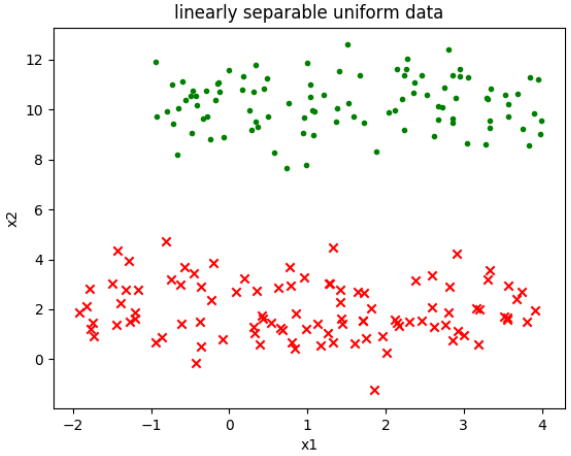
SVM实验报告

苏智龙
2018Z8017761055

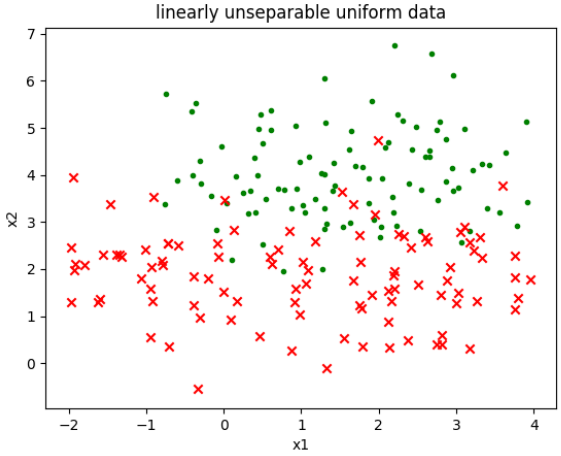
代码见末尾附录

（一）数据生成

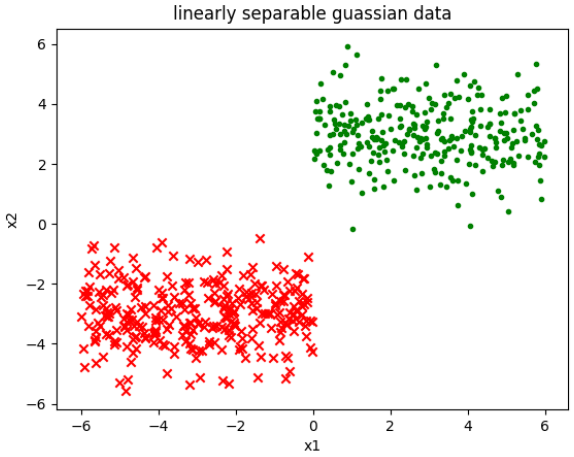
1、两组线形均匀分布的数据（完全线性可分）



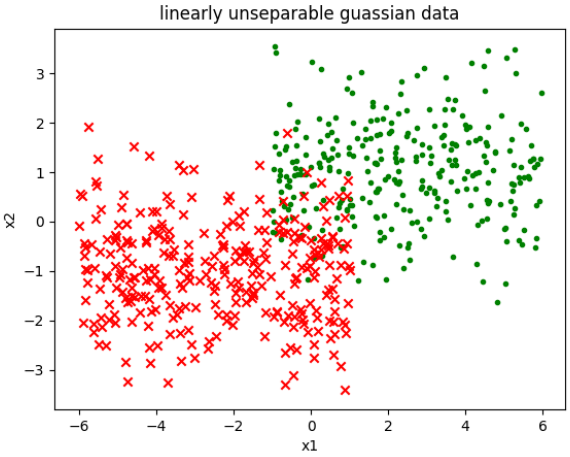
2、两组线形均匀分布的数据（线性不可分）



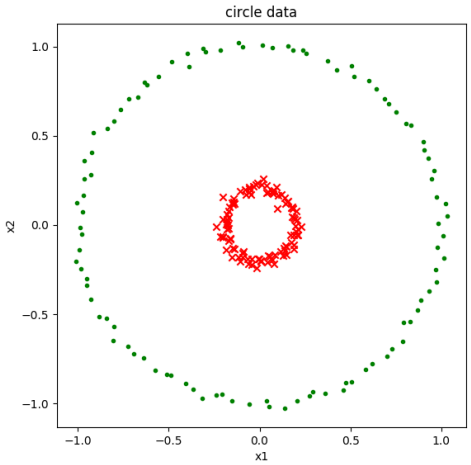
3、两组高斯分布的数据（完全线性可分）



4、两组高斯分布的数据（线性不可分）

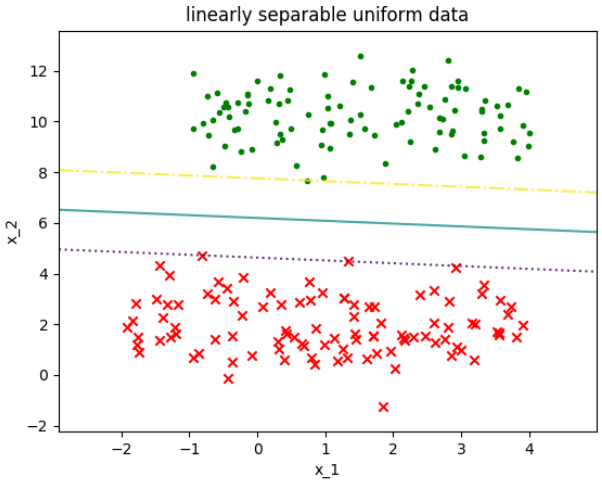


5、环状数据

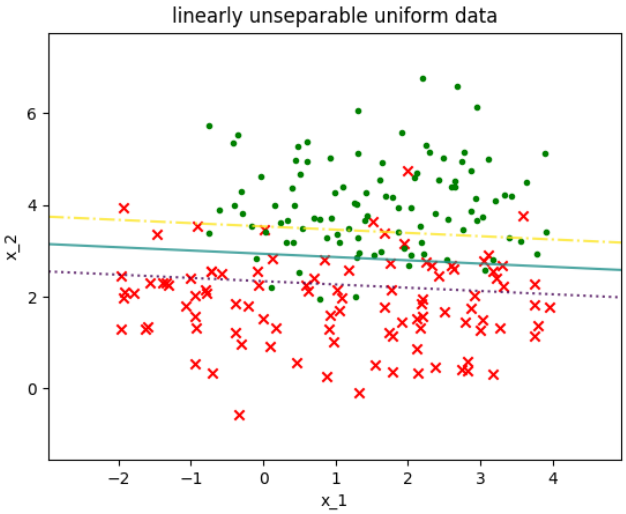


（二）使用svm进行二分类

1、使用线性 svm 对(一) 1 中生成的数据进行分类，并画出分类界面

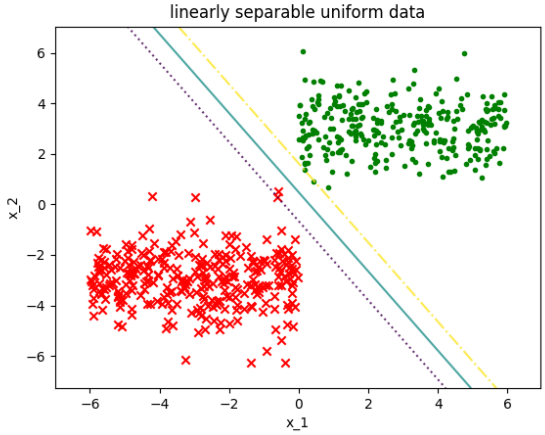


2、使用线性 svm 对(一) 2 中生成的数据进行分类，并画出分类界面

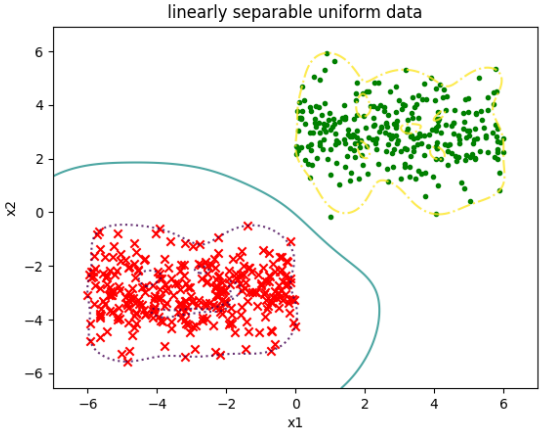


3、使用 svm 对(一) 3 中生成的数据进行分类，并画出分类界面

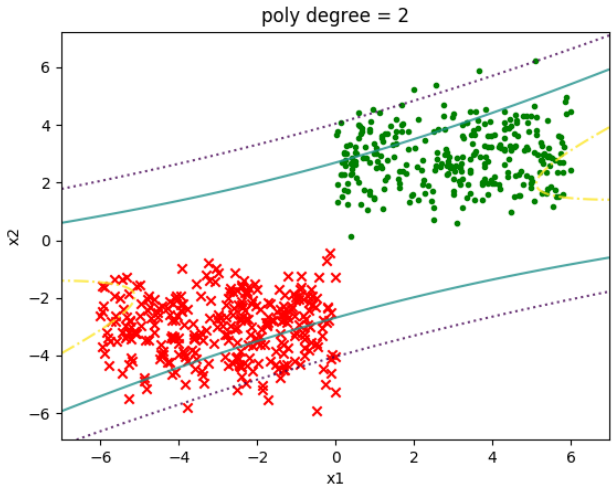
（1）使用线性svm

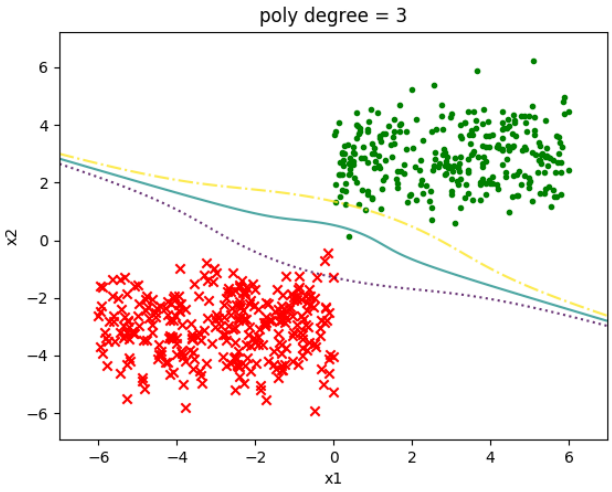


（2）使用rbf核



