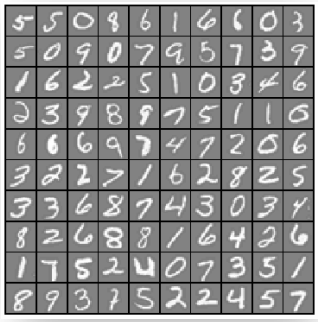
Распознавание рукописных цифр с помощью нейронной сети



В этой работе необходимо реализовать нейронную сеть прямого распространения для распознавания рукописных цифр. Обучение нейросети уже выполнено, поэтому следует использовать уже рассчитанные параметры модели. В процессе выполнения работы нужно реализовать алгоритм расчета значения сети, используя эти параметры для определения того, какая цифра изображена.

Введение в нейронные сети

Нейронная сеть представляет собой сеть, состоящую из соединенных между собой нескольких нейронов. Каждый нейрон можно представить простой моделью:

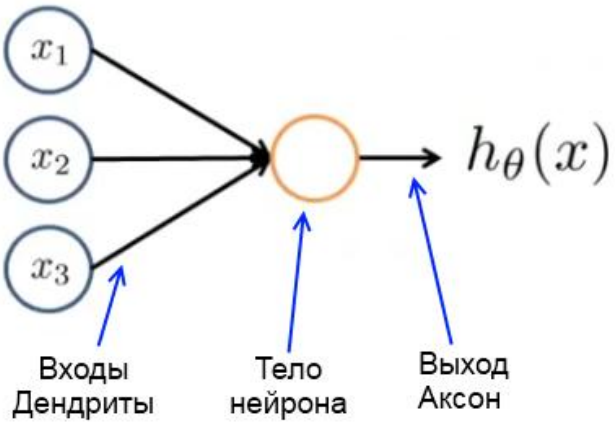


Рисунок 1 Нейрон

Нейрон состоит из нескольких входов – дендритов, тела нейрона и одного выхода – аксона. Значение выхода можно однозначно рассчитать из входов, используя параметры θ – весовые коэффициенты, соединяющие дендриты с телом нейрона. Значение рассчитывается с помощью функции $g(z)$ под названием сигмоид, используемой также в логистической регрессии и методе опорных векторов.

В общем виде:

$$g(z) = \frac{1}{1 + e^{-z}}$$

Или для нейрона:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta x}}$$

Здесь θ – весовые коэффициенты (параметры модели).

x – значения на входах нейрона.

На рисунке 2 приведен возможный вариант нейросети.

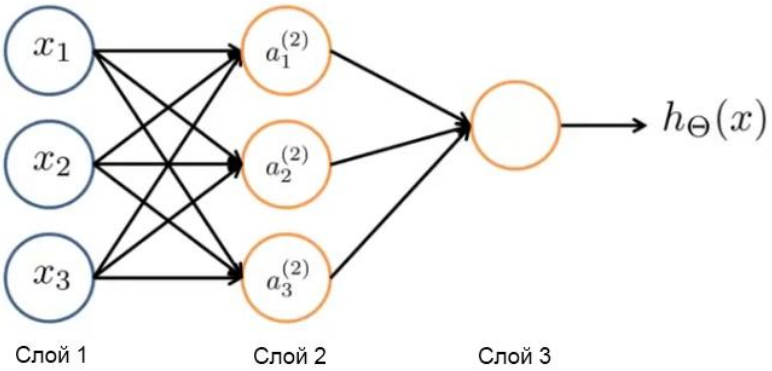


Рисунок 2 Пример простой нейронной сети

Эта сеть состоит из 3 слоев. Первый слой – входной. Состоит из трех нейронов. Второй слой – скрытый. Состоит так же из трех нейронов. Последний слой выходной, состоит из одного единственного нейрона. В случае многоклассовой классификации, как в данной лабораторной работе, в выходном слое может быть несколько нейронов по числу распознаваемых классов.

Сеть прямого распространения (Feedforward)

Нейронная сеть, в которой все связи направлены строго от входных нейронов к выходным, называется сетью прямого распространения (Feedforward). Именно этот вид нейронной сети рассматривается в лабораторной работе. Также такую сеть называют многослойным персептроном (MLP – Multi-Layer Perceptron).

