Empirical Evaluation of Machine Learning Algorithms on MNIST data

Harshit Gupta
18111020

I G Prasad
18111025

Jatin Dev
18111027

Mohit Malhotra
18111042

Sri Divya Yaddanapudi
18111073

Diclaimer: The work carried out in the project has not been re-used from any another course project at IITK or elsewhere.

1 Problem Description and Motivation

Classification is one of the important problem in machine learning, having its application in almost all domains such as medicine, E-commerce, robotics, security etc. Plenty of classification algorithms have been already proposed and demand to find further advanced algorithms and tools to solve problems experienced in classification is increasing over time. Hence there is a need to evaluate and compare different algorithms to understand the advantages and disadvantages of each algorithm, type of data they are best suited for, domains on which these algorithms perform better etc. Detailed evaluation of existing methods lead to further improvements and also contribute in designing new advanced algorithms.

In this project we tried to analyze some standard classification algorithms like Decision Trees, Random Forests, Softmax regression, KNN, SVM, Feed Forward Neural Networks, CNN etc. The main objective of this project is to compare the above mentioned classification algorithms with different parameter setting and on different amount of data sets. This project also compares the different models with distorted, white-noise MNIST data and combination of both.

2 Related Work

A lot of work has already been done in this area. In this paper [2], performance of SVMs, neural networks, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees and boosted stumps was evaluated on eleven binary classification problems using different performance metrics such as accuracy, F-score, ROC area, precision/recall, etc.

Performance of algorithms may vary when applied on high dimensional data. In this paper [3], performance of SVMs, neural networks, logistic regression, naive bayes, KNN, random forests, bagged decision trees, perceptrons has been evaluated on high dimensional(750 - 700K) data using three performance metrics accuracy, area under the ROC curve(AUC) and squared loss.

In this project, we have made an attempt to compare classification algorithms like decision trees, random forests, KNN, SVM, neural networks, CNN, softmax regression on MNIST dataset and compare them on performance metrics like accuracy, precision, recall, negative-log-loss, training time, test time. Further, we have also reduced the amount of training data to check which model performs better with less training data. We have also applied these models on noisy data, i.e, we have trained the models with stanadard MNIST data and have tested them on three noisy data sets namely, n-mnist-with-awgn, n-mnist-with-motion-blur and n-mnist-with-reduced-contrast-and-awgn.

3 Tools and Softwares Used

1. Scikit-learn - Is a python based library for machine learning. It contains various models for machine learning algorithms like regression, classficiation, clustering. It also has components to calculate various metrics for the model for the given input data.The implementation of the algorithms is very efficient with explained API documentation available. For this project we have used API of classification algorithms.
2. Python - Language used to run the metrics on the algorithms. It has good libraries developed for machine learning.
3. Tensor Flow - Google provided open source library, used for Deep Learning networks. It is created to run machine learning algorithms on low end processing systems too.
4. Keras - A simple python library which provides API to build complex neural network systems. It is on top of one of three libraries Tensorflow, CNTK or Therano. Very good for prototyping.

4 Machine Configuration

- Processor : Intel Core i5
- Ram : 4GB

5 Description of Data

We are evaluating different Machine Learning algorithms for classification on MNIST dataset available here. This dataset contains 28×28 size images of handwritten digits 0-9. These digits have been centered and size-normalized. It contains 60,000 training examples and 10,000 test examples. This dataset is a subset of large dataset available from NIST(National Institute of Standards and Technology). The set of images in MNIST dataset is a combination of images from two NIST's dataset: Special database 1 (consists of digits written by high school students) and special database 3(consists of digits written by United States Census Bureau). This dataset is provided by Yann LeCun, Professor at The Courant Institute of Mathematical Sciences, New York University.

Figure 1: Sample Images from MNIST dataset (*Source: Wikipedia*)

We also tried the algorithms on noisy data. For that purpose we used another dataset available here. These dataset were created using MNIST dataset by adding:

1. Additive white gaussian noise (n-mnist-with-awgn)
2. Motion blur (n-mnist-with-motion-blur)
3. A combination of additive white gaussian noise and reduced contrast to the MNIST dataset(n-mnist-with-reduced-contrast-and-awgn)

n-mnist-with-awgn : The AWGN dataset is created using an Additive White Gaussian Noise with signal-to-noise ratio of 9.5. This emulates significant background clutter.

n-mnist-with-motion-blur : The Motion Blur filter emulates a linear motion of a camera by pixels, with an angle of degrees. The filter becomes a vector for horizontal and vertical motions. In this dataset,a value of 5 pixels and value of 15 degrees is used in the counterclockwise direction.

n-mnist-with-reduced-contrast-and-awgn : In this dataset, the contrast range was scaled down to half and was applied with an Additive White Gaussian Noise with signal-to-noise ratio of 12. This emulates background clutter along with significant change in lighting conditions.

