机器学习 SVM 作业

姓名： 张泽群 学号： 19049100002 班级：1班

SVM.1

SVM 1.

$$K(x, x_i) = (1 + x \cdot x_i)^2$$

- 对于四个输入样本 $[x_1, x_2, x_3, x_4] \rightarrow$ 其中

输入	响应 y
$(-1, -1)$	-1
$(-1, +1)$	$+1$
$(+1, -1)$	$+1$
$(+1, +1)$	-1

- 核矩阵

$$K = \begin{bmatrix} K(x_1,x_1), & K(x_1,x_2), & K(x_1,x_3), & K(x_1,x_4) \\ K(x_2,x_1), & K(x_2,x_2), & K(x_2,x_3), & K(x_2,x_4) \\ K(x_3,x_1), & K(x_3,x_2), & K(x_3,x_3), & K(x_3,x_4) \\ K(x_4,x_1), & K(x_4,x_2), & K(x_4,x_3), & K(x_4,x_4) \end{bmatrix}$$

推导

- 对核函数有 $K(x,x_i) = (1 + x_1 x_{i1} + x_2 x_{i2})^2 = 1 + x_1^2 x_{i1}^2 + 2 x_1 x_2 x_{i1} x_{i2} + x_{i2}^2 x_2^2 + 2 x_1 x_{i1} + 2 x_2 x_{i2}$
- 则特征向量 $\phi(x) = [1, x_1^2, \sqrt{2} x_1 x_2, x_2^2, \sqrt{2} x_1, \sqrt{2} x_2]^T$

$$\phi(x_i) = [1, x_{i1}^2, \sqrt{2} x_{i1} x_{i2}, x_{i2}^2, \sqrt{2} x_{i1}, \sqrt{2} x_{i2}]^T$$

(1) • 将XOR的四个样本代入 $K(x,x_i)$ 中得到核矩阵：

$$K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

(2) • 目标函数的对偶形式为

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(x, x_i)$$

$$\Rightarrow \max_{\alpha} \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 + 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2)$$

SVM.2

答：通过重新构造后，SVM算法便可以应用到多分类问题中去。具体的解决方案主要有两大类，第一类是**直接法**，这种方法的核心是直接**改造优化的目标函数和限制条件**，将多分类问题中的多个参数整合到一个函数中，处理多分类问题。这种方法看似简单，但其计算复杂度比较高，实现起来比较困难，只适合用于小型问题中。另一类是**间接法**，主要是通过组合多个二分类器来实现多分类器的构造，常见的方法有one-against-one和one-against-all两种。

1.One-Versus-Rest（一对多）

训练时依次把某个类别的样本归为一类,其他剩余的样本归为另一类，这样k个类别的样本就构造出了k个SVM。分类时将未知样本分类为具有最大分类函数值的那类。

2.One-Versus-One（一对一）

这种方案的具体做法是在任意两类样本之间设计一个SVM，因此k个类别的样本就需要设计k(k-1)/2个SVM。当对一个未知样本进行分类时，最后得票最多的类别即为该未知样本的类别。

3.Many vs. Many（多对多） MvM是每次将若干个类作为正类，若干个其他类作为反类。OvO和OvR是MvM的特例。MvM的正、反类构造最常用的MvM技术：“纠错输出码”（Error Correcting Output Codes，简称ECOC）。 ECOC过程主要分为两步： 1.编码：对N个类别做M次划分，每次划分将一部分类别划为正类，一部分划为反类，从而形成一个二分类训练集；这样一共产生M个训练集，可训练出M个分类器。 2.解码：M个分类器分别对测试样本进行预测，这些预测标记组成一个编码。将这个预测编码与每个类别各自的编码进行比较，返回其中距离最小的类别作为最终预测结果。

4.Directed Acyclic Graph SVM（有向无环图）

是由Platt提出的决策导向的循环图DAG导出的，是针对“一对一”SVMS存在误分，拒分现象提出的。这种方法的训练过程类似于“一对一”方法，类别数增加时，速度快于前两者，且简单易行，对于一般规模的多类分类问题行之有效。是基于一对一方式的优化，不会出现拒分。