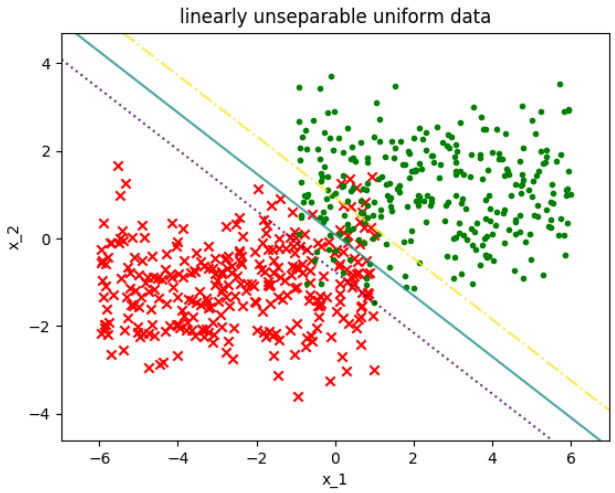
（3）使用多项式核



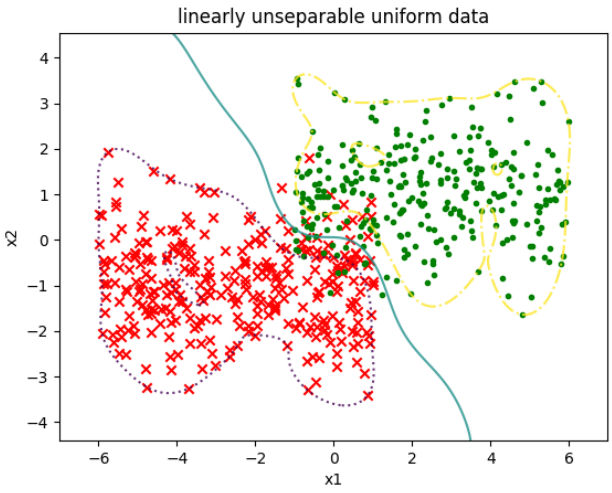


4、使用 svm 对(一) 4 中生成的数据进行分类，并画出分类界面

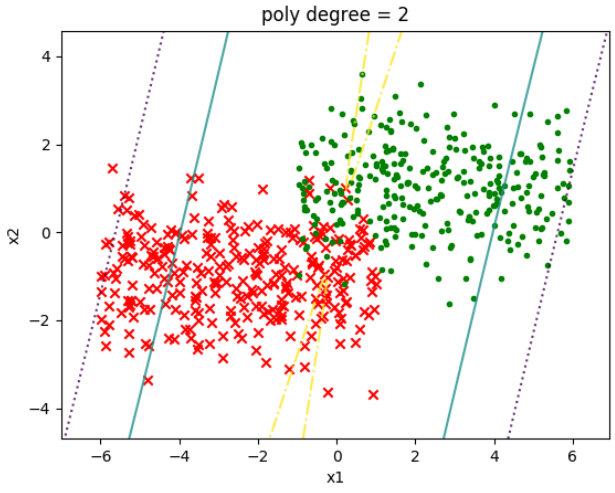
（1）使用线性svm

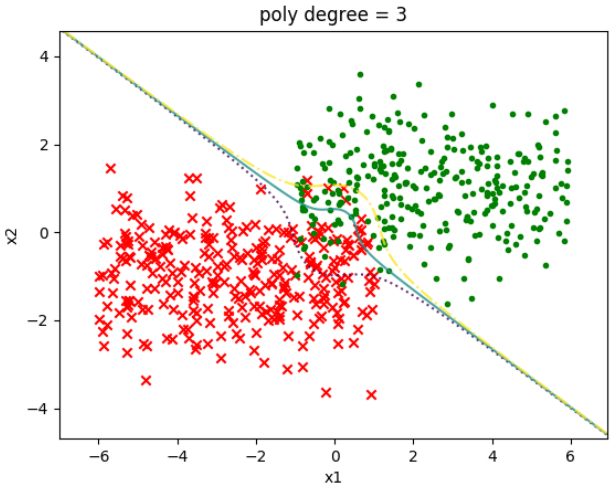


（2）使用rbf核

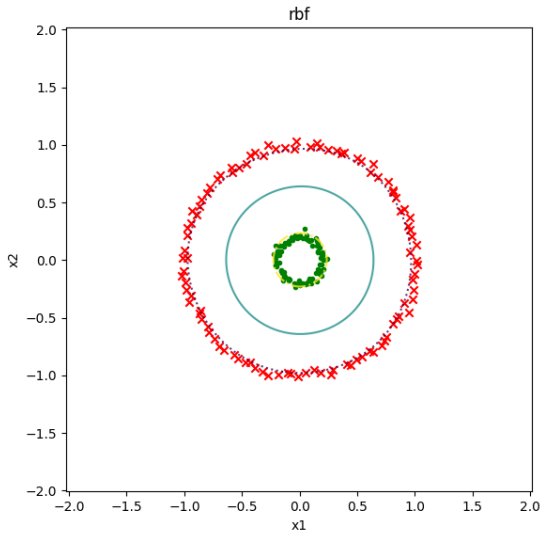


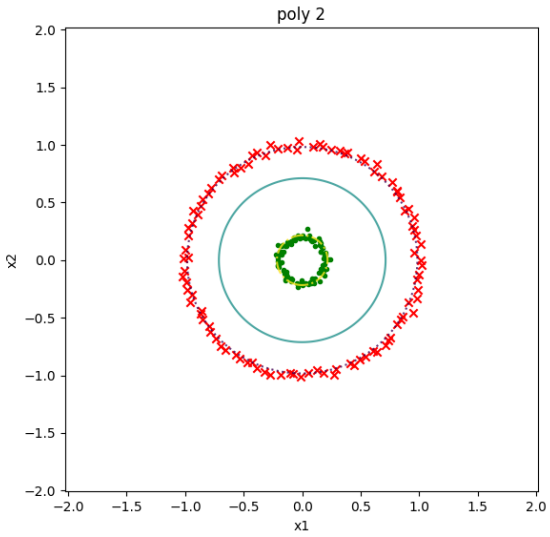
（3）使用多项式核

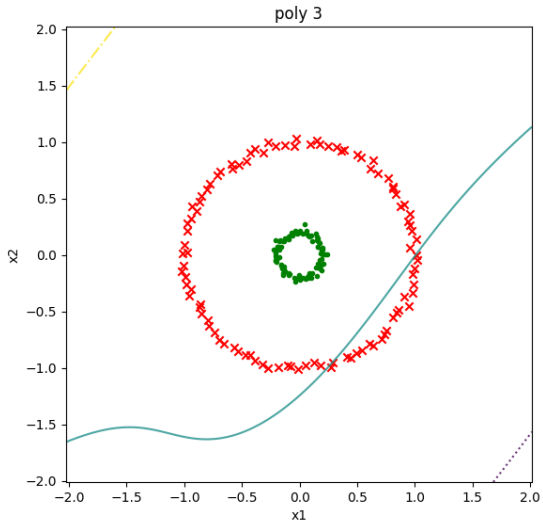


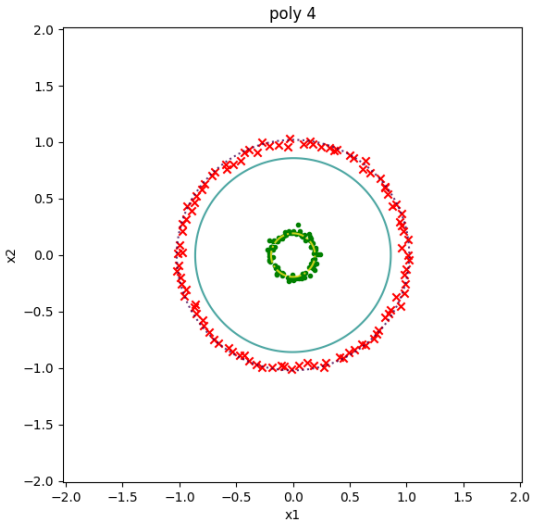


5、使用 svm 对(一) 5 中生成的数据进行分类，并画出分类界面









附录：

数据生成代码：data_generater.py

```
#python3
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles
#线性可分均匀数据
def linear_sep(fig=True):
    x1 = np.random.uniform(-1, 4, 100)#numpy.random.uniform(low,high,size)
    x2 = np.random.uniform(-2, 4, 100)
    y1 = [10 for x in x1] + np.random.normal(0,1,100)
