Comparative Study of Classifiers in Handwritten Digit Classification

Amandeep Singh
Northeastern University
Boston, MA
singh.aman@husky.neu.edu.com

Sangeetha Chandrashekar
Northeastern University
Boston, MA
chandrashekar.s@husky.neu.edu

ABSTRACT

This paper discusses the predicament of multiclass data classification and details out various approaches(KNN, Logistic Regression, BackPropagation) solving it hence drawing a comparative study among the classifiers based upon their accuracy, performance, execution time.

KEYWORDS

Multi-Class Classification,Pattern Recognition, MNIST, Handwriting recognition, KNN; Logistic Regression, Neural Net;

1 INTRODUCTION

Pattern recognition in the context of artificial intelligence is the classification of patterns and regularities in the given input and finding the best-matched answer using supervised or unsupervised learning algorithms. Pattern recognition is one of the most recognized problems to solve since it has numerous beneficial real-world applications like diagnosis using medical reports, check sorting and stock market prediction for financial institutes, traffic control systems, guidance systems, image processing and so on. Handwritten digit recognition is a small section of the broader pattern recognition spectrum. Handwritten digit recognition using artificial intelligence has been one of the oldest classification problems. This classification problem originally showed up in the sorting of mail by zip code for US Postal Service, which aimed at sorting mail at a quicker rate and with fewer labor costs. Although recognizing handwritten digits seems to be a simple problem from a human perspective, teaching a machine the different strokes that make up a digit and also to amalgamate a digit written in various handwriting is quite an issue. This recognition problem has various algorithms that are used to produce almost accurate results for a given input [5, 9]. With the rise in the size of data that is being processed on a day-to-day basis towards a range of thousands of terabytes and petabytes, we are in need of efficient algorithms to handle and classify any given size of data.

In this paper, we use three classification algorithms, k- nearest neighbors [6], logistic regression [4] and back propagation [2] for deriving valid outputs and comparison between the three algorithms. We use Modified National Institute of Standards and Technology's handwritten digit database as input to the algorithms. [7]

, ,
©

2 CLASSIFIERS

This section discusses the various classifiers used in our study.

2.1 K Nearest Neighbours

KNN is the simplest algorithm with no learning perspective. The logic is to compare a test case with all the training values and get the distance between them. The prediction here is given by the majority in the nearest distance neighbors. The logic behind is that the similar content would form clusters and would lie near to ones on class type. The distance taken here is Euclidian distance and K is taken as 10.

Algorithm 1 K Nearest Neighbour

function Classification(X, Y, x) ▷ Where X - training data, Y - training labels, x - test sample
  for $i = 1$ to m **do**
    compute distance $d(X_i, x)$
  compute set I containing indexes for k smallest $d(X_i, x)$
return majority label for $Y_i where i \in I$

2.2 Logistic Regression

Classification can be seen as a problem of regression in which value of y are limited in either of the classes.Hence to limit the value of y in the the classification problem we choose hypothesis $h_\theta(x)$ so $y \in \{0, 1\}$ [8].

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + \exp -\theta^T x} \tag{1}$$

where

$$g(z) = \frac{1}{1 + exp(-z)} \tag{2}$$

is called the logistic or sigmoid function. Figure(1) is the graph showing $g(z)$. Here it can be noticed that as $x \rightarrow \infty\ y- > 1$ and as $x \rightarrow -\infty\ y- > 0$. Hence $h_\theta(x)$ is bounded between {0,1}. Now we assume

$$\begin{aligned} P(y = 1|x; \theta) &= h_\theta(x) \\ P(y = 0|x; \theta) &= 1 - h_\theta(x) \end{aligned} \tag{3}$$

Now the likelihood function for m data can be written as

$$\begin{aligned} L(\theta) &= p(\vec{y}\,|X; \theta) \\ &= \prod_{i=1}^{m} (h_\theta(x^i))^{y^i} (1 - h_\theta(x^i))^{1-y^i} \end{aligned} \tag{4}$$

