

**模式识别大作业**

题 目 Otto Group 商品识别

学 院 信息科学与工程学院

专 业 信息与通信工程

组 员 方观寿，桂少军，季思凡，杜嘻嘻，马雪娇

指导教师 赵海涛

**完成日期： 2018 年 12 月15日**

# 摘 要

Otto Group是世界上最大的电子商务公司之一，在全世界范围内，它每天会卖出数百万件商品。每件商品所属的类别分别是Class\_1～ Class\_9。对于这家公司的来说，货物供给和需求分析是非常重要的信息。现给定一些商品的多个特征，设计一个算法模型判断一个商品所属的类别。SVM（support vector machines）支持向量机是一种二分类模型，它的目的是寻找一个超平面来对样本进行分割，分割的原则是间隔最大化，最终转化为一个凸二次规划问题来求解。由简至繁的模型包括：当训练样本线性可分时，通过硬间隔最大化，学习一个线性可分支持向量机；当训练样本近似线性可分时，通过软间隔最大化，也就是加入松弛变量和惩罚因子已达到允许系统错分的情况，学习一个线性支持向量机；当训练样本线性不可分时，通过核技巧和软间隔最大化，学习一个非线性支持向量机，核函数通常会使用高斯核函数。对支持向量的求解通常比较复杂，所以这里使用求解对偶函数的方法来求解原问题的解。

该实验的数据有93个特征，一个类别标签，分为9类，显然是一个多分类问题。根据分析，特征类型是非线性的。按照SVM的解决方法会将数据映射到高维，映射到高维无疑会增加计算的复杂度，而且是指数式增加。回了解决这一问题，如果映射正好是核函数则会改善计算的复杂度。根据核函数的性质，核函数的内积等于内积再做核函数运算。利用这一性质就可以在不增加运算复杂度的情况下对数据进行高维分类。本次数据分类中采用高斯核函数，在该实验中，高斯核函数具有最优的效率。除了svm方法，后面还提供了决策树和神经网络两种方法。

关键词： 多分类，SVM，决策树，随机森林，神经网络

方法一： SVM（支持向量机）

1. **题目简介**

Otto Group是世界上最大的电子商务公司之一，在全世界范围内，它每天会卖出数百万件商品。每件商品所属的类别分别是Class\_1～ Class\_9。对于这家公司的来说，货物供给和需求分析是非常重要的信息。现给定一些商品的多个特征，设计一个算法模型判断一个商品所属的类别。数据中提炼了93个特征向量和商品属于什么类别的标签。要求训练出合适的模型，预测在不同特征情况下商品属于的类别。

本题是一个典型的多分类问题先从train.csv中提取每个商品的多项特征（feature）和类别(label)，使用feature和label进行模型训练。商品的每个特征都是数值型，在使用SVM等分类器时注意将数据范围标准化。题目要求预测商品属于Class\_1～ Class\_9的概率，最后使用训练好的模型预测test.csv中每件商品的类别，要求得出每个商品属于每个类的概率。train.csv文件中包含特征和标签，包含了49502个样本数据，用于训练分类样本；test是测试数据，包含12355个样本，用于检测训练模型的效果。

1. **整体解决方案**
2. **方案设计**

本次作业的操作流程就如下图所示，



由图可知，该系统主要分为数据处理部分以及训练预测模型两部分。本次实验也将围绕这两个部分进行探究。

1. **方案分析**

首先，数据处理方面经过初步分析，没有任何的特征，分布杂乱，根据网上的提示若使用SVM方法进行模型预测时需要将数据范围标准化，也就是对特征的取值范围进行合理的约束。

其次，建立预测模型是使用SVM算法，当然处理的方法还有很多，比如：主流的“logistic regression + L1正则”， Decision Tree决策树。但无论用何算法，本次实验的核心在于通过已有数据特征预测样本的类型。

根据本学期所熟悉的算法，该实验选择了SVM支持向量机方法来预测该模型。SVM的最终判别函数只由少数的支持向量所确定，计算的复杂性取决于支持向量的数目，而不是样本空间的维数，这在某种意义上避免了“维数灾”。少数支持向量决定了最终结果，这不但可以帮助我们抓住关键样本、“剔除”大量冗余样本，而且注定了该方法不但算法简单，而且具有较好的“鲁棒”性。由于有较为严格的统计学习理论，应用SVM方法建立的模型具有较好的推广能力。SVM方法可以给出所建模型的推广能力的确定的界，这是目前其它任何学习方法所不具备的。建立任何一个数据模型，人为的干预越少越客观。与其他方法相比，建立SVM模型所需要的先验干预较少。

1. **数据处理**
   * 1. **数据提取**

首先查看一下原始数据的结构，如下表所示：

|  |  |  |
| --- | --- | --- |
| id | feat\_1,feat\_2,……,feat\_93 | target |
| 1 | 0,0,2,35,……,68,94 | Class\_4 |
| 2 | 0,1,2,6,……,56,1 | Class\_8 |
| 3 | 0,0,0,47,……,0,0 | Class\_6 |
| 4 | 45,0,0,3,……,56,0 | Class\_5 |
| 5 | 0,0,0,1,……,2,3 | Class\_1 |
| 6 | 0,0,1,2,……,3,6 | Class\_2 |
| 7 | 0,1,0,1,……,4,36 | Class\_9 |
| 8 | 2,2,2,1,……,2,45 | Class\_7 |
| 9 | 1,2,0,0,……,2,41 | Class\_3 |

表中的数据只是截取了原始数据中的一部分。

第一列数据表示的是样本编号，不同编号代表不同的样本，这个数据在求解过程中几乎没有使用到，因为使用矩阵保存数据可以严格保证每行用户和数据相对应，所以可以不用使用该数据。

