CSE574: Introduction to ML(Fall'18)

Project 3: Classification on MNIST and USPS Dataset

Vishal Shivaji Gawade
Department of Computer Science
University at Buffalo
Buffalo, NY,14226
vgawade@buffalo.edu

Abstract

The goal of this project is to implement machine learning methods for the task of classification. We will first implement an ensemble of four classifiers for a given task. Then the results of the individual classifiers are combined to make a final decision.

1.Introductiom

The classification task will be to recognizing a 28×28 gray scale handwritten digit image and identify it as a digit among 0, 1, 2, ..., 9. We will train the following four classifiers using MNIST digit images.

1. Multilayer perceptron neural network classifier.
2. Random Forest classifier.
3. SVM classifier.
4. Logistic regression, which we implement using backpropagation.

After our four models are trained on MNIST dataset, our task is to use these models to predict output of MNIST testing set which is of 10000 images and on 20,000 images of USPS dataset. We are going to use confusion matrix for evaluation of our models. And finally, we are going to combine the results of all four model's prediction and apply majority voting using ensemble and predict the final results. Then we will compare results of predicted output and actual target values using confusion matrix and observe how combined model predictions compare to individual model predictions. I am using confusion matrix to calculate the accuracy. Also, we will see does "No Free Lunch" theorem applies to our models and problem in hand.

Out of total 70,000 image samples of MNIST we are going to use 60,000 for training models and 10,000 for testing the model. We are saving the predicted values in csv file so that I can use them later. Let's see all four models one by one.

2. Train using Neural network classifier

For neural network we are going to use two hidden layers with relu activation function and output layer with softmax as an activation function. We are going to use 512 nodes for first hidden layer, 256 for second and as our output is 10 classes,10 nodes for output layer. Also, we are going to use drop out of 0.3 for hidden layers. For optimizing the model, we are going to use Adadelta as an optimizer and categorial cross entropy as a loss function. For epochs we are setting it to 20 and model batch size for training will be set to 512.For training the model we are representing the target value in one hot vector representation because we don't want to take integer values and cause matrix multiplication give more weightage to bigger integers. This is how we are going to setup our neural network. For more details I have mentioned comments in main.ipynb file.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               401920
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
activation_2 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                2570
_________________________________________________________________
activation_3 (Activation)    (None, 10)                0
=================================================================
Total params: 535,818
Trainable params: 535,818
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: Hidden layers and nodes used

2.1. Model Performance

I tried different tune parameters and below table shows their results on MNIST test data and USPS data.

Table 1: Model Settings for NN

Optimizer	Activation Function	No of Epochs	Accuracy on MNIST Test Data	Accuracy on USPS Test Data
Adadelta	Relu	10	96.44	44.92
Adadelta	Relu	20	98.33	52.89
Adagrad	tanh	5	94.09	34.78
Adadelta	Relu	15	97.44	48.76

As we are getting max accuracy at second row. We will stick with that network setting for out neural network.

Model Trained -

Now let's see the accuracy graph and confusion matrix for MNIST test data.

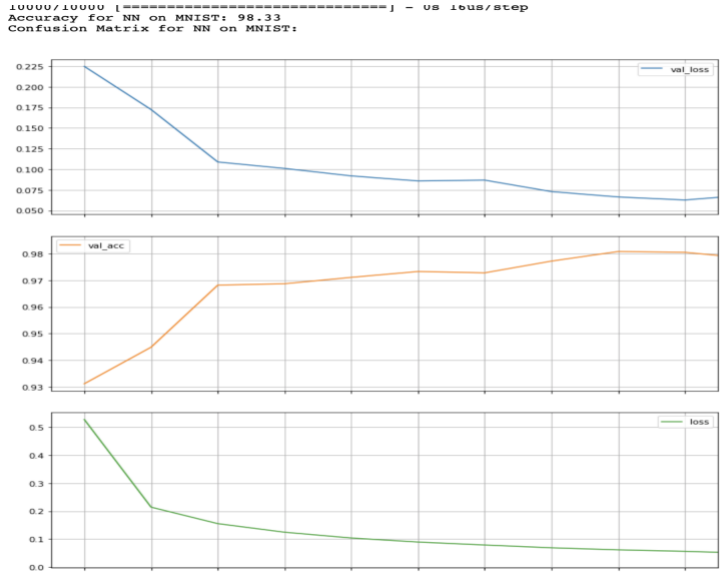

Figure 2: Different graphs for NN

As we can see from graphs value loss is almost less than 0.1 for out neural network model. Also, after certain number of epochs we constantly achieving greater than 98% accuracy.

