

机器学习笔记

黎雷蕾

2017 年 10 月 31 日

摘要

学医三年，自谓天下无不治之症；
行医三年，始信世间无可用之方。——孙思邈

纸上得来终觉浅，绝知此事要躬行。——陆游

绳锯木断，水滴石穿。——罗大经

目录

1 线性模型 (linear model)	5
1.1 基本形式	5
1.2 线性回归 (linear regression)	5
1.3 多元线性回归 (multivariate linear regression)	6
1.4 对数线性回归 (log-linear regression)	7
1.5 对数几率回归/逻辑回归 (logistic regression)	7
1.6 线性判别分析 (linear discriminant analysis, LDA)	9
1.7 多元 LDA	10
1.8 多分类学习	11
1.9 类别不平衡问题 (class imbalance)	11
1.10 小结	11
2 决策树 (decision tree)	13
2.1 基本流程	13
2.2 划分选择	13
2.2.1 信息增益 (information gain)	13
2.2.2 信息增益率 (information gain ratio)	14
2.3 基尼指数 (Gini index)	14
2.4 剪枝处理 (pruning)	15
2.4.1 预剪枝 (prepruning)	15
2.4.2 后剪枝 (postpruning)	15
2.5 连续之和缺失值	16

2.5.1	连续值	16
2.5.2	缺失值	16
2.6	多变量决策树 (multivariate decision tree)	17
2.7	随机森林 (Random Forest)	17
2.8	迭代决策树 (Gradient Boost Decision Tree, GBDT)	19
2.9	小结	19
3	神经网络 (neural networks)	20
3.1	神经元模型	20
3.2	感知机 (Perceptron) 和多层网络	20
3.3	误差逆传播算法 (BackPropagation, BP)	21
3.4	全局最小与局部极小	24
3.5	其他常见神经网络	25
3.5.1	径向基函数网络, RBF	25
3.5.2	ART 网络	25
3.5.3	SOM 网络	25
3.5.4	级联相关网络	25
3.5.5	Elman 网络	26
3.5.6	Boltzmann 机	26
3.6	深度学习	26
3.7	小结	26
4	支持向量机 (Support Vector Machine,SVM)	27
4.1	间隔与支持向量	27
4.2	对偶问题	28
4.3	核函数	30
4.4	软间隔与正则化	32
4.5	支持向量回归 (Support Vector Regression,SVR)	34
4.6	核方法	36
4.7	小结	38

5	贝叶斯分类器 (bayes classifier)	39
5.1	贝叶斯公式	39
5.2	贝叶斯决策论 (bayesian decision theory)	39
5.3	极大似然估计 (Maximum Likelihood Estimation,MLE)	40
5.4	朴素贝叶斯分类器 (Naïve Bayes Classifier)	40
5.5	半朴素贝叶斯分类器 (semi-Naïve Bayes Classifier)	41
5.6	贝叶斯网 (Bayesian network)	41
5.6.1	贝叶斯网 (Bayesian network)-学习	42
5.6.2	贝叶斯网 (Bayesian network)-推断	43
5.7	EM(Expectation-Maximization) 算法	43
5.8	小结	44
6	集成学习 (ensemble learning)	45
6.1	个体与集成	45
6.2	Boosting	46
6.3	Bagging 与随机森林	49
6.3.1	Bagging	49
6.3.2	随机森林 (Random Forest,RF)	50
6.4	结合策略	51
6.4.1	平均法 (averaging)	51
6.4.2	投票法 (voting)	52
6.4.3	学习法	52
6.5	多样性	53
6.5.1	误差-分歧分解	53
6.5.2	多样性的度量 (diversity measure)	54
6.5.3	多样性增强	55
6.6	小结	56
7	聚类 (clustering)	57
7.1	聚类任务	57
7.2	性能度量	57

7.3	距离计算	59
7.4	原型聚类	60
7.4.1	k 均值算法 (k-means)	61
7.4.2	学习向量量化	62
7.4.3	高斯混合聚类	63
7.5	密度聚类 (density-based clustering)	66
7.6	层次聚类 (hierarchical clustering)	68
7.7	小结	70
8	降维与度量学习	71
8.1	k 近邻学习	71
8.2	低维嵌入	71
8.3	主成分分析 (Principal Component Analysis,PCA)	73
8.4	核化线性降维	75
8.5	流形学习 (manifold learning)	76
8.5.1	等度量映射 (Isometric Mapping,Isomap)	76
8.5.2	局部线性嵌入 (Locally Linear Embedding,LLE)	77
8.6	度量学习	78
8.7	小结	79
9	计算学习理论 (Computational Learning Theory)	80
9.1	基础知识	80
9.2	PAC 学习	81
9.3	有限假设空间	83
9.3.1	可分情形	83
9.3.2	不可分情形	84
9.4	VC 维	85

Chapter 1

线性模型 (linear model)

1.1 基本形式

给定一个由 d 个属性描述的样本 $x = (x_1; x_2; \cdots; x_i)$, 其中 x_i 表示 x 在第 i 个属性上的取值, 那么线性模型可以表示为:

$$f(x) = w_1x_1 + w_2x_2 + \cdots + w_dx_d + b \quad (1.1)$$

用向量形式来表示:

$$f(x) = w^T x + b \quad (1.2)$$

其中 $w = (w_1; w_2; \cdots; w_d)$, 一旦确定 w, b 模型就可以得到确定。

1.2 线性回归 (linear regression)

线性回归试图学习:

$$f(x_i) = wx_i + b, \text{ 使得 } f(x_i) \simeq y_i \quad (1.3)$$

为了得到最好的 w^*, b^* , 我们可以采用均方误差 (欧氏距离) 最小化的

方法:

$$\begin{aligned} E_{(w,b)} = (w^*, b^*) &= \arg \min_{(w,b)} \sum_{i=1}^m (f(x_i) - y_i)^2 \\ &= \arg \min_{(w,b)} \sum_{i=1}^m (y_i - wx_i - b)^2 \end{aligned} \quad (1.4)$$

求解 1.4 的过程, 被称为线性回归模型的最小二乘“参数估计 (parameter estimation)”: 将其分别对 w, b 求偏导:

$$\begin{aligned} \frac{\partial E_{(w,b)}}{\partial w} &= 2 \left(w \sum_{i=1}^m x_i^2 - \sum_{i=1}^m (y_i - b)x_i \right) \\ \frac{\partial E_{(w,b)}}{\partial b} &= 2 \left(wb - \sum_{i=1}^m (y_i - wx_i) \right) \end{aligned} \quad (1.5)$$

令 1.5 中两式为 0, 即可求出 w, b 的最优解的闭式解:

$$\begin{aligned} w &= \frac{\sum_{i=1}^m y_i(x_i - \bar{x})}{\sum_{i=1}^m x_i^2 - \frac{1}{m}(\sum_{i=1}^m x_i)^2} \\ b &= \frac{1}{m} \sum_{i=1}^m (y_i - wx_i) \\ \bar{x} &= \frac{1}{m} \sum_{i=1}^m x_i, \quad \bar{x} \text{ 为 } x \text{ 的均值} \end{aligned} \quad (1.6)$$

1.3 多元线性回归 (multivariate linear regression)

上一节中的 x 仅由单个属性描述, 若其由 d 个属性进行了描述, 就可以拓展为多元线性回归。

将 w, b 写成向量形式 $\hat{w} = (w; b)$, 同时把数据集表示成一个 $m \times (d+1)$ 大小的矩阵 X (m 代表样本个数, d 代表样本对应的属性个数), X 中的元素 x_{ij} 代表第 i 个样本的第 j 个属性, 最后一列恒为 1:

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{md} & 1 \end{pmatrix} = \begin{pmatrix} x_1^T & 1 \\ x_2^T & 1 \\ \vdots & \vdots \\ x_m^T & 1 \end{pmatrix} \quad (1.7)$$

与 1.4 类似

$$\begin{aligned} E_{\hat{w}} = \hat{w}^* &= \arg \min_{\hat{w}} (y - X\hat{w})^T (y - X\hat{w}) \\ \frac{\partial E_{\hat{w}}}{\partial \hat{w}} &= 2X^T(X\hat{w} - y) \end{aligned} \quad (1.8)$$

若 $X^T X$ 为满秩矩阵或者正定矩阵，则：

$$\hat{w}^* = (X^T X)^{-1} X^T y \quad (1.9)$$

令 $\hat{x}_i = (x_i, 1)$ ，则最终学得的多元回归模型：

$$f(\hat{x}_i) = \hat{x}_i^T (X^T X)^{-1} X^T y \quad (1.10)$$

1.4 对数线性回归 (log-linear regression)

一般来说我们得到的线性回归模型可以简写为：

$$y = w^T x + b \quad (1.11)$$

我们把 y 取对数，那它的本质是试图让 $e^{w^T x + b}$ 逼近 y ，即：

$$\ln y = w^T x + b \quad (1.12)$$

一般地，考虑单调可微的函数 $g(\cdot)$ ，令：

$$y = g^{-1}(w^T x + b) \quad (1.13)$$

这个模型就被称为广义上的线性模型。

1.5 对数几率回归/逻辑回归 (logistic regression)

首先介绍 *Sigmoid* 函数：

$$y = \frac{1}{1 + e^{-z}} \quad (1.14)$$

它将 z 值转化为一个接近 0 或者 1 的 y 值，并且在 $z = 0$ 附近变化很陡，带入上节的对数几率函数：

$$y = \frac{1}{1 + e^{-(w^T x + b)}} \quad (1.15)$$

若将 y 视为样本 x 作为正例的可能性，那么 $1 - y$ 为其反例的可能性，两者比值取对数：

$$\ln \frac{y}{1 - y} = w^T x + b \quad (1.16)$$

这个比值称为对数几率 (log odds, 也叫 logit)。

若将 y 视为后验概率 $p(y = 1|x)$ ，则：

$$\begin{aligned} p(y = 1|x) &= \frac{e^{w^T x + b}}{1 + e^{w^T x + b}} \\ p(y = 0|x) &= \frac{1}{1 + e^{w^T x + b}} \end{aligned} \quad (1.17)$$

我们可以采用极大似然估计法 (maximum likelihood method) 来估计 w 和 b ，那么上述的对数似然可以写为：

$$\ell(w, b) = \sum_{i=1}^m \ln p(y_i | x_i; w, b) \quad (1.18)$$

令 $\beta = (w; b)$, $\hat{x} = (x; 1)$ ，那么 $w^T x + b = \beta^T \hat{x}$ 。

令 $p_1(\hat{x}; \beta) = p(y = 1|\hat{x}; \beta)$ ，那么 $p_0(\hat{x}; \beta) = p(y = 0|\hat{x}; \beta) = 1 - p_1(\hat{x}; \beta)$

那么 1.18 可重写为：

$$p(y_i | x_i; w, b) = y_i p_1(\hat{x}_i; \beta) + (1 - y_i) p_0(\hat{x}_i; \beta) \quad (1.19)$$

由上述几个公式，重写 1.18:

$$\ell(\beta) = \sum_{i=1}^m (-y_i \beta^T \hat{x}_i + \ln(1 + e^{\beta^T \hat{x}_i})) \quad (1.20)$$

1.20 是高阶可导的连续凸函数，根据凸优化理论，梯度下降法或者牛顿迭代法均可以求出最优解。

$$\beta^* = \arg \min_{\beta} \ell(\beta) \quad (1.21)$$

1.6 线性判别分析 (linear discriminant analysis, LDA)

LDA 思想特别朴素：给定训练样例集，设法将样例投影到一条直线上，使得同类样例的投影点尽可能的近，异类样例的投影点尽可能远。即不同分类的样例在直线上是聚集在一起的，像一个个部落一样。

设 μ_0, μ_1 分别是两个分类的样本中心点，那么他们在直线上的投影分别是 $w^T \mu_0, w^T \mu_1$ 。若将所有样本点都投影到直线上，这两类样本的协方差分别为 $w^T \sum_0 w, w^T \sum_1 w$ 。

- 欲使同类样例投影点尽可能接近，可以让协方差尽可能小： $w^T \sum_0 w + w^T \sum_1 w$
- 欲使异类样例的投影点尽可能远离，可以让类中心之间的距离尽可能大： $\|w^T \mu_0 - w^T \mu_1\|_2^2$

综合上面两点，最大化目标 J 可写为：

$$\begin{aligned} J &= \frac{\|w^T \mu_0 - w^T \mu_1\|_2^2}{w^T \sum_0 w + w^T \sum_1 w} \\ &= \frac{w^T (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T w}{w^T (\sum_0 + \sum_1) w} \end{aligned} \quad (1.22)$$

定义“类内散度矩阵”(within-class scatter matrix)：

$$\begin{aligned} S_w &= \sum_0 + \sum_1 \\ &= \sum_{x \in X_0} (x - \mu_0)(x - \mu_0)^T + \sum_{x \in X_1} (x - \mu_1)(x - \mu_1)^T \end{aligned} \quad (1.23)$$

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

$$S_b = (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T \quad (1.24)$$

重写 1.22:

$$J = \frac{w^T S_b w}{w^T S_w w} \quad (1.25)$$

由于 1.25 的解与 w 的长度无关, 只与其方向有关, 不是一般性, 令 $w^T S_w w = 1$, 最大化分子 $w^T S_b w$, 一般采用拉格朗日乘子法。可得:

$$w = S_w^{-1}(\mu_0 - \mu_1) \quad (1.26)$$

为了求 S_w^{-1} , 通常的做法是对 S_w 进行奇异值分解, $S_w = U \Sigma V^T$, 得到 $S_w^{-1} = V \Sigma^{-1} U^T$, 从而进行求解。

结合贝叶斯理论, 当两类数据满足先验概率相同、服从高斯分布且协方差相等, LDA 可以达到最优分类

1.7 多元 LDA

假定存在 N 个类, 且第 i 类示例数为 m_i , 我们可以定义 S_w, S_b 和全局散度矩阵 S_t

$$\begin{aligned} S_w &= \sum_{i=1}^N S_{w_i} = \sum_{x \in X_i} (x - \mu_i)(x - \mu_i)^T \\ S_b &= \sum_{i=1}^N m_i (\mu_i - \mu)(\mu_i - \mu)^T \\ S_t &= S_w + S_b = \sum_{i=1}^m (x_i - \mu)(x_i - \mu)^T \end{aligned} \quad (1.27)$$

其中 μ 是所有示例的均值向量。通常知道上式三者中的两者即可, 采用优化目标:

$$\max_W \frac{\text{tr}(W^T S_b W)}{\text{tr}(W^T S_w W)} \quad (1.28)$$

$$S_b W = \lambda S_w W \quad (1.29)$$

故 W 的闭式解是 $S_w^{-1} S_b$ 的 $N - 1$ 个最大广义特征值所对应的特征向量组成的矩阵。由于投影有降维的作用, 故 LDA 也被视为一种经典的监督降维技术。

1.8 多分类学习

经典的拆分策略有三个，涉及到编码不详细说明，详情见周志华《机器学习》p.63 ~ p.66:

- 一对一: One vs One, OvO
- 一对其余: One vs Rest, OvR
- 多对多: Many vs Many, MvM

1.9 类别不平衡问题 (class imbalance)

若训练集中正 (m^+) 反 (m^-) 例子数目不均等，那么：

$$\frac{y}{1-y} > \frac{m^+}{m^-}, \text{预测为正例} \quad (1.30)$$

故对样本进行再缩放 (rescaling):

$$\frac{y'}{1+y'} = \frac{y}{1+y} \times \frac{m^-}{m^+} \quad (1.31)$$

总之，处理类别不平衡主要有三种方法：

- 欠采样 (undersampling): 去除一定样本，使得正负样本数目趋于平衡。
- 过采样 (oversampling): 增加一些样本。
- 再缩放策略。

1.10 小结

本章是机器学习理论的基础基础章节，介绍的是最基本的线性模型。

- 线性模型：
 - 形式简单、易于建模，许多功能更加强大的非线性模型大都都是在线性模型的基础上通过引入层级结构或者高维映射而得。

- 线性模型中的 w 具有很好的解释性，便于理解。
- 逻辑回归，又名对数几率回归:
 - 可以直接对分类可能性进行建模，无需事先假设数据的分布，就可以避免假设分布不准确所带来的问题。
 - 它不仅可以进行分类，还可以得到近似的概率预测。
 - 对率函数是任意阶可导的凸函数，具有很好的数学性质许多数值化方法都可以用于求取最优解。
- 线性判别分析 (LDA): 是一种采用投影的策略，被视为一种经典的监督降维技术。

Chapter 2

决策树 (decision tree)

2.1 基本流程

决策树其实是一个递归建树的流程，具体步骤如下：

1. 对于属性集 A ，采用信息增益或者基尼指数等方式确定当做根节点的 a_i
2. 根据 a_i 的取值将样本分成几份 $D = \{D_1^{a_i}, D_2^{a_i}, \dots, D_n^{a_i}\}$
3. 对于每一个样本子集 $D_j^{a_i}$ ，重复上述的 (1), (2) 两步。直到样本子集里的每个样本属于同一类别 C 。

2.2 划分选择

2.2.1 信息增益 (information gain)

