**一、bagging和boosting**

先介绍Bagging方法：

Bagging即套袋法，其算法过程如下：

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（在训练集中，有些样本可能被多次抽取到，而有些样本可能一次都没有被抽中）。共进行k轮抽取，得到k个训练集。（k个训练集之间是相互独立的）

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。（注：这里并没有具体的分类算法或回归方法，我们可以根据具体问题采用不同的分类或回归方法，如决策树、感知器等）

3. 对分类问题：将上步得到的k个模型采用投票的方式得到分类结果；对回归问题，计算上述模型的均值作为最后的结果。（所有模型的重要性相同）

Boosting：

在分类问题中它通过改变训练样本的权重，学习多个分类器，并将分类器进行线性组合。

AdaBoosting方式每次使用的是全部的样本，每轮训练改变样本的权重。下一轮训练的目标是找到一个函数f 来拟合上一轮的残差。当残差足够小或者达到设置的最大迭代次数则停止。Boosting会减小在上一轮训练正确的样本的权重，增大错误样本的权重。（对的残差小，错的残差大）梯度提升的Boosting方式是使用代价函数对上一轮训练出的模型函数f的偏导来拟合残差。

Bagging，Boosting二者之间的区别

Bagging和Boosting的区别：

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）预测函数权重：

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）是否可以并行计算：

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）为什么说bagging是减少variance，而boosting是减少bias？

**Adaboost优缺点：**

Adaboost是一种加和模型，每个模型都是基于上一次模型的错误率来建立的，过分关注分错的样本，而对正确分类的样本减少关注度，逐次迭代之后，可以得到一个相对较好的模型。是一种典型的boosting算法。下面是总结下它的优缺点。

优点：泛化错误率低，易编码，可以应用在大部分分类器上，无参数可调整。adaboost是一种有很高精度的分类器。可以使用各种方法构建子分类器，Adaboost算法提供的是框架。当使用简单分类器时，计算出的结果是可以理解的，并且弱分类器的构造极其简单。简单，不用做特征筛选。不容易发生overfitting。

缺点：对离群数据点敏感。

适用数据类型：数值型和标称型数据。

**决策树优缺点：**

（1） 决策树的优点：

 直观，便于理解，小规模数据集有效

 执行效率高，执行只需要一次构建，可反复使用

（2）决策树的缺点：

 处理连续变量不好，较难预测连续字段

 类别较多时，错误增加的比较快

 对于时间序列数据需要做很多的预处理

 可规模性一般

 实际分类的时候只能根据一个字段进行

**随机森林优缺点：**

优点：

实现简单，训练速度快，泛化能力强，可以并行实现，因为训练时树与树之间是相互独立的；

相比单一决策树，能学习到特征之间的相互影响，且不容易过拟合；

能处理高维数据（即特征很多），并且不用做特征选择，因为特征子集是随机选取的；

对于不平衡的数据集，可以平衡误差；

相比SVM，不是很怕特征缺失，因为待选特征也是随机选取；

训练完成后可以给出哪些特征比较重要。

缺点：

在噪声过大的分类和回归问题还是容易过拟合；

相比于单一决策树，它的随机性让我们难以对模型进行解释。

**GBDT优缺点：**

优点：

预测精度高，适合低维数据，能处理非线性数据，可以灵活处理各种类型的数据，包括连续值和离散值。

在相对少的调参时间情况下，预测的准备率也可以比较高。这个是相对SVM来说的。

使用一些健壮的损失函数，对异常值的鲁棒性非常强。比如 Huber损失函数和Quantile损失函数。

缺点：

由于弱学习器之间存在依赖关系，难以并行训练数据。不过可以通过自采样的SGBT来达到部分并行。

如果数据维度较高时会加大算法的计算复杂度

**随机森林和GBDT的区别：**

1、随机森林采用的bagging思想，而GBDT采用的boosting思想。这两种方法都是Bootstrap思想的应用，Bootstrap是一种有放回的抽样方法思想。虽然都是有放回的抽样，但二者的区别在于：Bagging采用有放回的均匀取样，而Boosting根据错误率来取样（Boosting初始化时对每一个训练样例赋相等的权重1／n，然后用该算法对训练集训练t轮，每次训练后，对训练失败的样例赋以较大的权重），因此Boosting的分类精度要优于Bagging。Bagging的训练集的选择是随机的，各训练集之间相互独立，弱分类器可并行，而Boosting的训练集的选择与前一轮的学习结果有关，是串行的。

