所示.



附图C.1 均匀分布的概率密度函数

$$p(x | a, b) = U(x | a, b) = \frac{1}{b-a}; \quad (\text{C.1})$$

$$\mathbb{E}[x] = \frac{a+b}{2}; \quad (\text{C.2})$$

$$\text{var}[x] = \frac{(b-a)^2}{12}. \quad (\text{C.3})$$

不难发现, 若变量  $x$  服从均匀分布  $U(x | 0, 1)$  且  $a < b$ , 则  $a + (b - a)x$  服从均匀分布  $U(x | a, b)$ .

#### C.1.2 伯努利分布

以瑞士数学家雅各布·伯努利 (Jacob Bernoulli, 1654–1705) 的名字命名.

伯努利分布(Bernoulli distribution)是关于布尔变量  $x \in \{0, 1\}$  的概率分布, 其连续参数  $\mu \in [0, 1]$  表示变量  $x = 1$  的概率.

$$P(x | \mu) = \text{Bern}(x | \mu) = \mu^x(1 - \mu)^{1-x}; \quad (\text{C.4})$$

$$\mathbb{E}[x] = \mu ; \quad (C.5)$$

$$\text{var}[x] = \mu(1 - \mu) . \quad (C.6)$$

### C.1.3 二项分布

二项分布(binomial distribution)用以描述  $N$  次独立的伯努利实验中有  $m$  次成功(即  $x = 1$ )的概率, 其中每次伯努利实验成功的概率为  $\mu \in [0, 1]$ .

$$P(m | N, \mu) = \text{Bin}(m | N, \mu) = \binom{N}{m} \mu^m (1 - \mu)^{N-m} ; \quad (C.7)$$

$$\mathbb{E}[x] = N\mu ; \quad (C.8)$$

$$\text{var}[x] = N\mu(1 - \mu) . \quad (C.9)$$

对于参数  $\mu$ , 二项分布的共轭先验分布是贝塔分布. 共轭分布参见 C.2.

当  $N = 1$  时, 二项分布退化为伯努利分布.

### C.1.4 多项分布

若将伯努利分布由单变量扩展为  $d$  维向量  $\mathbf{x}$ , 其中  $x_i \in \{0, 1\}$  且  $\sum_{i=1}^d x_i = 1$ , 并假设  $x_i$  取 1 的概率为  $\mu_i \in [0, 1]$ ,  $\sum_{i=1}^d \mu_i = 1$ , 则将得到离散概率分布

$$P(\mathbf{x} | \boldsymbol{\mu}) = \prod_{i=1}^d \mu_i^{x_i} ; \quad (C.10)$$

$$\mathbb{E}[x_i] = \mu_i ; \quad (C.11)$$

$$\text{var}[x_i] = \mu_i(1 - \mu_i) ; \quad (C.12)$$

$$\text{cov}[x_j, x_i] = \mathbb{I}[j = i] \mu_i . \quad (C.13)$$

在此基础上扩展二项分布则得到多项分布(multinomial distribution), 它描述了在  $N$  次独立实验中有  $m_1$  次  $x_1 = 1$  的概率.

对于参数  $\mu$ , 多项分布的共轭先验分布是狄利克雷分布. 共轭分布参见 C.2.

$$P(m_1, m_2, \dots, m_d | N, \boldsymbol{\mu}) = \text{Mult}(m_1, m_2, \dots, m_d | N, \boldsymbol{\mu})$$

$$= \frac{N!}{m_1! m_2! \dots m_d!} \prod_{i=1}^d \mu_i^{m_i} ; \quad (C.14)$$

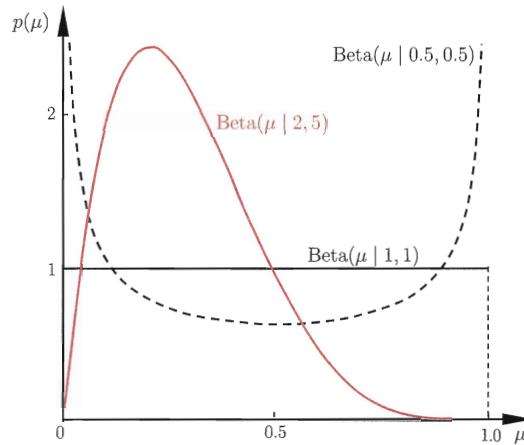
$$\mathbb{E}[m_i] = N\mu_i ; \quad (C.15)$$

$$\text{var}[m_i] = N\mu_i(1 - \mu_i) ; \quad (\text{C.16})$$

$$\text{cov}[m_j, m_i] = -N\mu_j\mu_i . \quad (\text{C.17})$$

### C.1.5 贝塔分布

贝塔分布(Beta distribution)是关于连续变量  $\mu \in [0, 1]$  的概率分布, 它由两个参数  $a > 0$  和  $b > 0$  确定, 其概率密度函数如附图C.2 所示.



