1. **实验环境**

PyCharm-162.1237.1

TextMate , python 2,7

1. **实验题目**
2. 用不同算法进行集成学习
3. **实验数据**

数据是位于“代码”文件夹下的Data-Ass2.mat，是3000\*3的数据，前两维是数据特征，后一维是数据类别

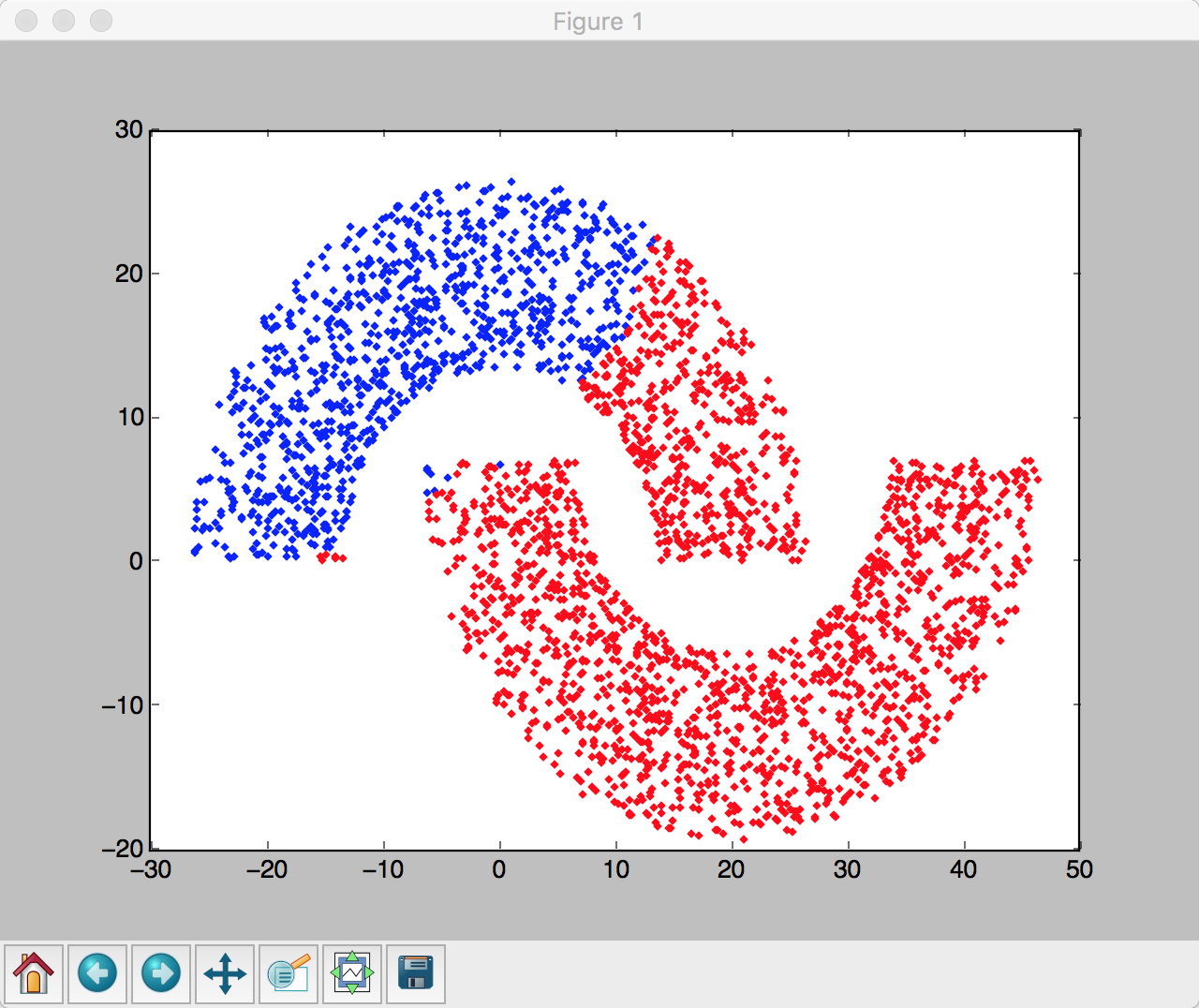
1. **实验思想**

把多个决策准确率在50%以上的决策器聚集起来共同投票进行一项决策，其中准确率越高的投票的权重越大，显然在一般情况下可以提高决策的准确性。

