Comparing Convolutional Neural Network and Softmax Regression for MNIST dataset

Manasi P. Paste
Department of Computer Science
South Dakota School of Mines and Technology
Rapid City, SD, 57701
manasi.paste@gmail.com

Abstract

This study aims to perform a comparison between the Convolution Neural Network and Soft-max regression on the MNIST dataset. The paper goes over a brief background of MNIST dataset and Tensor Flow framework. Convolution Neural Network are designed to handle 2D shapes. Soft-max regression is a generalization of logistic regression to handle classification in to multiple classes. The paper explains these two models' architecture as they are coded up in the python in TensorFlow framework. We find that CNN performs better than soft-max regression. This paper also compares and explains different existing models for MNIST dataset. For educational purposes, this paper explains different components of a neural net like activation function, learning rate, stochastic gradient descent etc. Further scope of experiments for CNN is discussed along with its area of applications.

Keywords: Convolutional Neural Network, Softmax Regression, Dropout, Stochastic Gradient Descent, Cross Entropy, Learning Rate, Back propagation. Principal Component analysis

1 Introduction

1.1 Dataset

MNIST (modified National Institute of Standards and Technology) dataset is a popular collection of handwritten digit images used widely in optical character recognition and machine learning research[1]. The training dataset was collected from American Census Bureau while the testing dataset was collected from American high school students[2]

http://yann.lecun.com/exdb/mnist

Each handwritten digit image is a gray scale image of size 28 X 28 pixels. The labels of each image is an integer from 0 to 9. The database size is 70000, with training dataset size is 60000 and test dataset size is 10000.

In 1998, the highest error rate was listed as 12%. Since then a lot of improvement has been observed in the performances of algorithms applied to MNIST dataset. In 2012, using convolutional nets with width normalization gave error rate of 0.23% [2]. Since the purpose of this study was to understand convolutional nets and regression models, MNIST database was chosen as its both popularly used and simpler.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

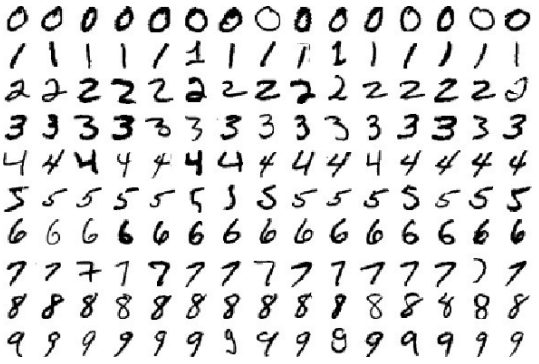

Figure 1: MNIST dataset sample images

1.2 Framework

For this project TensorFlow is used. TensorFlow is an open-source software library that is widely used for machine learning applications such as neural network. Its library allows high performance numerical computations. TensorFlow has well designed Python API as well as C++, Java and Go APIs[3].

In 2011, Google Mind constructed a machine learning framework called DistBelief. The back-propagation usage alongside different upgrades by the Google group saw 25% lessening in errors in the Speech recognition model[3]. TensorFlow is Google Brain's second generation system, which got its name from the fact that neural networks do operations on multidimensional data arrays, often referred to as "tensors"[3]

`https://github.com/tensorflow/tensorflow`

2 Models

For this study, Convolutional Neural Network (CNN) and Regression model were chosen for comparison. The code for these models was replicated from TensorFlow repository. Softmax-regression code was modified to plot graphs.

2.1 Convolutional Neural Network

CNN is designed to take advantage of 2D structure of an input image. It consists of a number of convolutional and pooling layers. A convolutional layer will have a certain number of filters, each representing a certain feature for example straight edges, colors or curve detectors. For MNIST dataset, the original image is of size 28 X 28 X 1. The first convolutional layer has 32 filters each of size 5 x 5 and uses ReLU as the activation function. This should have reduced the shape to 24 X 24. But in order to preserve the size of the original image the filter perimeter is padded with 0s. So we have 28 X 28 X 32 output from the first convolutional layer.

Pooling layer subsamples from each of the feature map produced by the convolutional layer[4]. The pooling layer is of size 2X2 and has the stride set to 2. It picks up the max value from the 2 X 2 matrix it hovers over. this is called max pooling. There are other ways to do the pooling like taking an average but max pooling was preferred because it takes existing values from the matrices. Averaging might give us a value that might not one of the values from the matrix. This pooling layer will reduce the input layer by 50%. So now we have output of the shape 14 X 14 X 32.

