> 本文由 简悦 SimpRead 转码， 原文地址 www.cnblogs.com

阅读目录

因为要准备面试, 本文以李航的《统计学习方法》为主, 结合西瓜书等其他资料对机器学习知识做一个整理.

回到顶部

知识点

- **进程和线程**: 进程和线程都是一个时间段的描述，是 CPU 工作时间段的描述，不过是颗粒大小不同. 进程就是包换上下文切换的程序执行时间总和 = CPU 加载上下文 + CPU 执行 + CPU 保存上下文. 线程是共享了进程的上下文环境的更为细小的 CPU 时间段。
- **判别式模型和生成式模型**:

1. 判别式模型直接学习**决策函数 f(X)** 或**条件概率分布 P(Y|X)** 作为预测的模型. 往往准确率更高, 并且可以简化学习问题. 如 k 近邻法 / 感知机 / 决策树 / 最大熵模型 / Logistic 回归 / 线性判别分析 (LDA)/ 支持向量机 (SVM)/Boosting / 条件随机场算法 (CRF)/ 线性回归 / 神经网络
2. 生成式模型由数据学习**联合概率分布 P(X,Y)**, 然后由 P(Y|X)=P(X,Y)/P(X) 求出条件概率分布作为预测的模型, 即生成模型. 当存在隐变量时只能用生成方法学习. 如混合高斯模型和其他混合模型 / 隐马尔可夫模型 (HMM)/ 朴素贝叶斯 / 依赖贝叶斯 (AODE)/LDA 文档主题生成模型

- **概率质量函数, 概率密度函数, 累积分布函数**:

1. 概率质量函数 (probability mass function，PMF) 是离散随机变量在各特定取值上的概率。
2. 概率密度函数（p robability density function，PDF ）是对 连续随机变量 定义的，本身不是概率，只有对连续随机变量的取值进行积分后才是概率。
3. 累积分布函数（cumulative distribution function，CDF） 能完整描述一个实数随机变量 X 的概率分布，是概率密度函数的积分。对於所有实数 x ，与 pdf 相对。

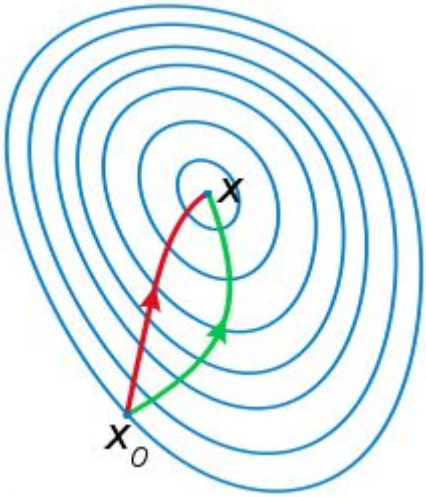
- **极大似然估计**: 已知某个参数能使这个样本出现的概率最大，我们当然不会再去选择其他小概率的样本，所以干脆就把这个参数作为估计的真实值

- **最小二乘法**: 二乘的英文是 least square, 找一个（组）估计值, 使得实际值与估计值之差的平方加总之后的值 $Q = min \sum_{i}^{n} (y_{ie} - y_i)^2$ 最小. 求解方式是对参数求偏导, 令偏导为 0 即可. **样本量小**时速度快.
- **梯度下降法**: 负梯度方向是函数值下降最快的方向, 每次更新值都等于原值加学习率 (**步长**) 乘损失函数的**梯度**. 每次都试一个步长看会不会下降一定的程度, 如果没有的话就按比例减小步长. 不断应用该公式直到收敛, 可以得到局部最小值. 初始值的不同组合可以得到不同局部最小值. 在最优点时会有震荡.

1. **批量梯度下降 (BGD)**: 每次都使用所有的 m 个样本来更新, 容易找到全局最优解, 但是 m 较大时速度较慢 $\theta_j^{'} = \theta_j + \frac{1}{m} \sum_{i=1}^{m} (y^i - h_\theta(x^i)) x_j^i$
2. **随机梯度下降 (SGD)**: 每次只使用一个样本来更新, 训练速度快, 但是噪音较多, 不容易找到全局最优解, 以损失很小的一部分精确度和增加一定数量的迭代次数为代价, 换取了总体的优化效率的提升. 注意控制步长缩小, 减少震荡.

$$\theta_j^{'} = \theta_j + (y^i - h_\theta(x^i)) x_j^i$$

3. **小批量梯度下降 (MBGD)**: 每次使用一部分样本来更新.

- **牛顿法**: 牛顿法是**二次收敛**, 因此收敛速度快. 从几何上看是每次用一个二次曲面来拟合当前所处位置的局部曲面, 而梯度下降法是用一个平面来拟合. 

红色的是牛顿法的迭代路径, 绿色的是梯度下降法的迭代路径. 牛顿法起始点不能离极小点太远, 否则很可能不会拟合.

1. **黑塞矩阵**是由目标函数 f(x) 在点 X 处的二阶偏导数组成的 n*n 阶对称矩阵。
2. **牛顿法**: 将 f(x) 在 x(k) 附近进行**二阶泰勒展开**:

$$f(x) = f(x^{(k)}) + g_k^{\mathrm{T}}(x - x^{(k)}) + \frac{1}{2}(x - x^{(k)})^{\mathrm{T}} H(x^{(k)})(x - x^{(k)}),$$

其中 gk 是 f(x) 的梯度向量在 x(k) 的值, H(x(k)) 是 f(x) 的黑塞矩阵在点 x(k) 的值. 牛顿法利用极小点的必要条件 f(x) 处的梯度为 0, 每次迭代中从点 x(k) 开始, 假设 $\nabla f(x^{(k+1)}) = 0$, 对二阶泰勒展开求偏导有 $\nabla f(x) = g_k + H_k(x - x^{(k)})$, 代入得到

$g_k + H_k(x^{(k+1)} - x^{(k)}) = 0$, 即 $x^{(k+1)} = x^{(k)} - H_k^{-1} g_k$, 以此为迭代公式就是牛顿法.

- **拟牛顿法**: 用一个 **n 阶正定矩阵 Gk**=G(x(k)) 来**近似代替**黑塞矩阵的**逆矩阵**就是拟牛顿法的基本思想. 在牛顿法中黑塞矩阵满足的条件如下: $g_{k+1} - g_k = H_k(x^{(k+1)} - x^{(k)})$, 令 $y_k = g_{k+1} - g_k$, $\delta_k = x^{(k+1)} - x^{(k)}$, 则有 $H_k^{-1} y_k = \delta_k$, 称为拟牛顿条件. 根据选择 Gk 方法的不同有多种具体实现方法.

1. **DFP 算法**: 假设每一步 $G_{k+1} = G_k + P_k + Q_k$, 为使 Gk+1 满足拟牛顿条件, 可使 Pk 和 Qk 满足 $P_k y_k = \delta_k$, $Q_k y_k = -G_k y_k$, 例如取 $P_k = \frac{\delta_k \delta_k^{\mathrm{T}}}{\delta_k^{\mathrm{T}} y_k}$, $Q_k = -\frac{G_k y_k y_k^{\mathrm{T}} G_k}{y_k^{\mathrm{T}} G_k y_k}$, 就得到迭代公式
$$G_{k+1} = G_k + \frac{\delta_k \delta_k^{\mathrm{T}}}{\delta_k^{\mathrm{T}} y_k} - \frac{G_k y_k y_k^{\mathrm{T}} G_k}{y_k^{\mathrm{T}} G_k y_k}$$
2. **BFGS 算法**: 最流行的拟牛顿算法. 考虑用 Bk 逼近黑塞矩阵, 此时相应的拟牛顿条件是 $B_{k+1}\delta_k = y_k$, 假设每一步 $B_{k+1} = B_k + P_k + Q_k$, 则 Pk 和 Qk 满足 $P_k \delta_k = y_k$, $Q_k \delta_k = -B_k \delta_k$, 类似得到迭代公式
$$B_{k+1} = B_k + \frac{y_k y_k^{\mathrm{T}}}{y_k^{\mathrm{T}} \delta_k} - \frac{B_k \delta_k \delta_k^{\mathrm{T}} B_k}{\delta_k^{\mathrm{T}} B_k \delta_k}.$$

- **先验概率和后验概率**:

1. 先验概率就是事情发生前的预测概率.
2. 后验概率是一种条件概率，它限定了事件为隐变量取值，而条件为观测结果。一般的条件概率，条件和事件可以是任意的.
3. 贝叶斯公式 P(y|x) = (P(x|y) * P(y)) / P(x) 中, P(y|x) 是后验概率, P(x|y) 是条件概率, P(y) 是先验概率.

- **偏差, 方差, 噪声**:

1. 偏差: 度量了学习算法的期望预测和真实结果偏离程度
2. 方差: 度量了同样大小的训练集的变动所导致的学习性能的变化，即刻画了数据扰动所造成的影响
3. 噪声: 可以认为是数据自身的波动性，表达了目前任何学习算法所能达到泛化误差的下限
4. **泛化误差**可以分解为偏差、方差与噪声之和

- **对偶原理**: 一个优化问题可以从主问题和对偶问题两个方面考虑. 在推导对偶问题时, 通过将拉格朗日函数对 x 求导并使导数为 0 来获得对偶函数. 对偶函数给出了主问题最优解的下界, 因此对偶问题一般是凸问题, 那么只需求解对偶函数的最优解就可以了.
- **KKT 条件**: 通常我们要求解的最优化条件有如下三种:

1. 无约束优化问题: 通常使用求导, 使导数为零, 求解候选最优值
2. 有等式约束的优化问题: 通常使用拉格朗日乘子法, 即把等式约束用拉格朗日乘子和优化问题合并为一个式子, 通过对各个变量求导使其为零, 求解候选最优值. 拉格朗日乘数法其实是 KKT 条件在等式约束优化问题的简化版.

3. 有不等式约束的优化问题: 通常使用 KKT 条件. 即把不等式约束, 等式约束和优化问题合并为一个式子. 假设有多个等式约束 h(x) 和不等式约束 g(x)

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i h_i(\boldsymbol{x}) + \sum_{j=1}^{n} \mu_j g_j(\boldsymbol{x}) \ ,$$

则不等式约束引入的 KKT 条件如下: $\begin{cases} g_j(\boldsymbol{x}) \leqslant 0; \\ \mu_j \geqslant 0 \ ; \\ \mu_j g_j(\boldsymbol{x}) = 0 \ . \end{cases}$, 实质是最优解在 g(x)<0 区域内时, 约束条件不起作用, 等价于对μ置零然后对原函数的偏导数置零; 当 g(x)=0 时与情况 2 相近. 结合两种情况, 那么只需要使 L 对 x 求导为零, 使 h(x) 为零, 使μg(x) 为零三式即可求解候选最优值.

- **性能度量**:

1. **准确度**, 最常用, 但在数据集不平衡的情况下不好
2. **Precision(精确度 / 查准率)**:P=TP/(TP+FP)
3. **Recall(召回率 / 查全率)**:R=TP/(TP+FN)
4. **Fβ度量**: $F_\beta = \frac{(1+\beta^2)rp}{\beta^2 * p + r}$, 当β=1 时退化为 F1 度量, 是精确率和召回率的调和均值.
5. **TPR(真正例率)**:TPR=TP/(TP+FN)
6. **FPR(假正例率)**:FPR=FP/(TN+FP)
7. **PR 曲线**: 纵轴为 Precision, 横轴为 Recall, 一般使用平衡点 (BEP, 即 Precsion=Recall 的点) 作为衡量标准.
8. **ROC(接受者操作特征) 曲线**: 纵轴为 TRP, 横轴为 FPR, 在绘图时将分类阈值依次设为每个样例的预测值, 再连接各点. ROC 曲线围住的面积称为 AOC,AOC 越大则学习器性能越好.