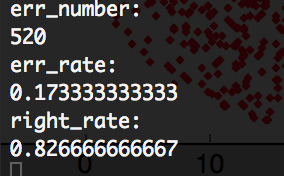
1. **实验过程**

我用了Knn,BP神经网络以及SVM三种算法

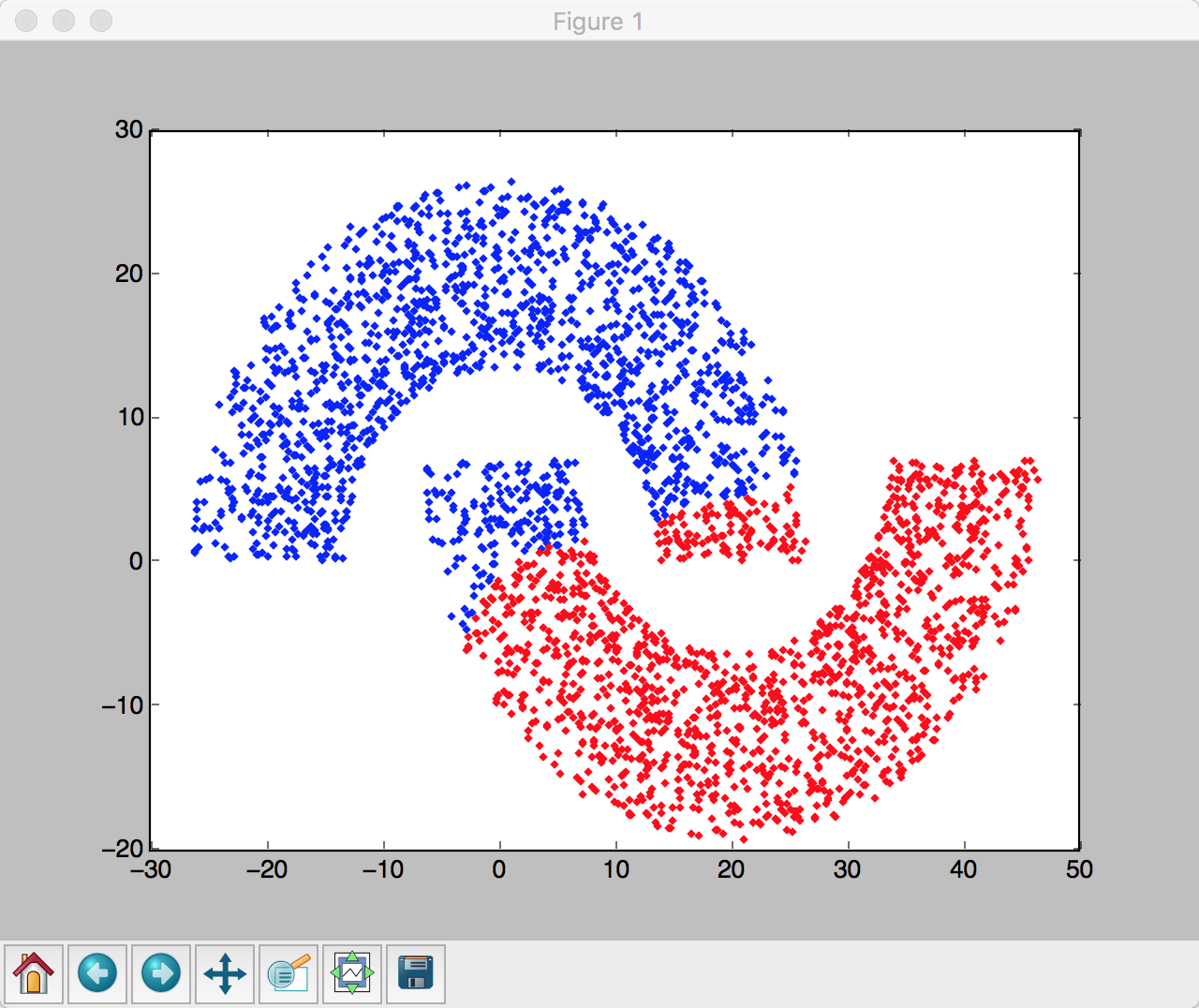
* 1. 单独使用Knn的时候



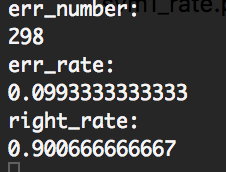
这里我故意把KNN的训练集弄得比较小，我只用了前11个点来训练它，使得算法有一定误差。可以看到它的错误点有520个，正确率约为82.6%



