# Concepts / 概念

简述二叉树

二叉树是n个有限元素的集合，该集合或者为空、或者由一个称为根（root）的元素及两个不相交的、被分别称为左子树和右子树的二叉树组成。

简述满二叉树

一个二叉树，如果每一个层的结点数都达到最大值，则这个二叉树就是满二叉树。

简述完全二叉树

一棵深度为k的有n个结点的二叉树，对树中的结点按从上至下、从左到右的顺序进行编号，如果编号为i（1≤i≤n）的结点与满二叉树中编号为i的结点在二叉树中的位置相同，则这棵二叉树称为完全二叉树。

简述二叉树的前中后序遍历算法

前序遍历：若二叉树为空树，则执行空逻辑，否则：

访问根节点

递归前序遍历左子树

递归前序遍历右子树

中序遍历：若二叉树为空树，则执行空逻辑，否则：

递归中序遍历左子树

访问根节点

递归中序遍历右子树

后序遍历：若二叉树为空树，则执行空逻辑，否则：

递归后序遍历左子树

递归后序遍历右子树

访问根节点

简述解决Hash冲突的方法

开放定址法：当发生哈希冲突时，如果哈希表未被装满，那么可以把这个值存放到冲突位置中的下一个空位置中去

链地址法：对相同的哈希地址，设置一个单链表，单链表内放的都是哈希冲突元素。

简述AVL树

AVL树是一种改进版的搜索二叉树，其引入平衡因子（左子支高度与右子支高度之差的绝对值），通过旋转使其尽量保持平衡。任何一个节点的左子支高度与右子支高度之差的绝对值不超过1。

简述红黑树

红黑树本身是有2-3树发展而来，红黑树是保持黑平衡的二叉树，其查找会比AVL树慢一点，添加和删除元素会比AVL树快一点。增删改查统计性能上讲，红黑树更优。红黑树主要特征是在每个节点上增加一个属性表示节点颜色，可以红色或黑色。红黑树和 AVL 树类似，都是在进行插入和删除时通过旋转保持自身平衡，从而获得较高的查找性能。红黑树保证从根节点到叶尾的最长路径不超过最短路径的 2 倍，所以最差时间复杂度是 O(logn)。红黑树通过重新着色和左右旋转，更加高效地完成了插入和删除之后的自平衡调整。

简述稳定排序和非稳定排序的区别

稳定排序：排序前后两个相等的数相对位置不变，则算法稳定 非稳定排序：排序前后两个相等的数相对位置发生了变化，则算法不稳定

常见的稳定排序算法有哪些

插入排序、冒泡排序、归并排序

常见的不稳定排序算法有哪些

希尔排序、直接选择排序、堆排序、快速排序

简述插入排序

插入排序：每一趟将一个待排序记录按其关键字的大小插入到已排好序的一组记录的适当位置上，直到所有待排序记录全部插入为止。

排序算法稳定。时间复杂度 O(n²)，空间复杂度 O(1)。

简述希尔排序

希尔排序：把记录按下标的一定增量分组，对每组进行直接插入排序，每次排序后减小增量，当增量减至 1 时排序完毕。

排序算法不稳定。时间复杂度 O(nlogn)，空间复杂度 O(1)。

简述直接选择排序

直接选择排序：每次在未排序序列中找到最小元素，和未排序序列的第一个元素交换位置，再在剩余未排序序列中重复该操作直到所有元素排序完毕。

排序算法不稳定。时间复杂度 O(n²)，空间复杂度 O(1)。

简述堆排序

堆排序：将待排序数组看作一个树状数组，建立一个二叉树堆。通过对这种数据结构进行每个元素的插入，完成排序工作。

排序算法不稳定，时间复杂度 O(nlogn)，空间复杂度 O(1)。

简述冒泡排序

冒泡排序：比较相邻的元素，如果第一个比第二个大就进行交换，对每一对相邻元素做同样的工作。

排序算法稳定，时间复杂度 O(n²)，空间复杂度 O(1)。

简述快速排序

快速排序：随机选择一个基准元素，通过一趟排序将要排序的数据分割成独立的两部分，一部分全部小于等于基准元素，一部分全部大于等于基准元素，再按此方法递归对这两部分数据进行快速排序。

排序算法不稳定，时间复杂度 O(nlogn)，空间复杂度 O(logn)。

简述归并排序

归并排序：将待排序序列分成两部分，然后对两部分分别递归排序，最后进行合并。排序算法稳定，时间复杂度都为 O(nlogn)，空间复杂度为 O(n)。

简述图

图是由顶点集合和顶点之间的边集合组成的一种数据结构，分为有向图和无向图。

有向图：边具有方向性

无向图：边不具有方向性

简述邻接矩阵

用一个二维数组存放图顶点间关系的数据，这个二维数组称为邻接矩阵。对于无向图，邻接矩阵是对称矩阵

简述邻接表

邻接表是通过链表表示图连接关系的一种方。对于表头结点所对应的顶点存在相邻顶点，则把相邻顶点依次存放于表头结点所指向的单向链表中。

简述图的深度优先搜索DFS

将图中每个顶点的访问标志设为 FALSE, 之后搜索图中每个顶点，如果未被访问，则以该顶点V0为起始点出发，访问此顶点，然后依次从V0的各个未被访问的邻接点出发深度优先搜索遍历图，直至图中所有和V0有路径相通的顶点都被访问到。

简述图的广度优先搜索BFS

从图中的某个顶点V0出发，并在访问此顶点之后依次访问V0的所有未被访问过的邻接点，之后按这些顶点被访问的先后次序依次访问它们的邻接点，直至图中所有和V0有路径相通的顶点都被访问到。

简述最小生成树和其对应的算法

对于有 n 个结点的原图，生成原图的极小连通子图，其包含原图中的所有 n 个结点，并且有保持图连通的最少的边。

普里姆算法：取图中任意一个顶点 v 作为生成树的根，之后往生成树上添加新的顶点 w。在添加的顶点 w 和已经在生成树上的顶点v 之间必定存在一条边，并且该边的权值在所有连通顶点 v 和 w 之间的边中取值最小。之后继续往生成树上添加顶点，直至生成树上含有 n-1 个顶点为止。

克鲁斯卡尔算法：先构造一个只含 n 个顶点的子图 SG，然后从权值最小的边开始，若它的添加不使 SG 中产生回路，则在 SG 上加上这条边，如此重复，直至加上 n-1 条边为止。

简述最短路径算法

Dijkstral算法为求解一个点到其余各点最小路径的方法，其算法为：

假设我们求解的是顶点v到其余各个点的最短距离。n次循环至n个顶点全部遍历：

从权值数组中找到权值最小的，标记该边端点k

打印该路径及权值

如果存在经过顶点k到顶点i的边比v->i的权值小

更新权值数组及对应路径

简述堆

堆是一种完全二叉树形式，其可分为最大值堆和最小值堆。

最大值堆：子节点均小于父节点，根节点是树中最大的节点。

最小值堆：子节点均大于父节点，根节点是树中最小的节点。

简述set

Set是一种集合。集合中的对象不按特定的方式排序，并且没有重复对象。

The NLP-AD Team aims to utilize existing and develop new Natural Language Processing (NLP) techniques to extract issue-centric sentiments expressed through the business press and social media as well as in official documents, such as government reports. In addition to English media, the Team will cover Chinese and other media.

The guiding principle is predicated on belief and experiences suggesting that sentiments expressed in texts need to be harnessed with a clear application focus. This issue-centric NLP undertaking has been motivated by the Credit Research Initiative’s (CRI) effort to incorporate incremental credit-focused information conveyed in business media on exchange-listed corporations globally.

