CSE574: Project 3 - Classification

Nitin Nataraj (50246850) & Arun Krishnamurthy (50247445)

November 21, 2017

1 Logistic Regression

1.1 Performing logistic regression on MNIST dataset

The MNIST dataset was downloaded from `tensorflow.examples.tutorials.mnist` and logistic regression was performed over 1000 epochs. The trained model was then tested on `mnist.validation` and `mnist.test` datasets and the following results were obtained.

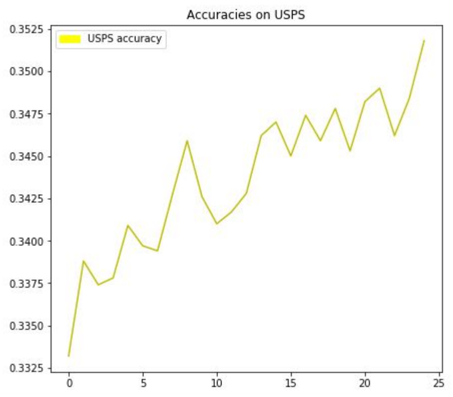


Figure 1: Loss on MNIST

1.2 Testing the model on the USPS dataset

The model trained on MNIST dataset is then trained on the USPS dataset that was provided on `UBLears` portal.

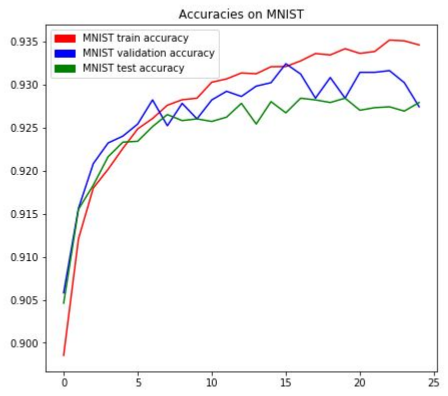


Figure 2: Loss on MNIST

1.3 Tuning Hyperparameters

Different hyperparameter values were tested to see which values were better. As we can see here, of the given iterations, the best accuracy on MNIST test data is 92.79%, when epochs is set to 25, batch size to 100 and learning rate to 1e-4.

	Iteration 1	Iteration 2	Iteration 3
Learning Rate	1e-3	1e-2	1e-4
Batch Size	100	100	100
Epoch	50	50	25
Validation Accuracy (%)	92.88	92.88	92.74
Test Accuracy (%)	92.76	92.76	92.79
USPS Accuracy (%)	34.78	34.78	35.18

Figure 3: Hyperparameter Tuning

The results obtained were:

```
Validation accuracy: 0.9274
Test accuracy: 0.9279
USPS Accuracy: 0.3518
```

The loss was `132.554`

2 Single Layer Neural Network

2.1 Training a single layer neural network on MNIST dataset

The MNIST dataset was downloaded from `tensorflow.examples.tutorials.mnist` and a single layer neural network was trained on this data. The trained model was then tested on `mnist.validation` and `mnist.test` datasets and the following results were obtained.

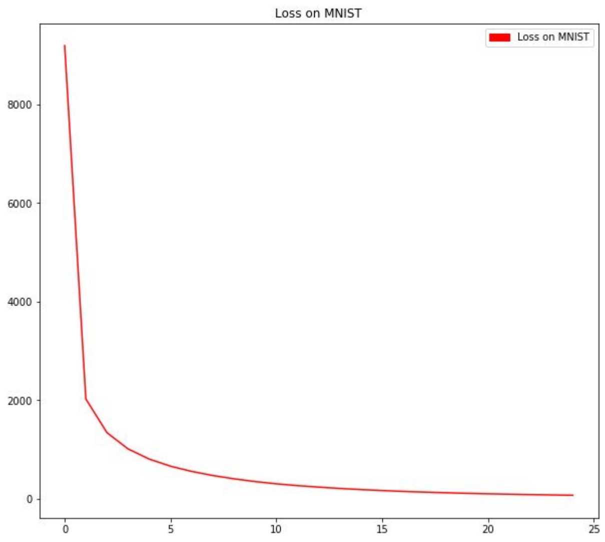


Figure 4: Loss on MNIST

2.2 Testing the neural network on the USPS dataset

The model trained on MNIST dataset is then trained on the USPS dataset that was provided on `UBLears` portal.