- **损失函数和风险函数**:

1. 损失函数度量模型一次预测的好坏. 常用的损失函数有: 0-1 损失函数, 平方损失函数, 绝对损失函数, 对数似然损失函数.
2. 损失函数的期望是理论上模型关于联合分布 P(X,Y) 的平均意义下的损失, 称为风险函数, 也叫**期望风险**. 但是联合分布是未知的, 期望风险不能直接计算.
3. 当样本容量 N 趋于无穷时经验风险趋于期望风险, 但现实中训练样本数目有限.

- **经验风险最小化和结构风险最小化:**

1. 模型关于训练数据集的平均损失称为经验风险. 经验风险最小化的策略就是最小化经验风险. 当样本数量足够大时学习效果较好. 比如当模型是条件概率分布, 损失函数是对数损失函数时, 经验风险最小化就等价于极大似然估计. 但是当样本容量很小时会出现过拟合.
2. 结构风险最小化等于正则化. 结构风险在经验风险上加上表示模型复杂度的正则化项. 比如当模型是条件概率分布, 损失函数是对数损失函数, 模型复杂度由模型的先验概率表示时, 结构风险最小化就等价于最大后验概率估计.

- **过拟合**是指学习时选择的模型所包含的参数过多, 以致于对已知数据预测得很好, 但对未知数据预测很差的现象. 模型选择旨在避免过拟合并提高模型的预测能力.

- **正则化**是模型选择的典型方法. 正则化项一般是模型复杂度的单调递增函数, 比如模型参数向量的范数.
- **交叉验证**是另一常用的模型选择方法, 可分为简单交叉验证, K 折交叉验证, 留一交叉验证等.

回到顶部

感知机

- 感知机是**二类分类**的线性模型, 属于判别模型. 感知机学习旨在求出将训练数据进行线性划分的分离超平面. 是神经网络和支持向量机的基础.
- **模型**: $f(x)=\text{sign}(w\cdot x+b)$,w 叫作权值向量, b 叫做偏置, sign 是符号函数.
- **感知机的几何解释**:wx+b 对应于特征空间中的一个分离超平面 S, 其中 w 是 S 的法向量, b 是 S 的截距. S 将特征空间划分为两个部分, 位于两个部分的点分别被分为正负两类.
- **策略**: 假设训练数据集是线性可分的, 感知机的损失函数是误分类点到超平面 S 的总距离. 因为误分类点到超平面 S 的距离是 $\frac{1}{\|w\|}|w\cdot x_0+b|$, 且对于误分类的数据来说, 总有 $-y_i(w\cdot x_i+b)>0$ 成立, 因此不考虑 1/||w||, 就得到感知机的损失函数: $L(w,b)=-\sum_{x_i\in M}y_i(w\cdot x_i+b)$, 其中 M 是误分类点的集合. 感知机学习的策略就是选取使损失函数最小的模型参数.
- **算法**: 感知机的最优化方法采用**随机梯度下降法**. 首先任意选取一个超平面 w0,b0, 然后不断地极小化目标函数. 在极小化过程中一次随机选取一个误分类点更新 w,b, 直到损失函数为 0. $w\leftarrow w+\eta y_i x_i$ $b\leftarrow b+\eta y_i$, 其中η表示步长. 该算法的直观解释是: 当一个点被误分类, 就调整 w,b 使分离超平面向该误分类点接近. 感知机的解可以不同.
- **对偶形式**: 假设原始形式中的 w0 和 b0 均为 0, 设逐步修改 w 和 b 共 n 次, 令 a=nη, 最后学习到的 w,b 可以表示为 $w=\sum_{i=1}^{N}\alpha_i y_i x_i$ $b=\sum_{i=1}^{N}\alpha_i y_i$. 那么对偶算法就变为设初始 a 和 b 均为 0, 每次选取数据更新 a 和 b 直至没有误分类点为止. 对偶形式的意义在于可以将训练集中实例间的内积计算出来, 存在 Gram 矩阵中, 可以大大加快训练速度.

回到顶部

k 近邻法

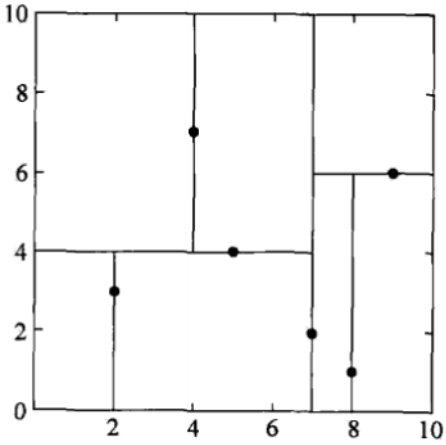
- k 近邻法根据其 **k 个最近邻**的训练实例的类别, 通过多数表决等方式进行预测. k 值的选择, 距离度量及分类决策规则是 k 近邻法的三个基本要素. 当 k=1 时称为最近邻算法.
- **模型**: 当训练集, 距离度量, k 值以及分类决策规则确定后, 特征空间已经根据这些要素被划分为一些子空间, 且子空间里每个点所属的类也已被确定.
- **策略**:

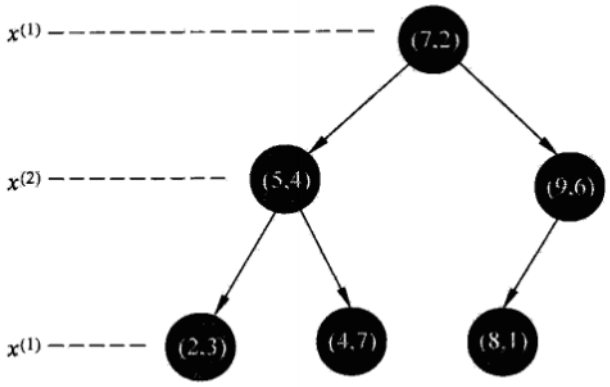
1. **距离**: 特征空间中两个实例点的距离是相似程度的反映, k 近邻算法一般使用欧氏距离, 也可以使用更一般的 Lp 距离或 Minkowski 距离.

2. **k 值**:k 值较小时, 整体模型变得复杂, 容易发生过拟合. k 值较大时, 整体模型变得简单. 在应用中 k 一般取较小的值, 通过交叉验证法选取最优的 k.
3. **分类决策规则**:k 近邻中的分类决策规则往往是多数表决, 多数表决规则等价于经验风险最小化.

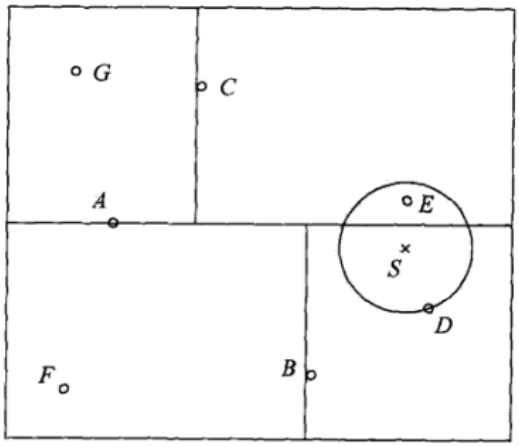
- **算法**: 根据给定的距离度量, 在训练集中找出与 x 最邻近的 k 个点, 根据分类规则决定 x 的类别 y.
- **kd 树**:

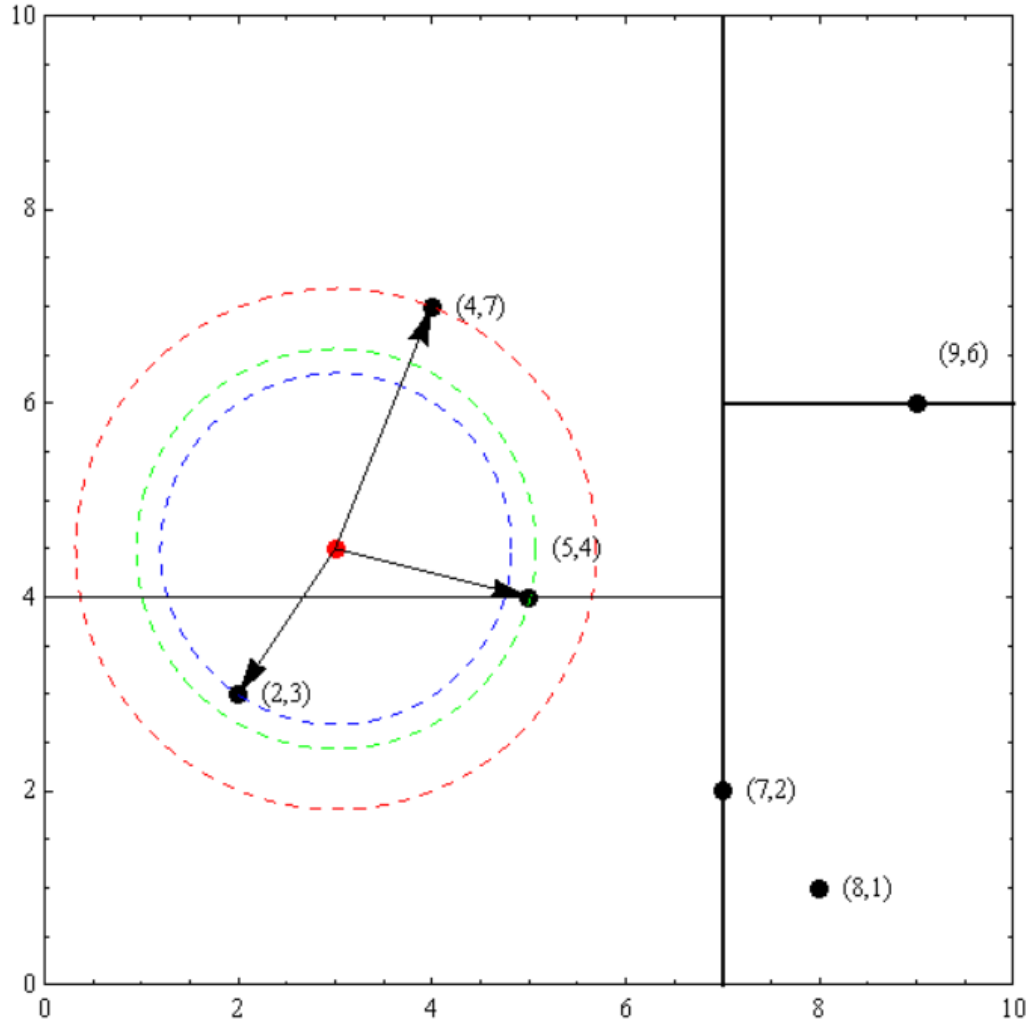
1. kd 树就是一种对 k 维空间中的实例点进行存储以便对其进行快速检索的树形数据结构. kd 树更适用于**训练实例数远大于空间维数**时的 k 近邻搜索.
2. **构造**: 可以通过如下**递归**实现: 在超矩形区域上选择一个**坐标轴**和此坐标轴上的一个**切分点**, 确定一个超平面, 该超平面将当前超矩形区域切分为两个子区域. 在子区域上重复切分直到子区域内没有实例时终止. 通常依次选择坐标轴和选定坐标轴上的**中位数点**为切分点, 这样可以得到平衡 kd 树.





