机器学习问题

为什么用交叉熵代替二次代价函数?

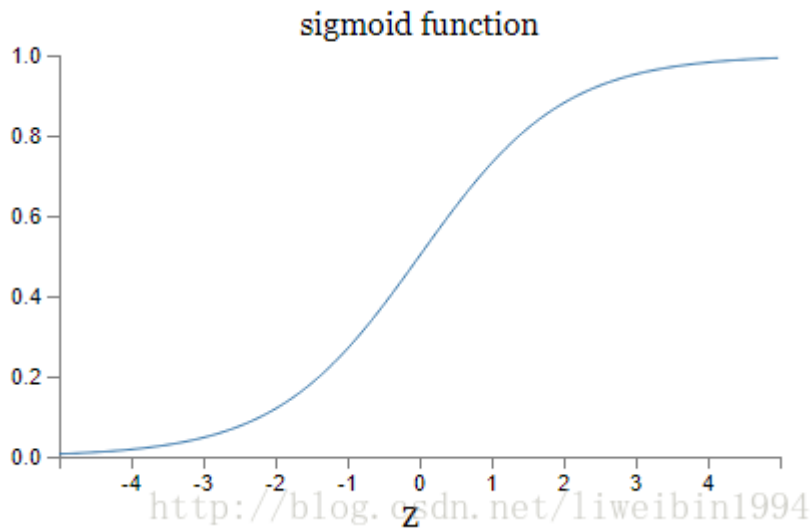
二次代价函数

$$J=\frac{1}{2n}\sum_x\|y(x)-a(x)\|^2$$

其对w和b的偏导数为

$$\frac{\partial J}{\partial w}=(a-y)\sigma'(z)x$$

$$\frac{\partial J}{\partial b}=(a-y)\sigma'(z)$$



偏导数受到激活函数的导数影响，sigmoid函数的导数在sigmoid函数值接近0和1时非常小，会导致一些实例在刚开始训练时学习的非常慢。

而使用交叉熵

$$J=-\frac{1}{n}\sum_x ylna+(1-y)ln(1-a)$$

其对w和b的偏导数为

$$\frac{\partial J}{\partial w_j}=\frac{1}{n}\sum_x x_j(\sigma(z)-y)$$

$$\frac{\partial J}{\partial b}=\frac{1}{n}\sum_x (\sigma(z)-y)$$

权重学习的速度受到sigmoid(z)-y影响，更大的误差，就有更快的学习速度，避免了二次代价函数的学习缓慢的情况。

隐马尔科夫模型基本问题及其相应的算法：

1、评估问题：概率计算问题：给定模型和观测序列，计算在模型下观测序列出现的概率。

前向、后向算法解决的是一个评估问题，即给定一个模型，求某特定观测序列的概率，用于评估该序列最匹配的模型。

2、模型学习问题：已知观测序列，估计模型中的参数，使得在该模型下观测序列概率最大，即用极大似然估计的方法估计参数。

Baum-Welch算法解决的是一个模型训练问题，即参数估计，是一种无监督的训练方法，主要通过EM迭代实现；即只有观测序列，无状态序列时训练模型。

极大似然估计：观测序列和相应的状态序列都存在的监督学习算法，用来估计参数。

3、解码问题/预测问题：已知模型和观测序列，给定观测序列，求最可能的对应的状态序列。

维特比算法解决的是给定一个模型和某个特定的输出序列，求最可能产生这个输出的状态序列。如通过海藻变化（输出序列）来观测天气（状态序列），是预测问题，通信中的解码问题！

线性分类器最佳准则

- 感知器算法 感知准则函数
- 支持向量机
- 线性判别分析 Fisher准则

贝叶斯分类器不是线性分类器

SVM可通过正则化系数控制模型的复杂度，避免过拟合。

解决类别不均衡问题

- 重采样
- 欠采样
- 调整权值

机器学习中做特征选择时，常用的方法

通常来说，从两个方面考虑来选择特征：

- 特征是否发散：如果一个特征不发散，例如方差接近于0，也就是说样本在这个特征上基本上没有差异，这个特征对于样本的区分并没有什么用。
- 特征与目标的相关性：这点比较显见，与目标相关性高的特征，应当优选选择。除方差法外，本文介绍的其他方法均从相关性考虑。

根据特征选择的形式又可以将特征选择方法分为3种：

- Filter：过滤法，按照发散性或者相关性对各个特征进行评分，设定阈值或者待选择阈值的个数，选择特征。
    - 方差选择法

      先要计算各个特征的方差，然后根据阈值，选择方差大于阈值的特征。
    - 相关系数法

      先要计算各个特征对目标值的相关系数以及相关系数的P值
    - 卡方检验

      卡方检验这个统计量的含义就是自变量与因变量的相关性
    - 互信息法

      递归消除特征法使用一个基模型来进行多轮训练，每轮训练后，消除若干权值系数的特征，再基于新的特征集进行下一轮训练
    - 也是评价定性自变量对定性因变量的相关性的
- Wrapper：包装法，根据目标函数（通常是预测效果评分），每次选择若干特征，或者排除若干特征。
    - 递归特征消除法

      递归消除特征法使用一个基模型来进行多轮训练，每轮训练后，消除若干权值系数的特征，再 基于新的特征集进行下一轮训练
- Embedded：嵌入法，先使用某些机器学习的算法和模型进行训练，得到各个特征的权值系数，根据系数从大到小选择特征。类似于Filter方法，但是是通过训练来确定特征的优劣。
    - 基于惩罚项的特征选择法

      比如L1正则化，可以帮助选择特征
    - 基于树模型的特征选择法

      树模型中GBDT也可用来作为基模型进行特征选择

机器学习中的生成模型和判别模型：

生成模型

- 朴素贝叶斯
- 高斯混合模型
- 隐马尔科夫模型
- 马尔科夫随机场
- LDA主题模型

判别模型

- 逻辑回归
- K近邻
- SVM
- 神经网络
- 条件随机场

PDF PMF CDF

PDF：概率密度函数（probability density function）, 在数学中，连续型随机变量的概率密度函数（在不至于混淆时可以简称为密度函数）是一个描述这个随机变量的输出值，在某个确定的取值点附近的可能性的函数。

PMF : 概率质量函数（probability mass function), 在概率论中，概率质量函数是离散随机变量在各特定取值上的概率。

CDF : 累积分布函数 (cumulative distribution function)，又叫分布函数，是概率密度函数的积分，能完整描述一个实随机变量X的概率分布。

缺失值的处理方法

由于调查、编码和录入误差，数据中可能存在一些无效值和缺失值，需要给予适当的处理。常用的处理方法有：估算，整例删除，变量删除和成对删除。

估算(estimation)。最简单的办法就是用某个变量的样本均值、中位数或众数代替无效值和缺失值。这种办法简单，但没有充分考虑数据中已有的信息，误差可能较大。另一种办法就是根据调查对象对其他问题的答案，通过变量之间的相关分析或逻辑推论进行估计。例如，某一产品的拥有情况可能与家庭收入有关，可以根据调查对象的家庭收入推算拥有这一产品的可能性。

整例删除(casewise deletion)是剔除含有缺失值的样本。由于很多问卷都可能存在缺失值，这种做法的结果可能导致有效样本量大大减少，无法充分利用已经收集到的数据。因此，只适合关键变量缺失，或者含有无效值或缺失值的样本比重很小的情况。

变量删除(variable deletion)。如果某一变量的无效值和缺失值很多，而且该变量对于所研究的问题不是特别重要，则可以考虑将该变量删除。这种做法减少了供分析用的变量数目，但没有改变样本量。

成对删除(pairwise deletion)是用一个特殊码(通常是9、99、999等)代表无效值和缺失值，同时保留数据集中的全部变量和样本。但是，在具体计算时只采用有完整答案的样本，因而不同的分析因涉及的变量不同，其有效样本量也会有所不同。这是一种保守的处理方法，最大限度地保留了数据集中的可用信息。

采用不同的处理方法可能对分析结果产生影响，尤其是当缺失值的出现并非随机且变量之间明显相关时。因此，在调查中应当尽量避免出现无效值和缺失值，保证数据的完整性。

基于核的机器学习方法

- SVM
- LDA线性判别分析
- 径向基函数（Radial Basis Function ，RBF)

