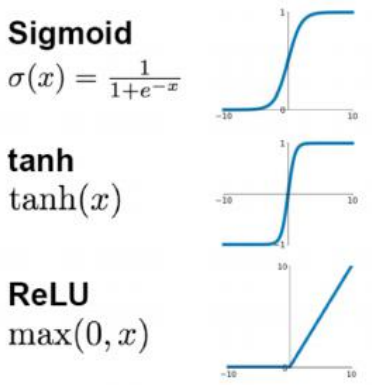
机器学习面试题

目录

激活函数

1、请问人工神经网络中为什么 ReLU 要好过于 tanh 和 Sigmoid function？



第一，采用Sigmoid 等函数，算激活函数时（指数运算），计算量大，反向传播求误差梯度时，求导涉及除法和指数运算，计算量相对大，而采用ReLU 激活函数，整个过程的计算量节省很多。

第二，对于深层网络，Sigmoid 函数反向传播时，很容易就会出现梯度消失的情况（在 Sigmoid 接近饱和区时，变换太缓慢，导数趋于 0，这种情况会造成信息丢失），这种现象称为饱和，从而无法完成深层网络的训练。而ReLU 就不会有饱和倾向，不会有特别小的梯度出现。

第三，ReLU 会使一部分神经元的输出为 0，这样就造成了网络的稀疏性，并且减少了参数的相互依存关系，缓解了过拟合问题的发生（以及一些人的生物解释 balabala）。当然现在也有一些对 ReLU 的改进，比如 PReLU，random ReLU 等，在不同的数据集上会有一些训练速度上或者准确率上的改进。

现在主流的做法，会多做一步 batch normalization，尽可能保证每一层网络的输入具有相同的分布 。而较新的 paper ，他们在加入bypass connection 之后，发现改变 batch normalization 的位置会有更好的效果。

逻辑回归：

1、能写一下逻辑回归的损失函数吗？为什么不用 MSE(L2 loss)作为损失函数

L（yhat，y） = -ylog（yhat）-（1-y）log（1-yhat）

不用 MSE 做损失函数的原因：

a. 损失函数的角度：逻辑回归预测函数是非线性的，采用 MSE 得到的损失函数是非凸函数，会存在很多局部极小值，梯度下降法可能无法获得全局最优解。

b.极大似然的角度: 采用极大似然法估计逻辑回归模型的参数，最终得到的对数似然函数形式与对数损失函数一致。

2、逻辑回归和最大似然有什么关系？

通过极大似然法可以求解逻辑回归模型中的参数。最大似然估计就是通过已知结果去反推最大概率导致该结果的参数。极大似然估计是概率论在统计学中的应用，它提供了一种给定观察数据来评估模型参数的方法，即“模型已定，参数未知”，通过若干次试验，观察其结果，利用实验结果得到某个参数值能够使样本出现的概率为最大，则称为极大似然估计。逻辑回归是一种监督式学习，是有训练标签的，就是有已知结果的，从这个已知结果入手，去推导能获得最大概率的结果参数θ，只要我们得出了这个参数，那我们的模型就自然可以很准确的预测未知的数据了。

3、逻辑回归用梯度下降优化，学习率对结果有什么影响？

a.学习率过低则模型训练速度会慢

b.学习率过高则模型训练会在全局最优点附近震荡，甚至不收敛

4、逻辑回归中样本不均衡我们怎么处理？

a. 调整分类阈值，不统一使用 0.5，根据样本中类别的比值进行调整。

b.多类样本负采样。进一步也可将多类样本负采样构建多个训练集，最后聚合多个模型的结果。

c.少类样本过采样。过采样的方法大致有三种：

c1: 随机复制

c2: 基于聚类的过采样

c3: SMOTE

d. 改变性能指标，推荐采用 ROC AUC、F1 Score4.

e.模型训练增加正负样本惩罚权重,少类样本权重加大，增大损失项。

5、lr 模型是线性模型还是非线性， 为什么？ 能推导它的原理吗？

lr 模型是广义线性模型，lr 的原始形式是非线性的，决定这个复合函数是否是线性的，要看 g（x）的形式，假如阈值为 0.5，那么 g(x)>=0 时，预测为 1，否则预测为 0.因此，这里的 g(x)，实际是一个决策面，这个决策面的两侧分别是正例和负例。逻辑回归的作用是把决策面两侧的点映射到逻辑回归曲线阈值的两侧。

属于线性模型。

因为 logistic 回归的决策边界（decision boundary）是线性的：

$$P(Y=1|\mathbf{x},\boldsymbol{\theta}) = P(Y=0|\mathbf{x},\boldsymbol{\theta})$$

$$\frac{1}{1+e^{-\boldsymbol{\theta}^T\cdot\mathbf{x}}} = \frac{e^{-\boldsymbol{\theta}^T\cdot\mathbf{x}}}{1+e^{-\boldsymbol{\theta}^T\cdot\mathbf{x}}}$$

$$1 = e^{-\boldsymbol{\theta}^T\cdot\mathbf{x}}$$

$$\boldsymbol{\theta}^T\cdot\mathbf{x} = 0$$

6、lr 使用注意事项，相比别的分类模型为什么使用它？

注意，建模数据量不能太少，目标变量中每个类别所对应的样本数量要足够充分，才能支持建模排除共线性问题（自变量间相关性很大）。

异常值会给模型带来很大干扰，要剔除。

逻辑回归不能处理缺失值，所以之前应对缺失值进行适当处理。

相对于别的模型用 lr 的好处是在于在数据量比较大的情况下越复杂的模型可能会越慢，lr 就显得会快上很多了

7、（欢聚时代 YY）LR 和 DNN 联系和区别是什么？

二者都是用于二分类问题，且无法学习比较复杂的非线性模型，区别在于 DNN 是神经网络，输出结果为 1，-1，lr 输出结果为 0，1；DNN 的激活函数为 sign，lr 的激活函数为 sigmoid。

8、LR/SVM/softmax/Adaboost 损失函数之间的差别？

softmax 模型是 LR 模型从二分类推广到多分类的模型，两者损失函数都是对数损失。SVM 使用的是合页损失，分类边界只受限于支持向量。

Adaboost 损失函数是指数损失。

9、LR/DT/SVM 模型之间对比

LR 是一个分类模型，DT 和 SVM 既可以做分类也可以做回归。LR 的决策面与线性 SVM 的决策面基本相同，原理上来说，分类问题 SVM 的决策面由支持向量决定，LR 的决策面是由所有数据点决定，更进一步的说，LR 不同数据点对决策面的贡献是不同的(由参数更新函数决定)，离决策面近的贡献大，因此 SVM 相对更加健壮，LR 的决策面可能会受到正负样本点的分布的影响。DT 与两者之间的差别在于决策面是阶梯型的，这是因为本质上说决策树进行决策时，样本空间是由各个特征的某个值进行划分的因此平行于特征轴，所以导致决策面是这种形态。

10、（拼多多）LR 怎么优化

优化方法，梯度下降，牛顿法

1 梯度下降法

可以理解为每次按照梯度下降的方向下降一定的步长，以寻找最优解的位置

同理此时求的是代价函数的最小值 → 梯度下降；求最大值时用梯度上升法，即系数 1/m 为正

特点：

梯度下降法比较万能，基本都能用

此处使用梯度下降，也就是说求似然函数的最小值（似然函数是代价函数），将 1/m 前改为 + 则梯度上升法，可以求最大值

梯度下降法一般很慢

2 牛顿迭代法

思路 ：对 \beta β 给定一个初始值，进行迭代

特点：

很快，比梯度下降法快得多

注意以上 \beta β 的二阶导(黑塞矩阵 Hessian)不一定是可逆的, 迭代公式可能根本进行不下去

同理由于迭代步长固定为 1，容易出现无法收敛的情况

Hessian 矩阵不可导时，考虑拟牛顿法；无法收敛时考虑阻尼牛顿法

3.拟牛顿法（BFGS）

考虑到梯度下降法很慢，而牛顿法存在黑塞矩阵不可逆的问题，考虑拟牛顿法思路（BFGS）：通过迭代的方式，求得一个黑塞矩阵逆的近似矩阵，在每次进行目标值的迭代时，沿着迭代方向进行 armijo 搜索确定的步长的迭代，余下思路与牛顿法相同

BFGS 法整体步骤：

用所有下标带 k 的先给出 beta_k+1 （得到的过程中步长因子需要迭代

判断 此时的 l(b) 目标函数下降足够少, 即退出迭代

计算 y_k

迭代黑塞矩阵的近似值 G 矩阵

重新迭代计算新的 beta

11、（拼多多）谈谈牛顿法

收敛速度快，指数级收敛，但是不保证线性收敛，只保证二阶导收敛。

牛顿法比梯度下降法收敛的要快，这是因为牛顿法是二阶收敛，梯度下降是一阶收敛。事实上，梯度下降法每次只从当前位置选一个上升速度最大的方向走一步，牛顿法在选择方向时，不仅会考虑上升速度是否够大，还会考虑你走了一步之后，上升速度是否会变得更大。，所以所需要的迭代次数更少。从几何上说，牛顿法就是用一个二次曲面去拟合你当前所处位置的局部曲面，而梯度下降法是用一个平面去拟合当前的局部曲面，通常情况下，二次曲面的拟合会比平面更好，所以牛顿法选择的下降路径会更符合真实的最优下降路径。

12、（拼多多）牛顿法怎么优化

假设要求解方程 f(x)=0，

牛顿法就是用泰勒公式将 f(x)在 x0 处一阶展开，即

f(x)=x0+f′(x0)* (x−x0)

令 f(x)=0

得到 x1=x0−x0/f′(x0)

直观的来看，就是在 x0 处沿着一阶导数的方向做一条切线，然后取切线与 x 轴的交点作为新的近似值。

这里可以用牛顿法求 w

w 的最优解，因为最优解满足

∂l(w)/∂=0,所以就可以用牛顿法来求解了

牛顿法比梯度下降法收敛的要快，这是因为牛顿法是二阶收敛，梯度下降是一阶收敛。事实上，梯度下降法每次只从当前位置选一个上升速度最大的方向走一步，牛顿法在选择方向时，不仅会考虑上升速度是否够大，还会考虑你走了一步之后，上升速度是否会变得更大。，所以所需要的迭代次数更少。从几何上说，牛顿法就是用一个二次曲面去拟合你当前所处位置的局部曲面，而梯度下降法是用一个平面去拟合当前的局部曲面，通常情况下，二次曲面的拟合会比平面更好，所以牛顿法选择的下降路径会更符合真实的最优下降路径。