3. **搜索**: 从根节点出发, 若目标点 x 当前维的坐标小于切分点的坐标则移动到左子结点, 否则移动到右子结点, 直到子结点为叶结点为止. 以此叶结点为 "当前最近点", **递归**地向上回退, 在每个结点:(a) 如果该结点比当前最近点距离目标点更近, 则以该结点为 "当前最近点"(b)"当前最近点" 一定存在于该结点一个子结点对应的区域, 检查该结点的另一子结点对应的区域是否与以目标点为球心, 以目标点与 "当前最近点" 间的距离为半径的超球体相交. 如果相交, 移动到另一个子结点, 如果不相交, 向上回退. 持续这个过程直到回退到根结点, 最后的 "当前最近点" 即为最近邻点.





回到顶部

朴素贝叶斯

- 朴素贝叶斯是基于**贝叶斯定理**和**特征条件独立假设**的分类方法. 首先学习输入 / 输出的联合概率分布, 然后基于此模型, 对给定的输入 x, 利用贝叶斯定理求出后验概率最大的输出 y. 属于生成模型.
- **模型**: 首先学习先验概率分布 $P(Y=c_k), \quad k=1,2,\cdots,K$, 然后学习条件概率分布

$$P(X=x|Y=c_k)=P(X^{(1)}=x^{(1)},\cdots,X^{(n)}=x^{(n)}|Y=c_k)$$

如果估计实际, 需要指数级的计算, 所以朴素贝叶斯法对条件概率分布作了条件独立性的假设, 上式

变成 $\prod_{j=1}^{n} P(X^{(j)}=x^{(j)} \mid Y=c_k)$. 在分类时, 通过学习到的模型计算后验概率分布, 由贝叶斯定理得到

$$P(Y=c_k \mid X=x)=\frac{P(X=x \mid Y=c_k)P(Y=c_k)}{\sum_k P(X=x \mid Y=c_k)P(Y=c_k)}$$, 将条件独立性假设得到的等式代入, 并且注意到分母都是相同的, 所以得到朴素贝叶斯分类器:

$$y=\arg\max_{c_k} P(Y=c_k)\prod_j P(X^{(j)}=x^{(j)} \mid Y=c_k)$$

- 朴素贝叶斯将实例分到后验概率最大的类中, 这等价于期望风险最小化.
- **算法**: 使用**极大似然估计法**估计相应的先验概率

$$P(Y=c_k)=\frac{\sum_{i=1}^{N} I(y_i=c_k)}{N}, \quad k=1,2,\cdots,K$$ 和条件概率

$$P(X^{(j)}=a_{jl} \mid Y=c_k)=\frac{\sum_{i=1}^{N} I(x_i^{(j)}=a_{jl}, y_i=c_k)}{\sum_{i=1}^{N} I(y_i=c_k)}$$, 计算条件独立性假设下的实例各个取值的可能性

$$P(Y=c_k)\prod_{j=1}^{n} P(X^{(j)}=x^{(j)} \mid Y=c_k), \quad k=1,2,\cdots,K$$, 选取其中的最大值作为输出.
- 用极大似然估计可能会出现所要估计的概率值为 0 的情况, 在累乘后会影响后验概率的计算结果, 使分类产生偏差. 可以采用**贝叶斯估计**, 在随机变量各个取值的频数上赋予一个正数.

$$P_\lambda(X^{(j)}=a_{jl} \mid Y=c_k)=\frac{\sum_{i=1}^{N} I(x_i^{(j)}=a_{jl}, y_i=c_k)+\lambda}{\sum_{i=1}^{N} I(y_i=c_k)+S_j\lambda}$$.Sj 为 j 属性可能取值数量, 当λ=0 时就是极大似然估计. 常取λ=1, 称为**拉普拉斯平滑**.
- 如果是连续值的情况, 可以假设连续变量服从高斯分布, 然后用训练数据估计参数.

$$P(X_i=x_i \mid Y=y_i)=\frac{1}{\sqrt{2\pi\sigma_{ij}^{2}}} e^{-\frac{(x_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

回到顶部

决策树

- 决策树是一种基本的分类与回归方法. 它可以认为是 **if-then 规则**的集合, 也可以认为是定义在特征空间与类空间上的**条件概率分布**. 主要优点是模型具有可读性, 分类速度快.
- **模型**: 分类决策树由**结点**和**有向边**组成. 结点分为**内部结点** (表示一个特征或属性) 和**叶结点** (表示一个类). 决策树的路径具有**互斥且完备**的性质.
- **策略**: 决策树学习本质上是从训练数据集中归纳出一组分类规则. 我们需要的是一个与训练数据**矛盾较小**, 同时具有很好的**泛化能力**的决策树. 从所有可能的决策树中选取最优决策树是 NP 完全问题, 所以现实中常采用**启发式方法**近似求解.
- **算法**: 决策树学习算法包含**特征选择**, **决策树的生成**与**决策树的剪枝过程**. 生成只考虑局部最优, 剪枝则考虑全局最优.
- **特征选择**: 如果利用一个特征进行分类的结果与随机分类的结果没有很大差别, 则称这个特征是**没有分类能力**的. 扔掉这样的特征对决策树学习的精度影响不大.

1. **信息熵:** 熵是衡量**随机变量不确定性**的度量. 熵越大, 随机变量的不确定性就越大. 信息熵是信息量的期望 $H(X) = -\sum_{x\varepsilon X} P(x)logP(x))$. 条件熵表示在已知随机变量 X 的条件下随机变量 Y 的不确定性. $H(Y \mid X) = \sum_{i=1}^{n} p_i H(Y \mid X = x_i)$
2. **信息增益**: 表示得知特征 X 的信息而使得类 Y 的信息的**不确定性减少**的程度. 定义为集合 D 的经验熵与特征 A 在给定条件下 D 的经验条件熵之差 $g(D, A) = H(D) - H(D \mid A)$, 也就是训练数据集中类与特征的**互信息**.
3. **信息增益算法**: 计算数据集 D 的经验熵 $H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|}$, 计算特征 A 对数据集 D 的经验条件熵 $H(D \mid A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|}$, 计算信息增益, 选取信息增益最大的特征.
4. **信息增益比**: 信息增益值的大小是相对于训练数据集而言的, 并无绝对意义. 使用信息增益比 $g_R(D, A) = \frac{g(D, A)}{H(D)}$ 可以对这一问题进行校正.

- **决策树的生成**:

1. **ID3 算法**: 核心是在决策树各个结点上应用**信息增益准则**选择信息增益最大且大于阈值的特征, 递归地构建决策树. ID3 相当于用极大似然法进行概率模型的选择. 由于算法只有树的生成, 所以容易产生过拟合.
2. **C4.5 算法**:C4.5 算法与 ID3 算法相似, 改用**信息增益比**来选择特征.

- **决策树的剪枝**:

1. 在学习时过多考虑如何提高对训练数据的正确分类, 从而构建出过于复杂的决策树, 产生**过拟合**现象. 解决方法是对已生成的决策树进行简化, 称为剪枝.
2. 设树的叶结点个数为 | T|, 每个叶结点有 Nt 个样本点, 其中 k 类样本点有 Ntk 个, 剪枝往往通过极小化决策树整体的损失函数 $C_\alpha(T) = \sum_{t=1}^{|T|} N_t H_t(T) + \alpha |T|$ 来实现, 其中经验熵

$H_t(T) = -\sum_k \frac{N_{tk}}{N_t} \log \frac{N_{tk}}{N_t}$. 剪枝通过加入 a|T | 项来考虑模型复杂度, 实际上就是用正则化的极大似然估计进行模型选择.

3. **剪枝算法**: 剪去某一子结点, 如果生成的新的整体树的**损失函数值**小于原树, 则进行剪枝, 直到不能继续为止. 具体可以由动态规划实现.

- **CART 算法**:

1. CART 既可以用于**分类也**可以用于**回归**. 它假设决策树是**二叉树**, 内部结点特征的取值为 "是" 和 "否". 递归地构建二叉树, 对回归树用**平方误差**最小化准则, 对分类数用**基尼指数**最小化准则.
2. **回归树的生成**: 在训练数据集所在的输入空间中, 递归地将每个区域划分为两个子区域. 选择第 j 个变量和它取的值 s 作为切分变量和切分点, 并定义两个区域

   $R_1(j,s) = \{x \mid x^{(j)} \leqslant s\}$ 和 $R_2(j,s) = \{x \mid x^{(j)} > s\}$, 遍历变量 j, 对固定的 j 扫描切分点 s, 求解

   $\min_{j,s}\left[\min_{c_1}\sum_{x_i \in R_1(j,s)}(y_i - c_1)^2 + \min_{c_2}\sum_{x_i \in R_2(j,s)}(y_i - c_2)^2\right]$. 用选定的对 (j,s) 划分区域并决定相应的输出值

   $\hat{c}_m = \frac{1}{N_m}\sum_{x_i \in R_m(j,s)} y_i, \quad x \in R_m, \quad m = 1,2$, 直到满足停止条件.
3. **基尼指数**: 假设有 K 个类, 样本属于第 k 类的概率为 pk, 则概率分布的基尼指数为

   $\text{Gini}(p) = \sum_{k=1}^{K} p_k(1-p_k) = 1 - \sum_{k=1}^{K} p_k^2$, 表示不确定性. 在特征 A 的条件下集合 D 的基尼指数定义为 $\text{Gini}(D,A) = \frac{|D_1|}{|D|}\text{Gini}(D_1) + \frac{|D_2|}{|D|}\text{Gini}(D_2)$, 表示分割后集合 D 的不确定性. 基尼指数越大, 样本集合的**不确定性**也就越大.
4. **分类树的生成**: 从根结点开始, 递归进行以下操作: 设结点的训练数据集为 D, 对每个特征 A 和其可能取的每个值 a, 计算 A=a 时的基尼指数, 选择**基尼指数最小**的特征及其对应的切分点作为**最优特征**与**最优切分点**, 生成两个子结点, 直至满足**停止条件**. 停止条件一般是结点中的样本个数小于阈值, 或样本集的基尼指数小于阈值, 或没有更多特征.
5. **CART 剪枝**:

   Tt 表示以 t 为根结点的子树, |Tt | 是 Tt 的叶结点个数. 可以证明当 $\alpha = \frac{C(t) - C(T_t)}{|T_t| - 1}$ 时, Tt 与 t 有相同的损失函数值, 且 t 的结点少, 因此 t 比 Tt 更可取, 对 Tt 进行剪枝. **自下而上**地对各内部结点 t 计算 $g(t) = \frac{C(t) - C(T_t)}{|T_t| - 1}$, 并令 a=min(g(t)), **自上而下**地访问内部节点 t, 如果有 g(t)=a, 进行剪枝, 并对 t 以**多数表决法**决定其类, 得到子树 T, 如此循环地生成一串**子树序列**, 直到新生成的 T 是由根结点单独构成的树为止. 利用**交叉验证法**在子树序列中选取最优子树.

- 如果是**连续值**的情况, 一般用**二分法**作为结点来划分.

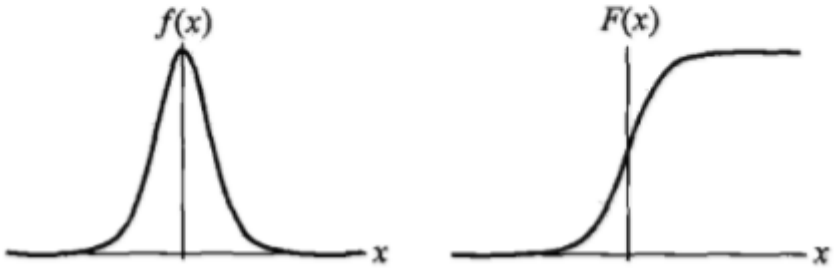
回到顶部

logistic 回归和最大熵模型

$$F(x)=P(X\leqslant x)=\frac{1}{1+\mathrm{e}^{-(x-\mu)/\gamma}}$$

$$f(x)=F'(x)=\frac{\mathrm{e}^{-(x-\mu)/\gamma}}{\gamma(1+\mathrm{e}^{-(x-\mu)/\gamma})^2}$$

- **逻辑斯谛分布**: 分布函



数 f(x)以点 (μ,1/2) 为中心对称,γ的值越小, 曲线在中心附近增长得越快.