In addition to complementing the CRI’s global corporate default prediction platform with NLP enabled alternative data, the Team’s efforts will also be directed at:

Extracting Greenness Sentiment to support the Green Finance research drive at AIDF and

Creating alternative data for SME credit analysis catered to lending institutions.

#### NLP-AD

NLP-AD 团队旨在利用现有和开发新的自然语言处理 (NLP) 技术来提取通过商业媒体和社交媒体以及政府报告等官方文件表达的以问题为中心的情绪。除英文媒体外，该团队还将覆盖中文和其他媒体。

该指导原则基于信念和经验，表明需要以明确的应用重点来利用文本中表达的情感。这项以问题为中心的 NLP 事业受到信用研究计划 (CRI) 的推动，该计划旨在整合全球交易所上市公司商业媒体中传达的增量信用信息。

除了使用支持 NLP 的替代数据补充 CRI 的全球企业违约预测平台外，该团队还将致力于：

提取绿色情绪以支持 AIDF 的绿色金融研究活动和为中小企业信用分析创建替代数据，以满足贷款机构的需求。

## 损失函数 / 如何选择损失函数

1. 什么是损失函数？

一言以蔽之，损失函数（loss function）就是用来度量模型的预测值f(x)与真实值Y的差异程度的运算函数，它是一个非负实值函数，通常使用L(Y, f(x))来表示，损失函数越小，模型的鲁棒性就越好。

2. 为什么使用损失函数？

损失函数使用主要是在模型的训练阶段，每个批次的训练数据送入模型后，通过前向传播输出预测值，然后损失函数会计算出预测值和真实值之间的差异值，也就是损失值。得到损失值之后，模型通过反向传播去更新各个参数，来降低真实值与预测值之间的损失，使得模型生成的预测值往真实值方向靠拢，从而达到学习的目的。

1. Regression Loss Functions
   1. Mean Squared Error Loss (L2 Loss)

在回归问题中，均方误差损失函数用于度量样本点到回归曲线的距离，通过最小化平方损失使样本点可以更好地拟合回归曲线。无参数、计算成本低和具有明确物理意义等优点。

* 1. Mean Squared Logarithmic Error Loss
  2. Mean Absolute Error Loss (L1 Loss)

对离群点有很好的鲁棒性，但它在残差为零处却不可导。另一个缺点是更新的梯度始终相同，也就是说，即使很小的损失值，梯度也很大，这样不利于模型的收敛。针对它的收敛问题，一般的解决办法是在优化算法中使用变化的学习率，在损失接近最小值时降低学习率

* 1. Smooth L1 Loss

主要用在目标检测中防止梯度爆炸。smooth L1和L1-loss函数的区别在于，L1-loss在0点处导数不唯一，可能影响收敛。smooth L1的解决办法是在0点附近使用平方函数使得它更加平滑。

L1 loss that is less sensitive to outliers than the L2 loss used in R-CNN and SPPnet." 也就是smooth L1 loss让loss对于离群点更加鲁棒，即：相比于L2损失函数，其对离群点、异常值（outlier）不敏感，梯度变化相对更小，训练时不容易跑飞。

1. Binary Classification Loss Functions
   1. Binary Cross-Entropy
   2. Hinge Loss

An alternative to cross-entropy for binary classification problems is the hinge loss function, primarily developed for use with Support Vector Machine (SVM) models.

* 1. Squared Hinge Loss

1. Multi-Class Classification Loss Functions
   1. Multi-Class Cross-Entropy Loss

用于评估当前训练得到的概率分布与真实分布的差异情况。为了使神经网络的每一层输出从线性组合转为非线性逼近，以提高模型的预测精度，在以交叉熵为损失函数的神经网络模型中一般选用tanh、sigmoid、softmax或ReLU作为激活函数。

* 1. Sparse Multiclass Cross-Entropy Loss
  2. Kullback Leibler Divergence Loss

KL散度（ Kullback-Leibler divergence）也被称为相对熵(relative entropy)，是一种非对称度量方法，常用于度量两个概率分布之间的距离。KL散度也可以衡量两个随机分布之间的距离，两个随机分布的相似度越高的，它们的KL散度越小，当两个随机分布的差别增大时，它们的KL散度也会增大，因此KL散度可以用于比较文本标签或图像的相似性。基于KL散度的演化损失函数有JS散度函数。JS散度也称JS距离，用于衡量两个概率分布之间的相似度，它是基于KL散度的一种变形，消除了KL散度非对称的问题，与KL散度相比，它使得相似度判别更加准确。

1. 基于距离度量的损失函数
   1. L2损失函数 均方误差损失函数（MSE）
   2. L1损失函数
   3. Smooth L1损失函数
   4. huber损失函数
2. 基于概率分布度量的损失函数

基于概率分布度量的损失函数是将样本间的相似性转化为随机事件出现的可能性，即通过度量样本的真实分布与它估计的分布之间的距离，判断两者的相似度，一般用于涉及概率分布或预测类别出现的概率的应用问题中，在分类问题中尤为常用。

* 1. KL散度函数（相对熵relative entropy）
  2. 交叉熵损失
  3. softmax损失函数
  4. Focal loss

focal loss的引入主要是为了解决难易样本不均衡的问题，注意有区别于正负样本不均衡的问题。

通常情况下，损失函数的选取应从以下方面考虑：

（1） 选择最能表达数据的主要特征来构建基于距离或基于概率分布度量的特征空间。

（2）选择合理的特征归一化方法，使特征向量转换后仍能保持原来数据的核心内容。

（3）选取合理的损失函数，在实验的基础上，依据损失不断调整模型的参数，使其尽可能实现类别区分。

（4）合理组合不同的损失函数，发挥每个损失函数的优点，使它们能更好地度量样本间的相似性。

（5）将数据的主要特征嵌入损失函数，提升基于特定任务的模型预测精确度。

### 为什么使用交叉熵

在机器学习中，我们希望模型在训练数据上学到的预测数据分布与真实数据分布越相近越好，上面讲过了，用相对熵，但是为了简便计算使用交叉熵就可以了。

注意：此处真实数据分布指的就是训练数据的分布(标注)。

二分类中交叉熵损失函数：

交叉熵损失函数一般用来**代替均方差损失函数与sigmoid激活函数组合**。

梯度下降的目的，直白地说：是减小真实值和预测值的距离，而损失函数用来度量真实值和预测值之间距离，所以梯度下降目的也就是减小损失函数的值。怎么减小损失函数的值呢？，所以我们要做的就是不断修改损失函数参数 的值以使损失函数越来越小。

均方差对参数的偏导的结果都乘了sigmoid的导数，偏导数很有可能接近于0。

参数=参数-学习率×损失函数对参数的偏导

可知，偏导很小时，参数更新速度会变得很慢，而当偏导接近于0时，参数几乎就不更新了。

反观交叉熵对参数的偏导就没有sigmoid导数，所以不存在这个问题。这就是选择交叉熵而不选择均方差的原因。

Table

Description automatically generated

Softmax函数是二分类函数Sigmoid在多分类上的推广，目的是将多分类的结果以概率的形式展现出来。

如果模型输出为互斥类别，且只能选择一个类别，则采用Softmax函数计算该网络的原始输出值。

Sigmoid函数可以用来解决多标签问题，Softmax函数用来解决单标签问题。[1]

对于某个分类场景，当Softmax函数能用时，Sigmoid函数一定可以用。

## Random Forest VS GBM

**random forest** uses **bagging** technique to make predictions.

* In bagging technique, a data set is divided into n samples using randomized sampling. Then, using a single learning algorithm a model is built on all samples. Later, the resultant predictions are combined using voting or averaging. Bagging is done in parallel.