Figure 2: n-mnist-with-awgn

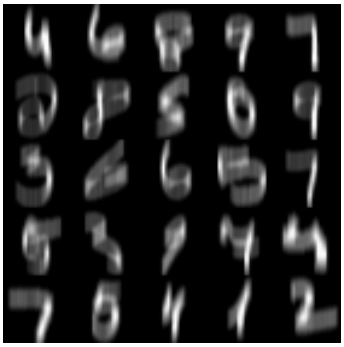

Figure 3: n-mnist-with-motion-blur

Figure 4: n-mnist-with-reduced-contrast-and-awgn

Source: csc.lsu.edu/ saikat/n-mnist/

6 Classification Algorithms

6.1 KNN

KNN is a non parametric classification algorithm in which the entire training set is used during the prediction. Specified number of nearest neighbours(K) are used for prediction of classes for the new training data. Increasing the value of K results in smoother prediction, however increasing it beyond the threshold will underfit and results in poor training and test errors. In this project, the value of K has been varied to find the optimum value of K for the given training dataset and to generalize how bias and variance vary with respect to different value of K.

6.2 Softmax Regression

Softmax Regression generalizes Logistic Regression for Multi-class classification. In our project, we have tried out different solvers: lbfgs, newton-cg, sag and liblinear available in sklearn library in Python.
liblinear: This solver uses a coordinate descent (CD) algorithm that solves optimization problems by successively performing approximate minimization along coordinate directions or coordinate hyperplanes.
newton-cg: Uses a quadratic approximation(first and second order derivatives). At each iteration it approximates $f(x)$ by a quadratic function around x_n, and then takes a step towards the maximum/minimum of that quadratic function.
lbfgs: Limited-memory BroydenFletcherGoldfarbShanno Algorithm. It is analogue of the Newtons Method but here the Hessian matrix is approximated using updates specified by gradient evaluations (or approximate gradient evaluations). The term Limited-memory simply means it stores only a few vectors that represent the approximation implicitly.
sag: By incorporating a memory of previous gradient values the SAG method achieves a faster convergence rate than black-box SG methods.

6.3 SVM

Support vector machines are used for classification. SVM allows us to find the hyper planes between 2 classes and also guarantees the margin to be maximized. SVMs can be easily kernelized to classify data which are not linearly separable. The disadvantage with SVM is that it takes more training time when huge training data is available. But on the better side SVM is not much affected by noise and

not much prone to over fitting. Linear SVM and SVM with polynomial kernel have been trained on the MNIST data-set in this project for our comparison. The degree of the polynomial kernel has been varied to compare between different models.

6.4 Decision Trees

A regression and classification technique. It acts swiftly on the test data. It can draw both linear and non linear decision boundaries. There are many variations of decision trees: ID3(Iterative Dichotomiser 3), C4.5, CART(Classification and Regression Tree), CHAID(CHi- squared Automatic Interaction Detector). The time taken for training depends on the number of the features and the depth on which the tree is being trained. The visualization of the trees are simple and easy to understand. In this project the decision trees have been trained with different depths, number of features, type of splitter to use..

6.5 Random Forest

Random Forest is an Ensemble classification. The forest is created by a number of decision trees. The decision trees are picked at random. The accuracy of the model is picked up when we are using an ensemble classification with number a decision trees. In the project we have tested with varying number of estimators, depths for each decision tree, number of features in each decision tree, criteria for the comparison. Also compared the results with grid and random search.

