

机器学习复习笔记

机器学习复习笔记

绪论

模型评估与选择

经验误差与过拟合

模型选择

主要考察知识点

聚类

聚类算法任务

性能度量及距离计算

常见聚类方法

原型聚类

密度聚类

层次聚类

主要考察方向

回归

回归任务、线性回归、回归指标

最优化模型

多项式回归

欠拟合、过拟合、泛化能力

向量相关性

岭回归算法

Lasso 回归和 ElasticNet 回归

局部加权线性回归

K 近邻法

主要考察方向

分类

分类算法基础

逻辑回归模型

多分类逻辑回归

Softmax 回归模型

线性判别分析(LDA)

多分类学习

类别不平衡问题

主要考察方向

决策树

决策树分类算法

连续值与缺失值

预剪枝与后剪枝处理

回归树

多变量决策树

主要考察方向

贝叶斯分类

概率模型

朴素贝叶斯分类器

半朴素贝叶斯分类器

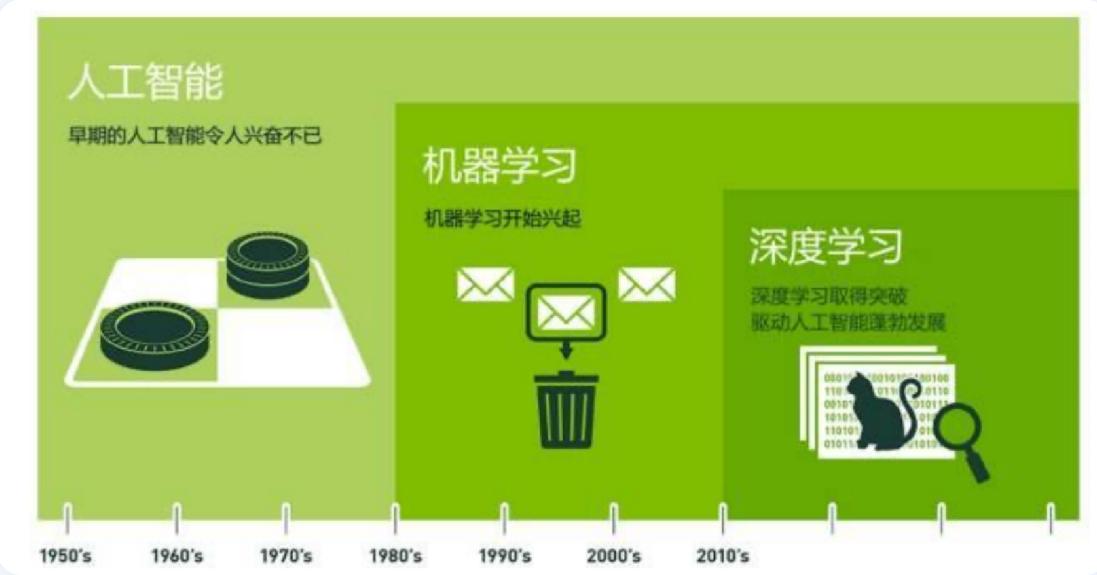
EM 算法

主要考察方向
集成学习
 个体与集成
 Boosting
 Bagging与随机森林
 结合策略
 多样性
支持向量机
 间隔与支持向量
 对偶问题
 软间隔
 核函数
 支持向量回归
 核方法
 主要考察方向
神经网络
 神经网络发展史
 神经元模型
 多层神经网络
 其他常见神经网络
深度学习
 概述
 卷积神经网络
 循环神经网络
结束语

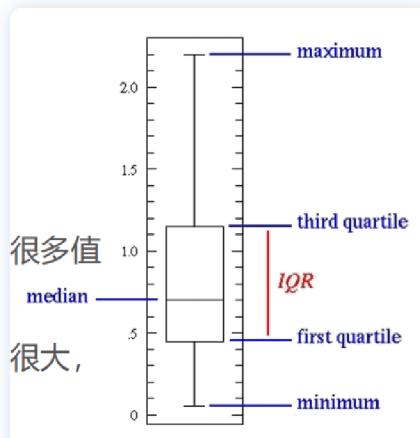
绪论

- 机器学习是什么
 - 机器学习与深度学习、神经网络的关系

传统神经网络是实现机器学习中分类、聚类、回归等模型的重要方法。后来人们发现改进的神经网络可以自动提取特征，从而发展为深度学习的分支



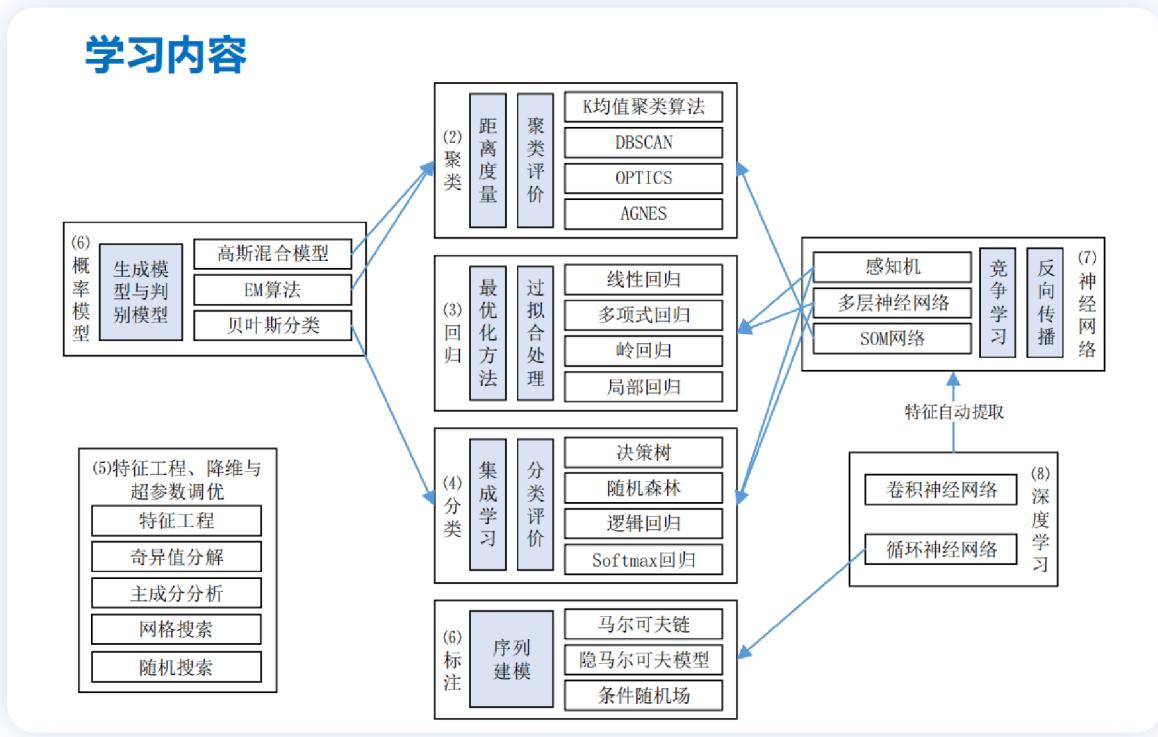
- 发展历程
- 机器学习算法
 - 监督学习、无监督学习、半监督学习
根据是否有标签的训练数据来学习一个模型
 - 聚类模型、分类模型、回归模型和标注模型
聚类模型用于将训练数据按照某种关系划分为多个族
分类模型用于将事物判定为预先设定的多个类别中的某一个
回归模型的标签值不再是离散的值了而是连续的
标注模型用于处理前后有关联的序列问题，输入一个观测序列，输出一个标签序列
 - 数据集统计



- 泛化能力
机器学习适用于新样本的能力称为泛化能力
- 归纳偏好

奥卡姆剃刀是一种常用的，选最简单的那个
没有免费午餐定理，脱离具体问题，空谈算法无意义

- 学习之路



模型评估与选择

经验误差与过拟合

- 错误率和误差

错误率：错分样本的占比

误差：训练误差：训练集上，测试误差：测试集，泛化误差：除训练集的所有样本

目标：选取泛化误差最小的学习器，但实际只能使经验误差最小

- 过拟合和欠拟合

把训练样本学的“太好了”和对训练样本的一般性质都没学好

模型选择

- 评估方法

- 留出法

直接将数据集划分为两个互斥集合

- 交叉验证法

将数据集分层采样划分为k个大小相似的互斥子集，每次用k-1个子集的并集作为训练集，余下的子集作为测试集，最终返回k个测试结果的均值

- 自助法

以自助采样法为基础，对数据集 D 有放回采样m次得到训练集D'，D/D'用做测试集

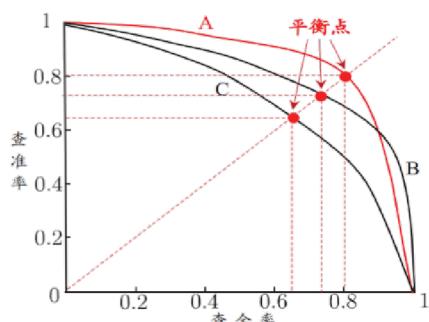
- 性能度量

- 回归任务中最常用的性能度量是“均方误差”
- 错误率：分错样本占总样本的比例
- 精度：分对样本占总样本的比例
- 查准率P、查全率R、F1

$$F1 = \frac{2 \times P \times R}{P + R} =$$

- P-R曲线

根据学习器的预测结果按正例可能性大小对样例进行排序，并逐个把样本作为正例进行预测，则可以得到查准率-查全率曲线，简称“P-R曲线”



