-

Machine Learning Notes

Machine Learning Notes

zhuynan

Soochow University

Mechanical and Electrical Engineering College

March 2018

目录

Chapter 1

第一章绪论

1. 根据训练数据是否拥有标记信息，学习任务大致分为：监督学习和无监督学习两类。分类（预测离散值）是前者的代表，回归（预测连续值）是后者的代表。
2. 奥卡姆剃刀（Occam's Razor）原则：若有多个假设与观察一致，则选最简单的那个。例如：曲线我们认为“更平滑”为更简单。
3. 没有免费的午餐定理（No Free Lunch Theorem, NFL 定理）：任何两个学习算法的期望性能是相同的，即总误差相同。NFL 定理的重要前提：所有“问题”出现的机会相同，或所有问题同等重要。但是实际生活中总是针对特定的问题，所以总会有较优的算法。

Chapter 2

第二章　模型评估与选择

2.1　评估方法

2.1.1　留出法 (hold-out)

将数据集 D 划分为两个互斥集合：训练集 S 和测试集 T，即：$D = S \cup T, S \cap T = \phi$。数据集划分时，采用分层采样 (stratified sampling)，单次留出法得到的结果往往不稳定可靠，所以一般若干次随机划分，重复进行实验评估后取平均值作为最终评估结果。

常用的划分比例是：将约 $\frac{2}{3} \sim \frac{4}{5}$ 的样本用于训练，剩余样本用于测试。

2.1.2　交叉验证法 (cross validation)

将数据集 D 划分为 k 个大小相似的互斥子集，即 $D = D_1 \cup D_2 \cup \cdots \cup D_k, D_i \cap D_j = \phi (i \neq j)$，每个子集通过分层采样得到尽可能数据分布一致的样本，每次使用 k-1 个子集的并集作为训练集，剩下的一个子集作为测试集，共得到 k 个测试结果，取平均值。通常把交叉验证法称为 k 折交叉验证 (k-fold cross validation)。k 最常用取值为 10。

特例：**留一法** (Leave-One-Out，简称 LOO)。数据集 D 有 m 个样本，且 k=m。留一法的评估结果通常较为准确，但未必永远更优且计算开销很大。

2.1.3　自助法 (bootstrapping)

对于含有 m 个样本的数据集 D，采样产生含有 m 个样本的数据集 D'：每次随机从 D 中挑选一个样本放入 D' 中，并将此样本放回 D 中，重复 m 次。某个样本不出现在 D' 中的概率是：$(1 - \frac{1}{m})^m$，取极限得到：

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

即初始数据集 D 中约有 36.8% 的样本未出现在采样数据集 D' 中，于是用 D' 作训练集，$D \setminus D'$ 坐测试集。

自助法在数据集小，难以有效划分训练集和测试集时很有用；且自助法能从初始数据集中产生多个不同的训练集，对**集成学习**等有很大优势。但自助法改变了初始数据集的样本分布，会引入估计误差。

2.2　机器学习的参数

机器学习常涉及两类参数：算法参数和模型参数。前者也称为“超参数”，数目常在 10 以内，由人工设定；后者数目可能很多，通过学习产生。

2.3 性能度量

2.3.1 均方误差 (mean squared error)

给定样例集 $D = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$，其中 y_i 是示例 x_i 的真实标记，令 $f(x_i)$ 是 x_i 在学习器 f 上的预测结果。

回归任务：

$$E(f; D) = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)^2 \tag{2.1}$$

更一般的，对于数据分布 D 和概率密度函数 $p(\cdot)$，均方误差可表示为：

$$E(f; D) = \int_{x\sim D}(f(x_i) - y_i)^2 p(x)dx \tag{2.2}$$

2.3.2 查准率 (precision) 与查全率 (recall)

查准率亦称“准确率”，查全率亦称“召回率”。

TP, FP, TN, FN 分别表示真正例 (true positive)，假正例 (false positive)，真反例 (true negative)，假反例 (false negative)，有 $TP + FP + TN + FN =$ 样例总数。

查准率：$P = \frac{TP}{TP+FP}$，查全率：$R = \frac{TP}{TP+FN}$。查准率和查全率是一对矛盾的度量。一般说，当查准率高时，查全率往往偏低；当查全率高时，查准率往往偏低。只有在一些简单的任务中，才可能使查准率和查全率都很高。

“平衡点”(Break-Even Point，简称 BEP) 是当“查准率 = 查全率”时的取值，即 $BEP = P = R$。

2.3.3 F_1 度量

F_1 是基于查准率和查全率的调和平均 (harmonic mean) 定义的：

$$\frac{1}{F_1} = \frac{1}{P} + \frac{1}{R} \Longrightarrow F_1 = \frac{2 \times P \times R}{P + R} = \frac{2 \times TP}{\text{样例总数} - TP - TN} \tag{2.3}$$

F_β 则是加权调和平均：

$$\frac{1}{F_\beta} = \frac{1}{1+\beta^2}(\frac{1}{P} + \frac{\beta^2}{R}) \Longrightarrow F_\beta = \frac{(1+\beta^2) \times P \times R}{(\beta^2 \times P) + R} \tag{2.4}$$

其中 $\beta > 0$ 度量了查全率对查准率的相对重要性。当 $\beta = 1$ 时，退化为标准的 F_1；当 $\beta > 1$ 时，查全率有更大影响；当 $\beta < 1$ 时，查准率有更大影响。

2.3.4 ROC 与 AUC

ROC 全称为：受试者工作特征 (Receiver Operating Characteristic) 曲线。根据学习器的预测结果对样例进行排序，按此顺序**逐个把样本作为正例**进行预测，每次计算 FPR 和 TPR，分别以它们作为横/纵坐标作图。横轴 $FPR = \frac{FP}{TN+FP}$ 是“假正例率”(False Positive Rate)，纵轴 $TPR = \frac{TP}{TP+FN}$ 是“真正例率”(True Positive Rate).

绘制 ROC 图的过程：给定 m^+ 个正例和 m^- 个反例，根据学习器预测结果对样例进行排序，然后把分类阈值设为最大，即把所有样例均预测为反例，此时 TPR 和 FPR 均为 0，在坐标 (0,0) 处标记一个点。然后**将分类阈值依次设为每个样例的预测值，即依次将每个样例划分为正例**。设前一个标记点坐标为 (x, y)，当前若为真正例，则对应标记点的坐标为 $(x, y + \frac{1}{m^+})$；若当前为假正例，则对应标记点的坐标为 $(x + \frac{1}{m^-}, y)$，然后用线段连接相邻点即可。

通过比较 ROC 曲线下的面积，即 AUC(Area Under ROC Curve)，来比较学习器的优劣。AUC 可估算为：

$$AUC = \frac{1}{2}\sum_{i=1}^{m-1}(x_{i+1} - x_i)\cdot(y_i + y_{i+1}) \tag{2.5}$$

AUC 考虑的是样本预测的排序质量。给定 m^+ 个正例和 m^- 个反例，令 D^+ 和 D^- 分别表示正/反例集合，则排序损失 (rank loss) 定义为：

$$\ell_{rank} = \frac{1}{m^+m^-}\sum_{x^+\in D^+}\sum_{x^-\in D^-}\left(\mathbb{I}(f(x^+) < f(x^-)) + \frac{1}{2}\mathbb{I}(f(x^+) = f(x^-))\right) \tag{2.6}$$

即考虑每一对正反例。若正例的预测值小于反例，则代价加一，若相等则代价加 0.5。ℓ_{rank} 表示的是 ROC 曲线上的面积：若一个正例在 ROC 曲线上对应标记点的坐标为 (x, y)，则 x 恰是排序在其之前的反例所占的比例，即假正例率。因此有：

$$AUC = 1 - \ell_{rank} \tag{2.7}$$

2.4 比较检验

2.4.1 假设检验

假设检验的逻辑是：**全称命题只能被否证而不能被证明。**因为个案不足以证明全称命题，但是可以否定全称命题。因此通过否证假设的反面（虚无假设），或者"拒绝"假设的反面，来证明假设。由于抽样的原因，样本并不可能绝对地否证虚无假设。在个案中，小概率事件可以等同于不可能发生的事件。我们在这个意义上去在一定的事先约定的概率水平上去拒绝虚无假设。

在包含 m 个样本的测试集上，泛化错误率为 ϵ 的学习器被测得测试错误率为 $\hat{\epsilon}$ 的概率：

$$P(\hat{\epsilon};\epsilon) = \binom{m}{\hat{\epsilon}\times m}\epsilon^{\hat{\epsilon}\times m}(1-\epsilon)^{m-\hat{\epsilon}\times m} \tag{2.8}$$

求导可得：

$$\frac{\partial P(\hat{\epsilon};\epsilon)}{\partial \epsilon} = \binom{m}{\hat{\epsilon}\times m}\epsilon^{\hat{\epsilon}\times m-1}(1-\epsilon)^{m-\hat{\epsilon}\times m-1}(\hat{\epsilon}-\epsilon)\times m \tag{2.9}$$

所以当 $\hat{\epsilon} = \epsilon$ 时，$P(\hat{\epsilon};\epsilon)$ 最大，当 $|\epsilon - \hat{\epsilon}|$ 增大时，$P(\hat{\epsilon};\epsilon)$ 减小。

二项检验 (binomial test)：考虑假设"$\epsilon \le \epsilon_0$"，则在 $1-\alpha$（此为置信度,confidence）的概率内所能观察到的最大错误率为：

$$\bar{\epsilon} = \max\epsilon \quad \text{s.t.} \quad \sum_{i=\epsilon_0\times m+1}^{m}\binom{m}{i}\epsilon^i(1-\epsilon)^{m-i} < \alpha \tag{2.10}$$

当接受假设"$\epsilon \le \epsilon_0$"时，需满足测试错误率 $\hat{\epsilon}$ 小于临界值 $\bar{\epsilon}$，即能以 $1-\alpha$ 的置信度认为学习器的泛化错误率不大于 ϵ_0。

t 检验 (t-test)：假设得到了 k 个测试错误率 $\hat{\epsilon_1}, \hat{\epsilon_2}, \cdots, \hat{\epsilon_k}$，则平均测试误差 μ 和方差 σ^2 为：

$$\mu = \frac{1}{k}\sum_{i=1}^{k}\hat{\epsilon_i} \tag{2.11}$$

$$\sigma^2 = \frac{1}{k-1}\sum_{i=1}^{k}(\hat{\epsilon_i} - \mu)^2 \tag{2.12}$$

由于这 k 个测试错误率可看作泛化错误率 ϵ_0 的独立采样，则变量：

$$\tau_t = \frac{\sqrt{k}(\mu - \epsilon_0)}{\sigma} \tag{2.13}$$

服从自由度为 k-1 的 t 分布。**需要注意这里的 $\mu, \sigma^2, \epsilon_0$ 分别为样本均值，样本方差和总体均值。**

2.4.2 交叉验证 t 检验

对于学习器 A 和学习器 B，使用 k 折交叉验证法得到的测试错误率分别为 $\epsilon_1^A, \epsilon_2^B, \cdots, \epsilon_k^A$ 和 $\epsilon_1^B, \epsilon_2^B, \cdots, \epsilon_k^B$，令 $\delta_i = \epsilon_i^A - \epsilon_i^B$，若两个学习器的性能相同，则差值均值应为 0。对假设“学习器 A 与学习器 B 性能相同”做 t 检验，计算差值的均值 μ 和方差 σ^2，在显著度 α 下，若：

$$\tau_t = |\frac{\sqrt{k}\mu}{\sigma}|$$

小于临界值 $t_{\alpha/2,k-1}$，则接受假设。其中 $t_{\alpha/2,k-1}$ 是自由度为 k-1 的 t 分布上尾部累积分布为 $\alpha/2$ 的临界值。

2.4.3 McdNemar 检验

对于学习器 A 和学习器 B 分别用 $e_{00}, e_{01}, e_{10}, e_{11}$ 表示 A 和 B 的分类结果都正确，B 正确 A 错误，A 正确 B 错误，AB 均错误。如果两学习器性能相同，则 $e_{01} = e_{10}$，那么变量 $|e_{01} - e_{10}$ 应当服从正态分布. McNemar 检验考虑变量

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

服从自由度为 1 的 χ^2 分布，即标准正态分布变量的平方。分子中有-1，是因为 $e_{01} + e_{10}$ 通常很小，需要考虑连续性校正。给定显著度 α，当以上变量值小于临界值 χ_α^2 时，不能拒绝假设，即应认为两学习器的性能没有显著差别。

补充：**p 值 (p-value) 指的是：在原假设 H_0 成立时，出现现状或更差的情况的概率**。例如抛 20 次硬币，出现 18 次正面，则出现现状或更差的情况是：出现 18，19，20 次正面和出现 0，1，2 次正面，所以 p 值为：$p = (\frac{1}{2})^{20}\left(\binom{20}{18} + \binom{20}{19} + \binom{20}{20} + \binom{20}{0} + \binom{20}{1} + \binom{20}{2}\right)$。

2.4.4 Friedman 检验与 Nemenyi 后续检验

2.5 偏差与方差

泛化误差（在训练集上得到的模型在新样本上的输出误差）可分解为偏差（bias，期望输出与真实标记的差别），方差（variance，预测输出与期望输出的差别）与噪声（noise，数据集中标记和真实标记的差别）之和。

Chapter 3

第三章　线性模型

3.1　线性回归

3.1.1　最小二乘法 (least square method)

基于均方误差 (mean square error, MSE) 最小化的模型求解方法。均方误差定义为:

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

通过对 E 分别求 w 和 b 的偏导，并令其等于 0 可得:

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

$$b=\frac{1}{m}\sum_{i=1}^{m}(y_i-wx_i)$$

其中 $\bar{x}=\frac{1}{m}\sum_{i=1}^{m}x_i$ 为 x 的均值。

为什么可以这样求解呢? 因为损失函数是一个**凸函数**（是向下凸，类似 U 型曲线），导数为 0 表示该函数曲线最低的一点，此时对应的参数值就是能使均方误差最小的参数值。特别地，要判断一个函数是否凸函数，可以求其**二阶导数**，若二阶导数在区间上**非负**则称其为凸函数，若在区间上恒大于零则称其为严格凸函数。

3.1.2　多元线性回归

令 $\hat{\mathbf{w}}=(\mathbf{w};b)$。把数据集表示为 $m\times(d+1)$ 大小的矩阵，每一行对应一个样例，前 d 列是样例的 d 个属性，**最后一列恒置为 1**，对应偏置项。把样例的真实标记也写作向量形式，记作 $\mathbf{y}$。则此时损失函数为:

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

同样使用最小二乘法进行参数估计，首先对 $\hat{\mathbf{w}}$ 求导:

$$
\begin{aligned}
dE_{\hat{\mathbf{w}}} &= (-Xd\hat{\mathbf{w}})^T(\mathbf{y}-X\hat{\mathbf{w}}) + (\mathbf{y}-X\hat{\mathbf{w}})^T(-Xd\hat{\mathbf{w}}) \\
&= (Xd\hat{\mathbf{w}})^T(X\hat{\mathbf{w}}-\mathbf{y}) + (X\hat{\mathbf{w}}-\mathbf{y})^TXd\hat{\mathbf{w}} \\
&= (X\hat{\mathbf{w}}-\mathbf{y})^TXd\hat{\mathbf{w}} + (X\hat{\mathbf{w}}-\mathbf{y})^TXd\hat{\mathbf{w}} \\
&= 2(X\hat{\mathbf{w}}-\mathbf{y})^TXd\hat{\mathbf{w}} \qquad (this\ is\ a\ scalar) \\
&= tr\left((\frac{\partial E_{\hat{\mathbf{w}}}}{\partial\hat{\mathbf{w}}})^Td\hat{\mathbf{w}}\right) \\
&= (\frac{\partial E_{\hat{\mathbf{w}}}}{\partial\hat{\mathbf{w}}})^Td\hat{\mathbf{w}}
\end{aligned}
$$

所以有:

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

当 X^TX 是可逆矩阵，也即**满秩矩阵**（full-rank matrix）或正定矩阵（positive definite matrix）时，令该式值为 0 可得到 $\hat{\mathbf{w}}$ 的闭式解:

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

现实任务中 X^TX 往往不是满秩的，常见的做法是引入**正则化（regularization）**项。

3.2 对数几率回归（逻辑回归）

对数几率函数（logistic function）:

$$y = \frac{1}{1+e^{-(\mathbf{w^Tx}+b)}}$$

该式可以改写为:

$$\ln\frac{y}{1-y} = \mathbf{w^Tx}+b$$

其中，$\frac{y}{1-y}$ 称作几率（odds），把 y 理解为该样本是正例的概率，把 $1-y$ 理解为该样本是反例的概率，而几率表示的就是**该样本作为正例的相对可能性**。若几率大于 1，则表明该样本更可能是正例。对几率取对数就得到对数几率（log odds，也称为 logit）。几率大于 1 时，对数几率是正数。

3.2.1 极大似然法（maximum likelihood method）

所谓极大似然，就是最大化预测事件发生的概率，也即**最大化所有样本的预测概率之积**。令 $p(c=1|\mathbf{x})$ 和 $p(c=0|\mathbf{x})$ 分别代表 y 和 $1-y$。简单变换一下公式，可以得到:

$$p(c=1|\mathbf{x}) = \frac{e^{\mathbf{w^Tx}+b}}{1+e^{\mathbf{w^Tx}+b}}$$

$$p(c=0|\mathbf{x}) = \frac{1}{1+e^{\mathbf{w^Tx}+b}}$$

由于预测概率都是小于 1 的，如果直接对所有样本的预测概率求积，所得的数会非常非常小，当样例数较多时，会超出精度限制。所以，一般来说会对概率去对数，得到**对数似然（log-likelihood）**，此时求所有样本的预测概率之积就变成了求所有样本的对数似然之和 *。对率回归模型的目标就是最大化对数似然，对应的似然函数是:

$$\ell(\mathbf{w},b) = \sum_{i=1}^{m}\ln p(c_i|\mathbf{x_i};\mathbf{w};b) = \sum_{i=1}^{m}\ln(c_ip_1(\hat{\mathbf{x_i}};\beta)+(1-c_i)p_0(\hat{\mathbf{x_i}};\beta))$$

可以理解为若标记为正例，则加上预测为正例的概率，否则加上预测为反例的概率。其中 $\beta = (\mathbf{w}; b)$。

对该式求导，令导数为 0 可以求出参数的最优解。特别地，此时发现似然函数的导数和损失函数是等价的，所以说**最大似然解等价于最小二乘解**。最大化似然函数等价于最小化损失函数：

$$E(\beta) = \sum_{i=1}^{m}(-y_i\beta^T\hat{x}_i + \ln(1 + e^{\beta^T\hat{\mathbf{x}_\mathbf{i}}}))$$

这是一个关于 β 的高阶可导连续凸函数，可以用最小二乘求（要求矩阵的逆，计算开销较大），也可以用数值优化算法如**梯度下降法**（gradient descent method）、**牛顿法**（Newton method）等逐步迭代来求最优解（可能陷入局部最优解）。

3.3 线性判别分析

在线性判别分析（Linear Discriminant Analysis，简称 LDA）中，不再是拟合数据分布的曲线，而是将所有的数据点投影到一条直线上，使得**同类点的投影尽可能近，不同类点的投影尽可能远**。

同类样例的投影值尽可能相近意味着**同类样例投影值的协方差应尽可能小**；然后，异类样例的投影值尽可能远离意味着**异类样例投影值的中心应尽可能大**。合起来，就等价于最大化：

$$J = \frac{\|\mathbf{w}^T\mu_0 - \mathbf{w}^T\mu_1\|_2^2}{\mathbf{w}^T\Sigma_0\mathbf{w} + \mathbf{w}^T\Sigma_1\mathbf{w}} = \frac{\mathbf{w}^T(\mu_0 - \mu_1)(\mu_0 - \mu_1)^T\mathbf{w}}{\mathbf{w}^T(\Sigma_0 + \Sigma_1)\mathbf{w}}$$

其中，分子的 μ_i 表示第 i 类样例的均值向量 * 即表示为向量形式后对各维求均值所得的向量）。分子表示的是两类样例的均值向量投影点（也即类中心）之差的 ℓ_2 范数的平方，这个值越大越好。分母中的 Σ_i 表示第 i 类样例的**协方差矩阵**。分母表示两类样例投影后的协方差之和，这个值越小越好。

LDA 可从贝叶斯决策理论的角度阐释，并可证明：当两类数据同先验，满足高斯分布且协方差相等时，LDA 可达到最优分类。

3.4 多分类学习

基于一些策略，把多分类任务分解为多个二分类任务，利用二分类模型来解决问题。有三种最经典的拆分策略，分别是一对一，一对其余，和多对多。

一对一（One vs. One，简称 OvO）的意思是把所有类别两两配对。假设样例有 N 个类别，OvO 会产生 $\frac{N(N-1)}{2}$ 个子任务，每个子任务只使用两个类别的样例，并产生一个对应的二分类模型。测试时，新样本输入到这些模型，产生 $\frac{N(N-1)}{2}$ 个分类结果，最终预测的标记由投票产生，也即把被预测得最多的类别作为该样本的类别。

一对其余（One vs. Rest，简称 OvR）产生 N 个二分类模型，测试时，新样本输入到这些模型，产生 N 个分类结果，若只有一个模型预测为正例，则对应的类别就是该样本的类别；若有多个模型预测为正例，则选择置信度最大的类别（参考模型评估与选择中的**比较检验**）。