RF原理：RF是Bagging的扩展变体，它在以决策树为基础学习器构建Bagging集成的基础上，进一步在决策树的训练过程中引入了随机特征选择，其流程大概是：

1. 随机选择样本（有放回抽样）。
2. 随机选择特征属性，构建决策树。
3. 随机森林投票/平均 得到预测值。

因此防止过拟合能力更强，降低方差。

Bagging，即套袋法，算法过程如下：

1. 从原始样本集中抽取训练集。每轮从原始样本集中使用Bootstraping的方法抽取n个训练样本，共进行k轮抽取，得到k个训练集。
2. 每次使用一个训练集得到一个模型，k个训练集共得到k个模型。（模型可以根据具体问题具体选取，如决策树，SVM，感知机。）

**GBM** uses **boosting** techniques to make predictions.

* In boosting, after the first round of predictions, the algorithm weighs misclassified predictions higher, such that they can be corrected in the succeeding round. This sequential process of giving higher weights to misclassified predictions continue until a stopping criterion is reached.

Random forest improves model accuracy by **reducing variance** (mainly). The trees grown are uncorrelated to maximize the decrease in variance. On the other hand, GBM improves accuracy by **reducing both bias and variance** in a model.

### RandomForest、XGBoost、GBDT和LightGBM

RandomForest、XGBoost、GBDT和LightGBM都属于集成学习 (Ensemble Learning)。

GBDT、XGBoost和LightGBM算法都是基于梯度提升机（Gradient Boosting Machine，GBM）的思想。

集成学习通过构建并结合多个分类器来完成学习任务，也称为多分类系统，集成学习的目的是通过结合多个机器学习分类器的预测结果来改善基本学习器的泛化能力和鲁棒性。

集成学习方法大致分为两类：

* 基本学习器之间存在强依赖关系、必须串行生成的序列化方法，即Boosting提升方法。
* 基本学习器之间不存在强依赖关系、可同时生成的并行化方法，即Bagging (bootstrap aggregating)方法。

Bagging和Boosting采用的都是**采样-学习-组合**的方式，但在细节上有一些不同，如

* **Bagging中每个训练集互不相关，也就是每个基分类器互不相关，而Boosting中训练集要在上一轮的结果上进行调整，也使得其不能并行计算**
* **Bagging中预测函数是均匀平等的，但在Boosting中预测函数是加权的**

XGBoost原理：XGBoost属于集成学习Boosting，是在GBDT的基础上对Boosting算法进行的改进，并加入了模型复杂度的正则项。GBDT是用模型在数据上的负梯度作为残差的近似值，从而拟合残差。XGBoost也是拟合数据残差，并用泰勒展开式对模型损失残差的近似，同时在损失函数上添加了正则化项。

GBDT算法原理：指通过在残差减小的梯度方向建立boosting tree（提升树），即gradient boosting tree（梯度提升树）。每次建立新模型都是为了使之前模型的残差往梯度方向下降。

### XGBoost与GBDT算法的区别：

传统的GBDT在优化的时候只用到了一阶导数信息，而XGBoost则对代价函数进行了二阶泰勒展开，得到一阶和二阶导数，并且XGBoost在代价函数中加入了正则项，用于控制模型的复杂度。

另外XGBoost还支持线性分类器，通过在代价函数中加入正则项，降低了模型的方差，使学习出来的模型更加简单，避免过拟合。

**LightGBM**

lightGBM，它是微软出的新的boosting框架，基本原理与XGBoost一样，使用基于学习算法的决策树，只是在框架上做了一优化（重点在模型的训练速度的优化）。

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。

### lightGBM与XGBoost的区别：

1 xgboost采用的是level-wise的分裂策略，而lightGBM采用了leaf-wise的策略。

level-wise：指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，带来了没必要的开销。

leaf-wise：指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，容易出现过拟合，因此需要做最大深度限制，从而避免过拟合。

2 lightGBM使用了基于histogram（直方图）的决策树算法，而XGBoost使用了exact算法（需要提前预排序）。

能减少内存消耗(#data\* #features \* 1Bytes)，降低计算代价，因为其只需要保存特征离散化之后的值。

而XGBoost使用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。

XGBoost的预排序算法在选择好分裂特征计算分裂受益时，需要遍历所有样本的特征值，时间复杂度为(#data)，与数据量成正比。而lightGBM的直方图只需要遍历分区的直方桶即可，时间复杂度为(#bin)。

3 lightGBM的直方图还能做减差加速作用。

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即可得到，从而实现加速。

4 lightGBM支持直接输入类别categorical的特征feature。

在对离散特征分裂时，每个取值都当作一个桶，分裂时的增益为“是否属于某个类别category”的gain。类似与one-hot编码。

lightGBM实现可多线程优化。（即能够在同一时间执行多个线程，提升整体性能）

lightGBM做了并行处理。

feature parallel（特征并行处理）：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worker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。lightgbm的做法是每个worker都拥有所有数据，再分割。——我也没懂

data parallel（数据并行处理）：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。

**相对熵**又称KL散度,如果我们对于同一个随机变量 x 有两个单独的概率分布 P(x) 和 Q(x)，我们可以使用 KL 散度（Kullback-Leibler (KL) divergence）来衡量这两个分布的差异

维基百科对相对熵的定义

In the context of machine learning, DKL(P‖Q) is often called the information gain achieved if P is used instead of Q.

即如果用P来描述目标问题，而不是用Q来描述目标问题，得到的信息增量。

在机器学习中，P往往用来表示样本的真实分布，比如[1,0,0]表示当前样本属于第一类。Q用来表示模型所预测的分布，比如[0.7,0.2,0.1]

直观的理解就是如果用P来描述样本，那么就非常完美。而用Q来描述样本，虽然可以大致描述，但是不是那么的完美，信息量不足，需要额外的一些“信息增量”才能达到和P一样完美的描述。如果我们的Q通过反复训练，也能完美的描述样本，那么就不再需要额外的“信息增量”，Q等价于P。

在机器学习中，我们需要评估label和predicts之间的差距，使用KL散度刚刚好，即DKL(y||y^)DKL(y||y^)，由于**KL散度中的前一部分−H(y)不变**，故在优化过程中，只需要关注交叉熵就可以了。所以一般在机器学习中直接用用交叉熵做loss，评估模型。

### Batch norm 作用

加速收敛；防止过拟合

因为神经网络本质上是学习数据的分布情况，而mini-batch每次都会有不同的分布，也就是说，每次训练的数据都有不同的分布，数据抖动很大，所以权重变化也会很大。而batch norm会将数据归一化，减少不同batch间数据的抖动情况，从而提高训练速度

BN每次的mini-batch的数据都不一样，但是每次的mini-batch的数据都会对moving mean和moving variance产生作用，可以认为是引入了噪声，这就可以认为是进行了data augmentation，而data augmentation被认为是防止过拟合的一种方法。因此，可以认为用BN可以防止过拟合。

### dropout

在dropout部分， 我们已经解释过， 之所以dropout可以抑制overfitting, 是因为在**训练阶段**， 我们引入了 随机性(随机cancel一些Neuron)，减少网络的匹配度, 在测试阶段， 我们去除掉随机性， 并通过期望的方式marginalize随机影响。

