机器学习、数据挖掘、数据分析岗面试总结

2018年07月25日 23:53:02 阅读数：158

算法面试注意细节

算法工程师岗位（机器学习、数据挖掘、数据分析等）面试主要包含两大块：首先是基本的 Coding 能力；其次就是机器学习算法的理论与应用能力。

基本的 Coding 能力考察，主要是让你写一些数据结构与算法设计的常见算法题，比如链表类、二叉树、排序、查找、动态规划等。这方面建议《剑指 Offer》+ LeetCode 结合的方式进行练习。

机器学习算法理论与应用能力不用说，应付面试看本课程就行了。当然师父领进门，修行在个人，想要在这方面深造，还需要大量实践和广泛学习优秀的论文和开源源代码。

算法面试常见问题

1.模型融合方法有哪些

主要有以下几个：

```
Voting
Averaging
Bagging
Boosting
Stacking
```

2.特征选择方法

主要有：

```
去掉取值变化小的特征
递归特征消除法
基于学习模型的特征排序
线性模型和正则化
```

3.学习项目的一般步骤

主要包括：

```
问题抽象，模型选择；
数据获取；
特征工程（数据清洗，预处理、采样等）；
模型训练，调优；
模型验证、误差分析；
模型融合；
模型上线。
```

4.推导 LR 算法

5.多参数调参方法

网格化搜索：对于多个参数，首先根据经验确定大致的参数范围。然后选择较大的步长进行控制变量的方法进行搜索，找到最优解后；然后逐步缩小步长，使用同样的方法在更小的区间内寻找更精确的最优解。

