**机器学习算法面试总结**

1、监督学习与非监督学习

（1）监督学习有训练集与测试样本，在训练中找规律，有目标值和特征值；

（2）非监督学习没有训练集，只有一组数据，在组内寻找规律；

2、分类与聚类

（1）聚类分析是一种分类的多元统计分析方法。按照个体或样品的特征将它们分类，使同一类别内的个体具有尽可能高的同质性，而不同类别之间则应具有尽可能高的异质性。

（2）聚类分析在没有训练集的条件下把样本划分若干类，自动标记确定

（3）分类分析类是确定的要做的是将每条记录标记出来

**无监督学习方法**

3、最近邻（KNN）算法，用距离衡量样本之间的相似度

（1）原理：

* 计算训练样本和测试样本中每个样本点的距离（距离度量方式：欧式距离、夹角余弦等）；
* 对上面所有的距离值进行排序；
* 选前k个最小距离的样本；
* 根据这k个样本的标签进行投票，得到最后的分类类别；

（2）优缺点：

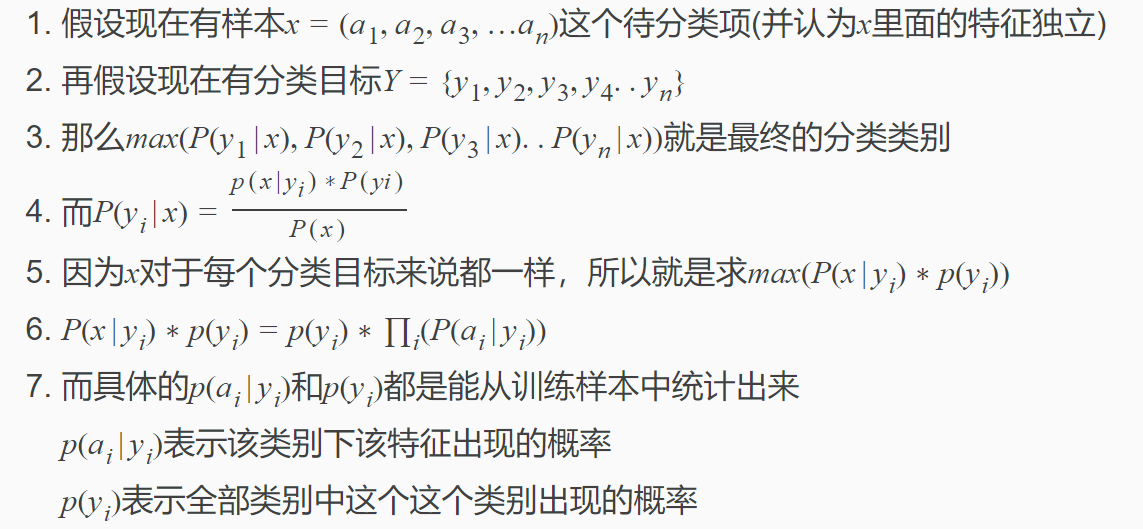
* 优点：思想简单，理论成熟，既可以用来做分类也可以用来做回归；可用于非线性分类；准确度高，对数据没有假设，对outlier不敏感；
* 缺点：计算量大；样本不平衡问题；需要大量的内存；

（3）python 代码

* from sklearn.neighbors import KNeighborsClassifie
* model=KNeighborsClassifier(n\_neighbors=6)
* model.fit(X, y)
* predicted= model.predict(x\_test)

4、朴素贝叶斯算法，以样本可能属于某类的概率来作为分类依据

（1）原理：



（2）优缺点：

* 对小规模的数据表现很好，适合多分类任务，适合增量式训练；
* 对输入数据的表达形式很敏感（离散、连续，极大极小值）；

（3）注意点：

* 当某个类别下某个特征划分没有出现时，引入Laplace校验，计数加1；
* 特征不独立时，使用贝叶斯网络；

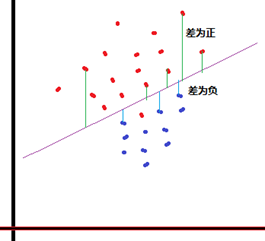
（4）python 代码

* from sklearn.naive\_bayes import GaussianNB
* model = GaussianNB()
* model.fit(X, y)
* predicted= model.predict(x\_test)