首先需要了解信息熵 (information entropy)，信息熵代表了信息不确定的程度：信息熵越大，说明信息越不确定，那么纯度越低；反之，若信息熵很小，说明信息确定程度高，那么信息纯度越高。假设当前样本集合 D 中第 k 类样本所占的比例为 p_k , ($k = 1, 2, \dots, |\mathcal{Y}|$)，那么信息熵可以被定义为：

$$Ent(D) = - \sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k \quad (2.1)$$

由于信息熵取值和纯度大小呈反比，为了便于理解，我们引入“信息增益”，假设一个属性 a 的取值为 $\{a^1, a^2, \dots, a^V\}$ ，那么样本集合 D 可以按照 a 分成 V 份，假设按照 a^v 分的样本子集是 D^v ，那么信息增益 $Gain(D, a)$ 可以写为：

$$Gain(D, a) = Ent(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} Ent(D^v) \quad (2.2)$$

为了得到最大的信息增益，我们可以求出所有属性 a^i 的信息增益 $Gain(D, a^i)$ ，从中选择信息增益最大的 a_* 作为当前的节点，这个思想就是**ID3 决策树学习算法**。

$$a_* = \arg \max_{a \in A} Gain(D, a) \quad (2.3)$$

2.2.2 信息增益率 (information gain ratio)

信息增益准则会对取值数目较多的属性有所偏好，为了减少这种不利影响，我们采用增益率 (gain ratio) 来划分最优属性，这个思想就是**C4.5 决策树算法**。

$$Gain_ratio(D, a) = \frac{Gain(D, a)}{IV(a)} \quad (2.4)$$

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

2.3 基尼指数 (Gini index)

CART 决策树采用基尼指数来选择划分属性，数据集 D 的纯度可以用基尼值 $Gini(D)$ 来测量：

$$Gini(D) = \sum_{k=1}^{|\mathcal{Y}|} \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^{|\mathcal{Y}|} p_k^2 \quad (2.5)$$

和信息熵类似，基尼值越小，数据集的纯度越高。基尼指数可以定义为：

$$Gini_index(D, a) = \sum_{v=1}^V \frac{|D^v|}{|D|} Gini(D^v) \quad (2.6)$$

与信息增益相反，我们选择基尼系数最小的属性当做划分属性：

$$a_* = \arg \min_{a \in A} \text{Gini_index}(D, a) \quad (2.7)$$

2.4 剪枝处理 (pruning)

剪枝是为了避免决策树算法是否进入‘过拟合’的手段。

2.4.1 预剪枝 (prepruning)

预剪枝是指在决策树生成过程中对当前节点进行估计，若当前节点的划分不能带来决策树泛化性能的提升，则停止划分并将当前节点标记为叶节点。

- 计算不分叶节点之前验证集的精度 p_{pre} 。
- 计算分开的叶节点之后的验证集精度 p_{post}
- 若 $p_{post} > p_{pre}$ 则扩展该节点，否则直接将其作为叶节点。
- 预剪枝可以减少很多不必要的分支，时间开销较小，但是会带来更大的欠拟合风险。

2.4.2 后剪枝 (postpruning)

- 和预剪枝类似，也是采用划分前后的验证集精度来决定是否进行剪枝。
- 不同的是，后剪枝是先分叶节点后再剪枝。
- 相比于预剪枝，后剪枝通常可以保留更加多的叶节点，所以后剪枝的欠拟合风险较小，泛化性能优于预剪枝，但是后剪枝需要在决策树生成后才能进行，训练的开销要大于预剪枝。

2.5 连续之和缺失值

2.5.1 连续值

对于连续值，一般采用连续属性离散化技术，最简单的策略是采用二分法 (bi-partition) 对连续属性进行处理，**C4.5 决策树算法就是采用这种方法。**

给定样本集 D 和连续属性 a ，假定 a 在 D 上出现了 n 个不同的取值，将其按从大到小的顺序进行排列，记为 $\{a_1, a_2, \dots, a_n\}$ ，基于划分点 t 可将 D 分为子集 D_t^- 和 D_t^+ ，其中 $a_i \leq a_t, a_i \in D_t^-$ 且 $a_j > a_t, a_j \in D_t^+$ ，对于相邻的属性 a_t, a_{t+1} ， t 在区间 $[a_t, a_{t+1})$ 上取任意值的划分相同。那么对于一个连续属性 a_t 我们就可以考察包含 $n - 1$ 个衰术的划分点集合，即：

$$T_n = \left\{ \frac{a_i + a_{i+1}}{2} \mid 1 \leq i \leq n - 1 \right\} \quad (2.8)$$

上述公式的意思就是把区间 $[a_t, a_{t+1})$ 上的中位点 $\frac{a_i + a_{i+1}}{2}$ 作为候选的划分点，从而连续值就变成了离散值。

2.5.2 缺失值

对于缺失值，我们通常是采用给特定的信息增益赋予权值 ρ 的方式。

1. 假设某一个属性 a_i 的集合为 A ，缺失的属性集合为 A^* ，那么在 D 上有， $|D_A| = |A| + |A^*|$ ，其中 $|A|$ 代表集合 A 中的属性个数。
2. 我们把 A 当做一个无缺失值的属性，计算出相应的信息增益 $Gain(A, a_i)$ 。
3. 实际上属性的信息增益：

$$Gain(D_A, a_i) = \rho \times Gain(A, a_i) = \frac{|A|}{|A| + |A^*|} \times Gain(A, a_i) \quad (2.9)$$

4. 若用该属性作为父节点，那么缺失值将同时进入所有的子节点，在每个节点计算信息增益时，它的权重：

$$\rho_{child} = \frac{\text{该子节点不缺失的属性个数}}{\text{父节点不缺失属性的个数}} \quad (2.10)$$

2.6 多变量决策树 (multivariate decision tree)

上述所有的决策树算法的节点都是以单个属性为准，而我们实际的情况下，经常会用到多变量作为决策树的分界点，每个非叶结点都是形如 $\sum_{i=1}^d w_i a_i = t$ 的线性分类器，这个分类器可能会采取 *softmax* 之类的方式进行决策，不太便于解释。

2.7 随机森林 (Random Forest)

虽然拥有剪枝技术，但是决策树还是会存在过拟合的问题，随机森林可以很好地解决这个问题：

随机森林顾名思义，是用随机的方式建立一个森林，森林里面有很多的决策树组成，随机森林的每一棵决策树之间是没有关联的。在得到森林之后，当有一个新的输入样本进入的时候，就让森林中的每一棵决策树分别进行一下判断，看看这个样本应该属于哪一类（对于分类算法），然后看看哪一类被选择最多，就预测这个样本为那一类。

随机森林是一个最近比较火的算法，它有很多的优点：

- 在数据集上表现良好
- 在当前的很多数据集上，相对其他算法有着很大的优势
- 它能够处理很高维度（feature 很多）的数据，并且不用做特征选择
- 在训练完后，它能够给出哪些 feature 比较重要
- 在创建随机森林的时候，对误差 (generalization error) 使用的是无偏估计
- 训练速度快
- 在训练过程中，能够检测到 feature 间的互相影响
- 容易做成并行化方法，实现比较简单

每棵树的按照如下规则生成：

1. 如果训练集大小为 N ，对于每棵树而言，随机且有放回地从训练集中的抽取 N 个训练样本（这种采样方式称为 bootstrap sample 方法），作为该树的训练集；如果不进行随机抽样，每棵树的训练集都一样，那么最终训练出的树分类结果也是完全一样的，这样的话完全没有 bagging 的必要；我理解的是这样的：如果不是有放回的抽样，那么每棵树的训练样本都是不同的，都是没有交集的，这样每棵树都是”有偏的”，都是绝对”片面的”（当然这样说可能不对），也就是说每棵树训练出来都是有很大的差异的；而随机森林最后分类取决于多棵树（弱分类器）的投票表决，这种表决应该是”求同”，因此使用完全不同的训练集来训练每棵树这样对最终分类结果是没有帮助的，这样无异于是”盲人摸象”。
2. 如果每个样本的特征维度为 M ，指定一个常数 $m \ll M$ ，随机地从 M 个特征中选取 m 个特征子集，每次树进行分裂时，从这 m 个特征中选择最优的；
3. 每棵树都尽最大程度的生长，并且没有剪枝过程。

两个随机性的引入对随机森林的分类性能至关重要。由于它们的引入，使得随机森林不容易陷入过拟合，并且具有很好得抗噪能力（比如：对缺省值不敏感）。

随机森林分类效果（错误率）与两个因素有关：

- 森林中任意两棵树的相关性：相关性越大，错误率越大；
- 森林中每棵树的分类能力：每棵树的分类能力越强，整个森林的错误率越低。
- 减小特征选择个数 m ，树的相关性和分类能力也会相应的降低；增大 m ，两者也会随之增大。所以关键问题是如何选择最优的 m （或者是范围），这也是随机森林唯一的一个参数。

2.8 迭代决策树 (Gradient Boost Decision Tree, GBDT)

首先要了解 Boost 算法：原始的 Boost 算法是在算法开始的时候，为每一个样本赋上一个权重值，初始的时候，大家都是一样重要的。在每一步训练中得到的模型，会使得数据点的估计有对有错，我们就在每一步结束后，增加分错的点的权重，减少分对的点的权重，这样使得某些点如果老是被分错，那么就会被“严重关注”，也就被赋上一个很高的权重。然后等进行了 N 次迭代（由用户指定），将会得到 N 个简单的分类器（basic learner），然后将它们组合起来（比如说可以对它们进行加权、或者让它们进行投票等），得到一个最终的模型。

而 Gradient Boost 与传统的 Boost 的区别是，每一次的计算是为了减少上一次的残差 (residual)，而为了消除残差，我们可以在残差减少的梯度 (Gradient) 方向上建立一个新的模型。所以说，在 Gradient Boost 中，每个新的模型的简历是为了使得之前模型的残差往梯度方向减少，与传统 Boost 对正确、错误的样本进行加权有着很大的区别。

建立 GBDT 流程：

1. 给定一个初始化的值， $F_0(x)$
2. 迭代 M 次，建立 M 颗决策树
3. 对 $F_i(x)$ 进行逻辑回归。
4. 求得残差减少的梯度方向
5. 类似随机森林方式投票确定当前决策树，与原来的决策树合在一起作为一个新的模型。

2.9 小结

这章主要是介绍了决策树算法，随机森林和 GBDT 会在书上第八章再详细进行说明。

Chapter 3

神经网络 (neural networks)

3.1 神经元模型

设输入的样本属性 X 和对应的连接权重 W , 阈值 θ , 那么神经元可以被表示为:

$$y = f(WX - \theta) = f\left(\sum_{i=1}^n w_i x_i - \theta\right) \quad (3.1)$$

其中 y 表示输出, $f(x)$ 表示激活函数 (activation function)。

3.2 感知机 (Perceptron) 和多层网络

最简单的有两层 (输入层和输出层)。如图所示:

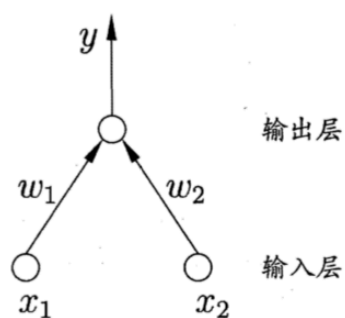


图 5.3 两个输入神经元的感知机网络结构示意图

一般的，给定训练的数据集，权重 $w_i (i = 1, 2, \dots, n)$ 和阈值 θ 可以通过学习得到。对于训练的样例 (x, y) ，若当前感知机的输出为 \hat{y} ，感知机的权重可以调整为：

$$\begin{aligned} w_i &\leftarrow w_i + \Delta w_i, \\ \Delta w_i &= \eta(y - \hat{y})x_i \end{aligned} \quad (3.2)$$

其中 $\eta \in (0, 1)$ 称为学习率 (learning rate)，若感知机预测正确， $\hat{y} = y$ ，否则根据错误程度进行权重调整。

单层感知机只能解决线性可分的问题，若不是线性可分的问题，那么我们得改成多层感知机才行，如输入层与输出层之间加入一层隐藏层。如下图就是典型的多层前馈神经网络 (multi-layer feedforward neural networks)：

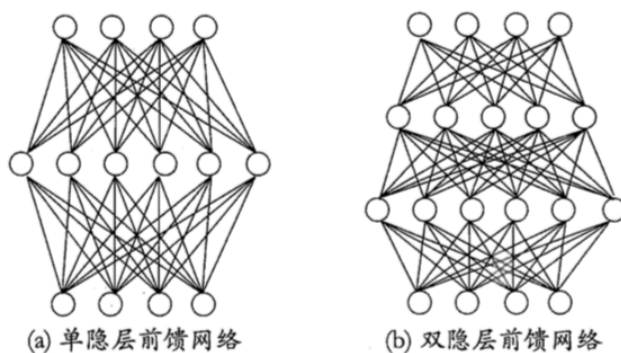


图 5.6 多层前馈神经网络结构示意图

3.3 误差逆传播算法 (BackPropagation, BP)

BP 算法可以说是 NN 算法中最杰出的代表，大部分时间都是用于多层前馈网络。

给定训练集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$, $x_i \in \mathbb{R}^d, y_i \in \mathbb{R}^l$ ，即输入示例由 d 个属性进行描述，输出 l 维向量，即 d 个输入神经元和 l 个输出神经元， q 个隐藏层神经元。其中输出层的第 j 个神经元的阈值为 θ_j ，隐层第 h 个神经元用 γ_h ，输入层第 i 个神经元和隐层的第 h 个神经元连接权

值为 v_{ih} ，隐层第 h 个神经元与输出层第 j 个神经元之间的连接权值为 w_{hj} 。记隐层第 h 个神经元接收到的输入为 $a_h = \sum_{i=1}^d v_{ih}x_i$ ，输出层第 j 个神经元接收到的输入为 $\beta_j = \sum_{h=1}^q w_{hj}b_h$ ，其中 b_h 为隐层第 h 个神经元的输出，如下图所示：

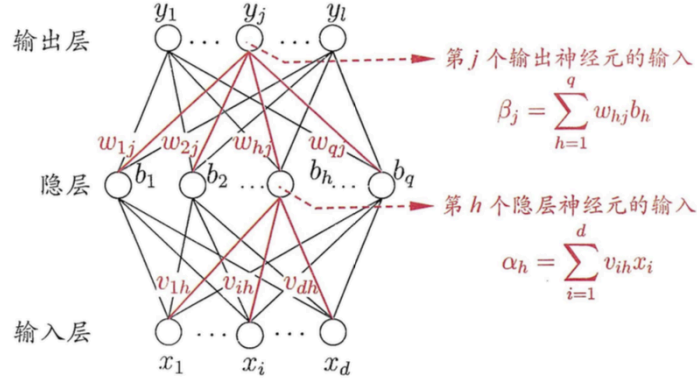


图 5.7 BP 网络及算法中的变量符号

对训练例子 (x_k, y_k) ，假定神经网络的输出为 $\hat{y}_k = (\hat{y}_1^k, \hat{y}_2^k, \dots, \hat{y}_l^k)$ ，可以表示为：

$$\hat{y}_j^k = f(\beta_j - \theta_j) \quad (3.3)$$

那么在 (x_k, y_k) 的均方误差为：

$$E_k = \frac{1}{2} \sum_{j=1}^l (\hat{y}_j^k - y_j^k)^2 \quad (3.4)$$

BP 算法实际上是一个迭代学习的过程，下面以隐藏层到输出层的连接权值 w_{hj} 为例进行推导。由于 BP 算法基于梯度下降 (gradient descent) 策略，以目标的负梯度方向对参数进行调整，对上面的均方误差 E_k 及给定的学习率 η ，有：

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}} \quad (3.5)$$

而从图 3.3 可以看出 w_{hj} ，首先影响第 j 个输出层神经元的输入值 β_j ，再影响其输出值 \hat{y}_j^k ，最后影响到 E_k ，根据复合微分和马尔科夫过程，有：

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}} \quad (3.6)$$

而根据 $\beta_j = \sum_{h=1}^q w_{hj} b_h$ ，可得：

$$\frac{\partial \beta_j}{\partial w_{hj}} = b_h \quad (3.7)$$

我们选择 *Sigmoid* 函数作为激活函数，是由于它有一个特别好的性质：

$$f'(x) = f(x)(1 - f(x)) \quad (3.8)$$

根据公式 3.3和 3.4可得：

$$g_j = -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} = -(\hat{y}_j^k - y_j^k) f'(\beta_j - \theta_j) = \hat{y}_j^k (1 - \hat{y}_j^k) (y_j^k - \hat{y}_j^k) \quad (3.9)$$

这个式子表示 g_j 可以采用实际输出 \hat{y}_j^k 和样本真实输出 y_j^k 进行表示，那么结合前面的式子 3.5-3.9：

$$\Delta w_{hj} = \eta g_j b_h \quad (3.10)$$

同理，我们可以求出其他的参数：

$$\Delta \theta_j = -\eta g_j$$

$$\Delta v_{ih} = \eta e_h x_i$$

$$\Delta \gamma_h = -\eta e_h$$

其中：

$$\begin{aligned} e_h &= -\frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} = -\sum_{j=1}^l \frac{\partial E_k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} f'(\alpha_h - \gamma_h) \\ &= b_h(1 - b_h) \sum_{j=1}^l w_{hj} g_j \end{aligned} \quad (3.11)$$

