* **Xgboost的推导过程：**

1. boosting都是前向分布算法：
2. 原目标函数：
3. 二阶泰勒展开：





1. 去掉以前固定结构的常数项：



1. 将目标函数表示为树结构：

令表示分到的叶节点，表示叶节点的权重，则：



1. 求目标函数的最小值，及叶节点的输出：

令：

求得：

将代入上式：

求得：

1. 由此得出打分函数：



1. 通过遍历所有特征的所有取值，寻找使得Gain最大时对应的分裂方式来构建K棵决策树，最终的预测结果是这K棵树预测的求和。

* **gbdt推导**

1. 初始化：
2. 训练1...K棵决策树：k=1...K
3. 计算负梯度：，其中i=1...m
4. 拟合叶节点区域：根据梯度残差数据集拟合回归树，得到叶节点区域，j=1...J
5. 每个叶节点的取值为：
6. 更新本轮强学习器：
7. 最终的GBDT模型为：

* **Adaboost的推导：**

1. 初始化样本的权值分布：
2. 循环K次，训练K个分类器
3. 在权值分布的数据集上训练基分类器
4. 计算分类误差：
5. 计算分类器的权重：
6. 更新下一轮的权值分布：



，其中

注：误分类的样本的权值更新是相同的

1. 加权投票法组合成最终的分类器



* **RF的推导**

1. 并行训练多棵决策树

对于每棵树的训练：

* 1. 用自助法进行样本采样
  2. 每次节点分裂，先随机采样K个特征，再从中选择1个最优特征。

1. 将每棵决策树的结果“投票判决”或“取平均”作为最终的预测。

* **xgboost原理**

Xgboost（或者说一般的boosting）是基于前向分步框架的算法，每一轮的强学习器是上一轮强学习器和当前基学习器的累加。我们希望训练一个基学习器使得整个强学习器的损失达到最小。

基学习器训练中最核心的点是决策树的分裂规则，这个分裂规则是基于打分函数确定的，具体是“gain=当前树的最小损失-节点分裂后的最小损失”，选择增益最大（损失减少最多）的节点进行分裂。接下来具体讲一下xgb的打分函数是怎么推导出来的。

**Xgb的损失函数：**首先xgb损失=原始损失+正则化项；原始损失函数是当前的y与强学习器的损失之和，正则化项为叶节点个数惩罚和叶节点权重惩罚。

**损失函数的变形：**再对这个损失函数做二阶泰勒展开，在这一步引入了一阶导信息和二阶导信息；然后把这个损失函数写成树结构的形式格式，具体是q(x)表示样本x分到的树节点编号，wq(x)表示该编号的节点的输出。

**基于该损失函数求极小值：**直接对损失函数求导，可以得到最小的损失以及对应的叶节点输出。基于刚刚推导的极小损失值，可以确定打分函数gain=当前树的最小损失-节点分裂后的最小损失

**树的分裂方法：**对于每棵树的分裂，遍历所有的特征和所有的取值，也即（特征，值）对，根据上面的打分函数，选择增益最大的（特征，值）来对数据进行切分（也即节点的分裂），并且得到分裂后每个节点的输出，按这样的方法一共训练n棵树；（xgb选(特征，值)的方法不是遍历扫描，而是取分位点处的值计算增益）

**树的集成：**最后对所有的树进行集成，具体为将每棵树的预测结果相加得到最终的值；如果是分类问题，将上面的损失函数换为对数似然损失，将最终的输出结果经过sigmoid函数映射为概率。

**其他的点：**

1. 可以设置缩减（也即学习率），让每棵树的输出×学习率，削弱每棵树的影响，而让树的棵树更多。
2. 对于防止过拟合，**xgb的调参**方式有：
   1. 设置行采样和列采样：设为0.9左右，让每个分类器有区别。
   2. 设置正则化参数（两个）：增大正则化参数，可以减低模型复杂度。
   3. 单棵决策树处理：剪枝，最大树深（减少），叶节点保留样本（增大）
3. 对缺失值，xgb可以自动学习分裂的策略
4. Xgb在实现上支持并行，这个并行不是指树的维度（依然要根据前一棵树才能训练下一棵树），而是在特征的维度，也即同时计算所有的（特征，取值）对，选择最佳的（特征，取值）对进行分裂。

* **Xgboost和GBDT的区别：**

1. GBDT以CART作为基学习器，而Xgboost可以支持CART和线性模型作为基学习器
2. GBDT在训练时只使用了损失函数的一阶导信息，只用到了一阶导信息；Xgboost对损失函数进行二阶泰勒展开，同时使用了一阶导和二阶导信息
3. Xgboost在目标函数里加入了正则化项（惩罚叶节点的个数与叶节点的取值），用于缓解过拟合。
4. GBDT在每一轮的训练时使用了全部数据，Xgboost支持列采样和行采样（“无放回”）
5. Xgboost能自动学习缺失值的处理策略。
6. Xgboost支持并行

7）Xgboost使用可并行的近似直方图算法，用于高效生成候选的分割点

其他比较重要的特点（GBDT和Xgboost都有）：

1. GBDT和Xgboost都采用Shrinkage（缩减），将每个预测值乘以一个系数，用于削弱每次预测的影响，一般将学习率设置小一点，将迭代次数设置大一点。

* **Xgb和lgbm的区别：**

1. 采用直方图算法
2. 直方图作差加速
3. Xgb同一层所有节点都分裂；lgbm每一层选取最大增益的节点分裂
4. 并行优化

* **GBDT与RF比较。**

相同点：

1）都是由多棵决策树组成，最后的预测结果，由多棵树一起决定

不同点：

1. RF可以由分类树组成，也可以由回归树组成；GBDT只由回归树组成，对于分类任务，GBDT采用对数似然损失函数，并且最终预测结果经过sigmoid函数转化为概率
2. RF并行生成，每棵树都是独立的；GBDT串行生成，每棵树都要依赖上一课决策树
3. RF提升的是模型的方差；GBDT提升的是模型的偏差
4. RF不容易过拟合，而GBDT容易过拟合
5. RF将每棵决策树的结果做投票（分类）或取平均（回归）作为输出；GBDT将每棵决策树的结果加起来作为输出（分类采用sigmoid转化为概率）
6. RF的行采样是有放回采样（自助采样），GBDT则是无返回采样

