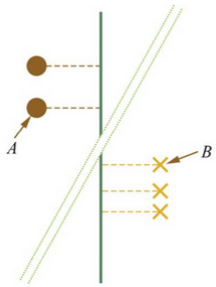
经典算法

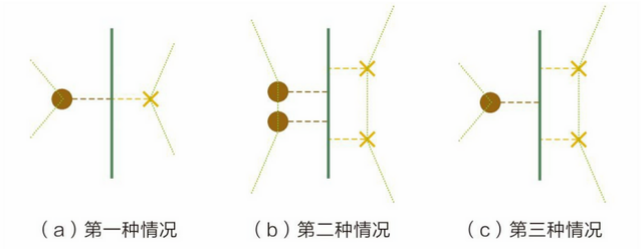
支持向量机

1. **关键知识点：**
   - 分类超平面（hyperplane）
   - 核映射（Kernel Mapping）
   - SMO算法（Sequential Minimal Optimization）
2. **什么是SVM模型：**
   - 分类模型，目标是找到一个距离最近点最远（边缘最大化）的超平面
   - 距离分类超平面最近的点叫做支持向量，模型只需要考虑这些点便能找到最优的超平面（SVM模型运行效率极高的原因之一）
   - 通过核方法将非线性可分的样本点映射到更高维度的空间，再利用超平面进行线性分类

3. **空间上线性可分的两类点，被投影到SVM的分类超平面上，还是线性可分吗?**
    - 非线性可分
    - 最简单的反例：只有两个样本点，超平面就是两点的中垂线，映射之后两点重合，不可分
    - 事实上，对于任意两组线性可分的点，它们在SVM的分类超平面上的投影都是不可分的
    - 因为SVM的分类结果只依赖于支持向量，所以我们可以考虑一个只有支持向量的情况：支持向量在绿色实线上的投影线性可分，但是我们可以证明绿色实线此时已经不是最优分类



    - 或者，根据超平面分离定理（Seperating Hyperplane Theorem）：SVM的分类超平面实际上就是两组样本中距离最近的点的中垂线，两组点在中垂线上的映射都是无法线性可分的



4. **是否存在一组参数使SVM的训练误差为0?**

- 一个使用高斯核训练的SVM中，若训练集中不存在两个点在同一位置，则存在一组系数$a_1, a_2, ..., a_m, b$ 和高斯参数γ使得该SVM的训练误差为0。

  1. SVM的预测公式可以写为：

  $$f(x)=\sum_{i=1}^{m}\alpha_i y^{(i)}K(x^{(i)},x)+b,$$

  2. 让$\alpha_i = 1$以及$b = 0$，则有：

  $$\begin{aligned} f(x)&=\sum_{i=1}^{m}\alpha_i y^{(i)}K(x^{(i)},x)+b \\ &=\sum_{i=1}^{m}y^{(i)}K(x^{(i)},x) \\ &=\sum_{i=1}^{m}y^{(i)}\mathrm{e}^{-\|x-x^{(i)}\|^2/\gamma^2}. \end{aligned}$$

  3. 由于不存在两个相同的点，那么对于任何$i! = j$，有$x_i - x_j >= e$，将x_j带入上面的式子：

  $$f(x^{(j)})=\sum_{i=1}^{m}y^{(i)}\mathrm{e}^{-\|x^{(j)}-x^{(i)}\|^2/\gamma^2},$$

  $$f(x^{(j)})-y^{(j)}=\sum_{i=1,i\neq j}^{m}y^{(i)}\mathrm{e}^{-\|x^{(j)}-x^{(i)}\|^2/\gamma^2},$$

  $$\|f(x^{(j)})-y^{(j)}\|\leqslant\sum_{i=1,i\neq j}^{m}\mathrm{e}^{-\|x^{(j)}-x^{(i)}\|^2/\gamma^2}.$$

  由题意知$\|x^{(i)}-x^{(i)}\|\geq\varepsilon$，取$\gamma=\varepsilon/\sqrt{\log m}$，可将式（3.12）重写为

  $$\begin{aligned} \|f(x^{(j)})-y^{(j)}\|&\leqslant\|\sum_{i=1,i\neq j}^{m}\mathrm{e}^{-\|x^{(j)}-x^{(i)}\|^2/\gamma^2}\| \\ &\leqslant\|\sum_{i=1,i\neq j}^{m}\mathrm{e}^{-\log m}\|=\frac{m-1}{m}<1. \end{aligned}$$