5、logistic逻辑回归，通过大量的数据拟合出一条曲线的表达式，以这条曲线区分测试样本实现分类。

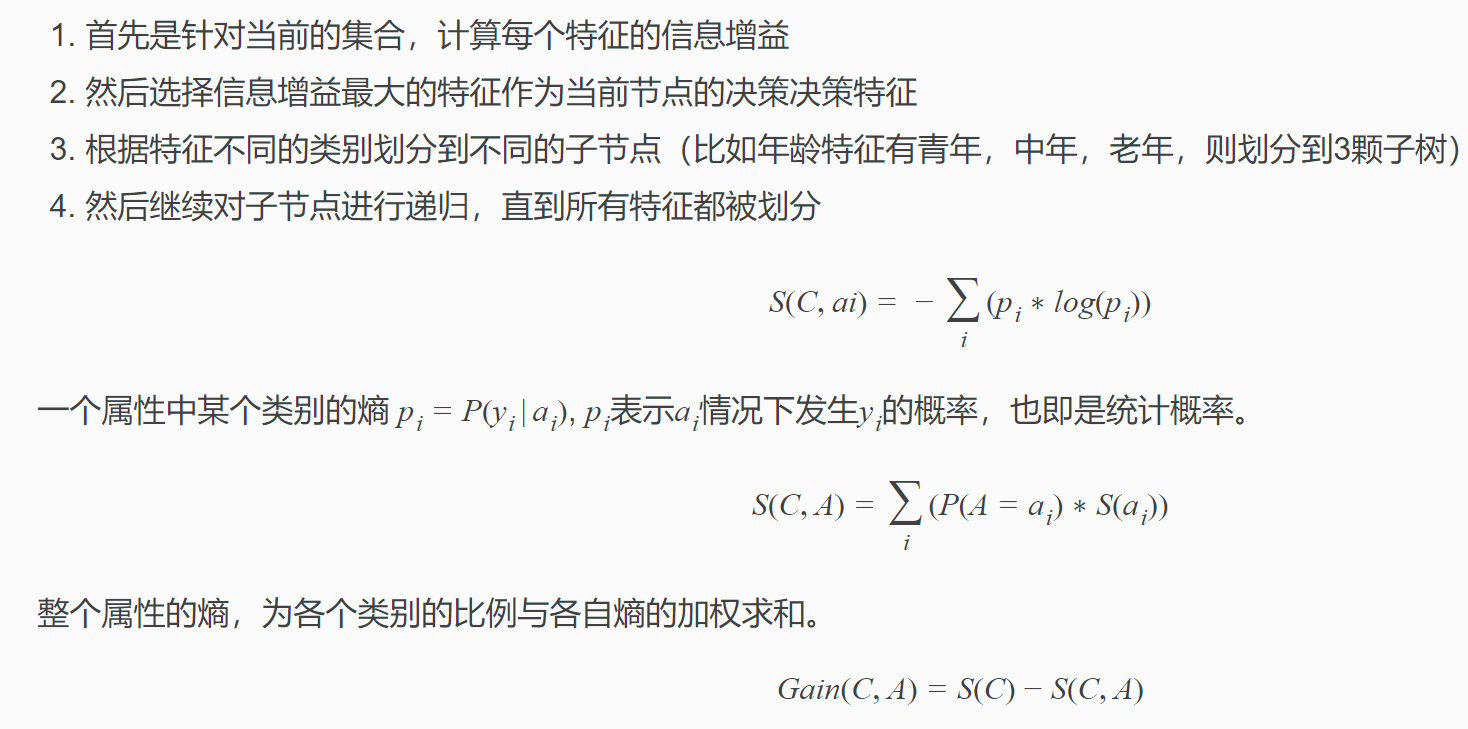
（1）训练：不断用样本特征值代入算式，计算出结果后跟其实际标签进行比较，根据差值来修正参数，然后再代入新的样本值计算，循环往复，直到无需修正或已到达预设的迭代次数。