* **Adaboost对误分类样本的权值更新是否相同：**

**答：相同的。**

更新下一轮的权值分布：



，其中

从下面的公式中可以看到，权值公式只有和的区别：



* **Bagging和boosting的区别**

1. 样本的选择上：

Bagging：每个基分类器的训练集，是在原数据集中用自助法采样（有放回采样）得到

Boosting：每个基分类器的训练集都是原始数据集，但数据集的每个样本权重不一样

1. 样本的权重上：

Bagging：每个样本的权重相等。

Boosting：根据错误率调整样本的权重，分错的样本权重会更大

1. 基模型的依赖：

Bagging：各个基分类器无依赖，可以并行训练

Boosting：各个基分类器只能顺序生成，每一轮分类器的训练都依赖上一轮的预测结果

1. 模型集成：

Bagging：每个基分类器的权重相等

Boosting：每个基分类器都有相应的权重，分类误差率小的基分类器权重大。

1. 性能提升：

Bagging：通过消除方差来提升模型性能。

Boosting：通过消除偏差来提高性能。

* **集成模型（Xgboost，GBDT，RF）的优点：**

1. 不需要对连续变量进行归一化
2. 不需要对类别变量进行one-hot编码
3. 不需要做特征选择
4. 可以输出特征重要性

* **DT的推导**

信息增益：



信息增益比：



基尼指数：



最小二乘：



Id3：根据“信息增益最大”原则选择特征，对于每个数据子集，用选出的特征对数据进行划分。最终的输出为叶节点最多的类

C4.5：根据“信息增益比最大”原则选择特征

CART：

分类：根据“基尼指数最小”原则选择特征和最优分割点，对于每个数据子集，用选出的特征和对应的分割点对数据进行划分。最终的输出为叶节点最多的类

回归：根据“最小二乘”原则选择特征和最优分割点，最终的输出为叶节点的均值。

* **决策树怎么处理连续值（C4.5，CART）：**

1. 对该特征进行排序
2. 所有可能的划分点是“相邻取值的中点”：
3. 用“Gain、Gini、最小二乘”准则做遍历，选出最佳的特征

* **决策树怎么剪枝**

1. **预剪枝：**设置验证集，在决策树的生成过程中，如果节点分裂后的泛化性能没有提升，则该节点不再分裂并设置为叶节点。
2. **后剪枝：**设置验证集，先生成整棵决策树，自底向上对于每个子节点，如果剪枝后的泛化性能提升，则剪枝，也即该节点设置为叶节点。

* **LR的推导：**

模型：，其中：

损失函数的推导：











注：这是一个凸函数，存在有最优解

梯度下降法求解参数：



* **LR为什么要对连续特征进行离散化：**

1. 逻辑回归属于线性模型，表达能力受限；单个变量离散化为N个后，每个变量有单独的权重，相当于为模型引入非线性（可以从分段函数分析），提升模型的表达能力
2. 离散化可以做交叉特征，如特征A离散化为M个，特征B离散化为N个交叉后可以达到M\*N个特征；（特征的升维类似于核函数的特征映射）
3. 离散化后对异常样本有很强的鲁棒性：不会因为一个很大的异常值而对模型由很大的干扰
4. 离散化能够降低LR的复杂度，能避免过拟合：因为一个特征的权重很大，模型就会很容易过度依赖这个特征，这个特征的微小变化会对模型产生重要的影响；特征离散化之后，特征的权重会变为多个，连续特征对模型的影响力被分散；因此缓解过拟合；
5. 离散化后存储为稀疏向量，稀疏向量的内积计算快，存储方便，易于扩展

* **SVM的推导**

**一、线性可分：**

1. **模型：**，
2. **间隔：**
   1. 函数间隔：
   2. 几何间隔：
3. **构造原始优化问题：**



1. **构造对偶优化问题：**

拉格朗日函数：；其中

先求极小：

令偏导为0：







再求极大：





1. **SMO算法求解模型参数：**
2. **参数还原：**
3. **最终的对偶支持向量机：**，其中
4. **带松弛变量和惩罚参数：**

约束条件带松弛变量，目标函数带惩罚因子



1. **非线性支持向量机与核函数**

用核函数表示特征映射后的内积：

* **Svm和逻辑回归的区别：**

1. 相同点：

都是线性分类器，分离超平面都是

1. 不同点：
2. 根本的区别：LR使用“log loss”，svm使用“hing loss”；这导致优化的时候，LR计算损失的时候考虑了全部的样本点（对全部样本做优化），而svm计算损失的时候只考虑了作为支持向量的样本点（只对支持向量做优化）。

Log loss：

Hing loss：

1. 二分类的时候，LR通过输出的概率和设定的阈值进行分类，可以直接得到预测为正类概率；SVM直接输出分离超平面，需要进行距离归一化才能输出概率。
2. LR可以使用softmax直接扩展为多分类器，而svm扩展到多分类需要使用ovo和ovr的策略。
3. Svm可以使用核函数解决非线性问题；LR应该不能使用kenerl吧！