2、组成随机森林的树可以是分类树，也可以是回归树；而GBDT只能由回归树组成。

3、组成随机森林的树可以并行生成；而GBDT只能是串行生成。

4、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。

5、随机森林对异常值不敏感；GBDT对异常值非常敏感。

6、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。

7、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。

---------------------

**逻辑斯特回归优缺点：**

优点：

• 形式简单，模型的可解释性非常好。从特征的权重可以看到不同的特征对最后结果的影响，某个特征的权重值比较高，那么这个特征最后对结果的影响会比较大。

• 模型效果不错。在工程上是可以接受的（作为baseline)，如果特征工程做的好，效果不会太差，并且特征工程可以大家并行开发，大大加快开发的速度。

• 训练速度较快。分类的时候，计算量仅仅只和特征的数目相关。并且逻辑回归的分布式优化sgd发展比较成熟，训练的速度可以通过堆机器进一步提高，这样我们可以在短时间内迭代好几个版本的模型。

• 资源占用小,尤其是内存。因为只需要存储各个维度的特征值，。

• 方便输出结果调整。逻辑回归可以很方便的得到最后的分类结果，因为输出的是每个样本的概率分数，我们可以很容易的对这些概率分数进行cutoff，也就是划分阈值(大于某个阈值的是一类，小于某个阈值的是一类)。

缺点:

• 准确率并不是很高。因为形式非常的简单(非常类似线性模型)，很难去拟合数据的真实分布。

• 很难处理数据不平衡的问题。举个例子：如果我们对于一个正负样本非常不平衡的问题比如正负样本比 10000:1.我们把所有样本都预测为正也能使损失函数的值比较小。但是作为一个分类器，它对正负样本的区分能力不会很好。

• 处理非线性数据较麻烦。逻辑回归在不引入其他方法的情况下，只能处理线性可分的数据，或者进一步说，处理二分类的问题 。

• 逻辑回归本身无法筛选特征。有时候，我们会用gbdt来筛选特征，然后再上逻辑回归。

**K-Means聚类**

优点

算法简单，容易实现 ；

对处理大数据集，该算法是相对可伸缩的和高效率的，因为它的复杂度大约是O(nkt)，其中n是所有对象的数目，k是簇的数目,t是迭代的次数。通常k<<n。这个算法通常局部收敛。

算法尝试找出使平方误差函数值最小的k个划分。当簇是密集的、球状或团状的，且簇与簇之间区别明显时，聚类效果较好。

缺点

对数据类型要求较高，适合数值型数据；

可能收敛到局部最小值，在大规模数据上收敛较慢

K值比较难以选取；

对初值的簇心值敏感，对于不同的初始值，可能会导致不同的聚类结果；

不适合于发现非凸面形状的簇，或者大小差别很大的簇。

对于”噪声”和孤立点数据敏感，少量的该类数据能够对平均值产生极大影响。

**SVM支持向量机:**

高准确率，为避免过拟合提供了很好的理论保证，而且就算数据在原特征空间线性不可分，只要给个合适的核函数，它就能运行得很好。在动辄超高维的文本分类问题中特别受欢迎。可惜内存消耗大，难以解释，运行和调参也有些烦人，而随机森林却刚好避开了这些缺点，比较实用。