第2列到第94数据是商品的特征信息，商品的分类由这些特征决定。

最后一列是商品的分类，这些商品一共有9个类，这列数据给出了不同商品特征所属的类别。

考虑到时间以及编程复杂度问题，在原始数据表中去掉了那些特征数据超过70个零的样本。

* + 1. **代码实现：**

训练集数据在train.csv文件中，采用feat\_1~feat\_93的数据作为测试数据，Lable就是要求的Y.测试集数据在test.csv文件中。使用pandas库的函数从csv文件中取出是DataFrame数据格式，先对多余70个零值的数据删除。对于测试集的数据应为只有测试数据而没有测试结果，所以在本次实验中只能得出概率数据，验证结果只能提交到网上进行评测。

def load\_dataset():

train\_dataset = pd.read\_csv('train.csv',parse\_dates=True,usecols=range(1,95))

train\_dataset = train\_dataset.dropna(thresh=70)

train\_dataset = train\_dataset.values

train\_set\_x\_orig = train\_dataset[:,0:93]

train\_set\_y\_orig = train\_dataset[:,93:94]

test\_dataset1 = pd.read\_csv('test.csv',parse\_dates=True,usecols=range(1,94))

test\_dataset = test\_dataset.dropna(thresh=70)

test\_dataset\_x\_orig = test\_dataset.values

return train\_set\_x\_orig,train\_set\_y\_orig,test\_set\_x\_orig

* + 1. **数据预处理**

经过观察，本次实验数据不存在空缺的数据，且特征值的变化范围也是在一个合理的范围，所以数据预处理并不存在很多的复杂工作。根据网上的提示，仅对数据进行简单的数据范围标准化即可。实现方法如下：

def normorlize\_data(matrix\_data\_orig):

#将数据集进行归一化处理

scaler = MinMaxScaler( )

scaler.fit(matrix\_data\_orig)

scaler.data\_max\_

return scaler.transform(matrix\_data\_orig)

* + 1. **数据结果处理**

由于该实验没有给出测试集的结果，所以只能将得到每个样本分别数据每个分类的概率，然后按照要求提交该结果到网上进行测试。网上测试采用将使用 multi-class logarithmic loss 作为最后评判标准，公式如下：

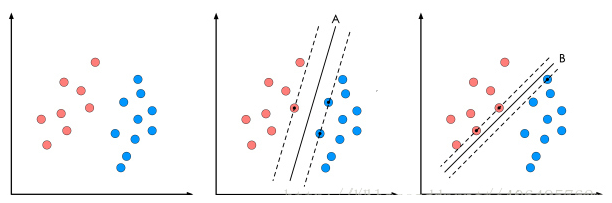


其中*N*代表测试数据集中的商品数量，*log*使用自然对数，*y*​*ij*​​表示商品*i*是否属于*j*，如果是则*y*​*ij*​​=1，否则为0。*p*​*ij*​​代表你预测商品*i*属于*class*​*j*​​的概率。

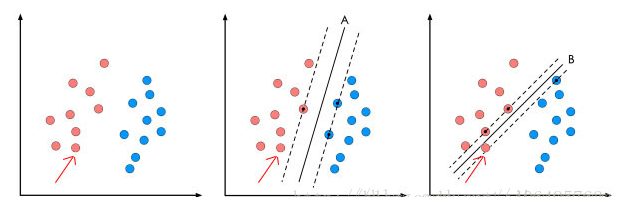
1. **SVM预测模型**

在该实验中使用SVM进行分类，SVM支持向量机（support vector machines）是一种二分类模型，它的目的是寻找一个超平面来对样本进行分割，分割的原则是间隔最大化，最终转化为一个凸二次规划问题来求解。由简至繁的模型包括：当训练样本线性可分时，通过硬间隔最大化，学习一个线性可分支持向量机；当训练样本近似线性可分时，通过软间隔最大化，学习一个线性支持向量机；当训练样本线性不可分时，通过核技巧和软间隔最大化，学习一个非线性支持向量机。

* 1. **线性SVM**
  2. **线性可分的二分类问题**：



上图中红色和蓝色分别表示不同的两个类别，数据为线性可分，但是将其分开的直线不止一条，右边两张图分别给出了不同的方法。黑色实线为“决策面”，每个决策面对应一个线性分类器，两者的性能是有差距的。



决策面不同的情况下，添加一个红色的点，显然中间的图仍然能够很好的分类，但是右边的图已经分类错误了，所以中间的图得决策面优于右图得决策面。

* 1. **选择决策面**

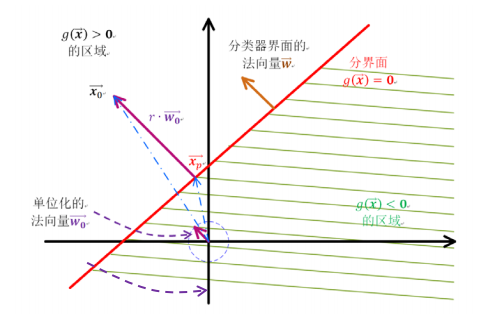
在保证决策面方向不变且不会出现错分样本的情况下移动决策面，会在原来的决策面两侧找到两个极限位置（越过该位置就会产生错分现象），如虚线所示。

虚线的位置由决策面的方向和距离原决策面最近的几个样本的位置决定。而这两条平行虚线正中间的分界线就是在保持当前决策面方向不变的前提下的最优决策面。两条虚线之间的垂直距离就是这个最优决策面对应的分类间隔。显然每一个可能把数据集正确分开的方向都有一个最优决策面（有些方向无论如何移动决策面的位置也不可能将两类样本完全分开），而不同方向的最优决策面的分类间隔通常是不同的，那个具有“最大间隔”的决策面就是SVM要寻找的最优解。

而这个真正的最优解对应的两侧虚线所穿过的样本点，就是SVM中的支持样本点，称为“支持向量”。

学习的目标是在特征空间中找到一个分离超平面，能将实例分到不同的类中。

因此我们希望最后能获得线性分类器，到数据集合得最短距离尽可能大。



图a 分界面外一点到分界面距离得示意图

上图所示得红线，为以分类器，在分类器边界上的样本则有。根据图示，令分界面外一点到分界面得距离为，在分界面上的投影为，分界面的单位法向量为。假设（的情况，讨论是类似的），根据上图有：





因此有：







因为有

则有：



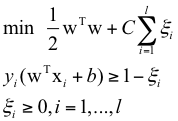
从而可得：



加入绝对值符号是因为与法方向的关系。如的情况，我们会得出，考虑两种情况，有。

* 1. **线性不可分的二分类问题**

线性可分的SVM不具有太多的实用价值，因为现实问题中样本一般都不是线性可分的，接下来我们将它进行扩展，得到能够解决线性不可分问题的模型。为了处理这个问题，当线性不可分时通过加上松弛变量和惩罚因子对错误分类的样本进行惩罚，可以得到如下最优化问题：

****

其中是松弛变量，如果它不为0，表示样本突破了不等式约束条件。C为惩罚因子，是人工设定的大于0的参数，用来对突破了不等式约束条件的样本进行惩罚。可以证明这个问题是凸优化问题，因此可以保证求得全局最优解。