* **SVM有哪些常用的核函数，讲一下这些核函数**

线性核：

多项式核：——将原始特征空间映射到更高维

高斯核：——将原始特征空间映射到无穷维

拉普拉斯核：

Sigmoid核：

非线性SVM的一个根本假设是，在原始特征空间的数据如果不能线性切分，可以对“原始特征空间”经过一个高维映射，再在这个“映射特征空间”做线性切分。

但这样会存在两个问题，一是这个映射不是那么好找，二是即使找到了映射，这个“映射特征空间”的维度也很高，计算量非常大。

在对偶支持向量机中，推导出了两个特征向量的内积，核函数可以直接表示这两个向量映射后的内积，而不必显式地定义映射；这样就相当于学习出了非线性支持向量机。

**多项式核函数的原理：**将原始特征空间映射到更高维

假设输入二维向量，多项式次数：

，



可以还原函数的映射：

也即输出从2维映射到3维：，

**高斯核函数的原理：**将原始特征空间映射到无穷维

假设输入二维向量，且：

，



而根据泰勒展开式：，因此：



从上可以看出，高斯核函数是无限个不同阶的多项式核函数的累加，每个多项式核函数能映射到有限的高维，无限个多项式就映射到了无穷维。

* **SVM调参：**

SVM的主要参数为：惩罚系数C，高斯核函数参数，多项式核函数参数；

选择高斯核：C越大，越小，模型越复杂，越容易过拟合。

选择多项式核：C越大，越大，模型越复杂，越容易过拟合。

* **二分类怎么处理为多分类（SVM如何扩展到多分类）：**

1. **OVO：**将“两个类别的数据”两两组合，训练n(n-1)/2个模型（每个模型的数据为两类的数据）。最终的预测可以是投票决定，也可是将每个类别的概率加起来取概率最大的类别
2. **OVR：**取其中1类的数据作为正样本，其余数据作为负样本，训练n个模型（每个模型的数据为全量数据），最终的输出为概率最大的类别。

* **NB的推导**

一句话说一下NB：取后验概率最大的类作为预测。

联合概率与边缘概率：

贝叶斯公式：

简版：

详细版：

注意：是条件概率，是后验概率

朴素贝叶斯公式：



朴素贝叶斯模型：

参数估计：

1. 
2. 

* **讲一下朴素贝叶斯**

朴素贝叶斯模型的预测方法是，给定输入特征，计算预测为每个类别的概率，最后取最大概率的类别

**朴素贝叶斯模型：**根据贝叶斯原理，，其中分母对于所有的类别是一样的，可以去掉；再根据朴素贝叶斯的基本假设——条件独立假设，也就是认为特征与特征之间是没有关联的；可以把条件概率分解为特征的每个维度的条件概率之间的连乘，，这样模型就分解为两部分的参数。

**参数的训练：**求解这些参数的过程可以直接根据数据进行频率统计，，（y给定的条件下，特征的每个维度的每种取值的概率）

**预测过程：**对于新的输入样本，将特征分解为每个维度当前取值的概率的连乘，直接按照上面的统计结果代入决策公式计算便可

* **KNN流程：**

距离计算：

曼哈顿距离：

欧式距离：

1. 计算测试样本与所有训练样本的距离
2. 选择距离最小的K个训练样本
3. k个训练样本投票输出，或均值输出

* **KNN的三要素：**

k值的选择、距离度量、分类决策规则

* **聚类：**

**常用的聚类算法有哪些：**

k均值聚类

谱聚类

层次聚类

密度聚类

* **Kmean的过程描述**

1. 随机选定K个样本作为聚类中心
2. 遍历所有的样本，计算每个样本到所有聚类中心的距离，将每个样本归到最近聚类中心的簇。
3. 遍历所有的簇，将每个簇的均值向量作为新的聚类中心。
4. 重复2) ~ 3)步，直到均值向量不再被更新。

Kmean++对初始化聚类中心的处理：

1. 先随机选择一个样本作为聚类中心
2. 计算每个样本到“已选出的聚类中心的最小距离”；余下的样本被选择为下一个聚类中心的原则为“取根据距离越大，被选取的概率越大的原则”
3. 重复2)，直到选出K个聚类中心。

* **手写kmean的代码：这个很吊哦！！！**
* **Kmean聚类怎么选择k值**

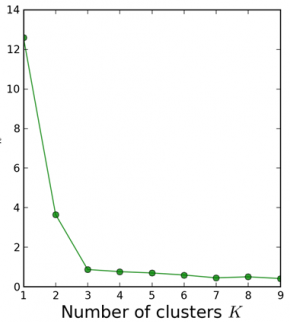
1. 手肘法：

核心指标数：误差平方和SSE；（表示所有聚类的误差，表示聚类的好坏）



其中，是第k个簇，p是属于的样本点，是的聚类中心。

手肘法的核心思想是：随着聚类数k的增大，样本划分会更加精细，每个簇的聚合程度会逐渐提高，那么误差平方和SSE自然会逐渐变小。并且，当k小于真实聚类数时，由于k的增大会大幅增加每个簇的聚合程度，故SSE的下降幅度会很大，而当k到达真实聚类数时，再增加k所得到的聚合程度回报会迅速变小，所以SSE的下降幅度会骤减，也就是说SSE和k的关系图是一个手肘的形状，而这个肘部对应的k值就是数据的真实聚类数。



1. 轮廓系数法：

* **降低过拟合和欠拟合风险的方法：**

过拟合是指模型在训练集上表现很好，但在测试集上表现较差。欠拟合是指模型在训练集和测试集上的表现均不好

过拟合的原因：

1. 样本的噪声数据干扰过大，大到模型过分记住了噪声的特征，而忽略了真实的输入输出关系。
2. 模型的复杂度过高，导致过分学习了“局部规律”，而忽略了“普遍规律”

过拟合的办法：

1. 增加训练数据（最有效的方法）
2. 降低模型复杂度：
3. 采用复杂度低的模型（如LR，DT）
4. 设置正则化
5. 控制每棵树的树深（tree base model）
6. 每棵树设置叶子节点最小样本个数（tree base model）
7. 剪枝（tree base model）
8. 设置列采样（xgb）
9. 设置神经网络隐藏层神经元的个数，层数（DNN）
10. 神经网络加drop\_out（DNN）
11. 训练提早停止（xgb，神经网络）
12. 采用集成学习的Bagging方法（RF）

欠拟合：

1. 增加新特征
2. 增加模型复杂度
3. 减小正则化系数

* **偏差和方差**



**偏差：**度量算法的期望预测与真实结果的偏离程度，刻画学习器本的拟合能力

**方差：**度量训练集的变化导致学习性能的变化，刻画数据集扰动造成的影响

* **为什么需要对数值类型的特征做归一化：**

归一化是指将所有的特征都统一到一个大致相同的区间内，常用的方法主要有两种：

1. 线性归一化（最大最小归一化）：

将原始数据映射到[0，1]的范围内，实现等比例缩放



1. 零均值归一化：

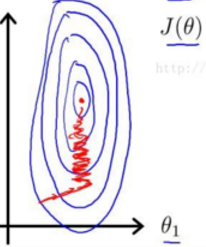
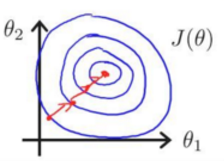
将原始数据映射到均值为0，标准差为1的分布上。