多对多（Many vs. Many，简称 MvM）是每次将多个类作为正例，其他的多个类作为反例。OvO 和 OvR 都是 MvM 的特例。书中介绍的是一种比较常用的 MvM 技术——**纠错输出码**（Error Correcting Outputs Codes，简称 ECOC）。

MvM 的正反例划分不是任意的，必须有特殊的构造，否则组合起来时可能就无法定位到预测为哪一类了。ECOC 的工作过程分两步：

- 编码：对应于训练。假设有 N 个类别，计划做 M 次划分，每次划分把一部分类别划为正类，一部分类别划分为反类，最终训练出 M 个模型。而每个类别在 M 次划分中，被划为正类则记作 +1，被划为负类则记作-1，于是可以表示为一个 M 维的编码。
- 解码：对应于预测。把新样本输入 M 个模型，所得的 M 个预测结果组成一个预测编码。把这个预测编码和各个类别的编码进行比较，跟哪个类别的编码**距离**最近就预测为哪个类别。

此处的距离有汉明距离，欧式距离等。

为什么称这种方法为**纠错**输出码呢？因为 ECOC 编码对分类器的错误有一定的容忍和修正能力。即使预测时某个分类器预测成了错误的编码，在解码时仍然有机会产生正确的最终结果。具体来说，对同一个学习任务，**编码越长，纠错能力越强**。但是相应地也需要训练更多分类器，增大了计算和存储的开销。对同等长度的编码来说，理论上任意两个类别之间的编码距离越远，纠错能力越强。

3.5 类别不平衡问题

欠采样（undersampling）针对的是负类，也即移取训练集的部分反例，使得正类和负类的样例数目相当。由于丢掉了大量反例，所以时间开销也大大减少。但是带来一个问题就是，随机丢弃反例可能会丢失一些重要信息。书中提到一种解决方法是利用**集成学习机制**，将反例划分为多个集合，用于训练不同的模型，从而使得对每个模型来说都进行了欠采样，但全局上并无丢失重要信息。

过采样（oversampling）针对的是正类，也即增加训练集的正例，使得正类和负类的样例数目相当。过采样的时间开销会增大很多，因为需要引入很多正例。注意！过采样不能简单地通过重复正例来增加正例的比例，这样会引起严重的过拟合问题。一种较为常见的做法是对已有正例进行**插值**来产生新的正例。

阈值移动（threshold-moving）利用的是**再缩放**思想。前面对数几率回归中，几率 $\frac{y}{1-y}$ 表示正例的相对可能性，默认以 1 作为阈值，其实是假设了样本的真实分布为正例反例各一半。但这可能不是真相，假设我们有一个存在类别不平衡问题的训练集，正例数目为 m^+，反例数目为 m^-，可以重定义：

$$\frac{y'}{1-y'} = \frac{y}{1-y} \times \frac{m^-}{m^+}$$

这就是再缩放（rescaling）。当几率大于 $\frac{m^+}{m^-}$ 时就预测为正例。但必须注意，这种思想是基于观测几率近似真实几率这一假设的，现实任务中这一点未必成立。

Chapter 4

第四章　决策树

4.1　基本流程

一棵决策树（decision treee）可以分成三个部分：叶节点，非叶节点，分支。叶节点对应决策结果，也即分类任务中的类别标记；非叶节点（包括根节点）对应一个判定问题（某属性 =?）；分支对应父节点判定问题的不同答案（可能的属性值），可能连向一个非叶节点的子节点，也可能连向叶节点。

决策树生成是一个**递归过程**：

1. 传入训练集和属性集
2. 生成一个新节点
3. 若此时数据集中所有样本都属于同一类，则把新节点设置为该类的叶节点，然后**返回** [1]。
4. 若此时属性集为空，或者数据集中所有样本在属性集余下的所有属性上取值都相同，无法进一步划分，则把新节点设置为叶节点，类标记为数据集中样本数最多的类，然后**返回** [2]
5. 从属性集中选择一个最优划分属性
   - 为该属性的每个属性值生成一个分支，并按属性值划分出子数据集
   - 若分支对应的子数据集为空，无法进一步划分，则直接把子节点设置为叶节点，类标记为父节点数据集中样本数最多的类，然后**返回** [3]
   - 将子数据集和去掉了划分属性的子属性集作为算法的传入参数，继续生成该分支的子决策树。

3 处返回中的第 2 处和第 3 处设置叶节点的类标记原理有所不同。第 2 处将类标记设置为当前节点对应为数据集中样本数最多的类，这是利用当前节点的**后验分布**；第 3 处将类标记设置为为父节点数据集中样本数最多的类，这是把父节点的样本分布作为当前节点的**先验分布**。

先验概率（prior probability）：在获得某些信息之前对某个变量 p 的概率分布情况进行猜测。先验概率仅仅依赖主观上的经验估计。后验概率（posterior probability）：在相关证据或背景给定并纳入考虑之后的条件分布，即关于参数 θ 在给定的证据信息 X 下的后验概率是：$P(\theta|X)$。似然函数（likelihood function）：对于结果 X，在参数集合 θ 上的似然，就是在参数给定条件下观察到 X 值的条件分布：$P(X|\theta)$。

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$$

先验就是设定一种情形，似然就是看这种情形下发生的可能性，两者合起来就是后验的概率。

$$Posterior\ Probability \propto Likelihood\ \times Prior\ Probability$$

后验分布正比于先验分布 × 似然函数。$\propto$ 表示正比于。

4.2 划分选择

划分的目标是：尽可能让每个划分子节点中都是同一类别，即结点的纯度越高越好。

4.2.1 信息增益– ID3

信息熵（information entropy）：

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

其中 $|\mathcal{Y}|$ 为类别集合，p_k 为该类样本占样本总数的比例。信息熵越大，表示样本集的混乱程度越高，纯度越低。

信息增益（information gain）是 ID3 算法采用的选择准则，定义如下：

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

它描述的是按某种属性划分后纯度的提升，**信息增益越大，代表用属性 a 进行划分所获得的纯度提升越大**。其中 V 表示属性 a 的属性值集合，D^v 表示属性值为 v 的数据子集。求和项也称为条件熵，可以理解为它是先求出每个数据子集的信息熵，然后按每个数据子集占原数据集的比例来赋予权重，比例越大，对提升纯度的帮助就越大。

4.2.2 增益率– C4.5

增益率（gain ratio）是 C4.5 算法采用的选择准则，定义如下：

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

其中，

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

IV 称为属性的固有值（intrinsic value），它的定义和信息熵是类似的，信息熵衡量的是样本集在类别上的混乱程度，而固有值衡量的是样本集在某个属性上的混乱程度。**固有值越大，则该属性混乱程度越高，可能的取值越多。**

之所以要定义增益率是为了**避免模型过份偏好用取值多的属性作划分**。这是使用信息增益作准则非常容易陷入的误区，比方说每个样本都有一个“编号”属性，这个属性的条件熵肯定是最小的，但如果选择了该属性作为根节点，那么构建出的决策树就没有任何意义了，因为这个模型根本不具备泛化性能。

C4.5 并非直接选择增益率最高的属性，它使用了一个**启发式**：先从属性集中找到信息增益高于平均水平的属性作为候选，然后再比较这些候选属性的增益率，从中选择增益率最高的。

4.2.3 基尼指数–CART

基尼指数（Gini index）是 CART 算法采用的选择准则，定义如下：基尼值：

$$Gini(D) = \sum_{k=1}^{|\mathcal{Y}|} \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^{|\mathcal{Y}|} p_k^2$$

其中 p_k 为该类样本占样本总数的比例。

基尼指数：

$$Gini\_index(D,a)=\sum_{v=1}^{V}\frac{|D^v|}{|D|}Gini(D^v)$$

基尼值是另一种衡量样本集纯度的指标。反映的是**从一个数据集中随机抽取两个样本，其类别标志不同的概率**。基尼值越小，样本集的纯度越高。

4.3 剪枝处理

剪枝（pruning）是决策树学习算法应对**过拟合**的主要手段。判断剪枝是否有用即判断剪枝后模型的泛化性能有没有提升。

4.3.1 预剪枝

预剪枝（prepruning）是在决策树生成的过程中，对每个节点在划分前先进行估计，若当前节点的划分不能带来决策树泛化性能提升（划分后在测试集上错得更多了 / 划分前后在测试集上效果相同），就停止划分并将当前节点标记为叶节点。

4.3.2 后剪枝

后剪枝（postpruning）是先从训练集生成一颗完整的决策树，然后自底向上地逐个考察非叶节点，若将该节点对应**的子树替换为叶节点**能带来决策树泛化性能的提升，则将该子树替换为叶节点。实际任务中，即使没有提升，只要不是性能下降，一般也会剪枝，因为根据奥卡姆剃刀准则，简单的模型更好。

特别地，只有一层划分（即只有根节点一个非叶节点）的决策树称为决策树桩（decision stump）。

4.3.3 优缺点

预剪枝是一种贪心策略，因为它在决策树生成时就杜绝了很多分支展开的机会，所以不但降低了过拟合的风险，同时也 ** 显著减少了模型的训练时间开销和测试时间开销 **。但是这种贪心策略有**可能导致欠拟合**，因为有可能当前划分不能提升模型的泛化性能，但其展开的后续划分却会显著提升泛化性能，在预剪枝中这种可能被杜绝了。

后剪枝是种比较保守的策略，欠拟合的风险很小，泛化性能往往优于预剪枝的决策树。但是由于后剪枝是在生成了完整决策树后，自底向上对所有非叶节点进行考察，所以**训练时间开销**要比未剪枝决策树和预剪枝决策树都大得多。

4.4 连续与缺省值

4.4.1 连续值处理

前面线性模型已经谈到了离散属性连续化，而决策树模型需要的则是**连续属性离散化**，因为决策树每次判定只能做有限次划分。最简单的一种离散化策略是 C4.5 算法采用的二分法（bi-partition）。

给定一个包含连续属性 a 的数据集，并且 a 在数据集中有 n 个不同取值，我们先把属性 a 的 n 个属性值从小到大进行排序。所谓“二分”是指将这些属性值分为两个类别（比如把身高这一属性分为高于 170 和低于 170 两个类别）。

这就产生了一个新问题，怎么找到合适的划分点（例如上面例子的 170）呢?

在对连续属性值排序完之后，由于有 n 个不同取值，取**每两个取值的平均值作为划分点**的话，就有 $n-1$ 个候选划分点。按照相应准则（比方说用 ID3 算法的话就是信息增益）进行 $n-1$ 次判断。每次拿出一个候选划分点，把连续属性分为两类，转换为离散属性，然后基于这个基础计算准则，最终选出一个最优的属性值划分点。

注意：和离散属性不同，连续属性用于当前节点的划分后，其**后代节点依然可以使用该连续属性进一步划分**。例如，当前节点用身高低于 170 划分了，那么它的后代节点还可以用身高低于 160 来进一步划分。

4.4.2 缺失值处理

假设数据集为 D，有缺失值的属性为 a，令 $\tilde{D}$ 表示 D 中没有缺失属性 a 的样本子集。定义变量：

$$\rho = \frac{\sum_{\mathbf{x}\in\tilde{D}} w_{\mathbf{x}}}{\sum_{\mathbf{x}\in D} w_{\mathbf{x}}}$$
$$\tilde{p_k} = \frac{\sum_{\mathbf{x}\in\tilde{D}_k} w_{\mathbf{x}}}{\sum_{\mathbf{x}\in\tilde{D}} w_{\mathbf{x}}}, \quad (1 \le k \le |\mathcal{Y}|)$$
$$\tilde{r_v} = \frac{\sum_{\mathbf{x}\in\tilde{D}^v} w_{\mathbf{x}}}{\sum_{\mathbf{x}\in\tilde{D}} w_{\mathbf{x}}}, \quad (1 \le v \le V)$$

ρ 表示无缺失值样本所占的比例；$\tilde{p_k}$ 表示无缺失值样本中第 k 类所占的比例；$\tilde{r_v}$ 表示无缺失值样本中在属性 a 上取值 a^v 的样本所占的比例。

注意：这里的 $w_{\mathbf{x}}$ 表示样本的权值，它是含缺失值样本参与建模的一种方式。在根节点处初始时，所有样本 $\mathbf{x}$ 的权重都为 1。

接下来重新定义信息熵和信息增益，推广到样本含缺失值的情况：

$$Ent(\tilde{D}) = -\sum_{k=1}^{|\mathcal{Y}|} \tilde{p_k} log_2 \tilde{p_k}$$

$$Gain(D,a) = \rho \times Gain(\tilde{D},a) = \rho \times (Ent(\tilde{D}) - \sum_{v=1}^{V} \tilde{r_v} Ent(\tilde{D^v}))$$

按照新的定义来计算包含缺失值的属性的信息增益，然后和其他属性的信息增益相比，选出最优的划分属性。

假设有一个包含缺失值的属性被计算出是最优划分属性，那么我们就要按该属性的不同取值划分数据集了。缺失该属性值的样本怎么划分呢？答案是**按概率划分**，这样的样本会被**同时**划入所有子节点，并且其权重更新为对应的 $\tilde{r_v} \dot{w}_{\mathbf{x}}$。

可以把无缺失值的决策树建模想象为各样本权值恒为 1 的情形，它们只对自己所属的属性值子集作贡献。而样本含缺失值时，它会以不同的概率对所有属性值子集作贡献。

4.5 多变量决策树

前面提到的决策树都是单变量决策树（univariate decision tree），即在每个节点处做判定时都只用到一个属性。它有一个特点，就是形成的分类边界都是轴平行（axis-parallel）的，即分类边界由若干个与坐标轴平行的分段组成。

如果把属性都当作坐标空间中的坐标轴，由于我们建模时假设样本的各属性之间是没有关联的，所以**各坐标轴是相互垂直的**。而决策数每次只取一个确定的属性值来划分，就等同于画一个垂直于该属性坐标轴的超平面（只有两个属性时就是一条线），它与其他坐标轴都是平行的，这就是轴平行，最终由多个与坐标轴平行的超平面组成分类边界。

这样有一个弊端就是，如果真实分类边界特别复杂，就需要画出很多超平面（线），在预测时就需要继续大量的属性测试（遍历决策树）才能得到结果，预测时间开销很大。

多变量决策树（multivariate decision tree），不再是选择单个最优划分属性作为节点，而是试图寻找一个**最优的多属性的线性组合**作为节点，它的每个非叶节点都是一个形如 $\sum_{i=1}^{d} w_i a_i = t$ 的线性分类器。多变量决策树的决策边界能够斜着走，甚至绕曲线走，从而用更少的分支更好地逼近复杂的真实边界。

Chapter 5

第五章　神经网络

5.1　神经元模型

神经网络是由具有适应性的简单单元组成的广泛**并行互连**的网络，它的组织能够模拟生物神经系统对真实世界物体所做出的反应。神经元（neuron）模型是神经网络最基本的组成成分。M-P 神经元模型：神经元收到来自 n 个其他神经元传递过来的输入信号，这些输入信号通过带权重的连接进行传递，将神经元接收到的总输入值和神经元的阈值进行比较，然后通过激活函数（activation function）处理以产生神经元的输出。

$$y = f(\sum_{i=1}^{n} w_i x_i - \theta) \tag{5.1}$$

激活函数一般要求的性质：

- 非线性：内在反映的是模型的非线性
- 可微性：以支持梯度下降法等基于微分的优化方法
- 单调性：以保证单层网络是凸约束，从而存在最优解
- 输出值范围受限：输出值有限时，寻优算法（如梯度下降）变得更稳定。

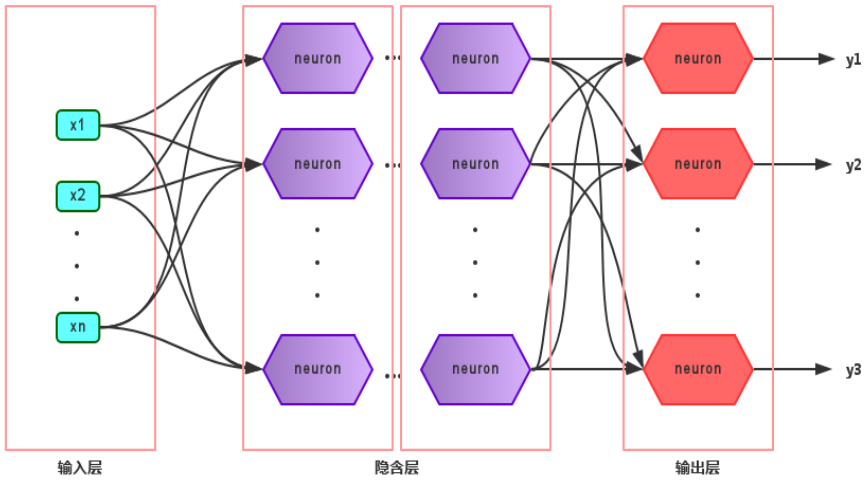
常用的激活函数：
对数几率函数：$\sigma(x) = \text{sigmod}(x) = \frac{1}{1+e^{-x}}$
双曲正切函数：$\tanh(x) = 2\sigma(2x) - 1$
修正线性函数：$\text{ReLU}(x) = \max(0, x)$
径向基函数（Radial Basis Function）：高斯径向基函数：$P(x, x') = e^{-\frac{(x-x')^2}{2\sigma^2}}$，其中 σ 为自由参数。
softmax 函数：softmax 函数常用于将多个标量映射为一个概率分布。$\text{softmax}(x_k) = \frac{e^{x_k}}{\sum_{i=1}^{n} e^{x_i}}$

5.2　感知机与多层网络

感知机（Perceptron）是由输入层和输出层是 M-P 神经元的两层神经元组成。因为感知机只有输出层神经元进行激活函数处理，即只拥有一层功能神经元，**只能解决线性可分问题**。

多层功能神经元可以解决非线性可分问题。输入层和输出层之间的隐含层与输出层神经元都是拥有激活函数的功能神经元。最为常见的多层神经网络——**多层前馈神经网络**（multi-layer feedforward neural networks），它有以下特点：

- 每层神经元与下一层神经元全互连
- 神经元之间不存在同层连接



多层前馈神经网络

- 神经元之间不存在跨层连接

神经网络的学习其实就是调整各神经元之间的连接权（connection weight）以及各神经元的阈值。

5.3 误差逆传播算法

误差逆传播算法（error BackPropagation，简称 BP）也称为反向传播算法，是最为成功的一种神经网络学习方法之一。一般而言，BP 神经网络是指用 BP 算法训练的多层前馈神经网络，但 BP 算法也能训练其他类型的神经网络，如递归神经网络。

5.3.1 标准 BP 算法

假设要训练的是一个单隐层的前馈神经网络，BP 算法使用**均方误差作为性能度量**，基于梯度下降（gradient descent）策略，以目标函数的负梯度方向对参数进行调整。

输入：训练集 $D = (\mathbf{x}_k, \mathbf{y}_k)_{k=1}^m$，学习率 η。
过程：
1：在 $(0, 1)$ 范围内随机初始化网络中所有连接权和阈值
2：**repeat**
3：　**for all** $(\mathbf{x}_k, \mathbf{y}_k \in D)$ **do**
4：　　根据当前参数计算出样本的输出 $\hat{\mathbf{y}}_k$
5：　　计算输出层神经元的梯度项 g_j
6：　　计算隐层神经元的梯度项 e_h
7：　　更新连接权与阈值
8：　**endfor**
9：**until** 达到停止条件
输出：连接权与阈值确定的多层前馈神经网络

所谓逆传播其实就是从输出层开始逐步往后更新，因为输出层的误差确定后就可以对输出层的连接权和阈值进行更新，并且可以推算出隐含层输出的“真实值”，从而计算出隐含层的“误差”，然后更新

隐含层的连接权和阈值。BP 就是这样一种利用一层层倒推来最终更新整个神经网络的方法，每一层的更新公式其实和感知机用的是类似的。

在学习过程中，学习率 η 控制着每一轮迭代的更新步长，太大则容易振荡，太小则收敛速度太慢。所以常常设置 η 和迭代次数相关。

5.3.2 累积 BP 算法

BP 算法的目标是最小化训练集 D 上的累积误差：

$$E = \frac{1}{m}\sum_{k=1}^{m} E_k$$

而标准 BP 算法每输入一个样例就进行一次更新，所以它的参数更新非常频繁，而且不同样例可能会对更新起到抵消效果，从而使得模型需要更多次迭代才能到达累积误差的极小点。

如果把更新方式变为每输入一遍训练集进行一次更新，就得到累积 BP 算法，更新公式需要重新推导一下。这样更新一次就称为一轮（one round，亦称 one epoch）学习。

5.3.3 过拟合问题

鉴于 BP 神经网络强大的表达能力，很容易会遇到过拟合问题。主要有以下两种应对策略：

- **早停**（early stopping）：把数据集分为训练集和验证集，若训练集误差降低但测试集误差升高，就停止训练，并返回具有最小验证集误差的连接权和阈值。
- **正则化**（regularization）：在目标函数中添加一个用于描述网络复杂度的部分，比如连接权和阈值的平方和。这样训练时就会偏好较小的连接权和阈值，从而令输出更“光滑”。带正则化项的目标函数如下：

  $$E = \lambda\frac{1}{m}\sum_{k=1}^{m} E_k + (1-\lambda)\sum_i w_i^2$$

  其中 λ 是用于对经验误差和网络复杂度折中的参数，常通过交叉验证法来估计。

为什么可以用正则化来避免过拟合呢？ 过拟合的时候，**拟合函数的系数往往非常大**【过拟合，就是拟合函数需要顾忌每一个点，最终形成的拟合函数波动很大。在某些很小的区间里，函数值的变化很剧烈。这就意味着函数在某些小区间里的导数值（绝对值）非常大】，而正则化是通过约束参数的范数使其不要太大，所以可以在一定程度上减少过拟合情况。

