Project Report 3
Classification

CSE-574 - Fall 2017

Deepika Chaudhary, Manpreet Dhanjal

(UB IT Name: d25, PN: 50248725), (UB IT Name: dhanjal, PN: 50248990)

Introduction:

This project asks us to implement classification of hand written digits using logistic regression, single layer neural network and convolutional neural network using tensor flow library. The training was done on MNIST dataset and in the end the trained models were tested with UPS dataset to check the validity of 'no free lunch theorem'.

Data Partition:

The MNIST has 2 subsets – training dataset containing 60000 images and testing dataset containing 10000 images. We further divided the training dataset into 2 subsets, one for training and one for validation and tuning of hyper parameters. The validation set constituted of about 10% of training dataset.

Logistic Regression:

We used 1-of-K coding scheme t = [t1, ..., tK]. for our multiclass classification task. The model can be represented as:

$$p\left(\mathcal{C}_k|\mathbf{x}\right) = y_k\left(\mathbf{x}\right) = \frac{\exp\left(a_k\right)}{\sum_j \exp\left(a_j\right)}$$

where the activation a_k are given by $a_k = w^{\top}_k x + b_k$.

The gradient of error is given by

$$\nabla_{\mathbf{w}_j} E\left(\mathbf{x}\right) = \left(y_j - t_j\right)\mathbf{x}$$

The weights are updated by the following method

$$\mathbf{w}_j^{t+1} = \mathbf{w}_j^t - \eta \nabla_{\mathbf{w}_j} E\left(\mathbf{x}\right)$$

We used Mini batch SGD method to update the weights. The computation of gradient error can be done by matrix multiplication and dividing the data in batches largely out performs the speed of computing the gradient error.

MNIST DATA:

The input has 60000 x 784 features which has to be classified into 10 models

Hyper Parameter Tuning: Lambda and Learning Rate(Eta)

The learning rate and lambda are selected via grid search. The model is first trained using the training data of around 55000. The weight estimated through this training is then given to validation data and the accuracy is seen in all the combinations of lambda and eta. The maximum accuracy was estimated to be 96.3% on validation data at eta = 0.9 and lambda=0.01.

Lambda →

<--Learning Rate

	0.001	0.01	0.1
0.01	0.9358	0.9358	0.9358
0.03	0.9388	0.9388	0.9388
0.09	0.9424	0.9424	0.9424
0.3	0.9516	0.9516	0.9516
0.9	0.963	0.963	0.963

Table: Grid Search

Accuracy on training data = 92.15%
Accuracy on validation data = 96.3%
Accuracy on testing data = 92.26%

Results:

```
Final eta is0.9
Final lambda is 0.01
Training accuracy is 0.9215
Testing accuracy is 0.9226
```

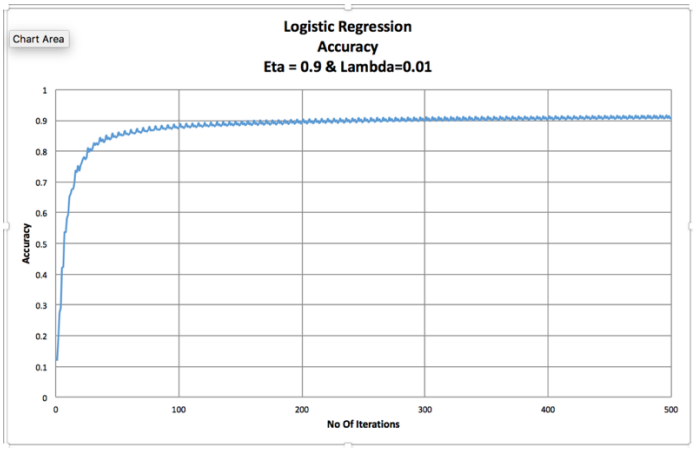

Figure: Testing accuracy with iterations

USPS DATA

The input has 20000 x 784 features which has to be classified into 10 models
Accuracy on training data = 36.58%

Single Layer Neural Network (using back-propagation):

Neural network is an information processing paradigm inspired by the way the human neural system works. A neural network has a large number of interconnected processing elements which processes the information together and learn about the information provided so that it is able to perform functions like recognition or data classification when provided with new data.

We have implemented the feed forward with back propagation neural network. It works as follows:

A single layer neural network has one input layer, one output layer and one hidden layer. There are two sets of weights, one for the weights between input and hidden layer and other for the weights between hidden and output layer. The product of input features and weights of layer 1 acts as input to the hidden layer where an activation function is applied to it. We have used hyperbolic tangent as the activation function and softmax at the output layer.

