数据挖掘面试题链接地址：<https://zhuanlan.zhihu.com/p/40559938>

机器学习面试干货精讲：<https://zhuanlan.zhihu.com/p/32877396?refer=c_152307828>

1. 如何平衡偏差和偏差

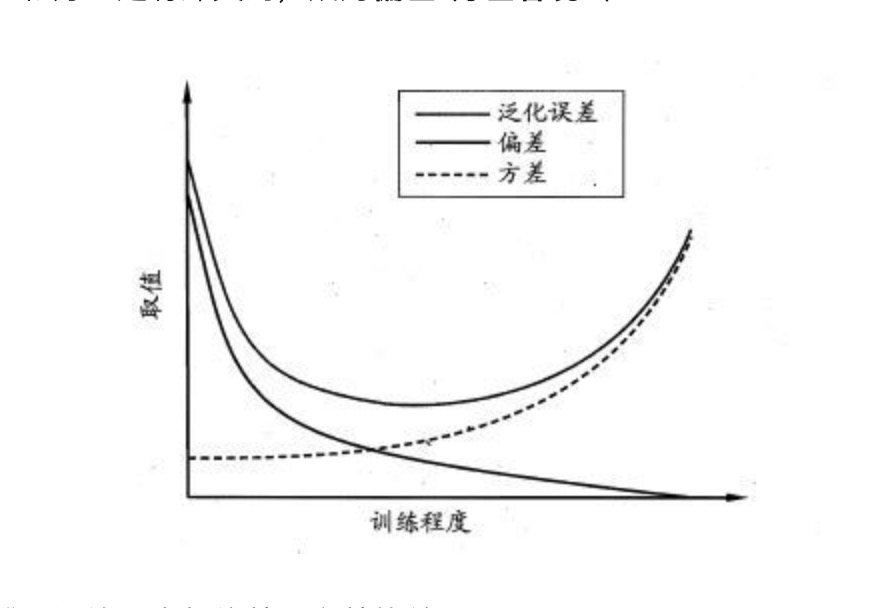
参考链接：<https://zhuanlan.zhihu.com/p/31852037>

偏差：度量了学习算法的期望预测与真实结果的偏离程度，刻画学习算法本身的拟合能力，偏差值越大，越偏离真实值

方差：衡量的距离期望值的波动性，方差越大，数据波动性越大，数据分布越分散

噪声：表达了当前任务上任何学习算法所能达到的期望泛化误差的下界，也就是最小值

泛化误差可以分解为偏差和方差和噪声之和



周志华老师描述了训练程度和偏差，方差的关系：

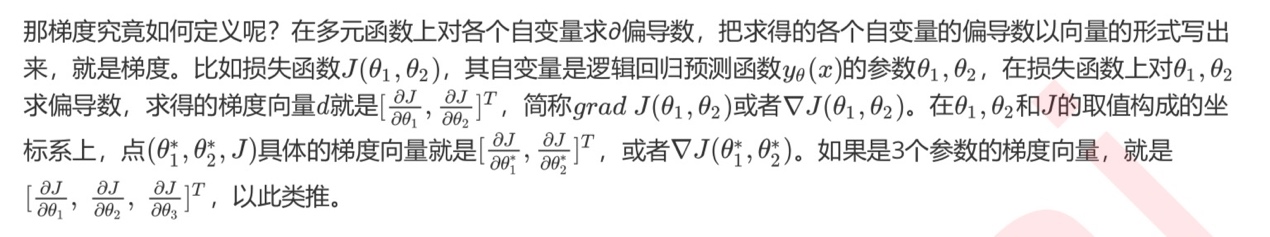
1）当训练程度不足时，学习器的拟合能力不够强，训练数据的扰动不足使学习器产生显著变化，偏差主导泛化误差

2）当训练程度加深，学习器的拟合能力逐渐增强，训练数据发生的扰动逐渐被学习器学到，方差主导泛化误差

3）训练程度充足后，学习器的拟合能力已经非常强，训练数据发生的轻微扰动都会导致学习器发生显著变化。训练数据的非全局特征如果被学习器学到，将会发生过拟合

1. 梯度下降

参考地址：<https://zhuanlan.zhihu.com/p/52048551>



eg：在机器学习中，我们使用梯度下降来求解一个损失函数J(θ)的最小值，就是我们沿着函数梯度下降的反方向不断更新损失函数的变量θ，来不断地让它去逼近最小值

1. 解释一下过拟合和欠拟合，如何解决这两种问题

参考链接：<https://zhuanlan.zhihu.com/p/47474872>

这就涉及到模型的泛化能力，如果模型在训练集上表现很差，在测试集上表现也很差，那么我们说模型可能存在欠拟合。我们可以增加模型参数，比如：增加模型复杂度，构建更多的特征，减少正则项