* 1. 单独使用BP神经网络的时候

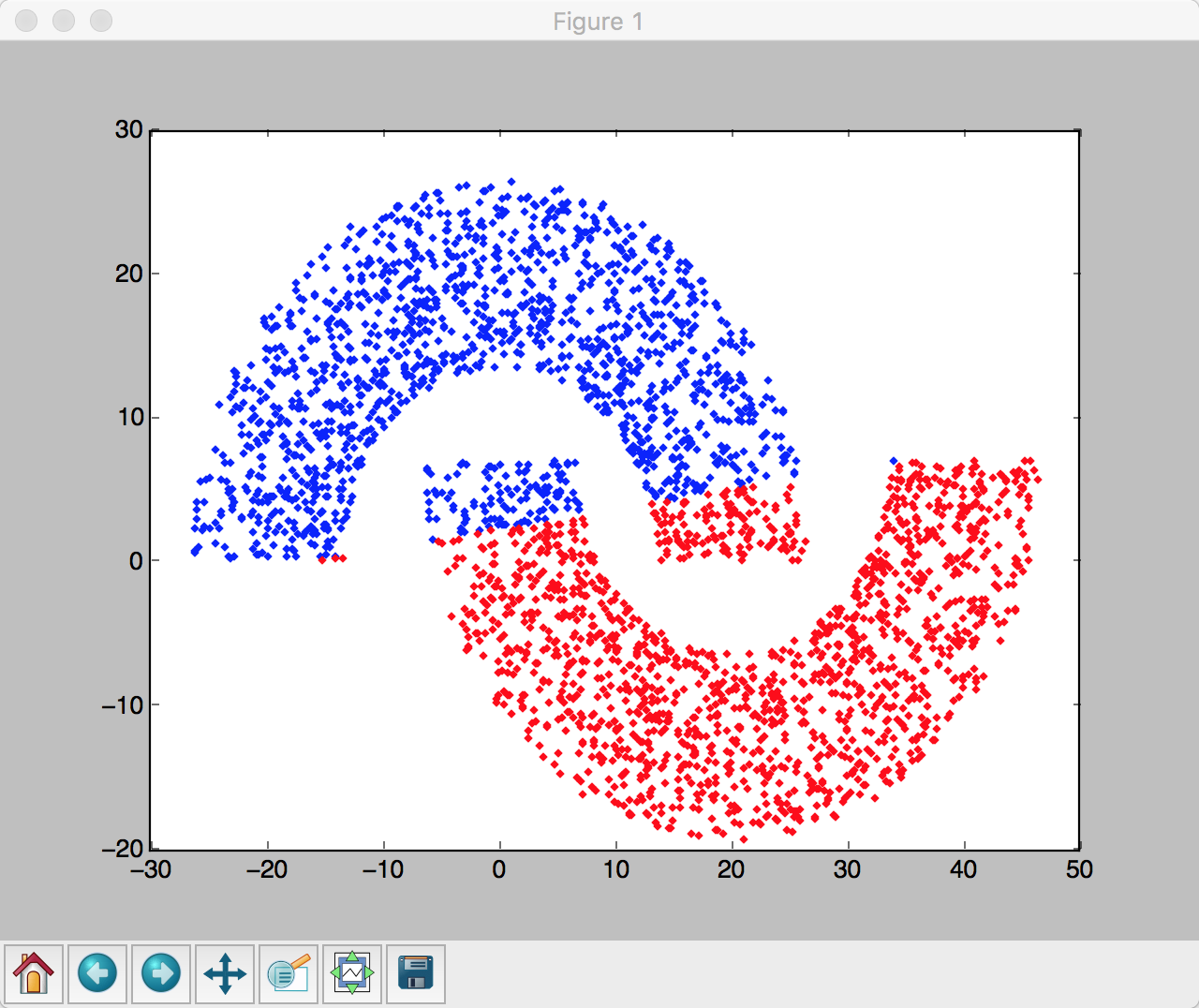
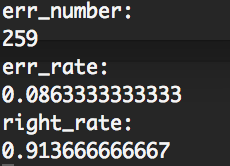