附图C.2 贝塔分布的概率密度函数

$$\begin{aligned} p(\mu | a, b) &= \text{Beta}(\mu | a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1} \\ &= \frac{1}{B(a,b)}\mu^{a-1}(1-\mu)^{b-1} ; \end{aligned} \quad (\text{C.18})$$

$$\mathbb{E}[\mu] = \frac{a}{a+b} ; \quad (\text{C.19})$$

$$\text{var}[\mu] = \frac{ab}{(a+b)^2(a+b+1)} , \quad (\text{C.20})$$

其中  $\Gamma(a)$  为 Gamma 函数

$$\Gamma(a) = \int_0^{+\infty} t^{a-1}e^{-t}dt , \quad (\text{C.21})$$

$B(a, b)$  为 Beta 函数

$$B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} . \quad (\text{C.22})$$

当  $a = b = 1$  时, 贝塔分布退化为均匀分布.

### C.1.6 狄利克雷分布

以德国数学家狄利克雷(1805—1859)的名字命名.

狄利克雷分布(Dirichlet distribution)是关于一组  $d$  个连续变量  $\mu_i \in [0, 1]$  的概率分布,  $\sum_{i=1}^d \mu_i = 1$ . 令  $\boldsymbol{\mu} = (\mu_1; \mu_2; \dots; \mu_d)$ , 参数  $\boldsymbol{\alpha} = (\alpha_1; \alpha_2; \dots; \alpha_d)$ ,  $\alpha_i > 0$ ,  $\hat{\alpha} = \sum_{i=1}^d \alpha_i$ .

$$p(\boldsymbol{\mu} | \boldsymbol{\alpha}) = \text{Dir}(\boldsymbol{\mu} | \boldsymbol{\alpha}) = \frac{\Gamma(\hat{\alpha})}{\Gamma(\alpha_1) \dots \Gamma(\alpha_d)} \prod_{i=1}^d \mu_i^{\alpha_i-1}; \quad (\text{C.23})$$

$$\mathbb{E}[\mu_i] = \frac{\alpha_i}{\hat{\alpha}}; \quad (\text{C.24})$$

$$\text{var}[\mu_i] = \frac{\alpha_i(\hat{\alpha} - \alpha_i)}{\hat{\alpha}^2(\hat{\alpha} + 1)}; \quad (\text{C.25})$$

$$\text{cov}[\mu_j, \mu_i] = \frac{\alpha_j \alpha_i}{\hat{\alpha}^2(\hat{\alpha} + 1)}. \quad (\text{C.26})$$

当  $d = 2$  时, 狄利克雷分布退化为贝塔分布.

### C.1.7 高斯分布

高斯分布(Gaussian distribution)亦称正态分布(normal distribution), 是应用最为广泛的连续概率分布.

对于单变量  $x \in (-\infty, \infty)$ , 高斯分布的参数为均值  $\mu \in (-\infty, \infty)$  和方差  $\sigma^2 > 0$ . 附图 C.3 给出了在几组不同参数下高斯分布的概率密度函数.

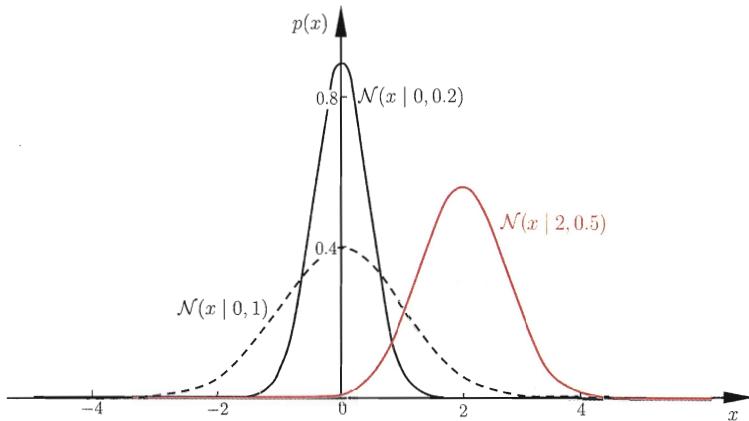
$$p(x | \mu, \sigma^2) = \mathcal{N}(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}; \quad (\text{C.27})$$

$$\mathbb{E}[x] = \mu; \quad (\text{C.28})$$

$$\text{var}[x] = \sigma^2. \quad (\text{C.29})$$

对于  $d$  维向量  $\mathbf{x}$ , 多元高斯分布的参数为  $d$  维均值向量  $\boldsymbol{\mu}$  和  $d \times d$  的对称正定协方差矩阵  $\boldsymbol{\Sigma}$ .