6.XGBoost 算法的主要参数及其作用

主要有：

- **objective**

学习任务，如选择 `multi:softmax` 则为多分类；

- **gamama**

用于控制是否后剪枝的参数，越大越保守，一般0.1、0.2这样子；

- **max_depth**

构建树的深度，越大越容易过拟合；

- **lambda**

控制模型复杂度的权重值的 L2 正则化项参数，参数越大，模型越不容易过拟合；

- **num_round**

XGBoost 中梯度提升树的个数，它的值越大，模型的学习能力越强；

- **eta**

单颗提升树的学习率，通过缩减特征的权重使提升计算过程更加保守。

7.Bagging 和 Boosting 的区别

主要表现在：

- **样本选择上**

Bagging： 训练集是在原始集中有放回选取的，从原始集中选出的各轮训练集之间是独立的。

Boosting： 每一轮的训练集不变，只是训练集中每个样例在分类器中的权重发生变化。而权值是根据上一轮的分类结果进行调整。

- **样例权重**

Bagging： 使用均匀取样，每个样例的权重相等；

Boosting： 根据错误率不断调整样例的权值，错误率越大则权重越大。

- **预测函数**

Bagging： 所有预测函数的权重相等；

Boosting： 每个弱分类器都有相应的权重，对于分类误差小的分类器会有更大的权重。

- **并行计算**

Bagging： 各个预测函数可以并行生成；

Boosting： 各个预测函数只能顺序生成，因为后一个模型参数需要前一轮模型的结果。

8.PCA 与 LDA 降维原理

9.为什么 L1 正则可以实现参数稀疏，而 L2 正则不可以？

首先，加入正则化的目标函数表达式如下：

$$Obj(w) = \min_{w} E_D(w) + \lambda E_R(w)$$

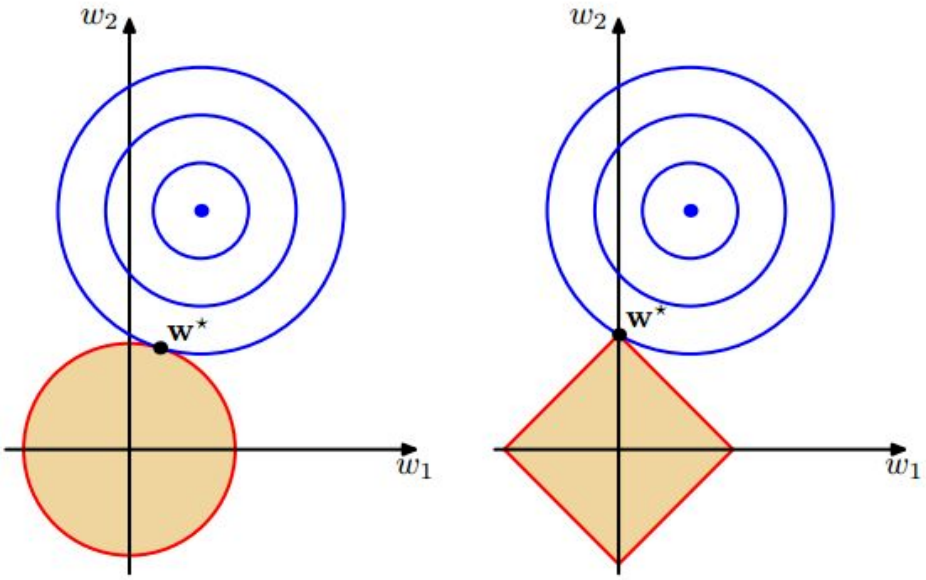
其中，$E_D(w)$ 是损失函数，$E_R(w)$ 是正则化项；

学过最优化的同学都知道，上述表达式等价于：

$$\min_{w} E_D(w)$$
$$s.t.\ E_R(w) \leq \eta$$

即对一个特定的 λ 总存在一个 η 使得这两个问题是等价的。

这个优化问题，把 w 的解限制在如下图所示的黄色区域内，同时使得经验损失尽可能小：



当 $E_R(w)$ 为 L2 正则（如左图）时，此时可行解空间为一个多维空间中的超球体； 当 $E_R(w)$ 为 L1 正则（如右图）时，此时可行解空间为一个多维空间中的多面棱锥体；而蓝色区域为损失函数的可行区域，它们焦点就是可行解。这样，我们很直观地就明白为什么 L1 能产生稀疏解了。

10.如何解决数据不平衡问题

（1）训练集重新采样

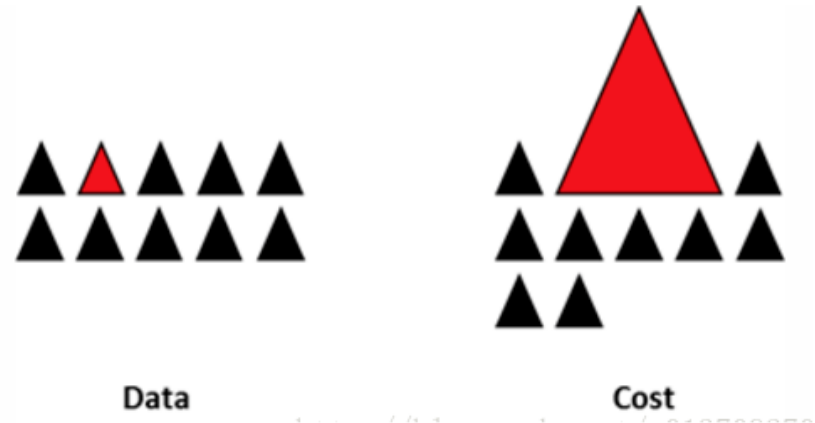
有两种方法使不平衡的数据集来建立一个平衡的数据集——欠采样和过采样:

欠采样是通过减少丰富类的大小来平衡数据集，当数据量足够时就该使用此方法。通过保存所有稀有类样本，并在丰富类别中随机选择与稀有类别样本相等数量的样本，可以检索平衡的新数据集以进一步建模；

相反，当数据量不足时就应该使用过采样，它尝试通过增加稀有样本的数量来平衡数据集，而不是去除丰富类别的样本的数量。通过使用重复、自举或合成少数类过采样等方法（SMOTE）来生成新的稀有样本。

（2）设计适用于不平衡数据集的模型

通过设计一个代价函数来惩罚稀有类别的错误分类而不是分类丰富类别，可以设计出许多自然泛化为稀有类别的模型。如下图所示，加大稀有类别分类错误时的惩罚：



（3）聚类丰富类

该方法类似于欠采样，但不同的是，它不是简单地从富有类原始数据中直接选择出 K 个样例，而是采用聚类的思想。首先采用聚类的方法产生 K 个聚类中心，然后用这 K 个聚类中心与稀有类别进行训练。

