1分类算法常用哪些？

KNN

LR（逻辑回归）

SVM

决策树、随机森林

神经网络

2 回归算法一般用哪些？

线性回归

决策树、GBDT

XGboost

神经网络

3 回归方法中使用的评价指标是哪些?

均方误差（MSE）：

均方根误差（RMSE）

MAE(平均绝对误差)

R Squared

4 分类中使用的损失函数是哪些？为什么是交叉熵而不是mae？

0-1损失函数（zero-one loss）：在感知机中，使用的该损失函数

对数损失函数（log loss）：用于最大似然估计，等价于交叉熵损失函数

指数损失函数（exponential loss）：在adaboost中使用的该损失函数

合页损失函数（hinge loss）：在SVM中使用的该损失函数

交叉熵损失函数（cross-entropy loss）：用于分类任务

在LR与神经网络中，分类最常使用的是交叉熵\*：

因为平方损失函数权重更新过慢，采用交叉熵损失函数可以完美解决过慢的问题，它具有“误差大的时候，权重更新快；误差小的时候，权重更新慢”的良好性质。

sigmoid作为激活函数的时候，如果采用均方误差损失函数，那么这是一个非凸优化问题，不宜求解，容易陷入局部最优解。而采用交叉熵损失函数依然是一个凸优化问题，更容易优化求解。

5对于小数据集一般怎么处理呢？小数据集中如何防止过拟合？

第一个思路：

小数据集配合神经网络时，必须要考虑过拟合的问题。

最有效的方法应该是扩充数据集：数据集越大，网络泛化性能越好，所以努力扩充数据集，通过平移、翻转、旋转、放缩、随机截取、加噪声、色彩抖动等等方式。

其他防止过拟合的方法：比如l1、l2，dropout、BN、验证集、模型集成

第二个思路：

神经网络需要大数据，所以可以考虑其他机器学习方法。

6常用什么激活函数？

sigmoid

tanh

ReLU

Leaky ReLU

Mish激活函数

[链接](7. https://blog.csdn.net/weixin\_42057852/article/details/84644348?spm=1001.2101.3001.6661.1&utm\_medium=distribute.pc\_relevant\_t0.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1.pc\_relevant\_antiscanv2&depth\_1-utm\_source=distribute.pc\_relevant\_t0.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1.pc\_relevant\_antiscanv2&utm\_relevant\_index=1)

7 sigmoid函数有使用过吗？与relu激活函数有什么不同？

sigmoid型函数是第一个被广泛应用于神经网络的激活函数。经过sigmoid型函数作用后，输出的值范围在[0,1]之间。但是sigmoid型函数的输出存在均值不为0的情况，并且存在梯度消失的问题，在深层网络中被其他激活函数替代。在逻辑回归中使用的该激活函数用于输出分类。

函数不同

sigmoid存在梯度消失的问题；relu型函数可以有效避免梯度消失的问题

8SGD,Momentum,Adagard,Adam原理

SGD为随机梯度下降,每一次迭代计算数据集的mini-batch的梯度,然后对参数进行跟新。

Momentum参考了物理中动量的概念,前几次的梯度也会参与到当前的计算中,但是前几轮的梯度叠加在当前计算中会有一定的衰减。

Adagard在训练的过程中可以自动变更学习的速率,设置一个全局的学习率,而实际的学习率与以往的参数模平方和的开方成反比。

Adam利用梯度的一阶矩估计和二阶矩估计动态调整每个参数的学习率,在经过偏置的校正后,每一次迭代后的学习率都有个确定的范围,使得参数较为平稳。

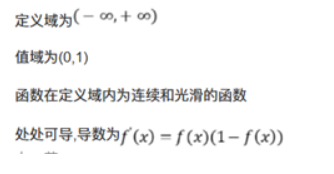
链接

9L1不可导的时候该怎么办

当损失函数不可导,梯度下降不再有效,可以使用坐标轴下降法,梯度下降是沿着当前点的负梯度方向进行参数更新,而坐标轴下降法是沿着坐标轴的方向,假设有m个特征个数,坐标轴下降法进参数更新的时候,先固定m-1个值,然后再求另外一个的局部最优解,从而避免损失函数不可导问题。

使用Proximal Algorithm对L1进行求解,此方法是去优化损失函数上界结果。

10sigmoid函数特性



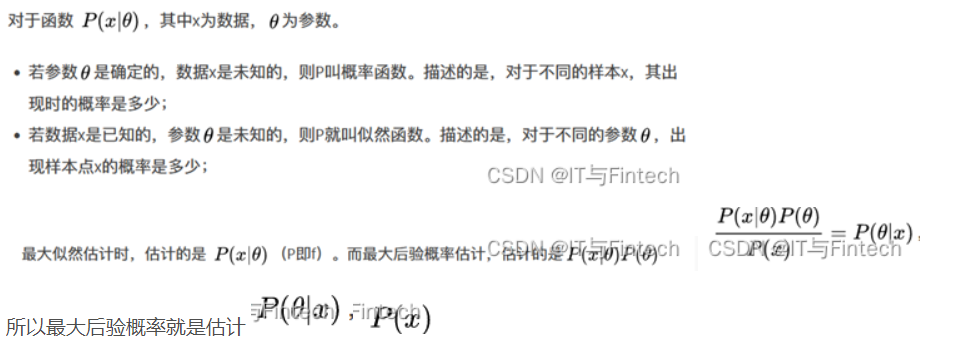
11面试中的概率问题

概率链接

切比雪夫不等式

最大似然估计：

最大后验概率估计：

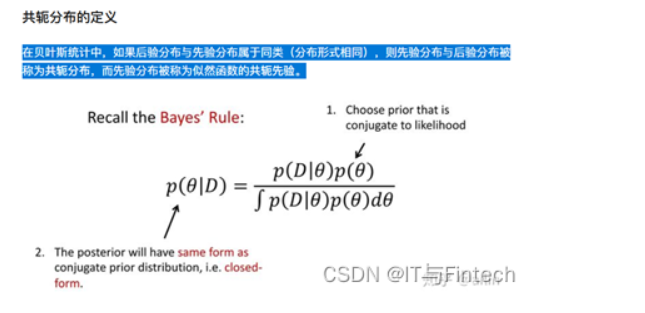


所以最大后验概率就是估计

12各种分布及其期望、方差、分布函数

链接

13什么是共轭先验分布

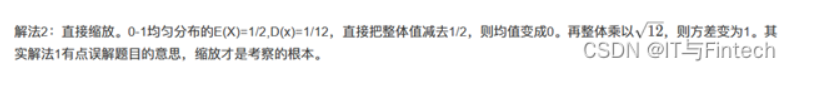


14频率学派和贝叶斯学派的区别

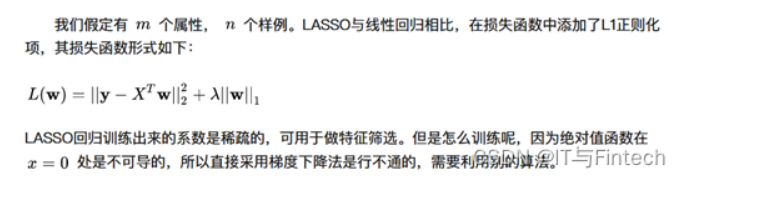
频率派认为模型参数是固定的,一个模型在无数次抽样后,参数是不变的；频率派认为模型不存在先验

贝叶斯学派认为数据才是固定的而参数并不是。贝叶斯派认为模型存在先验。

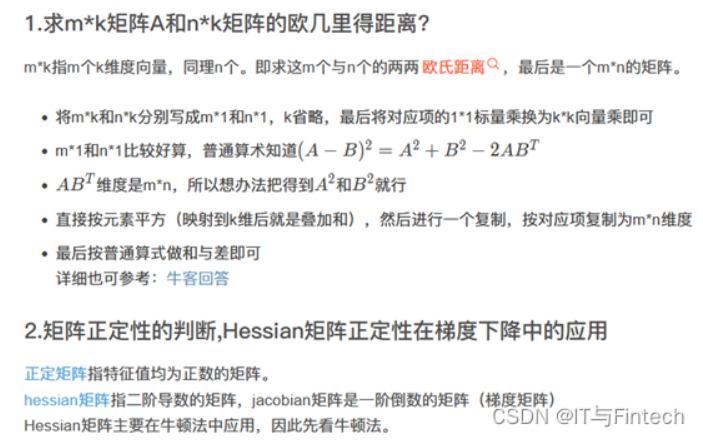
1. 0~1均匀分布的随机器如何变化成均值为0，方差为1的随机器



16Lasso的损失函数

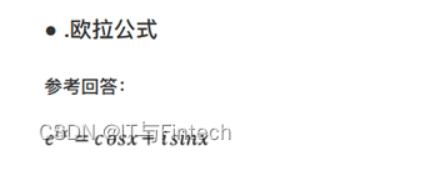


17线性代数

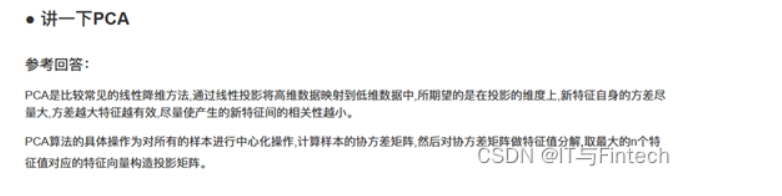


若矩阵所有特征值均不小于0,则判定为半正定。若矩阵所有特征值均大于0,则判定为正定。在判断优化算法的可行性时Hessian矩阵的正定性起到了很大的作用,若Hessian正定,则函数的二阶偏导恒大于0,函数的变化率处于递增状态,在牛顿法等梯度下降的方法中,Hessian矩阵的正定性可以很容易的判断函数是否可收敛到局部或全局最优解。

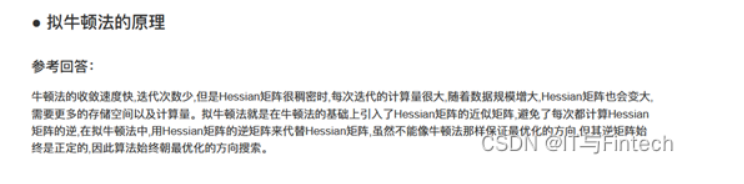
18欧拉公式



19PCA



20拟牛顿法



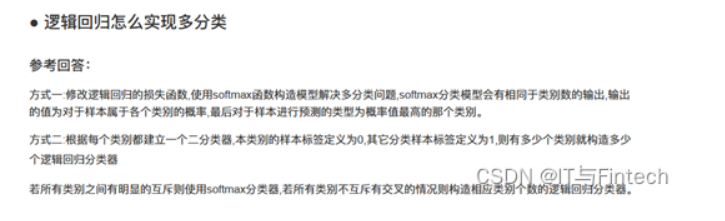
21交叉熵与logistic回归

交叉熵损失按数推导：

链接1

链接2

逻辑回归理解：链接



22SVM中什么时候用线性核什么时候用高斯核?

当数据的特征提取的较好,所包含的信息量足够大,很多问题是线性可分的那么可以采用线性核。若特征数较少,样本数适中,对于时间不敏感,遇到的问题是线性不可分的时候可以使用高斯核来达到更好的效果。

23什么是支持向量机,SVM与LR的区别?

支持向量机为一个二分类模型,它的基本模型定义为特征空间上的间隔最大的线性分类器。而它的学习策略为最大化分类间隔,最终可转化为凸二次规划问题求解。

LR是参数模型,SVM为非参数模型。LR采用的损失函数为logisticalloss,而SVM采用的是hingeloss。在学习分类器的时候,SVM只考虑与分类最相关的少数支持向量点。LR的模型相对简单,在进行大规模线性分类时比较方便

24监督学习和无监督学习的区别

输入的数据有标签则为监督学习,输入数据无标签为非监督学习

25机器学习中的距离计算方法?



26朴素贝叶斯（naive Bayes）法的要求是？

贝叶斯定理、特征条件独立假设

解析：朴素贝叶斯属于生成式模型，学习输入和输出的联合概率分布。给定输入x，利用贝叶斯概率定理求出最大的后验概率作为输出y。

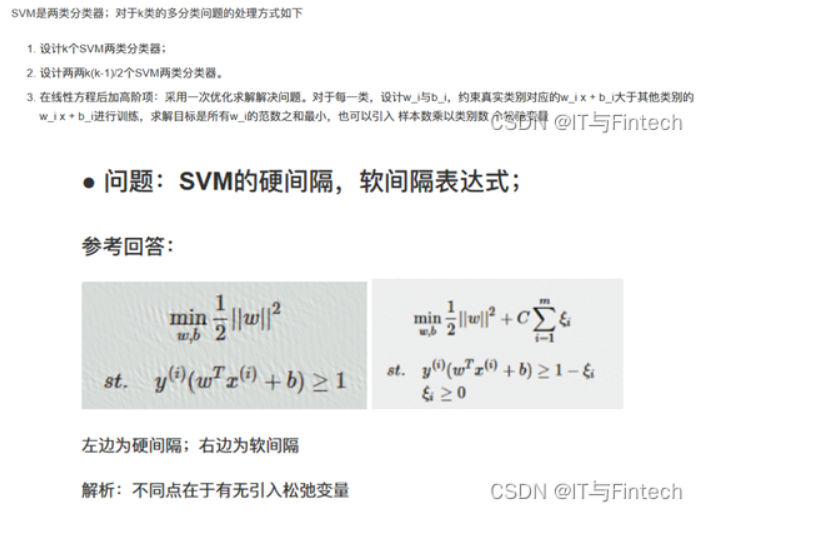
27训练集中类别不均衡，哪个参数最不准确？

准确度（Accuracy）

解析：举例，对于二分类问题来说，正负样例比相差较大为99:1，模型更容易被训练成预测较大占比的类别。因为模型只需要对每个样例按照0.99的概率预测正类，该模型就能达到99%的准确率。

28问题：SVM的作用，基本实现原理；

SVM可以用于解决二分类或者多分类问题，此处以二分类为例。SVM的目标是寻找一个最优化超平面在空间中分割两类数据，这个最优化超平面需要满足的条件是：离其最近的点到其的距离最大化，这些点被称为支持向量



29SVM的物理意义是什么；

构造一个最优化的超平面在空间中分割数据

30SVM使用对偶计算的目的是什么，如何推出来的，手写推导；

目的有两个：一是方便核函数的引入；二是原问题的求解复杂度与特征的维数相关，而转成对偶问题后只与问题的变量个数有关。由于SVM的变量个数为支持向量的个数，相较于特征位数较少，因此转对偶问题。通过拉格朗日算子发使带约束的优化目标转为不带约束的优化函数，使得W和b的偏导数等于零，带入原来的式子，再通过转成对偶问题。

