信息基础2 作业

徐瑞 20180012045

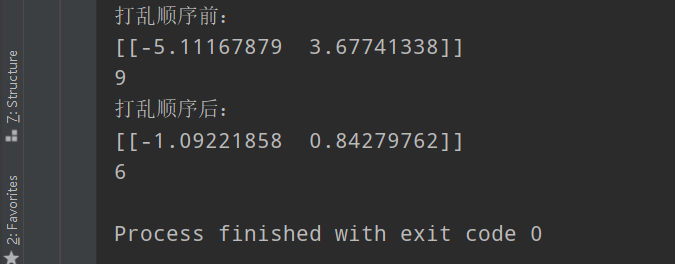
1. Perceptron

**问题1：**

**Consider running the Perceptron algorithm on some sequence of examples S (an example is a data point and its label). Let S′ be the same set of examples as S, but presented in a different order.（考虑在示例S的某些序列上运行Perceptron算法（示例是数据点及其标签）。 令S'是与S相同的一组示例，但是顺序不同。）**

1. Does the Perceptron algorithm necessarily make the same number of mistakes on S as it does on S′?( Perceptron算法是否必然在S上犯下与在S'上犯同样数量的错误？)

**答：不相同**



1. If so, why? If not, show such an S and S′ where the Perceptron algorithm makes a different number of mistakes on S′ than it does on S.( 如果是，为什么？ 如果不是，请显示这样的S和S'，其中Perceptron算法在S'上犯的错误数量与在S'上犯的错误数量不同。)

答：S1 = [[-6.69981438 0.66072861]

[-0.09597482 1.21837517]

[ 1.09221858 -0.84279762]

[ 0.03761913 0.94111016]

[ 0.90929317 -1.16953996]

[-0.16124602 1.04585258]

[-0.6967333 0.94885796]

[-0.6104734 1.07047058]

[-1.66657506 1.26994355]

[ 0.52937853 1.00556159]

[-0.53903288 1.24010824]]

S1\_y = [1 1 0 1 0 1 1 1 1 1]

S2 = [[ 1.09221858 -0.84279762]

[ 0.52937853 1.00556159]

[-0.6967333 0.94885796]

[ 0.90929317 -1.16953996]

[-6.69981438 0.66072861]

[ 0.03761913 0.94111016]

[-0.09597482 1.21837517]

[-0.6104734 1.07047058]

[-1.66657506 1.26994355]

[-0.16124602 1.04585258]]

S2\_y = [0 1 1 0 1 1 1 1 1 1]

2. A proposed kernel

**问题2：**

**Consider the following kernel function:**

1. Prove this is a legal kernel. That is, describe an implicit mapping Φ : such that K(x, x′) = Φ(x) Φ(x′). (You may assume the instance space X is finite.)(证明这是合法的内核。 也就是说，描述一个隐式映射Φ：使得K（x，x'）=Φ（x）Φ（x'）。 （您可以假定实例空间X是有限的。）

**答：易知题目给出的核函数是在当x与x’相等时取1，不相等时取0，相当于逻辑运算中的同或运算，即K（x，x’）=xx’+’**

1. In this kernel space, any labeling of points in X will be linearly separable. Justify this claim.( 在这个内核空间中，X中的点的任何标记都是线性可分离的。 证明此主张合理。)

**答：无论x与x’中的点取什么类型的值，经过核函数之后都会变成0或1，在核空间的标记是线性可分的。**

1. Since all labelings are linearly separable, this kernel seems perfect for learning any target function. Why is this actually a bad idea?（由于所有标记都是线性可分离的，因此该内核似乎非常适合学习任何目标函数。 为什么这实际上是一个坏主意？）

**答：虽然这种内核可以让任何数据都变得线性可分，但是同时也会失去一些数据本身的特征，可能这些特征正是我们想要的，但是机器没有学习到这些，可能会导致分类结果误差过大，泛化能力不强。**

3.SVM分类散点

**问题3：Gaussian Data**

有2类二维空间点，A类和B类。

A类点以(0, 0)为中心、(1, 0; 0, 1)为协方差矩阵的二维高斯分布；B类点以(1, 2) 为中心、(1, 0; 0, 2)为协方差矩阵的二维高斯分布；随机生成300个A类点，200个B类点，并用SVM的方法（可以调用机器学习库)进行分类(选择合适的kernel)。