- **逻辑斯谛回归模型**: 对于给定的输入 x, 根据 $P(Y=1\mid x)=\frac{\exp(w\cdot x+b)}{1+\exp(w\cdot x+b)}$

和 $P(Y=0\mid x)=\frac{1}{1+\exp(w\cdot x+b)}$ 计算出两个条件概率值的大小, 将 x 分到概率值较大的那一类. 将偏置 b 加入到权值向量 w 中, 并在 x 的最后添加常数项 1, 得到

$P(Y=1\mid x)=\frac{\exp(w\cdot x)}{1+\exp(w\cdot x)}$ 和 $P(Y=0\mid x)=\frac{1}{1+\exp(w\cdot x)}$.

如果某事件发生的概率是 p, 则该事件发生的**几率** (此处几率指该事件发生概率与不发生概率之比) 是 p/1-p, **对数几率**是 log(p/1-p), 那么 $\log\frac{P(Y=1\mid x)}{1-P(Y=1\mid x)}=w\cdot x$, 也就是说在逻辑斯谛回归模型中, 输出 Y=1 的对数几率是输入 x 的**线性函数**, 线性函数值越接近正无穷, 概率值就越接近 1, 反之则越接近 0.

- **似然估计**: 给定 x 的情况下参数θ是真实参数的可能性.
- **模型参数估计**: 对于给定的二分类训练数据集, 对数似然函数为

$$\begin{aligned}L(w)&=\sum_{i=1}^{N}[y_i\log\pi(x_i)+(1-y_i)\log(1-\pi(x_i))]\\&=\sum_{i=1}^{N}\left[y_i\log\frac{\pi(x_i)}{1-\pi(x_i)}+\log(1-\pi(x_i))\right]\\&=\sum_{i=1}^{N}[y_i(w\cdot x_i)-\log(1+\exp(w\cdot x_i))]\end{aligned}$$

, 也就是**损失函数**.

其中 P(Y=1|x)=π(x), 对 L(w) 求极大值, 就可以得到 w 的估计值. 问题变成了以对数似然函数为目标函数的最优化问题.

- **多项逻辑斯谛回归**: 当问题是多分类问题时, 可以作如下推广: 设 Y 有 K 类可能取值,

$$P(Y=k|x)=\frac{\exp(w_k\cdot x)}{1+\sum_{k=1}^{K-1}\exp(w_k\cdot x)},\quad k=1,2,\cdots,K-1$$

$$P(Y=K|x)=\frac{1}{1+\sum_{k=1}^{K-1}\exp(w_k\cdot x)}$$

, 实际上就是 **one-vs-all** 的思想, 将其他所有类当作一个类, 问题转换为二分类问题.

- **最大熵原理**: 学习概率模型时, 在所有可能的概率模型中, **熵最大**的模型是最好的模型. 直观地, 最大熵原理认为模型首先要满足已有的事实, 即**约束条件**. 在没有更多信息的情况下, 那些不确定的部分都是 " **等可能的** ".
- **最大熵模型**: 给定训练数据集, 可以确定联合分布 P(X,Y) 的经验分布

$$\tilde{P}(X=x,Y=y)=\frac{\nu(X=x,Y=y)}{N}$$

和边缘分布 P(X) 的经验分布

$$\tilde{P}(X=x)=\frac{\nu(X=x)}{N}$$

, 其中 v 表示频数, N 表示样本容量. 用**特征函数 f(x,y)**=1 描述 x 与 y 满足某一事实, 可以得到特征函数关于 P(X,Y) 的经验分布的期望值和关于模型 P(Y|X) 与 P(X) 的经验分布的期望值, 假设两者相等, 就得到了**约束条件**

$$\sum_{x,y}\tilde{P}(x)P(y|x)f(x,y)=\sum_{x,y}\tilde{P}(x,y)f(x,y)$$

. 定义在条件概率分布 P(Y|X) 上的条件熵为 $H(P)=-\sum_{x,y}\tilde{P}(x)P(y|x)\log P(y|x)$, 则**条件熵最大**的模型称为最大熵模型.

- **最大熵模型的学习**就是求解最大熵模型的过程. 等价于**约束最优化问题**

$$\max_{P\in C}\quad H(P)=-\sum_{x,y}\tilde{P}(x)P(y|x)\log P(y|x)$$

$$\text{s.t.}\quad E_P(f_i)=E_{\tilde{P}}(f_i),\quad i=1,2,\cdots,n$$

$$\sum_y P(y|x)=1$$

, 将求最大值问题改为

$$\min_{P\in C}\quad -H(P)=\sum_{x,y}\tilde{P}(x)P(y|x)\log P(y|x)$$

等价的求最小值问题

$$\text{s.t.}\quad E_P(f_i)-E_{\tilde{P}}(f_i)=0,\quad i=1,2,\cdots,n$$

$$\sum_y P(y|x)=1$$

. 引入**拉格朗日乘子**

$$L(P,w) \equiv -H(P) + w_0\left(1 - \sum_y P(y|x)\right) + \sum_{i=1}^{n} w_i (E_{\tilde{P}}(f_i) - E_P(f_i))$$

$$= \sum_{x,y} \tilde{P}(x)P(y|x)\log P(y|x) + w_0\left(1 - \sum_y P(y|x)\right)$$

$$+ \sum_{i=1}^{n} w_i \left(\sum_{x,y} \tilde{P}(x,y) f_i(x,y) - \sum_{x,y} \tilde{P}(x)P(y|x) f_i(x,y)\right)$$

将原始问题 $\min_{P\in C} \max_{w} L(P,w)$ 转换为无约束最优化的**对偶问题**

$\max_{w} \min_{P\in C} L(P,w)$. 首先求解内部的**极小化问题**, 即求 L(P,W) 对 P(y|x) 的偏导数

$$\frac{\partial L(P,w)}{\partial P(y|x)} = \sum_{x,y} \tilde{P}(x)(\log P(y|x) + 1) - \sum_y w_0 - \sum_{x,y}\left(\tilde{P}(x)\sum_{i=1}^{n} w_i f_i(x,y)\right)$$

$$= \sum_{x,y} \tilde{P}(x)\left(\log P(y|x) + 1 - w_0 - \sum_{i=1}^{n} w_i f_i(x,y)\right)$$

$$P_w(y|x) = \frac{1}{Z_w(x)} \exp\left(\sum_{i=1}^{n} w_i f_i(x,y)\right)$$

, 并令偏导数等于 0, 解得 . 可以证

$$Z_w(x) = \sum_y \exp\left(\sum_{i=1}^{n} w_i f_i(x,y)\right)$$

明对偶函数等价于对数似然函数, 那么对偶函数极大化等价于最大熵模型的**极大似然估计**

$L(w) = \sum_{x,y} \tilde{P}(x,y)\sum_{i=1}^{n} w_i f_i(x,y) - \sum_x \tilde{P}(x)\log Z_w(x)$. 之后可以

用最优化算法求解得到 w.

- 最大熵模型与逻辑斯谛回归模型有类似的形式, 它们又称为**对数线性模型**. 模型学习就是在给定的训练数据条件下对模型进行极大似然估计或正则化的极大似然估计.
- **算法**: 似然函数是**光滑的凸函数**, 因此多种最优化方法都适用.

1. **改进的迭代尺度法 (IIS)**: 假设当前的参数向量是 w, 如果能找到一种方法 **w->w+δ**使对数似然函数值变大, 就可以**重复**使用这一方法, 直到找到最大值.
2. 逻辑斯谛回归常应用梯度下降法, 牛顿法或拟牛顿法.

回到顶部

支持向量机

- **模型**: 支持向量机 (SVM) 是一种**二类分类模型**. 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. 支持向量机还包括**核技巧**, 使它成为实质上的非线性分类器. **分离超平面** $w^* \cdot x + b^* = 0$, **分类决策函数** $f(x) = \text{sign}(w^* \cdot x + b^*)$.

- **策略**: **间隔最大化**, 可形式化为一个求解**凸二次规划**的问题, 也等价于正则化的**合页损失函数**的最小化问题.
- 当训练数据**线性可分**时, 通过硬间隔最大化, 学习出**线性可分支持向量机**. 当训练数据**近似线性可分**时, 通过软间隔最大化, 学习出**线性支持向量机**. 当训练数据**线性不可分**时, 通过使用核技巧及软间隔最大化, 学习**非线性支持向量机**.
- **核技巧**: 当输入空间为欧式空间或离散集合, 特征空间为希尔伯特空间时, 核函数表示将输入从输入空间**映射**到特征空间得到的特征向量之间的**内积**. 通过核函数学习非线性支持向量机等价于在高维的特征空间中学习线性支持向量机. 这样的方法称为核技巧.
- 考虑一个二类分类问题, 假设输入空间与特征空间为两个不同的空间, 输入空间为**欧氏空间或离散集合**, 特征空间为**欧氏空间或希尔伯特空间**. 支持向量机都将输入映射为特征向量, 所以支持向量机的学习是在**特征空间**进行的.
- 支持向量机的最优化问题一般通过对偶问题化为**凸二次规划问题**求解, 具体步骤是将等式约束条件代入优化目标, 通过求偏导求得优化目标在不等式约束条件下的极值.
- **线性可分支持向量机**:

1. 当训练数据集线性可分时, 存在无穷个分离超平面可将两类数据正确分开. 利用**间隔最大化**得到**唯一**最优分离超平面 $w^* \cdot x + b^* = 0$ 和相应的分类决策函数 $f(x) = \text{sign}(w^* \cdot x + b^*)$ 称为线性可分支持向量机.

2. **函数间隔**: 一般来说, 一个点距离分离超平面的**远近**可以表示分类预测的**确信程度**. 在超平面 $w \cdot x + b = 0$ 确定的情况下, |wx+b | 能够相对地表示点 x 距离超平面的远近, 而 wx+b 与 y 的符号是否一致能够表示分类是否正确. 所以可用 $\hat{\gamma}_i = y_i(w \cdot x_i + b)$ 来表示分类的正确性及确信度, 这就是**函数间隔**. 注意到即使超平面不变, 函数间隔仍会受 w 和 b 的绝对大小影响.

3. **几何间隔**: 一般地, 当样本点被超平面正确分类时, 点 x 与超平面的距离是 $\gamma_i = y_i\left(\frac{w}{\|w\|} \cdot x_i + \frac{b}{\|w\|}\right)$, 其中 ||w|| 是 w 的 **l2 范数**. 这就是**几何间隔**的定义.

   定义超平面关于训练数据集 T 的几何间隔为超平面关于 T 中所有样本点的几何间隔之**最小值** $\gamma = \min_{i=1,\cdots,N} \gamma_i$. 可知 $\gamma = \frac{\hat{\gamma}}{\|w\|}$, 当 ||w||=1 时几何间隔和函数间隔**相等**.

