Predicting rainfall using Ensembles of Ensembles.* †

Prolok Sundaresan, Varad Meru, and Prateek Jain‡
University of California, Irvine
{sunderap,vmeru,prateekj}@uci.edu

Abstract

Regression is an approach for modeling the relationship between data `X` and the dependent variable `y`. In this report, we present our experiments with multiple approaches, ranging from Ensemble of learners to Deep Learning Networks on the weather modeling data to predict the rainfall. The competition was held on the online data science competition portal 'Kaggle'. The results for weighted ensemble of learners gave us a top-10 ranking, with the testing root-mean-squared-error being 0.5587.

1 Introduction

The fundamental task of the in-class Kaggle competition is to predict the amount of rainfall at a particular location using satellite data.

2 The DataSet

2.1 Features

2.2 Visualizing the Data

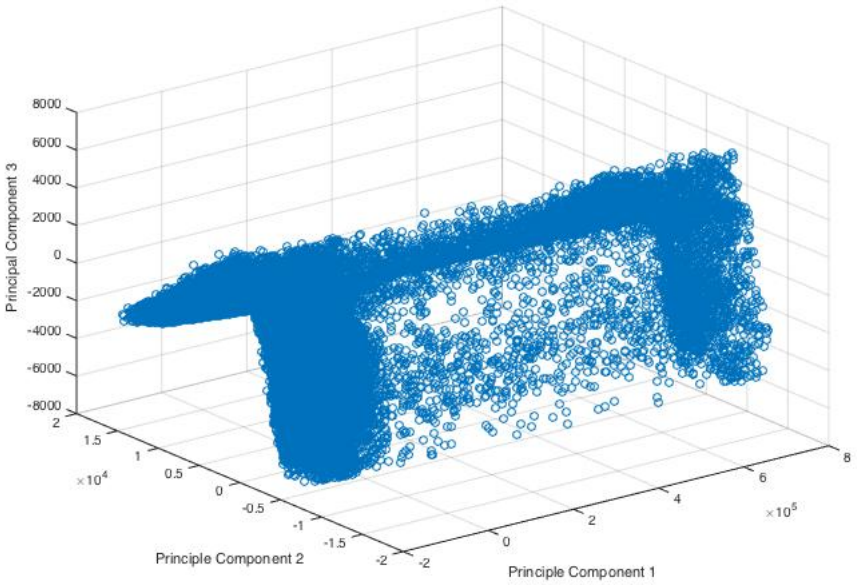
Visualizing the data was a difficult task since the data was in 91 dimensions. In order to look for patterns in the data and visualized it, we applied SVD technique to reduce the dimensionality of the features to 2 principle dimensions. Then we applied k means clustering with k=5 on the data with 91 dimensions and plotted the assignments in the 2 dimensional transformed feature space. We saw patterns in the data(show the figure). Espeically some points were densely clustered and some were sparse.

To visualize it better, we transformed the feature in 3 dimensional space, and saw that the points were clustered around 3 planes.

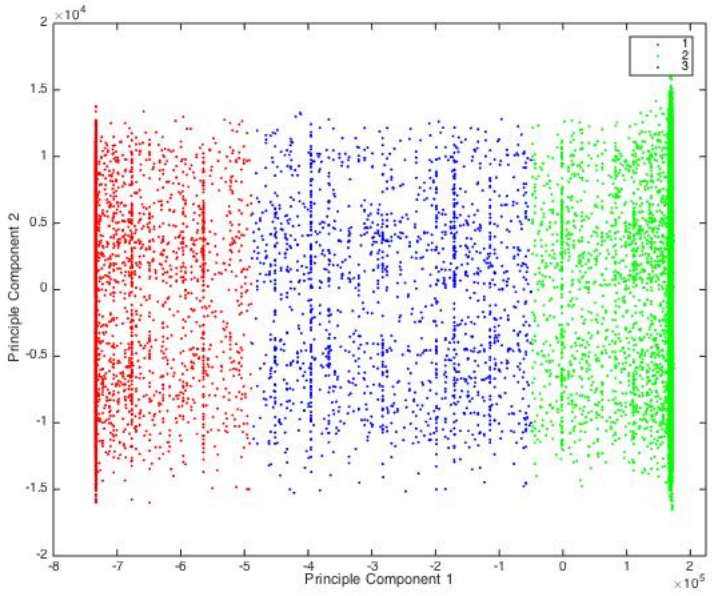
*The online competition is available at the Kaggle website `https://inclass.kaggle.com/c/how-s-the-weather`. The name of the team was `foo_bar`

†This work was does as a part of the project for CS 273: Machine Learning, Fall 2014, taught by Prof. Alexander Ihler.

