**优化问题：**

**第二个问题：L1 L2的作用，为什么有这样的作用？一般求L1的优化方法（坐标下降，LARS角回归）; L1为什么能让参数稀疏，L2为什么会让参数趋于较小值，L1优化方法; L2正则的本质？**

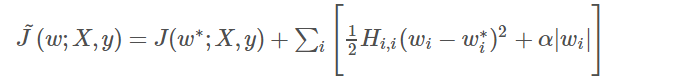
**答案：**

**L1与L2的作用：**

1. **L1作用：产生稀疏权值，特征权重系数可能为0，可以用于特征选择。**
2. **L2作用：让参数趋向于较小值，可以防止模型过拟合，一定程度上，L1也可以防止过拟合。**

**L1作用方式：**

1. **L1方式：当在原始损失函数后面添加L1正则化项时，相当于做了一个约束，变成了在约束问题下求出最小值的解 。在损失函数中引入L1正则项的时候，会遇到L1正则项的求导问题。将L1正则化目标函数采用二次近似分解成关于参数的求和**





**当最优解小于a/h的时候，正则化后w的最优值是0，在此w维度方向**

**上原来损失函数对正则化损失函数的贡献会被抵消；当大于的时候，正**

**化之后w值不为0，但是会减少一定的值。**

1. **L1优化方法：坐标轴下降法https://www.cnblogs.com/pinard/p/6018889.html**
   1. **数学依据：一个可微的凸函数J，其中Q是一个参数空间中的n维向量，如果存在一个点，在该点对应的每一个坐标轴的值处，凸函数在该坐标轴处都是最小值，那么该点就是一个全局最小值。**
   2. **坐标轴下降法在每次迭代过程中在当前点处沿一个坐标轴方向进行一维搜索，固定其他坐标轴方向，找到函数的极小值**
   3. **坐标下降分为外循环和内循环，外循环是对所有样本进行一次迭代，内循环是对坐标轴各位维度进行一次迭代。**
   4. **坐标轴下降法不需要利用梯度导数，只需要按照某一个坐标轴方向进行搜索最小值。**

**L2作用方式：**

1. **有带有L2正则项的损失函数求导时，可以发现在每步执行梯度下降时会将特征权重乘以一个权重向量，会使得特征权重逼近0。**

**L2正则的本质：限制解空间范围，缩小解空间，控制模型复杂度。**

**第三个问题：常见损失函数**

**答案：**

1. **0-1损失函数**
2. **平方损失函数**
3. **交叉熵损失函数**
4. **合页损失函数**

**第四个问题：SGD BGD MBGD**

**答案：**

1. **BGD：批量梯度下降：使用全部训练集进行优化，每次使用全部的训练集样本计算损失函数的梯度，然后使用学习率朝着梯度下降的方向更新模型参数。**

**优点：**

1. **每次更新都会朝着最终方向进行，能够保证最终收敛域极值点（凸函数收敛域全局极值点，非凸函数能够保证收敛域局部极值点）**
2. **能够并行实现**

**缺点：**

1. **由于使用的全部的样本进行每次更新，每次更新所需要的时间特征长，并且会消耗大量的内存。**
2. **不能在线更新模型参数**

**SGD：随机梯度下降：每次使用一个样本进行学习，使用学习率朝着梯度下降的方向进行更新。**

**优点：**

1. **由于每次只使用一个样本，每次迭代时间特别短。**
2. **能够在线更新模型参数。**
3. **训练速度快，例如样本量特别大时，可能用到其中一部分样本就收敛。**

**缺点：**

1. **每次更新并不一定朝着最终方向进行，会有波动存在。由于hesse矩阵病态问题存在。**
2. **不能并行实现**

**MBGD：小批量梯度下降：结合了BGD和SGD，在每次更新速度和迭代次数之间进行权衡。相对于梯度下降，每次只使用小部分样本进行训练，对内存的消耗程度低，提高了每次更新速度。相对于随机梯度下降，降低了更新的波动性，更新更加稳定，能够并行实现。**

**第五个问题：如何处理样本非均衡问题？数据不均衡如何解决，抽样得到的分类准确率如何转换为原准确率。**

**答案：从两个方面进行考虑：数据层面、算法层面**

**数据层面：**

**样本不平衡问题不是特别严重：**

1. **过采样：基本思想就是通过改变训练数据的分布来消除或减少数据的不平衡，通过增加少数类样本来提高少数类的分类性能，最简单方法是简单复制少数类样本，缺点是容易过拟合，没有给少数类增加新的信息；改进的过采样方法通过在少数类中加入随机高斯噪声或产生新的合成样本，例如SMOTE。**
2. **欠采样：通过减少多数类样本来提高分类器性能，最简单的方法是通过随机地去掉一些多数类样本来减少多数类的规模，缺点是会丢失很多信息，不能充分利用信息；还可以采用聚类的方法，将大类聚类成多个簇，使用簇中心作为样本，和小样本一起训练，优点是在一定程度上保留了大类在特征空间分布特征，又降低了大类的数目。**
3. **SMOTE：基本思想是构造在原数据集中不存在的新样本；基于距离度量选择两个或更多的相似样本，然后选择其中一个样本，并随机选择一定数量的邻居样本对选择的样本添加噪声，每次处理一个属性，优点是合理的对小样本的特征分布进行一定的扩展）**
4. **对小类错分样本加权惩罚：对分类器的小类样本数据增加权值，降低大类样本的权值，从而使得分类器将重点集中在小类样本身上。一个具体做法就是，在训练分类器时，若分类器将小类样本分错时额外增加分类器一个小类样本分错代价，这个额外的代价可以使得分类器更加“关心”小类样本。**
5. **分治方法：将大类中样本聚类到L个聚类中，然后训练L个分类器；每个分类器使用大类中的一个簇与所有的小类样本进行训练得到；最后对这L个分类器采取少数服从多数对未知类别数据进行分类，如果是连续值（预测），那么采用平均值。**

**样本不平衡问题特别严重时：**

1. **一分类：当正负样本比例严重失衡时，靠单纯的采样和数据合成不能很好解决问题，因为上面方法虽然解决了正负样本比例问题，但是却偏离了原始数据的真实分布情况，模型不能反映真实的情况，有很大偏差。最常见一分类方法是one-class-svm，基本思路：利用高斯核函数将样本空间映射到核空间，在核空间中找到一个能够包含多数类所有数据的球体，在判别时，若在球体内，则判别为多数类，否则少数类。**
2. **离群点异常检测：基于密度的异常点检测、基于邻近度的异常点检测，统计的方法，要对具体问题具体分析。**

**算法层面：**

1. **代价敏感：设计出代价敏感的分类算法。通过对小样本赋予较高的代价，大样本赋予较小的代价；还可以在损失函数中引进类似召回率的评价指标。**
2. **使用bagging、boosting、ensemble方法**
3. **使用auc、roc等方法对模型效果进行评价。**

**抽样得到的准确率如何转换：**

**第六个问题：如何处理数据缺失？**

**答案：存在两种方法，分为删除缺失值和缺失值插补**

**缺失值删除：**也就是将存在遗漏信息属性值的对象（元组，记录）删除，从而得到一个完备的信息表。这种方法简单易行，在对象有多个属性缺失值、被删除的含缺失值的对象与信息表中的数据量相比非常小的情况下是非常有效的，类标号（假设是分类任务）缺少时通常使用。然而，这种方法却有很大的局限性。它是以减少历史数据来换取信息的完备，会造成资源的大量浪费，丢弃了大量隐藏在这些对象中的信息。在信息表中本来包含的对象很少的情况下，删除少量对象就足以严重影响到信息表信息的客观性和结果的正确性；当每个属性空值的百分比变化很大时，它的性能非常差。因此，当遗漏数据所占比例较大，特别当遗漏数据非随机分布时，这种方法可能导致数据发生偏离，从而引出错误的结论。

**缺失值插补：因为一个属性值的缺失而放弃大量的其他属性值，删除浪费了很多信息。**

1. **填固定值，例如数据是表示wifi连接强度，对于缺失值可以填充-100db。**
2. 绝对均值**：**
3. **条件均值：能够估计已知数据丢失值的概率密度函数，计算期望值。**
4. 条件概率分布填充：**根据缺失值的统计特性进行填充，不是填充固定值，而是有其分布的随机抽取值替代**