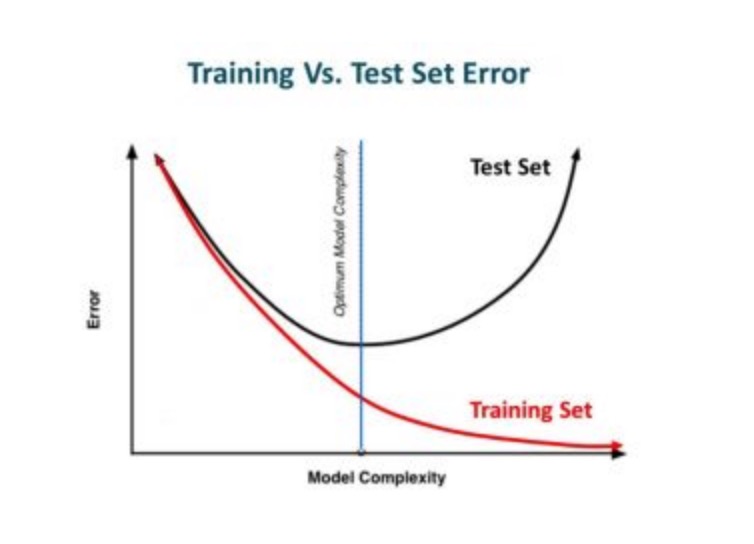
如果在训练集上表现很好，但在测试集上表现很差，那么模型可能存在过拟合问题。我们可以增大训练集，减少模型复杂度，增大正则项

1. 如何处理维度灾难
2. 什么是正则化项。为什么要使用正则化，说出一些常用的正则化方法

参考地址：<https://www.zhihu.com/question/20924039/answer/503837047>

正则化是一种技术，它向学习算法略微做些修正，从而让模型能更好的泛化。这样反过来能提高模型在不可见数据上的性能。

使用正则化的原因：虽然模型的发展方向是正确的。随着模型越来越复杂，训练误差有所下降，但是测试误差却没有下降，（ps：随着训练强度的加强，模型努力地学习训练数据中的细节和噪声数据，学习过了头，就会有如下的表现）如下图：



常用的正则化方法：

1）、L1 和L2正则化

会在原来损失函数的基础上加上一个正则项作为新的损失函数

L1范式表现为参数向量中的每个参数的绝对值之和，L2范式表现为参数向量中每个参数平方和的开方值

L1 和L2正则化区别：虽然L1 和L2正则化都可以控制过拟合，但是效果并不相同。当正则化强度逐渐增大时，参数的取值会逐渐变小，但是L1正则化会将参数压缩为0，L2正则化只会让参数尽量小，但不会取到0。

L1正则化逐渐加强的过程，携带信息量小，对模型贡献不大的特征的参数，会比携带信息量大，对模型贡献大的特征的参数更快变成0，所以L1正则化的本质就是一个特征选择的过程，掌管了参数的稀疏性。L1正则化越强，参数向量中越多的参数为0，参数越稀疏，选出来的特征就越少，以此来防止过拟合。如果特征量比较大，数据维度很高，我们倾向使用L1正则化。

L2正则化加强的过程，会尽量让每个特征都对模型有一些小的贡献，但携带信息少，对模型贡献不大的参数会非常接近0，通常来说，如果我们的主要目的只是为了防止过拟合，选择L2正则化就足够了。但是如果选择L2正则化后还是过拟合，模型在未知数据集上效果表现很差，就可以考虑L1正则化

2）丢弃法

3）数据增强法

减缓过拟合的简单方法就是增加训练集的数量。在机器学习中有时无法增加训练数据的数量，或者寻找标签数据的成本很高。我们可以通过一些其他方法扩充数据，eg：比如我们在处理手写数据集时，我们可以通过旋转，缩放，变换，裁剪这些手段来扩充数据集