5.4 局部最小

因为用梯度下降搜索最优解可能会陷入非全局最小解的局部极小解，在现实任务中，人们会使用以下这些策略试图跳出局部极小：

- **以多组不同参数值初始化多个神经网络**：经过训练后，取误差最小的解作为最终参数。这种方法相当于从多个不同的初始点开始搜索，从而增加找到全局最小解的可能性。
- **模拟退火**（simulated annealing）技术：每次迭代都以一定的概率接收比当前解差的结果，从而有机会跳出局部极小（当然也有可能跳出全局最小），每次接受“次优解”的概率会随着时间推移逐渐减小，从而保证了算法的稳定。
- **随机梯度下降**（stochastic gradient descent，简称 SGD）：在计算梯度时加入了随机因素，因此即便陷入局部极小点，梯度依然可能不为 0，从而有机会跳出局部极小，继续搜索。

除了这些方法之外，**遗传算法**也常用于训练神经网络以逼近全局最小。当这些技术大多是启发式，没有理论的保障。

5.5 其他常见神经网络

5.5.1 RBF 网络

RBF（Radial Basis Function）网络是一种单隐层前馈神经网络，它使用径向基函数作为隐层神经元的激活函数，输出层则直接使用隐层神经元的线性组合。

5.5.2 ART 网络

竞争型学习（competitive learning）是神经网络中常用的一种无监督学习策略。使用该策略时，网络中的输出神经元相互竞争，每次只有一个竞争获胜的神经元被激活，其它输出神经元被抑制，这种机制又称为胜者通吃（winner-take-all）。

ART（Adaptive Resonance Theory，自适应谐振理论）网络是竞争型学习的重要代表。该网络由四部份组成：**比较层、识别层、识别阈值、重置模块**。比较层就是输入层，只负责把样本传递给识别层。识别层也即输出层，但识别层的每个神经元对应一个模式类，而且神经元的数目可以在训练过程中动态增加以增加新的模式类。

识别层的每个神经元有一个对应的模式类的代表向量，每次输入一个样本，各识别层神经元相互竞争，代表向量与样本距离最小的胜出并被激活。获胜神经元会向其他神经元发送信号，抑制其激活。如果样本与获胜神经元的距离小于识别阈值，则被分到对应的模式类。否则，重置模块会在识别层新增一个神经元，代表向量为该样本。

ART 能有效缓解竞争型学习中的**可塑性-稳定性窘境**（stability-plasticity dilemma）。可塑性指神经网络要有学习新知识的能力，稳定性则指神经网络在学习新知识时要保持对旧知识的记忆。

ART 具备可塑性和稳定性，因此能进行**增量学习**（incremental learning）和**在线学习**（online learning）。

增量学习可以理解为建立模型后再收到新的样例时可以对模型进行更新，但不用重新训练整个模型，先前学得的有效信息会被保存。它可以逐个新样例进行更新，也能以批模型（batch-mode），每次用一批新样例来更新。

在线学习则可以理解为每拿到一个新样本就进行一次模型更新，不需要一开始就准备好完整的训练集，每次收到新样例都能继续训练。可以看作增量学习的一个特例。

5.5.3 SOM 网络

SOM（Self-Organizing Map，自组织映射）网络，又称为自组织特征映射网络或 Kohonen 网络。同样是一种竞争学习型无监督神经网络，只有输入层和输出层两层，输出层以矩阵形式排列。与样本距离最近的输出层神经元获胜，称为最佳匹配单元（best matching unit）。最佳匹配单元和邻近神经元的权向量会被调整，使得下次遇到相似的样本时距离更小。如此迭代，直至收敛。

5.5.4 级联相关网络

级联相关（Cascade-Correlation）网络是一种典型的结构自适应网络，这类网络不仅通过训练来学习合适的连接权和阈值等参数，还会在训练过程中找到最符合数据特点的网络结构。

级联相关神经网络有两个主要成分：

- 级联：指建立层次连接的层级结构。开始训练时，只有输入层和输出层，随着训练进行逐渐加入隐层神经元，从而建立层级结构。注意，**隐层神经元的输入端连接权是冻结固定的**。
- 相关：指通过**最大化新神经元的输出与网络误差之间的相关性**（correlation）来训练相关的参数。

5.5.5 Elman 网络

递归神经网络（recurrent neural networks，简称 RNN）允许网络中出现环形结构，即一些神经元的**输出可以反馈回来当输入信号**，从而能够处理与时间有关的动态变化。

Elman 网络是最常用的递归神经网络之一，只有一个隐层，并且隐层神经元的输出会被反馈，在下一时刻与输入层神经元的输入信号一起作为隐层神经元的新输入。隐层神经元一般采用 Sigmoid 函数作为激活函数，并用 BP 算法训练整个网络。

5.5.6 Boltzmann 机

神经网络中有一类基于能量的模型（energy-based model），**把网络状态定义为一个能量，能量最小时网络达到理想状态**，模型的学习过程就是最小化能量函数。Boltzmann 机就是这样的模型，同时也是一种 RNN。

Boltzmann 机的神经元分为显层与隐层，显层用于表达数据的输入与输出，隐层则是数据的内在。每个神经元只有 0、1 两种状态，也即抑制和激活。

标准的 Boltzmann 机是全连接图，即任意两个神经元之间都相连。但复杂度太高，难以用于解决现实任务，实际应用中用的是**受限 Boltzmann 机**（Restricted Boltzmann Machine，简称 RBM），把标准 Boltzmann 机退化为二部图，只保留显层和隐层之间的连接，同一层直接不相连。

5.6 深度学习

理论上，参数越多，模型复杂度就越高，容量（capability）就越大，从而能完成更复杂的学习任务。怎么增大模型复杂度呢？两个办法，一是**增加隐层的数目**，二是**增加隐层神经元的数目**。前者更有效一些，因为它不仅增加了功能神经元的数量，还增加了激活函数嵌套的层数。但是对于多隐层神经网络，经典算法如标准 BP 算法往往会在误差逆传播时发散（diverge），无法收敛到稳定状态。

那要怎么有效地训练多隐层神经网络呢？一般来说有以下两种方法：

- **无监督逐层训练**（unsupervised layer-wise training）：每次训练一层隐结点，把上一层隐结点的输出当作输入来训练，本层隐结点训练好后，输出再作为下一层的输入来训练，这称为预训练（pre-training）。全部预训练完成后，再对整个网络进行微调（fine-tuning）训练。一个典型例子就是深度信念网络（deep belief network，简称 DBN）。这种做法其实可以视为把大量的参数进行分组，先找出每组较好的设置，再基于这些局部最优的结果来训练全局最优。
- **权共享**（weight sharing）：令同一层神经元使用完全相同的连接权，典型的例子是卷积神经网络（Convolutional Neural Network，简称 CNN）。这样做可以大大减少需要训练的参数数目。

事实上，深度学习可以理解为一种特征学习（feature learning）或者表示学习（representation learning），无论是 DBN 还是 CNN，都是通过多个隐层来把初始与输出目标联系不大的输入表示转化为与输出目标更密切的表示，使原来只通过单层映射难以完成的任务变为可能。

Chapter 6

第六章　支持向量机

6.1　间隔与支持向量

6.1.1　点到超平面距离

样本空间中任一点 $\mathbf{x}$ 到超平面 $(\mathbf{w}, b)$ 的距离为：

$$r = \frac{|\mathbf{w}^T\mathbf{x} + b|}{\|\mathbf{w}\|}$$

证明方法：
1）点到直线距离公式。
2）向量方法。投影向量 $\mathbf{w}$ 垂直于超平面，点 x 对应向量 $\mathbf{x}$，过点 x 作超平面的垂线，交点 x_0 对应向量 $\mathbf{x_0}$。假设由点 x_0 指向点 x 的向量为 $\mathbf{r}$，长度（也即点 x 与超平面的距离）为 r。由向量加法定义可得 $\mathbf{x} = \mathbf{x_0} + \mathbf{r}$。其中向量 $\mathbf{r}$ 等于这个方向的单位向量乘上 r，也即有 $\mathbf{r} = \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot r$。因此又有 $\mathbf{x} = \mathbf{x_0} + \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot r$。由于点 x_0 在超平面上，所以有 $\mathbf{w}^T\mathbf{x_0} + b = 0$。由 $\mathbf{x} = \mathbf{x_0} + \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot r$ 可得 $\mathbf{x_0} = \mathbf{x} - \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot r$，代入直线方程消去 $\mathbf{x_0}$：

$$\mathbf{w}^T\mathbf{x_0} + b = \mathbf{w}^T(\mathbf{x} - \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot r) + b = 0$$

简单变换即可得到:

$$r = \frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$$

又因为取距离为正值，所以要加上绝对值符号：

$$r = \frac{|\mathbf{w}^T\mathbf{x} + b|}{\|\mathbf{w}\|}$$

6.1.2　间隔

根据定义，所有支持向量都满足：

$$\mathbf{w}^T\mathbf{x} + b = +1, \quad y_i = +1$$

$$\mathbf{w}^T\mathbf{x} + b = -1, \quad y_i = -1$$

代入前面的距离公式可以得到支持向量到超平面的距离为 $\frac{1}{\|\mathbf{w}\|}$。

定义间隔（margin）为两个异类支持向量到超平面的距离之和：

$$\gamma = 2 \cdot \frac{1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

SVM 的目标便是找到**具有最大间隔**（maximum margin）的划分超平面，也即找到使 γ 最大的参数 $\mathbf{w}$ 和 b：

$$\max_{\mathbf{w},b} \frac{2}{\|\mathbf{w}\|} \quad s.t. \quad y_i(\mathbf{w}^T\mathbf{x}+b) \geq 1, \quad i = 1, 2, ..., m$$

看上去间隔只与 $\mathbf{w}$ 有关，但实际上位移项 b 也通过约束影响着 $\mathbf{w}$ 的取值，进而对间隔产生影响。由于最大化 $\|\mathbf{w}\|^{-1}$ 等价于最小化 $\|\mathbf{w}\|^2$，所以可以重写目标函数为：

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad s.t. \quad y_i(\mathbf{w}^T\mathbf{x}+b) \geq 1, \quad i = 1, 2, ..., m \tag{6.1}$$

这便是支持向量机的基本型。

6.2 对偶问题

式（1）是一个**带约束的凸二次规划**（convex quadratic programming）问题（凸问题就意味着必定能求到全局最优解，而不会陷入局部最优）。下面介绍一种求解方法。

首先为式（1）的每条约束添加拉格朗日乘子 $a_i \geq 0$（对应 m 个样本的 m 条约束），得到该问题的拉格朗日函数：

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{m} a_i(1 - y_i(\mathbf{w}^T\mathbf{x}+b)) \tag{6.2}$$

其中 $\mathbf{a} = (a_1; a_2; ...; a_m)$，对拉格朗日函数求 $\mathbf{w}$ 和 b 的偏导，并令偏导为 0 可以得到：

$$\mathbf{w} = \sum_{i=1}^{m} a_i y_i \mathbf{x}i \tag{6.3}$$

$$0 = \sum_{i=1}^{m} a_i y_i \tag{6.4}$$

将式（3）代入式（2）可以消去 $\mathbf{w}$ 和 b，然后再考虑式（4）的约束就得到了式（1）的对偶问题（dual problem）：

$$\max_{\mathbf{a}} \sum_{i=1}^{m} a_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j y_i y_j \mathbf{x}_i^T \mathbf{x}j \quad s.t. \quad \sum_{i=1}^{m} a_i y_i = 0, \quad a_i \geq 0, \quad i = 1, 2, ..., m \tag{6.5}$$

只要求出对偶问题的解 $\mathbf{a}$，就可以推出 $\mathbf{w}$ 和 b，从而得到模型（不过实际计算时一般不这样做，特别是需要用核函数映射到高唯空间时，因为映射后做内积很困难，而用少量支持向量进行表示，在原始空间进行计算显然更优）：

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}^T\mathbf{x} + b \\ &= \sum_{i=1}^{m} a_i y_i \mathbf{x}_i^T \mathbf{x} + b \end{aligned} \tag{6.6}$$

注意，由于式（1）的约束条件是不等式约束，所以求解过程要求满足 KKT（Karush-Kuhn-Tucker）条件：

$$\begin{cases} a_i \geq 0; \\ y_i f(\mathbf{x}_i) - 1 \geq 0; \\ a_i(y_i f(\mathbf{x}_i) - 1) = 0. \end{cases} \tag{6.7}$$

KKT 条件说明了，对任何一个样本来说，要么对应的拉格朗日乘子为 0，要么函数间隔等于 1（即式（1）的约束条件取等号）。如果拉格朗日乘子为 0，则这个样本对式（6）毫无贡献，不会影响到模型；如果函数间隔为 1，则表明这个样本位于最大间隔边界上，是一个支持向量。它揭示了 SVM 的一个重要性质：**最终模型只与支持向量有关，因此训练完成后，大部分的训练样本都不需保留**。

通常使用 **SMO 算法**求解式（5）。

6.3 核函数

6.3.1 如何处理非线性划分

在现实任务中，我们更常遇到的是在原始样本空间中非线性可分的问题。对这样的问题，一种常用的思路是将样本从原始空间映射到一个更高维的特征空间，使得样本在该特征空间中线性可分。幸运的是，**只要原始空间是有限维的（也即属性数目有限），那就必然存在一个高维特征空间使样本线性可分**。

6.3.2 核函数

假设用 $\phi(\mathbf{x})$ 来表示映射所得的特征向量。则在映射的高维特征空间中，用于划分的线性超平面可以表示为：

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

类似式（1），可以得到此时的目标函数为：

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad s.t. \quad y_i(\mathbf{w}^T\phi(\mathbf{x}) + b) \geq 1, \quad i = 1, 2, ..., m \tag{6.8}$$

对应的对偶问题为：

$$\max_{\mathbf{a}} \sum_{i=1}^{m} a_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j y_i y_j \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) \quad s.t. \ \sum_{i=1}^{m} a_i y_i = 0, \ a_i \geq 0, \ i = 1, 2, ..., m \tag{6.9}$$

注意到对偶问题中，涉及到 $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ 的计算，也即 x_i 和 x_j 映射到高维特征空间后的内积（比如 $x_i = (1, 2, 3)$，$x_j = (4, 5, 6)$，那么内积 $x_i^T x_j$ 就等于 $1 * 4 + 2 * 5 + 3 * 6 = 32$），由于特征空间维数可能很高，所以直接计算映射后特征向量的内积是很困难的，如果映射后的特征空间是无限维，根本无法进行计算。为了解决这样的问题，就引入了核函数（kernel function）。

假设输入空间是二维的，每个样本点有两个属性 x 和 y，存在映射将每个样本点映射到三维空间：

$$\phi(\mathbf{x}) = \phi(x, y) = (x^2, \sqrt{2}xy, y^2)$$

给定原始空间中的两个样本点 $\mathbf{v}_1 = (x_1, y_1)$ 和 $\mathbf{v}_2 = (x_2, y_2)$，则它们映射到高维特征空间后的内积可以写作：

$$\begin{aligned}
&\phi(\mathbf{v}_1)^T\phi(\mathbf{v}_2) =< \phi(\mathbf{v}_1), \phi(\mathbf{v}_2) > \\
=&< (x_1^2, \sqrt{2}x_1y_1, y_1^2), (x_2^2, \sqrt{2}x_2y_2, y_2^2) > \\
=& x_1^2x_2^2 + 2x_1x_2y_1y_2 + y_1^2y_2^2 = (x_1x_2 + y_1y_2)^2 \\
=&< \mathbf{v}_1, \mathbf{v}_2 >^2 \\
=& \kappa(\mathbf{v}_1, \mathbf{v}_2)
\end{aligned}$$

高维特征空间中两个点的内积，可以写成一个关于原始空间中两个点的函数 $\kappa(\cdot, \cdot)$，这就是核函数。

为什么需要核函数?

假设原始空间是二维的，那么对于两个属性 x 和 y，取一阶二阶的组合只有 5 个（也即 x^2，y^2，x，y，xy）。但当原始空间是三维的时候，仍然取一阶二阶，组合就多达 19 个了（也即 x，y，z，xy，xz，

yz, x^2y, x^2z, y^2x, y^2z, z^2x, z^2y, x^2yz, xy^2z, xyz^2, x^2y^2z, x^2yz^2, xy^2z^2, xyz)。随着原始空间维数增长，新空间的维数是呈爆炸性上升的。然而有了核函数，我们就可以在原始空间中通过函数 $\kappa(\cdot;\cdot)$ 计算（这称为**核技巧**（kernel trick)），而不必直接计算高维甚至无穷维特征空间中的内积。

使用核函数后，对偶问题式（9）可以重写为:

$$\max_{\mathbf{a}} \sum_{i=1}^{m} a_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j y_i y_j \kappa(\mathbf{x}_i;\mathbf{x}j) \quad s.t.\ \sum_{i=1}^{m} a_i y_i = 0,\ a_i \geq 0, \quad i = 1,2,...,m \tag{6.10}$$

求解后得到的模型可以表示为:

$$f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b = \sum_{i=1}^{m} a_i y_i \phi(\mathbf{x}_i)^T\phi(\mathbf{x}) + b = \sum_{i=1}^{m} a_i y_i \kappa(\mathbf{x}_i;\mathbf{x}) + b$$

这表明了模型最优解可通过训练样本的核函数展开，称为支持向量展式（support vector expansion)。

注意，**核函数本身不等于映射。它只是用来计算两个数据点映射到高维空间之后的内积的一种简便方法。** 当发现数据在原始空间线性不可分时，会有把数据映射到高维空间来实现线性可分的想法，比方说引入原有属性的幂或者原有属性之间的乘积作为新的维度。假设把数据点都映射到了一个维数很高甚至无穷维的特征空间，而模型求解和预测的过程需要用到映射后两个数据点的内积，这时直接计算就没辙了。但又幸运地发现，**原来高维空间中两点的内积在数值上等于原始空间通过某个核函数算出的函数值，无需先映射再求值**，就很好地解决了计算的问题了。

6.3.3 核函数定理

给定一个输入空间 $\mathcal{X}$，函数 $\kappa(\cdot;\cdot)$ 是定义在 $\mathcal{X}\times\mathcal{X}$ 上的对称函数。当且仅当对于任意数据集 $D = \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m$, 对应的**核矩阵（kernel matrix）都是半正定**的时候，κ 是核函数。

任何一个核函数都隐式地定义了一个称为“再生核希尔伯特空间(Reproducing Kernel Hilbert Space，简称 RKHS)”的特征空间。

6.3.4 常用核函数

- 线性核：$\kappa(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$
- 多项式核：$\kappa(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i^T\mathbf{x}_j)^d$，其中 $d \geq 1$ 为多项式的次数，d=1 时退化为线性核
- 高斯核（亦称 RBF 核)：$\kappa(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2})$，$\sigma > 0$ 为高斯核的带宽（width)
- 拉普拉斯核：$\kappa(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|}{\sigma})$，$\sigma > 0$
- Sigmoid 核：$\kappa(\mathbf{x}_i,\mathbf{x}_j) = \tanh(\beta\mathbf{x}_i^T\mathbf{x}_j + \theta)$，tanh 为双曲正切函数，$\beta > 0, \theta < 0$

核函数组合:

- 若 κ_1 和 κ_2 都是核函数，则 $a\kappa_1 + b\kappa_2$ 也是核函数，其中 $a > 0, b > 0$。
- 若 κ_1 和 κ_2 都是核函数，则其直积 $\kappa_1 \otimes \kappa_2(\mathbf{x},\mathbf{z}) = \kappa_1(\mathbf{x},\mathbf{z})\kappa_2(\mathbf{x},\mathbf{z})$ 也是核函数。
- 若 κ_1 是核函数，则对于任意函数 $g(\mathbf{x})$，$\kappa(\mathbf{x},\mathbf{z}) = g(\mathbf{x})\kappa_1(\mathbf{x},\mathbf{z})g(\mathbf{z})$ 也是核函数。

6.4 软间隔与正则化

6.4.1 软间隔

软间隔是相对于硬间隔（hard margin）的一个概念，硬间隔要求所有样本都必须划分正确，也即约束:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$

软间隔则允许某些样本不满足约束（根据约束条件的不同，有可能某些样本出现在间隔内，甚至被误分类）。此时目标函数可以重写为：

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \ell_{0/1}(y_i(\mathbf{w}^T\mathbf{x} + b) - 1) \tag{6.11}$$

其中 $\ell_{0/1}$ 是 0/1 损失函数：

$$\ell_{0/1}(z) = \begin{cases} 1, & if\ z < 0; \\ 0, & otherwise. \end{cases} \tag{6.12}$$

它的含义很简单：如果分类正确，那么函数间隔必定大于等于 1，此时损失为 0；如果分类错误，那么函数间隔必定小于等于-1，此时损失为 1。

而 C 则是一个大于 0 的常数，当 C 趋于无穷大时，式（11）等效于带约束的式（1），因为此时对误分类的惩罚无限大，也即要求全部样本分类正确。当 C 取有限值时，允许某些样本分类错误。

由于 0/1 损失函数是一个非凸不连续函数，所以式（11）难以求解，于是在实际任务中，采用一些凸的连续函数来取替它，这样的函数就称为**替代损失（surrogate loss）函数**。