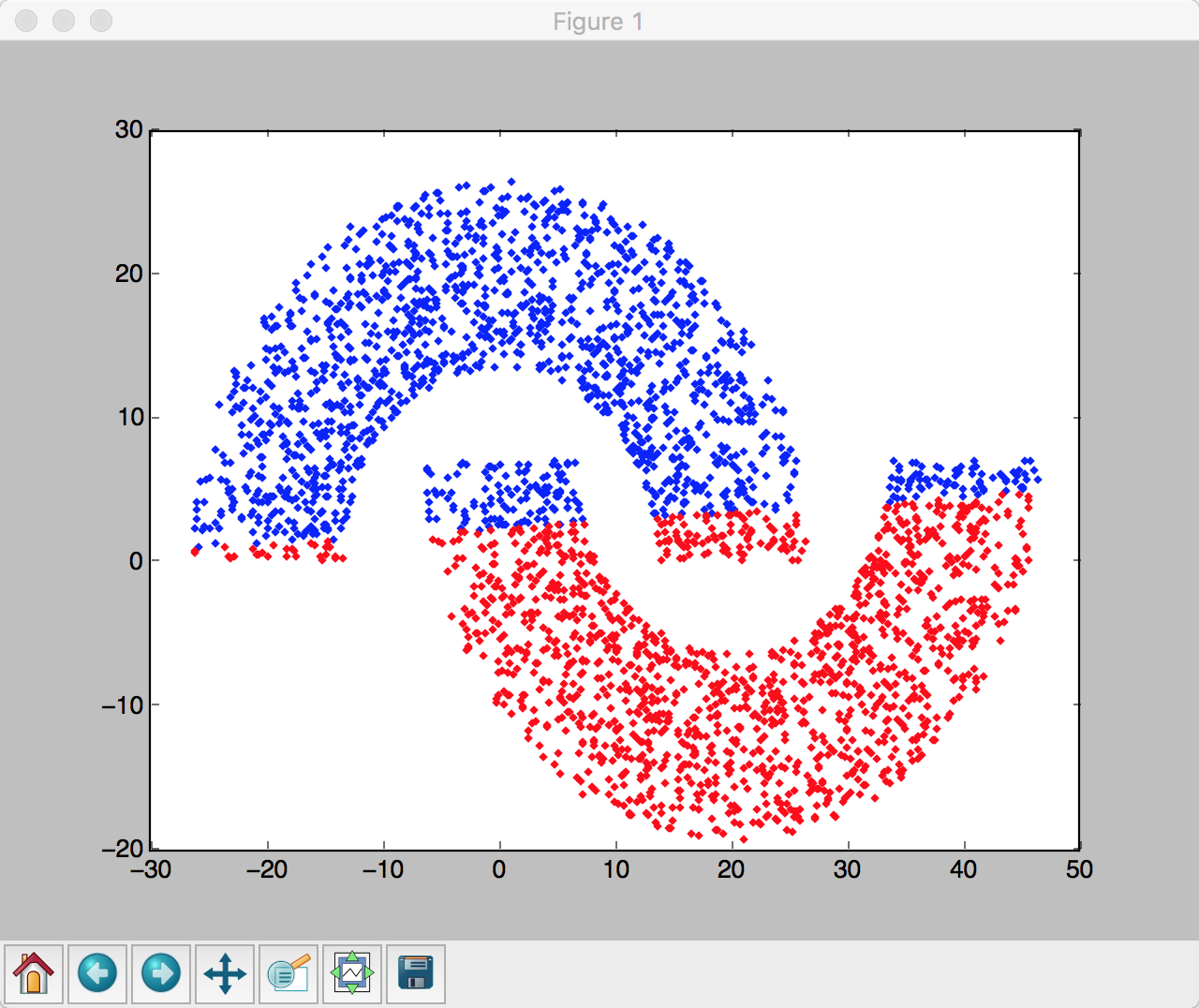
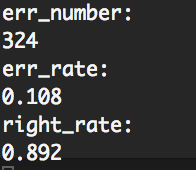
可以看到它的错误点有298个，正确率约为90.1%