**第七个问题：如何处理异常点，异常值的影响，如何消除？**

**答案：**

异常值，即在数据集中存在不合理的值，又称离群点。

影响：异常值可以大幅度改变数据分析和统计建模的结果。

统计判断方法：

1. 正太分布z分数
2. 箱型图分析
3. 根据先验知识进行判别

处理方法：

1. 删除
2. 视为缺失值，按照缺失值处理
3. 平均值修正

**第八个问题：特征工程之连续特征与离散特征处理方法介绍。**

**答案：**<https://blog.csdn.net/u014135752/article/details/80789251、>

**第九个问题：特征怎么选择（基尼系数、信息增益比、fisher准则、PCA）**

**答案：**

**特征选择和降维异同点：两者达到的效果是一样的，都是试图去减少特征数据集中属性的数目；两者采用方式不一样，降维主要是通过属性间的关系，如组合不同的属性得到新的属性，改变原来的特征空间；而特征选择的方法是从原始特征数据集中选择出子集，是一种包含关系，没有更改原来的特征空间。**

**特征数量过多的影响：**

1. **特征个数越多，分析特征、训练模型所需的时间久越长，模型也会越复杂。**
2. **特征个数越多，容易引起维数灾难。**
3. **特征个数越多，容易导致机器学习中经常出现特征稀疏的问题，模型效果下降。**
4. **对于模型来说，可能会病态情况，即解出来的参数会因为样本的微小变化而出现大的波动。**

**特征选择的目标：**

1. **剔除不相关，冗余，没有差异刻画能力的特征，提高预测的准确性**
2. **构造更快，消耗更低的预测模型**
3. **能够对模型有更高的理解和解释。**

**特征选择的作用：减少特征数量、降维，使模型泛华能力更强，减少过拟合。**

**选择方法：一般是先使用统计的方法去掉部分与目标变量不相关的特征或者方差很小的特征，从而减少后续搜索的规模；然后使用树模型对剩余的特征和样本进行选择。**

**统计方法（Filter）：一般用作预处理，与算法无关。**

1. **使用方差选择方法：计算各个特征的方差，选择方差大于阈值的特征，方差小说明数据没有什么区别，对目标的刻画能力很差。**
2. **卡方检验：主要比较两个变量之间的关联性分析。**
3. **Pearson相关系数：两个变量的协方差除以两个变量的标准差，衡量的是变量之间的线性相关性，绝对值越大，表示线性相关性越强，绝对值越小，表示线性相关性越弱。**
4. **距离度量：好的特征子集应该使得属于同一类样本的距离尽量小，属于不同类之间的距离尽量远。**
5. **信息增益：类似于决策树中衡量方法。**

**Wrapper方法：将特征子集的选择看做是一个搜寻最优问题，生成不同的特征组合，对组合进行评价，再与其他组合进行比较。**

1. **启发式搜索：采用贪心的思想，某些算法没有考虑特征之间的相关性，而单纯考虑单个特征对最终结果的影响，启发式考虑了特征之间相关性。**

**评价准则是学习算法的结果。贪心的算法容易陷入局部最优。**

* 1. **序列前向选择**
     1. **算法描述：特征子集从空集开始，每次选择一个特征加入特征子集，使得特征函数最优。就是说，每次都选择一个使得评价函数最优的特征加入。**
     2. **评价：缺点是只能加入特征而不能去除特征，例如：特征A完全依赖于B和C，如果先加入B C，则A是多余的，如果先加入了A，也会将B和C加进去，这样就有多余的特征A。**
  2. **序列后向选择**
     1. **算法描述：从特征全集O开始，每次从特征集O中剔除一个特征x，使得剔除特征x后评价函数值达到最优。**
     2. **算法评价：序列后向选择与序列前向选择正好相反，它的缺点是特征只能去除不能加入。**
  3. **双向搜索**
     1. **算法描述：使用序列前向选择(SFS)从空集开始，同时使用序列后向选择(SBS)从全集开始搜索，当两者搜索到一个相同的特征子集C时停止搜索。**
  4. **增L去R选择**
     1. **算法描述：算法从空集开始，每轮先加入L个特征，然后从中去除R个特征，使得评价函数值最优**
     2. **增L去R选择算法结合了序列前向选择与序列后向选择思想， L与R的选择是算法的关键。**
  5. **树模型/回归模型进行特征选择：采用回归模型，例如lr，去掉权重低的特征，或者利用决策树，对特征进行选择。**

**基于惩罚项的特征选择：**

1. **L1正则化：L1正则化将系数w的l1范数作为惩罚项加到损失函数上，由于正则项非零，这就迫使那些弱的特征所对应的系数变成0。L1正则化像非正则化线性模型一样也是不稳定的，如果特征集合中具有相关联的特征，当数据发生细微变化时也有可能导致很大的模型差异。**
2. **L2正则化：L2正则化对于特征选择来说一种稳定的模型，不像L1正则化那样，系数会因为细微的数据变化而波动。所以L2正则化和L1正则化提供的价值是不同的，L2正则化对于特征理解来说更加有用：表示能力强的特征对应的系数是非零。**

**第十个方法：降维的主要方法 PCA原理**

**答案：**

1. **PCA： 原理见纸质资料**
2. **SVD:**
3. **FA**
4. **ICA**

**第十二个问题：KKT条件是什么？完整描述**

**答案：KKT条件是确定某点为极值点的必要条件，如果讨论的是凸规划问题，那么也是充分条件。**

**对凸优化问题,KKT一共包含三个条件：**

1. **拉格朗日函数对自变量和拉格朗日乘子的一阶导数为0**
2. **最优解必须满足对应的等式约束、不等式约束以及拉格朗日乘子的约束**
3. **互补松弛条件：不等式约束的拉格朗日乘子和不等式约束乘积为0.**

**第十三个问题：最小二乘与极大似然函数的关系？**

**最小二乘法解释：最小二乘是基于欧式距离，最合理的估计函数的参数估计量应该使得模型能最好地拟合样本数据，也就是估计值和观测值之差的平方和最小。**

**极大似然法的解释：极大似然法是预计概率分布。最合理的参数估计量应该使得从模型中抽取该n组样本后观测值的概率最大，也就是概率分布函数或者说是似然函数最大。在最大似然中，通过选择参数，使得已知数据在参数的条件下出现的概率最大，使用似然函数来衡量概率。即利用已知的样本结果，反推最有可能（最大概率）导致这样结果的参数值。**

**两者之间的关系：如果我们认为观察到的数据是由模型的真实输出加上高斯噪声得到的，那么对模型的参数的最大似然估计即等价于最小二乘法。即在误差服从高斯分布的情况下，最小二乘法等价于极大似然估计。**

**第十四个问题：分类的评价标准，准确度，AUC，召回率等等**

**答案：见纸质资料，不再描述**

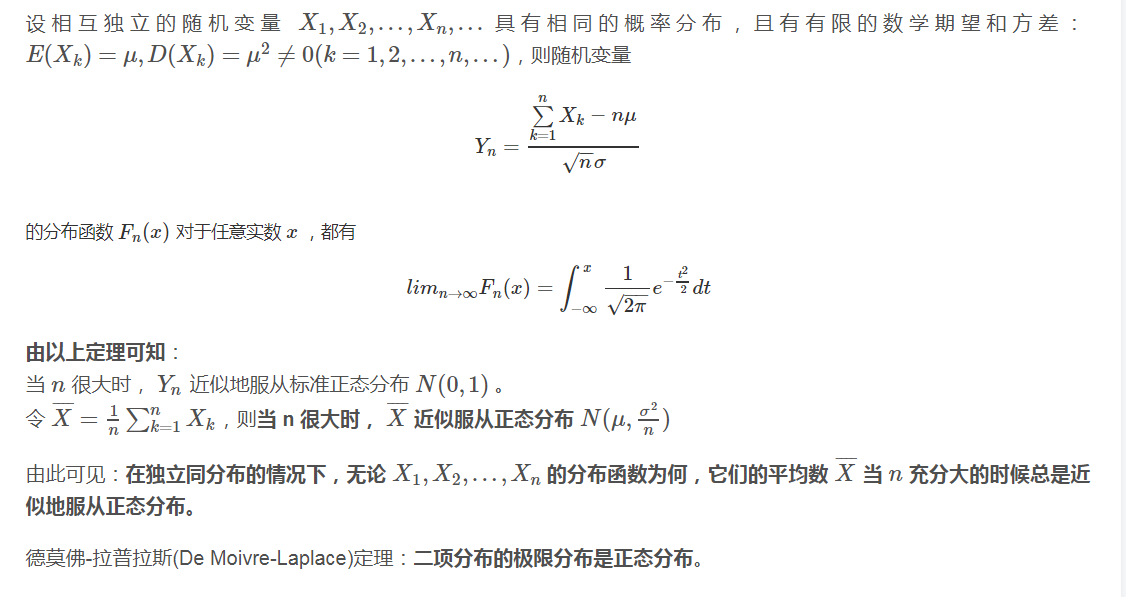
**第十七个问题：线性模型为何用的最小二乘作为损失函数而不用似然函数或者交叉熵？**

