Finding Optimized Machine Learning Model For Recognizing English Handwritten Digit

Nowfel Mashnoor
Roll: 1503069

Amir Faruk
Roll: 1503075

Abstract

This paper is about the comparison between different Machine Learning models(classifiers) trained and tested on MNIST dataset. For declaring a model as best, we only considered low error score. A standard machine learning library written in Python Programming Language is used during this research.

1 Introduction

Handwritten Digit Recognition has been very successful in recent years. A lot of research and studies has been done in recent years on it like Devnagari Handwritten Character Recognition[1]. Handwritten digit recognition technique is used in various fields like PDA, bank cheque, handwritten fields in form etc.[2] Using machine learning technique, which which can be briefly defined as enabling computers make successful predictions using past experiences, [3] handwritten digit recognition system is greatly improved. Handwritten Digit recognition is a supervised learning algorithm problem. There are many classifier in supervised learning like Neural Network, Decision Tree, Bayesian Network, Support Vector Machine(SVM), Random Forest etc[4]. A comparison study has been already done where *Base Linear Classifier, Baseline Nearest Neighbor Classifier, Large Fully Connected Multi-Layer Neural Network, Tangent Distance Clasifier(TDC), LeNet 4 With KNN, Optimal Margin Classifier* are compared among.[5] In our research we are going to compare among Random Forest and Artificial Neural Network. We will chose the best classifier among them based on their accuracy on testing set.

2 MNIST Dataset

For training and testing our classifiers, we used MNIST (Modified National Institute of Standards and Technology) Dataset. This dataset contains *70000* images. Among 70000 images, 60000 images are for training and 10000 images are for testing[6]. It is basically a subset of NIST dataset. the black and white images from NIST were normalized to fit into a 28x28 pixel and converter to grayscale levels.[7]

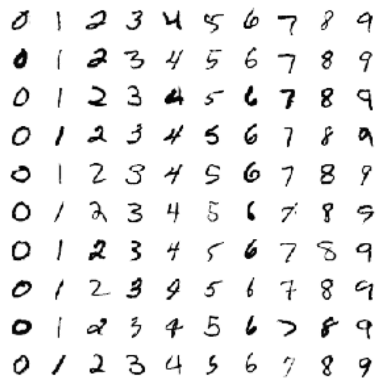

Figure 1: MNIST Dataset Sample Images

For each and every classifier we are using in our study, we will use use all the 60000 images for training and the rest 10000 images for evaluating our models.

3 Scikit Learn Machine Learning Library

Scikit-learn is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems[8]. Most of the algorithms are written in Python programming language where some of the algorithms are written in Cython for achieving optimal performance. Also for efficient operation on large arrays and multidimensional matrices, we used NumPy library.[9] For graphing and other visualization purpose, we used Matplotlib, which is other Python Library. [10] For training and testing our classifiers, we will use the Scikit Learn Library. This library also provides tools for model evaluation.

4 The Classifiers

We have trained and tested three classifiers on MNIST Dataset. We are going to give a quick and brief description of all of them.

4.1 Random Forest

Random Forest is a powerful Machine Learning model. It can do both regression and classification.[11] It is basically a collection of random decision trees. Each decision tree predict a class and after all the prediction, random forest outputs that class which got the most vote. The main benefit of Random Forest over decision tree is that it removes the problem of over fitting of decision tree.[12] Random forest uses Bootstrap aggregating or Tree Bagging

