

# 机器学习笔记

陈鸿峥

2020.07 \*

## 目录

<b>1</b>	<b>简介</b>	<b>3</b>
1.1	分类 . . . . .	3
1.2	历史 . . . . .	3
<b>2</b>	<b>基础概念</b>	<b>4</b>
2.1	训练集与测试集 . . . . .	4
2.2	参数选择 . . . . .	5
2.3	性能度量 . . . . .	5
<b>3</b>	<b>线性模型</b>	<b>6</b>
3.1	线性回归 . . . . .	6
3.2	线性分类 . . . . .	7
3.3	线性判别分析 . . . . .	7
3.4	多分类学习 . . . . .	8
3.5	类别不平衡问题 . . . . .	8
<b>4</b>	<b>决策树</b>	<b>9</b>
4.1	信息增益 . . . . .	9
4.2	连续值处理 . . . . .	11
4.3	缺失值处理 . . . . .	11
4.4	多变量决策树 . . . . .	11
<b>5</b>	<b>神经网络</b>	<b>12</b>
5.1	感知机与单层神经网络 . . . . .	12
5.2	多层神经网络 . . . . .	13
5.3	深度学习 . . . . .	14
5.4	Transformer . . . . .	15
5.5	神经网络的历史发展 . . . . .	16

---

\*Build 20200727

<b>6</b>	<b>支持向量机</b>	<b>16</b>
6.1	支持向量	17
6.2	核函数	18
6.3	支持向量回归	19
<b>7</b>	<b>贝叶斯优化</b>	<b>20</b>
7.1	高斯混合模型	20
7.2	EM算法	21
<b>8</b>	<b>集成学习</b>	<b>21</b>
8.1	Boosting	22
8.2	Bagging	23
8.3	Stacking	24
8.4	总结	25
<b>9</b>	<b>聚类</b>	<b>26</b>
9.1	原型聚类	27
9.2	密度聚类	27
9.3	层次聚类	28
<b>10</b>	<b>降维与度量学习</b>	<b>29</b>
10.1	k近邻学习	29
10.2	主成分分析	30
10.3	核化线性降维	31
10.4	流形学习	31
10.5	自编码器	31
<b>11</b>	<b>特征选择</b>	<b>31</b>
11.1	子集搜索与子集评价	32
11.2	特征选择方法	32
<b>12</b>	<b>半监督学习</b>	<b>33</b>
12.1	生成式方法	33
12.2	半监督SVM	34
12.3	图半监督学习	34
12.4	半监督聚类	35
<b>13</b>	<b>推荐系统</b>	<b>35</b>
<b>14</b>	<b>参考资料</b>	<b>35</b>

本笔记对应周志华的《机器学习》（西瓜书）。

# 1 简介

机器学习理论主要是设计和分析一些让计算机可以自动“学习”的算法，数据挖掘是用人工智能、机器学习、统计学和数据库的交叉方法在相对较大型的数据集中发现模式的计算过程。大体上看，数据挖掘可以视为机器学习和数据库的交叉，它主要利用机器学习界提供的技术来分析海量数据，利用数据库界提供的技术来管理海量数据。

”A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .”  
—Tom Mitchell

## 1.1 分类

机器学习(machine learning)通常可以分为以下几类：

- 监督学习(supervised learning)：有标签(label)
  - 回归(regression)：连续值
  - 分类(classification)：离散值
- 无监督学习(unsupervised learning)：无标签
  - 降维
  - 聚类(clustering)
- 强化学习(reinforcement learning)：延后的标签

## 1.2 历史

- 推理期(1950-1970)：逻辑理论家(Logic Theorist)A. Newell & H. Simon(1975图灵奖)
- 知识期(1970s)：知识工程之父E. A. Feigenbaum(1994图灵奖)
- 学习期(1980s)：决策树、归纳逻辑程序设计(Prolog)

机器学习的五大学派(tribe)：

- 符号主义(symbolist)
- 联结主义(connectionist)
- 进化主义(evolutionaries)
- 贝叶斯主义(bayesians)
- 类比主义(analogizers)



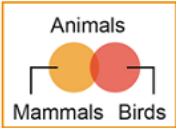




# A look at

## Machine learning evolution

### Overview

For decades, individual “tribes” of artificial intelligence researchers have vied with one another for dominance. Is the time ripe now for tribes to collaborate? They may be forced to, as collaboration and algorithm blending are the only ways to reach true artificial general intelligence (AGI). Here’s a look back at how machine learning methods have evolved and what the future may look like.

### What are the five tribes?

Symbolists	Bayesians	Connectionists	Evolutionaries	Analogizers
				
Use symbols, rules, and logic to represent knowledge and draw logical inference	Assess the likelihood of occurrence for probabilistic inference	Recognize and generalize patterns dynamically with matrices of probabilistic, weighted neurons	Generate variations and then assess the fitness of each for a given purpose	Optimize a function in light of constraints (“going as high as you can while staying on the road”)
Favored algorithm Rules and decision trees	Favored algorithm Naive Bayes or Markov	Favored algorithm Neural networks	Favored algorithm Genetic programs	Favored algorithm Support vectors

Source: Pedro Domingos, *The Master Algorithm*, 2015

## 2 基础概念

### 2.1 训练集与测试集

- 留出法(hold-out): 直接将数据集划分为两个互斥的集合, 一个作为训练集, 另一个作为测试集
  - 注意数据分布的一致性, 通过多次随机划分取平均保证
  - 通常用 $2/3 \sim 4/5$ 的样本用于训练, 其余用作测试
- 交叉验证法(cross validation)/ $k$ 折(fold)交叉验证: 划分为 $k$ 个互斥子集, 其中 $k - 1$ 个用于训练, 最后一个用于验证, 训练 $k$ 次, 对这 $k$ 次结果取平均
  - 通常采用10次10折交叉验证, 每次都换划分方式, 确保随机性
- 自助法(bootstrapping): 从原始数据集中放回采样得到新数据集作为测试集
  - 在数据集较小的、难以有效划分训练/测试集时比较有用

- 但改变了初始数据集分布，会引入估计偏差

注意：通常将习得模型实际使用中遇到的数据称为测试集，而将训练的数据划分为训练集与验证集(validation)

## 2.2 参数选择

参数通常包括

- 模型本身的参数(parameter)：通过学习改变
- 超参数(superparameter)：预先设定，调参实际上就是在选择算法

## 2.3 性能度量

- 回归：通常采用均方误差(MSE)
- 分类：错误率、精度

对于二分类问题，有混淆矩阵(confusion matrix)

	预测正	预测反
真正正	TP(真正例)	FN(假反例)
真实反	FP(假正例)	TN(真反例)

$$\text{查准率 } P = \frac{TP}{TP + FP}$$

$$\text{查全率 } R = \frac{TP}{TP + FN}$$

为了衡量机器学习算法的泛化性能，需要知道以下指标：

- 方差(variance)：度量同样大小的训练集的变动所导致的学习性能的变化，即刻画数据扰动所造成的影响

$$\mathbb{D}(\mathbf{x}) = \mathbb{E}_D \left[ (f(\mathbf{x}; D) - \bar{f}(\mathbf{x}))^2 \right]$$

- 偏差(bias)：度量学习算法的期望预测和真实结果的偏离程度，即刻画学习算法本身的拟合能力

$$b^2(\mathbf{x}) = (\bar{f}(\mathbf{x}) - y)^2$$

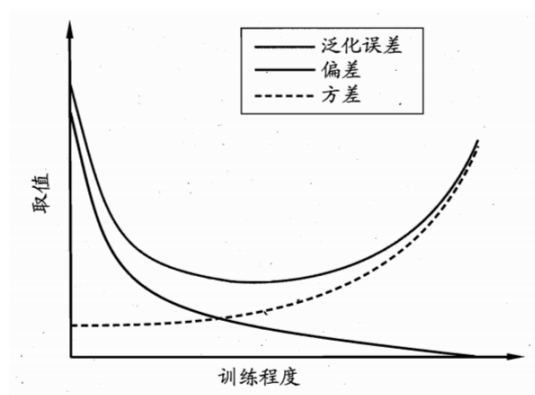
- 噪声(noise)：表达在当前任务上任何学习算法所能达到的期望泛化误差的下界

$$\varepsilon^2 = \mathbb{E}_D \left[ (y_D - y)^2 \right]$$

泛化误差可以分解为

$$E(f; D) = b^2(\mathbf{x}) + \mathbb{D}(\mathbf{x}) + \varepsilon^2$$

即泛化性能是由学习算法的能力、数据的充分性及学习任务本身的难度决定的



### 3 线性模型

线性模型

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

由于 $\mathbf{w}$ 直观表达了各属性在预测中的重要性，因此线性模型具有很好的可解释性(comprehensibility)。

#### 3.1 线性回归

数据集 $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ，每个样本 $\mathbf{x}_i$ 都由 $d$ 个属性描述，多元线性回归希望(multivariate linear regression)学到

$$f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i + b, \text{ s.t. } f(\mathbf{x}_i) \simeq y_i$$

将 $\mathbf{w}$ 和 $b$ 写在一起变成 $\mathbf{w} \leftarrow \begin{bmatrix} \mathbf{w} & b \end{bmatrix}$ ，并设

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

为已知， $\mathbf{w}$ 为需要训练的权重。再将标记写成向量形式 $\mathbf{y} = [y_1 \ y_2 \ \cdots \ y_m]$ ，进而得到最小二乘优化

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \|\mathbf{y} - X\mathbf{w}\|_2^2$$

对 $\mathbf{w}$ 求导有

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

当 $X^T X$ 满秩或正定时，令上式为0有

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

但现实中大多数时候 $X^T X$ 都非可逆阵，故常用正则化方法。

## 3.2 线性分类

单位阶跃(unit-step)函数

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

不连续, 故用对数几率(logistic)函数替代

$$y = \frac{1}{1 + e^{-z}}$$

这是一种Sigmoid函数, 将线性表达式代入有

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

进而有

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

将 $y$ 视为样本 $\mathbf{x}$ 为正例的可能性, 则 $1 - y$ 为反例可能性, 两者比值 $y/(1 - y)$ 称为几率(odds), 反映了 $\mathbf{x}$ 作为正例的相对可能性。对数几率又称logit, 故这种方法又称为逻辑斯蒂(logistic)回归, 但其实是分类学习方法, 是一种广义的线性模型。

