Offline Handwritten Character Recognition

Nikhil Karthik Pamidimukkala and Group

April 28, 2019

My Contributions

3. Feature Extraction - Nikhil
4. Similarity of Characters - Nikhil

Classifers - SVM, Random Forest - Nikhil

Abstract

The task of accurately classifying offline handwritten characters is an important step in building an automated handwriting recognition system. However, this a challenging task due to the inherent variations in handwriting style of people. In this project, we explored different classifiers and their performance in recognizing letter of 35 uppercase handwritten characters which consists of alphanumerics (0-9, A - Z). The classifiers are built based on data which consists of 3136 pixels which form a 56 by 56 bitmap representing the handwritten character. In this paper we present the methodologies followed and results obtained from our analysis.

1. Introduction

Offline Handwritten Character recognition is an interesting yet challenging area of research due to the variations in handwriting style of people.It has a wide range of applications such as recognizing signatures, identifying letter zip codes and forensic evidence etc. However, while building an automated handwriting identification systems it is imperative to have an idea of the letter of a particular ink-blob (character). This information can help in the prediction of the writer of the document which corresponds to the mechanics used to generate the ink-blobs and independent of the author of the document. To perform this task, the first step is to efficiently preprocess data from the documents. In this project, we used two preprocessed data set which consists of 3136 pixels represented by binary numbers for each character that form a 56 by 56 bitmap as shown in Figure 1.

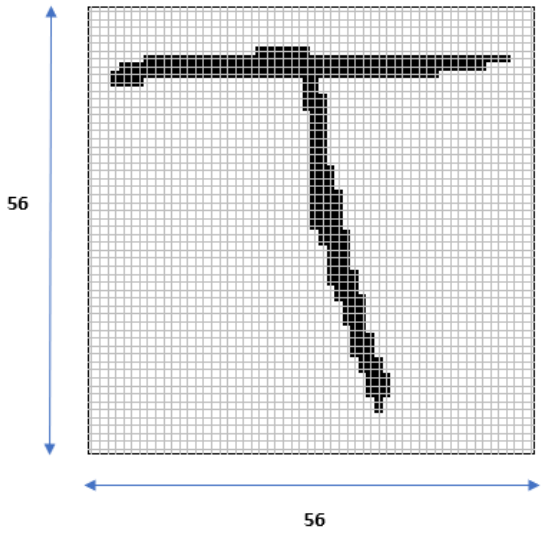

Figure 1: 56 by 56 image of character 'T'

1.1 Data Description

It is more often in real-world that data without labels is more abundant than that of data with labels. To build our classifiers we had two data sets at our disposal.

- **Labeled Data** : 1000 handwritten characters with labels which are Upper Case characters (A-Z) except X and numerics 0-9. The class distribution of characters in the labeled data is shown in Figure 2.

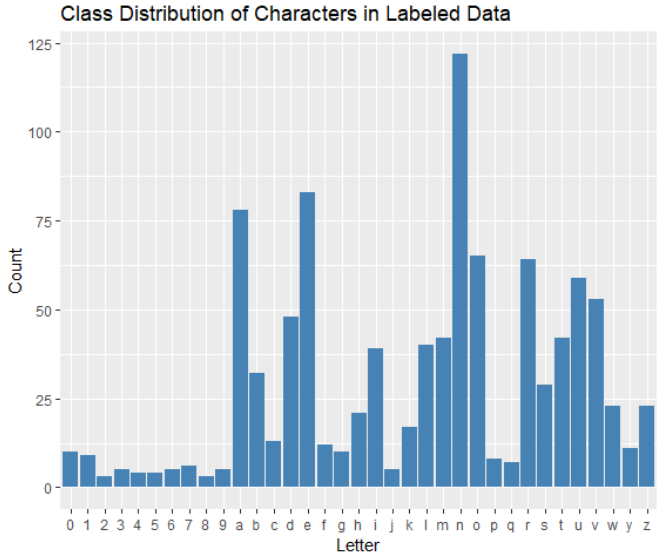

Figure 2: Class Distribution of characters in Labeled Data

- **Unlabeled Data** : 10000 handwritten characters without labels.

It is very important to efficiently leverage the unlabeled data. The process we adopted to label the unlabeled is mentioned in section **[6]**. The classifiers were built using both these data sets.