，其中为均值，为标准差

**对数值型归一化的原因是：**

没归一化之前，代价函数“等值图”的轮廓会是“扁长的”，在找到最优解前，梯度下降的过程是曲折的，训练耗时且不易收敛。

进行了特征归一化以后，代价函数“等值图”的轮廓会是“偏圆”的，梯度下降过程更加笔直，训练快且易收敛。

需要注意的是，一般用梯度下降法的模型需要归一化，如线性回归，逻辑回归，支持向量机，神经网络等；而**决策树模型是不需要归一化**的，因为决策树的节点分裂主要考察（特征，取值）的信息增益，而对特征进行归一化不会改变特征在各分割点处的信息增益

* **类别型特征的处理**

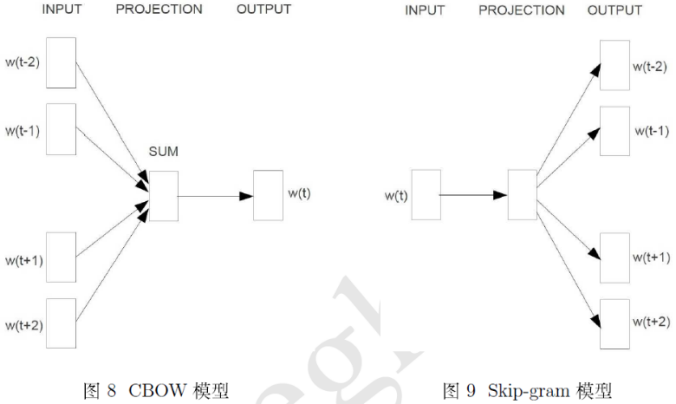
1. **序号编码：**处理类别间具有大小关系数据
2. **One-hot编码：**类别间无大小关系
3. **二进制编码：**

* **特征选择的方法**

1. **过滤式：**先对原始特征进行“过滤”，在用过滤后的特征来训练模型。如信息增益法，卡方检测法，皮尔森相关系数法等
2. **包裹式：**以最终使用的学习器的性能作为评价准则来搜索特征子集。如前向搜索，后向搜索，特征递归消除
3. **嵌入式：**将特征选择的过程与学习器的训练过程融为一体，即在训练的过程中自动完成了特征选择。如L1正则化，树模型特征选择

**卡方检测：**主要是比较两个及两个以上样本率( 构成比）以及两个分类变量的关联性分析。其根本思想就是在于比较理论频数和实际频数的吻合程度或拟合优度问题

* **Word2vec原理**



这个图只是说了一个大概意思，并没有体现模型的细节。

**Word2vec基本思想是：**把一个词映射成一个固定维度的向量，映射后这些词如果语义相近则向量夹角小，如果语义相反则夹角大。

**CBOW：**根据上下文的词预测当前词的生成概率

**Skip-gram：**根据当前词预测上下文各词的生成概率

实践表明，CBOW在小型语料中表现更好，而Skip-gram在大型语料中表现更好

* **Tfidf的公式及其表示的含义**

（注：tf为词频，idf为文档逆频率）





或：

**TFIDF的思想：**一个词在本文章中出现的频率越高，而在其他文章中出现的频率越低，则认为这个词具有很好的区分能力。

+1主要是为了平滑，防止某些生僻词没有出现在语料库中，而导致分母为0。

* **L1，L2正则化的区别（常问的题目）**

1. Lsso回归：

岭回归：

1. 相同点：

毫无疑问，L1，L2正则化都可以用来降低过拟合的风险，

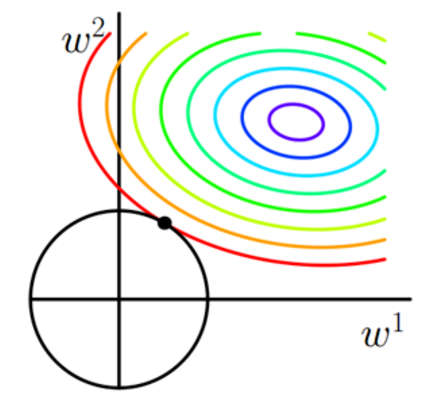
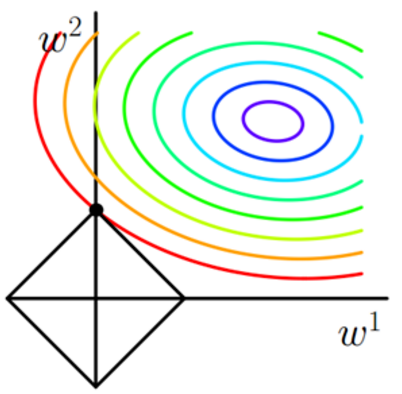
1. 不同点：

L2正则化会让模型所有的参数都变小，但一般不会变为0，以此构造一个更简单的模型；

L1正则化很容易获得“稀疏解”，也即w中很多项为0；相当于将不重要的特征的权重置为0，实现了特征选择。

1. L1、L2产生这种解的原因解释：

假设特征是二维的，那么对应的参数也是二维的，即和；以和为坐标轴，的等值线是图中的彩色圆，的等值线是黑色的圆，的等值线是一个菱形；整个损失函数要在“平方误差项”与“正则化项”之间折中，即的取值要出现在两个等值线的交点处；从坐标轴可以看出，L1正则化的解很容易出现在坐标轴处，即或为0；而L2正则化的解一般出现在某个象限中，也即和不为0；因此L1正则化具有“稀疏性”