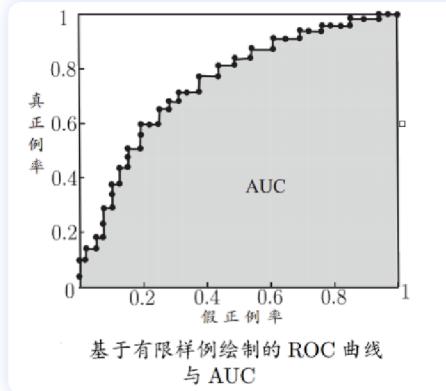
平衡点是曲线上“查准率=查全率”时的取值，可用来用于度量P-R曲线有交叉的分类器性能高低

P-R曲线与平衡点示意图

- ROC曲线

横轴：假正率 纵轴：真正例率

- AUC



The bigger, the better

- 比较检验

- 两学习器比较

交叉验证t检验

McNemar检验

- 多学习器比较

▪ Friedman检验

▪ Nemenyi后续检验

- 偏差与方差

- 通过实验可以估计学习算法的泛化性能

$$\begin{aligned}
 E(f; D) &= E_D[(f(x; D) - y_D)^2] \\
 &= E_D[(f(x; D) - \bar{f}(x))^2] + (\bar{f}(x) - y)^2 + E_D[(y - y_D)^2] \\
 &= bias^2(x) + var(x) + \varepsilon^2
 \end{aligned}$$

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

- 偏差度量了学习算法期望预测与真实结果的偏离程度；即刻画了学习算法本身的能力；
- 方差度量了同样大小训练集的变动所导致的学习性能的变化；即刻画了数据扰动所造成的影响；
- 噪声表达了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；即刻画了学习问题本身的难度。

- Bias-variance dilemma

一般而言，偏差与方差存在冲突：

- 训练不足时，学习器拟合能力不强，偏差主导
- 随着训练程度加深，学习器拟合能力逐渐增强，方差逐渐主导
- 训练充足后，学习器的拟合能力很强，方差主导

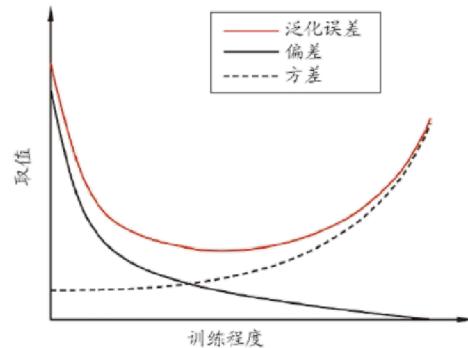


图 2.9 泛化误差与偏差、方差的关系示意图

主要考察知识点

根据往年考察题目的观察，主要考察性能度量的相关计算，根据语意进行计算

查全率: 真实正例被预测为正例的比例

真正例率: 真实正例被预测为正例的比例

显然查全率与真正例率是相等的。

查准率: 预测为正例的实例中真实正例的比例

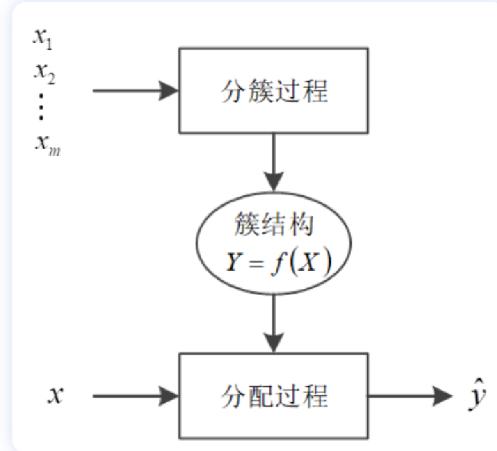
假正例率: 真实反例被预测为正例的比例

- 如何绘制 ROC 曲线
 - 根据学习器的预测结果进行排序
 - 对所有结果进行遍历，如果为正例则对 Y 轴加 $1/m$ ，如果为负例则对 X 轴加 $1/n$ ，(m 为正例个数， n 为负例个数)
 - 连起来所有的点

聚类

聚类算法任务

聚类算法在“无监督学习”任务中研究最多，应用最广，将样本划分为若干个不相交的子集，可以作为一个单独的过程，同样可以作为学习任务的前驱过程



性能度量及距离计算

- 聚类性能度量

- 外部指标

将聚类模型与参考模型进行比较

- 内部指标

直接考察聚类结果而不用任何参考模型

- 距离计算

聚类得到的划分为 $C = \{C_1, C_2, \dots, C_k\}$, 参考模型给出的簇划分为 C^* ,

令 λ 与 λ^* 分别表示 C 与 C^* 的簇标记向量

$$a = |SS| = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}$$

$$b = |SD| = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}$$

$$c = |DS| = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}$$

$$d = |DD| = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}$$

Jaccard系数 (0, 1区间内越大越好)

$$JC = \frac{a}{a + b + c} ([0, 1] \text{区间内, 越大越好})$$

FM指数 (0, 1区间内越大越好)

$$FMI = \sqrt{\frac{a}{a + b} \cdot \frac{a}{a + c}} ([0, 1] \text{区间内, 越大越好})$$

Rand指数 (0, 1区间内越大越好)

$$RI = \frac{2(a + d)}{m(m - 1)} ([0, 1] \text{区间内, 越大越好})$$

DB指数 (越小越好)

Dunn指数 (越大越好)

■ 各类距离

$$\text{簇内样本平均距离: } \text{avg}(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \leq i \leq j \leq |C|} \text{dist}(x_i, x_j)$$

$$\text{簇内样本最远聚类: } \text{diam}(C) = \max_{1 \leq i \leq j \leq |C|} \text{dist}(x_i, x_j)$$

$$\text{簇间最近距离: } d_{min}(C) = \min_{x_i \in C_i, x_j \in C_j} \text{dist}(x_i, x_j)$$

$$\text{簇间中心点距离: } d_{cen}(C) = \text{dist}(\mu_i, \mu_j)$$

■ 轮廓系数 (SC)

SC值高表示自身簇聚类匹配较好，与其他簇匹配较差，即簇内密集，簇间疏散。

■ 常用距离

闵可夫斯基距离：

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

p=2:欧式距离

p=1:曼哈顿距离

- VDM处理无序属性，例如{1,2,3}这样的为有序属性，{飞机，火车，轮船}这样的为无序属性
- MinkovDMP处理混合属性
- 余弦相似度

$$\begin{aligned} \cos \theta &= \frac{x_i \cdot x_j}{\|x_i\| \cdot \|x_j\|} \\ &= \frac{\sum_{l=1}^n x_i^{(l)} x_j^{(l)}}{\sqrt{\sum_{l=1}^n (x_i^{(l)})^2} \cdot \sqrt{\sum_{l=1}^n (x_j^{(l)})^2}} \end{aligned}$$

常见聚类方法

原型聚类

通常情况下，算法先对原型进行初始化，再对原型进行迭代更新求解

- K均值聚类

K均值聚类算法的基本思想是让簇内的样本点更“紧密”一些，也就是说，让每个样本点到本簇中心的距离更近一些。使每个样本点到本簇中心的距离的平方和尽量小就是k-means算法的优化目标。每个样本点到本簇中心的距离的平方和也称为误差平方和 (SSE)，在优化算法中称为损失函数或代价函数。

算法流程

步数	操作
1	随机产生k个初始簇中心（或者随机选择k个点作为初始簇中心）
2	对每一点，计算与所有簇中心的距离，将其分配到最近的簇
3	如果没有点发生分配结果改变，则结束，否则继续下一步
4	计算新的簇中心
5	跳到第2步

- 学习向量量化(LVQ)

与一般聚类算法不同的是，LVQ假设数据样本带有类别标记，学习过程中利用样本的这些监督信息来辅助聚类。给定样本集，LVQ的目标是学得一组n维原型向量，每个原型向量代表一个聚类簇。

输入: 样本集 $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$;
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; $i^* = \arg \min_{i \in \{1, 2, \dots, q\}} d_{ji}$;
- 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
- 13: **return** 当前原型向量

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

- 高斯混合聚类

高斯混合聚类采用概率模型来表达聚类原型

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 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leq i \leq k\}$
- 2: **repeat**
- 3: **for** $j = 1, \dots, m$ **do**
- 4: 根据(9.30)计算 \mathbf{x}_j 由各混合成分生成的后验概率, 即
 $\gamma_{ji} = p_M(z_j = i \mid \mathbf{x}_j)$ ($1 \leq i \leq k$)
- 5: **end for**
- 6: **for** $i = 1, \dots, k$ **do**
- 7: 计算新均值向量: $\boldsymbol{\mu}'_i = \frac{\sum_{j=1}^m \gamma_{ji} \mathbf{x}_j}{\sum_{j=1}^m \gamma_{ji}}$;
- 8: 计算新协方差矩阵: $\boldsymbol{\Sigma}'_i = \frac{\sum_{j=1}^m \gamma_{ji} (\mathbf{x}_j - \boldsymbol{\mu}'_i)(\mathbf{x}_j - \boldsymbol{\mu}'_i)^\top}{\sum_{j=1}^m \gamma_{ji}}$;
- 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 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leq i \leq k\}$ 更新为 $\{(\alpha'_i, \boldsymbol{\mu}'_i, \boldsymbol{\Sigma}'_i) \mid 1 \leq i \leq k\}$
- 12: **until** 满足停止条件
- 13: $C_i = \emptyset$ ($1 \leq i \leq k$)
- 14: **for** $j = 1, \dots, m$ **do**
- 15: 根据(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**
- 18: **return** 簇划分结果

