Лабораторная работа №8. Обучение нейронной сети.

В данной работе вам предстоит реализовать алгоритм обратного распространения и использовать его для обучения нейронной сети с регуляризацией в задаче распознавания рукописных символов из базы данных MNIST (https://ru.wikipedia.org/wiki/MNIST), которая уже использовалась в предыдущих лабораторных работах. Обучающий набор содержит 5000 изображений размером 20×20 пикселей, каждый пиксел принимает целочисленные значения 0-255. Теоретический материал для данной лабораторной работы можно найти в разделе 5 учебного пособия.

Для распознавания рукописных символов мы будем использовать нейронную сеть следующей архитектуры, показанной на рис. 1.

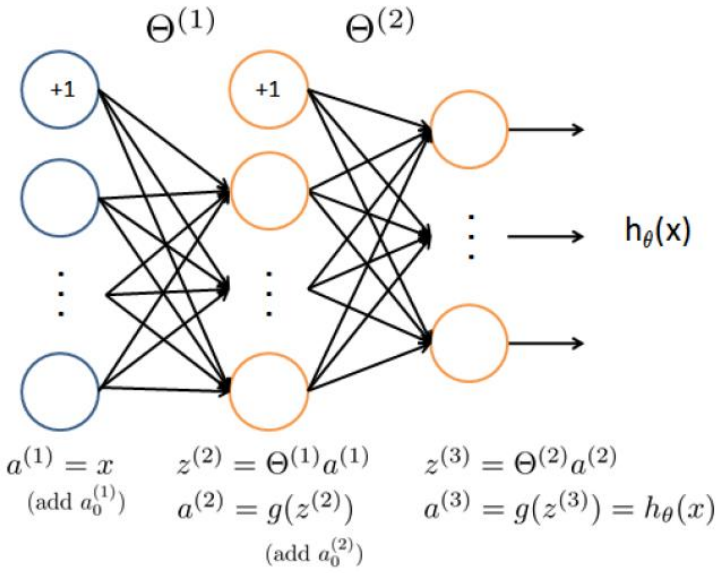

Рис. 1. Архитектура нейронной сети

Сеть состоит из трех слоев. Входной слой $a^{(1)}$ составляют 400 входных параметров X (значения признаков каждого символа). Также у нас присутствует входной параметр $a_0^{(1)}=1$. Второй скрытый слой $a^{(2)}$ составляют 25 нейронов. В скрытом слое присутствует нейрон $a_0^{(2)}=1$. Третий выходной слой $a^{(3)}$ содержит 10 нейронов (по одному на каждый класс).

Десять выходных значений соответствуют десяти распознаваемым классам изображений рукописных символов. Первый нейрон выдает значение вероятности того, что на вход был подан символ ‘0’, второй нейрон – символ ‘1’ и т.д., десятый нейрон соответствует символу ‘9’. Мы будем относить распознаваемое изображение к тому классу, нейрон которого выдал наибольшее значение.

1 Загрузка и визуализация данных

На этом шаге загружается файл ex3data1.npy, который содержит обучающий набор данных в виде матрицы X и вектора y в формате NumPy.

Как и в прошлой лабораторной работе, матрица X размером 5000×400 содержит 5000 изображений. Строка матрицы X длиной 400 элементов хранит изображение 20×20 (всего 400 пикселей) в следующем виде: сначала 20 чисел значений яркости первой строки изображение, затем второй и т.д. до последней строки. Таким образом вместо матрицы 20×20 один обучающий пример задается вектором из 400 параметров.

Вектор y длиной 5000 элементов для каждого обучающего примера X_i хранит

индекс класса. Класс y_i=0 соответствует цифре «0», y_i=1 соответствует цифре «1» и т.д., y_i=9 соответствует цифре «9».

После загрузки выбирается случайных 100 изображений из обучающего набора и передаются в функцию display_data для отображения на экране. Функция принимает следующие параметры:

X – матрица элементов для отображения размером (width×height) строк и (el_width×el_height) столбцов;

width – сколько изображений разместить по горизонтали;

height – сколько изображений разместить по вертикали;

el_width – размер одного изображения по горизонтали;

el_height – размер одного изображения по вертикали.

Найдите функцию в программе и допишите ее содержимое, чтобы она отображала таблицу width×height изображений рукописных цифр на одной картинке.

Когда функция готова, запустите программу, вы должны увидеть изображение из 100 случайных рукописных символов из обучающего набора.

Удалите оператор return после первого шага в функции main, чтобы перейти ко второму шагу.