13、（拼多多）Cross Encropy 是怎么推导出来的

交叉熵（Cross Entropy）是 Shannon 信息论中一个重要概念，主要用于度量两个概率分布间的差异性信息。语言模型的性能通常用交叉熵和复杂度（perplexity）来衡量。交叉熵的意义是用该模型对文本识别的难度，或者从压缩的角度来看，每个词平均要用几个位来编码。复杂度的意义是用该模型表示这一文本平均的分支数，其倒数可视为每个词的平均概率。平滑是指对没观察到的 N 元组合赋予一个概率值，以保证词序列总能通过语言模型得到一个概率值。通常使用的平滑技术有图灵估计、删除插值平滑、Katz 平滑和 Kneser-Ney 平滑。

14、（天眼查）为什么逻辑回归做对数转换时选用以 e 为底？

因为 ln'(x) = 1/x，求导比较好做计算

15、（每日优鲜）lr 为什么不采用 mse 而是采用交叉熵损失？

lr 使用 mse 为损失函数时，是非凸的损失函数，不方便优化；而对数损失(二元交叉熵损失)能保证损失函数为凸函数，优化到全局最低点。

如果采用 mse 作为损失函数，最后的梯度计算中会存在 sigmoid 函数的导数项。而交叉熵损失不存在这个问题，只有差值与特征值的乘积。

sigmoid 函数的导数项其梯度是小于等于 0.25 的，而且在预测概率接近 1 或者接近 0 时存在梯度饱和区问题，使得训练困难。这也是深度学习中梯度消失的原因之一。

16、（阿里高德）交叉熵的物理含义

交叉熵是由相对熵导出的，相对熵的含义是衡量两个概率分布 p 与 q 之间的差异。

相对熵的前半部分是交叉熵，后半部分是常数，相对熵达到最小值的时候，交叉熵也达到了最小值，所以交叉熵也可以衡量计算之后的概率分布 q 与真实概率分布 p 之间的差异。

交叉熵最小值等同于最大似然估计。

17、最小二乘法的解

最小二乘法是勒让德(A. M. Legendre)于 1805 年在其著作《计算慧星轨道的新方法》中提出的。它的主要思想就是求解未知参数，使得理论值与观测值之差（即误差，或者说残差）的平方和达到最小：

$$E=\sum_{i=1}^{n} e_i^2=\sum_{i=1}^{n}(y_i-\hat{y})^2$$

我们从矩阵的角度来理解，首先我们给出一个矩阵中的定义：

$$R(A)=\{Ax \mid x \in R^n\}, A \in R^{n \times n}$$

有了上面的定义之后，我们就可以写出最小二乘问题的矩阵形式：

$$\exists b \notin R(A), b \in R^n, \min_{x \in R^n} \| Ax-b \|_2$$

是求在欧几里得空间中以 2-范数作为距离，使得向量 Ax 与 b 之间距离最小的 x。我们的目标是求：

$$\min_{x\in R^n} \| Ax-b \|_2$$

当然我们知道，使得距离最小的向量 x 与使得距离平方最小的向量 x 是相同的，于是我们可以将所求的目标改写为：

$$\min_{x\in R^n} \| \mathrm{Ax}-\mathrm{b} \|_2^2$$

结合一些矩阵、行列式的知识，我们知道：

$$\| Ax-b \|_2^2=(Ax-b)^T*(Ax-b)$$

我们知道，求最极值问题直接对应的就是导数为零，因此我们试图将所给出的原式的矩阵形式求导。对原式化简并求其对 x 的导数：

$$\| Ax-b \|_2^2=x^TA^TAx-b^TAx-x^TA^Tb+b^Tb$$

求导得到：

$$\frac{\partial \| Ax-b \|_2^2}{\partial x}=2A^TAx-2A^Tb=0$$

于是我们就得到了，最小二乘法解的矩阵形式：

$$x=(A^TA)^{-1}A^Tb$$

注：这里是最简单的线性最小二乘法

SVM：

1、（网易游戏）svm，lr 的区别联系，重点介绍下 svm 算法。

a.支持向量机只考虑局部的边界线附近的点，而逻辑回归考虑全局（远离的点对边界线的确定也起作用，虽然作用会相对小一些。

b.SVM 是结构最小化算法，LR 需另外在损失函数上添加正则项。

c.LR 输出具有自然的概率意义，支持向量机输出不具有概率意义。

d.SVM 采用核函数解决非线性问题，LR 通常不采用核函数的方法。

假设： n= 特征数量，m= 训练样本数量

1）如果 n 相对于 m 更大，比如 n = 10,000，m = 1,000，则使用 LR 或线性 SVM

理由：特征数相对于训练样本数已经够大了，使用线性模型就能取得不错的效果，不需要过于复杂的模型；

2）如果 n 较小，m 比较大，比如 n = 10，m = 10,000，则使用 SVM（高斯核函数）

理由：在训练样本数量足够大而特征数较小的情况下，可以通过使用复杂核函数的 SVM 来获得更好的预测性能，而且因为训练样本数量并没有达到百万级，使用复杂核函数的 SVM 也不会导致运算过慢；

3）如果 n 较小，m 非常大，比如 n = 100, m = 500,000，则应该引入 / 创造更多的特征，然后使用 lr 或者线性核函数的 SVM

理由：因为训练样本数量特别大，使用复杂核函数的 SVM 会导致运算很慢，因此应该考虑通过引入更多特征，然后使用线性核函数的 SVM 或者 lr 来构建预测性更好的模型。

2、（网易游戏）核函数要满足什么性质，常见的核函数有哪些。

核函数的性质

核函数必须是连续的，对称的，并且最优选地应该具有正（半）定 Gram 矩阵。据说满足 Mercer 定理的核是正半定数，意味着它们的核矩阵只有非负特征值。使用肯定的内核确保优化问题将是凸的和解决方案将是唯一的。

然而，许多并非严格定义的核函数在实践中表现得很好。一个例子是 Sigmoid 内核，尽管它广泛使用，但它对于其参数的某些值不是正半定的。 Boughorbel（2005）也实验证明，只有条件正定的内核在某些应用中可能胜过大多数经典内核。

内核还可以分为各向异性静止，各向同性静止，紧凑支撑，局部静止，非稳定或可分离非平稳。此外，内核也可以标记为 scale-invariant（规模不变）或 scale-dependent（规模依赖），这是一个有趣的属性，因为尺度不变内核驱动训练过程不变的数据的缩放。

补充：Mercer 定理：任何半正定的函数都可以作为核函数。所谓半正定的函数 f(xi,xj)，是指拥有训练数据集合（x1,x2,...xn)，我们定义一个矩阵的元素 aij = f(xi,xj)，这个矩阵式 n*n 的，如果这个矩阵是半正定的，那么 f(xi,xj)就称为半

正定的函数。这个 mercer 定理不是核函数必要条件，只是一个充分条件，即还有不满足 mercer 定理的函数也可以是核函数

几种常用的核

1 线性核

线性内核是最简单的内核函数。它由内积<x，y>加上可选的常数 c 给出。使用线性内核的内核算法通常等于它们的非内核对应物，即具有线性内核的 KPCA 与标准 PCA 相同。

2 多项式核函数

多项式核是非固定内核。 多项式内核非常适合于所有训练数据都归一化的问题。

3 高斯核

高斯核是径向基函数核的一个例子。

可调参数 sigma 在内核的性能中起着主要作用，并且应该仔细地调整到手头的问题。 如果过高估计，指数将几乎呈线性，高维投影将开始失去其非线性功率。 另一方面，如果低估，该函数将缺乏正则化，并且决策边界将对训练数据中的噪声高度敏感。

4 指数的内核

指数核与高斯核密切相关，只有正态的平方被忽略。 它也是一个径向基函数内核。

5 拉普拉斯算子核

拉普拉斯核心完全等同于指数内核，除了对 sigma 参数的变化不那么敏感。作为等价的，它也是一个径向基函数内核。

3、（网易游戏）高斯核函数在使用过程中要注意什么。

高斯核

高斯核是径向基函数核的一个例子。

可调参数 sigma 在内核的性能中起着主要作用，并且应该仔细地调整到手头的问题。 如果过高估计，指数将几乎呈线性，高维投影将开始失去其非线性功率。 另一方面，如果低估，该函数将缺乏正则化，并且决策边界将对训练数据

中的噪声高度敏感。

4、SVM 的原理是什么？

SVM 是一种二分类模型，学习的目标是在特征空间中找到一个分离超平面，且此超平面是间隔最大化的最优分离超平面，最终可转化为一个凸二次规划问题求解。

5、SVM 为什么采用间隔最大化？

间隔最大化使超平面所产生的结果是最鲁棒的，对未见示例的泛化能力最强。

6、什么样的函数可以作为 SVM 的核函数？

由于一般我们说的核函数都是正定核函数，这里我们直说明正定核函数的充分必要条件。一个函数要想成为正定核函数，必须满足他里面任何点的集合形成的 Gram 矩阵是半正定的。

7、SVM 对缺失数据敏感吗，为什么？

敏感。SVM 模型涉及到样本点“距离”的度量，缺失数据会非常重要。

8、SVM 的损失函数是什么形式？可以用梯度下降优化吗？

SVM 的损失函数是 hinge 损失。hinge 损失函数是凸函数，可以使用梯度下降进行优化。由于 hinge 损失函数不可导，可采用次梯度下降法进行优化。

9、SVM 的高斯核为什么会把原始维度映射到无穷多维？

对于高斯核为什么可以将数据映射到无穷多维，我们可以从泰勒展开式的角度来解释，

首先我们要清楚，SVM中，对于维度的计算，我们可以用内积的形式，假设函数：

$\kappa(x_1, x_2) = (1, x_1 x_2, x^2)$ 表示一个简单的从二维映射到三维。

则在SVM的计算中，可以表示为：

$\kappa(x_1, x_2) = 1 + x_1 x_2 + x^2$

再来看e^x泰勒展开式：

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}$$

所以这个无穷多项的式子正是对于e^x的近似，e^x所对应的映射：

$$\kappa(x) = \left(1, x, \frac{x^2}{2!}, \frac{x^3}{3!}, \ldots, \frac{x^n}{n!}\right)$$

再来看高斯核：

$$\kappa(x_1, x_2) = e^{\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right)}$$

将泰勒展开式带入高斯核，我们得到了一个无穷维度的映射：

$$\kappa(x_1, x_2) = 1 + \left(-\frac{||x_1 - x_2||^2}{2\sigma^2}\right) + \frac{(-\frac{\|x_1 - x_2\|^2}{2\sigma^2})^2}{2!} + \ldots + \frac{(-\frac{\|x_1 - x_2\|^2}{2\sigma^2})^3}{3!} + \ldots + \frac{(-\frac{\|x_1 - x_2\|^2}{2\sigma^2})^n}{n!}$$

