

机器学习笔记

陈鸿峥

2020.06 *

目录

1	简介	2
1.1	分类	2
1.2	历史	3
2	基础概念	4
2.1	训练集与测试集	4
2.2	参数选择	4
2.3	性能度量	4
3	线性模型	5
3.1	线性回归	5
3.2	线性分类	6
3.3	线性判别分析	7
3.4	类别不平衡问题	7
4	决策树	8
4.1	信息增益	8
4.2	连续值处理	9
5	神经网络	10
5.1	感知机与单层神经网络	10
5.2	多层神经网络	11
5.3	深度学习	11
5.4	神经网络的历史发展	12

*Build 20200626

6	支持向量机	12
6.1	支持向量	12
6.2	核函数	14
6.3	支持向量回归	14
6.4	核方法	15
7	贝叶斯优化	15
8	集成学习	15
8.1	Boosting	16
8.2	Bagging	17
8.3	Stacking	18
9	聚类	18
9.1	原型聚类	18
10	降维与度量学习	19
10.1	k近邻学习	19
10.2	主成分分析	19
11	参考资料	19

本笔记对应周志华的《机器学习》(西瓜书)。

1 简介

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)



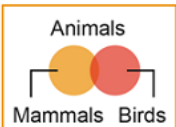

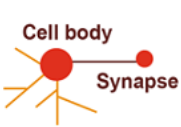


A look at

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 ~ 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
真实正	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

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

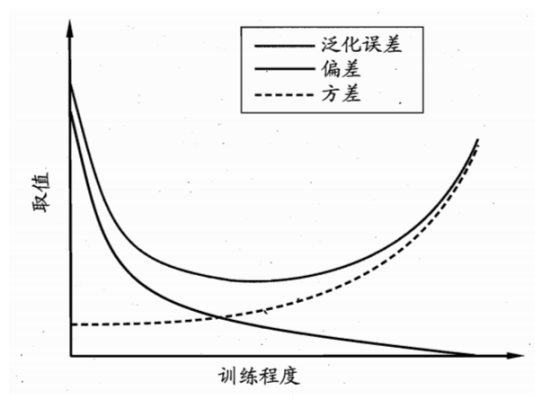
- 噪声(noise): 表达在当前任务上任何学习算法所能达到的期望泛化误差的下界

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

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 $\mathbf{w} \leftarrow [\mathbf{w} \ b]$ ，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。

将 y 视为后验概率估计，有

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

显然有

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

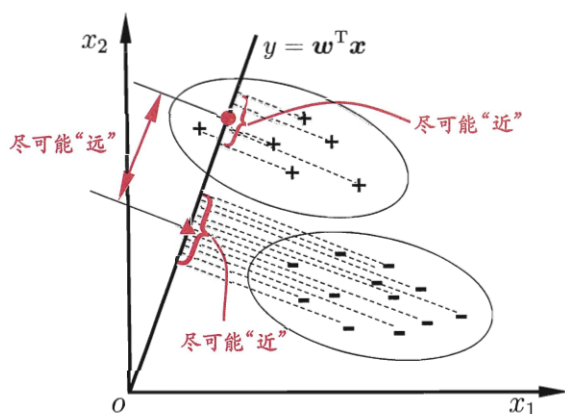
$$\mathbb{P}(y = 0 | \mathbf{x}) = \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x} + b}}$$

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

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

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。



最优化广义瑞利商

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

其中 S_w 为类内散度矩阵， S_b 为类间散度矩阵。

LDA也是经典的监督降维技术。

3.4 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

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

其中

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

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。

- 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。

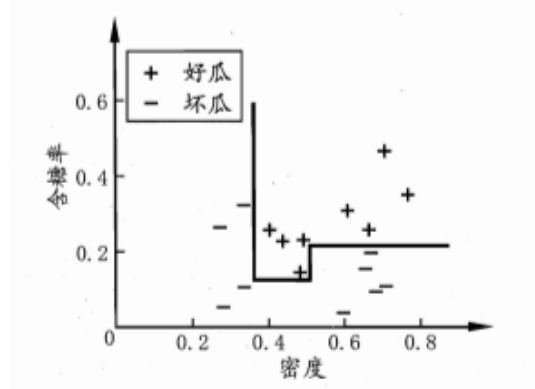
4.2 连续值处理

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。

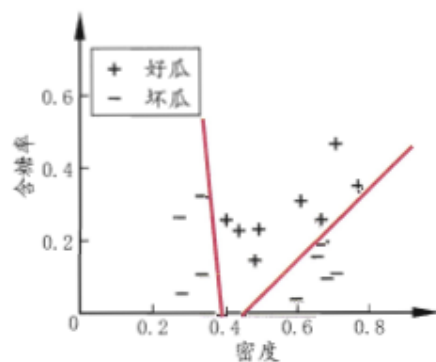
假定连续属性 a 在 D 上出现了 n 个不同的取值，将这些值从小到大排序记为 $\{a^1, a^2, \dots, a^n\}$ ，基于划分点 t 可以将 D 分为子集 D_t^- 和 D_t^+ ，进而可以考虑 $n - 1$ 个候选划分点集合

