Machine Learning
Contest Report

Deddy Jobson(EE15B125) & Shubhangi Ghosh(EE15B129)
Team Name: Ensemble, nous gagnons!
Team Members: juste moi(Deddy Jobson), Shubhangi Ghosh
Department of Electrical Engineering

November 12, 2017

1 INTRODUCTION:

This report presents the techniques used to handle a classification problem of 29 classes with 9501 training instances of 2600 dimensions. The evaluation metric is the mean F1-score.

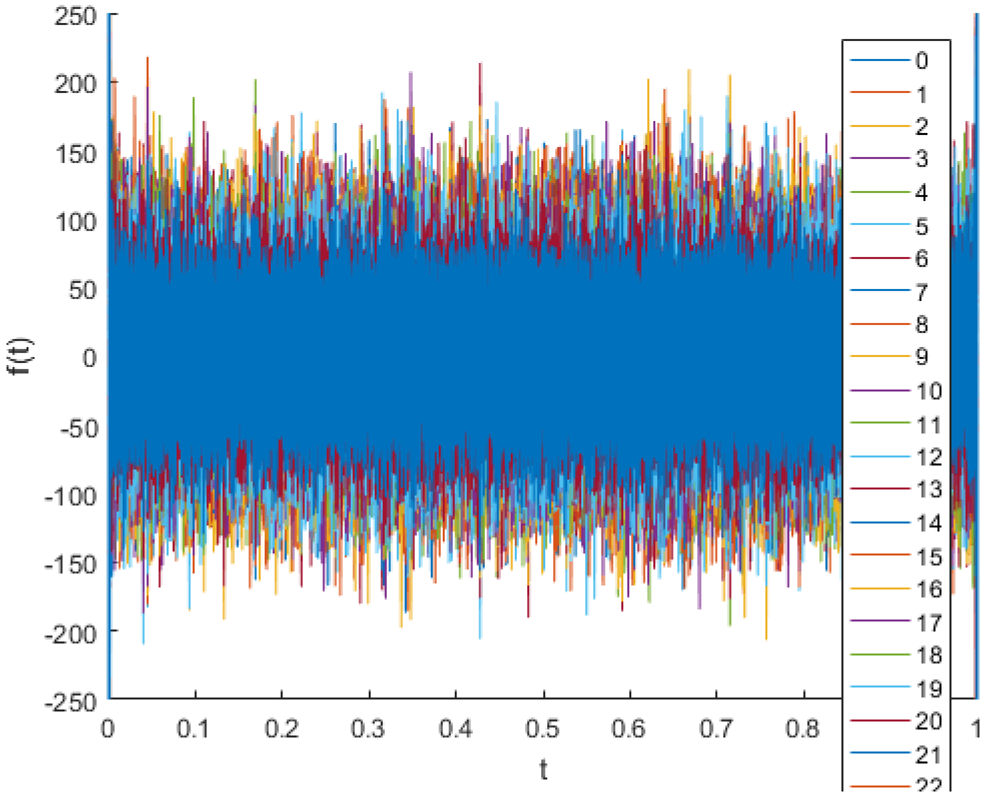
2 DATA INTERPRETATION AND CORRESPONDING INTUITONS:

1. The number of dimensions is extremely high, which gives us the intuition that Naive Bayes (with multinomial likelihood), could be a good classifier to try out. Since Random Forests behave really well when irrelevant features are present or the features have skewed distributions, trying out Random forests may be a good idea.

2. As the number of dimensions is very high, the dataset is very **sparse**. So techniques like dimensionality reduction should be tried. Also, methods using euclidean distance measures would fail in such high dimensions. So Manhattan distance is used wherever possible.

3. The number of training instances is pretty low, keeping in mind the huge number of dimensions. Also, since there are a large number of classes(29), the number of training instances that we would encounter per class is very low. Since SVMs give good results with less data, since the classifier boundary depends only on the support vectors, SVM would be a good classifier to try out. Bagging SVMs may also be a good ideas.

4. Since the data is most likely not linearly classifiable, Gradient Boosted Decision Trees might be a good classifier. The data can be shown to be non-linearly classifiable by it's Andrew's plot(which is a way of plotting and checking linearity of high dimensional data) :



All the classes appear to be completely mixed up, so the data is definitely not linearly separable. Gradient Boosted Decision Trees or Boosting of linear classifiers, such as SVMs may give good results.

5. The data has a lot of missing values, so classifiers which can handle missing values, such as Decision Trees with efficient methods to handle missing values must be used.