**答案：一般对于误差而言，我们假设他是一个均值为0的高斯分布（此处问题：为什么我们常常假设误差之类都是服从高斯分布），样本之间是相互独立的，使用最大似然，我们既可以得到损失函数是平方损失函数，即可以认为用的是最小二乘作为损失函数。**

**第十八个问题：为什么常常假设数据服从高斯分布？**

**答案：主要是由于中心极限定理**

**中心极限定理：如果一个事物受到多种因素的影响，不管每个因素是什么分布，因素加和后，结果的平均值服从正态分布。**



**第十九个问题： 拉格朗日法，对偶问题，以及好处**

**答案：**

**拉格朗日：是一种寻找有等式约束条件的函数的最大值、最小值的优化方法，在求解函数最优值过程中，约束条件会给求取最优值带来困难，拉格朗日法可以有效解决这类问题。**

**对偶问题：原始问题是先对拉格朗日函数先取最大化，再取最小化；对偶问题是对拉格朗日函数先取最小化，再取最大化；调换对偶问题中对拉格朗日函数取最大化、最小化的顺序即可得到与原问题等价的优化问题。**

**KKT条件：是拉格朗日乘子法的拓展,是一种求取含不等式约束条件的函数最优值的方法。**

**模型问题：**

**第一个问题：交叉验证的几种方法？**

**答案：**

1. **K折交叉验证：将数据分成K份，每次取一个作为测试集，其余作为训练集，求k词模型准确率的平均值。**
   1. **评价：严格来说不算交叉验证，结果具有随机性，并不具有说服性。**
2. **简单交叉验证：取出一部分作为测试集，其余是训练集**
   1. **结果较有说服性**
3. **留一法：每次只取出一个样本作为测试集，其余作为训练集**
   1. **每一回合几乎所有样本都用作训练集，最接近原始样本分布，评估结果可靠，但是计算量特别大。**

**第二个问题：模型融合怎么做？**

**融合：Ensemble Learning 是指将多个不同的 Base Model 组合成一个 Ensemble Model 的方法。它可以同时降低最终模型的 Bias 和 Variance（证明可以参考这篇论文，我最近在研究类似的理论，可能之后会写新文章详述)，从而在提高分数的同时又降低 Overfitting 的风险。**

**模型融合条件：**

1. **Base Model 之间的相关性要尽可能的小。**这就是为什么非 Tree-based Model 往往表现不是最好但还是要将它们包括在 Ensemble 里面的原因。Ensemble 的 Diversity 越大，最终 Model 的 Bias 就越低。
2. **Base Model 之间的性能表现不能差距太大。**这其实是一个 **Trade-off**，在实际中很有可能表现相近的 Model 只有寥寥几个而且它们之间相关性还不低。但是实践告诉我们即使在这种情况下 Ensemble 还是能大幅提高成绩。

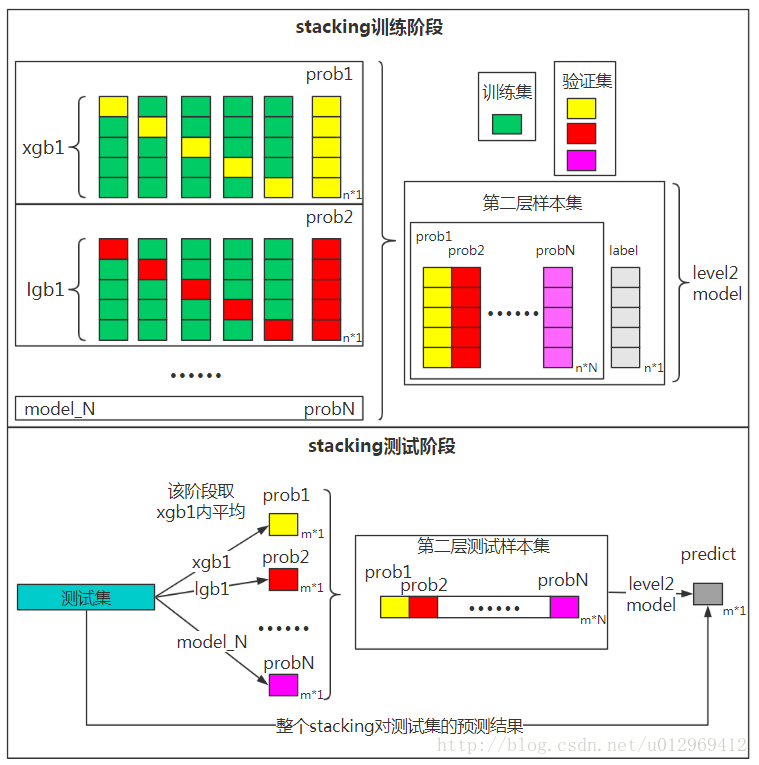
**简单融合：**

1. **多数表决融合：默认所有模型是一样重要的，对于分类来说，取出现次数最多的那个认为是正确答案。**
2. **加权表决融合：通常对表现好的模型赋予更大的权值。**
3. **对结果取平均：可以减少过拟合，模型对噪声的考虑会因为取平均的原因而减少。**

**Stacking：使用基础模型的预测作为堆叠模型的特征。因此，堆叠模型能够辨别哪个模型表现良好，哪个模型表现不佳。**

**原理：使用很多基分类器，比如逻辑回归，随机森林，GBDT等分类器，预测的结果作为特征，再用分类器预测作为最终结果。**

**一图看懂stacking：https://blog.csdn.net/u012969412/article/details/76636336**



**Blending：用不相交的数据训练不同的基模型，将它们的输出取平均**

**两层blengding：训练集划分为两部分d1 d2，测试集为test**

**第一层：用d1训练多个模型，将其对d2和test的预测结果作为第二层的新特征**

**第二层：用d2的新特征和标签训练新的分类器，然后把test的新特征输入，结果就是最终的预测值。**

**优点：**

1. **比stacking简单（不需要用k折交叉验证）**

**缺点：**

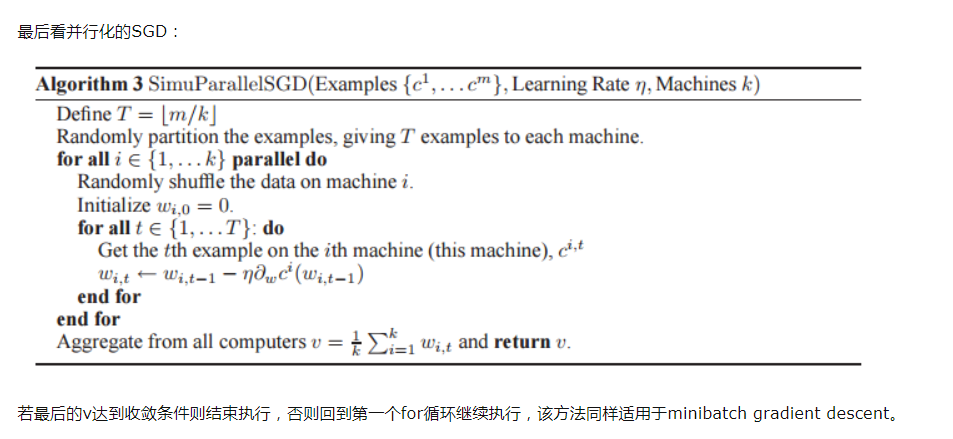
1. **对数据的利用不充分，stack阶段只利用了部分数据。**
2. **可能会过拟合**

