# 分类算法总结

## k近邻算法

### 原理

近朱者赤近墨者黑，如果一个待分类样本在特征空间的k个最相似（即特征空间中K近邻）的样本中的大多数属于某一个类别，则这样本也属于这个类别

### 输入

**数据集**：样本中数据集中每个样本是有标签的

**样本间距离的计算方法**：欧式距离、余弦距离、汉明距离、曼哈顿距离

**K值得选取：**K值太大，分类偏差大（极端方法，K值是所有样本数）

K值太小，方差大容易过拟合

K值一般在3~10之间，或者是k等于训练数据的平方根

### 输出

预测样本的分类结果

### 优点

1、简单，易于理解，易于实现，无需参数估计，无需训练

2、对异常值不敏感（个别噪音数据对结果的影响不是很大）

3、适合对稀有事件进行分类

4、适合于多分类问题（对象具有多个类别标签），KNN要比SVM表现要好

### 缺点

1、对测试样本分类时的计算量大，内存开销大（需要保存全部的数据集），因为对每一个待分类的文本都要计算它到全体已知样本的距离，才能求得它的K个最近邻点（耗时）。

2、可解释性差，无法告诉你哪个变量更加重要，无法给出决策树那样的规则/

3、K值得选择：样本不均衡时，容易偏向样本中类别多的那个类别/

4、KNN是一种消极学习方法、懒惰算法

**积极学习法**：先根据训练集构造出分类模型，根据分类模型对测试集分类。

**消极学习法**（基于实例的学习法）：推迟建模，当给定训练元组时，简单地存储训练数据，一直等到给定一个测试样本。

### 性能问题

使用KNN，可以很容易的构造模型，但在对待分类样本进行分类时，为了获得K近邻，必须采用暴力搜素的方式，扫描全部训练样本并计算其与待分类样本之间的距离，系统开销很大

### 应用KNN常见的问题

K值设定

**经验规则：k一般低于训练样本数的平方根**

类别如何判定最合适？

投票法没有考虑近邻的距离的远近，距离更近的近邻也许更应该决定最终的分类，所以加权投票法更恰当一些。

如何选择合适的距离衡量？

高维度对距离衡量的影响：众所周知当变量数越多，欧式距离的区分能力就越差

训练样本是否要一视同仁？

在训练集中，有些样本可能是更值得依赖的。 可以给不同的样本施加不同的权重，加强依赖样本的权重，降低不可信赖样本的影响

### KNN算法的应用领域

推荐系统、字符识别、文本分类、图像识别、改进约会网站、手写识别系统、电影分类

### 代码

dataSetSize = dataSet.shape[0]

a = array([4,3])

diffMat = tile(intX,(dataSetSize,1))-dataSet

sqDiffMat = diffMat \*\*2

sqDistances = sqDiffMat.sum(axis=1)

distances = sqDistances\*\*0.5

sortedDistIndicies = distances.argsort()

classCount={}

for i in range(k):

voteIlabel = labels[sortedDistIndicies[i]]

classCount[voteIlabel]=classCount.get(voteIlabel,0) + 1

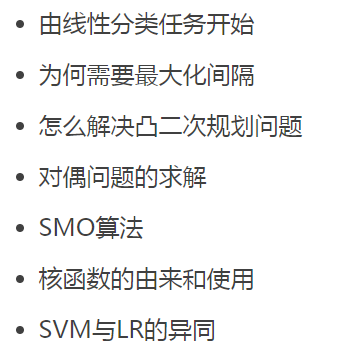
print classCount

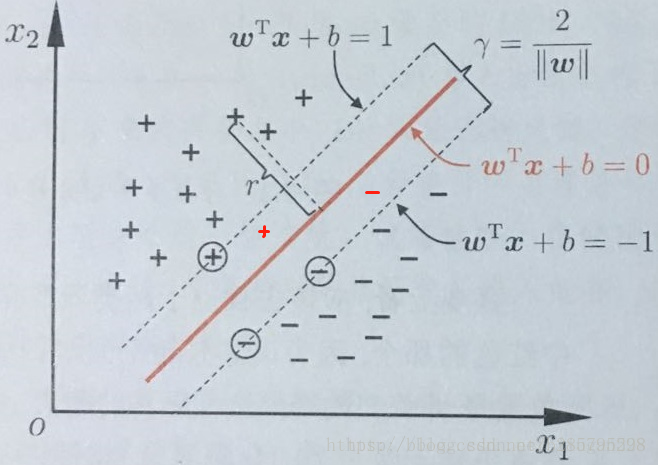
sortedClassCount = sorted(classCount.iteritems(),key = operator.itemgetter(1),reverse =True)