2. Project Snapshot

To address the challenge at hand the following steps were followed:

- Feature extraction
- Split the feature extracted labeled data into Training and Validation sets.
- Train multiple classifiers on the training data.
- Select one classifier with the highest validation/test set accuracy.
- Use the selected classifier along with a manual labelling approach to label the unlabeled data which is explained in detail in section **[6]**
- Combine the labeled and unlabeled (after labeling). Again split this combined data into training and validation sets.
- Train the classifiers on the new training set and select a model with optimal validation set accuracy metrics.
- Use the selected classifier to predict labels for another 30000 unlabeled characters.

3. Feature Extraction

The handwritten characters have 3136 pixels and it is not feasible to fit a model directly using these pixels as features. Especially in high dimensionality settings such as this, feature extraction is very important for building models as there is a high chance of overfitting with features as is. We explored two different approaches for feature extraction which are discussed below.

3.1 Principal Components Analysis

Principal Components Analysis is an unsupervised learning and dimensionality reduction technique which allows to summarize the high dimensional data with a small number of representative variables called principal components. The dimension of feature space is reduced in the direction along which the original data is highly variable. Each principal component is a linear combination of p features. PCA is also useful for data visualization as shown in Figure 3. The scatterplot of the first two principal components indicates that the data is non-linearly seperable. However, in a multi-class classfication setting if the classification is performed in an one-vs-one or one-vs-all manner, the decision boundary between few classes can be linear.

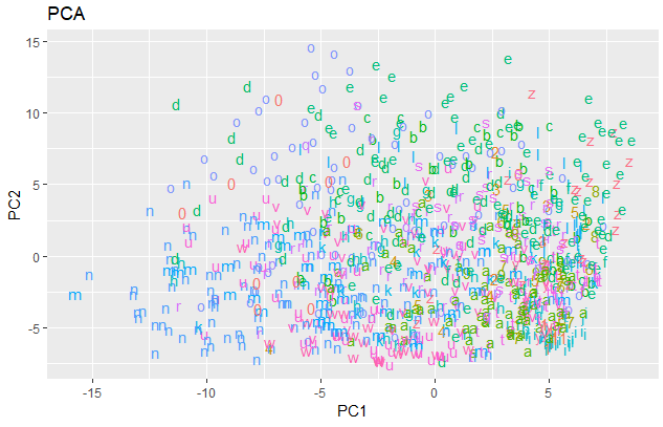

Figure 3: Scatterplot first two Principal Components

The labeled data is split into training and validation sets in 60:40 ratio. We perform PCA on the training to reduce the 3136 pixels into a lower dimension representation and use a selected number of principal components which explains most variance as predictors for the classifier. To select an appropriate number of principal components, the proportion of variance explained (PVE) and Cumulative proportion of variance explained (CPVE) by the principal components can be used which is shown in Figure 4.

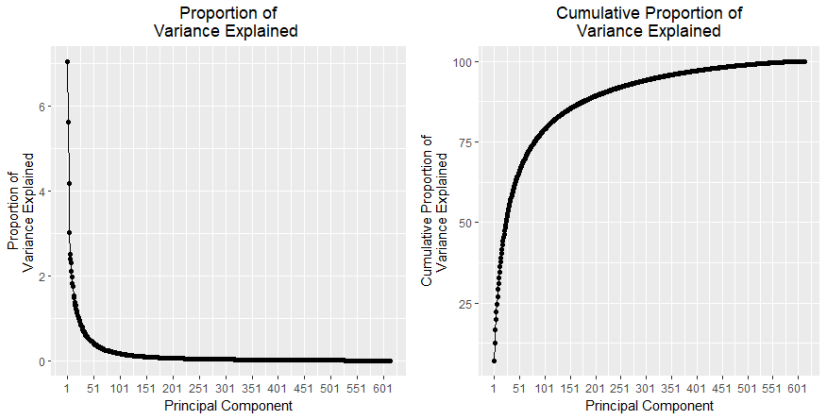

Figure 4: PVE and CPVE by Principal Components on Training Data

We also used 10 - Fold Cross Validation accuracy from different classifiers to determine the optimal number of principal components. As seen from Figure 5, the optimal accuracy is obtained at 60% variability explained which is obtained by 39 principal components. Therefore, the 39 principal components are used as predictors for training the classfiers. The details about building classifiers is explained in detail in section [**5**].