Рассмотрим порядок расчета выходного значения сети по известным значениям входов и параметрам.

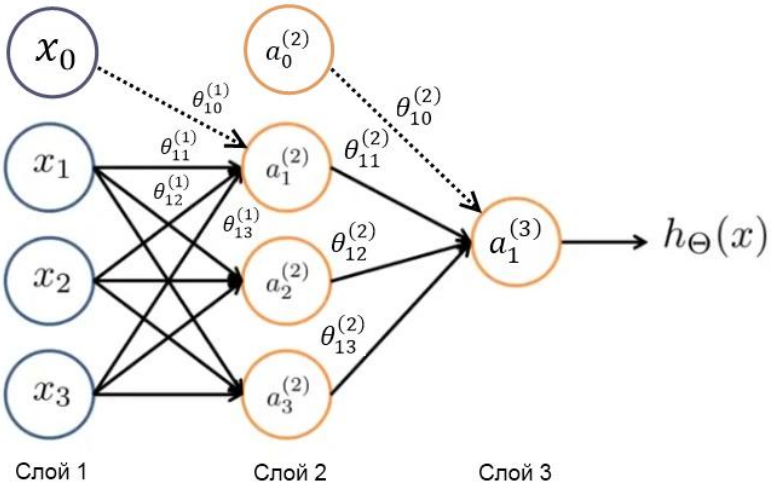


Рисунок 3 Расчет значения нейросети

На каждом слое нейросети (кроме последнего) добавляется по одному нейрону (x_0 и $a_0^{(2)}$), значения которых равны +1. Эти нейроны нужны для того, чтобы возможно было использовать векторизацию вычислений для расчета выходного значения сети.

Для того, чтобы вычислить значение нейрона $a_1^{(2)}$ со второго слоя (см. рисунок 3), нужно воспользоваться формулой:

$$a_1^{(2)} = g\left(\theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3\right)$$

Здесь применяется сигмоид $g(z)$. А в качестве параметра – сумма значений входов, поэлементно перемноженных на параметры модели.

Аналогично можно рассчитать значения на выходе остальных нейронов второго слоя сети:

$$a_2^{(2)} = g\left(\theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \theta_{23}^{(1)}x_3\right)$$

$$a_3^{(2)} = g\left(\theta_{30}^{(1)}x_0 + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3\right)$$

Рисунок 4 поясняет суть индексов в этих формулах:

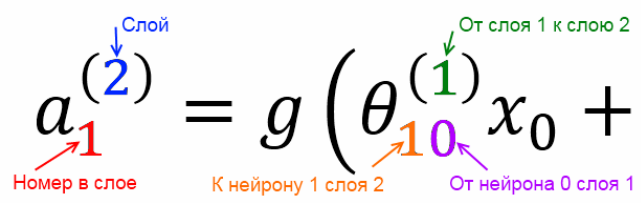


Рисунок 4 Пояснение к индексам в формуле

Используя тот же подход, можно рассчитать и значение на выходе нейросети. Оно будет вычисляться по формуле:

$$h_\theta(x) = a_1^{(3)} = g\left(\theta_{10}^{(2)}a_0^{(2)} + \theta_{11}^{(2)}a_1^{(2)} + \theta_{12}^{(2)}a_2^{(2)} + \theta_{13}^{(2)}a_3^{(2)}\right)$$

Векторная реализация расчета выхода нейронной сети

На практике чаще всего используют векторную (основанную на операциях с векторами и матрицами) реализацию этого алгоритма. Векторная реализация проще и эффективнее с точки зрения производительности, так как во многих языках существуют высокопроизводительные реализации библиотек для матричных вычислений.

