Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Компьютерных сетей и систем

Кафедра Информатики

ЛАБОРАТОРНАЯ РАБОТА №2
«Логистическая регрессия. Многоклассовая классификация»

БГУИР 1-40 81 04

Магистрант:
гр. 858642
Кукареко А.В.

Проверил:
Стержанов М. В.

Минск, 2019

ХОД РАБОТЫ

Задание.

Набор данных ex2data1.txt представляет собой текстовый файл, содержащий информацию об оценке студента по первому экзамену (первое число в строке), оценке по второму экзамену (второе число в строке) и поступлении в университет (0 - не поступил, 1 - поступил).

Набор данных ex2data2.txt представляет собой текстовый файл, содержащий информацию о результате первого теста (первое число в строке) и результате второго теста (второе число в строке) изделий и результате прохождения контроля (0 - контроль не пройден, 1 - контроль пройден).

Набор данных ex2data3.mat представляет собой файл формата *.mat (т.е. сохраненного из Matlab). Набор содержит 5000 изображений 20x20 в оттенках серого. Каждый пиксель представляет собой значение яркости (вещественное число). Каждое изображение сохранено в виде вектора из 400 элементов. В результате загрузки набора данных должна быть получена матрица 5000x400. Далее расположены метки классов изображений от 1 до 9 (соответствуют цифрам от 1 до 9), а также 10 (соответствует цифре 0).

1. Загрузите данные ex2data1.txt из текстового файла.
2. Постройте график, где по осям откладываются оценки по предметам, а точки обозначаются двумя разными маркерами в зависимости от того, поступил ли данный студент в университет или нет.
3. Реализуйте функции потерь J(θ) и градиентного спуска для логистической регрессии с использованием векторизации.
4. Реализуйте другие методы (как минимум 2) оптимизации для реализованной функции стоимости (например, Метод Нелдера — Мида, Алгоритм Бройдена — Флетчера — Гольдфарба — Шанно, генетические методы и т.п.). Разрешается использовать библиотечные реализации методов оптимизации (например, из библиотеки scipy).
5. Реализуйте функцию предсказания вероятности поступления студента в зависимости от значений оценок по экзаменам.
6. Постройте разделяющую прямую, полученную в результате обучения модели. Совместите прямую с графиком из пункта 2.
7. Загрузите данные ex2data2.txt из текстового файла.
8. Постройте график, где по осям откладываются результаты тестов, а точки обозначаются двумя разными маркерами в зависимости от того, прошло ли изделие контроль или нет.

9. Постройте все возможные комбинации признаков x1 (результат первого теста) и x2 (результат второго теста), в которых степень полинома не превышает 6, т.е. 1, x1, x2, x12, x1x2, x22, …, x1x25, x26 (всего 28 комбинаций).
10. Реализуйте L2-регуляризацию для логистической регрессии и обучите ее на расширенном наборе признаков методом градиентного спуска.
11. Реализуйте другие методы оптимизации.
12. Реализуйте функцию предсказания вероятности прохождения контроля изделием в зависимости от результатов тестов.
13. Постройте разделяющую кривую, полученную в результате обучения модели. Совместите прямую с графиком из пункта 7.
14. Попробуйте различные значения параметра регуляризации λ. Как выбор данного значения влияет на вид разделяющей кривой? Ответ дайте в виде графиков.
15. Загрузите данные ex2data3.mat из файла.
16. Визуализируйте несколько случайных изображений из набора данных. Визуализация должна содержать каждую цифру как минимум один раз.
17. Реализуйте бинарный классификатор с помощью логистической регрессии с использованием векторизации (функции потерь и градиентного спуска).
18. Добавьте L2-регуляризацию к модели.
19. Реализуйте многоклассовую классификацию по методу “один против всех”.
20. Реализуйте функцию предсказания класса по изображению с использованием обученных классификаторов.
21. Процент правильных классификаций на обучающей выборке должен составлять около 95%.
22. Ответы на вопросы представьте в виде отчета.

Результат выполнения:

1. Загрузите данные ex2data1.txt из текстового файла.

```
df1 = pandas.read_csv('ex2data1.txt', header=None, names=['exam_1', 'exam_2', 'result'])
```

```
df1.head()
```

	exam_1	exam_2	result
0	34.623660	78.024693	0
1	30.286711	43.894998	0
2	35.847409	72.902198	0
3	60.182599	86.308552	1
4	79.032736	75.344376	1

2. Постройте график, где по осям откладываются оценки по предметам, а точки обозначаются двумя разными маркерами в зависимости от того, поступил ли данный студент в университет или нет.