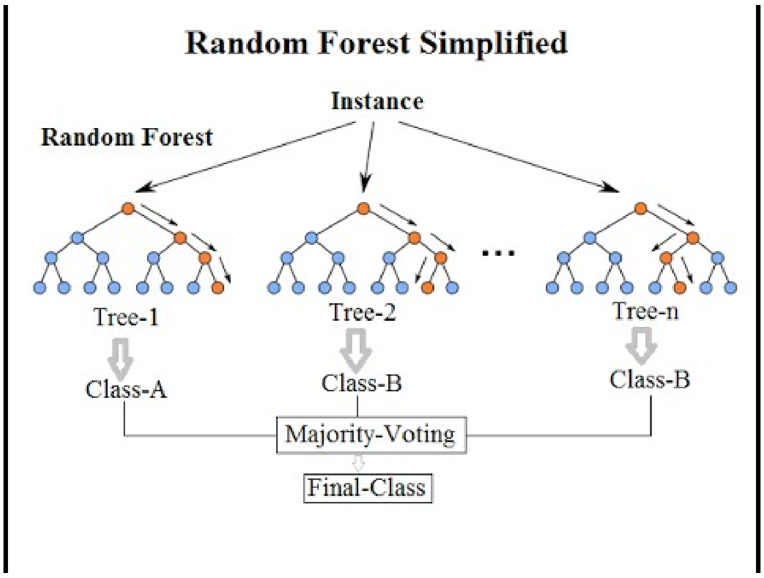

Figure 2: Random Forest

algorithm. It reduces variance and helps to prevent over fitting. In decision tree learning, for splitting the variable at each step, different metrics can be used like Gini Impurity, Information Gain. We used Gini Index or Gini Impurity in our work. For calculating Gini Index, we have to use the following formula:

$$I_G(P) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k$$

Where,
J = No. of Classes
i ϵ 1, 2, 3, ..., J

p_i = Probability of an item with label i being chose
p_k = Probability of a mistake categorizing that item [13]

4.2 Artificial Neural Network

Artificial Neural Network(ANN) is a computing system inspired by biological neural network that constitute animal brains.[14] It is one of the best models for both classification and regression. Generally Neural Networks learns from example. It is given some training set and from that training set, it generates a set of weight vectors. This training is can be done using various learning algorithm. In our work, we have use backpropagation algorithm which is a popular method for designing a multi layer neural network.[15] A Neural Network system has some layer. Each layer contains some neurons. The first layer is called input layer and the last layer is called output layer. The other layers that lies in between is called hidden layers. Each neuron is given a vector from previous layer as input and then it passes the value to a non-linear transfer function and generates a input for the next layer.[16] The transfer function, by which the neuron takes input and gives a output

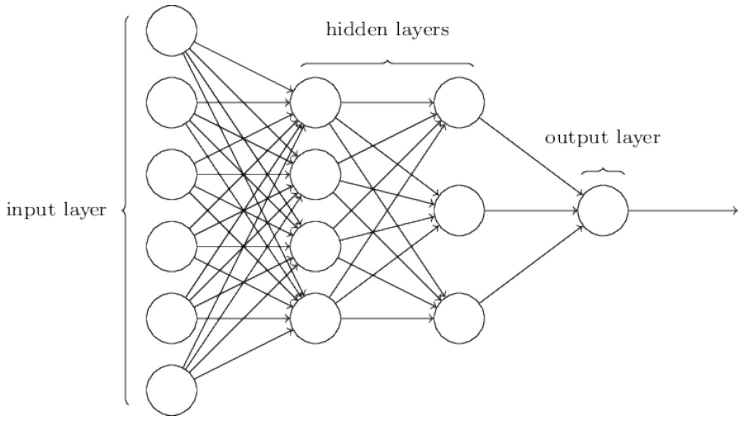

Figure 3: Neural Network

is also called activation function. There are many activation functions like Logistic or Sigmoid, Softsign[17], Inverse square root unit[18], rectified linear unit(ReLU)[19], Exponential Linear Unit (ELU)[20] In our work we have used Sigmoid as transfer function. Also used backpropagation algorithm as it is capable of handling large problems.[21]

5 Hyperparameters

Hyperparamters basically are the tuning knobs of a learning algorithm. There parameters are set before the training of the machine learning model.[22] By changing the parameters, we can tune the model.[23] After many trial and error, we have set these hyper parameter values.

5.1 For Random Forest

The number of trees in the forest = 10
criterion = gini
Minimum samples split = 2
Minimum Samples Leaf = 1
Min Weight Fraction Leaf = 0
min impurity decrease = 0
The number of jobs to run in parallel for both fit and predict = 2

5.2 Artificial Neural Network

Hidden Layer Size = 30, 30, 30
Activation = Logistic(Sigmoid)
The Solver for Weight Optimization = stochastic gradient descent
Learning Rate = 0.001
Tolerance = 1e-4

6 Model Evaluations

After testing the model, we have tested all our classifiers using the test dataset of MNIST. The following tables will show our findings on this test. We not only calculated the error rate of the model, but also Mean Accuracy, F1 Score, Recall Score and Confusion matrix.