**第三个问题：sgd的并行化思路，sgd并行化思路展开，一个并行任务运行很慢怎么办？**

**答案：传统的SGD在本质上的串行性使其不适合在大数据上并行。**

**基本思想：允许多个模型副本同时处理样本集的不同子集，并周期性地对各个模型副本的结果进行归并。**

**异步：**<https://blog.csdn.net/itplus/article/details/31831661>**（模型并行化+数据并行化）**



**第四个问题：了解各种优化算法不？梯度下降和随机梯度下降的区别？牛顿法和拟牛顿法的区别？为啥提出拟牛顿？**

**答案：见纸质资料**

**逻辑斯特回归：**

**第一个问题：为什么要用sigmoid函数作为分类的概率分布?**

答案：有逻辑斯特回归的约束条件和最大熵原理可以推导出多分类的概率函数就是softmax函数。推导过程见资料P1-2。

**第二个问题：简单介绍逻辑斯特回归**

答案：逻辑回归是对数据类别的概率进行建模的，假设数据服从伯努利分布，通过极大似然函数，运用梯度下降法求解参数，达到二分类的目的。

**第三个问题：为什么使用交叉熵？**

答案：

1. 逻辑回归极大似然函数取对数就是交叉熵函数，取对数的好处是方便计算和防止数值下溢。
2. 在逻辑回归模型中，当使用交叉熵函数时，梯度下降只和x, y，预测概率有关，与sigmoid函数的梯度无关，损失函数训练求解参数会很快，更新速度稳定。

**第四个问题：为什呢不用平方损失函数？**

答案：

1. 平方损失函数第梯度与sigmoid的梯度有关，而sigmoid函数在定义域内的梯度不大于0.25，在自变量比较大的时候，sigmoid的梯度几乎为0，梯度下降特别缓慢。
2. LR使用平方损失函数无法保证损失函数是凸函数，在梯度优化过程中，求解的解有可能是局部最小，不是全局的最小值
3. 对数损失函数，因为取对数，求导数很方便。

**第五个问题：若存在很多高相关度的特征，会有什么影响？**

答案：若损失函数最终收敛的情况下，很多高相关的特征不会影响分类的准确度，但是高相关的特征权值会变小，并且训练速度会变慢。

**第六个问题：如果去掉高相关的特征会有什么好处？**

答案：

1. 特征区分度变大，模型的可解释性变得更高，重要的特征权值会更大。
2. 会提高训练速度，特征变多，参数会变多，训练时间增加。

**第七个问题：优缺点**

答案：

优点：

1. 形式简单，可解释性较好，特征权重大，说明特征可能更加重要。
2. 模型效果一般还不做，如果特征工程做的好，也能取得不错得效果。
3. 训练速度较快，梯度下降时只有训练速度和sigmoid函数梯度无关，计算量之和特征数量有关系，可以并行处理。
4. 资源占用较小，只需要存储各个维度的特征值。在时间和内存需求上相当高效，可以适用于分布式实现，还有在线算法。
5. 结果方便调整，可以调整阈值获得不同的分类效果。

缺点：

1. 准确率并不是很高，因为还是类似于线性，很难去拟合复杂的分布。
2. 很难处理样本不均衡问题，正负样本不均衡是，把所有样本判别为一类损失函数也特别小
3. 很难处理非线性数据，数据特征有缺失的时候或者特征空间很大的时候表现并不好。
4. 无关根据特征权重筛选特征。

**第八个问题:sigmoid函数有什么优缺点？**

答案：

优点：

1. 平滑线，可以很容易抑制误差，例如当x=10和x=20时，sigmoid的值差不多一样，即数据如果异常对sigmoid影响很小。
2. Sigmoid函数求导很容易。
3. 可以作为概率来解释模型。

缺点

1. Sigmoid函数过饱和，丢失梯度，当输出值接近0或者1的时候，梯度也接近于0，如果函数的梯度与sigmoid梯度有关，则函数梯度会变得很小，梯度下降速度会变慢。
2. Sigmoid函数输出不是0中心的，0中心解释：当激活函数输入是0的时候，输出也是0。Sigmoid不满足这个要求
3. Sigmoid中含有指数运算，特别耗费计算资源，幂指数加1，数据计算量翻一倍。

**第九个问题：逻辑回归与线性回归的区别？**

答案：

1. 目的不一样：线性回归是用于回归问题，而逻辑回归是用于分类问题。
2. 目标不一样：线性回归是对预测值与真实值之间的误差进行优化，并且假设误差服从高斯分布；而逻辑回归是对分类的概率进行优化，假设数据服从伯努利分布。
3. 损失函数不一样：线性回归采用的是最小二乘法，而逻辑回归采用的最大似然函数

**第十个问题：逻辑回归对特征有什么要求，是否需要做离散化，离散化好处与坏处？**

答案：需要做离散化处理

优势：

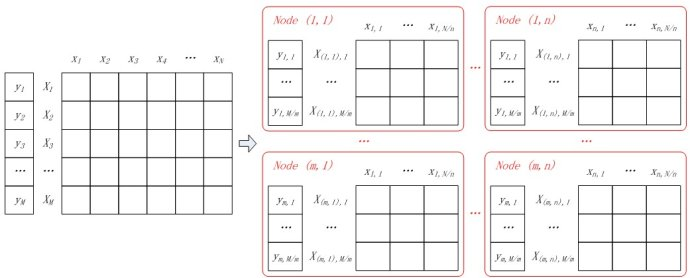
1. 离散后特征向量更加稀疏，稀疏向量内积乘法运算速度快，计算结果方便存储；
2. 离散化之后的特征对异常数据具有很强的鲁棒性：比如一个特征是年龄大于30岁，异常年龄300岁会对模型有很大影响。离散化之后300岁子对应于一个特征，如果训练数据中没有出现300岁，自然对模型没有影响，如果训练数据中出现了，也只是这个特征对模型有影响，正常年龄的特征对模型不会有影响（离散化之后特征的权值变小了）。
3. 逻辑回归属于广义线性模型，表达能力受限，单变量离散化之后，相当于引入了非线性，能够提升模型表达能力，对复杂的分布更好的拟合。例如：一个特征是连续的，对它进行求导，导数也是线性的，如果进行离散化，导数是一个分段的线性函数，相当于引入了非线性。
4. 特征离散化之后可以进行特征交叉，例如A特征离散化之后为M个特征，B特征离散化之后为N个特征，交叉可以得到M\*N个变量，进一步引入了非线性，提升了模型表达能力。
5. 特征离散化之后，模型更加稳定，比如对用户年龄进行离散化之后，20-30作为一个区间，不会因为一个用户年龄增长就会产生比较大的变化；但是划分区间很重要，在相邻处的样本会相反。
6. 特征离散化之后，对模型进行了简化，降低了模型过拟合的风险。当使用离散特征时，如果这个特征权重比较大，模型会很依赖这个特征，这个特征的一个微笑变化就是对最终结果产生比较大的变化，即泛华能力变差，容易过拟合。离散化之后，权重变为多个，之前连续特征对模型影响力被分散弱化，降低了过拟合的风险。

**第十一个问题：逻辑回归的参数是否可以分布式求解，如何做到分布式？**

**答案：可以并行求解。**

**过程：**

1. **将M个样本的标签构成一个M维的向量，M个N维特征向量构成一个M\*N的样本矩阵。其中特征矩阵每一行为一个特征向量(M行)，列为特征维度（N列）**
2. **数据分割：假设计算节点排成m行n列，按行将样本进行划分，每个计算节点分配M/m个样本特征向量和标签；按列对特征向量进行划分，每个节点上特征向量分配N/n维特征。**



1. **目标函数的梯度计算公式，依赖于两个计算结果，特征权重向量和特征向量x（i）的相乘, 标量（预测值与真实值之差）与的相乘。可以将目标函数的梯度分成两个并行化计算步骤和两个结果归并步骤。**
2. **行节点并行化：行号相同的各个节点并行计算点乘，即预测值。**
3. **归并：对行号相同的节点归并点乘结果，计算得到的结果需要返回到该行的所有节点中，即每行的n个节点共享计算结果。**
4. **列节点并行化：列好相同的每个节点独立计算标量与特征向量的第j个分量之间的乘积。**
5. **归并：列号相同的节点进行归并，得到第j个权重向量更新之后的值。**