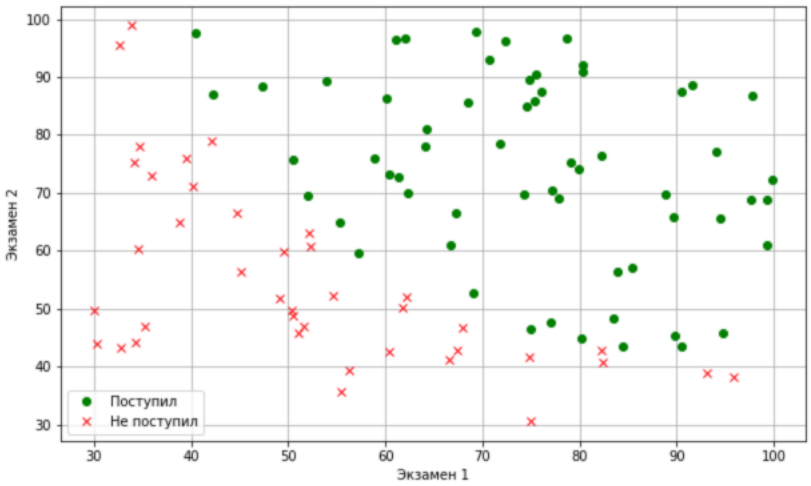

Рисунок 1 - график зависимости поступления студентов от их оценок.

3. Реализуйте функции потерь J(θ) и градиентного спуска для логистической регрессии с использованием векторизации.

Функция гипотеза:

```
def sigmoid(z):
    return 1.0 / (1 + np.exp(-z))

def h(theta, X):
    return sigmoid(np.dot(X, theta))
```

Функция потерь (или стоимости):

```
def J(theta, X, Y):
    m = len(X)

    h_res = h(theta, X)

    e1 = np.dot(-Y.T, np.log(h_res))
    e2 = np.dot((1 - Y).T, np.log(1 - h_res))

    return (1 / m) * (e1 - e2).item()
```

Функция градиентного спуска:

```
def gd(X, Y, iterations = 200, alpha = 0.01):
    n = X.shape[1]
    m = len(Y)

    theta = np.zeros([n, 1])

    j_hist = []

    for i in range(iterations):
        h_res = h(theta, X)

        dt = np.dot(X.T, (h_res - Y))

        theta = theta - (alpha / m) * dt

        j_hist.append(J(theta, X, Y))

    return theta, np.asarray(j_hist)
```

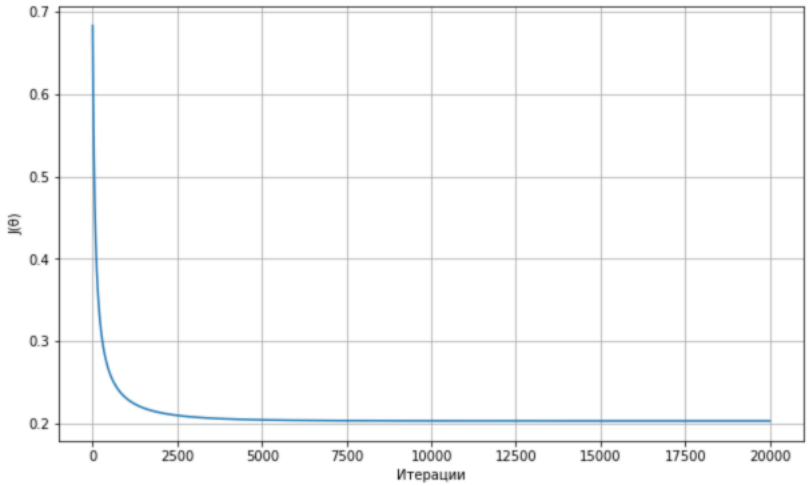

Рисунок 2 – график функции стоимости.

4. Реализуйте другие методы (как минимум 2) оптимизации для реализованной функции стоимости (например, Метод Нелдера — Мида, Алгоритм Бройдена - Флетчера - Гольдфарба - Шанно, генетические методы и т.п.). Разрешается использовать библиотечные реализации методов оптимизации (например, из библиотеки scipy).

Функция алгоритма «Нелдера – Мида»:

```
def nelder_mead(X, Y):
    result = optimize.minimize(J, x0=np.zeros([X.shape[1], 1]), args=(X, Y), method='Nelder-Mead')
    return result.x, result.fun
```

Функция алгоритма «Бройдена - Флетчера - Гольдфарба - Шанно»:

```
def bfgs(X, Y):
    result = optimize.minimize(J, x0=np.zeros([X.shape[1], 1]), args=(X, Y), method='BFGS')
    return result.x, result.fun
```

5. Реализуйте функцию предсказания вероятности поступления студента в зависимости от значений оценок по экзаменам.