6. Missing values can either be handled by:

   (a) Imputation

   (b) Introducing a new categorical attribute "Missing"
   Datapoints could go missing due to a systematic reason, but since our attributes were numeric and not categorical, we haven't implemented this.

   (c) Surrogate splits on decision trees.

7. For simplicity, Imputation has been used to handle missing values. A python module called **fancyimpute** has been used to implement more advanced forms of Imputation. sklearn Imputer with mean Imputation has also been used with some classifiers. (I think handling of missing values deserves its own section. We can try to mention mean imputation, knn, etc.)

3 MISSING VALUE IMPUTATION METHODS THAT HAVE BEEN TRIED:

3.1 MICE:

- MICE stands for Multivariate Imputation by Chained Equations.
- In this method, the missing variables were predicted from all the other variables.
- Unlike other methods, the prediction was made taking all missing values into account, not independantly.

3.2 Soft Impute:

- This method of imputation tries to reduce the rank of the resulting matrix.
- However, the performance of this imputation method is not as good as that of the following method.

3.3 Knn:

- In this method, the missing attributes of each instance has been filled with the corresponding attribute of the nearest neighbour.
- The nearest neighbour was found using the attributes that were not missing in both instances.
- In this case, Euclidian distance was used, however other methods like Manhattan distance could also be used.
- The best results were obtained when k was set to 3.

3.4 IMPUTATION WITH STRATEGY MEAN

1. This has the benifit of not changing the sample mean for the feature.
2. Mean imputation attenuates any correlations involving the features that are imputed, since there is no guaranteed relationship between imputed feature and other features, unlike Regression Imputation.

4 DIMENSIONALITY REDUCTION METHODS THAT HAVE BEEN TRIED:

4.1 PCA:

- The dimensions were filtered based on the ration of it's variance with that of the Principal Component Axis.
- The threshold of 0.95 was set so that any component whose variance is less than 5% of PCA is removed.
- 871 dimensions were left, but the performance of the classifiers dropped significantly, so it wasn't used further.

4.2 Isomap:

- This is a non linear method of dimension reduction.
- The dimensions were reduced keeping the ratio of distance of neighbours fixed.
- The performance was worse than PCA, so in the end, all features were used.

4.3 FEATURE SELECTION USING VARIANCE THRESHOLD:

1. Since this is a problem with large no. of dimensions, dimensionality reduction methods have been tried.
2. Dimensions above a certain variance threshold were chosen, to keep only most useful dimensions.

3. But since all dimensions had similar variances, this wasn't very helpful, and only few features were eliminated.

4. This gave an improvement of about 2% in cross-validation scores.

5 CLASSIFIERS THAT HAVE BEEN TRIED:

5.1 RANDOM FORESTS:

5.1.1 PREPROCESSING:

1. **IMPUTATION:**
   Values have been Imputed with Sklearn Imputer with Strategy Mean.

2. **STANDARDISATION:**

   (a) Training data has been standardised to 0 mean and unit variance.

   (b) Test data has been fit according to the standardisation.

   (c) Random forests typically don't require data standardisation, but standardisation has been implemented anyway since, comparing and splitting between very large or very small numbers may cause trouble.

5.1.2 INTUTION:

1. Decision trees are known to be more robust to numerical instabilities. Hence, the given data has a lot of missing values, which have been imputed with mean strategy, numerical inconsistencies in the data may be high. Hence, this classifier has been tried.

2. Random Forests are known to minimise the high variance problem that decision trees usuall have, hence random forests are expected to give more robust classifiers.

5.1.3 PARAMETER SETTING:

Best-fit Parameters have been estimated by cross-validation:

1. min_samples_split=160

2. min_samples_leaf=80

5.1.4 ACCURACY SCORES:

For these best-fit parameters, the F-score on test set was : 18.16%.

5.1.5 WHY RANDOM FORESTS MAY HAVE FAILED:

1. Random Forests generally needs larger number of instances to work its randomization concept well and generalize to the data.

2. Since the no. of training instances here is relatively low keeping in mind the no. of dimensions, random forests may have failed.

5.2 GRADIENT BOOSTED DECISION TREES:

5.2.1 PREPROCESSING:

1. **IMPUTATION:**
   Values have been Imputed with Sklearn Imputer with Strategy Mean.
2. **STANDARDISATION:**
   (a) Training data has been standardised to 0 mean and unit variance.
   (b) Test data has been fit according to the standardisation.
   (c) GBDTs also don't essentially require data standardisation, but data has still been standardises because:
      i. Since the range of values of raw data varies widely, objective functions such as squared error loss function will not work properly without normalization.
      ii. If one of the features has a broad range of values, the distance measure will be governed by this particular feature.