* **梯度下降法：**

1. **梯度：**对多元函数求偏导，并以向量的形式展示出来（注意，多元函数的梯度是向量）；也即：；在梯度的方向上，函数增加得最快，在负梯度的方向上，函数减少得最快。
2. **梯度下降法的描述**：
3. 随机初始化所有参数
4. 对于每个参数：；

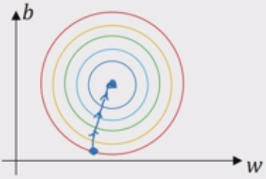
注：每次的所有的参数要一起更新，这样才能表示往负梯度的方向更新

1. 重复2），直到所有参数的更新都小于；输出最终的参数。
2. **梯度下降法的解释**：

求函数的“极小值点”，可以转化为求的解，但这个解通常是不好直接求的，因此考虑通过迭代的方法来求解。

因为往负梯度的方向是函数值下降最快的方向，也就是每次往负梯度的方向走一小步，可以最快地收敛到。因此GD思想为：对于参数，每次往负梯度的方向走一小步，直到梯度为0；

更新公式为：，其中表示负梯度，表示每次更新的步长。



1. **梯度下降法家族**：

标准梯度下降：使用全部样本更新参数

随机梯度下降：随机使用一个样本更新参数

小批量梯度下降：使用一小部分数据更新参数

* **牛顿法：**

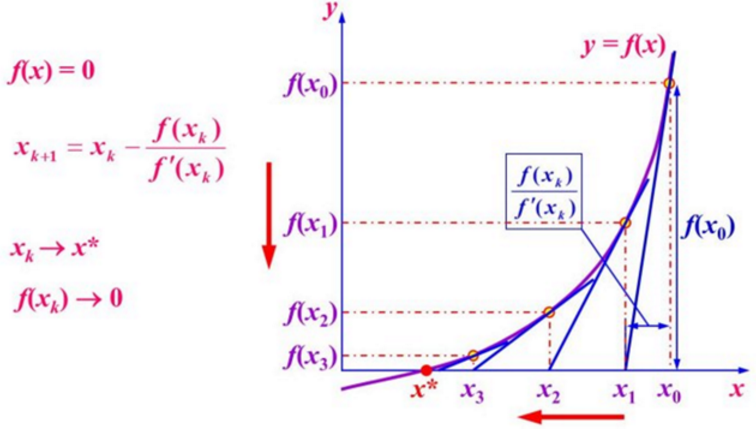
求函数的“极小值点”，可以转化为求的解；令，“极小值点”为的解。

对进行一阶泰勒展开：

，也即：

求得：，也即

写成迭代的形式：



上面的式子是一维的情况（因为只有一维才能直接除以二阶导），牛顿法也适用于高维的情况，此时泰勒展开式为：，迭代公式为：

其中是梯度向量，是海森矩阵的逆。





* **拟牛顿法：**
* **最小二乘法的描述：**
* **梯度下降法、牛顿法、拟牛顿法的原理：**

这三个算法都是求极小值点。

**梯度下降法：**对于函数，参数沿着负梯度的方向更新，能使函数值减少最快，因此GD的思想是：参数每次往负梯度的方向更新一小步，直到的梯度接近0；也即：

**牛顿法：**将求解极小值点的问题，转化为求的解；将在初始点处做1阶泰勒近似，解方程求出，作为下次迭代初始点，也即：求方程的解，解得：，如此循环，直到几乎不更新。（如图中不断用切线做与X轴的交点，逐步逼近曲线与x轴的交点）

**拟牛顿法：**牛顿法的缺点是在高维向量的求解中，需要求解海森矩阵的逆，计算复杂度很高；主要的想法是用一个矩阵来近似代替，

具体的过程很复杂，真的不会了啊！！！

* **说说常见的算法，以及优缺点：**

1. 批量（标准）梯度下降：

优点：收敛速度快

缺点：容易陷入局部最小点

1. 随机梯度下降：

优点：在一定程度上课解决局部最小值问题

缺点：收敛慢

1. 小批量梯度下降：

结合批量和随机的优缺点，做综合衡量

1. 牛顿法：

优点：收敛更快

缺点：每一步迭代，都需要计算海森矩阵的逆，计算复杂度高

1. 拟牛顿法：

改进迭代过程中的求解，通过用一个矩阵来逼近或实现

* **如何处理数据不平衡：**

**训练：**

1. 过采样：从少数类的样本集中，“有放回”地随机重复采样，以得到更多的样本
2. 欠采样：从多数类的样本集中，随机采样较少的样本。
3. 设置类别权重：通过改变模型训练时的目标函数来校正样本不平衡（如代价敏感学习中不同类别有不同的权重，直接在模型里设置class\_weight参数）