‡Prolok Sundaresan: Student# 66008474, Varad Meru: Student# 26648958, Prateek Jain: Student# 28321844



(a) Visualizing the data in 3 dimensions



(b) Train-Validation-Test error plot for 20 neuron hidden layer

Figure 1: Visualizing the principle components of Data

3 Machine Learning Models

3.1 Mixture of Experts

As seen from our visualization, we could identify two highly dense areas of the feature data on either side of a region of sparsely distributed data. The idea behind using the mixture of experts approach was, that intuitively, it would be difficult for a single regressor to fit the dataset, since the distribution is non-uniform.

We decided to split the data into clusters. To cluster the data, we used several intitialization of the k means algorithm with the kmeans++ intitialization. We decided to use K=3 for 3 evident clustering of the data in the visualization.

Since each of the clusters got a subset of a points from the original dataset, number of data points per cluster was not a very large number. Our concern with this was that any model we chose would overfit the data in its cluster.

Therefore, we used the ensemble method of gradient boosting for each of the clusters. Since, in gradient boosting, we start with an underfitting model and then gradually add complexity, the chances of overfitting would be less in this model. We decided to use Decision stumps as our regressors for the boosting algorithm.

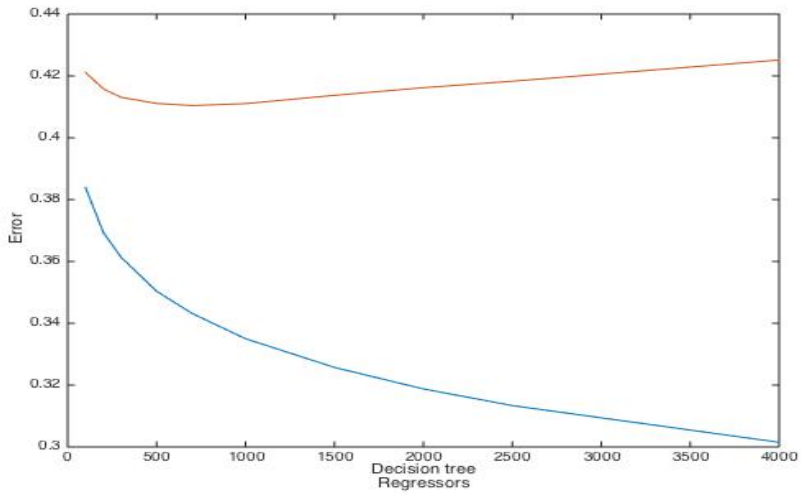


Figure 2: Mixture of Experts Error

3.2 Neural Networks

We implemented various types of neural networks, ranging from single layer networks to 3-layer sigmoidal neural networks.

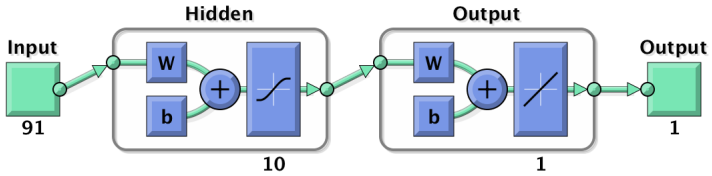


Figure 3: Single Layer Architecture.

Single Layer Network

We build the neural network using the MATLAB's Neural-Network-Toolkit and PyBrain library implemented in Python. For the MATLAB implementation, there were various runs made for different number of neurons in the hidden layer. The architecture of the neural network can be seen **??**. The Figure 4 show the train-test-validation plots for different network architectures. The dataset was distributed into 70% (Training), 20% (Validation) and 10% (Testing) section for the neural network to run. The Figure 3.2 shows the performance of the models learned. It was seen that the neural networks started to overfit as the number of neurons were increased more than 40.

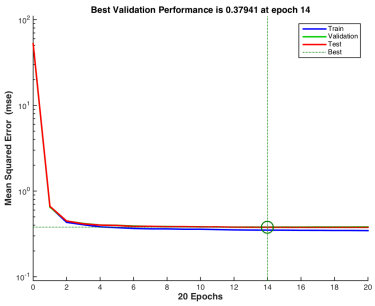
# of Neurons	Training Error (RMSE)	Testing Error (RMSE)
10	0.5986	0.61341
20	0.5875	0.61301
50	0.5852	0.62889

Table 1: RMSE Error rates for different network architectures.

