МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОННОЙ ТЕХНИКИ

Институт системной и программной инженерии   
и информационных технологий (Институт СПИНТех)

Лабораторная работа № 3

по курсу «Интеллектуальные системы»

Статистические методы. Метод опорных векторов.

Выполнил:

Залесский О.А. гр. ПИН-12М

Проверил преподаватель:

проф., д.ф.-м. н. Рычагов М.Н.

Москва, МИЭТ, 2023

**Загрузка и визуализация данных**

mat = scipy.io.loadmat('ex3data1.mat')

X = np.array(mat["X"])

y = np.array(mat["y"])

def plotData(X, y):

# Указание: Реализуйте функцию, которая будет визуализировать набор данных

y = y.ravel()

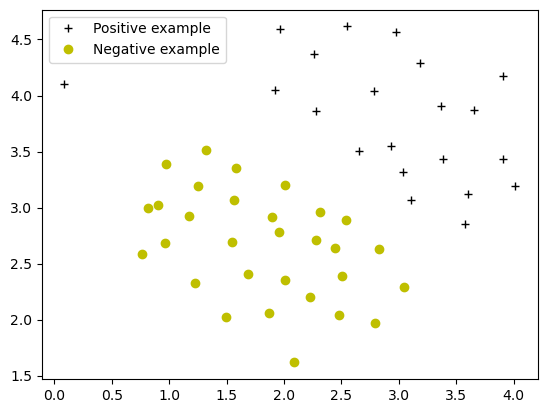
plt.plot(X[:,0][y == 1], X[:,1][y == 1], "k+", label = 'Positive example')

plt.plot(X[:,0][y == 0], X[:,1][y == 0], "yo", label = 'Negative example')

plt.legend()

plt.show()

plotData(X, y)



## Обучение линейного классификатора SVM

## def svmTrain(X, y, C, kernelFunction, sigma = None, tol=1e-3, max\_passes=-1):

## y = y.flatten()

## if kernelFunction == "gaussian\_rbf":

## clf = svm.SVC(C = C, kernel="rbf", tol=tol, max\_iter=max\_passes, verbose=True)

## return clf.fit(gaussianKernelGramMatrix(X,X, sigma), y)

## else:

## clf = svm.SVC(C = C, kernel=kernelFunction, tol=tol, max\_iter=max\_passes, verbose=True)

## return clf.fit(X, y)

def visualizeBoundaryLinear(X, y, model, c):

w = model.coef\_[0]

b = model.intercept\_[0]

xp = np.linspace(X[:,0].min(), X[:,0].max(), 100)

yp = - ((w[0] \*xp) + b) / w[1]

plt.plot(xp, yp, linewidth = 3, color = 'blue', label='Граница классов')

plt.legend(fontsize=15)

plt.title('\n C = {:d} \n'.format(c), fontsize=20)

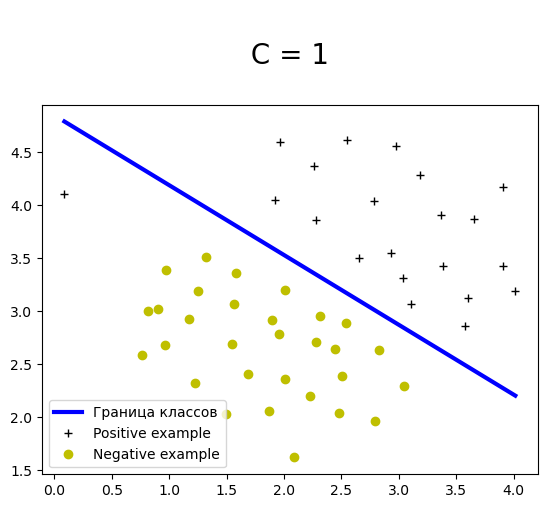
plotData(X, y)