降维算法

算法名称	线性/非线性	有监督/无监督	(超)参数	是否去中心化	目标	假设	涉及矩阵	解的形式
PCA	线性	无监督	W,d	是	降维后的低维样本之间每一维方差尽可能大	低维空间相互正交	C,W	取C前d个最大特征值对应特征向量排列成线性变换W的列
MDS	非线性	无监督	d	是	降维的同时保证数据之间的相对关系不变	已知高维空间的N个样本之间的距离矩阵	E,A	取E前d个最大特征值对应特征向量排列成低维矩阵Z的列
LDA	线性	有监督	W,d	否	降维后同一类样本之间协方差尽可能小，不同类中心距离尽可能大	数据能够被分成d+1类	S_w, S_b, W	取 $S_w^{-1}S_b$ 特征分解的前d个最大特征值对应特征向量排列成W
Isomap	非线性	无监督	d,k	是	降维的同时保证高维数据的流形不变	高维空间的局部区域上某两点距离可以由欧式距离算出	E,A	与MDS一致
LLE	非线性	无监督	d,k	是	降维的同时保证高维数据的流形不变	高维空间的局部区域上某一点是相邻K个点的线性组合，低维空间各维正交	F,M	取M前d个非0最小特征值对应特征向量构成
t-SNE	非线性	无监督	K=2/3	-	降维到二维或者三维可视化	在高维空间中，一个点的取值服从以另外一个点为中心的高斯分布。在低维空间中，两个点之间的欧式距离服从自由度为1的t分布	P,Q	梯度下降的方式来更新低维空间的Z
Auto encoder	非线性	无监督	W_l, l, D_l	-	这个网络能够重构输入数据	网络能够学习到数据内部的一些性质或者结构	W_l	网络最后一层的输出

HMM MEMM CRF

HMM:隐马尔可夫模型

MEMM: 最大熵隐马尔可夫模型

CRF：条件随机场

CRF 的优点：特征灵活，可以容纳较多的上下文信息，能够做到全局最优

CRF 的缺点：速度慢

CRF没有HMM那样严格的独立性假设条件，因而可以容纳任意的上下文信息。和HMM相比，特征设计灵活（与ME一样）

（2）同时，和MEMM相比，由于CRF计算全局最优输出节点的条件概率，它还克服了最大熵马尔可夫模型标记偏置（Label-bias）的缺点。

（3）和MEMM相比，CRF是在给定需要标记的观察序列的条件下，计算整个标记序列的联合概率分布，而不是在给定当前状态条件下，定义下一个状态的状态分布。

缺点：训练代价大、复杂度高

梯度提升决策树（GBDT）和随机森林（RF）的异同

GBDT（Gradient Boosting Decision Tree）

DT + Boosting = GBDT GBDT是一种boosting算法。boosting工作机制：先从初始训练集训练处一个基学习器，然后在根据基学习器的表现对训练样本分布进行调整，使得先前的基学习器做错的训练样本在后续获得更多关注（增加错误样本权重），然后基于调整后的样本分布训练下一个基学习器，如此重复，直到基学习器达到指定的T时，最终将T个基学习器进行加权结合，得出预测。
RF（Random Forest） DT + Bagging = RF 随机森林是bagging的一种扩展，在k个数据集选择的时候后，引入了随机属性选择。加入所有属性个数为d，k是随机选择的属性个数。那么k=d的时候，就没有改变。那么k=1的时候后，随机选择一个属性用于计算。推荐的k=log2d. 随机森林的基学习器一般是决策树算法-主要，也有神经网络。

随机森林是对bagging算法的一点改动，但是根能提现样本集之间的差异性。会提高最终预测结果的泛化能力。

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

1、组成随机森林的树可以是分类树，也可以是回归树；而GBDT只由回归树组成

2、组成随机森林的树可以并行生成；而GBDT只能是串行生成

3、对于最终的输出结果而言，随机森林采用多数投票等；而GBDT则是将所有结果累加起来，或者加权累加起来

4、随机森林对异常值不敏感，GBDT对异常值非常敏感

5、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

6、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

处理欠拟合和过拟合

(1)解决欠拟合的方法：

```
1、增加新特征，可以考虑加入进特征组合、高次特征，来增大假设空间;
2、尝试非线性模型，比如核SVM 、决策树、DNN等模型;
3、如果有正则项可以较小正则项参数 $\lambda$;
4、Boosting ,Boosting 往往会有较小的 Bias，比如 Gradient Boosting 等.
```

(2)解决过拟合的方法：

1、交叉检验，通过交叉检验得到较优的模型参数;
2、特征选择，减少特征数或使用较少的特征组合，对于按区间离散化的特征，增大划分的区间;
3、正则化，常用的有 L1、L2 正则。而且 L1正则还可以自动进行特征选择;
4、如果有正则项则可以考虑增大正则项参数 lambda;
5、增加训练数据可以有限的避免过拟合;
6、Bagging ,将多个弱学习器Bagging 一下效果会好很多，比如随机森林等.

（3）DNN中常见的方法：

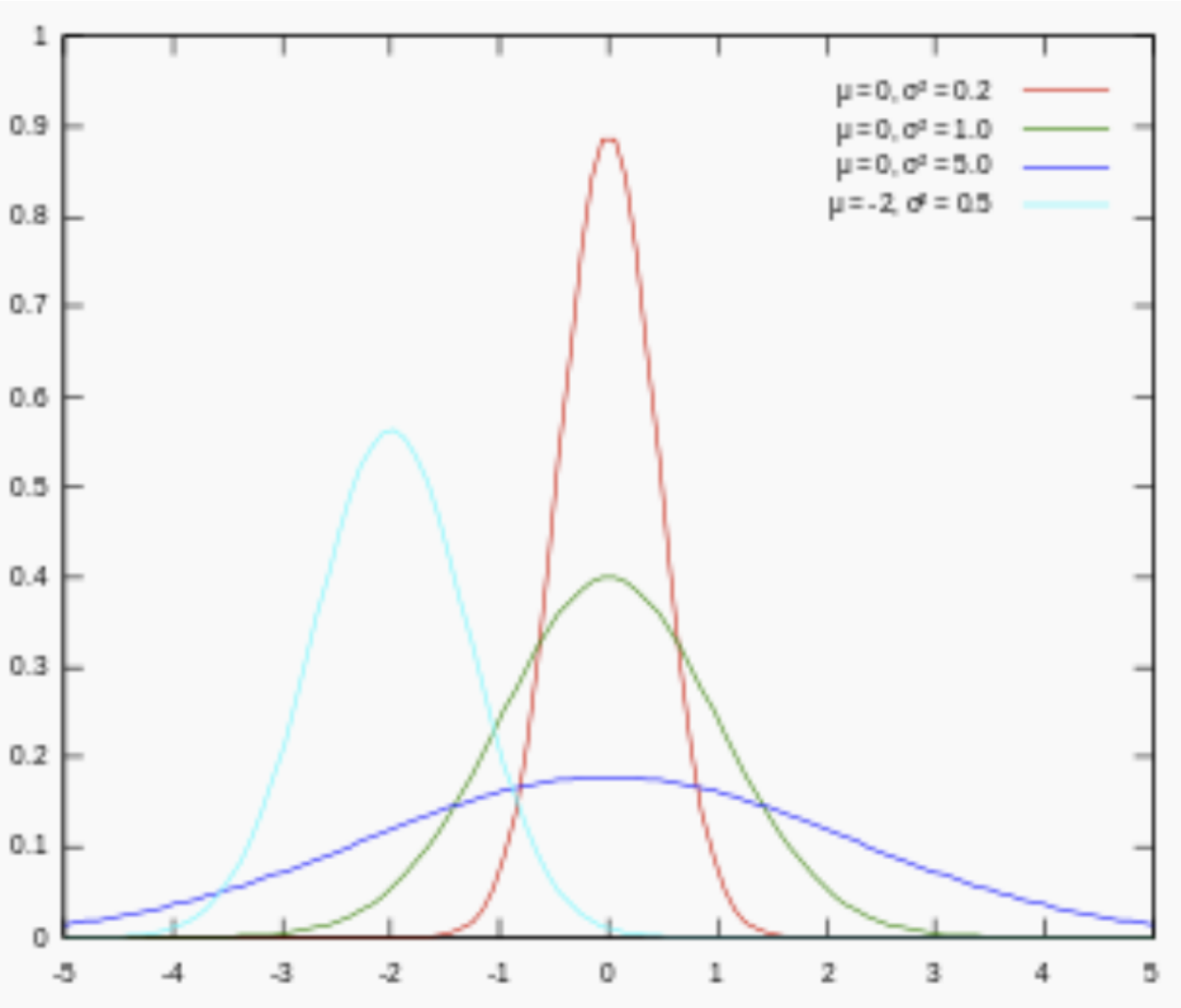
1、早停策略。本质上是交叉验证策略，选择合适的训练次数，避免训练的网络过度拟合训练数据。
2、集成学习策略。而DNN可以用Bagging的思路来正则化。首先我们要对原始的m个训练样本进行有放回随机采样，构建N组m个样本的数据集，然后分别用这N组数据集去训练我们的DNN。即采用我们的前向传播算法和反向传播算法得到N个DNN模型的W,b参数组合，最后对N个DNN模型的输出用加权平均法或者投票法决定最终输出。不过用集成学习Bagging的方法有一个问题，就是我们的DNN模型本来就比较复杂，参数很多。现在又变成了N个DNN模型，这样参数又增加了N倍，从而导致训练这样的网络要花更加多的时间和空间。因此一般N的个数不能太多，比如5-10个就可以了。
3、DropOut策略。所谓的Dropout指的是在用前向传播算法和反向传播算法训练DNN模型时，一批数据迭代时，随机的从全连接DNN网络中去掉一部分隐藏层的神经元。 在对训练集中的一批数据进行训练时，我们随机去掉一部分隐藏层的神经元，并用去掉隐藏层的神经元的网络来拟合我们的一批训练数据。使用基于dropout的正则化比基于bagging的正则化简单，这显而易见，当然天下没有免费的午餐，由于dropout会将原始数据分批迭代，因此原始数据集最好较大，否则模型可能会欠拟合。