    y2 = [2 for x in x2] + np.random.normal(0,1,100)
    x = np.array(list(zip(x1,y1))+list(zip(x2,y2)))
    y = np.array([1 if i <100 else 0 for i in range(200)])
    #可视化
    if fig==True:
        fig = plt.figure()
        ax1 = fig.add_subplot(1, 1, 1)
        ax1.set_title('linearly separable uniform data')
        ax1.set_xlabel('x1')
        ax1.set_ylabel('x2')
        ax1.scatter(x1, y1, c='g', marker='.')
        ax1.scatter(x2, y2, c='r', marker='x')
        plt.show()
    return x,y
#线性不可分均匀数据
def linear_unsep(fig=True):
    x1 = np.random.uniform(-1, 4, 100)
    x2 = np.random.uniform(-2, 4, 100)
    y1 = [4 for x in x1] + np.random.normal(0,1,100)
    y2 = [2 for x in x2] + np.random.normal(0,1,100)
    x = np.array(list(zip(x1, y1)) + list(zip(x2, y2)))
    y = np.array([1 if i < 100 else 0 for i in range(200)])
    #可视化
    if fig==True:
        fig = plt.figure()
        ax1 = fig.add_subplot(1, 1, 1)
        ax1.set_title('linearly unseparable uniform data')
        ax1.set_xlabel('x1')
        ax1.set_ylabel('x2')
        ax1.scatter(x1, y1, c='g', marker='.')
        ax1.scatter(x2, y2, c='r', marker='x')
        plt.show()
    return x,y
#线性可分高斯数据
def guassian_sep(fig=True):
    x1 = np.random.uniform(0, 6, 300)
    x2 = np.random.uniform(-6, 0, 300)
    y1 =  np.random.normal(3, 1, 300)
    y2 =  np.random.normal(-3, 1, 300)
    x = np.array(list(zip(x1, y1)) + list(zip(x2, y2)))
    y = np.array([1 if i < 300 else 0 for i in range(600)])
    #可视化数据
    if fig==True:
        fig = plt.figure()
        ax1 = fig.add_subplot(1, 1, 1)
```

```
        ax1.set_title('linearly separable guassian data')
        ax1.set_xlabel('x1')
        ax1.set_ylabel('x2')
        ax1.scatter(x1, y1, c='g', marker='.')
        ax1.scatter(x2, y2, c='r', marker='x')
        plt.show()
    return x, y
#线性不可分高斯数据
def guassian_unsep(fig=True):
    x1 = np.random.uniform(-1, 6, 300)
    x2 = np.random.uniform(-6, 1, 300)
    y1 =  np.random.normal(1, 1, 300)
    y2 =  np.random.normal(-1, 1, 300)
    x = np.array(list(zip(x1, y1)) + list(zip(x2, y2)))
    y = np.array([1 if i < 300 else 0 for i in range(600)])
    #可视化
    if fig==True:
        fig = plt.figure()
        ax1 = fig.add_subplot(1, 1, 1)
        ax1.set_title('linearly unseparable guassian data')
        ax1.set_xlabel('x1')
        ax1.set_ylabel('x2')
        ax1.scatter(x1, y1, c='g', marker='.')
        ax1.scatter(x2, y2, c='r', marker='x')
        plt.show()
    return x, y
#环状数据
def circle(fig=True):
    x, y = make_circles(200,shuffle=True, noise=0.02, factor=0.2)
    x1 = np.array([x[i] for i in range(200) if y[i] == 0])
    x2 = np.array([x[i] for i in range(200) if y[i] == 1])
    #可视化
    if fig==True:
        fig = plt.figure()
        ax1 = fig.add_subplot(1, 1, 1)
        ax1.set_title('circle data')
        ax1.set_xlabel('x1')
        ax1.set_ylabel('x2')
        ax1.scatter(x1[:,0], x1[:,1], c='g', marker='.')
        ax1.scatter(x2[:, 0], x2[:, 1], c='r', marker='x')
        plt.show()
    return x, y
```