另外，上述问题是满足Slater条件的。如果令

则有

C:\Users\IFREEW~1\AppData\Local\Temp\WeChat Files\7b6f1fb266891aaf63842681872fd2d3.png

不等式条件严格满足，因此强对偶条件成立，原问题和对偶问题有相同的最优解。因此可以转化成对偶问题求解，这样做的原因是原问题的不等式约束太多太复杂，不易于求解。

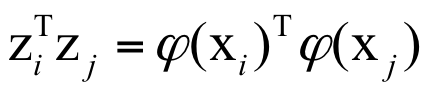
* 1. **非线性分类器(核函数问题)**

虽然加入了松弛变量和惩罚因子，但支持向量机还是一个线性模型，只是允许错分样本的存在，这从它的决策函数也可以看出来。接下来要介绍的核映射使得支持向量机成为非线性模型，决策边界不再是线性的超平面，而可以是形状非常复杂的曲面。

如果样本线性不可分，可以对特征向量进行映射将它转化到一般来说更高维的空间，使得在该空间中是线性可分的，这种方法在机器学习中被称为核技巧。核映射将特征向量变换到另外一个空间：

C:\Users\IFREEW~1\AppData\Local\Temp\WeChat Files\5c424453096154accce2779465372bc8.jpg

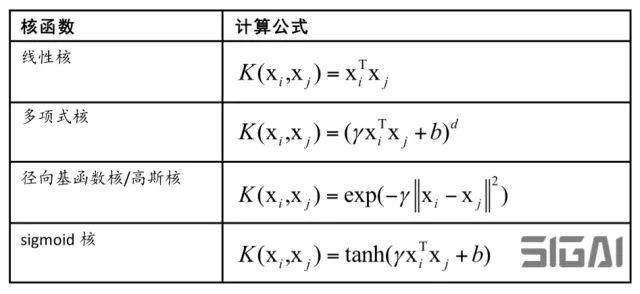
对偶问题计算的是两个样本向量之间的内积，因此映射后的向量在对偶问题中为：



直接计算效率太低，而且不容易构造映射函数。如果映射函数选取得当，能够确保存在函数K，使得下面等式成立：

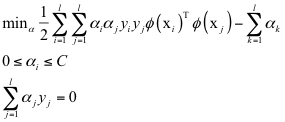
C:\Users\IFREEW~1\AppData\Local\Temp\WeChat Files\3e8add7894317a8c378a6286cb3b7bb3.png

这样只需先对向量做内积然后用函数K进行变换，这等价于先对向量做核映射然后再做内积。在这里我们看到了求解对偶问题的另外一个好处，对偶问题中出现的是样本特征向量之间的内积，而核函数刚好作用于这种内积，替代对特征向量的核映射。满足上面条件的函数称为核函数，常用的核函数有以下几种：

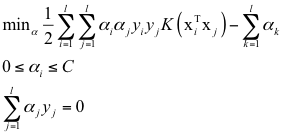


核函数的精妙之处在于不用真的对特征向量做核映射，而是直接对特征向量的内积进行变换，而这种变换却等价于先对特征向量做核映射然后做内积。

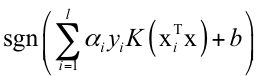
为向量加上核映射后，要求解的最优化问题变为：



根据核函数满足的等式条件，它等价于下面的问题：



最后得到的分类判别函数为：



和不用核映射相比，只是求解的目标函数、最后的判定函数对特征向量的内积做了核函数变换。如果K是一个非线性函数，上面的决策函数则是非线性函数，此时SVM是非线性模型。当训练样本很多、支持向量的个数很大的时候，预测时的速度是一个问题，因此很多时候我们会使用线性支持向量机。

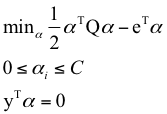
如果我们定义矩阵Q，其元素为：

C:\Users\IFREEW~1\AppData\Local\Temp\WeChat Files\e185d05606fc118976c8d010fd70f6a3.jpg

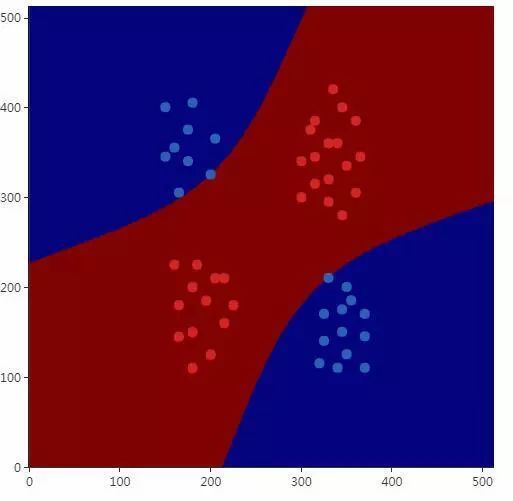
同时定义矩阵K，其元素为：

C:\Users\IFREEW~1\AppData\Local\Temp\WeChat Files\137b48023b201914db5b420e1b6c9355.jpg

对偶问题可以写成矩阵和向量形式：



可以证明，这个对偶问题同样是凸优化问题，这是由核函数的性质保证的。下图是使用高斯核的SVM对异或问题的分类结果：



只要参数设置得当，使用高斯核的支持向量机确实能解决非线性分类问题，分类边界可以是非常复杂的曲线。

* 1. **算法实现**

1. **加载数据**

from load\_data import load\_dataset

train\_set\_x\_orig , train\_set\_y , test\_set\_x\_orig , test\_set\_y = load\_dataset()

1. **训练模型**
   * 1. **导入数据包**

from time import time

from sklearn.svm import SVC

from loaddatas import load\_dataset

from sklearn import svm

from sklearn.externals import joblib

本次实验主要用了sklearn中的SVC和svm函数，time函数用来计算训练模型所需花费的时间，joblib函数仅仅是用来保存训练模型，方便于其他数据集测试。

* + 1. **创建模型**

#SVM参数设置

n\_classes = 9#表明有样本分类为9类

print("Fiting the classifier to the training set")

t0 = time()#计算时间

param\_grid = {'c':[100000],'gamma':[0.00025,0.00001,0.00005,0.0001,0.0005,0.001,0.005,0.01,0.1],}

#构建SVM模型

clf = svm.SVC(C=0.8, kernel='rbf', gamma=20, decision\_function\_shape='ovr',probability=True)

