International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Handwritten English Character Recognition Using Logistic Regression and Neural Network

Tapan Kumar Hazra[1], Rajdeep Sarkar[2], Ankit Kumar[3]

[1]Department of Information Technology, Institute of Engineering and Management, Salt Lake, Kolkata-700091
[2]Department of Information Technology, Institute of Engineering and Management, Salt Lake, Kolkata-700091
[3]Department of Information Technology, Institute of Engineering and Management, Salt Lake, Kolkata-700091

Abstract: ***Hand written character recognition is a challenging task often resulting in ambiguous labels. Using the concepts of Machine learning we have tried to develop an Optical Character Recognition (OCR) system where an algorithm is trained on a data set of known letters and then can learn to accurately classify new data. Our optical character recognition (OCR) system for handwritten English characters comprises of two steps-Generating training set data using an OCR tool and then Applying different machine learning algorithm on the training set and starts the learning process. A variety of algorithms have shown good accuracy for the handwritten letters, two of which are looked here.***

Keywords: neural network, classification, optical character recognition, regularization, logistic regression

1. Introduction

Optical Character Recognition (OCR) an area of computer science that started developing as early as 1950, currently encompasses two previously distinct areas - pure optical character recognition, using optical techniques such as mirrors and lenses and digital character recognition, using scanners and computer algorithms. The intent of this paper is to focus on recognition of single handwritten character, by viewing it as a data classification problem. Analysis of the result should give us an idea on the viability of the algorithms implemented for use in complete OCR application.

Success of optical character recognition depends on a number of factors, two of which are feature extraction and classification algorithms. We find English OCR an interesting problem. This is mainly because OCR itself lies in between one dimensional natural language process (NLP) and three dimensional real world projected down to image in computer vision.

The way we extract features for training/testing our model is by first writing the text on a white or light background. Then we use the detection of white spaces as a means of segmenting the image into regions that contains lines of text. We can reapply the process to segment the line regions into smaller regions that contain words and letters.

While some OCR algorithms work with grey-scale images, many convert their input into binary images during the early stages of processing. For our implementation we have done both, but have used the grey-scale intensity values as input features. We have not used binarized image because it outputs intensity in terms of only 0 and 1 which may not give as good result as a grey-scale image. Given that we have image regions that contain text, whether single word regions or whole slabs of text, the goal here is to identify image pixels that belong to text and those that belong to the background.

In our project we have used a tool [4] for feature extraction which is the intensity of each pixel. This tool is completely open source, but we have made certain modifications to it according to our need. The data generated by this tool is then used to train our algorithm. In our work, most characters are well structured and well-separated [2], which is different from general vision recognition tasks where objects have much variation.

2. Related Work

2.1 OCR tool

Optical character recognition is a combination of natural language processing (NLP) and Computer Vision. In our application, language modeling is less important since we have more clues from the image likelihood, and the vision clues are more robust since it is an intrinsic 2Dimage.

Below is given how our OCR tool works:

1. In first phase, the image containing the text is converted into gray-scale (preprocessing).

Figure 1: gray-scaled then binarized image

2. In next phase, the system finds characters by layout analysis. Here the text is separated into unconnected characters.

3. The tool then generates a labeled data set for the identified characters. The data generated is used to train our model.
4. Lastly test characters are classified based on model from training data. This is referred to as recognition.

2.2 Classification

In machine learning, classification [1] is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations whose category membership is known. Classification is an example of pattern recognition.

In the terminology of machine learning, classification is considered an instance of supervised learning, i.e. learning where a training set of correctly identified observations is available. The corresponding unsupervised procedure is known as clustering, and involves grouping data into categories based on some measure of inherent similarity or distance.

2.2.1 Definition

We first review the general goal of Machine Learning Algorithm: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

For classification problem, the task is to tell things different apart and the experience E is to learn the suitable representation of the true data distribution based on the performance measured by P on training data. It can be viewed as an optimization problem over a hypothesis space h. In the parametric setting, the objective function is given by $f(\Theta;x)$ where Θ is the model parameter and x is a given sample. An empirical objective is given in equation (1):

$$f(\Theta) = \frac{1}{m}\sum_{i=1}^{m} f(\theta\,\Theta; x_i) \qquad (1)$$

here $x_1, x_2, \ldots, x_m$ are our samples.

2.2.2 Parametric vs Nonparametric

In terms of modeling the hypothesis space, classification algorithms can be divided into parametric ones and nonparametric ones. In the parametric setting where the hypothesis space is a family of parameterized function, no matter Frequentist, Bayesianist or Learning Theorist, no matter Discrimitive, Generative or Algorimitic, we end up with an object function which we have to optimize.

