Ensemble Classification

Pratik Kubal
University At Buffalo
pkubal@buffalo.edu

Abstract

We compare the performance of different classification methods of Logistic Regression, Random Forests, Neural Networks and SVM and then create an ensemble based on training of MNIST dataset. We also evaluate the performance of each of the methods and ensemble over a previously unseen dataset of US Postal Service(USPS) and draw inferences about ensembles and 'No Free Lunch' Theory.

1 Introduction

We first create four classifers based on validation set of MNIST and tune the hyperparameters. Optimal Hyperparameters are compared with the testing set of MNIST and the USPS. We then implement ensemble of these classfiers using different techniques discussed and compare its Testing Accuracy. We also find that ensemble outperforms all the individual classifers.

2 Methods

2.1 Logistic Regression

Logistic Regression can be done using different functions. Sigmoid functions work for a Binary classification, however, when we have a multiclass classification then the Sigmoid functions poses significant challenges such as ambiguous regions[1] when used with different number of discriminant functions.
A Softmax Function takes an exponent of the activation and normalizes it by taking the percentage of each category. This step could very well be considered as a Probablity of a prediction being in a particular class. Basically, a Softmax Functions is similar to Sigmoid function(Gives probabalities of being in a class) but there are few differences, one being the use of exponent function.
The problems in using exponent function is that high values are reached easily and soon after few iterations we will face problems. We have solved it using a constant *log C* used by Xianxu Hou[2], such as if $a_k = {w_k}^t \Phi + b_k$

Then, $logC = -max(a_k)^{(i)}$

The Softmax function after adding constant is:

$$\frac{e^{a_k + logC}}{\sum_{j=1}^{10} e^{a_k + logC}}$$

While working with MNIST dataset we have preprocessed the data by normalizing it according to the Mean and Standard deviation so as to reach the minima quicker. We experiment with various mini batch sizes for the SGD in our expirementation phase.

Preprint. Work in progress.

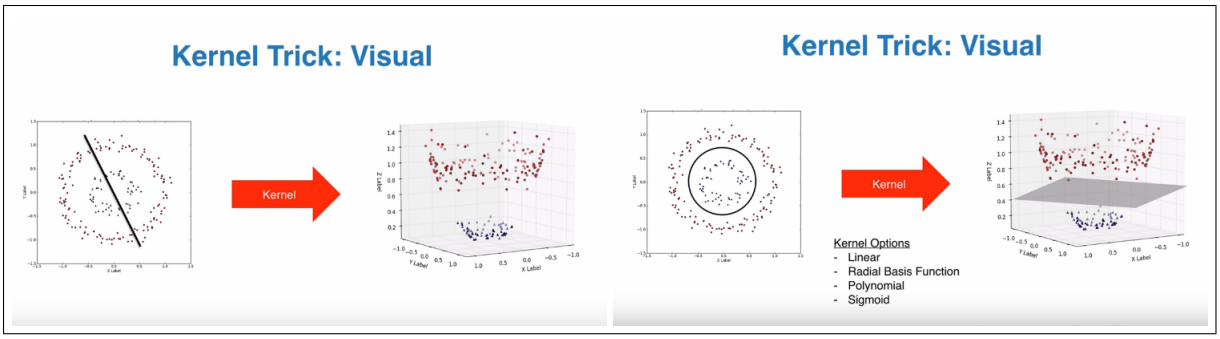

Figure 1: Seperating Support vectors in multi-dimensional space

2.2 Support Vector Machine

Low values of gamma indicate that the far away support vectors or points are taken into consideration. While High values of gamma indicate that the support vectors closer to the hyperplane can pull the decision boundary or hyperplane towards them. Thereby, if a low value of gamma was taken, these points would be misclassified.
The C parameter in general says where should we draw the hyperplane, there are two options. If we put an high value of C then the model will try to get all support vectors classified correctly, while if we took a low C it will misclassify some of the support vectors. It goes without saying that, keeping high values of C might affect the generalization capabality of the model. C is a hyperparameter for SVM which seperates the support vectors in an n dimensional space by creating a hyperplane. Values of C has direct corelation with the dimensions created to seperate the support vectors, overfitting has a typical trait of all support vectors being in their own dimensions. While having bad values of gamma such has having very high value of gamma will make the hyperplane seperating each point very closely, while very low values of gamma will make the hyperplane linear after projecting it to a lower dimensional space. Consider Fig(1)[1]