4. **硬间隔最大化**: 对线性可分的训练集而言, 这里的间隔最大化又称为**硬间隔最大化**. 直观解释是对训练集找到几何间隔最大的超平面意味着以**充分大的确信度**对训练数据进行分类. 求最大间隔分离超平面即约束最优化问题:

$$\max_{w,b} \quad \gamma$$

$$\text{s.t.} \quad y_i\left(\frac{w}{\|w\|} \cdot x_i + \frac{b}{\|w\|}\right) \geqslant \gamma, \quad i = 1,2,\cdots,N$$

   , 将几何间隔用函数间隔表示 $\max_{w,b} \quad \frac{\hat{\gamma}}{\|w\|}$, 并且注意到函

$$\text{s.t.} \quad y_i(w \cdot x_i + b) \geqslant \hat{\gamma}, \quad i = 1,2,\cdots,N$$

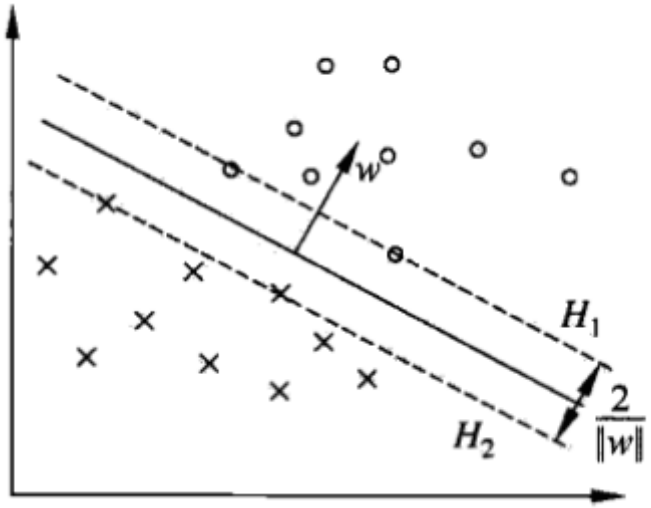
   数间隔的取值并不影响最优化问题的解, 不妨令函数间隔 = 1, 并让最大化 1/||w|| 等价为最小化

||w||^2/2, 问题变为**凸二次规划问题**

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2$$

$$\text{s.t.} \quad y_i(w \cdot x_i + b) - 1 \geqslant 0, \quad i = 1,2,\cdots,N$$

5. **支持向量和间隔边界**: 与分离超平面距离**最近的样本点**的实例称为**支持向量**. 支持向量是使最优化问题中的约束条件等号成立的点. 因此对 y=+1 的正例点和 y=-1 的负例点, 支持向量分别在超平面 H1:wx+b=+1 和 H2:wx+b=-1.H1 和 H2 平行, 两者之间形成一条长带, 长带的宽度$\frac{2}{\|w\|}$称为**间隔**,H1 和 H2 称为**间隔边界**. 在决定分离超平面时只有支持向量起作用, 所以支持向量机是由很少的"重要的" 训练样本确定的. 由对偶问题同样可以得到支持向量一定在间隔边界上.



6. **对偶算法**: 引进拉格朗日乘子, 定义拉格朗日函数

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

, 根据拉格朗日对偶性, 原始问题的对偶问题是极大极小问题: $\max_{\alpha} \min_{w,b} L(w,b,\alpha)$. 先求对 w,b 的**极小值**. 将 L(w,b,a) 分别对 w,b 求偏导数并令其等于 0, 得

$$w = \sum_{i=1}^{N}\alpha_i y_i x_i$$

, 代入拉格朗日函数

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

得

$$L(w,b,\alpha)=\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(x_i\cdot x_j)-\sum_{i=1}^{N}\alpha_i y_i\left(\left(\sum_{j=1}^{N}\alpha_j y_j x_j\right)\cdot x_i+b\right)+\sum_{i=1}^{N}\alpha_i$$

$$=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(x_i\cdot x_j)+\sum_{i=1}^{N}\alpha_i$$

, 这就是极小值. 接下来对极小值求对 a 的极大, 即是**对偶问题**

$$\max_{\alpha}\ -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(x_i\cdot x_j)+\sum_{i=1}^{N}\alpha_i$$

$$\text{s.t.}\quad \sum_{i=1}^{N}\alpha_i y_i=0$$

. 将求极大转换为求极小

$$\alpha_i\geqslant 0,\quad i=1,2,\cdots,N$$

$$\min_{\alpha}\quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(x_i\cdot x_j)-\sum_{i=1}^{N}\alpha_i$$

$$\text{s.t.}\quad \sum_{i=1}^{N}\alpha_i y_i=0$$

. 由 **KKT 条件**成立得到

$$\alpha_i\geqslant 0,\quad i=1,2,\cdots,N$$

$$w^*=\sum_{i=1}^{N}\alpha_i^* y_i x_i$$

, 其中 j 为使 aj>*0 的下标之一. 所以问题就变为求对*

$$b^*=y_j-\sum_{i=1}^{N}\alpha_i^* y_i(x_i\cdot x_j)$$

偶问题的解 a, 再求得原始问题的解 w,*b*, 从而得分离超平面及分类决策函数可以看出 w * 和 b * 都只依赖训练数据中 ai*>0 的样本点 (xi,yi), 这些实例点 xi 被称为**支持向量**.

- **线性支持向量机**:

1. 如果训练数据是**线性不可分**的, 那么上述方法中的不等式约束并不能都成立, 需要修改硬间隔最大化, 使其成为**软间隔最大化**.
2. 线性不可分意味着某些**特异点**不能满足函数间隔大于等于 1 的约束条件, 可以对每个样本点引进一个**松弛变量**, 使函数间隔加上松弛变量大于等于 1, 约束条件变为

$y_i(w\cdot x_i+b)\geqslant 1-\xi_i$, 同时对每个松弛变量, 支付一个代价, 目标函数变为

$\frac{1}{2}\|w\|^2+C\sum_{j=1}^{N}\xi_i$, 其中 C>0 称为**惩罚参数**,C 值越大对误分类的惩罚也越大. 新目标函数包含了两层含义: 使**间隔尽量大**, 同时使误分类点的**个数尽量小**.

3. **软间隔最大化**: 学习问题变成如下**凸二次规划**问题:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{s.t.} \quad y_i(w\cdot x_i + b) \geqslant 1-\xi_i, \quad i=1,2,\cdots,N$$

$$\xi_i \geqslant 0, \quad i=1,2,\cdots,N$$

, 可以证明 w 的解是唯一的, 但 b 的解存在一个**区间**. 线性支持向量机包含线性可分支持向量机, 因此**适用性更广**.

4. **对偶算法**: 原始问题的对偶问题是, 构造**拉格朗日函数**

$$L(w,b,\xi,\alpha,\mu) \equiv \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(w\cdot x_i+b)-1+\xi_i) - \sum_{i=1}^{N}\mu_i\xi_i$$

$$w = \sum_{i=1}^{N}\alpha_i y_i x_i$$

, 先求对 w,b,ξ的**极小值**, 分别求偏导并令导数为 0, 得 $\sum_{i=1}^{N}\alpha_i y_i = 0$, 代入原函数, 再对极小值求 a 的**极大值**, 得到

$$C-\alpha_i-\mu_i = 0$$

$$\max_{\alpha} \quad -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i\cdot x_j) + \sum_{i=1}^{N}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{N}\alpha_i y_i = 0$$

, 利用后三条约束**消去μ**, 再将

$$C-\alpha_i-\mu_i = 0$$

$$\alpha_i \geqslant 0$$

$$\mu_i \geqslant 0, \quad i=1,2,\cdots,N$$

求极大转换为**求极小**, 得到**对偶问题**

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i\cdot x_j) - \sum_{i=1}^{N}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{N}\alpha_i y_i = 0$$

. 由 **KKT 条件**成立可以得到

$$0 \leqslant \alpha_i \leqslant C, \quad i=1,2,\cdots,N$$

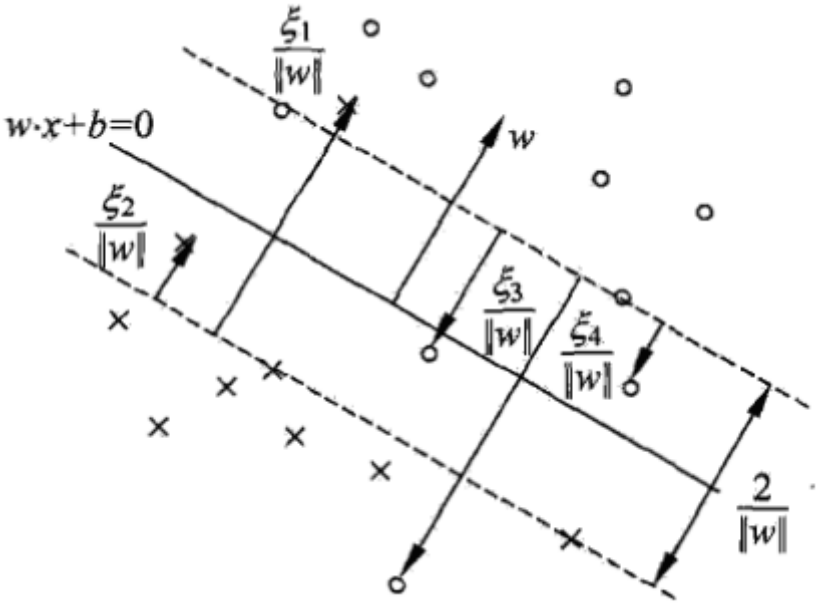
$$w^* = \sum_{i=1}^{N} \alpha_i^* y_i x_i$$

,j 是满足 0<aj<C *的下标之一. 问题就变为选择惩罚参*

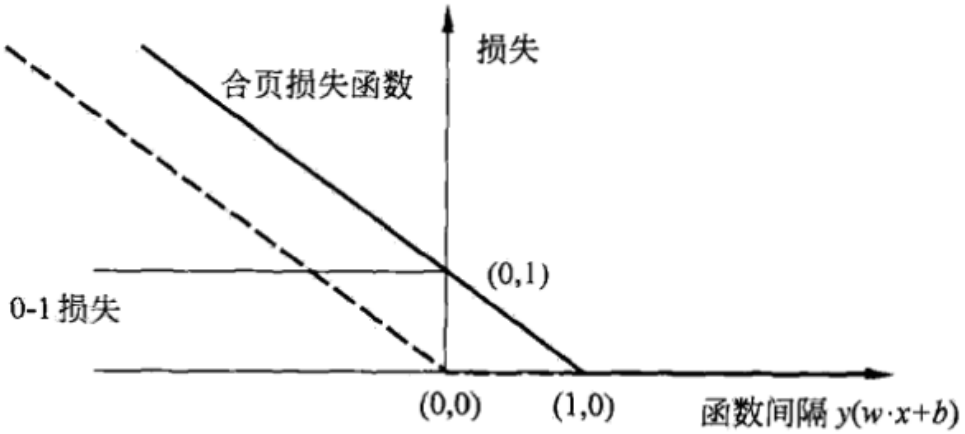
$$b^* = y_j - \sum_{i=1}^{N} y_i \alpha_i^* (x_i \cdot x_j)$$

*数 C>0, 求得对偶问题 (**凸二次规划问题**) 的**最优解 α***, 代入计算 w * 和 b*, 求得分离超平面和分类决策函数. 因为 b 的解并不唯一, 所以实际计算 b * 时可以取所有样本点上的**平均值**.

5. **支持向量**: 在**线性不可分**的情况下, 将对应与 ai*>0 的样本点 (xi,yi) 的实例点 xi 称为**支持向量**. 软间隔的支持向量或者在间隔边界上, 或者在间隔边界与分类超平面之间, 或者再分离超平面误分一侧.



6. **合页损失函数**: 可以认为是 0-1 损失函数的上界, 而线性支持向量机可以认为是优化合页损失函数构成的目标函数.