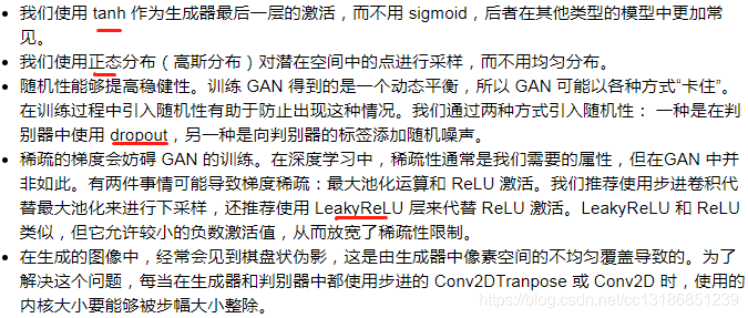
在BatchNormalization中， 训练阶段， 我们随机选取了Batch进行Normalization, 并计算running mean等， 在测试阶段， 应用running\_mean这些训练参数来进行整体Normalization， 本质上是 在Marginalize训练阶段的随机性。 因此， BatchNormalization也提供了 Regularization的作用， 实际应用中证明， NB在防止过拟合方面确实也有相当好的表现。

这一类抑制过拟合的方法有着统一的思想， 即：

***在训练阶段引入随机性， 防止过度匹配； 在测试阶段通过求期望等方式在全局去除掉这种随机性， 从而获得确定而准确的结果。***

这与 L1/L2等正则项通过直接对 训练参数添加某种惩罚的方法 区别明显。

#### GAN



GoodFellow原始GAN论文生成器最后一层激励函数使用的是sigmoid函数，后来有论文指出使用tanh函数收敛更快效果更好，上篇博客也是使用tanh函数，但是发现效果很差，后来自己做了一下实验，把tanh改成sigmoid生成的质量更高，这就造成了一个矛盾，后来仔细检查代码，发现了一个问题所在。一般我们习惯上将输入图片简单归一化到[0,1],但是这时使用sigmoid激活函数比tanh更好，因为sigmoid值域就是[0,1],而tanh函数值域是[-1,1],所以如果生成器最后一层采用tanh激活函数，**切记要把输入缩放在[-1,1]之间，**缩放方式可以简单采取为：img = img/255.0\*2-1.0,这样发现GAN的训练更稳定，收敛速度确实更快，生成质量确实更高。

生成器和判别器最好都使用leaky\_relu作为激活函数，出了生成器最后一层使用tanh,以及判别器最后一层使用sigmoid之外，这样可以防止在训练过成中的梯度消失现象，效果确实比用relu作为激活函数，得到的图片质量更高

由于GAN的终极目标还是为了得到高质量的生成图片，判别器的工作相对与生成器的工作更轻松，所以建议在写生成器代码时，生成器比判别器网络更复杂，会有更好的效果

#### CNN 相关

1.卷积的概念

　　对图像（不同的数据窗口数据）和滤波矩阵（一组固定的权重：因为每个神经元的多个权重固定，所以又可以看做一个恒定的滤波器 filter）**做内积**（逐个元素相乘再求和）的操作就是所谓的『卷积』操作，也是卷积神经网络的名字来源。

2.图像卷积输出size 计算公式

　　output\_size = (input\_size - kernel\_size + 2\*padding)/ stride +1

3.感受野的概念和计算公式

　　感受野： **CNN每一层输出的特征图(feature map)上的像素点在原始图像上映射的区域大小.。**

4.解释权值共享

　　卷积的时候，卷积核上面的一组权重是恒定不变的，也可以这样说，用一个卷积核去卷积一张图，这张图每个位置是被同样数值的卷积核操作的，权重是一样的，也就是参数共享。

5.CNN的特性

　　a.局部连接

　　b.权值共享

　　c.池化操作

　　d.多层系统

## pooling 的作用及 max-pooling 和 average-pooling 的应用场景

　　pooling：对输入的特征图进行压缩，一方面使特征图变小，简化网络计算复杂度；一方面进行特征压缩，提取主要特征。

　　通常来讲，max-pooling的效果更好，虽然max-pooling和average-pooling都对数据做了下采样，但是max-pooling感觉更像是做了特征选择，选出了分类辨识度更好的特征，提供了非线性。 pooling的主要作用一方面是去掉冗余信息，一方面要保留feature map的特征信息，在分类问题中，我们需要知道的是这张图像有什么object，而不大关心这个object位置在哪，在这种情况下显然max pooling比average pooling更合适。在网络比较深的地方，特征已经稀疏了，从一块区域里选出最大的，比起这片区域的平均值来，更能把稀疏的特征传递下去。

　　average-pooling:更强调对整体特征信息进行一层下采样，在减少参数维度的贡献上更大一点，更多的体现在信息的完整传递这个维度上，在一个很大很有代表性的模型中，比如说DenseNet中的模块之间的连接大多采用average-pooling，在减少维度的同时，更有利信息传递到下一个模块进行特征提取。

　　average-pooling在全局平均池化操作中应用也比较广，在ResNet和Inception结构中最后一层都使用了平均池化。有的时候在模型接近分类器的末端使用全局平均池化还可以代替Flatten操作，使输入数据变成一位向量。

7. resnet提出解决的问题

　　解决深层卷积神经网络在反向传播中梯度消失问题。

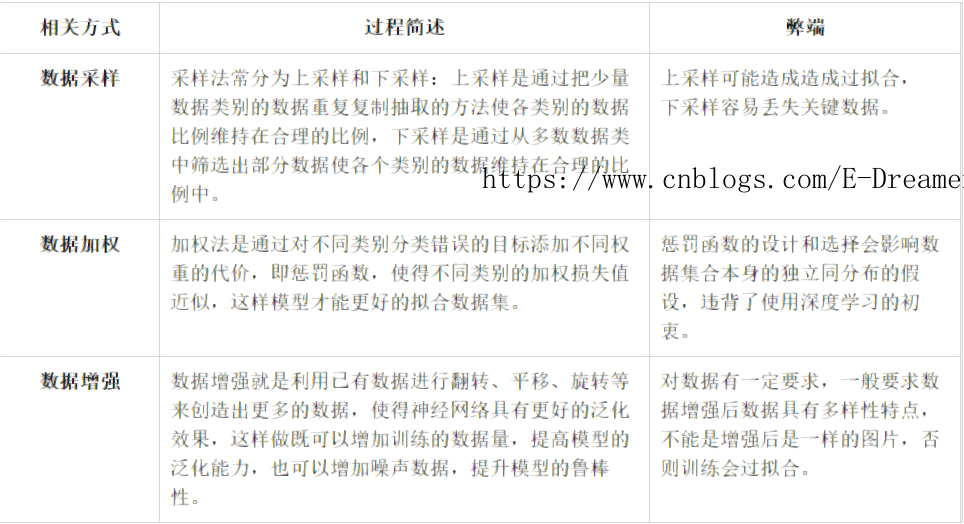
8.CNN参数与计算量的计算

　　卷积输入为 W \* H \* C ，卷积核 K \* K \* N ，输出Ｗ１​\* Ｈ１​\*Ｃ１​

计算量：Ｗ１​\* Ｈ１​\* Ｃ１​ \* K \* K \* C

参数量：C1​\* K \* K \* C

解决类别不平衡的相关方式



Adam优化器，结合AdaGrad和RMSProp两种优化算法的优点。对梯度的一阶矩估计（First Moment Estimation，即梯度的均值）和二阶矩估计（Second Moment Estimation，即梯度的未中心化的方差）进行综合考虑，计算出更新步长。

