机器学习面试

如何做梯度并行运算

试图跳出局部最小的方法

1. 以多组不同参数值初始化多个神经网络，按标准方法（梯度下降）训练后，取其中误差最小的解作为最终参数
2. 模拟退火：在每一步都以一定的概率接受比当前解更差的结果
3. 随机梯度下降
4. 遗传算法

关于决策树

最优属性的划分基本思想：希望分支结点所包含的样本尽可能属于同一类别，即结点的“纯度”要越来越高

- 信息增益：

信息熵：$Ent(D) = -\sum_{k=1}^{|Y|} p_k log_2 p_k$，其中$p_k$为当前样本集合D中第k类样本所占比例。

熵越小，D的纯度越高

用属性a对样本集D进行划分的信息增益：$Gain(D,a) = Ent(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} Ent(D^v)$

其中，属性a有V个不同取值$\{a^1, a^2, ..., a^v\}$，

若使用a划分则会产生V个不同分支点，其中D^v第v个分支点包含D中所有在属性a上取值为a^v的样本的

- 增益率

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

- 基尼系数

基尼值

$$\begin{aligned}Gini(D)&=\sum_{k=1}^{|Y|}\sum_{k^{'}\neq k}p_kp_k^{'}\\&=1-\sum_{k=1}^{|Y|}p^2\end{aligned}$$

则属性a的基尼指数为$Gini_index(D,a)=\sum\limits_{v=1}^{V}\frac{|D^v|}{|D|}Gini(D^v)$，选择那个使得划分后的基尼指数最小的属性作为最优划分属性

剪枝：

- 预剪枝，在决策树生成中就剪
- 后剪枝，生成一颗完整的决策树后，自底向上对非叶节点进行考虑

一般来说，后剪枝欠拟合风险较小，泛化性能往往优于预剪枝

关于SVM

1. 目标函数是啥

$$\begin{aligned}&\min_{\omega,b}\frac{1}{2}\left\|\omega\right\|\\&s.t.\quad y_i(\omega^Tx_i+b)\geqslant 1,i=1,2,...,m.\end{aligned}$$

2. 什么是KKT条件

3. 关于核函数

若不存在一个超平面将训练样本正确分类（线性可分），可以将样本从原始空间映射到一个更高维的特征空间使之线性可分（一定存在），如$x \mapsto \phi(x)$y

有

$$\begin{aligned}&\min_{\omega,b}\frac{1}{2}\|\omega\|\\&s.t.\quad y_i(\omega^T\phi(x_i)+b)\geqslant 1,i=1,2,...,m.\end{aligned}$$

但化为对偶问题，需要求内积$\phi(x_i)^T\phi(x_j)$，由于特征空间维数可能很高，甚至无穷维，因此计算困难，这时希望能找到一个函数：$\kappa(x_i,x_j)=\phi(x_i)^T\phi(x_j)$，即向量在特征空间的内积等于它们在原始样本空间通过函数κ计算的结果，这个函数就称为核函数。

只要一个对称函数所对应的核矩阵是半正定的，它就能作为核函数。

常用核函数：

表 6.1 常用核函数

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

以下这些组合也是核函数：

- 核函数的线性组合 $\gamma_1\kappa_1+\gamma_2\kappa_2$
- 核函数的直积$\kappa_1\otimes\kappa_2(x,z)=\kappa_1(x,z)\kappa_2(x,z)$
- 对任意函数$g(x)$，有$\kappa(x,z)=g(x)\kappa_1(x,z)g(z)$

特征空间的好坏对SVM的性能至关重要，而核函数也隐式定义了这个空间。经验：对文本数据通常采用线性核，情况不明时可先尝试高斯核

4. 软间隔

核函数往往难以确定，可以允许某些样本不满足约束$y_i(\omega^T x_i+b)\geq 1$

因为在最大化间隔时不满足约束的样本应尽可能少，于是目标函数可写为：

$$\min_{\omega,b}\frac{1}{2}||\omega||^2+C\sum_{i=1}^{m}l_{0/1}(y_i(\omega^T x_i+b)-1)$$

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

l是0/1损失函数

5. 支持向量回归（SVR）

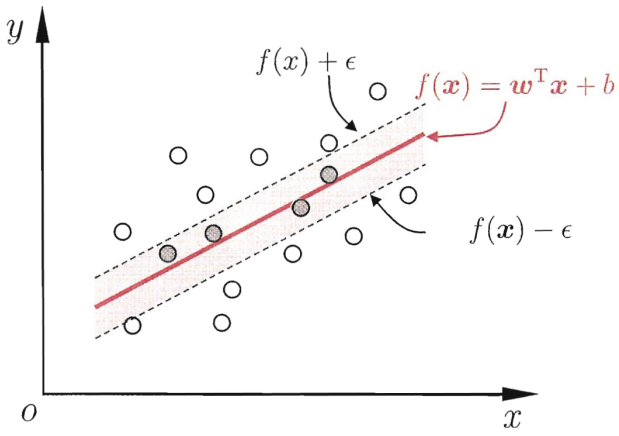


图 6.6 支持向量回归示意图. 红色显示出 ϵ-间隔带, 落入其中的样本不计算损失.