线性分类器与非线性分类器的区别以及优劣

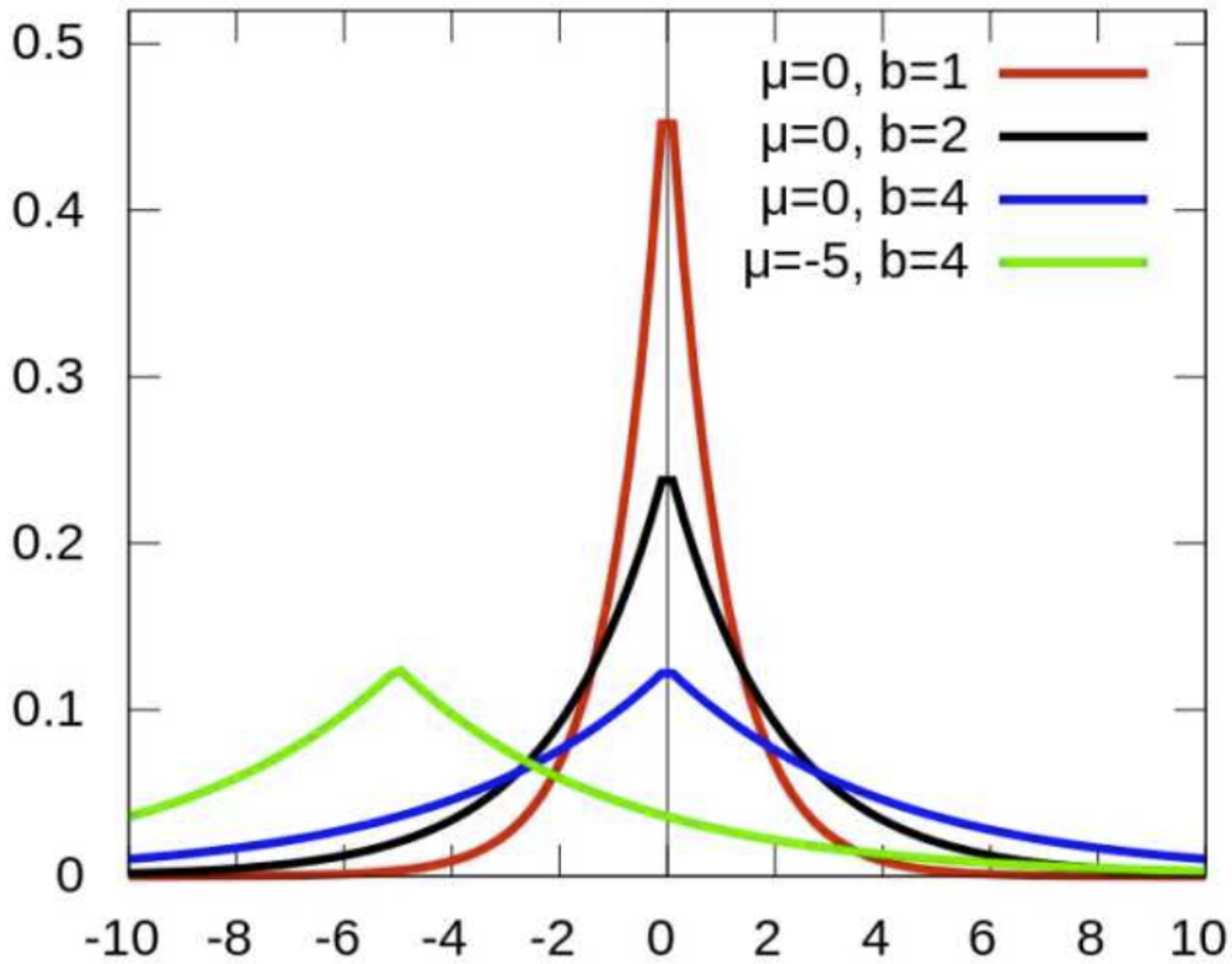
线性和非线性是针对，模型参数和输入特征来讲的；比如输入x，模型y=ax+ax^2那么就是非线性模型，如果输入是x和X^2则模型是线性的。 线性分类器可解释性好，计算复杂度较低，不足之处是模型的拟合效果相对弱些。 非线性分类器效果拟合能力较强，不足之处是数据量不足容易过拟合、计算复杂度高、可解释性不好。 常见的线性分类器有：LR,贝叶斯分类，单层感知机、线性回归 常见的非线性分类器：决策树、RF、GBDT、多层感知机 SVM两种都有（看线性核还是高斯核）

L1和L2正则先验分别服从什么分布

面试中遇到的，L1和L2正则先验分别服从什么分布，L1是拉普拉斯分布，L2是高斯分布。 先验就是优化的起跑线, 有先验的好处就是可以在较小的数据集中有良好的泛化性能，当然这是在先验分布是接近真实分布的情况下得到的了，从信息论的角度看，向系统加入了正确先验这个信息，肯定会提高系统的性能。 对参数引入高斯正态先验分布相当于L2正则化, 这个大家都熟悉：



对参数引入拉普拉斯先验等价于 L1正则化, 如下图：



从上面两图可以看出, L2先验趋向零周围, L1先验趋向零本身。

随机森林如何评估特征重要性?