构建模型仅仅一个函数就足够了，这得益于python高度的函数集成，在sklearn库中不仅包含了支持向量机的训练模型，还有随机森林等训练模型。C=0.8是松弛变量；kernel是选择了高斯(rbf)核函数。Probablity=True，该参数默认为false，但该实验要求输出概率，所以这里改为了true，这也大大增加了训练的时间。

* + 1. **训练模型**

clf = clf.fit(X\_train,Y\_train)

这个函数，输入X\_train,Y\_train。分别表示训练的特征和训练的标签。因为函数给出了训练的模型，这里只需要简单的输入就搭建好模型，可以进行训练了。训练结果储存在clf变量中。

1. **得出结果**

pred\_label = clf.predict\_proba(X\_test)

joblib.dump(pred\_label,"pred\_label.m")

np.savetxt('submission.csv',pred\_label,delimiter=',')

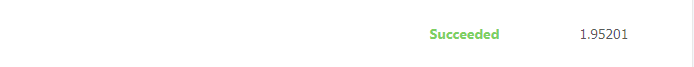
第一行是使用训练好的结果clf，得到所需的概率输出，clf.predict\_proba就可以得到每个样本分别属于每个类的概率。函数joblib.dump是用来存储模型训练结果的，该结果可用于其他数据进行测试，而不需要每次进行训练然后进行预测。np.savetext函数则是将概率输出结果保存为.csv文件。

1. **测试结果和结果分析**

因为test文件中没有给出测试集的结果，所以没法在本地得出模型的准确率，将得到的概率文件上传到网上进行自动评测。下图是使用SVM构建的模型，训练了5个小时，最终得出的结果：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | Class\_1 | Class\_2 | Class\_3 | Class\_4 | Class\_5 | Class\_6 | Class\_7 | Class\_8 | Class\_9 |
| 49503 | 0.030512 | 0.259652 | 0.129278 | 0.04356 | 0.043734 | 0.228736 | 0.045977 | 0.137756 | 0.080795 |
| 49504 | 0.030512 | 0.259652 | 0.129278 | 0.04356 | 0.043734 | 0.228736 | 0.045977 | 0.137756 | 0.080795 |
| 49505 | 0.030512 | 0.259652 | 0.129278 | 0.04356 | 0.043734 | 0.228736 | 0.045977 | 0.137756 | 0.080795 |
| 49506 | 0.030512 | 0.259652 | 0.129278 | 0.04356 | 0.043734 | 0.228736 | 0.045977 | 0.137756 | 0.080795 |
| 49507 | 0.030512 | 0.259652 | 0.129278 | 0.04356 | 0.043734 | 0.228736 | 0.045977 | 0.137756 | 0.080795 |
| 49508 | 0.030512 | 0.259652 | 0.129278 | 0.04356 | 0.043734 | 0.228736 | 0.045977 | 0.137756 | 0.080795 |
| 49509 | 0.030512 | 0.259652 | 0.129278 | 0.04356 | 0.043734 | 0.228736 | 0.045977 | 0.137756 | 0.080795 |
| 49510 | 0.030512 | 0.259652 | 0.129278 | 0.04356 | 0.043734 | 0.228736 | 0.045977 | 0.137756 | 0.080795 |
| 49511 | 0.030512 | 0.259652 | 0.129278 | 0.04356 | 0.043734 | 0.228736 | 0.045977 | 0.137756 | 0.080795 |
| 49512 | 0.030512 | 0.259652 | 0.129278 | 0.04356 | 0.043734 | 0.228736 | 0.045977 | 0.137756 | 0.080795 |

这是部分概率结果，使用SVM不仅花费时间长，得到的结果还不好，这对该实验来说显然不是最好的解决方法。就算调整了参数，得到的结果还是差不多。这和SVM不适合大样本分类有很大的关系，得到的损失结果如下图：



最终的结果是1.95201，这显然是很差的结果，经过调参，数据预处理等处理，结果还是差不多。于是试着使用随机森林进行训练，仅仅花了30s的训练时间，使用了300课随机树得到的结果如下图：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | Class\_1 | Class\_2 | Class\_3 | Class\_4 | Class\_5 | Class\_6 | Class\_7 | Class\_8 | Class\_9 |
| 49503 | 0.003333 | 0 | 0.003333 | 0 | 0 | 0.99 | 0 | 0 | 0.003333 |
| 49504 | 0 | 0.003333 | 0.003333 | 0.003333 | 0 | 0.01 | 0.003333 | 0.973333 | 0.003333 |
| 49505 | 0.436667 | 0.01 | 0.006667 | 0.003333 | 0.033333 | 0.113333 | 0.113333 | 0.18 | 0.103333 |
| 49506 | 0.013333 | 0.126667 | 0.073333 | 0.026667 | 0.006667 | 0.16 | 0.06 | 0.5 | 0.033333 |
| 49507 | 0.01 | 0.463333 | 0.46 | 0.003333 | 0.043333 | 0.003333 | 0.006667 | 0.006667 | 0.003333 |
| 49508 | 0.003333 | 0.263333 | 0.703333 | 0.013333 | 0 | 0.01 | 0.003333 | 0.003333 | 0 |
| 49509 | 0.003333 | 0.003333 | 0 | 0 | 0 | 0.973333 | 0.01 | 0.006667 | 0.003333 |
| 49510 | 0 | 0.103333 | 0.796667 | 0.1 | 0 | 0 | 0 | 0 | 0 |
| 49511 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 49512 | 0.103333 | 0.06 | 0.006667 | 0.023333 | 0.006667 | 0.57 | 0.036667 | 0.126667 | 0.066667 |
| 49513 | 0.026667 | 0.306667 | 0.346667 | 0.206667 | 0.003333 | 0.03 | 0.056667 | 0.02 | 0.003333 |
| 49514 | 0.013333 | 0.003333 | 0.006667 | 0.003333 | 0.003333 | 0.036667 | 0.023333 | 0.893333 | 0.016667 |
| 49515 | 0.016667 | 0.02 | 0.013333 | 0.04 | 0.776667 | 0.09 | 0.023333 | 0.016667 | 0.003333 |
| 49516 | 0.003333 | 0.9 | 0.083333 | 0.013333 | 0 | 0 | 0 | 0 | 0 |
| 49517 | 0.02 | 0.006667 | 0.003333 | 0.003333 | 0 | 0.886667 | 0.036667 | 0.043333 | 0 |
| 49518 | 0.146667 | 0.016667 | 0.01 | 0.003333 | 0.003333 | 0.12 | 0.096667 | 0.5 | 0.103333 |