The second convolutional layer has 64 filters with size 5 X 5. Again the image is padded with zeros around the border to preserve the shape. We get an output of 14 X 14 X 64 from the second convolutional layer. The second pooling layer again reduces the output size to 7 X 7 X 64. This layer is followed by a dense layer of 1024 neurons and uses ReLU as an activation function. ReLU (Rectified Linear Units) as the advantage that it trains the network many times faster than other activation function[5].

$$f(x) = max(x, 0)$$

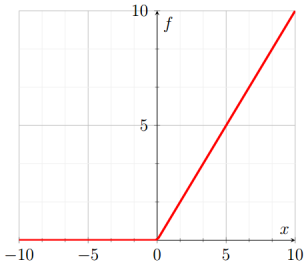

Figure 2: ReLU graph

Over fitting the data is one of the major issues with neural networks. We observe reduced errors in the training set but testing set errors increase. The method of preventing over fitting is called regularization. To over come over fitting we need to make the model generalize. **Dropout** is the technique used to prevent over fitting in the dense layer of our convolutional neural network. This technique basically randomly drops units from the neural network. Dropout rate is a hyper-parameter that defines the probability of retaining a unit[6], each one mapping to classes from 0 to 9. Sometimes over fitting can be taken care of by simply stopping the training early. The number of training steps is just another hyper-parameter that we can tune for **early stopping**. **Bootstrapping** is another regularization which combines different models to vote on the test output. This method assumes that not all the models will not make same errors.

To optimize the neural network **Stochastic Gradient Descent(SGD)** algorithm was used. SGD updates parameters with respect to few examples or mini batches instead of entire training set[7]. SGD comes with two advantages. One, it leads to stable convergence to the minima and two, it is computationally light[7].

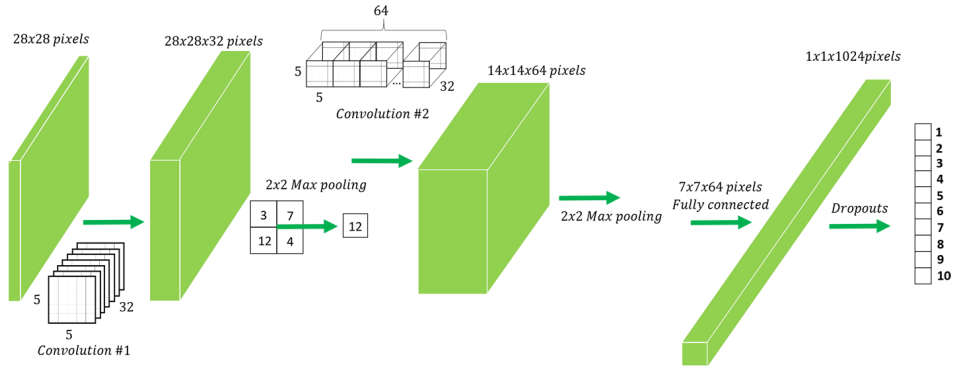

Figure 3: Visualization of CNN architecture

2.2 Regression model

Simple regression model is usually Y = WX + b. But since we want to perform multi-class classification, Softmax regression model was chosen. Softmax regression model applies softmax function to the regression model output.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

Figure 4: Visualization of Softmax regression model

Softmax function, also known as normalized exponential function, converts a N-dimensional vector of real values to a N-dimensional vector (σ (z)) of real values in the range (0, 1) [8]. The advantages of softmax function is that it gives values from 0 to 1 and that all the values sum up to 1. So this value is treated as the likelihood (probability) of a certain input being classified into a given class. The formula for softmax function:

$$\sigma(z) = \frac{e^{z_j}}{\Sigma_{k=1}^{K} e^{z_k}}$$

for j = 1 ... N

This model flattened the input images into a 784 - dimensional vector. The weights and bias are initialized to zero. The weight matrix, W, has a shape [784, 10]. When you multiply the 784-dimensional input matrix it produces 10- dimensional vector. The bias, b, has shape of [10].

Cross entropy, measure of how inefficient our predictions are for describing the truth, is calculated[5]. Gradient descent optimizer is used to minimize the cross entropy. It modifies the variables (W, b) and reduce the loss.The parameter for this optimizer is learning rate. This uses an algorithm called **back propagation**. Back - prop algorithm is used to adjust the weights during the training process. The first part is forward pass, where the input is feed into the model and output is obtained. Then, a loss function is used to calculate the error. Most commonly used loss function is MSE (mean squared error). The reason for opting to square the errors instead of just taking Formula for sum of squared errors:

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

Then gradient dE/dW is calculated and passed backward through the network to update the weights such that the loss is reduced. Therefore we always move along the negative gradient. The equation to update weights is:

$$w = w_i - learning_Rate \frac{dl}{dW}$$

Learning rate is a hyper-parameter that controls how much we are adjusting the weights/biases of our network with respect to the loss gradient [9]. A very low rate means we will be traveling along the downward slope slowly, making sure we don't miss any local minima. But this is will take a long time for convergence. If learning rate is too large we can miss the minimum and fail to converge. This study has chosen 0.4 as a the learning rate. The graph (Fig 5) below explains the effect of learning rate of the loss rate.