衡量变量重要性的方法有两种，Decrease GINI 和 Decrease Accuracy：

1) Decrease GINI： 对于回归问题，直接使用argmax(VarVarLeftVarRight)作为评判标准，即当前节点训练集的方差Var减去左节点的方差VarLeft和右节点的方差VarRight。

2) Decrease Accuracy：对于一棵树Tb(x)，我们用OOB样本可以得到测试误差1；然后随机改变OOB样本的第j列：保持其他列不变，对第j列进行随机的上下置换，得到误差2。至此，我们可以用误差1-误差2来刻画变量j的重要性。基本思想就是，如果一个变量j足够重要，那么改变它会极大的增加测试误差；反之，如果改变它测试误差没有增大，则说明该变量不是那么的重要。

bagging与boosting两种集成模型的偏差bias以及方差variance 的理解

Bagging对样本重采样，对每一重采样得到的子样本集训练一个模型，最后取平均。由于子样本集的相似性以及使用的是同种模型，因此各模型有近似相等的bias和variance（事实上，各模型的分布也近似相同，但不独立）。由于 $E[\frac{\sum X_i}{n}] = E[X_i]$，所以bagging后的bias和单个子模型的接近，一般来说不能显著降低bias。另一方面，若各子模型独立，则有 $Var(\frac{\sum X_i}{n}) = \frac{Var(X_i)}{n}$，此时可以显著降低variance。若各子模型完全相同，则 $Var(\frac{\sum X_i}{n}) = Var(X_i)$，此时不会降低variance。bagging方法得到的各子模型是有一定相关性的，属于上面两个极端状况的中间态，因此可以一定程度降低variance。为了进一步降低variance，Random forest通过随机选取变量子集做拟合的方式de-correlated了各子模型（树），使得variance进一步降低。

（用公式可以一目了然：设有i.d.的n个随机变量，方差记为 σ^2，两两变量之间的相关性为 ρ，则 $\frac{\sum X_i}{n}$ 的方差为 $\rho * \sigma^2 + (1-\rho) * \sigma^2/n$，bagging降低的是第二项，random forest是同时降低两项。详见ESL p588公式15.1）

boosting从优化角度来看，是用forward-stagewise这种贪心法去最小化损失函数 $L(y, \sum_i a_i f_i(x))$。例如，常见的AdaBoost即等价于用这种方法最小化exponential loss：$L(y, f(x)) = exp(-yf(x))$。所谓forward-stagewise，就是在迭代的第n步，求解新的子模型f(x)及步长a（或者叫组合系数），来最小化 $L(y, f_{n-1}(x) + af(x))$，这里 $f_{n-1}(x)$ 是前n-1步得到的子模型的和。因此boosting是在sequential地最小化损失函数，其bias自然逐步下降。但由于是采取这种sequential、adaptive的策略，各子模型之间是强相关的，于是子模型之和并不能显著降低variance。所以说boosting主要还是靠降低bias来提升预测精度。

另外，计算角度来看，两种方法都可以并行。bagging, random forest并行化方法显而意见。boosting有强力工具stochastic gradient boosting，其本质等价于sgd，并行化方法参考async sgd之类的业界常用方法即可。

XGBoost的优势

XGBoost算法可以给预测模型带来能力的提升。当我对它的表现有更多了解的时候，当我对它的高准确率背后的原理有更多了解的时候，我发现它具有很多优势：

1. 正则化

- 标准GBM的实现没有像XGBoost这样的正则化步骤。正则化对减少过拟合也是有帮助的。
- 实际上，XGBoost以“正则化提升(regularized boosting)”技术而闻名。

2. 并行处理

- XGBoost可以实现并行处理，相比GBM有了速度的飞跃。
- 不过，众所周知，Boosting算法是顺序处理的，它怎么可能并行呢？每一课树的构造都依赖于前一棵树，那具体是什么让我们能用多核处理器去构造一个树呢？我希望你理解了这句话的意思。如果你希望了解更多，点击这个链接。
- XGBoost 也支持Hadoop实现。
- boosting不是一种串行的结构吗?怎么并行的？注意xgboost的并行不是tree粒度的并行，xgboost也是一次迭代完才能进行下一次迭代的（第t次迭代的代价函数里包含了前面t-1次迭代的预测值）。xgboost的并行是在特征粒度上的。我们知道，决策树的学习最耗时的一个步骤就是对特征的值进行排序（因为要确定最佳分割点），xgboost在训练之前，预先对数据进行了排序，然后保存为block结构，后面的迭代中重复地使用这个结构，大大减小计算量。这个block结构也使得并行成为了可能，在进行节点的分裂时，需要计算每个特征的增益，最终选增益最大的那个特征去做分裂，那么各个特征的增益计算就可以开多线程进行。 可并行的近似直方图算法。树节点在进行分裂时，我们需要计算每个特征的每个分割点对应的增益，即用贪心法枚举所有可能的分割点。当数据无法一次载入内存或者在分布式情况下，贪心算法效率就会变得很低，所以xgboost还提出了一种可并行的近似直方图算法，用于高效地生成候选的分割点。

3. 高度的灵活性

- XGBoost 允许用户定义**自定义优化目标和评价标准**
- 它对模型增加了一个全新的维度，所以我们的处理不会受到任何限制。

4. 缺失值处理

- XGBoost内置处理缺失值的规则。
- 用户需要提供一个和其它样本不同的值，然后把它作为一个参数传进去，以此来作为缺失值的取值。XGBoost在不同节点遇到缺失值时采用不同的处理方法，并且会学习未来遇到缺失值时的处理方法。

5. 剪枝

- 当分裂时遇到一个负损失时，GBM会停止分裂。因此GBM实际上是一个**贪心算法**。
- XGBoost会一直分裂到指定的最大深度(max_depth)，然后回过头来剪枝。如果某个节点之后不再有正值，它会去除这个分裂。
- 这种做法的优点，当一个负损失（如-2）后面有个正损失（如+10）的时候，就显现出来了。GBM会在-2处停下来，因为它遇到了一个负值。但是XGBoost会继续分裂，然后发现这两个分裂综合起来会得到+8，因此会保留这两个分裂。

6. 内置交叉验证

- XGBoost允许在每一轮boosting迭代中使用交叉验证。因此，可以方便地获得最优boosting迭代次数。
- 而GBM使用网格搜索，只能检测有限个值。

7. 在已有的模型基础上继续

- XGBoost可以在上一轮的结果上继续训练。这个特性在某些特定的应用上是一个巨大的优势。
- sklearn中的GBM的实现也有这个功能，两种算法在这一点上是一致的。

xgboost和lightgbm

xgboost和lightgbm支持缺失值处理，不过后者也支持类别特征处理，不需要进行one-hot encoding。

LightGBM优点如下

1. 速度提升 减少内存使用

XGBoost使用基于预排序的算法，难以优化。

LightGBM使用基于直方图的算法，将连续特征离散化，来提升速度，减少内存使用。LightGBM在速度与内存方面的具体优点为

- 减少计算split gain的成本
- 使用直方图减法来提升速度
- 减少内存使用
- 减少并行计算通信成本

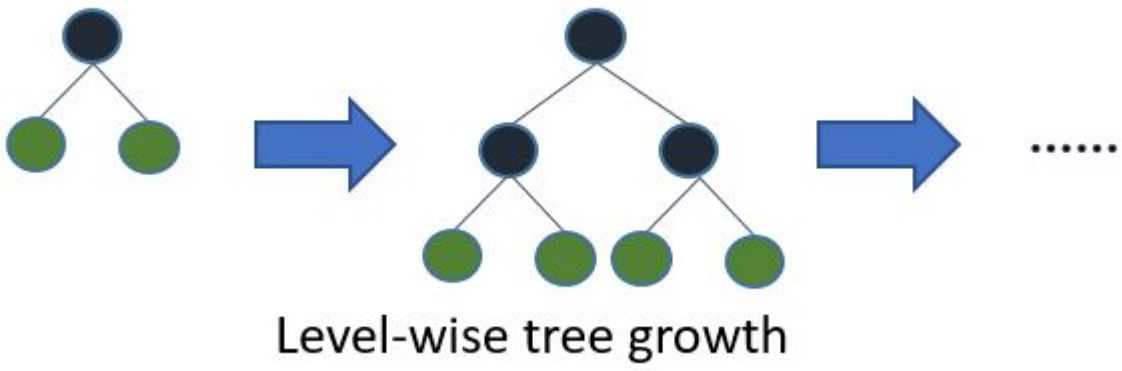
2. 适用于稀疏数据

只需O(2 * #non_zero_data)来为稀疏特征构建直方图

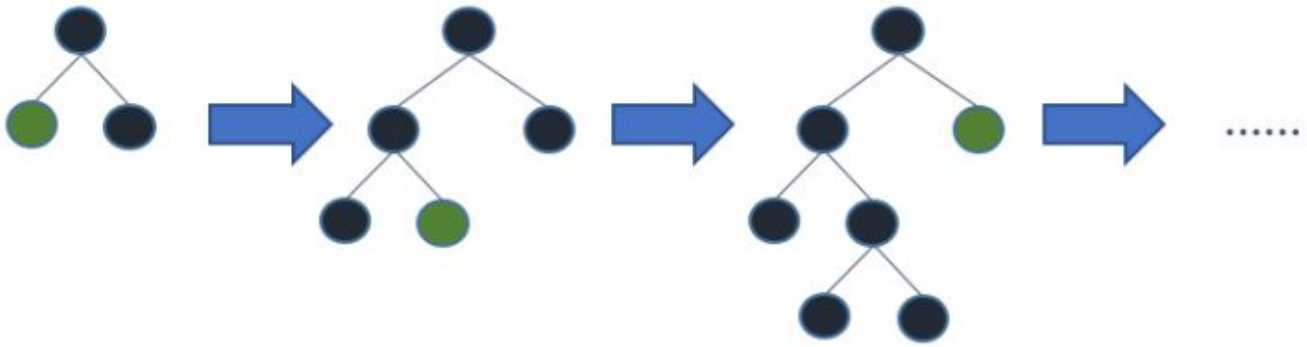
3. 准确率提升

- LightGBM采用Leaf-wise (Best-first) 树增长方式

大部分gradient boosting machine算法采用level(depth)-wise树增长方式，如下图



LightGBM采用Leaf-wise (Best-first) 树增长方式，如下图



Leaf-wise tree growth

Leaf-wise (Best-first) 算法选择使得loss下降最快的方式来生长叶子。达到相同叶子数的时候，leaf-wise算法比level-wise算法loss下降得更多。