那么，对于 x_1和 x_2的内积形式符合在SVM中无穷维度下的内积计算，即高斯核将数据映射到无穷高的维度。

核函数的本质简要概括下，即以下三点：

1.实际中，我们会经常遇到线性不可分的样例，此时，我们的常用做法是把样例特征映射到高维空间中去(映射到高维空间后，相关特征便被分开了，也就达到了分类的目的)；

2.但进一步，如果凡是遇到线性不可分的样例，一律映射到高维空间，那么这个维度大小是会高到可怕的(如 19 维乃至无穷维)。

3.此时，核函数就隆重登场了，核函数的价值在于它虽然也是将特征进行从低维到高维的转换，但核函数绝就绝在它事先在低维上进行计算，而将实质上的分类效果表现在了高维上，也就如上文所说的避免了直接在高维空间中的复杂计算。

10、为什么要将求解 SVM 的原始问题转换为其对偶问题？

对偶问题往往更容易求解。

11、为什么 SVM 要引入核函数？

原始样本空间可能会线性不可分，这样需要将原始空间映射到一个更高维的特征空间，使得样本在这个特征空间线性可分。样本映射到高维空间后的内积求

解通常是困难的，引入核函数可简化内积的计算。

DT：

1、解释一下决策树的建模过程？

分类决策树模型是一种描述对实例进行分类的树形结构。决策树由结点和有向边组成。结点有两种类型：内部结点和叶结点。内部结点表示一个特征或属性，叶节点表示一个类。

用决策树分类，从根节点开始，对实例的某一特征进行测试，根据测试结果，将实例分配到其子节点；每一个子节点对应着该特征的一个取值。如此递归地对实例进行测试并分配，直至达到叶节点。最后将实例分到叶节点的类中。

决策树学习算法包含特征选择、决策树的生成和决策树的剪枝过程。

2、有几种不同的决策树，区别在哪？

ID3 决策树的输入训练数据是一组带有类别标记的样本，构造的结果是一棵多叉树。树的分支节点一般表示为一个逻辑判断，如形式为 a=aj 的逻辑判断，其中 a 是属性，aj 是该属性的所有取值。如果选择在某一节点上用哪个特征呢？标准是在该节点上选取能对该节点处的训练数据进行最优划分的属性。划分的标准是信息增益（Information Gain），即划分前后数据集的熵的差异。取能带来最大信息增益的那个 feature 作为当前划分标准。

ID3：

离散特征（标称型数据）、贪心算法、信息增益、特征所有取值切分（非二分）

缺点：

无剪枝策略，容易过拟合；信息熵的设定使得高基数类别特征的条件熵会很大。因为高基数类别代表了树要分裂非常非常多的叶子节点，并且每个叶子节点上的样本树很少，越小的数据子集其“纯度”显然越容易高，导致了条件熵会趋向于 0，信息增益会很大；只能用于处理离散分布的特征并且只能处理分类问题；

没有考虑缺失值。

C4.5：

离散/连续特征（对连续属性扫描排序，设定阈值，二分样本）、信息增益比（引入分裂信息(Split information)的项来惩罚取值较多的 Feature）、可剪枝（合并叶节点）、可处理缺失值（可参考缺失值处理）

CART 包含的基本过程有分裂，剪枝和树选择。

分裂：分裂过程是一个二叉递归划分过程，其输入和预测特征既可以是连续型的也可以是离散型的，CART 没有停止准则，会一直生长下去；剪枝：采用代价复杂度剪枝，从最大树开始，每次选择训练数据熵对整体性能贡献最小的那个分裂节点作为下一个剪枝对象，直到只剩下根节点。CART 会产生一系列嵌套的剪枝树，需要从中选出一颗最优的决策树；树选择：用单独的测试集评估每棵剪枝树的预测性能（也可以用交叉验证）。

CART 在 C4.5 的基础上进行了很多提升。

C4.5 为多叉树，运算速度慢，CART 为二叉树，运算速度快；C4.5 只能分类，CART 既可以分类也可以回归；CART 使用 Gini 系数作为变量的不纯度量，减少了大量的对数运算；CART 采用代理测试来估计缺失值，而 C4.5 以不同概率划分到不同节点中；CART 采用“基于代价复杂度剪枝”方法进行剪枝，而 C4.5 采用悲观剪枝方法。

3、决策树的缺失值和数值型特征分别是怎么处理的？

数值型特征：

采用二分法对连续属性进行处理。给定样本集 D 和连续属性 a，假定 a 在 D 上出现了 n 个不同的取值，将这些值从小到大排序，记为（a^1,a^2,…,a^n）。对连续属性 a，我们可考察包含 n-1 个元素的候选划分点集合

缺失值：

(1). 在选择分裂属性的时候，训练样本存在缺失值，如何处理

假如你使用 ID3 算法，那么选择分类属性时，就要计算所有属性的熵增(信息增益，Gain)。假设 10 个样本，属性是 a,b,c。在计算 a 属性熵时发现，第 10 个样本的 a 属性缺失，那么就把第 10 个样本去掉，前 9 个样本组成新的样本集，

在新样本集上按正常方法计算 a 属性的熵增。然后结果乘 0.9（新样本占 raw 样本的比例），就是 a 属性最终的熵。

(2). 分类属性选择完成，对训练样本分类，发现属性缺失怎么办？

比如该节点是根据 a 属性划分，但是待分类样本 a 属性缺失，怎么办呢？假设 a 属性离散，有 1,2 两种取值，那么就把该样本分配到两个子节点中去，但是权重由 1 变为相应离散值个数占样本的比例。然后计算错误率的时候，注意，不是每个样本都是权重为 1，存在分数。

(3). 训练完成，给测试集样本分类，有缺失值怎么办？

这时候，就不能按比例分配了，因为你必须给该样本一个确定的 label，而不是薛定谔的 label。这时候根据投票来确定，或者填充缺失值。

4、（滴滴）CART 树是如何做分类的，是如何做回归的，是如何处理多分类的？

特征选择

特征选择决定了使用哪些特征来做判断。在训练数据集中，每个样本的属性可能有很多个，不同属性的作用有大有小。因而特征选择的作用就是筛选出跟分类结果相关性较高的特征，也就是分类能力较强的特征。

在特征选择中通常使用的准则是：信息增益。

决策树生成：

选择好特征后，就从根节点触发，对节点计算所有特征的信息增益，选择信息增益最大的特征作为节点特征，根据该特征的不同取值建立子节点；对每个子节点使用相同的方式生成新的子节点，直到信息增益很小或者没有特征可以选择为止。

决策树剪枝：

剪枝的主要目的是对抗「过拟合」，通过主动去掉部分分支来降低过拟合的风险。

ID3 算法：

ID3 是最早提出的决策树算法，他就是利用信息增益来选择特征的。

C4.5 算法：

他是 ID3 的改进版，他不是直接使用信息增益，而是引入“信息增益比”指

标作为特征的选择依据。

CART（Classification and Regression Tree）：

这种算法即可以用于分类，也可以用于回归问题。CART 算法使用了基尼系数取代了信息熵模型。

算法输入是训练集 D，基尼系数的阈值，样本个数阈值。输出是决策树 T。

我们的算法从根节点开始，用训练集递归的建立 CART 树。

1) 对于当前节点的数据集为 D，如果样本个数小于阈值或者没有特征，则返回决策子树，当前节点停止递归。

2) 计算样本集 D 的基尼系数，如果基尼系数小于阈值，则返回决策树子树，当前节点停止递归。

3) 计算当前节点现有的各个特征的各个特征值对数据集 D 的基尼系数，对于离散值和连续值的处理方法和基尼系数的计算见第二节。缺失值的处理方法和上篇的 C4.5 算法里描述的相同。

4) 在计算出来的各个特征的各个特征值对数据集 D 的基尼系数中，选择基尼系数最小的特征 A 和对应的特征值 a。根据这个最优特征和最优特征值，把数据集划分成两部分 D1 和 D2，同时建立当前节点的左右节点，做节点的数据集 D 为 D1，右节点的数据集 D 为 D2.

5) 对左右的子节点递归的调用 1-4 步，生成决策树。

对于生成的决策树做预测的时候，假如测试集里的样本 A 落到了某个叶子节点，而节点里有多个训练样本。则对于 A 的类别预测采用的是这个叶子节点里概率最大的类别。

CART 回归树构建过程如下

输入：训练数据集 D；

输出：回归树 $f(x)$.

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并决定每个子区域上的输出值，构建二叉决策树：

（1）选择最优切分变量 j 与切分点 s，求解

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right] \tag{5.21}$$

遍历变量 j，对固定的切分变量 j 扫描切分点 s，选择使式 (5.21) 达到最小值的对 (j,s).

（2）用选定的对 (j,s) 划分区域并决定相应的输出值：

$$R_1(j,s)=\{x\,|\,x^{(j)}\leqslant s\},\quad R_2(j,s)=\{x\,|\,x^{(j)}>s\}$$

$$\hat{c}_m=\frac{1}{N_m}\sum_{x_i\in R_m(j,s)}y_i,\quad x\in R_m,\quad m=1,2$$

（3）继续对两个子区域调用步骤 (1)，(2)，直至满足停止条件.

（4）将输入空间划分为 M 个区域 $R_1,R_2,\cdots,R_M$，生成决策树：

$$f(x)=\sum_{m=1}^{M}\hat{c}_m I(x\in R_m)$$

■

5、决策树怎么控制过拟合?

决策树生成算法递归地产生决策树，直到不能继续下去为止。这样产生的树往往对训练数据的分类很准确，但对未知的测试数据的分类却没有那么准确，即过拟合现象。过拟合的原因在于决策树过于复杂，通过剪枝，从已生成的树上裁剪掉一些子树或叶结点，并将其根节点或父节点作为新的叶结点，从而简化树模型，降低过拟合。

常用的剪枝方法有预剪枝和后剪枝。

我们还可以通过控制树深，对样本或特征进行采样，控制分裂叶子节点最少样本数，最小 gain 提升等来控制过拟合。

6、（网易游戏）树模型，boost tree 介绍下

决策树这种算法有着很多良好的特性，比如说训练时间复杂度较低，预测的过程比较快速，模型容易展示（容易将得到的决策树做成图片展示出来）等。但是同时，单决策树又有一些不好的地方，比如说容易 over-fitting，虽然有一些方法，如剪枝可以减少这种情况，但是还是不够的。

模型组合（比如说有 Boosting，Bagging 等）与决策树相关的算法比较多，这些算法最终的结果是生成 N(可能会有几百棵以上）棵树，这样可以大大的减

少单决策树带来的毛病，有点类似于三个臭皮匠等于一个诸葛亮的做法，虽然这几百棵决策树中的每一棵都很简单（相对于 C4.5 这种单决策树来说），但是他们组合起来确是很强大。