$$\begin{aligned} p(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ &= \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}; \end{aligned} \quad (\text{C.30})$$



附图C.3 高斯分布的概率密度函数

$$\mathbb{E}[x] = \mu ; \quad (C.31)$$

$$\text{cov}[x] = \Sigma . \quad (C.32)$$

## C.2 共轭分布

假设变量  $x$  服从分布  $P(x | \Theta)$ , 其中  $\Theta$  为参数,  $X = \{x_1, x_2, \dots, x_m\}$  为变量  $x$  的观测样本, 假设参数  $\Theta$  服从先验分布  $\Pi(\Theta)$ . 若由先验分布  $\Pi(\Theta)$  和抽样分布  $P(X | \Theta)$  决定的后验分布  $F(\Theta | X)$  与  $\Pi(\Theta)$  是同种类型的分布, 则称先验分布  $\Pi(\Theta)$  为分布  $P(x | \Theta)$  或  $P(X | \Theta)$  的共轭分布(conjugate distribution).

例如, 假设  $x \sim \text{Bern}(x | \mu)$ ,  $X = \{x_1, x_2, \dots, x_m\}$  为观测样本,  $\bar{x}$  为观测样本的均值,  $\mu \sim \text{Beta}(\mu | a, b)$ , 其中  $a, b$  为已知参数, 则  $\mu$  的后验分布

$$\begin{aligned} F(\mu | X) &\propto \text{Beta}(\mu | a, b) P(X | \mu) \\ &= \frac{\mu^{a-1} (1-\mu)^{b-1}}{B(a, b)} \mu^{m\bar{x}} (1-\mu)^{m-m\bar{x}} \\ &= \frac{1}{B(a+m\bar{x}, b+m-m\bar{x})} \mu^{a+m\bar{x}-1} (1-\mu)^{b+m-m\bar{x}-1} \\ &= \text{Beta}(\mu | a', b') , \end{aligned} \quad (C.33)$$

亦为贝塔分布, 其中  $a' = a + m\bar{x}$ ,  $b' = b + m - m\bar{x}$ , 这意味着贝塔分布与伯努利分布共轭. 类似可知, 多项分布的共轭分布是狄利克雷分布, 而高斯分布的共轭分布仍是高斯分布.

这里仅考虑高斯分布方差已知、均值服从先验的情形.

先验分布反映了某种先验信息, 后验分布既反映了先验分布提供的信息、又反映了样本提供的信息。当先验分布与抽样分布共轭时, 后验分布与先验分布属于同种类型, 这意味着先验信息与样本提供的信息具有某种同一性。于是, 若使用后验分布作为进一步抽样的先验分布, 则新的后验分布仍将属于同种类型。因此, 共轭分布在不少情形下会使问题得以简化。例如在式(C.33)的例子中, 对服从伯努利分布的事件  $X$  使用贝塔先验分布, 则贝塔分布的参数值  $a$  和  $b$  可视为对伯努利分布的真实情况(事件发生和不发生)的预估。随着“证据”(样本)的不断到来, 贝塔分布的参数值从  $a, b$  变化为  $a + m\bar{x}, b + m - m\bar{x}$ , 且  $a/(a + b)$  将随着  $m$  的增大趋近于伯努利分布的真实参数值  $\bar{x}$ 。显然, 使用共轭先验之后, 只需调整  $a$  和  $b$  这两个预估值即可方便地进行模型更新。

### C.3 KL散度

KL散度(Kullback-Leibler divergence), 亦称相对熵(relative entropy)或信息散度(information divergence), 可用于度量两个概率分布之间的差异。给定两个概率分布  $P$  和  $Q$ , 二者之间的KL散度定义为

这里假设两个分布均为连续型概率分布; 对于离散型概率分布, 只需将定义中的积分替换为对所有离散值遍历求和。

$$\text{KL}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx, \quad (\text{C.34})$$

其中  $p(x)$  和  $q(x)$  分别为  $P$  和  $Q$  的概率密度函数。

KL散度满足非负性, 即

$$\text{KL}(P\|Q) \geq 0, \quad (\text{C.35})$$

当且仅当  $P = Q$  时  $\text{KL}(P\|Q) = 0$ 。但是, KL散度不满足对称性, 即

$$\text{KL}(P\|Q) \neq \text{KL}(Q\|P), \quad (\text{C.36})$$

度量应满足四个基本性质, 参见9.3节。

因此, KL散度不是一个度量(metric)。

若将KL散度的定义(C.34)展开, 可得

$$\begin{aligned} \text{KL}(P\|Q) &= \int_{-\infty}^{\infty} p(x) \log p(x) dx - \int_{-\infty}^{\infty} p(x) \log q(x) dx \\ &= -H(P) + H(P, Q), \end{aligned} \quad (\text{C.37})$$