当数据量太小时，Leaf-wise算法会过拟合，LightGBM使用数最大深度参数来防止过拟合。

- 针对categorical特征优化split

我们经常对categorical特征进行one-hot编码，但这在树学习中并不好，因为对于可取值众多的categorical特征，学习到的树模型会非常不平衡，树深度需要很深才能达到好的准确率。

解决方法是：将categorical特征分成两部分，有 $2^{(k-1)} - 1$ 种可能划分，存在算法在 $k * log(k)$ 时间内找到最优划分。基本思想是将categorical特征根据与训练目标的相关度进行排序，更具体的，将categorical特征的直方图根据$sum_gradient / sum_hessian$ 值进行排序。

4. 网络通信优化

使用collective通信方法，包括"All reduce", "All gather" and "Reduce scatter"，比点对点通信方法有更好效果。

5. 并行学习优化

- 特征并行
- 数据并行
- 投票并行

6. 支持GPU

7. 应用广泛

- regression, the objective function is L2 loss
- binary classification, the objective function is logloss
- multi classification
- lambdarank, the objective function is lambdarank with NDCG

k-means算法时间复杂度

从算法过程中可以看到，需要求n个数据点和k个中心的距离，外面有多次迭代，迭代次数为m。则k-means算法的时间复杂度为： O(mnkd)，m与数据集本身的分布情况和初始中心点位置有关。n为数据集中数据样本数量，k为聚类个数，d为数据的维数。

空间复杂度：O((n+k)d)，其中，k为簇的数目，n为样本数，d为维数

马氏距离和欧氏距离

欧氏距离具有平移和旋转不变形

马氏距离除具有平移和旋转不变性外，还具有尺度不变形，并且不受量纲影响。

马氏距离（Mahalanobis Distance）是由印度统计学家马哈拉诺比斯（P. C. Mahalanobis）提出的，表示数据的**协方差距离**。它是一种有效的计算两个未知样本集的相似度的方法。与欧氏距离不同的是它考虑到各种特性之间的联系并且是尺度无关的（scale-invariant），即独立于测量尺度。

对于一个均值为

$$\mu = (\mu_1, \mu_2, \mu_3, \ldots, \mu_p)^T$$

协方差矩阵为∑的多变量

$$x = (x_1, x_2, x_3, \ldots, x_p)^T$$

其马氏距离为：

$$D_M(x) = \sqrt{(x-\mu)^T \sum^{-1} (x-\mu)}$$

深度学习调参技巧

训练技巧对深度学习来说是非常重要的，作为一门实验性质很强的科学，同样的网络结构使用不同的训练方法训练，结果可能会有很大的差异。这里我总结了近一年来的炼丹心得，分享给大家，也欢迎大家补充指正。

参数初始化。

下面几种方式,随便选一个,结果基本都差不多。但是一定要做。否则可能会减慢收敛速度，影响收敛结果，甚至造成Nan等一系列问题。

下面的n_in为网络的输入大小，n_out为网络的输出大小，n为n_in或(n_in+n_out)*0.5

Xavier初始法论文：http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

He初始化论文：https://arxiv.org/abs/1502.01852

- uniform均匀分布初始化： w = np.random.uniform(low=-scale, high=scale, size=[n_in,n_out])
- - Xavier初始法，适用于普通激活函数(tanh,sigmoid)：scale = np.sqrt(3/n)

    - He初始化，适用于ReLU：scale = np.sqrt(6/n)
- normal高斯分布初始化： w = np.random.randn(n_in,n_out) * stdev # stdev为高斯分布的标准差，均值设为0
-
    - Xavier初始法，适用于普通激活函数 (tanh,sigmoid)：stdev = np.sqrt(n)
    - He初始化，适用于ReLU：stdev = np.sqrt(2/n)
- svd初始化：对RNN有比较好的效果。参考论文：https://arxiv.org/abs/1312.6120

数据预处理方式

- zero-center ,这个挺常用的. X -= np.mean(X, axis = 0) # zero-center X /= np.std(X, axis = 0) # normalize
- PCA whitening,这个用的比较少.

训练技巧

- 要做梯度归一化,即算出来的梯度除以minibatch size
- clip c(梯度裁剪): 限制最大梯度,其实是value = sqrt(w1^2+w2^2....),如果value超过了阈值,就算一个衰减系系数,让value的值等于阈值: 5,10,15
- dropout对小数据防止过拟合有很好的效果,值一般设为0.5,小数据上dropout+sgd在我的大部分实验中，效果提升都非常明显.因此可能的话，建议一定要尝试一下。 dropout的位置比较有讲究, 对于RNN,建议放到输入->RNN与RNN->输出的位置.关于RNN如何用dropout,可以参考这篇论文:http://arxiv.org/abs/1409.2329
- adam,adadelta等,在小数据上,我这里实验的效果不如sgd, sgd收敛速度会慢一些，但是最终收敛后的结果，一般都比较好。如果使用sgd的话,可以选择从1.0或者0.1的学习率开始,隔一段时间,在验证集上检查一下,如果cost没有下降,就对学习率减半. 我看过很多论文都这么搞,我自己实验的结果也很好. 当然,也可以先用ada系列先跑,最后快收敛的时候,更换成sgd继续训练.同样也会有提升. 据说adadelta一般在分类问题上效果比较好，adam在生成问题上效果比较好。
- 除了gate之类的地方,需要把输出限制成0-1之外,尽量不要用sigmoid,可以用tanh或者relu之类的激活函数.1. sigmoid函数在-4到4的区间里，才有较大的梯度。之外的区间，梯度接近0，很容易造成梯度消失问题。2. 输入0均值，sigmoid函数的输出不是0均值的。
- rnn的dim和embdding size,一般从128上下开始调整. batch size,一般从128左右开始调整.batch size合适最重要,并不是越大越好.
- word2vec初始化,在小数据上,不仅可以有效提高收敛速度,也可以可以提高结果.
- 尽量对数据做shuffle
- LSTM 的forget gate的bias,用1.0或者更大的值做初始化,可以取得更好的结果,来自这篇论文:http://jmlr.org/proceedings/papers/v37/jozefowicz15.pdf, 我这里实验设成1.0,可以提高收敛速度.实际使用中,不同的任务,可能需要尝试不同的值.
- Batch Normalization据说可以提升效果，不过我没有尝试过，建议作为最后提升模型的手段，参考论文：Accelerating Deep Network Training by Reducing Internal Covariate Shift
- 如果你的模型包含全连接层（MLP），并且输入和输出大小一样，可以考虑将MLP替换成Highway Network,我尝试对结果有一点提升，建议作为最后提升模型的手段，原理很简单，就

是给输出加了一个gate来控制信息的流动，详细介绍请参考论文: http://arxiv.org/abs/1505.00387
- 来自@张馨宇的技巧：一轮加正则，一轮不加正则，反复进行。

Ensemble

Ensemble是论文刷结果的终极核武器,深度学习中一般有以下几种方式

- 同样的参数,不同的初始化方式
- 不同的参数,通过cross-validation,选取最好的几组
- 同样的参数,模型训练的不同阶段，即不同迭代次数的模型。
- 不同的模型,进行线性融合. 例如RNN和传统模型.