2.2.3 Binary-class vs. Multi-class

In practice Neural Networks can be easily applied for multi-class classification. But, since Logistic Regression is a binary classifier, there is no direct methodology on multi-class problems. One natural approach is to reduce the multi-class problem to several binary ones. This can be done either by one-vs.-one or one-vs.-all schemes.

3. Data Sets

Since the algorithms we used are supervised learning algorithms so we had to develop a way to generate the labeled data sets. For this we have used our OCR tool which gives us our required labeled data set as shown in the following figure:

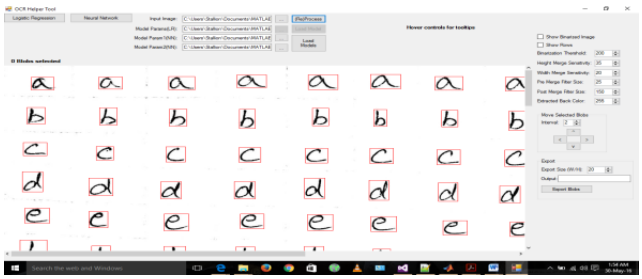

Figure 2: OCR tool

This tool recognizes individual characters and for each character outputs a 20*20 matrix. This gives us a feature set of 400 features for each training example.

To obtain these features we first convert our image to gray scale. Then each individual character is recognized and the matrix is formed by the intensity value of every pixel.

4. Algorithms Implemented

We realized that the essence of the project was to understand the intricacies of different machine learning algorithms and to learn which algorithm gives good result for which use case. With this philosophy in mind, we wrote our own implementation of Regularized Logistic Regression [3] and Neural Network.

The following figure shows how a supervised learning algorithm works:

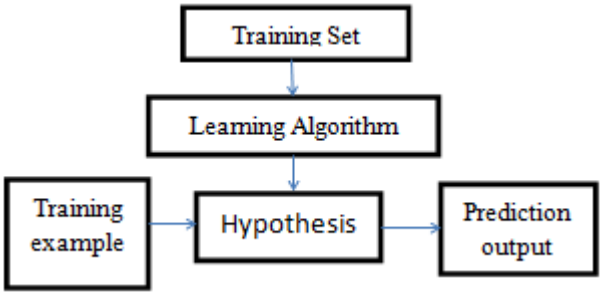

Figure 3: Supervised learning algorithm working

4.1 Regularized Logistic Regression

Logistic regression is a binary classifier. But our project had 26 different classes, so we implemented a one vs. all extension to classify our 26 characters. The goal of this algorithm is to minimize the cost function given by equation (2):

$$J(\Theta) = -\frac{1}{m}\{\sum_{i=1}^{m} y^{(i)} \log\left(h_{\Theta\Theta}(x^{(i)})\right) + (1-y^{(i)}) \log(1-h_{\Theta\Theta}(x^{(i)}))\} \quad (2)$$

Where, $h_{\Theta\Theta}$ (x) = $\frac{1}{(1+e^{-\Theta\Theta^T X})}$ is the sigmoid function or logistic function.

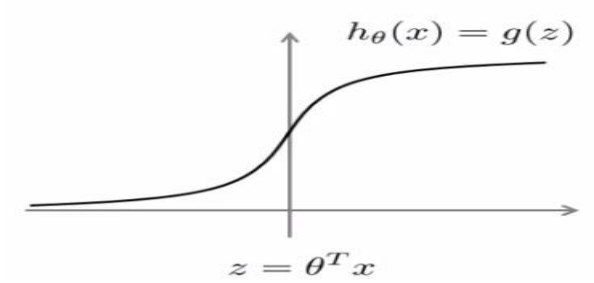

Figure 4: Logistic function

4.1.1 Interpreting hypothesis output

When our hypothesis ($h_{\Theta\Theta}$ (x)) outputs a number, we treat that value as the estimated probability that y=1 on input x. For example, If x is a feature vector with x_0=1 and x_1 = tumor Size and we get $h_{\Theta\Theta}$ (x) = 0.7, this tells that the patient has a 70% chance of the tumor being malignant. We can write this with the following notation:

$$h_{\Theta\Theta}(x) = P(y=1|x;\theta) \quad (3)$$

Equation (3) represents the probability that y=1, given x, parameterized by θ.

4.1.2 Overfitting with Logistic Regression

If we have large number of features, the learned hypothesis may fit the training set well, but fails to generate new examples, i.e. predict the output on new data. This is called over fitting. Since our data set has large number of features (400), there is a high probability that our hypothesis will overfit our training data. To overcome this there are mainly two ways:

4.1.2.1 Reducing the number of features

To reduce the number of features we can either select which features to keep or we can use Model Selection algorithms. But, in reducing the number of features we lose some information. We can also select those features which minimize data loss, but even so, some information is still lost.