**第十三个问题：LR为啥是个线性模型？**

**答案：本质就是线性的，只是特征到结果映射用的是sigmoid函数，或者说回归边界是线性的，即P(Y=1|x)=P(Y=0|x)时有W\*x=0 。**

**第十四个问题：有的逻辑回归损失函数中为啥要加-1\*m ？**

**答案：防止损失函数受到样本规模的影响，例如在小批量梯度下降中，如果每次样本不同，损失函数函数会随着样本规模而变化。**

**第十五个问题：softmax损失函数推导？**

**答案：见资料P3。**

**第十六个问题：样本不均衡造成的影响？如何处理？**

**答案：**

**影响：逻辑回归目标函数考虑的是最大化每个实例被正确分类概率的对数之和，而不考虑该实例是少数类（负类）还是多数类（正类），这样在分类时负类样本全部判别为负类或全部判别为正类对损失函数影响不大，即优化损失函数很难正确对负类进行划分，这样会导致将更多的少数类误分为多数类。**

**解决方法：**

1. **数据层面：第一个：重新采样使得新的数据集相对平衡，包括过采样和欠采样。**
2. **算法层面：代价敏感学习，组合方法、基于局部聚类。**
3. **代价敏感学习：在训练过程中不同的类分类错误会导致不同的惩罚力度，还可以在损失函数中考虑不同类的召回率重新定义损失函数**
4. **组合方法：构建多个基分类器，组合各个基分类器的分类结果，得出最终的结果，对逻辑回归来说，对多数类使用欠采样，训练多个模型，和随机森林一样，求平均。**

**SVM：**

**第一个问题：简单介绍SVM？**

**答案：SVM是一种二分类模型，基本模型是定义在特征空间上的间隔最大的线性分类器，学习目标是在特征空间中找到一个分离超平面，能够将训练数据正确分类，并且使得分离超平面关于训练数据的间隔最大化，形式化为一个求解凸二次规划问题，通过优化凸二次规划问题求解参数，达到二分类的问题。**

**第二个问题：SVM和感知机的异同？**

**答案：**

**相同**

1. **都是解决分类问题**
2. **都是寻找将数据线性可分的超平面**
3. **两者都可以使用核技巧。**

**不同：**

1. **感知机考虑的是误分类样本点，而svm考虑的支持向量来学习分类器。**
2. **感知机学习的分离超平面不唯一，svm学习的分离超平面是唯一的。**
3. **SVM比感知机考虑的更多，不仅要讲训练数据分开，还要使得分离间隔最大化，比感知机要求更严格，自带L2正则化**
4. **SVM引入松弛变量可以处理近似线性可分数据和线性不可分数据，感知机不可以。**

**第三个问题：如何处理SVM中的过拟合？**

**答案：SVM过拟合主要是因为训练数据中的异常点，这些点严重偏离正常位置，决定SVM最优分类超平面的恰恰是那些占少数的支持向量，如果支持向量中存在异常点，那么很容易过拟合。解决过拟合的方法是引入松弛变量，引入松弛变量能够使SVM在一定程度上容忍异常点存在，因为引入松弛变量后，所有点到超平面的距离约束不需要大于等于1了。但是如果对异常点过度容忍会导致很多超平面都是最优超平面，可以对松弛变量进行L2正则化处理，即加上松弛变量的平方L2范数，即希望松弛变量存在来解决异常点问题，又不希望松弛变量太大导致分类解决太差。**

**第四个问题: 松弛变量以及惩罚变量相关知识**

**答案：**

**松弛变量：**

1. **松弛变量主要是为了解决近似线性可分数据中存在的离群点问题，使得分离超平面能够在一定程度上容忍离群点的存在。**
2. **只有离群点才有对应的松弛变量。离群点距离分类超平面越远，松弛变量越大，非离群点松弛变量为0。**
3. **惩罚因子决定了又多重视离群点带来的损失，显然当所有离群点的松弛变量的和一定时，惩罚因子越大，对目标函数的损失越大。惩罚因子大代表着模型对离群点的容忍程度，惩罚因子越大，模型对松弛变量的容忍程度越小，当松弛变量无穷大时，模型退化成“硬间隔分类器”.**
4. **惩罚因子是一个需要实现指定的值，可以使用参数寻优的过程选择合适的惩罚因子。**

**第五个问题：样本不均衡对SVM的影响？如何解决？**

**答案：**

**影响：如果两个类别的样本数量差异很大，例如正类的样本远远大于负类的样本，由于学习目标的最大化间隔，所以最优分离超平面会偏向负类方向，样本不平衡程度越大，分离超平面的平移程度就越大。**

**解决样本不均衡问题：给样本数量少的负类更大的惩罚因子，更大的惩罚因子表示模型更加重视这部分样本；给样本数量多的更小的惩罚因子。样本不平衡之所以会造成最优分离超平面偏离，真实原因是样本数量少的样本分布不够广阔，所以根据两类样本的分布程度来确定惩罚因子，分布半径大的样本拥有更小的惩罚因子。**

**第五个问题：SVM对偶性的作用，为什么求解对偶问题更加高效？**

**答案：**

**等式约束的最优化问题：**

**目标是寻找在若干等式约束条件下的最小值，当寻找约束的最优点时，约束虽然减少了需要搜寻的范围，但是却使问题变得更加复杂，为了使得问题更加易于处理，我们方法是把目标函数和约束全部融入一个新的函数，即拉格朗日函数，通过拉格朗日函数来寻找最优点。**

**SVM-不等式约束问题：**

**同样将拉格朗日函数应用到不等式约束上，必须使得不等式约束的拉格朗日乘子正负性与不等式约束的方向相反，否则拉格朗日函数趋向于无穷大，当拉格朗日乘子满足条件时，可以将不等式约束问题转换的拉格朗日函数，即对偶问题的原始形式。**

**如果直接对原始问题进行求解，由于不等式拉格朗日乘子的不等式约束存在，还是相当于求解一个不等式约束问题，求解过程很难实现。原始问题是一个极小极大问题，将原始问题的极小极大进行对转，转变成极大极小问题，即原问题的对偶问题，可以看出原始问题的最优值一定大于等于对偶问题的最优值。**

**在一定的条件下，即目标函数和不等式约束是凸函数，等式约束是仿射函数，并且不等式约束是严格可行的，则原始问题和对偶问题的最优值相同的充分必要条件是原始问题的最优解和对偶问题的最优解满足KKT条件。**

**KKT条件库恩-塔克条件：见上，不再解释。**

**SVM对偶性作用：**

1. **SVM对偶问题更容易求解，在对偶问题中，只用求解alpha系数，而只有支持向量的alpha系数不为0。**
2. **可以引入核函数，进而将SVM推广到非线性分类问题。**