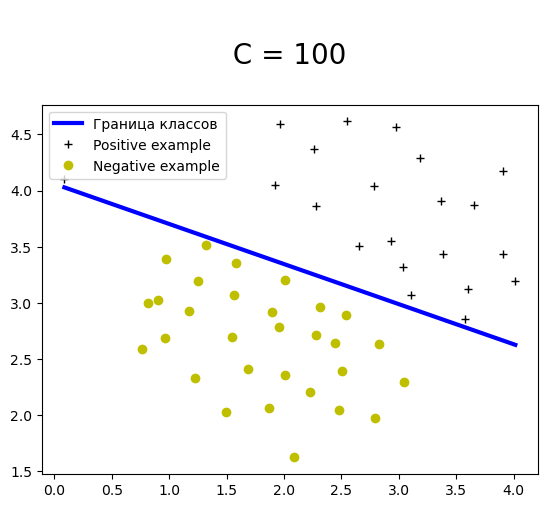
C = [1, 100, 1000]

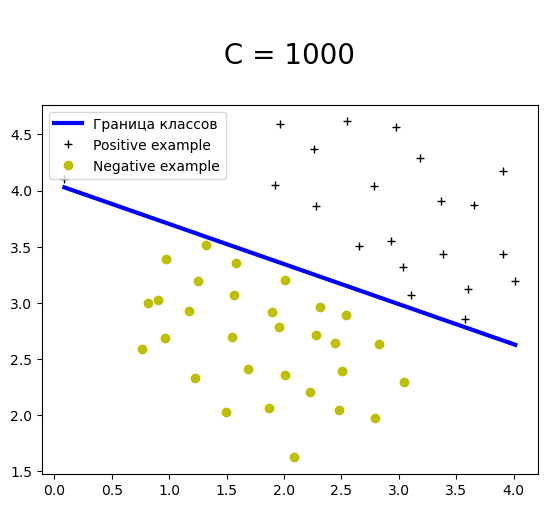
for c in C:

model = svmTrain(X, y, c, "linear")

visualizeBoundaryLinear(X, y, model, c)







При коэффициенте С = 1000 портится классификация, так область границы между опорными векторами сужается, что противоречит идее метода.

## Применение радиальной базисной функции (ядра) Гаусса

def gaussianKernel(x1, x2, sigma):

'''

Под ядром Гаусса подразумевается функция, определяющая сходство пары образцов на основании оценки расстояния между ними.

Возвращаемой величиной является переменная sim

Следует определить векторы x1 и x2 как векторы-столбцы

# Указание: Запрограммируйте функцию, табулирующую близость векторов x1 и x2, вычисляя значение ядра Гаусса, с параметром sigma

sim = np.exp(-np.sum(np.square(x1 - x2)) / (2 \* (sigma\*\*2)))

return sim

def gaussianKernelGramMatrix(X1, X2, sigma):

gram\_matrix = np.zeros((X1.shape[0], X2.shape[0]))

for i, x1 in enumerate(X1):

for j, x2 in enumerate(X2):

gram\_matrix[i, j] = gaussianKernel(x1, x2, sigma)

return gram\_matrix

x1 = np.array([1, 2, 1])

x2 = np.array([0, 4, -1])

sigma = 2

sim = gaussianKernel(x1, x2, sigma)

print("Gaussian Kernel between x1 =", x1, ", x2 =", x2, ", sigma =", sigma, ":\n{:f}".format(sim))

Gaussian Kernel between x1 = [1 2 1] , x2 = [ 0 4 -1], sigma = 2 :