2.3 Random Forest

Based on sampling different subsets(cross validation) over the dataset, Random Forest constructs n different decision trees. Here the number of decision trees or 'n' is a parameter here. In the expiremention phase we use 'gini impurity' to measure the quality of a spilt. Gini impurity is given by: $1 - \sum_{i=1}^{j} p^2{}_i$
Since Entropy calculates the log it might be computationally slower than gini impurity.
The final prediction is based on ensemble of decision trees, output from each decision tree constructed is taken into account and the weightages of these trees are decided by the cross validation method.

2.4 Neural Networks

We are using two types of Neural networks, Convolutional Neural Networks(CNN) and Deep Neural networks(DNN). Both are similar but according to Deep Learning Book by Ian Goodfellow et. al define convolutional networks as the networks which use convolution instead of a matrix multiplication in at least one of their layers. In addition to this CNN might also involve operation of Pooling.
CNN can be thought as a complex layer involving convolution of the Image with a kernel. Strides are the shifts by which the Kernel moves. The Pooling layer or downsampling layer samples the area around the pool size given and here stride works same as during convolution.
Another key difference between Deep Neural Networks and Neural networks is Sparse Connectivity[3], for CNN the ouputs affected by unit x is the kernel size specified but for DNN all the outputs are affected by unit x. Example, If kernel size is (5,5) then 5 outputs(inputs to next layer) are affected by x, in DNN every hidden layer will be connected to every other hidden layer. For CNN even if direct connections are sparse due to the former difference, the connection reach or the receptive field of

[1] Alice Zhao (https://www.youtube.com/watch?v=N1vOgolbjSc)

Table 1: HyperParameters for Logistic Regression

Alpha(LearningRate)	Batches Size	Validation Accuracy
0.00003	400	78.72
0.00003	100	79.02
0.00003	1	92.56
0.0003	1	92.78

units in a deep layer network is significant. Strides would also increase the receptive field of CNN, and make as if units in deepest layer are connected to the input image or the input layer.

2.5 Ensemble

We use different techniques for implmenting Ensemble over the above classfication methods, first we tried implmenting a basic Naive Soft Voting based Ensemble. It takes the weighted average of the probablities of each candidate model. This weight is determined by brute forcing for accuracy over a test set - Which is computationally difficult because SVM takes long to train and predict. We were able to brute force with three models and then brute force with remaining permutations for SVM. However, there is a flaw in this method, the test set isn't representative of the problem.
Therefore, we need to train another model for predicting the accurate weights for the model for which we will use Random Forests classifier. There are two ways to implement this, we can use the predicted classes or the probabalities. We call the former model as Hard Random Forest Ensemble and the latter as Soft Forest Ensemble(Based on Soft and Hard Voting).
In Hard Random Forest Ensemble we use the predicted classes as input to Random Forest in training and testing. It involves four feature inputs from the four candidate model.
In Soft Random Forest Ensemble we use the predicted probabalities as input to Random Forest in training and testing. It now has 40 feature inputs for 10 categories for each candidate model.

3 Experimentation

3.1 Logistic Regression

We first normalize the data by preprocessing it by replacing each instance by substracting it from its mean and dividing it by Standard Deviation, we will use this techniques at all places so as to decrease the training time required.
For Logistic regression we use try SGD with different batch sizes and alpha. Between each epoch we shuffle the data so as to remove any cycles if present in the data.
The Validation accuracies for different hyperparameters can be found in Table1. While the Validation and Training accuracy graph for optimal hyperparameter can be found in Figure2. The confusion matrix for MNIST and USPS for testing data with Recall and Precision can be found in Table2 and Table3

3.2 Neural Networks

We experiment with Convolutional Neural Networks and Deep Neural networks with different model types(Given in 4 , 5 and 6) and hyperparamters and measure the Testing Accuracy over it. These results can be seen in Table.7
Moreover the confusion matrix for MNIST is given in 8 while for USPS is given in 9. We are using the DNN1 as our candidate model to the ensemble.