优点: 无须事先假设数据分布, 可以得到类别的近似概率预测, 可直接应用现有的数值优化算法求得最优解。

将 $y$ 视为后验概率估计, 有

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

显然有

$$\begin{aligned} \mathbb{P}(y = 1 | \mathbf{x}) &= \frac{e^{\mathbf{w}^T \mathbf{x} + b}}{1 + e^{\mathbf{w}^T \mathbf{x} + b}} \\ \mathbb{P}(y = 0 | \mathbf{x}) &= \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x} + b}} \end{aligned}$$

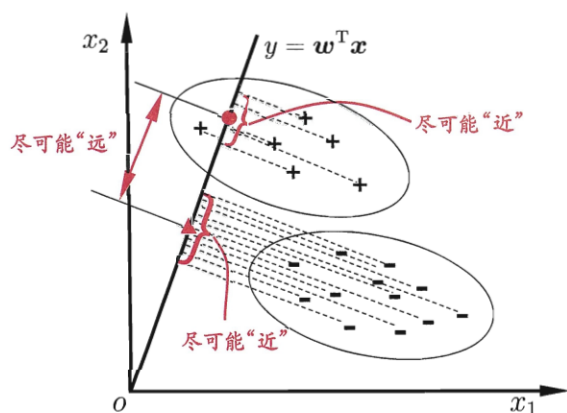
给定数据集 $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ , 由极大似然法估计 $\mathbf{w}$ 和 $b$ 有

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

可以用梯度下降或牛顿法进行求解。

## 3.3 线性判别分析

线性判别分析(Linear Discriminant Analysis, LDA)[Fisher, 1936]同样用于二分类, 希望将样例投影到一条直线上, 使得同类样例投影点尽可能近, 异类样例投影点尽可能远。可通过同类样例投影点协方差尽可能小, 异类样例类中心距离尽可能大实现。



最优化广义瑞利商

$$J = \frac{\mathbf{w}^T (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1) (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^T \mathbf{w}}{\mathbf{w}^T (\Sigma_0 + \Sigma_1) \mathbf{w}}$$

$$= \frac{\mathbf{w}^T S_b \mathbf{w}}{\mathbf{w}^T S_w \mathbf{w}}$$

其中  $S_w$  为类内散度矩阵,  $S_b$  为类间散度矩阵。

$$S_w = \Sigma_0 + \Sigma_1$$

$$= \sum_{\mathbf{x} \in X_0} (\mathbf{x} - \boldsymbol{\mu}_0)(\mathbf{x} - \boldsymbol{\mu}_0)^T + \sum_{\mathbf{x} \in X_1} (\mathbf{x} - \boldsymbol{\mu}_1)(\mathbf{x} - \boldsymbol{\mu}_1)^T$$

$$S_b = (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^T$$

LDA也是经典的监督降维技术。

### 3.4 多分类学习

利用二分类学习器解决多分类问题, 拆分策略:

- 一对一(One vs. One, OvO): 共  $\binom{n}{2} = n(n-1)/2$  个学习器, 投票产生最终结果
- 一对其余(One vs. Rest, OvR): 一类作为正例, 其他作为反例,  $n$  个学习器, 置信度最大的作为最终类别
- 多对多(Many vs. Many, MvM)

### 3.5 类别不平衡问题

- 欠采样(undersampling): 减少样例
- 过采样(oversampling): 增加样例
- 阈值移动(threshold-moving)/再缩放(rescaling)

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



## 4 决策树

### 4.1 信息增益

假设当前样本 $D$ 中第 $k$ 类样本所占比例为 $p_k$ （如好瓜坏瓜二分类 $M$ 就是2），则 $D$ 的信息熵为

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

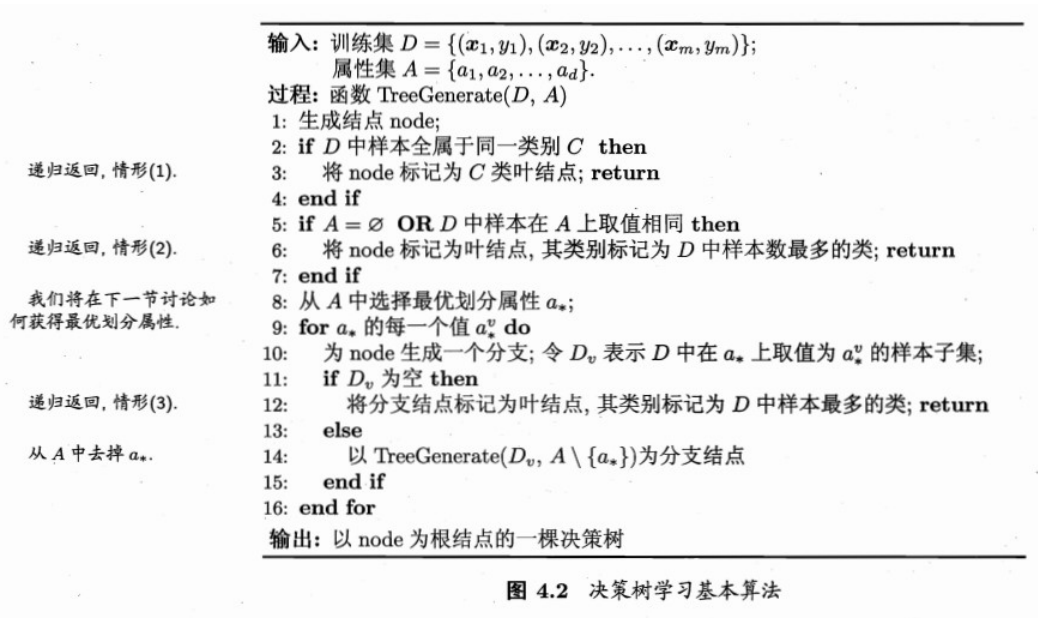
$Ent(D)$ 的值越小，则 $D$ 的纯度越高。

假设离散属性 $a$ 有 $V$ 个可能取值 $\{a^1, a^2, \dots, a^V\}$ （如颜色属性，红蓝绿，则 $a^1 = R, a^2 = B, a^3 = G$ ），则产生 $V$ 个分支结点，第 $v$ 个结点包含取值为 $a^v$ 的样本，记为 $D^v$ 。进而可以给不同分支结点赋予权值，并计算用属性 $a$ 进行划分的信息增益(information gain)

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

一般信息增益越大，意味着依据 $a$ 划分所获得的纯度提升越大。因此每次划分采用最大信息增益的属性进行划分，此即ID3（迭代二分器，Iterative Dichotomiser）决策树学习算法[Quinlan, 1986]。





注意先前划分的指标后面可能依然会被作为划分标准, 只要信息增益最大。

通过剪枝来避免过拟合

- 预剪枝: 对每个结点在划分前进行估计, 若划分不能提升决策树泛化性能, 则停止划分并将当前结点标记为叶结点, 类别标记为训练样例最多的类别。
- 后剪枝: 先完整生成一棵决策树, 然后自底向上对非叶结点进行考察, 若结点对应的子树替换成叶结点能够带来决策树泛化性能提升, 则将该子树替换为叶结点。

类似地采用其他指标可以得到其他决策树算法:

- C4.5(Classifier)算法[Quinlan, 1993]: 用增益率(gain ratio)来选择最优划分属性

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

其中

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

为属性 $a$ 的固有值(intrinsic value)。但C4.5采用启发式算法选择划分, 而不是单纯基于增益率。增益率准则对可取值数目较少的属性有所偏好。

- CART(Classification and Regression Tree)算法[Breiman, 1984]: 采用基尼指数(Gini index)

$$\text{Gini}(D) = \sum_{k=1}^M \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^M p_k^2$$

反映了从数据集 $D$ 中随机抽取两个样本, 类别标记不一致的概率。

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

选择基尼指数较小的进行划分。CART算法包括决策树生成和决策树剪枝两个部分，回归树最小化平方误差，分类树最小化基尼指数。

理想的决策树应该是叶子结点数最少、叶子结点深度最小、叶子结点数最少且叶子结点深度最小的树。找到这种最优的决策树是NP难题。决策树优化的目的就是要找到尽可能趋向于最优的决策树。

## 4.2 连续值处理

连续属性的离散化常用二分法，这是在C4.5算法中采用的方法。

假定连续属性 $a$ 在 $D$ 上出现了 $n$ 个不同的取值，将这些值从小到大排序记为 $\{a^1, a^2, \dots, a^n\}$ ，基于划分点 $t$ 可以将 $D$ 分为子集 $D_t^-$ 和 $D_t^+$ ，进而可以考虑 $n-1$ 个候选划分点集合

$$T_a = \left\{ \frac{a^i + a^{i+1}}{2} \mid 1 \leq i \leq n-1 \right\}$$

## 4.3 缺失值处理

$\tilde{D}$ 表示 $D$ 中在属性 $a$ 上没有缺失值的样本子集， $\tilde{D}^v$ 表示 $\tilde{D}$ 中在属性 $a$ 上取值为 $a^v$ 的样本子集， $\tilde{D}_k$ 表示 $\tilde{D}$ 中属于第 $k$ 类的样本子集。为每个样本 $\mathbf{x}$ 赋予权重 $w_{\mathbf{x}}$ ，定义

$$\begin{aligned} \rho &= \frac{\sum_{\mathbf{x} \in \tilde{D}} w_{\mathbf{x}}}{\sum_{\mathbf{x} \in D} w_{\mathbf{x}}} \\ \tilde{p}_k &= \frac{\sum_{\mathbf{x} \in \tilde{D}_k} w_{\mathbf{x}}}{\sum_{\mathbf{x} \in \tilde{D}} w_{\mathbf{x}}} \quad (1 \leq k \leq |\mathcal{Y}|) \\ \tilde{r}_v &= \frac{\sum_{\mathbf{x} \in \tilde{D}^v} w_{\mathbf{x}}}{\sum_{\mathbf{x} \in \tilde{D}} w_{\mathbf{x}}} \quad (1 \leq v \leq V) \end{aligned}$$