4）提前停止（Early Stopping）

1. 讲解一下PCA原理

参考链接：<https://zhuanlan.zhihu.com/p/21580949>

参考链接：<https://zhuanlan.zhihu.com/p/47769240>

PCA是一种无监督模型，在具有m条自变量的数据中，PCA从中提取p个自变量，p<=m，可以较好包含数据绝大部分信息

1. X：首先对于m个自变量的矩阵，对X进行特征缩放
2. A: X的方差矩阵
3. 对A的特征值和特征向量，并从大到小进行排列
4. 选取P值：解释方差的百分比，并选取前P个特征值，使得前P个的特征值之和比上所有的特征值之和大于等于P

对应前P个特征值的特征向量，即为所选取的主成分分析

1. 为什么在神经网络中Relu激活函数会比Sigmoid激活函数用的更多？

参考链接：<https://www.zhihu.com/question/29021768/answer/43488153>

1）采用sigmoid等函数，算激活函数时（指数运算），计算量大，反向传播求误差梯度时，求导涉及除法，计算量相对大，而采用Relu激活函数，整个过程的计算量会节省很多

2）对于深层网络，sigmoid函数反向传播时，很容易出现梯度消失的情况（ps：sigmoid接近饱和区，变换太缓慢，导数趋于0，这种情况会造成信息丢失），从而无法完成深层网络的训练。

3）Relu会使一部分神经元的输出为0，这样就造成了网络的稀疏性，并且减少了参数的相互依存关系，从而缓解了过拟合问题的发生

1. 什么是数据标准化，为什么要进行数据标准化？

数据标准化：

每个特征维度代表含义可能不同，可能在量纲，数量级差异很大，为了可以进行综合比较，进行分析数据前，需要统一比较标准，消除特征之间的差异性，使数据具有可比性

1. 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

降维的好处：

1. 把数据维度降到2维或者3维并把将其可视化，便于观察数据的分布
2. 加快计算速度
3. 节省存储空间
4. 去除一些冗余特征
5. 如何处理缺失值数据？

1）删除 对于大量数据来说，如果缺失的数据量很小，可以直接简单粗暴一点直接删除

2）对于一些数值型数据，我可以采用均值，众数，中位数填充，对于一些字符类型的我们可以采用众数填充

3）使用拉格朗日函数填充

4）使用随机森林回归填充

1. 解释聚类算法

聚类算法是一种无监督学习算法。

K\_Means算法是将一组N个样本的特征矩阵X划分为K个无交集的簇，直观上来看簇是一组一组聚集在一起的数据，每一个簇中的数据可以看成是同一类。簇就是聚类结果的表现。

K\_Means的核心任务就是根据我们设定好的超参数k，找出k个最优的质心，并将距离质心距离最近的这些数据分别分配到这些质心代表的簇中去。

具体过程如下：

1. 随机抽取K个样本作为最初的质心
2. 开始循环
3. 将每个样本点分配到距离到距离他们最近的质心，生成k个簇
4. 对于每个簇，计算所有被分配到该簇的样本点均值作为新的质心
5. 当质心的位置不在发生变化时，迭代停止，聚类完成

典型的算法：K\_Means,DBSCAN,层次聚类，光谱聚类

1. 你会如何进行探索性数据分析(EDA)？

EDA的目的就是去挖掘数据中的重要信息。

首先我拿到数据，会找业务人员进行确认数据特征代表的含义以及特征的分布范围

1）借助pandas工具，读取数据

data.shape 查看数据的结构 看看数据有多少样本，样本有多少个特征

data.info 查看特征数据看看哪些是分类型数据，哪些是连续型数据，以及他们的数据类型

data.isnull().mean() 查看缺失值占总样本的比例

2）数据清洗工作 接着分析特征，删除一些可能对数据分析和预测没用的特征

删除数据中重复的数据，填补缺失值（ps:平均数，众数，中位数，拉格朗日函数填充，随机森林回归填充等等）

data.describe([0.01, 0.1, 0.25, 0.50, 0.75, 0.9, 0.99]).T 借助

describe更好查看数据的均值，方差，最小值，最大值，看看大概看看数据是不是存在左偏，右偏问题，还可以根据数据特征范围限制，常识，大致判断是否存在异常值，这个最终还是要跟业务人员确认的