其中  $H(P)$  为熵(entropy),  $H(P, Q)$  为  $P$  和  $Q$  的交叉熵(cross entropy)。在信

息论中, 熵  $H(P)$  表示对来自  $P$  的随机变量进行编码所需的最小字节数, 而交叉熵  $H(P, Q)$  则表示使用基于  $Q$  的编码对来自  $P$  的变量进行编码所需的字节数. 因此, KL 散度可认为是使用基于  $Q$  的编码对来自  $P$  的变量进行编码所需的“额外”字节数; 显然, 额外字节数必然非负, 当且仅当  $P = Q$  时额外字节数为零.



## 后记

写作本书的主因，是 2016 年准备在南京大学开设“机器学习”课。十五年前笔者曾主张开设此课，但那时国内对机器学习闻之不多，不少人听到这个名字的第一反应是“学习什么机器？”学校估计学生兴趣不大，于是笔者开设了“数据挖掘”这门名字听上去就觉得很有用的课。被评为省优秀研究生课程后，又给本科生单开了一门“数据挖掘导论”。这两门课很受欢迎，选修学生很多，包括不少外来蹭听生。虽然课上有一多半其实在讲机器学习，但笔者仍一直希望专开一门机器学习课，因笔者以为机器学习迟早会变成计算机学科的基础内容。

图灵奖得主 E. W. Dijkstra 曾说“计算机科学并不仅是关于计算机，就像天文学并不仅是关于望远镜”。正如天文学早期的研究关注如何制造望远镜，计算机科学早期研究是在关注如何令计算机运转。到了今天，建造强大的天文望远镜虽仍重要，但天文学更要紧的是“用”望远镜来开展研究。类似地，计算机科学发展至今，也该到了从关注“造”计算机转入更关注“用”计算机来认识和改造世界的阶段，其中最重要的无疑是用计算机对数据进行分析，因为这是计算的主要目的，而这就离不开机器学习。十多年前在国内某次重要论坛上笔者刚抛出此观点就被专家迎头痛斥，但今日来看，甚至很多计算机学科外人士都已对机器学习的重大价值津津乐道，现在才开设机器学习基础课似乎已有点嫌晚了。

1995 年在南大图书馆偶然翻看了《机器学习：通往人工智能的途径》，这算是笔者接触机器学习的开始。那时机器学习在国内问津者寥寥，甚至连科研人员申请基金项目也无合适代码方向可报。周边无专家可求教，又因国内科研经费匮乏而几无国际交流，加之学校尚无互联网和电子文献库，能看到的最新文献仅是两年前出版且页数不全的某 IEEE 汇刊……可谓举步维艰，经历的困惑和陷阱不可胜数。笔者切身体会到，入门阶段接触的书籍是何等重要，对自学者尤甚。一本好书能让人少走许多弯路，材料不佳则后续要花费数倍精力方能纠偏。中文书当然要国人自己来写。虽已不需靠“写书出名”，且深知写教科书极耗时间精力，但踌躇后笔者仍决定动手写这本书，唯望为初学者略尽绵薄之力。

有人说“一千个人眼中就有一千个哈姆雷特”，一个学科何尝不是如此。之所以不欲使用市面上流行的教科书（主要是英文的），除了觉得对大多数中国学生来说中文教科书更便于学习，另一个原因则是希望从笔者自己的视角来展现机器学习。

2013 年中开始规划提纲，由此进入了焦躁的两年。该写哪些内容、先写什么后写什么、从哪个角度写、写到什么程度，总有千丝万缕需考虑。及至写作进行，更是战战兢兢，深恐不慎误人子弟。写书难，写教科书更难。两年下来，甘苦自知。子曰：“取乎其上，得乎其中；取乎其中，得乎其下”，且以顶级的态度，出一本勉强入得方家法眼之书。

本书贯穿以西瓜为例,一则因为瓜果中笔者尤喜西瓜,二则因为西瓜在笔者所生活的区域有个有趣的蕴义。朋友小聚、请客吃饭,菜已全而主未知,或馔未齐而人待走,都挺尴尬。于是聪明人发明了“潜规则”:席终上西瓜。无论整盘抑或小碟,宾主见瓜至,则心领神会准备起身,皆大欢喜。久而久之,无论菜肴价格贵贱、场所雅鄙,宴必有西瓜。若将宴席比作(未来)应用系统,菜肴比作所涉技术,则机器学习好似那必有的西瓜,它可能不是最“高大上”的,但却是离不了的、没用上总觉得不甘心的。