优点

可以解决高维问题，即大型特征空间；

能够处理非线性特征的相互作用；

无局部极小值问题。（相对于神经网络等算法）；

无需依赖整个数据；

可以提高泛化能力；

需要对数据提前归一化，很多人使用的时候忽略了这一点，毕竟是基于距离的模型，所以LR也需要归一化

缺点

当观测样本很多时，效率并不是很高；

一个可行的解决办法是模仿随机森林，对数据分解，训练多个模型，然后求平均，时间复杂度降低p倍，分多少份，降多少倍

对非线性问题没有通用解决方案，有时候很难找到一个合适的核函数；

对缺失数据敏感；

对于核的选择也是有技巧的（libsvm中自带了四种核函数：线性核、多项式核、RBF以及sigmoid核）：

第一，如果样本数量小于特征数，那么就没必要选择非线性核，简单的使用线性核就可以了；

第二，如果样本数量大于特征数目，这时可以使用非线性核，将样本映射到更高维度，一般可以得到更好的结果；

第三，如果样本数目和特征数目相等，该情况可以使用非线性核，原理和第二种一样。

对于第一种情况，也可以先对数据进行降维，然后使用非线性核，这也是一种方法。

**LR和SVM异同：**

不同点：1、损失函数不同

2、输入自变量不同（lr全部，svm是支持向量）

3、输出不同（lr是概率）

4、svm有约束条件，LR无约束

5、lr可解释性强，SVM先投影到更高维分类再投影到低维空间

6、svm对非线性问题更好更快实际上差不多，svm常用的核函数为高斯核和非线性核函数，LR也可以映射到高维，但采用核函数映射到高维更快

7、svm自带有约束条件正则，泛化能力比LR好，LR是无约束正则

相同点：

1、都是二分类

2、都采用正则化

3、都是监督学习

4、判别模型

**朴素贝叶斯优缺点：**

朴素贝叶斯属于生成式模型（关于生成模型和判别式模型，主要还是在于是否是要求联合分布），非常简单，你只是做了一堆计数。如果注有条件独立性假设（一个比较严格的条件），朴素贝叶斯分类器的收敛速度将快于判别模型，如逻辑回归，所以你只需要较少的训练数据即可。即使NB条件独立假设不成立，NB分类器在实践中仍然表现的很出色。它的主要缺点是它不能学习特征间的相互作用，用mRMR中R来讲，就是特征冗余。引用一个比较经典的例子，比如，虽然你喜欢Brad Pitt和Tom Cruise的电影，但是它不能学习出你不喜欢他们在一起演的电影。

优点：

朴素贝叶斯模型发源于古典数学理论，有着坚实的数学基础，以及稳定的分类效率。

对小规模的数据表现很好，能个处理多分类任务，适合增量式训练；

对缺失数据不太敏感，算法也比较简单，常用于文本分类。

缺点：

需要计算先验概率；

分类决策存在错误率；

对输入数据的表达形式很敏感。

朴素贝叶斯算法优点

1、对大数量训练和查询时具有较高的速度。即使使用超大规模的训练集，针对每个项目通常也只会有相对较少的特征数，并且对项目的训练和分类也仅仅是特征概率的数学运算而已。

2、支持增量式运算。即可以实时的对新增的样本进行训练。

3、朴素贝叶斯对结果解释容易理解。

二、朴素贝叶斯缺点

1、由于使用了样本属性独立性的假设，所以如果样本属性有关联时其效果不好。

KNN算法；

优点

1、KNN是一种在线技术，新数据可以直接加入数据集而不必进行重新训练

理论成熟，思想简单，既可以用来做分类也可以用来做回归；

可用于非线性分类；

训练时间复杂度为O(n)；

对数据没有假设，准确度高，对outlier不敏感；

缺点

计算量大；

样本不平衡问题（即有些类别的样本数量很多，而其它样本的数量很少）；

需要大量的内存；

人工神经网络

一、 神经网络优点

1、分类准确度高，学习能力极强。

2、对噪声数据鲁棒性和容错性较强。

3、有联想能力，能逼近任意非线性关系。

二、神经网络缺点

1、神经网络参数较多，权值和阈值。

2、黑盒过程，不能观察中间结果。

3、学习过程比较长，有可能陷入局部极小值。

)EM最大期望算法

EM算法是基于模型的聚类方法，是在概率模型中寻找参数最大似然估计的算法，其中概率模型依赖于无法观测的隐藏变量。E步估计隐含变量，M步估计其他参数，交替将极值推向最大。