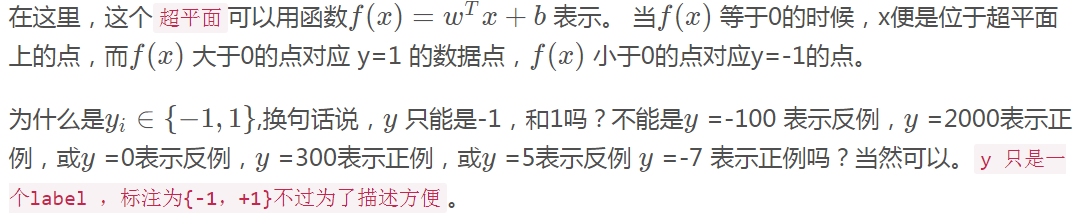
return sortedClassCount[0][0]

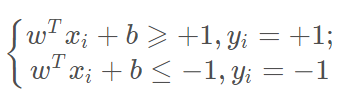
## 支持向量机

### 原理



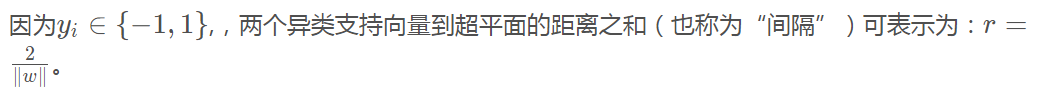


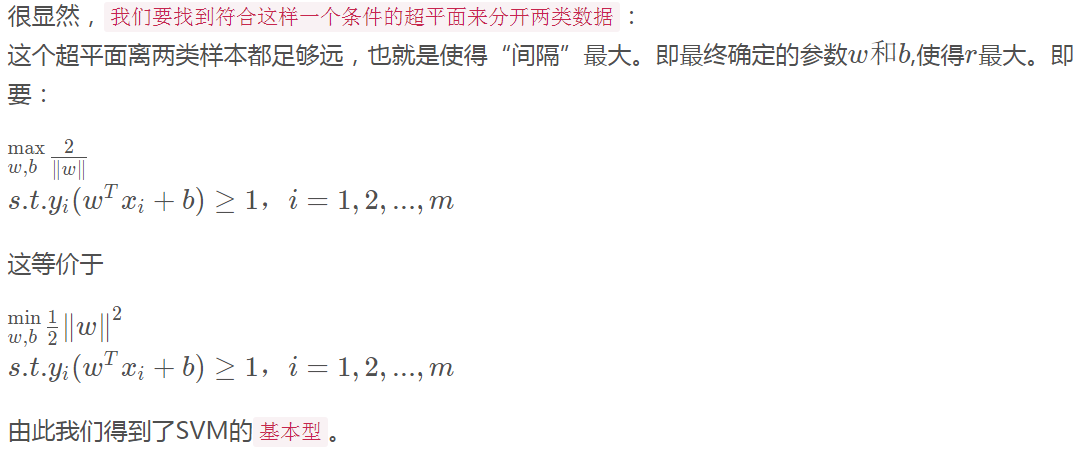


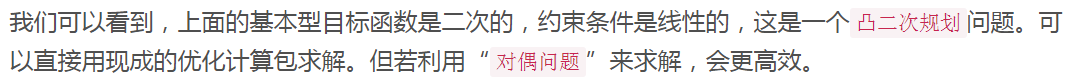


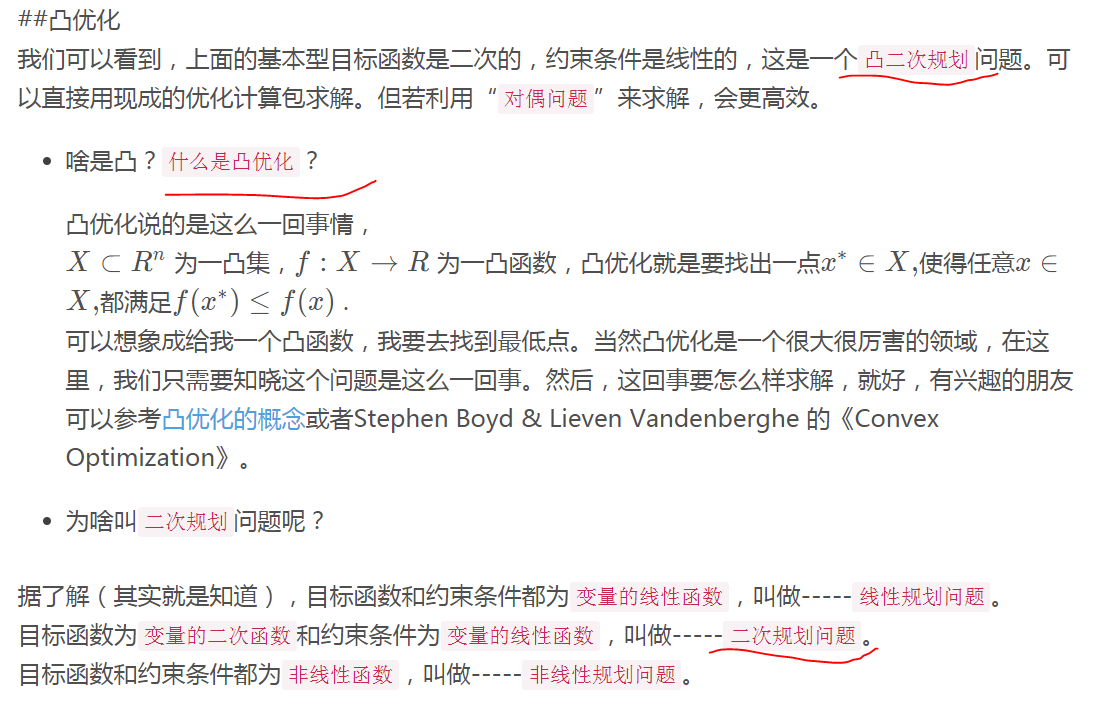
点到直线的距离为：

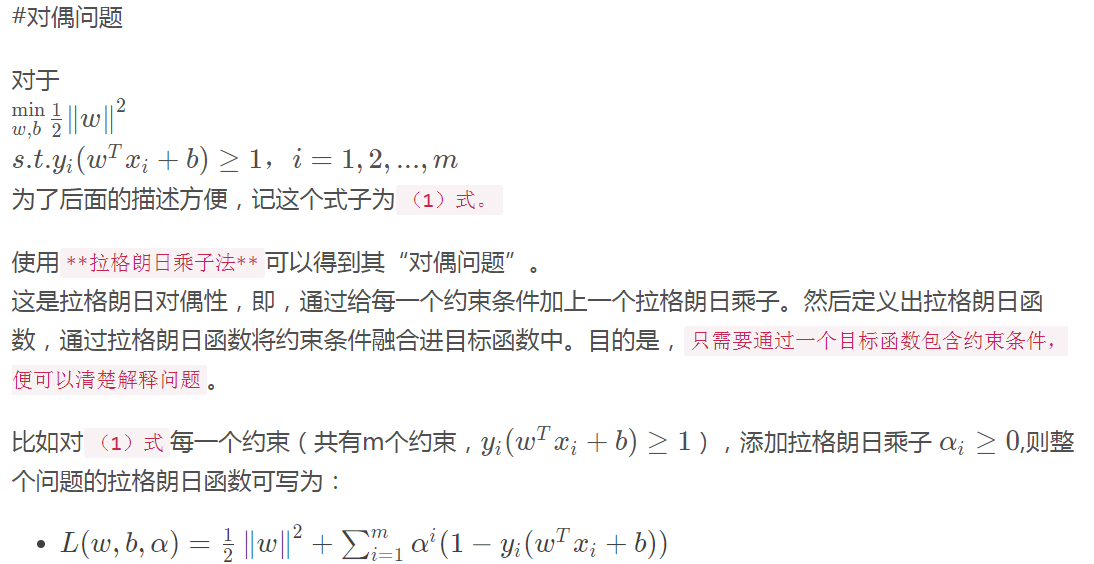