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

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。



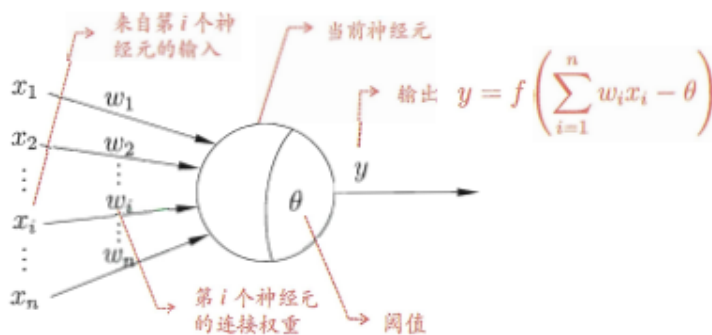
如果采用线性分类器，则变成多变量决策树，可以实现更加复杂的划分。



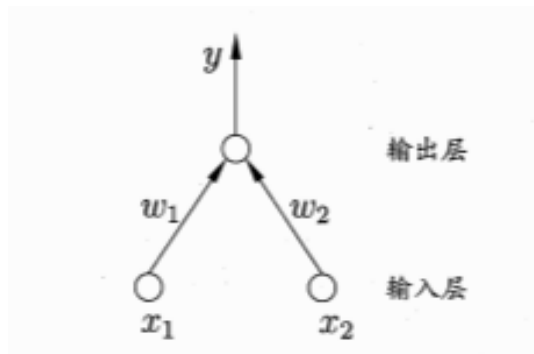
5 神经网络

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。

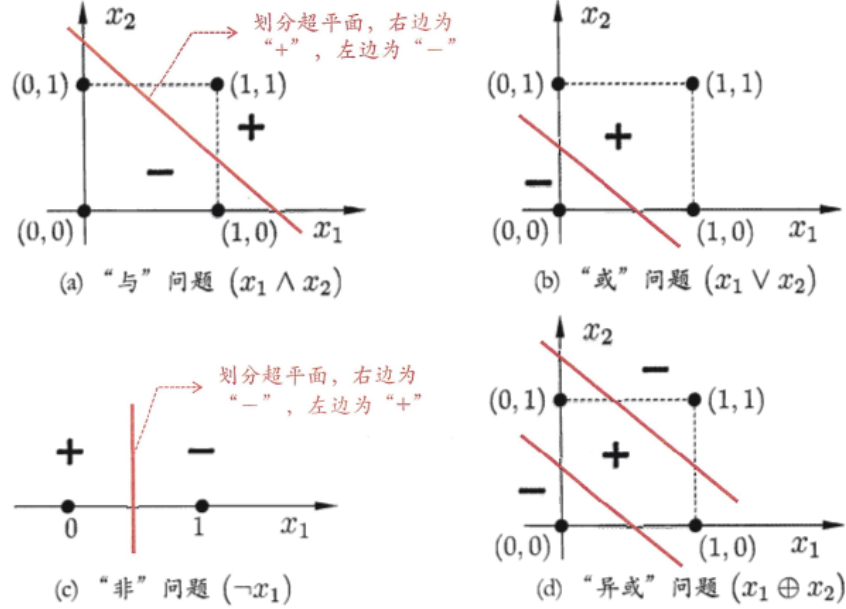
感知机的学习规则非常简单，对训练样例 (\mathbf{x}, y) ，如果当前感知机的输出为 \hat{y} ，则感知机的权重依照下式修改