EM算法比K-means算法计算复杂，收敛也较慢，不适于大规模数据集和高维数据，但比K-means算法计算结果稳定、准确。EM经常用在机器学习和计算机视觉的数据集聚（Data Clustering）领域。

什么时候用归一化？什么时候用标准化？

  （1）如果对输出结果范围有要求，用归一化。

  （2）如果数据较为稳定，不存在极端的最大最小值，用归一化。

  （3）如果数据存在异常值和较多噪音，用标准化，可以间接通过中心化避免异常值和极端值的影响。

他的经验：1） 在分类、聚类算法中，需要使用距离来度量相似性的时候、或者使用PCA技术进行降维的时候，第二种方法(Z-score standardization)表现更好。2）

在不涉及距离度量、协方差计算、数据不符合正太分布的时候，可以使用第一种方法或其他归一化方法。比如图像处理中，将RGB图像转换为灰度图像后将其值限定在

[0 255]的范围。

哪些模型必须归一化/标准化？

（1）SVM

  不同的模型对特征的分布假设是不一样的。比如SVM 用高斯核的时候，所有维度共用一个方差，这不就假设特征分布是圆的么，输入椭圆的就坑了人家，所以简单的归一化都还不够好，来杯白化才有劲。比如用树的时候就是各个维度各算各的切分点，没所谓。

（2）KNN

  需要度量距离的模型，一般在特征值差距较大时，都会进行归一化/标准化。不然会出现“大数吃小数”。

（3）神经网络

  1）数值问题

  归一化/标准化可以避免一些不必要的数值问题。输入变量的数量级未致于会引起数值问题吧，但其实要引起也并不是那么困难。因为tansig（tanh）的非线性区间大约在[-1.7，1.7]。意味着要使神经元有效，tansig( w1x1 + w2x2 +b) 里的 w1x1 +w2x2 +b 数量级应该在 1 （1.7所在的数量级）左右。这时输入较大，就意味着权值必须较小，一个较大，一个较小，两者相乘，就引起数值问题了。

  假如你的输入是421，你也许认为，这并不是一个太大的数，但因为有效权值大概会在1/421左右，例如0.00243，那么，在matlab里输入 421·0.00243 == 0.421·2.43，会发现不相等，这就是一个数值问题。

  2）求解需要

  a. 初始化：在初始化时我们希望每个神经元初始化成有效的状态，tansig函数在[-1.7, 1.7]范围内有较好的非线性，所以我们希望函数的输入和神经元的初始化都能在合理的范围内使得每个神经元在初始时是有效的。（如果权值初始化在[-1,1]且输入没有归一化且过大，会使得神经元饱和）

  b. 梯度：以输入-隐层-输出这样的三层BP为例，我们知道对于输入-隐层权值的梯度有2ew(1-a^2)\*x的形式（e是誤差，w是隐层到输出层的权重，a是隐层神经元的值，x是输入），若果输出层的数量级很大，会引起e的数量级很大，同理，w为了将隐层（数量级为1）映身到输出层，w也会很大，再加上x也很大的话，从梯度公式可以看出，三者相乘，梯度就非常大了。这时会给梯度的更新带来数值问题。

  c. 学习率：由（2）中，知道梯度非常大，学习率就必须非常小，因此，学习率（学习率初始值）的选择需要参考输入的范围，不如直接将数据归一化，这样学习率就不必再根据数据范围作调整。 隐层到输出层的权值梯度可以写成 2ea，而输入层到隐层的权值梯度为 2ew(1-a^2)x ，受 x 和 w 的影响，各个梯度的数量级不相同，因此，它们需要的学习率数量级也就不相同。对w1适合的学习率，可能相对于w2来说会太小，若果使用适合w1的学习率，会导致在w2方向上步进非常慢，会消耗非常多的时间，而使用适合w2的学习率，对w1来说又太大，搜索不到适合w1的解。如果使用固定学习率，而数据没归一化，则后果可想而知。

  d.搜索轨迹：已解释

（4）PCA