7、决策树/随机森林的特征重要度是怎么获得的？

主要有两种方式：

a.考察样本经过节点后不纯度减少的值的大小，值越大则特征越重要。

b.通过包外数据（out of bag）计算特征加入噪声前后对模型预测准确率的影响，影响越大则特征越重要。

xgboost 中的特征重要度，也可以考虑特征用作分裂的次数

8、（拼多多）决策树，比如 ID3 怎么做特征的离散化的

思想

从信息论的知识中我们知道：信息熵越大，从而样本纯度越低，。ID3 算法的核心思想就是以信息增益来度量特征选择，选择信息增益最大的特征进行分裂。算法采用自顶向下的贪婪搜索遍历可能的决策树空间（C4.5 也是贪婪搜索）。其大致步骤为：

初始化特征集合和数据集合；

计算数据集合信息熵和所有特征的条件熵，选择信息增益最大的特征作为当前决策节点；

更新数据集合和特征集合（删除上一步使用的特征，并按照特征值来划分不同分支的数据集合）；

重复 2，3 两步，若子集值包含单一特征，则为分支叶子节点。

划分标准

ID3 使用的分类标准是信息增益，它表示得知特征 A 的信息而使得样本集合不确定性减少的程度。

9、树形结构为什么不需要归一化？

因为数值缩放不影响分裂点位置，对树模型的结构不造成影响。按照特征值

进行排序的，排序的顺序不变，那么所属的分支以及分裂点就不会有不同。而且，树模型是不能进行梯度下降的，因为构建树模型（回归树）寻找最优点时是通过寻找最优分裂点完成的，

因此树模型是阶跃的，阶跃点是不可导的，并且求导没意义，也就不需要归一化。

既然树形结构（如决策树、RF）不需要归一化，那为何非树形结构比如 Adaboost、SVM、LR、Knn、KMeans 之类则需要归一化呢？

对于线性模型，特征值差别很大时，比如说 LR，我有两个特征，一个是(0,1)的，一个是(0,10000)的，运用梯度下降的时候，损失等高线是椭圆形，需要进行多次迭代才能到达最优点。

但是如果进行了归一化，那么等高线就是圆形的，促使 SGD 往原点迭代，从而导致需要的迭代次数较少。

除了归一化，我们还会经常提到标准化，那到底什么是标准化和归一化呢？

标准化：特征均值为 0，方差为 1

公式：

X-np.mean(x)

Np.std(x)

归一化：把每个特征向量（特别是奇异样本数据）的值都缩放到相同数值范围，如[0,1]或[-1,1]。

最常用的归一化形式就是将特征向量调整为 L1 范数（就是绝对值相加），使特征向量的数值之和为 1。而L2 范数就是欧几里得之和。

data_normalized = preprocessing.normalize(data , norm="L1")

公式：

$$\frac{x-\min(x)}{\max(x)-\min(x)}$$

这个方法经常用于确保数据点没有因为特征的基本性质而产生较大差异，即确保数据处于同一数量级（同一量纲），提高不同特征数据的可比性。

RF：

1、 树模型对于缺失值如何处理？

首先，给缺失值预设一些估计值，比如数值型特征，选择其余数据的中位数或众数作为当前的估计值，然后，根据估计的数值，建立随机森林，把所有的数据放进随机森林里面跑一遍。记录每一组数据在决策树中一步一步分类的路径，然后来判断哪组数据和缺失数据路径最相似，引入一个相似度矩阵，来记录数据之间的相似度，比如有 N 组数据，相似度矩阵大小就是 N*N，如果缺失值是类别变量，通过权重投票得到新估计值，如果是数值型变量，通过加权平均得到新的估计值，如此迭代，直到得到稳定的估计值。xgboost 把缺失值当做稀疏矩阵来对待，本身的在节点分裂时不考虑的缺失值的数值。缺失值数据会被分到左子树和右子树分别计层损失，选择较优的那一个。如果训练中没有数据缺失，预测时出现了数据缺失，那么默认被分类到右子树

Boosting：

1、（百度） Xgboost 的原理能讲一下么？ xgboost 是如何做分类的， 能解释一下原理么?

xgb 是一种 tree boosting 的可扩展机器学习系统，属于监督学习，支持并行处理，是一个优化的分布式梯度增强库，旨在实现高效、灵活和便携。

xgb 做分类是对于每个样本，每棵树都会预测出一个结果，对于二分类问题来说，将所有数的分数相加，经过 sigmoid 转换，得到预测为 1 的概率

2、（百度）RF 和 GBDT 的区别

组成随机森林的树可以分类树也可以是回归树，而 GBDT 只由回归树组成组成随机森林的树可以并行生成，而 GBDT 是串行生成随机森林的结果是多数表决表决的，而 GBDT 则是多棵树累加之和随机森林对异常值不敏感，而 GBDT

对异常值比较敏感随机森林是通过减少模型的方差来提高性能，而 GBDT 是减少模型的偏差来提高性能的随机森林不需要进行数据预处理，即特征归一化。而 GBDT 则需要进行特征归一化

3、（百度）为什么 xgboost 要求二阶导数

泰勒展开逼近残差，二阶比一阶精度高

通用性，所有二阶可导的 loss function 都可以用

有些函数梯度在一阶上变化小，二阶变化大，参考牛顿法

4、介绍一下 RandomForest、Adaboost、GBDT

RandomForest：随机森林是 Bagging 的一个变体。其以决策树为基学习器，在决策树的训练过程中引入了随机属性选择。与 Bagging 相比，其基学习器的多样性不仅来自样本扰动，还来自属性扰动，最终集成的泛化性能进一步提升。

Adaboost：Adaboost 是一种 Boosting 方法，其核心思想有两点：a.提高前一轮训练中错误分类的的样本的权重，而降低哪些被正确分类样本的权重 b.加权多数表决的方法，加大分类误差率小的弱分类器的权重，减少分类误差率大的弱分类器的权重 . 最终得到一个多个弱学习器的加权组合。

GBDT： Boosting Tree 是以分类树或回归树为基本分类器的提升方法，GBDT 是 Boosting Tree 的 一种改进，利用损失函数的负梯度作为残差的近似值，解决了 Boosting Tree 对一般损失函数优化困难的问题。

5、GBDT/xgboost 每次获得的新一轮决策树模型，通常都要乘以系数，为什么

系数是学习率，主要是为了削弱每棵树的影响，让后面有更大的学习空间。

6、xgboost 和 GBDT 有什么区别

a. GBDT 以 CART 作为基分类器，xgboost 还支持线性分类器.

b. GBDT 在优化时只用到一阶导数信息，xgboost 则对损失函数进行了二阶泰勒展开，同时用到了一阶和二阶导数。（xgboost 支持自定义一阶和二阶导数，即自定义损失函数）.

c. xgboost 在代价函数里加入了正则项(对每棵树的叶子节点个数和权重都做了惩罚)，用于控制模型的复杂度。

d.Shrinkage（缩减），相当于学习速率（xgboost 中的 eta）。主要是为了削弱每棵树的影响，让后面有更大的学习空间.

e.列抽样（column subsampling）。xgboost 借鉴了随机森林的做法，支持列抽样，不仅能降低过拟合，还能减少计算.

f.对于特征的值有缺失的样本，xgboost 可以自动学习出它的分裂方向。

g.可并行的近似直方图算法。树节点在进行分裂时，我们需要计算每个特征的每个分割点对应的增益，即用贪心法枚举所有可能的分割点。当数据无法一次载入内存或者在分布式情况下，贪心算法效率就会变得很低，所以 xgboost 还提出了一种可并行的近似直方图算法，用于高效地生成候选的分割点。

h.xgboost 工具支持并行。

7、xgboost 为了控制过拟合做了什么优化

a.在损失函数里加入了正则项。正则项里包含了树的叶子节点个数、每个叶子节点上输出的 score 的 L2 模的平方和。

b.增加 Shrinkage（缩减），相当于学习速率（xgboost 中的 eta）。主要是为了削弱每棵树的影响，让后面有更大的学习空间.

c.列抽样（column subsampling)

8、xgboost 的并行化是如何做的

xgboost 的并行化的实现是通过决策树分裂过程中分裂结点特征的并行计算实现的，即并行每个特征在同层结点上的增益计算过程。

xgboost 在实现上支持并行化，这里的并行化并不是类似于 rf 那样树与树之间的并行化，xgboost 同 boosting 方法一样，在树的粒度上是串行的，但是在构建树的过程中，也就是在分裂节点的时候支持并行化，比如同时计算多个属性的多个取值作为分裂特征及其值，然后选择收益最大的特征及其取值对节点分裂。

9、xgboost 的树生长时的精确分裂与近似分裂分别是怎么做的？

精确分裂也叫作贪心分裂，是遍历所有特征中可能的分裂点位置。

当数据量非常大难以被全部加载进内存时或者分布式环境下时，贪心算法将不再合适。近似分裂通过特征的分布，按照分位数确定一组候选分裂点，通过遍历所有的候选分裂点来找到最佳分裂点。分位数的确定考虑了样本点在当前特征下的权重，权重值取损失函数的二阶导数在该样本点下的值。

分裂算法有两种，一种是精确的分裂，一种是近似分裂算法，精确分裂算法就是把每个属性的每个取值都当作一次阈值进行遍历，采用的决策树是 CART。近似分裂算法是对每个属性的所有取值进行分桶，按照各个桶之间的值作为划分阈值，xgboost 提出了一个特殊的分桶策略，一般的分桶策略是每个样本的权重都是相同 的，但是 xgboost 使每个样本的权重为损失函数在该样本点的二阶导

10、xgboost 有什么缺点(level-wise 和预排序等角度)

a.level-wise 建树方式对当前层的所有叶子节点一视同仁，有些叶子节点分裂收益非常小，对结果没影响，但还是要分裂，加重了计算代价。

b.预排序方法空间消耗比较大，不仅要保存特征值，也要保存特征的排序索引，同时时间消耗也大，在遍历每个分裂点时都要计算分裂增益(不过这个缺点可以被近似算法所克服)

11、xgboost 和 LightGBM 有什么不同

(1).lightgbm 比 xgboost 的优点

更快的训练速度

更低的内存消耗

更好的准确率

分布式支持，可以快速处理海量数据

LightGBM 提出的主要原因是为了解决 GBDT 在处理海量数据遇到的问题，让 GBDT 可以更好更快地用于工业实践。

(2).Histogram 算法

直方图算法的基本思想是先把连续特征值离散化为 k 个整数，同时构造一个宽度为 k 的直方图。在遍历数据的时候，根据离散化后的值作为索引在直方图累

积统计量，当遍历一次数据后，直方图累积了需要的统计量。然后根据直方图的离散值，遍历寻找最优的分割点。

原始的 xgboost 需要存储特征值与特征值预排序后的索引，而 lightgbm 只需要保存特征离散化后的值。这个值一般用 8 位整型存储就足够了。内存降低。

在分裂节点的时间复杂度方面，每次 xgboost 需要遍历所有数据，时间复杂度为 O(#data#features)，lightgbm 为 O(k#features)