6.1 Precision Score

Classifier	Cls 0	Cls 1	Cls 2	Cls 3	Cls 4	Cls 5
RF	0.95829196	0.97650131	0.92701422	0.92625369	0.95829095	0.94557823
ANN	0.93841336	0.94482759	0.80925401	0.8387698	0.89559877	0.80462725

Classifier	Cls 6	Cls 7	Cls 8	Cls 8	Cls 9
RF	0.95829196	0.97650131	0.92701422	0.92625369	0.95829095
ANN	0.93841336	0.94482759	0.80925401	0.8387698	0.89559877

6.2 Average Accuracy

Random Forest	Artificial Neural Netwrok	K-Nearest Neighbor
0.875087508750875	0.9478947894789479	0.0

6.3 F1 Score

Classifier	Cls 0	Cls 1	Cls 2	Cls 3	Cls 4	Cls 5
RF	0.96203796	0.98380744	0.93937947	0.9286066	0.94977169	0.92699491
ANN	0.92823955	0.95511983	0.81970349	0.86413826	0.89331291	0.7497006

Classifier	Cls 6	Cls 7	Cls 8	Cls 9
RF	0.96675462	0.95022178	0.93067227	0.9341258
ANN	0.92169312	0.91530461	0.84148728	0.83781095

6.4 Recall Score

Classifier	Cls 0	Cls 1	Cls 2	Cls 3	Cls 4	Cls 5
RF	0.98365679	0.99030837	0.95348837	0.93366337	0.95315682	0.91816143
ANN	0.91828396	0.96563877	0.83042636	0.89108911	0.8910387	0.70179372

6.5 Confusion Matrices

6.5.1 For Random Forest

$$CM = \begin{bmatrix} 965 & 0 & 1 & 1 & 1 & 2 & 3 & 1 & 5 & 0 \\ 0 & 1118 & 3 & 2 & 1 & 2 & 2 & 1 & 6 & 0 \\ 6 & 4 & 984 & 6 & 6 & 1 & 4 & 12 & 5 & 4 \\ 1 & 3 & 11 & 958 & 1 & 13 & 0 & 7 & 10 & 6 \\ 5 & 2 & 4 & 0 & 937 & 2 & 4 & 4 & 5 & 19 \\ 6 & 1 & 2 & 26 & 5 & 836 & 4 & 1 & 4 & 7 \\ 12 & 6 & 0 & 0 & 4 & 11 & 920 & 1 & 4 & 0 \\ 5 & 6 & 25 & 9 & 4 & 0 & 0 & 962 & 3 & 14 \\ 14 & 1 & 13 & 29 & 10 & 21 & 8 & 6 & 863 & 9 \end{bmatrix}$$

6.5.2 For Artificial Neural Network

$$CM = \begin{bmatrix} 899 & 0 & 43 & 2 & 0 & 23 & 8 & 1 & 2 & 1 \\ 0 & 1096 & 2 & 2 & 0 & 1 & 1 & 1 & 30 & 2 \\ 13 & 3 & 857 & 3 & 8 & 43 & 25 & 7 & 53 & 20 \\ 1 & 5 & 9 & 900 & 0 & 55 & 0 & 7 & 33 & 0 \\ 0 & 0 & 10 & 0 & 875 & 0 & 19 & 4 & 5 & 69 \\ 21 & 4 & 52 & 142 & 0 & 626 & 5 & 0 & 38 & 4 \\ 18 & 1 & 38 & 0 & 20 & 3 & 871 & 0 & 5 & 2 \\ 1 & 31 & 3 & 1 & 2 & 0 & 1 & 924 & 19 & 46 \\ 4 & 14 & 37 & 18 & 6 & 13 & 0 & 7 & 860 & 15 \\ 1 & 6 & 8 & 5 & 66 & 14 & 2 & 40 & 25 & 842 \end{bmatrix}$$