MNIST result –

Accuracy- 98.33 %

Now let's evaluate confusion matrix for MNIST

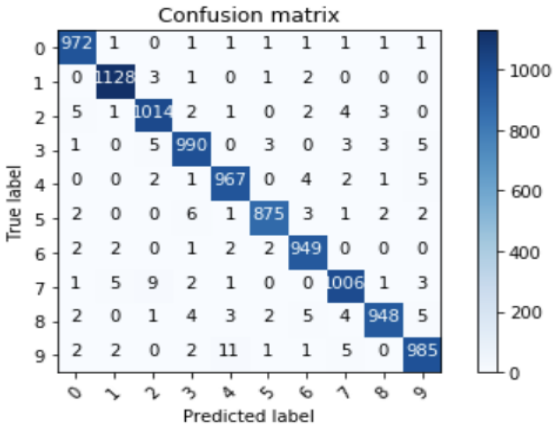

Figure 3: Confusion matrix for MNIST NN

As we can see from confusion matrix we predicted 9 as a 4 most of the times compare to other digits in neural network because of the way people write it. Diagonal elements represent the correctly predicted results. The addition of all diagonal elements is number of correctly predicted samples. All these properties of confusion matrix apply for all models.

USPS result -

```
Accuracy for NN on USPS: 52.89764488224
Confusion Matrix for NN USPS:
```

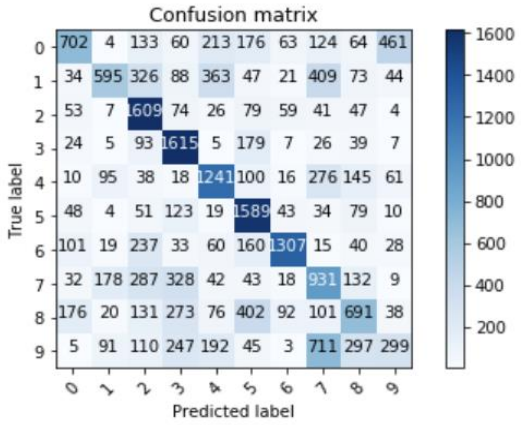

Figure 4: Confusion matrix for USPS NN

As we can see from matrix, we predicted 9 as 7, 711 times on USPS data. After that we predicted 0 as a 9, 461 times. Confusion matrix shows us how many times samples got predicted as another number as well as how many times it got predicted correctly.

3. Train using Random Forest classifier

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

3.1. Model Performance

For random forest we only have one parameter to tune, n_estimators. We will see how model performs when we tweak this parameter.

Table 2: Model Settings for RF

n_estimators	Test Accuracy (MNIST)	Accuracy (USPS)
10	94.85	29.19
30	96.24	36.70
50	96.69	38.59
80	96.85	38.93
100	97	39.63

As we can see, if we keep on increasing number of decision trees in random forest we see increase in accuracy. After certain number we see very little improvement in accuracy. Therefore, lets stick to 100 value of n_estimators.

MNIST result –

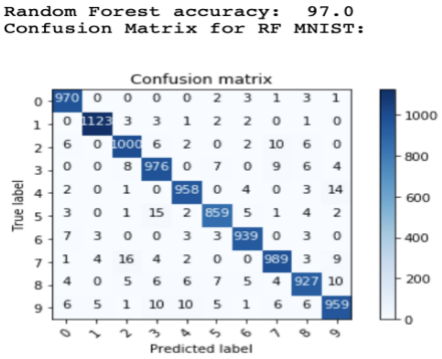

Figure 5: Confusion matrix for MNIST RF

As we can see we predicted almost 97% samples correctly. For wrong predictions, we predicted 7 as 2, 16 times. This is the most wrong predicted value for our random forest model compare to other digits. People sometimes write seven like two, this's maybe reason for it being mostly predicting wrong by our model.