(3).带深度限制的 leaf-wise 的叶子生长策略

xgboost 采用 level-wise 叶子生长策略，其过一次数据对每一层所有节点做无差别分裂。但是有些节点的分裂增益非常小，对结果影响不大，会带来不必要的开销。

lightgbm 会从当前所有叶子节点中找出分裂增益最大的叶子节点进行分裂，如此循环。与 level-wise 相比，在分裂次数相同的情况下，leaf-wise 可以降低更多的误差，获得更好的精度。其缺点是，可能会生成比较深的决策树，产生过拟合。因此要对 lightgbm 的生成策略中限定最大深度 max_depth

(4).直方图差加速

通常构造叶子节点的直方图需要遍历一遍叶子节点数据。但是其直方图也可以通过父节点的直方图与他兄弟节点的直方图做差得到。可以提升效率。

(5).直接支持类别特征

xgboost 对于类别特征并不是直接支持的，需要进行 one-hot 编码等操作。基于 one-hot 构造的树，很有可能是不平衡的并且需要很深才可以达到足够的精度。

lightgbm 直接将类别特征分为两堆，共有 2^(k-1)-1 种可能。而且 lightgbm 采取了优化算法，时间复杂度只需要 O(k * log(k))的时间复杂度。

(6).网络通讯优化

在并行学习时，lightgbm 采用集体通讯策略（collective communication algorithm）,而不是点对点的通讯策略。

(7).并行方面优化

XGBoost 的并行不是树粒度的而是特征粒度的，随机森林就是树粒度的并行。

寻找分裂点的时候，算法中先是遍历所有特征再遍历每个特征下的所有值。

遍历特征下所有值时要求值是排序好的，这样就可以使用差加速。

如果不排序，那么计算分类时候的损失函数减少量就没法达到 O(1)的复杂度，因为二叉树的分裂是> x,分到 a 子树这样的形式。

在建树的过程中，最耗时是找最优的切分点，而这个过程中，最耗时的部分是将数据排序。为了减少排序的时间，提出 Block 结构存储数据。

Block 中的数据以稀疏格式 CSC 进行存储

Block 中的特征进行排序（不对缺失值排序，排序只有一次）

Block 中特征还需存储指向样本的索引，这样才能根据特征的值来取梯度。

一个 Block 中存储一个或多个特征的值

(8).Feature Parallel 特征并行

原始的特征并行步骤

a. 垂直方向分割数据集，不同的机器使用不同的特征集，也是对应的不同的数据集

b. 每一个 worker 找到当前 feature set 的最优分割点

c. worker 间沟通找到全局最优分割点

d. 有全局最优分割点的 worker 执行分裂，并将 split 结果发送给其他 worker

e. 其他 worker 执行 split

这种并行方式不仅要交换最优分割点还要交换分割结果。在 data 规模很大时，feature 并行并不能加速。

lightgbm 特征并行策略

a. 不同 worker 均保留所有的数据集，但是使用不同的特征集

b. worker 在当前 feature set 找到局部最优分割点

c. worker 间交流找到全局唯有分割点，并且因为每一个 worker 都有全部数据集，所以可以直接执行 split

(9).Data Parallel 数据并行

数据并行则是让不同的机器先在本地构造直方图，然后进行全局的合并，最后在合并的直方图上面寻找最优分割点。

基于投票的数据并行

在直方图合并的时候，通信代价比较大，基于投票的数据并行能够很好的解

决这一点（笔者建议在设置参数时候，尽量设置为数据并行，不要设置投票并行，因为投票并行可能产生局部最优分割点。这是因为投票并行也是基于数据并行，而每个 worker 只统计了局部数据的最优分割点，可能在投票的过程中这个局部数据的最优分割点不能代表全局最优分割点）。

12、level-wise 的生长和 leaf-wise 的生长有什么不同，优缺点是什么？

Level-wise 过一次数据可以同时分裂同一层的叶子，容易进行多线程优化，也好控制模型复杂度，不容易过拟合。但实际上 Level-wise 是一种低效的算法，因为它不加区分的对待同一层的叶子，带来了很多没必要的开销，因为实际上很多叶子的分裂增益较低，没必要进行搜索和分裂。

Leaf-wise 则是一种更为高效的策略，每次从当前所有叶子中，找到分裂增益最大的一个叶子，然后分裂，如此循环。因此同 Level-wise 相比，在分裂次数相同的情况下，Leaf-wise 可以降低更多的误差，得到更好的精度。Leaf-wise 的缺点是可能会长出比较深的决策树，产生过拟合。因此 LightGBM 在 Leaf-wise 之上增加了一个最大深度的限制，在保证高效率的同时防止过拟合。

13、lightGBM 有哪些优化点

a. 更快的训练速度

b. 更低的内存消耗

c. 更好的准确率

d.分布式支持，可以快速处理海量数据

14、能解释一下 LightGBM 里基于 histogram 的决策树算法吗?

直方图算法的基本思想是先把连续的浮点特征值离散化成 k 个整数，同时构造一个宽度为 k 的直方图。在遍历数据的时候，根据离散化后的值作为索引在直方图中累积统计量，当遍历一次数据后，直方图累积了需要的统计量，然后根据直方图的离散值，遍历寻找最优的分割点。

15、实际上 xgboost 的近似直方图算法也类似于 lightgbm 这里的直方图算法，

为什么 xgboost 的近似算法比 lightgbm 还是慢很多呢？

xgboost 在每一层都动态构建直方图， 因为 xgboost 的直方图算法不是针对某个特定的 feature，而是所有 feature 共享一个直方图(每个样本的权重是二阶导)，所以每一层都要重新构建直方图，而 lightgbm 中对每个特征都有一个直方图，所以构建一次直方图就够了。

lightgbm 有一些工程上的 cache 优化

16、LightGBM 对类别型是怎么处理的?

直接支持类别特征(Categorical Feature)，不需要进行 0/1 展开。在对类别特征计算分割增益的时候，不是按照数值特征那样由一个阈值进行切分，而是直接把其中一个类别当成一类，其他的类别当成另一类。这实际上与 0/1 展开的效果是一样的。

在对离散特征分裂时，每个取值都当作一个桶，分裂时的增益算的是"是否属于某个 category"的 gain。类似于 one-hot 编码。

17、lightgbm 哪些方面做了并行?

目前支持特征并行和数据并行的两种。特征并行的主要思想是在不同机器在不同的特征集合上分别寻找最优的分割点，然后在机器间同步最优的分割点。数据并行则是让不同的机器先在本地构造直方图，然后进行全局的合并，最后在合并的直方图上面寻找最优分割点。

feature parallel

一般的 feature parallel 就是对数据做垂直分割（partiion data vertically，就是对属性分割），然后将分割后的数据分散到各个 workder 上，各个 workers 计算其拥有的数据的 best splits point, 之后再汇总得到全局最优分割点。但是 lightgbm 说这种方法通讯开销比较大，lightgbm 的做法是每个 worker 都拥有所有数据，再分割

data parallel

传统的 data parallel 是将对数据集进行划分，也叫 平行分割(partion data horizontally)， 分散到各个 workers 上之后，workers 对得到的数据做直方图，汇

总各个 workers 的直方图得到全局的直方图。 lightgbm 也 claim 这个操作的通讯开销较大，lightgbm 的做法是使用”Reduce Scatter“机制，不汇总所有直方图，只汇总不同 worker 的不同 feature 的直方图(原理？)，在这个汇总的直方图上做 split，最后同步。

18、xgboost 和 LightGBM 有哪些控制过拟合的手段，通常需要调整的参数有哪些?

设置较少的直方图数目，`max_bin`

设置较小的叶节点数`num_leaves`

设置参数`min_data_in_leaf`和`min_sum__hessian_in_leaf`

使用 bagging 进行行采样`bagging_fraction`和`bagging_freq`

`feature_fraction`,列采样

控制树深`max_depth`

正则化`lambda_l1` `lambda_l2`

切分的最小收益`min_gain_to_split`

19、xgboost 对于缺失值，训练和预测的时候都是怎么处理的？

xgboost 将缺失值看作一类数据，实际处理时，可以将缺失值设置成 missing=-999 或 missing=-9999。

在寻找 split point 的时候，不会对该特征为 missing 的样本进行遍历统计，只对该列特征值为 non-missing 的样本上对应的特征值进行遍历，通过这个技巧来减少了为稀疏离散特征寻找 split point 的时间开销。

在逻辑实现上，为了保证完备性，会分别处理将 missing 该特征值的样本分配到左叶子结点和右叶子结点的两种情形，计算增益后选择增益大的方向进行分裂即可。

如果在训练中没有缺失值而在预测中出现缺失，那么会自动将缺失值的划分方向放到右子树。

20、lgb,xgb 区别，为什么 lgb 快

选取特征最佳分割点时，Lgb 使用的是 histogram 算法，这种只需要将数据分割成不同的段即可，不需要进行预先的排序。占用的内存更低，数据分割的复杂度更低。

Xgb 采用的是 Level-wise 的树生长策略，Lgb 采用的是 leaf-wise 的生长策略，以最大信息增益为导向。后者进度更高，容易过拟合，所以要控制最大深度。

Lgb 优化了对类别特征的支持，可以直接输入类别特征，不需要额外的 0/1 展开，并在决策树算法上增加了类别特征的决策规则。

Lgb 做了很多的并行优化策略。

21、lgb,xgb 如何防止过拟合，有哪些参数？

1、损失函数中加入了正则化项，相当于预剪枝

2、shrinkage

即在迭代中为树的叶子结点乘以一个权重衰减稀疏，以削弱每棵树的影响力，为后面的树留下提升空间

3、列采样，即特征采样。

有按层采样和建树之前随机采样两种方式。

其中按层采样是在同一层的结点进行分裂之前随机选择部分特征，对这些部分特征进行遍历，寻找最优切分点，而不用遍历全部特征。

建树之前随机选择特征是在建树之前就选择部分特征，在之后的结点的分裂中，只使用这部分特征

4、行采样，也可以理解成样本采样。

利用的是 bagging 思想，训练时选取部分样本进行训练，增加了树的多样性

参数有 max_depth,learning_rate,n_estimators,min_child_weight,等

22、lgb 二分类的损失函数是什么？

交叉熵

23、（百度）Adaboost 原理

AdaBoost 是英文"Adaptive Boosting"（自适应增强）的缩写，它的自适应在

于：前一个基本分类器被错误分类的样本的权值会增大，而正确分类的样本的权值会减小，并再次用来训练下一个基本分类器。同时，在每一轮迭代中，加入一个新的弱分类器，直到达到某个预定的足够小的错误率或达到预先指定的最大迭代次数才确定最终的强分类器。

