西安电子科技大学

模式识别 课程实验报告

人工智能学院 1920032 班

姓名 付凯文 学号 19170100004

实验日期 2021 年 11 月 28 日

成绩:
实验报告内容基本要求及参考格式 一、实验目的 二、实验基本原理及步骤 三、实验仿真结果与分析 四、实验中遇到的问题及解决方法（至少 3 个，每人至少写 1 个，写清楚谁的问题和解决方法）

-

实验三　不同核函数对 SVM 分类器的影响

一、　实验目的

1. 学习并掌握 SVM 算法
2. 利用 SVM 算法对 iris 数据集进行分类
3. 比较不同核心函数的差别
4. 理解并分析不同核函数对 SVM 分类器的影响

二、　实验基本原理及步骤

SVM 算法简介：

支持向量机（support vector machines, SVM）是一种二分类模型，它的基本模型是定义在特征空间上的间隔最大的线性分类器，间隔最大使它有别于感知机；SVM 还包括核技巧，这使它成为实质上的非线性分类器。SVM 的的学习策略就是间隔最大化，可形式化为一个求解凸二次规划的问题，也等价于正则化的合页损失函数的最小化问题。SVM 的的学习算法就是求解凸二次规划的最优化算法。

非线性 SVM 算法原理：

对于输入空间中的非线性分类问题，可以通过非线性变换将它转化为某个维特征空间中的线性分类问题，在高维特征空间中学习线性支持向量机。由于在线性支持向量机学习的对偶问题里，目标函数和分类决策函数都只涉及实例和实例之间的内积，所以不需要显式地指定非线性变换，而是用核函数替换当中的内积。核函数表示，通过一个非线性转换后的两个实例间的内积。具体地，$k(x,z)$ 是一个函数，或正定核，意味着存在一个从输入空间到特征空间的映射 $\emptyset(x)$，对任意输入空间中的x,z，有：

$$k(x,z)=\emptyset(x)*\emptyset(z)$$

在线性支持向量机学习的对偶问题中，用核函数$k(x,z)$替代内积，求解得到的就是非线性支持向量机：　　$f(x)=\text{sign}\left(\sum_{i=1}^{N}\alpha_i^* y_i K(x,x_i)+b^*\right)$

综合以上讨论，我们可以得到非线性支持向量机学习算法如下：

输入：训练数据集

$$T=\{(x_1,y_1),(x_2,y_2)\ldots\ldots(x_N,y_N)\},$$
$$x_i\in R^n,\ \ y_i\in\{+1,-1\},i=1,2\ldots\ldots N$$

输出：分离超平面和分类决策函数

首先选取适当的核函数$k(x,z)$和惩罚参数$C>0$,构造并求

$$\min_{\boldsymbol{\alpha}}\frac{1}{2}\sum_{i=1}^{N}\ \sum_{j=1}^{N}\ \alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i,\boldsymbol{x}_j)-\sum_{i=1}^{N}\alpha_i$$

$$s.t.\sum_{i=1}^{N}\alpha_i y_i=0\quad 0\le\alpha_i\le C$$

得到最优解：

$$\alpha^*=(\alpha_{1,}^*,\ \alpha_{2,}^*\ldots\ldots.\alpha_{N,}^*)^T$$

之后选择α^*的一个分量α_j^*满足条件$0\le\alpha_j^*\le C$ ，计算

$$b^*=y_j-\sum_{i=1}^{N}\alpha_i^* y_i K(\boldsymbol{x}_i,\boldsymbol{x}_j)$$

则分类决策函数：

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i^* y_i K(x, x_i) + b^*\right)$$