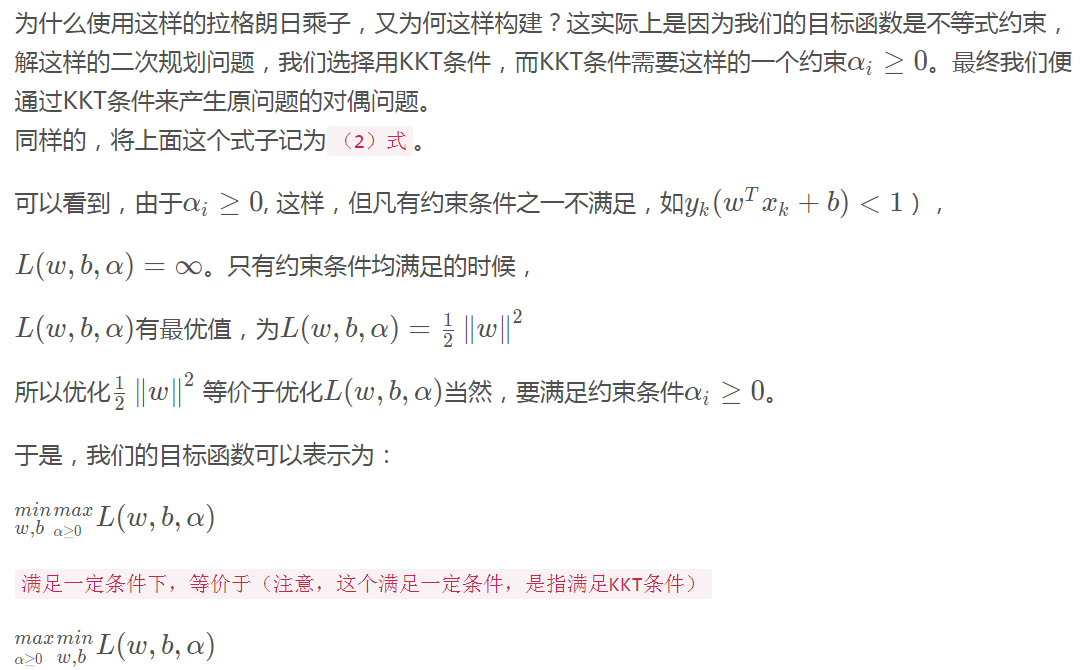


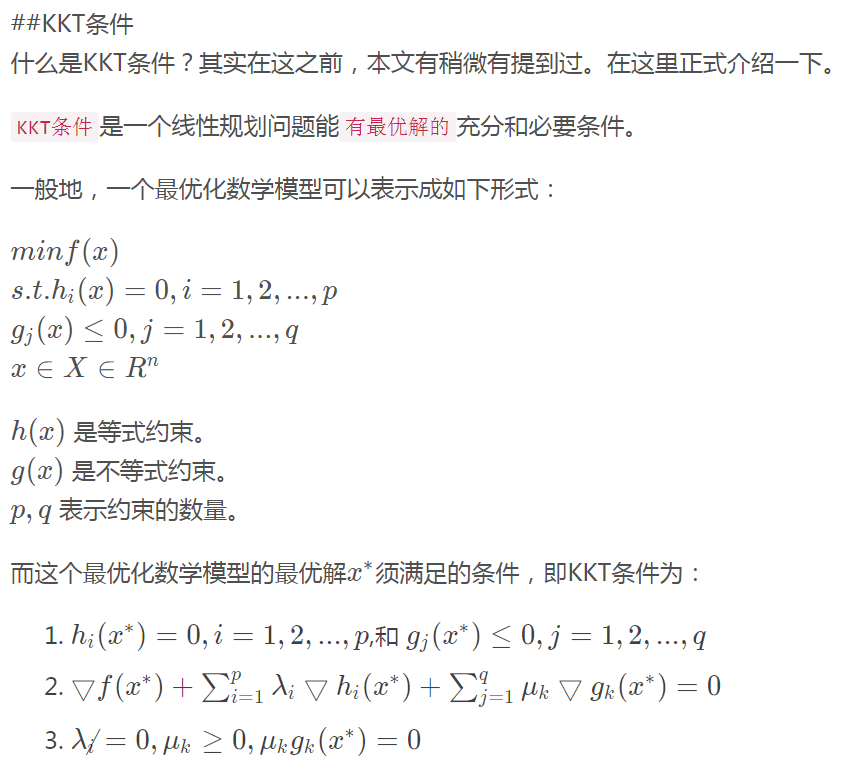


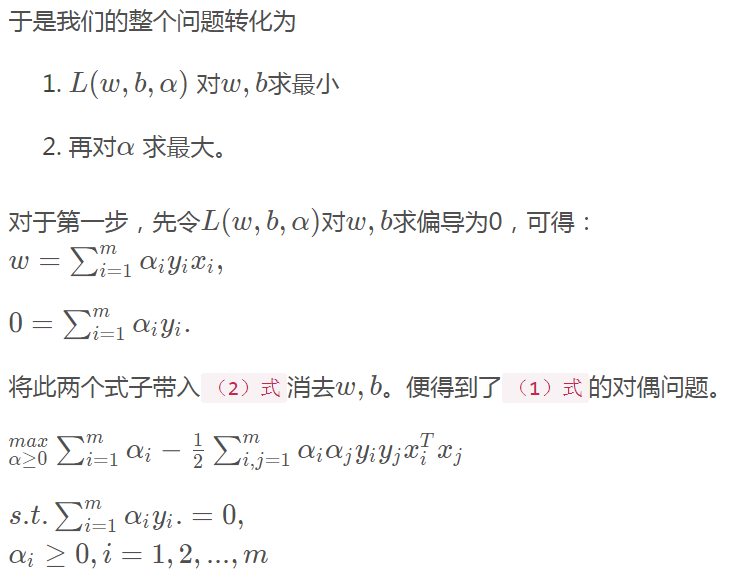


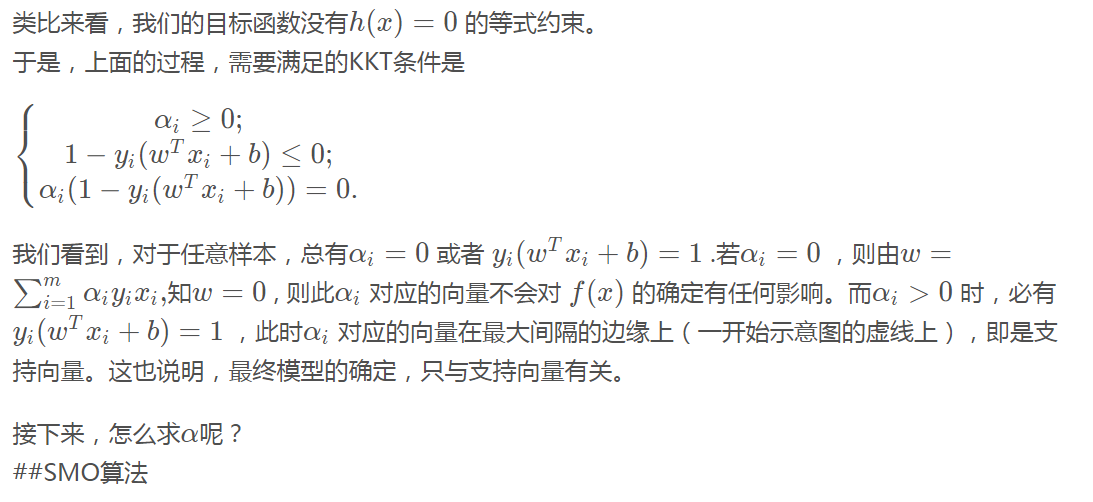








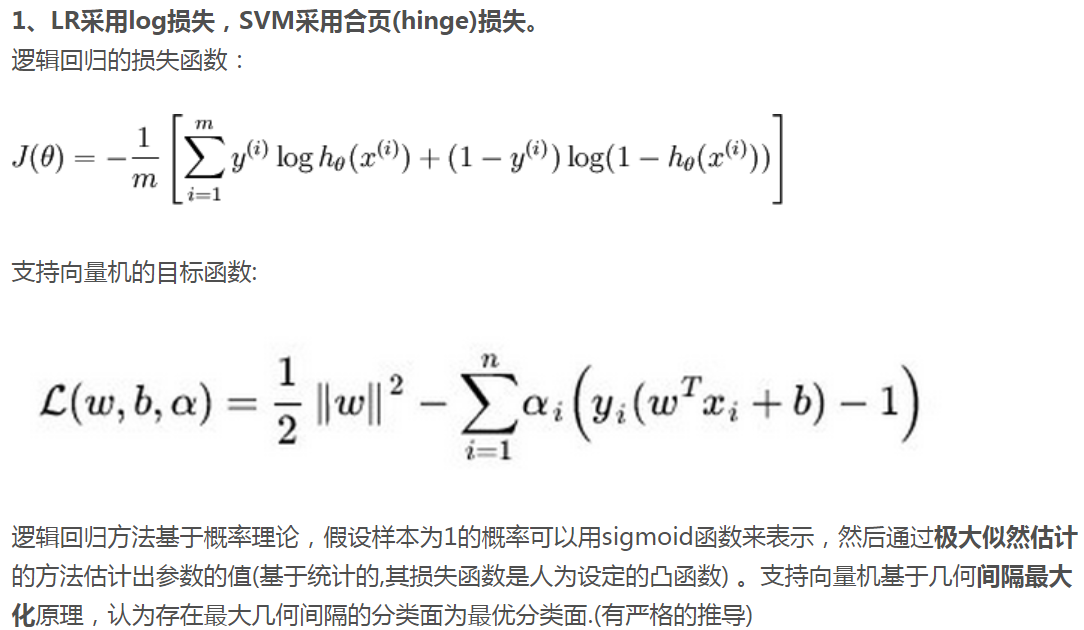




<https://blog.csdn.net/b285795298/article/details/81977271>

### 损失函数

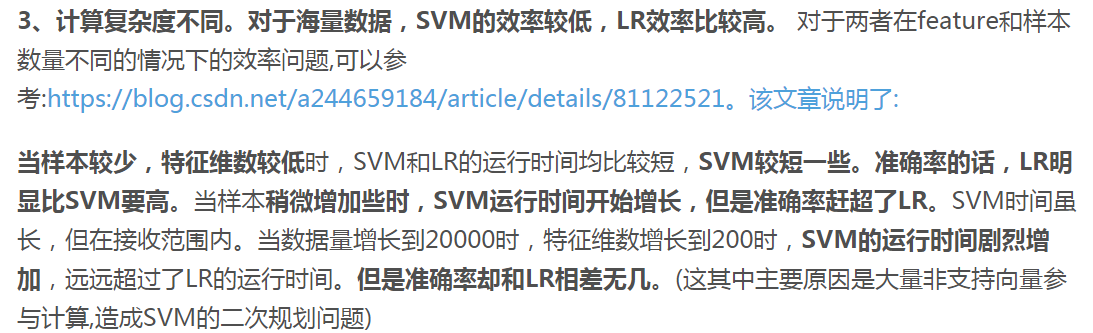