3.3 Random Forests

We start with default value of number of tree which is 10. We see that as we increase the number of trees the validation accuracy increases till we reach 3000 after which the accuracy decreases,can be seen in Table 10.

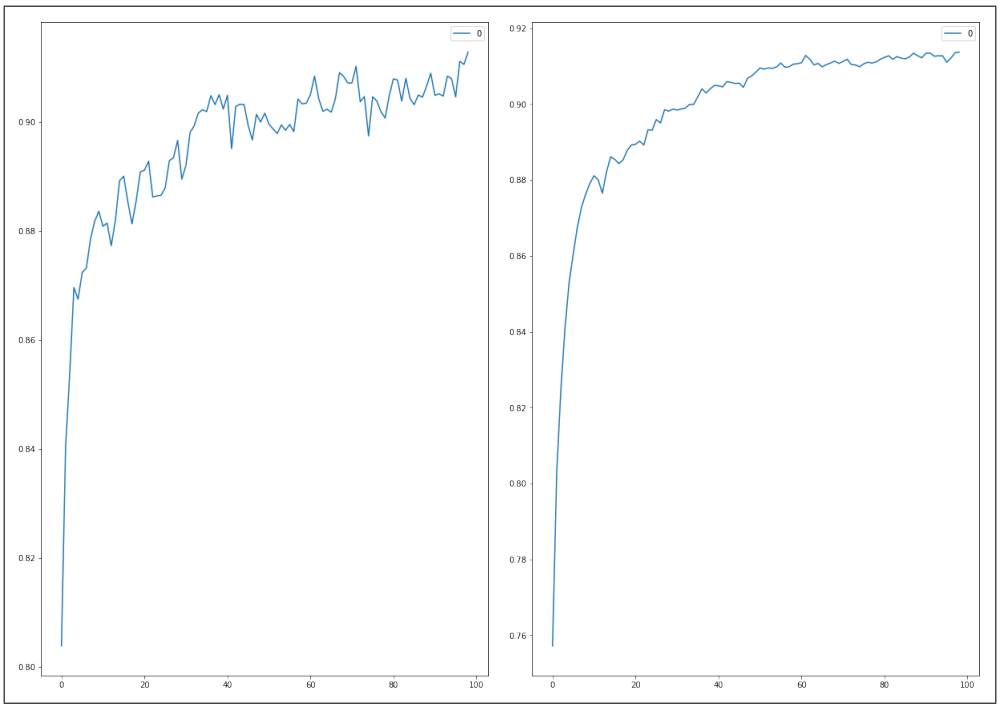

Figure 2: Logistic Regression Training and Validation accuracy for alpha 0.0003

Table 2: Confusion Matrix, Recall and Precision for MNIST Logistic Regression

	Actual Class											
Predicted Class	0	1	2	3	4	5	6	7	8	9	Recall	Precision
0	957	0	1	1	1	6	9	2	3	0	97.65	95.03
1	0	1107	2	2	1	2	4	2	15	0	97.53	95.18
2	7	10	927	18	8	4	12	9	34	3	89.83	93.35
3	3	2	17	917	1	30	3	12	18	7	90.79	91.06
4	2	4	6	1	916	0	9	4	7	33	93.28	92.62
5	7	4	2	32	9	786	17	7	24	4	88.12	89.22
6	11	4	6	1	7	17	908	2	2	0	94.78	93.32
7	2	10	23	6	6	0	0	948	1	32	92.22	92.40
8	6	14	8	16	10	31	11	13	856	9	87.88	88.80
9	12	8	1	13	30	5	0	27	4	909	90.09	91.17

Table 3: Confusion Matrix, Recall and Precision for USPS Logistic Regression

	Actual Class											
Predicted Class	0	1	2	3	4	5	6	7	8	9	Recall	Precision
0	526	88	104	91	28	143	129	144	222	29	26.30	34.97
1	19	353	84	69	103	63	27	266	83	159	17.65	28.79
2	167	210	1126	139	28	198	368	45	57	76	56.33	46.64
3	65	217	103	932	19	134	86	373	259	388	46.60	36.18
4	147	364	34	16	999	34	67	68	133	123	49.95	50.33
5	123	101	125	394	65	900	294	111	350	35	45.00	36.03
6	243	39	176	41	69	222	703	32	199	19	35.15	40.33
7	205	187	37	48	156	71	113	317	81	402	15.85	19.60
8	127	307	142	200	310	188	62	426	441	359	22.05	17.21
9	378	134	68	70	223	47	151	218	175	410	20.50	21.88

.