常用的 SVM 核函数有以下四种，分别为线性核，高斯核，多项式核，sigmoid 核，他们的表达式如下：

线性核：$k(x,z) = x \cdot z$

多项式核：$k(x,z) = (\gamma x \cdot z + r)^d$

Sigmoid 核：$k(x,z) = \tanh(\gamma x \cdot z + r)$

高斯核：$k(x,z) = \exp(-\gamma||x - z||^2)$

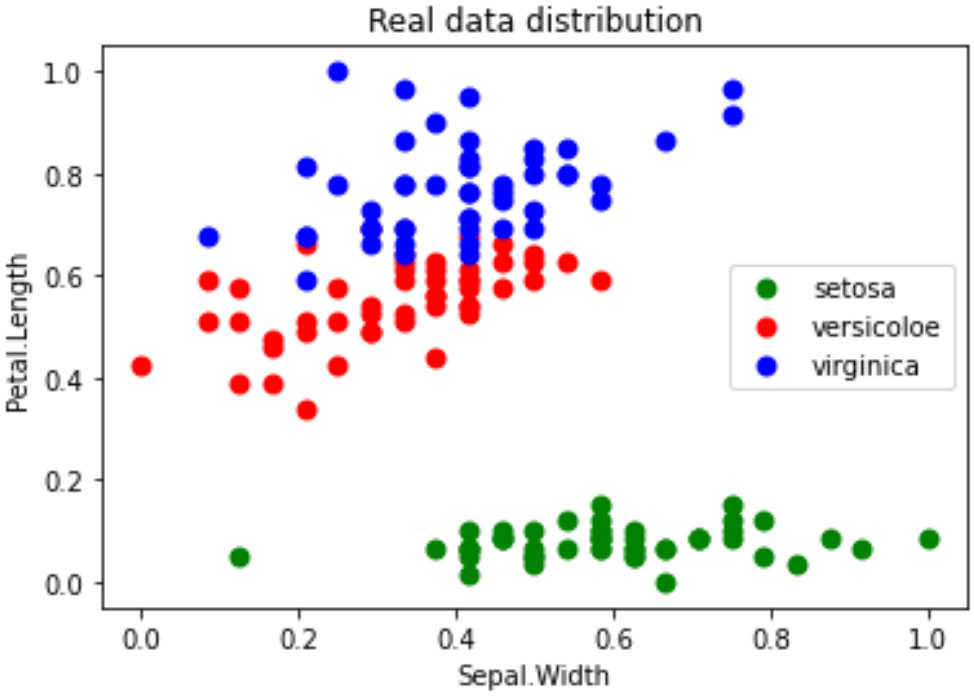
实验步骤：

1. 准备实验所需的数据集，本实验中，我们使用 iris 数据集，对数据集进行归一化处理，归一化所用公式为$\boldsymbol{x}_i = (\boldsymbol{x}_i - \boldsymbol{x}_{\min})/(\boldsymbol{x}_{max} - \boldsymbol{x}_{\min})$
2. 随机选取两个维度作为x, y轴，将归一化的数据集进行可视化处理。
3. 将数据随机打乱，之后对数据集进行划分，训练集/测试集 = 7:3
4. 选取 C=1，分别对不同核函数的 SVM 模型进行训练同时记录训练所用时间。
5. 利用测试集对训练好的不同模型进行测试，计算各个模型预测的正确率。
6. 为了更加直观的对比不同核函数对 SVM 分类器的影响，选取 Sepal Width 和 Petal Length 两个维度作为特征，重新训练二维的 SVM 分类器。并绘制出各个分类器的分别面。
7. 比较不同核函数的差别，并分析核函数对 SVM 分类器的影响。

三、 实验仿真结果及分析

实验仿真结果：

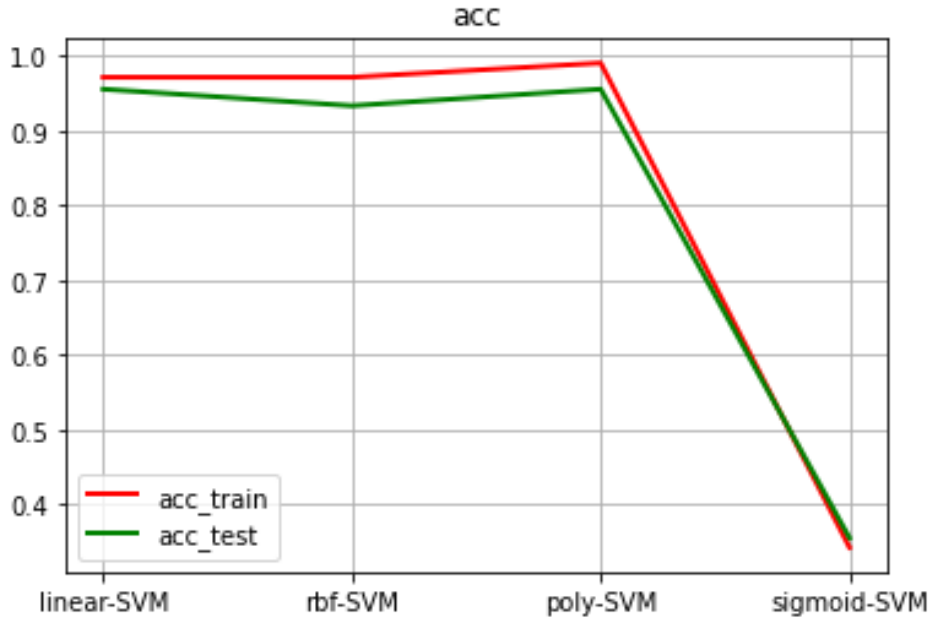
我们首先导入数据集并进行归一化处理，之后 Sepal Width 和 Petal Length 两个特征作为 X,Y 轴，对数据集进行可视化。结果如下图所示：