* 1. 单独使用SVM的时候

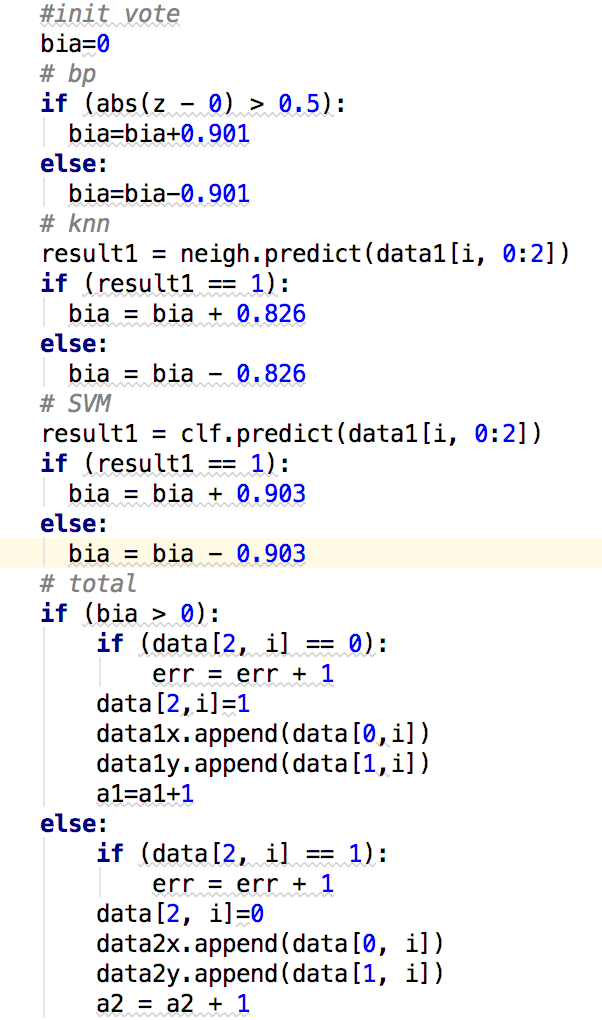
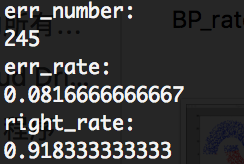
用一般的SVM会出现没有误差的情况，所以这里用了线性的SVM，它每次学习的不同带来的结果也不同，但是可以看到正确率也是维持在一个范围的，此时算它平均正确率约为90.3%