USPS result –

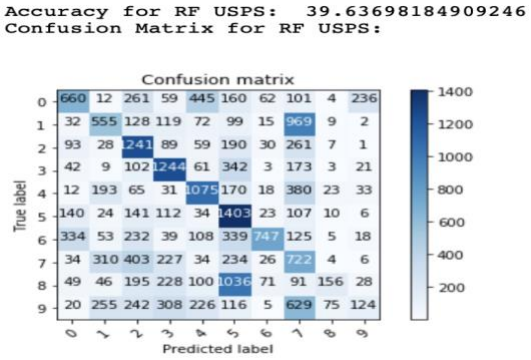

Figure 6: Confusion matrix for USPS RF

For USPS dataset, we predicted 1 as 7 almost 969 times. This is may be because people write one like seven. Also, we predicted 8 as a 5 almost 1036 times, these values are very big and hampered accuracy of our model on USPS data.

4. Train using SVM classifier

Support vector machines are a set of supervised learning methods used for classification, regression and outliers detection.
The advantages of support vector machines are:

1. Effective in high dimensional spaces.
2. Still effective in cases where number of dimensions is greater than the number of samples.
3. Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
4. Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

1. If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
2. SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

The first person who tried to train MNIST dataset tried it on SVM classifier. Now let's see how our model performs.

4.1. Model Performance

Table 3: Model Settings for SVM

Model Setting	MNIST test accuracy	USPS accuracy
Kernel='linear', C=2, gamma=1	93.8	28.42
Kernel='linear' (Keeping other parameters to default)	94.12	29.27
Kernel='poly' (Keeping other parameters to default)	94.33	36.74
Kernel='rbf' (Keeping other parameters to default)	94.45	38.89

As we see, we are getting maximum accuracy for rbf kernel. But as it is computationally taking lot of time. We are going to stick with first row combination for further evaluation.

MNIST result –

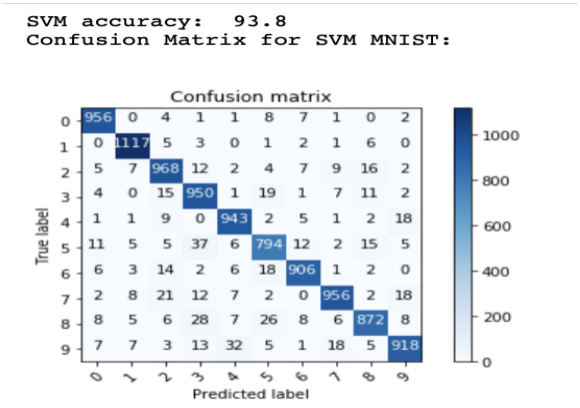

Figure 7: Confusion matrix for MNIST SVM

From confusion matrix of SVM MNIST, we can infer that we predicted 5 as a 3 almost 37 times, most for any digit. We almost predicted all one (1) digit sample correctly for SVM model. But still SVM is predicting more false results compare to neural network.

USPS result –

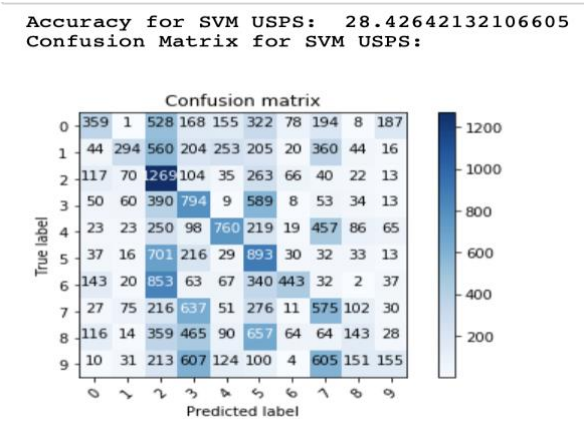

Figure 8: Confusion matrix for USPS SVM

From above confusion matrix, we can infer that there is lot of inconsistency in predicating correct results. For SVM we predicted digit two almost correctly for every two's sample. We cannot form a solid conclusion regarding any other digits from confusion matrix.