学习率 $\eta \in (0, 1)$ 控制算法每一轮迭代中的更新步长，若太大则容易造成震荡，太小会导致收敛速度太慢。

BP 算法的目标是要最小化训练集 D 上的积累误差：

$$E = \frac{1}{m} \sum_{k=1}^m E_k \quad (3.12)$$

一般来说，只要一个包含足够多神经元的隐层，多层前馈网络就能以任意精度逼近任意复杂度的连续函数，实践中通常采用试错法 (trial-by-error) 进行调整。

为了防止过拟合，通常采用两种策略：

- 早停 (early stopping): 将数据分成训练集和测试集，在更新权值时，若训练集误差降低而测试集误差升高，则停止训练。
- 正则化 (regularization): 在目标函数中增加一个用于描述网络复杂度的部分，如连接层权值与阈值的平方和，用 E_k 表示第 k 个训练样例上的误差， w_i 表示连接权值和阈值的平方和，那么 E 可以改写为：

$$E = \lambda \frac{1}{m} \sum_{k=1}^m E_k + (1 - \lambda) \sum_i w_i^2 \quad (3.13)$$

其中 $\lambda \in (0, 1)$ 表示经验误差与网络复杂度这两项的折中，通常采用交叉验证法进行估计。

3.4 全局最小与局部极小

一般来说，用的最多的参数寻优的方法是梯度搜索。梯度下降法是沿着函数值下降得最快的方向进行搜索，若误差函数在当前值为零了，说明已经到达局部极小，但是不一定到达全局最小。

为了跳出局部极小，通常采用如下策略：

- 以多组不同的参数值初始化神经网络，但是可能会陷入多个不同的局部极小。
- 模拟退火 (simulated annealing)，其策略是在每一步有一定的概率接收比当前解更差的结果，即选择次优解。
- 采用随机梯度算法，这样可以保证即使到局部极小时它的梯度仍不为零，从而跳出局部极小。
- 或者采用遗传算法 (genetic algorithms)。

3.5 其他常见神经网络

3.5.1 径向基函数网络, RBF

RBF 网络是一种单隐层神经网络, 采用径向基函数作为隐层神经元的激活函数:

$$\begin{aligned}\varphi(x) &= \sum_{i=1}^q w_i \rho(x, c_i) \\ \rho(x, c_i) &= e^{-\beta_i \|x - c_i\|^2}\end{aligned}\tag{3.14}$$

已经被证明: 拥有足够度的隐层神经元的 RBF 网络, 能以任意精度逼近任意连续函数。

3.5.2 ART 网络

ART 网络是竞争 (competitive learning) 型网络, 是神经网络中常用的无监督学习网络, 其原理是所有网络的输出神经元都进行相互的竞争, 每一时刻有且仅有一个神经元获胜而被激活, 剩下的神经元都处于抑制状态。

ART 可以缓解竞争学习中的‘可塑性-稳定性窘境 (stability-plasticity dilemma)’, 可塑性 (plasticity) 指的是神经网络要有学习新知识的能力, 稳定性 (stability) 指的是神经网络在学习心得知识时要保持旧的知识, 这样 ART 可以进行增量学习 (incremental learning) 或者在线学习 (online learning)。

3.5.3 SOM 网络

自组织映射网络 (Self-Organizing Map), 是一种竞争学习型的无监督映射网络, 它能够将高维数据映射到低维空间, 同时保持输入数据在高维空间的拓扑结构。

3.5.4 级联相关网络

级联相关 (Cascade-Correlation) 网络能够在训练过程中改变自身的网络拓扑结构, 与一般的前馈神经网络相比, 级联相关网络无需设置网络层数、隐层神经元数目, 训练速度较快, 但是在数据量较小时容易陷入过拟合。

3.5.5 Elman 网络

递归神经网络 (recurrent neural networks,RNN) 允许网络中出现环形结构, 从而让一些神经元的输出反馈来作为输入信号, 使得网络在 t 时刻的输出状态不仅与 t 时刻的输入有关, 还与 $t - 1$ 时刻的网络状态有关, 从而能处理与实践相关的动态变化。Elman 网络就是常用的 RNN 一种。

3.5.6 Boltzmann 机

一种基于能量 (energy) 的模型, 当能量最小化时取得网络的理想状态。

3.6 深度学习

3.7 小结

Chapter 4

支持向量机 (Support Vector Machine, SVM)

4.1 间隔与支持向量

在样本空间中，划分超平面可以通过如下线性方程组进行描述：

$$w^T x + b = 0 \quad (4.1)$$

样本空间中任一点 x ，到超平面 (w, b) 的距离可以写为：

$$\tau = \frac{|w^T x + b|}{\|w\|} \quad (4.2)$$

假设超平面 (w, b) 能使训练样本正确分类，即对于 $(x_i, y_i) \in D$ ，有：

$$\begin{cases} w^T x_i + b \geq +1, y_i = +1; \\ w^T x_i + b \leq -1, y_i = -1. \end{cases} \quad (4.3)$$

对于距离超平面最近的几个点使得上式的等号成立，那么它们被称为“支持向量 (support vector)”，如下图所示，两个异类支持向量到超平面的距离之和为：

$$\gamma = \frac{2}{\|w\|} \quad (4.4)$$

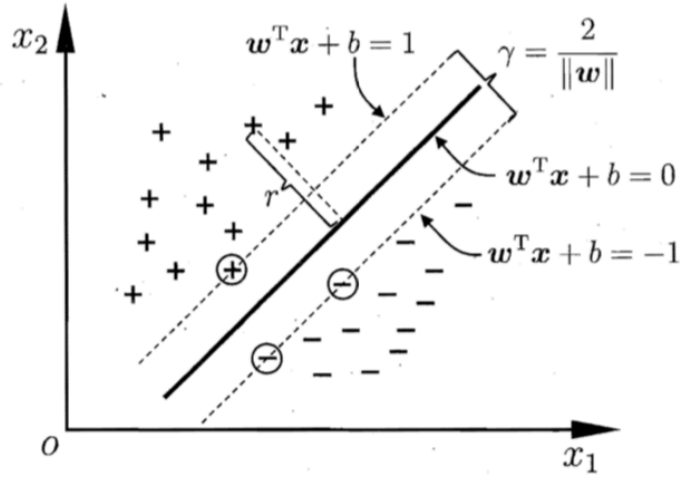


图 6.2 支持向量与间隔

欲找到最大间隔 (maximum margin) 来划分超平面, 也就是要找到能满足约束的 w, b , 使得 γ 最大, 即:

$$\max_{w, b} \frac{2}{\|w\|}, \quad s.t. \quad y_i(w^T x_i + b) \geq 1, i = 1, 2, \dots, m \quad (4.5)$$

即只需要最大化 $\frac{1}{\|w\|}$, 即最小化 $\|w\|^2$ 即可, 所以将上式改写一下:

$$\min_{w, b} \frac{1}{2} \|w\|^2, \quad s.t. \quad y_i(w^T x_i + b) \geq 1, i = 1, 2, \dots, m \quad (4.6)$$

这个就是支持向量机 (Support Vector Machine, SVM) 的基本型.

4.2 对偶问题

公式 4.6 是一个凸二次规划问题, 可以使用拉格朗日乘子法得到其的“对偶问题”(dual problem), 即对公式 4.6 每个约束添加拉格朗日乘子 $\alpha_i \geq 0$:

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 + \sum_{i=1}^m \alpha_i (1 - y_i(w^T x_i + b)) \quad (4.7)$$

令 $L(w, b, \alpha)$ 对 w 和 b 的偏导为零可得:

$$\begin{aligned} w &= \sum_{i=1}^m a_i y_i x_i \\ 0 &= \sum_{i=1}^m a_i y_i \end{aligned} \quad (4.8)$$

将 4.8 代入 4.7, 可以将 w 和 b 消去, 就可以得到 4.6 的对偶问题:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, \alpha_i \geq 0, i = 1, 2, \dots, m \end{aligned} \quad (4.9)$$

求解出 α 后, 再求出 w 和 b 即可求出模型:

$$f(x) = w^T x + b = \sum_{i=1}^m \alpha_i y_i x_i^T x + b \quad (4.10)$$

上述过程要满足 KKT(Karush-Kuhn-Tucker) 条件, 即要求:

$$\begin{cases} \alpha_i \leftarrow 0; \\ y_i f(x_i) - 1 \geq 0; \\ \alpha_i (y_i f(x_i) - 1) = 0 \end{cases} \quad (4.11)$$

上述约束条件显示了 SVM 一个重要的性质: 训练完成后, 大部分的训练样本都不需要保留, 最终模型仅仅与支持向量有关。

为了求解 4.9, 我们常常采用 SMO 算法, 其基本思路: 先固定 α_i 之外的所有参数, 然后求 α_i 上的极值。由于存在约束 $\sum_{i=1}^m \alpha_i y_i = 0$, 所以 SMO 每次选择两个变量 α_i 和 α_j 并固定其它参数, 不断重复以下的步骤知道收敛:

- 选取一对需要更新的变量 α_i 和 α_j ;
- 固定 α_i 和 α_j , 求解 3.9 获得更新后的 α_i 和 α_j ;

一般来说, 只要 α_i 和 α_j 有一个不满足 KKT 条件, 目标函数 4.9 就会在迭代中减小, 违背 KKT 的程度越大, 那么目标函数值减幅也会更大。基于这个性质 SMO 先选取违背 KKT 条件程度最大的变量, 第二个变量选取一个使得目标函数值减小最快的变量, 为了简便, 所选取的两个变量对应的样本之间的间隔应该最大。

SMO 算法之所以高效, 是应为它仅仅需要优化两个参数 α_i 和 α_j , 将 4.9 重写为:

$$\begin{aligned} \alpha_i y_i &= \alpha_j y_j = c, \alpha_i \geq 0, \alpha_j \geq 0, \\ c &= - \sum_{k \neq i, j} \alpha_k y_k \end{aligned} \quad (4.12)$$

用 4.12 消去 4.9 中的变量 α_j , 得到一个关于 α_i 的单变量二次规划的问题, 仅有约束 $\alpha_i \geq 0$ 。

对于任意的支持向量 (x_s, y_s) 都有 $y_s f(x_s) = 1$, 即:

$$y_s \left(\sum_{i \in S} \alpha_i y_i x_i^T x_s + b \right) = 1 \quad (4.13)$$

其中 $S = \{i | \alpha_i > 0, i = 1, 2, \dots, m\}$ 为了简便, 直接求解所有支持向量的平均值得到偏置 b :

$$b = \frac{1}{|S|} \sum_{s \in S} \left(y_s - \sum_{i \in S} \alpha_i y_i x_i^T x_s \right) \quad (4.14)$$

4.3 核函数

在实际应用中, 往往遇到的是线性不可分的问题, 遇到这种问题, 一般的做法是将样本映射到一个更加高维的空间, 使得样本能够在这个特征空间内线性可分。

令 $\phi(x)$ 表示将 x 映射后的特征向量, 于是, 在特征空间中划分超平面所对应的模型可表示为:

$$f(x) = w^T \phi(x) + b \quad (4.15)$$

类似上面的 4.6 和 4.9, 有:

$$\min_{w, b} \frac{1}{2} \|w\|^2 \quad s.t. \quad y_i (w^T \phi(x_i) + b) \geq 1, \quad i = 1, 2, \dots, m \quad (4.16)$$

其对偶问题为:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \\ \text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, \alpha_i \geq 0, i = 1, 2, \dots, m \end{aligned} \quad (4.17)$$

假设一个函数 $\kappa(x_i, x_j)$ 表示 $\phi(x_i)^T \phi(x_j)$ 来计算内积, 这个函数 κ 我们称为‘核函数’(kernel function):

$$\kappa(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle = \phi(x_i)^T \phi(x_j) \quad (4.18)$$

重写 4.17:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j) \\ \text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, \alpha_i \geq 0, i = 1, 2, \dots, m \end{aligned} \quad (4.19)$$

求解后:

$$f(x) = w^T \phi(x) + b = \sum_{i=1}^m \alpha_i y_i \kappa(x, x_i) + b \quad (4.20)$$

一般来说, 只要一个对称函数所对应的核矩阵是半正定的, 它就能作为核函数使用, 下图是一些常用的核函数:

表 6.1 常用核函数

名称	表达式	参数
线性核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$	
多项式核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^d$	$d \geq 1$ 为多项式的次数
高斯核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\ \mathbf{x}_i - \mathbf{x}_j\ ^2}{2\sigma^2}\right)$	$\sigma > 0$ 为高斯核的带宽(width)
拉普拉斯核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\ \mathbf{x}_i - \mathbf{x}_j\ }{\sigma}\right)$	$\sigma > 0$
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, $\beta > 0, \theta < 0$

核函数还会满足下面三个性质:

- 若 κ_1, κ_2 为核函数，那么对于任意正数 γ_1, γ_2 ，其线性组合也为核函数

$$\gamma_1 \kappa_1 = \gamma_2 \kappa_2 \quad (4.21)$$

- 若 κ_1, κ_2 为核函数，则核函数的直积也为核函数

$$\kappa_1 \otimes \kappa_2(x, z) = \kappa_1(x, z) \kappa_2(x, z) \quad (4.22)$$

- 若 κ_1 为核函数，则对任意函数 $g(x)$ ，

$$\kappa(x, z) = g(x) \kappa_1(x, z) g(z) \quad (4.23)$$

也是核函数

4.4 软间隔与正则化

在现实任务中，我们往往很难确定一个适合的核函数使得训练样本在特征空间中线性可分，很难断定这个貌似线性可分的结果是不是由于过拟合造成的。缓解该问题的一个办法是允许 SVM 在一些样本上出错，为此我们引入‘软间隔’(soft margin)，但是在最大化间隔的同时，不满足约束的样本应该尽可能减少，故优化目标可以写为：

$$\min_{w, b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \ell_{0/1}(y_i(w^T x_i + b) - 1) \quad (4.24)$$

其中： $C > 0$ 是一个常数， $\ell_{0/1}$ 是一个“0/1 损失函数”

$$\ell_{0/1}(z) = \begin{cases} 1, & \text{if } z < 0; \\ 0, & \text{otherwise.} \end{cases} \quad (4.25)$$

显由于 $\ell_{0/1}$ 性质不太好（非凸，非连续），我们常用其它一些函数进行代替，称为代替损失 (surrogate loss)，下面列举了常用的代替损失函数，如下图所示：

$$\begin{aligned} \text{hinge 损失: } \ell_{\text{hinge}}(z) &= \max(0, 1 - z) \\ \text{指数损失 (exponential loss): } \ell_{\text{exp}}(z) &= \exp(-z) \\ \text{对率损失 (logistic loss): } \ell_{\text{log}}(z) &= \log(1 + \exp(-z)) \end{aligned} \quad (4.26)$$

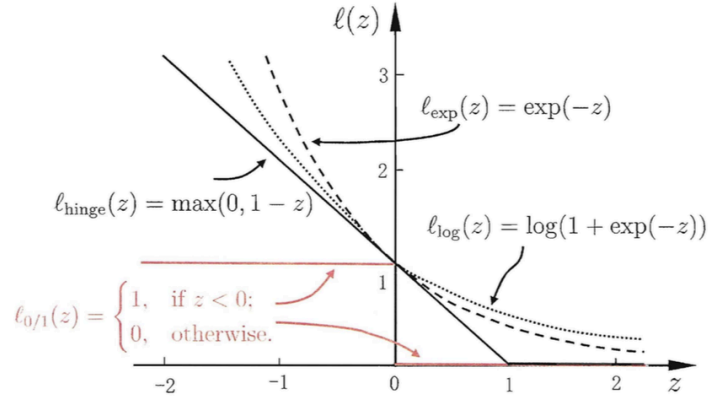


图 6.5 三种常见的替代损失函数: hinge损失、指数损失、对率损失

若采用 hinge 损失, 那么改写 4.24:

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \max(0, 1 - y_i(w^T x_i + b)) \quad (4.27)$$

引入松弛变量 (slack variables) $\xi_i \geq 0$, 重写上式:

$$\begin{aligned} \min_{w,b,\xi_i} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, 2, \dots, m \end{aligned} \quad (4.28)$$

这称为常用的‘软间隔 SVM’, 通过拉格朗日乘子法可得:

$$\begin{aligned} L(w, b, \alpha, \xi, \mu) = \\ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i + \sum_{i=1}^m \alpha_i (1 - \xi_i - y_i(w^T x_i + b)) - \sum_{i=1}^m \mu_i \xi_i \end{aligned} \quad (4.29)$$

其中 $\alpha_i \geq 0, \mu_i \geq 0$ 是拉格朗日乘子。令 $L(w, b, \alpha, \xi, \mu)$ 对 w, b, ξ_i 求偏导为零可得:

$$\begin{aligned} w &= \sum_{i=1}^m \alpha_i y_i x_i \\ 0 &= \sum_{i=1}^m \alpha_i y_i \\ C &= \alpha_i + \mu_i \end{aligned} \quad (4.30)$$