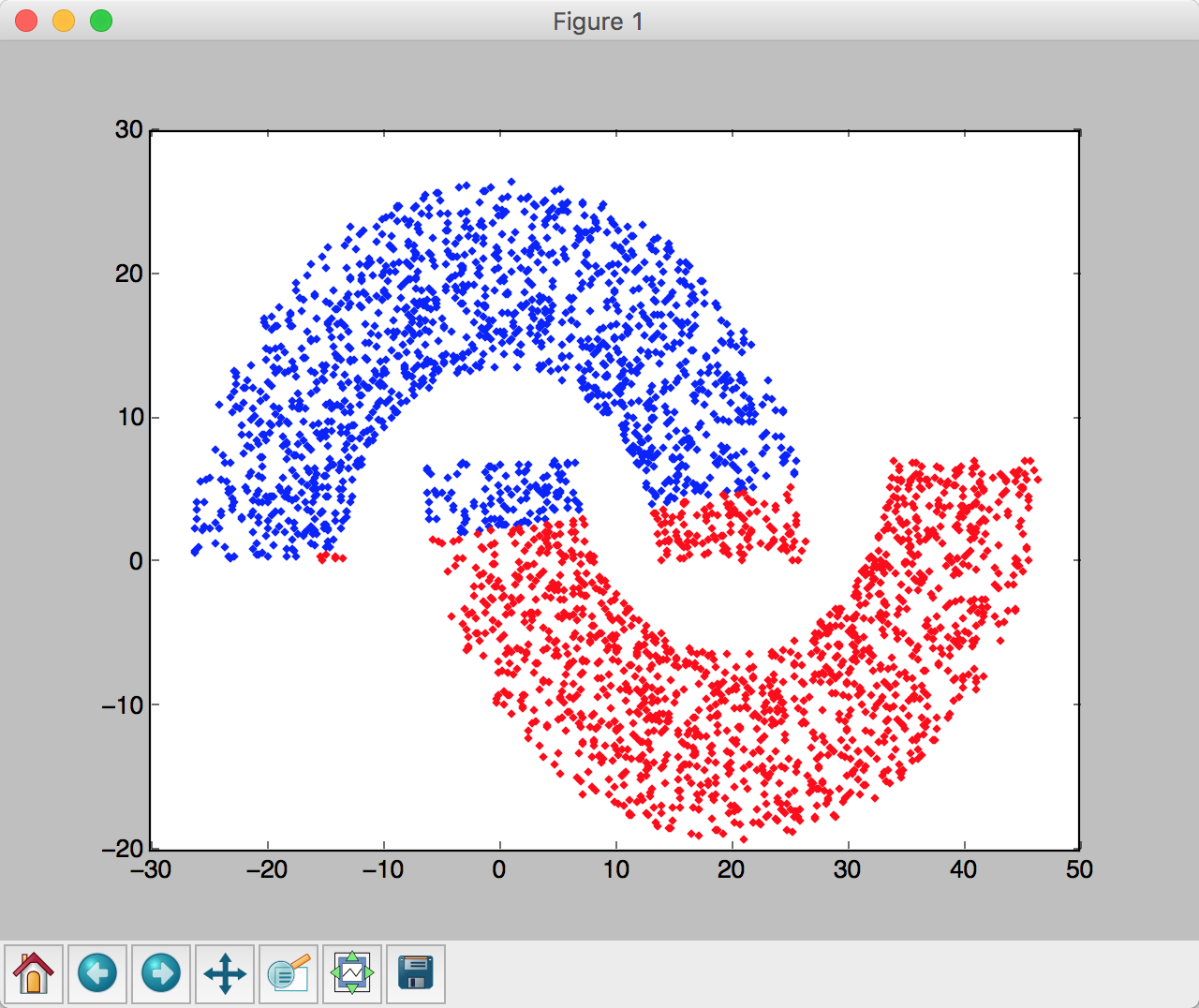
 

* 1. 集成三个算法

对一个点分类时分别用三个算法对它进行分类。分类完以后按之前的正确率为权重投票（这里是改变bia的值），然后根据投票结果（bia正负）决定该点的类别

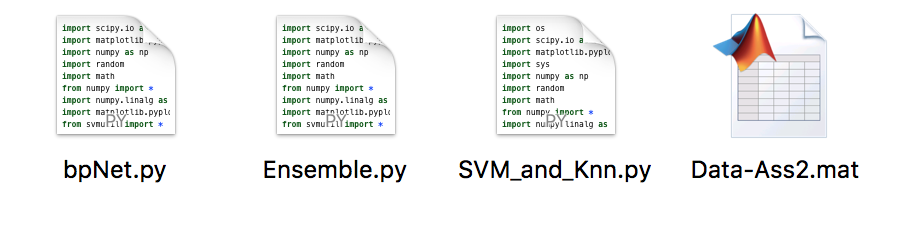
 

可以看到，此时的分类效果比前三个算法单独的时候都要好，此时的正确率达到了91.8%,比单独Knn提高了9.2%,比单独Bp提高了1.7%，比单独SVM提高了1.5%，所以说集成学习还是可以提高学习的准确率的。



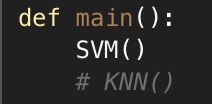
1. **收获与感悟**
   1. 经过这次实验首先我更加熟悉了python的使用
   2. 我了解了集成学习的原理和使用
2. **代码**

位于“代码”文件夹下

****

bpNet.py是神经网络的算法

SVM\_and\_Knn.py是SVM和Knn两个算法，在main方法下面把不要用的注释掉就可以了



Ensemble.py是总的集成学习