Adaboost 算法可以简述为三个步骤：

（1）首先，是初始化训练数据的权值分布 D1。假设有 N 个训练样本数据，则每一个训练样本最开始时，都被赋予相同的权值：w1=1/N。

（2）然后，训练弱分类器 hi。具体训练过程中是：如果某个训练样本点，被弱分类器 hi 准确地分类，那么在构造下一个训练集中，它对应的权值要减小；相反，如果某个训练样本点被错误分类，那么它的权值就应该增大。权值更新过的样本集被用于训练下一个分类器，整个训练过程如此迭代地进行下去。

（3）最后，将各个训练得到的弱分类器组合成一个强分类器。各个弱分类器的训练过程结束后，加大分类误差率小的弱分类器的权重，使其在最终的分类函数中起着较大的决定作用，而降低分类误差率大的弱分类器的权重，使其在最终的分类函数中起着较小的决定作用。

换而言之，误差率低的弱分类器在最终分类器中占的权重较大，否则较小。

23、（欢聚时代 YY）gbdt 的负梯度在算什么，为什么 xgb 加了二阶梯度会更好

Xgboost 官网上有说，当目标函数是 MSE 时，展开是一阶项（残差）+二阶项的形式（官网说这是一个 nice form），而其他目标函数，如 logloss 的展开式就没有这样的形式。为了能有个统一的形式，所以采用泰勒展开来得到二阶项，这样就能把 MSE 推导的那套直接复用到其他自定义损失函数上。简短来说，就是为了统一损失函数求导的形式以支持自定义损失函数。这是从为什么会想到引入泰勒二阶的角度来说的

二阶信息本身就能让梯度收敛更快更准确。这一点在优化算法里的牛顿法里已经证实了。可以简单认为一阶导指引梯度方向，二阶导指引梯度方向如何变化。这是从二阶导本身的性质，也就是为什么要用泰勒二阶展开的角度来说的

24、（欢聚时代 YY）lgb 对比 xgb 和原始 gbdt 的优缺点是什么

1.Lgb 使用的是 histogram 算法，这种只需要将数据分割成不同的段即可，不需要进行预先的排序。占用的内存更低，数据分割的复杂度更低。

2.Xgb 采用的是 Level-wise 的树生长策略，Lgb 采用的是 leaf-wise 的生长策略，以最大信息增益为导向。后者进度更高，容易过拟合，所以要控制最大深度。

3.Lgb 优化了对类别特征的支持，可以直接输入类别特征，不需要额外的 0/1 展开，并在决策树算法上增加了类别特征的决策规则。

4.Lgb 做了很多的并行优化策略。

①.特征并行，每个 worker 留有一份完整的数据集，但是每个 worker 仅在特征子集上进行最佳切分点的寻找；worker 之间需要相互通信，通过比对损失来确定最佳切分点；然后将这个最佳切分点的位置进行全局广播，每个 worker 进行切分即可。

②.数据并行，当数据量很大，特征相对较少时，可采用数据并行策略。对数据水平切分，每个 worker 上的数据先建立起局部的直方图，然后合并成全局的直方图，采用直方图相减的方式，先计算样本量少的节点的样本索引，然后直接相减得到另一子节点的样本索引，这个直方图算法使得 worker 间的通信成本降低一倍，因为只用通信比此样本量少的节点。

③.投票并行，当数据量和维度都很大时，选用投票并行，该方法是数据并行的一个改进。数据并行中的合并直方图的代价相对较大，尤其是当特征维度很大时。

大致思想是：每个 worker 首先会找到本地的一些优秀的特征，然后进行全局投票，根据投票结果，选择 top 的特征进行直方图的合并，再寻求全局的最优分割点。

25、RF 与 boosting 之间差别？GBDT 和 xgboost 之间差别？

RF 是一种以决策树为基模型的 bagging 算法，RF 通过对样本以及特征采样构建数据集，再利用DT 进行集成，是一种天然的并行模型。

boosting 是一种串行集成方法，主要的思想是通过每一次增加一个基学习器来减少集成模型与预测值之间的误差，代表的模型包括 Adaboost,GBDT 以及

xgboost，Adaboost 是通过对样本点的权值调整(错误样本权值增大，正确样本权值降低)达到训练多个模型的目的。

GBDT 与 xgboost 主要区别分为以下几点：

①GBDT 优化方法使用了一阶导数 xgboost 使用了二阶导数(牛顿法)优化

②xgboost 模型本省增加了正则化项，包括叶子节点的二范数以及树结构的惩罚项

③xgboost 本身有一些并行化的操作，对特征进行分 bin 求优再汇总求优

④xgboost 使用了 shrinkage方法，降低每次基学习器的拟合效果，防止过拟合

⑤xgboost支持特征抽样

⑥xgboost 可对特征有缺失的样本自动处理

⑦可以对特征预排序进而使得特征选择的时候可以并行化价快速度

⑧除了 CART 还支持线性分类器

26、xgboost 有哪些参数会影响模型的复杂度？影响是怎样的？

eta，学习率，小一点不容易过拟合(shrinkage)

max_depth，树深度，越大越容易过拟合

min_child_weight，越大越容易欠拟合

max_leaf_nodes，越大越容易过拟合

gamma，分裂损失最小值，越大越容易欠拟合

alpha，L1 正则化系数，越大越容易欠拟合

lambda，L2 正则化系数，越大越容易欠拟合

27、 xgboost 的并行化体现在哪？xgboost 的多分类如何做？xgboost 的近似算法怎么做的？

并行化体现在特征并行化。

xgboost 多分类使用one vs rest方法

近似算法体现在特征值的选取上，使用了近似直方图算法生成候选的分割点

28、xgboost 用的哪个基分类器

XGBoost 目前支持三种基分类器，分别是 gbtree，gblinear 和 dart，可以使用 booster 参数修改。其中，gbtree 和 dart 是基于树的分类器，dart 主要多了 Dropout，而 gblinear 是线性分类器。一般情况下 gbtree 都足够了。

29、bagging,boosting 区别

样本选择上: Bagging 采取 Bootstraping 的是随机有放回的取样，Boosting 的每一轮训练的样本是固定的，改变的是买个样的权重。

样本权重上：Bagging 采取的是均匀取样，且每个样本的权重相同，Boosting 根据错误率调整样本权重，错误率越大的样本权重会变大

预测函数上：Bagging 所以的预测函数权值相同，Boosting 中误差越小的预测函数其权值越大。

并行计算: Bagging 的各个预测函数可以并行生成;Boosting 的各个预测函数必须按照顺序迭代生成。

朴素贝叶斯：

1、写一下贝叶斯公式?

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

$P(A|B)$：A事件的后验概率

$P(A)$：A事件的先验概率

$P(B)$：B事件的先验概率

$P(B|A)$：B事件的后验概率

2、朴素贝叶斯是一个什么算法，能解释一下吗?

朴素贝叶斯算法是基于贝叶斯公式的，通过计算各个类别的后验概率来判断分类，后验概率最大的类别是分类结果。贝叶斯公式+条件独立+平滑，常用的平滑是拉普拉斯平滑，在 NLP 中词频统计的时候伯努利模型和多项式模型的平

滑方式不同。贝叶斯公式应用于分类问题可用下式表示：

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

c：类别

x：特征

由于 P(x|c)是所有特征的联合概率，难以从训练样本直接估计，所以采用“条件独立性假设”，假设所有特征相互独立。则：

$$P(c|x) = \frac{P(x|c) * P(c)}{P(x)} = \frac{P(c)}{P(x)} \prod_{i=1}^{d} P(x_i|c)$$

d：特征数目x_i：第i个特征上的取值

由于对所有类别来说 P(x)相同，则

$$P(c|x) \propto P(c) \prod_{i=1}^{d} P(x_i|c)$$

当样本数充足时，可容易估计出 P(c)和 P(x_i|c)，进而得到 P(c|x)。

3、如果出现计数为 0 导致概率为 0，这种情况在朴素贝叶斯计算里是怎么解决的?

采用拉普拉斯平滑进行修正：

$$\hat{P}(c) = \frac{|D_c| + 1}{|D| + N}$$

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

N：训练集D中可能的类别数

N_i：第i个特征可能的取值数

生成模型&判别模型：

1、(网易游戏)生成模型，判别模型的区别

生成模型

举例：HMM Bayes LDA

计算思路： 通过学习全部样本的先验和条件概率分布，求出后验概率。

特征：可得到联合概率分布 P(XY)

判别模型

举例：SVM 感知器 knn 决策树 LR CRF

计算思路： 直接学习 P(Y|X) 或者 学习决策边界

特征：不可学的到联合概率分布 P(XY)

直接对比

生成式模型举例：利用生成模型是根据山羊的特征首先学习出一个山羊的模型，然后根据绵羊的特征学习出一个绵羊的模型，然后从这只羊中提取特征，放到山羊模型中看概率是多少，在放到绵羊模型中看概率是多少，哪个大就是哪个。判别式模型举例：要确定一个羊是山羊还是绵羊，用判别模型的方法是从历史数据中学习到模型，然后通过提取这只羊的特征来预测出这只羊是山羊的概率，是绵羊的概率。

只有生成模型能检测异常值。由于生成模型完全学习了所有的分布，所以它可以用来检测某个值是否异常：P(X)是否太小

生成模型的处理过程会告诉你关于数据的一些统计信息（p(x|y) 分布

判定模型，就是只有一个模型，你把测试用例往里面一丢，label 就出来了，如 SVM。生成模型，有多个模型（一般有多少类就有多少个），你得把测试用例分别丢到各个模型里面，最后比较其结果，选择最优的作为 label，如朴素贝叶斯

2、(网易游戏)有哪些是生成模型，哪些是判别模型

生成模型

举例：HMM Bayes LDA

计算思路： 通过学习全部样本的先验和条件概率分布，求出后验概率。

特征：可得到联合概率分布 P(XY)

判别模型

举例：SVM 感知器 knn 决策树 LR CRF

计算思路： 直接学习 P(Y|X) 或者 学习决策边界

特征：不可学的到联合概率分布 P(XY)

EM：

1、（网易游戏）Em 算法

EM(Expectation maximization)算法，也即期望最大化算法，作为“隐变量”（属性变量不可知）估计的利器在自然语言处理（如 HMM 中的 Baum-Welch 算法）、高斯混合聚类、心理学、定量遗传学等含有隐变量的概率模型参数极大似然估计中有着十分广泛的应用。EM 算法于 1977 年由 Arthur Dempster, Nan Laird 和 Donald Rubin 总结提出，其主要通过 E 步（exceptation），M 步（maximization）反复迭代直至似然函数收敛至局部最优解。由于其方法简洁、操作有效，EM 算法曾入选“数据挖掘十大算法”，可谓是机器学习经典算法之一。