Table 4: Convolutional Neural Network

Layer (type)	Output Shape	Param	Other HyperParam
conv2d (Conv2D)	(None, 9, 9, 64)	25664	kernel_size=(20, 20), strides=(1, 1),activation='tanh'
max_pooling2d (MaxPooling1)	(None, 4, 4, 64)	0	pool_size=(2, 2), strides=(2, 2)
flatten_1 (Flatten)	(None, 1024)	0	
dense_1 (Dense)	(None, 1024)	1049600	activation='relu'
dense_2 (Dense)	(None, 10)	10250	activation='softmax'

Table 5: Deep Neural Network with Low Hidden Layers(DNN1)

Layer (type)	Output Shape	Param	Other HyperParam
dense_1 (Dense)	(None, 1024)	803840	activation='tanh'
dense_2 (Dense)	(None, 2048)	2099200	activation='relu'
dense_3 (Dense)	(None, 10)	20490	activation='softmax'

Table 6: Deep Neural Network with High Hidden Layers(DNN2)

Layer (type)	Output Shape	Param	Other HyperParam
dense_1 (Dense)	(None, 32)	25120	activation='sigmoid'
dense_2 (Dense)	(None, 128)	4224	activation='tanh'
activation (Activation)	(None, 128)	0	activation='relu'
dropout_1 (Dropout)	(None, 128)	0	
dense_3 (Dense)	(None, 128)	16512	
dense_4 (Dense)	(None, 10)	1290	activation='softmax'

Table 7: Testing Accuracy for Different Neural Network Models

	Testing Accuracy		
Dataset	CNN	DNN1	DNN2
MNIST	98.41	98.23	97.14
USPS	4.575	50.61	37.83

Table 8: Confusion Matrix, Recall and Precision for MNIST Neural Networks

	Actual Class											
Predicted Class	0	1	2	3	4	5	6	7	8	9	Recall	Precision
0	972	1	0	1	0	0	2	1	3	0	99.18	98.98
1	0	1129	1	2	0	1	0	2	0	0	99.47	99.56
2	2	1	1009	6	1	1	1	6	4	1	97.77	98.82
3	1	0	1	977	0	16	0	7	3	5	96.73	97.70
4	0	0	3	1	960	0	3	3	1	11	97.76	98.66
5	2	0	0	4	0	879	2	1	1	3	98.54	96.91
6	4	2	0	1	2	2	942	1	4	0	98.33	98.95
7	1	1	5	2	1	0	0	1012	4	2	98.44	97.40
8	0	0	2	5	2	3	1	5	952	4	97.74	97.64
9	0	0	0	1	7	5	1	1	3	991	98.22	97.44

Table 9: Confusion Matrix, Recall and Precision for USPS Neural Networks

	Actual Class											
Predicted Class	0	1	2	3	4	5	6	7	8	9	Recall	Precision
0	836	4	66	93	186	154	111	150	91	309	41.80	59.29
1	71	575	144	144	321	131	91	349	92	82	28.75	56.10
2	54	4	1548	86	13	86	79	57	66	6	77.44	62.47
3	13	4	71	1465	3	347	6	21	61	9	73.25	47.61
4	32	71	30	23	1002	129	34	320	301	58	50.10	55.45
5	54	3	58	68	5	1647	15	55	82	13	82.35	51.10
6	116	15	201	44	38	224	1203	40	69	50	60.15	75.14
7	47	233	216	566	31	50	6	690	149	12	34.50	28.42
8	150	12	90	328	69	362	50	87	825	27	41.25	40.16
9	37	104	54	260	139	93	6	659	318	330	16.50	36.83

Table 10: HyperParameters for Random Forest

Number of Trees in Forest	Validation Accuracy
10	86.17
100	94.54
1500	95.21
2500	98.92
3000	95.08

The optimal hyperparameters for Random Forests can be seen in table 11 and 12.