联立上面两个公式，可得 4.28 的对偶问题：

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, 0 \leq \alpha_i \leq C, i = 1, 2, \dots, m. \end{aligned} \quad (4.31)$$

软间隔与硬间隔的唯一差别是在于对偶变量的约束不同，前者是 $0 \leq \alpha_i \leq C$ ，后者是 $0 \leq \alpha_i$ ，那么软间隔 KKT 条件相应改为：

$$\begin{cases} \alpha_i \geq 0, & \mu_i \geq 0, \\ y_i f(x_i) - 1 + \xi_i \geq 0, \\ \alpha_i (y_i f(x_i) - 1 + \xi_i) = 0, \\ \xi_i \geq 0, & \mu_i \xi_i = 0 \end{cases} \quad (4.32)$$

通过上述 KKT 条件，可以看出软间隔 SVM 最终模型仅仅与支持向量有关，即通过采用 hinge 损失函数仍然保持了稀疏性。

我们可以把上面的优化目标写成更加一般的形式，其中第一项用来描述划分超平面的‘间隔’大小，另一项 $\sum_{i=1}^m \ell(f(x_i), y_i)$ 用来表述训练集上的误差：

$$\min_f \Omega(f) + C \sum_{i=1}^m \ell(f(x_i), y_i) \quad (4.33)$$

其中 $\Omega(f)$ 称为结构分析按 (structural risk)，用来描述模型 f 的某些性质， $\sum_{i=1}^m \ell(f(x_i), y_i)$ 称为经验风险 (empirical risk)，用于描述模型与训练数据的契合程度， C 对两者进行权重。 $\Omega(f)$ 也称为正则化项， C 称为正则化函数。

4.5 支持向量回归 (Support Vector Regression, SVR)

SVR 与传统的回归模型有些许不同，他假设我们能容忍模型输出 $f(x)$ 与真实输出 y 之间最多有 ϵ ，当且仅当 $|f(x) - y| > \epsilon$ 才计算损失，相当于以 $f(x)$ 为中间，构建一个宽度为 2ϵ 的间隔带，若样本落入这个间隔带，则认为分类时正确的。

SVR 问题可以形式化为:

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \ell_{\epsilon}(f(x_i) - y_i) \quad (4.34)$$

其中 ℓ_{ϵ} 称为 ϵ -不敏感损失函数 (ϵ -insensitive loss):

$$\ell_{\epsilon}(z) = \begin{cases} 0, & \text{if } |z| \leq \epsilon \\ |z| - \epsilon, & \text{otherwise} \end{cases} \quad (4.35)$$

引入松弛变量 ξ_i 和 $\hat{\xi}_i$, 重写上式:

$$\begin{aligned} \min_{w,b,\xi_i,\hat{\xi}_i} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m (\xi_i + \hat{\xi}_i) \\ \text{s.t.} \quad & f(x_i) - y_i \leq \epsilon + \xi_i, \quad y_i - f(x_i) \leq \epsilon + \hat{\xi}_i, \\ & \xi_i, \hat{\xi}_i \geq 0, i = 1, 2, \dots, m \end{aligned} \quad (4.36)$$

引入拉格朗日乘子法: $\mu_i \geq 0, \hat{\mu}_i \geq 0, \alpha_i \geq 0, \hat{\alpha}_i \geq 0$:

$$\begin{aligned} L(w, b, \alpha, \hat{\alpha}, \xi, \hat{\xi}, \mu, \hat{\mu}) = \\ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m (\xi_i + \hat{\xi}_i) - \sum_{i=1}^m \mu_i \xi_i - \sum_{i=1}^m \hat{\mu}_i \hat{\xi}_i + \\ \sum_{i=1}^m \alpha_i (f(x_i) - y_i - \epsilon - \xi_i) + \sum_{i=1}^m \hat{\alpha}_i (y_i - f(x_i) - \epsilon - \hat{\xi}_i) \end{aligned} \quad (4.37)$$

对上式进行求导:

$$\begin{aligned} w &= \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) x_i \\ 0 &= \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) \\ C &= \alpha_i + \mu_i \\ C &= \hat{\alpha}_i + \hat{\mu}_i \end{aligned} \quad (4.38)$$

所以我们可以计算出 SVR 的对偶问题:

$$\begin{aligned} \max_{\alpha, \hat{\alpha}} \quad & \sum_{i=1}^m y_i (\hat{\alpha}_i - \alpha_i) - \epsilon (\hat{\alpha}_i + \alpha_i) - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m (\hat{\alpha}_i (\hat{\alpha}_j - \alpha_j) x_i^T x_j) \\ \text{s.t.} \quad & \sum_{i=1}^m (\hat{\alpha}_i + \alpha_i) = 0, \quad 0 \leq \alpha_i, \hat{\alpha}_i \leq C \end{aligned} \quad (4.39)$$

满足以下 KKT 条件:

$$\begin{cases} \alpha_i (f(x_i) - y_i - \epsilon - \xi_i) = 0, \\ \hat{\alpha}_i (y_i - f(x_i) - \epsilon - \hat{\xi}_i) = 0, \\ \alpha_i \hat{\alpha}_i = 0, \xi_i \hat{\xi}_i = 0, \\ (C - \alpha_i) \xi_i = 0, (C - \hat{\alpha}_i) \hat{\xi}_i = 0 \end{cases} \quad (4.40)$$

综上, SVR 的解形可以写为:

$$f(x) = \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) x_i^T x + b \quad (4.41)$$

能使得上式中的 $\hat{\alpha}_i - \alpha_i \neq 0$ 的样本即为 SVR 的支持向量, 他们必然落于 ϵ -间隔带外, 在得到 α_i 后, 若 $0 < \alpha_i < C$, 则必有 $\xi_i = 0$, 进而有:

$$b = y_i + \epsilon - \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) x_i^T x \quad (4.42)$$

一般在实践中, 我们采取更加鲁棒的做法: 选取多个满足条件 $0 < \alpha_i < C$ 的样本后求解 b 取平均值, 考虑到映射的形式, SVR 可表示为:

$$f(x) = \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) \kappa(x, x_i) + b \quad (4.43)$$

其中 $\kappa(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ 为核函数。

4.6 核方法

表示定理: 满足 Ω 单挑递增, ℓ 非负损失函数, 对于优化问题:

$$\min_{h \in \mathbb{H}} F(h) = \Omega(\|h\|_{\mathbb{H}}) + \ell(h(x_1), h(x_2), \dots, h(x_m)) \quad (4.44)$$

的解总可写为：

$$h^*(x) = \sum_{i=1}^m \alpha_i \kappa(x, x_i) \quad (4.45)$$

一般来说，人们把这种方法称为核方法 (kernel methods)，最常见的就是通过引入核函数来核化将线性学习期拓展为非线性拓展，从而得到核线性判别分析 (Kernelized Linear Discriminant Analysis, KLDA).

我们可以假设通过某种映射 $\phi : \mathcal{X} \rightarrow \mathbb{F}$ 将样本空间到一个特征空间 \mathbb{F} ，然后在 \mathbb{F} 中执行线性判别分析，以求得：

$$h(x) = w^T \phi(x) \quad (4.46)$$

那么 KLDA 的学习目标：

$$\max_w J(w) = \frac{w^T S_b^\phi w}{w^T S_w^\phi w} \quad (4.47)$$

其中 $w^T S_b^\phi w$ 表示 \mathbb{F} 中的类间散度矩阵，而 $w^T S_w^\phi w$ 类内散度矩阵。令 X_i 表示第 $i \in 0, 1$ 类样本的集合，其样本数为 m_i ，总样本数为 $m = m_0 + m_1$ ，那么第 i 类样本在特征空间 \mathbb{F} 的均值为：

$$\mu_i^\phi = \frac{1}{m_i} \sum_{x \in X_i} \phi(x) \quad (4.48)$$

两个散度矩阵分别为：

$$\begin{aligned} S_b^\phi &= (\mu_1^\phi - \mu_0^\phi)(\mu_1^\phi - \mu_0^\phi)^T \\ S_w^\phi &= \sum_{i=0}^1 \sum_{x \in X_i} (\phi(x) - \mu_i^\phi)(\phi(x) - \mu_i^\phi)^T \end{aligned} \quad (4.49)$$

我们用核函数 $\kappa(x, x_i)$ 来表示，那么 $h(x)$ 可写为：

$$h(x) = \sum_{i=1}^m \alpha_i \kappa(x, x_i) \quad (4.50)$$

故：

$$w = \sum_{i=1}^m \alpha_i \phi(x_i) \quad (4.51)$$

令 $K \in \mathbb{R}^{m \times m}$ 为 κ 的核矩阵, 那么:

$$\begin{aligned}\hat{\mu}_0 &= \frac{1}{m_0} K l_0, \\ \hat{\mu}_1 &= \frac{1}{m_1} K l_1, \\ M &= (\hat{\mu}_0 - \hat{\mu}_1)(\hat{\mu}_0 - \hat{\mu}_1)^T, \\ N &= K K^T - \sum_{i=0}^l m_i \hat{\mu}_i \hat{\mu}_i^T,\end{aligned}\tag{4.52}$$

于是 4.47 等价于:

$$\max_{\alpha} J(\alpha) = \frac{\alpha^T M \alpha}{\alpha^T N \alpha}\tag{4.53}$$

用线性判别分析求解方法即可得到 α , 从而求出 $h(x)$ 。

4.7 小结

Chapter 5

贝叶斯分类器 (bayes classifier)

由于在深极做过一次 ppt 的演讲了，所以这一章不用写得太详细。

5.1 贝叶斯公式

贝叶斯公式的本质就是通过条件概率 (也叫似然) 之间的转化，建立先验概率 $P(x|c)$ 与后验概率 $P(c|x)$ 之间的联系。

$$P(c|x) = \frac{P(c, x)}{P(x)} = \frac{P(x|c)P(c)}{P(x)} \quad (5.1)$$

5.2 贝叶斯决策论 (bayesian decision theory)

假设有 N 种可能的类别标记。即 $\mathcal{Y} = \{c_1, c_2, \dots, c_N\}$, λ_{ij} 是将一个真实标记为 c_j 的样本误分类为 c_i 的损失，那么基于后验概率 $P(c_i|x)$ 就可以获得将样本 x 分类为 c_i 所产生的期望损失，即 x 上的条件风险 (conditional risk)。

$$R(c_i|x) = \sum_{j=1}^N \lambda_{ij} P(c_j|x) \quad (5.2)$$

我们的目标即找出 x 的分类，使得期望风险 $R(c|x)$ 最小：

$$h^*(x) = \arg \min_{c \in \mathcal{Y}} R(c|x) \quad (5.3)$$

5.3 极大似然估计 (Maximum Likelihood Estimation, MLE)

- 对 θ_c 进行极大似然估计。就是寻找能最大化似然 $P(D_c|\theta_c)$ 的参数值 $\hat{\theta}_c$ 。换句话说，就是遍历 θ_c 所有可能的取值，找出一个使数据出现的“可能性”最大的一个。
- 为了加快计算速度和减少溢出的可能性，可以用取对数相加代替连乘的操作：

$$LL(\theta_c) = \log P(D_c|\theta_c) = \sum_{x \in D_c} \log P(x|\theta_c) \quad (5.4)$$

- 此时参数 θ_c 的极大似然估计 $\hat{\theta}_c$ 可以写为：

$$\hat{\theta}_c = \arg \max_{\theta_c} LL(\theta_c) \quad (5.5)$$

- 总之，MLE 的思想可以总结为：已知某个参数能使这个样本出现的概率最大，我们当然不会再去选择其他小概率的样本，所以干脆就把这个参数作为估计的真实值。

5.4 朴素贝叶斯分类器 (Naïve Bayes Classifier)

- 朴素 (Naïve)，指的是“属性条件独立性假设”，即对于已知类别，假设所有的属性相互独立。聊天监控系统里采用的贝叶斯算法就是基于 NBC 的。
- 在这个前提下，贝叶斯公式可以改写：

$$P(c|x) = \frac{P(c)P(x|c)}{P(x)} = \frac{P(c)}{P(x)} \prod_{i=1}^d P(x_i|c) \quad (5.6)$$

其中， d 为属性的数目， x_i 为 x 在第 i 个属性上的取值。

- 由上，我们可得朴素贝叶斯分类器的表达式。

$$h_{nb}(x) = \arg \max_{c \in \mathcal{Y}} P(c) \prod_{i=1}^d P(x_i|c) \quad (5.7)$$

- 对于贝叶斯分类器，若前提有大量的训练集，我们可以事先训练贝叶斯分类所有的概率估值，进行测试时，我们只需进行查表操作即可，借助哈希表、二叉树等数据结构进行存储，贝叶斯分类器的时间复杂度为 $O(n)$ 。

5.5 半朴素贝叶斯分类器 (semi-Naïve Bayes Classifier)

- NBC 是基于属性之间是相互独立的假设，但是在现实条件下这个假设是很难实现的，所以我们提出了半朴素贝叶斯分类器，它的基本想法是适当考虑一部分属性间的相互依赖信息，从而既不需要进行完全联合计算，又不至于彻底忽略了比较强的属性依赖关系。
- 采用得比较多得是“独依赖估计”(One-Dependent Estimator, ODE), 指的是每个属性最多仅依赖一个其他属性，即：

$$P(c|x) \propto P(c) \prod_{i=1}^d P(x_i|c, pa_i) \quad (5.8)$$

其中 pa_i 为属性 x_i 所依赖的属性，称为 x_i 的父属性。

5.6 贝叶斯网 (Bayesian network)

- 贝叶斯网也称为信念网 (belief network)。它借助有向无环图来刻画属性间的依赖关系，并使用条件概率表来描述属性的联合概率分布。
- 贝叶斯网 B 可以表示成如下公式：

$$B = \langle G, \Theta \rangle \quad (5.9)$$

其中 G 表示一个有向无环图， Θ 定量描述两个属性之间的直接依赖关系。假设属性 x_i 在 G 中的父结点集为 π_i ，则 Θ 包含了每个属性的条件概率表 $\theta_{x_i|\pi_i} = P_B(x_i|\pi_i)$ 。

- 贝叶斯网学习的首要任务是通过训练集构建一个最合理的贝叶斯网, 一般采用评分搜索的办法。
- 首先定义一个评分函数 (score function), 基于信息论准则, 其目标是找到一个能以最小编码长度描述训练模型, 即“最小描述长度”(Minimal Description Length, MDL)

5.6.1 贝叶斯网 (Bayesian network)-学习

求 MDL 的过程可以描述如下:

1. 给定训练集 $D = \{x_1, x_2, \dots, x_m\}$, 那么贝叶斯网 $B = \langle G, \Theta \rangle$ 的评分函数可以写为:

$$s(B|D) = f(\theta)|B| - LL(B|D) \quad (5.10)$$

其中 $|B|$ 是贝叶斯网的参数个数; $f(\theta)$ 表示描述每个参数所需的字节数, $LL(B|D)$ 表示贝叶斯网 B 的对数似然。

$$LL(B|D) = \sum_{i=1}^m \log P_B(x_i) \quad (5.11)$$

2. 学习任务转化为一个优化任务, 即寻找一个贝叶斯网 B 使评分函数 $s(B|D)$ 最小。
3. 若 $f(\theta) = 0$, 即不计算对网络编码的长度, 评分函数退化成负对数似然, 那么学习任务退化成极大似然估计。

$$s(B|D) = -LL(B|D) = -\sum_{i=1}^m \log P_B(x_i) \quad (5.12)$$

4. 若 $B = \langle G, \Theta \rangle$ 的网络结构 G 固定, 则 $s(B|D)$ 等价于对参数 Θ 的极大似然估计, 那么 $\theta_{x_i|\pi_i}$ 可以直接在训练数据 D 上通过经验估计得到:

$$\theta_{x_i|\pi_i} = \hat{P}_D(x_i|\pi_i) \quad (5.13)$$

其中 $\hat{P}_D(\cdot)$ 是 D 上的经验分布。

5. 为了最小化评分函数 $s(B|D)$ ，只需要对网络结构进行搜索，而候选结构的最优参数可以直接在训练集上计算得到。
6. 搜索出贝叶斯网最优结构是一个 NP 难的问题，难以快速求解，一般常用两种方法保证在有限时间内求得近似解。
 - 采用贪心策略，从某个网络结构出发，每次调整一条边，直到评分函数值不再降低为止。
 - 通过网络结构施加约束来削减搜索空间，例如将网络结构限定为树形结构。

5.6.2 贝叶斯网 (Bayesian network)-推断

- 通过前面的训练和学习，贝叶斯网就可以通过一些属性变量的观测值来推测其他属性变量的取值，这个过程我们称为“推断”(inference)，已知变量观测值称为“证据”(evidence)。
- 理想情况下是直接根据贝叶斯网定义的联合概率分布计算后验概率，但前面已经说明搜索最优结构是 NP 难的，在现实应用中，贝叶斯网的近似推断常使用吉布斯采样 (Gibbs sampling) 来完成。