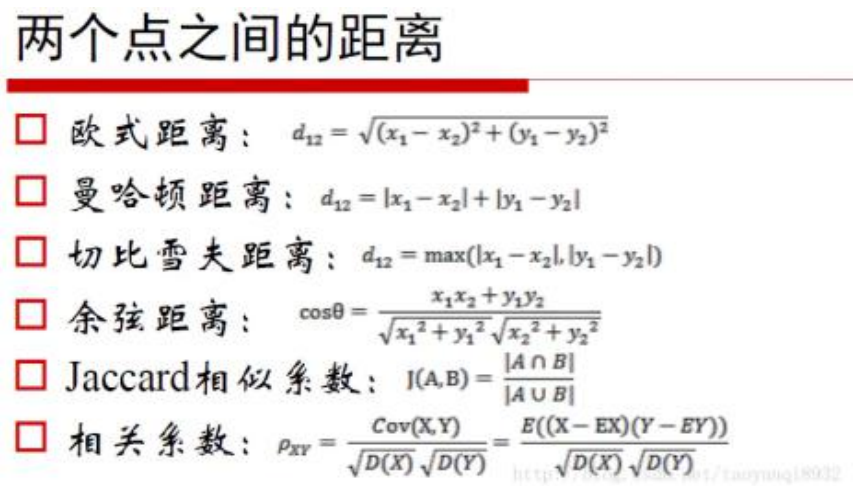
聚类算法：

1、（网易游戏）kmeans 算法介绍下，kmeans 跟 em 的联系，怎么自动确定 k 值。用什么统计量衡量簇的好坏。优缺点

K 值：要得到的簇的个数

质心：每个簇的均值向量，即向量各维取平均即可

距离量度：常用欧几里得距离和余弦相似度（先标准化）



算法流程：

(1)、首先确定一个 k 值，即我们希望将数据集经过聚类得到 k 个集合。

(2)、从数据集中随机选择 k 个数据点作为质心。

(3)、对数据集中每一个点，计算其与每一个质心的距离（如欧式距离），离哪个质心近，就划分到那个质心所属的集合。

(4)、把所有数据归好集合后，一共有 k 个集合。然后重新计算每个集合的质心。

(5)、如果新计算出来的质心和原来的质心之间的距离小于某一个设置的阈值（表示重新计算的质心的位置变化不大，趋于稳定，或者说收敛），我们可以认为聚类已经达到期望的结果，算法终止。

(6)、如果新质心和原质心距离变化很大，需要迭代 3~5 步骤。

如果用数据表达式表示，假设簇划分为$(C_1, C_2, \dots C_k)$，则我们的目标是最小化平方误差E：

$$E=\sum_{i=1}^{k}\sum_{x\in C_i}||x-\mu_i||_2^2$$

其中μ_i是簇C_i的均值向量，有时也称为质心，表达式为：

$$\mu_i=\frac{1}{|C_i|}\sum_{x\in C_i}x$$

如果我们想直接求上式的最小值并不容易，这是一个NP难的问题，因此只能采用启发式的迭代方法。

算法优点：

①算法简单

②只需要调 k

算法缺点

①结果受初值影响不稳定，解决方法多次初始取最优，kmeans++

②受异常点影响大，可以使用k-medias 算法

③数据如果不是凸集效果较差，因为假设数据服从正态分布，是 GMM 模型的特殊形式。

2、（百度）聚类的方法都有什么

聚类分析的算法可以分为划分法、层次法、基于密度的方法、基于网格 bai 的方法、基于模型的方法。

(1)、划分法，给定一个有 N 个元组或者纪录的数据集，分裂法将构造 K 个分组，每一个分组就代表一个聚类，K<N。

(2)、层次法，这种方法对给定的数据集进行层次似的分解，直到某种条件满足为止。

(3)、基于密度的方法，基于密度的方法与其它方法的一个根本区别是：它不是基于各种各样的距离的，而是基于密度的。这样就能克服基于距离的算法只能发现“类圆形”的聚类的缺点。

(4)、图论聚类方法解决的第一步是建立与问题相适应的图，图的节点对应于被分析数据的最小单元，图的边（或弧）对应于最小处理单元数据之间的相似性度量。

(5)、基于网格的方法，这种方法首先将数据空间划分成为有限个单元的网格结构,所有的处理都是以单个的单元为对象的。

(6)、基于模型的方法，基于模型的方法给每一个聚类假定一个模型，然后去寻找能够很好的满足这个模型的数据集。

3、（百度）Kmeans 的流程方法停止条件

流程：

（1）K 如何确定

（2）初始质心的选取

（3）距离的度量

（4）质心的计算

（5）算法停止条件

（6）空聚类的处理

停止条件：

目标函数达到最优，对于不同的距离度量，目标函数往往不同。我们往往认为簇的质心到各个点的距离越小，簇越紧凑。

采用欧式距离时：目标函数一般为最小化对象到其簇质心的距离的平方和。

采用余弦相似度时，目标函数一般为最大化对象到其质心的余弦相似度和。

4、 聚类算法 K-means 与 gmm 的差异与用途

在 GMM 中加入下面的三个条件即可转换成 k-means，故有人称 GMM 为 soft-EM,k-means 为 hard-EM(本来 GMM 的隐变量 z(向量形式，每个分量为 0 或 1，可以有多个分量为 1)为 soft 形式，可能属于多个类，在 k-means 的情况下只能属于一个类(z 中只能有一个 1))

1. 权重均一致，即每个高斯分布等概率地产生样本， $\alpha_j = \frac{1}{k}$ ；

2. 协方差矩阵 Σ_j 为单位矩阵；

3. 我们认为样本归属于产生概率最大的那一个高斯分布，即 $\tilde{z}_{ij} = P(z_{ij}|\mathbf{x}_i, \theta^m) = 1$ ，意味着 $\tilde{z}_{ij} = 0/1$ ；

降维算法：

1、（网易游戏）PCA 算法原理，跟 svd 的区别。特征值越大代表什么？ 特征向量代表什么。特征向量有什么性质，是单位向量吗？

PCA 属于一种线性、非监督、全局的降维算法，旨在找到数据的主成分，并利用这些主成分表征原始数据，从而达到降维的目的。

算法步骤

样本归一化

求解协方差矩阵

求解特征值和特征向量

选择主要成分

转换特征降维的数据

降维的优化目标：将一组 N 维向量降为 K 维（K 大于 0，小于 N），其目标是选择 K 个单位（模为 1）正交基，使得原始数据变换到这组基上后，各字段两两间协方差为 0，而字段的方差则尽可能大（在正交的约束下，取最大的 K 个方差）

PCA 的两面性

优点是完全无参数限制的。在 PCA 的计算过程中完全不需要人为的设定参数或是根据任何经验模型对计算进行干预，最后的结果只与数据相关，与用户是独立的。

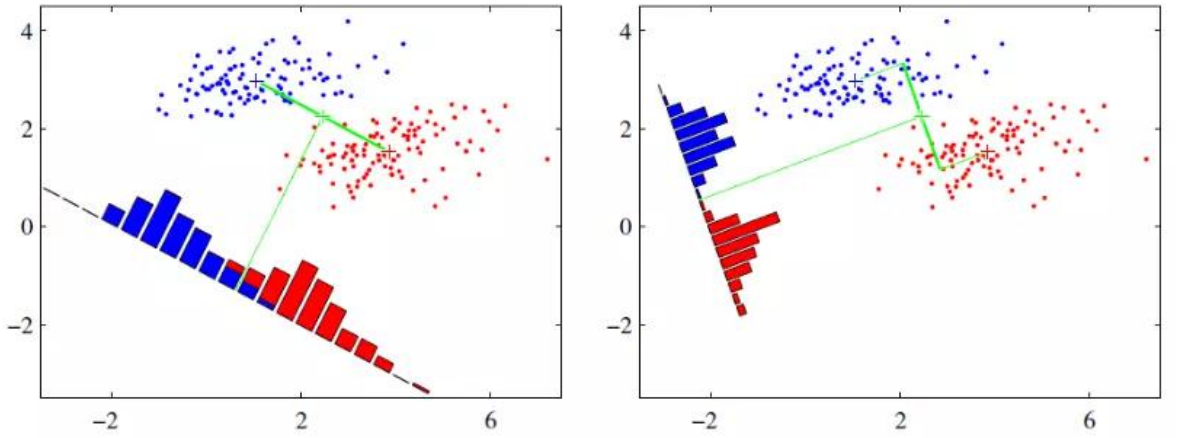
上边同时也可以看作是缺点。如果用户对观测对象有一定的先验知识，掌握了数据的一些特征，却无法通过参数化等方法对处理过程进行干预，可能会得不到预期的效果，效率也不高。

2、（网易游戏）知道 LDA 吗

LDA 的核心思想： 类内小，类间大

线性判别分析（Linear Discriminant Analysis, 以下简称 LDA）是一种监督学习的降维技术，也就是说它的数据集的每个样本是有类别输出的。这点和 PCA 不同，PCA 是不考虑样本类别输出的无监督降维技术。LDA 的思想可以用一句话概述，就是“投影后类内方差最小，类间方差最大”，什么意思呢？我们要将数据在低维度上进行投影， 投影后希望每一种类别数据的投影点尽可能的接近，而不同类别的数据的类别中心之间的距离尽可能的大。

可能这句话有点抽象，那我们先看看最简单的情况。假设我们有两类数据分别是红色和蓝色，如下图所示，这些数据特征是二维的，我们希望将这些数据投影到一维的一条直线，让每一种类别数据的投影点尽可能的接近，而红色和蓝色数据中心之间的距离尽可能的大。



特征工程：

1、特征工程有什么作用？

数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限。通过特征过程获得重要和有效的特征，机器学习才能有更好的效果。

2、请简述有哪些你知道的特征工程和他们的操作？

特征工程包括三方面：

a. 数据获取及清洗

数据需考虑数据对任务是否有帮助及数据是否能够被采集。数据清洗主要是去除脏数据，需结合业务对数据进行过滤，例如去除爬虫、作弊等数据，同时对数据异常点进行检测。

b.特征处理

特征处理主要是通过以下一些方法：

特征处理

缺失值补充

幅度调整/归一化

离散化

one-hot/哑编码

Hash 与聚类

统计计算

数据变换（多项式、指数、对数）

特征组合

c.特征选择

特征选择主要考虑两方面：

c1.特征是否发散：方差大小

c2.特征与目标的相关性：与目标相关性高的特征，应当优先选择。

3、有哪些特征选择的方法？

特征选择方法有三类：过滤式、包裹式和嵌入式。

过滤式：先对数据集进行特征选择，再训练学习器，特征选择过程与后续学习器无关。主要有 方差选择法、相关系数法、卡方检验、互信息法。

包裹式：给学习器选择最有利于其性能的特征子集。常用 RFE 算法。

嵌入式：特征选择过程与学习器训练过程融为一体，两者在同一个优化过程中完成。通常使用 L1 正则化。

4、（美团）样本不均衡怎么处理

(1)、收集更多的数据

(2)、改变模型评估方法

(3)、数据集重采样

(4)、合成样本

(5)、尝试不同的算法

(6)、使用罚分模型

(7)、尝试以更多不同的视角来看待样本不均衡问题

其实，样本不均衡就是不同类别的数据不同使得模型学习时候偏向各个类别的权重不同，而我们要做的，其实就是如何均衡各个类别的权重，无论是上采样，下采样，抑或是更改 loss，给数据量少的类别的 loss 给多权重，更直接的，在某些库的分类器中我们可以看到给某些样本直接赋予权重，这些虽然看起来不同，