5. Train using Logistic regression classifier

For logistic regression implementation we used mini batch stochastic gradient decent. Mini-batch stochastic gradient descent is something between batch gradient descent and stochastic gradient descent. In each iteration of the mini-batch SGD, it samples a small chunk of samples z1, z2, ...,zm from the training data and uses this chunk to update the parameters w:

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

5.1. Model Performance

Table 4: Model Settings with respect to learning rate

Regulariser (λ)	Learning Rate	Batch Size	Epoch	Accuracy Rate on MNIST Test Data	Accuracy Rate on USPS Test Data
0.0001	0.1	2048	5	87.729	33.491
0.0001	0.5	2048	5	90.560	35.096
0.0001	0.9	2048	5	91.070	35.291

The learning rate parameter determines how fast or slow we are moving towards the optimal weights. Through tuning the parameter, we found that although testing accuracy changes very little in a range for 0.5 and 0.9 learning rate, hence 0.5 learning rate is the best fit for our model.

Table 5: Model Settings with respect to regulariser rate

Regulariser (λ)	Learning Rate	Batch Size	Epoch	Accuracy Rate on MNIST Test Data	Accuracy Rate on USPS Test Data
0.1	0.5	2048	5	85.73	33.06
0.01	0.5	2048	5	89.94	35.01
0.001	0.5	2048	5	90.36	35.17

Further tweaking the regulariser on network settings where we received the highest accuracy, did not have favorable impact on accuracy of MNIST test data for values 0.01 and 0.001

Table 6: Model Settings with respect to number of epochs

Regulariser(ƛ)	Learning Rate	Batch Size	Epoch	Accuracy Rate on MNIST Test Data	Accuracy Rate on USPS Test Data
0.0001	0.5	2048	5	90.51	35.02
0.0001	0.5	2048	10	91.21	35.51

Further tweaking the epoch hyper-parameter on network settings where we received the highest accuracy is 10.
I tried different batch sizes also. We got maximum performance for 2048 batch size.
We are going to use epoch=10, batch size=2048, Learning rate=0.5 and ƛ=0.001 for further evaluation.

MNIST result –

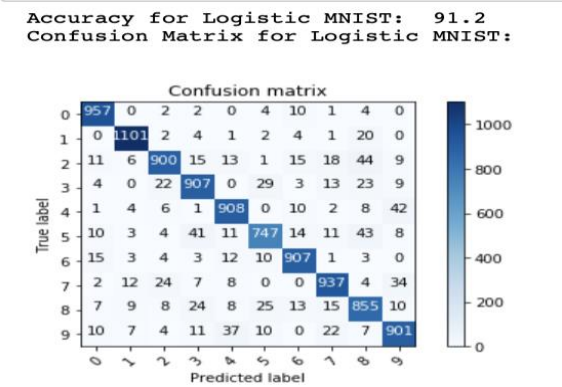

Figure 9: Confusion matrix for MNIST Logistic

From confusion matrix, we can say that we predicted 2 as an 8, almost 44 times. We predicted 2,5 and 9 more incorrectly compare to other digits. The confusion matrix looks cleaner but it is still not better than neural network model.