6.6 Feedforward Neural Networks

Neural networks consist of an input layer, an output layer and one or more hidden layers. Here, we have used the following architectures of feed forward neural networks.
For 1 hidden layer, number of hidden units(K)= 2, 4, 8, 16,....., 1024
For 2 hidden layers, number of hidden units in 1st layer(K) = 2, 4, 8,...., 512 and number of hidden units in 2nd layer(L) = 2, 4,... K/2
For 3 hidden layers, number of hidden units in 1st layer(K) = 2, 4, 8,...., 256, number of hidden units in 2nd layer(L) = 2, 4,... K/2 and number of hidden units in 3rd layer(M) = 2, 4, ..., L/2
For each architecture, we regularize the model using l2 regularizer with regularization hyperparameter 0, 0.01 and 0.1

6.7 Convolution Neural Network

CNN uses an input layer, an output layer and multiple hidden layers. The hidden layer contains convolutional layer, pooling layers and fully connected layers. In our implementation of CNN, we have used a Convolution layer followed by a Maxpooling layer where a 2D matrix is used for pooling. The maxpooling layer is followed by a flattening layer and then a fully connected hidden layer with ReLU activation layer is used with varying number of hidden units(we are trying different values here). Then we have applied a dropout layer to prevent overfitting and then output layer has 10 units for each class. These units output the score for each class, the class with maximum score is predicted. We have tried for different values of kernel size, pooling window size and number of units in hidden layer.

7 Experimental Results

7.1 Model Performance for entire MNIST dataset :

This table shows the best obtained result for various models along with the parameter configuration used.

Model	Configuration	Accuracy	Train time(secs)	Test time(secs)	Negative log loss	F1 Score	Precison	Recall
Random Forest	250 estimators	97.25	571.42	5.379	0.234	0.97	0.97	0.97
Decision Tree	Depth = 25	88.31	31.85	1.367	3.463	3.463	3.463	3.463
Linear SVM	C=1	90.66	495	0.54	0.427	0.91	0.91	0.91
KNN	K=3	97.05	30.75	913.03	0.4687	0.97	0.97	0.97
Soft max	lbfgs solver	91.72	80	0.06	0.379	0.92	0.92	0.92
Kernelized SVM	Polynomial Kernel, d = 2	98.06	1941	104	0.062	0.97	0.97	0.97
NN	L1(512units), L2(256units)	98.45	167.78	1.66	0.062	0.98	0.98	0.98
CNN	kerne(5,5) pool(3,3) 128 units	99.09	677	36	0.33	0.99	0.99	0.99

7.2 Accuracy for varying training Training Examples :

Model vs Training examples	100	500	750	1000	5000	10000	20000	30000
Decision Tree	38.41	53.75	63.01	67.96	78.38	81.06	84.06	84.85
Random Forest(250)	66.53	83.77	87.52	88.63	94.25	35.90	96.23	96.44
KNN(k=3)	64.76	80.61	84.07	86.22	93.39	94.63	-	-
Softmax	67.82	82.38	83.13	83.08	85.46	87.02	90.18	91.27
Linear SVM	72	83.8	83.6	83.2	84.48	88.52	90.17	90.78
Kernelized SVM(d=2)	65.49	86.23	89.82	90.54	94.96	96.17	97.18	97.79
NN	66.87	85.03	88.28	88.94	94.93	96.11	97.35	97.24
CNN	66.25	87.57	91.84	93.52	96.93	98.14	98.64	98.76

7.3 Performance of Different Models for Blurred, White-Noise Data and combination :

Model/Noise	n-mnist-with-awgn	n-mnist-with-motion-blur	n-mnist-with-reduced-contrast-and-awgn
KNN(K = 3)	96.85	93.27	77.66
Decision Tree	25.82	49.79	21.77
Random Forest	58.59	78.08	52.11
Softmax	36.68	87.71	29.34
Kernelized SVM	89.84	95.96	78.04
Neural Networks	72.12	96.44	90.64
CNN	82.37	90.08	62.0