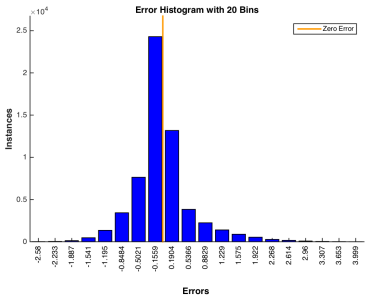
It was observed that the learner could not learn very accurately as the data a lot as the data was not much for the neural network to learn on.

Deep Networks

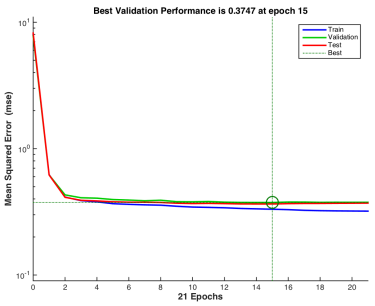
For this project, we tried using deep networks as well. The deep network was made using PyBrain. We tried using different activation functions and architectures to understand how deep networks would work. The architecture shown in section 3.2 had 2 layers - visible later



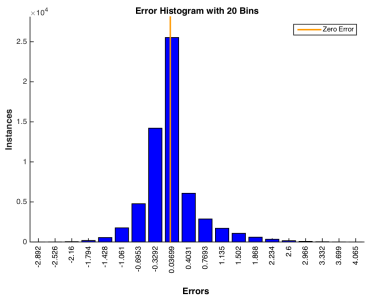
(a) Train-Validation-Test error plot for 10 neuron hidden layer



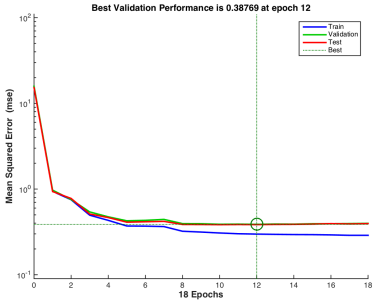
(b) Error distribution histogram for 10 neuron hidden layer



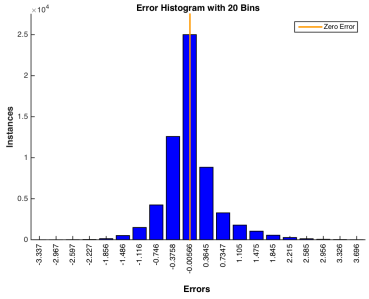
(c) Train-Validation-Test error plot for 20 neuron hidden layer



(d) Error distribution histogram for 20 neuron hidden layer

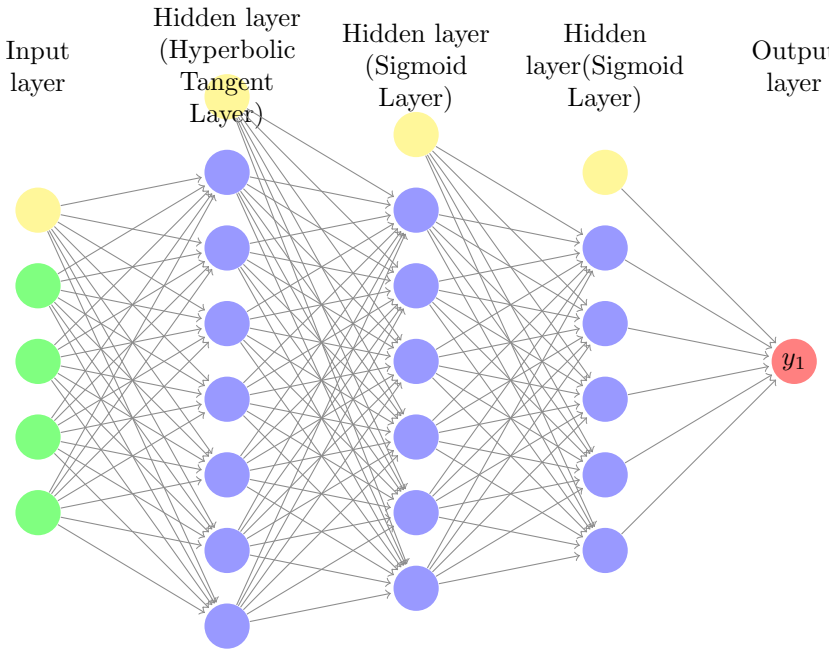


(e) Train-Validation-Test error plot for 50 neuron hidden layer



(f) Error distribution histogram for 50 neuron hidden layer

Figure 4: Plots of various Train-Validation-Test error for number of neurons = [10, 20, 50]



3.3 Gradient Boosting

3.4 Random Forests

4 Ensemble of all Learners

5 Conclusion

References