4.1.2.2 Regularization

The second and much better way to resolve overfitting is by using a technique which is called regularization. This allows us to keep all features, but reduces the magnitude of parameters Θ. The regularized logistic regression cost function is given in equation (4):

$$J(\Theta) = -\frac{1}{m}\{\sum_{i=1}^{m} y^{(i)} \log\left(h_{\Theta\Theta}(x^{(i)})\right) + (1-y^{(i)}) \log(1-h_{\Theta\Theta}(x^{(i)}))\} + \lambda/2m \sum_{j=1}^{n} \Theta_j^2 \quad (4)$$

Where, λ is the regularization parameter, $x^{(i)}$ is the set of features, $y^{(i)}$ is the set of labels, Θ is the parameter upon which the optimization is done, n is the number of features for each training example and m is the number of training example.

4.2 Neural Network

In artificial Neural Network, a neuron is a logistic unit. We feed input via input wires, the logistic unit does the computation and the output wire sends the output. The logistic computation is the same Logistic regression hypothesis calculation.

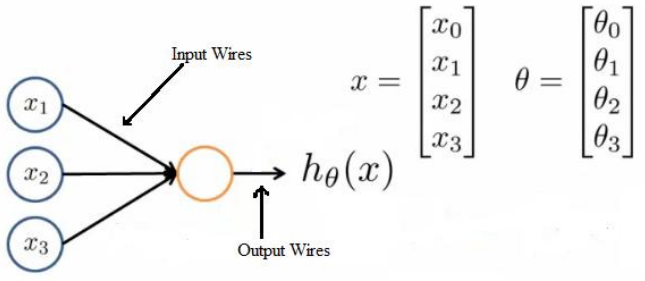

Figure 5: A single neuron

The above figure is an artificial neuron with a sigmoid (logistic) activation function. The Θ vector is also called the weights of the model.

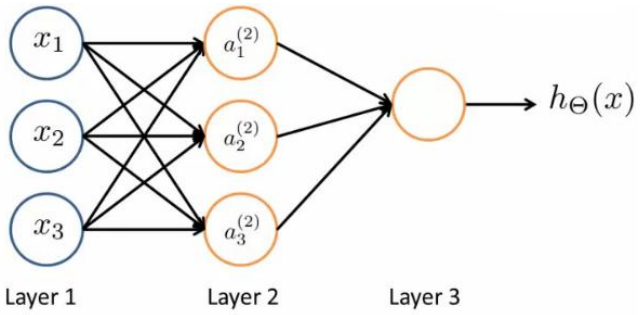

Figure 6: A neural network

In the above figure x_1, x_2, and x_3 constitute the input layer, a_1^2, a_2^2 and a_3^2 constitute the hidden layer and $h_{\Theta\Theta}(x)$ which produces the output is called the output layer. We can have as many hidden layer as our requirement.

4.2.1 Types of classification problem with Neural Network

There are basically two types of classification:

i. Binary classification

For binary class, there is one output, either 0 or 1. So there is only one output node and the value is going to be a real number and s_L= 1. So the output vector y is a k-dimensional vector of real numbers.

ii. Multi-class classification

For multi-class there are k distinct classifications. Typically k is greater than or equal to 3. Here $s_L = k$.

4.2.2 Implementation details

For Neural Network we have implemented the backpropagation algorithm. Our Neural Network has one input layer, one hidden layer, and one output layer. The output layer had 26 neurons or nodes each for one class. The cost function that was minimized in this algorithm is given in equation (5):

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} log\left(h_\Theta\left(x^{(i)}\right)\right)_k + (1 - y_k^{(i)})\, log\left(1 - \left(h_\Theta\left(x^{(i)}\right)\right)_k\right]\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta\Theta_{ji}^{(l)}\right)^2 \quad (5)$$

Where K = number of distinct classes, $s_l = k$.

While implementing the hidden layer we tried with many different numbers of neurons and finally found that the number of nodes in the hidden layer should be around mean of input layer and output layer.

There are basically two halves to the Neural Network cost function represented by equation (5).

4.2.3 First half

$$-\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} log\left(h_\Theta\left(x^{(i)}\right)\right)_k + (1 - y_k^{(i)})\, log\left(1 - \left(h_\Theta\left(x^{(i)}\right)\right)_k\right]\right] \quad (5a)$$

This says, for each training data example (i.e. 1 to m – the first summation) sum for each position in the output vector.

4.2.4 Second half

$$\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta\Theta_{ji}^{(l)}\right)^2 \quad (5b)$$

This massive regularization summation term is also called a weight decay term. The regularization term is very similar to that in Logistic Regression cost function given in equation (4).