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

• 进一步讨论

■ 簇中心的确定

求总SSE, 可以按簇计算SSE, 然后求和。

要使总SSE最小, 只需要每簇SSE最小, 因此, 计算簇中心就是要求使得簇内SSE最小的那个点。

■ 算法复杂度

设样本总数为 m , 分簇数为 k 。一次迭代过程中, 以样本与簇中心点距离计算为基本运算, 需要 mxk 次。如果迭代次数为 t 次, 则算法的时间复杂度是 $O(mkt)$

■ 局部最优与全局最优

如果初始点选不好, 就会陷入局部最优解, 而无法得到全局最优解, 这是因为SSE是所谓的非凸函数

■ K值的确定

根据SSE值来确定, 在大于一定分簇数时, SSE值趋于相对稳定。

■ 特征归一化

$$Standard(x_i^{(j)}) = \frac{x_i^{(j)} - \min x^{(j)}}{\max x^{(j)} - \min x^{(j)}}$$

• 改进算法

■ 二分k-means算法

步数	操作
1	将所有样本点放在一个簇内
2	当簇数目小于k时，对每一个簇
2. 1	计算SSE
2. 2	进行簇数目为2的k-means试分簇，并对分裂后的两个簇分别计算SSE
2. 3	计算减少的SSE值，并记录减少最多的SSE值
3	对记录到减少最多SSE值的簇进行分裂，跳到第2步

- k-means++算法

步数	操作
1	从样本集 \mathbf{S} 中随机选择1个样本点加入簇中心集合 U 中
2	对任一样本点 x ，计算它到 U 的距离 $D(x)$
3	将 $D(x)$ 转化为对应样本点 x 的概率 $p(x)$
4	按所有样本点的概率 $p(x)$ ，选择一个样本点加入簇中心集合 U
5	重复2、3、4步直至簇中心集合元素个数达到 k

密度聚类

密度聚类是通过密度进行分簇，这里的密度是指某样本点给定领域内其它样本点的数量。

- 密度相关的概念和定义

ϵ – 邻域

$$N_\epsilon(x_i) = \{x_j | dist(x_j, x_i) \leq \epsilon\}$$

- 核心点和边界点

核心点是领域中样本点数量不少于Minpts的点，相反的称为边界点

- 直接密度可达

若 x_i 位于 x_j 的领域内，且 x_i 是核心点，则称 x_j 可由 x_i 直接密度可达

- 密度可达

密度可达是由直接密度可达多次传递得到。

- 密度相连

对于点x和y，若存在点O，使得两点均可由O密度可达，则称两点密度相连

- DBSCAN算法

步数	操作
1	初始化当前簇号为0，初始化样本集内所有样本点的标签为noise，初始化种子集合seeds为空集合
2	从样本集中依序取一个标签为noise的点 x_i ，如果没有这样的点，则算法结束
3	检查 x_i 是否核心点，如果不是则转第2步
4	将 x_i 邻域内的所有标签为noise的点标记上当前簇号，并加入集合seeds（不包括 x_i ）
5	如果集合seeds为空，将当前簇号加1，并转第2步，否则从集合seeds中取一个点p
6	如果点p为非核心点，则从集合seeds中删除，并转第5步
7	将点p的邻域内的所有标签为noise的点标记上当前簇号，并入seeds集合，转第5步

58

- 领域参数和Minpts的确定

设k_dist为某点p到离它第k近的那个邻居点点距离，可以计算每个点的k_dist值，将他们按照大小进行排序作图，根据拐点来确定epsilon，将epsilon设为拐点的k_dist值，Minpts设为k。

- OPTICS

1. 点 x_i 的核心距离

只有核心点才有核心距离，且核心距离是使该点成为核心点的最小距离

2. 点 x_i 到点 x_j 的可达距离

只有核心点才能被可达，且可达距离是该核心点的核心距离或者两点间距离的最大值。

计算可达距离流程

OPTICS算法的核心思想是将每个点离最近聚集密集区的可达距离都计算出来，然后据此进行分簇。

步数	操作
1	设置领域半径 ϵ ，核心点最小领域点数 $MinPts$ ，初始化样本集内所有样本点的可达距离为极大值 inf ，创建两个队列：有序队列和结果队列
2	从样本集中选取一个核心点 x_i ，放入结果队列，如果没有这样的点，则算法结束
3	找到 x_i 的所有直接密度可达点，如果不在结果队列中，则计算其与 x_i 的可达距离，交放入有序队列，将有序队列按可达距离值从小到大排序
4	如果有序队列为空，则跳至第2步，否则，从有序队列中取出第1个点 P 放入结果队列，并对点 P ： 4.1 如果点 P 为非核心点，跳至第4步，否则取出该点的所有直接密度可达点 4.2 如果点 P 的直接密度可达点已经在结果队列中，跳至第4步 4.3 如果点 P 的直接密度可达点不在有序队列中，则计算与点 P 的可达距离，加入有序队列，并对有序队列按可达距离值从小到大排序，跳至第4步 4.4 如果点 P 的直接密度可达点已在有序队列中，则计算与点 P 的可达距离，如果新的可达距离小于原可达距离，则更新该点的可达距离，并重新排序，跳至第4步

层次聚类

层次聚类强调的是聚类执行的过程，分为自底向上的凝聚方法和自顶向下的分裂方法，凝聚方法是先将每一个样本点当成一个簇，然后逐步合并，分裂方法是将所有样本点放到一个簇内，然后逐步分解。

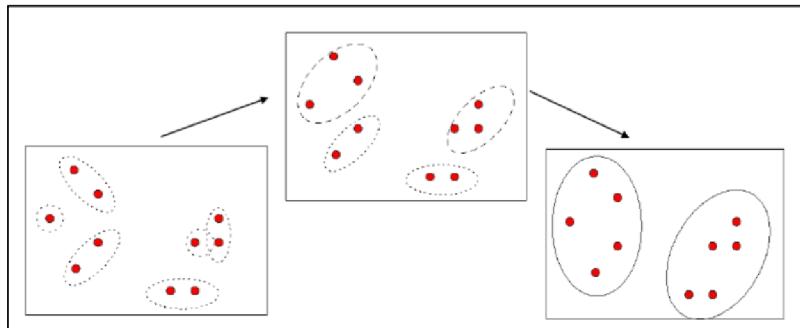
- **AGNES 算法（自底向上）**

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。

Step1: 将每个样本点作为一个簇

Step2: 合并最近的两个簇

Step3: 若所有样本点都存在与一个簇中，则停止；否则转到 Step2



- 网格聚类

强调的是分批统一处理，具体的做法是将特征空间划分为若干个网格，将网格内的样本点看做一个单元进行处理。

- 其他常见聚类方法

- K 均值聚类
- FCM
- LVQ
- 密度聚类：DBSCAN、OPTICS等
- 层次聚类：AGNES、DIANA 等

主要考察方向

- 常见聚类方法

- k均值聚类，学习向量量化，高斯混合聚类
- DBSCAN
- AGNES

回归

回归任务、线性回归、回归指标

基于损失函数最小的思想，学习得到一个模型，该模型是从实例特征向量到实数的映射

$$f(x) = W \cdot x^T + b$$

- 回归评价指标

- 残差：

$$s = |y_i - f(x_i)|$$

- 残差(误差)平方：

$$s_i^2 = (y_i - f(x_i))^2$$

- 误差平方和(SSE)

$$L(W) = s_1^2(W) + s_2^2(W) + \cdots + s_m^2(W)$$

- 均方误差(MSE)

$$MSE = \frac{L(W)}{m}$$

最优化模型

- 数学模型

$$\min_{x \in R} f(x) \quad s.t. \begin{cases} h_i(x) = 0 \\ g_i(x) \leq 0 \end{cases}$$

s.t.为subject to的缩写

- 迭代法

迭代法的核心是建立迭代关系式

- 先将方程式 $f(x)=0$ 变换为 $x=g(x)$
- 建立迭代关系式 $x_{k+1}=g(x_k)$
- 计算 x_k ,若收敛于 x^* ,那么 x^* 就是方程的跟

- 梯度下降法

$$x_{i+1} = x_i - \alpha \cdot \frac{df(x)}{dx} \Big|_{x=x_i}$$

- 结束条件：一般是迭代次数达到了最大设定，损失函数低于设定的阈值
- 步长的设定
- 特征归一化

$$\text{损失函数: } L(W) = \frac{1}{2} \sum_{i=1}^m s_i^2(W)$$

$$\text{回归系数更新: } w_{l+1}^{(j)} = w_i^{(j)} + \alpha \frac{dL(W)}{dw^{(j)}}$$

- 全局最优

Hessian矩阵

$$f(x) = f(x_0) + \left(\frac{\partial f}{\partial x^{(1)}}, \frac{\partial f}{\partial x^{(2)}} \right)_{x_0} \cdot \begin{pmatrix} \Delta x^{(1)} \\ \Delta x^{(2)} \end{pmatrix} + \dots$$