请解释你的SVM的调用方法，描述结果。

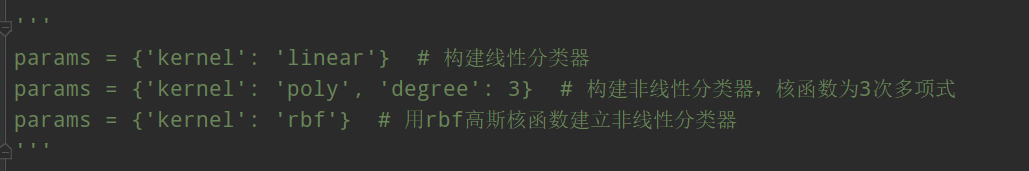
**解答：**数据集还是和以前一样，先生成500个点作为训练集，再生成500个点作为测试集，不过这次我把numpy的随机种子确定了，这样每次运行程序都会得到相同的训练集和数据集；通过调用sklearn中的svm模型，根据输入的数据不同可做不同的模型（若输入标签为连续值则做回归，若输入标签为分类值则用SVC()做分类），显然在这里我们的输入标签为分类值，使用svm中的svc（）函数，这个函数有很多的参数，参数详解如下：

* **C**：C-SVC的惩罚参数C，默认值是1.0

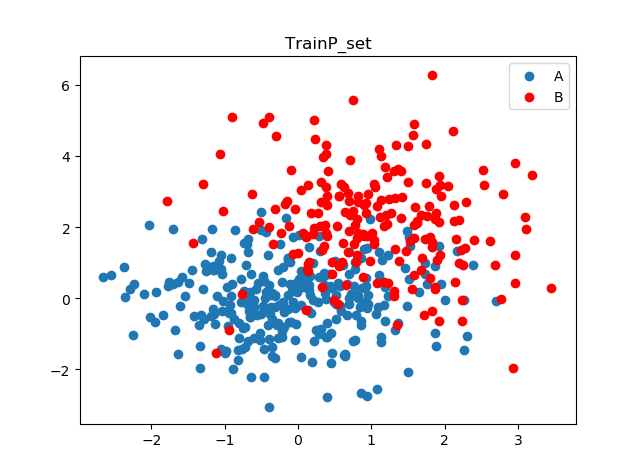
C越大，相当于惩罚松弛变量，希望松弛变量接近0，即对误分类的惩罚增大，趋向于对训练集全分对的情况，这样对训练集测试时准确率很高，但泛化能力弱。C值小，对误分类的惩罚减小，允许容错，将他们当成噪声点，泛化能力较强。

* **kernel** ：核函数，默认是rbf，可以是‘linear’,‘poly’, ‘rbf’
* liner – 线性核函数：u'v
* poly – 多项式核函数：(gamma\*u'\*v + coef0)^degree
* rbf – RBF高斯核函数：exp(-gamma|u-v|^2)
* **degree** ：多项式poly函数的维度，默认是3，选择其他核函数时会被忽略。
* **gamma** ： ‘rbf’,‘poly’ 和‘sigmoid’的核函数参数。默认是’auto’，则会选择1/n\_features
* **coef0** ：核函数的常数项。对于‘poly’和 ‘sigmoid’有用。
* **probability** ：是否采用概率估计.默认为False
* **shrinking** ：是否采用shrinking heuristic方法，默认为true
* **tol** ：停止训练的误差值大小，默认为1e-3
* **cache\_size** ：核函数cache缓存大小，默认为200
* **class\_weight** ：类别的权重，字典形式传递。设置第几类的参数C为weight \* C(C-SVC中的C)
* **verbose** ：允许冗余输出
* **max\_iter** ：最大迭代次数。-1为无限制。
* **decision\_function\_shape** ：‘ovo’, ‘ovr’ or None, default=None3
* **random\_state** ：数据洗牌时的种子值，int值

在这里我只修改了kernel这个参数，通过设置不同的核函数类型，来观察分类结果的好坏，我采用三种不同类型的核函数，分别是：



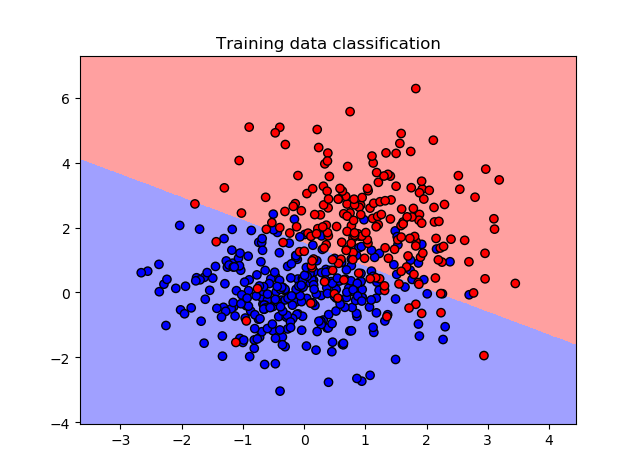
**训练集分布图：**

****

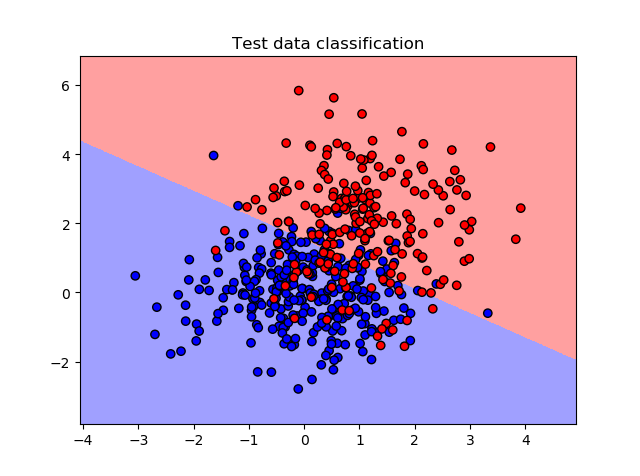
**分类结果如下：**

(蓝色代表A类点，红色代表B类点)

