# 关于机器学习knn算法及其样例，决策树基本原理

## 机器学习的四大任务

### 监督学习:

**知道目标的工作**

主要由两大任务构成，分类与回归：

1. **预测目标不同**：
   * **回归**的目标是预测连续型的输出变量。例如，预测房价、股票价格或温度等连续数值的问题都可以归类为回归任务。
   * **分类**的目标则是预测离散型的输出变量，通常是将输入数据分到预定义的类别中。例如，判断一封电子邮件是垃圾邮件还是正常邮件、预测图像中的物体类别等问题都属于分类任务。
2. **输出类型**：
   * **回归**输出的是连续的数值或实数，通常可以是任意的数值范围。
   * **分类**输出的是离散的类别标签，通常是预先定义好的有限集合中的一个值。
3. **算法和评估指标**：
   * **回归**问题通常使用的算法包括线性回归、决策树回归、支持向量回归等。评估回归模型的指标可以是均方误差（Mean Squared Error, MSE）、均方根误差（Root Mean Squared Error, RMSE）、决定系数 ( R^2 ) 等，用来衡量预测值与实际值之间的差距。
   * **分类**问题则涉及到算法如逻辑回归、决策树分类、支持向量机分类等。评估分类模型的指标包括准确率（Accuracy）、精确率（Precision）、召回率（Recall）、F1分数等，用来衡量分类模型的预测性能。
4. **应用领域**：
   * **回归**通常在需要预测数值型输出的场景中应用广泛，如金融、经济学、气象预测等。
   * **分类**则适用于需要进行预测类别或标签的场景，如文本分类、图像识别、医学诊断等。

**分类:**

knn-k近邻分类算法（绝大部分任务都可以使用，通常可以作为对比组）

决策树

基于概率的分类：朴素贝叶斯

Logistic回归

支持向量机

利用AdaBoost元算法提高分类性能

**回归：**

线性拟合回归

局部加权线性回归

树回归

### 无监督学习：

聚类

k-均值聚类对未标注数据分组

使用Apriori算法进行关联分析

使用FP-growth算法来高效发现频繁项集

## 研究机器学习算法的流程

**1.收集数据**

**2.准备输入数据**

**3.分析输入数据**

**4.训练算法**

**5.测试算法**

**6.使用算法**

## KNN-k近邻算法

### 模板

核心就是：通过已知数据的特征与标签对选定集合进行预测

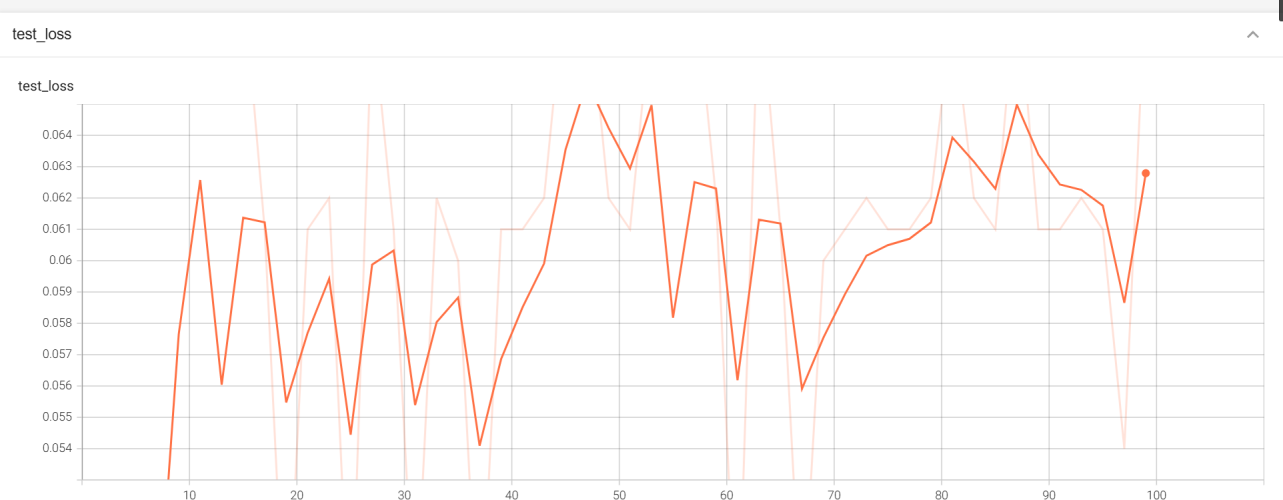
过程(只展示了欧氏距离，其他诸如切比雪夫距离有不同的应用场景)