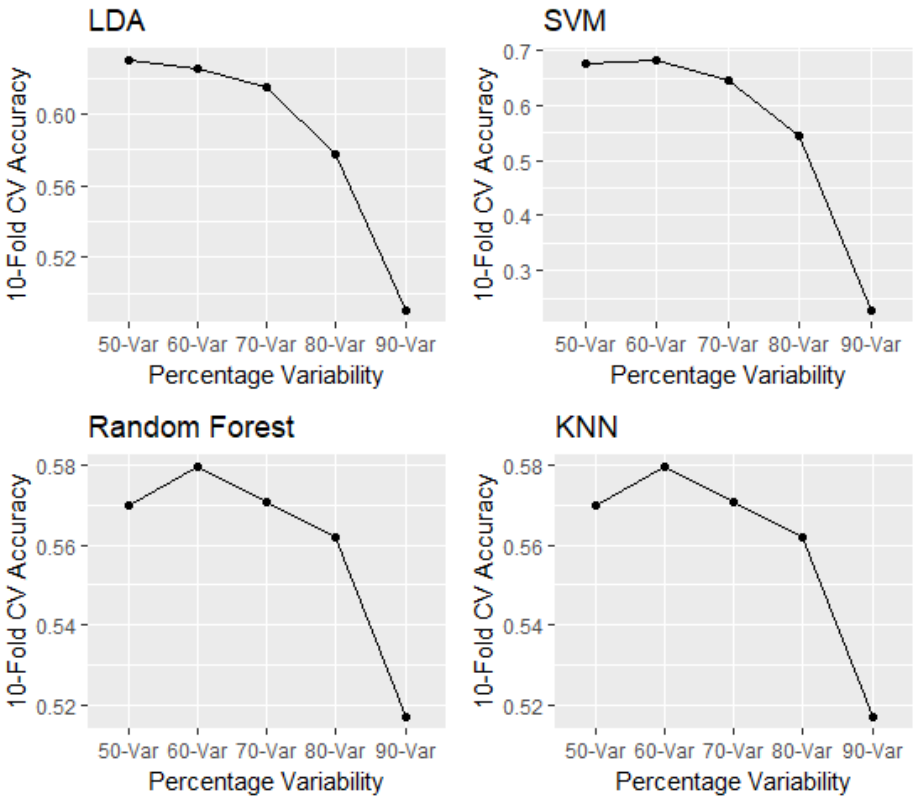

Figure 5: 10 Fold CV for classsifiers across different CVPE

3.2 Diagonal Based Feature Extraction

We also explored another feature extraction method called Diagonal Based Feature extraction [**1**] which was particularly developed for the purpose of handwritten character recognition. This algorithm is implemented with the following steps

- Divide the 56 x 56 pixel image into 49 equal zones where each zone is of size 8 x 8. The choice of this split is made to have equal sized zones.
- Features are extracted by moving along the diagonals of the 8 x 8 matrix of each zone.
- Each zone has 15 diagonal lines and the pixels along each diagonal is summed which form 15 sub-features. These 15 sub-features are averaged to form a single feature and placed in the corresponding zone.
- The above process is repeated for all zones and total of 49 features are extracted for each character. The entire process is shown in Figure 6.

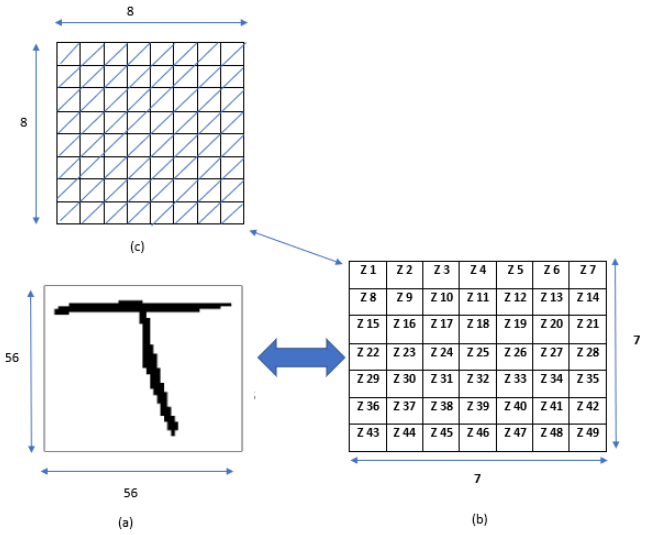

Figure 6: Diagonal Based Feature Extraction

4. Similarity of Characters

In the labeled data, we have 35 alphanumeric characters. And certain characters look similar. Although there are variations within each character due to inherent variation in writing styles of people, there are similar characters such as ‘o’ and ‘0’ , ‘1’ and ‘i’ etc. The following figure shows the mean of several characters.

