# Report

# Backpropagation Algorithm for Neural Networks

1641469 Long Cheng

## Introduction

This project implement the backpropagation algorithm for neural networks, the main task it achieved is recognizing some hand-written digit. The hand-written digit data was picked up from the UCI Machine Learning Repository, it has 1593 hand-written digits, this data set was divided into training data and testing data, after training the weight of this neural networks, we can calculate the result for each hand-written digit in the testing dataset, and get the accuracy for running testing data. The result shows that the accuracy can be around 93%. What’s more, I wrote a GUI for test a random digit to demo the result of this neural networks.

## Dataset

### data sources used

The data I used is download from: <http://archive.ics.uci.edu/ml/datasets/Semeion+Handwritten+Digit>*,* named *Semeion Handwritten Digit*. It has 1593 handwritten digits from around 80 persons were scanned, stretched in a rectangular box 16\*16 in a gray scale of 256 values. Actually, when load this .data file in MATLAB, it a 1593\*266 double matrix, each line is a hand-written digit sample, and the first 256 dimensions are a single pixel of a 16\*16 image. The last 10 dimensions are the digit it presents.

### 2.2 main characteristics

|  |  |
| --- | --- |
| Samples number | 1593 |
| Resolution of each hand-written digit | 16\*16 |
| Range of value for each pixel | [0, 1] |
| Number of Attributes | 256 |
| Associated Tasks | Classification |

## Data preprocessing operations

The data was already preprocessed by the owner in UCI Machine Learning Repository, we know that each image has RGB information, but this data only choose 256 gray scale. Furthermore, each pixel was scaled between 0 and 1 (setting to 0 every pixel whose value was under the value 127 of the grey scale (127 included) and setting to 1 each pixel whose ordinal value in the grey scale was over 127). So, all the image was scaled into a 16\*16 binary square box, final 256 binary attributes.

### Point out the label

The data we load in was a 1593\*266 matrix, the excess 10 attributes show the real digit this sample present. We know that there are 10 digits, 10 attributes are suitable for this.

As a neural network, I have a size 10 output, I used [1,0,0,0,0,0,0,0,0,0] to present 0. Only one attribute is 1, others are all 0, the position of 1 present the digit. This is what I done in neural networks, but in data prepare, I convert them into an integer for easier way to understand.

### Random extract testing data

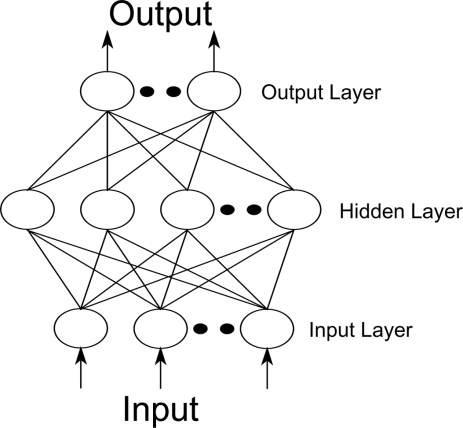
From this data, we have 1593 records. In most way of machine learning, it’s a routine that we should choose some of same for testing, not in training. In this demo, the percentage of testing data I chose was 75%. 25% data was randomly elected for accuracy calculation (398 records).

### Random initialize weight

For each theta, we initialize it in a range from [-epsilon, -epsilon], the epsilon I chose here is 0.12.

## Structure of the neural networks

### Structure

Like the structure in the right, the neural networks I implement in this demo have three layers: input layer, hidden layer and output layer, the size of them are 257, 201 and 10 respectively. The one more node in the input and hidden layer is a bias unit which output +1. In most neural networks, the hidden layer size is set near input size, so I chose 200 here.

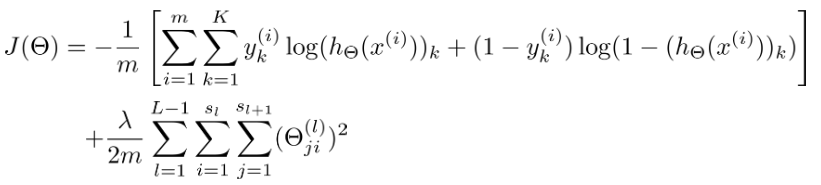
### 4.2 Reason of this structure

As we mentioned above, we have 256 attributes and 10 output dimensions, the one more node in input and hidden layer is bias unit. The size of hidden layer is recommended integer times than input layer size, so I chose 200 here.

## Feedforward implement

### Computational formula

The formula I used to implement the feedforward is:



Which is learned from Coursera ML Class. J represent the cost calculate from this set of weights, m - how many records are used in this training, k – number of outputs, y – the real digit it represents, h – outputs, L – number of layers. This formula is the cost function with regularization. Lambda always set 1 here.

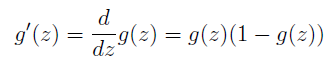
### Implement

We should calculate the cost each time we run the network, feedforward is a good way to test our weight on the lines. The implement here is mainly calculation about matrix, followed the formula and we can get the cost.

## Backpropagation

### Computational formula

First, we should know the sigmoid gradient function:



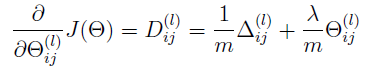
Then, we calculate the Delta backward, the formula for Delta are:







Delta presents how differences in each layer between my result and real digit. Once I get the Delta, the theta’s gradients for cost function can be calculate from:



### Implement

Like the formulas above, just some matrix calculation, this kind of backpropagation calculate the delta backward, and sum all the Delta for all samples. In our demo, we should know that the last formula is only apply to no bias node, bias node doesn’t need regularized.

## Learning the weight of each line of neural networks

After finished cost function and gradient computation, I used fmincg.m to learned a good set parameters. Fmincg.m is an optimized function for parameter’s training, it’s recommended in Stanford ML class. Give your cost function it will train the parameters for you.

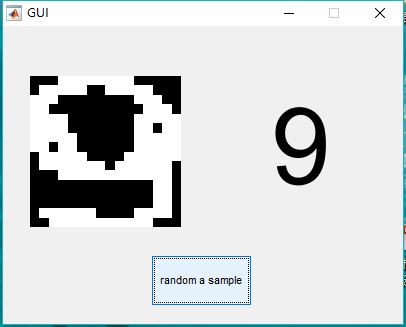
## Prediction result

### accuracy

After running this neural networks on testing data, about 50 iterations, the cost is decreased very slow, the accuracy I got for testing data was around 93% while almost 100% for the training data.

### GUI

I also wrote a GUI to demo the result, randomly chose a sample in the dataset, it will show the digit image and calculate the result of my networks. The GUI shows below.



## Possible improvements

For this small dataset, I can’t feel the time consumption, the accuracy is good enough now, but when the data comes bigger, it will need some methods to decrease the time. Maybe we can draw some features of these input images, make the input attribute smaller. What else, try different size of hidden layer may help the time consumption.