USPS result –

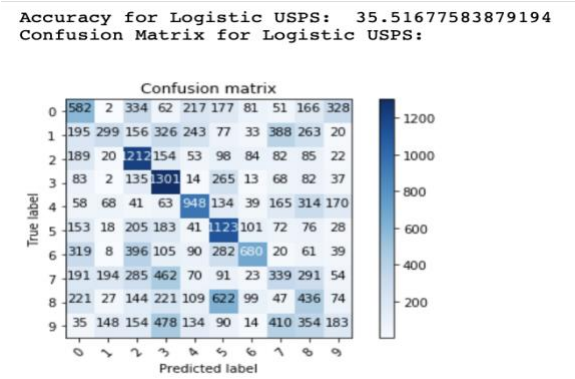

Figure 10: Confusion matrix for USPS Logistic

For USPS confusion matrix we predicted 2,3,4,5 digits more correctly as compare to other digits. Still it is not better than neural network model, which perform way better than logistic model

6. Combining all models (Ensemble Classifier)

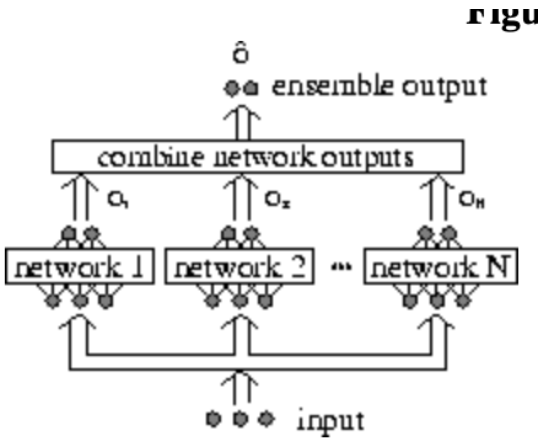

Figure 11: Ensemble classifier

Figure 11 illustrates the basic framework for a classifier ensemble. In this example, neural networks are the basic classification method, though conceptually any classification method (e.g., decision trees) can be substituted in place of the networks. Each network in Figure 11's ensemble (network 1 through network N in this case) is trained using the training instances for that network. Then, for each example, the predicted output of each of these networks (oi in Figure 11) is combined to produce the output of the ensemble (in Figure 11). We used majority voting to predict the output of ensemble. In majority voting we select the value on which major ensembles agree on.
Combining the output of several classifiers is useful only if there is disagreement among them. Also, combining several identical classifiers produces no gain.
We took the predicted output of above each four classifiers and using majority voting concept predicted final output values for ensemble classifier. Then we drew confusion matrix for ensemble classifier and evaluated the accuracy and error.

6.1 MNIST test set results

```
Accuracy of ensembler on MNIST 96.39999999999999
Confusion Matrix for ensembler on MNIST:
```

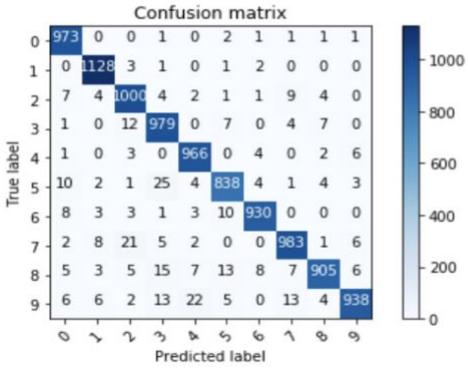

Figure 12: Confusion matrix for ensemble classifier MNIST

Figure 12 shows that we predicted digits 5,7,8,9 more wrong as compare to other digits for ensemble classifier. Also, we can draw a conclusion from confusion matrix and accuracy that ensemble classifier does not always perform better than individual classifiers. For our problem we see that neural network perform better than ensemble classifier. Our other models are dragging down the performance of ensemble classifier which causing ensemble performing worse than neural network model.