5.2.2 INTUTION:

1. This is again a decision tree approach and is expected to be good at handling numerical instabilities.
2. The data given is most likely not linearly separable and hence GBDTs are expected to be able to give arbitrary class boundaries, so this approach may prove to be helpful.

5.2.3 PARAMETER SETTING:

Best-fit Parameters have been estimated by cross-validation:

1. min_samples_split=160
2. min_samples_leaf=80

5.2.4 ACCURACY SCORES:

For these best-fit parameters, the F-score on test set was : 21.839%.

5.3 BAGGED GAUSSIAN NAIVE BAYES:

5.3.1 PREPROCESSING:

1. **IMPUTATION:**
   Values have been Imputed with Sklearn Imputer with Strategy Mean.
2. **STANDARDISATION:**
   (a) Training data has been standardised to 0 mean and unit variance.
   (b) Test data has been fit according to the standardisation.
   (c) Naive Bayes also typically doesn't require feature scaling, but it has been implemented anyway to handle numerical inconsistencies better.

5.3.2 INTUTION:

1. Naive Bayes is typically efficient at handling high dimensional problems, and hence was thought to be a classifier worth trying.

5.3.3 PARAMETER SETTING:

1. Default parameters have been used.
2. Cross validation could have been used to improve parameter settings.

5.3.4 ACCURACY SCORES:

For these best-fit parameters, the F-score on test set was : 22.6%.

5.4 ARTIFICIAL NEURAL NETWORKS

5.4.1 PREPROCESSING:

All the datapoints were bounded between zero and one, therefore no preprocessing of input data was required for ANNs after imputation of missing values.

5.4.2 PARAMETER SETTING:

As ANNs have many hyperparameters, a parameter search script was created in order to search for the optimal parameters. The set of hyperparameters that gave the best accuracy scores were then used to train the neural networks. Cross validation could be used in order to reduce the variance of predictions.

The following parameters could be varied

- Number of layers
- Number of perceptrons per layer
- Type of activation function
- Loss function to optimize
- Validation split
- Regularization parameters

Some of the parameters weren't varied due to lack of suitable alternatives such as categorical accuracy for loss function, but the rest were chosen after testing out cvarious configurations.

5.4.3 STRUCTURE :

Artificial Neural Networks have great flexibility when it comes to structure.

To begin with, the neural networks were created with a single hidden layer. The performance of those neural networks peaked at 32% categorical accuracy. The following were the structures used

- (2600,1397,29)
- (1300,597,29)
- (1300,252,29)

Then an additional layer was added to stengthen the neural network.

- (2600,1487,493,29)
- (1300,677,295,29)
- (1300,597,597,29)

This however didn't work as the neural network was already complex enough to overfit the data. Also, the performance improvement was more dependant on the width of the network rather than the depth.

So in the end, wider networks were used with regularization.

5.4.4 REGULARIZATION :

There are many ways to regularize neural networks. Some methods which have been tried here are

- L1 regularization
- L2 regularization
- Dropout
- Early stopping

In the end, a combination of L2 and dropout was used to prevent overfitting without underfitting the model to the data. Early stopping was also experimented on with the help of a validation set left untouched by the model.

5.4.5 ACCURACY SCORES:

The first neural networks had a low f1 score of about 0.13. Then after further improvements, the f score improved to about 0.32. Finally, with the help of multiple models and using a majority vote, the final prediction stands at 0.392.

5.5 BOOSTING ANNs WITH SVMs

5.5.1 PREPROCESSING:

1. The sum of class probabilities from Neural Networks has been scaled down to between 0 and 1.

5.5.2 PARAMETER SETTING:

By cross validation best fit parameters for SVM were found to be:

1. RBF kernel
2. c =100
3. gamma = 0.01

A parameter search script was created in order to search for the optimal parameters for Neural Networks.

5.5.3 ACCURACY SCORES:

1. The cross validation score was 41% in this case.
2. But the F-score on test set was merely 19%.

5.5.4 WHY THIS METHOD MAY HAVE FAILED:

1. Neural network probabilities may have overfit to training data.
2. Thus, results on test data may not have been so good.

6 References

- Distances in Higher Dimensions, Charu C. Aggarwal, Alexander Hinneburg, Daniel A. Keim
- Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.