Функция предсказания вероятности поступления студента:

```
def predict(theta, x):
    new_x = np.insert(norm_func(x), 0, 1, axis=0).reshape(1, -1)
    return (h(theta, new_x) >= 0.5).astype(int)

predict(gd_theta, [70, 55]).item()
```

```
1
```

6. Постройте разделяющую прямую, полученную в результате обучения модели. Совместите прямую с графиком из пункта 2.

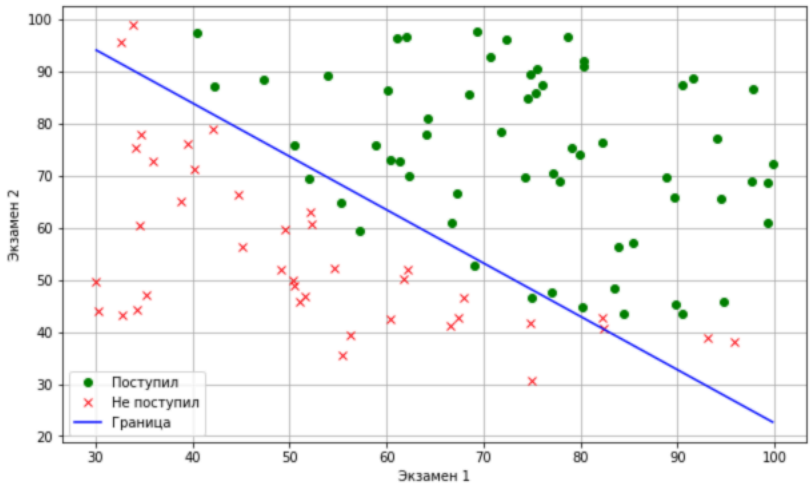

Рисунок 3 – график разделяющей прямой и данных из задания 2.

7. Загрузите данные ex2data2.txt из текстового файла.

```
ex2_df = pandas.read_csv('ex2data2.txt', header=None, names=['test_1', 'test_2', 'result'])
ex2_df.head()
```

	test_1	test_2	result
0	0.051267	0.69956	1
1	-0.092742	0.68494	1
2	-0.213710	0.69225	1
3	-0.375000	0.50219	1
4	-0.513250	0.46564	1

8. Постройте график, где по осям откладываются результаты тестов, а точки обозначаются двумя разными маркерами в зависимости от того, прошло ли изделие контроль или нет.

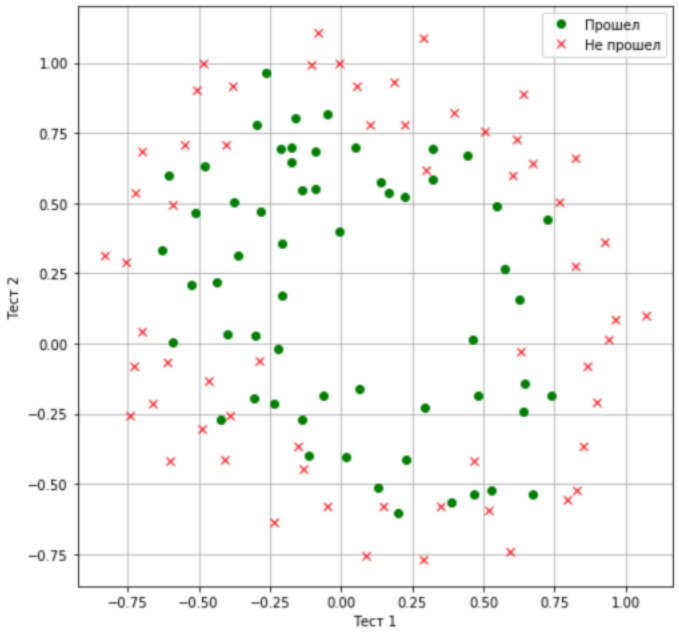

Рисунок 4 – график показывающий прохождение контроля изделием по двум тестам.

9. Постройте все возможные комбинации признаков x1 (результат первого теста) и x2 (результат второго теста), в которых степень полинома не превышает 6, т.е. 1, x1, x2, x12, x1x2, x22, …, x1x25, x26 (всего 28 комбинаций).