Рассмотрим набор уравнений для расчета значения нейросети, который был представлен выше:

$$a_1^{(2)} = g\left(\theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3\right)$$

$$a_2^{(2)} = g\left(\theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \theta_{23}^{(1)}x_3\right)$$

$$a_3^{(2)} = g\left(\theta_{30}^{(1)}x_0 + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3\right)$$

$$h_\theta(x) = a_1^{(3)} = g\left(\theta_{10}^{(2)}a_0^{(2)} + \theta_{11}^{(2)}a_1^{(2)} + \theta_{12}^{(2)}a_2^{(2)} + \theta_{13}^{(2)}a_3^{(2)}\right)$$

Введем матрицу $\theta^{(1)}$, которая описывает весовые коэффициенты для расчета значений слоя 2 из входных значений слоя 1:

$$\theta^{(1)} = \begin{bmatrix} \theta_{10}^{(1)} & \theta_{11}^{(1)} & \theta_{12}^{(1)} & \theta_{13}^{(1)} \\ \theta_{20}^{(1)} & \theta_{21}^{(1)} & \theta_{22}^{(1)} & \theta_{23}^{(1)} \\ \theta_{30}^{(1)} & \theta_{31}^{(1)} & \theta_{32}^{(1)} & \theta_{33}^{(1)} \end{bmatrix}$$

Тогда значения слоя 2 можно получить, перемножив значения матрицы $\theta^{(1)}$ на значения входного слоя x:

$$a^{(2)} = \begin{bmatrix} a_0^{(2)} = 1 \\ a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \end{bmatrix} = g\left(\begin{bmatrix} \theta_{10}^{(1)} & \theta_{11}^{(1)} & \theta_{12}^{(1)} & \theta_{13}^{(1)} \\ \theta_{20}^{(1)} & \theta_{21}^{(1)} & \theta_{22}^{(1)} & \theta_{23}^{(1)} \\ \theta_{30}^{(1)} & \theta_{31}^{(1)} & \theta_{32}^{(1)} & \theta_{33}^{(1)} \end{bmatrix} \times \begin{bmatrix} x_0 = 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = g\left(\begin{bmatrix} \theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3 \\ \theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \theta_{23}^{(1)}x_3 \\ \theta_{30}^{(1)}x_0 + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3 \end{bmatrix}\right)$$

$$= g\left(\theta^{(1)}x\right)$$

Если принять, что $x = a^{(1)}$, то можно записать $a^{(2)} = g\left(\theta^{(1)}a^{(1)}\right)$. Функция $g(z)$ применяется к каждому элементу получившегося вектора.

С помощью формулы $a^{(2)} = g\left(\theta^{(1)}a^{(1)}\right)$ можно вычислить значения на выходах нейронов второго слоя. Аналогично можно рассчитать значения на выходе нейрона третьего слоя сети. Пусть у нас имеется матрица $\theta^{(2)}$, описывающая весовые коэффициенты для расчета слоя 3 из слоя 2:

$$\theta^{(2)} = \begin{bmatrix} \theta_{10}^{(2)} & \theta_{11}^{(2)} & \theta_{12}^{(2)} & \theta_{13}^{(2)} \end{bmatrix}$$

Тогда значение нейросети можно вычислить по формуле:

$$h_\theta(x) = a_1^{(3)} = a^{(3)} = g\left(\begin{bmatrix} \theta_{10}^{(2)} & \theta_{11}^{(2)} & \theta_{12}^{(2)} & \theta_{13}^{(2)} \end{bmatrix} \times \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \end{bmatrix} \right)$$

$$= g\left(\theta_{10}^{(2)}a_0^{(2)} + \theta_{11}^{(2)}a_1^{(2)} + \theta_{12}^{(2)}a_2^{(2)} + \theta_{13}^{(2)}a_3^{(2)}\right) = g\left(\theta^{(2)}a^{(2)}\right)$$

Теперь, если собрать всё вместе:

$$h_\theta(x) = g\left(\theta^{(2)}a^{(2)}\right) = g\left(\theta^{(2)}g\left(\theta^{(1)}a^{(1)}\right)\right)$$

Приведенная выше формула позволяет вычислить выход нейросети по известным входам x и параметрам θ.

Внимание! При векторной реализации алгоритма важно помнить о добавлении нулевого нейрона со значением +1 на всех слоях, кроме последнего.

Структура нейросети

На рисунке 5 изображена структура нейросети. Она состоит из входного слоя, скрытого слоя и выходного слоя. Так как размер изображений цифр 20×20 точек, количество входных нейронов сети равно 400 (не считая входа x_0 с постоянным значением +1). Второй (скрытый) слой нейросети состоит из 25 нейронов. Последний выходной слой состоит из 10 нейронов по числу распознаваемых цифр – от 0 до 9.