### SVM与LR的不同点

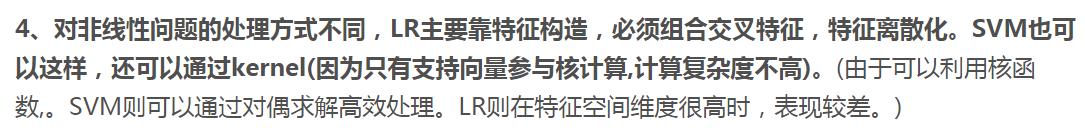


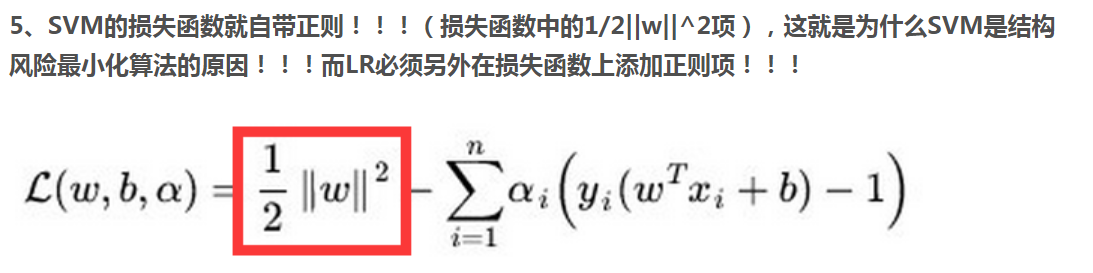


支持向量机改变非支持向量样本并不会引起决策面的变化

逻辑回归中任何样本都会引起决策面的变化







### 优点

1、对于线性不可分的情况可以通过核函数，映射到高维特征空间实现线性可分。

2、SVM学习问题可以表示为凸优化问题，因此可以利用已知的有效算法发现目标函数的全局最小值。而其他分类方法（如基于规则的分类器和人工神经网络）都采用一种基于贪心学习的策略来搜索假设空间，这种方法一般只能获得局部最优解。

3、小集群分类效果好

### 缺点

1、SVM仅仅只限于一个二类分类问题，对于多分类问题解决效果并不好。