（4）多模型 Bagging

我本人在随机森林算法思想的启发下，想出了在上述方法的基础上，将不同比例下训练出来的模型进行多模型 Bagging操作，具体的步骤如下：

- 对两类样本选取 N 组不同比例的数据进行训练并测试，得出模型预测的准确率：

$$P = \{p_i | i = 1, 2, \dots N\}$$

- 对上述各模型的准确率进行归一化处理，得到新的权重分布：

$$\Omega = \{\omega_i | i = 1, 2, \dots N\}$$

其中：

$$\omega_i = \frac{p_i}{\sum_{i=0}^{N} p_i}$$

- 按权重分布 Ω 组合多个模型，作为最终的训练器：

对于分类任务：

$$Model = \underset{i}{argmax} \sum_{y_i=i} \omega_i$$

对于回归任务：

$$Model = \sum_{i=1}^{N} \omega_i y_i$$

11.经验风险最小化与结构风险最小化

12.常见数据清洗方法

数据清洗主要用于解决两大类问题：数据值缺失和数据值异常。

- **缺失值处理**

1. **删除包含缺失值的样本：** 当样本数很多的时候，并且出现缺失值的样本在整个的样本的比例相对较小，这种情况下，我们可以使用最简单有效的方法处理缺失值的情况。那就是将出现有缺失值的样本直接丢弃。这是一种很常用的策略；
2. **均值填补法：** 根据缺失值的属性相关系数最大的那个属性把数据分成几个组，然后分别计算每个组的均值，把这些均值放入到缺失的数值里面就可以了；
3. **模型预测法：** 如果某一列属性值存在大量缺失，我们可以将剩余的属性当成特征，而该列属性中未缺失的当成预测值（回归分类都可以）。学习出一个模型，然后利用该模型对缺失值部分进行预测填补。

- **异常值检测和处理**

1. 统计学方法检测异常值；
2. 使用本课程学习到的 IForest算法检测异常值；
3. 使用One Class SVM检测异常值。

13.机器学习算法评价指标

在介绍算法评价指标之前我们需要搞清楚如下几个概念：

- **TP（True Positive）：** 被正确预测的正样本数；
- **FP（False Positive）：** 被错误预测的正样本数；
- **TN（True Negative）：** 被正确预测的负样本数；
- **FN（False Negative）：** 被错误预测的负样本数。

有了如上概念，我们可以定义如下所示的指标：

（1）**精确率（Precision）**

$$P = \frac{TP}{TP + FP}$$

（2）**召回率（Recall）**

$$R = \frac{TP}{TP + FN}$$

（3）**ROC 曲线与 AUC 值**

限于篇幅，大家可以自行查阅相关资料，但是**这个概念很重要。**

14.欠拟合与过拟合及其解决方法

- **欠拟合**

如下图3所示，欠拟合其实就是算法学习过程中未充分学习到数据的内在规律，导致学习的模型出现很大的偏差，预测准确率不高。

- **过拟合**

与欠拟合相反，过拟合是由于算法模型学习过于充分，导致模型过于复杂。不仅学习到了数据的通用规律，还学习出了训练样本的特有规律，导致模型在训练数据上具有很高的准确性。但是对未知样本的预测能力非常差，在机器学习领域有个专有名词叫做“泛化能力”非常差。

- **解决方法**

欠拟合的解决方法，其实很简单，我们可以增加模型的复杂度来实现。具体来讲，对于类似于神经网络或者 Boosting 算法我们可以增加算法的迭代轮数；对于决策树可以增加树的深度和叶子节点个数的方法等。

过拟合的解决方法，相对复杂一点，其实用一句话讲就是限制模型的复杂度。典型的在理论方面的做法是增加正则化项的方法来限制模型的复杂度；当然也可以通过增加训练样本来防止过拟合；对于神经网络或者深度学习来说，典型的 Dropout 就可以做到防止过拟合。