（2）测试：计算各点的y值到拟合线的垂直距离，如果距离>0，分为类A；距离<0，分为类B。

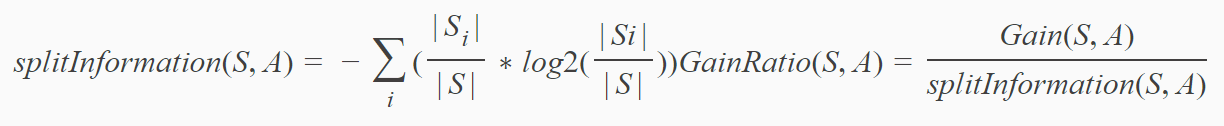


6、决策树算法，经决策属性划分后，数据无序度越来越低，即信息熵越来越小

（1）ID3原理：增益表示分类目标的熵减去当前属性的熵，增益越大，分类能力越强



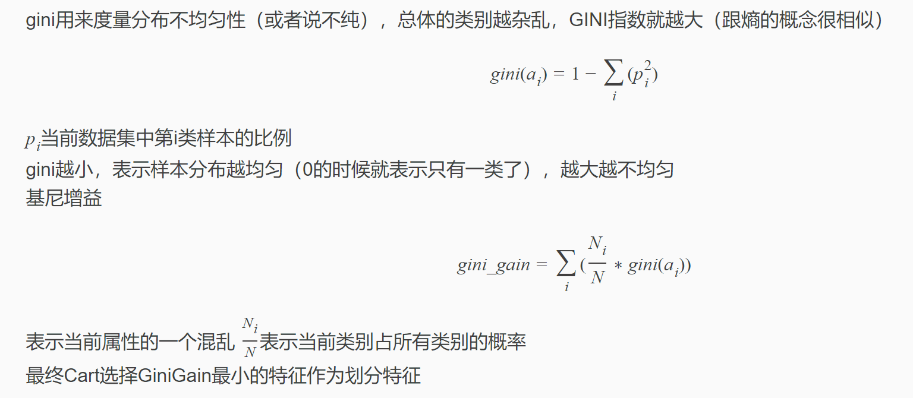
（2）C4.5：ID3的改进算法，使用信息增益率来进行属性的选择



（3）Cart：

* 首先是根据当前特征计算基尼增益
* 选择基尼增益最小的特征作为划分特征
* 从该特征中查找基尼指数最小的分类类别作为最优划分点
* 将当前样本划分成两类，一类是划分特征的类别等于最优划分点，另一类就是不等于
* 针对这两类递归进行上述的划分工作，直达所有叶子指向同一样本目标或者叶子个数小于一定的阈值

（4）Gini系数：



（5）停止条件：

* 每个叶子节点都只有一种类型的记录时停止，易过拟合
* 当叶子节点的记录树或节点的信息增益小于一定的阈值

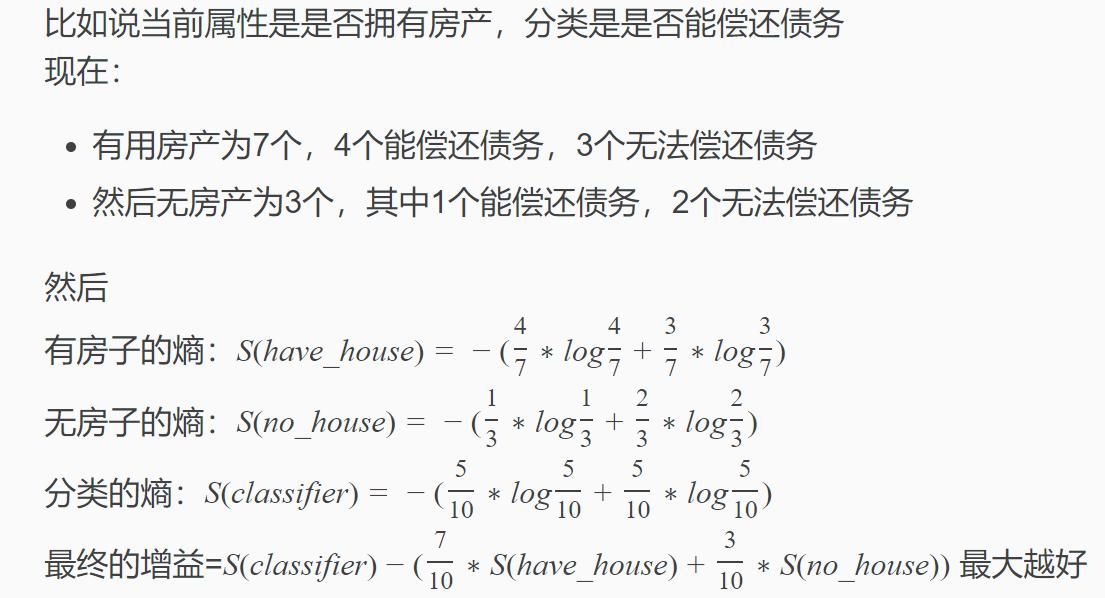
1. 解决过拟合：

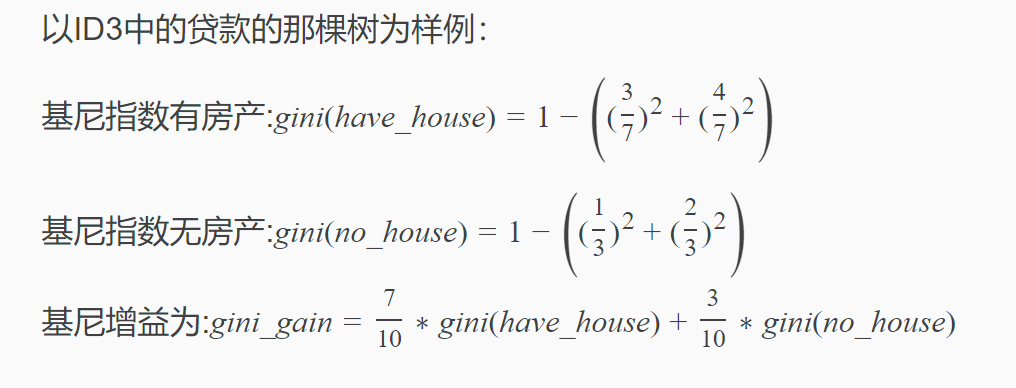
* 剪枝：前置剪枝：在分裂节点的时候设计比较苛刻的条件，如不满足则直接停止分裂（这样干决策树无法到最优，也无法得到比较好的效果）；后置剪枝：在树建立完之后，用单个节点代替子树，节点的分类采用子树中主要的分类（这种方法比较浪费前面的建立过程）；交叉验证；随机森林