2、仅局限于小集样本，对于观测样本太多时，效率较低；由于SVM是借助二次规划来求解支持向量，而求解二次规划将涉及m阶矩阵的计算（m为样本个数），当m数组很大时该矩阵的存储和计算将耗费大量的内存和运算时间

3、寻求合适的核函数相当难

### 应用领域

小样本分类

### 代码

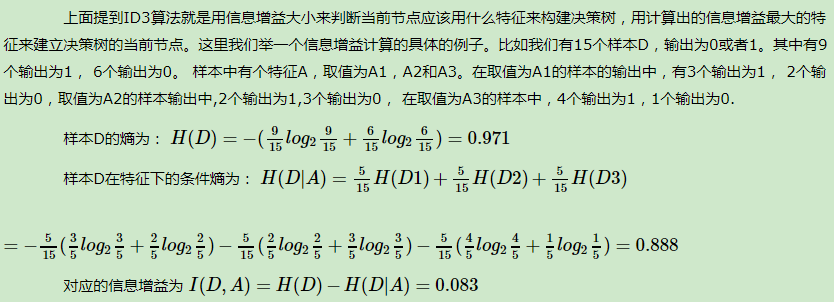
需要调试的参数，软间隔中有个惩罚错误分类因子C，还有核函数的选择，核函数里的参数也需要调整

## 决策树

### ID3原理



信息增益： ，熵越大，不确定性越大



决策树ID3正是采用这个信息增益为基础的，采用增益最大的特征作为划分特征

### 优点

简单

### 缺点

1、ID3没有考虑连续特征，比如长度，密度都是连续值，无法在ID3运用

2、ID3采用信息增益大的特征优先建立决策树节点。数据不平衡问题，会影响

3、ID3算法对缺失值得情况没有考虑

4、没有考虑过拟合问题

### 应用领域

### 代码

### C4.5原理



信息增益与特征熵之比

### 优点

1、将连续特征离散化。取相邻两样本值得平均数，分别计算以该点作为二元分类点时的信息增益。解决了ID3不能处理的连续值问题

2、信息熵增益与特征熵的比值，校正信息增益容易倾向于取值较多的特征的问题

3、解决缺失值问题。设置权重来解决

4、引入了正则化系数进行初步的剪枝

### 缺点

1. 容易过拟合
2. C4.5生成的是多叉树，即一个父节点可以有多个节点。很多时候，计算机中二叉树模型会比多叉树运算效率高。采用二叉树，可以提高效率
3. C4.5只能用于分类