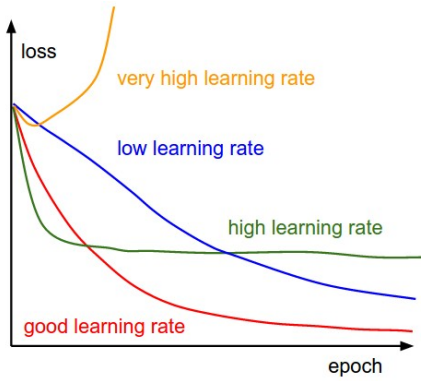

Figure 5: Visualization of how the choice of learning rate affects loss

3 Results

Accuracy rate for Convolutional Neural Network is approximate 97% and for the Softmax regression model is approximate 92%. Convolutional Neural Network runs for 20000 steps with a batch size of 100. The learning rate for this model 0.001. Softmax regression runs for 2000 steps with a batch size

of 100. Since MNIST a widely experimented dataset, there are numerous studies existing for it. Fig 6 is a graph plotted using *matplotlib* for the softmax regression model. This graph plots accuracy rate for each step of the training and validation step. The table below (Table 1) compares Test Error Rates for different models on MNIST.

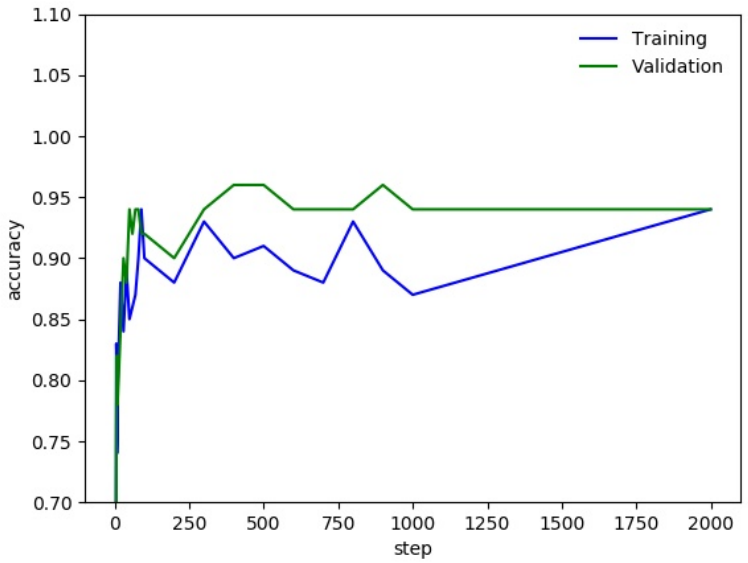

Figure 6: Accuracy rate vs Number of Steps for Training and Validation dataset

Table 1: Performance comparison of different models on MNIST[10]

Classifier	Preprocessing	Test Error Rates
Linear classifier (1-layer NN)	none	12.0
K-nearest-neighbours, Euclidean	deskewing	2.4
boosted stumps	none	7.7
40 PCA + quadratic classifier	none	3.3
Support Vector Machine, Gaussian Kernel	none	1.4
2-layer NN, 300 hidden units, mean square error	none	4.7
2-later NN, 300 hidden units	deskewing	1.6
Convolutional net LeNet-1	subsampling 16X16 pixels	1.7
Committee of 35 conv. net	width normalization	0.23

Preprocessing data is prepping of data before it is passed through the model. The purpose of this step is to either reduce features, clean the data for noise and missing points or to normalize it. De-skewing is a preprocessing method removing skewness (rotation/ slantness) of an image. This is done by rotating the image in an opposite direction but by same degrees as its skewness[11].

4 Discussion/ Conclusion

According to the validation accuracies of the two models, Convolutional Neural Network performs better in classifying the MNIST dataset than the regression model. Regression model CNN architecture has many advantages over the linear regression model. Unlike regression model, CNN preserves the 2D structure of the images and takes its advantage of it. It is capable of handling shifts and distortion in the images. Convolutional layer uses same coefficients across different locations in the space, therefore reducing memory requirement drastically[12].