分别为无缺失值样本所占的比例、无缺失样本中第 $k$ 类所占的比例、无缺失值样本中在属性 $a$ 上取值 $a^v$ 的样本所占的比例。显然有 $\sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k = 1$ 和 $\sum_{v=1}^V \tilde{r}_v = 1$ 。

可将信息增益推广为

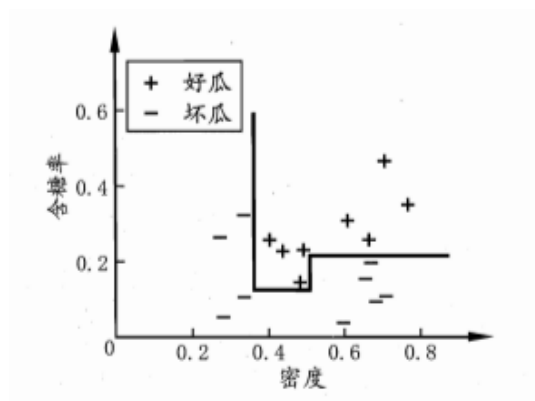
$$\begin{aligned} \text{Gain}(D, a) &= \rho \times \text{Gain}(\tilde{D}, a) \\ &= \rho \times \left( \text{Ent}(\tilde{D}) - \sum_{v=1}^V \tilde{r}_v \text{Ent}(\tilde{D}^v) \right) \end{aligned}$$

其中

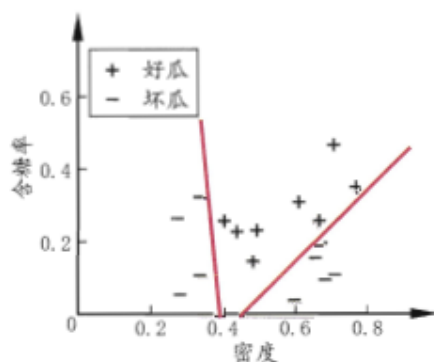
$$\text{Ent}(\tilde{D}) = - \sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k \log_2 \tilde{p}_k$$

## 4.4 多变量决策树

决策树形成的分类边界有一个明显的特点，即与坐标轴平行，故学习的结果具有较好的可解释性，因为每一段划分都直接对应某个属性的取值。但在学习任务的真实边界比较复杂时，就需要使用很多段划分才能得到比较好的近似。



如果采用线性分类器，则变成多变量决策树，可以实现更加复杂的划分。

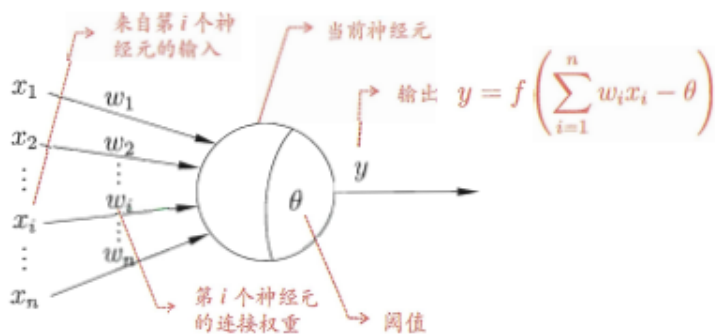


决策树可能会陷入局部最优（贪心增益最高），同时分类边界表达能力弱，之后可采用随机森林的方法对性能进行提升。

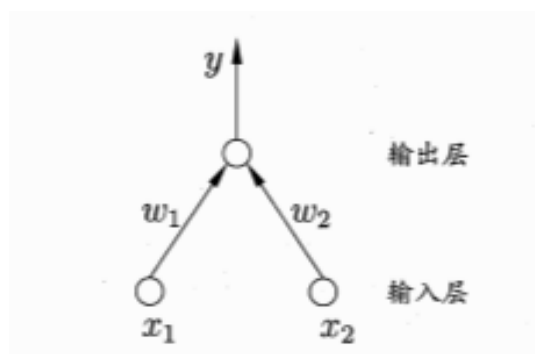
## 5 神经网络

### 5.1 感知机与单层神经网络

下图是M-P神经元(neuron)模型[McCulloch and Pitts, 1943]，一直沿用至今。



感知机(perceptron)由两层神经元组成，输入层接收外界输入信号后传递给输出层，输出层是M-P神经元，亦称“阈值逻辑单元”(threshold logic unit)。



感知机能够容易实现与或非运算，只要设定好特定的权重和阈值。

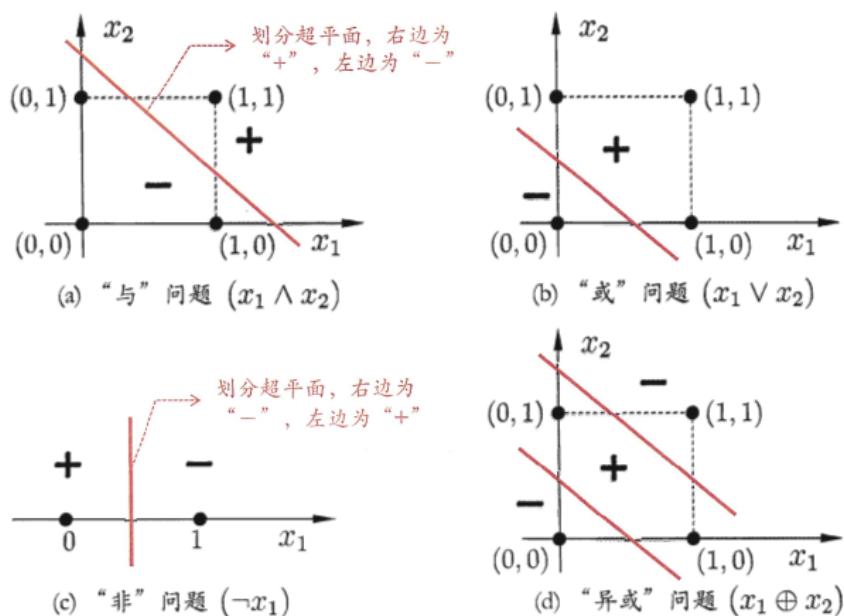
感知机的学习规则非常简单，对训练样例 $(\mathbf{x}, y)$ ，如果当前感知机的输出为 $\hat{y}$ ，则感知机的权重依照下式修改

$$w_i \leftarrow w_i + \Delta w_i$$

$$w_i = \eta(y - \hat{y})x_i$$

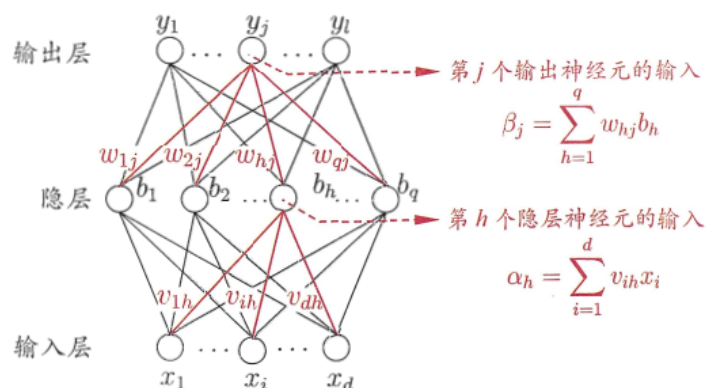
其中 $\eta \in (0, 1)$ 称为学习率(learning rate)。感知机预测正确则不发生变化，否则根据错误的程度对权重进行修改。

Minsky和Papert[1969]证明了若两类模式是线性可分的，则必然存在一个线性超平面将它们分开，感知机一定收敛；但非线性可分，如异或，感知机无法收敛。



## 5.2 多层神经网络

要解决非线性可分问题，则需要用到多层神经网络。



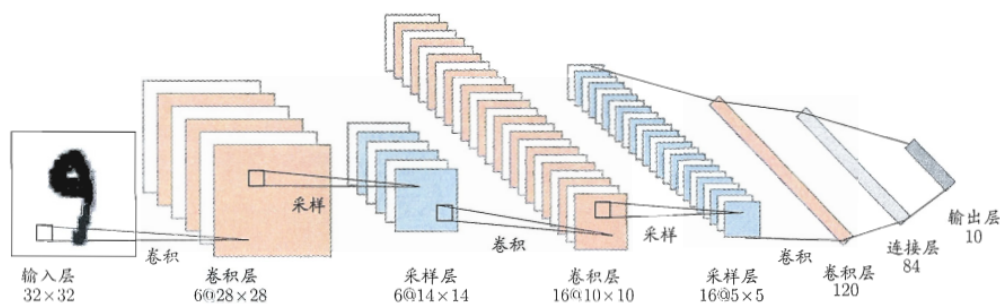
用反向传播算法(Back propagation, BP)进行学习。

通用近似定理[Hornik, 1989]证明，只需一个包含足够多神经元的隐层，多层前馈神经网络就能以任意精度逼近任意复杂度的连续函数。然而，如何设置神经元的个数却没有固定的标准。

### 5.3 深度学习

一般情况下，复杂模型的训练效率低，易陷入过拟合，因此难以受到人们青睐。但是随着云计算和大数据时代的到来，计算能力的大幅提高可以有效缓解训练的低效性，训练数据的大幅增加则可以降低过拟合风险，因此以深度学习(deep learning)为代表的复杂模型开始受到人们关注。

- 深度信念网络(Deep belief network, DBN)[Hinton, 2006]：每一层都是一个受限Boltzmann机
- 卷积神经网络(Convolutional neural network, CNN)[LeCun, 1995]：权值共享



深度学习实际上是通过多层处理，将初始的“低层”特征表示转化为“高层”特征表示后，用“简单的模型”就可以完成复杂的学习任务。（网络前若干层都是在特征表示，最后一层则是简单的分类。）因此可以将深度学习理解为进行特征学习或表示学习(representation learning)。