Hessian矩阵判断多元函数的极值

设 $f(x)$ 在 x_0 点处二阶连续可导，且有 $\nabla f(x_0) = 0$ ，那么：①当 $G(x_0)$ 是正定矩阵时， $f(x)$ 在 x_0 点处是极小值；②当 $G(x_0)$ 是负定矩阵时， $f(x)$ 在 x_0 点处是极大值；③当 $G(x_0)$ 是不定矩阵时， x_0 不是极值点；④当 $G(x_0)$ 是半正定矩阵或半负定矩阵时， x_0 点是可疑极值点。

求三元函数 $f(x, y, z) = x^2 + 2xy + 2y^2 + z^2 + 6x$ 的极值。

令各变量的梯度为0： $\frac{\partial f}{\partial x} = 2x + 2y + 6 = 0$ ， $\frac{\partial f}{\partial y} = 2x + 4y = 0$ ， $\frac{\partial f}{\partial z} = 2z = 0$ ，可得驻点 $(-6, 3, 0)$ 。

因此，Hessian矩阵为 $\begin{pmatrix} 2 & 2 & 0 \\ 2 & 4 & 0 \\ 0 & 0 & 2 \end{pmatrix}$ ，为正定矩阵，故该驻点是极小值点，极小值为 $f(-6, 3, 0) = -18$ 。

35

- 凸优化

凸集的定义：集合内 S 中任意的两点的连线上的点都在 S 内，则称集合 S 为凸集

凸函数：曲线上的任意两点连线上的所有点都在曲线上方

凸函数中局部最优点就是全局最优点

- 凸函数的判定

- 切线位于曲线下方
- 二阶导数大于等于0

- 牛顿法

用所谓的切线来建立迭代关系

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

.....

多项式回归

多项式回归，研究一个因变量与一个或多个自变量间多项式关系的回归分析方法。

- 转换为线性回归问题来解决

$$y_1 = x, y_2 = x^2, \dots$$

欠拟合、过拟合、泛化能力

- 训练集、验证集、测试集

训练集的数据要尽可能充分且分布平衡

- 泛化能力评估方法

- 留出法
- K-折交叉验证

- 奥卡姆剃刀定律：越简单越好

- 过拟合抑制

- 正则化方法：加一个正则化项，模型越复杂，值越大

- L2 正则化方法：给它加一个参数向量 W 的 L2 范数

采用 L2 范数和 L1 范数正则项的线性回归，分别称为岭回归和Lasso回归

- 早停法：在每一轮训练完后，就用验证集来验证泛化能力，如果n轮训练后泛化能力都没有提高，就停止训练。

- 随机失活：随机失活应用于神经网络的过拟合抑制，它通过随机使一部分神经元临时失效来达到目的。

向量相关性

用验证集的预测值组成的向量与实际标签值组成的向量之间的相关性来衡量算法的有效性

- 协方差

$$Cov(X, Y) = E\{[X - E(X)] \cdot [Y - E(Y)]\}$$

变化趋势相近时，协方差大于 0，如果相反，则小于 0，独立时，为 0

- 相关系数与相关距离

$$\rho_{XY} = \frac{Cov(X, Y)}{\sqrt{D(X)} \sqrt{D(Y)}}$$

岭回归算法

岭回归算法是在原线性回归的损失函数上增加 L2 正则项

$$L' = L(W) + \lambda W W^T$$

通过权重对系数进行了衰减，抑制了相关特征的系数发生过大变化，迫使它们向 0 趋近，与简单线性回归相比能取得更好的预测效果。

Lasso 回归和 ElasticNet 回归

Lasso 回归和岭回归的区别在于 Lasso 回归的惩罚项是基于 L1 范数

ElasticNet 回归是对 Lasso 回归和岭回归的融合，惩罚项是 L1 和 L2 项的融合

$$\lambda \left(\frac{1-\rho}{2} \|W\|_2^2 + \rho \|W\|_1 \right)$$

局部加权线性回归

局部加权线性回归模型根据训练样本点与预测点的远近设立权重，离预测点越近的点的权重就越大

K 近邻法

K 近邻法是一种简单而基本的机器学习方法，可用于求解分类和回归问题

K 近邻法用于回归分为两步

1. 根据 d ，找出 k 个距离 x 最近的样本，得到 x 的邻域 N
2. 计算 $v(N)$ 得到 x 的标签值

d 常用欧式距离， v 常用求均值函数

主要考察方向

- 计算 L1 范数和 L2 范数
 - L1 范数：向量中各元素的绝对值之和
 - L2 范数：向量中各元素的平方和然后求平方根
- K 近邻法考察过计算题
注意其计算步骤

分类

分类，就是将某个事物判定为属于预先设定的有限个集合中的某一个的过程。

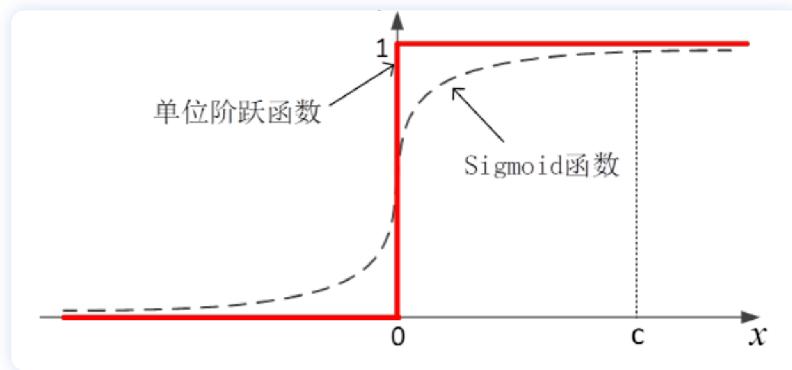
分类算法基础

- 平面上二分类的线性逻辑回归

通过一个线性方程来对样本点进行划分：

$$f_W(x) = w^{(0)} \cdot x^{(0)} + w^{(1)} \cdot x^{(1)} + \dots = W \cdot x = 0$$

- 单位阶跃函数



- 损失函数

$$L(W) = \sum_i |y_i - \hat{y}_i|$$

- 梯度下降法求解

$$W_{i+1}^{(j)} = W_i^{(j)} - \alpha \frac{\partial L}{\partial W_i^{(j)}}$$

逻辑回归模型

- 逻辑回归求解的步骤

- 确定一个合适的未知参数的预测函数（通过对样本的观察分析）
- 根据预测函数设计损失函数
- 通过某种方法从损失函数求出预测函数的未知参数

- 采用sigmoid函数作为分类函数

$$L_i(W) = \begin{cases} -\ln(\hat{y}_i), & \text{if } y_i = 1 \\ -\ln(1 - \hat{y}_i), & \text{if } y_i = 0 \end{cases}$$

- 二分类逻辑回归未知样本的判定

$$P(y = 1|x) = \frac{1}{1 + e^{-f_{\hat{W}}(x)}}$$

- 发生比

样本取正类和负类概率的比值称为发生比: $\frac{P(y = 1|x)}{P(y = 0|x)} = e^{f_{\hat{W}}(x)}$

多分类逻辑回归

一对一：两两配对训练模型，在预测时将实例提交有关所有模型，最后投票其类别

一对其余：将一个看为正类，其余看为负类，训练出k个模型

多对多：将若干类作为正类，其余作为负类

- 多分类输出概率

$$\begin{aligned} P(y = 1|x) &= P(y = K|x)e^{f_{W_1}(x)} \\ &\dots \\ P(y = K-1|x) &= P(y = K|x)e^{f_{W_{K-1}}(x)} \\ P(y = K|x) &= \frac{1}{\sum_{i=1}^{K-1} e^{f_{W_i}(x)} + 1} \end{aligned}$$

- Softmax函数

softmax函数将每一个分类结果转化为一个概率值：

$$p_k = \frac{e^{y_k}}{\sum_{i=1}^n e^{y_k}}, k = 1, 2, \dots, K$$

softmax还具有其归一化的功能，在神经网络中得到了非常广泛的应用

Softmax 回归模型

- 分类预测函数

$$p_k(x) = \frac{e^{W_k \cdot x}}{\sum_{k=1}^K e^{W_k \cdot x}}$$

- 损失函数

$$L(W) = - \sum_{i=1}^m \sum_{k=1}^K I(y_i = y_k) \ln \frac{e^{W_k \cdot x}}{\sum_{k=1}^K e^{W_k \cdot x}}$$

- 梯度下降法求解

$$W_{j+1} = W_j - \alpha \cdot \frac{\partial L(W)}{\partial W_j}$$

- Softmax回归与逻辑回归的关系

Softmax回归可看作二分类逻辑回归在多分类问题上的推广

线性判别分析(LDA)