- **非线性支持向量机**:

1. 如果分类问题是**非线性**的, 就要使用**非线性支持向量机**. 主要特点是使用**核技巧**.
2. **非线性分类问题**: 用线性分类方法求解非线性分类问题分为两步: 首先使用一个变换将原空间的数据映射到新空间, 然后在新空间里用线性分类学习方法从训练数据中学习分类模型.
3. **核函数**: 设 X 是**输入空间** (欧式空间的子集或离散集合),H 为**特征空间** (希尔伯特空间), 一般是**高维**甚至无穷维的. 如果存在一个从 X 到 H 的映射 $\phi(x): \mathcal{X} \to \mathcal{H}$ 使得对所有 x,z 属于 X, 函数 K(x,z) 满足条件 $K(x,z) = \phi(x) \cdot \phi(z)$, 点乘代表**内积**, 则称 K(x,z) 为**核函数**.

4. **核技巧**: 基本思想是通过一个**非线性变换**将输入空间对应于一个**特征空间**, 使得在输入空间中的**超曲面模型**对应于特征空间中的**超平面模型** (支持向量机). 在学习和预测中只定义核函数 K(x,z), 而**不显式**地定义映射函数. 对于给定的核 K(x,z), 特征空间和映射函数的取法并**不唯一**. 注意到在线性支持向量机的对偶问题中, 目标函数和决策函数都只涉及输入实例与实例之间的**内积**,xi`xj` 可以用核函数 `K(xi,xj)=Φ(xi)` Φ(xj) 来**代替**. 当映射函数是非线性函数时, 学习到的含有核函数的支持向量机是非线性分类模型. 在实际应用中, 往往依赖领域知识**直接选择**核函数.
5. **正定核**: 通常所说的核函数是指**正定核函数**. 只要满足正定核的充要条件, 那么给定的函数 K(x,z) 就是正定核函数. 设 K 是定义在 X*X 上的**对称函数**, 如果任意 xi 属于 X,K(x,z) 对应的 **Gram 矩阵** $K=\left[K(x_i,x_j)\right]_{m\times m}$ 是**半正定矩阵**, 则称 K(x,z) 是正定核. 这一定义在构造核函数时很有用, 但要验证一个具体函数是否为正定核函数并不容易, 所以在实际问题中往往应用已有的核函数.
6. **算法**: 选取适当的核函数 K(x,z) 和适当的参数 C, 将线性支持向量机对偶形式中的内积换成核函数,

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i,x_j)-\sum_{i=1}^{N}\alpha_i$$

构造并求解最优化问题 , 选择

$$\text{s.t.} \quad \sum_{i=1}^{N}\alpha_i y_i=0$$

$$0\leqslant\alpha_i\leqslant C, \quad i=1,2,\cdots,N$$

最优解 a * 的一个正分量 0<aj*<C 计算 $b^*=y_j-\sum_{i=1}^{N}\alpha_i^* y_i K(x_i\cdot x_j)$, 构造决策函数 $f(x)=\text{sign}\left(\sum_{i=1}^{N}\alpha_i^* y_i K(x\cdot x_i)+b^*\right)$.

- **常用核函数**:

1. **多项式核函数 (polynomial kernel function)**:$K(x,z)=(x\cdot z+1)^p$, 对应的支持向量机是一个 p 次多项式分类器, 分类决策函数为

$$f(x)=\text{sign}\left(\sum_{i=1}^{N_s}a_i^* y_i(x_i\cdot x+1)^p+b^*\right) .$$

2. **高斯核函数 (Gaussian krenel function)**: $K(x,z)=\exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$, 对应的支持向量机是高斯径向基函数 (RBF) 分类器. 分类决策函数为

$$f(x)=\text{sign}\left(\sum_{i=1}^{N_s}a_i^* y_i\exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)+b^*\right) .$$

3. **字符串核函数 (string kernel function)**: 核函数不仅可以定义在欧氏空间上, 还可以定义在**离散数据的集合**上. 字符串核函数给出了字符串中长度等于 n 的所有子串组成的特征向量的余弦相似度.

- **序列最小最优化 (SMO) 算法**:

1. SMO 是一种快速求解凸二次规划问题

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i,x_j)-\sum_{i=1}^{N}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{N}\alpha_i y_i = 0$$

$$0 \leqslant \alpha_i \leqslant C, \quad i=1,2,\cdots,N$$

的算法. 基本思路是: 如果所有变量都满足此优化问题的 KKT 条件, 那么解就得到了. 否则, 选择**两个变量**, 固定其他变量, 针对这两个变量构建一个二次规划问题. 不断地将原问题分解为**子问题**并对子问题求解, 就可以求解原问题. 注意子问题两个变量中只有一个是**自由变量**, 另一个由**等式约束**确定.

2. **两个变量二次规划的求解方法**: 假设选择的两个变量是 a1,a2, 其他变量是固定的, 于是得到子问题

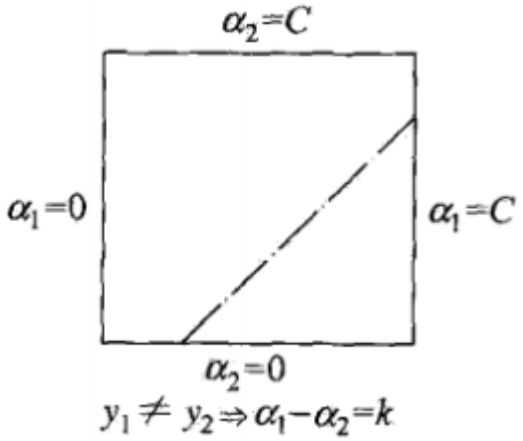
$$\min_{\alpha_1,\alpha_2} \quad W(\alpha_1,\alpha_2)=\frac{1}{2}K_{11}\alpha_1^2+\frac{1}{2}K_{22}\alpha_2^2+y_1y_2K_{12}\alpha_1\alpha_2$$
$$-(\alpha_1+\alpha_2)+y_1\alpha_1\sum_{i=3}^{N}y_i\alpha_iK_{i1}+y_2\alpha_2\sum_{i=3}^{N}y_i\alpha_iK_{i2}$$

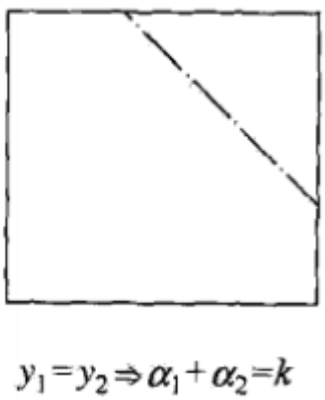
,ε

$$\text{s.t.} \quad \alpha_1y_1+\alpha_2y_2=-\sum_{i=3}^{N}y_i\alpha_i=\varsigma$$

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

是常数, 目标函数式省略了不含 a1,a2 的常数项. 考虑不等式约束和等式约束, 要求的是目标函数在一条平行于对角线的线段上的最优值





, 问题变为**单变量**的最优化问题. 假设初始可行解为 aold, 最优解为 anew, 考虑沿着约束方向未经剪辑的最优解 anew,unc(即未考虑不等式约束). 对该问题求偏导数, 并令导数为 0, 代入原式, 令

$$E_i=g(x_i)-y_i=\left(\sum_{j=1}^{N}\alpha_jy_jK(x_j,x_i)+b\right)-y_i, \quad i=1,2$$

, 得到 $\alpha_2^{\text{new,unc}}=\alpha_2^{\text{old}}+\frac{y_2(E_1-E_2)}{\eta}$, 经剪辑后 a2 的解是

$$\alpha_2^{new}=\begin{cases}H, & \alpha_2^{new,unc}>H\\ \alpha_2^{new,unc}, & L\leqslant\alpha_2^{new,unc}\leqslant H\\ L, & \alpha_2^{new,unc}<L\end{cases}$$,L 与 H 是 a2new 所在的对角线段端点的界. 并解得 $\alpha_1^{new}=\alpha_1^{old}+y_1y_2(\alpha_2^{old}-\alpha_2^{new})$.

3. **变量的选择方法**: 在每个子问题中选择两个变量优化, 其中至少一个变量是违反 KKT 条件的. 第一个变量的选取标准是**违反 KKT 条件最严重**的样本点, 第二个变量的选取标准是希望能使该变量有**足够大的变化**, 一般可以选取使对应的 | E1-E2 | 最大的点. 在每次选取完点后, **更新**阈值 b 和差值 Ei.

回到顶部

提升方法

- **提升 (boosting) 是**一种常用的统计学习方法, 是集成学习的一种. 它通过改变训练样本的权重 (概率分布), 学习**多个**弱分类器 (基本分类器), 并将这些分类器**线性组合**来构成一个强分类器提高分类的性能.
- **AdaBoost**:

1. AdaBoost 提高那些被前一轮弱分类器错误分类样本的权值, 而降低那些被正确分类样本的权值. 然后采取**加权多数表决**的方法组合弱分类器.
2. **算法**: 首先假设训练数据集具有均匀的权值分布 D1, 使用具有**权值分布** Dm 的训练数据集学习得到**基本分类器** Gm(x), 计算**分类误差率**

   $e_m=P(G_m(x_i)\neq y_i)=\sum_{i=1}^{N}w_{mi}I(G_m(x_i)\neq y_i)$ 和 Gm(x) 的**系数**

   $\alpha_m=\frac{1}{2}\log\frac{1-e_m}{e_m}$, 更新训练数据集的权值分布

   $D_{m+1}=(w_{m+1,1},\cdots,w_{m+1,i},\cdots,w_{m+1,N})$, 其中

   $$w_{m+1,i}=\begin{cases}\frac{w_{mi}}{Z_m}e^{-\alpha_m}, & G_m(x_i)=y_i\\ \frac{w_{mi}}{Z_m}e^{\alpha_m}, & G_m(x_i)\neq y_i\end{cases}$$ Zm 是使 Dm+1 成为概率分布的**规范化因子** $Z_m=\sum_{i=1}^{N}w_{mi}\exp(-\alpha_m y_i G_m(x_i))$. 重复上述操作 M 次后得到 M 个弱分类器, 构建线性组合得到**最终分类器**

   $$G(x)=\text{sign}(f(x))=\text{sign}\left(\sum_{m=1}^{M}\alpha_m G_m(x)\right)$$.

3. AdaBoost 算法也可以理解成模型为加法模型, 损失函数为指数函数, 学习算法为**前向分步算法**的二类分类学习方法.

- **前向分步算法**: 考虑加法模型 $f(x)=\sum_{m=1}^{M}\beta_m b(x;\gamma_m)$, 其中 b(x,γm) 为基函数,γm 为基函数的参数,βm 为基函数的系数. 在给定损失函数 L(y,f(x)) 的条件下, 学习加法模型就是求解损失函数极小化问题 $\min_{\beta_m,\gamma_m}\sum_{i=1}^{N}L\left(y_i,\sum_{m=1}^{M}\beta_m b(x_i;\gamma_m)\right)$ 前向分步算法求解的想法是: **从前往后**, 每一步只学习一个基函数及其系数, 优化

  $(\beta_m,\gamma_m)=\arg\min_{\beta,\gamma}\sum_{i=1}^{N}L\left(y_i,f_{m-1}(x_i)+\beta b(x_i;\gamma)\right)$, 得到参数βm和γm, 更新 $f_m(x)=f_{m-1}(x)+\beta_m b(x;\gamma_m)$, 逐步逼近优化目标. 最终得到加法模型.
- **提升树**:

1. 提升树是模型为加法模型, 算法为前向分布算法, 基函数为**决策树**的提升方法. 第 m 步的模型是 $f_m(x)=f_{m-1}(x)+T(x;\Theta_m)$, 通过经验风险极小化确定下一棵决策树的参数 $\hat{\Theta}_m=\arg\min_{\Theta_m}\sum_{i=1}^{N}L(y_i,f_{m-1}(x_i)+T(x_i;\Theta_m))$. 不同问题的提升树学习算法主要区别在于使用的**损失函数**不同.
2. **二类分类问题**: 只需将 AdaBoost 算法中的基本分类器限制为二类分类数即可.
3. **回归问题**: 如果将输入空间划分为 J 个互不相交的区域, 并且在每个区域上确定输出的常量 Cj, 那么树可表示为 $T(x;\Theta)=\sum_{j=1}^{J}c_j I(x\in R_j)$, 其中 $\Theta=\{(R_1,c_1),(R_2,c_2),\cdots,(R_J,c_J)\}$. 提升树采用**前向分步算法**:

   $$f_0(x)=0$$
   $$f_m(x)=f_{m-1}(x)+T(x;\Theta_m),\quad m=1,2,\cdots,M$$
   $$f_M(x)=\sum_{m=1}^{M}T(x;\Theta_m)$$

   . 当采用平方误差损失函数时, 损失变为 $L(y,f_{m-1}(x)+T(x;\Theta_m))$ $=[y-f_{m-1}(x)-T(x;\Theta_m)]^2$ $=[r-T(x;\Theta_m)]^2$, 其中 r 是当前模型拟合数据的**残差**. 每一步都只需**拟合残差**学习一个回归树即可.
4. **梯度提升树 (GBDT)**: 利用最速下降法的近似方法来实现每一步的优化, 关键在于用损失函数的**负梯度**在当前模型的值 $-\left[\frac{\partial L(y,f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$ 作为回归问题中提升树算法中的

残差的**近似值**, 每一步以此来估计回归树叶结点区域以拟合残差的近似值, 并利用线性搜索估计叶结点区域的值使损失函数最小化, 然后更新回归树即可.