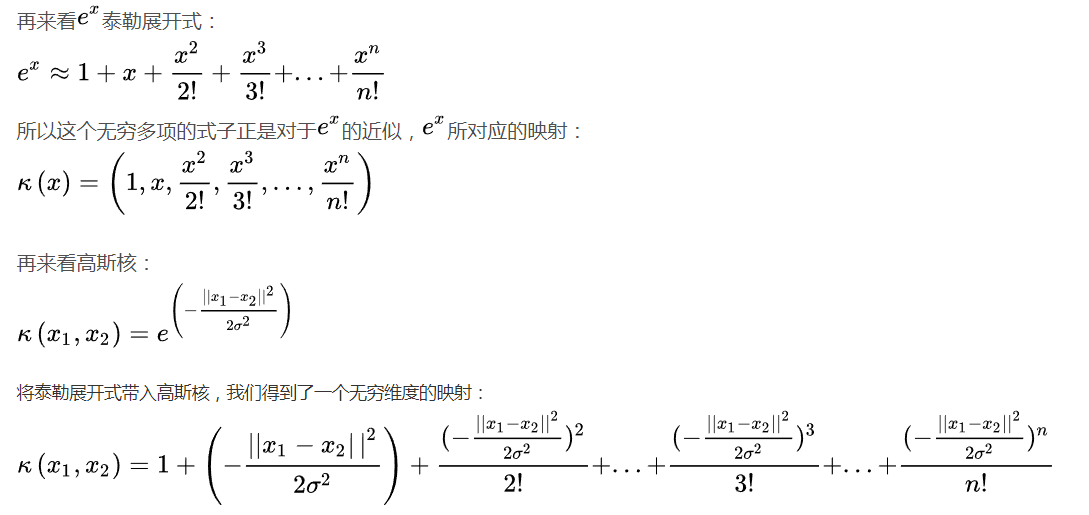
**第六个问题：SVM引入核函数本质？核函数有哪些，区别是什么，优缺点？**

**答案：**

**SVM引入核函数本质：通过非线性变换将输入空间对应于一个高维特征空间，然后在新的空间中使用线性分类的方法从训练数据中学习分类问题。**

**常见核函数：**

1. **线性核函数：主要用于线性可分的情况，特征空间与输入空间维度是一样的，其参数少速度快，对于线性可分数据，效果理想。即线性支持向量机**
2. **多项式核函数：多项式核函数可以实现将低维的输入空间映射到高纬的特征空间，但是多项式核函数的参数多，当多项式的阶数比较高的时候，核矩阵的元素值将趋于无穷大或者无穷小，计算复杂度会大到无法计算。即数值的上溢和下溢问题。**
3. **高斯核函数：高斯径向基函数是一种局部性强的核函数，其可以将一个样本映射到一个更高维的空间内，该核函数是应用最广的一个，无论大样本还是小样本都有比较好的性能，而且其相对于多项式核函数参数要少，因此大多数情况下在不知道用什么核函数的时候，优先使用高斯核函数。**
4. **为什么高斯核函数是无限维度：主要是由于指数的泰勒展开：**



**如果方差选择很大的话，在高次特征上的权重衰减的非常快，所以实际上可以近似看做一个低维度空间；如果高斯核函数的方差特别小，在高次特征上的权重衰减的非常慢，可以看做映射到一个高纬度空间，分离超平面局部变化会特别大，会将任意的数据映射为线性可分，会过度拟合训练数据，造成过拟合。**

**第七个问题：为什么svm可以使用核函数，LR不可以使用**

**答案：分类模型的结果是计算分离超平面，在计算过程中，SVM算法中只有支持向量参与了核计算，因为在SVM中w参数只和支持向量有关。然而在LR里，w参数和所有样本都相关，即每个样本点都参与了超平面的计算过程，在LR中应用核函数，每个样本点都必须参与核函数计算，计算复杂度非常高。**

**第八个问题：SVM与LR的异**

**答案：**

**相同：**

1. **LR和SVM都是分类算法。**
2. **如果不考虑核函数，LR和SVM都是线性分类算法，也就是说他们的分类决策面都是线性的。**
3. **LR和SVM都是监督学习算法**
4. **LR和SVM都是判别模型。判别模型会生成一个表示P(Y|X)的判别函数（或预测模型），而生成模型先计算联合概率p(Y,X)然后通过贝叶斯公式转化为条件概率。简单来说，在计算判别模型时，不会计算联合概率，而在计算生成模型时，必须先计算联合概率。或者这样理解：生成算法尝试去找到底这个数据是怎么生成的（产生的），然后再对一个信号进行分类。基于你的生成假设，那么那个类别最有可能产生这个信号，这个信号就属于那个类别。判别模型不关心数据是怎么生成的，它只关心信号之间的差别，然后用差别来简单对给定的一个信号进行分类。常见的判别模型有：KNN、SVM、LR，常见的生成模型有：朴素贝叶斯，隐马尔可夫模型。当然，这也是为什么很少有人问你朴素贝叶斯和LR以及朴素贝叶斯和SVM有什么区别。**

**不同：**

1. **loss function不同：不同的loss function代表了不同的假设前提，也就代表了不同的分类原理。逻辑回归方法基于概率理论，假设样本为1的概率可以用sigmoid函数来表示，然后通过极大似然估计的方法估计出参数的值，支持向量机​基于几何间隔最大化原理，认为存在最大几何间隔的分类面为最优分类面。**
2. **在线学习：SVM不支持在线学习，而LR支持在线学习**
3. **样本不平衡问题：支持向量机只考虑局部的边界线附近的点，而逻辑回归考虑全局（远离的点对边界线的确定也起作用），逻辑回归中改变任何样本都会引起决策面的变化。SVM不太依赖于数据的分布，所以数据是否平衡影响不是很大（有影响的）；LR依赖于数据的分布，受不平衡数据的影响较大，一般需要先对数据做balancing。**
4. **核函数：在解决非线性问题时，支持向量机采用核函数的机制，而LR通常不采用核函数的方法。解释见上。**
5. **规范化：线性SVM依赖数据表达的距离测度，所以需要对数据先做normalization，LR不受其影响。**
6. **结果可解释程度：LR可以给出每个点的概率，而SVM是非概率的。**
7. **泛化性能：LR不使用正则化项的时候，很容易过拟合，特征多的时候，模型会变得复杂；SVM的损失函数就自带正则！！！（损失函数中的1/2||w||^2项），这就是为什么SVM是结构风险最小化算法的原因。**
8. **样本特点：SVM对小样本高纬度数据的效果比较好，SVM学习的参数少，所以需要的样本少，而且参数只与支持向量有关，与特征数量无关，可以处理高维；而LR的参数是跟特征数量呈正比的。**

**树模型：**

**第一个问题：Bagging和boosting了解吗？**

**答案：**

**Ensemble learnng：集成学习通过构建多个学习器来完成学习任务；一般是通过算法产生一组个体学习器，再用某种策略将它们结合起来；通过对多个学习器的结合，可以获得比单一学习器更好的泛化性能。**

**根据个体学习器的生成方式，目前的集成学习方法大致可以分为两大类，即个体学习器间存在强依赖关系、必须串行生成的序列化方法（bagging），以及个体学习器间不存在强依赖关系、可同时生成的并行化方法（boosting）。**

**集成学习核心思想：如何训练多个弱分类器以及如何将这些弱分类器进行组合。**

**多个分类器的生成：可以采用随机选取数据进行分类器的训练，也可以采用不断调整错误分类的训练数据的权重生成新的分类器。**

**多个弱分类器怎么组合：基本分类器之间的整合方式，一般有简单多数投票，权重投票、贝叶斯投票**

**第二个问题：bagging/boosting和stacking/blending之间区别？**

**答案：**

**相同点：他们都是集成学习的方法，都是通过组合的方式来减少模型的过拟合。**

**不同点：**

1. **Bagging/boosting强调抽取数据的策略。两者都采取随机有放回取样的方式抽取数据，不同的是bagging中，所有数据被抽到的概率相同；在boosting中，每一轮被标错的数据会被增加权重，从而增加在下一轮学习中被抽取的概率。**
2. **Blending/stacking强调集成弱学习器输出的策略。在stacking中，所有弱学习器都被称为第一级学习器，他们的输出结果作为第二级学习器的输入，将第二级学习器的输出最为的预测结果。Stacking的概念，主要是针对voting/weighting/averaging等较为简单的集成策略，在一些较为复杂的问题上表现可能不太好。**

**第三个问题：bagging降低方差，boosting降低偏差？**

**答案：**

**偏差：模型预测的期望与真实值之间的差异程度**

**方差：模型预测的离散程度**

**Bagging降低方差：bagging是对样本重采样，对每一个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，因此各个模型具有近似相等的偏差(bias)和方差(variance)(事实上，各模型的分布也近似相同，也不独立)。若两个模型之间完全独立，则模型组合之后的方差是单个模型方差的1/n，若各个模型之间完全相同，则组合模型的方差等于单个模型的方差。所以会降低方差。**

**Bagging主要在于尽量使得模型独立。模型可以看做n个独立同分布随机样本的函数，因为bagging是从m个样本中抽样，不同模型的训练子集是有重合、交叠的，会导致两个模型是两组重叠的随机变量集合的函数，从而不独立，所以方差会降低。考虑极端情况，总共有m个样本，第一次抽样取得所有的m个样本，第二次抽样也是取得m个样本，则两次样本完全一样，模型也是几乎一摸一样，所以方差几乎没有改变。所以随机过程分为两个：第一个是训练集中的抽样具有随机性，即一定独立性；第二个是对抽样子集进行训练，也具有随机性（即使用两个同样的训练集去训练模型的时候也会有差别，主要原因是模型的随机性，例如初始化权重的随机性，即随机数种子）**