同类样例的投影点尽可能接近，让同类样例投影点的协方差尽可能小

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

- 最大化目标

$$J = \frac{w^T(\mu_0 - \mu_1)(\mu_0 - \mu_1)^Tw}{w^T(\sum_0 + \sum_1)w}$$

多分类学习

多分类学习方法，将二分类学习方法推广到多类

- 一对一
 - N 个类别两两配对
 - 各个二类任务学习分类器
 - 投票产生最终分类结果
- 一对其余
 - 某一类作为正例，其他反例
 - 各个二类任务学习分类器

- 置信度最大类别作为最终类别
- **两种策略比较**
 - 一对一：存储开销和测试时间大，训练时间短
 - 一对其余：存储开销和测试时间小，训练时间长
- 多对多

纠错输出码(ECOC):

- 海明距离：比较样本与类的编码，不一致的个数就是海明距离
- 欧氏距离：样本与类的差的平方和再加根号

类别不平衡问题

- 欠采样法
直接对训练集中的过多样本去掉一部分
- 过采样法
增加一些样本，简单的复制插值等
- 阈值移动法
根据正负样本的数量对阈值进行移动

主要考察方向

- 训练逻辑回归使用的方法
极大似然估计

决策树

决策树分类算法

- **基本思想**
生活中常用决策树的思想来做决定。
- **决策树模型**
 - 决策树是一种对测试样本进行分类的树形结构，该结构由结点和有向边组成。
 - 内部结点表示对样本的一个特征进行测试
 - 叶结点表示样本的一个分类
- **建立二叉决策树流程**

步数	操作
1	对输入的训练集，如果集合为空，算法结束
2	如果不能选择到一个合适的特征及其决策值，则建立叶子节点，算法结束
3	根据选择到的特征及其决策值，建立内部节点
4	依据选择到的特征及其决策值将输入的训练集划分为左、右两个子集，对每个子集应用本算法

- 信息量

信息就是对不确定性的消除

因此预言以往发生小概率的事件的消息所带来的信息量就要大，以往发生的概率叫做先验概率，用 p 表示

$$\text{信息量公式: } I(x) = \log\left(\frac{1}{p}\right) = -\log p$$

- 信息熵

把信源发出的所有事件的信息量求平均值，就可以刻画信源消除的平均不确定性，定义为信息熵

$$Ent(X) = E[I(x_i)] = - \sum_{i=1}^n p_i \log_2 p_i$$

样本集合的信息熵越大，说明各样本相对均衡，区别度就越小

- 信息增益

将样本集划分为两个的信息熵，按样本比例作加权的和：

$$Ent(D, F^{(j)} = f) = \frac{|D_1|}{|D|} Ent(D_1) + \frac{|D_2|}{|D|} Ent(D_2)$$

划分前后信息熵的减少量称为信息增益

$$\begin{aligned}
Gain(D, F^{(j)} = f) &= Ent(D) - Ent(D, F^{(j)} = f) \\
&= Ent(D) - \left(\frac{|D_1|}{|D|} Ent(D_1) + \frac{|D_2|}{|D|} Ent(D_2) \right) \\
&= Ent(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} Ent(D^v)
\end{aligned}$$

- 增益率

使用信息增益来选择特征时，算法会偏向于取值多的特征，也就是说取值越多可能会使信息增益越大，但并没有实际意义，故引入增益率

$$GainRatio(D, F^{(j)}) = \frac{Gain(D, F^{(j)})}{SplitInfo(F^{(j)})}$$

其中： $SplitInfo(F^{(j)}) = - \sum \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|}$ (D_i 是依据特征 $F^{(j)}$ 的取值划分的样本子集)

- 基尼指数

对于样本集 D ，其基尼指数为： $Gini(D) = 1 - \sum_{k=1}^K \left(\frac{|D_k|}{|D|} \right)^2 = 1 - \frac{\sum_{k=1}^K |D_k|^2}{|D|^2}$

将样本集 D 划分为独立的两个子集 D_1 和 D_2 ，其基尼指数为：

$$Gini(\{D_1, D_2\}) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

连续值与缺失值

- 连续值处理

连续属性离散化，常见做法：二分法

- 缺失值处理

- 仅通过无缺失值的样例来判断划分属性的优劣

- 多叉决策树

步数	操作
1	对输入的样本集，如果集合为空，算法结束
2	如果不能选择到一个合适的特征，则建立叶子节点，算法结束
3	根据选择到的特征，建立内部节点
4	依据选择到的特征将输入的样本集划分为若干个子集，对每个子集应用本算法

- 随机森林算法

步数	操作
1	从样本集中随机选择m个样本组成样本子集
2	从特征集中随机选择k个特征
3	对样本子集和选择的k个特征运用决策树算法，生成一颗决策树
4	重复上述过程n次

35

预剪枝与后剪枝处理

- 预剪枝

判断某个结点是否划分，计算不划分的精度，再计算划分的精度，从而决定是否对该树进行剪枝。

优点

- 降低过拟合风险
- 显著减少训练时间和测试时间开销

缺点

- 欠拟合风险

- 后剪枝

先从训练集生成一颗完整的决策树，然后自底向上地对非叶结点进行考察，若将该结点对应的子树替换为叶结点能带来决策树泛化性能提升，则将该子树替换为叶结点

优点

后剪枝比预剪枝保留了更多的分支，欠拟合风险小，泛化性能优于预剪枝

缺点

训练时间开销大

回归树

树模型解决回归问题的基本思想是将样本空间切分为多个子空间，在每个子空间中单独建立回归模型

- **CART树生成**

在回归问题中，标签值是连续的，将扎堆样本分在一起更合理些，所以使得切分的两个子集的方差和最小的那个值，称为最小剩余方差

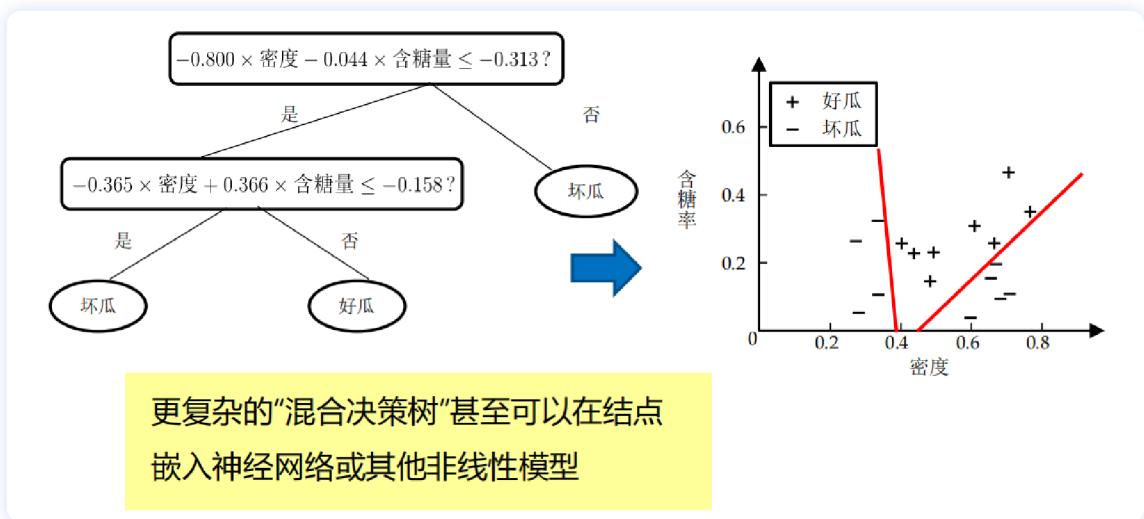
最小剩余方差

$$\sum_{i=1}^k (y_i - \hat{y}_1)^2 + \sum_{i=k+1}^m (y_i - \hat{y}_2)^2$$

步数	操作
1	待切分样本集数量是否少于数量阈值？如果少于数量阈值，将该样本集作为叶子节点输出，其预测值设为集内样本点的均值，算法结束，否则，进入下一步。
2	对每一个特征的每一个取值，将样本集试切分为大于和小于两个子集，计算剩余方差，记录下取得最小剩余方差的特征及其值和切分的左右子集。
3	如果最小剩余方差的最小值小于指标阈值，则将该样本集作为叶子节点输出，其预测值设为集内样本点的均值，算法结束，否则，分别对最小值的左右子集分别应用本算法。

多变量决策树

按照普通决策树，画出的图像是通过与轴平行的几条线来进行分割的，但是多变量决策树可以加入权重的一个线性组合



主要考察方向

- 计算信息增益
- 例题1
- 例题2
- 增益率
- PPT 无样例，后期有时间增加
- 基尼指数

贝叶斯分类

概率模型

- 条件概率

$$P(B|A) = \frac{p(AB)}{P(A)}$$

- 贝叶斯公式

$$P(A_k|B) = \frac{P(A_k) \cdot P(B|A_k)}{\sum_{i=0}^n P(A_i) \cdot P(B|A_i)} \quad (\text{也称为后验概率})$$

- 贝叶斯决策论

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

$$h^*(x) = \operatorname{argmin}_{c \in y} R(c|x)$$

$\lambda_{i,j}$, 当 $i = j$ 的时候等于 0, 其余等于 1

朴素贝叶斯分类器

朴素贝叶斯分类是基于贝叶斯定理与特征条件独立假定的分类方法

- 先验概率

对某一标签值 $y^{(1)}$, 其先验概率为 $P(Y = y^{(i)})$,
在标签取值 $y^{(i)}$ 时 X 为某一 x 的条件概率为 $P(X = x|Y = y^{(i)})$