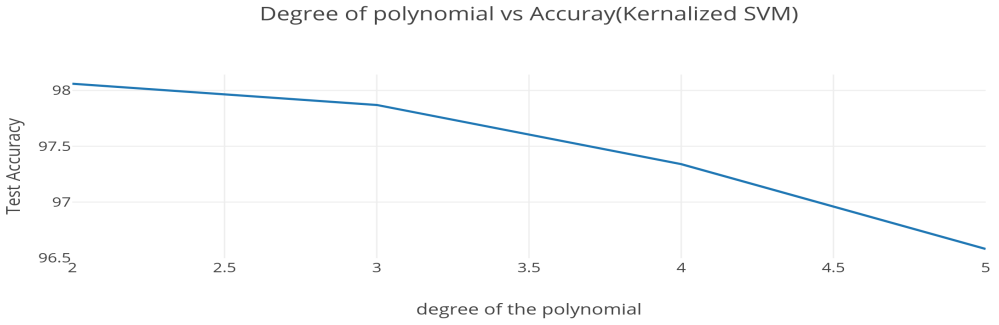

Figure 6

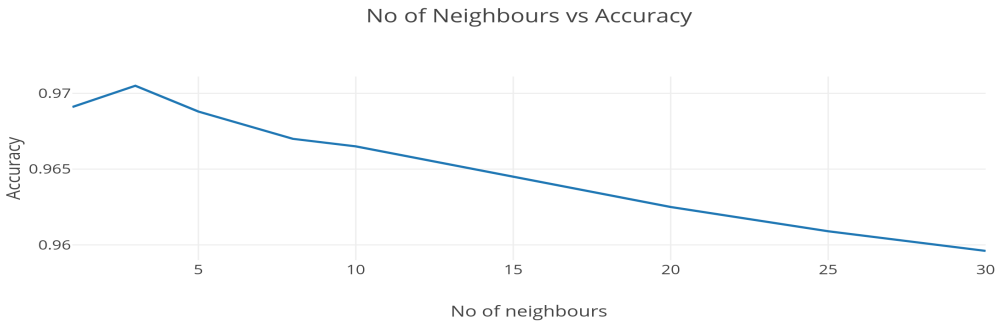

Figure 7

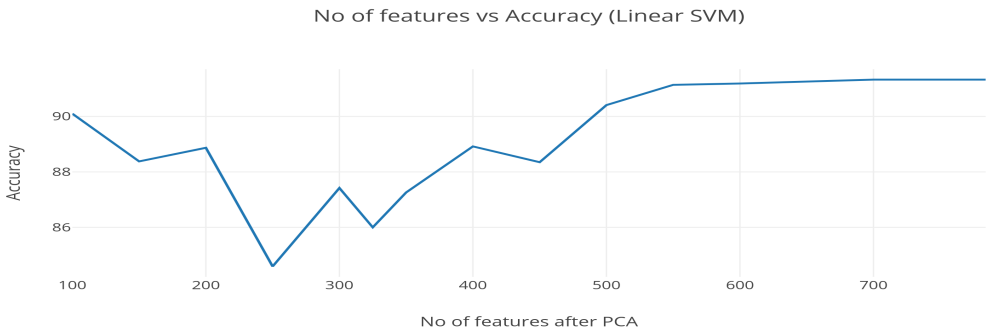

Figure 8

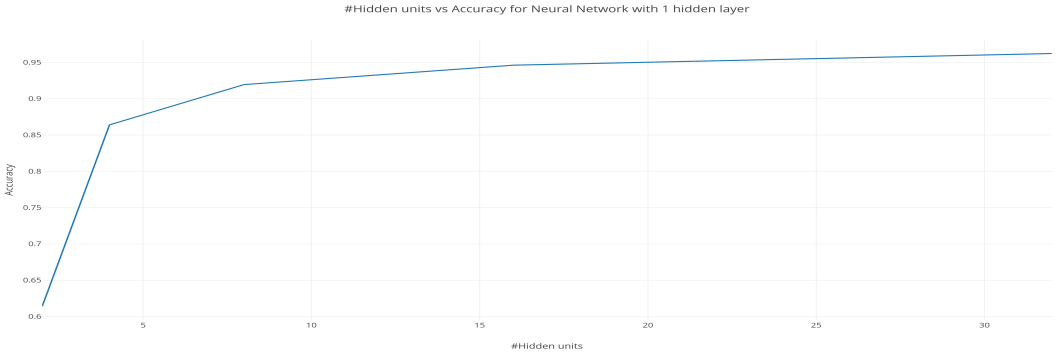

Figure 9

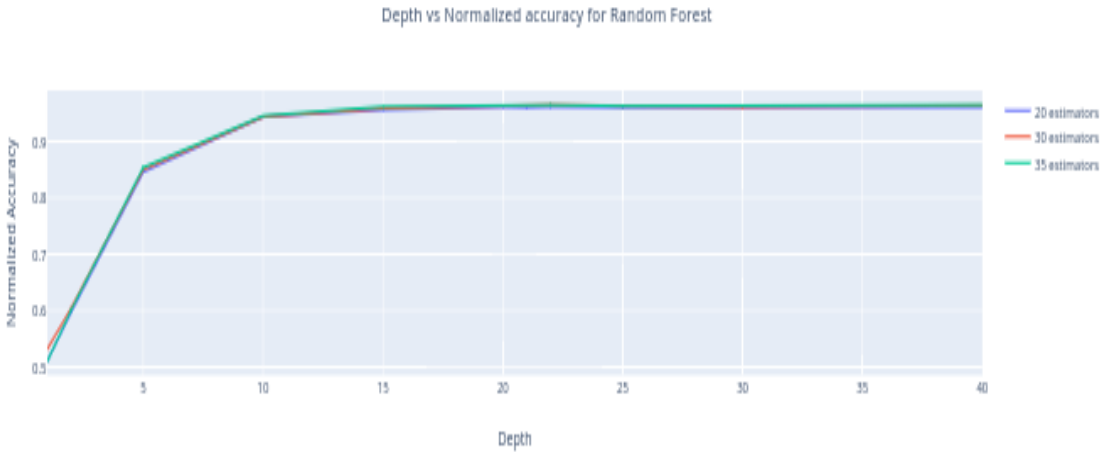

Figure 10

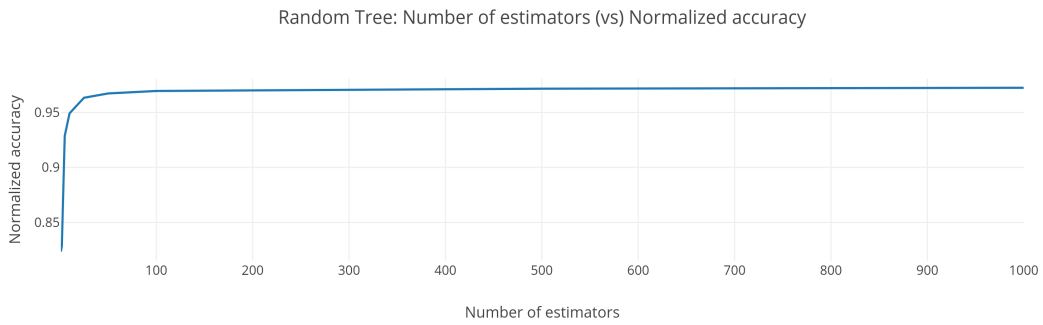

Figure 11

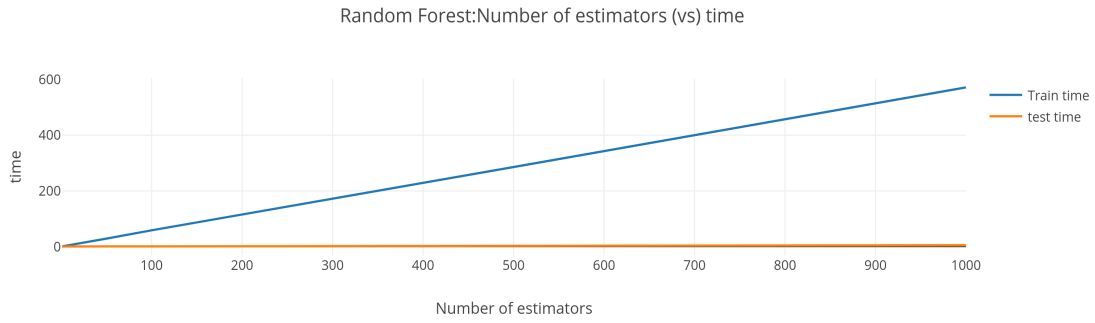

Figure 12

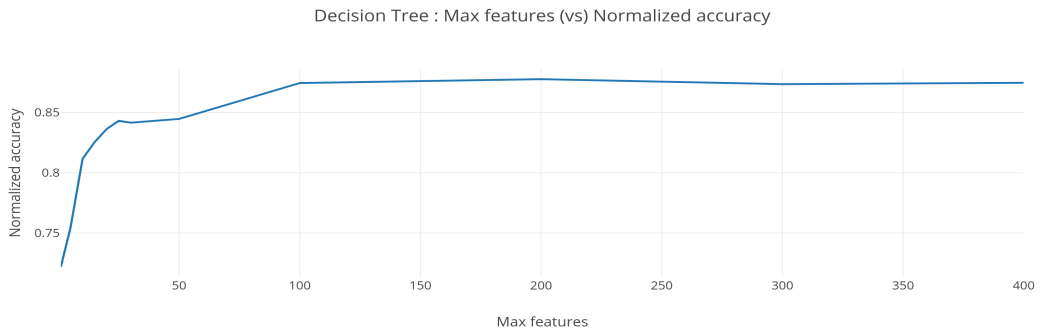

Figure 13

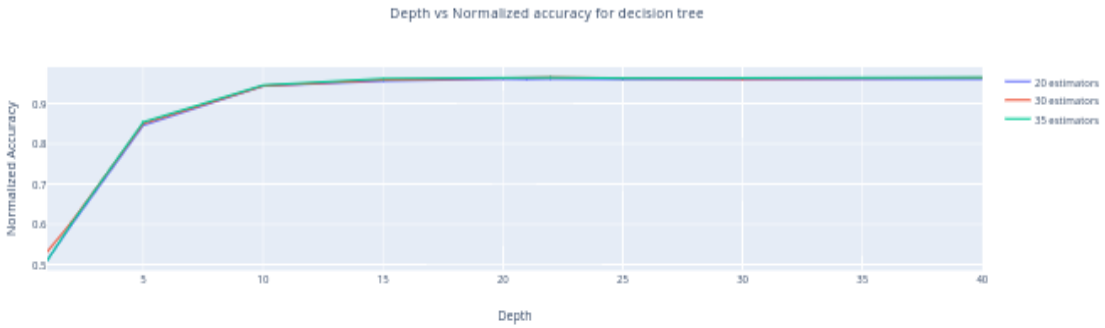

Figure 14

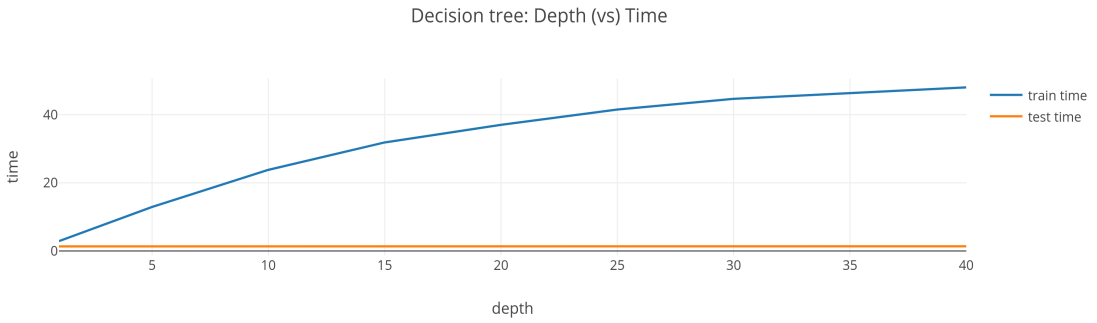

Figure 15

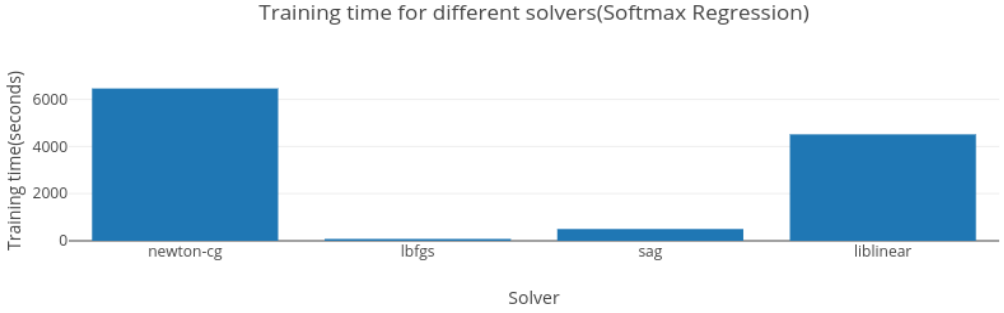

Figure 16

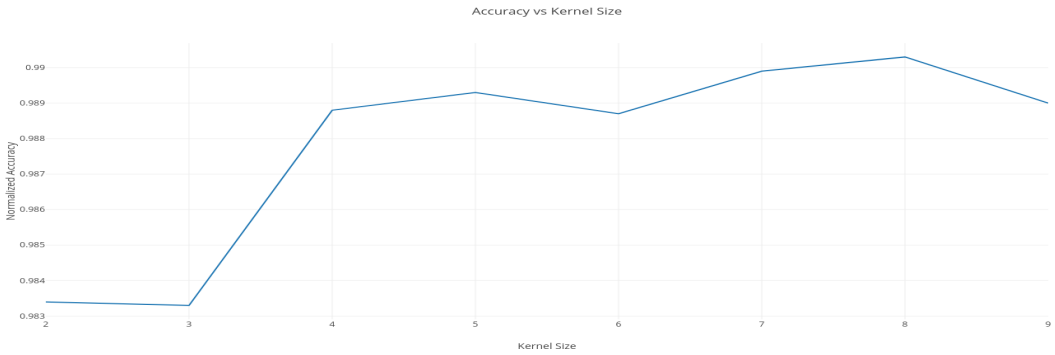

Figure 17: CNN: Accuracy vs Kernel Size

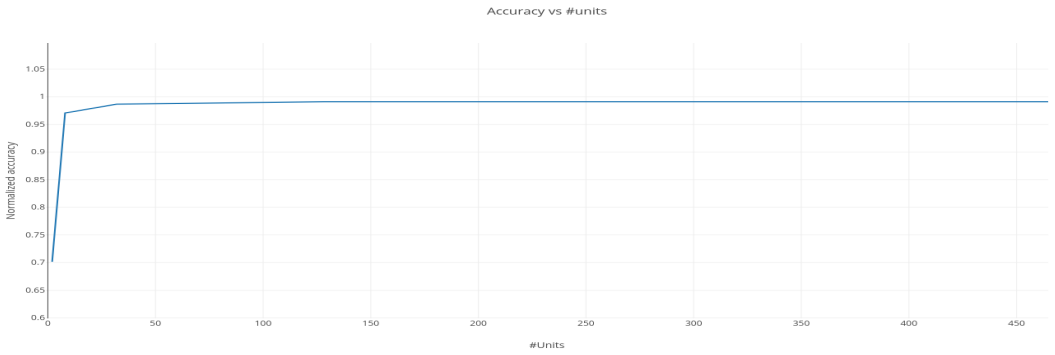

Figure 18: CNN: Accuracy vs Number of units in hidden layer

8 Takeaway from the project

- Learnt the pros and cons of various machine learning algorithms.
- Learnt how to cross-validate the model for hyper-parameter tuning.
