CSE 574: Project 3

Project Report

Submission Date: Dec 9 2015

Suramrit Singh/5016 9918/suramrit@buffalo.edu

1. Overview

The project required implementation of classification algorithms. The classification task was to recognize a 28 X 28 grey scale handwritten image and identify it as a digit among 0,1,2, ... ,9 The images used were MINST digit images.
The project required the following four tasks:

1. Train a linear logistic regression model and train it on MINST dataset using mini-batch gradient update and tune hyper-parameters.
2. Implement single hidden layer neural network, train it on the MNIST digit images and tune hyper-parameters such as the numbers of the units in the hidden layer.
3. Use a publicly available convolutional neural network pakage, train it on MNIST digit images and tune hyper-parameters.

1.1 Background

Terms and Concepts:

Logistic Regression:

Using 1-of-K coding scheme for our multiclass classification task. Our multiclass logistic regression model has the form.

$$\boldsymbol{p}(\boldsymbol{C}_{\boldsymbol{k}}|\boldsymbol{x}) = \boldsymbol{y}_{\boldsymbol{k}}(\boldsymbol{x}) = \frac{(\mathbf{exp}(\boldsymbol{a}_{\boldsymbol{k}}))}{\sum_{\boldsymbol{j}} \mathbf{exp}(\boldsymbol{a}_{\boldsymbol{j}}))}$$

called **soft-max regression**.

(1)

Where the activation a_k are given by $a_k = w_k + b_k$.

Here w(k) are the weights assigned to input parameters, which in our case are pixel values. And b(k) is the bias term associated with each class.

The cross-entropy error function for multiclass classification problem for a training sample **x** will be:

$$\boldsymbol{\Delta w_j E(x) = \sum_{k=1}^{K} t_k \, ln y_k} \tag{2}$$

The gradient of the error function in (2) then becomes

$$\boldsymbol{\Delta w_j E(x) = (y(j) - t(j))x} \tag{3}$$

We then use stochastic gradient descent, which uses first order derivatives to update the weight values.

$$\boldsymbol{w^{(\tau+1)} = w^{(\tau)} + \Delta w^{(\tau)}} \tag{4}$$

Where $\Delta \mathbf{w(\tau)} = -\boldsymbol{\eta}\ \mathbf{(\tau)\nabla E}$ are the weight updates. We then use the learning rate to converge to optimum values of weights that give us the solution for *wj*. We use a specific form of stochastic gradient descent called mini batch gradient descent. IN mini batch, we

With each iteration over the entire dataset, we change the parameters like batch size, learning rate **η** and number of passes over the data set.

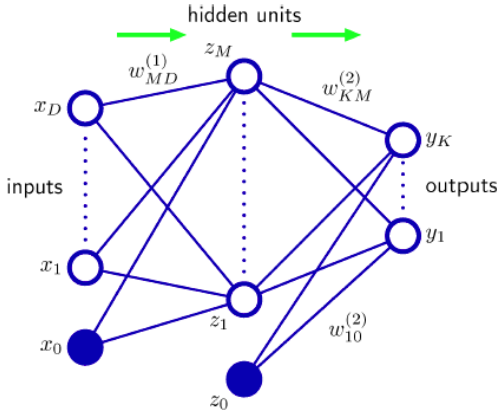
Single Layered Neural Network:

We have used a neural network with one hidden layer. The feed forward propagation for the single layered network is given by:

$$\boldsymbol{z_j = h(\sum_{i=1}^{D} w_{ji} x_i + b_j)} \tag{5}$$

Here z_j are the activation of the hidden layer and h(.) is the activation function fir the hidden layer. In this project we have used hyperbolic tangent as the activation function.

The backpropagation is done as follows,



$$\delta_k = y_k - t_k$$

$$\delta_j = h'(z_j) \sum_{k=1}^{K} w_{kj} \delta_k$$

The gradient of the error function is given by:

$$\frac{\partial E}{\partial w_{ji}^{(1)}} = \delta_j x_i, \qquad \frac{\partial E}{\partial w_{kj}^{(2)}} = \delta_k z_j$$

We then use stochastic gradient descent to train the neural network.

Convolutional Neural Network:

In order to implement convolutional neural network we use a public neural network package, DeepLearnToolbox by rasmusbergpalm. This library is publicly available at: https://github.com/rasmusbergpalm/DeepLearnToolbox/

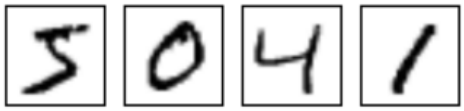
Filtering and Masking

In image processing, a common operation is to apply a filter to an image. The filter has a small "filter mask" or "kernel" which is "convolved" with the image.

This is the same basic operation as an MLP neuron–each of the corresponding components are multiplied together, then summed up.

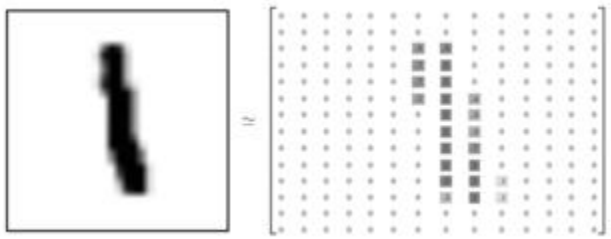
Data Set:

MNIST is a simple computer vision dataset. It consists of images of handwritten digits like these:

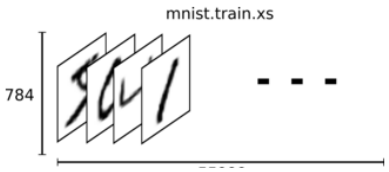


It also includes labels for each image, telling us which digit it is. For example, the labels for the above images are 5, 0, 4, and 1.