3.4 SVM

To counter a big dataset such as MNIST we first performed PCA dimensionality reduction. The optimal parameters were found using plotting the Variance vs Number of Features in the dataset. This figure can be seen in 3. However, after reducting the dimensions to account for 99% of the variance, we couldn't get good results. This is because principal components do not have any correlation to classification accuracy. It is an unsupervised algorithm, therefore we need a supervised algorithm of reducing the dimensions based on the classification labels of training data.
Therefore, Linear Discriminant might be a better choice, We reduce feature based on the variance which can be seen in Fig.4 and reduce the features to 8.
This reduction in size made gridSearch possible. GridSearch iterates through the parameter space while maximizing for a metric(Recall and Precision). GridSearch was run with parameters of kernels of 'linear,poly,sigmoid and rbf', C parameters of '1,2,3,4,5' and gamma parameters of

Table 11: Confusion Matrix, Recall and Precision for MNIST Random Forests

	Actual Class											
Predicted Class	0	1	2	3	4	5	6	7	8	9	Recall	Precision
0	965	1	1	0	0	4	8	1	0	0	98.47	96.79
1	0	1127	2	1	0	2	2	1	0	0	99.29	98.00
2	8	1	997	5	2	2	2	13	2	0	96.61	96.52
3	0	0	6	985	0	4	0	12	2	1	97.52	95.72
4	1	0	2	0	960	1	6	4	1	7	97.76	95.52
5	2	1	0	9	0	873	3	2	2	0	97.87	94.99
6	9	4	1	0	4	3	937	0	0	0	97.81	97.40
7	0	8	16	0	0	0	0	1001	0	3	97.37	95.15
8	4	1	7	17	9	23	3	8	893	9	91.68	98.89
9	8	7	1	12	30	7	1	10	3	930	92.17	97.89

Table 12: Confusion Matrix, Recall and Precision for USPS SVM

	Actual Class											
Predicted Class	0	1	2	3	4	5	6	7	8	9	Recall	Precision
0	573	2	428	19	285	248	73	44	6	322	28.65	42.16
1	110	429	285	137	273	180	46	501	22	17	21.45	44.73
2	128	18	1402	59	39	198	61	57	23	14	70.13	34.77
3	76	3	186	1123	11	483	5	70	27	16	56.15	50.72
4	18	67	91	14	1167	267	22	194	69	91	58.35	51.75
5	108	17	257	102	25	1367	60	43	15	6	68.35	28.94
6	197	7	489	24	98	394	748	13	7	23	37.40	66.02
7	50	225	457	265	57	416	15	452	41	22	22.60	23.61
8	73	25	209	193	87	1006	95	41	244	27	12.20	36.53
9	26	166	228	278	213	165	8	499	214	203	10.15	27.39

Table 13: HyperParameters for SVM

Kernel	C	Gamma	Testing Accuracy
rbf	1	1	17.59
rbf	2	1	99.25
rbf	2	0.5	98.27
rbf	3	0.1	95.42
linear	1.0	auto	94.03
rbf	1	auto	94.35

‘1,0.1,0.01,0.001’.
Some of the output from gridSearch for maximizing the Precision can be seen below:

```
Grid scores on development set:

0.916 (+/-0.004) for {‘C’: 1, ‘gamma’: 0.1, ‘kernel’: ‘rbf’}
...
0.917 (+/-0.003) for {‘C’: 2, ‘gamma’: 0.1, ‘kernel’: ‘rbf’}
...
0.917 (+/-0.002) for {‘C’: 4, ‘gamma’: 0.1, ‘kernel’: ‘rbf’}
...
0.917 (+/-0.003) for {‘C’: 5, ‘gamma’: 0.1, ‘kernel’: ‘rbf’}
...
0.892 (+/-0.004) for {‘C’: 1, ‘gamma’: 0.1, ‘kernel’: ‘linear’}
...
0.892 (+/-0.004) for {‘C’: 5, ‘gamma’: 0.0001, ‘kernel’: ‘linear’}
0.909 (+/-0.004) for {‘C’: 1, ‘gamma’: 0.1, ‘kernel’: ‘poly’}
...
0.011 (+/-0.000) for {‘C’: 5, ‘gamma’: 0.0001, ‘kernel’: ‘poly’}
0.641 (+/-0.010) for {‘C’: 1, ‘gamma’: 0.1, ‘kernel’: ‘sigmoid’}
0.890 (+/-0.004) for {‘C’: 1, ‘gamma’: 0.01, ‘kernel’: ‘sigmoid’}
...
0.886 (+/-0.004) for {‘C’: 5, ‘gamma’: 0.0001, ‘kernel’: ‘sigmoid’}
```