- 后验概率

$$\begin{aligned} P(Y = y^{(1)}|X = x) &= \frac{P(X = x, Y = y^{(i)})}{P(X = x)} \\ &= \frac{P(X = x|Y = y^{(l)})P(Y = y^{(l)})}{\sum_j^k P(X = x|Y = y^{(j)})P(Y = y^{(j)})} \end{aligned}$$

- 估计先验概率

p_i 是事件 $Y = y^{(l)}$ 发生的概率, 不发生的概率为 $1 - p_i$

$$\begin{aligned} \text{似然函数: } L(p_i) &= \prod_{i=1}^{M_l} p_i \cdot \prod_{i=1}^{m-M_l} (1 - p_i) \\ &= p_i^{M_l} (1 - p_i)^{m-M_l} \end{aligned}$$

对两边加 \ln , 然后对 p_i 求导, 等于零

$$\text{求得 } p_l = \frac{\sum_{i=1}^m I(y_i = y^{(l)})}{m}$$

- 估计条件概率

$$\begin{aligned} &P(X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)} \dots | Y = y^{(l)}) \\ &= \frac{P(X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)} \dots, Y = y^{(l)})}{P(Y = y^{(l)})} \end{aligned}$$

- 条件独立性假定下的条件概率

$$\begin{aligned} &P(X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)} \dots | Y = y^{(l)}) \\ &= P(X^{(1)} = x^{(1)} | Y = y^{(l)})P(X^{(2)} = x^{(2)} | Y = y^{(2)}) \dots \\ &\quad = \prod_{j=1}^n P(X^{(j)} = x^{(j)} | Y = y^{(l)}) \end{aligned}$$

$$P(X^{(j)} = x^{(j)} | Y = y^{(l)}) = \frac{\sum_{i=1}^m I(X_i^{(j)} = x^{(j)}, y_i = y^{(l)})}{\sum_{i=1}^m I(y_i = y^{(l)})}$$

- 生成式模型

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

- 极大似然估计

我们的任务就是通过训练集 D_c ，来估计参数

- 频率主义学派

参数虽然未知，但却存在客观值，因此可通过优化似然函数等准则来确定参数值

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

上式的连乘操作容易造成下溢，通常使用对数似然

$$\begin{aligned} LL(\theta_c) &= \log P(D_c|\theta_c) = \sum_{x \in D_c} \log P(x|\theta_c) \\ \theta_c &= \arg \max LL(\theta_c) \end{aligned}$$

- 贝叶斯学派

认为参数是未观察到的随机变量，因此可以假定参数服从一个先验分布

$$\begin{aligned} \mu_c &= \frac{1}{|D_c|} \sum_{x \in D_c} x \\ \sigma_c^2 &= \frac{1}{|D_c|} \sum_{x \in D_c} (x - \mu_c)(x - \mu_c)^T \end{aligned}$$

- 朴素贝叶斯分类器

朴素贝叶斯采用了“属性条件独立性假设”每个属性独立地对分类结果发生影响

对于所有类别来说 $P(x)$ 相同，因此贝叶斯判定准则有：

$$h_{nb}(x) = \operatorname{argmax}_{c \in \mathcal{Y}} P(c) \prod_{i=1}^d P(x_i|c)$$

- 一些计算步骤

$$\begin{aligned}
 - \quad P(c) &= \frac{|D_c|}{D} \\
 -\text{离散属性} \quad P(x_i|c) &= \frac{|D_{c,x_i}|}{D_c} \\
 -\text{连续属性} \quad P(x_i|c) &= \frac{1}{\sqrt{2\pi}\sigma_{c,i}} \exp\left(-\frac{(x_i - \mu_{c,i})^2}{2\sigma_{c,i}^2}\right)
 \end{aligned}$$

- 多项式朴素贝叶斯分类器

在用极大似然法对后验概率进行估计时，假设先验概率和条件概率服从多项式分布

$$\text{先验概率变为: } P(Y = y^{(l)}) = \frac{\sum_{i=1}^m I(y_i = y^{(l)}) + \alpha}{m + k\alpha}$$

$$\text{条件概率的估计变为: } P(X^{(j)} = x^{(j)} | Y = y^{(l)}) = \frac{\sum_{i=1}^m I(x_i^{(j)}, y_i) + \alpha}{\sum_{i=1}^m I(y_i = y^{(l)}) + S_j\alpha}$$

- 高斯朴素贝叶斯分类器

只是把正态分布变为了高斯分布

$$P(X^{(j)} = x^{(j)} | Y = y^{(l)}) = \frac{1}{\sqrt{2\pi\hat{\sigma}^2}} e^{-\frac{(x^{(j)} - \hat{\mu})^2}{2\hat{\sigma}^2}}$$

半朴素贝叶斯分类器

为了降低贝叶斯公式中后验概率的困难，朴素贝叶斯分类器采用的属性条件独立假设，若对属性条件独立假设进行一定程度的放松，就是半朴素贝叶斯分类器

- SPODE: 假设所有属性都依赖于同一属性，称为“超父”，然后通过交叉验证等模型选择方法来确定超父属性
- TAN: 以属性间的条件“互信息”为边的权重，构建完全图，再利用最大带权生成树算法，仅保留强相关属性间的依赖性

□ 简约步骤

- 计算任意两个属性之间的条件互信息 (conditional mutual information)

$$I(x_i, x_j | y) = \sum_{x_i, x_j; c \in y} P(x_i, x_j | c) \log \frac{P(x_i, x_j | c)}{P(x_i | c)P(x_j | c)}$$

- 以属性为结点构建完全图，任意两个结点之间边的权重设为 $I(x_i, x_j | y)$
- 构建此完全图的最大带权生成树，挑选根变量，将边设为有向；
- 加入类别节点y，增加从y到每个属性的有向边。

• AODE

□ AODE (Averaged One-Dependent Estimator) [Webb et al. 2005] 是一种基于集成学习机制、更为强大的分类器。

- 尝试将每个属性作为超父构建 SPODE
- 将具有足够训练数据支撑的SPODE集群起来作为最终结果

$$P(c | \mathbf{x}) \propto \sum_{i=1; |D_{x_i}| \geq m'}^d P(c, x_i) \prod_{j=1}^d P(x_j | c, x_i)$$

$$\hat{P}(x_i, c) = \frac{|D_{c,x_i}| + 1}{|D| + N_i} \quad \hat{P}(x_j | c, x_i) = \frac{|D_{c,x_i,x_j}| + 1}{|D_{c,x_i}| + N_j} N_i$$

• 贝叶斯网

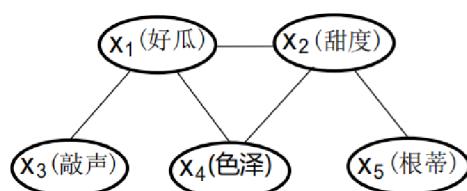
贝叶斯网络中的节点表示随机变量，有向边表示变量之间有因果关系(非条件独立)，两个用箭头连接的节点就会产生一个条件概率值

每一个节点在其直接前驱节点的值制定后，这个节点条件独立于其所有非直接前驱前辈节点。

- 分析有向图中变量间的条件独立性，可使用“有向分离” (D-separation)

- V型结构父结点相连
- 有向边变成无向边

由此产生的图称为道德图
(moral graph)

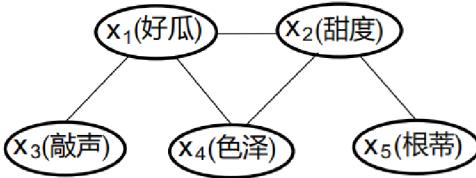


“有向分离” (D-separation)

先将有向图转变为无向图

- V型结构父结点相连
- 有向边变成无向边

道德图(moral graph)



若 x 和 y 能在图上被 z 分入两个连通分支，则有

$$x \perp y \mid z$$

$$\text{由图可得: } x_3 \perp x_4 \mid x_1$$

$$x_4 \perp x_5 \mid x_2$$

$$x_3 \perp x_2 \mid x_1$$

$$x_3 \perp x_5 \mid x_1$$

$$x_3 \perp x_5 \mid x_2$$

得到条件独立性关系之后，估计出条件概率表，就得到了最终网络

■ 结构学习：最小描述长度评估贝叶斯网与训练数据的契合程度

■ 推断：基于已知属性变量的观测值，推测其他属性变量的取值

近似推断：吉布斯采样（直觉不考）

EM 算法

在某些情况下，模型中含有无法明确观察到的隐参数，则无法直接用极大似然法来估计模型参数

它的基本思想是求期望和求极大化的逐步迭代。它先假定隐参数的值，然后基于已经观察到的样本数据和该假定，用极大似然法来估计其它参数。

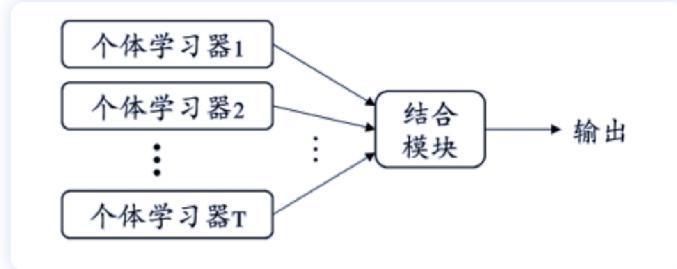
主要考察方向

- 样例
- 样例1
- 样例2

集成学习

个体与集成

集成学习通过构建并结合多个学习器来提升性能



- 简单分析

考虑二分类问题，假设基分类器的错误率为

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

- 假设集成通过简单投票法结合 T 个分类器，超过半数的基分类器正确则分类就正确
- 假设基分类器的错误率相互独立，则由Hoeffding不等式可得集成的错误率为

$$P(H(x) \neq f(x)) \leq \exp\left(-\frac{1}{2}T(1 - 2\epsilon)^2\right)$$

- 上式显示，在一定条件下，随着集成分类器数目的增加，集成错误率将指数级下降，最终趋向于 0

- 成功的集成学习方法

- 序列化方法：AdaBoost
- 并行化方法：Bagging

Boosting

个体学习器存在强依赖关系

串行生成

每次调整训练数据的样本分布

- Boosting算法
- AdaBoost算法

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 \mathfrak{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:    $h_t = \mathfrak{L}(D, \mathcal{D}_t);$ 
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: $H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$

Bagging与随机森林

个体学习器不存在强依赖关系

并行化生成

自助采样法

- Bagging算法

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 \mathfrak{L} ;

训练轮数 T .

