Лабораторная работа №4: «Нейронные сети»

Набор данных **ex4data1.mat** (такой же, как в лабораторной работе №2) 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).

Задание.

1. Загрузите данные **ex4data1.mat** из файла.
2. Загрузите веса нейронной сети из файла **ex4weights.mat**, который содержит две матрицы $\Theta^{(1)}$ (25, 401) и $\Theta^{(2)}$ (10, 26). Какова структура полученной нейронной сети?
3. Реализуйте функцию прямого распространения с сигмоидом в качестве функции активации.
4. Вычислите процент правильных классификаций на обучающей выборке. Сравните полученный результат с логистической регрессией.
5. Перекодируйте исходные метки классов по схеме one-hot.
6. Реализуйте функцию стоимости для данной нейронной сети.
7. Добавьте L2-регуляризацию в функцию стоимости.
8. Реализуйте функцию вычисления производной для функции активации.
9. Инициализируйте веса небольшими случайными числами.
10. Реализуйте алгоритм обратного распространения ошибки для данной конфигурации сети.
11. Для того, чтобы удостоверится в правильности вычисленных значений градиентов используйте метод проверки градиента с параметром $\varepsilon = 10^{-4}$.
12. Добавьте L2-регуляризацию в процесс вычисления градиентов.
13. Проверьте полученные значения градиента.
14. Обучите нейронную сеть с использованием градиентного спуска или других более эффективных методов оптимизации.
15. Вычислите процент правильных классификаций на обучающей выборке.
16. Визуализируйте скрытый слой обученной сети.
17. Подберите параметр регуляризации. Как меняются изображения на скрытом слое в зависимости от данного параметра?
18. Ответы на вопросы представьте в виде отчета.

Реализация:

In[1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pnd
```

1.1 Загрузите набор данных ex4data1.mat из файла.

In[2]:

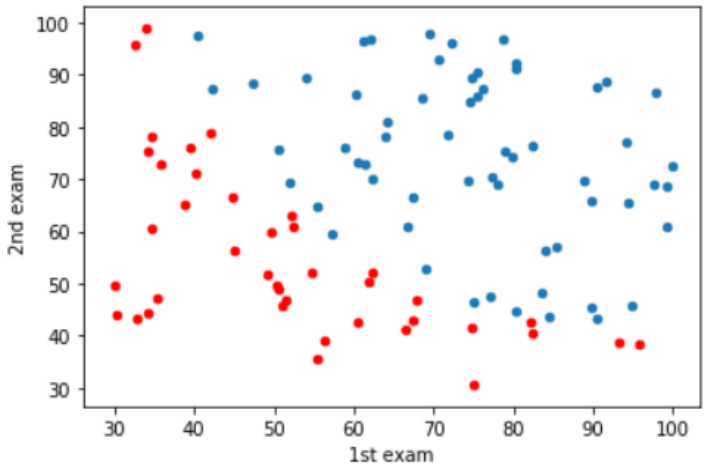
```
mat = loadmat('data/ex4data1.mat')
x_train, y_train = mat['X'], mat['y']
y_train = y_train.reshape(y_train.shape[0])

# replace all 10 to 0
y_train = np.where(y_train == 10, 0, y_train)
```

1.2 Загрузите веса нейронной сети из файла ex4weights.mat, который содержит две матрицы Θ(1) (25, 401) и Θ(2) (10, 26). Какова структура полученной нейронной сети?

In[3]:

```
weights = loadmat('data/ex4weights.mat')
theta1 = weights['Theta1']
theta2 = weights['Theta2']
```

Входной слой: 400 нейронов, Промежуточный слой: 25 нейронов, Выходной слой: 10 нейронов. theta0 - bias unit



1.3 Реализуйте функцию прямого распространения с сигмоидом в качестве функции активации

In[4]:

```
s_L = [400, 25, 10]

def sigmoid(z):
    return 1 / (1 + np.e ** (-z))

def insert_ones(x):
    if len(x.shape) == 1:
        return np.insert(x, 0, 1)
    return np.column_stack((np.ones(x.shape[0]), x))

def forward_propagation(x, thetas, cache=False):
    cur_activation = x.copy()
    activations = [cur_activation]

    for theta_i in thetas:
        temp_a = insert_ones(cur_activation)
        z_i = theta_i.dot(temp_a.T).T
        cur_activation = sigmoid(z_i)
        if cache:
            activations.append(cur_activation)

    return activations if cache else cur_activation

def unroll(weights):
    result = np.array([])

    for theta in weights:
        result = np.concatenate((result, theta.flatten()))

    return result

def roll(weights):
    weights = np.array(weights)
    thetas = []
    left = 0

    for i in range(len(s_L) - 1):
        x, y = s_L[i + 1], s_L[i] + 1
        right = x*y
        thetas.append(weights[left:left + right].reshape(x, y))
        left = right

    return thetas
```

1.4 Вычислите процент правильных классификаций на обучающей выборке. Сравните полученный результат с логистической регрессией.

In[5]:

```
def accuracy(predicted, y):
    correct_result_count = np.count_nonzero(predicted.argmax(axis=1) - y)
    return 1 - correct_result_count / y.shape[0]

weights = [theta1, theta2]
predicted = forward_propagation(x_train, weights)
accuracy(predicted, y_train)
```

Out[5]:

```
0.0011999999999999789
```

Как видно из результатов нейронная сеть не обучена. В случае логистической регрессии точность была выше 95%.

1.5 Перекодируйте исходные метки классов по схеме one-hot.

In[6]:

```
# |class|    |class1|class2|
# |1    | => |1     |0     |
# |2    |    |0     |1     |
In [47]:
def one_hot(y, classes_count=10):
    y_extended = np.zeros((len(y), classes_count))

    for i, y_i in enumerate(y):
        y_extended[i][y_i] = 1

    return y_extended
In [48]:
y_one_hot = one_hot(y_train)
```

1.6. Реализуйте функцию стоимости для данной нейронной сети

$$J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}log(h_\Theta(x^{(i)})) - (1-y^{(i)})log(1-(h_\Theta(x^{(i)}))]$$

In[7]:

```
def cost_func(X, y, weights):
    total_cost = 0
    K = y.shape[1]
    hyp = forward_propagation(X, weights)
    for k in range(K):
        y_k, hyp_k = y[:, k], hyp[:, k]
        cost_trues =  y_k * np.log(hyp_k)
        cost_falses = (1 - y_k) * np.log(ONE - hyp_k)
        cost = cost_trues + cost_falses
        total_cost += cost
    return -total_cost.sum() / y.shape[0]
```

1.7 Добавьте L2-регуляризацию в функцию стоимости.

$$J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}log(h_\Theta(x^{(i)})) - (1-y^{(i)})log(1-(h_\Theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\Theta_j^2$$

In[8]:

```
def cost_func_regularized(X, y, weights, reg_L=1):
    weights = roll(weights)
    reg = 0
    cost = cost_func(X, y, weights)

    for theta in weights:
        theta_R = theta[:, 1:]
        reg += (theta_R ** 2).sum()

    return cost + (reg_L / 2 / y.shape[0]) * reg
```

1.8 Реализуйте функцию вычисления производной для функции активации

In[9]:

```
def activation_der(act):
    return act * (1 - act)
```