上图是使用随机森林得到的结果，采用了300课随机树，得到的结果看起来更为合理一点，测试准确率如下图：



结果一下子到了0.58855，这是一个挺不错的成绩了，由于是接近50000个样本，数据量比较庞大，使用SVM不仅花费时间多，还得不到理想的结果，虽然在本次实验SVM不是最好的解决方案，但SVM在分类问题上还是发挥着很大的作用。

方法二：决策树与随机森林

一 决策树

1 概述

决策树(decision tree)是一种基本的分类与回归方法。决策树模型呈树形结构，在分类问题中，表示基于特征对实例进行分类的过程。它可以认为是if-then规则的集合，也可以认为是定义在特征空间与类空间上的条件概率分布。其主要优点是模型具有可读性，分类速度快。学习时，利用训练数据，根据损失函数最小化的原则建立决策树模型。预测时，对新的数据，利用决策树模型进行分类。决策树学习通常包括3个步骤：特征选择、决策树的生成和决策树的修剪。

2 简介

第一步：确定一个分裂属性（即以样本数据的哪个特征进行划分）。

此处确定最优划分特征的方法是整个决策树的关键部分。最优划分特征的选择基于一个目标：使得分裂后各个节点数据的“纯度”最高。即尽量使得通过该特征进行分类后的分支节点所包含的样本属于同一类别。选择一个合适的特征作为判断节点，可以快速的分类，减少决策树的深度。

2.1 信息增益

给出一个信息熵的定义，假设样本用D表示,



其中pi表示第i个类别在整个训练元组中出现的概率，可以用属于此类别元素的数量除以训练元组元素总数量作为估计。M为类别数，上例中是、否有能力偿还贷款，m=2。熵表示样本的混乱程度，样本数据越无序，越混乱，熵就越大。可以考虑通过比较划分特征前后样本数据熵的变化，来确定样本的纯度变化。

定义信息增益:



其中，a代表划分的特征，特征a有V个可能的取值（a1,a2,…aV），D中在特征a^(v)上取值为的所有样本定义为D^(v)，|D|表示D中样本个数。可以认为，信息增益越大，则意味着以特征a来进行划分，所获得的“纯度提升越大”。因此可以遍历所有特征，选取使得信息增益最大的特征作为当前结点的分裂特征。需要知道的是信息增益准则对可取值数目较多的特征有所偏好，如果将表1中的ID列也作为特征，可以计算其信息增益，是所有特征里面最大的，因为其将原始样本分成10个分支，且每个分支都只有一个样本，纯度自然是最高的，但这并没有泛化能力。

2.2 增益率

增益率是在信息增益偏向多取值特征的特性上做出的改进，减少了该偏向可能带来的不利影响。具体定义为：

 ，



该公式利用IV(a)，表示特征a的一个特性，特征的取值数目越多，则IV的值通常会越大，可以达到消除多取值特征带来的不利影响。

2.3基尼指数

基尼指数用于度量数据集D的纯度:



直观来说，Gini(D)反映了从数据集D随机选取两个样本，其类别标记不一致的概率，则基尼系数越小，代表其纯度越高。（与熵类似）选取合适的纯度量化方式，可以从当前样本数据中找到最好的划分特征，从而将数据集划分为若干个分支。

第二步：观察划分的各个分支，如果分支中样本数据均属于同一类别，则该分支应为叶节点，无需再进行计算；如果分支中样本所有特征都相同，无法再继续分解下去，那么当前分支就为叶节点，类别标记为当前分支中样本数最多的一种（多数表决）；如果以上均不符合，应针对每一组样本数据重复第一步的过程，将分支继续递归分解下去，直至每个分支的样本数据都具有相同的类别。

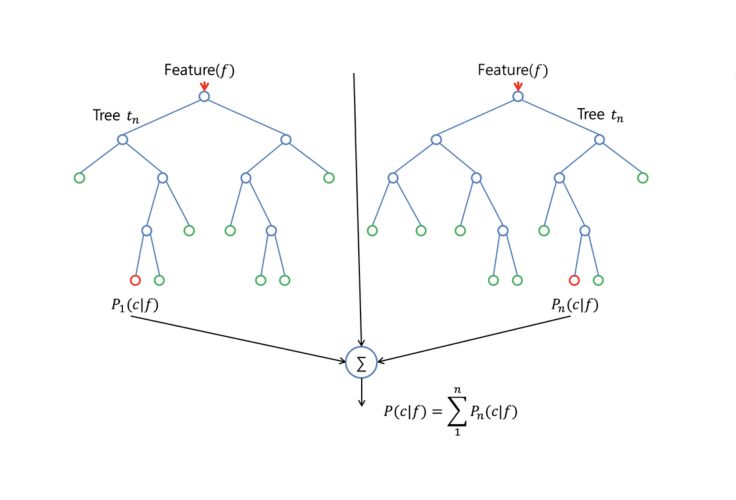
剪枝处理：剪枝操作是为了对付决策树学习算法中“过拟合”的情况，由于决策树算法会不断的重复特征的划分过程，或者由于噪声数据的存在，有时候会使得决策树分支过多，造成过拟合的情况，即对训练数据的分类很准确，但是对未知的测试数据的分类确没那么准确。这种情况下可以采用主动去掉分支的方法来降低过拟合的风险。一般存在“预剪枝”和“后剪枝”两种策略。

预剪枝即为在决策树生成过程中，对当前节点的划分结果进行评价，如果该划分不能带来决策树泛化能力（即处理未见过示例的能力）的提升，则停止划分，将当前结点标记为叶节点；后剪枝则是先生成一棵完整的决策树，然后自底向上的对非叶节点进行评价，如果剪掉该枝可以使得泛化性能提升，则将该子树替换为叶节点。预先剪枝可能过早的终止决策树的生长，后剪枝一般能够产生更好的效果。但后剪枝在子树被剪掉后，决策树生长的一部分计算就被浪费了。