The convolutional neural network in this experiment has accuracy rate of 97%. Other studies have CNN models that can achieve an accuracy rate of 99.77% for the MNIST database of handwritten digits. It is not only that CNN outperforms other algorithms when it comes to images, they even outperform humans such as in the task of classifying breeds of dogs or species of birds[12]

Linear Classifiers are the simplest neural networks. Each pixel contributes its to the weighted sum for each output unit. With 7850 free parameters, this classifier has an error rate of 12%[10]. K-nearest neighbors algorithm measures the Euclidean distance between the input images. In spite of the fact this model requires little training time, it has a large memory and recognition time requirement. Because the entire training set of 60,000 images must be made available at the run time[10].

Principal component analysis (PCA) is a statistical technique used to find patterns in the dataset by emphasizing on its variation. For MNIST data in the preprocessing stage, projection of the input pattern on the 40 principal components of the set of training vectors. Principal components are computed by subtracting the mean of each input component from the training vectors. Singular Value Decomposition is used to compute and diagonalize the covariance matrix of the resulting vectors [10].

Boosting is a method that combines multiple classifiers. Even though at first it looks like training data in multiple classifiers can be time expensive, but onces one of the classifiers gives out a very high confidence answer, the remaining classifiers are not run[10]. **Support Vector Machine(SVM)** is an efficient technique of representing complex surfaces in high-dimensional space. SVM is a good replacement for polynomial classifiers that our impractical for higher dimensional spaces[10].

Further study can be carried out by adding more convolutional layers and comparing their performances. Regression models can be further experimented by studying the performances at different learning rates for Gradient Descent Optimizer. Also, Tensor-board can be used to view live accuracy rate as the model is training.

Convolutional Neural Network has **hyper-parameters** like learning rate, number of epochs (number of times entire training set is passed through the model), batch size, Weight initialization and dropout rate.For further study, we can attempt to tune the hyper parameters for CNN. Though tuning hyper-parameter can be time consuming there are some efficient way to tune them.

- Learning rate can be optimized using optimizers like SGD, Adam, AdaDelta etc[13].
- Number of epoch have to be increased until the gap between test data error and training data error reduced to minimum possible size [13].
- An ideal batch size is usually 16 to 128[13].
- For weight initialization, uniform distribution usually works well[13].
- 0.5 is an ideal dropout rate[13].

There are numerous advanced applications of CNN. Face recognition uses CNN along with self organizing map neural network and we have seen significant results [14]. Some other applications are labeling scenes, action recognition, document analysis etc. Recently, CNN has been revolutionizing natural language processing too as its applications are found in speech recognition and text classification [14]. In conclusion we can say that when it comes to computer vision and natural speech recognition, Convolutional Neural Network has outperformed other models.

References

[1] L. Deng, "The mnist database of handwritten digit images for machine learning research," 2012. [Online]. Available: ieeexplore.ieee.org/document/6296535/

[2] "Mnist database." [Online]. Available: https://en.wikipedia.org/wiki/MNIST_database

[3] "Tensorflow." [Online]. Available: https://en.wikipedia.org/wiki/TensorFlow

[4] "Unsupervised feature learning and deep learning tutorial - pooling." [Online]. Available: ufldl.stanford.edu/tutorial/supervised/Pooling/

[5] TensorFlow, "Mnist for ml beginners." [Online]. Available: https://www.tensorflow.org/versions/r1.1/get_started/mnist/beginners

[6] I. S. Nitish Srivastava, Alex Krizhevsky, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*. [Online]. Available: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

[7] "Unsupervised feature learning and deep learning tutorial - stochastic gradient descent." [Online]. Available: http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/

[8] "Unsupervised feature learning and deep learning tutorial - softmax regression." [Online]. Available: http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/

[9] H. Zulkifli, "Understanding learning rates and how it improves performance in deep learning." [Online]. Available: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[11] "Deskewing." [Online]. Available: https://www.leadtools.com/help/leadtools/v19/dh/to/deskewing.html

[12] R. K. S. Hijazi, "Using convolutional neural networks for image recognition." [Online]. Available: https://ip.cadence.com/uploads/901/cnn_wp-pdf

[13] S. Lau, "A walkthrough of convolutional neural network - hyperparameter tuning." [Online]. Available: https://towardsdatascience.com/a-walkthrough-of-convolutional-neural-network-7f474f91d7bd

[14] P. G. R. C. Ashwin Bhandare, Maithili Bhide, "Applications of convolutional neural networks," 2016. [Online]. Available: https://pdfs.semanticscholar.org/89db/184cb8b1397a9aa35693f8dc03e1e728992a.pdf