31如果给你一些数据集，你会如何分类（我是分情况答的，从数据的大小，特征，是否有缺失，分情况分别答的）；

根据数据类型选择不同的模型，如Lr或者SVM，决策树。

假如特征维数较多，可以选择SVM模型，

如果样本数量较大可以选择LR模型，但是LR模型需要进行数据预处理；

假如缺失值较多可以选择决策树。

选定完模型后，相应的目标函数就确定了。还可以在考虑正负样例比比，通过上下集采样平衡正负样例比。

32如果数据有问题，怎么处理；

上下采样平衡正负样例比；2.考虑缺失值；3.数据归一化

常见的损失函数有：0-1损失函数、绝对损失函数、平方损失函数、对数损失函数

经验风险用来度量模型在整个训练集上的预测效果好坏，而损失函数用来度量模型在单个训练样本上的预测效果好坏

梯度（gradient）其实是一个向量，一个函数对于其自变量分别求偏导数，这些偏导数所组成的向量就是函数的梯度。

线性可分支持向量机的模型表达式和感知机是一样的，只不过对于模型参数的求解方式不一样，导致模型的泛化性能也不一样。感知机利用误分类最小的策略得到分离超平面，线性可分支持向量机利用间隔最大化的策略得到分离超平面。

线性可分支持向量机是通过最大化训练集的几何间隔来得到模型参数的。几何间隔最大化可以转化为不等式约束条件下的目标函数最小化问题。

线性可分支持向量机的模型参数通过训练集的每个样本和其对应的拉格朗日乘子计算而来，有多少个训练样本，就有多少个拉格朗日乘子；其中拉格朗日乘子大于零的样本特征向量就是支持向量

非线性支持向量机用来解决非线性分类问题，本质是将特征向量映射到新的特征空间中，从而使得数据在新的特征空间中线性可分，然后利用线性支持向量机的方法求解模型参数

33分层抽样的适用范围

分层抽样利用事先掌握的信息,充分考虑了保持样本结构和总体结构的一致性,当总体由差异明显的几部分组成的时候,适合用分层抽样。

34LR和线性回归的区别

线性回归用来做预测,LR用来做分类。线性回归是来拟合函数,LR是来预测函数。线性回归用最小二乘法来计算参数,LR用最大似然估计来计算参数。线性回归更容易受到异常值的影响,而LR对异常值有较好的稳定性。

35生成模型和判别模型基本形式，有哪些？

生成式：朴素贝叶斯、HMM隐含马尔柯夫模型、Gaussians、马尔科夫随机场

判别式：LR，SVM，神经网络，CRF，Boosting



36核函数的种类和应用场景

线性核、多项式核、高斯核、Sigmoid核函数。

特征维数高选择线性核

样本数量可观、特征少选择高斯核（非线性核）

样本数量非常多选择线性核（避免造成庞大的计算量）

当样本的特征很多且维数很高时可考虑用SVM的线性核函数。当样本的数量较多,特征较少时,一般手动进行特征的组合再使用SVM的线性核函数。当样本维度不高且数量较少时,且不知道该用什么核函数时一般优先使用高斯核函数,因为高斯核函数为一种局部性较强的核函数,无论对于大样本还是小样本均有较好的性能且相对于多项式核函数有较少的参数

37核函数的作用

核函数隐含着一个从低维空间到高维空间的映射,这个映射可以把低维空间中线性不可分的两类点变成线性可分的。

38分类算法列一下有多少种？应用场景

单一的分类方法主要包括：LR逻辑回归，SVM支持向量机，DT决策树、NB朴素贝叶斯、NN人工神经网络、K-近邻；

集成学习算法：基于Bagging和Boosting算法思想，RF随机森林,GBDT，Adaboost,XGboost

39SVM为什么使用对偶函数求解

对偶将原始问题中的不等式约束转为了对偶问题中的等式约束, 对偶问题更容易求解，因为不用求w了；而且更加方便了核函数的引入,同时也改变了问题的复杂度,在原始问题下,求解问题的复杂度只与样本的维度有关,在对偶问题下,只与样本的数量有关。

40ID3,C4.5和CART三种决策树的区别

ID3决策树优先选择信息增益大的属性来对样本进行划分,但是这样的分裂节点方法有一个很大的缺点,当一个属性可取值数目较多时,可能在这个属性对应值下的样本只有一个或者很少个,此时它的信息增益将很高,ID3会认为这个属性很适合划分,但实际情况下叫多属性的取值会使模型的泛化能力较差,

所以C4.5不采用信息增益作为划分依据,而是采用信息增益率作为划分依据。但是仍不能完全解决以上问题,而是有所改善

,这个时候引入了CART树,它使用gini系数作为节点的分裂依据。

41SVM和全部数据有关还是和局部数据有关?

SVM只和分类界限上的支持向量点有关,换而言之只和局部数据有关。

42为什么高斯核能够拟合无穷维度

因为将泰勒展开式代入高斯核,将会得到一个无穷维度的映射。

SVM的复杂度主要由支持向量数刻画，而不是数据的维度，因此相比其他方法，SVM不太容易过拟合。

43LR和SVM 区别

LR是参数模型，SVM是非参数模型。

2）从目标函数来看，区别在于逻辑回归采用的是logistical loss，SVM采用的是hinge loss.这两个损失函数的目的都是增加对分类影响较大的数据点的权重，减少与分类关系较小的数据点的权重。

3）SVM的处理方法是只考虑support vectors，也就是和分类最相关的少数点，去学习分类器。而逻辑回归通过非线性映射，大大减小了离分类平面较远的点的权重，相对提升了与分类最相关的数据点的权重。

4）逻辑回归相对来说模型更简单，好理解，特别是大规模线性分类时比较方便。而SVM的理解和优化相对来说复杂一些，SVM转化为对偶问题后,分类只需要计算与少数几个支持向量的距离,这个在进行复杂核函数计算时优势很明显,能够大大简化模型和计算。

5）logic 能做的 svm能做，但可能在准确率上有问题，svm能做的logic有的做不了。

44 L1和L2正则化的区别

L1是模型各个参数的绝对值之和,L2为各个参数平方和的开方值。

L1正则实质上是对模型参数分布做了拉普拉斯分布的先验性假设，L2正则是对模型参数分布做了高斯分布的先验性假设

L1更趋向于产生少量的特征,其它特征为0,最优的参数值很大概率出现在坐标轴上,从而导致产生稀疏

的权重矩阵,而L2会选择更多的特征，这些特征都会接近于0。最优的参数值很小概率出现在坐标轴上，因此每一维的参数都不会是0。当最小化||w||时，就会使每一项趋近于0

45Loss Function有哪些，怎么用？

平方损失（预测问题）、交叉熵（分类问题）、hinge损失（SVM支持向量机）、CART回归树的残差损失

46线性回归的表达式，损失函数；

线性回归y=wx+b，w和x可能是多维。线性回归的损失函数为平方损失函数。

反向求导推导

47知道哪些传统机器学习模型

1）.回归算法：回归算法是试图采用对误差的衡量来探索变量之间的关系的一类算法。回归算法是统计机器学习的利器。 常见的回归算法包括：最小二乘法（Ordinary Least Square），逐步式回归（Stepwise Regression），多元自适应回归样条（Multivariate Adaptive Regression Splines）以及本地散点平滑估计（Locally Estimated Scatterplot Smoothing）。

2）.基于实例的算法：基于实例的算法常常用来对决策问题建立模型，这样的模型常常先选取一批样本数据，然后根据某些近似性把新数据与样本数据进行比较。通过这种方式来寻找最佳的匹配。因此，基于实例的算法常常也被称为“赢家通吃”学习或者“基于记忆的学习”。常见的算法包括 k-Nearest Neighbor(KNN), 学习矢量量化（Learning Vector Quantization， LVQ），以及自组织映射算法（Self-Organizing Map，SOM）。深度学习的概念源于人工神经网络的研究。含多隐层的多层感知器就是一种深度学习结构。深度学习通过组合低层特征形成更加抽象的高层表示属性类别或特征，以发现数据的分布式特征表示。

3）.决策树学习：决策树算法根据数据的属性采用树状结构建立决策模型， 决策树模型常常用来解决分类和回归问题。常见的算法包括：分类及回归树（Classification And Regression Tree，CART），ID3 (Iterative Dichotomiser 3)，C4.5，Chi-squared Automatic Interaction Detection(CHAID), Decision Stump, 随机森林（Random Forest），多元自适应回归样条（MARS）以及梯度推进机（Gradient Boosting Machine，GBM）。

4）.贝叶斯方法：贝叶斯方法算法是基于贝叶斯定理的一类算法，主要用来解决分类和回归问题。常见算法包括：朴素贝叶斯算法，平均单依赖估计（Averaged One-Dependence Estimators，AODE），以及Bayesian Belief Network（BBN）。

5）.基于核的算法：基于核的算法中最著名的莫过于支持向量机（SVM）了。基于核的算法把输入数据映射到一个高阶的向量空间，在这些高阶向量空间里，有些分类或者回归问题能够更容易的解决。常见的基于核的算法包括：支持向量机（Support Vector Machine，SVM）， 径向基函数（Radial Basis Function，RBF)，以及线性判别分析（Linear Discriminate Analysis，LDA)等。

6）.聚类算法：聚类，就像回归一样，有时候人们描述的是一类问题，有时候描述的是一类算法。聚类算法通常按照中心点或者分层的方式对输入数据进行归并。所以的聚类算法都试图找到数据的内在结构，以便按照最大的共同点将数据进行归类。常见的聚类算法包括 k-Means算法以及期望最大化算法（Expectation Maximization，EM）。

7）.降低维度算法：像聚类算法一样，降低维度算法试图分析数据的内在结构，不过降低维度算法是以非监督学习的方式试图利用较少的信息来归纳或者解释数据。这类算法可以用于高维数据的可视化或者用来简化数据以便监督式学习使用。常见的算法包括：主成份分析（Principle Component Analysis，PCA），偏最小二乘回归（Partial Least Square Regression，PLS），Sammon映射，多维尺度（Multi-Dimensional Scaling, MDS）, 投影追踪（Projection Pursuit）等。

8）.关联规则学习：关联规则学习通过寻找最能够解释数据变量之间关系的规则，来找出大量多元数据集中有用的关联规则。常见算法包括 Apriori算法和Eclat算法等。

9）.集成算法：集成算法用一些相对较弱的学习模型独立地就同样的样本进行训练，然后把结果整合起来进行整体预测。集成算法的主要难点在于究竟集成哪些独立的较弱的学习模型以及如何把学习结果整合起来。这是一类非常强大的算法，同时也非常流行。常见的算法包括：Boosting，Bootstrapped Aggregation（Bagging），AdaBoost，堆叠泛化（Stacked Generalization，Blending），梯度推进机（Gradient Boosting Machine, GBM），随机森林（Random Forest）。

10）.人工神经网络：人工神经网络算法模拟生物神经网络，是一类模式匹配算法。通常用于解决分类和回归问题。人工神经网络是机器学习的一个庞大的分支，有几百种不同的算法。（其中深度学习就是其中的一类算法，我们会单独讨论），重要的人工神经网络算法包括：感知器神经网络（Perceptron Neural Network）, 反向传递（Back Propagation），Hopfield网络，自组织映射（Self-Organizing Map, SOM）。学习矢量量化（Learning Vector Quantization， LVQ）

48什么是DBSCAN

DBSCAN是一种基于密度的空间聚类算法,它不需要定义簇的个数,而是将具有足够高密度的区域划分为簇,并在有噪声的数据中发现任意形状的簇,在此算法中将簇定义为密度相连的点的最大集合

49k-means算法流程

从数据集中随机选择k个聚类样本作为初始的聚类中心,然后计算数据集中每个样本到这k个聚类中心的距离,并将此样本分到距离最小的聚类中心所对应的类中。将所有样本归类后,对于每个类别重新计算每个类别的聚类中心即每个类中所有样本的质心,重复以上操作直到聚类中心不变为止。

50LDA的原理

LDA是一种基于有监督学习的降维方式,将数据集在低维度的空间进行投影,要使得投影后的同类别的数据点间的距离尽可能的靠近,而不同类别间的数据点的距离尽可能的远。

51随机森林RF

通过对训练数据样本以及属性进行有放回的抽样（针对某一个属性随机选择样本）这里有两种，一种是每次都是有放回的采样，有些样本是重复的，组成和原始数据集样本个数一样的数据集；另外一种是不放回的抽样，抽取出大约60%的训练信息。由此生成一颗CART树，剩下的样本信息作为袋外数据，用来当作验证集计算袋外误差测试模型；把抽取出的样本信息再放回到原数据集中，再重新抽取一组训练信息，再以此训练数据集生成一颗CART树。这样依次生成多颗CART树，多颗树组成森林，并且他们的生成都是通过随机采样的训练数据生成，因此叫随机森林。RF可以用于数据的回归，也可以用于数据的分类。回归时是由多颗树的预测结果求均值；分类是由多棵树的预测结果进行投票。正式由于它的随机性，RF有极强的防止过拟合的特性。由于他是由CART组成，因此它的训练数据不需要进行归一化，因为每课的建立过程都是通过选择一个能最好的对数据样本进行选择的属性来建立分叉，因此有以上好处的同时也带来了一个缺点，那就是忽略了属性与属性之间的关系。

52KMeans讲讲，KMeans有什么缺点，K怎么确定

1初始化常数K，随机选取初始点为质心

2计算样本与每个质心之间的相似度，将样本归类到最相似的类中

3重新计算质心

4重复计算一下过程，直到质心不再改变

5输出最终的质心以及每个类

k-means存在缺点：

1）k-means是局部最优的，容易受到初始质心的影响；比如在下图中，因选择初始质心不恰当而造成次优的聚类结果。

2）同时，k值的选取也会直接影响聚类结果，最优聚类的k值应与样本数据本身的结构信息相吻合，而这种结构信息是很难去掌握，因此选取最优k值是非常困难的。

3）由于每次都要计算所有的样本与每一个质心之间的相似度，故在大规模的数据集上，K-Means算法的收敛速度比较慢。

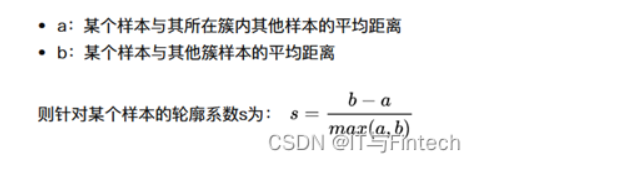
K值得确定：

法1：通过枚举，令k从2到一个固定值如10，在每个k值上重复运行数次kmeans(避免局部最优解)，并计算当前k的平均轮廓系数，最后选取轮廓系数最大的值对应的k作为最终的集群数目。