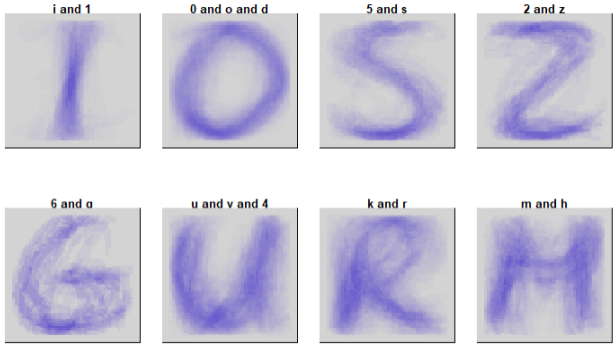

Figure 7: Means of Similar Characters

As seen from Figure 7, there are certain characters whose combined means have been plotted and yet there is no high contrast in them visually. We expect that the classfiers will have difficulty in differentiating these letters. Especially 'o' and '0' , 'i' and '1' have high similarity. A cluster analysis is also performed to analyze what characters are more similar. The diagonal based features for each class are averaged to get 35 rows, each for one unique character. Hierarchical clustering with euclidean dissimilarity measure and complete linkage is performed. The dendrogram obtained is shown in Figure 8. As expected we have certain similar characters grouped together.

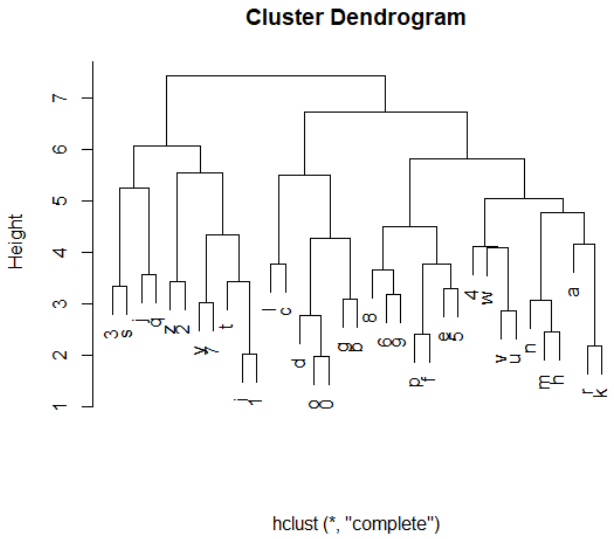

Figure 8: Hierarchical Clustering Using Diagonal Based Features

5. Classification using Labeled Data

There are numerous powerful classification techniques available to perform classfication. Our intention was to see whether linear and non-linear classifiers perform differently on this data. We chose non-linear classifiers such as Random Forests, K - Nearest Neighbours, Support Vector Machine with Non-Linear Kernels. The choice of linear classifiers were LDA and SVM Linear Kernel. These classifiers were trained using both Diagonal Based features and also PCA features with 60-40 split into train and validation sets to see which method was giving better results. The results of each classifier are presented below.

5.1 Support Vector Machine

Support Vector Machine is a supervised classification technique which aims to find the optimal hyperplane which maximizes the margin between classes. For the purpose of training and validation, the labeled data is split into 60 % training data and 40 % validation data. The hyperparameters for model are selected using 10-fold Cross validation. Cost and gamma parameters are tuned for the radial basis kernel. For the polynomial the cost and degree parameters are tuned. SVM model with linear kernel is also fit by tuning the cost. The training and test error obtained from the models with Diagonal Based Features (DBF) and PCA are reported in Table 1. Considering both feature sets, SVM RBF kernel and SVM Linear performed better than SVM Polynomial Kernel.