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

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

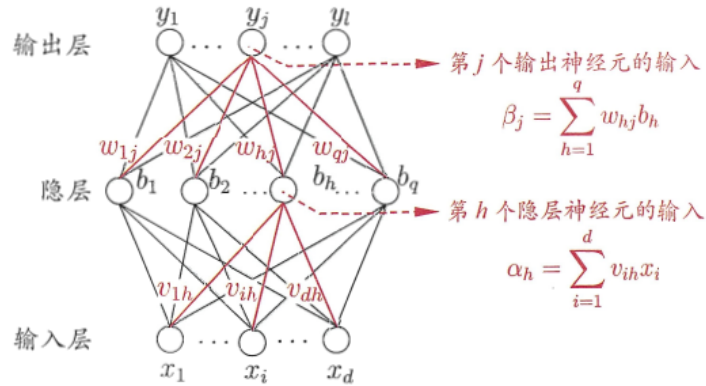
其中 $\eta \in (0, 1)$ 称为学习率(learning rate)。感知机预测正确则不发生变化，否则根据错误的程度对权重进行修改。

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。

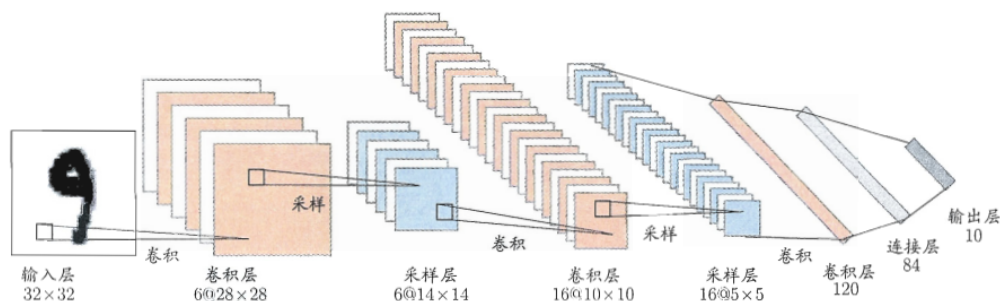
[Hornik, 1989]证明，只需一个包含足够多神经元的隐层，多层前馈神经网络就能以任意精度逼近任意复杂度的连续函数。然而，如何设置神经元的个数却没有固定的标准。

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。

5.4 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

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

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

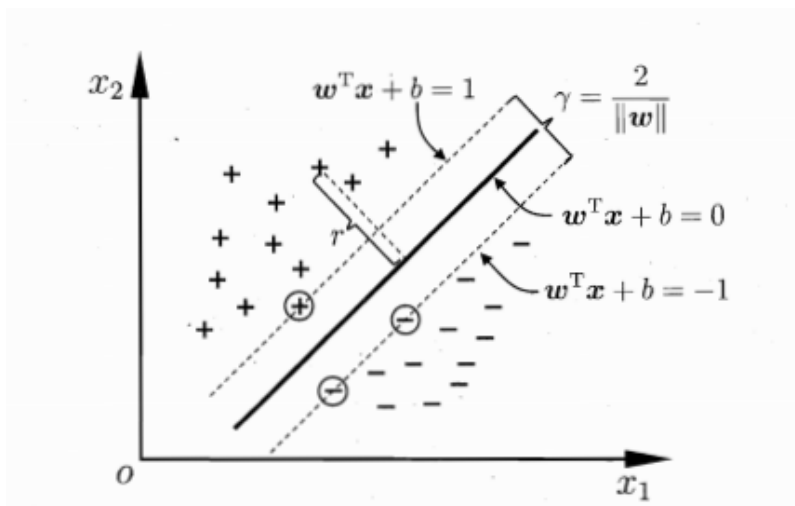
若超平面 (\mathbf{x}, b) 能将样本正确分类, 则

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

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。

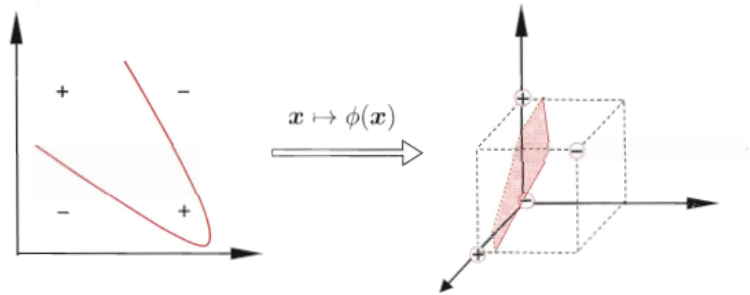
可以用拉格朗日乘子法求对偶问题并得到KKT条件, 可利用凸优化的方法进行求解。

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)