为什么RNN使用tanh不使用ReLU?

对CNN来说，使用sigmoid作为激活函数的话，导数在(0, 0,25]之间，小于1的数乘在一起，必然是越乘越小的，自然会导致梯度消失。而ReLU的导数为{0，1}，这样的话只要一条路径上的导数都是1，无论神经网络是多少层，这一部分的乘积都始终为1，因此深层的梯度也可以传递到浅层中。

而对RNN来说，如果使用ReLU，会导致非常大的输出值。

$$a_i = W f_{i-1} + U x_i + b_i$$
$$f_i = f[a_i]$$

在这个表示中，RNN每个阶段的输入是 x_i ，**和CNN每一层使用独立的参数 W_i 不同，原始的RNN在每个阶段都共享一个参数 W** 。如果我们假设从某一层开始输入 x_i 和偏置 b_i 都为0，那么最后得到的输出就是

$$f[W \ldots [W f[W f[W f[W f_i]]]]]$$

，**这在某种程度上相当于对参数矩阵 W 作连乘**，很显然，只要 W 有一个大于1的特征值，在经过若干次连乘后都会导致结果是一个数值非常庞大的矩阵。

另外一方面，将激活函数换成ReLU也不能解决梯度在长程上传递的问题，因为CNN中，每一层的参数是相互独立的，并且都是系数矩阵，进行连乘也不会导致梯度爆炸，而RNN中W参与了每个时间段的运算，连乘的话很容易导致梯度爆炸/消失问题。

一元线性回归的基本假设有：

（1）随机误差项是一个期望值或平均值为0的随机变量；

（2）对于解释变量的所有观测值，随机误差项有相同的方差；

（3）随机误差项彼此不相关；

（4）解释变量是确定性变量，不是随机变量，与随机误差项彼此之间相互独立；

（5）解释变量之间不存在精确的（完全的）线性关系，即解释变量的样本观测值矩阵是满秩矩阵；

（6）随机误差项服从正态分布

所谓多重共线性（Multicollinearity）是指线性回归模型中的解释变量之间由于存在精确相关关系或高度相关关系而使模型估计失真或难以估计准确。

神经网络中激活函数的真正意义？一个激活函数需要具有哪些必要的属性？还有哪些属是好的属性但不必要的？

1. 非线性：即导数不是常数。这个条件前面很多答主都提到了，是多层神经网络的基础，保证多层网络不退化成单层线性网络。这也是激活函数的意义所在。
2. 几乎处处可微：可微性保证了在优化中梯度的可计算性。传统的激活函数如sigmoid等满足处处可微。对于分段线性函数比如ReLU，只满足几乎处处可微（即仅在有限个点处不可微）。对于SGD算法来说，由于几乎不可能收敛到梯度接近零的位置，有限的不可微点对于优化结果不会有很大影响。
3. 计算简单：正如题主所说，非线性函数有很多。极端的说，一个多层神经网络也可以作为一个非线性函数，类似于Network In Network中把它当做卷积操作的做法。但激活函数在神经网络前向的计算次数与神经元的个数成正比，因此简单的非线性函数自然更适合用作激活函数。这也是ReLU之流比其它使用Exp等操作的激活函数更受欢迎的其中一个原因。
4. 非饱和性（saturation）：饱和指的是在某些区间梯度接近于零（即梯度消失），使得参数无法继续更新的问题。最经典的例子是Sigmoid，它的导数在x为比较大的正值和比较小的负值时都会接近于0。更极端的例子是阶跃函数，由于它在几乎所有位置的梯度都为0，因此处处饱和，无法作为激活函数。ReLU在x>0时导数恒为1，因此对于再大的正值也不会饱和。但同时对于x<0，其梯度恒为0，这时候它也会出现饱和的现象（在这种情况下通常称为dying ReLU）。Leaky ReLU[3]和PReLU的提出正是为了解决这一问题。
5. 单调性（monotonic）：即导数符号不变。这个性质大部分激活函数都有，除了诸如sin、cos等。个人理解，单调性使得在激活函数处的梯度方向不会经常改变，从而让训练更容易收敛。
6. 输出范围有限：有限的输出范围使得网络对于一些比较大的输入也会比较稳定，这也是为什么早期的激活函数都以此类函数为主，如Sigmoid、Tanh。但这导致了前面提到的梯度消失问题，而且强行让每一层的输出限制到固定范围会限制其表达能力。因此现在这类函数仅用于某些需要特定输出范围的场合，比如概率输出（此时loss函数中的log操作能够抵消其梯度消失的影响）、LSTM里的gate函数。
7. 接近恒等变换（identity）：即约等于x。这样的好处是使得输出的幅值不会随着深度的增加而发生显著的增加，从而使网络更为稳定，同时梯度也能够更容易地回传。这个与非线性是有点矛盾的，因此激活函数基本只是部分满足这个条件，比如Tanh只在原点附近有线性区（在原点为0且在原点的导数为1），而ReLU只在x>0时为线性。这个性质也让初始化参数范围的推导更为简单。额外提一句，这种恒等变换的性质也被其他一些网络结构设计所借鉴，比如CNN中的ResNet和RNN中的LSTM。

8. 参数少：大部分激活函数都是没有参数的。像PReLU带单个参数会略微增加网络的大小。还有一个例外是Maxout，尽管本身没有参数，但在同样输出通道数下k路Maxout需要的输入通道数是其它函数的k倍，这意味着神经元数目也需要变为k倍；但如果不考虑维持输出通道数的情况下，该激活函数又能将参数个数减少为原来的k倍。
9. 归一化（normalization）：这个是最近才出来的概念，对应的激活函数是SELU，主要思想是使样本分布自动归一化到零均值、单位方差的分布，从而稳定训练。在这之前，这种归一化的思想也被用于网络结构的设计，比如Batch Normalization。

EM算法、HMM、CRF

这三个放在一起不是很恰当，但是有互相有关联，所以就放在这里一起说了。注意重点关注算法的思想。 （1）EM算法 EM算法是用于含有隐变量模型的极大似然估计或者极大后验估计，有两步组成：E步，求期望（expectation）；M步，求极大（maxmization）。本质上EM算法还是一个迭代算法，通过不断用上一代参数对隐变量的估计来对当前变量进行计算，直到收敛。 注意：EM算法是对初值敏感的，而且EM是不断求解下界的极大化逼近求解对数似然函数的极大化的算法，也就是说EM算法不能保证找到全局最优值。对于EM的导出方法也应该掌握。 （2）HMM算法 隐马尔可夫模型是用于标注问题的生成模型。有几个参数（π，A，B）：初始状态概率向量π，状态转移矩阵A，观测概率矩阵B。称为马尔科夫模型的三要素。 马尔科夫三个基本问题：

- 概率计算问题：给定模型和观测序列，计算模型下观测序列输出的概率。–》前向后向算法
- 学习问题：已知观测序列，估计模型参数，即用极大似然估计来估计参数。–》Baum-Welch(也就是EM算法)和极大似然估计。
- 预测问题：已知模型和观测序列，求解对应的状态序列。–》近似算法（贪心算法）和维比特算法（动态规划求最优路径）

（3）条件随机场CRF 给定一组输入随机变量的条件下另一组输出随机变量的条件概率分布密度。条件随机场假设输出变量构成马尔科夫随机场，而我们平时看到的大多是线性链条随机场，也就是由输入对输出进行预测的判别模型。求解方法为极大似然估计或正则化的极大似然估计。 之所以总把HMM和CRF进行比较，主要是因为CRF和HMM都利用了图的知识，但是CRF利用的是马尔科夫随机场（无向图），而HMM的基础是贝叶斯网络（有向图）。而且CRF也有：概率计算问题、学习问题和预测问题。大致计算方法和HMM类似，只不过不需要EM算法进行学习问题。

