УДК 004

Нейронная сеть распознавания цифр на JavaScript

Голубь Илья Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Магистрант

Аннотация
В данной работе рассмотрен способ разработки нейронной сети с помощью языка программирования JavaScript библиотеки и Brain.js
Ключевые слова: Нейронная сеть, JavaScript,Brain.js, html, css

Neural network for recognizing numbers in JavaScript

Golub Ilya Sergeevich
Sholom-Aleichem Priamursky State University
Undergraduate

Abstract
In this paper, we consider a method for developing a neural network using the JavaScript programming language and Brain.js
Keywords: neural network: JavaScript,Brain.js, html,css

В реалиях современного мира, когда большую часть инженерных и компьютерных систем приводят к автоматизированному виду, либо же стремятся к тому, что бы автоматизировать процесс. Связанны эти попытки с тем, что компании стараются упростить процесс работы и уменьшить количество людей, которые требуются для полноценного функционирования предприятия. Одним из решений подобных задач является написание нейронной сети.

Нейронные сети это набор алгоритмов, смоделированных в общем виде по человеческому мозгу. Данные алгоритмы предназначены для распознавания закономерностей. С их помощью происходит интерпретация сенсорных данных, посредством своего рода машинного восприятия. Модели, которые используются нейронными сетями для распознания, являются числовыми, они содержатся в векторах. В данные вектора должны быть переведены все реальные данные[1]. Для реализации нейронных сетей в данных момент существует множество библиотек на множестве языков, для демонстрации выберем язык программирования JavaScript.

Целью исследования является показать работу нейронной сети, созданной с помощью языка программирования JavaScript.

Над исследованиями по данной теме занимались следующие авторы. М.А. Афанасьева рассмотрела в своей статье «Создание и обучение нейронных сетей в системе MATLAB» [3]. «Проект создания программной

системы для распознавания графических образов на основе нейронных сетей» был написан В.М. Хачумовым [4]. С.В. Маркова и К.Ю. Жигалов занимались исследованием «Применение нейронной сети для создания системы распознавания изображений» [5]. Описанием основных архитектур нейронных сетей и правил обучения занимались Martin Hagan, Howard Demuth, Mark Beale[2].

Для начала разработки понадобится редактор, в данной работе будет использоваться редактор от MicroSoft – Visual Studio Code.Так же, для более удобной работы можно использовать nodejs и npm. Для описание сети будем использовать Brain.js. Так же, в данной библиотеке есть примеры из документации. Вот, например, эмуляция функции XOR.

```
var brain = require('brain.js');
var net = new brain.NeuralNetwork();

net.train([{input: [0, 0], output: [0]},
           {input: [0, 1], output: [1]},
           {input: [1, 0], output: [1]},
           {input: [1, 1], output: [0]}]);

var output = net.run([1, 0]);  // [0.987]
console.log(output);
```

Рис. 1. XOR

У нас 2 входящих нейрона и один на выходе. Библиотека Brain.js сконфигурирует сама скрытый слой, и установить количество нейронов, которое сочтет нужным. Мы передали в метод train данные для обучения(обучающая выборка) массив состоит из объектов, которые содержат два свойства input и output, т.е. входящие и выходящие параметры. Нормализация данных не требуется т.к. библиотека сама уже нормализовала данные. На выходе получится число отличное от того, что было в примере, т.к. алгоритм обучения основан на случайных числах при подборе весовых коэффициентов. И, соответственно, метод run используется для получение ответа от нейронной сети по заданным параметрам.

Для обучения сети потребуется довольно массивная выборка данных. Такие данные есть в библиотеке MNIST digits.

После установки всех программ и библиотек, а так же проверки работы библиотеку можем приступить к написанию нейронной сети. Для начала ее нужно обучить.

```
const brain = require('brain.js');
var net = new brain.NeuralNetwork();
const fs = require('fs');
const mnist = require('mnist');
const set = mnist.set(1000, 0);
const trainingSet = set.training;
net.train(trainingSet,
    {
        errorThresh: 0.005,  // error threshold to reach
        iterations: 20000,   // maximum training iterations
        log: true,           // console.log() progress periodically
        logPeriod: 1,       // number of iterations between logging
        learningRate: 0.3    // learning rate
    }
);

let wstream = fs.createWriteStream('./data/mnistTrain.json');
wstream.write(JSON.stringify(net.toJSON(),null,2));
wstream.end();

console.log('MNIST dataset with Brain.js train done.')
```

Рис. 2. Обучающая выборка

Создали сеть, получили 1000 элементов для обучения, вызываем метод для обучения сети, сохраняем все в файл, так же надо не забыть создать подпапку в той же папке что и программа с именем data. После этого необходимо открыть файл для проверки результата.

```
iterations: 0 training error: 0.060402555338691676
iterations: 1 training error: 0.02802436102035996
iterations: 2 training error: 0.020358600820106914
iterations: 3 training error: 0.0159533285799183
iterations: 4 training error: 0.012557029942873513
iterations: 5 training error: 0.010245175822114688
iterations: 6 training error: 0.008218147206099617
iterations: 7 training error: 0.006798613211310184
iterations: 8 training error: 0.005629051609641436
iterations: 9 training error: 0.004910207736789503
MNIST dataset with Brain.js train done.
```