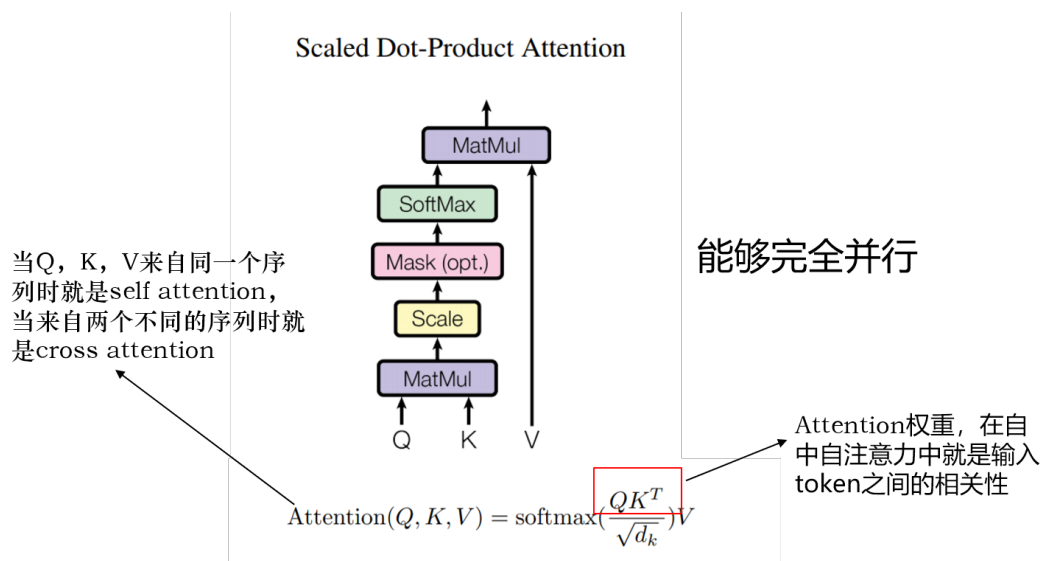
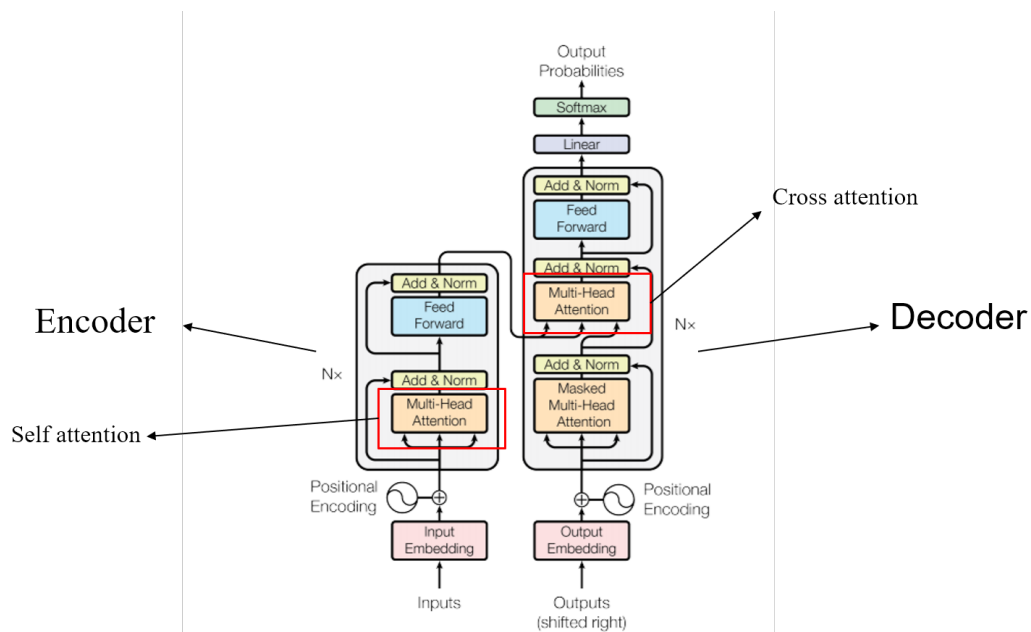
以往机器学习在用于现实任务时，描述样本的特征通常需要人类专家进行设计，这称为特征工程(feature engineering)。但现在有了深度学习，则是进一步将特征提取的工作交由机器来做。

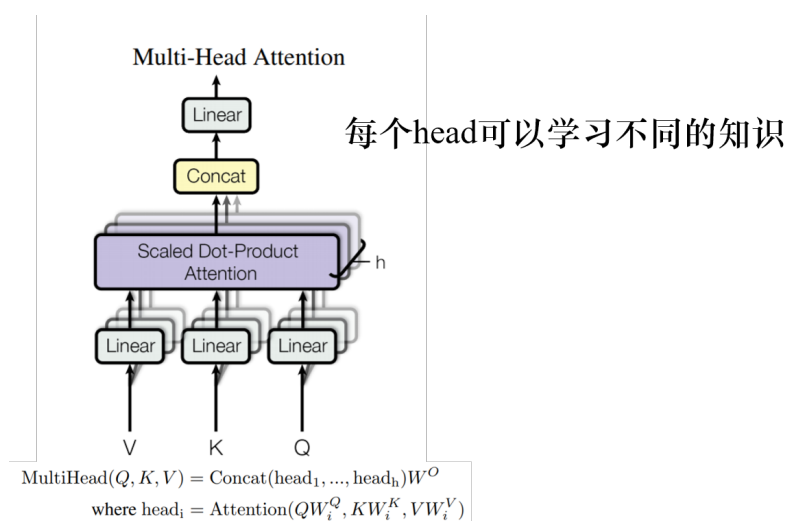
- 局部最优
- 梯度消失：ReLU、ResNet
- 过拟合：正则化（Dropout、早停）

- 优化: 适应性学习率 (Momentum、Nesterov、Adagrad、Adadelta、RMSprop、Adam)

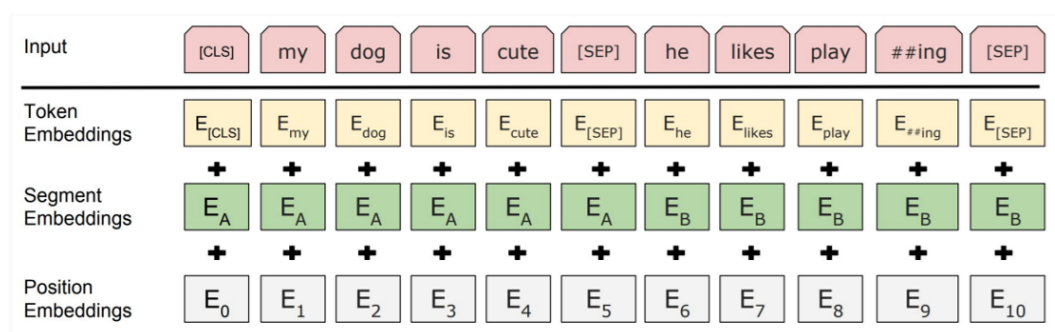
## 5.4 Transformer

Vaswani et al., *Attention Is All You Need*, NeurIPS, 2017





进而有BERT模型



## 5.5 神经网络的历史发展

- 1940s: M-P神经元模型
- 1950s: 感知机
- 1969: Minsky(MIT)出版了《感知机》一书，指出单层神经网络无法解决非线性问题，而多层神经网络训练算法看不到希望，直接导致神经网络研究进入了“冰河期”
- 1974: Werbos(Harvard)发明BP算法，但仍处于冰河期不受重视
- 1983: Hopfield(Caltech)利用神经网络在旅行商问题(TSP)上取得当时最好结果，后来Rumelhart等人重新发明BP算法，掀起神经网络第二次高潮
- 1990s: 统计学习理论和支持向量机的兴起，神经网络学习的理论性质不够清楚、试错性强、使用中充斥大量窍门的弱点更为明显，神经网络研究又陷入低谷
- 2010s: 算力提升+大数据涌现，神经网络在ImageNet上以绝对优势夺冠，神经网络迎来第三次高潮

## 6 支持向量机

支持向量机 (support vector machines, SVM) 是一种二分类模型，它的基本模型是定义在特征空间



上的间隔最大的线性分类器。

## 6.1 支持向量

超平面由下列方程决定

$$\mathbf{w}^T \mathbf{x} + b = 0$$

样本空间中任意点到超平面的距离为

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

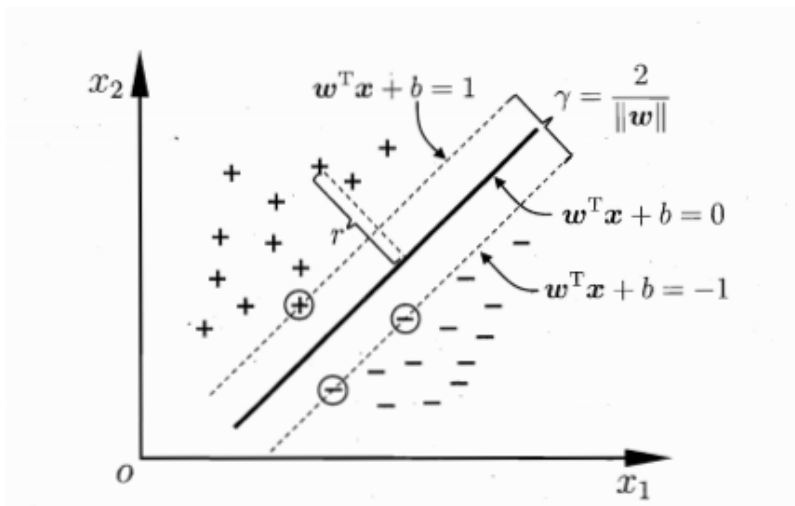
若超平面 $(\mathbf{x}, b)$ 能将样本正确分类，则

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq +1 & y_i = +1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 & y_i = -1 \end{cases}$$

距离超平面最近的使上式成立的点称为支持向量(support vector)，两个异类支持向量到超平面的距离之和为

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

被称之为间隔(margin)。



因此为找到最大间隔来划分超平面（分类结果最鲁棒，泛化能力最强），则需要求解下述最优化问题

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

显然，为了最大化间隔，只需最大化 $\|\mathbf{w}\|^{-1}$ ，等价于最小化 $\|\mathbf{w}\|^2$ ，于是上述优化问题可重写为

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

此即为支持向量机(Support Vector Machine, SVM)的基本型。

可以用拉格朗日乘子法求对偶问题并得到KKT条件，可利用凸优化的方法进行求解。

对偶问题为

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) - \sum_{i=1}^m \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned}$$