（4）HMM和CRF对比 其根本还是在于基本的理念不同，一个是生成模型，一个是判别模型，这也就导致了求解方式的不同。

CNN常用的几个模型

名称	特点
LeNet5	没啥特点-不过是第一个CNN应该要知道
AlexNet	引入了ReLU和dropout，引入数据增强、池化相互之间有覆盖，三个卷积一个最大池化+三个全连接层
VGGNet	采用1×1和3×3的卷积核以及2×2的最大池化使得层数变得更深。常用VGGNet-16和VGGNet19
Google Inception Net	这个在控制了计算量和参数量的同时，获得了比较好的分类性能，和上面相比有几个大的改进： 1、去除了最后的全连接层，而是用一个全局的平均池化来取代它；2、引入Inception Module，这是一个4个分支结合的结构。所有的分支都用到了1×1的卷积，这是因为1×1性价比很高，可以用很少的参数达到非线性和特征变换。 3、Inception V2第二版将所有的5×5变成2个3×3，而且提出来著名的Batch Normalization； 4、Inception V3第三版就更变态了，把较大的二维卷积拆成了两个较小的一维卷积，加速运算、减少过拟合，同时还更改了Inception Module的结构。
ResNet残差神经网络	1、引入高速公路结构，可以让神经网络变得非常深 2、ResNet第二个版本将ReLU激活函数变成y=x的线性函数

逻辑回归相关问题

（1）公式推导一定要会

（2）逻辑回归的基本概念　　这个最好从广义线性模型的角度分析，逻辑回归是假设y服从Bernoulli分布。

（3）L1-norm和L2-norm　　其实稀疏的根本还是在于L0-norm也就是直接统计参数不为0的个数作为规则项，但实际上却不好执行于是引入了L1-norm；而**L1norm本质上是假设参数先验是服从Laplace分布的，而L2-norm是假设参数先验为Gaussian分布**，我们在网上看到的通常用图像来解答这个问题的原理就在这。　　但是L1-norm的求解比较困难，可以用坐标轴下降法或是最小角回归法求解。

（4）LR和SVM对比　　首先，LR和SVM最大的区别在于损失函数的选择，LR的损失函数为Log损失（或者说是逻辑损失都可以）、而SVM的损失函数为hinge loss。

　　其次，两者都是线性模型。　　最后，SVM只考虑支持向量（也就是和分类相关的少数点）（5）LR和随机森林区别　　随机森林等树算法都是非线性的，而LR是线性的。LR更侧重全局优化，而树模型主要是局部的优化。（6）常用的优化方法　　逻辑回归本身是可以用公式求解的，但是因为需要求逆的复杂度太高，所以才引入了梯度下降算法。　　**一阶方法：梯度下降、随机梯度下降、**

mini 随机梯度下降法。　　二阶方法：牛顿法、拟牛顿法：　　这里详细说一下牛顿法的基本原理和牛顿法的应用方式。牛顿法其实就是通过切线与x轴的交点不断更新切线的位置，直到达到曲线与x轴的交点得到方程解。在实际应用中我们因为常常要求解凸优化问题，也就是要求解函数一阶导数为0的位置，而牛顿法恰好可以给这种问题提供解决方法。实际应用中牛顿法首先选择一个点作为起始点，并进行一次二阶泰勒展开得到导数为0的点进行一个更新，直到达到要求，这时牛顿法也就成了二阶求解问题，比一阶方法更快。我们常常看到的x通常为一个多维向量，这也就引出了Hessian矩阵的概念（就是x的二阶导数矩阵）。缺点：牛顿法是定长迭代，没有步长因子，所以不能保证函数值稳定的下降，严重时甚至会失败。还有就是牛顿法要求函数一定是二阶可导的。而且计算Hessian矩阵的逆复杂度很大。 拟牛顿法： 不用二阶偏导而是构造出Hessian矩阵的近似正定对称矩阵的方法称为拟牛顿法。拟牛顿法的思路就是用一个特别的表达形式来模拟Hessian矩阵或者是他的逆使得表达式满足拟牛顿条件。主要有DFP法（逼近Hession的逆）、BFGS（直接逼近Hession矩阵）、 L-BFGS（可以减少BFGS所需的存储空间）。

假设我们要解决一个二类分类问题, 我们已经建立好了模型, 输出是0或1, 初始时设阈值为0.5, 超过0.5概率估计, 就判别为1, 否则就判别为0 ; 如果我们现在用另一个大于0.5的阈值, 那么现在关于模型说法, 正确的是 : (C)

1. 模型分类的召回率会降低或不变
2. 模型分类的召回率会升高
3. 模型分类准确率会升高或不变
4. 模型分类准确率会降低

A. 1 B. 2 C.1和3 D. 2和4 E. 以上都不是

对应GradientBoosting tree算法， 以下说法正确的是 : C

A. 当增加最小样本分裂个数，我们可以抵制过拟合

B. 当增加最小样本分裂个数，会导致过拟合

C. 当我们减少训练单个学习器的样本个数，我们可以降低variance

D. 当我们减少训练单个学习器的样本个数，我们可以降低bias

A. 2 和 4

B. 2 和 3

C. 1 和 3

D. 1 和 4

下面的交叉验证方法：

i. 有放回的Bootstrap方法

ii. 留一个测试样本的交叉验证

iii. 5折交叉验证

iv. 重复两次的5折教程验证

当样本是1000时，下面执行时间的顺序，正确的是：答案: B

A. i > ii > iii > iv

B. ii > iv > iii > i

C. iv > i > ii > iii

D. ii > iii > iv > i

- **Boostrap方法是传统地随机抽样，验证一次的验证方法，只需要训练1次模型，所以时间最少。**
- **留一个测试样本的交叉验证，需要n次训练过程（n是样本个数），这里，要训练1000个模型。**
- **5折交叉验证需要训练5个模型。**
- **重复2次的5折交叉验证，需要训练10个模型。**

在一个线性回归问题中，我们使用 R 平方（R-Squared）来判断拟合度。此时，如果增加一个特征，模型不变，则下面说法正确的是？

A. 如果 R-Squared 增加，则这个特征有意义

B. 如果R-Squared 减小，则这个特征没有意义

C. 仅看 R-Squared 单一变量，无法确定这个特征是否有意义。

D. 以上说法都不对

答案：C

解析：线性回归问题中，R-Squared 是用来衡量回归方程与真实样本输出之间的相似程度。其表达式如下所示：

$$R^2 = 1 - \frac{\sum(y - \hat{y})^2}{\sum(y - \overline{y})^2}$$

上式中，分子部分表示真实值与预测值的平方差之和，类似于均方差 MSE；分母部分表示真实值与均值的平方差之和，类似于方差 Var。根据 R-Squared 的取值，来判断模型的好坏：如果结果是 0，说明模型拟合效果很差；如果结果是 1，说明模型无错误。一般来说，R-Squared 越大，表示模型拟合效果越好。**R-Squared 反映的是大概有多准，因为，随着样本数量的增加，R-Square必然增加，无**

法真正定量说明准确程度，只能大概定量。

对于本题来说，单独看 R-Squared，并不能推断出增加的特征是否有意义。通常来说，增加一个特征，R-Squared 可能变大也可能保持不变，两者不一定呈正相关。

如果使用校正决定系数（Adjusted R-Square）：

$$R^2 - adjusted = 1 - \frac{(1-R^2)(n-1)}{n-p-1}$$

其中，n 是样本数量，p 是特征数量。Adjusted R-Square 抵消样本数量对 R-Square的影响，做到了真正的 0~1，越大越好。

下列关于线性回归分析中的残差（Residuals）说法正确的是?

A. 残差均值总是为零

B. 残差均值总是小于零

C. 残差均值总是大于零

D. 以上说法都不对

答案：A

解析：线性回归分析中，目标是残差最小化。残差平方和是关于参数的函数，为了求残差极小值，令残差关于参数的偏导数为零，会得到残差和为零，即残差均值为零。

假如我们利用 Y 是 X 的 3 阶多项式产生一些数据（3 阶多项式能很好地拟合数据）。那么，下列说法正确的是（多选）?