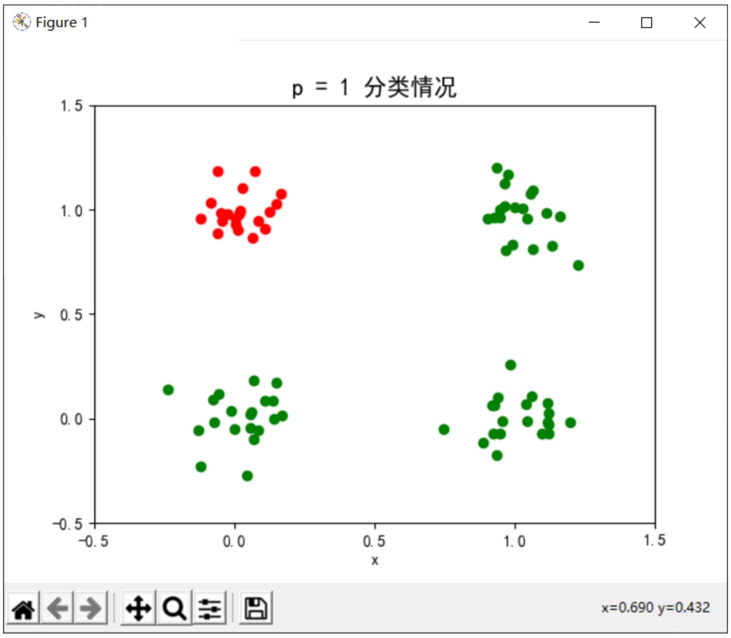
5.层次支持向量机

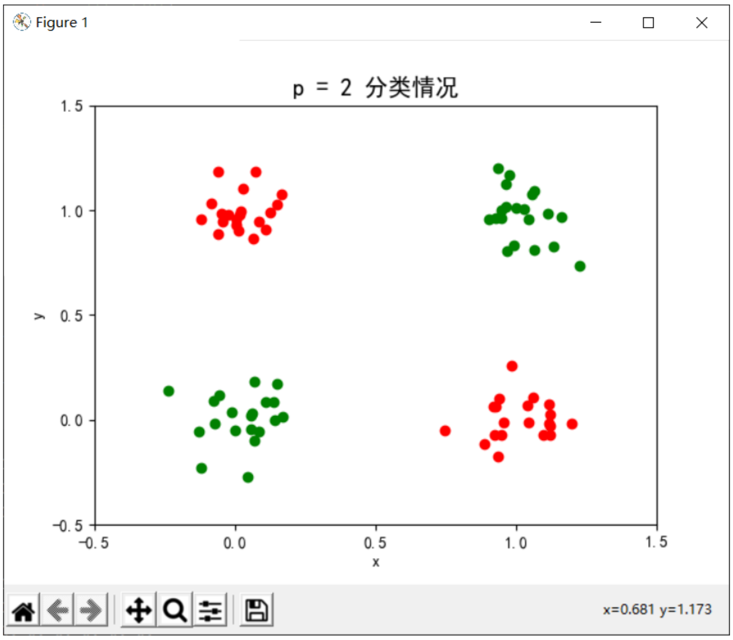
决策树的基本思想是从根节点开始，采用某种方法将该结点所包含的类别划分为两个子类，然后再对两个子类进一步划分，如此循环，直到子类中只包含一个类别为止。这样就得到一个倒立的二叉树。最后，在二叉树各决策节点训练支持向量机分类器，实现对识别样本的分类。决策树支持向量机多分类方法有很多种，不同方法的主要区别在于设计树结构的方法不同。

```python
# SVM.3
# 利用sklearn.svm.SVC()，C-支持向量分类器实现。


import numpy as np
from sklearn import svm
from matplotlib.pylab import plt

global p

def pltshow(data,labels,title):
    int_labels=map(int,list(labels))
    colors=['r','g']
    x,y = data.T
    for index,label in enumerate(int_labels):
        plt.scatter(x[index],y[index],color=colors[label])
    plt.xlabel('x')
    plt.ylabel("y")
    my_x_ticks = np.arange(-0.5, 2, 0.5)
    my_y_ticks = np.arange(-0.5, 2, 0.5)
    plt.xticks(my_x_ticks)
    plt.yticks(my_y_ticks)
    plt.title(title, fontsize=15)
    plt.show()


if __name__ == '__main__':
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    # 数据生成
    n=2
    sample_num=20
    sigma=0.01 # 样本偏移度
    cov = sigma * np.identity(n)
    mean = [[1, 1], [1, 0], [0, 1], [0, 0]]

    posdata1= np.random.multivariate_normal(mean[0], cov, sample_num)  # 正数据集
    posdata2 = np.random.multivariate_normal(mean[3], cov, sample_num)
    posd = np.vstack((posdata1,posdata2))
    negdata1 = np.random.multivariate_normal(mean[1], cov, sample_num) # 负数据集
    negdata2 = np.random.multivariate_normal(mean[2], cov, sample_num)
    negd = np.vstack((negdata1,negdata2))

    #数据集和标签集
    pos_num = posd.shape[0]   # 行数
    neg_num = negd.shape[0]
    labels = np.ones((pos_num + neg_num, 1))
    labels[pos_num:] = 0
    train_data = np.vstack((posd, negd))
    DataMat = np.array(train_data, dtype='float32')
```

```
    Labels = np.array(labels.reshape(-1))

    #预测
    for p in range(1,10):
        print("\n多项式核次方数 p = ",p)
        svc = svm.SVC(degree=p,coef0=1,gamma=1,kernel='poly')
        svc.fit(DataMat, Labels)
        print('标签集:')
        print(Labels)
        print("预测值:")
        pre=svc.predict(DataMat)
        print(pre)
        sum=0
        for en in range(len(Labels)):
            sum+=abs(np.float(Labels[en]-pre[en]))
        print('错误率 =',sum/len(Labels))
        pltshow(DataMat,pre,title='p = {} 分类情况'.format(p))
```