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

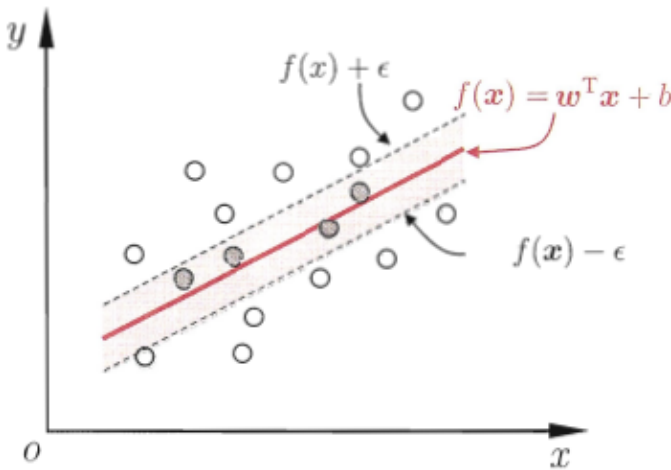
使得 \mathbf{x}_i 和 \mathbf{x}_j 在特征空间的内积等于它们在原始样本空间中通过核函数计算的结果。

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)

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 ϵ 的偏差



SVR问题可写为

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

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

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

6.4 核方法

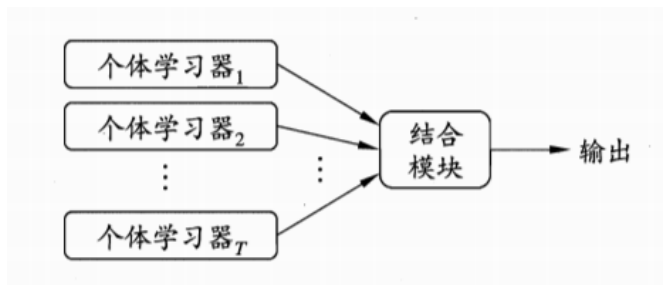
基于核函数的学习方法统称为核方法。最常见是通过“核化”（引入核函数），来将线性学习器扩展为非线性学习器。

7 贝叶斯优化

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），它的代表是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。

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

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

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

$$= \frac{\mathcal{D}_t(\mathbf{x}) \exp(-\alpha_t f(\mathbf{x}) h_t(\mathbf{x}))}{Z_t}$$
- 8: **end for**

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

图 1: AdaBoost算法

8.2 Bagging

Bagging是Bootstrap aggregating的缩写，用来生成多个分类器并且集成这些分类器形成一个整体模型。Bagging[Breiman, 1996]基于自助采样法（有放回的随机采样），采样出 T 个含 m 个样本的采样集 D_t ，然后基于每个采样集训练出一个基学习器 M_t ，再将这些基学习器结合。对未知样本 X 分类，每个分类器返回类预测，将得票最高的类赋予 X 。对于回归问题，则取每个分类器的预测值的平均值。

Bagging的时间复杂度低，与直接使用基学习器的复杂度同阶。个体学习器不存在强依赖关系，可以并行化生成。

随机森林[Breiman, 2001]是Bagging的一个扩展变体。传统决策树在选择划分属性时是在当前结点的属性集合（假定有 d 个属性）中选择一个最优属性；而在随机森林中则是对基决策树的每个结点，先从该结点的属性集合中随机选择一个包含 k 个属性的子集，然后再从这个子集中选择一个最优属性进行划分。这里的 k 即控制了随机程度，一般用 $k = \log_2 d$ [Breiman, 2001]。

随机森林中采用两到三层的随机性包括：

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 m 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。

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。

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。

贝叶斯模型平均(BMA)基于后验概率来为不同模型赋予权重，可视为加权平均的一种特殊实现。

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

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 μ_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。

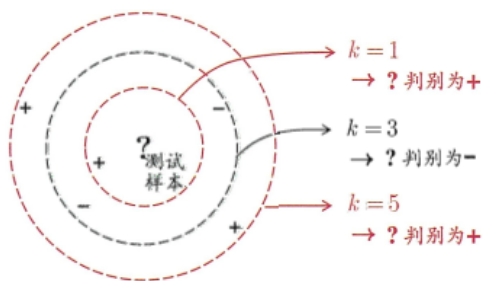
k邻近算法选择 k 个邻居，将其归为多的那个类别。对于categorical属性，有序关系的可直接编码为数字，没有序关系的则要用独热码。快速实现kNN算法可以用KD树，快速筛选出前 k 个距离最小的点。

10 降维与度量学习

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其最为靠近的 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。

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。



10.2 主成分分析

主成分分析(Principle Component Analysis, PCA)

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 XX^T
3. 对协方差矩阵 XX^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 $W = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'})$

11 参考资料

1. 周志华,《机器学习》, 2016