5.7 EM(Expectation-Maximization) 算法

- 在实际应用的时候，我们很难获得所有属性变量的值，即训练样本是不完整的，像这种无法获得属性变量的“未观测”变量，我们称为“隐变量”(latent variable)。
- EM(Expectation-Maximization) 算法，就是常用的估计参数隐变量的算法。它的基本思想很简单：若参数 Θ 已知，则可根据训练数据推断出最优隐变量 Z 的值 (E 步)；反之，若 Z 的值已知，则可方便地对 Θ 做极大似然估计 (M 步)。

若我们不是取 Z 的期望，而是基于 Θ^t 计算隐变量 Z 的概率分布 $P(Z|X, \Theta^t)$ ，那么 em 算法两步可以直接定义：

- E 步 (Expectation): 以当前参数 Θ^t 推断隐变量分布 $P(Z|X, \Theta^t)$, 并计算对数似然 $LL(\Theta|X, Z)$ 关于 Z 的期望。

$$Q(\Theta|\Theta^t) = \mathbb{E}_{Z|X, \Theta^t} LL(\Theta|X, Z) \quad (5.14)$$

- M 步 (Maximization): 寻找参数最大化期望似然:

$$\Theta^{t+1} = \arg \max_{\Theta} Q(\Theta|\Theta^t) \quad (5.15)$$

5.8 小结

贝叶斯模型是一个 precision 很高的模型, 而且属于线性模型, 十分便于工程实现。

最简单的朴素贝叶斯分类器, 在很多情况下都能够获得相当好的性能, 十分适合信息检索领域, 是常用的文本分类策略之一。

Chapter 6

集成学习 (ensemble learning)

6.1 个体与集成

集成学习指的是将几个弱学习器 (weaker learner, 一般就是指基学习器) 按照某种策略组合在一起进行学习。首先说明几个概念:

- 基学习器: 表示构成的集成所用的算法都是一致的
- 组件学习器: 表示构成的集成所用的算法都是不一致的

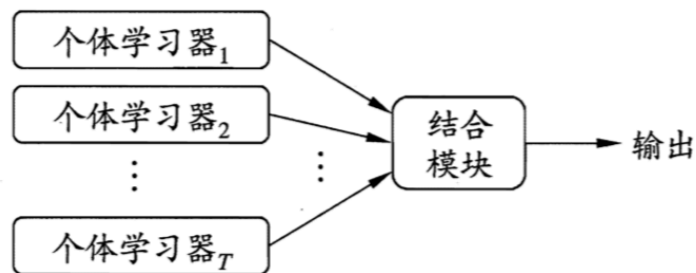


图 8.1 集成学习示意图

假定基分类器的错误率为 ϵ , 即对每个基分类器 h_i 有:

$$P(h_i(x) \neq f(x)) = \epsilon \quad (6.1)$$

若采用投票法，即超过半数基分类器认为是正确的，则集成分类就正确：

$$H(x) = \text{sign} \left(\sum_{i=1}^T h_i(x) \right) \quad (6.2)$$

若基分类器之间的错误率相互之间是独立的，那么根据 Hoeffding 不等式，集成错误率为：

$$P(h_i(x) \neq f(x)) = \sum_{k=0}^{[T/2]} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k} \leq \exp \left(-\frac{1}{2} T (1-2\epsilon)^2 \right) \quad (6.3)$$

说明随着个体分类器数目 T 的不断增大，集成的错误率将逐渐下降。当前二种集成学习的思想：

- 各个个体学习期之间存在强依赖的关系、必须串行生成序列化的方法，如 Boosting
- 各个个体学习期之间不存在强依赖关系、可以同时生成的并行化方法，如 Bagging 或者随机森林。

6.2 Boosting

Boosting 算法的工作机制很类似：

1. 从初始训练集中选择一个基学习器
2. 再根据基学习器的表现对训练样本分布进行调整，使之前分类错误的训练样本在后面受到更多关注。
3. 基于调整后的样本分布来训练下一个基学习器
4. 直到基学习器数目到达事先指定的值 T ，最终将这 T 个基学习器进行加权结合。

其中最出名的 Boosting 算法是 AdaBoost，一般用的是基学习器的线性组合：

$$H(x) = \sum_{t=1}^T \alpha_t h_t(x) \quad (6.4)$$

来最小化指数损失函数 (exponential loss function):

$$\ell_{\text{exp}}(H|\mathcal{D}) = \mathbb{E}_{x \sim \mathcal{D}}[e^{-f(x)H(x)}] \quad (6.5)$$

对应的算法流程图:算法用 \LaTeX 打起来太麻烦了,先用截图,以后有时间再改成 \LaTeX

```

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

过程:
1:  $\mathcal{D}_1(x) = 1/m$ .
2: for  $t = 1, 2, \dots, T$  do
3:    $h_t = \mathfrak{L}(D, \mathcal{D}_t)$ ;
4:    $\epsilon_t = P_{x \sim \mathcal{D}_t}(h_t(x) \neq f(x))$ ;
5:   if  $\epsilon_t > 0.5$  then break
6:    $\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$ ;
7:   
$$\mathcal{D}_{t+1}(x) = \frac{\mathcal{D}_t(x)}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(x) = f(x) \\ \exp(\alpha_t), & \text{if } h_t(x) \neq f(x) \end{cases}$$

      
$$= \frac{\mathcal{D}_t(x) \exp(-\alpha_t f(x) h_t(x))}{Z_t}$$

8: end for

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

```

图 8.3 AdaBoost 算法

对 $H(x)$ 进行偏导:

$$\frac{\partial \ell_{\exp}(H|\mathcal{D})}{\partial H(x)} = -e^{-H(x)}P(f(x)=1|x) + e^{H(x)}P(f(x)=-1|x) \quad (6.6)$$

令上式为 0, 有:

$$H(x) = \frac{1}{2} \ln \frac{P(f(x) = 1|x)}{P(f(x) = -1|x)} \quad (6.7)$$

于是：

$$\begin{aligned}
\text{sign}(H(x)) &= \text{sign}\left(\frac{1}{2} \ln \frac{P(f(x)=1|x)}{P(f(x)=-1|x)}\right) \\
&= \begin{cases} 1, & P(f(x)=1|x) > P(f(x)=-1|x) \\ -1, & P(f(x)=1|x) < P(f(x)=-1|x) \end{cases} \quad (6.8) \\
&= \arg \max_{y \in \{-1,1\}} P(f(x)=y|x)
\end{aligned}$$

上式表明，若使得指数损失函数最小化了，那么分类错误率也将得到最小化；在 AdaBoost 算法中，第一个基分类器 h_1 是通过直接将基学习算法用于初始数据分布而得到的，此后迭代地生成 h_t 和 α_t ，当基分类器 h_t 的基于分布 \mathcal{D} 产生后，该基分类器的权重 α_t 应使得 $\alpha_t h_t$ 最小化指数函数：

$$\ell_{\exp}(\alpha_t h_t | \mathcal{D}_t) = e^{-\alpha_t} (1 - \epsilon_t) + e^{\alpha_t} \epsilon_t \quad (6.9)$$

其中 $\epsilon_t = P_{x \sim \mathcal{D}_t}(h_t(x) \neq f(x))$ ，对 α_t 进行求导：

$$\frac{\partial \ell_{\exp}(\alpha_t h_t | \mathcal{D}_t)}{\partial \alpha_t} = -e^{-\alpha_t} (1 - \epsilon_t) \quad (6.10)$$

令上式为 0，可得：

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \quad (6.11)$$

AdaBoost 算法能在获得 H_{t-1} 之后的样本分布进行相应的调整，使得下一轮的基学习器 h_t 能够纠正 H_{t-1} 的一些错误，理想的情况下， h_t 能纠正 H_{t-1} 的所有错误，达到理想化：

$$\begin{aligned}
h_t(x) &= \arg \min_h \ell_{\exp}(H_{t-1} + h | \mathcal{D}) \\
&= \arg \max_h \mathbb{E}_{x \sim \mathcal{D}} = \left[\frac{e^{-f(x)H_{t-1}(x)}}{\mathbb{E}_{x \sim \mathcal{D}}[e^{-f(x)H_{t-1}(x)}]} f(x)h(x) \right] \quad (6.12)
\end{aligned}$$

其中 $\mathbb{E}_{x \sim \mathcal{D}}[e^{-f(x)H_{t-1}(x)}]$ 是一个常数，令 \mathcal{D}_t 表示一个分布：

$$\mathcal{D}_t(x) = \frac{\mathcal{D}(x)e^{-f(x)H_{t-1}(x)}}{\mathbb{E}_{x \sim \mathcal{D}}[e^{-f(x)H_{t-1}(x)}]} \quad (6.13)$$

根据数学的期望，这等价于：

$$h_t(x) = \arg \max_h \mathbb{E}_{x \sim \mathcal{D}_t} [f(x)h(x)] \quad (6.14)$$

由于 $f(x), h(x) \in \{-1, +1\}$ ，有：

$$h_t(x) = \arg \min_h \mathbb{E}_{x \sim \mathcal{D}_t} [\mathbb{I}(f(x) \neq h(x))] \quad (6.15)$$

综上：理想的 h_t 将在分布 \mathcal{D}_t 上最小化分类误差，类似于残差逼近的思想。考虑到 \mathcal{D}_t 和 \mathcal{D}_{t+1} 的关系，有：

$$\mathcal{D}_{t+1}(x) = \mathcal{D}_t(x) \cdot e^{-f(x)\alpha_t h_t(x)} \frac{\mathbb{E}_{x \sim \mathcal{D}} [e^{-f(x)H_{t-1}(x)}]}{\mathbb{E}_{x \sim \mathcal{D}} [e^{-f(x)H_t(x)}]} \quad (6.16)$$

对应算法流程图中的第七行样本分布的更新公式。

Boosting 算法要求基学习器能对特定的数据分布进行学习，可以通过‘重赋权法’实施，即在迭代的每一轮中，根据样本分布为每个训练样本重新赋予一个权重，对于无法接受代权样本的基学习方法，可以通过重采样法进行处理，即在每一轮学习中，根据样本分布对训练集进行重新采样，再进行训练。需要注意，Boosting 算法在训练每一轮都要检查当前生成的基学习器是否满足基本条件，一旦不满足，直接抛弃。

从偏差-方差分解的角度来说，Boosting 主要关注如何降低偏差，因此 Boosting 能基于泛化性能相当弱的学习器构建出很强的集成。

6.3 Bagging 与随机森林

在上一节中，我们可以知道，想得到泛化性能强的集成，集成中的个体学习器应当尽可能的相互独立，在实际应用中，应当使个体学习器尽可能有较大的差异。

6.3.1 Bagging

Bagging 可以说是并行式集成学习的著名代表，它是基于自主采样法 (bootstrap sampling)，给定包含 m 个样本的数据集，我们先随机选择一个样本放入采样集中，然后将其放回初始数据集。照这样，我们可以采样出 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组合，这就是 bagging 的基本思想。如下图所示：

输入： 训练集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$;
 基学习算法 \mathcal{L} ;
 训练轮数 T .

过程：

- 1: **for** $t = 1, 2, \dots, T$ **do**
- 2: $h_t = \mathcal{L}(D, \mathcal{D}_{bs})$
- 3: **end for**

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

图 8.5 Bagging 算法

假定基学习器的时间复杂度为 $O(m)$ ，那么 Bagging 的复杂度大概为 $T(O(m) + O(s))$ ，说明 Bagging 是一个近似的线性时间复杂度算法，能够不经修改地应用于多分类、回归等任务。

从第二章我们知道，每个基学习器只使用了大约 63.2% 的样本，意味着剩下的 36.8% 可以用于验证集对泛化性能的包外估计 (out-of-bag estimate)。包外样本有许多用途：

- 当基学习器是决策树时，可以采用包外样本进行减值减少过拟合
- 当基学习器是神经网络时，可以用包外样本来辅助早期的停止来减少过拟合。

6.3.2 随机森林 (Random Forest, RF)

随机森林是 Bagging 的一个扩展变体，RF 在以决策树为基学习器的基础上，进一步在决策树的训练过程中引入随机属性的选择，在 RF 中，对于基决策树的每个节点，先从该节点的属性集合中随机选择一个包含 k 个属性的子集，然后在这个子集中选择一个最优属性用于划分。这里的参数 k 表示

了随机性的引入程度，若 $k = d$ 表明与传统决策树相同，若 $k = 1$ 表明随机只选择一个属性用于划分，随机性最大，一般情况下，推荐选择 $k = \log_2 d$ 。

6.4 结合策略

组合策略值得是将几个基学习器组合在一起，来避免以下情况：

- 单学习器泛化性能不佳。
- 避免陷入局部极小。
- 可以扩大相应的假设空间。

6.4.1 平均法 (averaging)

对于数值型的输出， $h_i(x) \in \mathbb{R}$ ，最常用的就是平均法了：

- 简单平均 (simple averaging):

$$H(x) = \frac{1}{T} \sum_{i=1}^T h_i(x) \quad (6.17)$$

- 加权平均 (weighted averaging):

$$H(x) = \sum_{i=1}^T w_i h_i(x) \quad (6.18)$$

加权平均法的权重一般都是从训练数据中获得，但是由于数据不充分或者存在噪声，使得权重并不是完全可靠。若规模较大，容易陷入过拟合。一般而言，个体学习器的性能相差较大时采用加权平均法，相近时采用简单平均法。

6.4.2 投票法 (voting)

- 绝对多数投票法 (majority voting), 若某个 label 投票超过半数, 那么就预测为该 label。

$$H(x) = \begin{cases} c_j, & \text{if } \sum_{i=1}^T h_j^i(x) > 0.5 \sum_{k=1}^N \sum_{i=1}^T h_i^k(x); \\ reject, & \text{otherwise.} \end{cases} \quad (6.19)$$

- 相对多数投票法 (plurality voting), 选择得票数最高的 label, 若最高有多个, 则随机选择一个。

$$H(x) = c_{\arg \max_j \sum_{i=1}^T h_i^j(x)} \quad (6.20)$$

- 加权投票法 (weighted voting):

$$H(x) = c_{\arg \max_j \sum_{i=1}^T w_i h_i^j(x)} \quad (6.21)$$

6.4.3 学习法

学习法的原理是把初级学习器的输出结果当做次级训练器的样本进行训练。如 Stacking 算法:

输入: 训练集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$;
 初级学习算法 $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_T$;
 次级学习算法 \mathcal{L} .
 过程:
 1: **for** $t = 1, 2, \dots, T$ **do**
 2: $h_t = \mathcal{L}_t(D)$;
 3: **end for**
 4: $D' = \emptyset$;
 5: **for** $i = 1, 2, \dots, m$ **do**
 6: **for** $t = 1, 2, \dots, T$ **do**
 7: $z_{it} = h_t(\mathbf{x}_i)$;
 8: **end for**
 9: $D' = D' \cup ((z_{i1}, z_{i2}, \dots, z_{iT}), y_i)$;
 10: **end for**
 11: $h' = \mathcal{L}(D')$;
 输出: $H(\mathbf{x}) = h'(h_1(\mathbf{x}), h_2(\mathbf{x}), \dots, h_T(\mathbf{x}))$

图 8.9 Stacking 算法

6.5 多样性

6.5.1 误差-分歧分解

定义学习器 h_i 的分歧为:

$$A(h_i|x) = (h_i(x) - H(x))^2 \quad (6.22)$$

则集成的分歧:

$$\overline{A}(h|x) = \sum_{i=1}^T w_i A(h_i|x) = \sum_{i=1}^T w_i (h_i(x) - H(x))^2 \quad (6.23)$$

分歧项表明了个体学习器在样本 x 上的不一致性，即在一定程度上反映了个体学习器的多样性，那么个体学习器 h_i 和集成 H 的平方误差分别为：

$$\begin{aligned} E(h_i|x) &= (f(x) - h_i(x))^2 \\ E(H|x) &= (f(x) - H(x))^2 \end{aligned} \quad (6.24)$$

令 $\bar{E}(h|x) = \sum_{i=1}^T w_i \cdot E(h_i|x)$ 表示个体学习器误差的加权均值，有：

$$\bar{A}(h|x) = \bar{E}(h|x) - E(H|x) \quad (6.25)$$

令 $p(x)$ 表示样本的概率密度，则在全样本上有：

$$\begin{aligned} \sum_{i=1}^T w_i \int A(h_i|x)p(x)dx &= \\ \sum_{i=1}^T \int E(h_i|x)p(x)dx - \int E(H|x)p(x)dx \end{aligned} \quad (6.26)$$

其中 $E_i = \int E(h_i|x)p(x)dx$ 为全样本的泛化误差， $A_i = \int A(h_i|x)p(x)dx$ 为分歧项，而 $E = \int E(H|x)p(x)dx$ 为集成的泛化误差。

综上：

$$E = \bar{E} - \bar{A} \quad (6.27)$$

上式表明，个体学习器准确性越高，多样性越大，则集成越好。这就是集成-分期分解，但是这个理论中的 \bar{A} 不是一个可以直接操作的多样性度量，只适合进行回归学习而不是分类学习。

6.5.2 多样性的度量 (diversity measure)

多样性度量用于估算个体学习器的多样化程度，典型的做法是考虑个体分类器的两两相似/不相似性。

给定数据集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ ，对于二分类的任务， $y_i \in \{-1, +1\}$ ，则分类器 h_i 和 h_j 预测结果如下表：

	$h_i = +1$	$h_i = -1$
$h_j = +1$	a	c
$h_j = -1$	b	d

其中: $a + b + c + d = m$, 那么:

- 不合度量 (disagreement measure)

$$dis_{ij} = \frac{b + c}{m} \quad (6.28)$$

其中 $dis_{ij} \in [0, 1]$, 值越大说明多样性越大。

- 相关系数 (correlation coefficient)

$$\rho_{ij} = \frac{ad - bc}{\sqrt{(a + b)(a + c)(c + d)(b + d)}} \quad (6.29)$$

$\rho_{ij} \in [-1, 1]$, 若 h_i 与 h_j 无关, 那么 $\rho_{ij} = 0$, 正相关 $\rho_{ij} > 0$, 负相关则 $\rho_{ij} < 0$

- Q -统计量 (Q -stasistic)

$$Q_{ij} = \frac{ad - bc}{ad + bc} \quad (6.30)$$

Q_{ij} 与相关系数 ρ_{ij} 相同, 且 $|Q_{ij}| \leq |\rho_{ij}|$

- κ -统计量 (κ -statistic)

$$\kappa = \frac{p_1 - p_2}{1 - p_2} \quad (6.31)$$

其中, p_1 是两个分类器取得一致的概率, p_2 是两个分类器偶然达成一致的。由下面公式进行估算

$$\begin{aligned} p_1 &= \frac{a + d}{m} \\ p_2 &= \frac{(a + b)(a + c) + (c + d)(b + d)}{m^2} \end{aligned} \quad (6.32)$$

6.5.3 多样性增强

原理是在学习过程中引入随机性, 常见的做法是对数据样本、输入属性、输出熟悉、算法参数进行扰动。

- 数据样本扰动

给定初始数据集, 进行随机采样, 对于不稳定学习器 (那些训练样本稍微有些改变就会导致学习器有显著变动, 如决策树、神经网络等)

- 输入属性扰动

一般采用随机子空间算法 (random subspace):

输入: 训练集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$;
 基学习算法 \mathcal{L} ;
 基学习器数 T ;
 子空间属性数 d' .