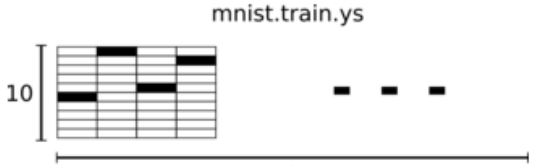
The dataset is split into 2 parts 60,000 data points of training data and 10,000 points of test data. Every MNIST data point has two parts: an image of a handwritten digit and a corresponding label. Each image is 28 pixels by 28 pixels. We can interpret this as a big array of numbers:



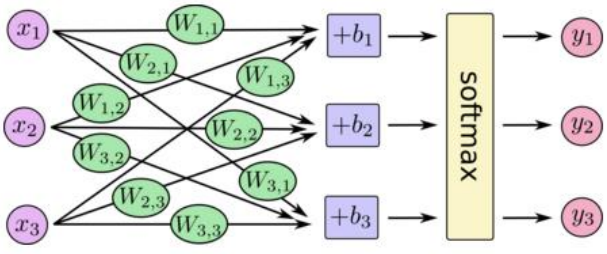
We can flatten this array into a vector of 28x28 = 784 numbers. The result is that mnist.train.images is a tensor (an n-dimensional array) with a shape of [60000, 784]. The first dimension indexes the images and the second dimension indexes the pixels in each image. Each entry in the tensor is the pixel intensity between 0 and 1, for a particular pixel in a particular image.



The corresponding labels in MNIST are numbers between 0 and 9, describing which digit a given image is of. In this case, the nth digit will be represented as a vector which is 1 in the nth dimensions. For example, 3 would be [0,0,0,1,0,0,0,0,0,0]



The softmax regression of eq (1) can then be visualized as,



and similarly the input tensor corresponds to the input layer in pour neural network.

Results and Observations:

1. Logistic Regression:

Approach: We randomly assign the weight and bias values initially and then run logistic regression over the entire dataset for a fixed number of passes. We keep the learning rate constant initially and then decrease the learning rate slightly whenever the weights are updated.

Results:

The parameter values that gave the best values(least misclassification error) were:

Learning Rate	2.6
Batch size for gradient descent	380
No of passes	10

The weights are found to be in the range:

Weight Minimum	-2.1949
Weight Maximum	3.8064

And the classification errors were found to be:

Misclassification on Training Data	~18%
Misclassification on Test Data	~18.5%

2. Single-Layered Neural Network:

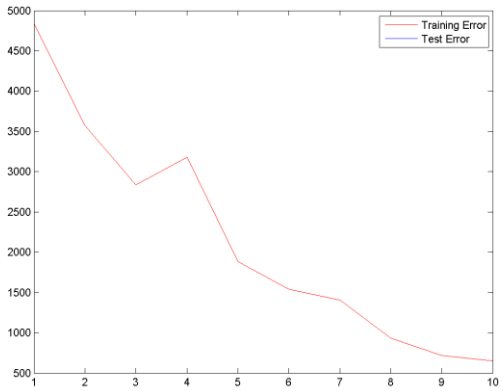
Approach: Similar to logistic regression, we assign random initial value to the weights and bias values for both the weights between the input layer and the hidden layer, and the hidden layer and the output layer of the neural network. We then take initial values of parameters and then train our model. We then divide our data into batches and call the forward propagation calculation for this particular batch, using hyperbolic tangent as our activation function.

We then call the gradient function to obtain the required gradient updates, with the differential of tanh function as:

$$\Delta(\tan h) = 1 - h^2$$

We then update the weights of the 2 sets of weights between the layers.

The figure shows Training Error and Test Error.



The weights were found to be in the range:

Weight Maximum	-2.1649
Weight Minimum	3.8064

The parameter values that gave the best values(least misclassification error) were:

Learning Rate	4.7
Batch Size	1000
No of Passes	10
No of units in the Hidden layer	500

The classification errors were found to be:

Misclassification on training set	~2%
Misclassification on test set	~6%

3. Convoluted Neural Network using DeepLearn

We used the DeepLearn tool for application of a convoluted neural network over the MNIST dataset. The parameters chosen were:

Learning Rate	1
Batch Size	50
No of passes(Epochs)	15
Layers	1(Input), 2(Convolution Layers), 3(Subsampling Layers)

We obtained an accuracy of 97.82% using the convoluted network.

Conclusion

We were able implement both logistic regression and a single layered neural network on MNIST Dataset and were able to tune the hyperparameters to reduce the misclassification error in logistic regression to 17% and in Neural Network to 6%. We also implement DeepLearn Tool in matlab to implement a convoluted neural network.

References

1] https://en.wikipedia.org/wiki/Hyperparameter_optimization

2] Christopher M. Bishop, Pattern Recognition and Machine Learning, Page 8-10.

3] https://en.wikipedia.org/wiki/Hyperparameter_optimization#Grid_search

4] Random Search for Hyper-Parameter Optimization ,James Bergstra Yoshua Bengio *et al. Journal of Machine Learning Research 13 (2012) 281-305*

5] https://chrisjmccormick.wordpress.com/2015/01/10/understanding-the-deeplearntoolbox-cnn-example/

6] https://www.tensorflow.org/versions/master/tutorials/mnist/beginners/index.html

7] http://deeplearning.net/tutorial/deeplearning.pdf