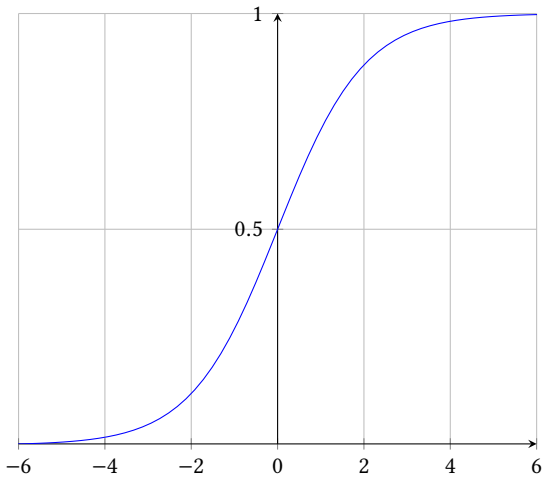


Figure 1: Sigmoid function

taking log of the likelihood

$$l(\theta) = \log L(\theta)$$
$$= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log (1 - h(x^{(i)})) \quad (5)$$

Applying gradient decent

$$\frac{\partial l(\theta)}{\theta_j} = (y - h_\theta(x))x_j \quad (6)$$

Therefore

$$\theta_j = \theta_j - \eta(y^{(i)} - h_\theta(x^i))x_j^{(i)} \quad (7)$$

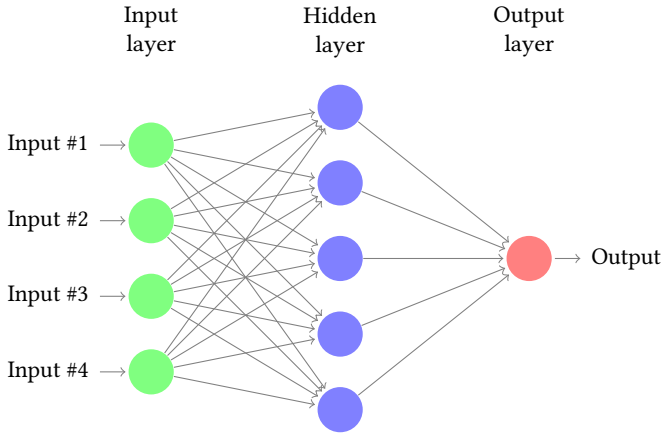
Since this would the prediction for one class hence using one vs rest approach to predict for all 10 classes[3] and choice made between two classes based upon the probability of accuracy of each class.

Algorithm 2 Logistic Regression

```
procedure LOGISTIC REGRESSION(X, Y, X_test)
    procedure INITIALIZATION
        param ← randomvaluessizeequaltototalfeatures
    procedure FITTING
        while notstoppingcriteria do
            ynew ← sigmoid(X · param)              ▷ equation 2
            param ← param − η((y − y_new) · X)     ▷ equation 7
    procedure PREDICT
        return sigmoid(X_test · param)             ▷ equation 2
```

2.3 Backpropogation

Backpropagation is a type of supervised learning classification algorithm which is used with neural net to find the error margin- the difference between the expected outcome and actual outcome - from each neuron participating in the neural net and adjust the input contributions so that the error is gradually minimalist. Hence we need to calculate the partial derivatives of the weights and the bias of the input layer at every updated calculation to achieve desired output.



There are three steps to backpropagation algorithm :

(1) Feedforward pass
(2) Error calculation
(3) Backward pass

2.3.1 *Feedforward Pass.*

$$a_i^k = b_i^k + \sum_{i=1}^{r_{k-1}} w_{ij}^k o_i^{k-1} \quad (8)$$

In the feedforward pass, the weights for each input is initially assigned at random and bias for each layer is given. Later the net input, a, is calculated using bias b and weights w and output o. The output o is generally 1 for input layer. The output for the derived net input is calculated using an activation function. The activation function used in this paper is called sigmoid function(2).

In equation (8) x is the net input and g(x) is the output for the layer.The same process is repeated with the hidden layer and we arrive at the final output.