最常用的有以下三种：

- hinge 损失：$\ell_{hinge}(z) = \max(0, 1 - z)$
- 指数损失（exponential loss）：$\ell_{\exp}(z) = \exp(-z)$
- 对率损失（logistic loss）：$\ell_{\log}(z) = \log(1 + \exp(-z))$

实际任务中最常用的是 **hinge 损失**，这里就以 hinge 损失为例，替代 0/1 损失函数，此时目标函数式（11）可以重写为：

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x} + b)) \tag{6.13}$$

引入松弛变量（slack variables）$\xi_i \geq 0$，可以把式（13）重写为：

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i \quad s.t. \quad y_i(\mathbf{w}^T\mathbf{x} + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, 2, ..., m \tag{14}$$

该式描述的就是软间隔支持向量机，其中**每个样本都对应着一个松弛变量，用以表征该样本误分类的程度，值越大，程度越高。**

6.4.2 支持向量机和逻辑回归的联系与区别

上面用的是 hinge 损失，不过也提到了还有其他一些替代损失函数，事实上，使用对率损失时，SVM 得到的模型和 LR 是非常类似的。

支持向量机和逻辑回归的相同点：

- 都是线性分类器，模型求解出一个划分超平面
- 两种方法都可以增加不同的正则化项
- 通常来说性能相当

支持向量机和逻辑回归的不同点：

- LR 使用对率损失，SVM 一般用 hinge 损失

- 在 LR 的模型求解过程中，每个训练样本都对划分超平面有影响，影响力随着与超平面的距离增大而减小，所以说 **LR 的解受训练数据本身的分布影响**；SVM 的模型只与占训练数据少部分的支持向量有关，所以说，**SVM 不直接依赖数据分布**，所得的划分超平面不受某一类点的影响
- 如果数据类别不平衡比较严重，LR 需要先做相应处理再训练，SVM 则不用
- **SVM 依赖于数据表达的距离测度**，需要先把数据**标准化**，LR 则不用（但实际任务中可能会为了方便选择优化过程的初始值而进行标准化）。如果数据的距离测度不明确（特别是高维数据），那么最大间隔可能就变得没有意义
- LR 的**输出有概率意义**，SVM 的输出则没有
- LR 可以**直接用于多分类任务**，SVM 则需要进行扩展（但更常用 one-vs-rest）
- LR 使用的对率损失是光滑的单调递减函数，无法导出支持向量，解依赖于所有样本，因此预测开销较大；SVM 使用的 hinge 损失有“零区域”，因此解具有稀疏性（书中没有具体说明这句话的意思，但按我的理解是解出的拉格朗日乘子 a 具有稀疏性，而不是权重向量 w），从而不需用到所有训练样本。
- **在实际运用中，LR 更常用于大规模数据集，速度较快；SVM 适用于规模小，维度高的数据集。**

  如果 Feature 的数量很大，跟样本数量差不多，这时候选用 LR 或者是 Linear Kernel 的 SVM。
  如果 Feature 的数量比较小，样本数量一般，不算大也不算小，选用 SVM+Gaussian Kernel。
  如果 Feature 的数量比较小，而样本数量很多，需要手工添加一些 feature 变成第一种情况。

6.5 核方法

6.5.1 表示定理（representer theorem）

令 $\mathbb{H}$ 为核函数 κ 对应的再生希尔伯特空间，$\|h\|_{\mathbb{H}}$ 表示 $\mathbb{H}$ 空间中关于 h 的范数，对于任意单调递增函数 $\Omega : [0, \infty] \longmapsto \mathbb{R}$ 和任意非负损失函数 $\ell : \mathbb{R}^m \longmapsto [0, \infty]$，优化问题

$$\min_{h \in \mathbb{H}} F(h) = \Omega(\|h\|_{\mathbb{H}}) + \ell(h(\mathbf{x}_1), h(\mathbf{x}_2), ..., h(\mathbf{x}_m)) \tag{6.14}$$

的解总可写为：

$$h^*(\mathbf{x}) = \sum_{i=1}^{m} a_i \kappa(\mathbf{x}, \mathbf{x}_i)$$

这个定理表明，**旨在最小化损失和正则化项之和的优化问题，解都可以表示为核函数的线性组合。**

基于核函数的学习方法，统称为核方法（kernal methods）。**最常见的就是通过核化（引入核函数），将线性学习器扩展为非线性学习器。**

Chapter 7

第七章　贝叶斯分类器

贝叶斯分类器（Bayes Classifier）是一种通过最大化后验概率进行单点估计的分类器。

7.1　贝叶斯决策论

7.1.1　贝叶斯最优分类器

以多分类任务为例，假设有 N 种标记，即 $\mathcal{Y} = c_1, c_2, ..., c_N$，用 λ_{ij} 表示把一个真实标记为 c_i 的样本误分类为 c_j 所产生的损失。那么将样本 $\mathbf{x}$ 分类为 c_i 的**期望损失**（expected loss）或者说，在样本 $\mathbf{x}$ 上的**条件风险**（conditional risk）：

$$R(c_i|\mathbf{x}) = \sum_{j=1}^{N} \lambda_{ij} P(c_j|\mathbf{x})$$

它描述的是，给定一个样本 $\mathbf{x}$，把它分类为 c_i 需要冒多大的风险。或者说，当样本真实标记不是 c_i 时，会有多大的损失。这个损失是一个求和，每一个求和项都是某一类别的后验概率和对应误分类损失代价的积。

在单个样本条件风险的基础上，可以定义总体风险：

$$R(h) = \mathbb{E}_{\mathbf{x}}[R(h(\mathbf{x}) \mid \mathbf{x})]$$

它描述的是，所有样本的条件风险的数学期望。其中 h 是一种用于产生分类结果的判断准则。

贝叶斯判定准则（Bayes decision rule）：要最小化总体风险，只需在每个样本上选择能使对应的条件风险 $R(c \mid \mathbf{x})$ 最小的标记。即：

$$h^*(\mathbf{x}) = \arg\min_{c \in \mathcal{Y}} R(c \mid \mathbf{x})$$

h^* 称为贝叶斯最优分类器（Bayes optimal classifier），对应的总体风险 $R(h^*)$ 称为贝叶斯风险（Bayes risk），而 $1 - R(h^*)$ 则反映了分类器所能达到的最好性能，也即模型精度的理论上限。

如果学习模型的目标是令分类错误率最小，那么分类正确时误分类损失 λ_{ij} 为 0，反之为 1。这是条件风险就是：

$$R(c \mid \mathbf{x}) = 1 - P(c \mid \mathbf{x})$$

要令风险最小，只需要选择使样本 $\mathbf{x}$ 后验概率最大的一个类别标记就可以了。

7.1.2 贝叶斯定理

从概率的角度来理解，机器学习的目标就是基于有限的训练样本集尽可能准确地估计出后验概率（当然，大多数机器学习技术无需准确估计出后验概率）。

获取后验概率主要有两种策略：

- 构建判别式模型（discriminative models）：给定样本 $\mathbf{x}$，直接对后验概率 $P(\mathbf{x} \mid c)$ 建模来预测 c。这类模型包括决策树、BP 神经网络、支持向量机等等。
- 构建生成式模型（generative models）：给定样本 $\mathbf{x}$，先对联合概率分布 $P(\mathbf{x}, c)$ 建模，然后再利用联合概率计算出后验概率 $P(c \mid \mathbf{x})$，也即 $P(c \mid \mathbf{x}) = \frac{P(\mathbf{x},c)}{P(\mathbf{x})}$。

又因为联合概率 $P(\mathbf{x}, c) = P(c \mid \mathbf{x}) \times P(\mathbf{x}) = P(\mathbf{x} \mid c) \times P(c)$，由此，能得到贝叶斯定理：

$$P(c \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid c) \times P(c)}{P(\mathbf{x})}$$

其中 $P(c \mid \mathbf{x})$ 是类标记 c 相对于样本 $\mathbf{x}$ 的**条件概率**;$P(\mathbf{x} \mid c)$ 是样本 $\mathbf{x}$ 相对于类标记 c 的**类条件概率**（class-conditional probability），或称为似然（likelihood），也由于得自 c 的取值而被称作 $\mathbf{x}$ 的后验概率。$P(c)$ 是 c 的**先验概率**（也称为边缘概率），之所以称为" 先验" 是因为它不考虑任何 $\mathbf{x}$ 方面的因素。在这里又称为类先验（prior）概率。$P(\mathbf{x})$ 是 $\mathbf{x}$ 的先验概率。在这里是用作归一化的证据（evidence）因子，与类标记无关。

类先验概率 P(c) 表示的是**样本空间中各类样本的比例**，根据大数定律，当训练集包含足够多的独立同分布样本时，类先验概率可以直接通过训练集中各类样本出现的频率进行估计。

7.2 极大似然估计

估计类条件概率的一种常用策略是：先假定该类样本服从某种确定的概率分布形式，然后再基于训练集中的该类样本对假定的概率分布的参数进行估计。

如果类 c 的样本服从参数为 θ_c（可能不止一个参数）的分布，那么从样本空间抽取到该类的某一个样本 $\mathbf{x}$ 的概率就是 $P(\mathbf{x} \mid \theta_c)$。使用 D_c 来表示训练集中类 c 的子集，可以定义数据集 D_c 的**似然（likelihood）**为：

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

由于连乘操作容易造成下溢，实际任务中通常使用**对数似然（log-likelihood）**代替：

$$LL(\theta_c) = \log P(D_c \mid \theta_c) = \sum_{\mathbf{x} \in D_c} \log P(\mathbf{x} \mid \theta_c)$$

所谓**极大似然估计**（Maximum Likelihood Estimation，简称 MLE）就是找出令似然最大的参数 θ_c。也即从 θ_c 的所有可能取值中找到一个令所抽取样本出现的可能性最大的值。求解的过程，就是求似然函数的导数，令导数为 0，得到似然方程，解似然方程得到最优解，也即该类样本分布的参数。

7.3 朴素贝叶斯分类器

朴素贝叶斯分类器（naive Bayes clssifier）采用**属性条件独立性假设**（attribute conditional independence assumption）。基于这个假设，可以把类条件概率写成连乘的形式，因此贝叶斯定理可重写为：

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

其中 d 为属性数目，x_i 为样本 $\mathbf{x}$ 在第 i 个属性上的取值。

又因为 $P(\mathbf{x})$ 与类别无关，所以朴素贝叶斯分类器的表达式可以写为：

$$h(\mathbf{x}) = \arg\max_{c\in\mathcal{Y}} P(c)\prod_{i=1}^{d} P(x_i \mid c)$$

当训练集包含足够多独立同分布样本时，类先验概率 $P(c)$ 可以直接算出，也即训练集该类样本的数目占训练集规模的比例：

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

而条件概率 $P(x_i \mid c)$，根据属性类型分离散和连续两种情况：

- 离散型属性：条件概率 $P(x_i \mid c)$ 可以估计为，在类别 c 的样本子集中，第 i 个属性取值为 x_i 的样本所占的比例：

$$P(x_i \mid c) = \frac{|D_{c,x_i}|}{|D_c|} \tag{7.1}$$

- 连续性属性：替换为概率密度函数，假设第 i 个属性服从高斯分布 (正态分布)，那么条件概率就写成 $p(x_i \mid c) \sim \mathcal{N}(\mu_{c,i}, \sigma_{c,i}^2)$，其中 $\mu_{c,i}$ 和 $\sigma_{c,i}^2$ 分别是第 c 类样本在第 i 个属性上取值的均值和方差。把属性取值 x_i 代入概率密度函数就可算出条件概率：

$$p(x_i \mid c) = \frac{1}{\sqrt{2\pi}\sigma_{c,i}} \exp\left(-\frac{(x_i-\mu_{c,i})^2}{2\sigma_{c,i}^2}\right) \tag{7.2}$$

7.3.1 拉普拉斯修正

若某个属性值在训练集中没有与某个类同时出现过，那么它对应的条件概率 $P(x_i \mid c)$ 就为 0。在连乘中，这就意味着整个式子值为 0 了，其他属性携带的信息都被抹去了。此时，就需要对概率值进行平滑（smoothing）了，最常用的是拉普拉斯修正（Laplacian correction），假设训练集中包含 N 个类别，第 i 个属性包含 N_i 种取值，则拉普拉斯修正把式（1）和式（2）修改为：

$$P(c) = \frac{|D_c|+1}{|D|+N} \tag{7.3}$$

$$P(x_i \mid c) = \frac{|D_{c,x_i}|+1}{|D_c|+N_i} \tag{7.4}$$

拉普拉斯修正保证了**不会因为训练集样本不充分导致概率估值为零**。但它实际上是假设了类别和属性值是均匀分布的，相当于额外引入了先验，这个假设并不总是成立。

7.3.2 比较

朴素贝叶斯分类器和前面学习的模型有一个不同的地方就是：并不是基于训练集和某些算法来学习模型的参数；而是利用训练集来算出一些概率，在预测时，根据新样本的情况，使用不同的概率计算出它被分到各个类的后验概率，然后取后验概率最大的一个类作为结果。

在实际任务中，有两种使用方式：

- 查表：若对预测速度要求较高，可以先根据训练集把所有涉及到的概率计算出来，然后存储好，在预测新样本时只需要查表然后计算就可以了。
- 懒惰学习：若数据更替比较频繁，也可以理解为用训练集算出的概率可能很快就失效了，更新换代的速度很快，那就采取懒惰学习（lazy learning）的方式，仅当需要预测时才计算涉及到的概率。

特别地，当采取了预先计算所有概率的方式时，如果有新数据加入到训练集，只需要更新新样本涉及到的概率（或者说计数）就可以了，可以很方便地实现**增量学习**。

7.4 半朴素贝叶斯分类器

朴素贝叶斯分类器基于属性条件独立性假设，每个属性仅依赖于类别；有时候属性之间会存在依赖关系，适当考虑一部分属性间的相互依赖信息，这就是半朴素贝叶斯分类器（semi-naive Bayes classifier）的基本思想。

独依赖估计（One-Dependent Estimator，简称 ODE）是半朴素贝叶斯分类器最常用的一种策略，它假设的是每个属性在类别之外最多仅依赖于一个其他属性。也即：

$$P(c \mid \mathbf{x}) \propto P(c) \prod_{i=1}^{d} P(x_i \mid c, pa_i)$$

其中 pa_i 是属性 x_i 依赖的另一属性，称为 x_i 的父属性。若已知父属性，就可以按式（4）来计算了。

确定每个属性的父属性的方法有 SPODE 和 TAN。

7.5 贝叶斯网

贝叶斯网（Bayesian network）亦称信念网（belief network），它借助**有向无环图**（Directed Acyclic Graph，简称 DAG）来刻画属性之间的依赖关系，并使用**条件概率表**（Conditional Probability Table，简称 CPT）来描述属性的联合概率分布。

贝叶斯网的学习包括结构的学习和参数的学习，而预测新样本的过程则称为推断（inference）。

7.5.1 EM 算法

前面讨论的极大似然估计方法是一种常用的参数估计方法，它是假设分布的形式，然后用训练样本来估计分布的参数。但实际任务中，遇到一个很大的问题就是**训练样本不完整**（样本某些属性的值缺失）。这时就需要用到 EM（Expectation-Maximization）算法了。将每个属性的取值看为一个变量，那么缺失的就可以看作“未观测”变量，称作**隐变量（latent variable）**。

整个训练集可以划分为已观测变量集 X 和隐变量集 Z 两部分。按照极大似然的思路，我们依然是想找出令训练集被观测到的概率最大的参数 Θ。也即最大化对数似然：

$$LL(\Theta \mid X, Z) = \ln P(X, Z \mid \Theta)$$

但是，由于 Z 是隐变量，无法观测到，所以上面这个式子实际是没法求的。

EM 算法的步骤如下：

1. 设定一个初始的 Θ
2. 按当前的 Θ 推断隐变量 Z 的（期望）值 **（E 步）**
3. 基于已观测变量 X 和步骤 2 得到的 Z 对 Θ 做最大似然估计得到新的 Θ **（M 步）**
4. 若未收敛（比方说新的 Θ 与旧的 Θ 相差仍大于阈值），就回到步骤 2，否则停止迭代

EM 算法可以看作是用**坐标下降（coordinate descent）法**来最大化对数似然下界的过程，每次固定 Z 或者 Θ 中的一个去优化另一个，直到最后收敛到局部最优解。

7.6 补充内容

朴素贝叶斯分类器的属性条件独立性假设在现实中很难成立，但事实上它在大多数情形下都有不错的性能。关于这点，有以下两种解释：

1. 对分类任务来说，只需各类别的条件概率排序正确，即使概率值不准确，也可以产生正确的分类结果；

2. 若属性间的相互依赖对所有类别影响都相同，或者依赖关系互相抵消，则属性条件独立性假设在降低开销的同时不会给性能带来负面影响。

注意，本章讨论的**贝叶斯分类器**和一般意义上的**贝叶斯学习**（Bayesian learning）是有很大差别的，本章讨论的贝叶斯分类器只是**通过最大化后验概率来进行单点估计**，获得的仅仅是一个数值；而贝叶斯学习则是进行**分布估计**或者说**区间估计**，获得的是一个分布。

Chapter 8

第八章　集成学习

集成学习 (ensemble learning) 通过构建整合多个学习器来完成学习任务。

8.1　个体与集成

当所有个体学习器都由同样的学习算法生成时，也即集成中只包含同种类型的个体学习器时，称为**同质**（homogeneous）**集成**，这些个体学习器又被称为基学习器（base learner），相应的学习算法称为基学习算法（base learning algorithm）；

当个体学习器由不同的学习算法生成时，称为**异质**（heterogenous）**集成**，这些个体学习器称为组件学习器（component learner）或直接称为个体学习器。

集成学习通过结合多个学习器，通常能获得比单一学习器更优越的泛化性能，对弱学习器（weak learner）（略优于随机猜测的学习器，例如二分类任务中精度略高于 50%）的提升尤为明显。要获得好的集成效果，个体学习器应该"好而不同"（准确性与多样性）。

关于 Hoeffding 不等式：假设抛硬币正面朝上的概率为 p，反面朝上的概率为 $1-p$。令 $H(n)$ 代表抛 n 次硬币所得正面朝上的次数，则最多 k 次正面朝上的概率为

$$P(H(n) \leq k) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}$$

对 $\delta > 0, \quad k = (p-\delta)n$，有 Hoeffding 不等式

$$P(H(n) \leq (p-\delta)n) \leq e^{-2\delta^2 n}$$

假设基学习器的错误率相互独立，则随着集成中个体分类器数目 T 的增大，集成的错误率将指数级下降，最终趋向于零。

证明：考虑二分类问题 $y \in \{-1, +1\}$ 和真实函数 f，假定基分类器的错误率为 ϵ，即对每个基分类器 h_i 都有：

$$P(h_i(\mathbf{x}) \neq f(\mathbf{x})) = \epsilon \tag{8.1}$$

假设集成通过**简单投票法**结合 T 个基分类器，若有超过半数的基分类器分类正确，则集成分类就正确：

$$H(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{T} h_i(\mathbf{x})\right) \tag{8.2}$$

假设基分类器的错误率相互独立，结合上面的 Hoeffding 不等式：令 $p = 1-\epsilon$，$k = \lfloor \frac{T}{2} \rfloor = (p-\delta) \times T$，

可得 $\delta = p - \frac{k}{T} = 1 - \epsilon - \frac{1}{T}\lfloor\frac{T}{2}\rfloor$，所以集成的错误率为：

$$\begin{aligned} P(H(\mathbf{x}) \neq f(\mathbf{x})) &= \sum_{k=0}^{\lfloor T/2 \rfloor} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k} \\ &\leq \exp\left(-2(1-\epsilon-\frac{1}{T}\lfloor\frac{T}{2}\rfloor)\right) \\ &= \exp\left(-\frac{1}{2}T(2-2\epsilon-1)\right) \\ &= \exp\left(-\frac{1}{2}T(1-2\epsilon)^2\right) \end{aligned} \tag{8.3}$$

得证。

上面的分析中有一个关键假设：**基学习器的误差相互独立**。然而，现实任务中，个体学习器均是为解决同一个问题训练出来的，它们显然不可能相互独立。因此，**集成学习研究的核心就是：如何产生的那个并结合“好而不同”的学习器。**

根据个体学习器的生成方式，集成学习方法大致可分为两个大类：

- 个体学习器间存在强依赖关系，必须串行生成的序列化方法，例如：Boosting 算法；
- 个体学习器间不存在强依赖关系，可同时生成的并行化方法，例如：Bagging，随机森林。

8.2 Boosting

Boosting 族算法的工作机制：先从初始训练集训练出一个基学习器，再根据基学习器的表现对**样本分布**进行调整，修正先前基学习器做错的训练样本；然后基于调整后的样本分布来训练下一个基学习器；如此重复进行，直至基学习器的数目达到指定的值 T，最终将这 T 个基学习器进行加权结合。

Boosting 族算法最著名的代表是 AdaBoost 算法，其中 $y_i \in \{-1, +1\}, f$ 是真实函数。

AdaBoost 算法

输入： 训练集 $D = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \cdots, (\boldsymbol{x_m}, y_m)\}$；基学习算法 $\mathfrak{L}$；训练轮数 T.