之后对数据集打乱后划分为训练集和测试集，两者的比例为 7:3，用划分好的数据集对不同核函数的 SVM 分类器进行训练并利用训练好的分类器对测试集进行测试，得到各个分类器的预测的准确率，如下图所示，红色的线为 SVM 对训练集的分类正确率，绿色线条为 SVM 对测试集分类的正确率。具体数据为：

train_acc: [0.9714285714285714, 0.9714285714285714, 0.9904761904761905, 0.34285714285714286]

test_acc: [0.9555555555555556, 0.9333333333333333, 0.9555555555555556, 0.35555555555555557]

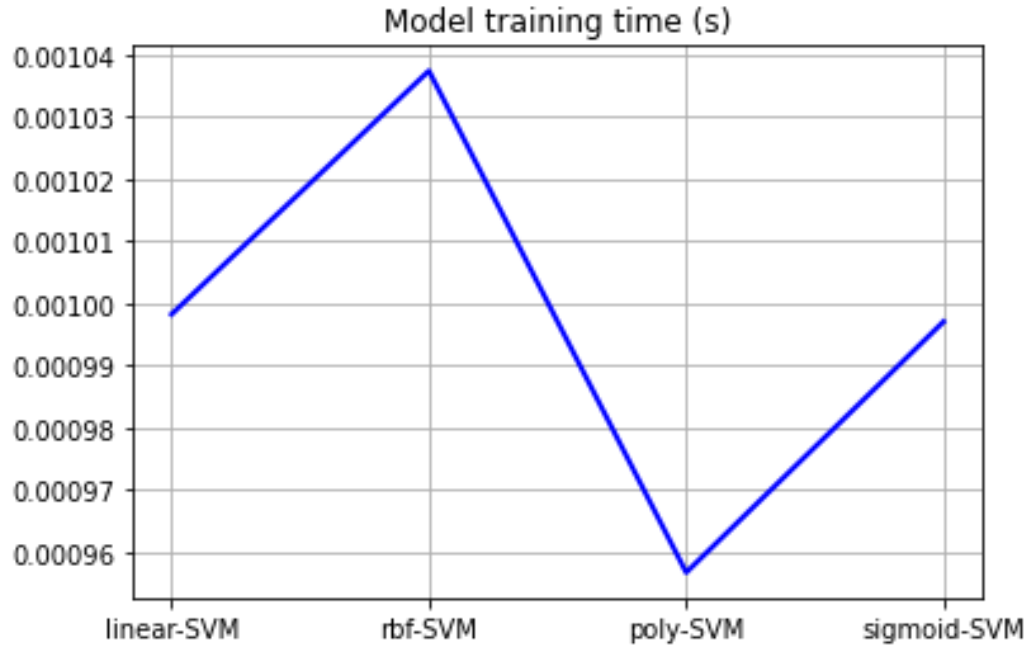


训练四种核函数 SVM 所用的时间为：

Model training time (s):

[0.000998258590, 0.001037359237, 0.000956773757, 0.000997066497]