#伪代码模型展示  
class KNN:  
 def \_\_init\_\_(self, data, labels,n):  
 self.data = data  
 self.labels = labels  
 self.n = n  
 def predict(self, vis, k): # 计算距离  
 data = self.data  
 labels = self.labels  
 n = self.n  
 # 计算距离并排序  
 data=np.int8(data)  
 diffmat = np.tile(vis, (data.shape[0], 1)) - data#计算差值和  
 sqDiffmat = diffmat \*\* 2#计算平方  
 dist = sqDiffmat.sum(axis=1)#开根号  
 ans = dist.argsort()#排序  
  
 # 统计最近邻的标签  
 label\_counts = {}  
 for i in range(k):  
 label = labels[ans[i]]  
 if label in label\_counts:  
 label\_counts[label] += 1  
 else:  
 label\_counts[label] = 1  
  
 # 按照出现次数降序排序  
 sorted\_labels = sorted(label\_counts.items(), key=lambda x: x[1], reverse=True)  
  
 # 返回出现次数最多的标签  
 return sorted\_labels[0][0]

### 样例：利用k近邻算法优化约会匹配网站

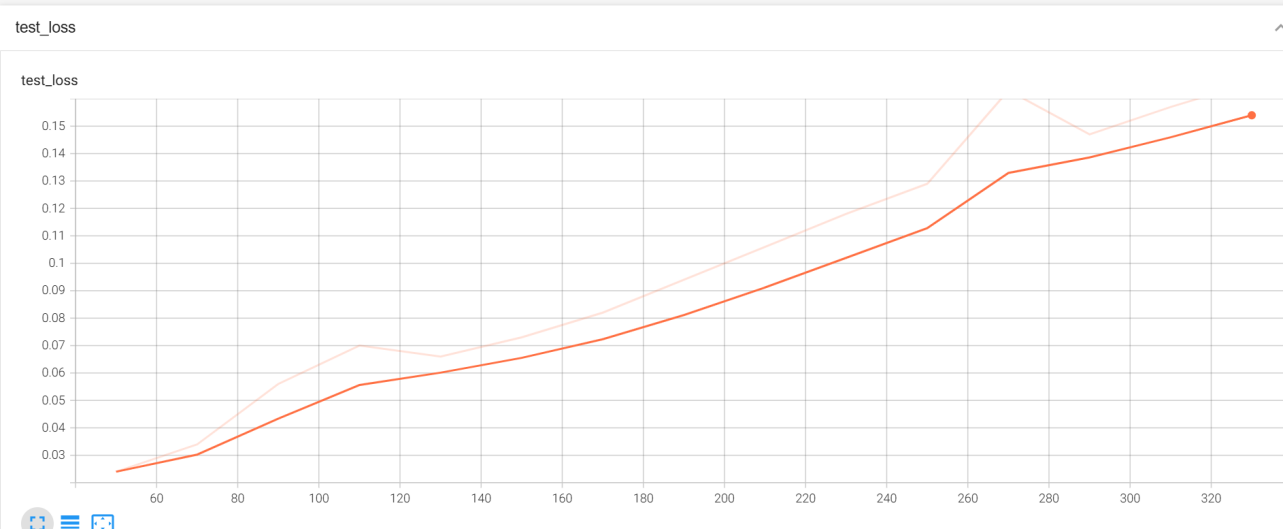
展示tensorboard跑出来的参数调整结果

#### 改变k



横坐标是k的大小，纵坐标是最后的loss，并未看出规律

#### 改变数据集与测试集的数量



横坐标是测试集的数量(由于测试集是从数据集中抽取出的)