1: $\mathcal{D}_1(\boldsymbol{x}) = \frac{1}{m}$ [$\mathcal{D}_t$ 为第 t 轮的分布]
2: **for** $t = 1, 2, \cdots, T$ **do**
3: $h_t = \mathfrak{L}(D, \mathcal{D}_t)$; [从数据集 D 中基于分布 $\mathcal{D}_t$ 训练得到分类器 h_t]
4: $\epsilon_t = P_{\boldsymbol{x}\sim\mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$ [估计 h_t 的误差]
5: **if** $\epsilon_t > 0.5$ **then**
6: ***break*** [检查当前基分类器是否比随机猜测要好]
7: **end if**
8: $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ [分类器权重更新公式]
9: $\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \exp\left(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x})\right)$ [更新样本分布，其中 Z_t 是规范化因子，以确保 $\mathcal{D}_{t+1}$ 是一个分布]
10: **end for**

输出： $H(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$

从偏差-方差分解的角度看，**Boosting 主要关注降低偏差**，因此 Boosting 能基于泛华性能相当弱的学习器构建出很强的集成。

下面介绍两种产生彼此有较大差异的基学习器的算法：Bagging 和随机森林。

8.3 Bagging

Bagging（由 Bootstrap AGGregatING 缩写而来）通过对训练样本采样，产生若干个不同的子集，从每个子集中训练出基学习器。

Bagging 直接基于**自助采样法**（bootstrap sampling）：给定包含 m 个样本的数据集，先随机取出一个样本放入采样集中，再把样本**放回**初始数据集，使得下次采样时该样本仍有可能被选中，这样经过 m 次随机采样后，得到含有 m 个样本的采样集。初始训练集中的样本在采样集里有的多次出现，有的从未出现，约有 63.2% 的样本出现在采样集里。

Bagging 算法的基本流程：先采样出 T 个含有 m 个训练样本的采样集，然后基于每个采样集训练出一个基学习器，再将这些基学习器进行结合。在结合时，Bagging 通常对分类任务使用**简单投票法**，对回归任务使用**简单平均法**。

Bagging 算法

输入： 训练集 $D = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \cdots, (\boldsymbol{x_m}, y_m)\}$；基学习算法 $\mathfrak{L}$；训练轮数 T.

1: **for** $t = 1, 2, \cdots, T$ **do**

2: $\quad h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$； $\quad$ [$\mathcal{D}_{bs}$ 是自主采样产生的样本分布]

3: **end for**

输出： $H(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$

与标准 AdaBoost 只适用于二分类任务不同的是，Bagging 能不经修改地用于多分类、回归等任务。

从偏差-方差分解的角度看，Bagging 主要关注**降低方差**，因此它在不剪枝决策树、神经网络等易受样本扰动的学习器上效果更明显。

8.4 随机森林

随机森林（(Random Forest, 简称 RF) 是 Bagging 的一个扩展体：在以决策树为基学习器构建 Bagging 集成的基础上，进一步在决策树的训练过程中引入了**随机属性选择**。因此，随机森林中基学习器的多样性不仅来自样本扰动，还来自属性扰动，这就使得最终集成的泛化性能可通过个体学习器之间差异度的增加而进一步提升。

随机森林的训练效率通常优于 Bagging。

8.5 结合策略

学习器结合的好处：

- 从统计方面看：由于学习任务的假设空间往往很大，可能有多个假设在训练集上达到同等性能，结合多个学习器可以降低使用单学习器可能因误选而导致泛化性能不佳的风险；
- 从计算方面看：结合可以降低陷入糟糕局部极小点的风险
- 从表示方面看：某些学习任务的真实假设可能不在当前学习算法所考虑的假设空间中，使用多个学习器结合，由于相应的假设空间有所扩大，有可能学得更好的近似。

假定集成包含 T 个基学习器 $\{h_1, h_2, ..., h_T\}$，其中 h_i 在示例 $\boldsymbol{x}$ 上的输出为 $h_i(\boldsymbol{x})$。下面是几种对 h_i 进行结合的常见策略。

8.5.1 平均法

- 简单平均法 (simple averaging)： $H(\boldsymbol{x}) = \frac{1}{T} \sum_{i=1}^{T} h_i(\boldsymbol{x})$
- 加权平均法 (weighted averaging)： $H(\boldsymbol{x}) = \frac{1}{T} \sum_{i=1}^{T} w_i h_i(\boldsymbol{x})$

其中 w_i 是个体学习器 h_i 的权重，通常要求 $w_i \geq 0, \sum_{i=1}^{T} w_i = 1$。

8.5.2 投票法

- 绝对大多数投票法：所有分类器在某标记上的得票超过半数，则预测为该标记，否则拒绝预测。
- 相对多数投票法：预测为得票最多的标记。
- 加权投票法

8.5.3 学习法

学习法即通过另一个学习器来结合基学习器，一个典型代表是 Stacking。把个体学习器称为初级学习器，把用于结合的学习器称为次级学习器或元学习器 (meta-learner)。

Stacking 算法

输入： 训练集 $D = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \cdots, (\boldsymbol{x_m}, y_m)\}$；初级学习算法 $\mathfrak{L}_1, \mathfrak{L}_2, \cdots, \mathfrak{L}_T$；次级学习算法 $\mathfrak{L}$.

for $t = 1, 2, \cdots, T$ **do**
  $h_t = \mathfrak{L}_t(D)$; [使用初级学习算法 $\mathfrak{L}_t$ 产生初级学习器 h_t]
end for
$D' = \varnothing$ [D' 是次级训练集]
for $i = 1, 2, \cdots, m$ **do**
  for $t = 1, 2, \cdots, T$ **do**
    $z_{it} = h_t(\boldsymbol{x}_i)$
  end for
  $D' = D' \cup ((z_{i1}, z_{i2}, \cdots, z_{iT}), y_i)$
end for
$h' = \mathfrak{L}(D')$

输出： $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \cdots, h_T(\boldsymbol{x}))$

次级训练集是由初级学习器产生的，如果直接用初级学习器的训练集来产生次级训练集，则很有可能产生过拟合。

将初级学习器的输出类概率作为次级学习器的输入属性，用多响应线性回归 (Multi-response Linear Regression，简称 MLR) 作为次级学习算法效果较好。

MLR 是基于线性回归的分类器，它对每个类分别进行线性回归，属于该类的训练样例所对应的输出被置位 1，其他类置为 0；测试样例将被分给输出值最大的类。

Chapter 9

第九章　聚类

常见的无监督学习任务有：聚类 (clustering)，密度估计 (density estimation)，异常检测 (anomaly detection) 等。

9.1　聚类性能度量

聚类的期望结果：簇内相似度 (intra-cluster similarity) 高且簇间相似度 (inter-cluster similarity) 低。

聚类性能度量分为两类：

- 将聚类结果与某个“参考模型”(reference model) 进行比较，称为外部指标 (external index)，主要考虑两个模型分类结果的相似性和相异性；
- 直接考察聚类结果而不利用任何参考模型，称为内部指标 (internal index)，主要考虑类间的“距离”。

9.2　距离计算

用函数 $\mathrm{dist}(\cdot,\cdot)$ 计算两个样本间的距离。定义“距离度量”(distance measure) 是满足下面性质的函数：

- 非负性：$\mathrm{dist}(\boldsymbol{x}_i,\boldsymbol{x}_j)\geq 0$
- 同一性：$\mathrm{dist}(\boldsymbol{x}_i,\boldsymbol{x}_j)=0$ 当且仅当 $\boldsymbol{x}_i=\boldsymbol{x}_j$
- 对称性：$\mathrm{dist}(\boldsymbol{x}_i,\boldsymbol{x}_j)=\mathrm{dist}(\boldsymbol{x}_j,\boldsymbol{x}_i)$
- 直递性 (三角不等式)：$\mathrm{dist}(\boldsymbol{x}_i,\boldsymbol{x}_j)\leq \mathrm{dist}(\boldsymbol{x}_i,\boldsymbol{x}_k)+\mathrm{dist}(\boldsymbol{x}_k,\boldsymbol{x}_j)$

给定样本 $\boldsymbol{x}_i=(x_{i1};x_{i2};\cdots;x_{in})$ 与 $\boldsymbol{x}_j=(x_{j1};x_{j2};\cdots;x_{jn})$，定义闵可夫斯基距离 (Minkowski distance)：

$$\mathrm{dist}_{mk}(\boldsymbol{x}_i,\boldsymbol{x}_j)=\left(\sum_{u=1}^{n}|x_{iu}-x_{ju}|^p\right)^{\frac{1}{p}}$$

对于 $\forall p\geq 1$，闵可夫斯基距离均满足距离度量的性质。$p=1$ 即为曼哈顿距离 (Manhattan distance)，$p=2$ 即为欧氏距离 (Euclidean distance).

连续属性可在属性值上直接计算距离，这样的属性称为“有序属性”，离散属性无法在属性值上计算距离，称为“无序属性”。闵可夫斯基距离可用于有序属性。

对无序属性采用 VDM(Value Difference Metric)。令 $m_{u,a}$ 表示在属性 u 上取值为 a 的样本数，$m_{u,a,i}$ 表示在第 i 个样本簇中在属性 u 上取值为 a 的样本数，k 为样本簇数，则属性 u 上两个离散值 a 与 b 之间的 VDM 距离为：

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

绝对值里即是每个样本簇上属性 u 取值为 a 的比例减去每个样本簇上属性 u 取值为 b 的比例。

9.3 原型聚类

假设聚类结构能通过一组原型刻画，算法先对原型进行初始化，然后对原型进行迭代更新求解。

9.3.1 k 均值算法

给定样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m\}$，$k$ 均值 (k-means) 算法针对聚类所得簇划分 $\mathcal{C} = \{C_1, C_2, \cdots, C_k\}$ 最小化平方误差：

$$E = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|_2^2 \tag{9.1}$$

其中 $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$ 是簇 C_i 的均值向量。E 的值越小，簇内样本相似度越高。

最小化式子 (9.1) 是 NP 难问题，k 均值算法采用贪心策略，通过迭代优化来近似求解。

---

k 均值算法

输入： 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m\}$，聚类簇数 k。

1: 从 D 中随机选择 k 个样本作为初始均值向量 $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_k\}$
2: **repeat**
3: 令 $C_i = \varnothing (1 \le i \le k)$
4: **for** $j = 1, 2, \cdots, m$ **do**
5: 计算样本 $\boldsymbol{x}_j$ 与各均值向量 $\boldsymbol{\mu}_i (1 \le i \le k)$ 的距离：$d_{ji} = \|\boldsymbol{x}_j - \boldsymbol{\mu}_i\|_2$；
6: 根据距离最近的均值向量确定 $\boldsymbol{x}_j$ 的簇标记：$\lambda_j = \arg \min_{i \in \{1,2,\cdots,k\}} d_{ji}$；
7: 将样本 $\boldsymbol{x}_j$ 划入相应的簇：$C_{\lambda_j} = C_{\lambda_j} \cup \{\boldsymbol{x}_j\}$;
8: **end for**
9: **for** $i = 1, 2, \cdots, k$ **do**
10: 计算新均值向量：$\boldsymbol{\mu}'_i = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$
11: 更新均值向量：$\boldsymbol{\mu}_i = \boldsymbol{\mu}'_i$
12: **end for**
13: **until** 当前均值向量均未更新

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

9.3.2 学习向量量化

与一般聚类算法不同的是，学习向量量化 (Learning Vector Quantization，简称 LVQ) **假设数据样本带有标记信息**。

9.4 密度聚类

密度聚类算法假设聚类结构能通过样本分布的紧密程度确定，从样本密度的角度考察样本之间的连续性，并基于可连接样本不断扩展聚类簇以获得最终的聚类结果。

LVQ 算法

输入: 样本集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
原型向量个数 q，各原型向量预设的类别标记 $\{t_1, t_2, \cdots, t_q\}$;
学习率 $\eta \in (0, 1)$。

初始化一组原型向量 $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_q\}$，可以随机选择 q 原型样本
repeat
  从样本集 D 中随机选择样本 $(\boldsymbol{x}_i, y_j)$;
  计算样本 $\boldsymbol{x}_j$ 与 $\boldsymbol{p}_i (1 \leq i \leq q)$ 的距离：$d_{ji} = \|\boldsymbol{x}_j - \boldsymbol{p}_i\|^2$
  找出与 $\boldsymbol{x}_j$ 距离最近的原型向量 $\boldsymbol{p}_{i*}$
  if $y_j = t_{i*}$ **then**
    $\boldsymbol{p}' = \boldsymbol{p}_{i*} + \eta \cdot (\boldsymbol{x}_j - \boldsymbol{p}_{i*})$ $\boldsymbol{x}_j$ 与 $\boldsymbol{p}_{i*}$ 的类别相同
  else
    $\boldsymbol{p}' = \boldsymbol{p}_{i*} - \eta \cdot (\boldsymbol{x}_j - \boldsymbol{p}_{i*})$ $\boldsymbol{x}_j$ 与 $\boldsymbol{p}_{i*}$ 的类别不同
  end if
  将原型向量 $\boldsymbol{p}_{i*}$ 更新为 $\boldsymbol{p}'$
until 满足停止条件，如迭代的次数等。

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

DBSCAN 算法 (Density-Based Spatial Clustering of Applications with Noise) 基于“邻域”刻画样本分布的紧密程度。

- $\boldsymbol{x}_j$ 的 ϵ-邻域是指样本集 D 中与 $\boldsymbol{x}_j$ 的距离不大于 ϵ 的样本构成的集合。
- 如果 $\boldsymbol{x}_j$ 的 ϵ-邻域至少包含 $MinPts$ 个样本，则称 $\boldsymbol{x}_j$ 是一个核心对象。
- $\boldsymbol{x}_j$ 在 $\boldsymbol{x}_i$ 的 ϵ-邻域中，且 $\boldsymbol{x}_i$ 是核心对象，则称 $\boldsymbol{x}_j$ 可由 $\boldsymbol{x}_i$ 密度直达。**通常密度直达关系不满足对称性。**
- $\boldsymbol{x}_j$ 可由 $\boldsymbol{x}_i$ 密度可达是指存在从 $\boldsymbol{x}_i$ 到 $\boldsymbol{x}_j$ 的密度直达序列。
- $\boldsymbol{x}_j$ 和 $\boldsymbol{x}_i$ 密度相连是指存在 $\boldsymbol{x}_k$ 使得 $\boldsymbol{x}_j$ 和 $\boldsymbol{x}_i$ 均可由 $\boldsymbol{x}_k$ 密度可达。

DBSCAN 的“簇”定义为由密度可达关系导出的最大的密度相连样本集合：**把一个点所有可达的点加入这个集合，并把可达点的所有可达点也加入**。初始点是核心对象集中的随机一个。

9.5 层次聚类

数据集的划分策略：自底向上的聚合策略和自顶向下的分拆策略。

AGNES(AGglomerative NESting 的缩写) 是一种自底向上聚合策略的层次聚类算法：先将数据集中的每个样本看做一个初始聚类簇，然后在算法运行的每一步中找出**距离最近**两个聚类簇进行合并，不断重复，直至达到预设的聚类簇个数。

两个聚类簇的距离有：最小距离 (两个簇的最近样本距离确定)，最大距离 (两个簇的最远样本距离确定)，平均距离 (两个簇的所有样本共同确定)。相应的 AGNES 算法依次称为单链接，全链接和均链接算法。

9.6 常用的距离

9.6.1 切比雪夫距离

切比雪夫距离 (Chebyshev distance) 是空间中的一种度量，二个点之间的距离定义为其各坐标数值差的最大值。

$$D(\boldsymbol{p},\boldsymbol{q}) = \max_i |\boldsymbol{p}_i - \boldsymbol{q}_i| = \lim_{k\to\infty}\left(\sum_{i=1}^{n}|\boldsymbol{p}_i-\boldsymbol{q}_i|^k\right)^{\frac{1}{k}}$$

9.6.2 马氏距离

马氏距离 (Mahalanobis distance) 表示数据的协方差距离。它是一种有效的计算两个未知样本集的相似度的方法。与欧氏距离不同的是它考虑到各种特性之间的联系（例如：一条关于身高的信息会带来一条关于体重的信息，因为两者是有关联的）并且是尺度无关的 (scale-invariant)，即独立于测量尺度。

有 m 个样本向量 $\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_m$，协方差矩阵记为 S，均值记为向量 $\boldsymbol{\mu}$，则其中一个样本向量 $\boldsymbol{x}_i$ 到 $\boldsymbol{\mu}$ 的马氏距离表示为：

$$D(\boldsymbol{x}_i) = \sqrt{(\boldsymbol{x}_i-\boldsymbol{\mu})^T S^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})}$$

如果协方差矩阵为单位矩阵，马氏距离就简化为欧式距离，如果协方差矩阵为对角阵，其也可称为正规化的马氏距离。马氏距离的优缺点：**量纲无关，排除变量之间的相关性的干扰**。

9.6.3 余弦相似度

余弦相似度 (Cosine Similarity)，又称为余弦相似性，是通过计算两个向量的夹角余弦值来评估他们的相似度。余弦相似度将向量根据坐标值，绘制到向量空间中，如最常见的二维空间。

$$\cos(\theta) = \frac{\sum_{i=1}^{n}\boldsymbol{x}_{1i}\times\boldsymbol{x}_{2i}}{\sqrt{\sum_{i=1}^{n}\boldsymbol{x}_{1i}^2}\times\sqrt{\sum_{i=1}^{n}\boldsymbol{x}_{2i}^2}}$$

两个向量有相同的指向时，余弦相似度的值为 1；两个向量夹角为 90° 时，余弦相似度的值为 0；两个向量指向完全相反的方向时，余弦相似度的值为-1。这结果是**与向量的长度无关的，仅仅与向量的指向方向相关**。

9.6.4 汉明距离

汉明距离 (Hamming distance)：两个等长字符串 s1 与 s2 之间的汉明距离定义为将其中一个变为另外一个所需要作的最小替换次数。例如字符串“1111”与“1001”之间的汉明距离为 2。可以利用动态规划求解。

应用：信息编码 (为了增强容错性，应使得编码间的最小汉明距离尽可能大)。

9.6.5 杰卡德相似系数

杰卡德相似系数 (Jaccard similarity coefficient) 表示两个集合 A 和 B 的交集元素在 A 与 B 的并集中所占的比例，用符号 $J(A,B)$ 表示。

$$J(A,B) = \frac{|A\cap B|}{|A\cup B|}$$

杰卡德相似系数是衡量两个集合的相似度一种指标。

与杰卡德相似系数相反的概念是杰卡德距离 (Jaccard distance)。杰卡德距离可用如下公式表示：

$$J_\delta(A,B) = 1 - J_{A,B}$$

9.6.6 皮尔森相关系数

皮尔森相关系数 (Pearson correlation coefficient)：也称皮尔森积矩相关系数 (Pearson product-moment correlation coefficient)，是一种线性相关系数。皮尔森相关系数是用来反映两个变量线性相关程度的统计量，用 r 表示。

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{x})(Y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{y})^2}}$$

其中 n 为样本量。r 描述的是两个变量间线性相关强弱的程度。r 的绝对值越大表明相关性越强。

9.6.7 K-L 散度

K-L 散度 (Kullback-Leibler Divergence)：即相对熵；是衡量两个分布 (P、Q) 之间的距离；越小越相似。

$$D(P\|Q) = \sum_{i=1}^{n} P(i) \log \frac{P(i)}{Q(i)}$$

Chapter 10

第十章　降维与度量学习

10.1　k 近邻学习

k 近邻 (k-Nearest Neighbor，简称 kNN) 学习是无监督学习方法，也是“懒惰学习”的著名代表。假设样本独立同分布，可以得到结论：**最近邻分类器的泛化错误率不超过贝叶斯最优分类器的错误率的两倍**。

令训练样本为 $\boldsymbol{z}$，$c^* = \arg\max_{c\in\mathcal{Y}} P(c|\boldsymbol{x})$ 表示贝叶斯最优分类器的结果，则：

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

10.2　低维嵌入

使用 kNN 时需要满足样本密度足够大，然而高维数据很容易出现样本稀疏，距离计算困难的问题，此时称为“维数灾难”(curse of dimensionality)。缓解维数灾难的一个方法是降维，降维时需要保证原始空间样本中的距离信息在低维空间中同样存在。

10.2.1　多维缩放降维法

多维缩放 (Multiple Dimensional Scaling，简称 MDS) 降维方法是基于线性变换进行降维的线性降维方法。

令 m 个样本在原始空间的距离矩阵为 $\boldsymbol{D} \in \mathbb{R}^{m\times m}$，其第 i 行 j 列的元素为 $dist_{ij}$ 为样本 $\boldsymbol{x}_i$ 到样本 $\boldsymbol{x}_j$ 的距离，降维后在 d' 维空间中的表示是 $\boldsymbol{z} \in \mathbb{R}^{d'\times m}, d' \le d$，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 $\|\boldsymbol{z}_i - \boldsymbol{z}_j\| = dist_{ij}$。

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

$$dist_{ij}^2 = \|\boldsymbol{z}_i\|^2 + \|\boldsymbol{z}_j\|^2 - 2\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_j = b_{ii} + b_{jj} - 2b_{ij}$$

设降维后的样本 $\boldsymbol{Z}$ 被中心化，即 $\sum_{i=1}^m \boldsymbol{z}_i = \boldsymbol{0}$，显然矩阵 $\boldsymbol{B}$ 的行和列之和均为零，即 $\sum_{i=1}^m b_{ij} = \sum_{j=1}^m b_{ij} = 0$。可以得到：

$$\sum_{i=1}^m dist_{ij}^2 = tr(\boldsymbol{B}) + mb_{jj}$$

$$\sum_{j=1}^{m} dist_{ij}^2 = tr(\boldsymbol{B}) + mb_{ii}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{m} dist_{ij}^2 = 2m\ tr(\boldsymbol{B})$$