于是SVR可形式化为：

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

f(x)是要学的模型

l_ϵ是不敏感损失函数：$l_{0/1}(z) = \begin{cases} 1, & \text{if } |z| \leq \epsilon \\ |z| - \epsilon, & otherwise \end{cases}$

贝叶斯分类器

- 贝叶斯决策论

N种可能的标记$\{c_1, c_2, ..., c_N\}$

λ_{ij}为将一个真实标记为c_j的样本x误分类为c_i所产生的损失，则

1. 样本x的条件风险（期望损失即为风险）为：$R(c_i|x) = \sum_{j=1}^{N} \lambda_{ij} P(c_j|x)$
2. 我们的任务是寻找一个判定准则h以最小化总体风险：$R(h) = \mathbb{E}_x[R(h(x)|x)]$
3. 贝叶斯判定准则：为了最小化总体风险，只需在每个样本上选择那个能使条件风险$R(c|x)$最小的类别标记，即$h^*(x) = \arg\min R(c|x)$，此时h^*称为贝叶斯最优分类器，与之对应的风险为贝叶斯风险
4. 若目标是最小化分类错误率（$\arg\min R(c|x)$），则误判损失可写为：

$$\lambda_{ij} = \begin{cases} 0, & \text{if } i == j \\ 1, & otherwise \end{cases}$$

5. 此时条件风险为$R(c_i|x) = 1 - P(c_i|x)$，所以贝叶斯最优分类器为：

$$h^*(x) = \arg\min(1 - P(c|x))，即h^*(x) = \arg\max P(c|x)$$

剩下就是如何计算后验概率$P(c|x)$的问题了

判别式模型和生成式模型

- 判别式模型：给定x，通过直接建模$P(c|x)$来预测c，如决策树，神经网络，支持向量机等
- 生成式模型：先对联合概率分布建模$P(x, c)$，再通过贝叶斯定理获得$P(c|x)$：

$P(c|x)=\frac{P(c)P(x|c)}{P(x)}$，其中$P(c)$为先验概率，$P(x|c)$可称为似然函数，$P(x)$是归一化证据因子（evidence）（对于连续型属性，可将概率质量函数P换成概率密度函数p，大P变小p）

极大似然估计（MLE）

- 频率主义学派：认为参数虽然未知，但却是客观存在的固定值，因此可通过优化似然函数来确定参数值

  做法：假设样本独立同分布，对于第c类的样本集D_c有$P(D_c|\theta)=\prod_{x\in D_c}P(x|\theta)$，为防止数据下溢，通常使用对数似然$LL(\theta)=logP(D_c|\theta)=\sum_{x\ inD_c}logP(x|\theta)$，然后：

$$\hat{\theta}=\arg\max_{\theta}LL(\theta)$$

- 贝叶斯学派：认为参数是未观察的随机变量，其本身也可有分布EM

朴素贝叶斯分类器

假设所有属性相互独立，则，d为属性的数量

所以此时贝叶斯判定准则为

对于离散属性，可使用高斯分布$p(x_i|c)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

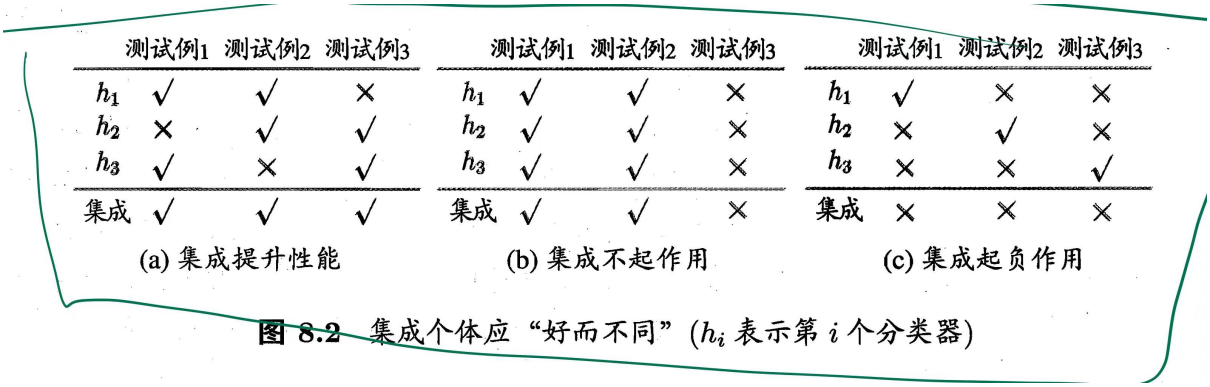
EM算法

常用于估计参数隐变量，是一种迭代式方法

基本思想：若模型参数θ已知，则可根据训练数据推断出最优隐变量Z的值（E步）；反之，若Z的值已知，则可方便地对参数θ做极大似然估计（M步）。具体的以θ^0为起点，迭代以下步骤直至收敛：