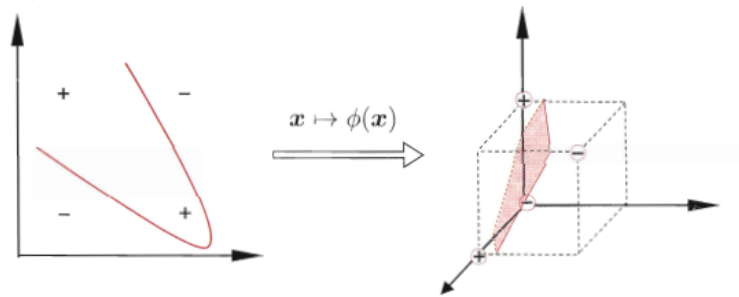
SVM可用SMO求解，通过选取一对更新的变量 $\alpha_i$ 和 $\alpha_j$ ，固定 $\alpha_i$ 和 $\alpha_j$ 以外的参数，求解对偶问题更新 $\alpha_i$ 和 $\alpha_j$ 。

支持向量机的解具有稀疏性：训练完成后，大部分的训练样本都不需保留，最终模型仅与支持向量有关。

支持向量机于[Cortes and Vapnik, 1995]正式发表，由于在文本分类任务中显示出卓越性能[Joachims, 1998]，很快成为机器学习主流技术，并直接掀起统计学习(statistical learning)在2000年前后的高潮。

## 6.2 核函数

如果将非线性可分样本从原始空间映射到更高维的特征空间，则可以使其在新的特征空间中线性可分。



由于 $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ 的维度可能很高，故不显式地设计核映射，而是设想有这样的核函数(kernel function)

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

使得 $\mathbf{x}_i$ 和 $\mathbf{x}_j$ 在特征空间的内积等于它们在原始样本空间中通过核函数计算的结果。

**定理 1** (Mercer定理 (充分非必要)). 只要一个对称函数所对应的核矩阵半正定，则它就能作为核函数来

使用。

几种常见的核函数如下

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)

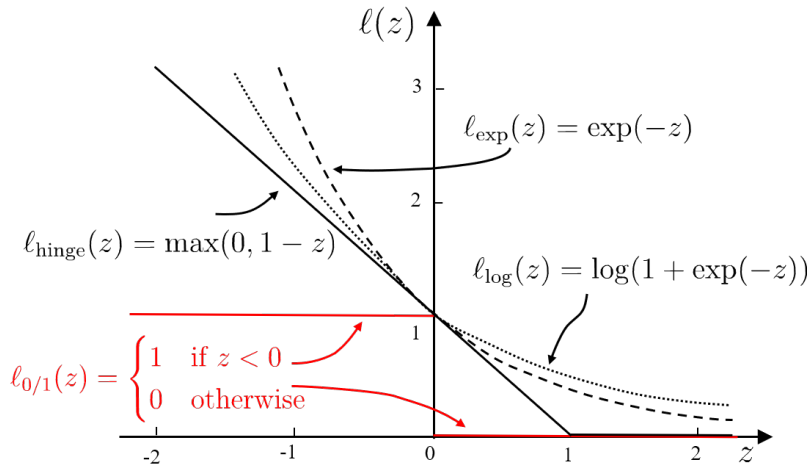
现实中, 很难确定合适的核函数使得训练样本在特征空间中线性可分; 一个线性可分的结果也很难断定是否是有过拟合造成的。软间隔允许支持向量机在一些样本上不满足约束。基本想法是最大化间隔的同时, 让不满足约束的样本尽可能少。

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

其中 $l_{0/1}$ 是0/1损失函数

$$l_{0/1} = \begin{cases} 1 & z < 0 \\ 0 & \text{otherwise} \end{cases}$$

存在的问题是0/1损失函数非凸、非连续, 不易优化。

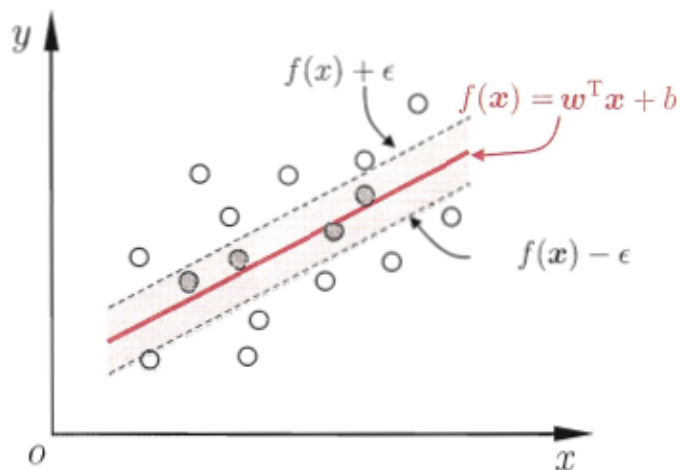


替代损失函数数学性质较好, 一般是0/1损失函数的上界。

根据KKT条件可推得最终模型仅与支持向量有关, 也即hinge损失函数依然保持了支持向量机解的稀疏性。

### 6.3 支持向量回归

支持向量回归(Support vector regression, SVR)允许 $f(\mathbf{x})$ 与 $y$ 之间最多有 $\varepsilon$ 的偏差



SVR问题可写为

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m l_{\epsilon}(f(\mathbf{x}_i) - y_i)$$

其中 $C$ 为正则化常数,  $l_{\epsilon}$ 为 $\epsilon$ -不敏感损失( $\epsilon$ -insensitive loss)函数

$$l_{\epsilon}(z) = \begin{cases} 0 & |z| \leq \epsilon \\ |z| - \epsilon & \text{otherwise} \end{cases}$$

## 7 贝叶斯优化

### 7.1 高斯混合模型

最常用的混合模型即高斯混合模型(Gaussian Mixture Model, GMM),

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} \mid \mu_k, \Sigma_k)$$

代表有 $K$ 个高斯分布进行混合, 其中 $\pi_k$ 为混合系数, 满足

$$\sum_k \pi_k = 1, \pi_k \geq 0, \forall k$$

求最大似然

$$\ln p(X \mid \pi, \mu, \Sigma) = \sum_{n=1}^N \ln \left( \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}^{(n)} \mid \mu_k, \Sigma_k) \right)$$

优化变量为 $\Theta = \{\pi_k, \mu_k, \Sigma_k\}$ 。

## 7.2 EM算法

未观测变量即隐变量(latent variable)，令 $X$ 表示已观测变量集， $Z$ 表示隐变量集， $\Theta$ 表示模型参数。若对 $\Theta$ 做极大似然估计，则应最大化对数似然

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

但由于 $Z$ 是隐变量，上式无法直接求解。这时可以通过对 $Z$ 计算期望，来最大化已观测数据的对数边际似然(marginal likelihood)

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

期望最大化(Expectation-Maximization, EM)算法是常用估计参数隐变量的方法（非梯度优化）。

- 期望(E)步：利用当前估计的参数值来计算对数似然的期望值
- 最大化(M)步：寻找能使E步产生的似然期望最大化的参数值

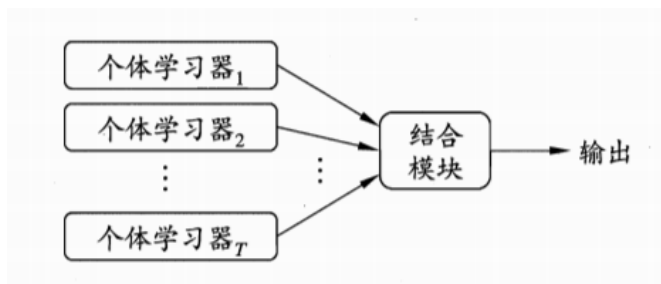
新得到的参数值重新被用于E步，直至收敛至局部最优解。

在GMM上用EM算法与K-means非常类似，只不过EM算法是软指派(soft assignment)，而且每一个中心根据软指派数据的加权平均进行移动。b

## 8 集成学习

集成学习(ensemble learning)通过构建并结合多个学习器来完成学习任务，通常可以获得比单一学习器更为显著的泛化性能。

同质(homogeneous)集成中的个体学习器称为基学习器(base learner)；而异质(heterogeneous)集成中的个体学习器则一般称为组件学习器(component learner)。



考虑二分类问题，假设基分类器的错误率为

$$\mathbb{P}(h_i(\mathbf{x}) \neq f(\mathbf{x})) = \varepsilon$$

假设集成通过简单投票法结合 $T$ 个分类器，若有超过半数的基分类器正确则分类正确

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

假设基分类器的错误率相互独立，则由Hoeffding不等式可得集成错误率为

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

即在一定条件下，随着集成分类器数目的增加，集成的错误率将指数级下降，最终趋于0。

但上面的分析有一假设：基学习器的误差相互独立。现实生活中个体学习器是为解决同一个问题而训练出来的，显然不可能相互独立。个体学习器的“准确性”和“多样性”之间本来就存在冲突。如何产生“好而不同”的个体学习器是集成学习研究的核心。

## 8.1 Boosting

提升(Boosting)可以将弱学习器提升为强学习器，后面的模型是基于前面模型的训练结果（误差），个体学习器之间存在**强依赖关系**，**串行生成**，它的代表是AdaBoost。

### 8.1.1 AdaBoost

每一轮的训练集不变，只是训练集中每个样例在分类器中的**权重发生变化**，而权值是根据上一轮的分类结果进行调整。在每一步迭代中，会给训练集中的样本赋予权重 $w_i$ ，样本的初始权重都一样为 $1/n$ 。每一步迭代中，被当前弱分类器**分错的样本权重会相应得到提高**，被当前弱分类器分对的样本权重则降低（即刻意练习学得不好的样本），弱分类器的权重则由当前分类器的加权错误率来决定。

对目标函数 $f(\mathbf{x})$ 进行多次逼近，通过不断拟合残差达到逼近的效果，可以按照下式不断迭代

$$\begin{array}{ll}f_1(\mathbf{x}) = \widehat{f}(\mathbf{x}) & h_1(\mathbf{x}) = f(\mathbf{x}) - f_1(\mathbf{x}) \\f_2(\mathbf{x}) = f_1(\mathbf{x}) + \widehat{h}_1(\mathbf{x}) & h_2(\mathbf{x}) = f(\mathbf{x}) - f_2(\mathbf{x}) \\f_3(\mathbf{x}) = f_2(\mathbf{x}) + \widehat{h}_2(\mathbf{x}) & h_3(\mathbf{x}) = f(\mathbf{x}) - f_3(\mathbf{x}) \\ \vdots & \vdots \\f_n(\mathbf{x}) = f_{n-1}(\mathbf{x}) + \widehat{h}_{n-1}(\mathbf{x}) & \end{array}$$