结合了momuntum 和 RMSProp

lr (float, 可选) – 学习率（默认：1e-3）

betas (Tuple[float, float], 可选) – 用于计算梯度以及梯度平方的运行平均值的系数（默认：0.9，0.999）

eps (float, 可选) – 为了增加数值计算的稳定性而加到分母里的项（默认：1e-8）

weight\_decay (float, 可选) – 权重衰减（L2惩罚）（默认: 0）

个人理解：

lr：同样也称为学习率或步长因子，它控制了权重的更新比率（如 0.001）。较大的值（如 0.3）在学习率更新前会有更快的初始学习，而较小的值（如 1.0E-5）会令训练收敛到更好的性能。

betas = （beta1，beta2）

beta1：一阶矩估计的指数衰减率（如 0.9）。

beta2：二阶矩估计的指数衰减率（如 0.999）。该超参数在稀疏梯度（如在 NLP 或计算机视觉任务中）中应该设置为接近 1 的数。

eps：epsilon：该参数是非常小的数，其为了防止在实现中除以零（如 10E-8）。

## 分类的评价指标

针对一个二分类问题，即将实例分成正类（positive）或负类（negative），在实际分类中会出现以下四种情况：

（1）若一个实例是正类，并且被预测为正类，即为真正类(True Positive TP)

（2）若一个实例是正类，但是被预测为负类，即为假负类(False Negative FN)

（3）若一个实例是负类，但是被预测为正类，即为假正类(False Positive FP)

（4）若一个实例是负类，并且被预测为负类，即为真负类(True Negative TN)

**准确率(Accuracy)**

定义：对于给定的测试数据集，分类器正确分类的样本数与总样本数之比。​

缺点：在正负样本不平衡的情况下，这个指标有很大的缺陷。例如：给定一组测试样本共1100个实例，其中1000个是正类，剩余100个是负类。即使分类模型将所有实例均预测为正类，Accuracy也有90%以上，这样就没什么意义了。

**精确率(Precision)、召回率(Recall)和F1值**

精确率是检索出相关文档数与检索出的文档总数的比率（正确分类的正例个数占分类为正例的实例个数的比例），衡量的是检索系统的查准率。

召回率是指检索出的相关文档数和文档库中所有的相关文档数的比率（正确分类的正例个数占实际正例个数的比例），衡量的是检索系统的查全率。

为了能够评价不同算法优劣，在Precision和Recall的基础上提出了F1值的概念，来对Precision和Recall进行整体评价。

我们当然希望检索结果（分类结果）的Precision越高越好，同时Recall也越高越好，但事实上这两者在某些情况下有矛盾的。比如极端情况下，我们只搜索出了一个结果，且是准确的（分类后的正确实例只有一个，且该实例原本就是正实例），那么Precision就是100%，但是Recall就会很低；而如果我们把所有结果都返回（所有的结果都被分类为正实例），那么Recall是100%，但是Precision就会很低。因此在不同的场合中需要自己判断希望Precision比较高还是Recall比较高。如果是做实验研究，可以绘制Precision-Recall曲线来帮助分析。

5，综合评价指标F-Measure

Precision和Recall指标有时候会出现矛盾的情况，这样就需要综合考虑他们，最常见的方法就是F-Measure（又称为F-Score）。F-Score是Precision和Recall的加权调和平均

当参数a = 1 a=1a=1时，就是最常见的F1.因此，F1综合了P和R的结果，当F1较高时则能说明试验方法比较有效。

**为什么引入ROC曲线？**

Motivation1：在一个二分类模型中，对于所得到的连续结果，假设已确定一个阈值，比如说0.6，大于这个值的实例为正类，小于这个值则为负类。如果减小阈值，比如减到0.5，固然能之别出更多的正类，也就是提高了识别出的正例占所有正例的比例，即TPR值变大；但同时也将更多的负实例当作了正实例，即，提高了FPR。为了形象化这一变化，引入了ROC，ROC可以用于评价一个分类器。

Motivation2：在类不平衡的情况下，如正样本90个，负样本10个，直接把所有样本分类为正样本，得到识别率为90%。但这显然是没有意义的。单纯根据Precision和Rcall来衡量算法的优劣已经不能表征这种病态问题。

6.3 ，什么是ROC曲线？

ROC曲线：接收者操作特征（receiver operating characteristic），ROC曲线上每个点反映着对同一信号刺激的感受性。

横轴：负正类率（FPR，特异度）；

纵轴：真正类率（TPR，灵敏度）。

**什么是AUC曲线？**

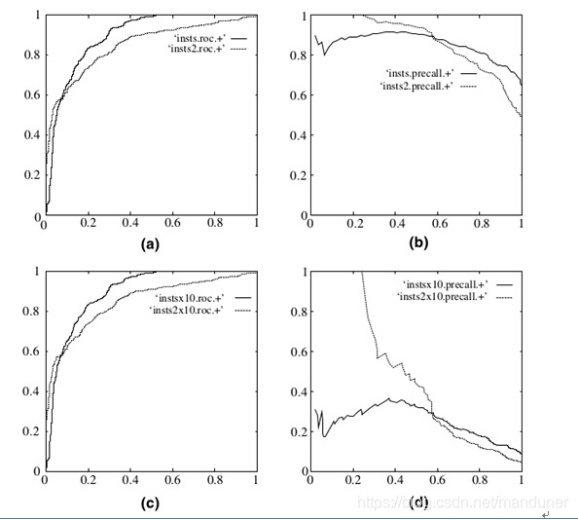
AUC(Area Under Curve)：ROC曲线下的面积，介于0.1和1之间。**AUC作为数值可以直观的评价分类器的好坏，值越大越好。**

AUC的物理意义：首先AUC值是一个概率值。假设分类器的输出是样本属于正类的Score（置信度），则AUC的物理意义为，任取一对（正、负）样本，正样本的Score大于负样本的Score的概率。

AUC为ROC曲线下的面积，那我们直接极端面积可得。面积为一个个小的梯形面积之和。计算的精度与阈值的精度有关。

**为什么使用ROC和AUC评价分类器呢？**

既然已经这么多标准，为什么还要使用ROC和AUC呢？因为ROC曲线有个很好的特性：当测试集中的正负样本的分布变换的时候，ROC曲线能够保持不变。在实际的数据集中经常会出现样本类不平衡，即正负样本比例差距较大，而且测试数据中的正负样本也可能随着时间变化。下图是ROC曲线和Presision-Recall曲线的对比：



决策树的生成主要分以下两步，这两步通常通过学习已经知道分类结果的样本来实现。

1. 节点的分裂：一般当一个节点所代表的属性无法给出判断时，则选择将这一节点分成2个子节点（如不是二叉树的情况会分成n个子节点）

2. 阈值的确定：选择适当的阈值使得分类错误率最小 （Training Error）。

比较常用的决策树有ID3，C4.5和CART（Classification And Regression Tree），CART的分类效果一般优于其他决策树。下面介绍具体步骤。

ID3: 由增熵（Entropy）原理来决定那个做父节点，那个节点需要分裂。对于一组数据，熵越小说明分类结果越好。

## Broadcast

## General semantics

Two tensors are “broadcastable” if the following rules hold:

* Each tensor has at least one dimension.
* When iterating over the dimension sizes, starting at the trailing dimension, the dimension sizes must either be equal, one of them is 1, or one of them does not exist.