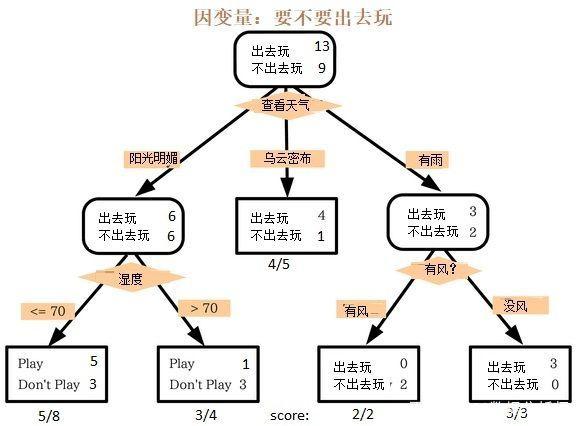
（7）优缺点：

* 计算量简单，可解释性强，比较适合处理有缺失属性值的样本，能够处理不相关的特征；
* 单颗决策树分类能力弱，并且对连续值变量难以处理；容易过拟合（后续出现了随机森林，减小了过拟合现象）

（8）实例：







（9）python 代码：

* 导入库：from sklearn import tree
* 创建树对象：model = tree.DecisionTreeClassifier(criterion='gini')
* 训练模：model.fit(X, y)
* 评估：model.score(X, y)
* 预测输出predicted= model.predict(x\_test)

7、随机森林算法

（1）原理：由很多随机决策树构成，它们之间没有关联。得到RF以后，在预测时分别对每一个决策树进行判断，最后使用Bagging的思想进行结果的输出（也就是投票的思想）

* 现有N个训练样本，每个样本的特征为M个，需要建K颗树
* 从N个训练样本中有放回（Bagging算法）的取N个样本作为一组训练集（其余未取到的样本作为预测分类，评估其误差）
* 从M个特征中取m个特征左右子集特征(m<<M)
* 对采样的数据使用完全分裂的方式来建立决策树，这样的决策树每个节点要么无法分裂，要么所有的样本都指向同一个分类
* 重复2的过程K次，即可建立森林

（2）参数：

* 树的个数 100~10000
* 叶子的深度 3~8
* 学习速率 0.01~1
* 叶子上最大节点树 20
* 训练采样比例 0.5~1
* 训练特征采样比例 sqrt(num)

（3）优缺点：

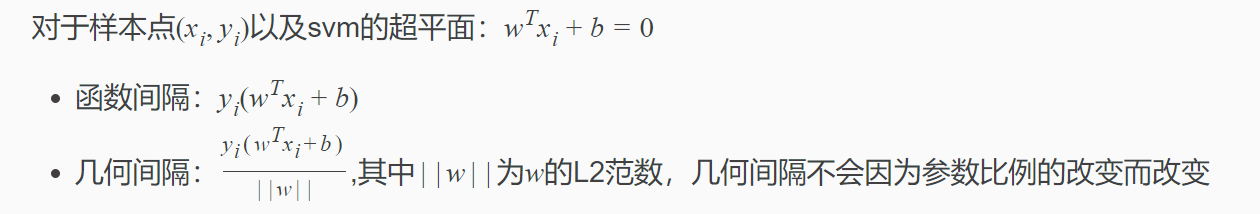
* 能够处理大量特征的分类，不用做特征选择
* 在训练完成之后能给出哪些feature的比较重要
* 训练速度很快，容易并行