本书写作过程从材料搜集,到习题设计,再到阅读校勘,都得到了笔者的很多学生、同事和学术界朋友的支持和帮助,在此谨列出他们的姓名以致谢意(姓氏拼音序):陈松灿,戴望州,高阳,高尉,黄圣君,黎铭,李楠,李武军,李宇峰,钱超,王魏,王威廉,吴建鑫,徐淼,俞扬,詹德川,张利军,张敏灵,朱军。书稿在 LAMDA 组学生 2015 年暑期讨论班上试讲,高斌斌、郭翔宇、李绍园、钱鸿、沈芷玉、叶翰嘉、张腾等同学又帮助发现了许多笔误。特别感谢李楠把笔者简陋的手绘图转变为精致的插图,俞扬帮助调整排版格式和索引,刘冲把笔者对封面设计的想法具体表现出来。

中国计算机学会终身成就奖得主、中国科学院院士陆汝钤先生是我国人工智能事业的开拓者之一,他在 1988 年和 1996 年出版的《人工智能》(上、下册)曾给予笔者很多启发。承蒙陆老师厚爱在百忙中为本书作序,不胜惶恐之至。陆老师在序言中提出的问题很值得读者在本书之后的进阶学习与研究中深思。

感谢清华大学出版社薛慧老师为本书出版所做的努力。十二年前笔者入选国家杰出青年科学基金时薛老师即邀著书,笔者以年纪尚轻、学力未逮婉辞。十年前“机器学习及其应用”研讨会(MLA)从陆汝钤院士肇始的复旦大学智能信息处理重点实验室移师南京,参会人数从复旦最初的 20 人,发展到 2010 年 400 余人,此后在清华、复旦、西电达 800 余人,今年再回南大竟至 1300 余人,场面热烈。MLA 倡导“学术至上、其余从简”,不搞繁文缛节,参会免费。但即便如此,仍有很多感兴趣的师生因旅费不菲而难以参加。于是笔者提议每两年以《机器学习及其应用》为题出版一本报告选集以飨读者。这个主意得到了薛老师、陆老师以及和笔者一起长期组织 MLA、去年因病去世的王珏老师的大力支持。此类专业性学术文集销量不大,出版社多半要贴钱。笔者曾跟薛老师说,自著的第一本中文书必交由薛老师在清华出版,或可稍为出版社找补。转眼《机器学习及其应用》系列已出到第六本,薛老师或以为十年前是玩笑话,某日告之书快完稿时她蓦然惊喜。

最后要感谢笔者的家人,本书几乎耗尽了两年来笔者所有的节假日和空闲时间。写作时垂髫犬子常跑来案边,不是问“爸爸去哪儿?”而是看几眼然后问“爸爸你又写了几页?”为了给他满意的答复,笔者埋头努力。

周志华  
2015 年 11 月于南京渐宽斋

# 索引

- 0/1损失函数, 130, 147  
 $5 \times 2$  交叉验证, 41  
 $\epsilon$ -贪心, 374
- AdaBoost, 173  
ART网络, 108
- Bagging, 178  
Bellman等式, 380  
Boltzmann分布, 111  
Boltzmann机, 111  
Boosting, 173, 190  
BP算法, 101  
BP网络, 101
- C4.5决策树, 78, 83  
CART决策树, 79
- ECOC, 64  
Elman网络, 111  
EM算法, 162, 208, 295, 335
- $F1$ , 32  
Fisher判别分析, 60  
Friedman检验, 42  
Frobenius 范数, 400
- Hoeffding不等式, 192, 268  
hinge损失, 130
- ID3决策树, 75  
ILP, 357, 364
- Jensen不等式, 268
- $K$ -摇臂赌博机, 373  
KKT条件, 124, 132, 135  
KL散度, 335, 414  
Kohonen网络, 109  
 $k$  折交叉验证, 26  
 $k$ 近邻, 225  
 $k$ 均值算法, 202, 218
- $L_1$ 正则化, 253  
 $L_2$ 正则化, 253
- LASSO, 252, 261  
Lipschitz条件, 253  
LVQ, 204, 218
- M-P神经元模型, 97  
McDiarmid不等式, 268  
MCMC, 331  
McNemar检验, 41  
MDP, 371  
Mercer定理, 137, 139  
MH算法, 333  
MvM, 63
- Nemenyi后续检验, 43
- OvO, 63  
OvR, 63
- P-R曲线, 31  
PAC辨识, 269  
PAC可学习, 269  
PAC学习算法, 270  
PCA, 229
- Q-学习, 387, 393
- Rademacher复杂度, 279  
RBF网络, 108  
ReLU, 114  
RIPPER, 353  
RKHS, 128  
ROC曲线, 33, 46
- S3VM, 298  
Sarsa 算法, 387, 390  
Sigmoid函数, 58, 98, 102  
Softmax, 375  
SOM网络, 109  
Stacking, 184  
SVM, 123
- TD学习, 386, 393  
Tikhonov正则化, 252
- VC维, 273, 274  
V型结构, 158
- WEKA, 16