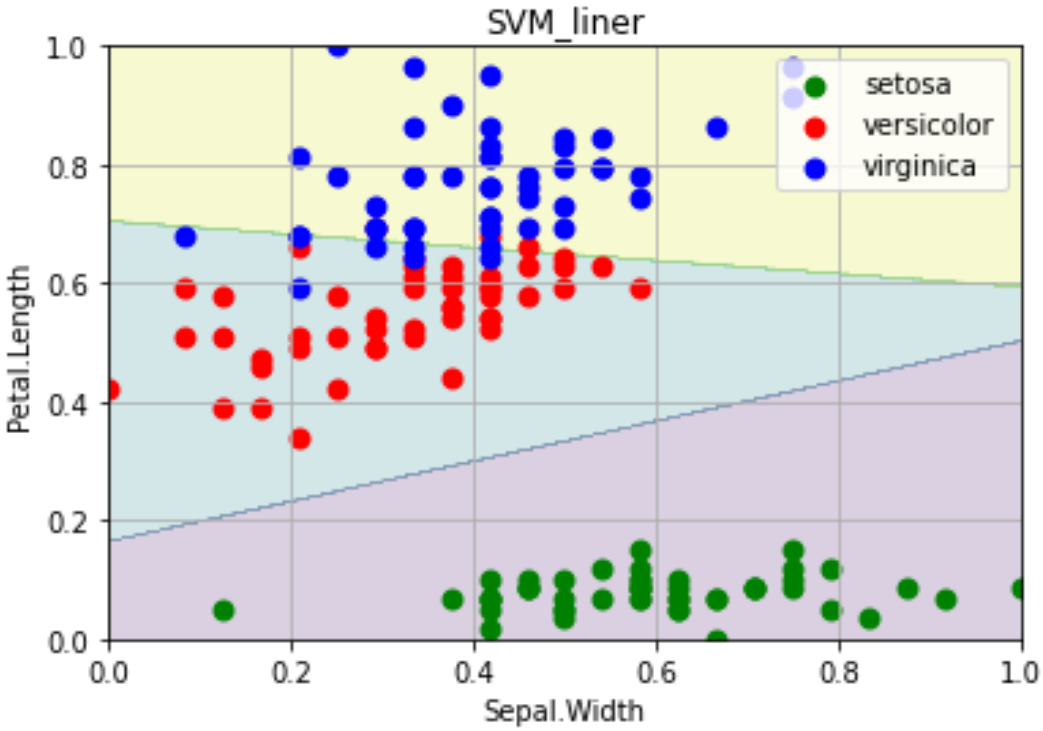
示意图如下：



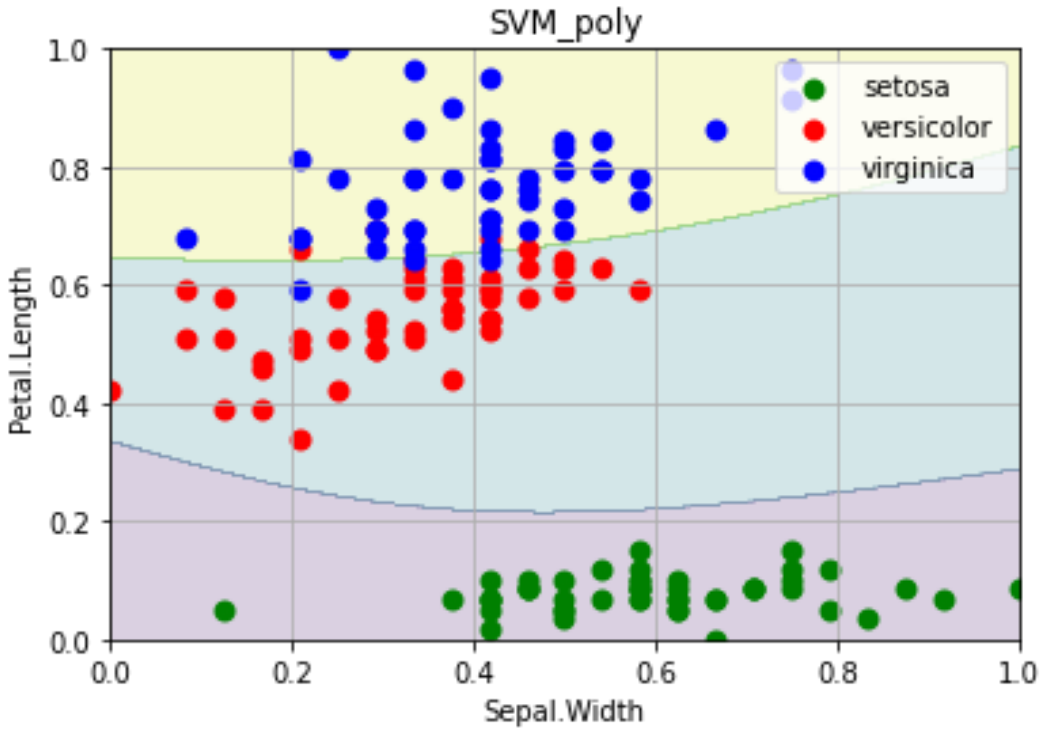
之后，为了直观的观察不同核函数的 SVM 分类器，选取 Sepal Width 和 Petal Length