	Train Error (%)		Test Error (%)	
Method	DBF	PCA	DBF	PCA
SVM Radial Kernel	3.43	14.84	27.39	38.50
SVM Polynomial Kernel	1.46	4.4	32.56	50.9
SVM Linear Kernel	1.46	12.23	31.78	39.27

Table 1: Train and Test Error Rates - SVM Models

5.2 Random Forest

Random Forest is an ensemble of decision trees built on bootstrapped training samples. Instead of considering all the predictors at each split, random forest considers only a random sample of m predictors usually square root of p. The choice of random forest is to see how a tree based method performs and also random forests overcome some disadvantages with bagging. Boosting was not considered as it is computational intensive. We used to 10-fold cross validation to determine the optimal number of predictors considered at each split as 7 for DBF and 2 for PCA features. This parameter is called mtry in the randomForest function. The training and test error rates of Random Forest Model with Diagonal Based Features (DBF) and PCA Features are reported in Table 2. The huge difference in training and test error is indicating that the random forest is overfitting.

	Train Error (%)		Test Error (%)	
Method	DBF	PCA	DBF	PCA
Random Forest	0	0	30.23	52.71

Table 2: Train and Test Error Rates - Random Forest

5.3 K - Nearest Neighbors

KNN is a simple classification approach which identifies the k points in the training data that are closest to a test observation and classifies based on the estimated conditional probability for a class j as the fraction of nearest points/neighbors whose response value equals j. In KNN, the classification can change drastically based on the value of k. We performed 10-fold Cross-Validation on the 60% training data to select an optimal k-value. The optimal value obtained is 3 for DBF and 5 for PCA features.The train and test error rate obtained are shown in Table 3.

	Train Error (%)		Test Error (%)	
Method	DBF	PCA	DBF	PCA
KNN	18.11	27.07	33.85	38.5

Table 3: Train and Test Error Rates - KNN

5.4 Linear Discriminant Analysis

Linear Discriminant Analysis involves the determination of linear combination of variables that best seperates classes. The discriminant function is obtained such that it maximizes the distance between the means of different catergories and minimize the variation within in each category. LDA assumes that each class has a normal distribution with a common covariance matrix for all classes.Gross violations of these assumptions can lead to poor performance. However, we wanted to test the performance of linear classifier. So we chose LDA. The model is fit on both the Diagonal Based Features and PCA Features. The training and test error rates obtained are shown in Table 4.

	Train Error (%)		Test Error (%)	
Method	DBF	PCA	DBF	PCA
LDA	16.15	19.90	36.95	39.79

Table 4: Train and Test Error Rates - LDA

6. Labeling Unlabeled Data

Training a classifier with more data can help improve its performance and generalizing capability. We labeled the unlabeled using a semi-autonomous approach which is as follows.

- Select a classifier with optimal test set accuracy obtained using labeled data.
- Predict the labels for the unlabeled data with the optimal classifier.
- Set up a console system such that if the prediction matches the printed pixel image, click '.' which stores the value of the label in a vector. If the prediction is wrong, directly enter the letter.

Although the above process involved some manual effort, we felt it is better to combine human effort and algorithm intelligence. However, there were certain challenges in labeling the unalabeled data which are discussed below.

6.1 Selecting an optimal classifier

We used two set of features for building the classifier i.e. Diagonal Based Features and PCA Features. For all the classifiers, the Diagonal Based Features gave better results. And the SVM with Radial Basis kernel gave the least test error rate followed by Random Forest, SVM Linear and Polynomial Kernels. We used the SVM RBF Kernel to predict the labels for the unlabeled data.