**1.linear线性核函数:**

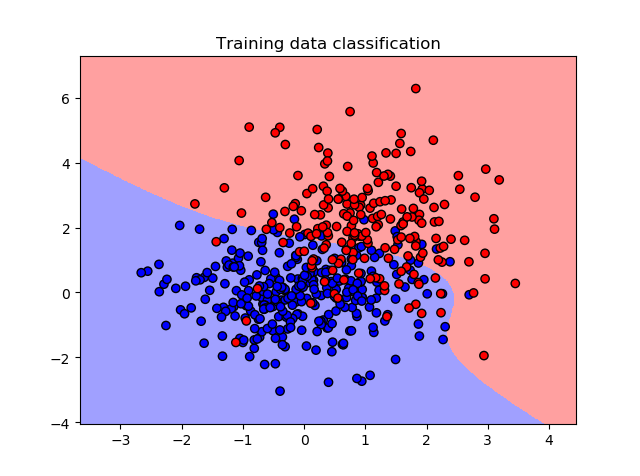
****

**训练集精确率为:0.864**

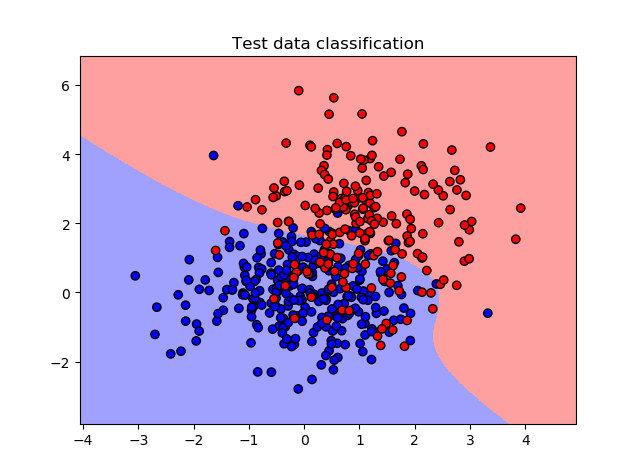
****

**测试集精确率为：0.854**

**2.poly非线性核函数，核函数为3次多项式:**

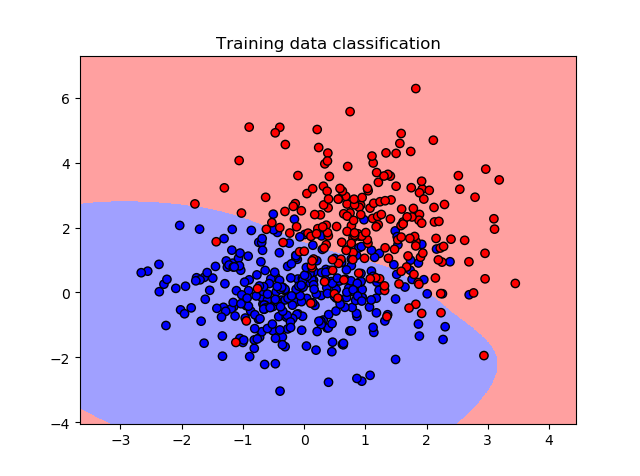
****

**训练集精确率为：0.874**

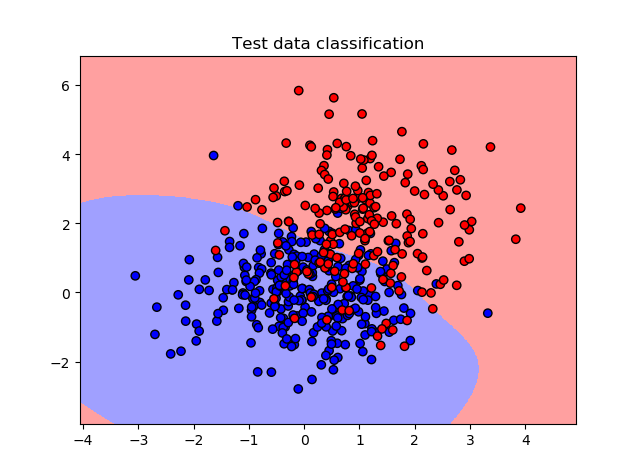
****

**测试集精确率为：0.858**

**3.** **rbf高斯核函数:**

****

**训练集精确率为：0.874**

****

**测试集精确率为：0.858**