2 Загрузка весов нейронной сети

На этом шаге выполняется загрузка весов заранее обученной нейронной сети из файла ex3weights.npy. Из файла загружаются матрицы Theta1 (матрица Θ_1 весов между первым и вторым слоем сети) и Theta2 (матрица Θ_2 весов между вторым и третьим слоем сети).

Поскольку второй слой имеет 25 нейронов и получает 400 входных параметров, матрица Theta1 имеет размер 25×401. Третий слой имеет 10 нейронов и получает на вход выходы от 25 нейронов второго слоя, поэтому Theta2 имеет размер 10×26.

Эти предварительно обученные веса мы будем использовать для проверки правильности написания функции стоимости прежде, чем написать процедуру обучения нейронной сети. В дальнейшем для распознавания рукописных символов эти матрицы учитываться использоваться не будут.

Также на этом шаге выполняется преобразование обеих матриц весов нейронной сети в один длинный вектор (сначала в вектор записывается первая строка первой матрицы, затем вторая строка и так далее до конца матрицы, затем точно также после нее в вектор записываются все значения второй матрицы, получается один длинный вектор). Это сделано потому, что библиотечные функции оптимизации, которые мы используем, принимают на вход только одномерный вектор параметров, по которым необходимо выполнить оптимизацию функции. Передать две матрицы произвольного размера нельзя. Поэтому для передачи в функцию оптимизации необходимо преобразовывать матрицы в вектор, а после оптимизации преобразовывать обратно. В программе эти действия уже заложены. Вам не нужно их программировать. Ознакомьтесь, как это выполняется.

Удалите оператор return, чтобы перейти к следующему шагу.

3 Алгоритм прямого распространения

На этом шаге вам необходимо написать процедуру прямого распространения для вычисления отклика нейронной сети и вычисления на его основе значения функции стоимости.

Найдите в программе функцию cost_function. В функции заложено четыре места, которые вы должны будете дополнить своим кодом в процессе выполнения работы. На

данный момент дополните часть в разделе для шага 3. Вы должны вычислить отклик нейронной сети и рассчитать значение функции стоимости *J*.

Функция принимает на вход следующие параметры:

weights – вектор, хранящий матрицы весов Θ_1 и Θ_2, как описано на шаге 2 (вам не нужно самостоятельно выполнять преобразование вектора в две матрицы, в начале функции выполняются необходимые действия и создаются две переменные Theta1 и Theta2);

input_size – размер первого слоя (входного вектора) нейронной сети;

hidden_size – размер скрытого слоя нейронной сети;

num_labels – размер выходного слоя нейронной сети (количество классов);

X – матрица элементов выборки;

y – вектор индексов классов элементов выборки;

lamb – параметр регуляризации.

На выходе функция возвращает значение функции стоимости *J* и вектор частных производных dth (равный по длине вектору weights). На данном шаге вам необходимо запрограммировать только вычисление значения *J*.

В процессе вычисления функции стоимости вам понадобится также значение функции сигмоиды. Найдите в программе функцию sigmoid и дополните необходимым кодом для вычисления сигмоиды от вектора значений.

После того, как функции готовы, запустите программу. Программа вычисляет функцию стоимости без регуляризации для предварительно загруженных весов. Полученное значение должно быть близко к ожидаемому.

Удалите оператор return, чтобы перейти к следующему шагу.

4 Регуляризация в функции стоимости

На данном шаге мы дополним вычисление значения функции стоимости регуляризацией. В функции cost_function внесите в раздел для шага 4 необходимый для этого код. После того, как функция готова, запустите программу. Если все реализовано правильно, то вычисленные значения функции стоимости с регуляризацией должны быть близки к ожидаемым.

Удалите оператор return, чтобы перейти к следующему шагу.

5 Производная функции сигмоиды

На данном шаге вам необходимо реализовать процедуру вычисления производной функции сигмоиды от произвольных значений. Эта функция необходима для вычисления ошибок скрытых слоев в алгоритме обратного распространения.

Найдите функцию sigmoid_gradient и дополните ее нужным кодом. Функция принимает на вход вектор z и должна вернуть вектор g того же размера, в котором каждый элемент g_i равен численному значению производной функции сигмоиды в точке z_i.

После того, как функция готова, запустите программу. Она вызывает написанную вами функцию для вычисления производной сигмоиды в некоторых заданных значениях. Ваши ответы должны быть близки к ожидаемым.

Удалите оператор return, чтобы перейти к следующему шагу.

6 Начальная инициализация весов

Перед обучением нейронной сети необходимо задать начальные значения матриц весов Θ_1 и Θ_2 случайными величинами.