Функция для построения полинома:

```
def gen_polynom_matrix(X, degrees):
    # первый столбик с единицами
    m = len(X)
    result = np.ones([m, 1])

    for i in range(1, degrees+1):
        for j in range(0, i+1):
            x1 = X[:,0] ** (i-j)
            x2 = X[:,1] ** (j)
            new_column = (x1 * x2).reshape(m, 1)
            result = np.hstack((result, new_column ))

    return result
```

Матрица построенная функцией:

```
X2_new = gen_polynom_matrix(X2, 6)
X2_new.shape
```

```
(118, 28)
```

10. Реализуйте L2-регуляризацию для логистической регрессии и обучите ее на расширенном наборе признаков методом градиентного спуска.

Функция градиентного спуска с L2-регуляризацией:

```
: def gd_step(theta, X, Y, lam=0):
      m = len(Y)
      h_res = h(theta, X)

      res = (1./m) * np.dot(X.T, (h_res - Y))

      res[1:] = res[1:] + (lam / m) * theta[1:]

      return res
```

11. Реализуйте другие методы оптимизации.

Функция алгоритма «Бройдена - Флетчера - Гольдфарба - Шанно» с L2-регуляризацией:

```
def bfgs_cust_reg(X, Y, l=0):
    init_theta = np.zeros([X.shape[1], 1]).flatten()

    result = optimize.minimize(J_reg, x0=init_theta, args=(X, Y.flatten(), l), method='BFGS', jac=gd_step)
    return result.x, result.fun
```

12. Реализуйте функцию предсказания вероятности прохождения контроля изделием в зависимости от результатов тестов.

Функция предсказания вероятности прохождения контроля изделием:

```
def predict_2(theta, x):
    return (h(theta, x) >= 0.5).astype(int)

print(f'Should equal to 1 - {predict_2(gd_reg_theta, [-0.25, 0.5]).item()}')
print(f'Should equal to 0 - {predict_2(gd_reg_theta, [1, 1]).item()}')
```

```
Should equal to 1 - 1
Should equal to 0 - 0
```

13. Постройте разделяющую кривую, полученную в результате обучения модели. Совместите прямую с графиком из пункта 7.

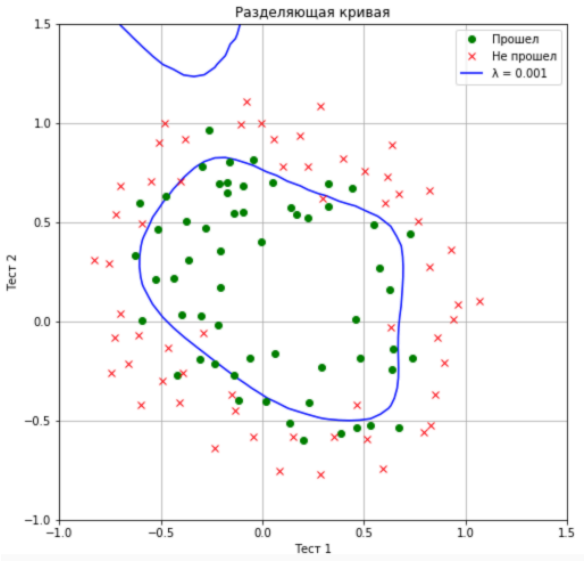

Рисунок 5 – график разделяющей кривой совмещенный с исходными данными (lambda=0.1)

14. Попробуйте различные значения параметра регуляризации λ. Как выбор данного значения влияет на вид разделяющей кривой? Ответ дайте в виде графиков.

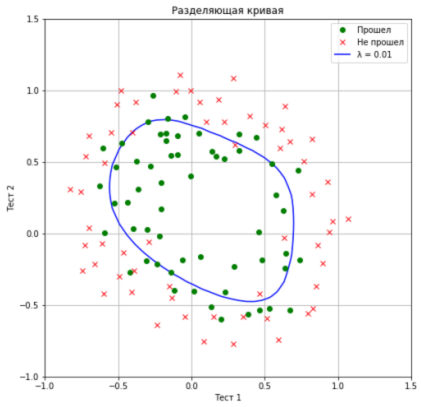

Рисунок 6 – график разделяющей кривой совмещенный с исходными данными (lambda=0.01)

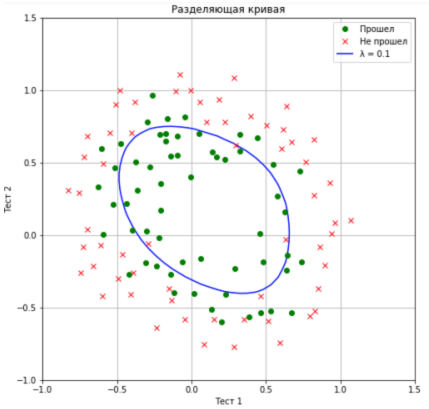

Рисунок 7 – график разделяющей кривой совмещенный с исходными данными (lambda=0.1)