2.3.2 *Error Calculation.*

$$E_{total} = \sum \frac{1}{2}(target - output)^2 \quad (9)$$

The error margin is calculated using the above equation where the *target* is the expected output and the *output* is the final output achieved after one iteration of feedforward pass. Then we calculate the error margin, also called as *gradient descent*, for each weight by calculating the partial derivative of the weight with respect to the total error.

$$\frac{\partial E}{\partial w_{ij}^k} \quad (10)$$

This can be calculated using chain rule :

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial a_j^k} \frac{\partial a_j^k}{\partial w_{ij}^k} \quad (11)$$

The right hand side of the equation (11) is called as the delta value for the output a .

2.3.3 Backward Pass. Calculating the gradient descent of a weight helps in adjusting the value of that weight such that the actual output is closer to the expected output. We can do this by using equation (12).

$$w_i^+ = w_i - \eta \frac{\partial E}{\partial w_{ij}^k} \tag{12}$$

The same procedure is applied to every weight at every layer starting from hidden layer and propagating back to the input layer. After several iterations we achieve the expected value with a minimalist error margin. There are 784 features in each input image of MNIST database that would become 784 neurons of input the layer, 30 neurons are used in hidden layer and 10 neurons for output layer since we are calculating the possibility of a particular input to be one of the 10 digits also η is the learning rate which is set to 3.5. The output with the least error is considered to be the answer for the particular image. In our paper we use 10 iterations to achieve the maximum accuracy.

Algorithm 3 BackPropogation

1: **for** *d* in *data* **do**
2: **procedure** FORWARD PASS
3: Starting form forward pass use eq.to do a forward pass through the network, computing the activity of neurons in each layer.
4: **procedure** BACKWARD PASS
5: Compute the derivatives of functions with respect to output layer activities
6: **for** *layer* in *layers* **do**
7: Compute the derivatives of the error function wrt inputs of the upper layer neurons
8: Compute the derivatives of the error function wrt weights b/w the outer layer and the layer below
9: Compute the derivatives of the error function wrt activities of the layer below.
10: Update the weights

3 EXPERIMENTS

3.1 Data Set

The data set used is MNSIT[7]. The data set has 55000 training images and 10000 test images. This data set is fixed with images centered and normalized. For ease of use the data downloading and division into training and test set is done using tensorflow[1]. Containing images in 28x28 format reduced to a 1D array of 784 features.

3.2 Environment

All experiments where carried out in AWS t2.large with two CPUs and 8Gb RAM. The code is written python 2.7.10.

3.3 Experiments

The experiments carried out are basically to draw comparative study among the above algorithms. We here are comparing on the bases of accuracy and time consumed by the method.