The LDA for MNIST isn’t the same for USPS, after normalizing the USPS dataset we don’t get good results. Even after fitting a non preprocessed dataset with the optimal parameters found by LDA or preprocessed fitting while gridSearch we don’t get good results in USPS(The Testing Accuracy is less than 15%).
We use rbf with C = 1 and gamma as auto because it gives best performance based on USPS and MNIST.

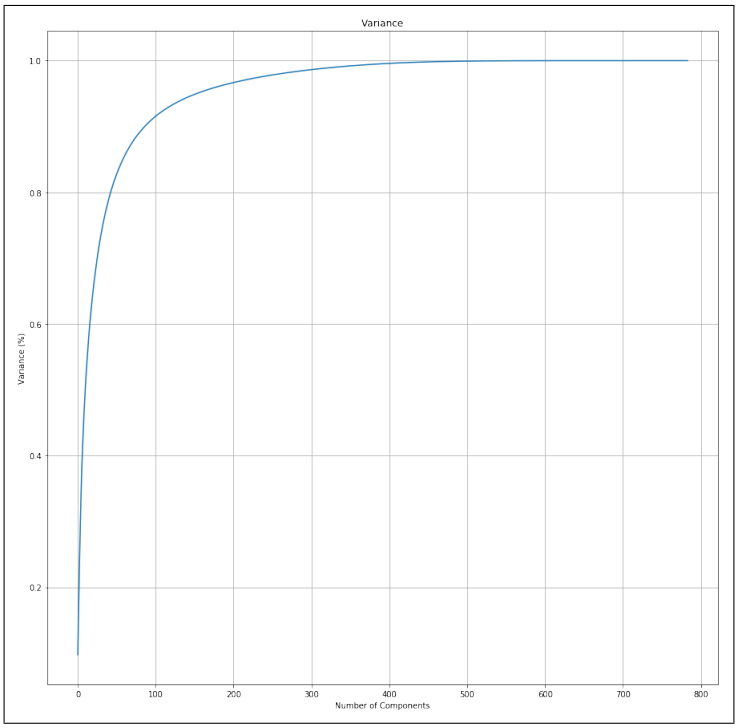

Figure 3: Principal Component Analysis Dimensionality Reduction

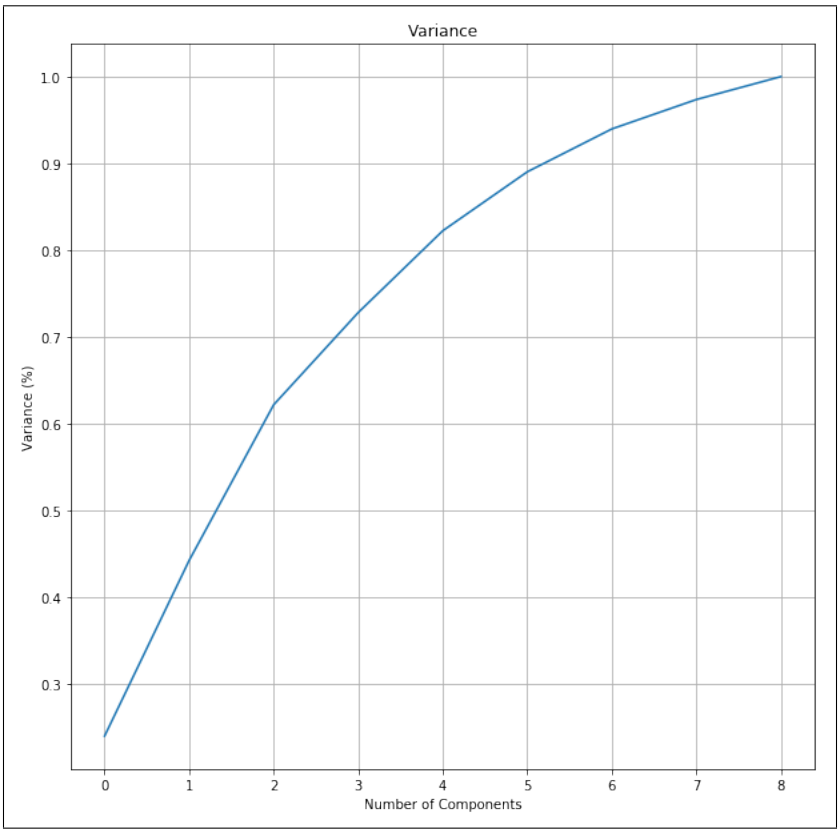

Figure 4: Linear Discriminant Analysis Dimensionality Reduction

Table 14: Confusion Matrix, Recall and Precision for MNIST SVM

	Actual Class											
Predicted Class	0	1	2	3	4	5	6	7	8	9	Recall	Precision
0	967	0	1	0	0	5	4	1	2	0	98.67	95.65
1	0	1120	2	3	0	1	3	1	5	0	98.68	96.97
2	9	1	962	7	10	1	13	11	16	2	93.22	94.04
3	1	1	14	950	1	17	1	10	11	4	94.06	92.68
4	1	1	7	0	937	0	7	2	2	25	95.42	92.96
5	7	4	5	33	7	808	11	2	10	5	90.58	92.66
6	10	3	4	1	5	10	924	0	1	0	90.58	92.66
7	2	13	22	5	7	1	0	954	4	20	92.80	95.12
8	4	6	6	14	8	24	10	8	891	3	91.48	93.99
9	10	6	0	12	33	5	1	14	6	922	91.38	93.98

Table 15: Confusion Matrix, Recall and Precision for USPS SVM

	Actual Class											
Predicted Class	0	1	2	3	4	5	6	7	8	9	Recall	Precision
0	650	11	267	52	450	169	62	85	2	252	32.50	47.86
1	44	552	117	99	49	106	28	990	14	1	27.60	37.73
2	97	24	1271	66	49	209	17	259	5	2	63.58	44.49
3	37	5	88	1303	53	314	3	177	4	16	65.15	55.66
4	12	192	53	22	1095	185	15	385	23	18	54.75	50.20