# can line up trailing dimensions

>>> **x=torch.empty(**5**,**3**,**4**,**1**)**

>>> **y=torch.empty(** 3**,**1**,**1**)**

# x and y are broadcastable.

# 1st trailing dimension: both have size 1

# 2nd trailing dimension: y has size 1

# 3rd trailing dimension: x size == y size

# 4th trailing dimension: y dimension doesn't exist

# Java / Java机制相关

反射

任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意方法和属性，动态获取信息以及动态调用对象方法的功能称为java语言的反射机制

使用场景：动态类加载，动态代理，反编译，Tomcat服务器。

缺点：性能是一个问题，反射相当于一系列解释操作，通知jvm要做的事情，性能比直接的java代码要慢很多。

# BigData / 大数据相关

Hdfs数据存储过程是怎样的

Hadoop的map reduce过程是怎样的

Spark有哪几种算子

Spark的shuffle过程是怎样的

如何解决spark的数据倾斜

Spark 如何优化两个大小表的join操作

大数据中寻找top-K的方法 （小顶堆min Heap），算法复杂度？（nlogK）

MySQL 相关

MyISAM和InnoDB差别

底层：MyISAM是非聚簇索引，即索引和数据分开存储，叶子节点是行的索引；InnoDB是聚簇索引，索引和数据放在一起，叶子节点是整行数据；

MyISAM不支持事务，InnoDB支持。MyISAM不支持行锁，只支持表锁，InnoDB支持行锁。

InnoDB没有保存具体的行数，所以在统计行数的时候会扫描全表；MyISAM有保存。

myisam的索引以表名+.MYI文件分别保存。innodb的索引和数据一起保存在表空间里。

To prepare, please think of previous experiences where you have demonstrated skill & experience as related to this role. Please ensure you can discuss and provide clear examples of:

* **The type of work you do**
* **Outline of your daily tasks**

Almost different every week, my mentor will told me what the following task. It depends on what kind of research they are focusing on or what kind of technical issues need to resolve. Small institute so very flexible,

* **Provide an overview of the tasks you enjoy and the tasks you dislike**

Pure researching work and developing work, which can explore a certain field of technique.

* **How you manage conflicting tasks and schedules**
* **Your career aspirations / reasons for leaving your current role / motivation for joining the company**

What is LDA?

(Not Linear Discriminant Analysis(LDA) )

## Latent Dirichlet Allocation

Latent Dirichlet allocation is one of the most popular methods for performing topic modelling. Each document consists of various words and each topic can be associated with some words. The aim behind the LDA is to find topics that the document belongs to, based on words contains in it. It assumes that documents with similar topics will use a similar group of words. This enables the documents to map the probability distribution over latent topics, and these topics are probability distribution.

## What is Topic Modeling

Topic modelling is a way of abstract modelling to discover the abstract ‘topics’ that occur in the collections of documents. The idea is that we will perform unsupervised classification on different documents, which find some natural groups in topics. Using topic modelling we can answer:

* What is the topic/main idea of the document?
* Given a document, can we find another document with a similar topic?
* How do topics field change over time?

For Scraper:

Selenium

import json

import re

import uuid

import numpy as np

import pandas as pd

import requests

from joblib import Parallel, delayed

Joblib is a set of tools to provide **lightweight pipelining in Python**. In particular:

1. transparent disk-caching of functions and lazy re-evaluation (memoize pattern)
2. easy simple parallel computing

from bs4 import BeautifulSoup

from selenium import webdriver

from pymongo import MongoClient

# Project / 项目【motivation、架构和创新点】

### Detection of Content Polluters on Twitter

Bag of word Filtering is about filtering out tokens from Bag of Words that appear in less than 15 documents and more than 0.5 documents (that is the fraction of the total corpus size). This filter itself can filter out all polluters with some False Positive cases in The Fake Project dataset.

And we use Latent Dirichlet Allocation to infer the topic distribution of the documents. Then use Jensen-Shannon divergence to compute distances between topics derived from each user. Then we observe the distance of topics from a particular user, and the difference in average distribution between genuine users and bots.

LDA是一种主题模型，它可以将文档集 中每篇文档的主题以概率分布的形式给出，从而通过分析一些文档抽取出它们的主题（分布）出来后，便可以根据主题（分布）进行主题聚类或文本分类。同时，它是一种典型的词袋模型，即一篇文档是由一组词构成，词与词之间没有先后顺序的关系。

train LDA models using TF-IDF for each tweet corpus. And compute words occurring in each topic and its relative weight

TF-IDF是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。

To quantify these differences, we used LDAvis[5] and compute saliency（推文要点）[2] and inter-topic distance.

**Preprocess:**

* stop word removal,
* tokenization,
* Stemming and Lemmatization

手机屏幕截图

描述已自动生成表格

描述已自动生成

We extracted users’ statistical features, used PageRank and Community Detection method to analyze the social network structure, to separate polluters and genuine users, trained LDA models, and Bag of Words filter to find the divergence of topics from different types of users.

RF

**核心思想：**将若干个弱分类器的分类结果进行投票选择，从而组成一个强分类器，这就是随机森林bagging的思想（关于bagging的一个有必要提及的问题：bagging的代价是不用单棵决策树来做预测，具体哪个变量起到重要作用变得未知，所以bagging改进了预测准确率但损失了解释性。）

XGBoost

算法非常热门，是很多的比赛的大杀器，但是在使用过程中，其训练耗时很长，内存占用比较大。

微软在GitHub的上开源了LightGBM。该算法在不降低准确率的前提下，速度提升了10倍左右，占用内存下降了3倍左右。LightGBM是个快速的，分布式的，高性能的基于决策树算法的梯度提升算法。可用于排序，分类，回归以及很多其他的机器学习任务中。

### A Fatigue Detection Algorithm based on Eye Condition

I think this is my most noteworthy project. This is the outcome of the undergraduate research training program. This is the starting point of my research and ignited my passion for coding. Because I was still a freshman at that time, only knew basic C++ and data structure knowledge. I knew nearly nothing about machine learning, but I chose to participate in this competition because I want to challenge myself. The most challenging part is I have no idea how to start at the beginning, and by which method can I implement eye status detection. So, I read many many papers, tried many many prototypes, and discussed with my supervisor many many times. Finally, I tear the problem apart into 3 parts and conquered them one by one. The first step from 0 to 1 is always the hardest. After I know some image processing, computer vision and machine learning knowledge, finding and implementing an appropriate method became much easier. I spent a whole summer vacation and another semester on it – and finally completed it and won first prize. This made me convinced I really enjoy learning these unknown computer knowledge and I can do it well.

Dataset - ibug 300-W face landmark dataset.

HOG(Histogram of Oriented Gradient) algorithm – face detection

GBDT(Gradient Boosting Decision Tree) – landmarking of feature points

Boosting指把多个弱学习器相加，产生一个新的强学习器。经典的例子有：adaboost, GBDT, xgboost等。

**Boosting** is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models is added.

ERT: Ensemble of Regression Trees

LBF（Face Alignment at 3000 FPS via Regressing Local Binary Features）这篇论文也是基于Tree的人脸关键点检测算法。LBF是基于Tree的方法，学习每个关键点的局部二值特征，然后将特征组合起来，使用线性回归检测关键点。与 LBF 不同的是， ERT 是在学习 Tree的过程中，直接将 shape 的更新值 ΔS存入叶子结点 leaf node. 初始位置 S 在通过所有学习到的 Tree后， mean shape 加上所有经过的叶子结点的ΔS，即可得到最终的人脸关键点位置。

为训练每一级的 regressor， 文章采用了 gradient tree boosting算法减小 initial shape 和 ground truth 的平方误差总和。

Xgboost 和 GBDT 的区别：

GBDT：

GBDT 它的非线性变换比较多，表达能力强，而且不需要做复杂的特征工程和特征变换。

GBDT 的缺点也很明显，Boost 是一个串行过程，不好并行化，而且计算复杂度高，同时不太适合高维稀疏特征；

传统 GBDT 在优化时只用到一阶导数信息。

Xgboost：

它有以下几个优良的特性：

显示的把树模型复杂度作为正则项加到优化目标中。

公式推导中用到了二阶导数，用了二阶泰勒展开。（GBDT 用牛顿法貌似也是二阶信息）

实现了分裂点寻找近似算法。

利用了特征的稀疏性。

数据事先排序并且以 block 形式存储，有利于并行计算。

基于分布式通信框架 rabit，可以运行在 MPI 和 yarn 上。（最新已经不基于 rabit 了）

实现做了面向体系结构的优化，针对 cache 和内存做了性能优化。

PERCLOS – Measuring

Some studies have shown that nearly 30% of traffic accidents are caused by fatigue. Therefore, it is imperative to study high-accuracy real-time fatigue state detection methods. In this paper, the ocular status of fatigue and the feasibility of PERCLOS to measure eye fatigue has been analyzed from the physiological point of view. The face detaction and feature point calibration are performed by extracting the HOG features combined with a linear classifier, and the degree of eye fatigue in each frame is further quantified and calculated, and the current fatigue state is determined by synthesizing the detection results within one cycle.

