Classification of Handwritten Digits

CSE 574 : Project #3

Adarsh Prakash | UBIT Name: adarshpr | Person #: 5020 8760

December 11, 2016

1 Overview

The goal of this project is to implement and evaluate Logistic Regression, Single Layer Neural Network and Convolutional Neural Network in recognizing handwritten digits. The MNIST dataset of images has been used to train, test and validate these models. An evaluation on data set of images from USPS has also been performed. This document briefly summarizes the evaluation of these models and their performance.

2 Software/Libraries Used

Python Version: 2.7
Python Modules: numpy, tensorflow, cv2, matplotlib

3 Data

3.1 MNIST

For training, testing and validating the models, the pickle file from the project documents - **mnist.pkl.gz** has been used. Please find the summary of the split of the data below (as provided):

- Training set: 50000 samples each with 784 features + respective labels
- Testing set: 10000 samples each with 784 features + respective labels
- Testing set: 10000 samples each with 784 features + respective labels

3.2 USPS

Unlike MNIST data set, USPS images are not normalized. So, images were processed as detailed by LeCunn for preparing the MNIST data set:

- Find the bounding box for the digit images
- Resize the bounded image and center it on (20 x 20) tile
- Add padding to prepare the final (28 x 28) image

After processing the images as described above, image vectors have been dumped into pickle file **usps_data.pkl.gz** for ease of use while training.

(Both **usps_data.pkl.gz** and the code for processing images and creating image vectors is available with this submission - *usps_process.ipynb* and *usps_dump.ipynb*)

4 Models

4.1 Logistic Regression

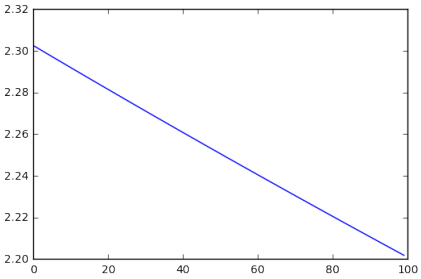
Hypothesis used,

$$y_k(x) = \frac{exp(a_k)}{\Sigma_j exp(a_j)}$$

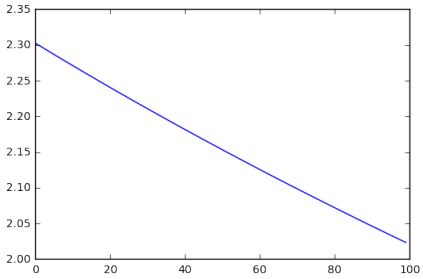
where, $a_k = W^T X + b_k$ and b_k is the bias for class k. The cost function used is shown below:

$$E(x) = -\Sigma_k t_k ln y_k$$

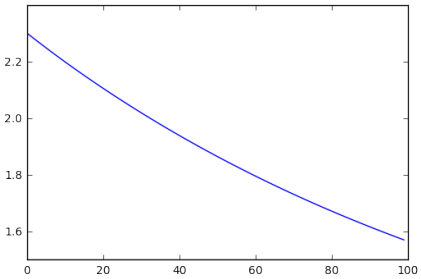
The following plots of Cost v/s Iterations detail the tuning of learning rate, α:



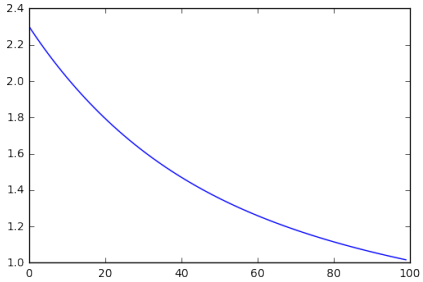
(a) $\alpha = 0.001$ for 100 iterations



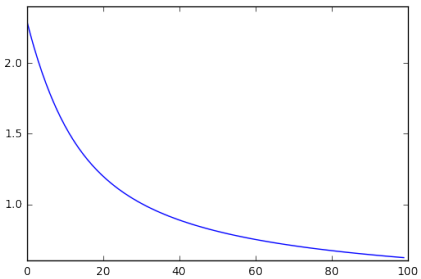
(b) $\alpha = 0.003$ for 100 iterations



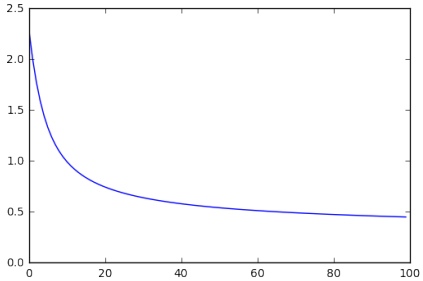
(c) $\alpha = 0.01$ for 100 iterations



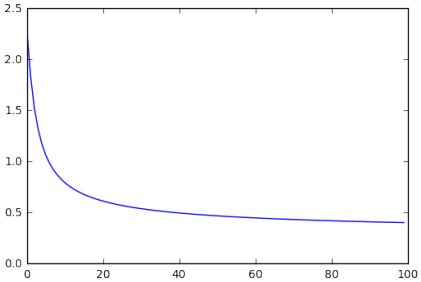
(d) $\alpha = 0.03$ for 100 iterations



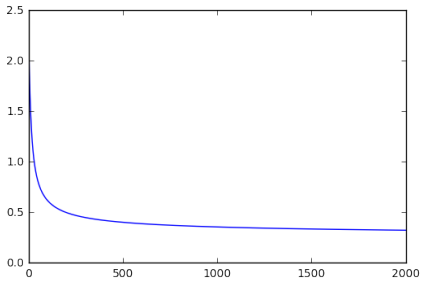
(e) $\alpha = 0.1$ for 100 iterations



(f) $\alpha = 0.3$ for 100 iterations



(g) $\alpha = 0.5$ for 100 iterations



(h) $\alpha = 0.5$ for 2000 iterations

Evaluation:

- Learning Rate, $\alpha = 0.5$
- Regularizer, $\lambda = 0.0003$
- Number of iterations = 2000
- Test Error: 0.0753
- Test Accuracy: 0.9247
- Validation Error: 0.0789
- Validation Accuracy: 0.9211
- USPS Error: 0.4181
- USPS Accuracy: 0.5819

4.2 Single Layer Neural Network

Activation of output layer happens by softmax,

$$Softmax, y_k(x) = \frac{exp(a_k)}{\Sigma_j exp(a_j)}$$

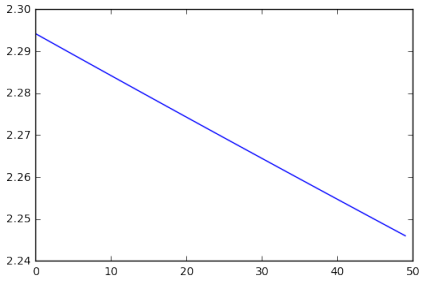
where, $a_k = \Sigma_{j=1}^{M} w_{kj}^{(2)} z_j + b_k^{(2)}$ and b_k is the bias for class k. Also, z_j is given by:

$$z_j = tanh(\Sigma_{i=1}^{D} w_{ji}^{(1)} x_i + b_j^{(1)})$$

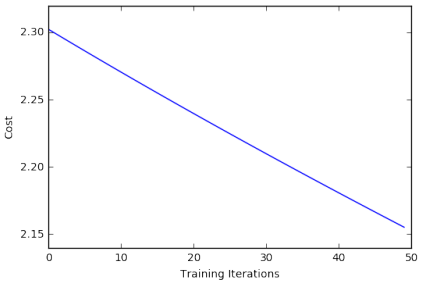
The cost function used is shown below:

$$E(x) = -\Sigma_k t_k ln y_k$$

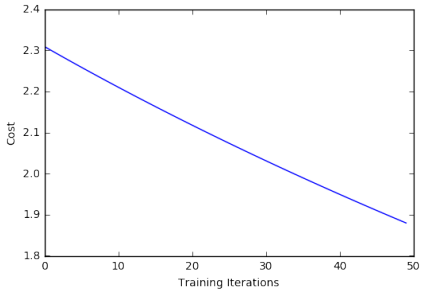
The following plots of Cost v/s Iterations detail the tuning of learning rate, α:



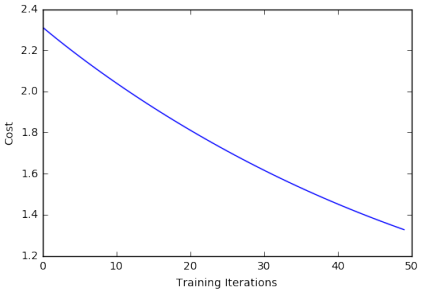
(a) $\alpha = 0.001$ for 50 iterations



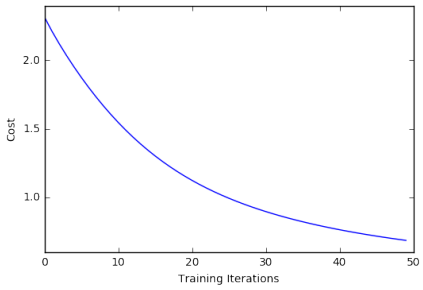
(b) $\alpha = 0.003$ for 50 iterations



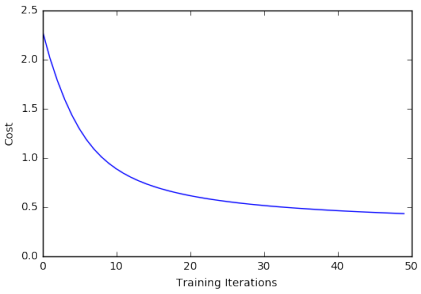
(c) $\alpha = 0.01$ for 50 iterations



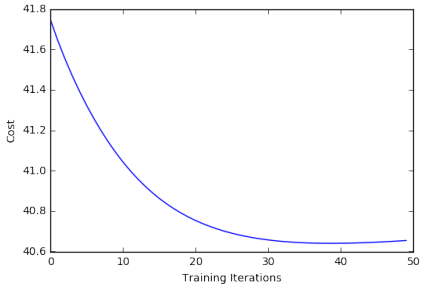
(d) $\alpha = 0.03$ for 50 iterations



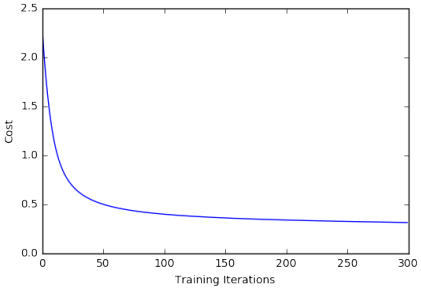
(e) $\alpha = 0.1$ for 50 iterations



(f) $\alpha = 0.3$ for 50 iterations



(g) $\alpha = 0.1$ - effect of regularization



(h) $\alpha = 0.5$ for 300 iterations

Evaluation:

- Learning Rate, $\alpha = 0.2$
- Regularizer, $\lambda = 0$
- Number of nodes in hidden layer = 785
- Number of iterations = 300
- Test Error: 0.0819
- Test Accuracy: 0.9181
- Validation Error: 0.0832
- Validation Accuracy: 0.9168
- USPS Error: 0.435
- USPS Accuracy: 0.565

4.3 Convolutional Neural Network

Tensorflow package was used to implement this model. Composition of the network is as follows:

- Input Layer
- First Convolution Layer
- Second Convolution Layer
- Dense Layer (with dropout adopted from tensorflow tutorial)
- Output Layer

The following table summarizes tuning of number of features for each image patch in First Convolution Layer and Second Convolution Layer:

# features in First Layer	# features in Second Layer	Validation Accuracy
64	128	0.9895
32	64	0.9906
32	32	0.9846
16	32	0.9891

Evaluation:

- Features per 5x5 image patch in First Convolution Layer = 32
- Features per 5x5 image patch in First Convolution Layer = 64
- Dropout threshold during training = 0.5 for each class
- Dropout threshold during test and validation = 1.0 for each class
- Test Accuracy: 0.9913
- Validation Accuracy: 0.9906
- USPS Accuracy: 0.875894

5 Conclusion

The following table summarizes the best results obtained during model evaluation:

	Testing Accuracy (%)	Validation Accuracy (%)	USPS Accuracy (%)
Logistic Regression	92.47	92.11	58.19
Single Layer NN	91.81	91.68	56.5
Convolutional NN	99.13	99.06	87.59

"No Free Lunch"
The previously summarized findings clearly support "no free lunch" theorem. Both Logistic Regression and Single Layer Neural Network, which try to generalize from training data perform similarly. However, Convolutional Neural Network which has been built to imitate function of human eye (5x5 image patches!), thereby using subtle domain bias, performs better (87%) relative to the other models.

6 References

[1] Lecture Notes, CSE 574, Prof. Sarguru H. Srihari

[2] Project Notes, CSE 574, Junfei Wang, Jun Chu and Kyung Won Lee

[3] Pattern Recognition and Machine Learning, Christopher M. Bishop

[4] NumPy v1.10 Manual, SciPy
`http://docs.scipy.org/doc/numpy-1.10.0/`

[5] Tensorflow Tutorials
`https://www.tensorflow.org/versions/r0.12/tutorials/index.html`