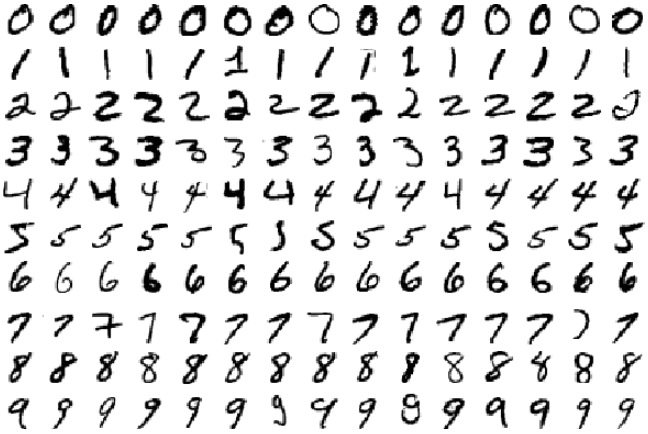


Figure 2: MNIST Data sample

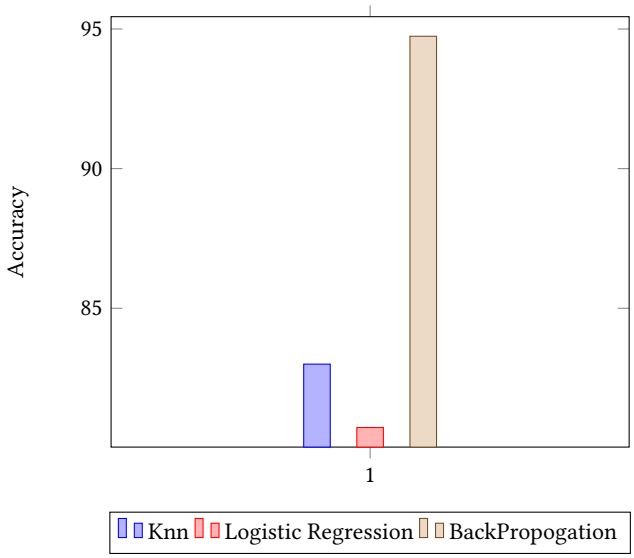


Figure 3: Accuracy Comparison

3.3.1 Comparison based on Accuracy. The algorithms are ran trained on a data set of 5000 images and then tested for 10000 images the results from various algorithms are noted and plotted as a graph:3 for a better and easier view. With y axis as accuracy percentage and x as the algorithm.

3.3.2 Comparison based on Time Taken. The algorithms are ran trained on a data set of 5000 images and then tested for 10000 images the results from various algorithms are noted and plotted as a graph:4 for a better and easier view. With y axis as time and x as the algorithm.

3.3.3 K value variation for Knn. The algorithms are ran trained on a data set of 750 images and then tested for 250 images the results from algorithm are noted and plotted as a graph:5 for a better and easier view. With y axis as accuracy and x as the K value.

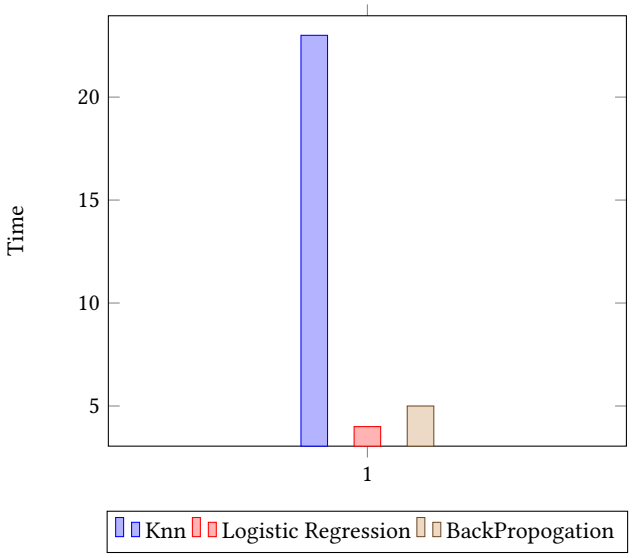


Figure 4: Time Comparison

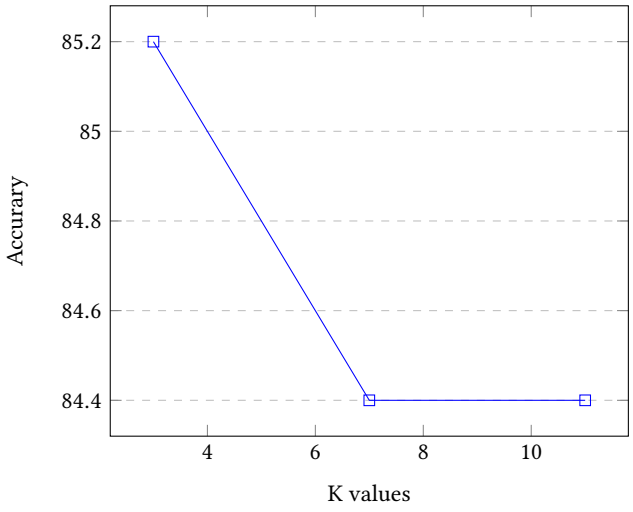


Figure 5: K variation for KNN

3.3.4 Learning rate value variation for Logistic Regression. The algorithms are ran trained on a data set of 5000 images and then tested for 10000 images the results from algorithm are noted and plotted as a graph:6 for a better and easier view. With y axis as accuracy and x as the learning rate(η) value.