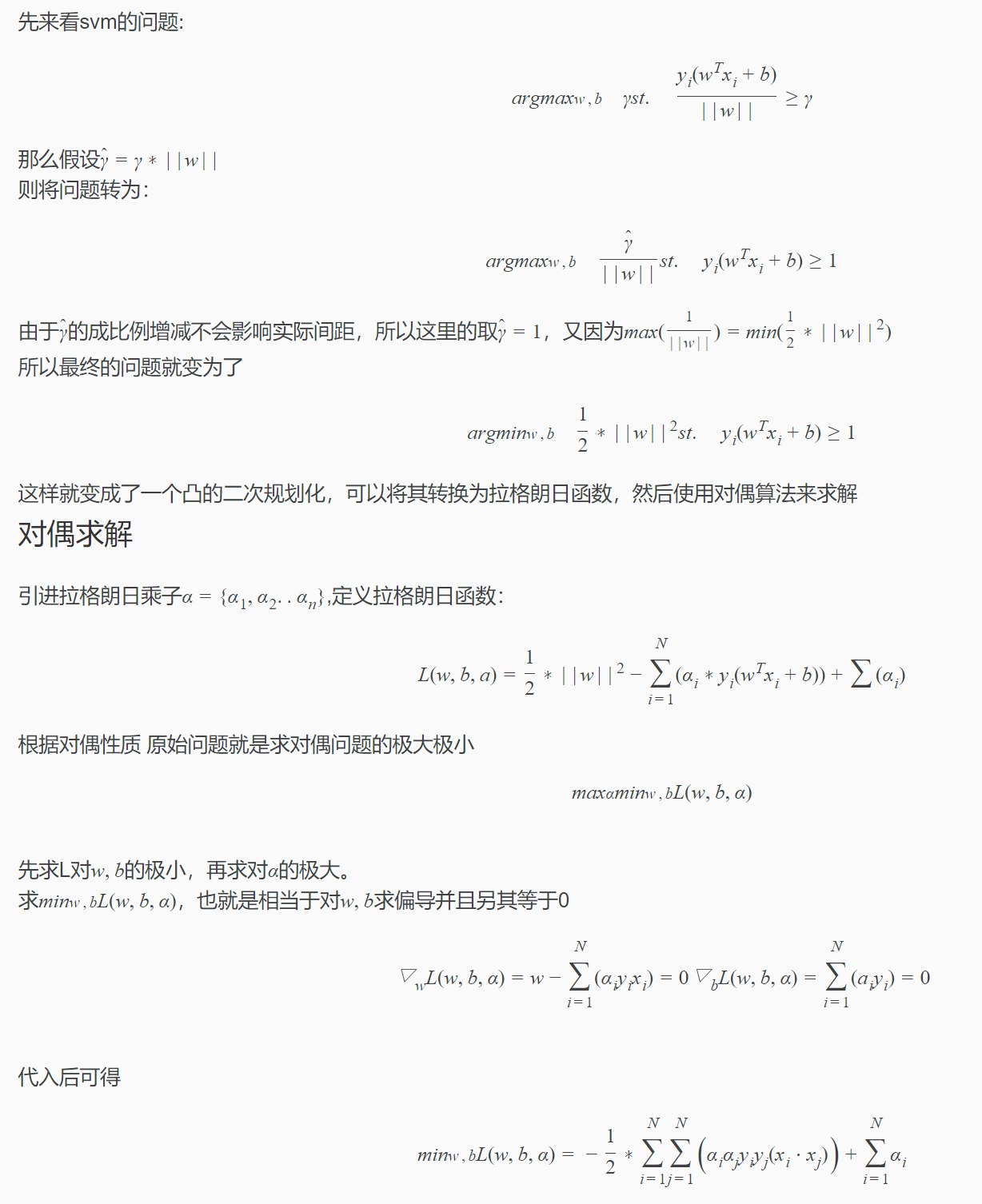
（4）python代码：

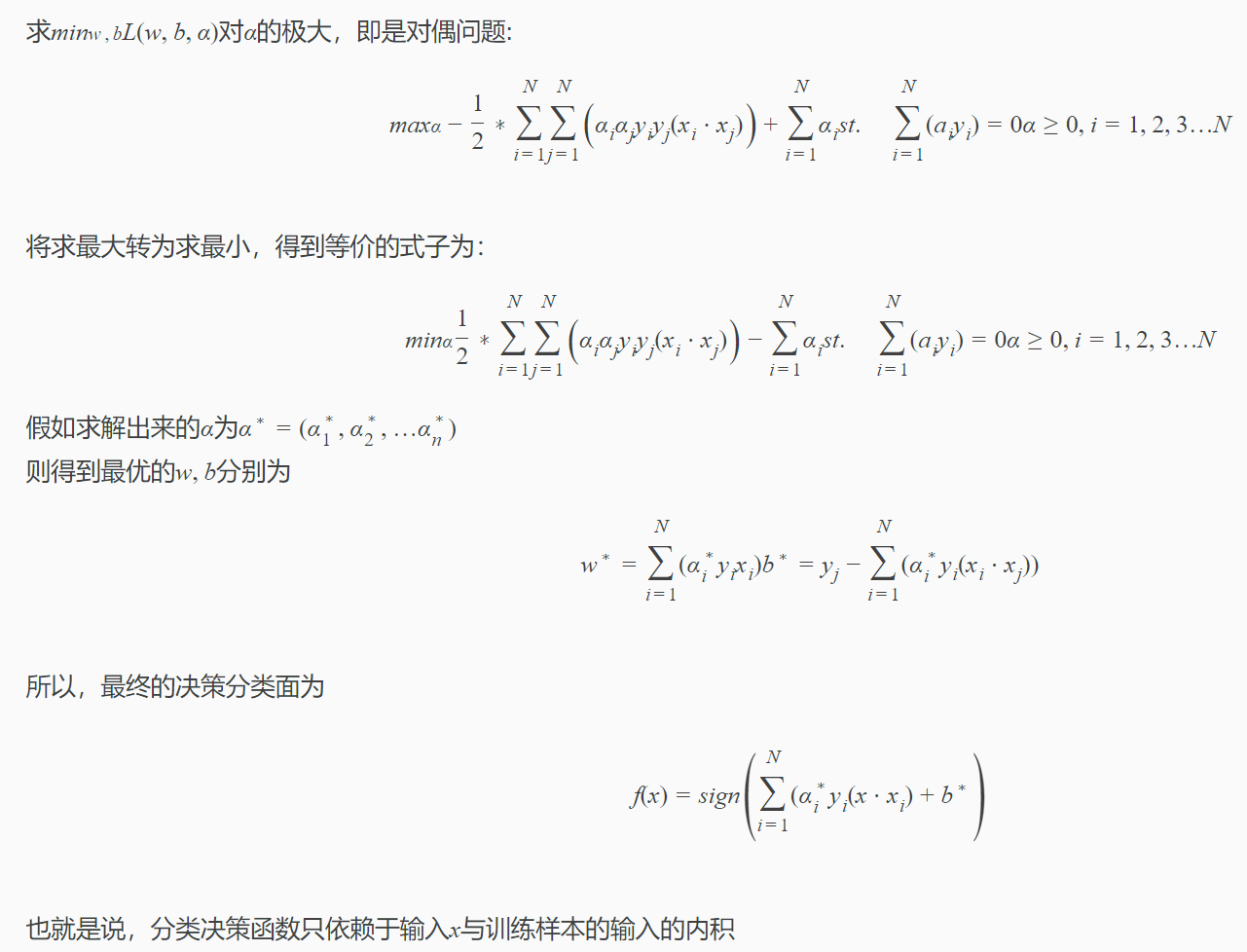
* from sklearn.ensemble import RandomForestClassifier
* model= RandomForestClassifier()
* model.fit(X, y)
* predicted= model.predict(x\_test)