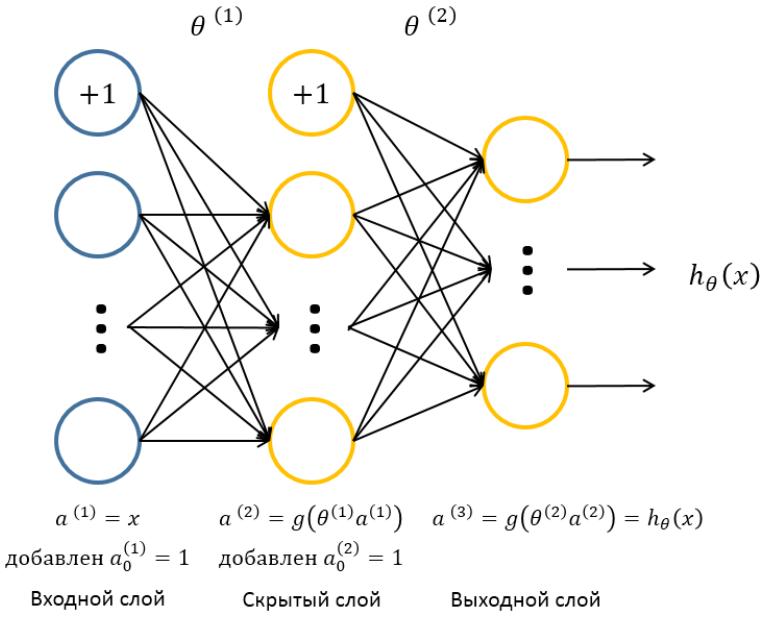


Рисунок 5 Структура нейронной сети распознавания цифр

Описание наборов данных

Тестовая выборка хранится в файле **test_set.mat** и состоит из 5000 рукописных цифр. Для загрузки данных в скрипт используйте функцию **scipy.io.loadmat** с именем файла в качестве параметра. Будет загружена матрица **X** и вектор **y**.

Матрица **X** представляет собой рукописные цифры. Она состоит из 5000 строк по числу тестовых примеров и 400 столбцов по числу точек на изображении. Каждое изображение цифры имеет размеры 20×20 точек, отсюда 400 столбцов в матрице **X**.

Вектор **y** – это вектор-столбец, состоящий из 5000 элементов. Каждый элемент определяет класс распознаваемой цифры. Цифре «0» соответствует класс 10, остальные классы соответствуют своим цифрам.

Параметры обученной нейросети хранятся в файле **weights.mat**. Параметры представляют собой 2 матрицы – **Theta1** и **Theta2**. В данной работе используется трёхслойная сеть, поэтому используются 2 матрицы параметров. Первая матрица содержит коэффициенты, связывающие первый и второй слои. Ее размер 25×401 (25 строк и 401 столбец). Число строк соответствует числу нейронов во втором слое, а число столбцов – числу нейронов в первом входном слое, плюс 1.

Аналогично матрица **Theta2** содержит матрицу коэффициентов, связывающих второй слой с третьим. Размер этой матрицы, исходя из количества нейронов во втором и третьем слоях, составляет 10×26.

$$Theta1 = \begin{bmatrix} \theta_{1,0}^{(1)} & \theta_{1,1}^{(1)} & \dots & \theta_{1,400}^{(1)} \\ \theta_{2,0}^{(1)} & \theta_{2,1}^{(1)} & \dots & \theta_{2,400}^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ \theta_{25,0}^{(1)} & \theta_{25,1}^{(1)} & \dots & \theta_{25,400}^{(1)} \end{bmatrix} \qquad Theta2 = \begin{bmatrix} \theta_{1,0}^{(2)} & \theta_{1,1}^{(2)} & \dots & \theta_{1,25}^{(2)} \\ \theta_{2,0}^{(2)} & \theta_{2,1}^{(2)} & \dots & \theta_{2,25}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ \theta_{10,0}^{(2)} & \theta_{10,1}^{(2)} & \dots & \theta_{10,25}^{(2)} \end{bmatrix}$$

Разъяснение индексов см. на рисунке 4.