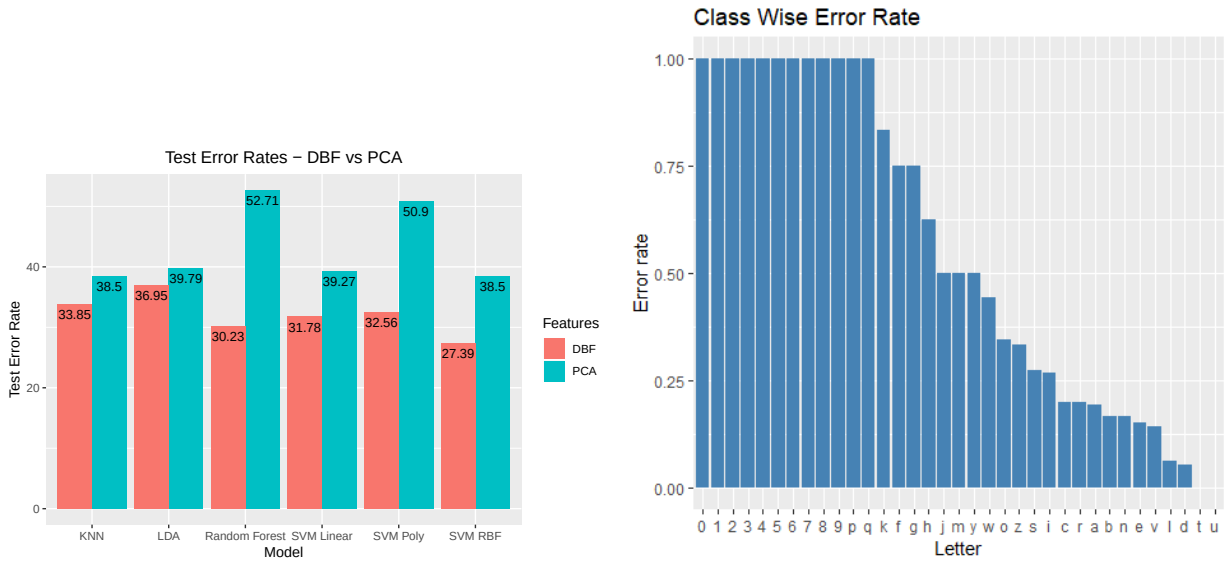

Figure 9: Left: Bar Plot Comparing of PCA and DBF Error Rates, Right: Classwise Error Rate SVM RBF Kernel

As seen from Figure 9 (Right), SVM with RBF Kernel misclassified all the numerics. Therefore, even using this model for predicting unlabeled data without a manual correction can lead to high bias if the unlabeled data is used to train classifiers. We went ahead with this model because of the manual correction done by us using the console system.

6.2 Similar Characters

During the labeling process, the challeging part was to label similar character. The similarity between characters such as 'o' and '0' , 'i' and '1', 'u' and 'v' etc. created confusion. We expect a small percentage of them to have errors.

7. Classification using Labeled and Unlabeled Data

After labeling the unlabeled data, we have a pool of 11000 handwritten character data. For selecting and recommending an optimal classifier, we chose to split the 11000 characters into 90 % Training set and 10 % Validation set. Then train the classifier using the train set and select a classifier with optimal validation set accuracy along with 5-Fold Cross-Validated Accuracy. Since, the Diagonal Based Features gave better results than PCA we only built the classifiers using Diagonal Based Features.

7.1 Support Vector Machine

Support Vector Machine with Radial basis function kernel, Polynomial kernel and Linear kernel are fit. The tuning parameters for radial basis kernel are cost and gamma. Cost value determines the cost of misclassification or violation of margin. As cost value is increased the margin is narrowed. Gamma value is a constant in the radial basis function which controls the influence of the support vectors observations. Large value of gamma restricts the influence only to the support vectors resulting in a narrow gaussian bell. For the polynomial kernel, the degree of the polynomial and cost are the tuning parameters. Cost is the parameter for SVM linear kernel. These parameters are tuned using Cross-validation. The train, test and 5-Fold Cross-validation Errors are reported along with the optimal parameters for each model.

Method	Train Error (%)	Test Error (%)	5- Fold CV Error (%)
SVM RB Kernel c =4,gamma = 0.02	2.56	9.57	11.26
SVM Polynomial Kernel c=8, d=3	2.11	9.84	13.75
SVM Linear Kernel c=0.1	10.19	13.84	15.38

Table 5: Train, Test and 5-Fold CV Error Error Rate- SVM

The test error rate of both non-linear kernels are distant. However, SVM with radial basis kernel performed better. Although the overall error rates are low, taking a look at the classwise error rate can tell which particular characters are being misclassified. Figure 10, shows that the classifiers are misclassifying numerics mostly.