其中 $tr(\boldsymbol{B}) = \sum_{i=1}^{m}\|\boldsymbol{z}_i\|^2$。令

$$dist_{i\cdot}^2 = \frac{1}{m}\sum_{j=1}^{m} dist_{ij}^2$$

$$dist_{\cdot j}^2 = \frac{1}{m}\sum_{i=1}^{m} dist_{ij}^2$$

$$dist_{\cdot\cdot}^2 = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} dist_{ij}^2$$

带入前面可得：

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

所以可以通过降维前后保持距离不变的距离矩阵 $\boldsymbol{D}$ 来求取内积矩阵 $\boldsymbol{B}$。

对矩阵 $\boldsymbol{B}$ 做特征值分解：$\boldsymbol{B} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathrm{T}}$，其中 $\boldsymbol{\Lambda}$ 是特征值构成的对角矩阵。选取其中的 d^* 个非零特征值，构成对角矩阵 $\boldsymbol{\Lambda}_*$，并令 $\boldsymbol{V}_*$ 为其对应的特征向量矩阵，则有：

$$\boldsymbol{Z} = \boldsymbol{\Lambda}_*^{\boldsymbol{1/2}}\boldsymbol{V}_*^{\boldsymbol{T}} \in \mathbb{R}^{d^*\times m}$$

10.3 主成分分析

主成分分析 (Principal Component Analysis，简称 PCA) 也是一种常用的降维方法。主成分分析的目标是：

$$\max_{\boldsymbol{W}} \mathrm{tr}(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{W}) \qquad \text{s.t.} \quad \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}$$

可以从**最近重构性** (样本点到划分超平面的距离足够近) 和**最大可分性** (样本点在超平面的投影尽可能分开) 两个角度得到上面的分析目标。从最大可分性的角度来看，样本点 $\boldsymbol{x}_i$ 在新空间中超平面上的投影为 $\boldsymbol{W}^T\boldsymbol{x}_i$，若所有样本点的投影尽可能分开，则应该使投影后样本点的方差最大化，而投影后的方差就是：

$$\sum_i \boldsymbol{W}^T\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{W}$$

PCA 的求解方法：对协方差矩阵 $\boldsymbol{X}\boldsymbol{X}^T$ 进行特征值分解，并将取前 d' 大的 d' 个特征值对应的特征向量构成坐标转化矩阵 $\boldsymbol{W}$。

10.4 核化线性降维

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”**。此时需要一个能够**显示**表示的核函数。

10.5 流形学习 (manifold learning)

流行是在局部与欧氏空间同胚的空间，即：在局部空间具有欧氏空间的性质，能用欧氏距离来进行距离计算。流行学习也可用于可视化。

10.5.1 等度量映射 (Isomap)

等度量映射 (Isometric Mapping，简称 Isomap) 的基本出发点，是认为低维流形嵌入到高维空间后，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行上是不可达的**。

低维嵌入流行上两点间的距离是"测地线"距离：样本空间曲面上的最短曲线距离。计算方法：利用流行在局部上与欧式空间同胚这个性质，对每个点基于欧氏距离找出其近邻点，然后建立近邻点连接图，途中近邻点之间存在连接，而非近邻点之间不存在连接。于是计算测地线距离就变成了**计算近邻连接图上两点之间最短路径问题**。可以使用 Dijkstra 算法、Floyd 算法等。

对近邻图的构造可以指定近邻的个数，也可以设置近邻距离阈值。**Isomap 保持了近邻样本之间的距离**。

10.5.2 局部线性嵌入 (LLE)

局部线性嵌入 (Locally Linear Embedding，简称 LLE) 保持了近邻样本之间的线性关系。

LLE 算法

输入: 样本集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_m,y_m)\}$;
近邻参数 k;
低维空间维数 d'。

初始化一组原型向量 $\{\boldsymbol{p}_1,\boldsymbol{p}_2,\cdots,\boldsymbol{p}_q\}$，可以随机选择 q 原型样本
for $i=1,2,\cdots,m$ **do**
  确定 $\boldsymbol{x}_i$ 的 k 近邻
  通过式子: $\min\limits_{\boldsymbol{w}_1,\boldsymbol{w}_2,\cdots,\boldsymbol{w}_m}\sum_{i=1}^{m}\|\boldsymbol{x}_i-\sum_{j\in Q_i}w_{ij}\boldsymbol{x}_j\|_2^2 \quad s.t. \quad \sum_{j\in Q_i}w_{ij}=1$ 求得 $w_{ij},j\in Q_i$
  对于 $j\notin Q_i$，令 $w_{ij}=0$
end for
计算 $\boldsymbol{M}=(\boldsymbol{I}-\boldsymbol{W})^T(\boldsymbol{I}-\boldsymbol{W})$
对 $\boldsymbol{M}$ 进行特征值分解
得到 $\boldsymbol{M}$ 的最小 d' 个特征值对应的特征向量

输出: 样本空间集 D 在低维空间的投影 $Z=\{\boldsymbol{z}_1,\boldsymbol{z}_2,\cdots,\boldsymbol{z}_m\}$

10.6 度量学习

度量学习 (metric learning)：每个空间对应了在样本属性上定义的一个距离度量，寻找合适的空间，也就是在寻找一个合适的距离度量。

要对距离度量进行学习，首先需要有一个便与学习的距离度量表达式，第九章总结了常用的距离计算方式。近邻成分分析 (Neighbourhood Component Analysis，简称 NCA) 是基于马氏距离的分析。

10.7 总结

懒惰学习方法主要有 k 近邻学习器，懒惰决策树；朴素贝叶斯分类器能以懒惰学习方式使用，也可以急切学习方式使用。

主成分分析是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)，其核化版本是 KLDA。

PCA 的优点：算法简单，具有线性误差；在计算过程中不需要人为的设定参数或是根据任何经验模型对计算进行干预，最后的结果只与数据相关，与用户是独立的；主成分可以用作多元回归和聚类分析的输入。PCA 的问题：存储空间大，计算复杂度高；有可能投影以后对数据的区分作用并不大，反而可能使得数据点揉杂在一起无法区分。

LDA 可以用于分类工作，但是对于样本维数大于样本数时不好处理。

常用的流行学习方法有：Isomap,LLE, 拉普拉斯特征映射，局部空间对齐等。

Isomap 虽然具有拓扑不稳定性，计算复杂，对噪声敏感，但仍然被广泛采用，效果良好。拉普拉斯特征映射思想简单，计算也简单，但是要求观测数据采样稠密，对噪声敏感性很大。

流形学习中非线性维数约简方法与线性维数约简方法相比的一个显著特点是分析中的**局部性**。现有非线性维数约简方法大多基于小的邻域学习, 期望通过在小邻域上的学习得到一个全局的坐标, 这往往是不现实的。因此非线性降维的一个研究方向是：如何将全局与局部数据学习结合起来。

Chapter 11

第十一章　特征选择与稀疏学习

特征选择的原因：必要性 (特征过多造成维数灾难) 和有利性 (去除冗余特征减低学习难度)。

11.1　子集搜索与评价

11.1.1　子集搜索

前向搜索：对于给定特征集，第一次选择最优的一个特征，使得在子集评价下最优，第二次从剩下的特征中再选择一个特征，使得这两个特征组成的特征子集在子集评价下最优，并且比一个特征时更优，依次进行，直到多选一个特征构成子集不更优或者选择完所有特征。

后向搜索：从全体特征子集中每次剔除一个特征，直到剩下的特征子集再剔除一个时效果更差。前向搜索和后向搜索结合就是双向搜索：一边增加选定的特征，一边删除无关特征。

无论是前/后/双向搜索都是基于贪心策略的。

11.1.2　子集评价

给定数据集 D，假定样本属性为离散型且 D 中第 i 类样本所占的比例为 $p_i (i = 1, 2, \cdots, |\mathcal{Y}|)$。对于属性子集 A，假定根据取其值将 D 分成了 V 个子集 $\{D^1, D^2, \cdots, D^V\}$, 每个子集中的样本在 A 上取值相同，计算属性子集 A 的信息增益为:

$$\mathrm{Gain}(A) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Ent}(D^v)$$

其中信息熵定义为:

$$\mathrm{Ent(D)} = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k$$

信息增益 $\mathrm{Gain}(A)$ 越大，意味着特征子集 A 包含的有助于分类的信息越多，所以信息增益越大越好。

11.2　特征选择方法

特征选择方法：特征子集搜索机制与子集评价机制相结合，例如决策树就是一种特征选择方法。常见的特征选择方法可分为三类：过滤式 (filter)，包裹式 (wrapper) 和嵌入式 (embedding)。

11.2.1 过滤式选择

先对数据集进行特征选择，然后再训练学习器，特征选择过程和后续学习器无关。**Relief**(Relevant Features) 是一种著名的过滤式特征选择方法。Relief 给每个属性 j 定义了一个“相关统计量”:

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

其中 x_a^j 表示样本 $\boldsymbol{x}_a$ 在属性 j 上的取值，$\boldsymbol{x}_{i,nh}$ 表示 $\boldsymbol{x}_i$ 在同类样本中的最近邻 (称为猜中近邻,near-hit)，$\boldsymbol{x}_{i,nm}$ 表示 $\boldsymbol{x}_i$ 的异类样本中的最近邻 (称为猜错近邻,near-miss)。$\text{diff}(x_a^j, x_b^j)$ 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，$\text{diff}(x_a^j, x_b^j)$ 为 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 均已规范化到区间 $[0, 1]$。

相关统计量越大分类能力越强。Relief 的时间开销随采样次数以及原始特征线性增长。

11.2.2 包裹式选择

包裹式特征选择直接把最终将要使用的学习器性能作为特征子集的评价准则。LVW(Las Vegas Wrapper) 是一个典型的包裹式特征选择方法。它在**拉斯维加斯方法** (Las Vegas method) 框架下使用随机策略来进行子集搜索，并最终分类器的误差为特征子集评价准则。

即：每次随机选择一个子集，如果这个子集在最终学习器上的误差更小，或者误差相同但是子集的基数更小那就替换这个子集为最终要选择的特征子集，直到随机选择的迭代次数到达给定值。

因为包裹式选择是针对学习器优化的，所以一般要比过滤式选择更好；但是因为包裹式选择中需要多次训练学习器，因此包裹式特征选择的**计算开销通常比较大**。

11.2.3 嵌入式选择

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，即：**引入正则化向**。

对于给定数据集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$，其中 $\boldsymbol{x} \in \mathbb{R}^d, y \in \mathbb{R}$。对于线性回归模型以平方误差为损失函数，则优化目标为:

$$\min_{\boldsymbol{w}} \sum_{i=1}^{m} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2$$

为了缓解过拟合问题，引入 L2 范数正则化，则有:

$$\min_{\boldsymbol{w}} \sum_{i=1}^{m} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

其中正则化参数 $\lambda > 0$，上式称为**岭回归** (ridge regression)。

若将 L2 正则化替换为 L1 正则化，则称为 LASSO(Least Absolute Shrinkage and Selection Operator)。

11.2.4 L0,L1,L2 正则化比较

L1 和 L2 都可以防止过拟合：L1 是舍弃了一些不重要的点；L2 是控制所有特征的权重。

L1 比 L2 更容易获得“稀疏解”，即它求得的 $\boldsymbol{w}$ 会有更少的非零向量。实际上，L0 比 L1 更容易获得稀疏解，但是通常不用 L0，因为 L0 范数不连续，难以优化求解，通常用 L1 来近似。

理解 L1 比 L2 更容易获得稀疏解即是理解 L1 情况下的损失函数为最小值的 $\boldsymbol{w}$ 解中有更多的 $w_i = 0$；也就是有更多的 $w_i = 0$ 使损失函数成立。因为损失函数是极小值，那么就要使更多的 $w_i = 0$ 的导数为 0，或者导数左右异号。如果原来的导数不为 0，那么 L2 正则化也不会为 0。【因为 $(cw_i^2)' = 2cw_i = 0$，相当于加正则项导数为 0。】而 L1 正则化有可能使导数左右异号【因为 $|cw_i|' = c \times \text{sign}(w_i) \times w_i$，只要 c 大于原来 0 处的导数的绝对值，$w_i = 0$ 就是一个极小值。】

11.3 稀疏表示与字典学习

稀疏表示就是当把数据集 D 考虑成一个矩阵时，每一行表示一个样本，每一列表示一个属性，尽可能去除特征无关的列，使得学习任务的难度有所降低，涉及的计算和存储开销也减少。

稀疏表示的好处：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，恰是因为文本数据在使用字频表示 (每个字看做一个特征，字在文档中出现的频率或次数作为特征的取值) 后具有高度的稀疏性，使大多数问题变得线性可分。**

字典学习 (dictionary learning)：为普通稠密表达的样本找到合适的字典 (以用于类似上面的字频表示)，使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。

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

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

其中 $\boldsymbol{B} \in \mathbb{R}^{d \times k}$ 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 $\boldsymbol{x}_i \in \mathbb{R}^d$ 的稀疏表示。上述式子中的第一项是希望 $\boldsymbol{\alpha}_i$ 能很好地重构 $\boldsymbol{x}_i$，第二项则是希望 $\boldsymbol{\alpha}_i$ 尽量稀疏。

11.4 压缩感知

压缩感知 (compressed sensing) 分为“感知测量”和“重构恢复”两个阶段。感知测量是指对原始信号进行处理以获得稀疏样本表示；而重构恢复则是基于稀疏性从少量观测中恢复原始信号，这是压缩感知的关键部分。

矩阵补全 (matrix completion) 技术：

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

其中 $\boldsymbol{X}$ 表示要恢复的稀疏信号，$\boldsymbol{A}$ 是已知的信号；Ω 是 $\boldsymbol{A}$ 中已知元素的下标集合。约束项表示恢复出的矩阵中 $(\boldsymbol{X})_{ij}$ 应当与观测到的对应元素相同。

求解上式是 NP 难的，但是注意到 $\text{rank}(\boldsymbol{X})$ 在集合 $\{\boldsymbol{X} \in \mathbb{R}^{m \times n} : \|\boldsymbol{X}\|_F^2 \leq 1\}$($\|\cdot\|_F$ 表示矩阵的 Frobenius 范数，即每个元素平方和后再开平方) 上的凸包是 $\boldsymbol{X}$ 的核范数：

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

其中 $\sigma_j(\boldsymbol{X})$ 表示 $\boldsymbol{X}$ 的奇异值，即矩阵的核范数为矩阵的奇异值之和，于是可以通过最小化矩阵核范数来近似求解矩阵补全的式子，即：

$$\min_{\boldsymbol{X}} \|\boldsymbol{X}\|_* \qquad \text{s.t.} \quad (\boldsymbol{X})_{ij} = (\boldsymbol{A})_{ij}, \quad (i,j) \in \Omega$$

上面的式子是一个**凸优化**问题，可以通过**半正定规划**求解。

Chapter 12

计算学习理论

12.1 基础知识

12.1.1 Jensen 不等式

对任意凸函数 $f(x)$ 有:

$$f(\mathbb{E}(x)) \leq \mathbb{E}(f(x))$$

先证明这个结论: $f(x)$ 是凸函数 $(f^{''}(x) \geq 0)$, 证明:

$$p_1 f(x_1) + p_2 f(x_2) \geq f(p_1 x_1 + p_2 x_2) \qquad \text{s.t. } p_1 + p_2 = 1, 0 < p_1 < 1$$

证明: 当 $p1 = p2$ 时, 等号成立。不妨设 $x_2 > x_1$, 则:

$$p_1 x_1 + p_2 x_2 - x_1 = (p_2 x_2) - (1 - p_1) x_1 = p_2 (x_2 - x_1) > 0 \tag{12.1}$$

所以: $p_1 x_1 + p_2 x_2 > x_1$, 所以有: $x_1 < z_1 < p_1 x_1 + p_2 x_2$。
由拉格朗日中值定理:

$$\frac{f(p_1 x_1 + p_2 x_2) - f(x_1)}{(p_1 x_1 + p_2 x_2) - x_1} = f'(z_1)$$

即:

$$f(p_1 x_1 + p_2 x_2) - f(x_1) = [(p_1 x_1 + p_2 x_2) - x_1] f'(z_1) = p_2 (x_2 - x_1) f'(z_1) \tag{12.2}$$

同理:

$$p_1 x_1 + p_2 x_2 - x_2 = (p_1 x_1) - (1 - p_2) x_2 = p_1 (x_1 - x_2) < 0 \tag{12.3}$$

所以: $p_1 x_1 + p_2 x_2 < x_2$, 所以有: $p_1 x_1 + p_2 x_2 < z_2 < x_2$ 。
由拉格朗日中值定理:

$$\frac{f(x_2) - f(p_1 x_1 + p_2 x_2)}{x_2 - (p_1 x_1 + p_2 x_2)} = f'(z_2)$$

即:

$$f(x_2) - f(p_1 x_1 + p_2 x_2) = [x_2 - (p_1 x_1 + p_2 x_2)] f'(z_2) = p_1 (x_2 - x_1) f'(z_2) \tag{12.4}$$

因为 $z_1 < z_2$ 且 $f(x)$ 为凸函数, 所以 $f''(x) \geq 0$, 所以: $f'(z_1) \leq f'(z_2)$。

将 $(12.2) \times p_1 - (12.4) \times p_2$ 可得:

$$p_1 p_2 (x_2 - x_1) f'(z_1) - p_1 p_2 (x_2 - x_1) f'(z_2) \leq 0$$

所以:

$$p_1 [f(p_1 x_1 + p_2 x_2) - f(x_1)] \leq p_2 [f(x_2) - f(p_1 x_1 + p_2 x_2)]$$

$$f(p_1 x_1 + p_2 x_2) \leq p_1 f(x_1) + p_2 f(x_2)$$

接着用数学归纳法证明最开始的 Jensen 不等式。不妨将原式子转化为:

$$f(\sum_{i=1}^{k} p_i x_i) \leq \sum_{i=1}^{k} p_i f(x_i)$$

当 $k+1$ 时:

$$\begin{aligned}
\sum_{i=1}^{k+1} p_i f(x_i) &= p_{k+1} f(x_{k+1}) + z_k \sum_{i=1}^{k} \frac{p_i}{z_k} f(x_i) \qquad z_k = \sum_{i=1}^{k} p_i \\
&\geq p_{k+1} f(x_{k+1}) + z_k f(\sum_{i=1}^{k} \frac{p_i}{z_k} p_{x_i}) \\
&\geq f(p_{k+1} x_{k+1} + z_k \sum_{i=1}^{k} \frac{p_i}{z_k} p_{x_i}) \qquad z_k + p_{k+1} = 1 \\
&= f(\sum_{i=1}^{k+1} p_i x_i)
\end{aligned}$$

12.1.2 Hoeffding 不等式

若 $x_1, x_2, \cdots, x_m$ 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:

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

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

12.1.3 McDiarmid

不等式若 $x_1, x_2, \cdots, x_m$ 为 m 个独立随机变量，且对任意 $1 \leq i \leq m$，函数 f 满足:

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

则对任意 $\epsilon > 0$，有:

$$P(f(x_1, \cdots, x_m) - \mathbb{E}(f(x_1, \cdots, x_m)) \leq \epsilon) \leq \exp(\frac{-2\epsilon^2}{\sum_i c_i^2})$$

$$P(|f(x_1, \cdots, x_m) - \mathbb{E}(f(x_1, \cdots, x_m))| \leq \epsilon) \leq 2\exp(\frac{-2\epsilon^2}{\sum_i c_i^2})$$

12.2 VC 维

VC 维 (Vapnik-Chervonenkis dimension) 中的一些概念。

增长函数 (growth function)$\Pi_{\mathcal{H}}(m)$ 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。如果第 i 个示例有 c_i 中标记，那么答案就是 $\prod c_i$。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。

对分 (dichotomy)：就是对示例集赋予标记的每一种可能结果。

打散 (shattering)：假设空间可以实现示例集上的所有对分，那么称示例集能被假设空间打散。

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。VC 维理解为给定一些已知位置标签未知的点集的集合，取其中满足这样条件的点集的最大基数作为 VC 维，条件是：对于这个固定已知位置的点集的任何一种标签赋予方式总是线性可分的。也因此二维实平面上所有线性划分构成的假设空间的 VC 维是 3。

VC 维的泛化误差界是分布无关 (distribution-free)、数据独立 (data-independent) 的。

证明：$\mathbb{R}^d$ 空间中线性超平面构成的假设空间的 VC 维是 $d+1$.
先证 VC 维 $\geq d+1$，即是证 VC 维 $= d+1$ 是可以成立的。

设数据集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_n, y_n)\}$，因为是 $\mathbb{R}^d$ 空间中，所以 $\boldsymbol{x}_i$ 中有 d 个属性，即 $\boldsymbol{x}_i \in \mathbb{R}^{1\times d}$，令 $\boldsymbol{X} = [\boldsymbol{x}_1; \boldsymbol{x}_2; \cdots, \boldsymbol{x}_n], \boldsymbol{Y} = [y_1, y_2, \cdots, y_n]$，存在线性划分即是存在 $\boldsymbol{w}, \boldsymbol{B}$ 使得如下成立：

$$\boldsymbol{X}\boldsymbol{w} + \boldsymbol{B} = \boldsymbol{Y}$$

其中 $\boldsymbol{B} = [b_1; b_2; \cdots; b_n]$。如果令 $\boldsymbol{X} = [[\boldsymbol{x}_1, 1]; [\boldsymbol{x}_2, 1]; \cdots; [\boldsymbol{x}_n, 1];]$；上式可以改写成：

$$\boldsymbol{X}\boldsymbol{W} = \boldsymbol{Y}$$

VC 维 $= d+1$ 成立，即是 $n = d+1$ 时上式可以成立，注意到此时 $\boldsymbol{X} \in \mathbb{R}^{(d+1)\times(d+1)}$，所以只要令 $rank(\boldsymbol{X}) = d+1$，此时 $\boldsymbol{W}$ 一定有解：

$$\boldsymbol{W} = \boldsymbol{X}^{-1}\boldsymbol{Y}$$

所以 VC 维 $= d+1$ 是成立的，也就是 VC 维 $\geq d+1$ 是成立的。

下面证明 VC 维 $\leq d+1$，也就是 VC 维 $= d+2$ 是不成立的。当 VC 维 $= d+2$ 时，即 $n = d+2$，此时 $\boldsymbol{X} \in \mathbb{R}^{(d+2)\times(d+1)}$，所以 $rank(\boldsymbol{X}) \leq d+1$，所以一定存在矩阵 $\boldsymbol{X}$ 中的一行是其他行的线性组合，即：

$$\boldsymbol{x}_j = \sum_{i, i\neq j} a_i \boldsymbol{x}_i$$

此时根据 $\boldsymbol{X}\boldsymbol{W}$ 计算 $\boldsymbol{x}_j$ 的解已经由其他的 $\boldsymbol{x}_i$ 决定了：

$$\boldsymbol{x}_j W = \sum_{i, i\neq j} a_i \boldsymbol{x}_i \boldsymbol{W} = \sum_{i, i\neq j} a_i y_i$$

那么只要令 $y_j \neq \text{sign}(\sum_{i, i\neq j} a_i y_i)$，$\boldsymbol{X}\boldsymbol{W} = \boldsymbol{Y}$ 就一定无解。也就是对任意的 $\boldsymbol{X}$，总能找到相应的标签 $\boldsymbol{Y}$ 使其无解。所以 VC 维不能等于 $d+2$。

综上：$\mathbb{R}^d$ 空间中线性超平面构成的假设空间的 VC 维是 $d+1$。

12.3 Rademacher 复杂度

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。

Rademacher 随机变量 σ_i：它以 0.5 的概率取值为-1，以 0.5 的概率取值为 +1.