**Boosting降低偏差：boosting是迭代算法，每一次迭代都是根据上一次迭代预测结果对样本进行加权，优化的是模型的误差，所以随着迭代进行，误差会越来越小，所以模型的偏差会越来越低。**

**第四个问题：bagging、boosting之间的区别？**

**答案：**

**样本选择：**

1. **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。（事实上由于样本有重合，并不是独立的）**
2. **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。**

**样例权重：**

1. **Bagging：使用均匀抽样，每个样例的权重相同**
2. **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。**

**预测函数：**

1. **Bagging：所有预测函数的权重相等。**
2. **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。**

**并行计算：**

1. **Bagging：各个预测函数可以并行生成。**
2. **Boosting：每次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。**

**Bagging + 决策树：随机森林**

**Adaboost + 决策树：提升树**

**Gradient boosting + 决策树：GBDT**

**决策树：**

**第一个问题：决策树各个流程，以及优缺点？**

**答案：**

**介绍：**

**决策树 是表示基于特征对实例进行分类的树形结构。  从给定的训练数据集中，依据特征选择的准则，递归的选择最优划分特征，并根据此特征将训练数据进行分割，使得各子数据集有一个最好的分类的过程。**

**模型：决策树本质上是要从训练数据中学习一组分类规则，与训练数据不相矛盾的决策时可能有很多个，也可能没有，我们需要的是一个与训练数据矛盾较小的决策树，同时对未知数据具有较好的泛化能力。**

**策略：决策树学习的损失函数常常是正则化的极大似然函数，决策树学习的策略是以正则化的极大似然函数为目标函数的最小化。**

**算法：从所有可能决策时中选出最优是一个NP难问题，使用启发式的方法进行选择。**

**决策树学习的算法是一个递归地选择最优特征，根据该特征对训练数据进行分割。首先，开始构建根节点，所有训练数据放在根节点，选择一个最优特征，按照这个特征对样本进行最优分类，如果子集不能正确的被分类，则继续选择最优特征，继续构建内部节点，直到所有样本都被分配到叶节点上，即生成了决策树。**

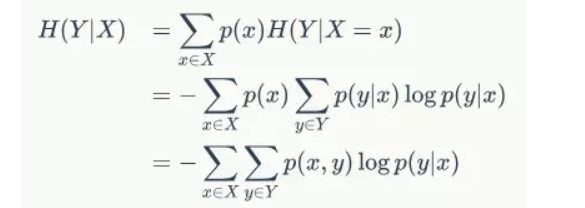
**决策树三要素：**

1. **特征选择：使用某特征对数据集划分之后，各数据子集的纯度要比划分前的数据集D的纯度高**
2. **决策树生成（模型的局部选择）**
3. **决策树的剪枝（全局选择）。**

**特征选择三种方式：信息增益（ID3），信息增益比(C4.5)，基尼指数**

**熵：表示随机变量不确定性的度量。**

**条件熵：H(Y|X)表示在已知随机变量X的条件下随机变量Y的不确定性。**



**信息增益I(Y,X) = H(Y) – H(Y|X) 表示得知特征X的信息而使得类Y的信息的不确定性减少的程度。**

**信息增益比g(D,A) = I(Y,X)/H(X)**

**特征选择（信息增益ID3）：**

1. **信息增益：给定训练集和特征，H(Y)表示对数据集进行分类的不确定性，H(Y|X)表示在特征X给定的条件下对训练集进行分类的不确定性，两者之差表示由于特征X而使得训练集进行分类不确定性的减少程度。不同的特征往往具有不同的信息增益，信息增益大的特征具有更强的分类能力。**
2. **理解：对于待划分的数据集D，其 entroy(前)是一定的，但是划分之后的熵 entroy(后)是不定的，entroy(后)越小说明使用此特征划分得到的子集的不确定性越小（也就是纯度越高），因此 entroy(前) -  entroy(后)差异越大，说明使用当前特征划分数据集D的话，其纯度上升的更快。而我们在构建最优的决策树的时候总希望能更快速到达纯度更高的集合，这一点可以参考优化算法中的梯度下降算法，每一步沿着负梯度方法最小化损失函数的原因就是负梯度方向是函数值减小最快的方向。同理：在决策树构建的过程中我们总是希望集合往最快到达纯度更高的子集合方向发展，因此我们总是选择使得信息增益最大的特征来划分当前数据集D。**
3. **缺点：信息增益在选择特征时偏向取值较多的特征，往往只利用了少数的特征，训练出来的树是一颗庞大而深度很浅的树，但是未利用其它实际上可能更加有效的分类信息，泛化能力会变差。**
4. **原因：当特征的取值较多时，根据此特征划分更容易得到纯度更高的子集，因此划分之后的熵更低，由于划分前的熵是一定的，因此信息增益更大，因此信息增益比较偏向取值较多的特征。**

**特征选择(信息增益比C4.5)：**

1. **信息增益比：信息增益比 = 惩罚参数\*信息增益。对特征的分布程度进行惩罚，当特征取值越少，特征的熵越小，惩罚参数越大，可以对信息增益偏向选择于选择取值更多的问题进行校正。**
2. **惩罚参数：数据集D以特征A作为随机变量熵的倒数，用来衡量属性分裂数据的广度和均匀性，改进了选择值多属性的缺点。**

**特征选择(基尼指数 CART)：CART是在给定输入随机变量X条件下输出随机变量Y的条件概率分布的学习方法。CART假设决策树是二叉树，内部结点特征的取值为“是”和“否”，即等价地递归分类每个特征。**

**CART策略：对于分类采用基尼指数最小化准则，对于回归采用平方误差最小化准则。**

**最小二乘回归树：在训练集的假设空间中，递归的每个子区域划分为两个子区域并决定每个子区域的输出值，构建二叉决策树。**

**最小基尼指数分类树：在训练集的假设空间中，递归的选择最优特征的最优二值切分点将子区域华为为两个子区域，区域输出值为区域内样本分类结果，构建二叉决策树。**

**基尼指数：表示训练集经过特征A后的不确定性，值越大，不确定性就越大**

**基尼指数计算：经过特征A后样本被分为D1 和D2**

**Gini(D,A) = (|D1|\*Gini(D1) + |D2|\*Gini(D2))/|D|**

**Gini(D): 1 - D中各种分类的样本比例平方和**

**第二个问题：决策树三种剪枝方式？**

**答案：**

**剪枝目的：决策树生成算法递归地产生决策树，直到不能继续下去为止，导致决策树对已知数据预测很准确，对未知测试数据预测分类并不准确，即过拟合现象。过拟合原因在学习时过多的考虑如何提高对训练数据的正确分类，从而构建出复杂的决策树。**

**剪枝：决策树算法在学习过程中为了正确的分类训练样本，不停的对样本进行划分，会导致整棵树的分支过多，容易过拟合，剪枝策略最基本有两种：预剪枝和后剪枝。**

**预剪枝：预剪枝是在构造决策树的过程中，先对每个节点在划分前进行估计，如果当前节点的划分不能带来模型泛化能力的提升，则不对当前节点进行划分并将当前节点标记为叶节点。**

**后剪枝：后剪枝就是先把整颗决策树构造完毕，然后自底向上的对非叶节点进行考察，若该非叶节点对应的子树替换为叶节点能够带来泛化性能的提升，则将该非叶节点对应的子树替换为叶节点。**

**Reduce error pruning(错误率降低剪枝)：使用测试集对决策树进行纠正，对决策树中每一个非叶节点的子树，将子树替换成叶节点，该叶节点类别为子树所覆盖的训练集中类别最多的类代替，比较两个决策树在测试集上的表现。若简化的决策树在测试集中的错误率少，并且该子树里面没有包含另外一个具有类似特性的子树（即把子树中的子树替换成叶节点后，测试集错误率下降），那么将该子树替换成叶节点，算法从下向上遍历所有子树。**