过程:

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

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

- 随机森林

Input: Data set $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$;
Feature subset size K .

Process:

1. $N \leftarrow$ create a tree node based on D ;
2. **if** all instances in the same class **then return** N
3. $\mathcal{F} \leftarrow$ the set of features that can be split further;
4. **if** \mathcal{F} is empty **then return** N
5. $\tilde{\mathcal{F}} \leftarrow$ select K features from \mathcal{F} randomly;
6. $N.f \leftarrow$ the feature which has the best split point in $\tilde{\mathcal{F}}$;
7. $N.p \leftarrow$ the best split point on $N.f$;
8. $D_l \leftarrow$ subset of D with values on $N.f$ smaller than $N.p$;
9. $D_r \leftarrow$ subset of D with values on $N.f$ no smaller than $N.p$;
10. $N_l \leftarrow$ call the process with parameters (D_l, K) ;
11. $N_r \leftarrow$ call the process with parameters (D_r, K) ;
12. **return** N

Output: A random decision tree

结合策略

- 投票法
 - 绝对多数投票法
 - 相对多数投票法
 - 加权投票法
- 平均法
 - 简单平均法
 - 加权平均法
- 学习法

多样性

- 误差-分歧分解

支持向量机

间隔与支持向量

- 间隔

$$\text{点与超平面的距离为: } \gamma_1 = \frac{\omega}{\|\omega\|} \cdot x_i + \frac{b}{\|\omega\|}$$

$$\text{超平面两类之间的间隔为: } \gamma = \frac{2}{\|\omega\|}$$

最大间隔: 寻找参数 ω 和 b , 使得 γ 最大

$$\operatorname{argmin} \frac{1}{2} \|\omega\|^2$$

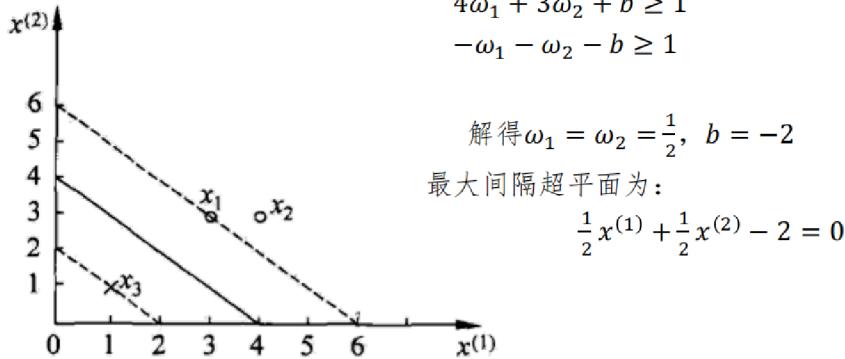
- 最大间隔超平面的存在唯一性

- 实例

- 如图所示的训练数据集, 其正例点是 $x_1 = (3,3)^T$, $x_2 = (4,3)^T$, 负例点是 $x_3 = (1,1)^T$, 试求最大间隔分离超平面。

构造约束最优化问题:

$$\begin{aligned} & \min_{\omega, b} \frac{1}{2} (\omega_1^2 + \omega_2^2) \\ \text{s.t. } & 3\omega_1 + 3\omega_2 + b \geq 1 \\ & 4\omega_1 + 3\omega_2 + b \geq 1 \\ & -\omega_1 - \omega_2 - b \geq 1 \end{aligned}$$



对偶问题

- 拉格朗日乘子法

- 第一步: 引入拉格朗日乘子 α 得到拉格朗日函数
- 第二步: 令拉格朗日函数对 w 和 b 的偏导为 0
- 第三步: 回代

- 线性可分支持向量机学习算法

- 第一步: 构造并求解约束最优化问题

$$\begin{aligned} & \min_{\alpha} \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j - \sum_{i=1}^m \alpha_i \\ \text{s.t. } & \sum_{i=1}^m \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad i = 1, 2, \dots, m. \end{aligned}$$

- 第二步: 计算 w 和 b
- 第三步: 求得分离超平面

- 实例

例子：已知训练数据集其正例是 $x_1 = (4, 4)^T$ ， $x_2 = (5, 3)^T$ ，负例点 $x_3 = (1, 1)^T$ ，试用线性可分支持向量机的对偶学习算法，求得分离超平面。

- 如图所示的训练数据集，其正例点是 $x_1 = (3, 3)^T$ ， $x_2 = (4, 3)^T$ ，负例点是 $x_3 = (1, 1)^T$ ，试求最大间隔分离超平面。

- 解的稀疏性

训练完成后，大部分的训练样本都不需要保留，最终模型仅与支持向量有关

KKT 条件

$$\begin{cases} \alpha_i \geq 0, \\ y_i f(\mathbf{x}_i) \geq 1, \\ \alpha_i(y_i f(\mathbf{x}_i) - 1) = 0. \end{cases}$$

软间隔

允许支持向量机在一些样本上不满足约束

- 0/1损失函数

基本想法：最大化间隔的同时，让不满足约束的样本尽可能少。

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

- 软间隔支持向量机

- 原始问题 $\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \max(0, 1 - y_i (\mathbf{w}^\top \mathbf{x}_i + b))$

- 引入“松弛变量”(slack variables) ξ_i

$$\begin{aligned} & \min_{\mathbf{w}, b, \xi_i} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i \\ & \text{s.t. } y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \end{aligned}$$

- 对偶问题

$$\begin{aligned} & \max_{\boldsymbol{\alpha}} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j && \text{与“硬间隔SVM”的区别} \\ & \text{s.t. } \sum_{i=1}^m \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, m. \end{aligned}$$

C越大，越容易过拟合

- 正则化

支持向量机学习模型的更一般形式

$$\min_f \Omega(f) + C \sum_{i=1}^m l(f(\mathbf{x}_i), y_i)$$

结构风险(structural risk)描述模型本身的某些性质

经验风险(empirical risk), 描述模型与训练数据的契合程度

- 正则化可理解为“罚函数法”
通过对不希望的结果施以惩罚，使得优化过程趋向于希望目标
- 从贝叶斯估计的角度，则可认为是提供了模型的先验概率

- 序列最小最优化算法(SMO)

如果所有变量的解都满足此最优化问题的KKT条件，那么可以得到这个最优化问题的解

- 基本思路：不断执行如下两个步骤直至收敛。
 - 第一步：选取一对需更新的变量 α_i 和 α_j 。
 - 第二步：固定 α_i 和 α_j 以外的参数，求解对偶问题更新 α_i 和 α_j 。
- 仅考虑 α_i 和 α_j 时，对偶问题的约束变为

$$\alpha_i y_i + \alpha_j y_j = - \sum_{k \neq i, j} \alpha_k y_k, \quad \alpha_i \geq 0, \quad \alpha_j \geq 0.$$

用一个变量表示另一个变量，回代入对偶问题可得一个单变量的二次规划，该问题具有闭式解。

- 偏移项b：通过支持向量来确定。

- SMO 算法

输入：线性可分训练数据集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$, 其中 $x_i \in \mathbb{R}^n$, $y \in \{-1, +1\}$, $i = 1, 2, \dots, m$, 精度 ε

输出：近似解 $\hat{\alpha}$

(1) 取初值 $\alpha^{(0)} = 0$, 令 $k = 0$;

(2) 选取优化变量 $\alpha_1^{(k)}, \alpha_2^{(k)}$, 解析求解两个变量的最优化问题

$$\min_{\alpha_1, \alpha_2} W(\alpha_1, \alpha_2) = \frac{1}{2} K_{11} \alpha_1^2 + \frac{1}{2} K_{22} \alpha_2^2 + y_1 y_2 K_{12} - (\alpha_1 + \alpha_2)$$

$$+ y_1 \alpha_1 \sum_{i=3}^m y_i \alpha_i K_{i1} + y_2 \alpha_2 \sum_{i=3}^m y_i \alpha_i K_{i2}$$

$$\text{s.t. } \alpha_1 y_1 + \alpha_2 y_2 = - \sum_{i=3}^m y_i \alpha_i$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2$$