时间复杂度：O(N)

轮廓系数取值范围为[-1,1]，取值越接近1则说明聚类性能越好，相反，取值越接近-1则说明聚类性能越差。

轮廓系数不应该用来评估不同聚类算法之间的优劣，比如Kmeans聚类结果与DBSCAN聚类结果之间的比较。缺点：不适合基高密度的聚类算法DBSCAN



53DBSCAN原理和算法伪代码，与kmeans，OPTICS区别

DBSCAN聚类算法原理

DBSCAN（Density-Based Spatial Clustering of Applications with Noise）聚类算法，它是一种基于高密度连通区域的、基于密度的聚类算法，能够将具有足够高密度的区域划分为簇，并在具有噪声的数据中发现任意形状的簇。我们总结一下DBSCAN聚类算法原理的基本要点：

https://blog.csdn.net/hansome\_hong/article/details/107596543

DBSCAN 算法是一种基于密度的聚类算法：

　　1.聚类的时候不需要预先指定簇的个数

　　2.最终的簇的个数不确定

2个算法参数：邻域半径R和最少点数目minpoints

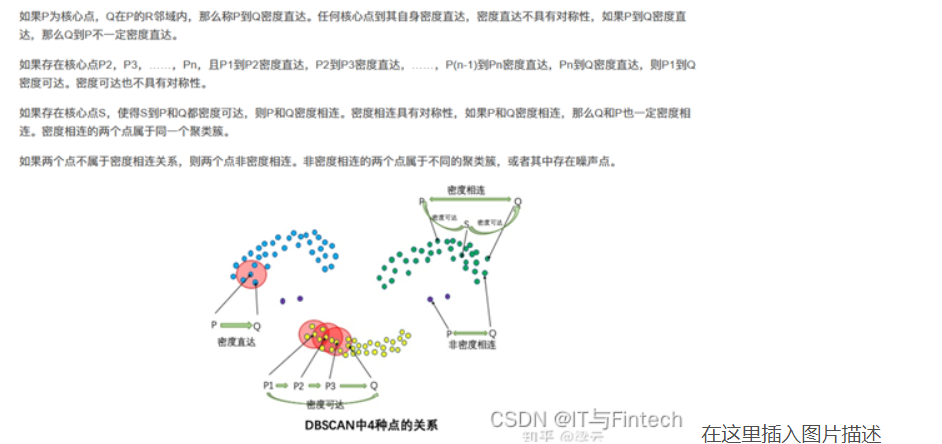
DBSCAN算法将数据点分为三类：

　　1.核心点：在半径Eps内含有超过MinPts数目的点。

　　2.边界点：在半径Eps内点的数量小于MinPts,但是落在核心点的邻域内的点。

　　3.噪音点：既不是核心点也不是边界点的点

4种点的关系：密度直达，密度可达，密度相连，非密度相连



在这里插入图片描述

![在这里插入图片描述](https://img-blog.csdnimg.cn/d7b337677a2f42e7a1e5c93cbf64a347.png

实例

3. 算法优缺点

和传统的 k-means 算法相比，DBSCAN 算法不需要输入簇数 k 而且可以发现任意形状的聚类簇，同时，在聚类时可以找出异常点。

DBSCAN 算法的主要优点如下。

1）可以对任意形状的稠密数据集进行聚类，而 k-means 之类的聚类算法一般只适用于凸数据集。

2）可以在聚类的同时发现异常点，对数据集中的异常点不敏感。

3）聚类结果没有偏倚，而 k-means 之类的聚类算法的初始值对聚类结果有很大影响。

DBSCAN 算法的主要缺点如下。

1）样本集的密度不均匀、聚类间距差相差很大时，聚类质量较差，这时用 DBSCAN 算法一般不适合。

2）样本集较大时，聚类收敛时间较长，此时可以对搜索最近邻时建立的 KD 树或者球树进行规模限制来进行改进。

3）调试参数比较复杂时，主要需要对距离阈值 Eps，邻域样本数阈值 MinPts 进行联合调参，不同的参数组合对最后的聚类效果有较大影响。

4）对于整个数据集只采用了一组参数。如果数据集中存在不同密度的簇或者嵌套簇，则 DBSCAN 算法不能处理。为了解决这个问题，有人提出了 OPTICS 算法。

5）DBSCAN 算法可过滤噪声点，这同时也是其缺点，这造成了其不适用于某些领域，如对网络安全领域中恶意攻击的判断

DBSCAN和Kmeans的区别：

1.DBSCAN自动地确定簇个数，对于K均值，簇个数需要作为参数指定。然而，DBSCAN必须指定另外两个参数：Eps（邻域半径）和MinPts（最少点数）。

2. K均值算法的时间复杂度是O(m)，而DBSCAN的时间复杂度是O(m^2)，

3．K均值一般聚类所有对象，而DBSCAN丢弃被它识别为噪声的对象。

4.K均值使用簇的基于原型的概念，而DBSCAN使用基于密度的概念

5.K均值很难处理非球形的簇和不同大小的簇。DBSCAN可以处理不同大小或形状的簇，并且不太受噪声和离群点的影响

6. 基本K均值算法等价于一种统计聚类方法（混合模型），假定所有的簇都来自球形高斯分布，具有不同的均值，但具有相同的协方差矩阵。DBSCAN不对数据的分布做任何假定

7. K均值可以发现不是明显分离的簇，即便簇有重叠也可以发现，但是DBSCAN会合并有重叠的簇。

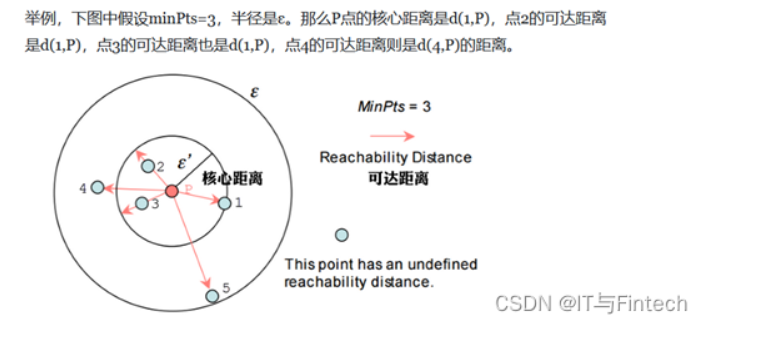
OPTICS：链接

DBSCAN算法中，有两个初始参数Eps（邻域半径）和minPts(Eps邻域最小点数)需要手动设置，并且聚类的结果对这两个参数的取值非常敏感，不同的取值将产生不同的聚类结果。为了克服DBSCAN算法这一缺点，提出了OPTICS算法，该算法可以让算法对半径Eps不再敏感。只要确定minPts的值，半径Eps的轻微变化，并不会影响聚类结果

2个新的定义：

核心距离：一个对象p的核心距离是使得其成为核心点的最小半径，如果p不是核心点，其可达距离没有定义。

可达距离：对于核心点x的邻点x1、x2、…xn 而言，如果他们到点x的距离大于点x’的核心距离，则其可达距离为该点到点x的实际距离；如果他们到点x的距离核心距离小于点x’的核心距离的话，则其可达距离就是点x的核心距离，



54OPTICS算法描述

输入：样本集D, 邻域半径ε, 给定点在ε领域内成为核心对象的最小领域点数MinPts

输出：具有可达距离信息的样本点输出排序



55层次聚类两种方式

层次聚类：

层次聚类就是一层一层的进行聚类，可以由上向下把大的类别（cluster）分割，叫作分裂法；也可以由下向上对小的类别进行聚合，叫作凝聚法；但是一般用的比较多的是由下向上的凝聚方法

　　1、分裂法：

　　分裂法指的是初始时将所有的样本归为一个类簇，然后依据某种准则进行逐渐的分裂，直到达到某种条件或者达到设定的分类数目。用算法描述：

　　输入：样本集合D，聚类数目或者某个条件（一般是样本距离的阈值，这样就可不设置聚类数目）

　　 输出：聚类结果

　　 1.将样本集中的所有的样本归为一个类簇；

　　repeat：

2.在同一个类簇（计为c）中计算两两样本之间的距离，找出距离最远的两个样本a,b；

3.将样本a，b分配到不同的类簇c1和c2中；

4.计算原类簇（c）中剩余的其他样本点和a，b的距离，若是dis(a)<dis(b)，则将样本点归到c1中，否则归到c2中；

　　 util： 达到聚类的数目或者达到设定的条件

　　2、凝聚法：

凝聚法指的是初始时将每个样本点当做一个类簇，所以原始类簇的大小等于样本点的个数，然后依据某种准则合并这些初始的类簇，直到达到某种条件或者达到设定的分类数目。用算法描述：

输入：样本集合D，聚类数目或者某个条件（一般是样本距离的阈值，这样就可不设置聚类数目）

输出：聚类结果

1.将样本集中的所有的样本点都当做一个独立的类簇；

repeat：

2.计算两两类簇之间的距离（后边会做介绍），找到距离最小的两个类簇c1和c2；

3.合并类簇c1和c2为一个类簇；

util： 达到聚类的数目或者达到设定的条件

56bagging和boosting的区别

Bagging是从训练集中进行子抽样组成每个基模型所需要的子训练集,然后对所有基模型预测的结果进行综合操作产生最终的预测结果。

Boosting中基模型按次序进行训练,而基模型的训练集按照某种策略每次都进行一定的转化,最后以一定的方式将基分类器组合成一个强分类器。

1）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。

2）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。

3）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。

4）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。

5)Bagging中的基模型为强模型,Boosting中的基模型为弱模型。

57XGBOOST和GDBT的区别

GDBT在函数空间中利用梯度下降法进行优化而XGBoost在函数空间中使用了牛顿法进行优化。即GDBT在优化中使用了一阶导数信息,而XGB对损失函数进行了二阶泰勒展开,用到了一阶和二阶倒数信息。

XGB在损失函数中加入了正则项(树叶子节点个数,每个叶子节点上输出score的L2模平方和。对于缺失的样本,XGB可以自动学习出它的分裂方向。

GDBT的节点分裂方式使用的是gini系数,XGB通过优化推导出分裂前后的增益来选择分裂节点。

XGB在处理每个特征列时可以做到并行。

58GDBT的原理,以及常用的调参参数

先用一个初始值去学习一棵树,然后在叶子处得到预测值以及预测后的残差,之后的树则基于之前树的残差不断的拟合得到,从而训练出一系列的树作为模型。

n\_estimators基学习器的最大迭代次数,

learning\_rate学习率，

max\_lead\_nodes最大叶子节点数,

max\_depth树的最大深度

min\_samples\_leaf叶子节点上最少样本数。

59stacking和blending的区别?

Stacking和blending的区别在于数据的划分,blending用不相交的数据训练不同的基模型,并将其输出取加权平均。而stacking是将数据集划分为两个不相交的集合,在第一个集合的数据集中训练多个模型,在第二个数据集中测试这些模型,将预测结果作为输入,将正确的标签作为输出,再训练一个高层的模型。

60AdaBoost和GBDT的区别,

AdaBoost通过调整错分的数据点的权重来改进模型,而GBDT是从负梯度的方向去拟合改进模型

AdaBoost改变了训练数据的权值,即样本的概率分布,减少上一轮被正确分类的样本权值,提高被错误分类的样本权值,而随机森林在训练每棵树的时候,随机挑选部分训练集进行训练。

在对新数据进行预测时,AdaBoost中所有树加权投票进行预测,每棵树的权重和错误率有关,而随机森林对所有树的结果按照少数服从多数的原则进行预测。

随机森林，它属于 Bagging 类的算法。使用随机森林解决回归问题，只需要将所有回归决策树的预测值取平均即可。Boosting 类算法在解决回归问题时，只需要将个体学习器的预测值加权求和即可

61gbdt推导

链接

是一种基于boosting集成学习思想的加法模型，是一种用于回归的机器学习算法，该算法由多棵回归决策树组成，所有树的结论累加起来做最终答案。训练时采用前向分布算法进行贪婪的学习，每次迭代都学习一棵CART树来拟合之前 t-1 棵树的预测结果与训练样本真实值的残差。GBDT中的决策树是个弱模型，深度较小一般不会超过5，叶子节点的数量也不会超过10，

GBDT:应用：：：二分类，多分类，特征组合，所有回归问题



GBDT和随机森林的相同点：

1）都是由多棵树组成；2）最终的结果都是由多棵树一起决定

GBDT和随机森林的不同点



随机森林对异常值不敏感，GBDT对异常值非常敏感；

随机森林对训练集一视同仁，GBDT是基于权值的弱分类器的集成；

随机森林是通过减少模型方差提高性能，GBDT是通过减少模型偏差提高性能

决策树模型的优点如下。

1）容易理解和解释，树可以被可视化。2）不需要太多的数据预处理工作，即不需要进行数据归一化，创造哑变量等操作。3）隐含地创造了多个联合特征，并能够解决非线性问题

62xgboost的特征重要性计算

特征重要性可以用来做模型可解释性 xgboost实现中Booster类get\_score方法输出特征重要性，其中importance\_type参数支持三种特征重要性的计算方法：

1.importance\_type=weight（默认值），特征重要性使用特征在所有树中作为划分属性的次数。

2.importance\_type=gain，特征重要性使用特征在作为划分属性时loss平均的降低量。

3.importance\_type=cover，特征重要性使用特征在作为划分属性时对样本的覆盖度

63xgboost原理，怎么防过拟合

XGBoost 属于加法模型，其基函数为回归决策树；

XGBoost 的目标函数为损失函数+正则化项，且损失函数使用了二阶泰勒展开；

XGBoost 使用前向分步算法，通过最小化目标函数来进行模型的优化与学习。

在xgboost调中，一般有两种方式用于控制过拟合：1）直接控制参数的复杂度：包括max\_depth， min\_child\_weight， gamma；2）add randomness来使得对训练对噪声鲁棒。包括subsample colsample\_bytree，或者也可以减小步长 eta，但是需要增加num\_round，来平衡步长因子的减小。

64xgboost，rf，lr优缺点场景

xgb

优缺点：

1）在寻找最佳分割点时，考虑传统的枚举每个特征的所有可能分割点的贪心法效率太低，xgboost实现了一种近似的算法。大致的思想是根据百分位法列举几个可能成为分割点的候选者，然后从候选者中根据上面求分割点的公式计算找出最佳的分割点。