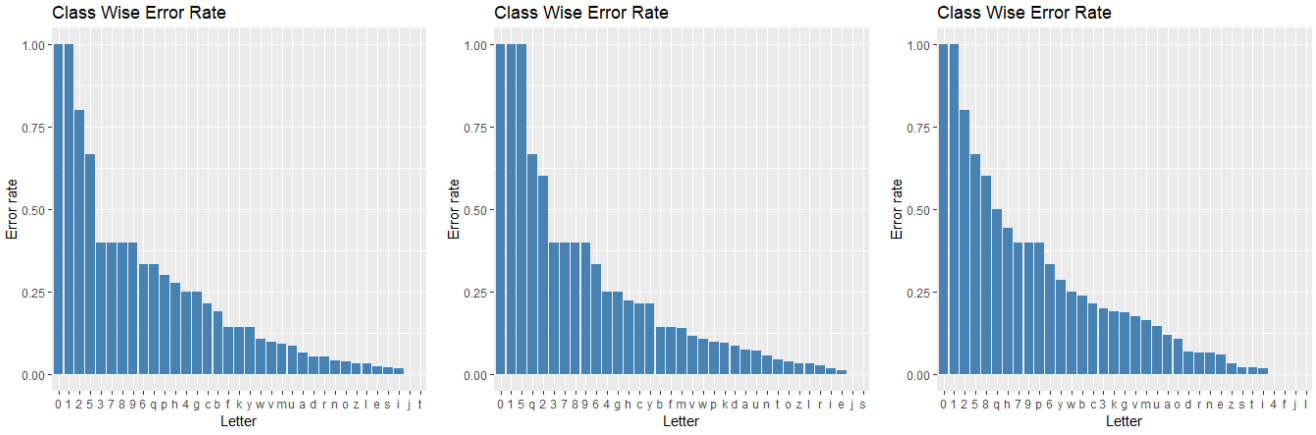

Figure 10: Left : SVM RB Kernel, Middle : SVM Polynomial Kernel, Right : SVM Linear Kernel

7.2 Random Forest

A random forest classifier is fit on the training set. The mtry parameter i.e. the number of predictors considered at each split is determined as 7 using 5-fold cross validation. Two types of error rates are reported for this model. They are the 5-fold Cross validation Error Rate on the training data and the validation set error rate using 10 % observations. The results are reported in the Table 5.

Method	Train Error (%)	Test Error (%)	5- Fold CV Error (%)
Random Forest	0.05	12.88	14.71

Table 6: Train, Test and 5-Fold CV Error Error Rates - Random Forest

The overall test error rate is decent but taking a looking at classwise error rates can help understand which classes are being misclassified most. Figure 12 shows that the classfier is struggling with numerics, especially 0,1,2,5 which were completely misclassified.

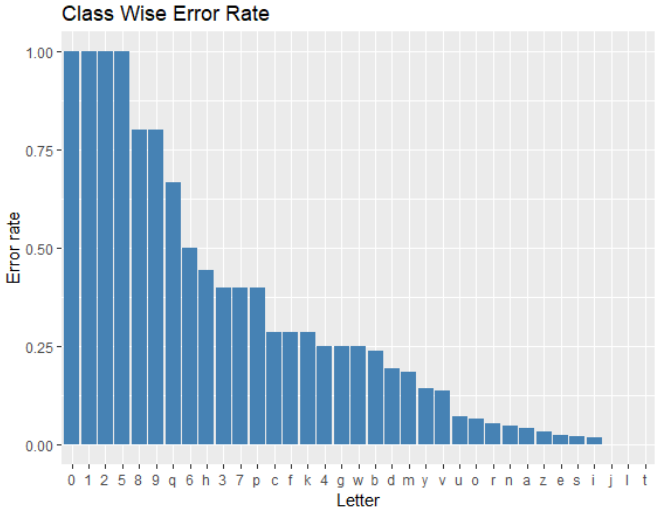

Figure 11: Classwise Error Rate of Random Forest

7.3 K - Nearest Neigbors

The classification done by KNN can change drastically with k. We performed 5-fold CV validation on the training data to select optimal k which is 5. Using k = 5, the KNN is model is fit on the training data. The test error rate and 5-Fold CV error rate which were obtained are reported in Table 7.