2）使用PCA降维，把数据降维到2维或3维，并将数据可视化，看看数据的大致分布

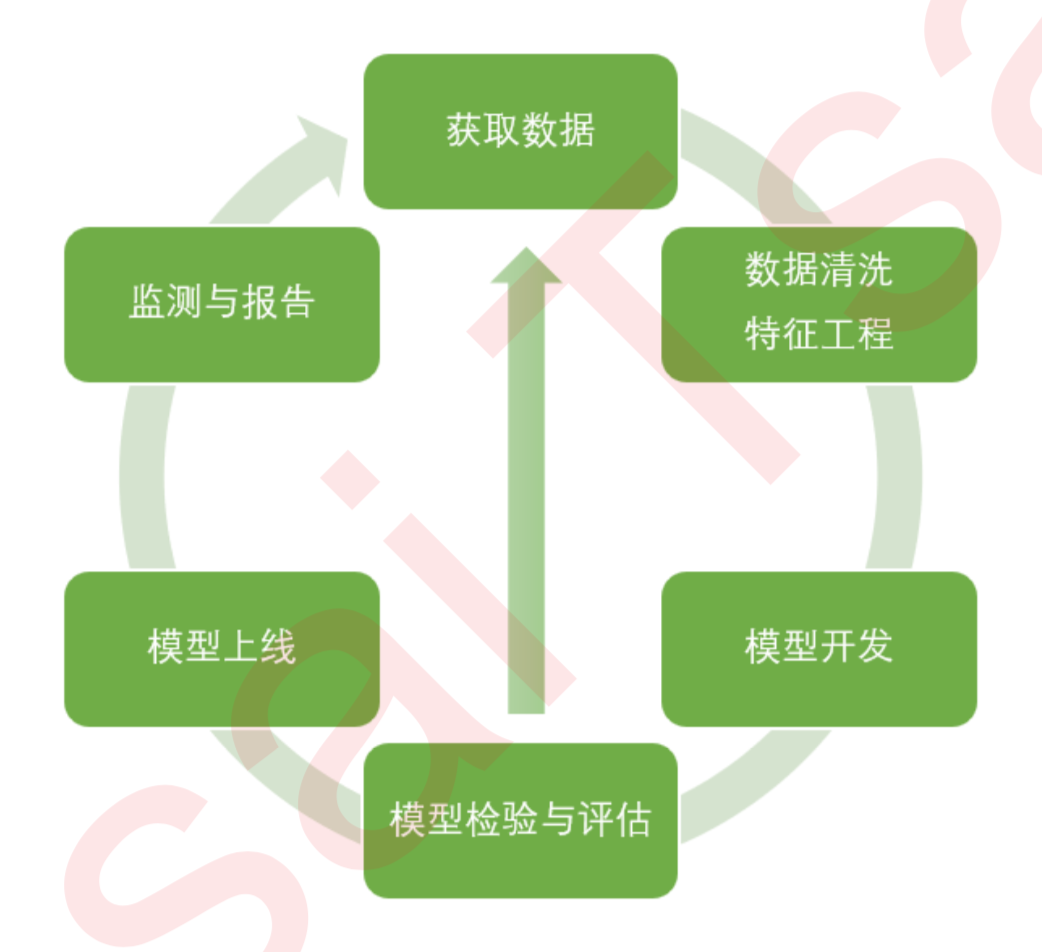
3）集合特征工程，比如构建新的特征

4）建立模型，根据上面数据的大致分布，初步形成数据建模的思路，如果是线性可分数据，可能接下来我们要尝试一下逻辑回归，svm，如果是非线性可分数据，可能会倾向于考虑多项式回归，决策树，随机森林，svm等等模型

5）建立模型后，查看正确率，召回率，ROC曲线, AUC面积等等指标

6）调参，并结合学习曲线，网格搜索，寻找最优参数，最后结合正确率，ROC曲线, AUC面积，召回率等等评测模型

具体流程可以参考逻辑回归的评分卡流程：



1. 决策树

决策树是一种非参数的有监督学习方法，他能够从一系列有特征和标签的数据中总结出决策规则，并用树形图的结构来呈现这些规则，以解决分类和回归的问题。

最初的问题所在的地方叫做根节点，在得到结论前的每一个问题都是中间节点，而得到的每一个结论都叫做叶子节点

1）决策树算法的两个核心问题：

第一个问题：如何从数据表中找出最佳节点和最佳分支？

我们定义了用来衡量分支质量的指标不纯度，不存度基于叶子节点来计算的。分类树的不纯度用基尼系数或者信息熵来衡量，回归树的不纯度用均方差来衡量。每次分支时，决策树对所有的特征进行不纯度计算，选取不纯度最低的特征进行分支，分支后，又再对被分支下的不同取值下，计算每个特征的不纯度，继续选取不纯度最低的特征进行分支。