两个维度的特征，重新训练二维的 SVM 分类器，并利用等高图的形式将分界面进行可视化。不同核函数 SVM 分类器的分界面如图所示：

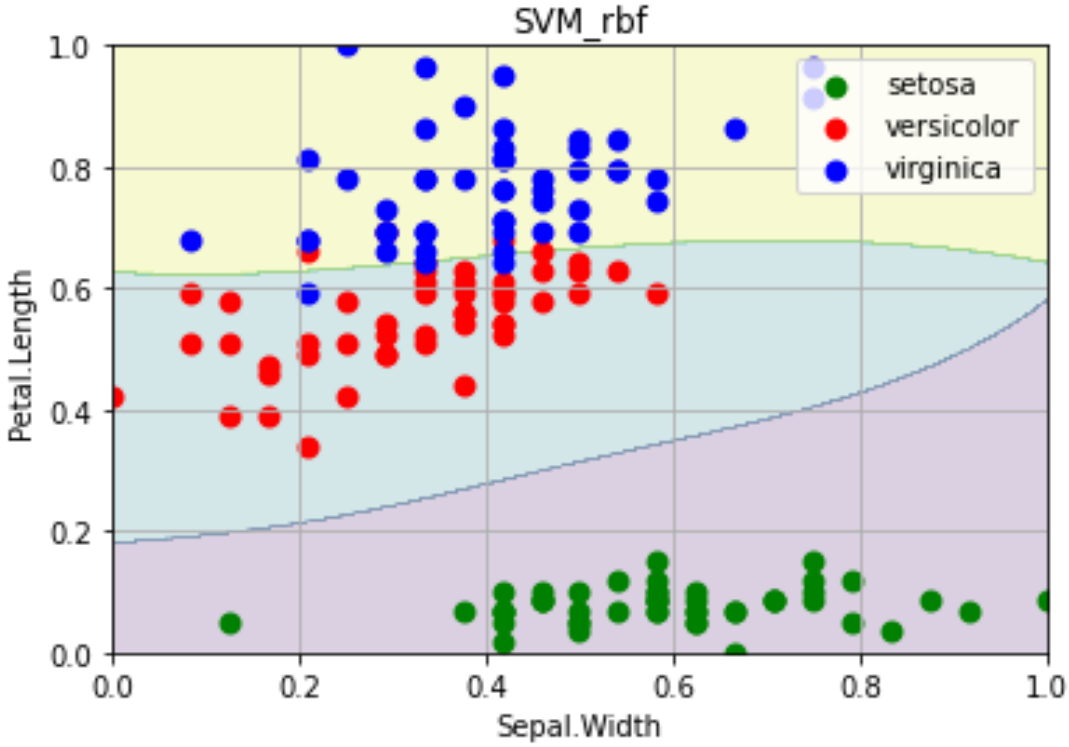
线性核函数：



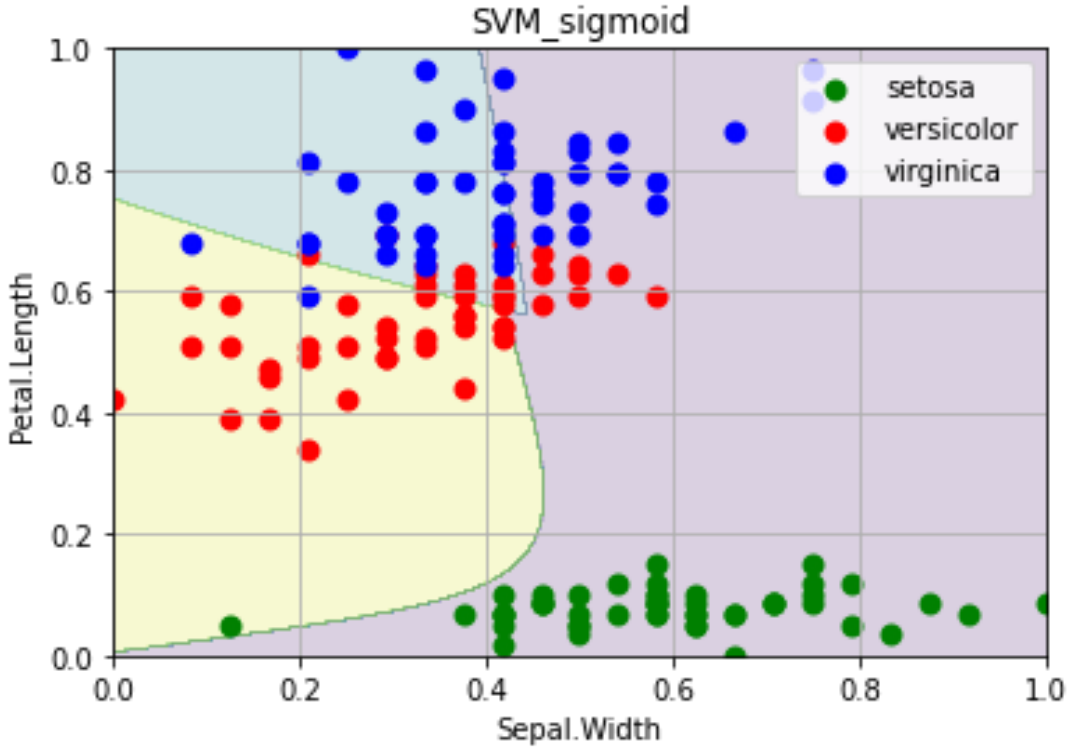
多项式核函数：



高斯核函数：



Sigmoid 函数核：



实验结果分析：

对实验结果进行对比分析，可以得出以下结论：

首先是训练时长，由于训练所用时间跟数据初始化关系密切，不同的初始化会引起同一核函数的 SVM 分类器训练时长在较大范围内变动。在进行了多次试验后可以得出，线性核 SVM 分类器训练所用平均时长最短，而高斯核 SVM 分类器的平均训练时间最长。

然后是分类的正确率，在进行多次试验后，得知线性核，高斯核，多项式核的 SVM 分类器均有着极高的分类正确率，但 sigmoid 核 SVM 的分类争取率始终很低，在百分之五十以下，可知在 iris 数据集上，sigmoid 核函数并不适用。

线性核计算量小，可以较快地解决线性可分问题，同时可解释性强，可以轻易知道哪些特征较为重要，缺点是只能解决线性可分割问题。

多项式核依靠升维将原本线性不可分的数据转化为线性可分的。可以解决非线性问题，同时可以通过主观设置幂数数来实现结果的预判。但是对于大数量级的幂数不太适用，且分类效果比较依赖参数的选择。

高斯核可以映射到无限维，决策边界更为多样，因为只有一个参数，相比于多项式核更容易选择。但可解释较差，计算速度缓慢且容易过拟合。