过程:

- 1: **for** $t = 1, 2, \dots, T$ **do**
- 2: $\mathcal{F}_t = \text{RS}(D, d')$
- 3: $D_t = \text{Map}_{\mathcal{F}_t}(D)$
- 4: $h_t = \mathcal{L}(D_t)$
- 5: **end for**

输出: $H(\mathbf{x}) = \arg \max_{y \in \mathcal{Y}} \sum_{t=1}^T \mathbb{I}(h_t(\text{Map}_{\mathcal{F}_t}(\mathbf{x})) = y)$

图 8.11 随机子空间算法

- 输出表示扰动

对输出进行扰动，比如随机改变一些样本的标记、引入纠错码等。

- 算法参数扰动

就是基学习器的调参

6.6 小结

Chapter 7

聚类 (clustering)

7.1 聚类任务

在无监督学习 (unsupervised learning) 中, 训练样本的标记信息是未知的, 目标是通过无标记的训练样本的学习来揭示数据的内在性质及规律, 此类学习任务一般就是研究聚类任务了。

聚类是指将数据集中的样本划分成若干个通常不相交的子集, 每个子集被称为一个簇 (cluster)。

7.2 性能度量

一般来说我们希望进行聚类后‘簇内相似度 (intra-cluster similarity)’高的同时‘簇间相似度 ()’

性能度量一般分为两类:

- 外部指标 (external index): 即将聚类结果与某个外部模型进行比较。
- 内部指标 (internal index): 只考察聚类结果而不利用任何模型。

假设数据集给出 $D = \{x_1, x_2, \dots, x_m\}$ 通过聚类给出的簇划分结果为 $C = \{C_1, C_2, \dots, C_n\}$, 参考模型给出的划分 $C^* = \{C_1^*, C_2^*, \dots, C_n^*\}, \lambda, \lambda^*$

分别表示对应的簇标记向量，将其两两配对：

$$\begin{aligned}
a &= |SS|, SS = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}, \\
b &= |SD|, SD = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}, \\
c &= |DS|, DS = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}, \\
d &= |DD|, DD = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}, \\
a + b + c + d &= \frac{m(m-1)}{2}
\end{aligned} \tag{7.1}$$

- 聚类的性能度量外部指标：

- Jaccard 系数：

$$JC = \frac{a}{a + b + c} \tag{7.2}$$

- FM 指数：

$$FMI = \sqrt{\frac{a}{a+b} \cdot \frac{a}{a+c}} \tag{7.3}$$

- Rand 指数：

$$RI = \frac{2(a+d)}{m(m-1)} \tag{7.4}$$

上述三个指标都在 $[0, 1]$ 区间，值越大越好。

- 聚类的性能度量内部指标：首先定义几个变量：

- $dist(\cdot, \cdot)$ 表示两个样本之间的距离；
- $avg(C)$ 表示簇 C 之间的样本的平均距离。

$$avg(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \leq i < j \leq |C|} dist(x_i, x_j) \tag{7.5}$$

- $diam(C)$ 表示簇 C 内样本的最远距离：

$$diam(C) = \max_{1 \leq i < j \leq |C|} dist(x_i, x_j) \tag{7.6}$$

- $d_{\min}(C_i, C_j)$ ，对应簇 C_i 到 C_j 最近样本间距：

$$d_{\min}(C_i, C_j) = \min_{x_i \in C_i, x_j \in C_j} dist(x_i, x_j) \tag{7.7}$$

- $d_{\text{cen}}(C_i, C_j)$, 表示簇 C_i 到 C_j 中心点的距离。

$$d_{\text{cen}}(C_i, C_j) = \text{dist}(\mu_i, \mu_j) \quad (7.8)$$

其中 $\mu = \frac{1}{|C|} \sum_{1 \leq i \leq |C|}$, 表示簇 C 的中心点

那么我们可以推出聚类的性能度量内部指标:

- DB 指数:

$$DBI = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left(\frac{\text{avg}(C_i) + \text{avg}(C_j)}{d_{\text{cen}}(\mu_i, \mu_j)} \right) \quad (7.9)$$

- Dunn 指数:

$$DI = \min_{1 \leq i \leq k} \left\{ \min_{j \neq i} \left(\frac{d_{\min}(C_i, C_j)}{\max_{1 \leq l \leq k} \text{diam}(C_l)} \right) \right\} \quad (7.10)$$

显然 DBI 越小越好而 DI 越大越好。

7.3 距离计算

函数 $\text{dist}(\cdot, \cdot)$, 记录了两个样本上的距离, 需满足以下的性质:

- 非负性: $\text{dist}(x_i, x_j) \geq 0$;
- 同一性: $\text{dist}(x_i, x_j) = 0$ 当且仅当 $x_i = x_j$;
- 对称性: $\text{dist}(x_i, x_j) = \text{dist}(x_j, x_i)$;
- 直递性: $\text{dist}(x_i, x_j) \leq \text{dist}(x_i, x_k) + \text{dist}(x_k, x_j)$

对于给定样本 $x_i = (x_{i1}; x_{i2}; \dots; x_{in})$ 和 $x_j = (x_{j1}; x_{j2}; \dots; x_{jn})$, 计算两点最常用的距离是‘闵可夫斯基距离 (Minkowski distance, MK)’

$$\text{dist}_{mk}(x_i, x_j) = \left(\sum_{u=1}^n |x_{iu} - x_{ju}|^p \right)^{\frac{1}{p}} \quad (7.11)$$

当 $p = 2$ ，即二维向量时，mk 距离即为欧氏距离 (Euclidean distance):

$$dist_{ed}(x_i, x_j) = \|x_i - x_j\|_{p=2} = \sqrt{\sum_{u=1}^n |x_{iu} - x_{ju}|^2} \quad (7.12)$$

当 $p = 1$ ，即曼哈顿距离 (Manhattan distance):

$$dist_{man}(x_i, x_j) = \|x_i - x_j\|_{p=1} = \sum_{u=1}^n |x_{iu} - x_{ju}| \quad (7.13)$$

mk 距离仅适合如 $\{1, 2, 3\}$ 之类的有序属性, 对于无序属性采用 VDM(Value Difference Metric) 计算距离;

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, 那么属性 u 上的两个离散值 a, b 之间的 VDM 距离为:

$$VDM_p(a, b) = \sum_{i=1}^k \left| \frac{m_{u,a,i}}{m_{u,a}} - \frac{m_{u,b,i}}{m_{u,b}} \right|^p \quad (7.14)$$

结合两种计算距离的算法, 假设有 n_c 个有序属性, $n - n_c$ 个无序属性, 令有序属性排列在无序属性之前, 则:

$$MinkovDM_p(x_i, x_j) = \left(\sum_{u=1}^{n_c} |x_{iu} - x_{ju}|^p + \sum_{u=n_c+1}^n VDM_p(x_{iu}, x_{ju}) \right)^{\frac{1}{p}} \quad (7.15)$$

当样本空间中不同属性的重要性不同, 可使用‘加权距离’。

7.4 原型聚类

聚类的常用算法, 先对原型进行初始化, 然后对原型进行更新迭代来进行求解。

7.4.1 k 均值算法 (k-means)

给定样本集 $D = \{x_1, x_2, \dots, x_m\}$, k-means 针对聚类所得簇划分 $C = \{C_1, C_2, \dots, C_k\}$ 的最小化平方误差为:

$$E = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|_2^2 \quad (7.16)$$

其中 $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$, E 的值越小说明簇内样本相似度越高:

Algorithm 1 k-means 算法

In: 样本集 $D = \{x_1, x_2, \dots, x_m\}$; 聚类簇数 k ;

Out: 簇划分 $C = \{C_1, C_2, \dots, C_k\}$;

```
1: function K-MEANS( $D, k$ )
2:   从  $D$  中随机选择  $k$  个样本作为初始的均值向量  $\{\mu_1, \mu_2, \dots, \mu_k\}$ ;
3:   repeat
4:     令  $C_i = \emptyset (1 \leq i \leq k)$ ;
5:     for  $j = 1, 2, \dots, m$  do
6:       计算样本  $x$  与各均值向量  $\mu_i (1 \leq i \leq k)$  的距离:  $d_{ji} = \|x_j - \mu_i\|_2$ .
7:       根据距离最近的均值向量确定  $x_j$  的簇标记:  $\lambda_j = \arg \min_{i \in \{1, 2, \dots, k\}} d_{ji}$ ;
8:       将样本  $x_j$  划入相应的簇:  $C_{\lambda_j} = C_{\lambda_j} \cup x_j$ ;
9:     end for
10:    for  $i = 1, 2, \dots, k$  do
11:      计算新的均值向量:  $\mu'_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$ ;
12:      if  $\mu'_i \neq \mu_i$  then;
13:        将当前均值向量  $\mu_i$  更新为  $\mu'_i$ ;
14:      else
15:        保持当前均值向量不变;
16:      end if
17:    end for
18:  until 当前的均值向量均未更新
19: end function
```

7.4.2 学习向量量化

跟 k-means 算法类似, 学习向量量化 (Learning Vector Quantization, LVQ) 试图找到一组原型向量来刻画聚类结构, 但是与一般的聚类算法不同的是, LVQ 假设样本带有类别标记, 学习的过程利用样本的监督信息来辅助聚类。算法如下图:

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

过程:

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

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

图 9.4 学习向量量化算法

7.4.3 高斯混合聚类

高斯混合聚类采用的是概率模型来表达聚类模型, 而 k-means 与 LVQ 是采用原型向量来刻画的。

对 n 维样本空间 \mathcal{X} 的随机向量为 x , 若 x 服从高斯分布, 其密度函数:

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \quad (7.17)$$

其中 μ 是 n 维均值向量, Σ 是 $n \times n$ 的协方差矩阵, 将概率密度函数记为 $p(x|\mu, \Sigma)$. 我们可以定义高斯混合分布为:

$$p_{\mathcal{M}}(x) = \sum_{i=1}^k \alpha_i \cdot p(x|\mu_i, \Sigma_i), \quad (7.18)$$

该分布由 k 个混合分布组成, 每个混合分布对应相应的一个高斯分布, 其中 μ_i 与 Σ_i 是第 i 个高斯混合成分的参数。而 $\alpha_i > 0$ 为对应的混合系数 (mixture coefficient)。

那么样本的生成过程可以由高斯分布给出：

1. 根据 $\alpha_1, \alpha_2, \dots, \alpha_k$ 定义的先验分布选择高斯混合成分，
2. 根据被选择的混合成分的概率密度函数进行采样并且生成对应的样本。

若训练集 $D = \{x_1, x_2, \dots, x_m\}$ 由上述过程进行生成，令随机变量 $z_j \in \{1, 2, \dots, k\}$ 表示样本 x_j 的高斯混合成分，那么 z_j 的后验分布为：

$$p_{\mathcal{M}}(z_j = i | x_j) = \frac{\alpha_i \cdot p(x_j | \mu_i, \Sigma_i)}{\sum_{l=1}^k \alpha_l \cdot p(x_j | \mu_l, \Sigma_l)} \quad (7.19)$$

将 $p_{\mathcal{M}}(z_j = i | x_j)$ 简记为 $\gamma_{ji} (i = 1, 2, \dots, k)$ ，当高斯混合分布已知时，高斯混合聚类将样本集 D 划分为 k 个簇 $C = \{C_1, C_2, \dots, C_k\}$ 每个样本 x_j 的簇标记记为 λ_j 如下确定：

$$\lambda_j = \arg \max_{i \in \{1, 2, \dots, k\}} \gamma_{ji} \quad (7.20)$$

对于模型参数 $\{(\alpha_i, \mu_i, \Sigma_i) | 1 \leq i \leq k\}$ 可以采用极大似然估计，最大化对数似然：

$$LL(D) = \ln \left(\prod_{j=1}^m p_{\mathcal{M}}(x_j) \right) = \sum_{j=1}^m \ln \left(\sum_{i=1}^k \alpha_i \cdot p(x_j | \mu_i, \Sigma_i) \right) \quad (7.21)$$

令 $\frac{\partial LL(D)}{\partial \mu_i} = 0$ ，有：

$$\sum_{j=1}^m \frac{\alpha_i \cdot p(x_j | \mu_i, \Sigma_i)}{\sum_{l=1}^k \alpha_l \cdot p(x_j | \mu_l, \Sigma_l)} (x_j - \mu_i) = 0 \quad (7.22)$$

由 7.19 及 $\gamma_{ji} = p_{\mathcal{M}}(z_j = i | x_j)$ ，有：

$$\mu_i = \frac{\sum_{j=1}^m \gamma_{ji} x_j}{\sum_{j=1}^m \gamma_{ji}} \quad (7.23)$$

令： $\frac{\partial LL(D)}{\partial \Sigma_i} = 0$ 得：

$$\sum_i = \frac{\sum_{j=1}^m \gamma_{ji} (x_j - \mu_i)(x_j - \mu_i)^T}{\sum_{j=1}^m \gamma_{ji}} \quad (7.24)$$

对于混合系数 α_i ，考虑 $LL(D)$ 的拉格朗日形式：

$$LL(D) + \lambda \left(\sum_{i=1}^k \alpha_i - 1 \right) \quad (7.25)$$

其中 λ 为拉格朗日乘子，取上式对 α_i 的偏导为 0，有：

$$\sum_{j=1}^m \frac{p(x_j | \mu_i, \Sigma_i)}{\sum_{l=1}^k \alpha_l \cdot p(x_j | \mu_l, \Sigma_l)} + \lambda = 0 \quad (7.26)$$

两边同时乘 α_i 并对所有样本进行求和可知 $\lambda = -m$ ，有：

$$\alpha_i = \frac{1}{m} \sum_{j=1}^m \gamma_{ji} \quad (7.27)$$

综上可以推导出高斯混合模型的 EM 算法，算法流程如下图所示：

- E 步：根据当前参数来计算每个样本属于每个高斯成分的后验概率 γ_{ji}
- M 步：更新模型参数 $\{(\alpha_i, \mu_i, \Sigma_i) | 1 \leq i \leq k\}$

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

过程:

- 1: 初始化高斯混合分布的模型参数 $\{(\alpha_i, \mu_i, \Sigma_i) \mid 1 \leq i \leq k\}$
- 2: **repeat**
- 3: **for** $j = 1, 2, \dots, m$ **do**
- 4: 根据式(9.30)计算 \mathbf{x}_j 由各混合成分生成的后验概率, 即
 $\gamma_{ji} = p_{\mathcal{M}}(z_j = i \mid \mathbf{x}_j) \ (1 \leq i \leq k)$
- 5: **end for**
- 6: **for** $i = 1, 2, \dots, k$ **do**
- 7: 计算新均值向量: $\mu'_i = \frac{\sum_{j=1}^m \gamma_{ji} \mathbf{x}_j}{\sum_{j=1}^m \gamma_{ji}}$;
- 8: 计算新协方差矩阵: $\Sigma'_i = \frac{\sum_{j=1}^m \gamma_{ji} (\mathbf{x}_j - \mu'_i)(\mathbf{x}_j - \mu'_i)^T}{\sum_{j=1}^m \gamma_{ji}}$;
- 9: 计算新混合系数: $\alpha'_i = \frac{\sum_{j=1}^m \gamma_{ji}}{m}$;
- 10: **end for**
- 11: 将模型参数 $\{(\alpha_i, \mu_i, \Sigma_i) \mid 1 \leq i \leq k\}$ 更新为 $\{(\alpha'_i, \mu'_i, \Sigma'_i) \mid 1 \leq i \leq k\}$
- 12: **until** 满足停止条件
- 13: $C_i = \emptyset \ (1 \leq i \leq k)$
- 14: **for** $j = 1, 2, \dots, m$ **do**
- 15: 根据式(9.31)确定 \mathbf{x}_j 的簇标记 λ_j ;
- 16: 将 \mathbf{x}_j 划入相应的簇: $C_{\lambda_j} = C_{\lambda_j} \cup \{\mathbf{x}_j\}$
- 17: **end for**

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

图 9.6 高斯混合聚类算法

7.5 密度聚类 (density-based clustering)

密度聚类假定聚类结构能够通过样本分布的紧密程度进行确定。

DBSCAN 是一种著名的密度聚类算法, 它基于一组领域参数 ϵ 来刻画样本分布的紧密程度, 首先定义几个概念, 给定数据集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$:

- ϵ -邻域: 对 $\mathbf{x}_j \in D$, 其 ϵ -邻域包含样本集 D 中与 \mathbf{x}_j 距离不大于 ϵ 的样本, 即 $N_\epsilon(\mathbf{x}_j) = \{\mathbf{x}_i \in D \mid \text{dist}(\mathbf{x}_i, \mathbf{x}_j) \leq \epsilon\}$;
- 核心对象 (core object): 若 \mathbf{x}_j 的 ϵ -邻域至少包含 $MinPts$ 个样本。即 $|N_\epsilon(\mathbf{x}_j)| \geq MinPts$, 则 \mathbf{x}_j 是一个核心对象。

- 密度直达 (directly density-reachable): 若 x_j 位于 x_i 的 ϵ -邻域中, 且 x_i 是核心对象, 则称 x_j 由 x_i 密度直达;
- 密度可达 (density-reachable): 对 x_i 与 x_j , 若存在样本序列 p_1, p_2, \dots, p_n , 其中 $p_1 = x_i, p_n = x_j$ 且 p_{i+1} 由 p_i 密度直达, 则称 x_j 由 x_i 密度可达;
- 密度相连 (density-connected): 对于 x_i 与 x_j , 若存在 x_k 使得 x_i 与 x_j 均有 x_k 密度可达, 那么称为 x_i 与 x_j 密度相连。

如下图所示:

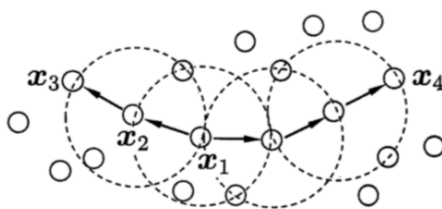


图 9.8 DBSCAN 定义的基本概念($MinPts = 3$): 虚线显示出 ϵ -邻域, x_1 是核心对象, x_2 由 x_1 密度直达, x_3 由 x_1 密度可达, x_3 与 x_4 密度相连。

基于如上的概念, DBSCAN 将簇定义为: 由密度可达关系导出的最大密度相连的样本集合。形式化来说, 给定邻域参数 $(\epsilon, MinPts)$, 簇 $C \subseteq D$ 满足以下两个性质的非空子集:

1. 连续性 (connextivity): $x_i \in C, x_j \in C \Rightarrow x_i$ 与 x_j 密度相连
2. 最大性 (maxmality): $x_i \in C, x_j$ 由 x_i 密度可达 $\Rightarrow x_j \in C$

算法流程如下图进行:

输入: 样本集 $D = \{x_1, x_2, \dots, x_m\}$;
邻域参数 $(\epsilon, MinPts)$.

过程:

- 1: 初始化核心对象集合: $\Omega = \emptyset$
- 2: **for** $j = 1, 2, \dots, m$ **do**
- 3: 确定样本 x_j 的 ϵ -邻域 $N_\epsilon(x_j)$;
- 4: **if** $|N_\epsilon(x_j)| \geq MinPts$ **then**
- 5: 将样本 x_j 加入核心对象集合: $\Omega = \Omega \cup \{x_j\}$
- 6: **end if**
- 7: **end for**
- 8: 初始化聚类簇数: $k = 0$
- 9: 初始化未访问样本集合: $\Gamma = D$
- 10: **while** $\Omega \neq \emptyset$ **do**
- 11: 记录当前未访问样本集合: $\Gamma_{old} = \Gamma$;
- 12: 随机选取一个核心对象 $o \in \Omega$, 初始化队列 $Q = \langle o \rangle$;
- 13: $\Gamma = \Gamma \setminus \{o\}$;
- 14: **while** $Q \neq \emptyset$ **do**
- 15: 取出队列 Q 中的首个样本 q ;
- 16: **if** $|N_\epsilon(q)| \geq MinPts$ **then**
- 17: 令 $\Delta = N_\epsilon(q) \cap \Gamma$;
- 18: 将 Δ 中的样本加入队列 Q ;
- 19: $\Gamma = \Gamma \setminus \Delta$;
- 20: **end if**
- 21: **end while**
- 22: $k = k + 1$, 生成聚类簇 $C_k = \Gamma_{old} \setminus \Gamma$;
- 23: $\Omega = \Omega \setminus C_k$
- 24: **end while**

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

图 9.9 DBSCAN 算法

7.6 层次聚类 (hierarchical clustering)

层次聚类试图在不同层次上对数据集进行划分, 从而形成树形的聚类结构。

AGNES 是一种采用自底向上聚合策略的层次聚类算法。他先将数据集中的每一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最近的两个聚类簇进行合并，直到聚类簇个数减小到阈值。定义下面三种聚类簇之间的距离：

- 最小距离： $d_{\min}(C_i, C_j) = \min_{x \in C_i, z \in C_j} dist(x, z)$
- 最大距离： $d_{\max}(C_i, C_j) = \max_{x \in C_i, z \in C_j} dist(x, z)$
- 平均距离： $d_{avg}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{z \in C_j} dist(x, z)$

综上 AGNES 算法可以定义出如下流程图：

输入: 样本集 $D = \{x_1, x_2, \dots, x_m\}$;
 聚类簇距离度量函数 d ;
 聚类簇数 k .

过程:

```

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

```

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

图 9.11 AGNES 算法

7.7 小结

Chapter 8

降维与度量学习

8.1 k 近邻学习

k 近邻学习 (k -Nearest Neighbor, k NN), 是一种常用的监督学习方法, 其原理很简单, 就是看测试集中离输入样本 x 中最近的 k 个样本中分类最多的一个, 将其结果作为 x 的分类结果进行返回。

k 近邻算法的是懒惰学习 (lazy learning) 的代表, 此类算法的最大特点就是在训练阶段仅仅将样本保存起来, 训练时间的开销为 0, 在收到测试样本之后再进行相应的处理, 那么那些在训练阶段就对样本进行学习的处理方法称为”急切学习”。

8.2 低维嵌入

高维空间会给计算距离造成大量的麻烦, 我们将其称为维数灾难。

缓解维数灾难的一个重要途径就是进行降维 (dimension reduction), 若要求原始空间中样本的距离在低维空间中继续得以维持, 即”多维缩放”, (Multiple Dimensional Scaling, MDS):

给定 m 个样本在原始的 d 维空间的距离矩阵为 $D \in \mathbb{R}^{m \times m}, dist_{ij}$ 指样本 x_i 和 x_j 的距离。我们的目标是要获得样本在 d' 维空间的表示 $Z \in \mathbb{R}^{d' \times m}, d' \leq d$, 且任意两个样本 x_i 和 x_j 在 d 维空间和 d' 维空间之间的距

离相同。

令 $B = Z^T Z \in \mathbb{R}^{m \times m}$ 表示降维后的样本内积矩阵, $b_{ij} = z_i^T z_j$ 有,

$$dist_{ij}^2 = \|z_i\|^2 + \|z_j\|^2 - 2z_i^T z_j = b_{ii} + b_{jj} - 2b_{ij} \quad (8.1)$$

令

$$\begin{aligned} dist_{i.}^2 &= \frac{1}{m} \sum_{j=1}^m dist_{ij}^2 \\ dist_{.j}^2 &= \frac{1}{m} \sum_{i=1}^m dist_{ij}^2 \\ dist_{..}^2 &= \frac{1}{m^2} \sum_{i=1}^m \sum_{j=1}^m dist_{ij}^2 \end{aligned} \quad (8.2)$$

综上可得:

$$b_{ij} = -\frac{1}{2}(dist_{ij}^2 - dist_{i.}^2 - dist_{.j}^2 + dist_{..}^2) \quad (8.3)$$

上述公式说明可以通过降维前后保持不变的距离矩阵 D 求内积矩阵 B . 对矩阵 B 进行特征值分解, 有 $b = V\Lambda V^T$, 其中 $\Lambda = diag(\lambda_1, \lambda_2, \dots, \lambda_d)$ 特征值构成的对角矩阵。且满足

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d \quad (8.4)$$

V 为特征向量矩阵, 假定其中有 d^* 个非零特征值, 它们构成对角矩阵 $\Lambda = diag(\lambda_1, \lambda_2, \dots, \lambda_{d^*})$, 令 V_* 表示对应的特征向量矩阵, 那么 Z 可以表达为:

$$Z = \Lambda_*^{\frac{1}{2}} V_*^T \in \mathbb{R}^{d^* \times m} \quad (8.5)$$

有时候为了实现有效的降维, 降维后的距离只需尽可能地接近原始空间的距离。此时去 $d' \ll d$ 个最大特征值构成的对角矩阵 $\tilde{\Lambda} = diag(\lambda_1, \lambda_2, \dots, \lambda_{d'})$, 令 \tilde{V} 表示相应的特征向量矩阵, 那么将 Z 表示为:

$$Z = \tilde{\Lambda}^{\frac{1}{2}} \tilde{V}^T \in \mathbb{R}^{d' \times m} \quad (8.6)$$

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

过程:

- 1: 根据式(10.7)~(10.9)计算 $dist_{i.}^2, dist_{.j}^2, dist_{..}^2$;
- 2: 根据式(10.10)计算矩阵 \mathbf{B} ;
- 3: 对矩阵 \mathbf{B} 做特征值分解;
- 4: 取 $\tilde{\mathbf{\Lambda}}$ 为 d' 个最大特征值所构成的对角矩阵, $\tilde{\mathbf{V}}$ 为相应的特征向量矩阵.

输出: 矩阵 $\tilde{\mathbf{V}}\tilde{\mathbf{\Lambda}}^{1/2} \in \mathbb{R}^{m \times d'}$, 每行是一个样本的低维坐标

图 10.3 MDS 算法

一般来说, 想要获得低维的子空间, 最简单的方式就是对原始的高维空间进行线性变换。给定 d 维空间中的样本 $X = (x_1, x_2, \dots, x_m) \in \mathbb{R}^{d \times m}$, 那么变化后的样本为:

$$Z = W^T X \quad (8.7)$$

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

给予线性变化来进行降维的方法称为线性降维方法。

8.3 主成分分析 (Principal Component Analysis, PCA)

PCA 是一种常用的降维方法, 我们设想存在一个超平面, 对于所有的样本:

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

我们先从最大重构性进行推导。

假定数据样本已经进行了中心化, 即 $\sum_i x_i = 0$; 再假定投影变化后的新坐标系为 $\{w_1, w_2, \dots, w_d\}$, 其中 $\|w_i\|_2 = 1, w_i^T w_j = 0 (i \neq j)$, 若丢弃新坐标系中的部分坐标, 即将维度降到 $d' < d$, 那么 x_i 在低维坐标系中的投影是 $z_i = (z_{i1}; z_{i2}; \dots; z_{id'})$, 其中 $z_{ij} = w_j^T x_i$ 指的是 x_i 在低维坐标系下第 j 维的坐标, 若基于 z_i 来重构 x_i , 则有 $\hat{x}_i = \sum_{j=1}^{d'} z_{ij} w_j$. 那么, 原样本点

x_i 与基于投影重构的样本点 \hat{x}_i 的距离为：

$$\begin{aligned} \sum_{i=1}^m \left\| \sum_{j=1}^{d'} z_{ij} w_j - x_i \right\|_2^2 = \\ \sum_{i=1}^m z_i^T z_i - 2 \sum_{i=1}^m z_i^T W^T x_i + \text{const} \propto -\text{tr} \left(W^T \left(\sum_{i=1}^m x_i x_i^T \right) W \right) \end{aligned} \quad (8.8)$$

根据最近重构性，上式应该要被最小化，考虑到 w_j 是标准正交基，而 $\sum_i x_i x_i^T$ 是协方差矩阵，有：

$$\min_W -\text{tr}(W^T X X^T W) \quad \text{s.t. } W^T W = I \quad (8.9)$$

上面这个式子就是主成分分析的主要优化目标。为了使得样本点的投影能够尽可能的分开，则应该使得投影后的样本点的方差能够最大化。假设样本投影后的方差是 $\sum_i W^T x_i x_i^T W$ ，于是优化目标可以写为

$$\max_W \text{tr}(W^T X X^T W) \quad \text{s.t. } W^T W = I \quad (8.10)$$

对上式采用拉格朗日乘子法可得：

$$X X^T W = \lambda W \quad (8.11)$$

只需对协方差矩阵 $X X^T$ 进行特征值分解，再将求得的特征值进行排序： $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$ ，再取前 d' 个特征值对应的特征向量构成 $W = (w_1, w_2, \dots, w_{d'})$ ，即是主成分分析的解了。算法流程图可以描述如下：

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

过程：

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

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

图 10.5 PCA 算法

8.4 核化线性降维

对应的是将高维空间经过非线性的映射到低维空间。

借鉴前面 SVM 的核函数方法，我们可以将转化矩阵进行核化，假设我们将在高维特征空间中把数据投影到由 W 确定的超平面上，即欲求解：

$$\left(\sum_{i=1}^m z_i z_i^T \right) = \lambda W \quad (8.12)$$

求出：

$$W = \frac{1}{\lambda} \left(\sum_{i=1}^m z_i z_i^T \right) = \sum_{i=1}^m z_i \frac{z_i^T W}{\lambda} = \sum_{i=1}^m z_i \alpha_i \quad (8.13)$$

其中 z_i 是样本点 x_i 在高维特征空间中的像，是由 x_i 通过映射 ϕ 映射产生的，那么可以改写 8.12 为核函数公式：

$$\left(\sum_{i=1}^m \phi(x_i) \phi(x_i)^T \right) W = \lambda W \quad (8.14)$$

再将 8.13 改写成：

$$W = \sum_{i=1}^m \phi(x_i) \alpha_i \quad (8.15)$$

核函数可以写为：

$$\kappa(x_i, x_j) = \phi(x_i)^T \phi(x_j) \quad (8.16)$$

综上：

$$\mathcal{K} A = \lambda A \quad (8.17)$$

其中 \mathcal{K} 是 κ 对应的核矩阵，显然上式是一个特征值分解问题，只需取 \mathcal{K} 的最大的 d' 个特征值对应的特征向量即可。

那么对于一个新的样本 x ，其投影之后的第 j ($j = 1, 2, \dots, d'$) 维坐标为：

$$z_j = \sum_{i=1}^m \alpha_i^j \kappa(x_i, x) \quad (8.18)$$

上述的公式说明，为了获得投影后的坐标，KPCA 需要对所有的样本进行求和，具有很大的开销。

8.5 流形学习 (manifold learning)

流形学习是一类借鉴了拓扑流形结构的降维方法，它在局部具有欧氏空间的性质，能够用欧式距离进行计算。若能够将低维流形嵌入高维空间中，就可以让高维空间的局部具有欧氏空间的性质，容易建立局部的降维映射，然后再将局部推广到全局中。

8.5.1 等度量映射 (Isometric Mapping, Isomap)

其思想是对于每个点基于欧式距离找出其邻近点，建立一个近邻连接图，那么计算问题就将转化为计算近邻连接图上两点的最短路径问题了。算法流程如下：

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

过程：

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

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

图 10.8 Isomap 算法

我们可以将训练样本的高维空间坐标作为输入，低维的空间坐标作为输出，训练一个回归学习器来对新样本进行一个预测，这是目前比较常用的方法。

对于如何构建近邻图，通常有两种办法：

- 指定近邻点的个数 k ，以最近的 k 个点作为近邻点，这样的图是 k 近邻图。
- 另一种是指定阈值 ϵ ，所有小于 ϵ 的点都会被认为是近邻点，这样的图被称为 ϵ 近邻图。

两种办法都存在一些问题，如果范围确定得太大，那么会把距离较远得点也算进来，造成”短路”，如果范围确定得太小，则会造成”断路”，从而损失掉一定的信息。

8.5.2 局部线性嵌入 (Locally Linear Embedding, LLE)

LLE 试图保存邻域内样本之间的线性关系，假设样本点 x_i 得坐标能够通过它的邻域样本 x_j, x_k 的坐标线性组合而重构得来的，即：

$$x_i = w_{ij}x_j + w_{ik}x_k + w_{il}x_l \quad (8.19)$$

LLE 希望在低维的空间中仍然能够保持这个趋势。首先对于每个样本 x_i 先找到它的邻近下标集合 Q_i ，然后计算出基于 Q_i 中的样本点对于 x_i 进行线性重构的系数 w_i ；

$$\min_{w_1, w_2, \dots, w_m} \sum_{i=1}^m \|x_i - \sum_{j \in Q_i} w_{ij}x_j\|_2^2 \quad s.t. \quad \sum_{j \in Q_i} w_{ij} = 1 \quad (8.20)$$

令 $C_{jk} = (x_i - x_j)^T(x_i - x_k)$ ，那么可以解出：

$$w_{ij} = \frac{\sum_{k \in Q_i} C_{jk}^{-1}}{\sum_{l, s \in Q_i} C_{ls}^{-1}} \quad (8.21)$$

那么 x_i 对应的低维空间的坐标 z_i 可以通过下面的式子进行求解：

$$\min_{z_1, z_2, \dots, z_m} \sum_{i=1}^m \|z_i - \sum_{j \in Q_i} w_{ij}z_j\|_2^2 \quad (8.22)$$

再令 $Z = (z_1, z_2, \dots, z_m) \in \mathbb{R}^{d' \times m}$, $(W)_{ij} = w_{ij}$ ，有：

$$M = (I - W)^T(I - W) \quad (8.23)$$

重写 8.22:

$$\min_Z \text{tr}(ZMZ^T) \quad s.t. \quad ZZ^T = I \quad (8.24)$$

其中 M 最小的 d' 个特征值对应的特征向量组成了矩阵 Z^T 。其算法如下图所示：

输入: 样本集 $D = \{x_1, x_2, \dots, x_m\}$;
 近邻参数 k ;
 低维空间维数 d' .