第二个问题：如何让决策树停止生长，如何防止过拟合？

每分枝一层，树整体的不纯度会越来越小，决策树追求的是最小不纯度。当没有更多的特征可用或者整体的不纯度指标已经最优，决策树就会停止生长

防止过拟合：决策树非常容易过拟合，我们对决策树进行剪枝，正确的剪枝是优化决策树算法的核心。

2）ID3,C4.5,CART著名的决策树算法：

ID3使用信息增益作为选择特征的准则

C4.5是使用信息增益比作为特征选择的准则

CART使用Gini指数作为特征选择的准则

信息熵概念：信息熵表示随机变量不确定的度量。熵越大，则随机变量的不确定性越大

信息增益：得知特征x的信息而使类Y的信息的不确定性减少的程度

g(D,A) = H(D) – H(D|A) H(D)为特征的熵，H(D|A)条件熵

3）ID3的局限性

只能处理一些离散型变量，如果要处理连续型变量，则首先需要把连续型变量离散化；对缺失值比较敏感，使用ID3之前需要提前对缺失值进行处理；没有剪枝设置，容易导致过拟合

1. 信息熵为什么以2为底

H = -∑p \* log(2)p 通常以2为底，因为信息熵的单位是bit

log如果以2为底数的话，信息熵的单位就是（bit），

以e为底数的话，信息熵的单位就是（nat），

以10为底数的话，单位就是（hat）

1. 集成算法

多个模型集成成为模型集成评估器，组成集成评估器的每个模型都叫做基评估器。

Bagging的核心思想就是构建多个相互独立的评估器，然后对其预测进行平均或则多数表决原则来决定集成评估器的结果。装袋法的代表模型就是随机森林。

Boosting的基评估器是相关的，是按顺序一一构建的。其核心思想就是结合弱评估器的力量一次次对难以评估的样本进行预测，从而形成一个强评估器。提升法的代表模型就是Adaboost和梯度提升树。

Bagging：通过有放回的随机抽样技术形成不同的训练数据。初始训练集中大约会有63.2%的数据出现在采样中，大约37数据在包外，当数据量足够大时，可以使用包外数据作为测试集。（ps：每一个样本被抽到的概率为1-（1-1/n）^n， 当n足够大时，这个概率收敛于1-1/e，大约63.2的数据会被抽到）

Bagging的代表模型就是随机森林：具体构造过程如下

1. 从原始训练集中使用bootstrap方法有放回随机抽样选出m个样本，共进行K次采样，生成K个新的训练集
2. 对于K训练集，我们分别训练K个决策树模型
3. 对于单个决策树模型，假设训练样本的特征个数为n，那么每次分裂时根据信息增益/信息增益比/基指数选择最好的特征进行分裂
4. 每棵树都一直这样分裂下去，直到该节点的所有训练样例都属于同一类
5. 将生成的多颗决策树组成随机森林。对于分类问题，多数投票原则决策最终的分类结果；对于回归，我们采用多棵树的均值作为最终的预测值

Boosting：工作机制如下

1. 先从训练集找那个学习一个基学习器
2. 然后根据学习器的表现对训练样本进行调整，使得先前基学习器做错的样本在后续收到更多的关注
3. 基于调整后的样本分布训练下一个基学习器
4. 如此反复，直到基学习器数据达到T, 最终对这T基学习器加权平均

AdaBoost 原理：

在Boosting中，每一轮，如何改变训练数据的概率分布或者权值分布？

提高那些被上一轮弱分类器误分样本的权值，降低那些被正确分类样本的权值。这样的话，分错的样本数据在下一次被抽取的概率就会增大

如何将弱分类器组合成强分类器？

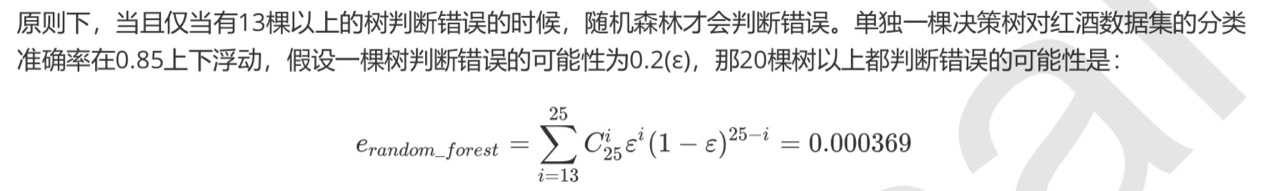
采用加权多数表决。具体的，加大分类错误率低的分类器的权值，使其在表决中起较大作用，减少分类误差率大的弱分类器的权值，使其在表决中起较小作用。