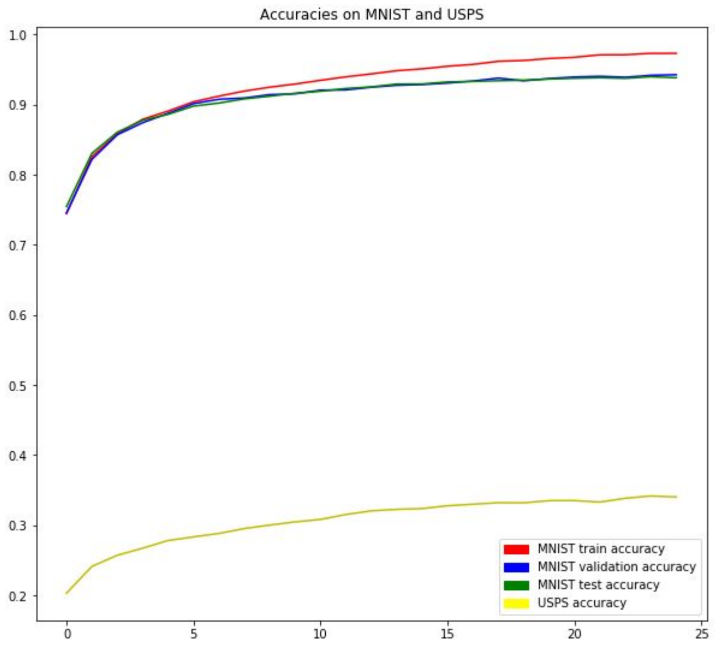
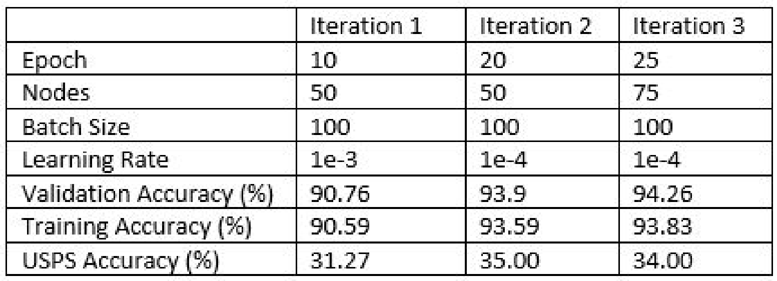


Figure 5: Accuracies on MNIST and USPS

2.3 Tuning Hyperparameters

Different hyperparameter values were tested to see which values were better. As we can see here, of the given iterations, the best accuracy on MNIST test data is 93.83%, when epochs is set to 25, nodes to 75, batch size to 100 and learning rate to 1e-4.



	Iteration 1	Iteration 2	Iteration 3
Epoch	10	20	25
Nodes	50	50	75
Batch Size	100	100	100
Learning Rate	1e-3	1e-4	1e-4
Validation Accuracy (%)	90.76	93.9	94.26
Training Accuracy (%)	90.59	93.59	93.83
USPS Accuracy (%)	31.27	35.00	34.00

Figure 6: Hyperparameter Tuning

The results obtained were:

```
Validation accuracy: 0.9426
Test accuracy: 0.9383
USPS Accuracy: 0.340017
```

The loss was `71.794952728319913`

3 Convolutional Neural Network

3.1 Training a CNN on MNIST dataset

The MNIST dataset was downloaded from `tensorflow.examples.tutorials.mnist` and a convolutional neural network was trained on this data. The trained model was then tested on `mnist.validation` and `mnist.test` datasets and the following results were obtained.