可以看出基本随着数据集的减少，误差逐渐增大

### 使用k-近邻算法识别手写数字

## 决策树

优点：树形结构计算复杂度低，输出结果易于理解，对中间值不敏感，可以处理不想关特征数据

缺点：可能出现过拟合

适用数据类型：数值型和标称型

### 数据划分

#### 使用香农(信息)熵评估数据的无序程度(信息增益)

信息定义：

l(xi)=-log2 p(xi) 其中p(xi)代表的是xi事件发生的概率

信息期望(香农熵)公式:

信息增益公式：信息增益越大说明熵(混乱度)的减少量越大，特征划分程度越好

选择一个最好的划分方式也就是信息增益最大的方式(信息增益是熵的减少或者数据无序度的减少)

#计算信息熵(划分数据集的方式----信息增益) 计算给定数据集的香农熵  
def calent(dataset):  
 num=len(dataset)  
 labelres={}#创建关于label数量的字典  
 for feat in dataset:  
 nowlabel=feat[-1]  
 labelres[nowlabel]=labelres.get(nowlabel,0)+1  
 shannon\_ent=0.0  
 for key in labelres: #求以2为底的对数  
 prob=float(labelres[key])/num  
 #prob 是选择该种类的概率 所以是该种的数量除以总数量  
 shannon\_ent-=prob\*log(prob,2)  
 return shannon\_ent  
#划分数据集---按照给定特征划分数据集 & 选择最好的数据集划分方式  
def splitdataset(dataset,axit,val):  
 retdataset=[]  
 for feat in dataset:  
 #抽取数据  
 if feat[axit]== val :  
 reduceFeatVec=feat[:axit]  
 reduceFeatVec.extend(feat[axit+1:])#extend建立新列表  
 retdataset.append(reduceFeatVec)#append塞入新列表  
 return retdataset  
 #选择最好的数据集划分方式  
 def best\_feat(dataset):  
 num=len(dataset[0])-1  
 base=calent(dataset)  
 bests=0.0;bestfeat=-1  
 for i in range(num):  
 fearlist=[example[i] for example in dataset]  
 unval=set(fearlist)  
 news=0.0  
 for val in unval:  
 subdataset=splitdataset(dataset,i,val)  
 prob=len(subdataset)/float(len(dataset))  
 news+=prob\*calent(subdataset)  
 ins=base-news#计算信息增益  
 if ins>bests: #信息增益越大说明熵的减少量越大，划分程度越好  
 bests=ins  
 bestfeat=i  
 return bestfeat

### 递归构造决策树

#递归建立决策树  
#投票表决  
def major(list):  
 count={}  
 for vote in list:  
 if vote not in count.keys():count[vote]=0  
 count[vote]+=1  
 sortcount=sorted(count.items(),key=operator.itemgetter(1),reverse=True)  
 return sortcount[0][0]  
  
def Tree(dataset,labels):  
 list=[example[-1] for example in dataset]  
 if list.count(list[0])==len(list): #类别完全相同是停止递归  
 return list[0]  
 if len(dataset[0])==1:#遍历完所有特征是进行投票表决划分  
 return major(list)  
 bestfeat=best\_model(dataset)#得到列表中包含所有属性值  
 bestfeatlabel=labels[bestfeat]  
 mytree={bestfeatlabel:{}}  
 del(labels[bestfeat])  
 featvals=[example[bestfeat] for example in dataset]  
 uniquevals=set(featvals)  
 for val in uniquevals:  
 sublabels=labels[:]  
 mytree[bestfeatlabel][val]=Tree(splitdataset(dataset,bestfeat,val),sublabels)  
 return mytree

### 可视化：Matplotlib注解绘制树形图

### 测试和存储分类器

使用pickle模块存储决策树

### 样例：使用决策树预测隐形眼镜类型

### 过度匹配问题

**1.减枝**，去掉一些不必要的叶子节点。如果叶子节点智能增加少量信息，则可以删除该节点，讲它并入到其他叶子节点中

**2.随机森林**，建立多颗决策树来进行投票，这样避免了单棵树的匹配选项过多的问题

本章采用的是ID3算法，后续还有CART算法构造决策树

ID3算法可用来划分标称型数据集(离散值)