**评价指标：**

F1，AUC

* **特征值与特征向量的计算**

，求A的特征值和特征向量

1. 原始方程：



1. 求解特征值：



1. 求解特征向量：

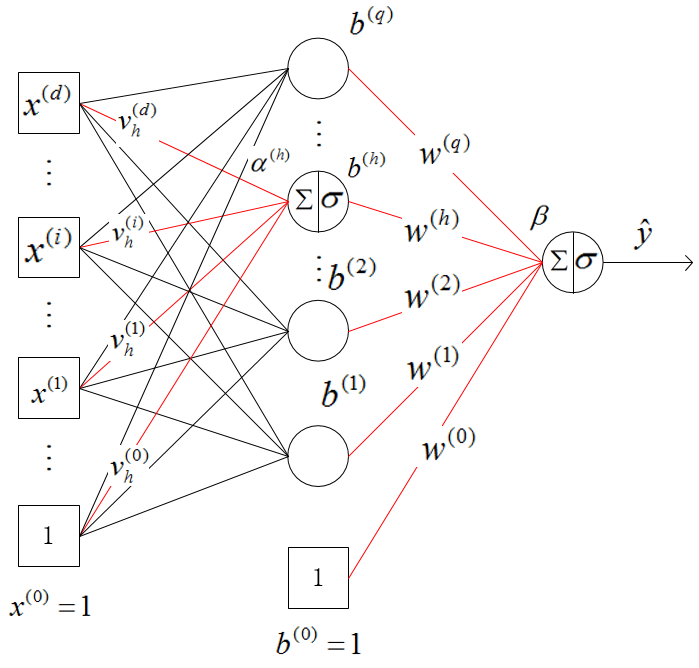
对于，带入原始方程为：，化简为；

解得

对于，带入原始方程为：，化简为；

解得

* **神经网络的BP过程：**

****

1. **损失函数：**
2. **BP算法：**

****

****

****

****

****

****

****

****

****

****

****

* **RNN的BPTT算法推导：**

****

1. **损失函数：**

****

1. **BPTT算法：**

****

****

****

****

****

以长为3的RNN为例（写成整体公式真的太麻烦了）：

****

****

以长为3的RNN为例：

****

* **激活函数：**

1. **Sigmoid函数的作用**





作用：将[-∞, +∞] 的输入，映射到[0, 1]之间，表示判别为“正类”的概率。

1. **Softmax函数的作用**





作用：将输出归一化到[0, 1]之间，表示每个类别的概率

1. **Relu：**隐含层最常用的激活函数



1. **Tanh：**

* **神经网络为什么会出现梯度消失和梯度爆炸，以及如何解决：**

**梯度消失和梯度爆炸出现的原因：**





神经网络的训练是采用方向传播算法，核心是链式求导法则，如上图，如果采用sigmoid函数，最终化简为导数与权值的乘积，但是sigmoid导数函数的图像是拱型，最大值为0.25；

**梯度消失：**此时如果初始化的时候权值小于1，（如0均值初始化——均值为0，标准差为1）；那么上式就变成一些列“小于0”的数的连乘，如果网络比较深，连乘的结果（梯度）就接近于0了。

**梯度爆炸：**如果初始化的时候，w过大，导致，网络很深后，连乘的结果（梯度）就会变得很大。

**梯度消失的解决：**

1. 采用relu激活函数（及其变种如Leaky-ReLU、P-ReLU、R-ReLU）代替sigmoid函数
2. 使用LSTM（相比于传统的RNN）
3. 采用批量归一化（batch\_normalization）

**梯度爆炸的解决：**

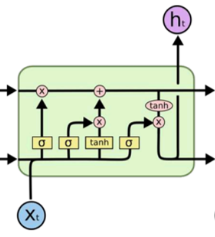
1. 使用 ReLU 激活函数
2. 使用LSTM
3. 重新设计网络模型
4. 使用梯度截断

* **RNN中为什么会出现梯度消失或度爆炸，LSTM是如何解决这个问题的：**

RNN是根据序列建模的网络，序列越长，网络的深度越深，从而对前面结构的权值更新越不明显（梯度消失），或者更新很大（梯度爆炸）。

LSTM对RNN的改进，最重要的两点是“避免长期依赖”和“梯度消失”的问题

* **讲一下LSTM**



LSTM也是一个处理序列问题的模型，它可以较好地解决RNN的“长期依赖”和“梯度消失”的问题。它由一个个cell组成，将序列每个时刻的信息依次输入到每个cell，来进行训练和预测。

主要讲一下cell的内部结构；一个cell由“3个门（遗忘门，输入门，输出门）”，“细胞状态”，“输入信息”，“输出信息”组成，门的作用是对信息进行抑制。具体的过程是： **第一步**，遗忘门连接到细胞状态，表示将不重要的信息丢弃；

**第二步**，确定当前步需要新加入的信息，首先是“当前输入和上一步的输出”经过线性组合并使用tanh激活得到候选信息，再经过输入门的抑制，得到当前的输入信息

**第三步**，将信息合并到细胞状态中：丢弃需要遗忘的信息，新加入当前的信息

**第四步**，基于细胞状态得到当前的输出：细胞状态经过tanh激活，再经过输出门的抑制，得到当前步的最终输出。

* **卷积核输出的维度的计算公式**

输入的维度：W\*W

卷积核的维度：F\*F

步长：S

Padding（补齐）：P

输出的维度：N\*N

N = (W - F + 2P) / S + 1

* **神经网络如何解决陷入局部极小值：**

1. **以多组不同的参数初始化多个神经网络**，训练之后取误差最小网络作为最终的网络。（其实每个网络都会陷入不同的局部极小，这样是为了获得接近全局极小的结果）
2. **使用“模拟退火”算法**。模拟退火算法每一步都以一定的概率接收比当前更差的结果，从而有助于“跳出”局部极小。类似的还有**“遗传算法”**。
3. **使用随机梯度下降**。SGD的过程中，即使陷入了局部极小，它计算出的梯度仍然可能不为0，这样就有机会跳出局部极小而继续搜索。
4. **使用带动量的梯度下降**。陷入局部极小后有机会跳出局部极小值。

* **Dropout过程的具体描述；为什么可以解决过拟合**

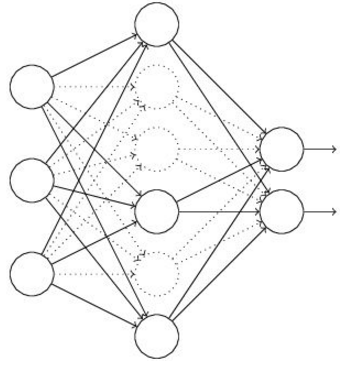
**训练过程：**

1. 随机“冷冻”N×p个隐藏层的节点（冷冻：节点的参数保持不变，但不参与计算；N是隐藏层节点个数；drop\_out=p）
2. 对于新的网络，在“小批量数据集”上经过前向传播和反向传播更新权值w
3. “解冻”所有节点。
4. 重复1) ~ 3)，直到网络收敛

**预测过程：**

使用全网络做预测，但隐藏层的参数要乘以概率p。





**Drop\_out为什么可以解决过拟合：**

drop\_out的过程相当于训练了“多个不同结构的网络”，然后对所有网络的输出取平均，类似于bagging的效果；每个网络会产生不同的过拟合，一些“反向的过拟合”可以相互抵消，就能实现整体上的减少过拟合