- 奥卡姆剃刀, 7, 17  
版本空间, 5  
半监督聚类, 240, 307  
半监督学习, 293, 294  
半监督支持向量机, 298  
半朴素贝叶斯分类器, 154  
半正定规划, 407  
包裹式特征选择, 250  
包外估计, 27, 179  
贝塔分布, 411  
贝叶斯定理, 148  
贝叶斯分类器, 164  
贝叶斯风险, 147  
贝叶斯决策论, 147  
贝叶斯模型平均, 185  
贝叶斯网, 156, 319, 339  
贝叶斯学习, 164  
贝叶斯最优分类器, 147  
本真低维空间, 232  
本真距离, 234  
必连约束, 239, 307  
边际独立性, 158  
边际分布, 328  
边际化, 158, 328  
边际似然, 163  
编码矩阵, 65  
变分推断, 334  
变量消去, 328  
标记, 2  
标记传播, 302  
标记空间, 3  
表格值函数, 388  
表示定理, 137  
表示学习, 114  
伯努利分布, 409  
不可分, 269, 272  
不可知PAC可学习, 273  
不一致, 269  
参数估计, 54  
参数调节, 28  
参数空间, 106  
策略, 372  
策略迭代, 381  
测地线距离, 234  
测试, 3  
测试样本, 3  
层次聚类, 214  
查全率, 30  
查询, 293  
查准率, 30  
差异性度量, 187  
超父, 155  
成对马尔可夫性, 325  
冲突消解, 348  
重采样, 177  
重赋权, 177  
簇, 3, 197  
错误率, 23, 29  
打散, 273  
带序规则, 348  
代价, 35, 47  
代价矩阵, 35  
代价敏感, 36, 67  
代价曲线, 36  
单隐层网络, 101  
道德图, 158  
等度量映射, 234  
低密度分隔, 298  
低维嵌入, 226  
低秩矩阵近似, 402  
狄利克雷分布, 412  
递归神经网络, 111  
典型相关分析, 240  
独立同分布, 3, 267  
独依赖估计, 154  
度量学习, 237  
端正图, 158  
对比散度, 112  
对分, 273  
对率函数, 58  
对率回归, 58, 132, 325  
对率损失, 130  
对偶函数, 405  
对偶问题, 123, 405  
对数几率函数, 58, 98  
对数几率回归, 57  
对数似然, 59, 149  
对数线性回归, 56  
多变量决策树, 88, 92  
多标记学习, 68  
多层前馈神经网络, 100  
多分类, 3  
多分类器系统, 171  
多分类学习, 63  
多核学习, 140  
多视图学习, 240, 304  
多维缩放, 227

- 多项分布, 410  
多样性, 172  
多样性度量, 187  
多元线性回归, 55  
二次规划, 406  
二分类, 3  
二项分布, 410  
二项检验, 38  
发散, 113  
罚函数法, 133  
反向传播算法, 101  
泛化, 3, 121, 350  
泛化误差, 23, 267  
非参数化方法, 340  
非度量距离, 201  
非线性降维, 232  
非线性可分, 99  
分层采样, 25  
分而治之, 74  
分类, 3  
分歧, 185, 304  
风险, 147  
符号主义, 10, 363  
概率近似正确, 268  
概率模型, 206, 319  
概率图模型, 156, 319  
概念类, 268  
概念学习, 4, 17  
感知机, 98  
高斯分布, 412  
高斯核, 128  
高斯混合, 206, 296  
割平面法, 139  
个体学习器, 171  
功能神经元, 99  
共轭分布, 413  
关系学习, 363  
广义  $\delta$  规则, 115  
广义瑞利商, 61  
广义线性模型, 57  
规范化, 36, 183  
规则, 347  
规则学习, 347  
归结商, 362  
归纳, 359  
归纳逻辑程序设计, 357, 364  
归纳偏好, 6  
归纳学习, 4, 11  
归一化, 36  
过采样, 67  
过滤式特征选择, 249  
过拟合, 23, 104, 191, 352  
过配, 23  
行列式, 399  
豪斯多夫距离, 220  
核范数, 260  
核方法, 137  
核函数, 126  
核化, 137, 232  
核化线性降维, 232  
核技巧, 127  
核矩阵, 128, 138, 233  
核线性判别分析, 137  
核主成分分析, 232  
合一, 361  
宏  $F_1$ , 32  
宏查全率, 32  
宏查准率, 32  
后剪枝, 79, 352  
划分超平面, 121, 298  
划分选择, 75, 92  
话题模型, 337  
回归, 3  
汇合, 114  
混合属性, 201  
混合专家, 191, 313  
混淆矩阵, 30  
基尼指数, 79  
基学习器, 171  
基学习算法, 171  
基于分歧的方法, 304  
基于能量的模型, 111  
机械学习, 11  
迹, 399  
迹范数, 260  
激活函数, 98  
吉布斯采样, 161, 334  
极大似然法, 59, 149, 297  
极大似然估计, 149, 328  
集成修剪, 191  
集成学习, 171, 311  
急切学习, 225  
级联相关, 110  
挤压函数, 98  
几率, 58