8、支持向量机算法

（1）原理：求解能正确划分训练样本并且其几何间隔最大化的超平面。







（2）多分类问题：

* 直接法：直接在目标函数上进行修改，将多个分类面的参数求解合并到一个最优化问题中，通过求解该优化就可以实现多分类（计算复杂度很高，实现较为困难）
* 间接法：一对多，其中某个类为一类，其余n-1个类为另一个类，会存在偏置，很不实用)；一对一(libsvm实现的方式)，任意两个类都训练一个分类器，那么n个类就需要n\*(n-1)/2个svm分类器。

（3）优缺点：

* 优点：使用核函数可以向高维空间进行映射，解决非线性的分类；分类思想很简单，就是将样本与决策面的间隔最大化；分类效果较好；
* 缺点：对大规模数据训练比较困难；无法直接支持多分类；

（4）python 代码：

* Import Libraryfrom sklearn import svm
* model = svm.svc(kernel='rbf', C=1000, gamma=0.5)
* model.fit(X, y)
* model.score(X, y)
* predicted= model.predict(x\_test)

**无监督学习方法**

9、k-means算法

（1）原理：

* 随机选择k个样本作为初始聚类中心
* 对剩下的样本，计算其到各个聚类中心的距离，根据与聚类中心的距离，将其归入最近族；
* 计算每个族内所有样本的均值，作为新的聚类中心
* 重复2，3，直到质心不在发生变化

（2）优缺点：

* 优点：算法简单、快速，局部收敛，找出使平方误差函数值最小的k个划分。
* 缺点：只有在簇的平均值被定义的情况下才能使用，且对有些分类属性的数据不适合；事先给k；对初值敏感，对于“噪声”和孤立点数据敏感

（3）python 代码:

* from sklearn.cluster import KMeans
* model= KMeans(n\_clusters=3, random\_state=0)
* model.fit(X)
* predicted= model.predict(x\_test)

**深度学习**

10、卷积神经网络

（1）卷积层：

* 局部感知：人的大脑识别图片的过程中，并不是一下子整张图同时识别，而是对于图片中的每一个特征首先局部感知，然后更高层次对局部进行综合操作，从而得到全局信息。举个例子来讲，一个32×32×3的RGB图经过一层5×5×3的卷积后变成了一个28×28×1的特征图，那么输入层共有32×32×3=3072个神经元，第一层隐层会有28×28=784个神经元
* 权值共享机制：对于每一层来讲，所有神经元对应的权值应该是相等的，也就是说，第一个神经元的参数向量为[w1,w2,…,w100]，那么其他同层的神经元也是[w1,w2,…,w100]，这就是权值共享。同一层下的神经元的连接参数只与特征提取的有关，而与具体的位置无关，因此可以保证同一层中所有位置的连接是权值共享的。

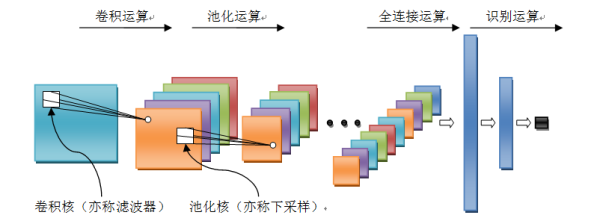
（2）池化层：也称为欠采样或下采样。主要用于特征降维，压缩数据和参数的数量，减小过拟合，同时提高模型的容错性。主要有：

* Max Pooling：最大池化
* Average Pooling：平均池化

（3）全连接层：将最后一个卷积层的输出压平，并把当前层的每个节点和下一层的节点相连。

（4）激活函数：首先ReLU，因为迭代速度快，但是有可能效果不加。如果ReLU失效的情况下，考虑使用Leaky ReLU或者Maxout

* Tanh函数在文本和音频处理有比较好的效果。
* Sigmoid函数
* Tanh函数
* ReLU
* Leaky ReLU
* ELU
* Maxout



11、LSTM网络

（1）原理：使用遗忘门、输入门和输出门来判断输入的脑电信息是否有用，对于前一时刻的脑电信息和现在时刻有用的脑电信息将被留下和输出，需要丢弃的脑电信息通过遗忘门被遗忘。LSTM结构处理EEG信号特征有如下几个步骤：