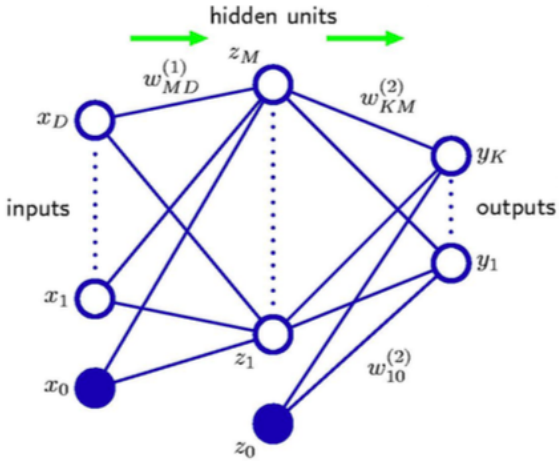

Figure: Neural Network

The gradient of the error function is calculated as follows:

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

where

$$\delta_k = y_k - t_k$$

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

We use stochastic gradient descent algorithm to update the weights

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \nabla_{\mathbf{w}} E(\mathbf{x})$$

Hyper Parameter Tuning (Lambda, Eta, Number of nodes):

For the single layer neural network, we need to tune certain hyper parameters like learning parameter eta, lambda and the number of nodes in the hidden layer. The typical values for eta and lambda lie between 0.01 to 0.1 and the number of nodes in the hidden layer should lie between the number of nodes in input layer and the number of nodes in the output layer.

We run a grid search for eta, lambda and number of nodes, train the neural network with the given parameters and test on the validation set and choose the parameter combination that gives the least classification error.
The values taken for eta for the grid search were: 0.01, 0.05, 0.1, 0.5
The values used for lambda for the grid search were: 0.01, 0.03, 0.1, 0.3
And, the number of nodes for the grid search were: 100, 300, 500, 700

Following are the accuracies on the validation data set for eta = 0.01 and various combinations of lambda and number of nodes

Lambda →

<--No of Nodes	**0.01**	**0.03**	**0.1**	**0.3**
100	80.58	80.05	81.93	80.2
300	83.58	83.78	84.41	84.23
500	84.76	85.18	84.91	85.15
700	85.83	85.23	85.85	86.01

Table: Grid Search

After running the grid search for various values of eta, the optimal values obtained were:
Eta = 0.1, Lambda = 0.01, Number of hidden layer nodes = 300

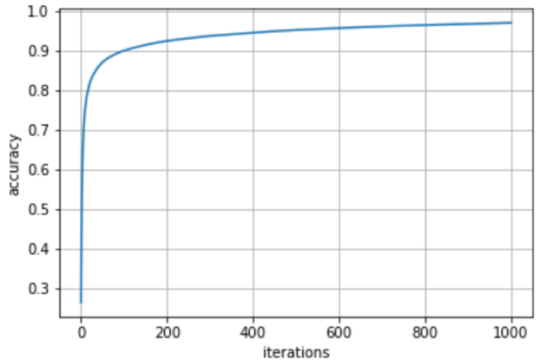

Figure: Testing accuracy with iterations

Training data accuracy: 97.06%
Testing accuracy on MNIST dataset: 95.05%
Testing accuracy on USPS dataset: 39.80%

Results:

```
Accuracy on train data:
0.970666666667
Accuracy on test data:
0.9505
Accuracy on USPS data:
0.39801990099504975
```

Convolutional Neural Network:

Convolutional neural network is a type of feed forward neural network consisting of multiple hidden layers. It is designed to require minimal pre-processing as compared to the normal multilayer neural networks in which pre-processing is required. A CNN typically consists of 3 types of layers defined as follows:

Convolutional layers: These layers apply a given number of filters of given size on the input data and extracts features from the data.

Pooling layers: A convolutional layer is always followed by a pooling layer. Pooling layers take the maximum values from the output convolutional layers and feed it to the next convolutional layer or fully connected layer. Pooling layer typically works to reduce the size of input data for easy processing.

Fully connected layers: The fully connected layers take input from the last pooling layer and acts as a true multilayer perceptron. The final output is obtained from a fully connected layer.

We have used the publically available tensorflow library to implement convolutional neural network. Following are the accuracy of the CNN model trained on the MNIST dataset using tensorflow.