采用 Sigmoid 函数作为核函数时，支持向量机实现的就是一种多层感知器神经网络，应用 SVM 方法，隐含层节点数目、隐含层节点对输入节点的权值都是在训练的过程中自动确定的。而且支持向量机的理论基础决定了它最终求得的是全局最优值而不是局部最小值，也保证了它对于未知样本的良好泛化能力而不会出现过学习现象。

在实际应用过程中，如特征维数较高则选择线性核，样本数目可观，特征少则选择非线性核。样本数目较大时选择线性核可以避免庞大的计算量。

同时，通过实验我们也可以得知，支持向量机性能的优劣主要取决于核函数的选取，所以对于一个实际问题而言，如何根据实际的数据模型选择合适的核函数从而构造 SVM 算法。目前比较成熟的核函数及其参数的选择都是人为的，根据经验来选取的，带有一定的随意性.在不同的问题领域，核函数应当具有不同的形式和参数，所以在选取时候应该将领域知识引入进来，但是目前还没有好的方法来解决核函数的选取问题。

四、 实验中遇到的问题及解决方法

遇到的问题：不理解 SVM 中惩罚系数 C 的作用

解决方法：在查阅相关资料并更改 C 进行了多次实验之后对 C 的作用有了深刻的理解。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，但容易过拟合，高方差，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，但过小容易欠拟合，高偏差。