求得最优解 $\alpha_1^{(k+1)}, \alpha_2^{(k+1)}$, 更新 α 为 $\alpha^{(k+1)}$;

(3) 若在精度 ε 范围内满足停机条件

$$\sum_{i=1}^m \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, i = 1, 2, \dots, m$$

$$y_i \cdot g(x_i) = \begin{cases} \geq 1, & \{x_i | \alpha_i = 0\} \\ = 1, & \{x_i | 0 < \alpha_i < C\} \\ \leq 1, & \{x_i | \alpha_i = C\} \end{cases}$$

其中 $g(x_i) = \sum_{j=1}^m \alpha_j y_j K(x_j, x_i) + b$, 则转 (4); 否则令 $k = k + 1$, 转 (2);

(4) 取 $\hat{\alpha} = \alpha^{(k+1)}$

核函数

如果原始空间是有限维，那么一定存在一个高维特征空间使样本可分

- 基本想法：不显式地设计核映射，而是设计核函数

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

- Mercer定理：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
- 常用核函数

名称	表达式	参数
线性核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$	
多项式核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j)^d$	$d \geq 1$ 为多项式的次数
高斯核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\ \mathbf{x}_i - \mathbf{x}_j\ ^2}{2\delta^2}\right)$	$\delta > 0$ 为高斯核的带宽(width)
拉普拉斯核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\ \mathbf{x}_i - \mathbf{x}_j\ }{\delta}\right)$	$\delta > 0$
Sigmoid核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta \mathbf{x}_i^\top \mathbf{x}_j + \theta)$	\tanh 为双曲正切函数, $\beta > 0, \theta < 0$

基本经验：文本数据常用线性核，
可通过函数组合得到：情况不明时可先尝试高斯核

- 非线性支持向量机学习算法

输入：训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ ，其中 $x_i \in \mathbb{R}^n$, $y \in \{-1, +1\}$, $i = 1, 2, \dots, N$

输出：分类决策函数

(1) 选取适当的核函数 $K(x, x_i)$ 和适当的参数 C ，构造并求解最优化问题

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

求得最优解 $\alpha^* = (\alpha_1^*, \alpha_2^*, \dots, \alpha_m^*)$

(2) 选择 α^* 的一个正分量 $0 < \alpha_j^* < C$ ，计算

$$b^* = y_j - \sum_{i=1}^m \alpha_i^* y_i K(x, x_i)$$

(3) 构造决策函数： $f(x) = \text{sign}(\sum_{i=1}^m \alpha_i^* y_i K(x, x_i) + b^*)$

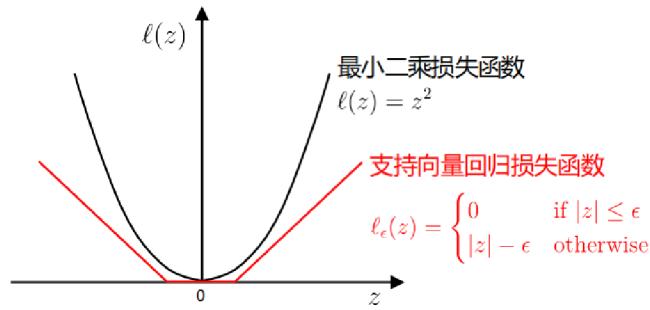
当 $K(x, x_i)$ 是正定函数时，该凸二次规划问题，解是存在的。

支持向量回归

支持向量回归允许模型输出和实际输出间存在 2σ 的误差

- 损失函数

落入中间 2σ 间隔带的样本不计算损失，从而使模型获得稀疏性



- 形式化

原始问题

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

对偶问题

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

预测

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

核方法

- 最大的区别：SVM的理论基础比神经网络更加坚实，有严格的数学推理
- 神经网络优化目标为经验风险最小化；支持向量机基于结构风险最小化，泛化能力优于神经网络。
- 支持向量机以统计学理论为基础，主要针对有限样本集，算法能够转化成凸优化问题，因而可以保证算法的全局最优化，避免了神经网络中常出现的局部最小值问题。

主要考察方向

- 样例计算
- 样例 2
- 例题

神经网络

神经网络发展史

神经元模型

神经网络是由具有适应性的简单单元组成的广泛并行互联的网络,它的组织能够模拟生物神经系统对真实世界物体所作出的反应

- M-P模型
- 激活函数
 - 阶跃函数
 - Sigmoid函数
$$g'(x)=g(x)(1-g(x))$$
 - ReLu函数
$$f(x)=\max(0,x)$$
- 感知机模型

感知机模型是二分类的线性分类模型, 它能对空间中线性可分的二分类样本点进行划分。

$$y = u\left(\sum_{i=1}^n w^{(i)}x^{(i)} - \theta\right) = u(W \cdot x^T - \theta)$$

- 感知机学习

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

- 学习率

学习率太小，收敛就非常慢

学习率太大，就可能导致网络的瘫痪和不稳定

自适应学习率

多层神经网络

- 多层前馈神经网络

- 误差逆传播算法

- 标准 BP 算法：每次针对单个训练样例更新权值与阈值

- 累计 BP 算法：最小化整个训练集上的累计误差

- 多层神经网络常用损失函数

- 相对熵损失函数和交叉熵损失函数

$$H(y, p) = -[y \log p + (1 - y) \log(1 - p)]$$

- 余弦相似度损失函数

$$\cos \theta = \frac{x_i \cdot x_j}{\|x_i\| \cdot \|x_j\|}$$

- 双曲余弦对数损失函数

$$\text{logcosh}(p, q) = \sum_{i=1}^n \log \frac{e^{q_i - p_i} + e^{-(q_i - p_i)}}{2}$$

- 多层神经网络常用优化算法

- 动量优化算法

加入动量之后的前进量是上一步的前进量和新梯度值

- 步长优化算法

随着迭代次数的增加，步长越来越小，可以防止在极小值附近来回振荡

- 结合动量和步长优化算法

- 多层神经网络中过拟合的抑制

- 正则化

- 早停法

- Dropout 法

- 进一步讨论

- 局部收敛

- 模拟退火：每一步都以一定概率接受比当前解更差的结果

- 随机梯度下降：在计算梯度时加入随机因素

- 遗传算法：常用来训练神经网络
- 梯度消散和梯度爆炸

其他常见神经网络

- RBF网络

RBF 网络是一种单隐层前馈神经网络, 它使用径向基函数作为隐层神经元激活函数, 而输出层则是隐层神经元输出的线性组合.

- ART 网络

ART网络性能依赖于识别阈值

- 自组织映射(SOM)网络

SOM 网络是一种竞争型的无监督神经网络, 它能将高维数据映射到低维空间（通常为2维）, 同时保持输入数据在高维空间的拓扑结构, 即将高维空间中相似的样本点映射到网络输出层中邻近神经元

- 级联相关网络

级联相关网络不仅利用训练样本优化连接权值, 阈值参数, 将网络的结构也当做学习的目标之一, 希望在训练过程中找到适合数据的网络结构.

- Elman网络
- 递归神经网络
- Boltzmann机

神经网络中有一类模型为网络定义一个“能量”, 能量最小化时网络达到理想状态, 而网络的训练就是在最小化这个能量函数.

概述

- 深度学习与神经网络

- 相同点：二者均采用分层结构，包括输入层、输出层组成的多层网络
- 不同点：神经网络采用 BP 算法调整参数，采用迭代法训练整个网络，深度学习采用逐层训练机制，采用该机制的原因在于如果采用 BP，残差传到最前面会很小

- 神经网络的局限性

- 比较容易过拟合，参数比较难调整
- 训练速度慢，层次较小的情况下效果不比其它方法更优

卷积神经网络

- 复杂模型训练方法

预训练+微调

权共享

卷积神经网络

- 卷积神经网络

- 卷积层
- 采样层
- 连接层

- 卷积层

Tensor是图片组成的多维数据

1. 一般会设置多个卷积核，将图片的长宽变小，厚度增加
2. 如果待处理张量规模很大，可以将卷积核由依次移动改为跳跃移动
3. 为了提取到边缘的特征，可以在待处理张量的边缘填充 0 再进行 juanji

- 池化层和Flatten层

1. 池化层与卷积层类似，只不过卷积核只取对应位置的最大值或平均值，分别称为最大池化或平均池化
2. Flatten层，只是将输入的多维数据拉成一维的

- 典型卷积神经网络

1. VGG-16,VGG-19
2. 残差网络

提出了抑制退化和梯度消散，加速训练收敛的方法

循环神经网络

循环神经网络是用于对序列的非线性特征进行学习的深度神经网络

- **基本单元**

类似于隐马尔可夫，把循环神经网络中间部分称为隐层，隐层的输出有两个一个是 y ，另一个反馈到自身

- **网络结构**

- One to many
- many to one
- Many to many

- **长短时记忆网络**

遗忘门用来控制上一步的状态输入到本步的量，也就是遗忘了上一步的状态的程度

结束语

Open source is a spirit that arises for freedom and equality.

如发现内容错误，请联系编辑者进行修正，该内容仅针对2023级的重点，但每年的核心应为固定的

E-mail:2303112308@qq.com

File Source Address:<https://wr0519.github.io/moan-blog/#/>