- 基于θ^t推断隐变量Z的期望，记为Z^tji
- 基于已观测变量X和Z^t，对参数θ做极大似然估计，记为θ^{t+1}

关于集成学习



	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	×	√	√
h_3	√	×	√
集成	√	√	√

(a) 集成提升性能

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	√	√	×
h_3	√	√	×
集成	√	√	×

(b) 集成不起作用

	测试例1	测试例2	测试例3
h_1	√	×	×
h_2	×	√	×
h_3	×	×	√
集成	×	×	×

(c) 集成起负作用

图 8.2 集成个体应“好而不同”(h_i 表示第 i 个分类器)

可见，要获得好的集成，个体学习器应“好而不同”：个体学习器要有一定的“准确性”，并且要有“多样性”。

然而，现实任务中，准确性和多样性本身就存在冲突：个体学习器是为解决同一个问题训练出来的，难以多样性。一般的，准确性很高之后，要增加多样性就需要牺牲准确性。

AdaBoost

- 每一轮如何改变训练数据的权值或概率分布？

提高那些被前一轮弱分类器错误分类样本的权值，降低那些被正确分类样本的权值

- 如何将弱分类器组合成一个强分类器？

加权多数表决，加大分类误差率小的弱分类器的权值，使其在表决中起较大的作用，减小分类误差率大的弱分类器的权值，使其在表决中起较小的作用。

算法过程：

给定二分类数据集$T = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N), y \in \{-1, +1\}$

1.初始化训练数据权重分布$D_1 = (w_{11}, w_{1i}, ..., w_{1N}), w_{1i} = \frac{1}{N}$

2.对m=1,2,...,M，反复学习基本分类器：

2.1 使用具有权值分布D_m的训练数据集，得到基本分类器$G_m(x) \to \{-1, +1\}$

2.2 计算$G_m(x)$在训练数据集上的误差率：

$e_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} I(G_m(x_i) \neq y_i)$，经计算$0 < e_m \leq \frac{1}{2}$，若误差率大于0.5，则退出迭代

2.3 计算$G_m(x)$的系数$\alpha_m = \frac{1}{2} log \frac{1-e_m}{e_m}$ ，用于计算分类器$G_m(x)$在最终分类器中的重要程度

2.4 更新下一轮训练数据集权重分布

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2}, ..., w_{m+1,N})$$
$$w_{m+1,i} = \frac{w_m i}{Z_m} e^{-\alpha_m y_i G_m(x_i)}, i = 1, 2, ..., N$$

其中Z_m为规范化因子：$Z_m = \sum_{i=1}^{N} w_{mi} e^{-\alpha_m y_i G_m(x_i)}$

权重更新解释：

$$w_{m+1,i} = \begin{cases} \frac{w_{mi}}{Z_m} e^{-\alpha_m} & \text{if } G_m(x_i) = y_i \\ \frac{w_{mi}}{Z_m} e^{\alpha_m} & \text{if } G_m(x_i) \neq y_i \end{cases}$$

当$G_m(x_i) == y_i$表示基分类器预测准确，此时两者同号，则$0 < e^{-\alpha_m} \leq 1$，从而降低了正确分类样本的权重；当$G_m(x_i) \neq y_i$表示基分类器预测错误，两者异号，则$e^{\alpha_m} > 1$，提高错误分类样本的权重

3. 构建基分类器线性组合$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$，得最终分类器：$G(x) = sign(f(x))$

可以证明AdaBoost的训练误差呈指数级下降

AdaBoost只适用于二分类

Bagging

基本思想：先随机取出一个样本放入采样集中，再把该样本放回初始数据集，使得下次采样时该样本仍有可能被选中（互相有交叠）。然后采样出T个含有m个训练样本的采样集后，基于每个采样集训练处一个基学习器，再将这些基学习器进行结合。

输入： 训练集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

过程：

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

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

图 8.5 Bagging 算法

最后对预测输出结合时，Bagging通常对分类任务使用简单的投票法，对回归任务使用简单平均法。

Bagging可以不经修改适用于多分类、回归等任务

随机森林

是Bagging一个变体，以基学习器构建Bagging集成的基础上，进一步在决策树的训练过程中引入随机属性的选择，对基决策树的每个结点，先从该节点的属性集合中随机选择一个包含k个属性的子集，然后再从在个子集中选择一个最优属性用于划分（传统决策树在选择划分属性时是在当前结点的属性集合中选择一个最优属性）

对学习器进行结合的策略

平均法

投票法

学习法：通过另一个学习器来进行结合。个体学习器称为初级学习器，用于结合的学习器称为次级学习器或元学习器