五、 附录

```
#author:fkw
#utf-8
#creat time:2021/11/28
#SVM
#dataset=iris
from sklearn import datasets
import matplotlib.pyplot as plt
import numpy as np
```

```
import time
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
#load the dataset
iris = datasets.load_iris()
# 特征值
diris = iris.data
# 标签
tiris = iris.target
labels = ['setosa', 'versicolor', 'virginica']
#数据可视化
# 数组归一化 x=(x-min)/(max-min)
for i in range(4):
    diris[:, i] = (diris[:, i]-np.min(diris[:, i]))/(np.max(diris[:, i])-np.min(diris[:, i]))
#绘制 dataset 散点图,选取 2,3 列作为 Xlable,ylabel
x_axis=diris[:,1]
y_axis=diris[:,2]
plt.scatter(x_axis[tiris==0],y_axis[tiris==0],c='green',marker='.',linewidths=5)
plt.scatter(x_axis[tiris==1],y_axis[tiris==1],c='red',marker='.',linewidths=5)
plt.scatter(x_axis[tiris==2],y_axis[tiris==2],c='blue',marker='.',linewidths=5)
plt.legend(["setosa","versicoloe","virginica"])
plt.title("Real data distribution")
plt.xlabel('Sepal.Width')
plt.ylabel('Petal.Length')
plt.show()
# 划分训练集和测试集
# 样本总数
num = diris.shape[0]
# 划分数据集 train/test=7:3
ratio = 7/3
# 测试集样本数目
num_test = int(num/(1+ratio))
# 训练集样本数目
num_train = num-num_test
# 产生样本标号并打乱
index = np.arange(num)
np.random.shuffle(index)
#前 30%做测试集
diris_test = diris[index[:num_test],:]
tiris_test = tiris[index[:num_test]]
```

```
#剩余的做训练集
diris_train = diris[index[num_test:],:]
tiris_train = tiris[index[num_test:]]
#构建分类器
#kernel 是不同的方法
svm_linear = SVC(C=1, kernel='linear')
svm_rbf= SVC(C=1, kernel='rbf')
svm_poly= SVC(C=1, kernel='poly')
svm_sigmoid= SVC(C=1, kernel='sigmoid')
#分类器的训练时间
t0=time.time()
svm_linear.fit(diris_train,tiris_train)
t1=time.time()
svm_rbf.fit(diris_train,tiris_train)
t2=time.time()
svm_poly.fit(diris_train,tiris_train)
t3=time.time()
svm_sigmoid.fit(diris_train,tiris_train)
t4=time.time()
#正确率
acc_linear_train=accuracy_score(tiris_train,svm_linear.predict(diris_train))
acc_linear_test=accuracy_score(tiris_test,svm_linear.predict(diris_test))
acc_linear_train=accuracy_score(tiris_train,svm_rbf.predict(diris_train))
acc_rbf_test=accuracy_score(tiris_test,svm_rbf.predict(diris_test))
acc_poly_train=accuracy_score(tiris_train,svm_poly.predict(diris_train))
acc_poly_test=accuracy_score(tiris_test,svm_poly.predict(diris_test))
acc_sigmoid_train=accuracy_score(tiris_train,svm_sigmoid.predict(diris_train))

acc_sigmoid_test=accuracy_score(tiris_test,svm_sigmoid.predict(diris_test))
#画图_不同 svm 的 acc
x=[0,1,2,3]
y_train=[acc_linear_train,acc_linear_train,acc_poly_train,acc_sigmoid_train]
y_test=[acc_linear_test,acc_rbf_test,acc_poly_test,acc_sigmoid_test]
plt.plot(x,y_train,'r-',lw=2,label=u'acc_train')
plt.plot(x,y_test,'g-',lw=2,label=u'acc_test')
plt.xticks(x, [u'linear-SVM', u'rbf-SVM', u'poly-SVM', u'sigmoid-SVM'], rotatio
n=0)
plt.legend(loc='lower left')
plt.title('acc')
plt.grid()
print('train_acc:',y_train)
print('test_acc:',y_test)
#画图_不同 svm 的 train_time
y_time=[t1-t0,t2-t1,t3-t2,t4-t3]
```

```
plt.plot(x,y_time,'b-',lw=2,label=u'acc_test')
plt.xticks(x, [u'linear-SVM', u'rbf-SVM', u'poly-SVM', u'sigmoid-SVM'], rotation=0)