但是其实都是为了均衡权重这一目的而来的。

5、有哪些异常值检测的方法？

数字异常值(Numeric outlier),使用IQR，适用于 1 维空间，一般 k 取 1.5

Z-score，适用于 1 维或者低维，假定数据服从高斯分布，对数据做归一化，判断异常点方法一般设置为 2.5/3/3.5

DBSCAN，基于 DBSCAN 聚类方法，适用于 1 维多维，基于数据密度的离群值的检验方法，主要将数据点分为核心点，边界点(在核心点ε内但

不是核心点)，除此之外的都是噪声点

Isolation Forest，一维或多维特征空间中大数据集的非参数方法，基于孤立数进行判断数据点是否是异常点，孤立数越小越可能是异常点

6、有哪些处理异常值的方法？

缺失值太多直接删除

缺失值不多，可以考虑填充，例如均值填充(正态分布)，众数或中位数填充(长尾分布)

将缺失值视为一个类别

对于某些模型可以不处理，例如树模型(sklearn 中只有 xgboost自己有处理的方式而已其他没有)

模型优化：

1、常用的防止过拟合的技术手段有哪些，l1-norm 和 l2-norm 的区别是什么？

常见防止过拟合一般有扩增数据集，添加正则化，重采样等； L1 正则化和 L2 正则化可以看做是损失函数的惩罚项。所谓惩罚是指对损失函数中的某些参数做一些限制。对于线性回归模型，使用 L1 正则化的模型叫做 Lasso 回归，使用 L2 正则化的模型叫做 Ridge 回归（岭回归）。

L1 正则化可以使得参数稀疏化，即得到的参数是一个稀疏矩阵，可以用于

特征选择。

稀疏性，说白了就是模型的很多参数是 0。通常机器学习中特征数量很多，例如文本处理时，如果将一个词组（term）作为一个特征，那么特征数量会达到上万个（bigram）。在预测或分类时，那么多特征显然难以选择，但是如果代入这些特征得到的模型是一个稀疏模型，很多参数是 0，表示只有少数特征对这个模型有贡献，绝大部分特征是没有贡献的，即使去掉对模型也没有什么影响，此时我们就可以只关注系数是非零值的特征。这相当于对模型进行了一次特征选择，只留下一些比较重要的特征，提高模型的泛化能力，降低过拟合的可能。

L2 正则化可以防止模型过拟合（overfitting）；一定程度上，L1 也可以防止过拟合。

2、（网易游戏）什么是过拟合，防止过拟合的方法有哪些。

overfiting 指的是在训练集效果极好，在测试集上效果一般，underfiting 指的是在训练集测试集上效果都不好

可以通过模型的评估指标在训练集测试集上的表现，对过拟合欠拟合进行判断

underfiting 缓解方法：

①增加可用特征

②使用更加复杂的模型

③减少正则化系数

overfiting 的缓解方法

①增加数据

②增加正则化系数

③调节模型参数，使之更加简单

④对于树模型可以对叶节点，树深度，节点分类最小值，shrinkage 等进行调节

⑤对数据进行随机采样提高随机性

⑥对于深度学习：可以对数据进行一系列变换平移旋转调节锐化等等增加样本，earlystopping，增加噪声，Dropout，bagging/boosting,贝叶斯方法(说白了就

是平均化/平滑)

3、正则化为什么可以防止过拟合。

L1 会趋向于产生少量的特征，而其他的特征都是 0，而 L2 会选择更多的特征，这些特征都会接近于 0。Lasso 在特征选择时候非常有用，而 Ridge 就只是一种规则化而已。在所有特征中只有少数特征起重要作用的情况下，选择 Lasso 比较合适，因为它能自动选择特征。而如果所有特征中，大部分特征都能起作用，而且起的作用很平均，那么使用 Ridge 也许更合适。

4、（网易游戏）了解凸优化吗

不严格的说，凸优化就是在标准优化问题的范畴内，要求目标函数和约束函数是凸函数的一类优化问题。

5、预测函数与代价函数的关系，在矩阵分解中如何体现的？

预测函数是输入样本 X 得到预测 y_hat 的映射过程。代价函数应该是和损失函数是一样的，是用来度量整个数据集

中真实值与预测值之间的差距，都是模型需要优化的目标，一般来说越小越好。

- 预测函数：

$$M'_{UI} = \sum_{k=1}^{K} P_{U,k} Q_{k,I}$$

- 代价函数：

$$SSE = E^2 = \sum_{U,I} (M_{U,I} - M'_{U,I})^2$$

6、梯度下降中局部最优解和全局最优解的关系？

梯度下降得到的是局部最优解，而不一定是全局最优解。只有凸函数才有全局最优解，使用梯度下降得到的解一般情况下是局部最优解。

凸优化之所以如此重要，是因为：

(1)、其应用非常广泛，机器学习中很多优化问题都要通过凸优化来求解；

(2)、在非凸优化中，凸优化同样起到重要的作用，很多非凸优化问题，可以转化为凸优化问题来解决；

(3)、如上引用所述，凸优化问题可以看作是具有成熟求解方法的问题，而其他优化问题则未必。

凸集，定义目标函数和约束函数的定义域。

凸函数，定义优化相关函数的凸性限制。

凸优化，中心内容的标准描述。

凸优化问题求解，核心内容。相关算法，梯度下降法、牛顿法、内点法等。

对偶问题，将一般优化问题转化为凸优化问题的有效手段，求解凸优化问题的有效方法。

5、如何观察过拟合？

模型在验证集合上和训练集合上表现都很好，而在测试集合上变现很差。过拟合即在训练误差很小,而泛化误差很大,因为模型可能过于的复杂

6、优化器详解

对于一个机器学习问题，在 tensorflow 中，需要先定义一个损失函数，然后优化器做的使就是最小化这个损失函数（梯度下降最常见的三种变形 BGD，SGD，MBGD）

7、各种模型的 feature importance 的度量

不基于模型的

特征房差较大的特征度较大(熵高，信息量大)

与 label 进行相关系数计算，相关系数越大重要度越高

线性模型/LR

根据得到模型的权重绝对值大小对特征重要度进行表征

Naive bayes

可以通过对相同的词在不同类别中的频率之比取 log 作为重要度衡量

RF/GDBT

①每个特征在每棵树上做了多大贡献再取平均值，比较特征之间贡献度，主要使用的是基尼系数，仔细来说是对所有包含此特征节点的树都计算某个特征在某棵树上分叉前后基尼系数的变化，之后进行归一化处理，作为特征重要度的指标

②袋外数据错误率评估，数据自变量值发生轻微的扰动之后正确率与分类前正确率的平均减少量，计算方法类似于基尼系数

xgboost

①weight，某个特征在所有树中用于分裂样本个的数

②gain，某个特征在所有树中涉及到的所有节点的平均增益(指标是 xgboost 推导出的特有的公式)

③cover，这个是通过被分到该节点的样本的二阶导数之和，举个例子来说，某个特征作为节点(假设只有一个该特征节点)的对应分割样本数为 10，那么此样本在这棵树的覆盖度就是 10。说得通俗一点就是该特征所涉及到的观测值的相对数量情况(MSE)。

模型评估方法：

1、（百度）混淆矩阵

混淆矩阵（Confusion Matrix），它的本质远没有它的名字听上去那么拉风。矩阵，可以理解为就是一张表格，混淆矩阵其实就是一张表格而已。

以分类模型中最简单的二分类为例，对于这种问题，我们的模型最终需要判断样本的结果是 0 还是 1，或者说是 positive 还是 negative。

我们通过样本的采集，能够直接知道真实情况下，哪些数据结果是 positive，哪些结果是 negative。同时，我们通过用样本数据跑出分类型模型的结果，也可以知道模型认为这些数据哪些是 positive，哪些是 negative。

因此，我们就能得到这样四个基础指标，我称他们是一级指标（最底层的）：

真实值是 positive，模型认为是 positive 的数量（True Positive=TP）

真实值是 positive，模型认为是 negative 的数量（False Negative=FN）：这就是统计学上的第一类错误（Type I Error）

真实值是 negative，模型认为是 positive 的数量（False Positive=FP）：这就是统计学上的第二类错误（Type II Error）

真实值是 negative，模型认为是 negative 的数量（True Negative=TN）

将这四个指标一起呈现在表格中，就能得到如下这样一个矩阵，我们称它为混淆矩阵。

2、 有哪些模型评估方法？

leave-one-out: 适用于数据量较大

K-Fold: 计算量比较大, 适用于数据量不大的时候

bootstrap: 小样本适用, 但会使得数据分布改变

注意采样的时候需要根据问题进行判断是否需要进行分层采样

3、 有哪些评估准则？

回归

MSE

MAE

R2: 表征模型对于观察值的拟合程度

RMSE

MAPE: MAPE=(∑((X-Y)/X)*100%)/N，X 为实测值，Y 为模拟值，N 为样本总数, 查看比例

二分类

accuracy: 不适用于非均衡数据

ROC: 适用于非均衡数据, 关注样本 rank 值不在意绝对值大小

PR curve: 主要适用于对 precision 或者 recall 敏感的问题

Fß score: 带权重的 PR 的调和平均, ß>1 recall 占权重大, 0<ß<1 precision 占权重大

多分类

micro AUC/F1

macro AUC/F1，主要区别在于平均的粒度层次不同

4、AUC 计算 auc 为什么稳定？

AUC 的全称是 AreaUnderRoc 即 Roc 曲线与坐标轴形成的面积，取值范围 [0, 1].

Roc 空间将伪阳性率(FPR)定义为 X 轴，真阳性率(TPR)定义为 Y 轴。

TPR: 在所有实际为阳性的样本中，被正确地判断为阳性之比率 TPR = TP/P = TP/(TP+FN)

FPR: 在所有实际为阴性的样本中，被错误地判定为阳性之比率 FPR = FP/N = FP/(FP + TN)

UC 最普遍的定义是 ROC 曲线下的面积。但其实另一种定义更常用，分别随机从正负样本集中抽取一个正样本，一个负样本，正样本的预测值大于负样本的概率。

auc 中的两个左边轴分别为正阳率与假阳率。正阳率代表模型判断对的正样本在真实正样本中所占比例，假阳率代表模型判断错的负样本在真实负样本中所占比例。

无论是正阳率还是假阳率，当进行过采样或者欠采样时，这两个值都不会发生改变。所以 auc 稳定。