AdaBoost优缺点：

优点：构造简单，容易理解，精度高，不易发生过拟合

缺点：对异常数据敏感，异常样本数据在迭代中可能会获得较高的权重，影响最终的强学习器的预测结果

1. 随机森林用什么方法，保证集成效果一定浩宇单个分类器



1. 调参的基本思想

1）通过学习曲线或者网格搜索

2）非常正确的调参思路和方法

3）对模型评估指标的理解

4）对数据的感觉和经验

5）使用你的洪荒之力不断地去调试

1. GBDT

参考链接：<https://www.cnblogs.com/pinard/p/6140514.html>

回归树（DT）：GBDT中的树全部都是回归树，核心就是累加所有的结果作为最终的结果。只有回归树的结果累加起来才有意义，分类树结果的累加是没有意义的。GBDT调整只有可以用于分类，但是内部还是回归树

梯度迭代（GB）：首先Boosting 是一种集成方法。通过弱分类器的组合得到的强分类器，他是串行的，几个弱分类器之间是一次训练的。GBDT的核心就在于，每一棵树学习基于之前所有树的结论和残差

Shrinkage：核心思想认为每次走一小步来逼近结果的效果，要比每次迈一大步很快逼近结果的方式更容易防止过拟合。

Shrinkage的具体做法：仍然以残差作为学习目标。但是对于残差学习出来的结果，只累加一小部分(step\*残差)逐步逼近目标，step一般都比较小0.01~0.001，导致各个树的残差是渐变不是陡变的。

本质上，Shrinkage为每一棵树设置了一个weight，累加时要乘以这个weight，但是和Gradient无关。这个weight就是step

GBDT和随机森林的区别：

GBDT和随机森林的相同点：都是由多棵树组成；最终的结果都是由多棵树共同决定；

GBDT和随机森林的不同点：

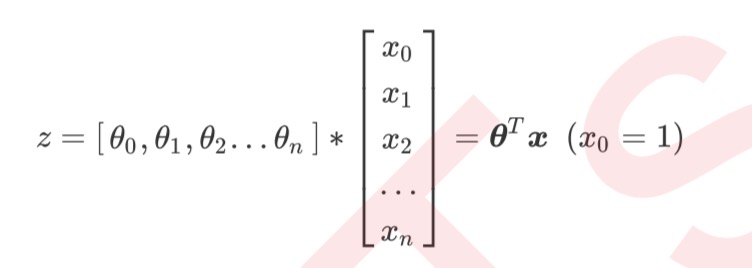
1. 组成随机森林的树可以是分类树、回归树；GBDT只能是回归树
2. 组成随机森林采用的是Bagging思想，随机森林的树可以使并行生成；GBDT采用的是Boosting思想，只能串行生成
3. 随机森林使用的是多数投票原则或者简单平均作为最终的返回值，GBDT则是所有结果进行加权累加后作为最终的结果
4. 随机森林对异常值不敏感，GBDT对异常值敏感
5. 随机森林通过减小模型的方差提高性能，GBDT通过减少模型偏差提高性能
6. Sigmoid函数和性质

Sigmoid函数是一个s型函数，当自变量趋近与正无穷时，因变量趋近于1，自变量趋近于负无穷时，因变量趋近于0，它能够将任何实数映射到（0，1）之间。因为这个性质，sigmoid函数也可以当作是归一化的一种方法，与MinMaxSclaer同理，是属于数据预处理中的缩放功能，可以将数据压缩到[0, 1]。区别在于MinMaxSclaer归一化后可以取到0和1，sigmoid函数只能无限趋近于0和1。