plt.title('Model training time (s)')
plt.grid()
print('Model training time (s):',y_time)
#绘制等高图
#data
diris = iris.data
# 标签
tiris = iris.target
diris=np.dstack((diris[:,1],diris[:,2]))[0]
#print(diris)
#grid_show.shape
svm_linear.fit(diris,tiris)
svm_rbf.fit(diris,tiris)
svm_poly.fit(diris,tiris)
svm_sigmoid.fit(diris,tiris)
#步长 0.01
step=0.001
x=np.arange(0,1+step,step)
y=np.arange(0,1+step,step)
#生成网路采样点
X,Y= np.meshgrid(x,y)
#测试点
grid_show = np.dstack((X.flat, Y.flat))[0]
#grid_show.shape
svm_linear_grid = svm_linear.predict(grid_show)
svm_rbf_grid = svm_rbf.predict(grid_show)
svm_poly_grid = svm_poly.predict(grid_show)
svm_sigmoid_grid = svm_sigmoid.predict(grid_show)

svm_linear_grid=np.mat(svm_linear_grid).reshape((len(x),len(y)))
svm_rbf_grid=np.mat(svm_rbf_grid).reshape((len(x),len(y)))
svm_poly_grid=np.mat(svm_poly_grid).reshape((len(x),len(y)))
svm_sigmoid_grid=np.mat(svm_sigmoid_grid).reshape((len(x),len(y)))


#颜色填充
plt.figure()
plt.contourf(X,Y,svm_linear_grid,alpha=0.2)
#画等高线
#plt.contour(X,Y,svm_linear_grid)
```

```
plt.scatter(x_axis[tiris==0],y_axis[tiris==0],c='green',marker='.',linewidths=5,label='setosa')
plt.scatter(x_axis[tiris==1],y_axis[tiris==1],c='red',marker='.',linewidths=5,label='versicolor')
plt.scatter(x_axis[tiris==2],y_axis[tiris==2],c='blue',marker='.',linewidths=5,label='virginica')
plt.title("SVM_liner")
plt.xlabel('Sepal.Width')
plt.ylabel('Petal.Length')
plt.legend(loc='upper right')
plt.grid()
plt.show()

plt.figure()
plt.contourf(X,Y,svm_rbf_grid,alpha=0.2)
#画等高线
#plt.contour(X,Y,svm_rbf_grid)
plt.scatter(x_axis[tiris==0],y_axis[tiris==0],c='green',marker='.',linewidths=5,label='setosa')
plt.scatter(x_axis[tiris==1],y_axis[tiris==1],c='red',marker='.',linewidths=5,label='versicolor')
plt.scatter(x_axis[tiris==2],y_axis[tiris==2],c='blue',marker='.',linewidths=5,label='virginica')
plt.title("SVM_rbf")
plt.xlabel('Sepal.Width')
plt.ylabel('Petal.Length')
plt.grid()
plt.legend(loc='upper right')
plt.show()

plt.figure()
plt.contourf(X,Y,svm_poly_grid,alpha=0.2)
#画等高线
#plt.contour(X,Y,svm_poly_grid)
plt.scatter(x_axis[tiris==0],y_axis[tiris==0],c='green',marker='.',linewidths=5,label='setosa')
plt.scatter(x_axis[tiris==1],y_axis[tiris==1],c='red',marker='.',linewidths=5,label='versicolor')
plt.scatter(x_axis[tiris==2],y_axis[tiris==2],c='blue',marker='.',linewidths=5,label='virginica')
plt.title("SVM_poly")
plt.xlabel('Sepal.Width')
plt.ylabel('Petal.Length')
plt.grid()
```

```
plt.legend(loc='upper right')
plt.show()

plt.figure()
plt.contourf(X,Y,svm_sigmoid_grid,alpha=0.2)
#画等高线
#plt.contour(X,Y,svm_sigmoid_grid)
plt.scatter(x_axis[tiris==0],y_axis[tiris==0],c='green',marker='.',linewidths=5
,label='setosa')
plt.scatter(x_axis[tiris==1],y_axis[tiris==1],c='red',marker='.',linewidths=5,l
abel='versicolor')
plt.scatter(x_axis[tiris==2],y_axis[tiris==2],c='blue',marker='.',linewidths=5,
label='virginica')
plt.title("SVM_sigmoid")
plt.xlabel('Sepal.Width')
plt.grid()
plt.ylabel('Petal.Length')
plt.legend(loc='upper right')
plt.show()
```