1. 随机有放回地抽取数据（使用Bagging）输入决策树模型
2. 随机选取 $k$ 个特征
3. 随机选择特征取值进行分割（不遍历特征所有取值）

注意由于建立决策树时的样本选择和特征选择所提供的随机性，因此不需要剪枝。

有两种指标衡量随机森林的表现：

- 分类间隔(margin)：森林中正确样本决策树的比例减去错误样本决策树的比例。假设样本 $A$ 有75%的树分类正确，那么分类间隔就是 $75\%-25\%=50\%$ 。通常希望分类间隔越大越好，因为大的间隔表示我们的分类效果比较稳定，泛化效果更好。
- 袋外错误率(out-of-bag error)：对每棵树来说，都有部分样本没有被抽样进入训练样本，这些即为袋外样本。随机森林对袋外样本的预测错误比率被称为袋外错误率。
  1. 对每个样本，计算把该样本作为袋外样本的树对该样本的分类情况
  2. 以简单多数投票作为该样本的分类结果
  3. 用误分样本个数占样本总数的比率作为随机森林的袋外错误率

优点：

- 能够处理很高维数据，且不用做特征选择
- 训练完可给出哪些特征比较重要
- 容易做成并行化方法

缺点：相对于决策树，执行速度慢，模型可解释性较差

### 8.3 Stacking

最简单的就是简单平均或加权平均。另外也可以通过投票法，绝对多数(majority)投票法、相对多数(plurality)投票法和加权投票法。

当训练数据很多时，更为强大的结合策略是使用学习法，即通过另一个学习器来进行结合。Stacking [Wolpert, 1992]是学习法的典型代表。这里把个体学习器称为次级学习器或元学习器(meta-learner)。

Stacking先从初始数据集中训练出初级学习器，然后**生成一个新数据集**用于训练次级学习器。在新数据集中，初级学习器的输出被当作样例输入特征，而初始样本的标记仍被当作样例标记。注意这个数据集的大小为 $mT$ ， $m$ 为原数据集训练样本数， $T$ 为学习器数目。



---

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

由于次级训练集是用初级学习器产生的，故直接用初级学习器的训练集产生次级学习器过拟合风险较大；因此常通过交叉验证或留一法，用训练初级学习器未使用的样本来产生次级学习器的训练样本。以 $k$ 折交叉验证为例，初始训练集 $D$ 被随机划分为 $k$ 个大小相似的集合 $D_1, D_2, \dots, D_k$ 。令 $D_j$ 和 $\bar{D}_j = D \setminus D_j$ 分别表示第 $j$ 折的测试集和训练集。给定 $T$ 个初级学习算法，初级学习器 $h_t^{(j)}$ 通过在 $\bar{D}_j$ 上使用第 $t$ 个学习算法而得。对于 $D_j$ 的每个样本 $\mathbf{x}_i$ ，令 $z_{it} = h_t^{(j)}(\mathbf{x}_i)$ ，则由 $\mathbf{x}_i$ 所产生的次级训练样例的实例部分为 $\mathbf{z}_i = [z_{i1} \ z_{i2} \ \dots \ z_{iT}]$ ，标记部分为 $y_i$ 。故整个交叉验证过程结束后，从这 $T$ 个初级学习器产生的次级训练集是 $D' = \{(\mathbf{z}_i, y_i)\}_{i=1}^m$ ，然后 $D'$ 将用于训练次级学习器。

总结Stacking的过程即：

1. 划分训练数据集为两（或多）个不相交的集合
2. 在第一个集合上训练多个学习器
3. 在第二个集合上测试这几个学习器
4. 把第三步得到的预测结果作为输入，把正确的回应作为输出，训练一个高层学习器

## 8.4 总结

多样性增强的方法：

- 数据样本扰动：通常基于采样法，Bagging中的自助采样法和AdaBoost中的序列采样。数据样本扰动对“不稳定基学习器”很有效。
  - 对数据样本扰动敏感的不稳定基学习器：决策树、神经网络
  - 对数据样本扰动不敏感的稳定基学习器：线性学习器、支持向量机、朴素贝叶斯、 $k$ 近邻
- 输入属性扰动：随机子空间算法，基于初始属性集，抽取出若干个属性子集，然后基于每个属性子集训练一个基学习器

## 9 聚类

直观来讲, 我们希望“物以类聚”, 即同一簇的样本尽可能彼此相似, 不同簇的样本尽可能不同。换言之, 聚类结果的簇内相似度(intra-cluster similarity)高, 且簇间相似度(inter-cluster similarity)低, 这样的聚类效果较好。

性能度量指标:

- 外部指标

- Jaccard系数(Jaccard Coefficient, JC)

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

- FM指数(Fowlkes and Mallows Index, FMI)

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

- Rand指数(Rand Index, RI)

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

- 内部指标: 考虑聚类结果的簇划分  $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$

- 簇  $C_l$  内样本间的平均距离

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

- 簇  $C_l$  内样本间的最远距离

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

- 簇  $C_i$  与簇  $C_j$  最近样本间的距离

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

- 簇  $C_i$  与簇  $C_j$  中心点间的距离

$$d_{cen}(C) = dist(\mu_i, \mu_j)$$

- DB指数(Davies-Bouldin Index, DBI): 越小越好

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

– Dunn指数(Dunn Index, DI): 越大越好

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

VDM(Value Difference Metric)可以用于处理无序属性, 令 $m_{u,a}$ 表示属性 $u$ 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,  $k$ 为样本数, 则属性 $u$ 上两个离散值 $a$ 与 $b$ 之间的VDM距离为

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

## 9.1 原型聚类

给定样本集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ , Kmeans针对聚类所得簇划分 $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ 最小化平方误差

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

其中 $\boldsymbol{\mu}_i$ 为簇 $C_i$ 的均值向量。上式一定程度上刻画了簇内样本围绕簇均值向量的紧密程度,  $MSE$ 越小则簇内样本相似度越高。

初始化每个簇的均值向量, 不断更新簇划分并重新计算均值, 若均值向量均未更新则停止。

## 9.2 密度聚类

此类算法假设聚类结构能通过样本分布的紧密程度来确定。通常情况下, 密度聚类算法从样本密度的角度来考察样本之间的可连接性, 并基于可连接样本不断扩展聚类簇来获得最终的聚类结果。

DBSCAN基于一组邻域参数 $(\epsilon, MinPts)$ 来刻画样本分布的紧密程度。给定数据集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ , 定义以下概念:

- $\epsilon$ -邻域:  $N_\epsilon = \{\mathbf{x}_i \in \mathcal{D} \mid \text{dist}(\mathbf{x}_i, \mathbf{x}_j) \leq \epsilon\}$
- 核心对象(core object): 若 $|N_\epsilon(\mathbf{x}_j)| \geq MinPts$ , 则 $\mathbf{x}_j$ 是一个核心对象
- 密度直达(directly density-reachable): 若 $\mathbf{x}_j$ 位于 $\mathbf{x}_i$ 的 $\epsilon$ -邻域中, 且 $\mathbf{x}_i$ 是核心对象, 则称 $\mathbf{x}_j$ 由 $\mathbf{x}_i$ 密度直达 (通常不满足对称性)
- 密度可达(density-reachable): 对 $\mathbf{x}_i$ 和 $\mathbf{x}_j$ , 若存在样本序列 $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n$ , 其中 $\mathbf{p}_1 = \mathbf{x}_i$ ,  $\mathbf{p}_n = \mathbf{x}_j$ 且 $\mathbf{p}_{i+1}$ 由 $\mathbf{p}_i$ 密度直达, 则称 $\mathbf{x}_j$ 由 $\mathbf{x}_i$ 密度可达 (单侧)
- 密度相连(density-connected): 对 $\mathbf{x}_i$ 与 $\mathbf{x}_j$ , 若存在 $\mathbf{x}_k$ 使得 $\mathbf{x}_i$ 与 $\mathbf{x}_j$ 均由 $\mathbf{x}_k$ 密度可达, 则称 $\mathbf{x}_i$ 与 $\mathbf{x}_j$ 密度相连 (满足对称性)

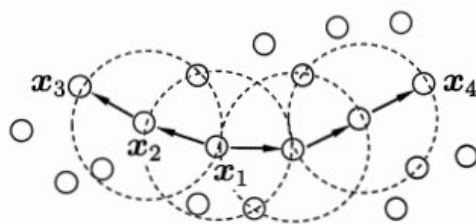


图 9.8 DBSCAN 定义的基本概念( $MinPts = 3$ ): 虚线显示出  $\epsilon$ -邻域,  $x_1$  是核心对象,  $x_2$  由  $x_1$  密度直达,  $x_3$  由  $x_1$  密度可达,  $x_3$  与  $x_4$  密度相连.

