Introduction to machine learning with a simple OCR system on uppercase English characters

*Ning Zhu*

*University of Nottingham, Ningbo*

*Email: scynz1@nottingham.edu.cn*

***Abstract* – As the rapid development of AI, machine learning**

**has been a basic skill almost all CS students need to learn. OCR system on English characters as a typical example of machine learning, can help students command a basic understanding of the machine learning. This paper will focus on how image preprocessing and neural networking combined could build up a simple OCR system with relatively high accuracy, show the basic steps in machine learning.**

1. **Introduction**

AI has been a critical part in human-life.They are nearly everwhere-vending machine in school buildings, cameras in the road, the famous AlphaGo in weiqi etc. In order to keep up with the rapid development of computer technology, CS students start with a simple OCR system which provides a basic understanding of machine learning with students.

This paper will focus on the process of building & training a neural network, as well as the preprocessing of data. In machine learning, a set of data is given and a trained model is created to predict the results. The neural network model proved to be effective in completing such task.

Neural network is a computing model with connected nodes, and its hierarchical structure is similar to that of neural network in brain. Neural network can ”learn“ from data, therefore it can train its pattern recognition, classify data and predict future events. Neural networks subdivide your input into multiple abstraction layers. For example, a large number of examples can be used to train whether the recognition mode is voice or image, just like the behavior of human brain. The behavior of neural network is determined by the connection mode of its elements and the strength or weight of these connections. During the training, the system will automatically adjust the relevant weights according to the specified learning rules until the neural network performs the required tasks normally. In genral, there are basically three parts in a neural network, an ***input layer***, many ***hidden layers*** and an ***output layer***.

The input layer obtains the training data and the expected output values from the file, which are regarded as ***data*** and ***label*** respectively. Then sends the processed ***data*** to the hidden layers.

The hidden layers are the main layers that process ***data*** and predict the results. Basically, for the layers in the hidden layers, the output y has a linear relationship with the input x. The relationship is done as shown below:

Where is the weight and is the bias.

Usually during coding, data flow is passed through layer by layer. The matrix of input, dotted with a matrix of weight and then added to a matrix of bias, will generate the matrix of output. The output matrix is then passed to the next layer as input ”x“. The process is as follows:

Usually, an ***activation layer*** is considered as an output layer. It normalizes the predictions and converts them into floating-point decimals between 0 and 1 which could be treated as probabilities. The ***Sigmoid Function*** is a very commonly used activation function:

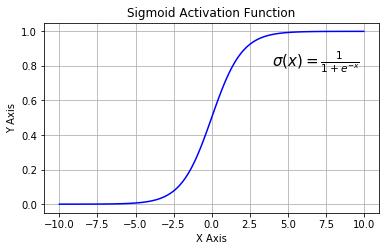


Figure 1: sigmoid function

Its value ranges from 0 to 1, and therefore is one of the optimal activation functions in the industry.

This whole process from input to output is known as ***forwarding***, the ***backwarding*** process which is used to modify the model will be introduced below.

In the ***backwarding*** process, there are two main concepts: ***cost function*** and ***backpropagation***.

***Loss*** is used to reflect the difference between the actual output value of the model and the expected correct output value. ***Loss*** also reflects the equality of a model. ***Cost function*** are used to caculate the ***loss.*** The ***cost function*** takes the weights, biases, input ***data*** and the ***label*** as its inputs. A common ***cost function*** used in machine-learning is the ***Quadratic Cost Function***, is shown as below:

Where is the parameters across the neurons (namely the weights and biases), is the corresponding***data*** and is the ***label***. In genral, the square of the difference between the actual output value and the expected output value.

***Backpropagation*** is essentially an algorithm that allows programs to perform ***gradient descent***：

IMG_256

Where is the parameters across the neurons (namely the weights and biases), is the current gradient, α is step lengrh we set.

The original meaning of gradient is a vector (vector), which means that the directional derivative of a function at this point gets the maximum value along this direction, that is, the function at this point changes the fastest along this direction (the direction of the gradient) and the rate of change is the maximum (the modulus of the gradient).

Through it, the machine-learning system modifies the values of the weights and biases in the hidden layers so as to decrease the ***loss***, and in turn to get a higher equality of the model.

To help you better understand how ***graduent descent*** being achieved in machine learning, specify steps will be introduced below.

After forwarding, the ***output layer*** takes the derivative of the cost function, plugs in its data and passes it to the last layer of the ***hidden layers***.

Note that the ***backpropagation algorithm*** works by the ***chain rule***:

***Backpropagation*** will stop at the first layer of the hidden layers, and get the gradient of current model. In that case, we just need to set a appropriate hyper-parameter α to represent the step and use the step and gradient to update weights, biases.

Given that the mean value of the sigmoid function is not 0. In the real use, the network should be trained with batch of input items, which means that 1000 or more pieces of items are parsed into the system at the same time. An epoch means that we have completed a ***forward propagation*** of all the data and updated the weights and biases with ***back propagation***.

An ***accuracy layer*** can also be added to directly repersent how accurate the prediction is (***Loss*** is not intuitive). It compares the predictions with the correct labels and quantizes the assessment. Using boolean values as results, the accuracy layer sums up the results and takes the average, and then prints it out for users to decide whether the neural network is good enough for application or not.

1. **Materials and Methodology**

For this task, Python 3 is the first-choice programming language. Its intuitive syntax and extensive machine-learning libraries makes coding more efficient. The following libraries are used:

* PIL
* Numpy

1. ***Calibration & Training***

The first thing is to acquire training datas. For this project, 17\*17 pixels grayscale images of hand written English letters are downloaded from [machine-learning data repositories.](https://github.com/znzz1/Machine-learning)

For convenient, we use the [proprecessed datas](https://github.com/znzz1/Machine-learning/blob/main/train.npy) from 17\*17 pixels grayscale images directly.

1. **References**

[1] <https://gimg2.baidu.com/image_search/src=http%3A%2F%2F5b0988e595225.cdn.sohucs.com%2Fimages%2F20171102%2F3c33d1500b5f44ba8ae8d07c0f534d45.png&refer=http%3A%2F%2F5b0988e595225.cdn.sohucs.com&app=2002&size=f9999,10000&q=a80&n=0&g=0n&fmt=jpeg?sec=1613806502&t>