Порядок выполнения работы

1. Создать рабочее окружение

Создать отдельный каталог для выполнения лабораторной работы и скопировать в него файлы с расширениями **.py** и **.mat** из архива задания. Создать в каталоге файл **run.py**, в который будет записываться код скрипта.

2. Добавить в скрипт код загрузки данных из файлов `test_set.mat` и `weights.mat`

Для загрузки данных из файлов **.mat** используется функция **scipy.io.loadmat**, которой передаётся имя файла. Не забудьте подключить модуль **scipy.io** в начале вашего скрипта.

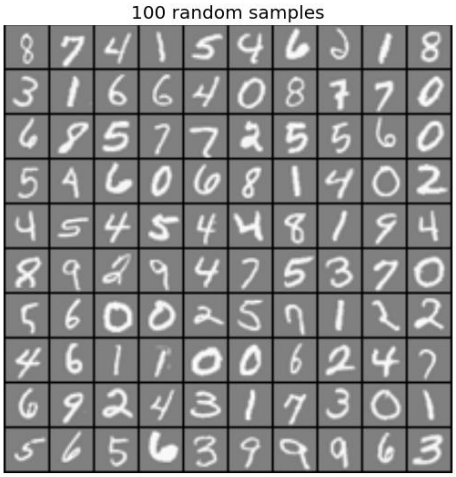
Из файла **test_set.mat** необходимо извлечь матрицу **X** по ключу '**X**' и вектор-столбец **y** по ключу '**y**'. Из файла **weights.mat** необходимо извлечь матрицы весовых коэффициентов **Theta1** и **Theta2** по ключам '**Theta1**' и '**Theta2**' соответственно.

Определите также параметр **m**, который должен равняться числу строк в матрице **X**.

3. Добавить код, выводящий на экран 100 случайных цифр из матрицы X

Для вывода случайных цифр на экран используется функция **displayData** из файла **displayData.py**. Подключите её в начале вашего скрипта. В функцию **displayData** передаются примеры из матрицы **X**.

Чтобы выбрать случайным образом 100 примеров из матрицы **X**, сначала сгенерируйте индексы строк. С помощью вызова функции **numpy.random.permutation(m)** получите массив индексов строк, перемешанных случайным образом. Затем возьмите первые 100 элементов полученного массива и используйте их как индексы примеров из матрицы **X**. Столбцы из матрицы **X** нужно брать все. Если всё сделано правильно, должен получиться следующий рисунок:



4. Дополнить код функции sigmoid в файле sigmoid.py

В файле **sigmoid.py** написать реализацию функции **sigmoid** по формуле

$$g(z) = \frac{1}{1 + e^{-z}}$$

Следует помнить, что в качестве параметра **z** в функцию может быть передан вектор или матрица произвольной размерности.

5. Реализовать вычисление отклика нейронной сети на входной пример

В файле **predict.py** реализовать вычисление отклика нейронной сети. При реализации нужно использовать векторную форму: $h_\theta(x) = g\left(\theta^{(2)}a^{(2)}\right) = g\left(\theta^{(2)}g\left(\theta^{(1)}a^{(1)}\right)\right)$.

Сначала необходимо определить количество примеров в матрице **X**. Количество примеров – это количество строк в **X**.

Затем необходимо добавить к матрице **X** единичный вектор-столбец. Он нужен для векторизации вычислений. Для генерации вектора-столбца используйте функцию **np.ones**. Присоедините вектор-столбец слева к матрице **X**, используя код ниже:

```
a1 = np.c_[ones, X]
```

Здесь **ones** – единичный вектор-столбец.

Вычислите значения на входах второго (скрытого) слоя нейросети ($\theta^{(1)}a^{(1)}$). Для этого нужно перемножить матрицу **a1** на коэффициенты **Theta1**, связывающие 1 и 2 слой нейросети. Матрицу **Theta1** необходимо транспонировать, чтобы перемножение выполнялось корректно. Для перемножения матриц используйте функцию **np.dot**.