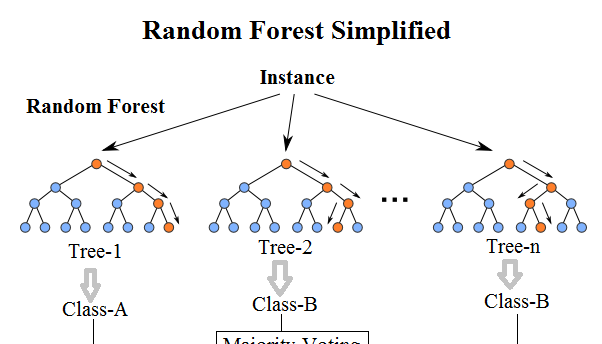
二 随机森林

随机森林是一种有监督学习算法。 就像你所看到的它的名字一样，它创建了一个森林，并使它拥有某种方式随机性。 所构建的“森林”是决策树的集成，大部分时候都是用“bagging”方法训练的。 bagging方法，即bootstrap aggrega[ti](http://www.elecfans.com/tags/%E5%BE%B7%E5%B7%9E%E4%BB%AA%E5%99%A8/" \t "/Users/victor-guix/_blank)ng，采用的是随机有放回的选择训练数据然后构造分类器，最后组合学习到的模型来增加整体的效果。

简而言之：随机森林建立了多个决策树，并将它们合并在一起以获得更准确和稳定的预测。随机森林的一大优势在于它既可用于分类，也可用于回归问题，这两类问题恰好构成了当前的大多数机器学习系统所需要面对的。 接下来，将探讨随机森林如何用于分类问题，因为分类有时被认为是机器学习的基石。 下图，你可以看到两棵树的随机森林是什么样子的：



除了少数例外，随机森林分类器使用所有的决策树分类器以及bagging 分类器的超参数来控制整体结构。 与其先构建bagging分类器，并将其传递给决策树分类器，您可以直接使用随机森林分类器类，这样对于决策树而言，更加方便和优化。要注意的是，回归问题同样有一个随机森林回归器与之相对应。



随机森林算法中树的增长会给模型带来额外的随机性。与决策树不同的是，每个节点被分割成最小化误差的最佳特征，在随机森林中我们选择随机选择的特征来构建最佳分割。因此，当您在随机森林中，仅考虑用于分割节点的随机子集，甚至可以通过在每个特征上使用随机阈值来使树更加随机，而不是如正常的决策树一样搜索最佳阈值。这个过程产生了广泛的多样性，通常可以得到更好的模型。

随机森林算法的另一个优点是可以很容易地测量每个特征对预测的相对重要性。 Sklearn为此提供了一个很好的工具，它通过查看使用该特征减少了森林中所有树多少的不纯度，来衡量特征的重要性。它在训练后自动计算每个特征的得分，并对结果进行标准化，以使所有特征的重要性总和等于1。

在决策树中，每个内部节点代表对一类属性的“测试”（例如，抛硬币的结果是正面还是反面)，每个分支代表测试的结果，每个叶节点代表一个类标签（在计算所有属性之后作出的决定）。叶子就是没有下一分支的节点。

通过查看特征的重要性，您可以知道哪些特征对预测过程没有足够贡献或没有贡献，从而决定是否丢弃它们。这是十分重要的，因为一般而言机器学习拥有的特征越多，模型就越有可能过拟合，反之亦然。

三 决策树与随机森林的区别

随机森林是决策树的集合，但仍有一些区别。如果您将带有特征和标签的训练数据集输入到决策树中，它将制定一些规则集，用于预测。

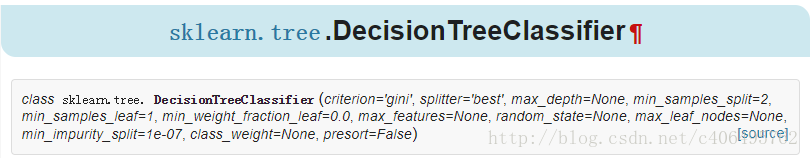
例如，如果您想预测某人是否会点击在线广告，可以收集该广告的过去点击人员以及能够描述其做决定的特征。一旦你将这些特征和标签放入决策树中，它会生成节点和一些规则，然后你就可以预测广告是否会被点击。但决策树通常通过计算信息增益和基尼指数来生成节点和规则时，相比之下，随机森林则是随机的。

另一个区别是“深度”决策树往往会遭遇过拟合问题。而随机森林则可以通过创建随机的特征子集并使用这些子集构建较小的树，随后组成子树，这种方法可以防止大部分情况的过拟合。要注意的是，这同时会使得计算速度变慢，并取决于随机森林构建的树数。

四 实验过程与结果

对于Otto\_Group数据集，包括训练集：49502个样本，93个数字特征，测试集：12376个样本，93个数字特征，通过这些特征讲商品分为9个类。用sklearn包对数据进行处理。

**4.1决策树实验**



4.1.1对sklearn.tree.DecisionTreeClassifier函数进行参数解读

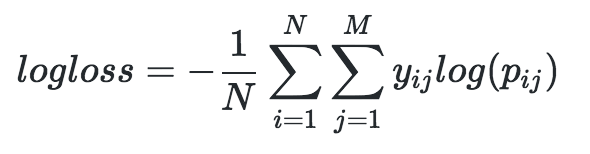
|  |
| --- |
| （1）criterion：特征选择，可选参数，默认是gini，可以设置为entropy。ID3使用的是entropy，CART使用的则是gini。  （2）max\_features：划分时考虑的最大特征数，可选参数，默认是None。  （3）max\_depth：决策树最大深，可选参数，默认是None。数据少或者特征少的时候可以不管这个值。如果模型样本量多，特征也多的情况下，推荐限制这个最大深度，具体的取值取决于数据的分布。常用的可以取值10-100之间。  （4）max\_leaf\_nodes：最大叶子节点数，可选参数，默认是None。通过限制最大叶子节点数，可以防止过拟合。如果特征不多，可以不考虑这个值，但是如果特征非常多的话，可以加以限制，具体的值可以通过交叉验证得到。  （5）min\_samples\_split：内部节点再划分所需最小样本数，可选参数，默认是2。这个值限制了子树继续划分的条件。如果样本量不大，不需要管这个值。如果样本量数量级非常大，则推荐增大这个值。 |

4.1.2经过实验，最终选择如下的参数的实验结果较好

|  |
| --- |
| Clf=tree.DecisionTreeClassifier(max\_depth=45,max\_features=45,min\_samples\_split=30,max\_leaf\_nodes=45) |

4.1.3结果

把结果上传到lincode上，结果如下，



其中NN代表测试数据集中的商品数量，log使用自然对数，表示商品i是否属于j，如果是则=1，否则为0。​​代表你预测商品i属于class​j​​的概率。



loss越小说明测试结果越好，kaggle上最高成绩是0.38左右，决策树做的效果并不是很好，因为这几个参数都很难找到最佳的，所以接下来使用随机森林。

**4.2 随机森林实验**

4.2.1对RandomForestClassifier参数进行解读

n\_estimators :森林里（决策）树的数目。

criterion : 衡量分裂质量的性能（函数）。 受支持的标准是基尼不纯度的"gini",和信息增益的"entropy"（熵）。可选择(默认值为“gini”)。