4、基于信息熵，涉及到大量对数运算，比较耗时

### CART分类树原理

基尼系数：

对于给定样本D，假设有K个类别，第k个类别的数量为 ，则样本D的基尼系数表达式为：

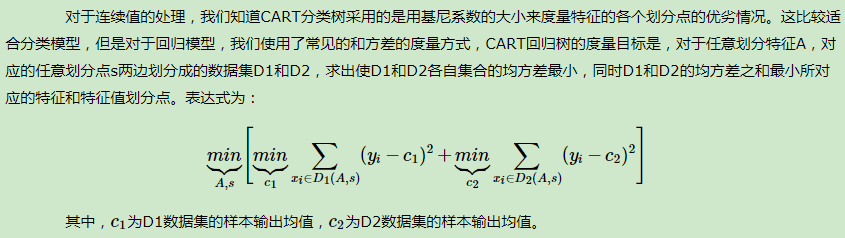
对于样本D，如果根据特征A的某个值a，把D分为D1和D2两部分，则在特征A的条件下，D的基尼系数表达式为：



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。

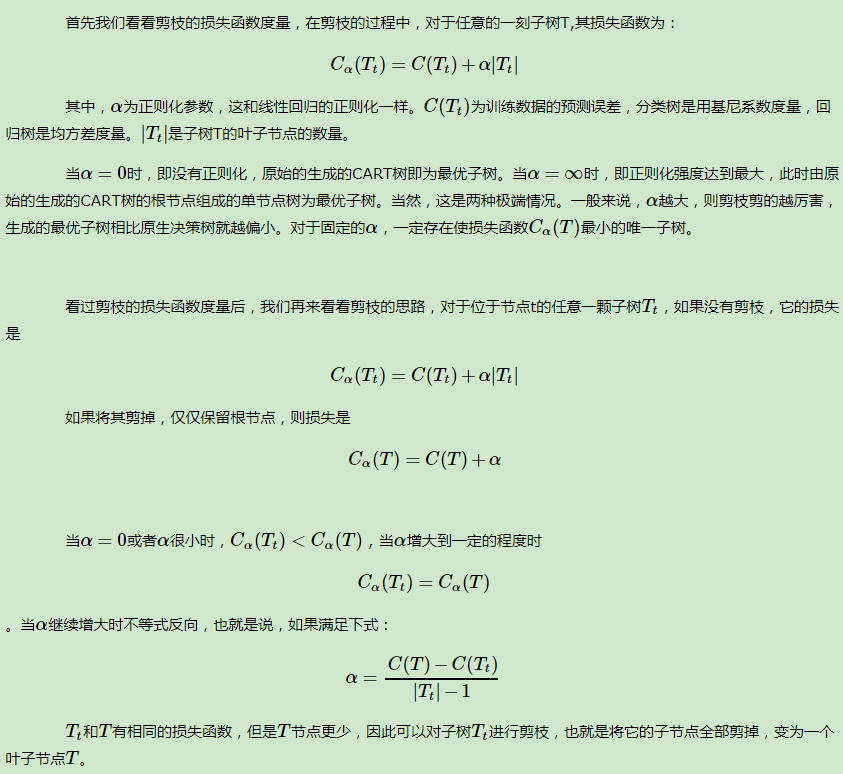
CART算法建立起来的是二叉树

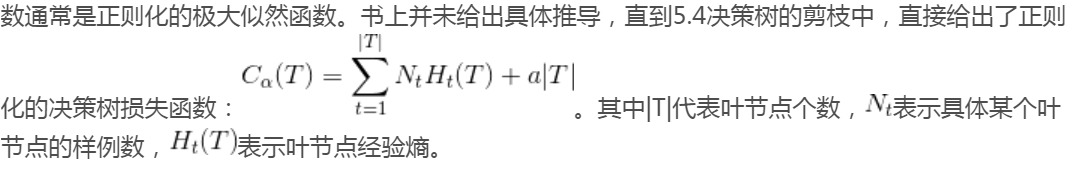
### CART回归原理

CART剪枝损失函数







剪枝过程，计算各个节点的正则化阈值，选择a最小的作为阈值，进行剪枝

### 决策树算法小结



优点

1）简单直观，生成的决策树很直观。

　　　　2）基本不需要预处理，不需要提前归一化，处理缺失值。

　　　　3）使用决策树预测的代价是O(log2m)O(log2m)。 m为样本数。

　　　　4）既可以处理离散值也可以处理连续值。很多算法只是专注于离散值或者连续值。

　　　　5）可以处理多维度输出的分类问题。

　　　　6）相比于神经网络之类的黑盒分类模型，决策树在逻辑上可以得到很好的解释

　　　　7）可以交叉验证的剪枝来选择模型，从而提高泛化能力。

　　　　8） 对于异常点的容错能力好，健壮性高。

缺点

1）决策树算法非常容易过拟合，导致泛化能力不强。可以通过设置节点最少样本数量和限制决策树深度来改进。

　　　　2）决策树会因为样本发生一点点的改动，就会导致树结构的剧烈改变。这个可以通过集成学习之类的方法解决。

　　　　3）寻找最优的决策树是一个NP难的问题，我们一般是通过启发式方法，容易陷入局部最优。可以通过集成学习之类的方法来改善。

　　　　4）有些比较复杂的关系，决策树很难学习，比如异或。这个就没有办法了，一般这种关系可以换神经网络分类方法来解决。

　　　　5）如果某些特征的样本比例过大，生成决策树容易偏向于这些特征。这个可以通过调节样本权重来改善。

## 朴素贝叶斯

### 原理



要预测当Y取哪个类别时， 概率最大，则采用上面的贝叶斯公式：目标是



由于对于所有的类别计算时，上诉的分母是一样的，因此，预测公式可以简化为：



接着利用朴素贝叶斯的独立性假设(如果特征之间非常不独立怎么办，那就尽量不要用朴素贝叶斯了)，就可以得到通常意义上的朴素贝叶斯推断公式：



### 优点

1、朴素贝叶斯模型有稳定的分类效率

2、对小规模的数据表现很好，能处理多分类任务，适合增量式训练，尤其是数据量超出内存时，可以一批批的去增量训练

3、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

1、理论上，朴素贝叶斯模型与其他分类方法相比具有最小的误差率。但是实际上并非总是如此，这是因为朴素贝叶斯模型给定输出类别的情况下,假设属性之间相互独立，这个假设在实际应用中往往是不成立的，在属性个数比较多或者属性之间相关性较大时，分类效果不好。而在属性相关性较小时，朴素贝叶斯性能最为良好。对于这一点，有半朴素贝叶斯之类的算法通过考虑部分关联性适度改进。

2、需要知道先验概率，且先验概率很多时候取决于假设，假设的模型可以有很多种，因此在某些时候会由于假设的先验模型的原因导致预测效果不佳。