5	140	23	132	54	19	1501	22	101	5	3	75.05	34.00
6	292	42	206	18	80	364	856	130	2	10	42.80	78.10
7	36	320	364	230	39	270	28	698	5	10	34.90	19.68
8	35	40	142	193	101	1163	57	93	166	10	8.30	52.37
9	15	254	217	304	246	133	8	629	91	103	5.15	24.23

4 Observation

4.1 No Free Lunch Theorem

The 'No Free Lunch Theorem' is *almost* evident in this problem. Almost because given that we do not know of prior information such as absence of USPS dataset or having an unknown dataset the 'No free Lunch theorem' would be applicable. Testing on USPS dataset is also a kind of prior which is influencing the hyperparameters in our model in selection of the perfect algorithm for both MNIST and USPS dataset.

Consider, SVM hyperparameters given in 13 we have found out that the choice of Hyper parameters of C = 2 and Gamma = 1 which maximizes the Accuracy for MNIST actually gives very low accuracy for USPS dataset. By considering this low accuracy as a prior we choose an algorithm which gave an a little low accuracy on MNIST but a little higher accuracy for USPS dataset. This is an effect of 'No Free Lunch Theorem', where the average performance of both is almost same.

This practically doesn't hold or we cannot prove due to multiple algorithms being used and different parameters being involved. It is also possible to get a different accuracy on USPS dataset by processing it differently. Since the effect of No Free Lunch Theorem is evident in a candidate classifier we say that 'No Free Lunch Theorem' is *almost* evident.