**Where applied the Eigenvector?**

The eigenvector is applied in face detection part based on histogram of oriented gradients.

A HOG-based face detection method involves first extracting HOG-based features from input image blocks through sliding window, and then using a linear classifier to

determine whether the object is a face.

HOG forms feature through the histogram of local cells and blocks of static images. After an image is loaded, the gradient of the image is first calculated and then the image is divided into cells. Next, the histogram of these cells is calculated, and a certain number of cells are grouped into slightly larger regions, called blocks. The histogram obtained for each block is L2-normalized, and the eigenvectors containing the histograms from all of the blocks are grouped together to form HOG eigenvectors in the histogram of the image.

HOG：**颜色空间归一化—>梯度计算—>梯度方向直方图—>重叠块直方图归一化—>HOG特征**

将图像划分成若干个**cells**（单元），**8x8=64**个像素为一个**cell**，相邻的cell之间不重叠。在每个cell内统计梯度方向直方图，将所有梯度方向划分为**9**个**bin**（即**9**维特征向量），作为直方图的横轴，角度范围所对应的梯度值累加值作为直方图纵轴

### Research and Implementation of Finger Vein Image Regeneration based on Generative Adversarial Networks

An algorithm of using conditional generative adversarial networks and morphological processing to refine the finger vein Infrared image and an algorithm of regenerating the artificial finger vein image by a bi-level generative adversarial networks model group.

A refined finger vein sketch is obtained from the trained pix2pix original-to-enhanced map model of the finger vein. Then obtained by morphological processing such as binarization, morphological corrosion and skeleton extraction.

The bi-level Cascading pix2pix takes such a hand-drawn refinement finger vein image as input, and after morphological expansion, it passes through the trained binarization-reinforcement image model and the reinforcement-original image model. Then gets the final artificial finger vein original image from the network group output.

Skeleton is 1 pixel wide topological image

pix2pix is a cGAN using one-to-one corresponding pictures as condition

Generator: U-net

Discriminator: 70\*70 PatchGAN

Morphological Binarization and Morphological Corrosion

UTFVP Fingervein Database, from The University of Twente

**Why using generative adversarial networks instead of other generative models like VAE?**

Ian Goodfellow, the one invented GAN says, compared to all other models he can think of: In terms of actual results, they seem to produce better samples than other models.

* the GAN framework can train any kind of generator net (in theory—in practice, it is pretty hard to use REINFORCE to train generator nets with discrete outputs). Most other frameworks require that the generator net has some functional form, like the output layer being Gaussian. Essentially all of the other frameworks require that the generator net put non-zero mass everywhere. GANs can learn models that generate points only on a thin manifold that goes near the data.
* There’s no need to design the model to obey any kind of factorization. Any generator net and any discriminator net will work.

Compared to the VAE, there is no variational lower bound. If the discriminator net fits perfectly, then the generator net recovers the training distribution perfectly. In other words, GANs are asymptotically consistent, while the VAE has some bias.

GANs’ training paradigm is adversarial, so that we aren’t using maximum likelihood learning (maximizing some sort of log likelihood) but learning through an critic, a discriminator which nudges the generator to produce more realistic samples. The images produced by GAN are results of not using an artificial, user defined loss function (usually L1 or L2) but a loss that is learnt from the data with the discriminator.

But I think between VAE and GAN, there is no need to point out that one is superior to the other and creates bias when the jury is still out on making such definitive judgments. Sometimes it is even wrong to think of them as competing models. GANs can be also treated as a loss function, in the VAE-GAN paper(1706.04987).

**In pix2pix Model**

For generator, pix2pix add skip connections, following the general shape of U-Net. Specifically, pix2pix add skip connections between each layer i and layer n-i, where n is the total number of layers. Each skip connection simply concatenates all channels.

U-Net is also based on the encoder-decoder idea, extracting the target feature through four downsampling layers(conv 3\*3, ReLU and max pool 2\*2), then classify its pixels one by one through four upsampling layers(conv 3\*3, ReLU and up-conv 2\*2).

表格

描述已自动生成

Figure 1 Comparation between U-net and Autoencoder

As mentioned before, because the L2 and L1 loss used in VAE, produces blurry results on image generation problems. These losses will fail to encourage high frequency crispness and restricting the GAN discriminator to only model high-frequency structure. So the patchGAN is used as discriminator and divided the images into patches and this discriminator tries to classify if each patch in an image is real or fake. Then take the averaging response to provide the ultimate output in order to make better classification on part of an image.

提出一种利用条件生成对抗网络结合形态学处理，细化得到指静脉走向示意图和一种通过二级网络生成对抗网络模型组生成人造指静脉图像的方法，实现了通过随机拓扑示意图的输入生成对应人造图像的过程。其中，细化的指静脉走向图由经过pix2pix训练好的指静脉原图到强化图的模型得到对应强化图，再通过二值化、形态学腐蚀以及骨架提取等形态学处理后得到；二级网络生成对抗网络模型组以此类手绘的细化指静脉走向示意图作为输入，通过形态学膨胀后依次经过训练好的指静脉二值化图到强化图和强化图到原图的二级pix2pix网络输出得到最终的人造指静脉原图。

**为什么选择Unet？**

作者提到，输入和输出图像的外表面(surface appearance)应该不同而潜在的结构(underlying structure)应该相似，对于image translation的任务来说，输入和输出应该共享一些底层的信息，因此使用Unet这种跳层连接(skip connection)的方法，这里说的跳层连接是 [公式] 层直接与 [公式] 层相加，如下所示：

**为什么选择PatchGAN?**

如果使用AE和VAE，用L1和L2 loss重建的图像很模糊，也就是说L1和L2并不能很好地恢复图像的高频部分(图像中的边缘等)，但能较好地恢复图像的低频部分(图像中的色块)。为了能更好得对图像的局部做判断，作者提出patchGAN的结构，也就是说把图像等分成patch，分别判断每个Patch的真假，最后再取平均！作者最后说，文章提出的这个PatchGAN可以看成所以另一种形式的纹理损失或样式损失。在具体实验时，不同尺寸的patch，最后发现70x70的尺寸比较合适。

GAN其实是一种相对于L1 loss更好的判别准则或者loss。有时候单独使用GAN loss效果好，有时候与L1 loss配合起来效果好。在pix2pix中，作者就是把L1 loss 和GAN loss相结合使用，因为作者认为L1 loss 可以恢复图像的低频部分，而GAN loss可以恢复图像的高频部分。

### FleaFair SDU – A Personalized Campus Second-hand Trading Platform

基于用户的协同过滤算法(user-based collaborative filtering，简称：ucf)

Step 1

Build a user behaviour model to quantify the user behaviour and corresponding weights.

Step 2

Split the data collected and train the model. Apply model with some other rules.

Step 3

Establish reasonable metrics to measure the performance of the model. Analyse recall, precision, and coverage. It provides a basis for the further update.