6.6 Overall Running Time

Random Forest = *2.2805209159851074 s*
ANN = *82.07250833511353 s*

Hardware Specification

Processor: Intel Corei7
Speed: 3.5GHz
RAM: 16 GiB DDR3
OS: macOS High Sierra

7 Discussion

From the above scores, it's clear that Random forest is showing much greater performance than the Artificifial Neural Network in this case. The average accuracy on MNIST Test Data Set, which is considered as one of the most vital factor of a model, is quiet good in Random Forest where ANN didn't show a good result. Though Neural Network showed a bit better result in labeling some characters, but in general, Random Forest did best. Though time required for training and evaluating totally depends on the performance of the hardware, random forest also wins here. But it's natural because neural network requires much computing power while training. It can be greatly improved by using GPU instead of CPU.[24] But scikit learn doesn't support GPU. We could use other libraries that supports GPU for training, but as our work mainly focus on the performance of the classifier, we didn't bother much about training and testing time required.

8 Conclusion

Our work isn't completed yet. We have to find more optimized hyper parameters for the classifiers. Here, we have just compared only two of the classifiers. But there are many other classifiers that are also needed to be considered. Better versions of Artificial Neural Network are there like Convolutional Neural Net(CNN), Reinforcement Neural Net(RNN), Deep Learning etc. This shows better performance in image categorizing or labeling. So we also want to work on those in future.

References

[1] U. Pal, T. Wakabayashi, and F. Kimura, "Comparative study of devnagari handwritten character recognition using different feature and classifiers," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pp. 1111–1115, IEEE, 2009.

[2] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 1, pp. 63–84, 2000.

[3] Y. Baştanlar and M. Özuysal, "Introduction to machine learning," in *miRNomics: MicroRNA Biology and Computational Analysis*, pp. 105–128, Springer, 2014.

[4] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[5] Y. LeCun, L. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, *et al.*, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural networks: the statistical mechanics perspective*, vol. 261, p. 276, 1995.

[6] E. Kussul and T. Baidyk, "Improved method of handwritten digit recognition tested on mnist database," *Image and Vision Computing*, vol. 22, no. 12, pp. 971–981, 2004.

[7] Y. LeCun, "Courant institute, nyu corinna cortes, google labs, new york the mnist database of handwritten digits."

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[9] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[10] G. Hackeling, *Mastering Machine Learning with scikit-learn.* Packt Publishing Ltd, 2014.

[11] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[12] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[13] Wikipedia, "Decision tree learning," 2018. [Online; accessed 05-June-2018].

[14] M. vanGerven and S. Bohte, "Artificial neural networks as models of neural information processing: Editorial on the research topic artificial neural networks as models of neural information processing," 2017.

[15] H. Leung and S. Haykin, "The complex backpropagation algorithm," *IEEE Transactions on Signal Processing*, vol. 39, no. 9, pp. 2101–2104, 1991.

[16] M. T. Hagan, H. B. Demuth, M. H. Beale, *et al.*, *Neural network design*, vol. 20. Pws Pub. Boston, 1996.

[17] J. Bergstra, G. Desjardins, P. Lamblin, and Y. Bengio, "Quadratic polynomials learn better image features," *Technical report, 1337*, 2009.

[18] B. Carlile, G. Delamarter, P. Kinney, A. Marti, and B. Whitney, "Improving deep learning by inverse square root linear units (isrlus)," *arXiv preprint arXiv:1710.09967*, 2017.

[19] G. E. Hinton, "Rectified linear units improve restricted boltzmann machines vinod nair," 2010.

[20] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[21] R. Hecht-Nielsen, “Theory of the backpropagation neural network,” in *Neural networks for perception*, pp. 65–93, Elsevier, 1992.

[22] Wikipedia, “Hyperparameter (machine learning) — Wikipedia, the free encyclopedia.” `http://en.wikipedia.org/w/index.php?title=Hyperparameter%20(machine%20learning)&oldid=843195812`, 2018. [Online; accessed 13-June-2018].

[23] F. Hutter, H. Hoos, and K. Leyton-Brown, “An efficient approach for assessing hyperparameter importance,” in *International Conference on Machine Learning*, pp. 754–762, 2014.

[24] B. J. Erickson, P. Korfiatis, Z. Akkus, T. Kline, and K. Philbrick, “Toolkits and libraries for deep learning,” *Journal of digital imaging*, vol. 30, no. 4, pp. 400–405, 2017.