- AdaBoost 产生的基础学习器有好有坏, 因此加入权重. 提升树产生的基础学习器是一个不断减少残差的过程, 并不是一个单独的分类器, 因此一般不加权重.
- **XGBoost**: 相比传统 GBDT 有以下优点:

1. 在优化时用到了二阶导数信息.
2. 在代价函数里加入了正则项.
3. 每次迭代后都将叶子结点的权重乘上一个系数, 削弱每棵树的影响.
4. 列抽样.
5. 在训练前对数据进行排序, 保存为 block 结构, 并行地对各个特征进行增益计算.

回到顶部

EM 算法

- EM 算法是一种**迭代**算法, 用于含有**隐变量**的概率模型参数的极大似然估计. 每次迭代由两步组成: E 步, 求**期望** (expectation),M 步, 求**极大值** (maximization), 直至收敛为止.
- **隐变量**: 不能被直接观察到，但是对系统的状态和能观察到的输出存在影响的一种东西.
- **算法**:

1. 选择参数的初始值θ(0), 开始迭代. 注意 EM 算法对初值是**敏感**的.
2. **E 步**:θ(i) 为第 i 次迭代参数θ的估计值, 在第 i+1 次迭代的 E 步, 计算

$$Q(\theta,\theta^{(i)}) = E_Z[\log P(Y,Z\mid\theta)\mid Y,\theta^{(i)}] = \sum_Z \log P(Y,Z\mid\theta)P(Z\mid Y,\theta^{(i)})$$

,P(Z|Y,θ(i)) 是在给定**观测数据** Y 和当前参数估计θ(i) 下**隐变量数据** Z 的条件概率分布.
3. **M 步**: 求使 Q(θ,θ(i)) 极大化的θ, 确定第 i+1 次迭代的参数的估计值

$$\theta^{(i+1)} = \arg\max_\theta Q(\theta,\theta^{(i)})$$

4. 重复 2 和 3 直到**收敛**, 一般是对较小的正数ε1 和ε2 满足

$$\|\theta^{(i+1)}-\theta^{(i)}\|<\varepsilon_1 \quad 或 \quad \|Q(\theta^{(i+1)},\theta^{(i)})-Q(\theta^{(i)},\theta^{(i)})\|<\varepsilon_2$$

则停止迭代.

- EM 算法是通过不断求解**下界**的极大化逼近求解对数似然函数极大化的算法. 可以用于生成模型的**非监督学习**. 生成模型由联合概率分布 P(X,Y) 表示. X 为观测数据, Y 为未观测数据.
- **高斯混合模型 (GMM)**: 高斯混合模型是指具有如下形式的概率分布模型:

$$P(y\mid\theta)=\sum_{k=1}^{K}\alpha_k\phi(y\mid\theta_k)$$. 其中

α_k 是系数，$\alpha_k\geqslant 0$，$\sum_{k=1}^{K}\alpha_k=1$；$\phi(y\mid\theta_k)$ 是高斯分布密度，$\theta_k=(\mu_k,\sigma_k^2)$，

，$\phi(y\mid\theta_k)=\frac{1}{\sqrt{2\pi}\sigma_k}\exp\left(-\frac{(y-\mu_k)^2}{2\sigma_k^2}\right)$ 称为第 k 个分模型.

- **高斯混合模型参数估计的 EM 算法**:

1. 取参数的初始值开始迭代

2. **E 步**: 计算分模型 k 对观测数据 yj 的**响应度**

$$\hat{\gamma}_{jk}=\frac{\alpha_k\phi(y_j\mid\theta_k)}{\sum_{k=1}^{K}\alpha_k\phi(y_j\mid\theta_k)},\qquad j=1,2,\cdots,N;\quad k=1,2,\cdots,K$$

3. **M 步**: 计算新一轮迭代的模型参数

$$\hat{\mu}_k=\frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}y_j}{\sum_{j=1}^{N}\hat{\gamma}_{jk}},\qquad k=1,2,\cdots,K$$

$$\hat{\sigma}_k^2=\frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}(y_j-\mu_k)^2}{\sum_{j=1}^{N}\hat{\gamma}_{jk}},\qquad k=1,2,\cdots,K$$

$$\hat{\alpha}_k=\frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}}{N},\qquad k=1,2,\cdots,K$$

4. 重复 2 和 3 直到对数似然函数

$$\log P(y,\gamma\mid\theta)=\sum_{k=1}^{K}n_k\log\alpha_k+\sum_{j=1}^{N}\gamma_{jk}\left[\log\left(\frac{1}{\sqrt{2\pi}}\right)-\log\sigma_k-\frac{1}{2\sigma_k^2}(y_j-\mu_k)^2\right]$$

收敛.

回到顶部

隐马尔可夫模型 (HMM)

- 隐马尔可夫模型是关于**时序**的概率模型, 描述由一个隐藏的马尔可夫链随机生成不可观测的**状态序列**, 再由各个状态生成一个观测而产生**观测随机序列**的过程.
- 设 Q 是所有可能的状态的集合, V 是所有可能的观测的集合

  $Q=\{q_1,q_2,\cdots,q_N\}$, $V=\{v_1,v_2,\cdots,v_M\}$,I 是长度为 T 的状态序列, O 是对应的观测序列 $I=(i_1,i_2,\cdots,i_T)$, $O=(o_1,o_2,\cdots,o_T)$,A 是**状态转移概率矩阵** $A=\left[a_{ij}\right]_{N\times N}$,aij 表示在时刻 t 处于状态 qi 的条件下在时刻 t+1 转移到状态 qj 的概率. B 是**观测概率矩阵** $B=\left[b_j(k)\right]_{N\times M}$,bij 是在时刻 t 处于状态 qj 的条件下生成观测 vk 的概率.π是**初始状态概率向量** $\pi=(\pi_i)$,πi 表示时刻 t=1 处于状态 qi 的概率. 隐马尔可夫模型由初始状态概率向量π, 状态转移概率矩阵 A 以及观测概率矩阵 B 确定.π和 A 决定即隐藏的**马尔可夫链**, 生成不可观测的**状态序列**.B 决定如何从状态生成观测, 与状态序列综合确定了**观测序列**. 因此, 隐马尔可夫模型可以用**三元符号**表示 $\lambda=(A,B,\pi)$.

- 隐马尔可夫模型作了两个**基本假设**:

1. **齐次马尔可夫性假设**: 假设隐藏的马尔可夫链在任意时刻 t 的状态只依赖于其前一时刻的状态.
2. **观测独立性假设**: 假设任意时刻的观测只依赖于该时刻的马尔可夫链的状态.

- 隐马尔可夫模型有三个基本问题, 即概率计算问题, 学习问题, 预测问题.
- **概率计算问题**: 给定模型 $\lambda=(A,B,\pi)$ 和观测序列 $O=(o_1,o_2,\cdots,o_T)$, 计算在模型λ下观测序列 O 出现的概率 P(O|λ).

1. **直接计算法**: 最直接的方法是列举所有可能长度为 T 的状态序列, 求各个状态序列 I 与观测序列 O 的联合概率, 但计算量太大, 实际操作不可行.
2. **前向算法**: 定义到时刻 t 部分观测序列为 o1~ot 且状态为 qi 的概率为**前向概率**, 记作

   $\alpha_t(i)=P(o_1,o_2,\cdots,o_t,i_t=q_i\mid\lambda)$. 初始化前向概率

   $\alpha_1(i)=\pi_i b_i(o_1),\quad i=1,2,\cdots,N$, 递推, 对 t=1~T-1,

   $\alpha_{t+1}(i)=\left[\sum_{j=1}^{N}\alpha_t(j)a_{ji}\right]b_i(o_{t+1}),\quad i=1,2,\cdots,N$, 得到

   $P(O\mid\lambda)=\sum_{i=1}^{N}\alpha_T(i)$. 减少计算量的原因在于每一次计算直接引用前一个时刻的计算结果, 避免重复计算.
3. **后向算法**: 定义在时刻 t 状态为 qi 的条件下, 从 t+1 到 T 的部分观测序列为 oi+1~oT 的概率为**后向概率**, 记作 $\beta_t(i)=P(o_{t+1},o_{t+2},\cdots,o_T\mid i_t=q_i,\lambda)$. 初始化后向概率

   $\beta_T(i)=1,\quad i=1,2,\cdots,N$, 递推, 对 t=T-1~1,

   $\beta_t(i)=\sum_{j=1}^{N}a_{ij}b_j(o_{t+1})\beta_{t+1}(j),\quad i=1,2,\cdots,N$, 得到

   $P(O\mid\lambda)=\sum_{i=1}^{N}\pi_i b_i(o_1)\beta_1(i)$.

- **学习算法**: 已知观测序列$O=(o_1,o_2,\cdots,o_T)$, 估计模型 $\lambda=(A,B,\pi)$的参数, 使得在该模型下观测序列概率 P(O|λ) 最大. 根据训练数据是否包括观察序列对应的状态序列分别由监督学习与非监督学习实现.

1. **监督学习**: 估计转移概率

   $\hat{a}_{ij}=\dfrac{A_{ij}}{\sum_{j=1}^{N}A_{ij}},\quad i=1,2,\cdots,N;\ j=1,2,\cdots,N$ 和观测概率

   $\hat{b}_j(k)=\dfrac{B_{jk}}{\sum_{k=1}^{M}B_{jk}},\quad j=1,2,\cdots,N;\ k=1,2,\cdots,M$. 初始状态概率πi 的估计为 S 个样本中初始状态为 qi 的频率.