* 遗忘门：控制是否被遗忘，决定丢弃细胞状态的哪些信息



其中，*ht-1*是上一序列的脑电信号输出，*xt*是当前细胞的脑电信号输入，激活函数一般是sigmoid函数， *ft*是遗忘门的输出。

* 输入门：控制当前脑电序列的位置，决定哪些新的脑电信息可以新增到细胞状态中



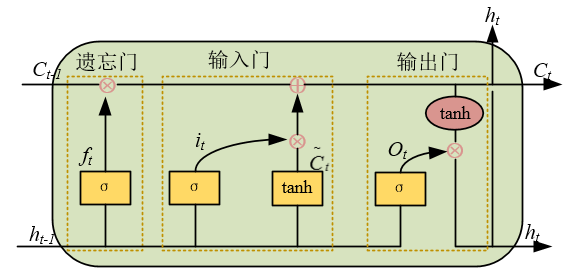
其中，it是sigmoid层的输出，是tanh层的输出。

* 细胞状态Ct的更新公式如下：



* 输出门：确定最终输出的脑电信号内容，更新输出值





（2）梯度消失：当神经网络有很多层，每个隐藏层都使用Sigmoid函数作为激励函数时，当神经网络特别深的时候，梯度呈指数级衰减，导数在每一层至少会被压缩为原来的1/4，当z值绝对值特别大时，导数趋于0。使得权重和偏差参数无法被更新，导致神经网络无法被优化，训练永远不会收敛到良好的解决方案

（3）梯度爆炸：将w初始化为一个较大的值时，当神经网络很深时，梯度呈指数级增长，最后到输入时，梯度将会非常大，会得到一个非常大的权重更新。

（4）解决方案：

* 非饱和的激活函数（如 ReLU）
* 批量规范化（Batch Normalization）
* 梯度截断（Gradient Clipping）

12、关联规则：是隐藏在数据项之间的关联或相互关系，即可以根据一个数据项的出现推导出其他数据项的出现。

* 第一阶段为从海量原始数据中找出所有的高频项目组;
* 第二阶段为从这些高频项目组产生关联规则。

13、数据预处理：包括数据清洗、数据集成、数据变换、数据归约。

* 数据清洗：选择子集、列名重命名、删除重复值、缺失值处理、一致化处理、数据排序、异常值处理

14、大数据数据处理与分析

（1）分布式计算模式和框架

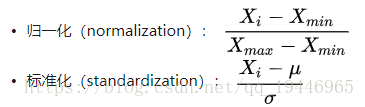
* 批处理：Hadoop、spark
* 流处理： Storm、Flink
* 图计算：Pregel、Graph X

（2）大数据分析

* 交互式查询：Hive、Pig、Spark、Sql
* 数据挖掘：Mahout
* 机器学习：Mllib

15、什么是数据标准化，为什么要进行数据标准化？

（1）定义：



（2）为什么要做数据归一化？

数据归一化后，损失函数变量前面的系数差距已不大，图像的等高面近似圆形，在梯度下降进行求解时能较快的收敛。一些机器学习算法需要计算样本之间的距离（如欧氏距离），例如 KNN、K-means 等。如果一个特征值域范围非常大，那么距离计算就主要取决于这个特征

（3）为什么要进行数据标准化？

数据标准化是预处理步骤，将数据标准化到一个特定的范围能够在反向传播中保证更好的收敛。如果不进行数据标准化，有些特征（值很大）将会对损失函数影响更大（就算这个特别大的特征只是改变了1%，但是他对损失函数的影响还是很大，并会使得其他值比较小的特征变得不重要了）。因此数据标准化可以使得每个特征的重要性更加均衡。

（4）区别与联系

* 相同点：归一化和标准化本质上都是一种线性变换
* 区别：归一化：缩放仅仅跟最大、最小值的差别有关，输出范围在0-1之间；标准化：缩放和每个点都有关系，通过方差（variance)体现出来，输出范围是负无穷到正无穷

（5）什么时候用归一化？什么时候用标准化？

如果对输出结果范围有要求，数据较为稳定，不存在极端的最大最小值，用归一化；如果数据存在异常值和较多矂音，用标准化，可以间接通过中心化避免貝常值和极端值的影响。

16、解释什么是降维，在哪里会用到降维，它的好处是什么？

降维是指通过保留一些比较重要的特征，去除一些冗余的特征，减少数据特征的维度。而特征的重要性取决于该特征能够表达多少数据集的信息，也取决于使用什么方法进行降维。而使用哪种降维方法则是通过反复的试验和每种方法在该数据集上的效果。一般情况会先使用线性的降维方法再使用非线性的降维方法，通过结果去判断哪种方法比较合适。而降维的好处是：