  4. 所以对于任何训练样本x_j，预测结果与真是标签的距离小于1，因为SVM处理的是二分类问题，误差小于1时不影响分类结果，所以训练误差为0。

5. **训练误差为0的SVM分类器一定存在吗?**
    - 上个问题找到了一组参数使得SVM的训练误差为0，但是这组参数不一定是SVM的一个解
    - 之前找到的参数能满足$y_j f(x_j) > 0$，现在需要参数需要满足一个更强的条件$y_j f(x_j) >= 1$，考虑在实际训练时我们没有加入松弛变量：

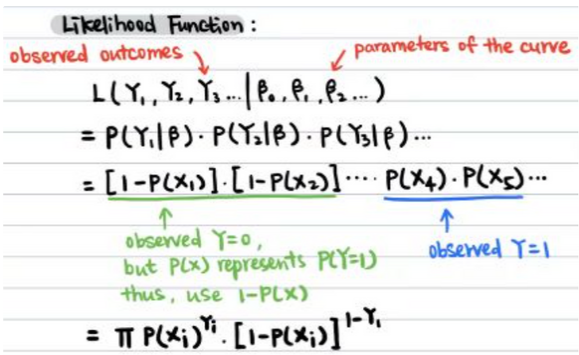
仍然固定b=0，于是预测公式$f(x) = \sum_{i=1}^{m} \alpha_i y^{(i)} K(x^{(i)}, x)$，将$y^{(i)} f(x^{(j)})$展开，有

$$
\begin{aligned}
y^{(j)} f(x^{(j)}) &= y^{(j)} \sum_{i=1}^{m} \alpha_i y^{(i)} K(x^{(i)}, x^{(j)}) \\
&= \alpha_j y^{(j)} y^{(j)} K(x^{(j)}, x^{(j)}) + \sum_{i=1, i \neq j}^{m} \alpha_i y^{(i)} y^{(j)} K(x^{(i)}, x^{(j)}) \\
&= \alpha_j + \sum_{i=1, i \neq j}^{m} \alpha_i y^{(i)} y^{(j)} K(x^{(i)}, x^{(j)}).
\end{aligned}
$$

    - 我们可以选择一个很大的α_j和一个很小的γ（使核映射项非常小），那么α_j会占据绝对主导权，满足$y_j f(x_j) >= 1$的条件
6. **加入松弛变量的SVM的训练误差可以为0吗?**
    - 背景：在实际应用中，使用SMO算法来训练一个加入了松弛变量的线性SVM模型，且惩罚因子C为任意未知常数
    - 使用SMO算法训练的线性分类器并不一定能得到训练误差为0模型，因为优化目标改变了，不再是训练误差最小化

逻辑回归

1. **逻辑回归相比于线性回归，有何异同?**
    - 区别：
        - 逻辑回归处理分类问题，线性回归处理回归问题
        - 逻辑回归中，因变量取值是一个二元分布（离散变量），模型学习后得出一个期望值（probability），并基于此进行预测分类
        - 线性回归中，预测的是因变量的真实数值，是连续变量
        - 逻辑回归可以说是在线性回归的基础上加了一个sigmoid函数，将线性回归输出的值归一化到[0,1]的区间内，即sigmoid函数起到一个映射的作用
        - 逻辑回归预测的期望值p满足$log(\frac{p}{1-p}) = \theta^T x$，即可以通过线性回归来求出log(odds)，odds是y=1事件发生的机率，所以逻辑回归的名字中也有回归一词
        - 我们可以把逻辑回归看作是广义线性模型（generalized linear regression），即在因变量y服从二元分布时的一个特殊情况
        - 在线性回归中，使用小二乘法（least square）求解时，我们认为因变量y服从正态分布
    - 相似：
        - 可以认为两者都使用了极大似然估计（maximum likelihood estimation）来训练模型，小二乘法实际上是MLE在因变量y服从正态分布假设下的一个简化，而逻辑回归则是学习这样的一个似然函数：



        - 二者在求解参数的过程中，一般都会使用梯度下降的方法，这也是所有监督学习模型的一个常见的相似之处

2. **如何使用逻辑回归处理多标签的分类问题?**
   - 如果一个样本只对应一个标签:
     - 我们可以采用多项逻辑回归（Multinomial LR），从k个标签中选取一个参考标签k，然后针对剩下的k-1个标签训练k-1个独立的二元逻辑回归模型（$odds = \frac{p_i}{p_k}$）：

$$\ln \frac{\Pr(Y_i = 1)}{\Pr(Y_i = K)} = \boldsymbol{\beta}_1 \cdot \mathbf{X}_i$$
$$\ln \frac{\Pr(Y_i = 2)}{\Pr(Y_i = K)} = \boldsymbol{\beta}_2 \cdot \mathbf{X}_i$$
$$\cdots\cdots$$
$$\ln \frac{\Pr(Y_i = K-1)}{\Pr(Y_i = K)} = \boldsymbol{\beta}_{K-1} \cdot \mathbf{X}_i$$