Рис. 3. Результаты обучения

Теперь необходимо сделать систему распознавания.

```
const brain = require('brain.js'),
      mnist = require('mnist');
var net = new brain.NeuralNetwork();
const set = mnist.set(0, 1);
const testSet = set.test;
net.fromJSON(require('./data/mnistTrain'));
var output = net.run(testSet[0].input);
console.log(testSet[0].output);
console.log(output);
```

Рис. 4. Система распознавания

Получается что у сейчас в системе распознавания один тестовый пример из выборки 10000 записей, так же в нее загружена натренированная модель. Пример выполнения данного кода.

```
[ 0, 0, 0, 1, 0, 0, 0, 0, 0, 0 ]
[ 0.0002863506627761867,
  0.00002389940760904011,
  0.00039954062883041345,
  0.9910109896013567,
  7.562879202664903e-7,
  0.0038756598319246837,
  0.00001675291955736278 6,
  0.0007205981595354964,
  0.13699517762991756,
  0.0011053963693377692 ]
```

Рис. 5. Пример выполнения

В примере на входящие в сеть нейроны поступили данные, первый массив это идеальный ответ, а на выходе сеть выдала массив элементов, один из них близок к единице, это тоже третий бит. В четвертом бите есть не стандартное число, таким образом записываются числа с плавающей точкой в JS. Распознавание прошло успешно.

Немного оптимизируем результаты функцией softmax:

```
function softmax(output) {
    var maximum = output.reduce(function(p,c) { return p>c ? p : c; });
    var nominators = output.map(function(e) { return Math.exp(e - maximum); });
    var denominator = nominators.reduce(function (p, c) { return p + c; });
    var softmax = nominators.map(function(e) { return e / denominator; });

    var maxIndex = 0;
    softmax.reduce(function(p,c,i){if(p<c) {maxIndex=i; return c;} else return p;});
    var result = [];
    for (var i=0; i<output.length; i++)
    {
        if (i==maxIndex)
            result.push(1);
        else
            result.push(0);
    }
    return result;
}
```

Рис.6. Функция softmax

Функцию лучше применить в конце нашей программы, прям в выводе в консоль.

```
console.log(softmax(output));
```

Рис. 7. Применение функции softmax

Теперь все работает в понятном для нас виде, запустим программу несколько раз.

```
[ 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 ]
[ 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 ]
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 ]
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 ]
[ 0, 0, 0, 0, 0, 0, 1, 0, 0, 0 ]
[ 0, 0, 0, 0, 0, 0, 1, 0, 0, 0 ]
```

Рис. 8. Результаты оптимизированные с помощью функции softmax

Иногда возможно получение не верного результата, это объясняется тем, что была выбрана не большая выборка дана а так же не достаточно строгая погрешность. Осталось только создать только внешний интерфейс. В данном случае он выглядит так:

```
<head>
    <meta charset="UTF-8">
    <title>MNIST Digits Simples</title>
    <!-- script src="./node_modules/mnist/dist/mnist.js"></script-->
    <script src="./lib/sketchpad.js?v1.1"></script>
    <script src="./dist/build.js"></script>
    <script>
        console.log(window["nn"]);
    </script>
    <style>
        body {
            margin: 10px;
            padding: 10px;
            font-family:Arial;
        }
        #container {
            position:relative;
            width: 500px;
        }
        #sketchpad {
            border: 5px solid #077;
        }
        #thumbnail {
            border: 1px solid #077;
        }
        #result {
            font-size: 128pt; /* Размер шрифта в пунктах */
            float: right;
            width: 120px;
            height: 280px;
            border: 5px solid #fff;
            text-align: center;
            line-height: 280px;
        }

    </style>
</head>
<body>
    <h1>Recognizing hand-written digits from MNIST dataset with Brain.js. Neural Networks live example.</h1>
    <p>
    Draw a digit in the box below and click the "recognize" button.
    </p>
    <br/>
    <div id="container">
        <canvas id="sketchpad" width="280" height="280">Sorry, your browser is not supported.</canvas>
        <canvas id="thumbnail" width="28" height="28">Sorry, your browser is not supported.</canvas>
        <div id="result"></div>
        <div style="clear: right"></div>
    </div>
    <button type="button" id="sketchClearButton">Clear</button>
    <button type="button" id="sketchRecogniseButton">Recognise</button>

```

Рис .9. Внешний вид нейросети

Таким образом, мы приходим к выводу, о том, что данная нейронная сеть готова к использованию и позволяет сделать UI интерфейс более легким в использовании. Так же то, что данная нейронная сеть основана на языке программирования JS позволяет разгрузить сервер, на котором расположен сайт, исполняя код на машине пользователя.

Библиографический список

1. A Beginner's Guide to Neural Networks and Deep Learning // skymind URL: https://skymind.ai/wiki/neural-network (дата обращения: 30.05.2020).
2. Beale H. D., Demuth H. B., Hagan M. T. Neural network design //Pws, Boston. 1996.

3. Афанасьева М.А. Создание и обучение нейронных сетей в системе MATLAB // Молодой ученый. 2014. № 4. С. 85-88.
4. Маркова С.В., Жигалов К.Ю. Применение нейронной сети для создания системы распознавания изображений // Фундаментальные исследования. 2017. № 8-1. С. 60-64.
5. Хачумов В.М. Проект создания программной системы для распознавания графических образов на основе нейронных сетей // Нейрокомпьютеры: разработка, применение. 2008. № 9. С. 52-55.