Training data accuracy = 99.89%
Testing accuracy on MNIST dataset: 99%
Testing accuracy on USPS dataset: 58.8%

Bayesian Logistic Regression:

We have implemented the code for Bayesian using description given below.
In Bayesian logistic regression, we start with an initial belief about the distribution that is the posterior, which is our updated belief about the weights given evidence, is proportional to our prior (initial belief) times the likelihood. Adding a prior is equivalent to adding a regularization term equivalent to L2 norm. Because we seek a Gaussian representation for the posterior distribution, it is natural to begin with a Gaussian prior, which we write in the general form

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$$

where m0 and S0 are fixed hyper parameters. The posterior distribution over w is given by

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{w})p(\mathbf{t}|\mathbf{w})$$

where t = (t1 , . . . , tN)T .

The likelihood function is maximized by taking log both sides then the above equation takes form:

$$\begin{aligned}\ln p(\mathbf{w}|\mathbf{t}) &= -\frac{1}{2}(\mathbf{w}-\mathbf{m}_0)^{\mathrm{T}}\mathbf{S}_0^{-1}(\mathbf{w}-\mathbf{m}_0) \\ &\quad + \sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} + \text{const}\end{aligned}$$

where y = σ(wTϕn)

To obtain a Gaussian approximation to the posterior distribution, we first maximize the posterior distribution to give the MAP (maximum posterior) solution wMAP, which defines the mean of the Gaussian. The covariance is then given by the inverse of the matrix of second derivatives of the negative log likelihood, which takes the form

$$\mathbf{S}_N = -\nabla\nabla \ln p(\mathbf{w}|\mathbf{t}) = \mathbf{S}_0^{-1} + \sum_{n=1}^{N} y_n(1-y_n)\boldsymbol{\phi}_n\boldsymbol{\phi}_n^{\mathrm{T}}.$$

The Gaussian approximation to the posterior distribution therefore takes the form

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\mathrm{MAP}}, \mathbf{S}_N).$$

Initially we select S0 as a diagonal matrix of size m x m where, m (784) is the no of features of any given sample N (let 60000). Here we are given data which has to be classified into 10 models. For that first the hyper parameters SN and Wmap are trained on the training data and then the prediction is done by the approximation given below using the updated S_N and Wmap:

Compute $\sigma(\kappa(\sigma_a^2)\mu_a)$

$$\mu_a = \mathrm{E}[a] = \int p(a)\,da = \int q(\boldsymbol{w})\boldsymbol{w}^T\phi\,d\boldsymbol{w} = \boldsymbol{w}_{map}^T\phi$$

$$\begin{aligned}\sigma_a^2 &= \mathrm{var}[a] = \int p(a)\left\{a^2 - \mathrm{E}[a]^2\right\}da \\ &= \int q(\boldsymbol{w})\left\{(\boldsymbol{w}^T\phi)^2 - (m_N^T\phi)^2\right\}d\boldsymbol{w} = \phi^T S_N\phi\end{aligned}$$

$$\kappa(\sigma^2) = (1+\pi\sigma^2/8)^{-1/2}$$

Initially Wmap is set to be a random Gaussian distribution which is then updated at every iteration using the formula

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta\sum_{i=1}^{m}\nabla_{\mathbf{w}}E(\mathbf{z}_i)$$

where gradient is given by taking derivative of equation:

$$\begin{aligned}\ln p(\mathbf{w}|\mathbf{t}) &= -\frac{1}{2}(\mathbf{w}-\mathbf{m}_0)^{\mathrm{T}}\mathbf{S}_0^{-1}(\mathbf{w}-\mathbf{m}_0) \\ &\quad + \sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} + \text{const}\end{aligned}$$

The process is repeated till we are able to classify our data correctly. The accuracy of the prediction is noted and the final weights and variance is used further on testing data.

Conclusion:

The free lunch theorem states that any two optimization algorithms are equivalent when their performance is averaged across all possible problems. In terms of machine learning, a highly optimized model might not work well for any set of new data. For this project assignment we developed models for hand written digit recognition using MNIST dataset as input and techniques like multiclass logistic regression, single layer neural network and convolutional neural networks with training accuracies of 92.15%, 97.06% and 99.89% respectively. However, these models give only 36.58%, 39.80% and 58.8% accuracy on USPS dataset. This proves the free lunch theorem.

References:

- https://en.wikipedia.org/wiki/Convolutional_neural_network
- https://www.tensorflow.org/tutorials/
- https://en.wikipedia.org/wiki/No_free_lunch_theorem
- https://www.youtube.com/watch?v=uSuEgl0n_Ls
- Pattern Recognition and Machine Learning By Christopher M Bishop
- https://www.youtube.com/watch?v=GxlHCGkdAZ0&t=1607s