**Rep方法动机：即使学习器可能会因为训练集中的随机错误和巧合所误导，但验证集中不太可能出现同样的错误，所以验证集可以用来剪枝、修正。**

**Pessimistic Error pruning(悲观错误剪枝)：使用伯努利分布对错误次数进行建模，认为通过rep方法校正后子树误判个数大于原先误判个数一个标准差菜不进行校正，缺点是可能过度剪枝。**

**基于代价函数的剪枝：通过极小化决策树整体的损失函数或代价函数。对于每个叶子节点来说，该叶节点覆盖的样本存在多个分类，可以计算得到该叶节点的熵，对每个叶子节点来说，熵乘以叶子节点中样本个数得到该叶子节点的损失函数，将所有叶子节点的损失函数加起来得到所有叶子节点的损失函数，即决策树的极大似然估计，加上正则项，正则项为叶子节点的个数。具体方法：递归的从树的叶节点向上回缩，回缩之前的损失函数和回缩之后的损失函数，若回缩后损失函数小，则应该剪枝。**

**第三个问题：ID3、C4.5、CART三种算法的优缺点对比？**

**答案：**

**ID3算法：**

1. **没有考虑连续特征，比如长度、密度都是连续值，无法在ID3中使用。**
2. **在同等分布的情况下，ID3偏向于选取特征取值较多的特征。**
3. **ID3对特征缺失值没有考虑。**
4. **没有考虑过拟合的问题**

**针对ID3出现的问题，C4.5进行了改进。改进如下：**

**C4.5:除了上面四个问题，C4.5相对于ID3思路区别不大。**

1. **针对不能处理连续特征，C4.5将连续特征离散化，假设有m个样本取值有m中，C4.5将特征值进行排序，取两个连续特征值的平均值作为切分点，使用信息增益最大的点作为该连续特征的二元离散切分点，连续特征在后续还可以继续参与子节点的产生选择过程。**
2. **针对信息增益偏向于取值较多的特征，C4.5对特征分布的广度和均匀性进行惩罚，可以校正这个问题。**
3. **缺失值处理问题，主要需要解决两个问题：第一个是样本某些特征缺失的情况下选择划分的属性（确定内部节点），第二个是在选择了划分属性（内部节点）后，对于在该属性上缺失特征的样本的处理。首先对于第一个问题：对于含有缺失值的特征A，C4.5将特征A数据分成两部分，对每个样本设置一个权重，然后划分数据，一部分是有特征值A的数据D1，另一部分是没有特征A的数据D2。然后对于数据集D1来计算特征A的各个特征值加权重后的信息增益比，最后乘以一个系数，这个系数是无特征A缺失样本加权后所占加权样本的比例。对于第二个子问题，确定节点后，将特征A缺失的样本同时划分到所有的子节点中，但是缺失样本在每个子节点的权重会改变，一般权重按照子节点中特征A无缺失的样本数量比例改变权重，即子节点中特征A无缺失样本的个数越多，权重越大。**
4. **对于过拟合问题，C4.5引入了正则化系数进行初步的剪枝。**

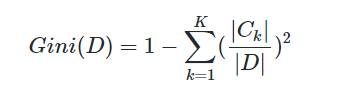
**C4.5虽然相对于ID3改进了很多，但是还有优化的空间（即缺点）：**

1. **针对过拟合问题，C4.5还可以优化。思路主要有两种，一种是预剪枝，即在生成决策树评估节点是否进行划分或剪枝；另一种是后剪枝，即先生成决策树，再通过交叉验证来剪枝，CART 会详细讲。**
2. **C4.5使用的是多叉树，即一个父节点可能有多个子节点，在计算机中，二叉树运算效率更高。如果采用二叉树，我可以提高效率。**
3. **C4.5由于采用熵运算，对数运算特别耗时，如果是连续值还有大量的排序运算。如何对模型进行简化，来达到减少运算强度并不牺牲太多准确性？**
4. **C4.5只能用于分类，而不能用于回归。**

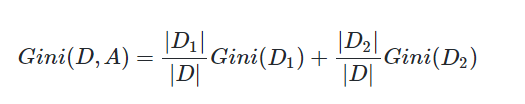
**CART针对以上四个问题加以改进，scikit-learn决策树即CART算法。由于CART既可以做回归，又可以做分类，分队对其进行介绍。**

**CART分类树：**

1. **特征选择：无论在ID3 还是 C4.5中，特征选择都是基于信息论的熵模型，涉及到大量的对数运算；CART分类树使用基尼系数代替信息增益比，基尼系数代表了模型的不纯度，基尼系数越小，则不纯度越低，特征越好，和信息增益比相反。**
   1. **给定样本D，假设有k个类别，第k个类别的数量为||,则样本D的系数表达式为：**



**对于样本D，如果根据特征A的某个值a，把D分成两部分D1、D2，则在特征A的条件下，D的基尼系数表达式为：**



**相对于熵运算，基尼指数运算简洁很多，而且基尼系数和熵之半的曲线非常接近**

* 1. **对于连续特征和离散特征的改进：CART对连续特征的处理和C4.5思想是一样的，区别在于选择划分点时的度量方式不同。如果是连续特征，则选择一次划分点之后还可以参与子节点的产生选择过程。**

**对于CART分类是离散值的处理问题，采用不停的二分离散特征（不同的是，而在ID3和C4.5中，离散特征只会参与一次节点的建立）。**

**CART回归树：与CART分类树的处理方法不同，决策树建立后做预测的方式不同**

1. **对于回归模型，CART回归树使用了常见的和方差度量方式，CART回归树的度量目标是，对于任意划分特征A，对应的任意划分点S两边划分的数据集D1和D2，求出使D1和D2各自集合的均方差最小，同时D1和D2的均方差之和最小所对应的特征和特征值划分。**
2. **对于决策树建立后的预测方式，CART分类树采用叶子节点里概率最大的类别作为当前节点的预测类别，而回归树输出不是类别，采用的是最终叶子节点覆盖样本的均值或中位数来预测输出结果。**
3. **CART剪枝：分类树和回归树在度量损失时一个使用均方差，一个使用基尼指数。**

**CART剪枝：CART剪枝可以分为两步，第一步是从原始决策树生成各种剪枝效果的决策树，第二部是用交叉验证的方式来检验剪枝后的预测能力，选择泛化预测能力最好的剪枝后的数作为最终的CART树。**

**第四个问题：决策树的并行策略？**

**第五个问题：判断决策树及随机森林是过拟合了还是欠拟合？**

**随机森林：**

**第一个问题：RF的随机性体现在哪里？它的代码中输出的特征重要程度是怎么进行计算？**

**第二个问题：Sklearn中树模型输出的特征重要程度是本身的还是百分比？**

**第三个问题：ID3+C4.5+优缺点+树的融合（GBDT，RF）+我的实现**

**第四个问题：提升树的思想，随机森林和提升树的区别。**

**第五个问题：结合偏差和方差说明gbdt和rf的区别，然后又围绕展开很多其他问题**

**第六个问题：做过广告点击率预估没？LR+GBDT和GBDT+FM怎么结合的知道不？**

**第七个问题：了解FM吗？GBDT的数据在使用前有什么需要注意的吗？**

**第八个问题：RF与xgboost的区别？怎样选取的特征？如何判断这些特征的重要程度？最后RF的层数和深度是多少？**

**第九个问题：介绍xgboost、gbdt、rf的区别**

**第十个问题：RF和GBDT的区别？二者的优化目标是什么？**

**聚类算法：**

**第一个问题：手写Kmeans**

**第二个问题：k-d树的原理**

**第三个问题：哈夫曼树在机器学习中的应用**

**第四个问题：Kmeans和KNN**

**第五个问题：关联规则中，置信度和支持度的概念？**

**第六个问题：K-means聚类个数选择，做什么样的试验来确定K**

**第七个问题：Kmeans中，现在给你n个样本点不在欧式空间中，无法度量距离。现在给了一个函数F，可以衡量任意两个样本点的相似度。请问Kmeans如何操作？**

想了一会，比如K=4的聚类。

1、首先，随机去4个点，作为初始类簇中心。

2、计算所有样本点与这4个点的F相似度。根据相似程度，把所有样本点分到4个类中。

3、在这4个类中，计算每一个样本点 i 到该类其他样本点的相似度和Si。取Si最大的那个点作为这个类的中心。

4、重复2、3步骤，直到类中心不再变化或者循环次数达到目标。

**EM算法：**

**第一个问题：EM算法数学原理**

**第二个问题：EM算法数学推导，如何证明算法收敛性？**

**第三个问题： Kmeans+EM**

**第四个问题：GMM原理，增大数据量是否会更好？**