函数空间 $\mathcal{F}$ 关于 Z 的**经验 Rademacher 复杂度**为：

$$\hat{\mathrm{R}}_Z(\mathcal{F}) = \mathbb{E}_{\boldsymbol{\delta}}\left[\sup_{f\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^{m} \delta_i f(\boldsymbol{z}_i)\right]$$

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。

函数空间 $\mathcal{F}$ 关于 $\mathcal{Z}$ 上**分布 $\mathcal{D}$ 的 Rademacher 复杂度**：

$$\mathrm{R}_m(\mathcal{F}) = \mathbb{E}_{Z\subseteq\mathcal{Z}:|Z|=m}\left[\hat{\mathrm{R}}_Z(\mathcal{F})\right]$$

基于 Rademacher 复杂度可得关于函数空间 $\mathcal{F}$ 的泛化误差界。

计算学习理论领域最好的学术会议是：国际计算学习理论会议 (COLT).

Chapter 13

第十三章　半监督学习

主动学习：在有标记数据比较少的情况下，主动向专家询问一些样本的标记加入训练集中，再对算法进行训练。

半监督学习 (semi-supervised learning)：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。半监督学习可分为**纯半监督学习** (pure semi-supervised learning) 和**直推学习** (transductive learning)。纯半监督学习假定训练数据中的未标记样本并非待预测的数据，也就是希望模型能够适用训练过程中未观察到的数据；直推学习假定未标记样本就是待预测数据，学习的目的就是在这些未标记样本上获得最优的泛化性能。

要利用未标记样本，就要对未标记样本做一些关于数据分布信息和类别标记联系的假设。假设分为两种：**聚类假设** (cluster assumption) 和**流形假设** (manifold assumption)。聚类假设考虑的是类别标记，通常用于分类任务，它假设数据存在簇结构，同一个簇的样本属于同一个类别。流形假设对输出值没有限制，它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。无论是聚类假设还是流形假设都是基于“相似的样本拥有相似的输出”这个基本的假设。

半监督学习方法主要有四种：**生成式方法、半监督 SVM、图半监督、基于分歧**的方法。其中前三种针对单学习器，最后一种是多学习器。

13.1　生成式方法

生成式方法 (generative method) 是直接基于生成式模型的方法，即无论是标记数据还是未标记数据都是由同一个模型“生成”的。把未标记数据看作是潜在模型的缺失参数，通过**基于 EM 算法进行极大似然估计**求解。

给定有标记样本集 $D_l = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_l, y_l)\}$ 和未标记样本集 $D_u = \{(\boldsymbol{x}_{l+1}, y_{l+1}), (\boldsymbol{x}_{l+2}, y_{l+2}), \cdots, (\boldsymbol{x}_{l+u}, y_{l+u})\}, l \ll u, l+u=m$。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，用极大似然估计来估计高斯混合模型的参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) | 1 \le i \le N\}$，其中 N 是样本类别标记个数，$y_i \in \boldsymbol{\mathcal{Y}}, |\boldsymbol{\mathcal{Y}}| = N$。每个类别对应一个高斯混合模型，因此数据样本的概率密度是：

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

$\Theta \in \{1, 2, \cdots, N\}$ 表示样本 $\boldsymbol{x}$ 隶属的高斯混合成分，$D_l \cup D_u$ 的对数似然是：

$$LL(D_l \cup D_u) = \sum_{(\boldsymbol{x}_j, y_j) \in D_l} \ln \left(\sum_{i=1}^{N} \alpha_i \cdot p(\boldsymbol{x}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \cdot p(y_j | \Theta = i, \boldsymbol{x}_j) \right) + \sum_{\boldsymbol{x}_j \in D_u} \ln \left(\sum_{i=1}^{N} \alpha_i \cdot p(\boldsymbol{x}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \right)$$

由有标记数据 D_l 的监督项和未标记数据的 D_u 无监督项组成。

用 EM 算法迭代更新求解时和第九章比较：第九章是无监督学习，完全没有标注信息；而此处是半监督学习，有部分样本已标注。

高斯混合模型还可以替换成混合专家模型和朴素贝叶斯模型。

13.2 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。其中最著名的是针对二分类问题的 TSVM(Transductive Support Vector Machine)。

给定有标记样本集 $D_l = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_l, y_l)\}$ 和未标记样本集 $D_u = \{(\boldsymbol{x}_{l+1}, y_{l+1}), (\boldsymbol{x}_{l+2}, y_{l+2}), \cdots, (\boldsymbol{x}_{l+u}, y_{l+u})\}, l \ll u, l+u=m, y_i \in \{-1, +1\}$。TSVM 的学习目标是为 D_u 中的样本给出预测标记 $\hat{\boldsymbol{y}} = (\hat{y}_{l+1}, \hat{y}_{l+2}, \cdots, \hat{y}_{l+u}), \hat{y}_i \in \{-1, +1\}$，使得：

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

其中 $(\boldsymbol{w}, b)$ 确定了划分超平面；$\boldsymbol{\xi}$ 为松弛向量，$\xi_i (i = 1, 2, \cdots, l)$ 对应于有标记样本，$\xi_i (i = l+1, l+2, \cdots, m)$ 对应于未标记样本；C_l, C_u 是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。相当于 $D_l, D_u, \hat{\boldsymbol{y}}, C_l, C_u$ 已知，求 $\boldsymbol{w}, b, \boldsymbol{\xi}$。

为什么说 $\hat{\boldsymbol{y}}$ 是已知呢？因为 TSVM 采用局部搜索来迭代求解上式的近似解。先用有标记样本学得一个 SVM，然后用这个 SVM 来预测未标记样本得到“伪标记”，从而 $\hat{\boldsymbol{y}}$ 已知。此时剩下的就是一个标准的 SVM 问题了。

13.3 图半监督学习

图半监督学习就是将数据集的样本看做图中的节点，把样本的相似程度看成图中节点之间边的强度，把标记过的样本染上相应的颜色，半监督学习就是颜色在图上的扩散传播过程。

给定有标记样本集 $D_l = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_l, y_l)\}$ 和未标记样本集 $D_u = \{(\boldsymbol{x}_{l+1}, y_{l+1}), (\boldsymbol{x}_{l+2}, y_{l+2}), \cdots, (\boldsymbol{x}_{l+u}, y_{l+u})\}, l \ll u, l+u=m$。基于 $D_l \cup D_u$ 构建图 $G = (V, E)$，其中结点集 $V = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_l, \boldsymbol{x}_{l+1}, \cdots, \boldsymbol{x}_{l+u}\}$，边集通过高斯函数 (径向基函数) 定义：

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

令学习器学的函数 $f: V \to \mathbb{R}$，对应的分类规则为 $y_i = \text{sign}(f(\boldsymbol{x}_i)), y_i \in \{-1, +1\}$，通过最小化能量函数：

$$\begin{aligned} \min_{\boldsymbol{f}} \quad E(f) &= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} (\boldsymbol{W})_{ij} (f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j))^2 \\ &= \boldsymbol{f}^T (\boldsymbol{D} - \boldsymbol{W}) \boldsymbol{f} \end{aligned}$$

其中 $\boldsymbol{D} = \text{diag}(d_1, d_2, \cdots, d_{l+u})$ 是一个对角矩阵，$d_i = \sum_{j=1}^{l+u} (\boldsymbol{W})_{ij}$ 为矩阵 $\boldsymbol{W}$ 的第 i 行元素之和，$\boldsymbol{W}$ 是对称矩阵。

通过对 $E(f)$ 求 $\boldsymbol{f}_u$ 的偏导，并令其等于 0，可求得：

$$\boldsymbol{f}_u = (\boldsymbol{D}_{uu} - \boldsymbol{W}_{uu})^{-1} \boldsymbol{W}_{ul} \boldsymbol{f}_l$$

而 $\boldsymbol{f}_l$ 通过已标记样本已知，所以 $\boldsymbol{f}_u$ 可求。

上述是二分类问题的标记传播，对于多分类问题的标记传播，参考论文[Zhou et al., 2004]。简单来讲就是通过构造标记传播矩阵 $\boldsymbol{S} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{W} \boldsymbol{D}^{-\frac{1}{2}}$，然后迭代计算预测函数：

$$\boldsymbol{F}(t+1) = \alpha \boldsymbol{S} \boldsymbol{F}(t) + (1 - \alpha) \boldsymbol{Y}$$

13.4 基于分歧的方法

把每个属性集看做一个视图，不同的视图具相同性 (关于输出空间 $\mathcal{Y}$ 是一致的)，协同训练。

首先在每个视图上基于标记样本训练出分类器，然后每个分类器分别挑选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标记样本进行训练更新，以此迭代。

13.5 半监督聚类

聚类任务通过标记样本可以获得必连约束集合 $\mathcal{M}$ 和勿连约束集合 $\mathcal{C}$，前者是指样本属于同一簇中，后者指样本必不属于同一簇中。

半监督聚类和一般聚类的区别在于：在聚类的时候对于 $\boldsymbol{x}_i$，查看它能否在不违背 $\mathcal{M},\mathcal{C}$ 的约束条件下，将其放在最近的某个聚类簇中，若不能则错误；若能，放进相应聚类簇，并查看下一个样本。

还可以通过少量的标记样本，获得初始化的 k-means 算法的 k 个聚类中心，然后再聚类迭代。但是这样显然有一种风险：少量的标记样本未必包含所有的标记信息，那么没被包含的标记信息就可能被忽略。

Chapter 14

第十四章　概率图模型

- 图模型
  - 有向图模型
    - 静态贝叶斯网络
    - 动态贝叶斯网络
      - 隐马尔可夫模型 (Hidden Markov Model)
      - 卡尔曼滤波器 (Kalman Filter)
      - ···
  - 无向图模型 → 马尔可夫网络
    - 吉布斯/玻尔兹曼机 (Gibbs/Boltzmann Machine)
    - 条件随机场 (Conditional Random Field)
    - ···

概率模型的学习即基于训练样本来估计变量分布的参数。**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。**马尔可夫链** (Markov Chain)：系统下一时刻的状态由当前状态决定，不依赖于以往的任何状态。

14.1　马尔可夫随机场

马尔可夫随机场 (Markov Random Field) 是典型的马尔可夫网 (无向图模型)。图中的每个节点表示一个或一组变量，节点之间的边表示两个变量之间的关系。还有一组定义在变量子集 (一般是团 (clique) 或者极大团 (maximal clique)) 上的**势函数** (potential functions)，亦称为因子 (factor)，势函数是非负的，主要用于定义概率分布函数。

对于 n 个变量 $\boldsymbol{x} = \{x_1, x_2, \cdots, x_n\}$，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 $\boldsymbol{x}_Q$，则**联合概率** $P(\boldsymbol{x})$ 定义为：

$$P(\boldsymbol{x}) = \frac{1}{Z} \prod_{Q \in \mathcal{C}} \psi_Q(\boldsymbol{x}_Q)$$

其中 ψ_Q 为与团 Q 对应的势函数，用于对团中的变量关系进行建模，$Z = \sum_{\boldsymbol{x}} \prod_{Q \in \mathcal{C}} \psi_Q(\boldsymbol{x}_Q)$ 为规范化因子，以确保 $P(\boldsymbol{x})$ 是被正确定义的概率。

为了减少计算开销，可以把联合概率基于极大团定义，$\mathcal{C}^*$ 为所有极大团构成的集合：

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

如果从节点集 A 中的节点到 B 中的节点都必须经过节点集 C 中的节点，则称节点集 A 和 B 被节点集 C 分离，C 称为“分离集”(separating set)。

马尔可夫随机场满足的性质：

- 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。(在给定分离集的条件下独立：$\boldsymbol{x}_A \perp \boldsymbol{x}_B|\boldsymbol{x}_C$)，令 A,B,C 分别对应单变量 x_A, x_B, x_C，则有：

$$P(x_A, x_B|x_C) = P(x_A|x_C)P(x_B|x_C)$$

- 局部马尔可夫性 (local Markov property)：给定某变量的邻接变量，则该变量条件独立于其他变量。令 V 为图的节点集,$n(v)$ 为节点 v 在图上的邻接节点，$n^*(v) = n(v) \cup \{v\}$，有 $\boldsymbol{x}_v \perp \boldsymbol{x}_{V \setminus n^*(v)}|\boldsymbol{x}_{n(v)}$

- 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。令图的节点集和边集分别为 V 和 E，则对图中的两个节点 u 和 v，若 $\langle u, v\rangle \notin E$，$\boldsymbol{x}_u \perp \boldsymbol{x}_v|\boldsymbol{x}_{V \setminus \langle u,v\rangle}$

为了满足非负性，**势函数常被定义为指数函数**，即：

$$\psi_Q(\boldsymbol{x}_Q) = e^{-H_Q(\boldsymbol{x}_Q)}$$

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

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

其中 α_{uv} 和 β_v 是参数，第一个加法项考虑每一对节点关系，第二个加法项考虑单节点。

14.2 条件随机场

条件随机场 (Conditional Random Field，简称 CRF) 是一种**判别式无向图模型**。令 $G = \langle V, E\rangle$ 表示节点与标记变量 $\boldsymbol{y}$ 中元素一一对应的无向图，y_v 表示与节点 v 对应的标记变量，$n(v)$ 表示节点 v 的邻接节点，若图 G 的每个变量 y_v 都满足马尔可夫性 (每个节点都仅受邻接节点影响)，即：

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

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

CRF 的特点是假设输出随机变量构成马尔可夫随机场。条件随机场和马尔可夫随机场均使用团上的势函数来定义概率，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**。

在条件随机场中，通过选用指数函数并引入特征函数 (feature function)，条件概率被定义为：

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

其中 $t_j(y_{i+1}, y_i, \boldsymbol{x}, i)$ 是定义在观测序列的两个相邻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。$s_k(y_i, \boldsymbol{x}, i)$ 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，λ_j 和 μ_k 为参数，Z 为规范化因子，用于确保式子是正确定义的概率。

14.3 精确推断

推断是指在基于概率图模型定义的联合概率分布，对目标变量的**边际分布** (marginal distribution) 或某些可观测变量为条件的**条件分布**进行推断。边际分布是指无关变量求和或积分后得到的结果。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，给定参数 Θ 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，称为**边际化** (marginalization)。**推断问题的核心就是如何高效地计算边际分布**。

精确推断是希望获得目标变量的边际分布或者条件分布的精确值；而近似推断是近似解。精确推断实质是动态规划算法，利用图模型所描述的条件独立性来削减计算量。精确推断方法有变量消去法和信念传播法。

14.3.1 变量消去法

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。这样子每次求积和求和都只在局部进行，仅与部分变量有关。但是如果需要计算多个边际分布，重复使用变量消去法会造成大量的冗余计算。

14.3.2 信念传播

信念传播 (Belief Propagation) 算法将变量消去法中的求和操作看做一个消息传递的过程，一个结点仅在接受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到的消息的承继。

14.4 近似推断

14.4.1 MCMC 采样

通过随机化采样的方式，用样本均值近似期望，因为计算概率的目的，就是基于概率计算期望。概率图模型中最常用的采样技术是：**马尔可夫链蒙特卡罗方法** (Markov Chain Monte Carlo, 简称 MCMC)。

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：

$$P(A) = \int_A p(x)\mathrm{d}x$$

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

$$p(f) = \mathbb{E}_p[f(X)] = \int_x f(x)p(x)\mathrm{d}x$$

如果 x 不是单变量，而是一个高维多元变量 $\boldsymbol{x}$，且服从一个非常复杂的分布，那么上面的积分就很困难。MCMC 先构造出服从 p 分布的独立同分布随机变量 $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N$，然后得到上面积分式子的无偏估计：

$$\hat{p}(f) = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}_i)$$

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：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 $\boldsymbol{x}$ 近似服从于分布 p。判断到达平稳状态的方法：令从状态 $\boldsymbol{x}$ 转移到状态 $\boldsymbol{x}'$ 的概率为 $T(\boldsymbol{x}'|\boldsymbol{x})$，$t$ 时刻状态的分布为 $p(\boldsymbol{x}^t)$，则如果某个时刻马尔可夫链满足平稳条件：

$$p(\boldsymbol{x}^t)T(\boldsymbol{x}^{t-1}|\boldsymbol{x}^t) = p(\boldsymbol{x}^{t-1})T(\boldsymbol{x}^t|\boldsymbol{x}^{t-1})$$

则称 $p(\boldsymbol{x})$ 是该马尔可夫链的平稳分布，且此时以收敛到平稳状态。

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。**

构造马尔可夫链的状态转移概率方法：**Metropolis-Hastings 算法** (简称 MH)。MH 算法会以一定的概率拒绝根据上一轮采样结果 $\boldsymbol{x}^{t-1}$ 采样获得的候选状态样本 $\boldsymbol{x}^*$，而直接选择上一轮的采样结果 $\boldsymbol{x}^{t-1}$ 作为当前这一轮的采样结果。

Algorithm 1 *

Metropolist-Hasting 算法

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

1: 初始化 $\boldsymbol{x}^0$

2: **for** $t = 1, 2, \cdots$ **do**

3: 根据 $Q(\boldsymbol{x}^*|\boldsymbol{x}^{t-1})$ 采样出候选样本 $\boldsymbol{x}^*$;
4: 根据均匀分布从 (0,1) 范围内采样出阈值 u
5: **if** $u \leq A(\boldsymbol{x}^*|\boldsymbol{x}^{t-1})$ [$A(\boldsymbol{x}^*|\boldsymbol{x}^{t-1})$ 是 $\boldsymbol{x}^*$ 被接受的概率] **then**
6: $\boldsymbol{x}^t = \boldsymbol{x}^*$
7: **else**
8: $\boldsymbol{x}^t = \boldsymbol{x}^{t-1}$
9: **end if**
10: **end for**
输出: 采样出的一个样本序列: $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots$

14.4.2 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**。

使用变分法时，最重要的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，然后用 EM 算法进行概率图模型的推断和参数估计。

14.5 话题模型

话题模型 (topic model) 是一种生成式有向图模型，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。典型代表是**隐狄利克雷分配模型** (Latent Dirichlet Allocation，简称 LDA)。

"词"(word) 是待处理数据的基本离散单元，"文档"(document) 是待处理的数据对象，它由一组词组成。用文档和词进行数据表示的方式成为"词袋"(bag-of-words)。"话题"(topic) 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。

LDA 认为每篇文档包含多个 (K) 个话题，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，具体由话题"生成"文档 t 的步骤:

1. 根据参数为 $\boldsymbol{\alpha}$ 的狄利克雷分布随机采样一个话题分布 Θ_t;
2. 按如下步骤生成文档中的 N 个词;
   - 根据 Θ_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 $\boldsymbol{\beta}_k$ 随机采样生成词。

Chapter 15

第十五章　规则学习

15.1　基本概念

从形式语言表达能力而言，规则可分为两类：**命题规则** (ropositional rule) 和**一阶规则** (first-order rule)。命题规则是由"原子命题"(propositional atom) 和逻辑连接词与、或、非和蕴含构成的简单陈述句。一阶规则 (first-order rule) 不像命题规则只处理简单的陈述命题，一阶逻辑还额外包含了断言和量化 (谓词和量词)，用来描述事物的属性和关系。一阶规则能表达复杂的关系，也被称为关系型规则 (relational rule)。

生成规则的方法也分为两种:

- 直接生成法 (Direct Method)：直接从训练集中归纳出规则
- 间接生产法 (Indirect Method)：从决策树转换而来

15.2　序列覆盖

序列覆盖 (Sequencial Covering) 即逐条归纳，属于直接生成法。在训练集上每学习到一条规则，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) 的策略。

15.2.1　规则产生的策略

- 自顶向下（top-down)：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，直到满足预定的条件为止。亦称"生成－测试"（generate-then-test）法，是规则逐渐"特化"（specialization）的过程。(类似决策树)
- 自底向上（bottom-top)：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，亦称数据驱动 (data-driven) 方法，是规则逐渐泛化 (generalazation) 的过程。

15.2.2　比较

- 第一种策略更容易产生泛化性能更好的规则，第二种策略更适合于训练样本较少的情况。
- 第一种策略对噪声的鲁棒性比后者强。命题规则学习中通常使用第一种策略.
- 第二种策略在一阶规则学习这类假设空间非常复杂的任务上使用较多。

15.3 剪枝优化

原因: 序列覆盖是一个贪心搜索的过程, 需要机制来**缓解过拟合**的风险. 决策树一样分为: 预剪枝和后剪枝.