svm主要代码：

```
#python3
# -*- coding: utf-8 -*-
import numpy as np
from sklearn import svm
import matplotlib.pyplot as plt
import data_generater

def linear_kernel(x,y,sep_title):
        # kernel='linear'时，为线性核，C越大分类效果越好，但有可能会过拟合（defaul C=1）。
        # kernel='rbf'时（default），为高斯核，gamma值越小，分类界面越连续；gamma值越大，分类界面越“散”，分类效果越好，但有可能会过拟合。
        # kernel='poly'时，多项式函数,degree 表示多项式的程度-----支持非线性分类。更高gamma值，将尝试精确匹配每一个训练数据集，可能会导致泛化误差和引起过度拟合问题。
        # kernel='sigmoid'时，支持非线性分类。更高gamma值，将尝试精确匹配每一个训练数据集，可能会导致泛化误差和引起过度拟合问题。
        model = svm.SVC(kernel='linear')
        model.fit(x, y)

        x_0, x_1 = x[:, 0], x[:, 1]
        x0_min, x0_max = x_0.min() - 1, x_0.max() + 1
        x1_min, x1_max = x_1.min() - 1, x_1.max() + 1
        xx, yy = np.meshgrid(np.arange(x0_min, x0_max, 0.01),
                             np.arange(x1_min, x1_max, 0.01))
        Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])
        Z = Z.reshape(xx.shape)

        x1 = np.array([x[i] for i in range(len(x)) if y[i] == 0])
        x2 = np.array([x[i] for i in range(len(x)) if y[i] == 1])
        fig = plt.figure()
        ax1 = fig.add_subplot(1, 1, 1)
        ax1.set_title('linearly '+sep_title+' uniform data')
        ax1.set_xlabel('x_1')
        ax1.set_ylabel('x_2')
        ax1.scatter(x1[:, 0], x1[:, 1], c='r', marker='x')
        ax1.scatter(x2[:, 0], x2[:, 1], c='g', marker='.')
        ax1.contour(xx, yy, Z, levels=[-1, 0, 1], linestyles=[':', '-', '-.'], alpha=0.8)
        plt.show()

def guassian_kernel(x,y,sep_title):
        model = svm.SVC(kernel='rbf', gamma=0.5)
        model.fit(x, y)

        x_0,x_1 = x[:,0], x[:,1]
        x0_min, x0_max = x_0.min() - 1, x_0.max() + 1
        x1_min, x1_max = x_1.min() - 1, x_1.max() + 1
        xx, yy = np.meshgrid(np.arange(x0_min, x0_max, 0.01),
                             np.arange(x1_min, x1_max, 0.01))
        Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])
        Z = Z.reshape(xx.shape)

        x1 = np.array([x[i] for i in range(len(x)) if y[i] == 0])
        x2 = np.array([x[i] for i in range(len(x)) if y[i] == 1])
        fig = plt.figure()
        ax1 = fig.add_subplot(1, 1, 1)
        ax1.set_title('linearly '+sep_title+' uniform data')
        ax1.set_xlabel('x1')
```

```
        ax1.set_ylabel('x2')
        ax1.scatter(x1[:, 0], x1[:, 1], c='r', marker='x')
        ax1.scatter(x2[:, 0], x2[:, 1], c='g', marker='.')
        ax1.contour(xx, yy, Z, levels=[-1,0,1],linestyles=[':', '-', '-.'],alpha=0.8)
        plt.show()

def poly_kernel(x,y,sep_title):
    for poly_degree in range(2,4):
            # poly_degree = 3
            model = svm.SVC(kernel='poly', degree=poly_degree)
            model.fit(x, y)

            x_0, x_1 = x[:, 0], x[:, 1]
            x0_min, x0_max = x_0.min() - 1, x_0.max() + 1
            x1_min, x1_max = x_1.min() - 1, x_1.max() + 1
            xx, yy = np.meshgrid(np.arange(x0_min, x0_max, 0.01),
                                 np.arange(x1_min, x1_max, 0.01))
            Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])
            Z = Z.reshape(xx.shape)

            x1 = np.array([x[i] for i in range(len(x)) if y[i] == 0])
            x2 = np.array([x[i] for i in range(len(x)) if y[i] == 1])
            fig = plt.figure()
            ax1 = fig.add_subplot(1, 1, 1)
            ax1.set_title('poly degree = '+str(poly_degree))
            ax1.set_xlabel('x1')
            ax1.set_ylabel('x2')
            ax1.scatter(x1[:, 0], x1[:, 1], c='r', marker='x')
            ax1.scatter(x2[:, 0], x2[:, 1], c='g', marker='.')