**precision**measures the proportion of correct positive predictions.

**Recall**measures the proportion of actual positive labels correctly identified by the model.

第一阶段

建立用户行为评分权重模型，达到对用户行为数据化和可视化，。

以电子商品平台为例：

某用户进入商品下单页权重2%；点击详情权重8%；收藏15%；支付20%；分享15%；好评20%；评分20%；

差评即分数为负数(向量为反方向)。

第二阶段

建立测试集和训练集。

第三阶段

建立合理分析指标 监视召回率、准确率和覆盖率。为模型后期修正提供依据。

计算公式

Euclidean distance 欧几里德距离 公式

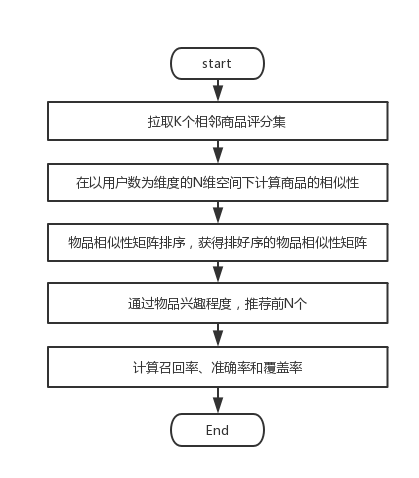
(为方便控制可取倒数Reciprocal，使结果分布在0—1之间)



余弦定理计算公式(N维空间)

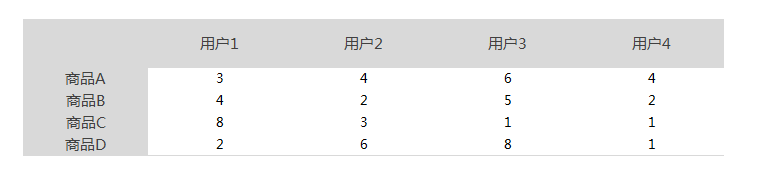
，Modulus: 模

计算流程



应用实例

【余弦公式和ICF为例】以用户实际评分为起点，建立商品评分矩阵(如下表)



通过计算4个用户(四维空间中)对4件商品的评分我们获得了用户间的相关性数据(如下表)。



系数浮动区间在-1~1之间，系数越靠近1，向量夹角越小，两件商品的相关性越高，由此可见A&B、A&D的相关性最高，C&D相关性很弱。

2、利用用户对某商品产生过的记录计算其相关性。

【例如】：某用户对商品A和商品B的行为得分为权重，对商品C和商品D进行加权排序，得分高者优先推荐。

根据相关性和加权评分后，商品C优先被推荐。

数据去噪修正

**Remove Super Popular**对大众化，一线流程产品进行剔除，原因是本来具有超高曝光率和知名度的产品，不推荐用户才能很快触达，不必进行不需要的推荐。

**Remove Repeated**对用户浏览和购买过的商品进行剔除，以免造成重复性。

**Categorize products**对商品分类，避免商品的跨类别推荐，造成用户并不需要此类商品。

【例如】：对某用户买衣服A，经过算法的综合排名，发现排第一的是方便面，排第六的才是衣服B，结果推荐了方便面岂不闹了笑话。但是对商品进行了归一分类，服装类商品只限推荐服装，这样服装B就会优先推荐。

对商品类别间建立合理的加分机制，并对低频商品建立合理的惩罚分值，使其推荐其他周边商品。

【例如】：家具类商品为低频商品，用户长时间内只需要买一件，购买后再次推荐也无法提升支付率。但是可以在用户支付下单后，通过计算，推荐家具的其它周边商品(例如：饰物、窗帘等)。由于设定合理的惩罚分值和相关商品类别的加分机制，可以一定程度上提高周边商品的推荐率，降低低频商品的推荐率，从而由侧面提升支付转化率。

方案弊端

数据稀疏性。由于此类协同过滤的模型需要有训练数据支撑，而在冷启动期间用户不会在数据模型中完成所有项目，所以数据会有稀疏性。

【解决思路】：可以按该类别商品的用户平均水平进行推荐(项目冷启动期间的方案待探讨)。

可扩展性。协同过滤算法能够容易地为千万记用户提供量化的推荐，但是对于电子商务网站，往往需要给成千上万的用户提供推荐，这就一方面需要提高响应速度，能够为用户实时地进行推荐;另一方面还应考虑到存储空间的要求，尽量减少推荐系统运行对系统的负担。

【解决思路】：划定计算范围，对无记录和类别相关性差距较大的商品、无操作记录的用户进行剔除性。由此减少计算压力。同时为提升用户体验，可以在离线期间对推荐数据进行训练和计算。但以上方案会在一定程度上影响到推荐的精确

#### 架构 / Architecture

Diagram

Description automatically generated

Controller层:Controller层负责具体的业务模块流程的控制，在此层里面要调用Serice层的接口来控制业务流程，控制的配置也同样是在Spring的配置文件里面进行，针对具体的业务流程，会有不同的控制器，我们具体的设计过程中可以将流程进行抽象归纳，设计出可以重复利用的子单元流程模块，这样不仅使程序结构变得清晰，也大大减少了代码量。

Service层：Service层主要负责业务模块的逻辑应用设计。同样是首先设计接口，再设计其实现的类，接着再Spring的配置文件中配置其实现的关联。这样我们就可以在应用中调用Service接口来进行业务处理。Service层的业务实现，具体要调用到已定义的DAO层的接口，封装Service层的业务逻辑有利于通用的业务逻辑的独立性和重复利用性，程序显得非常简洁。

DAO层：DAO层主要是做数据持久层的工作，负责与数据库进行联络的一些任务都封装在此，DAO层的设计首先是设计DAO的接口，然后在Spring的配置文件中定义此接口的实现类，然后就可在模块中调用此接口来进行数据业务的处理，而不用关心此接口的具体实现类是哪个类，显得结构非常清晰，DAO层的数据源配置，以及有关数据库连接的参数都在Spring的配置文件中进行配置。

View层 此层与控制层结合比较紧密，需要二者结合起来协同工发。View层主要负责前台jsp页面的表示，

### Load Balancer System, also include Reverse Proxy, Django Server

A **reverse proxy** is a web server that centralizes internal services and provides unified interfaces to the public. Requests from clients are forwarded to a server that can fulfill it before the reverse proxy returns the server's response to the client.

**How to implement Reverse Proxy:**

Use Django-revproxy module. Set up in settings.py. Then create a View that extends revproxy.views.ProxyView and set the upstrem attribute. Finally, add your view into url\_patterns in the urls.py

you should see the content of **(ProxyView**)http://example.com/ on **(Original URL)**<http://localhost:8000/>.

#### Load balancer vs reverse proxy

* Deploying a load balancer is useful when you have multiple servers. Often, load balancers route traffic to a set of servers serving the same function.
* Reverse proxies can be useful even with just one web server or application server, opening up the benefits described in the previous section.

#### Disadvantage(s): reverse proxy

* Introducing a reverse proxy results in increased complexity.
* A single reverse proxy is a single point of failure, configuring multiple reverse proxies (ie a [failover](https://en.wikipedia.org/wiki/Failover)) further increases complexity.

**Load Balancer**

Diagram

Description automatically generated

Can use: haproxy

Load balancers distribute incoming client requests to computing resources such as application servers and databases. In each case, the load balancer returns the response from the computing resource to the appropriate client. Load balancers are effective at:

* Preventing requests from going to unhealthy servers
* Preventing overloading resources
* Helping to eliminate a single point of failure

Load balancers can be implemented with hardware (expensive) or with software such as HAProxy.

# Question List

Tech Stack - improve my software engineering skills

How does the software engineering team plan projects?

### Can you describe how design decisions are made at your company?

(how collaborative it is and how the company resolves conflicts in the decision-making process.)

Ready for:

what was the most challenging work,how did you solve it,