Вычислите значения на выходе второго слоя нейросети ($a^{(2)}$ или $g\left(\theta^{(1)}a^{(1)}\right)$). Для этого примените активационную функцию – g (сигмоид) – к каждому элементу матрицы, полученной в результате перемножения на предыдущем этапе. После этого добавьте единичный столбец к матрице.

Вычислите значения на выходе нейросети ($h_\theta(x)$). Для этого перемножьте матрицу $a^{(2)}$ на транспонированную матрицу коэффициентов **Theta2** ($\theta^{(2)}$) и примените к каждому элементу полученной матрицы активационную функцию g (сигмоид).

Если вы всё сделали правильно, гипотеза $h_\theta(x)$ должна представлять собой матрицу из 5000 строк (по числу тестовых примеров) и 10 столбцов (по числу распознаваемых цифр). Каждая строка матрицы представляет собой 10 чисел в диапазоне от 0 до 1.

Индекс максимального числа в строке даёт искомую цифру (см. рисунок выше). Например, для первого примера максимальный отклик нейросети на 7 выходе (значение 0.9), значит для этого примера нейросеть предсказала значение «7».

Определите цифры, которая распознала нейросеть. Для этого используйте функцию **np.argmax**. В качестве дополнительного параметра укажите **axis=1**, тогда максимумы будут вычисляться не по столбцам, а по строкам. Прибавьте 1 к массиву, который вернёт функция **np.argmax**, так как распознаваемые цифры смещены относительно индексов в массиве на 1 (см. рисунок выше). Верните из функции итоговый массив с результатами предсказания.

6. Оценить точность распознавания

В файле **run.py** дополнить код так, чтобы оценивалось качество распознавания. Под качеством понимается отношение верно распознанных цифр к общему размеру выборки, умноженное на 100%.

Чтобы выполнить задание, необходимо сначала для тестовой выборки получить результат предсказания нейросети, а затем сравнить его с исходными фактическими значениями из выборки. Для получения предсказания нейросети используйте код:

```
pred = predict(Theta1, Theta2, X)
```

predict – это функция, которую вы модифицировали в задании 5. **Theta1** и **Theta2** – весовые коэффициенты обученной нейросети, которые вы ранее загрузили. **X** – матрица тестовых примеров.

Затем необходимо сравнить результаты предсказания сети с реальными значениями. Чтобы сравнить массив **pred** со значениями вектора-столбца **y**, необходимо сначала y преобразовать к одномерному массиву. Для этого используйте метод **y.ravel()**. Сравнение выполняется оператором **==**. Результат выполнения данного оператора – массив значений **True** и **False**. Чтобы оценить точность, нужно привести значения к типу **double**, передав результат сравнения в функцию **np.double()**, а затем посчитать среднее количество совпавших значений, которое можно вычислить функцией **np.mean**.

Если вы всё сделали правильно, должно получиться значение точности около **97.5%**.

7. Продемонстрировать классификацию для 5 случайных примеров

Используйте приведённый ниже код, чтобы выполнить предсказание для 5 случайных примеров из обучающей выборки:

```
rp = np.random.permutation(m)

plt.figure()
for i in range(5):

    X2 = X[rp[i],:]
    X2 = np.matrix(X[rp[i]])

    pred = predict(Theta1, Theta2, X2.getA())
    pred = np.squeeze(pred)
    pred_str = 'Neural Network Prediction: %d (digit %d)' % (pred, y[rp[i]])
    displayData(X2, pred_str)

    plt.close()
```

8. Вывести примеры, на которых нейронная сеть ошиблась

Выполните предсказание цифр как в задании 6. С помощью функции **np.where** определите индексы примеров, на которых нейросеть ошиблась. Параметром в функцию передаётся условие **pred != y.ravel()**. Функция **np.where** возвращает кортеж, в котором нас интересует только первое значение, поэтому используйте код **np.where(pred != y.ravel())[0]**, чтобы получить индексы примеров, на которых нейронная сеть сработала неверно.

Возьмите первые 100 примеров, а затем отобразите их функцией **displayData**. Если вы всё сделали верно, должна отобразиться приметно такая картинка:

Neural Network fails