            ax1.contour(xx, yy, Z, levels=[-1, 0, 1], linestyles=[':', '-', '-.'], alpha=0.8)
            plt.show()

def circle(x,y,sep_title):
    kernels = ['rbf','poly']
    for ker in kernels:
        if ker=='rbf':
                model = svm.SVC(kernel='rbf', gamma=0.7)

                model.fit(x, y)

                x_0, x_1 = x[:, 0], x[:, 1]
                x0_min, x0_max = x_0.min() - 1, x_0.max() + 1
                x1_min, x1_max = x_1.min() - 1, x_1.max() + 1
                xx, yy = np.meshgrid(np.arange(x0_min, x0_max, 0.01),
                                                 np.arange(x1_min, x1_max, 0.01))
                Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])
                Z = Z.reshape(xx.shape)

                x1 = np.array([x[i] for i in range(len(x)) if y[i] == 0])
                x2 = np.array([x[i] for i in range(len(x)) if y[i] == 1])
                fig = plt.figure()
                ax1 = fig.add_subplot(1, 1, 1)
                ax1.set_title('rbf')
                ax1.set_xlabel('x1')
                ax1.set_ylabel('x2')
                ax1.scatter(x1[:, 0], x1[:, 1], c='r', marker='x')
                ax1.scatter(x2[:, 0], x2[:, 1], c='g', marker='.')
                ax1.contour(xx, yy, Z, levels=[-1, 0, 1], linestyles=[':', '-', '-.'], alpha=0.8)
                plt.show()
        elif ker=='poly':
```

```
                for deg in range(2,5):
                        model = svm.SVC(kernel='poly', degree=deg)

                        model.fit(x, y)

                        x_0, x_1 = x[:, 0], x[:, 1]
                        x0_min, x0_max = x_0.min() - 1, x_0.max() + 1
                        x1_min, x1_max = x_1.min() - 1, x_1.max() + 1
                        xx, yy = np.meshgrid(np.arange(x0_min, x0_max, 0.01),
                                             np.arange(x1_min, x1_max, 0.01))
                        Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])
                        Z = Z.reshape(xx.shape)

                        x1 = np.array([x[i] for i in range(len(x)) if y[i] == 0])
                        x2 = np.array([x[i] for i in range(len(x)) if y[i] == 1])
                        fig = plt.figure()
                        ax1 = fig.add_subplot(1, 1, 1)
                        ax1.set_title('poly '+str(deg))
                        ax1.set_xlabel('x1')
                        ax1.set_ylabel('x2')
                        ax1.scatter(x1[:, 0], x1[:, 1], c='r', marker='x')
                        ax1.scatter(x2[:, 0], x2[:, 1], c='g', marker='.')
                        ax1.contour(xx, yy, Z, levels=[-1, 0, 1], linestyles=[':', '-', '-.'], alpha=0.8)
                                plt.show()

if __name__ == '__main__':
        sep=True
        destribute = 'circle'
        if sep==True and destribute=='linear':
                x,y = data_generater.linear_sep() #生成数据
                sep_title = 'separable'

        elif sep==False and destribute=='linear':
                x,y = data_generater.linear_unsep() #生成数据
                sep_title = 'unseparable'

        elif sep==True and destribute=='gauss':
                x,y = data_generater.guassian_sep() #生成数据
                sep_title = 'separable'

        elif sep==False and destribute=='gauss':
                x,y = data_generater.guassian_unsep() #生成数据
                sep_title = 'unseparable'

        elif destribute=='circle':
                x, y = data_generater.circle()
                sep_title = 'separable'

        linear_kernel(x,y,sep_title)
        guassian_kernel(x,y,sep_title)
        poly_kernel(x,y,sep_title)
        circle(x,y,sep_title)
```