* **什么是生成模型，什么是判别模型**

**生成模型**由数据学习出联合概率分布，然后通过边缘概率，求出条件概率分布。即

**判别模型**直接由数据学习出决策函数或条件概率分布。

* **模型评估的方法**

1. 留出法：70%的数据作训练集，30%的数据作验证集
2. 交叉验证法：
3. K折交叉验证：将所有样本划分为大小相等的k份，遍历所有子集，的将当前子集作为验证集，其余子集作为训练集，最终的结果是对k次评估取平均。
4. 留一验证：遍历所有样本，每次将1个样本作为验证集，其余样本作为训练集，最终结果为对所有评估取平均（是k折交叉验证的特例，此时k=m；适用于样本量很小的情况。）
5. 自助法：对原始数据集“有放回”地采样m次，得到大小为m的训练集（存在重复）；测试集为没有被采样到的样本（约有36.8%的数据）；适用于样本量小的情况。

* **稀疏矩阵是怎样存储的，实现一个稀疏矩阵相乘的算法（1、转化为稀疏矩阵，2、稀疏矩阵的乘法）**

1. 三元组表示法：，分别存储“行号，列号，数据”
2. 行逻辑连接顺序表：
3. 十字链表：

* **文本分类的基本流程：**

1. 数据预处理：除去垃圾词汇
2. 分词
3. 去停用词
4. 提取特征：
5. 单词的维度（即一个单词表示一个特征，一个句子表示一个样本）：

序号编码，one-hot编码，TF-idf加权，n-gram特征，word2vec特征

1. 构造分类器：LR，SVM，lstm，cnn，textcnn等

* **推荐系统：**

**评测指标：**

1. 预测准确度：
2. 针对评分预测：

RMSE：

MSE：

1. 针对topN预测：

精确度：

召回率：；其中R(u)表示推荐列表；T(u)表示行为列表

1. 覆盖率：
2. 多样性：

**协同过滤算法：**

怎么简洁说一下协同过滤：

协同过滤是推荐系统的经典算法，分为“基于用户的协同过滤”和“基于物品的协同过滤”；

基于用户的协同过滤：向目标用户推荐“与其相似的用户”喜欢的物品。

1. 找出与目标用户相似的k个用户。
2. 计算目标用户对这K个用户喜欢的物品的感兴趣程度
3. 按感兴趣程度排序，推荐给目标用户。

基于物品的协同过滤：向目标用户推荐“他喜欢物品”的相似物品”。

1. 找出“目标用户喜欢的物品”的相似物品
2. 计算目标用户对相似物品的感兴趣程度
3. 按感兴趣程度排序，推荐给目标用户

说实话，《推荐系统》那本书写得那么复杂，其实不就只是将“用户——物品评分矩阵”填写完整，然后topM推荐吗嘛

1. 基于用户的协同过滤：
2. 找出与目标用户相似的k个用户。

直接计算目标用户和所有用户之间的相似度，然后取TopK就可以。

1. Jacard相似度：
2. 余弦相似度：；

式中，表示用户u与用户v之间的相似度；表示用户u发生行为的物品向量，表示用户v发生行为的物品向量；N(u)表示用户u发生行为的物品集合，N(v)表示用户v发生行为的物品集合

1. 计算目标用户对这K个用户喜欢的物品的感兴趣程度

衡量用户u对物品i的感兴趣程度为：，其中表示k个相近的用户中，对物品i产生过行为的用户。表示用户u和用户v的相似度，表示用户v对物品i的兴趣程度（这里所有的）。

1. 按感兴趣程度排序，推荐给目标用户。
2. 基于物品的协同过滤：
3. 找出“目标用户喜欢的物品”的相似物品
4. ，其中表示对物品i有过行为的用户数量，该式表示物品i和物品j都喜欢的人数与喜欢物品i的人数之比（如果i和j相似的话，和应该差不多），但这样的表示也有问题。
5. （其实就是余弦相似度）
6. 计算目标用户对相似物品的感兴趣程度：

衡量用户u物品j的感兴趣程度为：，式中，用表示u产生过行为的物品中，与物品j相似的K个物品。

1. 按感兴趣程度排序，推荐给目标用户

* **分类模型的评价指标有哪些，做简要说明**

准确率（acc）：

正确率、召回率（precise，recall）：

F1：

Auc：

RME：

RMSE：

* **相似度计算的公式有哪些？写出具体的计算公式**

**欧式距离：**

**曼哈顿距离：**

**闵可夫斯基距离：**

**余弦相似度：**

**Jaccard相似度：，注：是布尔值的向量**

**皮尔森相关系数：**实在太麻烦了，不想搞了！！

* **马氏距离较欧式距离的优点：**

1. 尺度不变性；
2. 考虑了模式的分布

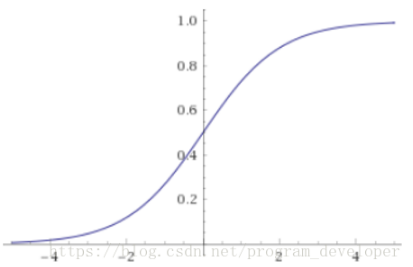
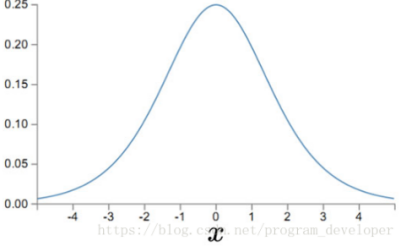
* **逻辑回归在什么场景下更合适：**

在高维特征的情况下更合适，因为逻辑回归是线性模型，是用一个超平面对特征空间进行切分，一般来说，数据在高维的情况下更容易线性可分（低维用非线性模型，高维用线性模型）