The figure given below shows the minimization of the cost function:

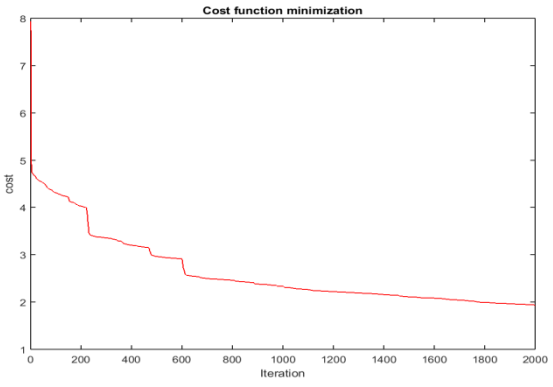

Figure 7: Cost function minimization

4.2.5 Parameters for controlling mapping from one layer to next

If a neural network has s_j units (neurons) in layer j and s_{j+1} units in layer j+1 then Θ^j will be of dimensions $[s_{j+1} * s_j+1]$

4.2.6 Gradient checking

Backpropagation has a lot of details; small bugs can be present and ruin it.

This may mean it looks like J (Θ) is decreasing, but in reality it may not be decreasing by as much as it should. So using a numeric method to check the gradient can help diagnose a bug. Gradient checking helps make sure an implementation is working correctly.

5. Result

We perform our training by taking samples of handwriting from two different individuals. We train our classifiers for both lower and upper case letters. Our training data contains 26 characters, and totally about 1100 training samples for each individual and each case. Testing data contains about 100 characters. The following tables show training versus testing accuracy for the two different learning algorithms we implemented:

Table 1: First person, lower case

Algorithm	Training Accuracy	Testing Accuracy
Logistic Regression	100%	79.22%
Neural Network	82.64%	59.74%

Table 2: Second person, lower case

Algorithm	Training Accuracy	Testing Accuracy
Logistic Regression	100%	74.36%
Neural Network	90.09%	55.13%

Table 3: First person, upper case

Algorithm	Training Accuracy	Testing Accuracy
Logistic Regression	100%	88.46%
Neural Network	97.52%	79.49%

Table 4: Second person, upper case

Algorithm	Training Accuracy	Testing Accuracy
Logistic Regression	100%	78.21%
Neural Network	81.79%	50.00%

6. Conclusion

Though our proposed method recognizes handwritten characters with about 80% accuracy for Logistic Regression and about 60% for Neural Networks, it's still not good enough for practical use. Constraint to the difficulty of obtaining ground truth data, we here only conduct experiment on a relatively small dataset. In future work, we wish we can train these methods with more data and have further testing.

7. Future Scope

1. We used raw pixel intensity as out features. But we can use different method to extract features which might improve our performance.
2. In Neural Network we used only one hidden layer. Neural Networks in computationally very expensive and it takes very long time to train in our machines. So, we can try to run Neural Networks with more number of hidden layers on high performance computing machines.
3. We can develop an Android/iOS application that identifies the handwriting in real time on clicking a picture of it.

References

[1] Bhandari, Aditya, Ameya Joshi, and Rohit Patki. "Bird Species Identification from an Image."
[2] URL:http://cs229.stanford.edu/proj2014/Aditya%20Bhandari,%20Ameya%20Joshi,%20Rohit%20Patki,%20Bird%20Species%20Identification%20from%20an%20Image.pdf
[3] Shih, Yichang, and Donglai Wei. "Machine Learning Final Project: Handwritten Sanskrit Recognition using a Multi-class SVM with K-NN Guidance." URL:http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.357.9935
[4] George Margulis- Optical Character Recognition: Classification of handwritten Digits and Computer Fonts. URL:http://cs229.stanford.edu/proj2011/Margulis-OpticalCharacterRecognition.pdf
[5] ml-ocr-tool. URL: https://github.com/ValYouW/ml-ocr-tool

Author Profile

Tapan Kumar Hazra completed his M.E degree from Jadavpur University, Kolkata, West Bengal, India. Since from 2003, he is working as Assistant Professor of Department of Information Technology at Institute of Engineering & Management, Salt Lake, Kolkata, West Bengal, India. His research interest includes Design and Analysis of Algorithms, Image Processing, Natural Language Processing, Sentiment Analysis, Machine learning, Cryptography.

Rajdeep Sarkar, is pursuing his final semester B.Tech degree in Information Technology from Institute of Engineering and Management affiliated to Maulana Abul Kalam Azad University of Technology, West Bengal. His research interest includes Machine learning and image processing.

Ankit Kumar is pursuing his final semester B.Tech degree in Information Technology from Institute of Engineering and Management affiliated to Maulana Abul Kalam Azad University of Technology, West Bengal. His research interest includes Machine learning.