Найдите функцию initialize_weights и дополните ее кодом так, чтобы выполнить начальную инициализацию. Значение каждого элемента матрицы весов рекомендуется

задавать случайным числом в интервале [-0.12, 0.12].

Функция принимает на вход два параметра:

in_layer_size – число нейронов предыдущего слоя сети;

out_layer_size – число нейронов следующего слоя сети.

Функция должна генерировать заполненную случайными числами матрицу размером (out_layer_size)×(in_layer_size+1).

После того, как функция готова, запустите программу. Ваши ответы должны быть близки к ожидаемым. Удалите оператор return, чтобы перейти к следующему шагу.

7 Алгоритм обратного распространения

Теперь имеются все необходимые процедуры для того, чтобы написать алгоритм обратного распространения, который вычисляет частные производные функции стоимости по каждому параметру нейронной сети.

Вернитесь к функции cost_function и в разделе для шага 7 дополните функцию кодом, реализующим данный алгоритм. Учтите, что в этой же функции выше (на шаге 3) уже вычислялись отклики нейронной сети каждого слоя. Вы можете воспользоваться этими значениями и не вычислять их снова. В программе уже создаются нулевые матрицы dth1 и dth2. Заполните эти матрицы правильными значениями частных производных функции стоимости: dth1 по весам Θ_1 и dth2 по весам Θ_2. В программе затем эти две матрицы преобразуются в одномерный вектор dth.

На данном шаге алгоритм обратного распространения вы должны реализовать без регуляризации.

После того, как функция готова, запустите программу. В программе также реализована численная оценка градиента (вам ее программировать не нужно, она уже готова). При помощи численной оценки вычисляются значения градиента параметров. В две колонки выводятся значения градиента, вычисленного вашей процедурой, и посчитанных численным методом. Они должны быть близкими.

Удалите оператор return, чтобы перейти к следующему шагу.

8 Регуляризация параметров нейронной сети

На данном шаге необходимо добавить регуляризацию к градиенту параметров нейронной сети. В функции cost_function в разделе для шага 8 дополните функцию кодом, реализующим регуляризацию при вычислении градиента.

После того, как функция готова, запустите программу. Аналогично предыдущему шагу, в программе численно оцениваются ожидаемые значения градиента с регуляризацией и выводится две колонки значений градиента с регуляризацией, посчитанного вашей функцией, и ожидаемого программой. Они должны быть близкими.

Удалите оператор return, чтобы перейти к следующему шагу.

9 Обучение нейронной сети

К этому моменту вы реализовали все функции, которые требовалось, чтобы обучить нейронную сеть. Далее вызывается процедура оптимизации методом Ньютона (функция fmin_tnc из библиотеки SciPy) для нахождения оптимальных значений весов нейронной сети, доставляющих минимум функции стоимости.

В функцию fmin_tnc передается в качестве аргумента написанная вами функция вычисления стоимости и градиента, инициализированные случайными числами вектор параметров, обучающая выборка (*X* и *y*), параметр регуляризации и другие. Функция возвращает вектор weights, заполненный оптимальными значениями весов нейронной сети, при которых достигается минимум функции стоимости.

10 Распознавание символов при помощи обученной нейронной сети

На данном шаге вам нужно при помощи алгоритма прямого распространения вычислить отклик нейронной сети на входной вектор x и правильно интерпретировать значения выходного вектора $h_\Theta(x)$ для определения цифры, которая отображена на x.

Полученный на предыдущем шаге вектор weights преобразуется в две матрицы Theta1 и Theta2 весов нейронной сети.

Найдите функцию predict и дополните ее кодом для классификации изображений. На вход функция принимает следующие параметры:

X – матрица входных изображений в том же формате, что и обучающая выборка;

Theta1 – матрица весов Θ_1;

Theta2 – матрица весов Θ_2.

Функция должна выдавать вектор p длиной, равной числу строк матрицы X. Значение элемента p_i должно быть равно цифре, изображенной на X_i.

После того, как функция готова, запустите программу. Написанная вами функция predict используется, чтобы получить отклик нейронной сети для всех примеров обучающей выборки и оценки качества модели на обучении. Полученное значение должно быть более 85%.

Удалите оператор return, чтобы перейти к следующему шагу.

11 Предсказание нейронной сетью случайных цифр

На последнем шаге программа использует обученную вами нейронную сеть для классификации случайных изображений из обучающей выборки. Нажмите в командном окне клавишу Enter, будет выбрано случайное изображение из набора данных, показано на экране и классифицировано. Результат классификации (к какому классу нейросеть относит символ) написан в командном окне. Остановить процесс можно вводом символа q и нажатием Enter.

На этом выполнение лабораторной работы №8 завершается.