2）xgboost考虑了训练数据为稀疏值的情况，可以为缺失值或者指定的值指定分支的默认方向，这能大大提升算法的效率，paper提到50倍。

3）特征列排序后以块的形式存储在内存中，在迭代中可以重复使用；虽然boosting算法迭代必须串行，但是在处理每个特征列时可以做到并行。

4）按照特征列方式存储能优化寻找最佳的分割点，但是当以行计算梯度数据时会导致内存的不连续访问，严重时会导致cache miss，降低算法效率。paper中提到，可先将数据收集到线程内部的buffer，然后再计算，提高算法的效率。

5）xgboost 还考虑了当数据量比较大，内存不够时怎么有效的使用磁盘，主要是结合多线程、数据压缩、分片的方法，尽可能的提高算法的效率。

适用场景：分类回归问题都可以。

Rf：

优点：

2）随机森林能处理很高维度的数据（也就是很多特征的数据），并且不用做特征选择。

3）在训练完之后，随机森林能给出哪些特征比较重要。

4）训练速度快，容易做成并行化方法(训练时，树与树之间是相互独立的)。5）在训练过程中，能够检测到feature之间的影响。

6）对于不平衡数据集来说，随机森林可以平衡误差。当存在分类不平衡的情况时，随机森林能提供平衡数据集误差的有效方法。

7）如果有很大一部分的特征遗失，用RF算法仍然可以维持准确度。

8）随机森林算法有很强的抗干扰能力（具体体现在6,7点）。所以当数据存在大量的数据缺失，用RF也是不错的。

9）随机森林抗过拟合能力比较强（虽然理论上说随机森林不会产生过拟合现象，但是在现实中噪声是不能忽略的，增加树虽然能够减小过拟合，但没有办法完全消除过拟合，无论怎么增加树都不行，再说树的数目也不可能无限增加的）。

10）随机森林能够解决分类与回归两种类型的问题，并在这两方面都有相当好的估计表现。（虽然RF能做回归问题，但通常都用RF来解决分类问题）。11）在创建随机森林时候，对generlization error(泛化误差)使用的是无偏估计模型，泛化能力强。

缺点：1）随机森林在解决回归问题时，并没有像它在分类中表现的那么好，这是因为它并不能给出一个连续的输出。当进行回归时，随机森林不能够做出超越训练集数据范围的预测，这可能导致在某些特定噪声的数据进行建模时出现过度拟合。2）对于许多统计建模者来说，随机森林给人的感觉就像一个黑盒子，你无法控制模型内部的运行。只能在不同的参数和随机种子之间进行尝试。4）对于小数据或者低维数据（特征较少的数据），可能不能产生很好的分类。（处理高维数据，处理特征遗失数据，处理不平衡数据是随机森林的长处）。5）执行数据虽然比boosting等快（随机森林属于bagging），但比单只决策树慢多了。

适用场景：数据维度相对低（几十维），同时对准确性有较高要求时。因为不需要很多参数调整就可以达到不错的效果，基本上不知道用什么方法的时候都可以先试一下随机森林。

Lr：

优点：实现简单，广泛的应用于工业问题上；

分类时计算量非常小，速度很快，存储资源低；

便利的观测样本概率分数；

缺点：当特征空间很大时，逻辑回归的性能不是很好；容易欠拟合，一般准确度不太高

不能很好地处理大量多类特征或变量；只能处理两分类问题（在此基础上衍生出来的softmax可以用于多分类），且必须线性可分；对于非线性特征，需要进行转换。

适用场景：LR同样是很多分类算法的基础组件，它的好处是输出值自然地落在0到1之间，并且有概率意义。因为它本质上是一个线性的分类器，所以处理不好特征之间相关的情况。虽然效果一般，却胜在模型清晰，背后的概率学经得住推敲。它拟合出来的参数就代表了每一个特征(feature)对结果的影响。也是一个理解数据的好工具。

65xgboost特征并行化怎么做的

决策树的学习最耗时的一个步骤就是对特征值进行排序,在进行节点分裂时需要计算每个特征的增益,最终选增益大的特征做分裂,各个特征的增益计算就可开启多线程进行。而且可以采用并行化的近似直方图算法进行节点分裂。

66xgboost和lightgbm的区别和适用场景

（1）xgboost采用的是level-wise的分裂策略，而lightGBM采用了leaf-wise的策略，区别是xgboost对每一层所有节点做无差别分裂，可能有些节点的增益非常小，对结果影响不大，但是xgboost也进行了分裂，带来了无必要的开销。 leaf-wise的做法是在当前所有叶子节点中选择分裂收益最大的节点进行分裂，如此递归进行，很明显leaf-wise这种做法容易过拟合，因为容易陷入比较高的深度中，因此需要对最大深度做限制，从而避免过拟合。

（2）lightgbm使用了基于histogram的决策树算法，这一点不同与xgboost中的 exact 算法，histogram算法在内存和计算代价上都有不小优势。1）内存上优势：很明显，直方图算法的内存消耗为(#data\* #features \* 1Bytes)(因为对特征分桶后只需保存特征离散化之后的值)，而xgboost的exact算法内存消耗为：(2 \* #data \* #features\* 4Bytes)，因为xgboost既要保存原始feature的值，也要保存这个值的顺序索引，这些值需要32位的浮点数来保存。2）计算上的优势，预排序算法在选择好分裂特征计算分裂收益时需要遍历所有样本的特征值，时间为(#data),而直方图算法只需要遍历桶就行了，时间为(#bin)

（3）直方图做差加速，一个子节点的直方图可以通过父节点的直方图减去兄弟节点的直方图得到，从而加速计算。

（4）lightgbm支持直接输入categorical 的feature，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。

（5）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。

67HMM隐马尔可夫模型的参数估计方法是？

期望最大化（Expectation-Maximum,EM）算法

68Bootstrap方法是什么？

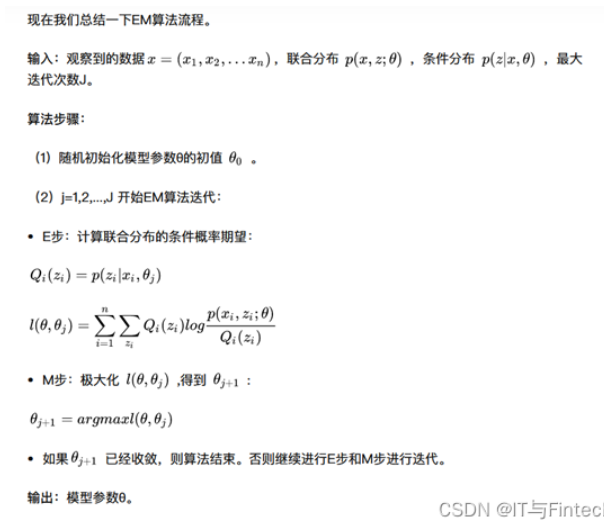
从一个数据集中有放回的抽取N次，每次抽M个。Bagging算法基于bootstrap。面试时结合Bagging算法讲述会更好。

69如何防止过拟合？

1.早停法；2.l1和l2正则化；3.神经网络的dropout；4.决策树剪枝；5.SVM的松弛变量；6.集成学习，扩增数据集,特征的筛选,

在训练和建立模型的时候，从相对简单的模型开始，不要一开始就把特征做的非常多，模型参数跳的非常复杂。

70EM算法推导



EM算法是否一定收敛？EM算法可以保证收敛到一个稳定点，即EM算法是一定收敛的。

————————————————

71 方差偏差的分解公式

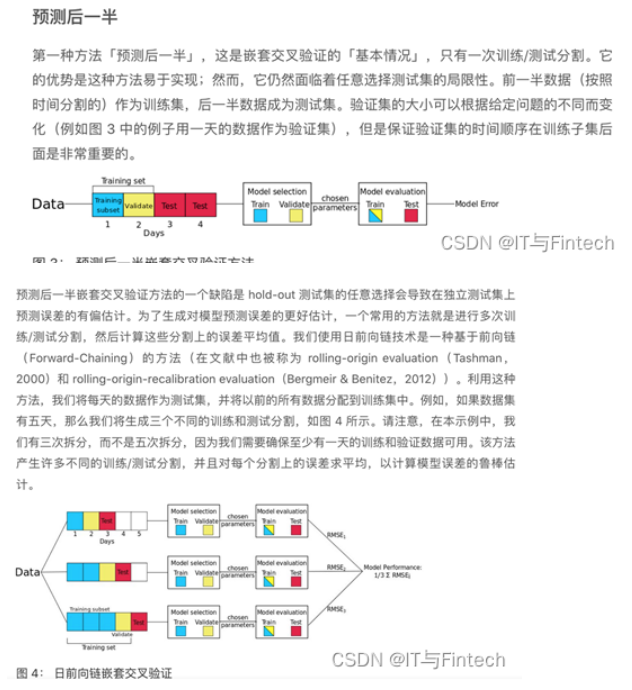
链接

sklearn）是针对Python 编程语言的免费软件机器学习库 [1] 。它具有各种分类，回归和聚类算法，包括支持向量机，随机森林，梯度提升，k均值和DBSCAN，

72对应时间序列的数据集如何进行交叉验证（两种嵌套交叉验证）

传统的交叉验证由于假定样本独立同分布，因此随机打乱分为训练集和验证集。

但是对于时间序列来讲，需要考虑序列间的时间依赖。1.预测后一半；2.日向前链



73正负样本不平衡的解决办法？评价指标的参考价值？

1重新采样数据集：上下采样法。

2使用正确的评估标准,当数据不平衡时可以采用精度,调用度,F1得分,MCC,AUC等评估指标

好的指标：ROC和AUC、F值、G-Mean；不好的指标：Precision、Recall

74迁移学习

迁移学习就是把之前训练好的模型直接拿来用，可以充分利用之前数据信息，而且能够避免自己实验数据量较小等问题。简单来讲就是给模型做初始化，初始化的数据来自于训练好的模型。

75AUC的理解

Auc体现出容忍样本倾斜的能力,只反应模型对正负样本排序能力的强弱,而其直观含以上是任意取一个正样本和负样本,正样本的得分大于负样本的概率。

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

它也是首先对score从大到小排序，然后令最大score对应的sample 的rank为n，第二大score对应sample的rank为n-1，以此类推。然后把所有的正类样本的rank相加，

76生成模型和判别模型的区别

生成模型是先从数据中学习联合概率分布,然后利用贝叶斯公式求得特征和标签对应的条件概率分布。判别模型直接学习条件概率分布,直观的输入什么特征就预测可能的类别。

77L1和L2正则

L范数（L1 norm）L1范数可以使权值稀疏，方便特征提取。 L2范数可以防止过拟合，提升模型的泛化能力。L1和L2的差别，一个让绝对值最小，一个让平方最小，

78ID3树用什么指标选择特征

基于信息增益最大的作为最优特征，以此为决策树的根节点

79特征工程的问题

特征工程包括数据与特征处理、特征选择和降维三部分。

数据与特征处理包括：

1.数据选择、清洗、采样

数据格式化；

数据清洗，填充缺失值、去掉脏数据，将不可信的样本丢掉，缺省值极多的字段考虑不用；

采样：针对正负样本不平衡的情况，当正样本远大于负样本时，且量都很大时，使用下采样，量不大时，可采集更多的数据或oversampling或修改损失函数；采样过程中可利用分层抽样保持不同类别数据的比例。

2.不同类型数据的特征处理

数值型：幅度调整/归一化、log等变化、统计值（例如max、min、mean、std）、离散化、分桶等

类别型：one-hot编码等

时间型：提取出连续值的持续时间和间隔时间；提取出离散值的“年”、“月”、“日”、“一年中哪个星期/季度”、“一周中的星期几”、“工作日/周末”等信息

文本型：使用If-idf特征

统计型：加减平均、分位线、次序、比例

意义：

对数据进行预处理，可提高数据质量；对数据进行清洗可填充缺失值、光滑噪声数据，识别和删除离群点数据，保证数据的一致性；

使用正确的采样方法可解决因数据不平衡带来的预测偏差；

对不同的数据类型进行不同的特征处理有助于提高特征的可用性，例如对数值型数据进行归一化可将数据转化到统一量纲下；对类别型数据，可用one-hot编码方法将类别数据数字化，数字化特征之后可更用来计算距离、相似性等；可从时间型数据当中提取中更多的时间特征

特征选择包括：

1.Filter：使用方差、Pearson相关系数、互信息等方法过滤特征，评估单个特征和结果值之间的相关程度，留下Top相关的特征部分。

2.Wrapper：可利用“递归特征删除算法”，把特征选择看做一个特征子集搜索问题，筛选各种特征子集，用模型评估效果。

3.Embedded：可利用正则化方式选择特征，使用带惩罚项的基模型，除了选择出特征外，同时也进行了降纬。

意义：

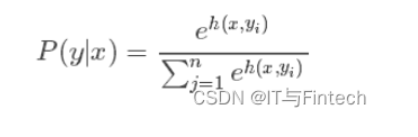
-剔除对结果预测不大的特征，减小冗余，选择有意义的特征输入模型，提高计算性能。

降维：

方法：主成分分析法（PCA）和线性判别分析（LDA）

意义：通过PCA或LDA方法，将较高纬度样本空间映射到较低维度的样本空间，从而达到降纬的目的，减少模型的训练时间，提高模型的计算性能

80 softmax公式



81lightgbm优势

1）更快的训练速度和更高的效率：LightGBM使用基于直方图的算法。

2）更低的内存占用：使用离散的箱子(bins)保存并替换连续值导致更少的内存占用。

3）更高的准确率(相比于其他任何提升算法)：它通过leaf-wise分裂方法产生比level-wise分裂方法更复杂的树，这就是实现更高准确率的主要因素。然而，它有时候或导致过拟合，但是我们可以通过设置|max-depth|参数来防止过拟合的发生。

4）大数据处理能力：相比于XGBoost，由于它在训练时间上的缩减，它同样能够具有处理大数据的能力。

5）支持并行学习。

Libsvm：svm的工具包

82MXNet和Tensorflow的区别

MXNet有两个主要的进程server和worker, server负责分布式存储模型参数，worker负责计算，且worker之间不能直接通信，只能通过server互相影响，mxnet常用来做数据并行，每个GPU设备上都训练完整的DL模型

Tensorflow有worker,server,client三种进程,worker是可以相互通信的,可以根据op的依赖关系主动收发数据。

MXNet常用来做数据并行,每个GPU设备上包含了计算图中所有的op,而Tensorflow可以由用户指定op的放置,一般情况下一个GPU设备负责某个和几个op的训练任务。

83Tensorflow的工作原理

Tensorflow是用数据流图来进行数值计算的,而数据流图是描述有向图的数值计算过程。在有向图中,节点表示为数学运算,边表示传输多维数据,节点也可以被分配到计算设备上从而并行的执行操作

84Tensorflow中interactivesession和session的区别

Tf. Interactivesession()默认自己就是用户要操作的会话,而tf.Session()没有这个默认,所以eval()启动计算时需要指明使用的是哪个会话。

85Keras

是一个由Python编写的开源人工神经网络库

86BatchNormalization的作用，为什么要在后面加伽马和贝塔，不加可以吗

神经网络在训练的时候随着网络层数的加深,激活函数的输入值的整体分布逐渐往激活函的取值区间上下限靠近,从而导致在反向传播时低层的神经网络的梯度消失。而Batch Normalization的作用是把一个batch内的所有数据，通过规范化的手段,将越来越偏的分布拉回到标准化的分布,使得激活函数的输入值落在激活函数对输入比较敏感的区域,从而使梯度变大,加快学习收敛速度,避免梯度消失的问题。

加入缩放平移变量γ和β的原因是： 不一定每次都是标准正态分布，也许需要偏移或者拉伸。直白的说就是为了保证每一次数据经过归一化后还保留原有学习来的特征，同时又能完成归一化操作，加速训练。 这两个参数是用来学习的参数。

1.保留网络各层在训练过程中的学习成果；

2. 保证激活单元的非线性表达能力；

3. 使批归一化具有自我关闭能力。

87梯度消失

在神经网络中，当前面隐藏层的学习速率低于后面隐藏层的学习速率，即随着隐藏层数目的增加，分类准确率反而下降了。这种现象叫做消失的梯度问题

88循环神经网络，为什么好?