A. 简单的线性回归容易造成高偏差（bias）、低方差（variance）

B. 简单的线性回归容易造成低偏差（bias）、高方差（variance）

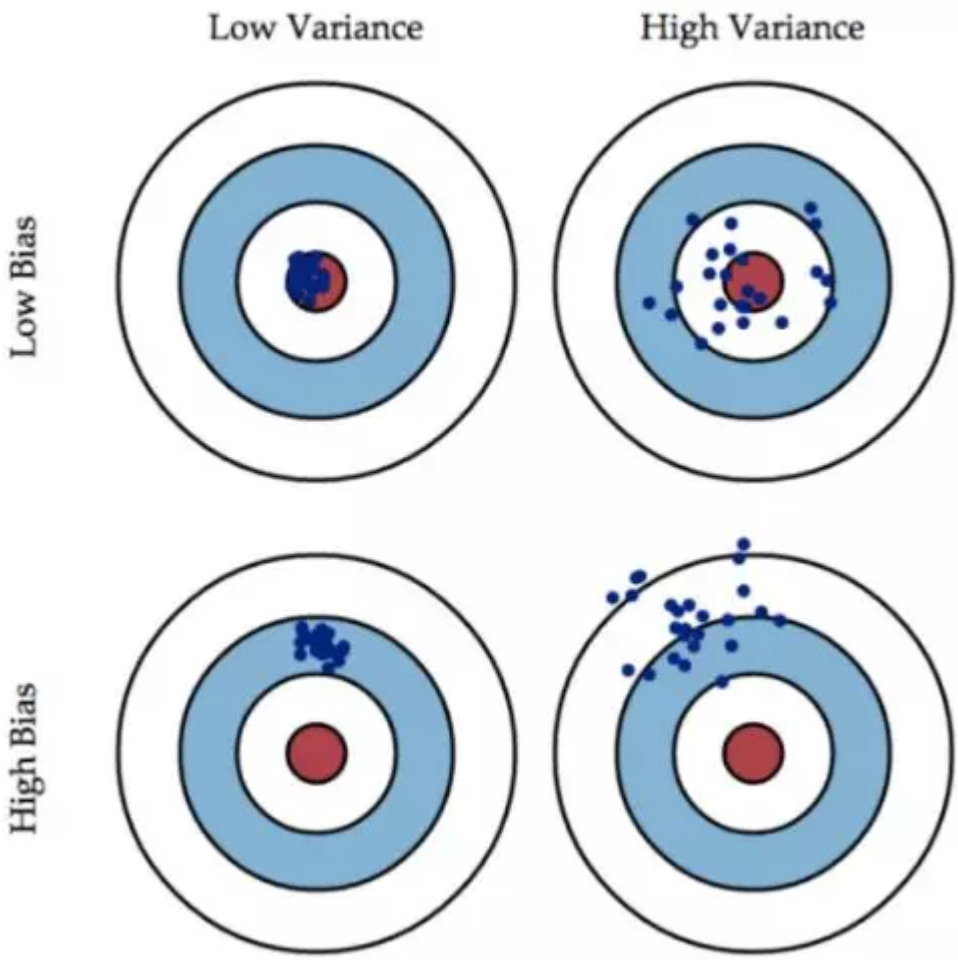
C. 3 阶多项式拟合会造成低偏差（bias）、高方差（variance）

D. 3 阶多项式拟合具备低偏差（bias）、低方差（variance）

答案：AD

解析：偏差和方差是两个相对的概念，就像欠拟合和过拟合一样。如果模型过于简单，通常会造成欠拟合，伴随着高偏差、低方差；如果模型过于复杂，通常会造成过拟合，伴随着低偏差、高方差。

用一张图来形象地表示偏差与方差的关系：



偏差（bias）可以看成模型预测与真实样本的差距，想要得到 low bias，就得复杂化模型，但是容易造成过拟合。方差（variance）可以看成模型在测试集上的表现，想要得到 low variance，就得简化模型，但是容易造成欠拟合。实际应用中，偏差和方差是需要权衡的。若模型在训练样本和测试集上都表现的不错，偏差和方差都会比较小，这也是模型比较理想的情况。

简述梯度下降法、牛顿法与共轭梯度法的原理，并比较异同点及各自的使用范围。

\1. 算法原理

梯度下降法是利用一阶的梯度信息找到函数局部最优解的一种方法。如果目标函数是一个凸优化问题，那么梯度下降法获得的局部最优解就是全局最优解。

牛顿法是利用局部的一阶和二阶偏导信息，去推测整个目标函数的形状，进而可以求得近似函数的全局最小值，然后将当前的最小值设定为近似函数的最小值。也就是说，牛顿法在二阶导数的作用下，从函数的凸性出发，直接搜索怎样到达极值点，即在选择方向时，不仅考虑当前坡度是否够大，还会考虑走了一步之后，坡度是否会变得更大。

共轭梯度法是介于最速下降法与牛顿法之间的一个方法，它仅需一阶导数信息。基本思想是把共轭性与最速下降法相结合，利用已知点处的梯度构造一组共轭方向，并沿这组方向进行搜索，求出目标函数的极小点。根据共轭方向的基本性质，这种方法具有二次终止性。核心迭代过程可以采取不同的方案，一种是直接延续，即总是用d^(k+1)=-g(k+1)+beta_k*d^(k)构造搜索方向；一种是把n步作为一轮，每搜索一轮之后，取一次最速下降方向，开始下一轮，此种策略称为“重置”。

\2. 比较异同点 共轭梯度法仅需一阶导数信息，克服了最速下降法收敛慢的缺点，又避免了牛顿法需要存储和计算Hesse矩阵并求逆的缺点。

牛顿法和梯度下降法都必须给定一个初始点，且都容易陷入局部最优。但在初始点选取合理的情况下，牛顿法比梯度下降法收敛速度更快。

牛顿法需要利用二阶导数信息，每次需要更新一个二维矩阵，计算难度和资源消耗相对梯度下降和共轭法都要更大，实际使用中常使用拟牛顿法。

牛顿法对初始值有一定要求，在非凸优化问题中（如神经网络训练），牛顿法很容易陷入鞍点（牛顿法步长会越来越小）；而梯度下降法则很容易逃离鞍点（因此在神经网络训练中一般使用梯度下降法，高维空间的神经网络中存在大量鞍点） 梯度下降法在靠近最优点时会震荡，因此步长调整在梯度下降法中是必要的，具体有adagrad, adadelta, rmsprop, adam等一系列自适应学习率的方法。

简述协方差的含义及协方差矩阵的计算过程。现有m条n列原始数据，试简述利用PCA算法对该数据进行降维的过程。

\1. 协方差就是计算了两个维度之间的相关性，即这个样本的这两个维度之间有没有关系。**（2分）**

协方差矩阵的计算： 1）先让样本矩阵中心化，即每一维度减去该维度的均值，使每一维度上的均值为0， 2）然后直接用新的到的样本矩阵乘上它的转置 3）然后除以(N-1)即可

\2. 按行或按列组织样本均可。

下面是按行组织样本：

1）将原始数据按行组成n行m列矩阵X，代表有n个数据，每个数据m个特征

2）将X的每一列（代表一个属性字段）进行零均值化，即减去这一列的均值

3）求出协方差矩阵C=1/n* XXT

4）求出协方差矩阵的特征值及对应的特征向量

5）将特征向量按对应特征值大小从上到下按列排列成矩阵，取前k列组成矩阵P

6）Y=XP即为降维到k维后的数据

谈谈广义线性模型在机器学习算法中的应用。在标准的SVM中，样本会被最优分类面分为+1或-1；而在实际应用中，后验概率是非常有用的。考虑如何使SVM实现后验概率的输出。

\1. 广义线性模型相对于经典的线性模型(y=wx+b)，核心在于引入了连接函数g(.)。

在机器学习中应用广泛，比如逻辑回归中的用sigmoid函数进行转换；在神经网络中的激活函数也是链接函数。

\2. 可以使用Sigmoid函数进行转换，对f(x)（即核函数）设学习参数，即Sigmoid((Af(x)+b))。概率SVM的构造主要分为两个步骤：首先训练标准的SVM模型。得到核函数的参数；其次训练Sigmoid函数，得到Sigmoid的参数。这样就可以对标准的SVM模型的输出值进行Sigmoid处理，将其转化为后验概率。

当训练样本数量趋向于无穷大时，在该数据集上训练的模型变化趋势，对于其描述正确的是（）

正确答案: C

A. 偏差(bias)变小 B. 偏差变大 C. 偏差不变 D. 不确定

解析：产生偏差的主要原因是无法拟合数据，可以通过选择一个新的网络（更复杂的网络）、延长训练时间或者使用更先进的算法等减小偏差。使用更多的数据可以使方差变小，而不是偏差。