2. **非监督学习 (Baum-Welch 算法)**: 将观测序列数据看作观测数据 O, 状态序列数据看作不可观测的**隐数据** I. 首先确定完全数据的对数似然函数: $\log P(O,I|\lambda)$. 求 Q 函数

$$Q(\lambda,\bar{\lambda})=\sum_I \log \pi_{i_1} P(O,I|\bar{\lambda}) + \sum_I\left(\sum_{t=1}^{T-1}\log a_{i_t i_{t+1}}\right)P(O,I|\bar{\lambda})+\sum_I\left(\sum_{t=1}^{T}\log b_{i_t}(o_t)\right)P(O,I|\bar{\lambda})$$

, 用拉格朗日乘子法极大化 Q 函数求模型参数 $\pi_i=\frac{P(O,i_1=i|\bar{\lambda})}{P(O|\bar{\lambda})}$,

$$a_{ij}=\frac{\sum_{t=1}^{T-1}P(O,i_t=i,i_{t+1}=j|\bar{\lambda})}{\sum_{t=1}^{T-1}P(O,i_t=i|\bar{\lambda})},$$

$$b_j(k)=\frac{\sum_{t=1}^{T}P(O,i_t=j|\bar{\lambda})I(o_t=v_k)}{\sum_{t=1}^{T}P(O,i_t=j|\bar{\lambda})}.$$

- **预测问题**: 也称为解码问题. 已知模型 $\lambda=(A,B,\pi)$和观测序列

  $O=(o_1,o_2,\cdots,o_T)$, 求对给定观测序列条件概率 P(I|O) 最大的状态序列

  $I=(i_1,i_2,\cdots,i_T)$.

1. **近似算法**: 在每个时刻 t 选择在该时刻最有可能出现的状态 it*, 从而得到一个状态序列作为预测的结果. 优点是**计算简单**, 缺点是不能保证状态序列整体是最有可能的状态序列.
2. **维特比算法**: 用**动态规划**求概率最大路径, 这一条路径对应着一个状态序列. 从 t=1 开始, 递推地计算在时刻 t 状态为 i 的各条部分路径的最大概率, 直至得到时刻 t=T 状态为 i 的各条路径的最大概率. 时刻 t=T 的最大概率即为**最优路径**的概率 P*, 最优路径的**终结点** it * 也同时得到, 之后从终结点开始由后向前逐步求得**结点**, 得到最优路径.

回到顶部

统计学习方法总结

方法	适用问题	模型特点	模型类型	学习策略	学习的损失函数	学习算法
感知机	二类分类	分离超平面	判别模型	极小化误分点到超平面距离	误分点到超平面距离	随机梯度下降
k 近邻法	多类分类，回归	特征空间，样本点	判别模型			
朴素贝叶斯法	多类分类	特征与类别的联合概率分布，条件独立假设	生成模型	极大似然估计，极大后验概率估计	对数似然损失	概率计算公式，EM 算法
决策树	多类分类，回归	分类树，回归树	判别模型	正则化的极大似然估计	对数似然损失	特征选择，生成，剪枝
逻辑斯谛回归与最大熵模型	多类分类	特征条件下类别的条件概率分布，对数线形模型	判别模型	极大似然估计，正则化的极大似然估计	逻辑斯谛损失	改进的迭代尺度算法，梯度下降，拟牛顿法
支持向量机	二类分类	分离超平面，核技巧	判别模型	极小化正则化合页损失，软间隔最大化	合页损失	序列最小最优化算法（SMO）
提升方法	二类分类	弱分类器的线性组合	判别模型	极小化加法模型的指数损失	指数损失	前向分步加法算法
EM 算法[①]	概率模型参数估计	含隐变量概率模型		极大似然估计，极大后验概率估计	对数似然损失	迭代算法
隐马尔可夫模型	标注	观测序列与状态序列的联合概率分布模型	生成模型	极大似然估计，极大后验概率估计	对数似然损失	概率计算公式，EM 算法
条件随机场	标注	状态序列条件下观测序列的条件概率分布，对数线性模型	判别模型	极大似然估计，正则化极大似然估计	对数似然损失	改进的迭代尺度算法，梯度下降，拟牛顿法

---------- **以下内容并非出自《统计学习方法》** ----------

回到顶部

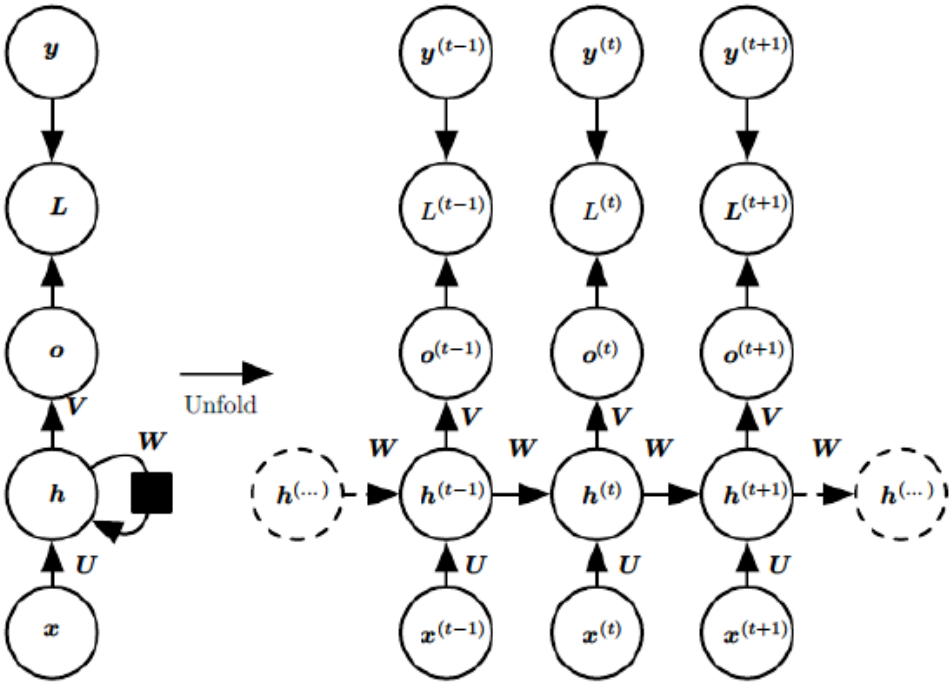
神经网络

- **神经元 (感知器)** 接收到来自 n 个其他神经元传递过来的**输入信号**, 这些输入信号通过带权重的连接进行传递, 神经元将接收到的总输入值与神经元的**阈值**进行比较, 然后通过**激活函数**处理以产生神经元的**输出**. 把许多个这样的神经元按一定的层次结构连接起来就得到了神经网络. 一般使用**反向传播 (BP) 算法**来进行训练.
- **反向传播 (BP) 算法**:

1. **前向传播**: 将训练集数据输入, 经过隐藏层, 到达输出层并输出结果.
2. 计算估计值与实际值之间的误差, 并将该误差从输出层向输入层**反向传播**.
3. 在反向传播的过程中, 根据误差使用**梯度下降**与**链式法则**调整各种参数的值.
4. 不断**迭代**直至收敛.

- **深度神经网络 (DNN)**: 可以理解为有很多隐藏层的神经网络. DNN 内部分为**输入层** (第一层), **隐藏层**, **输出层** (最后一层). 层与层之间是**全连接**的.
- **卷积神经网络 (CNN)**: 一般用于图像识别. 通过**卷积核**和**感受野**的乘积形成卷积后的输出. 在每一个卷积层之后, 通常会使用一个 **ReLU(修正线性单元) 函数**来把所有的负激活都变为零. 在几个卷积层之后也许会用一个**池化层 (采样层)** 来输出过滤器卷积计算的每个子区域中的最大数字或平均值.

- **循环神经网络 (RNN)**: 如果训练样本输入是**连续序列**, 则 DNN 和 CNN 不好解决. RNN 假设样本是基于序列的, 对应的输入是样本序列中的 x(t), 而模型在序列索引号 t 位置的隐藏状态 h(t) 由 x(t) 和 h(t-1) **共同决定**. 在任意序列索引号 t 有对应的模型预测输出 o(t). 也就是说, RNN 是包含循环的网络, 允许信息的**持久化**.



- **长短期记忆网络 (LSTM)**: 一种特殊的 RNN, 可以学习**长期依赖信息**.

回到顶部

K-Means

- K-Means 是**无监督**的**聚类**算法. 思想是对于给定的样本集, 按照样本之间的距离大小将样本集划分为 K 个簇, 让簇内的点尽量紧密地连在一起, 而让簇间的距离尽量的大.
- **传统算法**:

1. 用先验知识或交叉验证选择一个合适的 **k** 值.
2. 随机选择 k 个样本作为初始的**质心**. 注意初始化质心的选择对最后的聚类结果和运行时间都有很大的影响.
3. 计算每个样本点和各个质心的距离, 将样本点标记为**距离最小**的质心所对应的簇.
4. 重新计算每个**簇**的质心, 取该簇中每个点位置的平均值.
5. 重复 2,3,4 步直到 k 个质心都没有发生变化为止.

- **K-Means++**: 用于优化随机初始化质心的方法

1. 从输入样本点中随机选择一个点作为第一个质心.
2. 计算每一个样本点到已选择的质心中**最近质心**的距离 D(x).
3. 选择一个新的样本点作为新的质心, 选择原则是 D(x) 越大的点被选中的概率越大.
4. 重复 2 和 3 直到选出 k 个质心.

- **Elkan K-Means**: 利用两边之和大于第三边以及两边之差小于第三边来减少距离的计算. 不适用于特征稀疏的情况.
- **Mini Batch K-Means**: 样本量很大时, 只用其中的一部分来做传统的 K-Means. 一般多用几次该算法, 从不同的随即采样中选择最优的聚类簇.

回到顶部

Bagging

- Bagging 的弱学习器之间没有 boosting 那样的联系, 它的特点在于 " **随机采样** ", 也就是有放回采样. 因此**泛化能力**很强. 一般会随机采集和训练集样本数一样个数的样本. 假设有 m 个样本, 且采集 m 次, 当 m 趋向无穷大时不被采集到的数据占 1/e, 也就是 36.8%, 称为**袋外数据**, 可以用来检测模型的泛化能力. Bagging 对于弱学习器没有限制, 一般采用决策树和神经网络.
- **算法**:

1. 对于 t=1~T, 对训练数据进行第 t 次随机采样, 共采集 m 次, 得到包含 m 个样本的采样集 Dm.
2. 用采样集 Dm 训练第 m 个弱学习器 Gm(x)
3. 如果是分类, 则用**简单投票法**. 如果是回归, 则取 T 个弱学习器结果的**平均值**.

- **随机森林**: 使用 **CART 决策树**作为弱学习器, 然后每次不从 n 个样本特征中选择最优特征, 而是从随机选择的 **nsub** 个样本特征中来选择. 一般用交叉验证来获取合适的 nsub 值.

回到顶部

Apriori

- Apriori 是常用的挖掘出**数据关联规则**的算法, 用于找出数据值中**频繁**出现的数据集合. 一般使用支持度或者支持度与置信度的组合作为**评估标准**.
- **支持度**: 几个关联的数据在数据集中出现的次数占总数据集的比重

$$Support(X,Y) = P(XY) = \frac{number(XY)}{num(AllSamples)}$$

- **置信度**: 一个数据出现后. 另一个数据出现的概率

$$Confidence(X \Leftarrow Y) = P(X|Y) = P(XY)/P(Y)$$

- Apriori 算法的目标是找到最大的 **K 项频繁集**. 假设使用支持度来作为评估标准, 首先搜索出**候选 1 项集**及对应的支持度, **剪枝**去掉低于支持度的 1 项集, 得到**频繁 1 项集**. 然后对剩下的频繁 1 项集进行**连接**, 得到候选的频繁 2 项集..... 以此类推, 不断迭代, 直到无法找到频繁 k+1 项集为止, 对应的频繁 k 项集的集合即为输出结果.

回到顶部

降维方法

- 主成分分析 (PCA): 降维, 不断选择与已有坐标轴正交且方差最大的坐标轴.
- 奇异值分解 (SVD): 矩阵分解, 降维, 推荐系统.
- 线性判别分析 (LDA)

回到顶部

引用

1. 如何理解神经网络里面的反向传播算法？ - 陈唯源的回答 - 知乎
2. 刘建平 Pinard - 博客园