循环神经网络模型（RNN）是一种节点定向连接成环的人工神经网络，是一种反馈神经网络，RNN利用内部的记忆来处理任意时序的输入序列，并且在其处理单元之间既有内部的反馈连接又有前馈连接，,在RNN网络结构中中,隐藏层的输入不仅包括输入层的输出还包含上一时刻隐藏层的输出,网络会对之前的信息进行记忆并应用于当前的输入计算中。

89什么是Group Convolution

若卷积神将网络的上一层有N个卷积核,则对应的通道数也为N。设群数目为M,在进行卷积操作的时候,将通道分成M份,每个group对应N/M个通道,然后每个group卷积完成后输出叠在一起,作为当前层的输出通道。

90训练过程中,若一个模型不收敛,那么是否说明这个模型无效?导致模型不收敛的原因有哪些?

并不能说明这个模型无效,

导致模型不收敛的原因可能有 数据分类的标注不准确,样本的信息量太大导致模型不足以fit整个样本空间。学习率设置的太大容易产生震荡,太小会导致不收敛。可能复杂的分类任务用了简单的模型。数据没有进行归一化的操作。

91VGG使用3\*3卷积核的优势是什么? （采用连续的几个3x3的卷积核代替AlexNet中的较大卷积核（11x11，7x7，5x5））

2个33的卷积核串联和55的卷积核有相同的感知野,前者拥有更少的参数。多个33的卷积核比一个较大尺寸的卷积核有更多层的非线性函数,增加了非线性表达,使判决函数更具有判决性。



用C0个33Ci的卷积核进行卷积操作，所以参数量为33CiC0

2个·33参数量：18

1个·55参数量: 25

92Relu比Sigmoid的效果好在哪里?

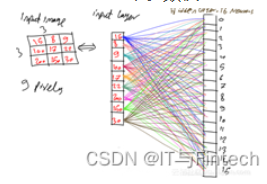
Sigmoid的导数只有在0的附近时有较好的激活性,而在正负饱和区域的梯度趋向于0,从而产生梯度弥散的现象,而relu在大于0的部分梯度为常数,所以不会有梯度弥散现象。

Relu的导数计算的更快。Relu在负半区的导数为0,所以神经元激活值为负时,梯度为0,此神经元不参与训练,具有稀疏性

93神经网络中权重共享的是？ 卷积神经网络、循环神经网络

卷积神经网络可以看做是在空间位置上共享参数，循环神经网络可以看做是在时间位置上共享参数。卷积神经网络通过卷积核来实现权重共享；循环神经网络在理论上可以看做是统一神经结构被无限复制的结果

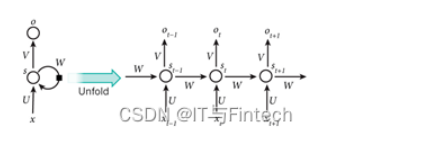
全连接层参数：由于全连接层只能接收一维向量，因此对于输入为M x N的图像，需要先展平为一维的(MxN) x 1。因此若得到的特征图feature map大小为P x Q（对应的一维向量为(PxQ) x 1），则map中每个元素需要MxN个权重，分别于输入图对应做加和，于是最后需要M x N x P x Q个参数权重



若原image大小为M x N x K1 单个卷积核大小为F x F x K1，共K2个卷积核，则总共需要F x F x K1 x K2个参数，之所以在卷积神经网络可以用到权值共享，则是考虑到对于一张图像的某个像素点，其往往具有局部相关性，即与其相邻的像素点相关性较大，越远相关性越小，这样通过一步步增加网络深度，也能慢慢学到图像的全局特征

RNN中需要用参数共享，一方面跟CNN一样，是为了减少参数量

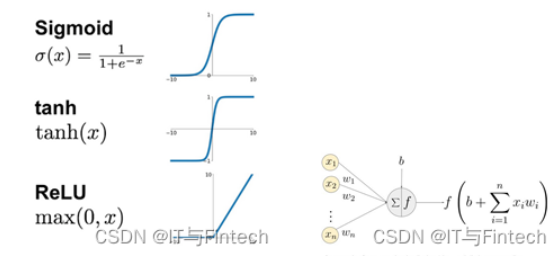
另一方面，由于原始RNN是变长的，因此若不共享，每个W都不同，那么需要设置多少个W就无法提前知晓了，实现上困难



94神经网络激活函数？

sigmod、tanh、relu

激活函数（activation functions）的目标是负责将神经元的输入映射到输出端，将神经网络非线性化，提高神经网络对模型的表达能力,解决线性模型所不能解决的问题。。激活函数是连续的（continuous），且可导的（differential）



当使用sigmoid作为激活函数时，随着神经网络隐含层（hidden layer）层数的增加，训练误差反而加大。表现为：梯度弥散：

靠近输出层的隐含层梯度较大，参数更新速度快，很快就会收敛；

靠近输入层的隐含层梯度较小，参数更新速度慢，几乎和初始状态一样，随机分布；

在含有四个隐藏层的网络结构中，第一层比第四层慢了接近100倍！

sigmoid缺点：

• 激活函数的计算量较大，在反向传播中，当求误差梯度时，求导涉及除法；

• 在反向传播中，容易就会出现梯度消失（函数值趋近于0和1时）和·梯度爆炸（x=0处），无法完成深层网络的训练；

• 函数的敏感区间较短，(-1,1)之间较为敏感，超过区间，则处于饱和状态

• 不以零为中心：sigmoid函数的输出恒为正值，不是以零为中心的，这会导致权值更新时只能朝一个方向更新，从而影响收敛速度。

1）tanh的导数取值范围在0至1之间，优于sigmoid的0至1/4，在一定程度上，减轻了梯度消失的问题（但仍然会有梯度消失问题），即梯度变化更快，也就是在训练过程中收敛速度更快。

2）输出值以 0 为中心，解决了sigmoid函数输出值只为正，梯度只向一个方向更新的问题。tanh的输出和输入能够保持非线性单调上升和下降关系，符合BP（back propagation）网络的梯度求解，容错性好，有界。

tanh和sigmoid都是全部激活（fire），使得神经网络较重（heavy）

缺点：依然存在sigmoid中梯度消失和爆炸的问题和指数运算计算量大的问题。

95relu，（在正区间解决梯度消失）

即Rectified Linear Unit，整流线性单元，激活部分神经元，增加稀疏性，当x小于0时，输出值为0，当x大于0时，输出值为x.

relu对比于sigmoid：

• sigmoid的导数，只有在0附近，具有较好的激活性，而在正负饱和区的梯度都接近于0，会造成梯度弥散；而relu的导数，在大于0时，梯度为常数，不会导致梯度弥散。

• relu函数在负半区的导数为0 ，当神经元激活值进入负半区，梯度就会为0，也就是说，这个神经元不会被训练，即稀疏性；

• relu函数的导数计算更快，程序实现就是一个if-else语句；而sigmoid函数要进行浮点四则运算，涉及到除法；

缺点：不以零为中心：和 Sigmoid 激活函数类似，ReLU 函数的输出不以零为中心，因此只存在正向梯度。

负值区域（x< 0 ）存在梯度消失问题。如果 x < 0，则神经元保持非激活状态，且在反向传播过程中「杀死」梯度。这样权重无法得到更新，网络无法学习。

在神经网络中，隐含层的激活函数，最好选择ReLU



所以，要结合具体问题以及激活函数的特点，恰当地选择。下面是一些经验，供参考：

• Sigmoid函数比较适合于二分类模型。

• 使用Sigmoid函数和tanh函数，要注意梯度消失问题。

• ReLU函数是应用比较广泛的激活函数，可以作为你的默认选项。

• 如果网络中存在大量未激活神经元，可以考虑leaky ReLU函数。

• ReLU函数应该只用于隐藏层。

• 如果是回归模型，在输出层上可以使用线性激活函数

当神经网络很深时，梯度呈指数级增长，最后到输入时，梯度将会非常大，我们会得到一个非常大的权重更新，这就是梯度爆炸的问题

96如何解决梯度消失和爆炸问题：

1非饱和的激活函数（如 ReLU）

2批量规范化（Batch Normalization）

3梯度截断（Gradient Clipping）

4更快的优化器

1）、使用 ReLU、LReLU、ELU、maxout 等激活函数

sigmoid函数的梯度随着x的增大或减小和消失，而ReLU不会。

2）、使用批规范化

通过规范化操作将输出信号x规范化到均值为0，方差为1保证网络的稳定性。从上述分析分可以看到，反向传播式子中有w的存在，所以w的大小影响了梯度的消失和爆炸，Batch Normalization 就是通过对每一层的输出规范为均值和方差一致的方法，消除了w带来的放大缩小的影响，进而解决梯度消失和爆炸的问题。

97在深度学习中，通常会finetuning（微调）已有的成熟模型，再基于新数据，修改最后几层神经网络权值，为什么？

实践中的数据集质量参差不齐，可以使用训练好的网络来进行提取特征。把训练好的网络当做特征提取器。

决定如何使用迁移学习的因素有很多，这是最重要的只有两个：新数据集的大小、以及新数据和原数据集的相似程度。有一点一定记住：网络前几层学到的是通用特征，后面几层学到的是与类别相关的特征。

1、新数据集比较小且和原数据集相似。因为新数据集比较小，如果fine-tune可能会过拟合；又因为新旧数据集类似，我们期望他们高层特征类似，可以使用预训练网络当做特征提取器，用提取的特征训练线性分类器。

2、新数据集大且和原数据集相似。因为新数据集足够大，可以fine-tune整个网络。

3、新数据集小且和原数据集不相似。新数据集小，最好不要fine-tune，和原数据集不类似，最好也不使用高层特征。这时可是使用前面层的特征来训练SVM分类器。

4、新数据集大且和原数据集不相似。因为新数据集足够大，可以重新训练。但是实践中fine-tune预训练模型还是有益的。新数据集足够大，可以fine-tine整个网络。

与重新训练相比，fine-tune要使用更小的学习率。

98微调时候网络参数是否更新？

答案：会更新。

finetune 的过程相当于继续训练，跟直接训练的区别是初始化的时候。

直接训练是按照网络定义指定的方式初始化。

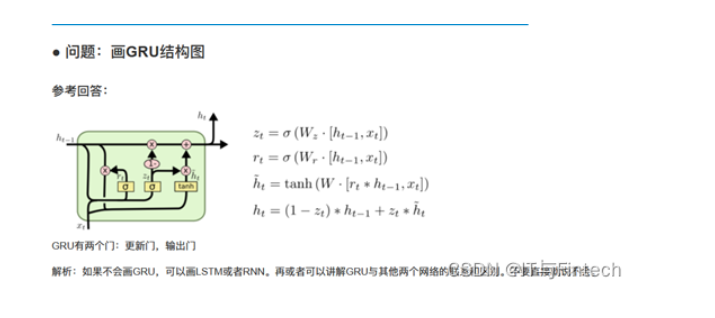
finetune是用你已经有的参数文件来初始化。

fine-tuning 模型的三种状态

状态一：只预测，不训练。 特点：相对快、简单，针对那些已经训练好，现在要实际对未知数据进行标注的项目，非常高效；

状态二：训练，但只训练最后分类层。 特点：fine-tuning的模型最终的分类以及符合要求，现在只是在他们的基础上进行类别降维。

状态三：完全训练，分类层+之前卷积层都训练 特点：跟状态二的差异很小，当然状态三比较耗时和需要训练GPU资源，不过非常适合fine-tuning到自己想要的模型里面，预测精度相比状态二也提高不少。



99Attention机制的作用

Attention简单理解就是权重分配。

减少处理高维输入数据的计算负担,结构化的选取输入的子集,从而降低数据的维度。

让系统更加容易的找到输入的数据中与当前输出信息相关的有用信息,从而提高输出的质量。

帮助类似于decoder这样的模型框架更好的学到多种内容模态之间的相互关系。

100Lstm和Gru的原理

Lstm由输入门,遗忘门,输出门和一个cell组成。第一步是决定从cell状态中丢弃什么信息,然后在决定有多少新的信息进入到cell状态中,最终基于目前的cell状态决定输出什么样的信息。

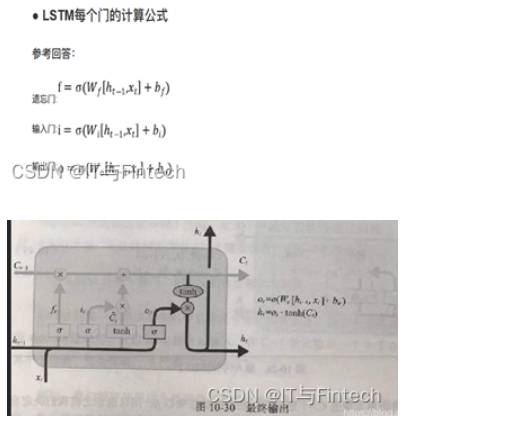
Gru由重置门和跟新门组成,其输入为前一时刻隐藏层的输出和当前的输入,输出为下一时刻隐藏层的信息。重置门用来计算候选隐藏层的输出,其作用是控制保留多少前一时刻的隐藏层。跟新门的作用是控制加入多少候选隐藏层的输出信息,从而得到当前隐藏层的输出。