- 计算学习理论, 267  
 加权距离, 201  
 加权平均, 182, 225  
 加权投票, 183, 225  
 加性模型, 173  
 假设, 2, 269  
 假设检验, 37  
 假设空间, 268  
 监督学习, 3  
 间隔, 122  
 简单平均, 182  
 剪枝, 79, 352  
 奖赏, 371  
 降维, 227  
 交叉验证成对  $t$  检验, 40  
 交叉验证法, 26  
 交叉熵, 415  
 街区距离, 200  
 阶跃函数, 57, 98  
 结构风险, 133  
 近端梯度下降, 253, 259  
 近邻成分分析, 238  
 近似动态规划, 393  
 近似推断, 161, 328, 331  
 精度, 23, 29  
 精确推断, 161, 328  
 经验风险, 133  
 经验风险最小化, 278  
 经验误差, 23, 267  
 径向基函数, 108  
 竞争型学习, 108  
 纠错输出码, 64  
 局部极小, 106  
 局部马尔可夫性, 324  
 局部线性嵌入, 235  
 矩阵补全, 259  
 聚类, 3, 197  
 聚类集成, 219  
 聚类假设, 294  
 距离度量, 199  
 距离度量学习, 201, 237  
 卷积神经网络, 113  
 决策树, 73, 363  
 决策树桩, 82  
 绝对多数投票, 182  
 均方误差, 29, 54  
 均匀分布, 409  
 均匀稳定性, 285  
 可分, 269, 270  
 可解释性, 115, 191  
 可塑性-稳定性窘境, 109  
 拉格朗日乘子法, 403  
 拉普拉斯修正, 153  
 拉斯维加斯方法, 251  
 懒惰学习, 154, 225, 240  
 累积误差逆传播, 105  
 类比学习, 11  
 类别不平衡, 66, 299  
 类间散度矩阵, 61, 138  
 类内散度矩阵, 61, 138  
 离散化, 83  
 离散属性, 200  
 联系函数, 57  
 连接权, 101, 104  
 连接主义, 10  
 连续属性, 200  
 链式法则, 103, 402  
 列联表, 41, 187  
 列名属性, 200  
 岭回归, 252  
 留出法, 25  
 流形假设, 240, 294  
 流形学习, 234  
 流形正则化, 240  
 逻辑文字, 347  
 码书学习, 255  
 马尔可夫决策过程, 371  
 马尔可夫链, 161, 320  
 马尔可夫随机场, 322  
 马尔可夫毯, 325  
 马尔可夫网, 319  
 曼哈顿距离, 200  
 没有免费的午餐定理, 9  
 蒙特卡罗方法, 251, 340, 384  
 密采样, 226  
 密度聚类, 211  
 免模型学习, 382  
 闵可夫斯基距离, 200, 220  
 命题规则, 348  
 模仿学习, 390  
 模拟退火, 107  
 模型选择, 24  
 默认规则, 348  
 逆归结, 359  
 逆强化学习, 391  
 欧氏距离, 200