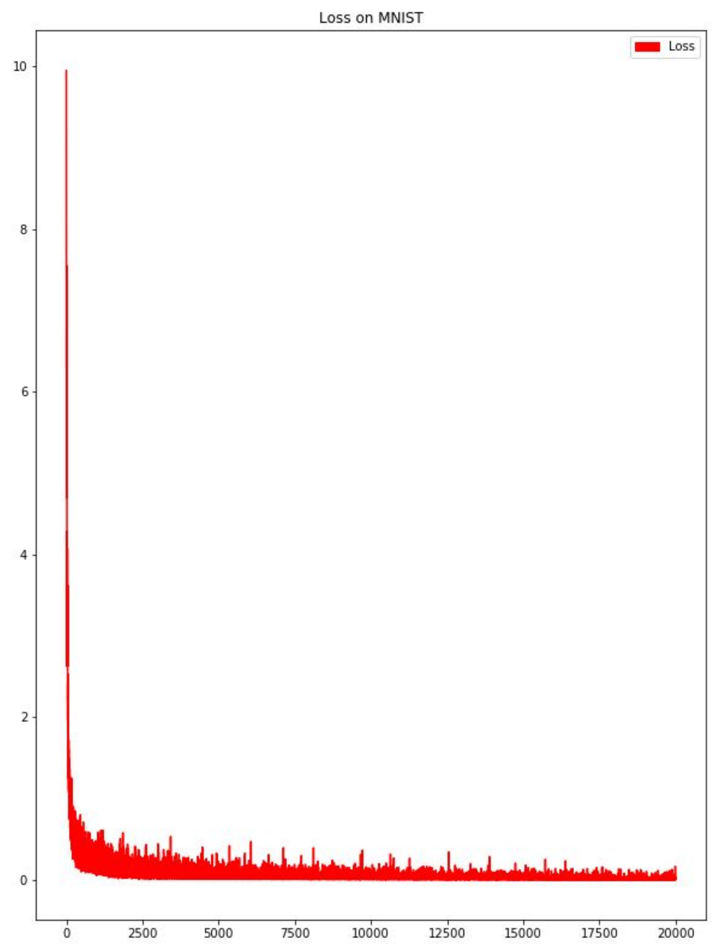


Figure 7: Loss on MNIST

3.2 Testing the CNN on the USPS dataset

The model trained on MNIST dataset is then trained on the USPS dataset that was provided on `UBLearns` portal. A dropout rate was 0.5 was used during training and a dropout rate of 1 was used while testing. A final loss of was obtained on the MNIST training set, and a classification accuracy of 99.11% was obtained on the MNIST training set. The same model when tested on the USPS training set, obtained a prediction accuracy of 67.53%.

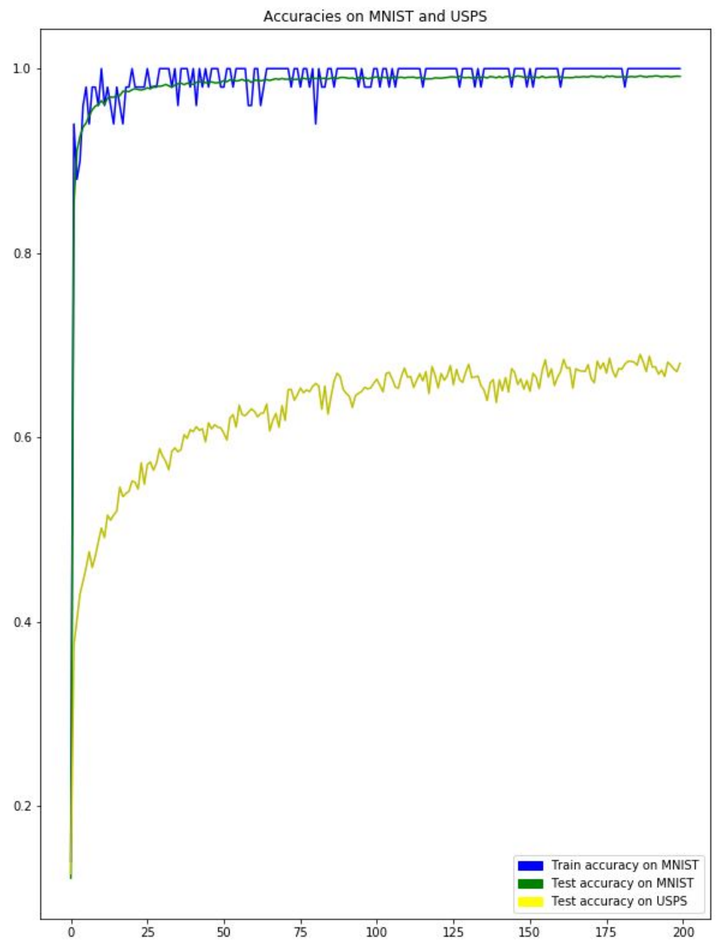


Figure 8: Accuracies on MNIST and USPS

The results obtained were:

```
Test accuracy: 0.9911
USPS Accuracy: 0.6753
```

3.3 No Free Lunch Theorem

The No Free Lunch Theorem states that if a model performs well on a particular dataset, then it pays for that with degraded performance on another dataset. As we can see that in all the above approaches, i.e. Logistic Regression, Single Layer Neural Network, and the Convolutional Neural Network, this theorem holds true, as the classification accuracy on the USPS dataset is far lower than that on the MNIST dataset where it performs very well.

3.4 Backpropogation

For this, first calculate the the error signal of the final layer, by obtaining the gradient of the cost function with respect to the outputs of the network. This expression will depend on the current training sample (input and output), as well as the chosen cost function. Then for backwards propagation, calculate the error signals of the neurons in each layer. Update the weights and biases using a learning rate parameter for scaling. Use regularization to prevent overfitting using the parameter lambda.

3.5 Bayesian Logistic Regression

Bayesian Logistic Regression calls for the Bayesian inference or treatment of regular logistic regression. In this case, we assume a prior distribution over the parameters of logistic regression, i.e. the weights. We obtain several Gaussian distributions over the weights, determined by the mean and variance of the respective distributions. We update the prior by using the likelihood information obtained from the data points we have for evidence. We then obtain the posterior distribution as per the following formula.

The posterior distribution over the weights is given by Bayes' rule:

$$p(\mathbf{w} \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \mathbf{w})\, p(\mathbf{w})}{P(\mathcal{D})} \propto P(\mathcal{D} \mid \mathbf{w})\, p(\mathbf{w}).$$

The normalizing constant is the integral required to make the posterior distribution integrate to one:

$$P(\mathcal{D}) = \int P(\mathcal{D} \mid \mathbf{w})\, p(\mathbf{w})\, \mathrm{d}\mathbf{w}.$$