     - 上述公式左右两边进行指数化处理：

$$\Pr(Y_i = 1) = \Pr(Y_i = K) e^{\boldsymbol{\beta}_1 \cdot \mathbf{X}_i}$$
$$\Pr(Y_i = 2) = \Pr(Y_i = K) e^{\boldsymbol{\beta}_2 \cdot \mathbf{X}_i}$$
$$\cdots\cdots$$
$$\Pr(Y_i = K-1) = \Pr(Y_i = K) e^{\boldsymbol{\beta}_{K-1} \cdot \mathbf{X}_i}$$

     - 我们需要所有的概率加起来等于1，由此可以得到：

$$\Pr(Y_i = K) = 1 - \sum_{k=1}^{K-1} \Pr(Y_i = K) e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i} \Rightarrow \Pr(Y_i = K) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$

     - 将得到的结果带入回公式中，相当于对预测结果通过一个softmax函数进行了归一化（同时softmax函数能放大结果中的差距）：

$$h_\theta(x) = \begin{bmatrix} p(y=1 \mid x;\theta) \\ p(y=2 \mid x;\theta) \\ \vdots \\ p(y=k \mid x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} \mathrm{e}^{\theta_j^{\mathrm{T}} x}} \begin{bmatrix} \mathrm{e}^{\theta_1^{\mathrm{T}} x} \\ \mathrm{e}^{\theta_2^{\mathrm{T}} x} \\ \vdots \\ \mathrm{e}^{\theta_k^{\mathrm{T}} x} \end{bmatrix},$$

     - 此时我们假设了每个样本属于不同标签的概率服从几何分布（geometric distribution），即$p(X = k) = (1-p)^{k-1}p$
     - 多项逻辑回归有参数冗余（redundancy）的特点，即参数同时加减一个向量不会改变预测结果，由此可反推出多项逻辑回归是二元逻辑回归的一个拓展
     - 选取不同的参考类别不会影响预测结果，但是会影响回归系数的意义解读，也会影响某个自变量的参数是否具有统计显著性
   - 如果一个样本可能属于多个标签:
     - 将问题拆分成多个二元分类问题，针对每个类别训练一个独立的二元逻辑回归模型，并独立地做出预测，这样就能判断一个样本是否属于多个标签

决策树

1. **什么是决策树?**
    - 一棵决策树由根节点、内部结点、叶结点以及有向边组成
    - 决策树的生产包含三个过程：特征选择、树的构造、树的剪枝

2. **有哪些常用的决策树算法，它们分别使用什么启发函数?**

- ID3（最大信息增益）
  - 在信息论中，熵（entropy）是表示随机变量不确定性的度量：

    在信息论与概率统计中，熵（entropy）是表示随机变量不确定性的度量．设X是一个取有限个值的离散随机变量，其概率分布为

    $$P(X=x_i)=p_i,\quad i=1,2,\cdots,n$$

    则随机变量X的熵定义为

    $$H(X)=-\sum_{i=1}^{n}p_i\log p_i \qquad (5.1)$$

  - 对于一个类别数量为K的样本集D，经验熵（empirical entropy）为：

    $$H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log_2\frac{|C_k|}{|D|},$$

    - C_k是样本集D中属于第k类的样本子集
  - 某个特征A对于数据集D的经验条件熵（empirical conditional entropy）为：

    $$H(D\mid A)=\sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)=\sum_{i=1}^{n}\frac{|D_i|}{|D|}\left(-\sum_{k=1}^{k}\frac{|D_{ik}|}{|D_i|}\log_2\frac{|D_{ik}|}{|D_i|}\right),$$

    - D_i表示D中特征A取第i个值的样本子集
  - 特征A带来的信息增益便是二者的差，即熵的减少量：

    $$g(D,A)=H(D)-H(D\mid A).$$

  - 在内部节点中，选择信息增益最大的特征进行分支，加快树的生长
- C4.5（最大信息增益比）
  - 特征A对于数据集D的信息增益比为：

    $$g_R(D,A)=\frac{g(D,A)}{H_A(D)},$$

  - 其中$H_A(D)$是数据集D关于A的取值熵（把A当作标签时数据集D的熵）：

    $$H_A(D)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\log_2\frac{|D_i|}{|D|},$$