4 CONCLUSIONS

On comparing the results returned by the three algorithms namely, K-nearest neighbors, Logistic Regression and Backpropagation, a general conclusion that we draw is that backpropogation performs much better than other classifiers mentioned above taking accuracy and time taken into consideration.

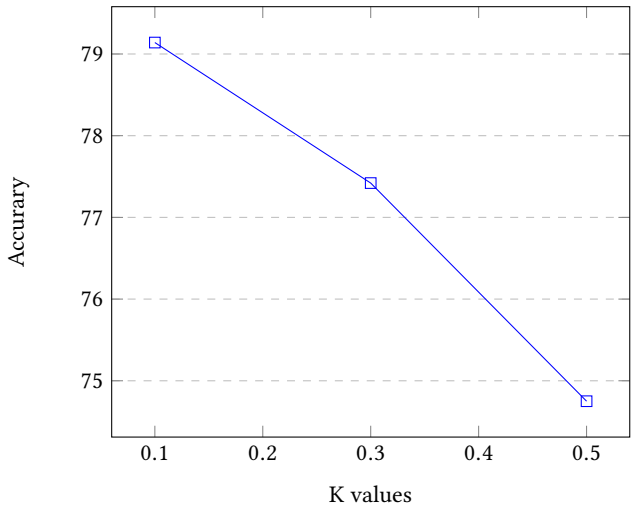


Figure 6: Learning rate variation: Logistic Regression

Figure (3) for experiment 3.3.1 show a clear distinction between the performance of all three algorithms. Backpropogation outperforms the other two by higher margin where result being 83,80.73,94.74 for KNN, logistic regression and backpropogation respectively. On further addition of the result from experiment 3.3.2 we observe a huge time difference between Knn and the other two, reason being that Knn is a simple algorithm with no supervised learning and it compares the test to all training data. The other two algorithm perform comparatively similar to each other. The results can be visually interpreted from figure(4).

On further experimenting with the k parameter of Knn 3.3.3 that is the number of nearest neighbour, we observe that there is a drop in accuracy with increasing value of k which ultimately achieves a constant accuracy range. This pattern can clearly be seen in the figure(5).

Lastly we note observations with the change in learning rate for logistic regression in experiment 3.3.4. We observe a fall in accuracy which could be due to lack of initial exploitation of a faster convergence with high learning rate.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Stacy C. Marsella for providing the opportunity to learn as well as experiment using the coursework and teaching assistant's for their valuable guidance.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). https://www.tensorflow.org/ Software available from tensorflow.org.

[2] W. A. M. Ahmed, E. S. M. Saad, and E. S. A. Aziz. 2001. Modified back propagation algorithm for learning artificial neural networks. In *Radio Science Conference,*

2001. NRSC 2001. Proceedings of the Eighteenth National, Vol. 1. 345–352 vol.1. https://doi.org/10.1109/NRSC.2001.929244

[3] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.

[4] J. R. Brzezinski and G. J. Knafl. 1999. Logistic regression modeling for context-based classification. In *Proceedings. Tenth International Workshop on Database and Expert Systems Applications. DEXA 99*. 755–759. https://doi.org/10.1109/DEXA.1999.795279

[5] Chayaporn Kaensar. 2013. *A Comparative Study on Handwriting Digit Recognition Classifier Using Neural Network, Support Vector Machine and K-Nearest Neighbor*. Springer Berlin Heidelberg, Berlin, Heidelberg, 155–163. https://doi.org/10.1007/978-3-642-37371-8_19

[6] J. Laaksonen and E. Oja. 1996. Classification with learning k-nearest neighbors. In *Neural Networks, 1996., IEEE International Conference on*, Vol. 3. 1480–1483 vol.3. https://doi.org/10.1109/ICNN.1996.549118

[7] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. (2010). http://yann.lecun.com/exdb/mnist/

[8] Andrew Ng. 2017. *cs229 course notes*. http://cs229.stanford.edu/notes/cs229-notes1.pdf.

[9] N. Pise and P. Kulkarni. 2016. Algorithm selection for classification problems. In *2016 SAI Computing Conference (SAI)*. 203–211. https://doi.org/10.1109/SAI.2016.7555983