运行结果:





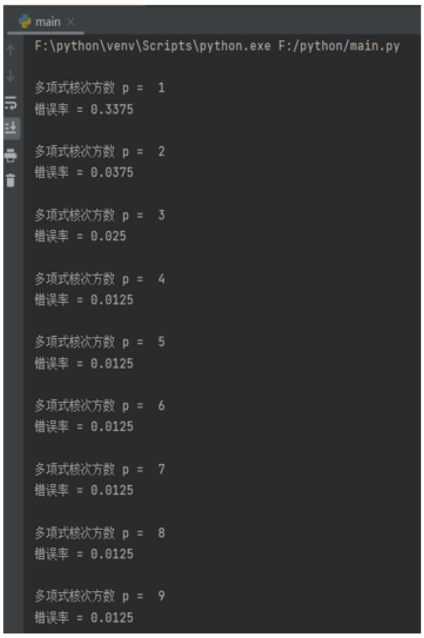

```
多项式核次方数 p =  1
标签集:
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0.]
预测值:
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0.]
错误率 = 0.25

多项式核次方数 p =  2
标签集:
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0.]
预测值:
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0.]
错误率 = 0.0

多项式核次方数 p =  3
```

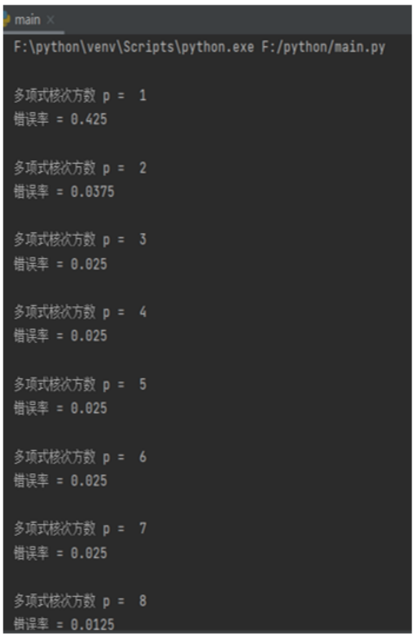
运行结果在惩罚松弛变量=1，最大迭代次数不限制的情况下，除了p=1，其余情况均可处理异或问题。

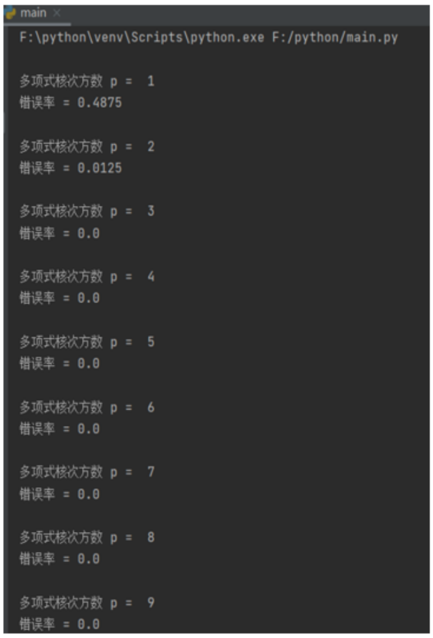
当然这个处理的效果也取决于样本的偏移程度，此处运行的样本偏移程度为0.01。

以下是不同样本偏移程度时的结果

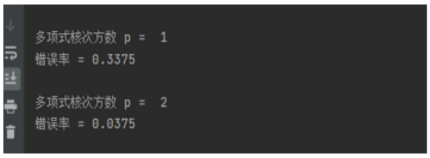
样本偏移= 0.5　　　　　　　　　　0.4　　　　　　　　　　0.3

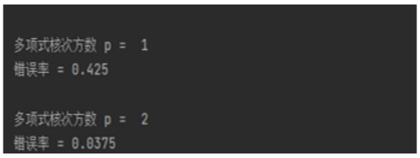






可见p=1 和 p=2 之间的错误率有着质的提升。





结论： 1. 因此 p 的最小数为2，即p至少要等于2才能解决异或问题。

此外，一般p=3，4时便可以取到最优效果。

2. 根据不同样本偏移度时 不同p的错误率比较结果，可以得出，当使用比最小值大的p值，p越大，其分类的精确程度就越高，同时计算量也越大。