进而DBSCAN将“簇”定义为：由密度可达关系导出的最大密度相连样本集合，即簇  $C \subset \mathcal{D}$  是满足以下性质的非空样本子集：

- 连接性：  $x_i \in C, x_j \in C \implies x_i$  与  $x_j$  密度相连
- 最大性：  $x_i \in C, x_j$  由  $x_i$  密度可达  $\implies x_j \in C$

实际上，若  $\mathbf{x}$  为核心对象，由  $\mathbf{x}$  密度可达的所有样本构成的集合  $X = \{\mathbf{x}' \in \mathcal{D} \mid \mathbf{x}' \text{ 由 } \mathbf{x} \text{ 密度可达}\}$ ，则  $X$  为满足连接性与最大性的簇。注意对于DBSCAN来说，可能存在不属于任何簇的样本，这些样本被认为是噪声(noise)或异常(anomaly)样本。

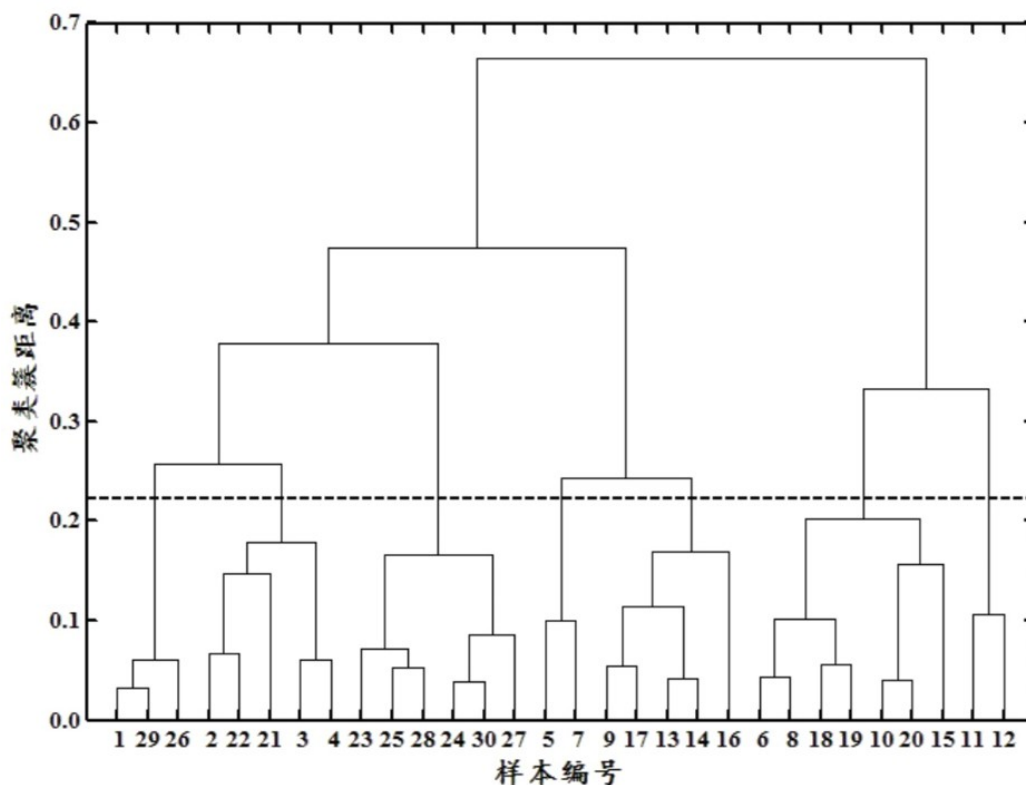
具体实现可先找出所有的核心对象，然后用队列维护进行扩展。

### 9.3 层次聚类

层次聚类试图在不同层次对数据集进行划分，从而形成树形的聚类结构。数据集划分既可采用“自底向上”的聚合策略，也可采用“自顶向下”的分拆策略。

AGNES算法（自底向上）：首先，将样本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距离最近的两个聚类簇进行合并，该过程不断重复，直到达到预设的聚类簇的个数。这里两个聚类簇  $C_i$  和  $C_j$  的距离，可以有3种度量方式：

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



## 10 降维与度量学习

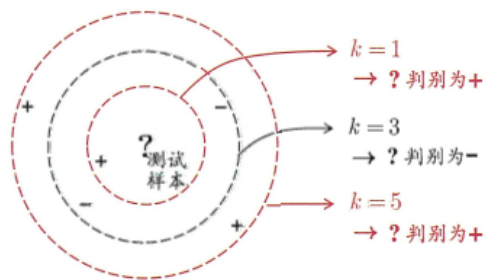
线性降维方法：对原始高维空间进行线性变换。给定 $d$ 维空间中的样本 $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m) \in \mathbb{R}^{d \times m}$ ，变换后得到 $d' \leq d$ 维空间中的样本 $Z = W^T X$ ，其中 $W \in \mathbb{R}^{d \times d'}$ 是变换矩阵， $Z \in \mathbb{R}^{d' \times m}$ 是样本在新空间中的表达。

### 10.1 k近邻学习

k近邻(k-nearest neighbor, kNN)是常用的监督学习方法。给定测试样本，基于某种距离度量找出训练集中与**最为靠近**的 $k$ 个训练样本，然后基于这 $k$ 个邻居的信息进行预测。在分类任务中可以采用**投票法**，在回归任务中可以采用**平均法**。

对于categorical属性，有序关系的可直接编码为数字，没有序关系的则要用独热码。快速实现kNN算法可以用KD树，快速筛选出前 $k$ 个距离最小的点。

kNN是懒惰学习(lazy learning)的著名代表，在训练阶段仅仅将样本保存起来，训练时间开销为0，待收到测试样本才进行处理。



这里距离的选择包括欧氏距离、闵可夫斯基距离、曼哈顿距离等。可以通过给每个特征/样本点加权来分配重要性。同时为避免属性之间的尺度不均，可以先做归一化 $(v_i - \min_j v_j) / (\max_j v_j - \min_j v_j)$ ，将特征值均映射到 $[0, 1]$ 的区间。

$k$ 太小则对噪声敏感， $k$ 太大则可能包含其他类别的很多点，经验法则取 $k < \sqrt{n}$ ， $n$ 为样本数目。

- 优点：直观易理解，可应用到不同分布的数据
- 缺点：分类时间长， $k$ 玄学选择，需要大规模样本

## 10.2 主成分分析

一般来说，想要获得低维子空间，最简单的方式是对原始高维空间进行线性变换。给定 $d$ 维空间中的样本 $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m) \in \mathbb{R}^{d \times m}$ 变换之后得到 $d' \leq d$ 维空间中的样本 $Z = W^T X$ ，其中 $W \in \mathbb{R}^{d \times d'}$ 为变换矩阵， $Z \in \mathbb{R}^{d' \times m}$ 是样本在新空间中的表达。基于线性变换来进行降维的方法称为线性降维方法，对低维子空间性质的不同要求可通过对 $W$ 施加不同的约束来实现。

主成分分析(Principle Component Analysis, PCA)希望找到这样的超平面具有这样的性质：

- 最近重构性：样本点到这个超平面的距离都足够近，对样本中心化 $\sum_i \mathbf{x}_i = \mathbf{0}$ ，再假定投影变换后得到的新坐标系为 $\{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_d\}$ ，其中 $\mathbf{w}_i$ 是标准正交基向量

$$\|\mathbf{w}_i\|_2 = 1, \mathbf{w}_i^T \mathbf{w}_j = 0 \ (i \neq j)$$

- 最大可分性：样本点在这个超平面上的投影尽可能分开

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

算法流程如下：输入样本集 $D = \{\mathbf{x}_i\}_{i=1}^m$ 和低维空间维数 $d'$

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 $X X^T$
3. 对协方差矩阵 $X X^T$ 做特征值分解
4. 取最大的 $d'$ 个特征值所对应的特征向量 $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}$
5. 输出投影矩阵 $W = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'})$

降维虽然会导致信息损失，但一方面舍弃这些信息后能使样本的采样密度增大，另一方面当数据受到噪声影响时，最小的特征值所对应的特征向量往往与噪声有关，舍弃可以起到去噪效果。

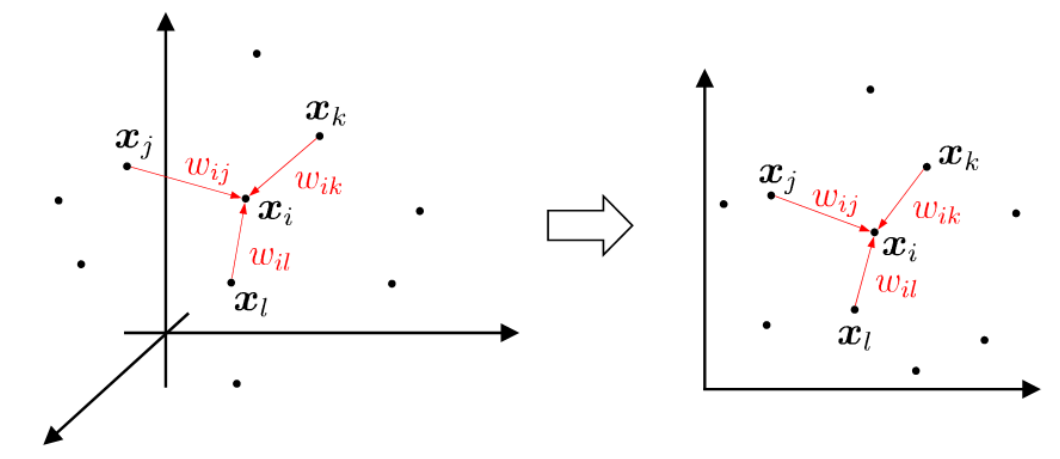
### 10.3 核化线性降维

线性降维方法假设从高维空间到低维空间的函数映射是线性的，然而，在不少现实任务中，可能需要非线性映射才能找到恰当的低维嵌入。

非线性降维的一种常用方法，是基于核技巧对线性降维方法进行核化(kernelized)。

### 10.4 流形学习

局部线性嵌入(Locally Linear Embedding, LLE)试图保持邻域内的线性关系<sup>2</sup>，并使得该线性关系在降维后的空间中继续保持。



$$\mathbf{x}_i = w_{ij}\mathbf{x}_j + w_{ik}\mathbf{x}_k + w_{il}\mathbf{x}_l$$

### 10.5 自编码器

自编码器(autoencoder)是一种前馈神经网络，输入 $\mathbf{x}$ ，预测 $\mathbf{x}$ 。最简单即单层的神经网络， $\tilde{\mathbf{x}} = W_2 W_1 \mathbf{x}$ ，计算平方误差

$$L(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2$$

将高维数据映射到二维来做可视化；以无监督方式学习抽象特征，进而可应用到有监督的任务中。

## 11 特征选择

特征选择是从原始特征中挑选出最有代表性、性能最好的特征，而特征提取则是用映射（或变换）的方法将原始特征变换维较少的新特征；降维则是通过原始特征构造新的特征，新的特征能够更好表示原始数据，且特征数量较原始特征少。