- How various parameters impact a model. Which parameter setting is better suited for which condition.
- Validation on what we learnt is actually what is happening
- Learnt how to use ensemble methods like Random Trees for prediction
- Learnt how to debug different ML Algorithms

9 Possible Future Improvements

Different classification algorithms were trained and tested on MNIST data without any data transformation/feature learning. However feature learning could be used before training any model using Deep learning, Landmark approach or any other sophisticated feature learning techniques to see if performance gets better. Techniques like deskewing could be used. In this project only MNIST data was considered to compare the performance of different Classification Algorithms. Other datasets from text or vision domains can be used and conclusions can be drawn regarding which model works better in what domain and under what assumptions.

10 Conclusion:

In this project we focused on comparing the classification algorithms on MNIST dataset. Different models have been trained with different hyper-parameters and with different amount of training data to compare how each model behaves. Also blurred , white noise and combination MNIST data have been tested against each model to check robustness of each model. We have also reduced the dimensionality of the data and compared the training and test time of different models for reduced data set.

SVM took longer training time for PCA reduced data than original data. One possibility of this could be sparsity in the original data and data became dense after applying PCA. Both kernelized SVM and KNN need training data to be present in memory during test time. However kernelized SVM took less test time than KNN as only the support vectors are used during test phase where as all training data are used in the case of KNN. Even though there is no training in KNN , KNN took a small amount of time to store the training data efficiently using KD-Tree or Ball Tree to efficiently find the nearest neighbours. CNN gave the best possible result among all other model as expected. Possible reason being CNN took advantage of locality and captured the local details of every pixel in the process called convolution. However when test data had some distortion and noise NN, K-Means and polynomial kernel SVM performed better than other models.

References

[1] Shweta C. Dharmadhikari, Maya Ingle, Parag Kulkarni
Empirical Studies on Machine Learning Based Text Classification Algorithms

[2] Rich Caruana, Alexandru Niculescu-Mizil
An Empirical Comparison of Supervised Learning Algorithms

[3] Rich Caruana, Nikos Karampatziakis, Ainur Yessenalina
An Empirical Evaluation of Supervised Learning in High Dimensions