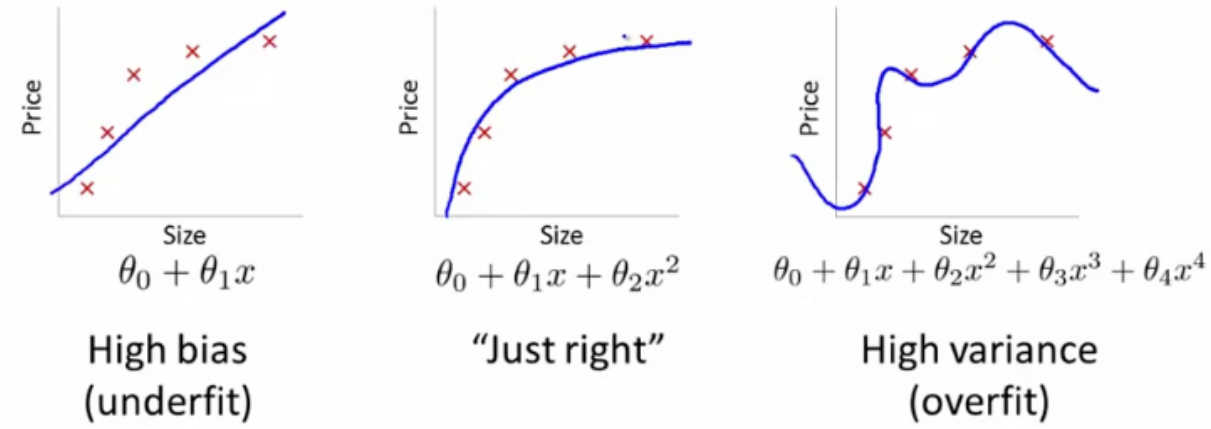


图3. 过拟合与欠拟合

15.GBDT 的原理

16.熵、条件熵、信息增益、信息增益比、Gini 指数

Gini指数： 在分类问题中，假设数据集 D 有 K 个类，样本点属于第 k 类的概率为 p_k，则概率分布的基尼指数定义为：

$$Gini(D) = \sum_{k=1}^{K} p_k(1-p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

同样的，在特征 A 的条件下，集合 D 的基尼指数定义为：

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

其中 D_1 表示特征 A 取值为 a 的样本集，$|D|$ 为全部训练集，$D_2 = D - D_1$。

17.为什么要使用 ROC 和 AUC

请参阅周志华《机器学习》P33。

18.XGBoost 是如何进行特征选择和特征重要性计算的

XGBoost 中计算特征重要性主要使用的是特征在分裂节点中使用的次数和总信息增益两种方法。

19.介绍几个在线学习方法

1. FTRL（Follow The Regularized Leader）
2. BPR（Bayesian Probit Regression）

它主要区别于传统算法训练与预测分离，或者说模型更新周期很长。而在线学习方法会实时将用户行为数据返回更新算法参数，使得模型更新很及时准确。

20.决策树的剪枝策略及其优缺点

决策树的剪枝方法主要分为两大类：预剪枝和后剪枝两种。

预剪枝： 当最优分裂点对应的增益值为负值是停止分裂。

它的优点是，计算时间上能保证最优；缺点则是将来的某个时刻也许能够获取更高的增益，也就是说它不能保证最优。

后剪枝： 将决策树增长到它的最大深度，递归的进行剪枝，剪去那些使得增益值为负值的叶子节点。

它的优点是能够保证决策树最优；缺点是比预剪枝计算复杂度高很多。

21.是交叉验证，它的作用是什么

首先，我们要知道交叉验证是一种常用的模型选择的方法。对于给定的样本集数据充足，我们可以简单地将数据分成三部分：**训练集**（Training Set）、**验证集**（Validation Set）、**测试集**（Test Set）。训练集用于训练模型，验证集用于模型的选择，而测试机则用于最终对学习方法的评估。

K-折交叉验证： 首先，将训练数据集随机地切分为 K 个互不相交的大小相同的子集；然后依次利用第 $k\ (k=1,2,\cdots,K)$ 个子集作为测试集，其余的 $K-1$ 个子集作为训练集。这样就训练出 K 个模型，并且每个模型都在相应的测试集上有它对应的误差。最终将平均测试误差作为该模型的误差，最后我们选择误差最小的模型。

留一法： 当 K-折交叉验证的 K=N（训练集大小）时，称为留一交叉验证，该方法主要用在训练数据集比较缺乏的情况。

综合以上，我们知道，交叉验证的作用是**用于进行模型选择和评估。**

常见机器学习算法优缺点

这个世界上没有完美的事物，机器学习算法也不例外。所有的机器学习或多或少都有其优缺点：

1. 存在即合理，这是每个算法能够广为人知的原因，也就是说它们都有优点；
2. 既然有这么多算法存在，即说明没有哪种算法是万能的，它们又有自身的不足之处。下面我们将常见机器学习算法优缺点总结如下：

KNN

- **优点**

1. 简单好用，容易理解，精度高，理论成熟，既可以用来做分类也可以用来做回归；
2. 可用于数值型和离散型数据；
3. 训练时间复杂度为O(n)，无数据输入假定；
4. 对异常值不敏感。

- **缺点**

1. 计算复杂性高；空间复杂性高；
2. 样本不平衡问题；

朴素贝叶斯

- **优点**

1. 生成式模型，通过计算概率来进行分类，可以用来处理多分类问题；
2. 对小规模的数据表现很好，适合多分类任务，适合增量式训练，算法也比较简单。

- **缺点**

1. 对输入数据的表达形式很敏感；
2. 由于朴素贝叶斯的“朴素”特点，所以会带来一些准确率上的损失；
3. 需要计算先验概率，分类决策存在错误率.