0.324652

## Визуализация обучающего набора 2

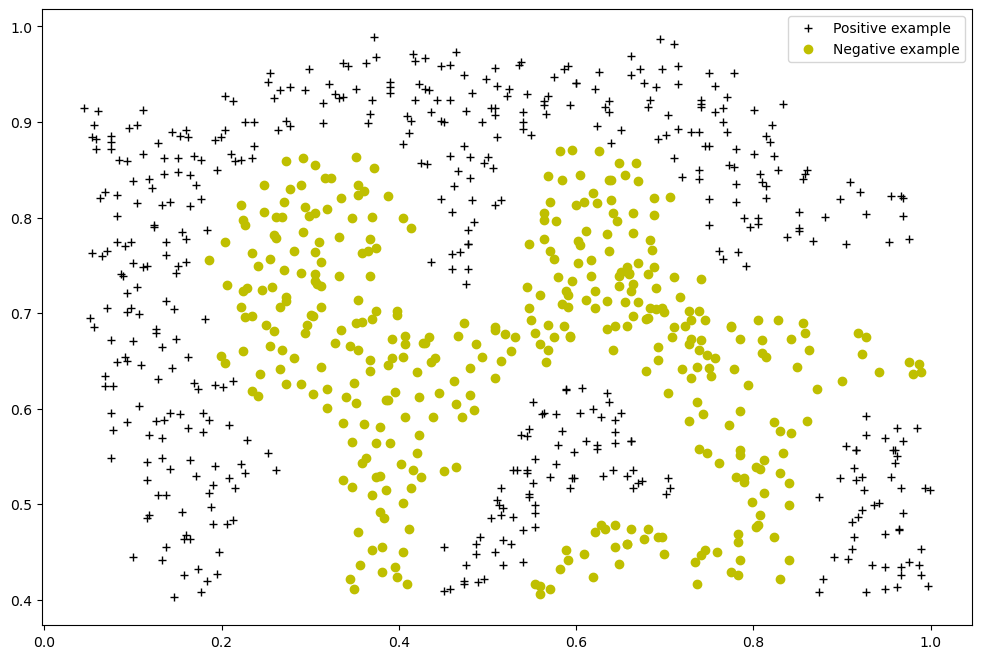
mat = scipy.io.loadmat('ex3data2.mat')

X = np.array(mat["X"])

y = np.array(mat["y"])

plt.figure(figsize=(12,8))

plotData(X, y)



## Обучение SVM с радиальной базисной функцией Гаусса (Набор данных 2)

def visualizeBoundary(X, y, model, sigma, varargin=0):

x1plot = np.linspace(X[:,0].min(), X[:,0].max(), 100).T

x2plot = np.linspace(X[:,1].min(), X[:,1].max(), 100).T

X1, X2 = np.meshgrid(x1plot, x2plot)

vals = np.zeros(X1.shape)

for i in range(X1.shape[1]):

this\_X = np.column\_stack((X1[:, i], X2[:, i]))

vals[:, i] = model.predict(gaussianKernelGramMatrix(this\_X, X, sigma))

plt.contour(X1, X2, vals, colors="blue", levels=[0], linewidth=10)

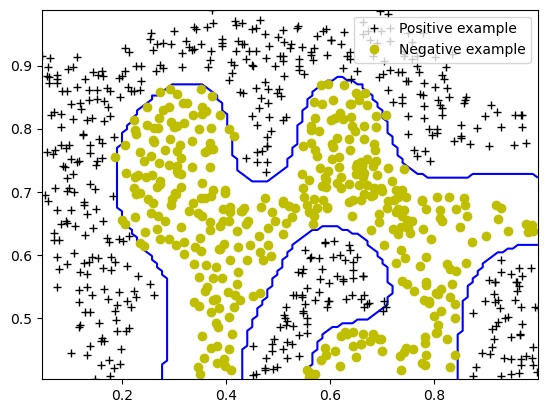
plotData(X,y)

C = 1

sigma = 0.1

model = svmTrain(X, y, C, "gaussian\_rbf", sigma)

visualizeBoundary(X, y, model, sigma)