（1）节省存储空间；

（2）加速计算速度（比如在机器学习算法中），维度越少，计算量越少，并且能够使用那些不适合于高维度的算法；

（3）去除一些冗余的特征；

（4）将数据维度降到2维或者3维使之能可视化，便于观察和挖掘信息。

（5）特征太多或者太复杂会使得模型过拟合。

17、如何处理缺失值数据？

（1）删除含有缺失值的个案

* 整例删除(casewise deletion)：剔除含有缺失值的样本，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。

（2）可能值插补缺失值

* 估算(estimation)：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。未充分考虑数据中已有的信息，误差可能较大。
* 利用同类均值插补。同均值插补的方法都属于单值插补，不同的是，它用层次聚类模型预测缺失变量的类型，再以该类型的均值插补。
* 极大似然估计（Max Likelihood ,ML）。在缺失类型为随机缺失的条件下，假设模型对于完整的样本是正确的，那么通过观测数据的边际分布可以对未知参数进行极大似然估计（Little and Rubin）。重要前提：适用于大样本。有效样本的数量足够以保证ML估计值是渐近无偏的并服从正态分布。但是这种方法可能会陷入局部极值，收敛速度也不是很快，并且计算很复杂。
* 多重插补（Multiple Imputation，MI）。多值插补的思想来源于贝叶斯估计，认为待插补的值是随机的，它的值来自于已观测到的值。具体实践上通常是估计出待插补的值，然后再加上不同的噪声，形成多组可选插补值。根据某种选择依据，选取最合适的插补值。

18、解决维度灾难问题：主成分分析法PCA，线性判别法LDA，奇异值分解简化数据、拉普拉斯特征映射，Lassio缩减系数法(L1)、小波分析法

19、在图像处理中为什么要使用卷积神经网络而不是全连接网络？

卷积过程是考虑到图像的局部特征，能够更加准确的抽取空间特征。而全连接会提取很多不相关的信息。其次，CNN有平移不变性，因为权值共享，图像平移了，卷积核仍可以识别出来，但是全连接则做不到。

20、神经网络为什么要用sigmoid函数？为什么要映射到0-1之间？

（1）对于深度神经网络，中间的隐层的输出必须有一个激活函数，否则多个隐层的作用和没有隐层相同。这个激活函数不一定是sigmoid，常见的有sigmoid、tanh、relu等。

（2）对于二分类问题，输出层是sigmoid函数。这是因为sigmoid函数可以把实数域光滑的映射到[0,1]空间。函数值恰好可以解释为属于正类的概率（概率的取值范围是0~1）。另外，sigmoid函数单调递增，连续可导，导数形式非常简单，是一个比较合适的函数

（3）对于多分类问题，输出层就必须是softmax函数了。softmax函数是sigmoid函数的推广

21、什么是偏倚（bias）、方差（variable）均衡？

* 偏倚：模型预测值与真实值的差异，会导致模型欠拟合
* 方差：不同训练数据训练的模型的预测值之间的差异，会导致模型过拟合

模型越复杂，偏倚越低，方差越大。

22、KNN和k-means聚类有什么不同？

KNN是一种监督学习算法，而k-means 是非监督的。这两种算法都需要计算样本之间的距离，而KNN算法训练集为带标签的数据，k-means聚类只需要一些未分类的数据点和阀值，算法会逐渐将样本点进行分成族类。

23、L1、L2正则之间有什么不同？

* L2正则：加入2范数，使得对权重进行衰减，从而达到惩罚损失函数的目的，防止模型过拟合。保留显著减小损失函数方向上的权重，而对于那些对函数值影响不大的权重使其衰减接近于0。
* L1正则：加入1范数，同样可以防止过拟合。它会产生更稀疏的解，即会使得部分权重变为0，达到特征选择的效果。

24、因果分析方法：



（1）拆解法：把一个结果指标，从多个角度拆解，找到影响它的原因

（2）相关系数法：通过计算，得出两列数字或者几列数字之间的关系

（3）趋势分析法

（4）控制变量法

25、过拟合和欠拟合

欠拟合是指模型不能在训练集上获得足够低的误差，即就是模型复杂度低，模型在训练集上就表现很差，没法学习到数据背后的规律。欠拟合基本上都会发生在训练刚开始的时候，经过不断训练之后欠拟合应该不怎么考虑了。但是如果真的还是存在的话，可以通过增加网络复杂度或者在模型中增加特征，这些都是很好解决欠拟合的方法。

过拟合是指训练误差和测试误差之间的差距太大。换句换说，就是模型复杂度高于实际问题，模型在训练集上表现很好，但在测试集上却表现很差。模型对训练集"死记硬背"（记住了不适用于测试集的训练集性质或特点），没有理解数据背后的规律，泛化能力差

获取和使用更多的数据（数据集增强）

采用合适的模型（控制模型的复杂度）

降低特征的数量

L1 / L2 正则化

Dropout

Early stopping（提前终止）