常用算法:

1. PRISM 算法: 第一个提出序列覆盖的算法, 基于准确率和覆盖率进行判断;
2. CN2 算法: 第一个考虑过拟合的算法, 预剪枝, 基于 AQ 算法并结合 ID3 算法处理噪音数据, 用熵对结果进行判断, 熵越小则质量越高;
3. FOIL 算法: 基于 CN2 算法的改进, 基于 FOIL 增益 (FOIL gain) 进行质量判断, 适用于一阶规则学习, 其中 FOIL 增益为:

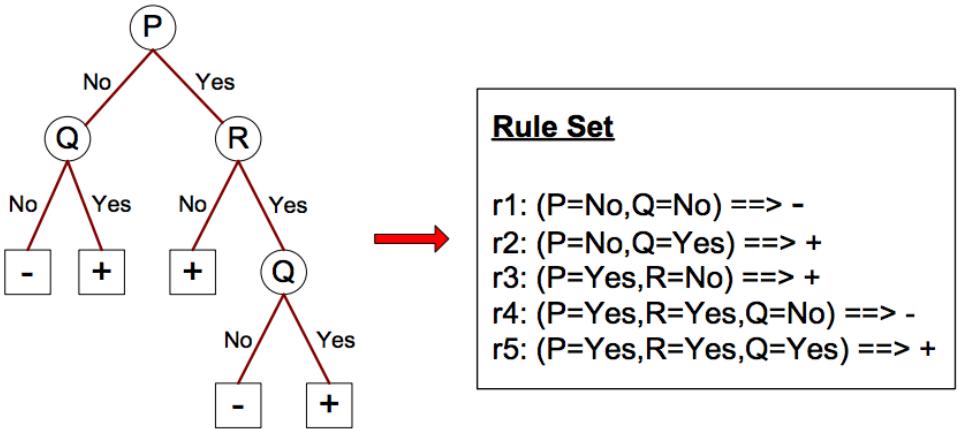
$$\text{F\_Gain} = \hat{m}_+ \times (\log_2 \frac{\hat{m}_+}{\hat{m}_+ + \hat{m}_-} - \log_2 \frac{m_+}{m_+ + m_-})$$

   其中 $\hat{m}_+, \hat{m}_-$ 分别是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。

4. REP 算法 (Reduce Error Pruning): $O(m^4)$, 后剪枝, 用准确率作为剪枝指标, 先生成 Rule 集合, 再剪枝, 具体是: 将样例集划分为训练集和验证集, 从训练集上学得规则集 R 后进行多轮剪枝, 在每一轮穷举所有可能的剪枝操作, 然后用验证集对剪枝产生的所有候选规则集进行比较, 保留最好的规则集进行下一轮剪枝 (也可以理解为如果错误增加了, 就剪掉, 否则保留)。时间复杂度是 $O(m^4)$, m 为训练样例数目。**REP 是针对规则集进行剪枝。**
5. IREP 算法 (Incremental REP): $O(m \log^2 m)$, 后剪枝, 相比 REP 高效, 每生成一个规则就进行剪枝验证: 在生成每条规则之前, 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。**IREP 仅对单条规则进行剪枝, 因此更高效。**
6. REIPPER 算法: 基于 IREP* 代替 IREP 的准确率, 防止局部最优, 泛化性能好, 学习速度比决策树更快。

15.4 间接方法

从决策树生成。


P
No
Yes
Q
R
No
Yes
No
Yes
-
+
+
Q
No
Yes
-
+
Rule Set
r1: (P=No,Q=No) ==> -
r2: (P=No,Q=Yes) ==> +
r3: (P=Yes,R=No) ==> +
r4: (P=Yes,R=Yes,Q=No) ==> -
r5: (P=Yes,R=Yes,Q=Yes) ==> +

Chapter 16

第十六章　强化学习

强化学习任务可以用 $E = \langle X, A, P, R\rangle$ 的马尔科夫决策过程 (Markov Decision Process, 简称 MDP) 表示，其中 E, X, A, P, R 分别表示当前环境、状态空间、动作空间、指定状态转移概率和奖赏函数。在强化学习中可以把状态、动作和策略分别对应监督学习中的示例、标记和分类器或回归器；但是强化学习中并没有监督学习中的有标记样本 (示例-标记对)，只有等到策略执行完，才能通过最终的学习结果才能评估动作执行是否正确。强化学习可以看做具有"延迟标记信息"的监督学习问题。

16.1　K-摇臂赌博机

16.1.1　$\epsilon-$ 贪心

每个臂被等概率探索。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 $v_1, v_2, \cdots, v_n$，则平均奖赏为:

$$\begin{aligned} Q(k) &= \frac{1}{n}\sum_{i=1}^{n} v_i \\ &= \frac{1}{n}((n-1) \times Q_{n-1}(k) + v_n) \\ &= Q_{n-1}(k) + \frac{1}{n}(v_n - Q_{n-1}(k)) \end{aligned}$$

16.1.2　Softmax 算法

平均奖赏越高的臂被选中的概率越大。Softmax 算法中摇臂概率的分配是基于 Boltzmann 分布的:

$$P(k) = \frac{e^{\frac{Q(k)}{\tau}}}{\sum_{i=1}^{K} e^{\frac{Q(i)}{\tau}}} \tag{16.1}$$

其中 $Q(i)$ 记录第 i 个臂的平均奖赏；$\tau > 0$ 称为"温度"; τ 越小则平均奖赏越高的摇臂被选取的概率越高。

16.1.3　UCB 方法

UCB 方法考虑了每个臂被探索的次数。UCB(Upper Confidence Bound，上置信界) 方法每次选择 $Q(k) + UC(k)$ 最大的摇臂，$UC(k)$ 定义为置信区间，例如:

$$Q(k) + \sqrt{\frac{2\ln n}{n_k}}$$

$\epsilon-$ 贪心算法

输入: 摇臂数 K; 奖赏函数 R; 尝试次数 T; 探索概率 ϵ。

$r = 0$
$\forall i = 1, 2, \cdots, K : Q(i) = 0, count(i) = 0$; $count(i)$ 记录臂 i 被选中的次数
for $i = 1, 2, \cdots, T$ **do**
  if rand() $< \epsilon$ **then**
    $k =$ 从 $1, 2, \cdots, K$ 中以均匀分布随机选取
  else
    $k = \arg\max_i Q(i)$
  end if
  $v = R(k)$;
  $r = r + v$;
  $Q(k) = \frac{Q(k) \times counk(k) + v}{count(k) + 1}$
  $count(k) = count(k) + 1$
end for

输出: 累积奖赏 r

Softmax 算法

输入: 摇臂数 K; 奖赏函数 R; 尝试次数 T; 探索概率 ϵ。

$r = 0$
$\forall i = 1, 2, \cdots, K : Q(i) = 0, count(i) = 0$; $count(i)$ 记录臂 i 被选中的次数
for $i = 1, 2, \cdots, T$ **do**
  $k =$ 从 $1, 2, \cdots, K$ 中以式子 (16.1) 选取
  $v = R(k)$;
  $r = r + v$;
  $Q(k) = \frac{Q(k) \times counk(k) + v}{count(k) + 1}$
  $count(k) = count(k) + 1$
end for

输出: 累积奖赏 r

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。

16.2 有模型学习

有模型学习 (model-based learning) 即马尔科夫决策过程四元组 $E = \langle X, A, P, R\rangle$ 均为已知。

16.2.1 策略评估

用 $V^\pi(x)$ 表示从状态 x 出发，使用策略 π 所带来的累积奖赏，可分为 T 步累积奖赏和 $\gamma \in (0,1)$ 折扣累计奖赏，可得状态值函数：

$$\begin{cases} V_T^\pi(x) = \mathbb{E}_\pi\left[\frac{1}{T}\sum_{t=1}^T r_t | x_0 = x\right] = \sum_{a\in A}\pi(x,a)\sum_{x'\in X} P_{x\to x'}^a(\frac{1}{T}R_{x\to x'}^a + \frac{T-1}{T}V_{T-1}^T(x')) \\ V_\gamma^\pi(x) = \mathbb{E}_\pi\left[\sum_{t=0}^{+\infty}\gamma^t r_{t+1} | x_0 = x\right] = \sum_{a\in A}\pi(x,a)\sum_{x'\in X} P_{x\to x'}^a(R_{x\to x'}^a + \gamma V_\gamma^T(x')) \end{cases}$$

函数 $Q^\pi(x,a)$ 表示从状态 x 出发，执行动作 a 后在使用策略 π 带来的累积奖赏，用 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，可得状态-动作值函数：

$$\begin{cases} Q_T^\pi(x,a) = \mathbb{E}_\pi\left[\frac{1}{T}\sum_{t=1}^T r_t | x_0 = x, a_0 = a\right] = \sum_{x'\in X} P_{x\to x'}^a(\frac{1}{T}R_{x\to x'}^a + \frac{T-1}{T}V_{T-1}^T(x')) \\ V_\gamma^\pi(x,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{+\infty}\gamma^t r_{t+1} | x_0 = x, a_0 = a\right] = \sum_{x'\in X} P_{x\to x'}^a(R_{x\to x'}^a + \gamma V_\gamma^T(x')) \end{cases}$$

16.2.2 策略改进

理想的策略应能最大化累积奖赏：

$$\pi^* = \arg\max_\pi \sum_{x\in X} V^\pi(x)$$

无论是状态值函数还是状态-动作值函数对于策略的每一点改进都是单调递增的。

16.2.3 策略迭代与值迭代

策略迭代 (policy iteration) 是按照策略评估-策略迭代-策略评估-策略迭代。。。以此迭代。
值迭代 (value iteration) 改进就是先更新值函数，最后在形成策略。

16.3 免模型学习

免模型学习 (model-free learning) 指算法不依赖与环境建模，有两个困难：**策略无法评估和奖赏函数未知**。也称为无模型学习。

16.3.1 蒙特卡罗强化学习

多次采样，求取平均累积奖赏来作为期望累积奖赏的近似。因为采样次数必须有限，所以这个方法更适合于 T 步累积奖赏的强化学习任务。采样时对策略的选取采用 $\epsilon-$ 贪心法，将确定性的策略 π 称为原始策略：

$$\pi^\epsilon(x) = \begin{cases} \pi(x), & \text{以概率} 1-\epsilon \\ A\text{中以均匀概率选取的动作}, & \text{以概率}\epsilon \end{cases}$$

因为这里被评估和被改进的是同一个策略，因此被称为“同策略”(one-policy) 蒙特卡罗强化学习算法。
异策略 (off-policy) 蒙特卡罗强化学习算法是策略评估时引用 ϵ-贪心，策略改进时改进原始策略。

基于 T 步累积奖赏的策略迭代算法

输入: MDP 四元组 $E = \langle X, A, P, R\rangle$; 累积奖赏参数 T

$\forall x \in X : V(x) = 0, \pi(x, a) = \frac{1}{|A(x)|}$
loop
  for $t = 1, 2, \cdots$ **do**
    $\forall x \in X : V'(x) = \sum_{a\in A} \pi(x, a) \sum_{x'\in X} P^a_{x\to x'}(\frac{1}{t} R^a_{x\to x'} + \frac{t-1}{t} V^T_{T-1}(x'))$
    if $t = T + 1$ **then**
      break
    else
      $V = V'$
    end if
  end for
  $\forall x \in X : \pi'(x) = \arg\max_{a\in A} Q(x, a)$;
  if $\forall x : \pi'(x) = \pi(x)$ **then**
    break
  else
    $\pi = \pi'$
  end if
end loop

输出: 最优策略 π

基于 T 步累积奖赏的值迭代算法

输入: MDP 四元组 $E = \langle X, A, P, R\rangle$; 累积奖赏参数 T; 收敛阈值 θ

$\forall x \in X : V(x) = 0$
for $t = 1, 2, \cdots$ **do**
  $\forall x \in X : V'(x) = \max_{a\in A} \sum_{x'\in X} P^a_{x\to x'}(\frac{1}{t} R^a_{x\to x'} + \frac{t-1}{t} V^T_{T-1}(x'))$
  if $\max_{x\in X} |V(x) - V'(x)| < \theta$ **then**
    break
  else
    $V = V'$
  end if
end for

输出: 最优策略 $\pi(x) = \arg\max_{a\in A} Q(x, a)$

同策略蒙特卡罗强化学习算法

输入: 环境 E; 动作空间 A; 起始状态 x_0; 策略执行步数 T

$Q(x,a)=0, count(x,a)=0, \pi(x,a)=\frac{1}{|A(x)|}$

for $s=1,2,\cdots$ **do**

在 E 中执行策略 π 产生轨迹: $<x_0,a_0,r_1,x_1,a_1,r_2,\cdots,x_{T-1},a_{T-1},r_T,x_T>$

for $t=0,1,\cdots,T-1$ **do**

$R=\frac{1}{T-t}\sum_{i=t+1}^{T} r_i$;

$Q(x_t,a_t)=\frac{Q(x_t,a_t)\times count(x_t,a_t)+R}{count(x_t,a_t)+1}$

$count(x_t,a_t)=count(x_t,a_t)+1$

end for

对所有已见状态 x:

$$\pi(x)=\begin{cases}\arg\max_{a'} Q(x,a'), & \text{以概率} 1-\epsilon \\ \text{以均匀概率从 A 中选取动作}, & \text{以概率}\epsilon\end{cases}$$

end for

输出: 策略 π

Algorithm 2 *

异策略蒙特卡罗强化学习算法

输入: 环境 E; 动作空间 A; 起始状态 x_0; 策略执行步数 T

$Q(x,a)=0, count(x,a)=0, \pi(x,a)=\frac{1}{|A(x)|}$

for $s=1,2,\cdots$ **do**

在 E 中执行 π 的 ϵ-贪心策略 π 产生轨迹: $<x_0,a_0,r_1,x_1,a_1,r_2,\cdots,x_{T-1},a_{T-1},r_T,x_T>$

$$p_i=\begin{cases}1-\epsilon+\epsilon/|A|, & a_i=\pi(x_i); \\ \epsilon/|A|, & a_i\neq\pi(x_i)\end{cases}$$

for $t=0,1,\cdots,T-1$ **do**

$R=\frac{1}{T-t}(\sum_{i=t+1}^{T} r_i)\prod_{i=t+1}^{T-1}\max(1,\frac{\mathbb{I}(a_i=\pi(x_i))}{p_i})$;

$Q(x_t,a_t)=\frac{Q(x_t,a_t)\times count(x_t,a_t)+R}{count(x_t,a_t)+1}$

$count(x_t,a_t)=count(x_t,a_t)+1$

end for

对所有已见状态 $\pi(x)=\arg\max_{a'} Q(x,a')$

end for

输出: 策略 π

16.3.2 时序差分学习

蒙特卡罗强化学习算法通过完成一个采样估计后再更新策略的值估计，解决了模型未知问题，但是效率很低。时序差分学习 (Temporal Difference Leanring) 则结合了动态规划和蒙特卡罗方法的思想，充分利用了强化学习任务重的 MDP 结构：每执行一步策略就更新一次值函数估计。

分别可得到同策略的 Sarasa 算法和异策略的 Q-学习算法 (Q-learning)。

16.4 值函数近似

前面的强化学习算法都是在**有限状态空间**上进行的，如果状态空间无限，那么就需要将连续状态空间的每一个状态表示成一个状态向量 $\boldsymbol{x}$，然后结合参数向量 $\boldsymbol{\theta}$ 来表示状态的线性函数 (也是值函数):

$$V_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x}$$

线性值函数近似 Sarsa 算法

输入: 环境 E；动作空间 A；起始状态 $\boldsymbol{x_0}$；奖赏折扣 γ；更新步长 α

$\boldsymbol{\theta} = 0, \boldsymbol{x} = \boldsymbol{x_0}, a = \pi(\boldsymbol{x}) = \arg\max_{a''} \boldsymbol{\theta}^T(\boldsymbol{x}; a'')$
for $s = 1, 2, \cdots$ **do**
  $r, \boldsymbol{x}' =$ 在 E 中执行动作 a 产生的奖赏与转移的状态;
  $a' = \pi^{\epsilon}(\boldsymbol{x}')$
  $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha(r + \gamma\boldsymbol{\theta}^T(\boldsymbol{x}'; a') - \boldsymbol{\theta}^T(\boldsymbol{x}; a))(\boldsymbol{x}; a)$;
  $\pi(\boldsymbol{x}) = \arg\max_{a''} \boldsymbol{\theta}^T(\boldsymbol{x}; a'')$;
  $\boldsymbol{x} = \boldsymbol{x}', a = a'$
end for

输出: 策略 π

16.5 模仿学习

16.5.1 直接模仿学习

直接模仿人类专家的“状态-动作对”。

16.5.2 逆强化学习

逆强化学习 (inverse reinforcement learning) 就是**从人类专家提供的范例数据中反推出奖赏函数。**

假设奖赏函数可以表达为状态特征的线性函数即: $R(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$，那么策略 π 的累积奖励可以写成状态向量的加权和的期望与系数 $\boldsymbol{w}$ 的内积:

$$\rho^{\pi} = \mathbb{E}\left[\sum_{i=0}^{+\infty} \gamma^t R(\boldsymbol{x}_t)|\pi\right] = \mathbb{E}\left[\sum_{i=0}^{+\infty} \gamma^t \boldsymbol{w}^T\boldsymbol{x}_t|\pi\right] = \boldsymbol{w}^T\mathbb{E}\left[\sum_{i=0}^{+\infty} \gamma^t \boldsymbol{x}_t|\pi\right]$$

把 $\mathbb{E}\left[\sum_{i=0}^{+\infty} \gamma^t \boldsymbol{x}_t|\pi\right]$ 简写为 $\bar{\boldsymbol{x}}^{\pi}$，把每条范例轨迹上的状态加权求和再平均记为 $\bar{\boldsymbol{x}}^*$，对于最优奖赏函数 $R(\boldsymbol{x}) = \boldsymbol{w}^{*T}\boldsymbol{x}$ 和其他任意策略产生的 $\bar{x}^{\pi}$，有:

$$\boldsymbol{w}^{*T}\bar{\boldsymbol{x}}^* - \boldsymbol{w}^{*T}\bar{\boldsymbol{x}}^{\pi} = \boldsymbol{w}^{*T}(\bar{\boldsymbol{x}}^* - \bar{\boldsymbol{x}}^{\pi}) \geq 0$$

所以有最优策略:

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \min_{\pi} \boldsymbol{w}^T(\bar{\boldsymbol{x}}^* - \bar{\boldsymbol{x}}^{\pi}) \qquad \text{s.t.} \quad \|\boldsymbol{w}\| \leq 1$$

16.6 其他

RL 的缺点:

- 需要大量有效的样本

迭代式逆强化学习算法

输入: 环境 E; 动作空间 A; 状态空间 X; 范例轨迹数据集 $D = \{\tau_1, \tau_2, \cdots, \tau_m\}$

1: $\bar{\boldsymbol{x}}^* =$ 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** $t = 1, 2, \cdots$ **do**
4: $\quad \bar{\boldsymbol{x}}^\pi =$ 从 π 的采样轨迹算出状态加权和的均值向量;
5: $\quad$ 求解 $\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \min_{\pi} \boldsymbol{w}^T(\bar{\boldsymbol{x}}^* - \bar{\boldsymbol{x}}^\pi) \qquad \text{s.t.} \quad \|\boldsymbol{w}\| \leq 1$
6: $\quad \pi =$ 在环境 $\langle X, A, R(\boldsymbol{x}) = \boldsymbol{w}^{*T}\boldsymbol{x} \rangle$ 中的求解最优策略
7: **end for**

输出: 奖赏函数 $R(\boldsymbol{x}) = \boldsymbol{w}^{*T}\boldsymbol{x}$ 与策略 π

- 最终表现很多时候并不好; 表现好的时候可能是过拟合
- 需要奖励函数，但是通常难以获得
- 因不当探索和利用 (exploration and exploitation) 导致的局部最优 (local optima) 难以避免
- 结果不稳定，而且很难重现

RL 表现好的条件:

- 容易获得大量样本
- 问题可以化简成一个简单的形式
- 可以进行左右互搏 (self-play)
- 奖励函数可以被定义; 奖励信号丰富，反馈及时

RL 未来的方向和发展:

- 局部最优或许已经足够好
- 靠硬件加速
- 增加更多的学习信号
- 用基于模型的 RL 来提高样本利用率
- 把 RL 当做一种微调手段 (fine-tuning)
- 迁移学习和强化学习结合
- 自学习奖励函数
- 好的先验知识可以大大降低学习时间
- 复杂的问题可能更容易用 RL 解决