* **Sigmoid函数的解释：**

1. 常说的sigmoid函数：

1. 逻辑斯蒂分布：

设x是连续随机变量，x服从逻辑斯蒂分布是指具有如下的分布函数：



其中，u是位置参数，为形状参数；是对称中心

* **数据预处理有哪些步骤：**

**缺失值**：

1. 离散：NAN，或者纵数
2. 连续：均值
3. 缺失太多的行或列：去掉

**连续值**：离散化：等宽，等频，一维聚类；

二值化：设置阈值

**等距（等宽）：**数据从小到大排序，从最小值到最大值之间，平均分成N个长度一样的区间，

**等频（等深）：**数据从小到大排序，先确定N个区间，让每个区间包含相等的实例个数。

**1维聚类：**

* **如何发现异常值：**

异常值是指那些明显偏离其余值的观测样本。

1. 直接根据业务进行统计分析：

比如，对一个变量设置一个阈值，超出阈值就是离群点。

1. 箱型图分析：

异常值定义为：小于，或大于的值；

其中为下四分位数，表示全部的观察值中有1/4的数据比该值小；

为上四分位数，表示全部的观察值中有1/4的数据比该值大；

表示上四分位与下四分位数之差。

四分位差：反映了中间50%数据的离散程度，其数值越小，说明中间的数据越集中；其数值越大，说明中间的数据越分散

###sql#################################################################

* **几个sql语句的执行顺序：**

其中select和from是必须的，其他关键词是可选的，这六个关键词的执行顺序与sql语句的书写顺序并不是一样的，而是按照下面的顺序来执行

from--join on--where--group by--having--select--distinct--order by,

From：需要从哪个数据表检索数据

Join... on：根据条件，进行表的连接

Where：按条件过滤表中数据

group by：将上面过滤出的数据分组

Having：对上面已经分组的数据，按条件进行过滤

Select：查看结果集中“选取的列”，或“选取列的计算结果”

Distinct：将重复的数据删除

order by：对返回的结果进行排序

* **常用的sql关键字：**

###概率题######################################################

考虑两队之间的足球比赛：队0和队1.假设65%的比赛队0胜出,剩余的比赛队1获胜.队0获胜的比赛中只有30%是在队1的主场,而队1取胜的比赛中75%是主场获胜.如果下一场比赛在队1的主场进行队1获胜的概率是多少?

**贝叶斯解法：**

设X：东道主球队；Y：获胜球队

；

；

因此：



**频率解法：**

设进行了100场比赛，

Count(队0胜) = 65，Count(队1胜) = 35

Count(队0胜，队1主场) = 65 \* 0.3 = 19.5

Count(队1胜，队1主场) = 35 \* 0.75 = 26.25

因此：



###算法编程######################################################

编程题数据输入：

数组：

arr\_tmp = input().split(',')

arr = []

for e in arr\_tmp:

arr.append(int(e))

单个数：

n = int(input())

###开放性####################################################

* **讲一下实习的项目：**

这个项目是讲：

数据来源：15家医院的，基础数据，体检数据，用药数据，疾病数据，icd数据，

数据预处理：数据的存储格式，命名，日期，数据选取等。

特征工程：记录信息整理成特征输入到模型。

基础——年龄，性别，科室，日期，住院天数等。

用药——滑窗统计，用药次数，用药种类，每重要的次数，种类等

体检——均值，最大，最小值的统计

疾病——19种疾病 + charlson评分

Icd——统计造影剂的次数

建模：LR，xgboost，lightgbm等

评估：auc，混淆矩阵，F1

* **讲一下数据分析做了哪些工作：**

1. 游客量预测：
2. 细粒的时间上，每天游客量有所有不同：因此将时间拆分更细，（原来的时间只是一个年-月-日的日期）
3. 节假日上有明显的节前节后效应，节假日前人数多，节假日后人数少：做了这个特征
4. 在黄金周时候，游客量明显激增：也做了特征
5. 游客量在时间轴上有明显的季节趋势性，并且周期大概在90和180天：因此做了时间滚动roolling特征，滚动的窗口大致取90,180
6. 进行log平滑后，数据上半部的波动变得小了，可能有一定异常值的处理功能，最终数据经过log平滑后再预测，结果有比较大的提升。
7. 购买时间预测：
8. 复购有比较明显的时间趋势（呈现峰值趋势），大比例在10~14天是复购的顶峰期：这方面做了比较多时间间隔特征
9. 浏览过后购买大部分集中在前面的几天：

3、O2O优惠券购买预测：

1. 优惠券领取得比较多，但购买比较少

* **英文面试（海信面试）：**

1. 自我介绍：

Hello,Interviewer,

my name is li Guang Chuang, I am from Guangdong Maoming, and now I live in Guangzhou, I am a master in Guangdong University of technology, I want to apply a job as a data mining engineer.

During my master period, I main learn the data mining, and I had learned some program language, such as “python, c, sql” and so on, and some data analysis tool , such as numpy, pandas, matplotlib; and I am good at machine learning and deep learning algorithm, and I can easy to use the sklearn, xgboost, lightgbm,keras,tensorflow to build model and solve the practical problem

And I had do some project about data mining, such as “predict whether user will use O2O coupons ”, “predict user’s next purchase time”, and “predict store’s visitors”. and I am as a intern at a company from June to August in this year, and I do some medical data mining job, such as “predict if a patient will have a sickness named AKI”

So,I think I have good foundation, and I have lots of experience in data mining filed. I believe I am good at this job.

1. 喜欢的兴趣爱好：

I love reading, running, and playing pingpong.

1. 你觉得你是一个怎么样的人：

I think I am a positive, responsible, and loving technology person.

* **你认为未来二十年，哪些算法会火：**

1. 深度学习
2. 集成学习
3. 强化学习
4. 迁移学习
5. FFM / LR：广告、点击率预估

* **怎么样获取“苹果应用商店”的软件下载量**

1. 进行分层统计，对“区域”做分层，对每个层做采样。
2. 对于每个区域，统计苹果手机的使用占比并计算区域内的苹果手机数量。采样统计区域内人均app下载量，该地区下载总量 = 人均下载量 \* 苹果手机数量
3. 将每个区域的统计量累加。

* **你看了那些书：**

算法类：

机器学习

统计学习方法

机器学习实战

百面机器学习

汉语自然语言处理

Python大战机器学习

Keras深度学习实战

TensorFlow深度学习实战

Python数据挖掘实战

利用Python进行数据分析

计算机基础类：

Python语言

大话数据结构

数据结构的Python语言描述

剑指offer