- CART（最大Gini指数）
  - Gini指数描述的是数据的不纯度（impurity）：

    $$\mathrm{Gini}(D)=1-\sum_{k=1}^{n}\left(\frac{|C_k|}{|D|}\right)^2.$$

  - 特征A的Gini指数是：

$$\mathrm{Gini}(D \mid A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} \mathrm{Gini}(D_i).$$

    - 每次迭代中选择Gini指数最小的特征，根据对应的切分点进行长枝分类
    - 与ID3和C4.5不同的是，CART是一颗二叉树，采用二元切割法
- 总结：
    - ID3采样信息增益作为评价标准，会倾向与取值较多的特征，因为特征取值越多，意味着更加细分，给定条件后不确定性会减少更多，这是一个缺点（容易导致过拟合）
    - C4.5通过引入信息增益比，弥补了这一缺点，一定程度上对取值较多的特征进行惩罚（取值越多取值熵越大）
    - ID3只能处理离散型变量，C4.5和CART可以处理连续型变量
        - C4.5通过对数据排序之后找到一个切分点，把连续型变量转换成布尔型
        - CART则是每次构建时都会对特征进行二值化分
    - ID3和C4.5只能应用于分类任务，CART还能应用于回归粪污（使用最小平方误差准则）
    - ID3对样本特征缺失值比较敏感，C4.5和CART可以对缺失值进行不同方式的处理
    - ID3和C4.5可以在每个结点上产生多叉分支，且同一个特征不会再不同层级反复使用
    - ID3和C4.5通过剪枝来权衡树的准确性和泛化能力，CART直接利用全部数据发现所有可能的树结构并进行对比

3. **如何对决策树进行剪枝（pruning）？**
    - 一颗完全生长的决策树很容易会过拟合，所以我们需要进行剪枝，提升模型泛化能力
    - 预剪枝（pre-pruning）：
        - 在生产决策树的过程中提前停止树的生长，在树中结点扩展之前，先计算当前的划分是否能提升模型的泛化能力，如果不能则停止生长
        - 当树深达到一定程度后，停止生长
        - 当当前节点的样本数量小于某个阈值，停止生长
        - 当最大的信息增益（information gain）小于某个阈值，停止生长
        - 该方法非常简单高效，适合解决大规模问题，但是如何准确的找到停止树生长的最佳阈值，需要一定的经验才能进行判断，且预剪枝会带来欠拟合的风险（有可能当前划分没达到一定的个信息增益，但是后续的划分可能会带来很大的增益）
    - 后剪枝（post-pruning）：
        - 先生成一颗完整的决策树，然后自底向上剪枝
        - 剪枝过程将子树删除，并用一个叶节点代替
        - 后剪枝通常能比预剪枝方法的得到泛化能力更强的决策树，但是时间消耗会更大
        - 常见后剪枝方法包括：
            - 错误率降低剪枝（reduced error pruning）
            - 悲观剪枝（pessimistic error pruning）
            - 代价复杂剪枝（cost complexity pruning）
            - 最小误差剪枝（minimum error pruning）
            - CVP（critical value pruning）
        - 代价复杂剪枝（CCP）：
            - 从完整的决策树T_0开始，生产一个子树序列$T_0, T_1, T_2 ... T_n$，其中T_{i+1}由T_i生成，根节点为T_n，根据误差选择表现最佳的子树
            - 从T_0开始，选择T_i中关于训练集误差增加率最小的结点进行剪枝，得到T_{i+1}
            - 当一棵树T在结点t处剪枝时，误差增加率为：

$$\alpha = \frac{R(t) - R(T_t)}{|L(T_t)| - 1}.$$

    - $|L(T_t)|$为子树T_t的叶节点个数，$R(t)$和$R(T_t)$分别是剪枝后结点t的误差和剪枝前子树T_t的误差
- 采用k-fold交叉验证，前k-1份数据用于生产完整的决策树，最后一份数据用于计算误差增加率，平均k次验证的误差增加率，判断最优剪枝结点，从而进行剪枝
- CCP剪枝法精度与REP差不多，但不需要额外的剪枝数据集，且形成的复杂度更小，但是生成子树序列的时间复杂度与结点数成二次关系，导致算法时间开销很大