101什么是dropout

在神经网络的训练过程中,对于隐藏层神经单元按一定的概率将其随机从网络中丢弃,从而达到对于每个mini-batch都是在训练不同网络的效果,防止过拟合。

102DropConnect的原理

防止过拟合方法的一种，对每个节点与之相连的输入权值以一定的概率清0。



103用过哪些移动端深度学习框架？

开源的有：小米的MACE，骁龙的SNPE，腾讯的FeatherCNN和ncnn，百度的mobile-deep-learning(MDL)；caffe、tensorflow lite都有移动端，只是可能没有上面的框架效率高。据传还有支付宝的xNN，商汤的PPL，不过都是自用，未开源。

104Adam

Adam 算法和传统的随机梯度下降不同。随机梯度下降保持单一的学习率（即 alpha）更新所有的权重，学习率在训练过程中并不会改变。而 Adam 通过计算梯度的一阶矩估计和二阶矩估计而为不同的参数设计独立的自适应性学习率。

105学习率的设定类型

1）固定学习率

每次迭代每个参数都使用同样的学习率。找到一个比较好的固定学习率非常关键，否则会导致收敛太慢或者不收敛。

2）不同的参数使用不同的学习率

如果数据是稀疏的且特征分布不均，似乎我们更应该给予较少出现的特征一个大的更新。这时可能需要对不同特征对应的参数设定不同的学习率。深度学习的梯度下降算法中Adagrad 和Adam方法都针对每个参数设置了相应的学习率

3）动态调整学习率

动态调整就是我们根据应用场景，在不同的优化阶段能够动态改变学习率，以得到更好的结果。动态调整学习率是本篇的重点内容，为了解决梯度学习在一些复杂问题时出现的挑战，数据科学家们在动态调整学习率的策略上做了很多研究和尝试。

4）自适应学习率

自适应学习率从某种程度上讲也算是动态调整学习率的范畴，不过更偏向于通过某种算法来根据实时情况计算出最优学习率，而不是人为固定一个简单策略让梯度下降按部就班地实行。

106RNN梯度消失问题,为什么LSTM和GRU可以解决此问题

RNN由于网络较深,后面层的输出误差很难影响到前面层的计算,RNN的某一单元主要受它附近单元的影响。

而LSTM因为可以通过阀门记忆一些长期的信息,相应的也就保留了更多的梯度。

而GRU也可通过重置和更新两个阀门保留长期的记忆,也相对解决了梯度消失的问题。

累乘变累加：RNN的梯度消失主要是因为在计算梯度的时候会出现隐藏层输出对于上一时刻的隐藏层输出求导，这个偏导算出来是(sigmoid’∈(0, 1/4)\*w)的结果会接近0，又随着序列的增长而不断连乘，这样就会导致远距离的输入对于梯度的贡献会逐渐消失，近距离的输入梯度主导，从而是模型无法学习长距离的依赖关系。LSTM其实也会有细胞状态对上一时刻的细胞状态的梯度连乘，但是这个梯度中包含了累加项（f\_t + …），其中有一项是忘记门输出，可以控制梯度不至于过大或者过小。

107RNN容易梯度消失，怎么解决？

1）、梯度裁剪（Clipping Gradient）

既然在BP过程中会产生梯度消失（就是偏导无限接近0，导致长时记忆无法更新），那么最简单粗暴的方法，设定阈值，当梯度小于阈值时，更新的梯度为阈值。

优点：简单粗暴

缺点：很难找到满意的阈值

2）、LSTM（Long Short-Term Memory）

一定程度上模仿了长时记忆，相比于梯度裁剪，最大的优点就是，自动学习在什么时候可以将error反向传播，自动控制哪些是需要作为记忆存储在LSTM cell中。一般长时记忆模型包括写入，读取，和忘记三个过程对应到LSTM中就变成了input\_gate,output\_gate,

forget\_gate,三个门，范围在0到1之间，相当于对输入输出进行加权的学习，利用大量数据来自动学习加权的参数（即学习了哪些错误可以用BP更新参数）。具体的公式表达：

优点：模型自动学习更新参数

108LSTM跟RNN有啥区别

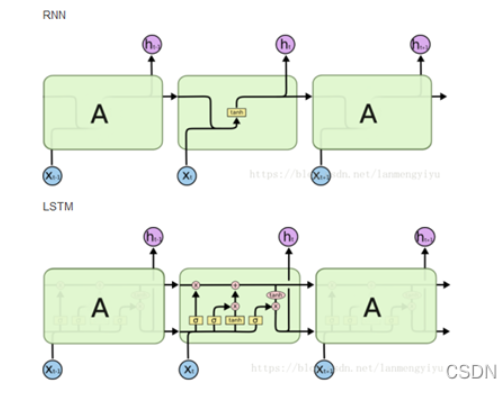
LSTM与RNN的比较

RNN在处理long term memory的时候存在缺陷，因此LSTM应运而生。LSTM是一种变种的RNN，它的精髓在于引入了细胞状态这样一个概念，不同于RNN只考虑最近的状态，LSTM的细胞状态会决定哪些状态应该被留下来，哪些状态应该被遗忘。

RNN在处理长期依赖（时间序列上距离较远的节点）时会遇到巨大的困难，因为计算距离较远的节点之间的联系时会涉及雅可比矩阵的多次相乘，这会带来梯度消失（经常发生）或者梯度膨胀（较少发生）的问题，这样的现象被许多学者观察到并独立研究。为了解决该问题，研究人员提出LSTM。

LSTM是门限RNN，其单一节点的结构如下图1所示。LSTM的巧妙之处在于通过增加输入门限，遗忘门限和输出门限，使得自循环的权重是变化的，这样一来在模型参数固定的情况下，不同时刻的积分尺度可以动态改变，从而避免了梯度消失或者梯度膨胀的问题。

下面来看一些RNN和LSTM内部结构的不同：



109LSTM与GRU区别

1）GRU和LSTM的性能在很多任务上不分伯仲。

2）GRU 参数更少因此更容易收敛，但是数据集很大的情况下，LSTM表达性能更好。

3）从结构上来说，GRU只有两个门（update和reset），LSTM有三个门（forget，input，output），GRU直接将hidden state 传给下一个单元，而LSTM则用memory cell 把hidden state 包装起来

110卷积层和池化层有什么区别



111GAN网络的思想

GAN用一个生成模型和一个判别模型,判别模型用于判断给定的图片是不是真实的图片,生成模型自己生成一张图片和想要的图片很像,开始时两个模型都没有训练,然后两个模型一起进行对抗训练,生成模型产生图片去欺骗判别模型,判别模型去判别真假,最终两个模型在训练过程中,能力越来越强最终达到稳态。

1121\*1的卷积作用

实现跨通道的交互和信息整合,

实现卷积核通道数的降维和升维,

实现多个feature map的线性组合,而且可是实现与全连接层的等价效果。

113怎么提升网络的泛化能力

从数据上提升性能:收集更多的数据,对数据做缩放和变换,特征组合和重新定义问题。

从算法调优上提升性能:用可靠的模型诊断工具对模型进行诊断,权重的初始化,用小的随机数初始化权重。对学习率进行调节,尝试选择合适的激活函数,调整网络的拓扑结构,调节batch和epoch的大小,添加正则化的方法,尝试使用其它的优化方法,使用early stopping。

114什么是seq2seq model

Seq2seq属于encoder-decoder结构的一种,利用两个RNN,一个作为encoder一个作为decoder。Encoder负责将输入序列压缩成指定长度的向量,这个向量可以看作这段序列的语义,而decoder负责根据语义向量生成指定的序列。



115神经网络为啥用交叉熵

通过神经网络解决多分类问题时，最常用的一种方式就是在最后一层设置n个输出节点， 一般情况下，最后一个输出层的节点个数与分类任务的目标数相等。假设最后的节点数为N，那么对于每一个样例，神经网络可以得到一个N维的数组作为输出结果，数组中每一个维度会对应一个类别。在最理想的情况下，如果一个样本属于k，那么这个类别所对应的的输出节点的输出值应该为1，而其他节点的输出都为0，即[0,0,1,0,….0,0]，这个数组也就是样本的Label，是神经网络最期望的输出结果，交叉熵就是用来判定实际的输出与期望的输出的接近程度。均方误差较为平缓，有可能收敛较慢

116注意力公式

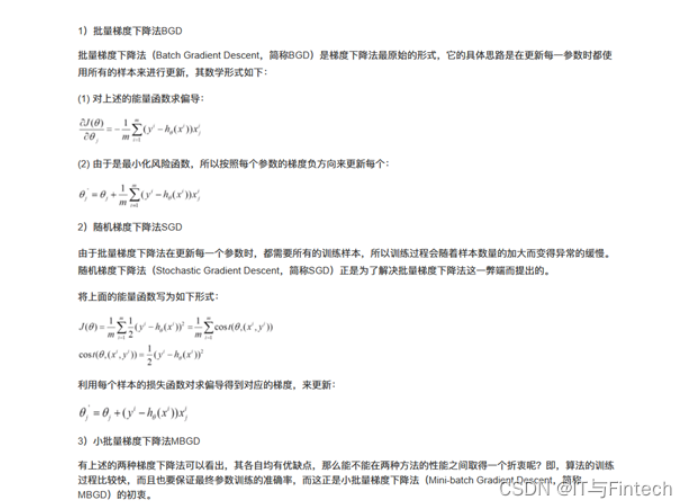
Soft attention、global attention、动态attention ，Hard attention ，静态attention

117论文flow情况

谈谈自己投稿的论文，论文投稿级别，论文内容，用到的方法，对比方法等

118推导LSTM正向传播和单向传播过程

链接



119DNN（深度神经网络）的梯度更新方式

120深度模型压缩方法

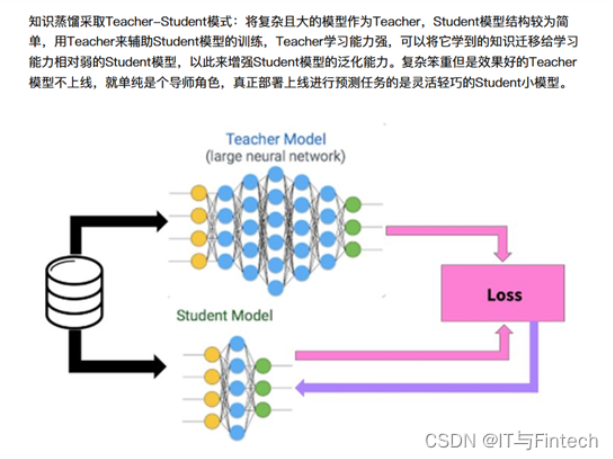
链接

• 参数修剪和共享（针对模型参数的冗余性）3类：模型量化和二进制化、参数共享、结构化矩阵

• 低秩因子分解（使用矩阵/张量分解来估计深度学习模型的信息参数）

• 转移/紧凑卷积滤波器（特殊的结构卷积滤波器来降低存储和计算复杂度）

• 知识蒸馏（学习一个蒸馏模型，训练一个更紧凑的神经网络来重现一个更大的网络的输出）



• 知识蒸馏是对模型的能力进行迁移，根据迁移的方法不同可以简单分为基于目标蒸馏（也称为Soft-target蒸馏或Logits方法蒸馏）和基于特征蒸馏的算法两个大的方向，下面我们对其进行介绍。

一般来说，参数修剪和共享，低秩分解和知识蒸馏方法可以用于全连接层和卷积层的CNN，但另一方面，使用转移/紧凑型卷积核的方法仅支持卷积层。

基于参数修剪/共享、低秩分解的模型可以从预训练模型或者从头开始训练，因此灵活而有效。然而转移/紧凑的卷积核和知识蒸馏模型只能支持从零开始训练。



121模型压缩效果评价指标有哪些？

运行效率、参数压缩率、准确率

122模型优化加速方法

模型优化加速能够提升网络的计算效率，具体包括：

Op-level的快速算法：FFT Conv2d (7x7, 9x9), Winograd Conv2d (3x3, 5x5) 等；

Layer-level的快速算法：Sparse-block net [1] 等；

优化工具与库：TensorRT (Nvidia), Tensor Comprehension (Facebook) 和 Distiller (Intel) 等；

123压缩和加速方法如何选择？

１）对于在线计算内存存储有限的应用场景或设备，可以选择参数共享和参数剪枝方法，特别是二值量化权值和激活、结构化剪枝。其他方法虽然能够有效的压缩模型中的权值参数，但无法减小计算中隐藏的内存大小（如特征图）。

２）如果在应用中用到的紧性模型需要利用预训练模型，那么参数剪枝、参数共享以及低秩分解将成为首要考虑的方法。相反地，若不需要借助预训练模型，则可以考虑紧性滤波设计及知识蒸馏方法。

３）若需要一次性端对端训练得到压缩与加速后模型，可以利用基于紧性滤波设计的深度神经网络压缩与加速方法。

４）一般情况下，参数剪枝，特别是非结构化剪枝，能大大压缩模型大小，且不容易丢失分类精度。对于需要稳定的模型分类的应用，非结构化剪枝成为首要选择。

５）若采用的数据集较小时，可以考虑知识蒸馏方法。对于小样本的数据集，学生网络能够很好地迁移教师模型的知识，提高学生网络的判别性。

６）主流的５个深度神经网络压缩与加速算法相互之间是正交的，可以结合不同技术进行进一步的压缩与加速。如：韩松等人结合了参数剪枝和参数共享；温伟等人以及Alvarez等人结合了参数剪枝和低秩分解。此外对于特定的应用场景，如目标检测，可以对卷积层和全连接层使用不同的压缩与加速技术分别处理。

GPU 加速的地方（指的是使用GPU的地方）

124用过哪些 Optimizer，效果如何

优化算法是不断迭代模型参数以降低模型损失函数的值

1）SGD；2）Momentum；3）Nesterov；4）Adagrad；5）Adadelta；6）RMSprop；7）Adam；8）Adamax；9）Nadam。