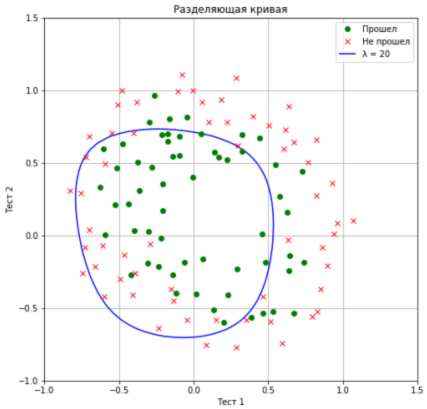

Рисунок 8 – график разделяющей кривой совмещенный с исходными данными (lambda=20)

По графику видно, что при приближении к lambda=1 разделяющая прямая становиться более скругленной, но как результат, контур становится меньше, и в него попадает меньше правильных данных. Увеличение lambda до 20 вообще сместило контур и сильно ухудшило качество модели.

15. Загрузите данные ex2data3.mat из файла.

```
img_data = scipy.io.loadmat('ex2data3.mat')
```

```
ex_3_X, ex_3_Y = img_data['X'], img_data['y']
```

```
ex_3_X.shape
```

```
(5000, 400)
```

16. Визуализируйте несколько случайных изображений из набора данных. Визуализация должна содержать каждую цифру как минимум один раз.

Рисунок 9 – визуализация данных из набора ex2data3.mat.

17. Реализуйте бинарный классификатор с помощью логистической регрессии с использованием векторизации (функции потерь и градиентного спуска).

Реализация бинарного классификатора была взята из предыдущих задания.

18. Добавьте L2-регуляризацию к модели.

Функция алгоритма «сопряженных градиентов» с L2-регуляризацией:

```
def fmin_cg_alg(X, Y, lam=0):
    init_theta = np.zeros([X.shape[1], 1]).flatten()
    result = optimize.fmin_cg(J_reg, fprime=gd_step, x0=init_theta, args=(X, Y.flatten(), lam), maxiter=50, disp=False,
    return result[0], result[1]
```

19. Реализуйте многоклассовую классификацию по методу “один против всех”.

Реализация многоклассовой классификации по методу “один против всех”.

```
def train_classifier(X, Y, lam=0):
    m = X.shape[0]
    n = X.shape[1]

    classes_count = 10

    thetas = np.zeros([classes_count, n])

    for klass in range(classes_count):
        class_index = klass if klass else 10 # 10 - это 0
        print(f'{klass} => {class_index}')
        replaced_Y = (Y == class_index).astype(int)
        #theta, cost = fmin_cg_alg(X, replaced_Y, lam)  # lam - 0.001 -> 95.6%
        theta, cost = bfgs_cust_reg(X, replaced_Y, lam) # lam - 0.001 -> 97.2%
        thetas[klass] = theta

    return thetas
```

20. Реализуйте функцию предсказания класса по изображению с использованием обученных классификаторов.

Функция предсказания класса по изображению:

```
def predClass(thetas, x):
    return np.argmax(h(thetas.T, x))
```

```
i = 3000
x = ex_3_X_ext[i]
y = ex_3_Y[i].item()

print(f'Class = {y}, predicted = {predClass(thetas, x)}')
```

```
Class = 6, predicted = 6
```

21. Процент правильных классификаций на обучающей выборке должен составлять около 95%.

```
def calc_accuracy(thetas, X, Y):
    m = X.shape[0]
    correct = 0

    for i in range(m):
        pred = predClass(thetas, X[i])
        pred = pred if pred else 10

        if pred == Y[i]:
            correct += 1

    return correct/m

print("Accuracy: %0.1f%%"%(100*calc_accuracy(thetas, ex_3_X_ext, ex_3_Y)))
```

```
Accuracy: 97.2%
```

Получена точность: 97,2%

Вывод.

В ходе выполнения лабораторной работы я изучил логистическую регрессию, изучил способы её реализации на языке python, попробовал готовые решения для оптимизации из пакета “оптимизаций” библиотеки scipy. Во второй части лабораторной работы был изучен алгоритм L2 регуляризации и его влияние на качество модели. В третьей части лабораторной работы я изучил алгоритм «один против всех» для многоклассовой классификации, реализовал его и получил точность 97.2% для набора картинок mnist.