6.2 USPS dataset results

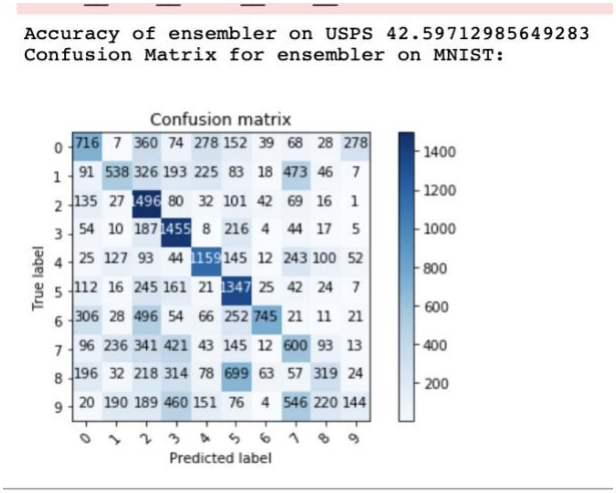

Figure 13: Confusion matrix for ensemble classifier USPS

Figure 13 shows that we predicted 2,3,4,5 digits more correctly compare to other digits for ensemble classifier. Also, we can draw a conclusion from confusion matrix and accuracy that ensemble classifier does not perform well on unknown datasets i.e. USPS. We trained our model on MNIST dataset but we are testing it on USPS dataset. This proves the "No Free Lunch" theorem applies for our given classification task and ensemble classifier. Also, USPS dataset performed better on neural network as compare to ensemble classifier. So, we can say that individual model performed better than ensemble classifier for our classification task and using ensemble classifier for tasks in not always beneficial.

7.Quetions and Answers

Q1. We test the MNIST trained models on two different test sets: the test set from MNIST and a test set from the USPS data set. Do your results support the "No Free Lunch" theorem?
A. The "No Free Lunch" theorem states that no optimization technique (algorithm/heuristic/meta-heuristic) is the best for the generic case and all special cases (specific problems/categories).
Our results do support "No Free Lunch" theorem. If we want to find out the reason, why it does support the "No Free Lunch" theorem look at the results for all four classifiers and ensemble classifier. Also, their results observation shows that indeed our classifiers does support "No Free Lunch" theorem.

Q2. Observe the confusion matrix of each classifier and describe the relative strengths/weaknesses of each classifier. Which classifier has the overall best performance?
A. Overall neural network performed better than other classifiers including ensemble classifier. After completing this classification problem, We should approach a classification problem with simple models first (e.g., logistic regression). In some cases, this may already solve our problem sufficiently well. However, if we are not satisfied with its performance and we have sufficient training data, we try to train a computationally more expensive neural network, which has the advantage to learn more complex, non-linear functions. We will use SVM for high dimensional spaces because it takes lot of computation time and cost for problems like classifying MNIST dataset compare to other models. Random forest classifier will handle the missing values and maintain the accuracy of a large proportion of data. Also, it has the power to handle a large data set with higher dimensionality and it handles overfitting issue quite well. Therefore, for high dimensionality we will use SVM or Random forest. For complex classification task we will use neural network. If our task is not complex enough and have limited training data, logistic regression can give us good results too.
For more explanation on each models' strengths and weakness, go through the topic of respective model in report.

Q3. Combine the results of the individual classifiers using a classifier combination method such as majority voting. Is the overall combined performance better than that of any individual classifier?
A. No, overall combined performance is not better than all individual classifier. But it is better than logistic regression and SVM classifier. We explained in details how ensemble classifier compares to other models. See topic 6 of report for more description.

8.Conclusion

1. We tested the MNIST trained models on two different test sets: the test set from MNIST and a test set from the USPS data set. Our results do support the "No Free Lunch" theorem.
2. After observing confusion matrix of each classifier and comparing accuracy we can say that neural network performed better for classifying MNIST as well as USPS data compare to other classifiers including ensemble classifier.
3. After combining the results of the individual classifiers using a classifier combination method majority voting, the overall combined performance is not better than that of any individual classifier.
4. We always don't get better performance if we use classifier combination method.
5. Combining the output of several classifiers is useful only if there is disagreement among them. Also, combining several identical classifiers produces no gain.

References

[1] https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume11/opitz99a-html/node2.html
[2] https://scikit-learn.org/stable/modules/svm.html
[3 https://www.toptal.com/machine-learning/ensemble-methods-machine-learning
[4] https://www.newgenapps.com/blog/random-forest-analysis-in-ml-and-when-to-use-it
[5] https://gogul09.github.io/software/first-neural-network-keras