（1）对于稀疏数据，尽量使用学习率可自适应的算法，不用手动调节，而且最好采用默认参数。

（2）SGD通常训练时间最长，但是在好的初始化和学习率调度方案下，结果往往更可靠。但SGD容易困在鞍点，这个缺点也不能忽略。

（3）如果在意收敛的速度，并且需要训练比较深比较复杂的网络时，推荐使用学习率自适应的优化方法。

（4）Adagrad，Adadelta和RMSprop是比较相近的算法，表现都差不多。

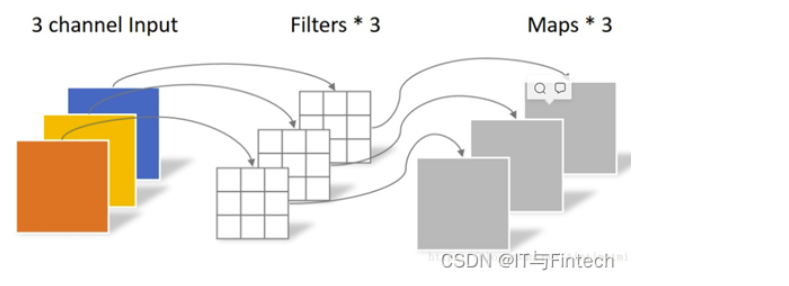
（5）在能使用带动量的RMSprop或者Adam的地方，使用Nadam往往能取得更好的效果。

125Depthwise卷积与Pointwise卷积

Depthwise(DW)卷积与Pointwise(PW)卷积，合起来被称作Depthwise Separable Convolution(参见Google的Xception)，该结构和常规卷积操作类似，可用来提取特征，但相比于常规卷积操作，其参数量和运算成本较低。所以在一些轻量级网络中会碰到这种结构如MobileNet。

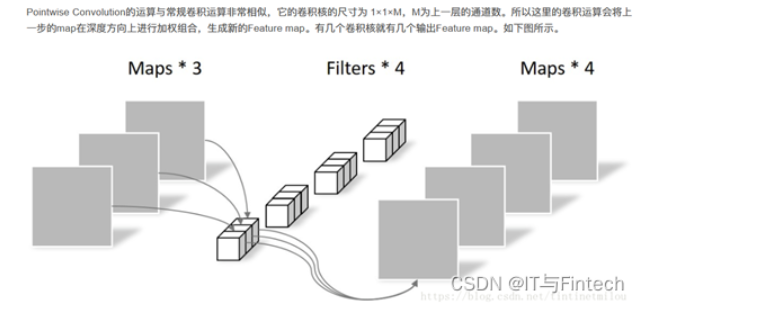
Depthwise Separable Convolution是将一个完整的卷积运算分解为两步进行，即Depthwise Convolution与Pointwise Convolution。

Depthwise Convolution的一个卷积核负责一个通道，一个通道只被一个卷积核卷积。上面所提到的常规卷积每个卷积核是同时操作输入图片的每个通道。Depthwise Convolution完成后的Feature map数量与输入层的通道数相同，无法扩展Feature map。而且这种运算对输入层的每个通道独立进行卷积运算，没有有效的利用不同通道在相同空间位置上的feature信息。因此需要Pointwise Convolution来将这些Feature map进行组合生成新的Feature map。



126Pointwise Convolution

Pointwise Convolution的运算与常规卷积运算非常相似，它的卷积核的尺寸为 1×1×M，M为上一层的通道数。所以这里的卷积运算会将上一步的map在深度方向上进行加权组合，生成新的Feature map。有几个卷积核就有几个输出Feature map。如下图所示。



127Hash表处理冲突的方法

开放定址法

链地址法 ：将所有Hash地址相同的记录都链接在同一链表中

再Hash法 ：同时构造多个不同的Hash函数,当产生冲突时,计算另一个Hash函数地址直到不再发生冲突为止

建立公共溢出区：将Hash表分为基本表和溢出表,若是与基本表发生冲突,都放入溢出表

Apriori原理：如果一个项集是频繁的，则它的所有子集一定也是频繁的；相反，如果项集是非频繁的，则它的所有超集也一定是非频繁的。

加盐hash：就是一个随机生成的字符串。我们将盐与原始密码连接（concat）在一起（放在前面或后面都可以），然后将concat后的字符串加密。采用这种方式加密密码，查表法就不灵了（因为盐是随机生成的）

128中缀表达式转后缀表达式

转换过程需要用到栈，具体过程如下：

1）如果遇到操作数，我们就直接将其输出。

2）如果遇到操作符，则我们将其放入到栈中，遇到左括号时我们也将其放入栈中。

3）如果遇到一个右括号，则将栈元素弹出，将弹出的操作符输出直到遇到左括号为止。注意，左括号只弹出并不输出。

4）如果遇到任何其他的操作符，如（“+”， “\*”，“（”）等，从栈中弹出元素直到遇到发现更低优先级的元素(或者栈为空)为止。弹出完这些元素后，才将遇到的操作符压入到栈中。有一点需要注意，只有在遇到" ) “的情况下我们才弹出” ( “，其他情况我们都不会弹出” ( "。

5）如果我们读到了输入的末尾，则将栈中所有元素依次弹出，并输出。

129大顶堆怎么插入、删除

插入: 需要先将要插入的结点x放在最底层的最右边，,此时需要找到新插入节点的父节点,对堆进行自下而上的调整（从小到大）使其变成一个大顶堆。 时间：O(logn)。 “结点上浮”

删除:将堆的最后一个元素填充到删除元素的位置,然后调整堆结构构造出新的大顶堆 ，并下调到合适位置，最后把该叶子删除。 “结点下沉”

130堆栈区别

堆和栈的区别：

一、堆栈空间分配区别：

1）、栈（操作系统）：由操作系统自动分配释放 ，存放函数的参数值，局部变量的值等。其操作方式类似于数据结构中的栈；

2）、堆（操作系统）： 一般由程序员分配释放， 若程序员不释放，程序结束时可能由OS回收，分配方式倒是类似于链表。

二、堆栈缓存方式区别：

1）、栈使用的是一级缓存，他们通常都是被调用时处于存储空间中，调用完毕立即释放；

2）、堆是存放在二级缓存中，生命周期由虚拟机的垃圾回收算法来决定（并不是一旦成为孤儿对象就能被回收）。所以调用这些对象的速度要相对来得低一些。

堆：内存中，存储的是引用数据类型，引用数据类型无法确定大小，堆实际上是一个在内存中使用到内存中零散空间的链表结构的存储空间，堆的大小由引用类型的大小直接决定，引用类型的大小的变化直接影响到堆的变化

栈：是内存中存储值类型的，大小为2M，超出则会报错，内存溢出

三、堆栈数据结构区别：

堆（数据结构）：堆可以被看成是一棵树，如：堆排序；

栈（数据结构）：一种先进后出的数据结构。

特点：先进后出

131栈溢出有哪些情况

1）、局部数组过大。当函数内部的数组过大时，有可能导致堆栈溢出。

2）、递归调用层次太多。递归函数在运行时会执行压栈操作，当压栈次数太多时，也会导致堆栈溢出。

3）指针或数组越界。这种情况最常见，例如进行字符串拷贝，或处理用户输入等等。

132红黑树

自平衡二叉查找树，用途是实现关联数组。

它可以在O(log n)时间内做查找，插入和删除，这里的n 是树中元素的数目。

set, multiset, map, multimap)应用了红黑树的变体

红黑树是每个节点都带有颜色属性的二叉查找树，颜色或红色或黑色。在二叉查找树强制一般要求以外，对于任何有效的红黑树我们增加了如下的额外要求:

性质1. 节点是红色或黑色。

性质2. 根节点是黑色。

性质3 每个叶节点（NIL节点，空节点）是黑色的。

性质4 每个红色节点的两个子节点都是黑色。(从每个叶子到根的所有路径上不能有两个连续的红色节点)

性质5. 从任一节点到其每个叶子的所有路径都包含相同数目的黑色节点。

红黑树的关键性质: 从根到叶子的最长的可能路径不多于最短的可能路径的两倍长

133对一千万个整数排序,整数范围在[-1000,1000]间,用什么排序最快?

计数排序,计数排序的基本思想为在排序前先统计这组数中其它数小于这个数的个数,其时间复杂度为 ,其中n为整数的个数,k为所有数的范围

134堆排序的思想

将待排序的序列构成一个大顶堆,这个时候整个序列的最大值就是堆顶的根节点,将它与末尾节点进行交换,然后末尾变成了最大值,然后剩余n-1个元素重新构成一个堆,这样得到这n个元素的次大值,反复进行以上操作便得到一个有序序列。

135快速排序的最优情况

链接

快速排序(Quick Sort)使用分治法策略。

它的基本思想是：选择一个基准数，通过一趟排序将要排序的数据分割成独立的两部分；其中一部分的所有数据都比另外一部分的所有数据都要小。然后，再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。

快速排序流程：

(1) 从数列中挑出一个基准值。

(2) 将所有比基准值小的摆放在基准前面，所有比基准值大的摆在基准的后面(相同的数可以到任一边)；在这个分区退出之后，该基准就处于数列的中间位置。

(3) 递归地把"基准值前面的子数列"和"基准值后面的子数列"进行排序。

快速排序是不稳定的算法，它不满足稳定算法的定义。

快速排序的最优情况是Partition每次划分的都很均匀,当排序的元素为n个,则递归树的深度为 。在第一次做Partition的时候需对所有元素扫描一遍,获得的枢纽元将所有元素一分为二,不断的划分下去直到排序结束,而在此情况下快速排序的最优时间复杂度为 。

136topK给出3种解法

1）局部淘汰法 – 借助“冒泡排序”获取TopK

（1）可以避免对所有数据进行排序，只排序部分；

（2）冒泡排序是每一轮排序都会获得一个最大值，则K轮排序即可获TopK。

时间复杂度：排序一轮是O(N)，则K次排序总时间复杂度为：O(KN)。

空间复杂度：O(K)，用来存放获得的topK，

2）局部淘汰法 – 借助数据结构"堆"获取TopK

我们使用小顶堆来实现。

取出K个元素放在另外的数组中，对这K个元素进行建堆。

然后循环从K下标位置遍历数据，只要元素大于堆顶，我们就将堆顶赋值为该元素，然后重新调整为小顶堆。

循环完毕后，K个元素的堆数组就是我们所需要的TopK。

时间复杂度：每次对K个元素进行建堆，时间复杂度为：O(KlogK)，加上N-K次的循环，则总时间复杂度为O((K+(N-K))logK)，即O(NlogK)，

空间复杂度：O(K)，

3）分治法 – 借助”快速排序“方法获取TopK

思路：

（1）比如有10亿的数据，找处Top1000，我们先将10亿的数据分成1000份，每份100万条数据。

（2）在每一份中找出对应的Top 1000，整合到一个数组中，得到100万条数据，这样过滤掉了999%%的数据。

（3）使用快速排序对这100万条数据进行”一轮“排序，一轮排序之后指针的位置指向的数字假设为S，会将数组分为两部分，一部分大于S记作Si，一部分小于S记作Sj。

（4）如果Si元素个数大于1000，我们对Si数组再进行一轮排序，再次将Si分成了Si和Sj。如果Si的元素小于1000，则我们需要在Sj中获取1000-count(Si)个元素的，也就是对Sj进行排序

（5）如此递归下去即可获得TopK。

时间复杂度：一份获取前TopK的时间复杂度：O((N/n)logK)。则所有份数为：O(NlogK)，但是分治法我们会使用多核多机的资源，比如我们有S个线程同时处理。则时间复杂度为：O((N/S)logK)。之后进行快排序，一次的时间复杂度为：O(N),假设排序了M次之后得到结果，则时间复杂度为：O(MN)。所以 ，总时间复杂度大约为O(MN+(N/S)logK) 。

空间复杂度：空间复杂度为O(N)。

137小顶堆的调整过程

堆排序的步骤分为三步:

1）建堆；2）交换数据；3）向下调整。

138BFS和DFS的实现思想

BFS:(1)顶点v入队列(2)当队列为非空时继续执行否则停止(3)从队列头取顶点v,查找顶点v的所有子节点并依次从队列尾插入(4)跳到步骤2

DFS:(1)访问顶点v并打印节点(2)遍历v的子节点w,若w存在递归的执行该节点。

139关联规则具体有哪两种算法,它们之间的区别

Apriori和FP-growth算法

Apriori多次扫描交易数据库,每次利用候选频繁集产生频繁集,

而FP-growth则利用树形结构,无需产生候选频繁集而直接得到频繁集,减少了扫描交易数据库的次数,提高算法的效率

但是Apriori有较好的扩展性可用于并行计算。一般使用Apriori算法进行关联分析,FP-growth算法来高效发现频繁项集。

140贪婪算法

贪婪算法(贪心算法)是指在对问题进行求解时，在每一步选择中都采取最好或者最优(即最有利)的选择，从而希望能够导致结果是最好或者最优的算法。

贪婪算法所得到的结果往往不是最优的结果(有时候会是最优解)，但是都是相对近似(接近)最优解的结果。

贪婪算法并没有固定的算法解决框架，算法的关键是贪婪策略的选择，根据不同的问题选择不同的策略。策略的选择必须具备无后效性，即某个状态的选择不会影响到之前的状态，只与当前状态有关，

其基本的解题思路为：

1)建立数学模型来描述问题;

2)把求解的问题分成若干个子问题;

3)对每一子问题求解，得到子问题的局部最优解;

4)把子问题对应的局部最优解合成原来整个问题的一个近似最优解。

位运算中异或的性质：两个相同数字异或=0，一个数和0异或还是它本身。

141什么是python的生成器?

python生成器是一个返回可以迭代对象的函数,可以被用作控制循环的迭代行为。生成器类似于返回值为数组的一个函数,这个函数可以接受参数,可以被调用,一般的函数会返回包括所有数值的数组,生成器一次只能返回一个值,这样消耗的内存将会大大减小。

使用了 yield 的函数被称为生成器（generator）

调用一个生成器函数，返回的是一个迭代器对象。



142迭代器?

迭代器有两个基本的方法：iter() 和 next()。

list=[1,2,3,4]

it = iter(list) # 创建迭代器对象

print (next(it))

143python中is和==的区别

is是用来判断两个变量引用的对象是否为同一个（is判断的是指针是否指向同一个位置）,==用于判断引用对象的值是否相等。可以通过id()函数查看引用对象的地址。

144python方法解析顺序

优先级从高到低为:实例本身 类 继承类(继承关系越近,越先定义,优先级越高)

145如何判断两个dict是否一样,list头上删除元素,字符串拼接?