3、由于我们是通过先验和数据来决定后验的概率从而决定分类，所以分类决策存在一定的错误率。

4、对输入数据的表达形式很敏感。

### 应用场景

文本分类

### 代码

对文本的统计有两种模式，词袋型和词库型

## 逻辑回归

### 原理

损失函数最小化，损失函数由最大似然法来推导



### 优点

1、形式简单，模型的可解释性非常好

2、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。

3、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

### 缺点

1、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

2、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。

3、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

4、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。

### 应用场景

### 代码

损失函数一般有四种，平方损失函数，对数损失函数，HingeLoss0-1损失函数，绝对值损失函数

调参包括，正则化系数，梯度下降次数，步长

库函数使用参数调整：

正则化选择参数：penalty，是L1还是L2

优化算法选择参数：solve 对逻辑所示函数的优化方法

分类方式选择参数：multi\_class 分类方式，一对多还是一对一

类型权重参数： class\_weight 标示样本中各类型的不同权重

样本权重参数： sample\_weight 样本不平衡问题中，对样本赋予的权重

# 深度学习算法总结

## 图像

## NLP

# 面试总结

## 平安

1、残差网络结构

2、lstm结构

3、卷积层作用

4、BN层作用

5、tf\_idf公式

6、逻辑回归的损失函数

7、大卷积核小卷积核区别

8、pooling层为什么能保存特征？会不会丢失特征？

9、BN层原理，为什么能防止过拟合

10、dropout

11、过拟合方法

12、调参调了哪些参数

编程：

1、回文检测

2、矩阵旋转90度不增加内存

## OPPO

1、python编程题：完全数、正则表达式

2、LSTM

3、你在项目中主要负责什么？

4、你的成果？

## 中信

### 技术面

1. 马尔科夫链一阶和二阶指的是什么
2. PCA算法
3. BN作用，里面的超参数是怎么调整的

### 经理面

1、你的优点是什么？

可从对这份工作的**热爱**、**兴趣**爱好入手

数学基础比较扎实，如果会有新的算法，理解掌握的会比较快

对编码敢兴趣，喜欢写代码，没事会去github上找一些源码，分析，自己也会写一下代码玩一玩

**态度方面**：对交代的任务会一定会在规定时间内完成，如果时间充裕会完美的完成，比如代码的优化，性能优化等

2、你最近写了多少行代码？

没统计过，因为代码很多，不同模块，写了很多，具体多少行还真没统计过

3、你们组的分工

项目驱动，来了额项目，先大致分裂，就比如说是回归问题还是分类问题，然后每个人会去试试模型，分析结果然后汇总，决定模型

4、你们的部门是哪个？

人工智能开发部

# 面试题目总结

## 笔试

## 技术

## 性格