4.2 Performance of Different Classifiers

We see that the Neural Networks gives best performance when considering only classifiers. Random Forest and SVM gives similar accuracies however, SVM is computationally expensive to train and predict while Random Forest has high space complexity, this is because the number of trees used were 2000. Logistic Regression performs poorly and had to early stop because the SGD algorithm was moving very slowly and the change in the accuracy wasn't significant. Ref table 16

A key detail in best two best performing Classifers of Random Forest and Neural networks is the

Table 16: Testing Accuracies for MNIST and USPS

	Testing Accuracy						
Dataset	LR	NN	RF	SVM	Naive Soft Voting	Hard RF Ensemble	Soft RF Ensemble
MNIST	92.31	98.23	96.68	94.35	97.77	98.12	98.43
USPS	33.79	50.60	40.98	38.54	47.95	49.54	50.90

Table 17: Confusion Matrix, Recall and Precision for MNIST Ensemble Soft RF

	Actual Class											
Predicted Class	0	1	2	3	4	5	6	7	8	9	Recall	Precision
0	972	1	0	1	0	0	2	1	3	0	99.18	98.88
1	0	1129	1	2	0	1	0	2	0	0	99.47	99.65
2	3	0	1018	0	1	0	1	4	4	1	98.64	98.74
3	1	0	1	986	0	8	0	7	3	4	97.62	98.40
4	0	0	3	1	963	0	3	0	1	11	98.06	98.77
5	2	0	0	4	0	880	1	1	1	3	98.65	97.99
6	4	2	0	1	2	2	942	1	4	0	98.33	99.05
7	1	1	6	2	0	0	0	1011	4	3	98.35	97.96
8	0	0	2	4	2	2	1	4	956	3	98.15	97.55
9	0	0	0	1	7	5	1	1	4	990	98.12	97.54

class *'3'*. The recall for it while using random forest is 97.52 while for Neural Networks is 96.73. This shows that even if the accuracy of Neural networks is greater than that of Random Forest, it is unable to predict the category. Based on Ensemble method we can see that the Recall for the category while using Soft Random Forest Ensemble went up to 97.62. Greater than that of Random Forest or Neural networks alone.

4.3 Performance of Ensemble

The Naive Soft Voting ensemble gives a performance averaged over all other candidate classifiers. Brute Forcing is computationally expensive and we need to find high variance of weights to compensate for low accuracy classfiers like Logistic Regression. Even if we do compensate as we talked about in 2.5 it is flawed. Therefore a Supervised approach based on unseen data seems like a good choice.
The Hard Random Forest Ensemble performs better than the previous ensemble but the Soft Random Forest Ensemble performs better. Due to probabalities being involved the Soft RF ensemble is able to perform better than all of the individual classifiers. It perfectly identifies the correct instances identified from each of the candidate models and uses them for prediction. See example in section 4.2.

5 Conclusion

We see that Ensemble method gives better results than all of the candidate classifiers combined. Using USPS dataset it is also seen that it is better able to generalize to new datasets. We also see that for ensemble methods to work they should interpret or use the outputs from candidate models perfectly, such as Naive soft voting wasn't able to outperform the candidate classifiers but Soft Random Forest able to outperform others.

Table 18: Confusion Matrix, Recall and Precision for USPS Ensemble Soft RF

	Actual Class											
Predicted Class	0	1	2	3	4	5	6	7	8	9	Recall	Precision
0	841	4	79	91	188	151	108	135	88	315	42.05	58.24
1	72	590	153	147	314	121	86	345	95	77	29.50	55.76
2	58	4	1576	73	15	80	75	50	62	6	78.84	60.78
3	18	4	71	1501	3	313	6	16	59	9	75.05	48.22
4	30	74	33	25	1017	128	33	302	299	59	50.85	56.40
5	58	4	62	72	5	1641	15	52	77	14	82.05	51.54
6	126	16	207	43	36	227	1199	36	60	50	59.95	75.74
7	50	237	248	560	31	52	6	663	141	12	33.15	28.98
8	152	10	94	333	62	380	49	75	817	28	40.85	40.27
9	39	115	70	268	132	91	6	614	331	334	16.70	36.95

References

[1] Bishop, Christopher M.(2006) - Pattern Recognition And Machine Learning Linear Models for Classification - Discriminant Functions Pg. 182

[2] Xianxu Hou (Apr 23, 2015) Logistic and Softmax Regression. Retrieved November 15, 2018, from https://houxianxu.github.io/2015/04/23/logistic-softmax-regression/

[3] Goodfellow .et-al (2016), Deep Learning. Ian Goodfellow and Yoshua Bengio and Aaron Courville, MIT Press, Chp 9 Convolutional Networks,Pg.331, http://www.deeplearningbook.org.