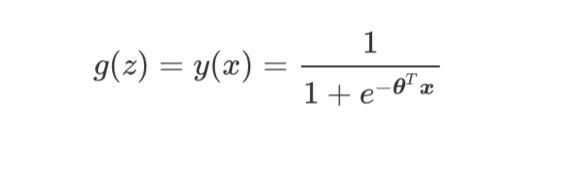
1. Logistic 回归

逻辑回归是一种名为“回归“的线性分类器，其本质是由线性回归变化而来的，一种广泛使用与分类问题中广义回归算法

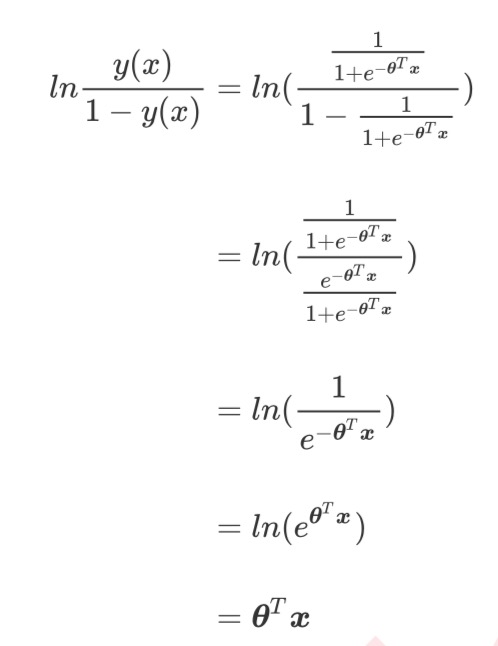
线性回归的表达式：



线性回归结合Sigmoid函数就形成二元逻辑回归的一般形式



形似几率 :p/ (1-p),在此基础上取对数就得到了对数几率：ln（p/(1-p)）



可以看出形似几率取对数的本质就是我们线性回归，我们实际上是在对线性回归模型的预测结果取对数几率来让其结果无限逼近0和1.因此对应的模型被称为“对数几率回归“，也就是我们的逻辑回归。

逻辑回归的核心任务：求解θ来构建一个尽量拟合数据的预测函数y(x),并通过向预测函数中输入特征矩阵来获取标签y值。

逻辑回归的数学目的：求解能够让模型对数据拟合程度最高的参数θ值的值，以此构建预测函数y(x),然后将特征矩阵输入预测函数来计算出逻辑回归的结果。

损失函数：我们使用损失函数这个评估指标，来衡量参数为θ的模型拟合训练集时产生信息损失的大小，并以此来衡量参数θ的优劣。

1. SVM 支持向量机

支持向量机的分类方法，是在这组分布中找出一个超平面作为决策边界，使模型在数据上的分类误差尽量小，尤其在未知数据集上的分类误差尽量小。

了解几个概念：超平面和边际

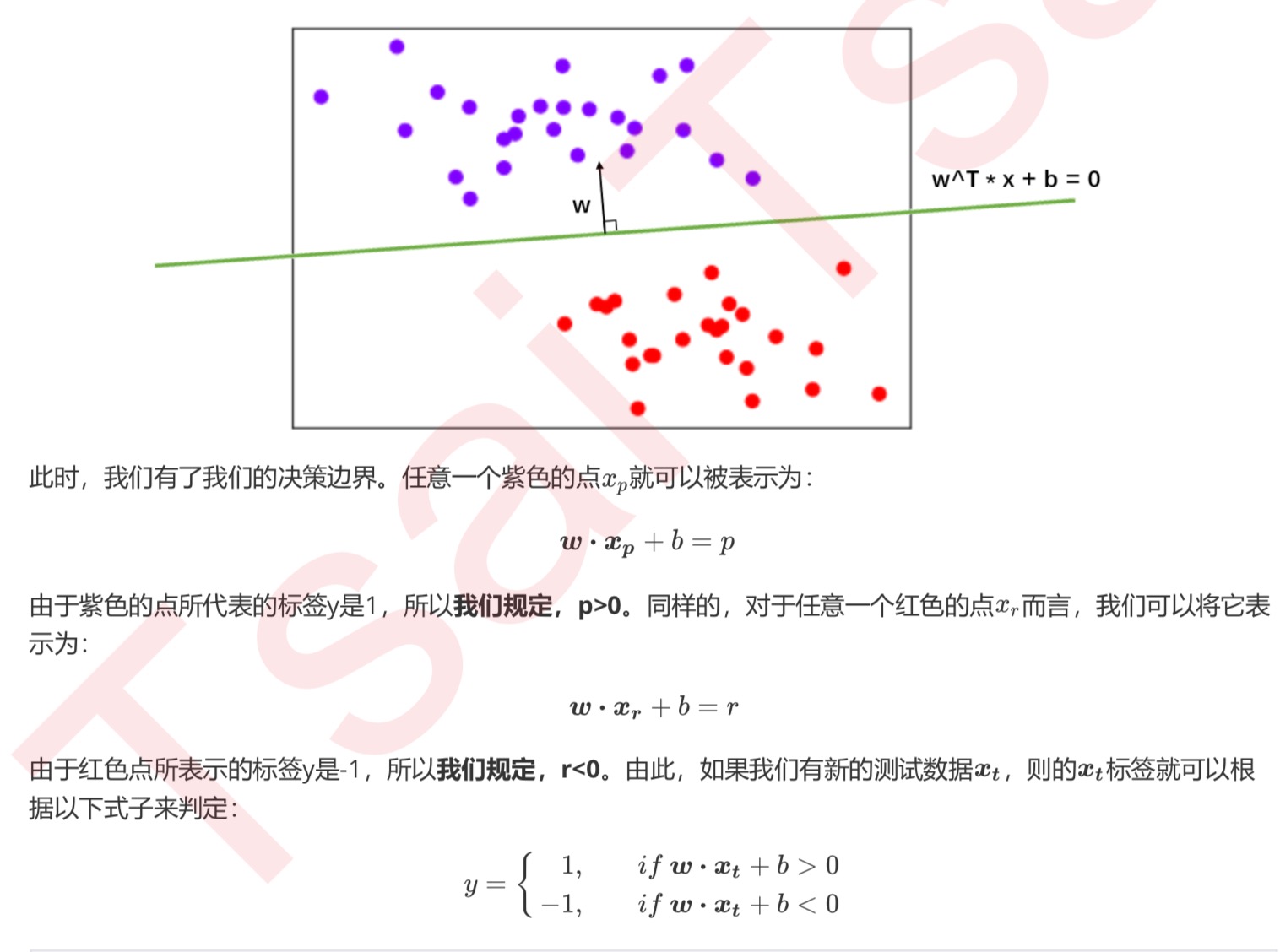
超平面：超平面比数据的维度少一维 eg：在二维数据中，超平面就是一条直线；在三维数据中，超平面就是一个面