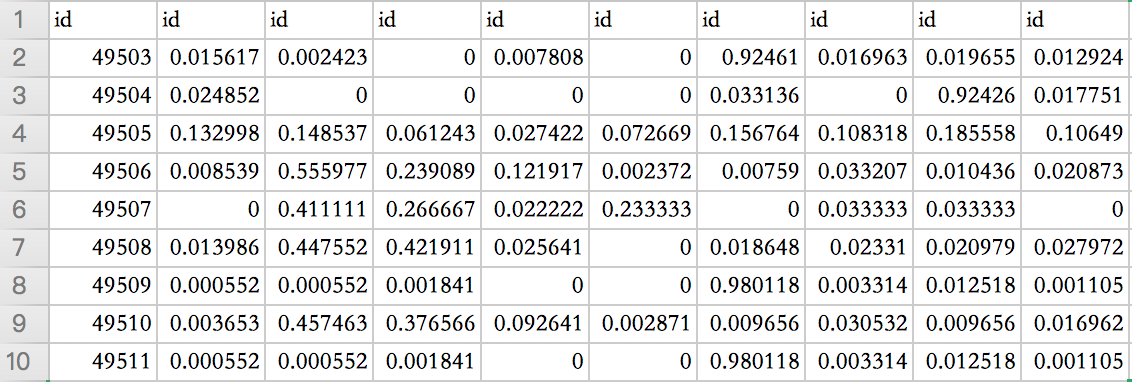
n\_estimators:用于拟合和预测的并行运行的工作（作业）数量。如果值为-1，那么工作数量被设置为核的数量。

其他的与决策树的类似

4.2.2 经过实验，当n\_estimators=300时，效果最好

|  |
| --- |
| clf = RandomForestClassifier(n\_estimators=300, n\_jobs=-1) |

4.2.3 预测概率如下



4.2.4 lintcode上的结果如下

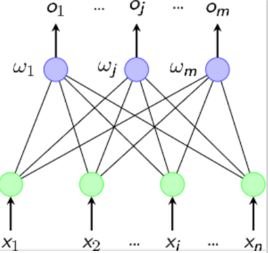


**方法三：BP神经网络**

1.概述

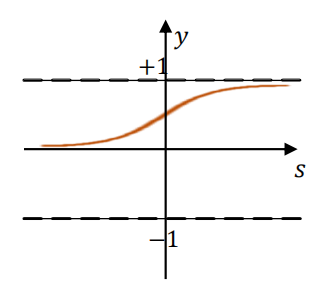
神经网络是由具有适应性的简单单元组成广泛互连的网络，它的组织能够模拟生物神经系统对真实世界物体所做出的交互反应。神经网络中最基本的成分是神经元模型，也就是上述定义上的“简单单元”。在生物神经网络当中，每个神经元都与其他神经元相连，当它“兴奋时”，就会向相连的神经元发送化学物质，从而改变这些神经元内的电位，而如果一个电位超过了阈值，就会使它被激活，即“兴奋”，像其它神经元发送化学物质。

将BP神经网络抽象简化成如下图（a）模型，第一层是输入层，第二层是隐含层，第三层是输出层，在这个模型当中，神经元接受n个其他神经元传过来的输入信号，输入信号经过带权重的连接传递，还要通过激活函数处理以产生神经元的输出。



（a）

激活函数是理想中的阶跃函数，将输入值映射为“0”或“1”，但由于阶跃函数不连续不光滑的性质，所有用sigmoid函数代替激活函数，模型如下图（b）：



(b)

把许多神经元按一定层次连接起来，就构成了整个神经网络。BP神经网络的计算过程由正向计算过程和反向计算过程组成。正向传播过程，输入模式从输入层经隐单元层逐层处理，并转向输出层，每一层神经元的状态只影响下一层神经元的状态。如果在输出层不能得到期望的输出，则转入反向传播，将误差信号沿原来的连接通路返回，通过修改各神经元的权值，使得误差信号最小。

1. 实验过程

2.1 原理图

由于输入的是93维数据，要转换成9类的分布概率，这里就选取了93->128->64->32->16->9的神经网络，使用relu激活函数，最后一层用softmax将预测值转化为每个分类的概率。

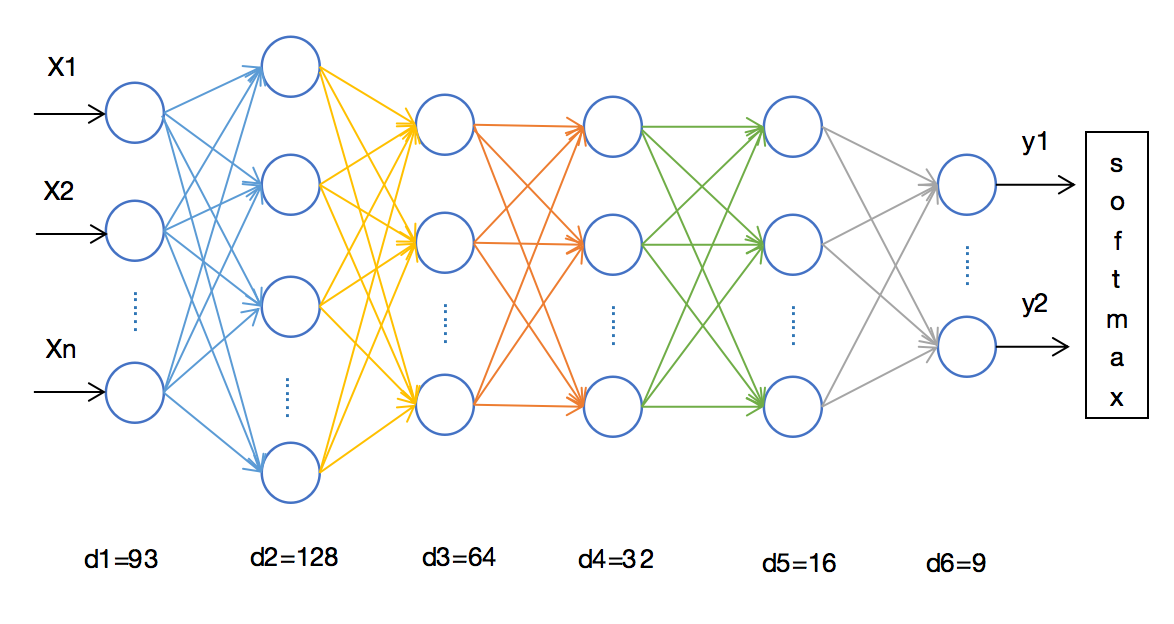
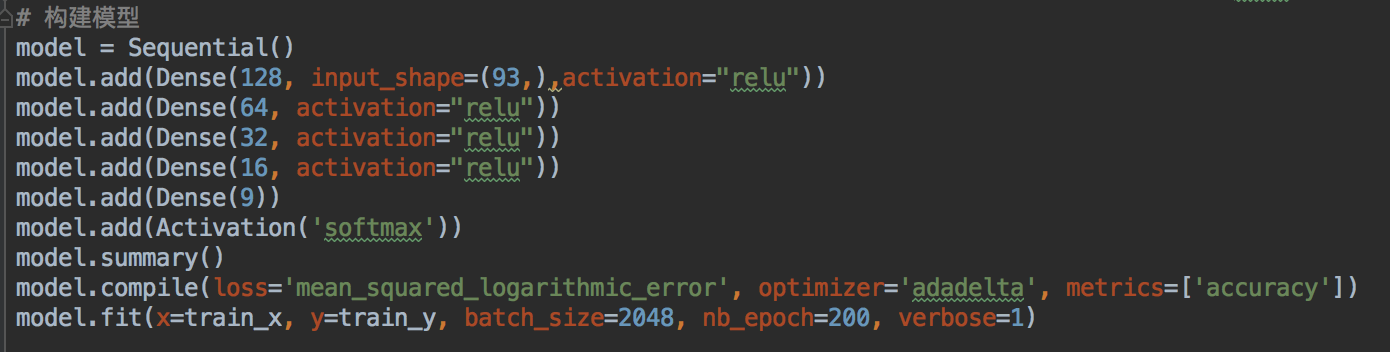