- 盘式记法, 334  
判别式模型, 148, 325  
判定树, 73  
偏差-方差分解, 44, 177  
偏好, 6  
平方损失, 54  
平衡点, 31  
平均场, 337  
平均法, 225  
平稳分布, 161  
朴素贝叶斯分类器, 150  
奇异值分解, 231, 402  
恰PAC可学习, 270  
迁移学习, 17  
嵌入式特征选择, 252  
欠采样, 67  
欠拟合, 23  
欠配, 23  
强化学习, 371  
切比雪夫距离, 200  
亲和矩阵, 301  
权共享, 113  
全局马尔可夫性, 323  
全局散度矩阵, 62  
全局最小, 106  
缺省规则, 348  
缺失值, 85  
人工神经网络, 97  
人工智能, 10  
冗余特征, 247  
软间隔, 129  
软间隔支持向量机, 131  
弱学习器, 171  
熵, 415  
上采样, 67  
深度学习, 113  
神经网络, 97  
神经元, 97  
生成式模型, 148, 295, 325  
胜者通吃, 108  
时间复杂度, 269  
时序差分学习, 386, 393  
示教学习, 11  
示例, 2  
势函数, 322  
视图, 304  
收敛, 99  
受限 Boltzmann 机, 112  
属性, 2  
属性空间, 2  
属性子集, 189  
数据集, 2  
数据挖掘, 14  
数据预处理, 247  
数值属性, 200  
似然, 148  
似然率, 352  
松弛变量, 130  
随机森林, 179  
随机子空间, 189  
探索-利用窘境, 374  
特化, 350  
特征, 2, 247  
特征工程, 114  
特征向量, 2  
特征选择, 247  
特征学习, 114  
梯度下降, 102, 254, 389, 407  
替代函数, 58  
替代损失, 130  
条件独立性假设, 150, 305  
条件风险, 147  
条件随机场, 325  
同父, 158  
统计关系学习, 364  
统计学习, 12, 139  
投票法, 172, 225  
图半监督学习, 300  
推断, 319  
微F1, 32  
微查全率, 32  
微查准率, 32  
维数约简, 227  
维数灾难, 227, 247  
未标记样本, 293  
稳定基学习器, 189  
稳定性, 284  
无导师学习, 3  
无关特征, 247  
无监督学习, 3, 197  
无监督逐层训练, 113  
无序属性, 200  
勿连约束, 239, 307  
误差, 23  
误差-分歧分解, 185

- 误差逆传播, 101  
稀疏编码, 255  
稀疏表示, 67, 255  
稀疏性, 67  
下采样, 67  
先验, 148  
线性超平面, 99  
线性核, 128  
线性回归, 53, 252  
线性降维, 229  
线性可分, 99, 126  
线性模型, 53  
线性判别分析, 60, 139  
相对多数投票, 183  
相对熵, 414  
相关特征, 247  
相似度度量, 201  
协同过滤, 259  
协同训练, 304  
斜决策树, 90  
信念传播, 330, 340  
信念网, 156  
信息散度, 414  
信息增益, 75, 248  
信息熵, 75  
序贯覆盖, 349  
选择性集成, 191  
学习, 2  
学习率, 99  
学习器, 2  
学习向量化, 204  
训练, 2  
训练集, 2  
训练误差, 23  
训练样本, 2  
压缩感知, 257  
哑结点, 99  
演绎, 359  
验证集, 28, 105  
样本, 2  
样本复杂度, 270  
样本空间, 2  
样例, 2  
一阶规则, 348  
一致性, 140  
遗传算法, 107  
异常检测, 219  
因子, 322  
隐变量, 162, 319  
隐狄利克雷分配模型, 337  
隐马尔可夫模型, 319  
硬间隔, 129  
优先级规则, 348  
有标记样本, 293  
有导师学习, 3  
有模型学习, 377  
有限假设空间, 270  
有向分离, 158  
有效性指标, 197  
有序属性, 200  
预剪枝, 79, 352  
阈值, 97, 104  
阈值逻辑单元, 98  
阈值移动, 67  
元规则, 348  
原型聚类, 202  
原子命题, 348  
再励学习, 371  
再平衡, 67  
再生核希尔伯特空间, 128  
再缩放, 67  
在线学习, 109, 241, 393  
早停, 105  
增长函数, 273  
增量学习, 92, 109  
增益率, 77  
召回率, 30  
真相, 2  
振荡, 99  
正态分布, 412  
正则化, 56, 105, 133  
证据, 148  
支持向量, 122  
支持向量回归, 133  
支持向量机, 123  
支持向量展式, 127  
直推学习, 295  
值迭代, 382  
值函数近似, 388  
指数损失, 130, 173  
置换, 361  
置信度, 38  
主成分分析, 229  
主动学习, 293  
状态-动作值函数, 377  
状态值函数, 377  
准确率, 30

- 子集评价, 248  
子集搜索, 248  
子空间, 189, 227  
自适应谐振理论, 108  
自助采样法, 178  
自助法, 27  
自组织映射, 109  
字典学习, 255
- 总体代价, 36  
最近邻分类器, 225  
最小二乘法, 54, 72  
最小描述长度, 159  
最小一般泛化, 358  
最一般合一置换, 361  
坐标下降, 163, 408