决策边界的边际：由决策边界分别向两边平移，直到碰到决策边界最近的样本点停下，形成两个新的超平面b1，b2， b1和b2之间的距离就是决策边界的边际。

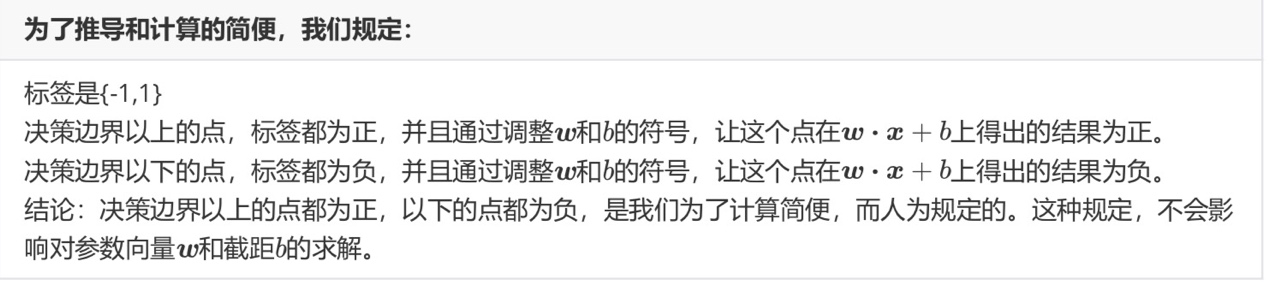
支持向量机就是通过找出边际最大的决策边界，来对数据进行分类的分类器。

在svm中我们使用w^T \* x + b 这个表达式来表示我们的决策边界。我们的目标就是求解能够让边际最大化的决策边界，所以我们要求解参数w和b

支持向量机的一些推导：规定紫色点的标签为+1，红色的点的标签为-1



这里假设p是正值，r是负值，是为了使它的符号和我们的标签的符号一致，为了我们后续计算和推导的简便。我们的规定如下：



非线性svm和核函数

核函数能够帮助我们解决三个问题：

1. 有了核函数，我们无需担心φ究竟应该是怎么样，因为非线性svm中的核函数都是正宗的核函数，确保了高维空间中任意两个向量的点积一定可以被低维空间中的两个向量的某种计算来表示
2. 使用核函数计算低纬度中的向量关系比计算原本φ(xi) . φ(xj)要简单太多了
3. 因为计算是在原始空间中进行，所以避免了维度诅咒问题
4. 想相信