图3 6层神经网络结构图

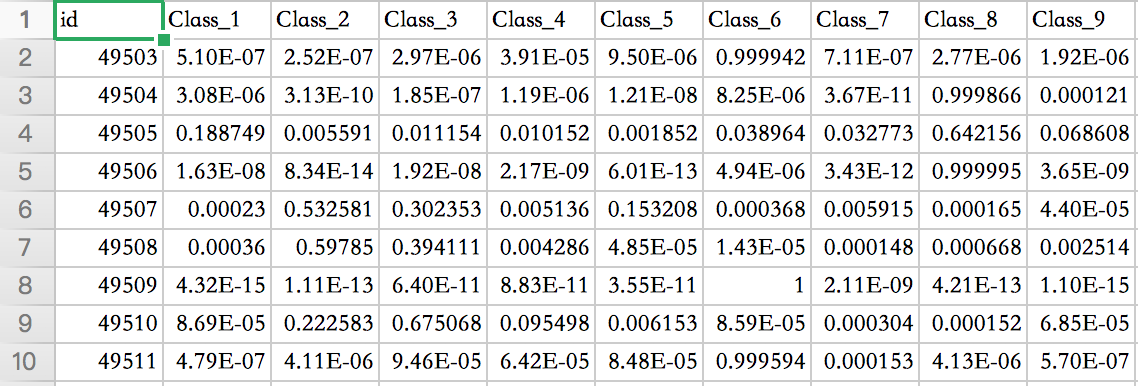
2.2 构建模型



loss采用公式为：

(log(clip(y\_pred,epsilon,infinite)+1)-log(clip(y\_true,epsilon,infinite)+1.))^2.mean(axis=-1)，这个就是加入了log对数，剔除不介于epsilon和infinite之间的预测值与实际值之后，然后取对数，作差，平方，累加求均值。

2.3预测概率结果



上传到lintcode的结果如下



**总 结**

经过几个星期的琢磨，一边学习，一遍查看别人的论文和实现方法，这个实验的模型总算还是建立起来了，但针对这个商品分类的问题却并没有得到很好的解决。本实验主要由两部分组成，一部分是数据处理，另一部分是训练模型。实验的难度主要在SVM原理的理解上，虽然python提供了SVM的训练模型，不需要自己去搭建，但该实验目的还是要了解其中的工作原理。在我的实验中，我认为对偶,KKT条件和核函数是SVM的难点也它的核心。因为满足KKT条件，所以原问题的解和对偶函数的解是可以一样的，核函数的作用就是将数据映射到高维进行分类，但又可以避免“维数灾难”。

SVM本是处理线性二分类问题，加入松弛变量和惩罚因子则允许存在错分情况。这也就到了线性不可分的情况，然后要求解线性不可分情况，若是直接求解基本上不可能实现，所以引入对偶问题的求解。当满足KKT条件之后，就可以求解对偶问题以得到原问题的解。光是得到线性不可分的模型和求解还不能解决生活中的问题，因为通常要求分类的情况都不是线性可分，基本上是非线性的。为了方便分类，想到将数据映射到高维进行分类。但这时会存在高维灾难，求解非常苦难，核函数的应用很好的解决了这个问题，不仅将数据映射到高维分类，还可以避免维数灾难。

SVM给出了一个分类问题很好的解决方式，使用至今已有20年，刚实用的时候引起分类界轰动，因其独有的准确度独领风骚。虽然现在的样本越来越大，使用SVM进行分类越来越感到吃力，但不可否定SVM在模式识别发展中的作用。通过本次实验，不仅让我对SVM有了深刻的理解，而且对模式识别也有了更深的认识。感谢老师的提点，也感谢同学的对我的无私帮助。

　随机森林的主要优点有：训练可以高度并行化，对于大数据时代的大样本训练速度有优势。个人觉得这是的最主要的优点。由于可以随机选择决策树节点划分特征，这样在样本特征维度很高的时候，仍然能高效的训练模型。在训练后，可以给出各个特征对于输出的重要性 由于采用了随机采样，训练出的模型的方差小，泛化能力强。对部分特征缺失不敏感。主要缺点有：在某些噪音比较大的样本集上，RF模型容易陷入过拟合。取值划分比较多的特征容易对RF的决策产生更大的影响，从而影响拟合的模型的效果。

**成员分工**

**代码编写，报告撰写：方观寿（Y30180616），桂少军（Y30180618）**

**报告撰写：季思凡（Y30180619），马雪娇（Y30180623），杜嘻嘻（Y30180615）**