过程:

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

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

图 10.10 LLE 算法

8.6 度量学习

前面的算法的目的是找到一个适合的低维空间来尽可能的还原高维空间, 度量学习是学习如何制造一个距离度量来直接衡量降维的样本:

对于两个 d 维的样本 x_i 和 x_j , 它们之间的平方欧式距离可以写为:

$$\begin{aligned}
 dist_{wed}^2(x_i, x_j) &= \|x_i - x_j\|_2^2 \\
 &= \sum_{k=1}^d w_k dist_{ij,k}^2 \\
 &= (x_i - x_j)^T W (x_i - x_j)
 \end{aligned} \tag{8.25}$$

其中 $dist_{ij,k}$ 指的是 x_i 和 x_j 在第 k 维的距离, w_k 指的是距离的权重。 W 是权重矩阵, $(W)_{ii} = w_i$.

W 所有的非对角元素都是 0, 说明 W 对应的属性都是正交不相关的, 这在实际的应用中是不合理的, 对此, 我们将 W 替换成一个普通的半正定

对称矩阵 M , 有正交基 P 使得 $M = PP^T$:

$$dist_{mah}^2(x_i, x_j) = (x_i - x_j)^T M (x_i - x_j) = \|x_i - x_j\|_M^2 \quad (8.26)$$

上述式子就是马氏距离 (Mahalanobis distance)。度量学习就是对 M 进行一系列的学习。常用的方法称为近邻成分分析 (Neighbourhood Component Analysis, NCA), NCA 在进行判别时通常采用多数投票法, 若替换成概率投票法, 对于任意样本, 有 x_j , 它对 x_i 分类影响的概率为:

$$p_{ij} = \frac{\exp(-\|x_i - x_j\|_M^2)}{\sum_l \exp(-\|x_i - x_l\|_M^2)} \quad (8.27)$$

计算 x_i 的留一法正确率, 即它被自身之外的所有样本正确分类的概率为:

$$p_i = \sum_{j \in \Omega_i} p_{ij} \quad (8.28)$$

其中 Ω_i 表示与 x_i 属于相同类别的样本的下标集合, 故整个样本采用留一法的正确率可以表示为:

$$\sum_{i=1}^m p_i = \sum_{i=1}^m \sum_{j \in \Omega_i} p_{ij} \quad (8.29)$$

综上, 我们可以得出 NCA 的优化目标了:

$$\min_P \quad 1 - \sum_{i=1}^m \sum_{j \in \Omega_i} \frac{\exp(-\|P^T x_i - P^T x_j\|_2^2)}{\sum_l \exp(-\|P^T x_i - P^T x_l\|_2^2)} \quad (8.30)$$

8.7 小结

Chapter 9

计算学习理论 (Computational Learning Theory)

9.1 基础知识

计算学习理论是通过计算来进行学习的理论，即有关于机器学习的理论基础。

给定样例集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$, $x_i \in \mathcal{X}$ 。假设所有的样本都是独立同分布的样本，并且服从一个隐含未知的分布 \mathcal{D} 令 h 为 \mathcal{X} 到 \mathcal{Y} 的一个映射，

其泛化误差为：

$$E(h; \mathcal{D}) = P_{x \sim \mathcal{D}}(h(x) \neq y) \quad (9.1)$$

其经验误差为：

$$\hat{E}(h; D) = \frac{1}{m} \sum_{i=1}^m \mathbb{I}(h(x_i) \neq y_i) \quad (9.2)$$

由于 D 是 \mathcal{D} 的独立同分布采样，那么 h 的经验误差的期望等于其泛化的误差，令 ϵ 为 $E(h)$ 的上限，即 $E(h) \leq \epsilon$ ，我们将 ϵ 称为误差参数。

对于两个映射 $h_1, h_2 \in \mathcal{X} \rightarrow \mathcal{Y}$ 来度量他们之间的差别：

$$d(h_1, h_2) = P_{x \sim \mathcal{D}}(h_1(x) \neq h_2(x)) \quad (9.3)$$

下面将会介绍一些常用的不等式：

- Jensen 不等式：对于任意的凸函数 $f(x)$ 有：

$$f(\mathbb{E}(x)) \leq \mathbb{E}(f(x)) \quad (9.4)$$

- Hoeffding 不等式：若 x_1, x_2, \dots, x_m 为 m 个独立随机变量，且满足 $0 \leq x_i \leq 1$ ，那么对于任意 $\epsilon > 0$ 有：

$$\begin{aligned} P\left(\frac{1}{m} \sum_{i=1}^m x_i - \frac{1}{m} \sum_{i=1}^m \mathbb{E}(x_i) \geq \epsilon\right) &\leq \exp(-2m\epsilon^2), \\ P\left(\left|\frac{1}{m} \sum_{i=1}^m x_i - \frac{1}{m} \sum_{i=1}^m \mathbb{E}(x_i)\right| \geq \epsilon\right) &\leq 2 \exp(-2m\epsilon^2) \end{aligned} \quad (9.5)$$

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

$$\sup_{x_1, \dots, x_m, x'_i} |f(x_1, \dots, x_m) - f(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_m)| \leq c_i, \quad (9.6)$$

对于任意 $\epsilon > 0$ ，有：

$$\begin{aligned} P(f(x_1, \dots, x_m) - \mathbb{E}(f(x_1, \dots, x_m)) \geq \epsilon) &\leq \exp\left(\frac{-2\epsilon^2}{\sum_i c_i^2}\right) \\ P(|f(x_1, \dots, x_m) - \mathbb{E}(f(x_1, \dots, x_m))| \geq \epsilon) &\leq 2 \exp\left(\frac{-2\epsilon^2}{\sum_i c_i^2}\right) \end{aligned} \quad (9.7)$$

9.2 PAC 学习

计算学习理论中最基本的就是概率近似正确 (Probably Approximately Correct, PAC)，令 c 表示概念 (concept)，这样就完成了样本空间 \mathcal{X} 到标记空间 \mathcal{Y} 的映射，它决定了样本示例 x 的真实标记 y ，若对于任何样例，均有 $c(x) = y$ 成立，则称 c 为目标概念；我们用 \mathcal{C} 表示概念类。

给定了学习算法 \mathcal{L} ，它所考虑的所有可能概念的集合就被称为假设空间 (hypothesis space)，我们可以用符号 \mathcal{H} 进行表示，学习算法一般会把自认

为可能的目标概念集中起来构成 \mathcal{H} ，对于 $h \in \mathcal{H}$ ，由于无法确定他是否是真的是目标概念，因此可以将其称为假设。

给定训练集 D ，我们希望学习算法 \mathcal{L} 所得到的模型所对应的假设 h 应该尽可能接近目标概念 c ，我们希望以比较大的把握学得比较好的模型，也就是说，以较大的概率学得误差满足预设上限的模型；形象地说，令 δ 表示置信度，定义：

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

$$P(E(h) \leq \epsilon) \geq 1 - \delta \quad (9.8)$$

那么称学习算法 \mathcal{L} 能从假设空间 \mathcal{H} 中辨识概念类 \mathcal{C} 。

这样的学习算法 \mathcal{L} 能以较大的概率 (至少 $1 - \delta$) 学得目标概念 c 的近似 (误差至多 ϵ)。为此我们定义：

PAC 可学习 (PAC Learnable): 令 m 表示从分布 \mathcal{D} 中独立同分布采样得到的样例数目， $0 < \epsilon, \delta < 1$ ，对于所有分布 \mathcal{D} ，若存在学习算法 \mathcal{L} 和多项式函数 $poly(\cdot, \cdot, \cdot, \cdot)$ ，使得对于任何 $m \geq poly(\frac{1}{\epsilon}, \frac{1}{\delta}, size(x), size(c))$ ，若 \mathcal{L} 能够从假设空间 \mathcal{H} 中 PAC 辨识概念类 \mathcal{C} ，则称为概念类 \mathcal{C} 是可 PAC 可学习的。

PAC 学习算法 (PAC Learning Algorithm): 若学习算法 \mathcal{L} 使得概念类 \mathcal{C} 为 PAC 可学习的，且 \mathcal{L} 的运行时间也是多项式函数 $poly(\frac{1}{\epsilon}, \frac{1}{\delta}, size(x), size(c))$ ，则称概念类 \mathcal{C} 是高效 PAC 可学习的。

样本复杂度 (Sample Complexity): 满足 PAC 学习算法 \mathcal{L} 所需的 $m \geq poly(\frac{1}{\epsilon}, \frac{1}{\delta}, size(x), size(c))$ 中的最小 m ，那我们将其称为学习算法 \mathcal{L} 的样本复杂度。

在 PAC 学习中一个关键因素是假设空间 \mathcal{H} 的复杂度，一般而言， \mathcal{H} 越大，其包含任意目标概念的可能性将会越大，但是从其中找到某个具体目标概念的难度也相应增大了。 $|\mathcal{H}|$ 有限时，我们称 \mathcal{H} 为“有限假设空间”，否则称为“无限假设空间”。

9.3 有限假设空间

9.3.1 可分情形

可分情形意味着目标概念 c 属于假设空间 \mathcal{H} , 即 $c \in \mathcal{H}$; 容易想到一种简单的学习策略, 对于任何训练集 D 上出现标记错误的假设肯定不是目标概念 c , 于是我们只需保留与 D 一致的假设, 剔除不一致的假设, 直到 \mathcal{H} 中仅剩下一个假设为止。但是 \mathcal{H} 中可能存在不止一个与 D 一致的等效假设, 这样的话上面的算法就失效了。

对于上述情况, PAC 学习只要训练集 D 的规模能使学习算法 \mathcal{L} 以概率 $1 - \delta$ 找到目标假设 ϵ 即可。

假定 h 的泛化误差大于 ϵ , 对于分布 \mathcal{D} 上随机采样得到的任何样例 (x, y) 都有:

$$P(h(x) = y) = 1 - P(h(x) \neq y) = 1 - E(h) < 1 - \epsilon \quad (9.9)$$

那么 h 于 D 表现一致的概率为:

$$P((h(x_1) = y_1) \wedge \cdots \wedge (h(x_m) = y_m)) = (1 - P(h(x) \neq y))^m < (1 - \epsilon)^m \quad (9.10)$$

由于我们事先不知道学习算法 \mathcal{L} 会输出 \mathcal{H} 的那个假设, 所以我们只需保证泛化误差大于 ϵ , 且训练集上表现完美的所有假设出现概率之和不大于 δ 即可:

$$P(h \in \mathcal{H} : E(h) > \epsilon \wedge \hat{E}(h) = 0) < |\mathcal{H}|(1 - \epsilon)^m < |\mathcal{H}|e^{-m\epsilon} \quad (9.11)$$

我们令 9.11, 不大于 δ : 即

$$|\mathcal{H}|e^{-m\epsilon} \leq \delta \quad (9.12)$$

可得:

$$m \geq \frac{1}{\epsilon} (\ln |\mathcal{H}| + \ln \frac{1}{\delta}) \quad (9.13)$$

上面的公式说明, 输出假设 h 的泛化误差随着样本数目的增多而收敛到 0, 其收敛速度为 $O(\frac{1}{m})$.

9.3.2 不可分情形

对于不可分的清醒，目标概率 c 往往不存在于假设空间 \mathcal{H} 中，假定 \mathcal{H} 中的任意一个假设都会在训练集上或多或少的错误，由 Hoeffding 不等式可知：

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

$$\begin{aligned} P(\hat{E}(h) - E(h) \geq \epsilon) &\leq \exp(-2m\epsilon^2) \\ P(E(h) - \hat{E}(h) \geq \epsilon) &\leq \exp(-2m\epsilon^2) \\ P(|E(h) - \hat{E}(h)| \geq \epsilon) &\leq 2\exp(-2m\epsilon^2) \end{aligned} \quad (9.14)$$

- 若训练集 D 包含 m 个从分布 \mathcal{D} 上独立同分布采样而得的样例， $0 < \epsilon < 1$ ，则对任意 $h \in \mathcal{H}$ ，下面的公式至少以 $1 - \delta$ 的概率成立。

$$\hat{E}(h) - \sqrt{\frac{\ln(2/\delta)}{2m}} \leq E(h) \leq \hat{E}(h) + \sqrt{\frac{\ln(2/\delta)}{2m}} \quad (9.15)$$

- 对于有限假设空间，且 $0 < \epsilon < 1$ ，则对任意 $h \in \mathcal{H}$ ，有：

$$P\left(|E(h) - \hat{E}(h)| \leq \sqrt{\frac{\ln|\mathcal{H}| + \ln(2/\delta)}{2m}}\right) \geq 1 - \delta \quad (9.16)$$

显然，当 $c \notin \mathcal{H}$ ，学习算法 \mathcal{L} 是无法学得目标概念 c 的 ϵ 近似的；但是，若假设空间 \mathcal{H} 给定时，其中必然存在一个泛化误差最小的假设，我们可以选择这个假设即可。这种方法被称为“不可知学习 (agnostic learning)”

不可知 PAC 可学习 (agnostic PAC learnable): 令 m 表示从分布 \mathcal{D} 中独立同分布采样得到的样例数目， $0 < \epsilon, \delta < 1$ ，对所有的分布 (D) ，若存在学习算法 \mathcal{L} 和多项式函数 $\text{poly}(\cdot, \cdot, \cdot, \cdot)$ ，使得对于任意 $m \geq \text{poly}(\frac{1}{\epsilon}, \frac{1}{\delta}, \text{size}(x), \text{size}(c))$ ，满足如下公式，则称假设空间 \mathcal{H} 是不可知 PAC 可学习的。

9.4 VC 维

现实的学习任务通常都是面临无限的假设空间，这时我们想度量假设空间的复杂度，最常见的方法就是考虑假设空间的 VC 维 (VapnikChervonenkis dimension)，给定假设空间 \mathcal{H} 和示例集 $D = \{x_1, x_2, \dots, x_m\}$ ，则 \mathcal{H} 中每个假设 h 都能对 D 中示例赋予标记，标记结果可以表示为：

$$h|_D = \{(h(x_1), h(x_2), \dots, h(x_m))\} \quad (9.17)$$

我们引入几个概念：

- 增长函数 (growth function): 对于所有的 $m \in N$ ，假设空间 \mathcal{H} 的增长函数 $\Pi_{\mathcal{H}}(m)$ 为：

$$\Pi_{\mathcal{H}}(m) = \max_{\{x_1, \dots, x_m\} \subseteq \mathcal{X}} |\{(h(x_1), \dots, h(x_m)) | h \in \mathcal{H}\}| \quad (9.18)$$

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) 和打散 (shattering): 增长函数可以用来估计经验误差与泛化误差之间的关系：对假设空间 \mathcal{H} ， $m \in N, 0 < \epsilon < 1$ 和任意 $h \in \mathcal{H}$ 有：

$$P(|E(h) - \hat{E}(h)| > \epsilon) \leq 4\Pi_{\mathcal{H}}(2m) \exp(-\frac{m\epsilon^2}{8}) \quad (9.19)$$

对于 m 个示例，最多有 2^m 个可能的结果。对于二分类问题