通过==可以判断两个dict是否相同

list.pop(0)删除list第一个元素

join()函数进行字符串拼接

146pytorch中cuda()作用,两个Tensor,一个加了cuda(),一个没加,相加后很怎样?

cuda()将操作对象放在GPU内存中,加了cuda()的Tensor放在GPU内存中而没加的Tensor放在CPU内存中,所以这两个Tensor相加会报错

device = torch.device(“cuda:1” )

model = model.to(device)

Tensor和numpy.ndarray之间可以相互转换：

• Numpy转Tensor: torch.from\_numpy(numpy矩阵)

• Tensor转Numpy: Torch矩阵.numpy()

• numpy与Tensor最大的区别就是在对GPU的支持上。Tensor只需要调用cuda()函数就可以将其转化为能在GPU上运行的类型。CPU转GPU: data.cuda()

• GPU转CPU: data.cpu()

147python中dict和list的区别,dict的内部实现

dict查找速度快,占用的内存较大,list查找速度慢,占用内存较小,

dict不能用来存储有序集合。

Dict用{}表示,list用[]表示。

dict是通过hash表实现的,dict为一个数组,数组的索引键是通过hash函数处理后得到的,hash函数的目的是使键值均匀的分布在数组中。

148python dict按照value进行排序

sorted（d.items(),key=lambda x:x[1],reverse=True）

149C++中static关键字的作用

c/c++共有

1）：修饰全局变量时，表明一个全局变量只对定义在同一文件中的函数可见。

2）：修饰局部变量时，表明该变量的值不会因为函数终止而丢失。

3）：修饰函数时，表明该函数只在同一文件中调用。

c++独有：

4）：修饰类的数据成员，表明对该类所有对象这个数据成员都只有一个实例。即该实例归 所有对象共有。

5）：用static修饰不访问非静态数据成员的类成员函数。这意味着一个静态成员函数只能访问它的参数、类的静态数据成员和全局变量



150Python多进程

方式一: os.fork()

方式二:使用multiprocessing模块:创建Process的实例，传入任务执行函数作为参数

方式三:使用multiprocessing模块:派生Process的子类，重写run方法

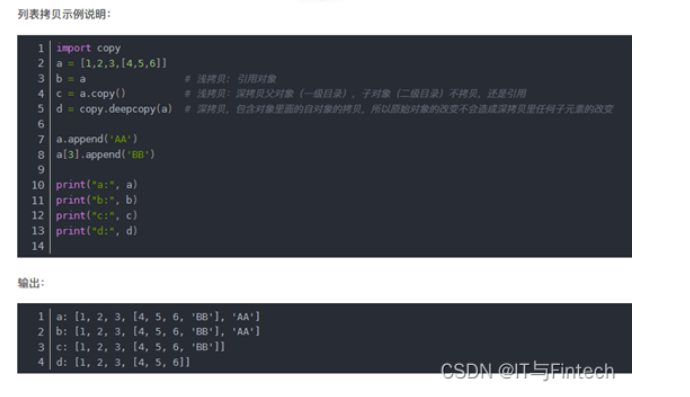
方式四:使用进程池Pool

深拷贝与浅拷贝的区别

深复制和浅复制最根本的区别在于是否是真正获取了一个对象的复制实体，而不是引用。

浅复制 —-只是拷贝了基本类型的数据，而引用类型数据，复制后也是会发生引用，我们把这种拷贝叫做“（浅复制）浅拷贝”，换句话说，浅复制仅仅是指向被复制的内存地址，如果原地址中对象被改变了，那么浅复制出来的对象也会相应改变。 拷贝父对象，不会拷贝对象的内部的子对象

深复制 —-在计算机中开辟了一块新的内存地址用于存放复制的对象。 完全拷贝了父对象及其子对象。



151在程序里面智能指针的名字是啥？

一个是shared\_ptr允许多个指针指向同一个对象；

一个是unique\_ptr独占所指向的对象；

还有一种伴随类weak\_ptr他是一种弱引用, 指向shared\_ptr所指向的对象。

152new，malloc区别

1）.malloc与free是C++/C语言的标准库函数，new/delete是C++的运算符。它们都可用于申请动态内存和释放内存。

2）.对于非内部数据类型的对象而言，光用maloc/free无法满足动态对象的要求。对象在创建的同时要自动执行构造函数，对象在消亡之前要自动执行析构函数。由于malloc/free是库函数而不是运算符，不在编译器控制权限之内，不能够把执行构造函数和析构函数的任务强加于malloc/free。

3）.因此C++语言需要一个能完成动态内存分配和初始化工作的运算符new，以一个能完成清理与释放内存工作的运算符delete。注意new/delete不是库函数。

4）.C++程序经常要调用C函数，而C程序只能用malloc/free管理动态内存

153虚函数和纯虚函数的区别

含有纯虚函数的类称为抽象类,只含有虚函数的类不能称为抽象类。

虚函数可以直接被使用,也可以被子类重载以后以多态形式调用,

而纯虚函数必须在子类中实现该函数才可使用,因为纯虚函数在基类中只有声明而没有定义。虚函数必须实现,对虚函数来说父类和子类都有各自的版本。

154纯虚函数怎么定义，写一个出来

纯虚函数是一种特殊的虚函数，在许多情况下，在基类中不能对虚函数给出有意义的实现，而把它声明为纯虚函数，它的实现留给该基类的派生类去做。这就是纯虚函数的作用。

纯虚函数是一种特殊的虚函数，它的一般格式如下：

class <类名>

{

virtual <类型><函数名>(<参数表>)=0;

…

};

155函数后面接const是什么意思？

这是把整个函数修饰为const，意思是“函数体内不能对成员数据做任何改动”。如果你声明这个类的一个const实例，那么它就只能调用有const修饰的函数

156函数指针

指向函数的指针 如果在程序中定义了一个函数，那么在编译时系统就会为这个函数代码分配一段存储空间，这段存储空间的首地址称为这个函数的地址。而且函数名表示的就是这个地址。既然是地址我们就可以定义一个指针变量来存放，这个指针变量就叫作函数指针变量，简称函数指针。

157c＋＋的一些库吗？

标准库

C++标准库，包括了STL容器，算法和函数等。

C++ Standard Library：是一系列类和函数的集合，使用核心语言编写，也是C++ISO自身标准的一部分。

Standard Template Library：标准模板库

C POSIX library：POSIX系统的C标准库规范

ISO C++ Standards Committee：C++标准委员会

框架

C++通用框架和库

Apache C++ Standard Library：是一系列算法，容器，迭代器和其他基本组件的集合

ASL：Adobe源代码库提供了同行的评审和可移植的C++源代码库。

Boost：大量通用C++库的集合。

BDE：来自于彭博资讯实验室的开发环境。

Cinder：提供专业品质创造性编码的开源开发社区。

Cxxomfort：轻量级的，只包含头文件的库，将C++ 11的一些新特性移植到C++03中。

Dlib：使用契约式编程和现代C++科技设计的通用的跨平台的C++库。

EASTL：EA-STL公共部分

ffead-cpp：企业应用程序开发框架

Folly：由Facebook开发和使用的开源C++库

JUCE：包罗万象的C++类库，用于开发跨平台软件

libPhenom：用于构建高性能和高度可扩展性系统的事件框架。

LibSourcey ：用于实时的视频流和高性能网络应用程序的C++11 evented IO

LibU：C语言写的多平台工具库

Loki：C++库的设计，包括常见的设计模式和习语的实现。

MiLi：只含头文件的小型C++库

openFrameworks ：开发C++工具包，用于创意性编码。

Qt：跨平台的应用程序和用户界面框架

Reason：跨平台的框架，使开发者能够更容易地使用Java，.Net和Python，同时也满足了他们对C++性能和优势的需求。

ROOT：具备所有功能的一系列面向对象的框架，能够非常高效地处理和分析大量的数据，为欧洲原子能研究机构所用。

STLport：是STL具有代表性的版本

STXXL：用于额外的大型数据集的标准模板库。

Ultimate++：C++跨平台快速应用程序开发框架

Windows Template Library：用于开发Windows应用程序和UI组件的C++库

Yomm11：C++11的开放multi-methods.

158Python中的各种锁：

一、全局解释器锁（GIL）

1、什么是全局解释器锁

每个CPU在同一时间只能执行一个线程，那么其他的线程就必须等待该线程的全局解释器，使用权消失后才能使用全局解释器，即使多个线程直接不会相互影响在同一个进程下也只有一个线程使用cpu，这样的机制称为全局解释器锁（GIL）。GIL的设计简化了CPython的实现，使的对象模型包括关键的内建类型，如：字典等，都是隐含的，可以并发访问的，锁住全局解释器使得比较容易的实现对多线程的支持，但也损失了多处理器主机的并行计算能力。

2、全局解释器锁的好处

1）、避免了大量的加锁解锁的好处

2）、使数据更加安全，解决多线程间的数据完整性和状态同步

3、全局解释器的缺点

多核处理器退化成单核处理器，只能并发不能并行。

4、GIL的作用：

多线程情况下必须存在资源的竞争，GIL是为了保证在解释器级别的线程唯一使用共享资源（cpu）。

二、同步锁

1、什么是同步锁？

同一时刻的一个进程下的一个线程只能使用一个cpu，要确保这个线程下的程序在一段时间内被cpu执，那么就要用到同步锁。

2、为什么用同步锁？

因为有可能当一个线程在使用cpu时，该线程下的程序可能会遇到io操作，那么cpu就会切到别的线程上去，这样就有可能会影响到该程　　序结果的完整性。

3、怎么使用同步锁？

只需要在对公共数据的操作前后加上上锁和释放锁的操作即可。

4、同步锁的所用：

为了保证解释器级别下的自己编写的程序唯一使用共享资源产生了同步锁。

三、死锁

1、什么是死锁？

指两个或两个以上的线程或进程在执行程序的过程中，因争夺资源或者程序推进顺序不当而相互等待的一个现象。

2、死锁产生的必要条件？

互斥条件、请求和保持条件、不剥夺条件、环路等待条件

3、处理死锁的基本方法？

预防死锁、避免死锁（银行家算法）、检测死锁（资源分配）、解除死锁：剥夺资源、撤销进程

四、什么是递归锁？

在Python中为了支持同一个线程中多次请求同一资源，Python提供了可重入锁。这个RLock内部维护着一个Lock和一个counter变量，counter记录了acquire的次数，从而使得资源可以被多次require。直到一个线程所有的acquire都被release，其他的线程才能获得资源。递归锁分为可递归锁与非递归锁。

五、什么是乐观锁？

假设不会发生并发冲突，只在提交操作时检查是否违反数据完整性。

六、什么是悲观锁？

假定会发生并发冲突，屏蔽一切可能违反数据完整性的操作。

七、python常用的加锁方式？

互斥锁、可重入锁、迭代死锁、互相调用死锁、自旋锁。

159Linux：ELF的bss段

ELF是Linux系统下的一种可执行可链接文件的格式,而bss段则是用于存放程序中未初始化的全局变量和静态局部变量。

160ip报文经过一个路由器改变哪些字段?

源目的MAC改变，TTL减1

161大小端存储：

• 大端（存储）模式：是指一个数据的低位字节序的内容放在高地址处，高位字节序存的内容放在低地址处。

• 小端（存储）模式：是指一个数据的低位字节序内容存放在低地址处，高位字节序的内容存放在高地址处。（可以总结为“小小小”即低位、低地址、小端）

162count(\*),count(1)和count(列名)的区别

count(\*),count(1)在统计的时候不会忽略Null,count(列名)在统计的时候会忽略Null

列名为主键，count(列名) 会比count(1) 和count(\*) 快；

多列且没有主键，count(1) 执行会比count(\*) 快。

若表中只有一个字段则count(\*)最快。

163排序

链接

在这里插入图片描述

164Python以一定的概率生成某个数

链接

165Python 中列表（ List ）中的 del，remove，和 pop 等的用法和区别

链接

pop

value = List.pop(index)

pop按照索引位置删除元素;

无参数时默认删除最后一个元素

返回删除的元素值

remove

remove 按照值删除，删除单个元素，

删除首个符合条件的元素，

返回值为空 None

del

del 根据索引位置来删除单个值或指定范围内的值

del是删除引用(变量)而不是删除对象(数据)，对象由自动垃圾回收机制（GC）删除

del List\_del\_1[1]

del List\_del\_2[2:4] # 删除[2,4)索引范围内的值

del List\_del\_3

166@property 与 @staticmethod 与 @classmethod 装饰器的介绍与使用@property是python的一种装饰器，是用来修饰方法的。通俗的理解就是：用访问类属性的方式，直接调用类方法

作用：

可以使用@property装饰器来创建只读属性，@property装饰器会将方法转换为相同名称的只读属性

可以与所定义的属性配合使用，这样可以防止属性被修改。

对类的私有属性进行操作



@staticmethod

普通的类内函数是需要实例化之后才可以调用的，但是 @staticmethod 静态方法无需实例化也可直接调用



@classmethod

@classmethod 类方法无需实例化也可直接调用。@classmethod装饰器定义的类方法需要传入类参数cls



167五大常见Python中错误和异常

捕获异常try 。。。except。。。else。。。finally

断言（assert）

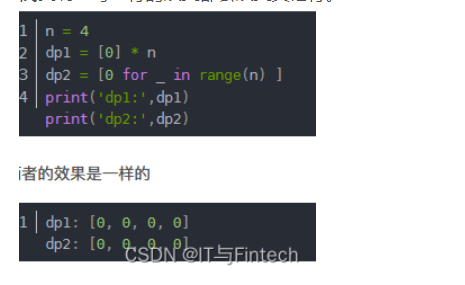
抛出异常(raise)

sys.exc\_info()

168python中[0 ]\* n与[0 for \_ in range(n)]的区别与联系

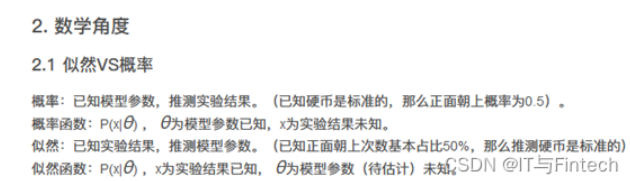
[0 ]\* n与[0 for \_ in range(n)]区别：前者是浅拷贝，也就是把一个列表重复了n次，而后者才是创建，深拷贝

浅复制：每一行的改变都会改变其他行。



169极大似然估计

利用实验结论反推模型参数，这就是“似然”（Likelihood）。



最大似然估计就是：通过真实的实验结果来推测未知的模型的参数，也就是找到一个合理的模型参数，使得实验发生的这个事实存在的概率最该。

170等概率输出0和1

链接

一个随机数发生器，以概率 P 产生0，概率 (1-P) 产生 1，请问能否利用这个随机数发生器，构造出新的发生器，以 1/2 的概率产生 0 和 1 。请写明结论及推理过程。（ 注意：这里的 p 相当于是未知的，后文会提到已知 p 的类型，解题思路是不同的 ）



知随机数生成函数f()，返回0的概率是60%，返回1的概率是40%。根据f()求随机数函数g()，使返回0和1的概率是50%，不能用已有的随机生成库函数

Python写法有答案