决策树

- **优点**

1. 概念简单，计算复杂度不高，可解释性强，输出结果易于理解；
2. 数据的准备工作简单，能够同时处理数据型和常规型属性，其他的技术往往要求数据属性的单一；
3. 对中间值得确实不敏感，比较适合处理有缺失属性值的样本，能够处理不相关的特征；
4. 应用范围广，可以对很多属性的数据集构造决策树，可扩展性强。决策树可以用于不熟悉的数据集合，并从中提取出一些列规则，这一点强于 KNN。

- **缺点**

1. 容易出现过拟合；
2. 对于那些各类别样本数量不一致的数据，在决策树当中，信息增益的结果偏向于那些具有更多数值的特征；
3. 信息缺失时处理起来比较困难，忽略数据集中属性之间的相关性。

SVM

- **优点**

1. 泛化能力比较强，泛化错误率低；
2. 解决小样本情况下的机器学习问题；
3. 可用于线性和非线性分类，也可以用于回归；
4. 可以解决高维问题，可以避免神经网络结构选择和局部极小点问题。

- **缺点**

1. 对参数调节和和函数的选择敏感，原始分类器不加修改仅适用于处理二分类问题；
2. 对于核函数的高维映射解释力不强，尤其是径向基函数；
3. 对缺失数据敏感；
4. 在数据量大的时候，计算开销太大。

逻辑回归

- **优点**

1. 实现简单，易于理解和实现；
2. 计算代价不高，速度很快，存储资源低；
3. 实现简单，广泛的应用于工业问题上；
4. 便利的观测样本概率分数。

- **缺点**

1. 容易欠拟合，分类精度可能不高；
2. 不能拟合非线性数据；
3. 对模型中自变量多重共线性较为敏感，例如两个高度相关自变量同时放入模型，可能导致较弱的一个自变量回归符号不符合预期，符号被扭转。需要利用因子分析或者变量聚类分析等手段来选择代表性的自变量，以减少候选变量之间的相关性。

AdaBoost

- **优点**

1. 很好的利用了弱分类器进行级联；
2. 可以将不同的分类算法作为弱分类器；
3. AdaBoost 具有很高的精度
4. 相对于 bagging 算法和 Random Forest 算法，AdaBoost 充分考虑的每个分类器的权重。

- **缺点**

1. AdaBoost 迭代次数也就是弱分类器数目不太好设定，可以使用交叉验证来进行确定；
2. 数据不平衡导致分类精度下降；
3. 训练比较耗时，每次重新选择当前分类器最好切分点。

GBDT

- **优点**

1. 可以灵活处理各种类型的数据，包括连续值和离散值；
2. 在相对少的调参时间情况下，预测的准备率也可以比较高（这个是相对 SVM 来说的）；
3. 使用一些健壮的损失函数，对异常值的鲁棒性非常强。比如 Huber 损失函数和 Quantile 损失函数。

- **缺点**

1. 由于弱学习器之间存在依赖关系，难以并行训练数据；
2. 不适合处理高维稀疏特征（相对于神经网络）。

随机森林

- **优点**

1. 它能够处理很高维度（feature 很多）的数据，并且不用做特征选择；
2. 能够计算特征重要性；
3. 模型泛化能力强；
4. 训练速度快，容易做成并行化方法。

- **缺点**

1. 随机森林已经被证明在某些噪音较大的分类或回归问题上会过拟；
2. 对于有不同取值的属性的数据，取值划分较多的属性会对随机森林产生更大的影响，所以随机森林在这种数据上产出的属性权值是不可信的。

神经网络

- **优点**

1. 分类准确度高，学习能力极强；
2. 对噪声数据鲁棒性和容错性较强；
3. 有联想能力，能逼近任意非线性关系。

- **缺点**

1. 神经网络参数较多，权值和阈值；
2. 黑盒过程，不能观察中间结果；
3. 学习过程比较长，有可能陷入局部极小值。