---

<sup>2</sup>与现在的GNN有异曲同工之妙

## 11.1 子集搜索与子集评价

特征选择的两个关键环节：子集搜索和子集评价

产生初始候选子集，评价候选子集的好坏，基于评价结果产生下一个候选子集。

- 前向搜索：逐渐增加相关特征
- 后向搜索：从完整特征集合开始，逐渐减少特征
- 双向搜索：每一轮逐渐增加相关特征，同时减少无关特征

## 11.2 特征选择方法

- 过滤型(filter)：先进行特征选择，再进行建模；评价指标：

– 皮尔逊相关系数：系数取值区间为 $[-1, 1]$ ， $-1$ 完全负相关， $+1$ 完全正相关， $0$ 则没有相关

$$r = \frac{\text{cov}(X, Y)}{\text{var}(X) \text{var}(Y)}$$

计算简单，但是不能反映变量之间的非线性关系

– 信息增益：特征子集 $A$ 确定了对数据集 $D$ 的一个划分， $A$ 上的取值将数据集 $D$ 分为 $V$ 份，每一份用 $D^v$ 表示， $\text{Ent}(D^v)$ 表示 $D^v$ 上的信息熵

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

- 封装型(wrapper)：直接将学习器的性能作为特征子集的评价标准，如召回率、准确率、AUC等指标。LVW (Las Vegas Wrapper)采用随机策略进行子集搜索，并以最终分类器的误差作为特征子集评价标准
  1. 在循环的每一轮随机产生一个特征子集
  2. 在随机产生的特征子集上通过交叉验证推断当前特征子集的误差
  3. 多次循环，在多个随机产生的特征子集中选择误差最小的特征子集作为最终解
- 嵌入型(embedded)：将特征选择和模型训练融为一体
  - 正则化方法：L1范数LASSO回归 $\lambda \|\mathbf{w}\|_1$ ，L2范数岭回归 $\lambda \|\mathbf{w}\|_2$
  - 随机森林/决策树：平均不纯度减少，平均精确率减少
  - 递归特征消除(Recursive Feature Elimination, RFE)：反复构建模型选出最好/最坏的特征，将选出来的特征放到一边，然后在剩余特征上重复这个过程，遍历所有特征。在这个过程中特征被消除的次序就是特征的排序。消除的稳定性很大程度取决于模型迭代时选用的底层模型。
- 无监督特征选择：Laplacian score选择那些能够最好保持数据流形结构的特征子集

数据和特征决定了机器学习的上限，而模型和算法意在逼近这个上限，特征工程就是最大限度地从原始数据中提取特征以供算法和模型使用。



## 12 半监督学习

让学习器不与外界交互、自动利用未标记样本来提升学习性能就是半监督学习<sup>3</sup>。

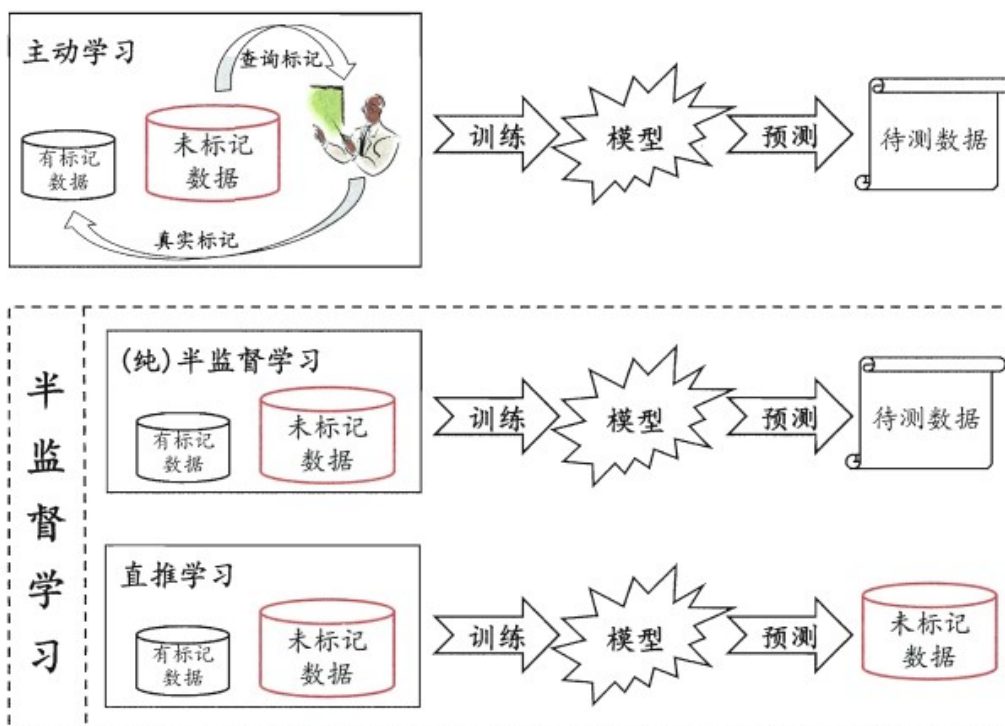


图 13.2 主动学习、(纯)半监督学习、直推学习

### 12.1 生成式方法

假设所有数据（无论是否有标记）都是由同一个潜在的模型生成的。

给定样本 $\mathbf{x}$ ，真实类别标记为 $y \in \mathcal{Y} = \{1, 2, \dots, N\}$ 。假设样本由高斯混合模型生成，且每个类别对应一个高斯混合成分，即数据样本基于以下概率密度生成

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

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

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

<sup>3</sup>如果是强化学习，则需要与外界进行交互获得反馈

验概率知

$$f(\mathbf{x}) = \arg \max_{j \in \mathcal{Y}} p(y = j | \mathbf{x})$$

$$= \arg \max_{j \in \mathcal{Y}} \sum_{i=1}^N p(y = j | \Theta = i, \mathbf{x}) \cdot p(\Theta = i | \mathbf{x})$$

用极大似然法可以估计参数，并用EM算法求解。

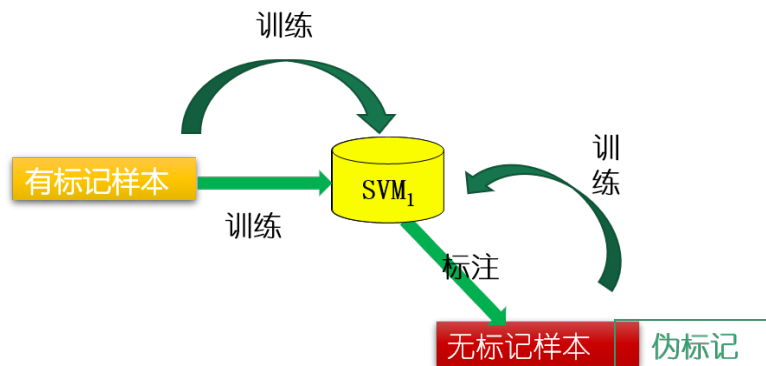
此类方法简单、易于实现，在有标记数据极少的情形下往往比其他方法性能更好。然而，此类方法有一个关键：模型假设必须准确，即假设的生成式模型必须与真实数据分布吻合；否则利用未标记数据反而会显著降低泛化性能。

## 12.2 半监督SVM

半监督SVM(Semi-Supervised SVM, S3VM)是SVM在半监督学习上的推广。S3VM试图找到能将两类有标记样本分开，且穿过数据低密度区域的划分超平面，采用的基本假设是低密度分隔(low-density separation)。

半监督SVM中最著名的是TSVM(Transductive SVM)。TSVM试图考虑对未标记样本进行各种可能的标记指派，尝试将每个未标记样本分别作为正例或反例，然后再所有这些结果中，寻求一个在所有样本上间隔最大化的划分超平面。一旦划分超平面得以确定，未标记样本的最终标记指派就是其预测结果。

TSVM采用局部搜索来迭代寻找近似解。



未标记样本进行标记指派及调整的过程中，有可能出现类别不平衡问题，即某类的样本远多于另一类。为了减轻类别不平衡性所造成的不利影响，可对算法稍加改进：将优化目标中的 $C_u$ 项拆分为 $C_u^+$ 与 $C_u^-$ 两项，并在初始化时令 $C_u^+ = \frac{u_-}{u_+} C_u^-$ 。

## 12.3 图半监督学习

给定一个数据集，我们可将其映射为一个图，数据集中每个样本对应于图中一个结点，若两个样本之间的相似度/相关性很高，则对应的结点之间存在一条边，边的强度(strength)正比于样本之间的相似度/相关性。

我们可将有标记样本所对应的结点想象为染过色，而未标记样本所对应的结点则尚未染色。于是，半监督学习就对应于“颜色”在图上扩散或传播的过程。

先基于标签数据与无标签数据的集合  $D_l \cup D_u$  建图  $G = (V, E)$ ，其中结点集

$$V = \{\mathbf{x}_1, \dots, \mathbf{x}_l, \mathbf{x}_{l+1}, \dots, \mathbf{x}_{l+u}\}$$

边集  $E$  可表示为一个亲和(affinity)矩阵，常基于高斯函数定义为

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

相似的样本应具有相似的标记,即得到最优结果于是可定义关于  $f$  的能量函数(energy function)。

## 12.4 半监督聚类

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

聚类任务中获得的监督信息大致有两种类型：

- 第一种类型是“必连”(must-link)与“勿连”(cannot-link)约束，前者是指样本必属于同一个簇，后者则是指样本必不属于同一个簇
- 第二种类型的监督信息则是少量的有标记样本

约束k均值(Constrained k-means)算法[Wagstaff et al., 2001]是利用第一类监督信息的代表。该算法是k均值算法的扩展，它在聚类过程中要确保“必连”关系集合与“勿连”关系集合中的约束得以满足，否则将返回错误提示。不冲突则选择最近的簇，冲突则尝试次近的簇。

第二类监督信息可假设少量有标记样本属于  $k$  个聚类簇。这样的监督信息利用起来很容易：直接将它们作为“种子”，用它们初始化  $k$  均值算法的  $k$  个聚类中心，并且在聚类簇迭代更新过程中**不改变**种子样本的簇隶属关系。这样就得到了约束种子k均值(Constrained Seed k-means)算法[Basu et al., 2002]。

## 13 推荐系统

$X$  为用户的集合， $S$  为推荐项的集合，效用函数  $u: X \times S \rightarrow R$ ， $R$  是评分的集合，全序。

## 14 参考资料

1. 周志华，《机器学习》，2016