## Визуализация обучающего набора 3

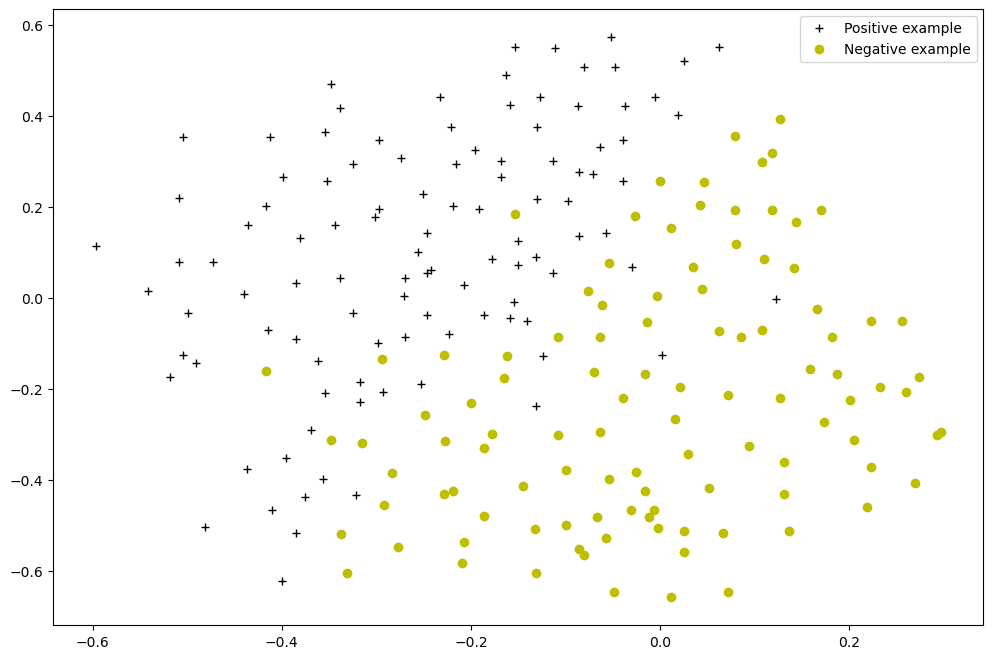
mat = scipy.io.loadmat('ex3data3.mat')

X = np.array(mat["X"])

y = np.array(mat["y"])

plt.figure(figsize=(12,8))

plotData(X, y)



def dataset3Params(X, y, Xval, yval):

'''

DATASET3PARAMS возвращает искомые параметры C и sigma для третьей части упражнения, в котором требуется определить оптимальные значения (C, sigma) для эффективного использования SVM с некоторой радиальной базисной функцией (например, c гауссовским ядром)

Следует запрограммировать функцию, используя метод перекрестной проверки (кросс-валидация).

Указание: необходимо также рассчитать ошибку для набора данных, выбранных для проверки. Ошибка определяет долю примеров для перекрестной проверки, классифицированных неправильно.

'''

values = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]

predictions = dict()

for C in values:

for sigma in values:

model = svmTrain(X, y, C, 'gaussian\_rbf', sigma, tol = 1e-5, max\_passes = -1)

prediction = model.predict(gaussianKernelGramMatrix(Xval, X, sigma))

predictions[(C, sigma)] = np.mean((prediction != yval))

C, sigma = min(predictions, key=predictions.get)

return C, sigma

## Обучение SVM с радиальной базисной функцией Гаусса (Набор данных 3)

Xval = np.array(mat["Xval"])

yval = np.array(mat["yval"])

# Задание: Определить оптимальные параметры С и σ, используя метод перекрестной проверки с помощью множества Хval, yval.

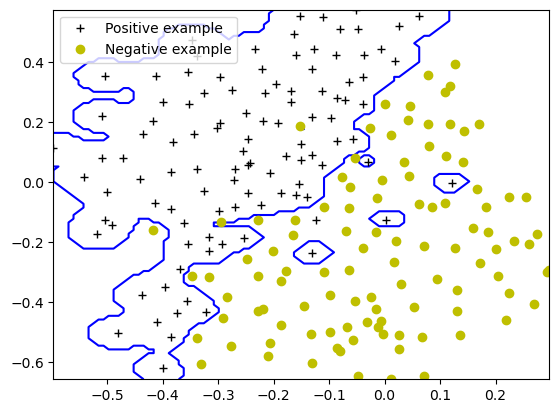
C, sigma = dataset3Params(X, y, Xval, yval)

print("Best parameters are C={:.2f}, sigma={:.2f}".format(C, sigma))

Best parameters are C=1.00, sigma=0.03

model = svmTrain(X, y, C, "gaussian\_rbf", sigma=sigma)

visualizeBoundary(X, y, model, sigma)



(Такая нелинейная граница скорее всего исходит из оценки точности классификации, так как она имеет меньший процент ошибки)