Method	Train Error (%)	Test Error (%)	5- Fold CV Error (%)
KNN k =5	10.67	14.81	17.02

Table 7: Train, Test and 5-Fold CV Error Error Rates - KNN

The classwise accuracy is shown in the following figure

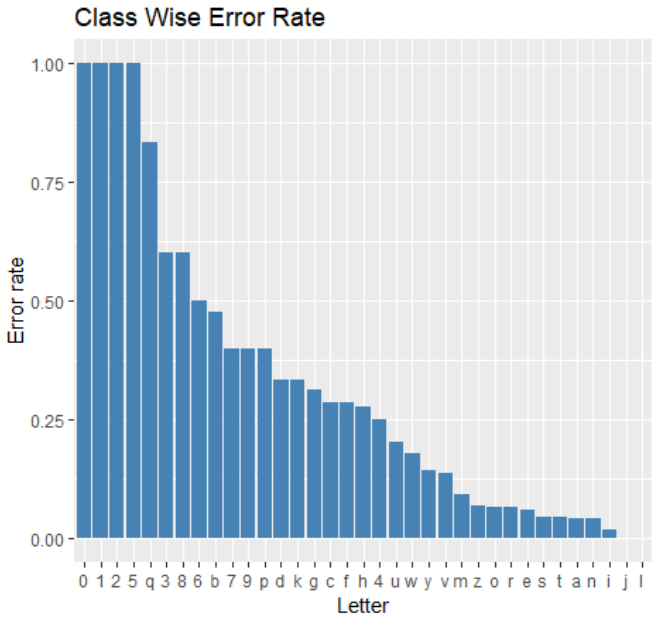

Figure 12: Classwise Error Rate of K Nearest Neighbors

7.4 LDA

We wanted to see how a linear classifier performs in comparison to non-linear classifier. We built a LDA model for this. Table 8 shows the obtained train, test and 5-Fold Cross-Validation Erros. The test error rate and 5-Fold CV error rate is higher than that of the other classifiers we built.

Method	Train Error (%)	Test Error (%)	5- Fold CV Error (%)
LDA	23.66	22.53	25.03

The classwise accuracy is shown in the following figure.

Table 8: Train, Test and 5-Fold CV Error Error Rates - LDA

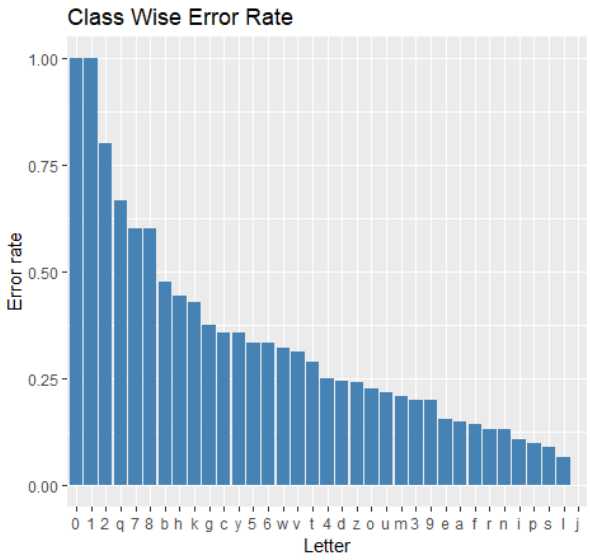

Figure 13: Classwise Error Rate of LDA

8. Recommended Classifier and Conclusion

Accuracy/Error rate of the classifier is the metric we considered to recommend a classifier. Among all the classifiers we built Support Vector Machine with Radial Kernel gave the least test error rate and also 5-fold Cross Validation Error Rate. Therefore we recommend this classifier.

Method	Train Error (%)	Test Error (%)	5- Fold CV Error (%)
SVM RB Kernel c =4,gamma = 0.02	2.56	9.57	11.26
SVM Polynomial Kernel c=8, d=3	2.11	9.84	13.75
SVM Linear Kernel c=0.1	10.19	13.84	15.38
Random Forest	0.05	12.88	14.71
KNN k =5	10.67	14.81	17.02
LDA	23.66	22.53	25.03

Table 9: Train, Test and 5-Fold CV Error Rates of Classifiers

Although the SVM - RBF model is performing well, the classwise error rates (Figure 10) shows that the classifier is not able to accurately classify numerics (0-9). Also taking a look what the top misclassifed labels are being misclassified as can show whether the model is making any egregious error such as misclassifying 'm' as an 'o'. The chord diagram in Figure 14, shows what the most misclassified labels are being classified as. The plot can be interpreted as **Actual Label —> Predicted Label / Percentage of Actual Label being misclassified as Predicted**.

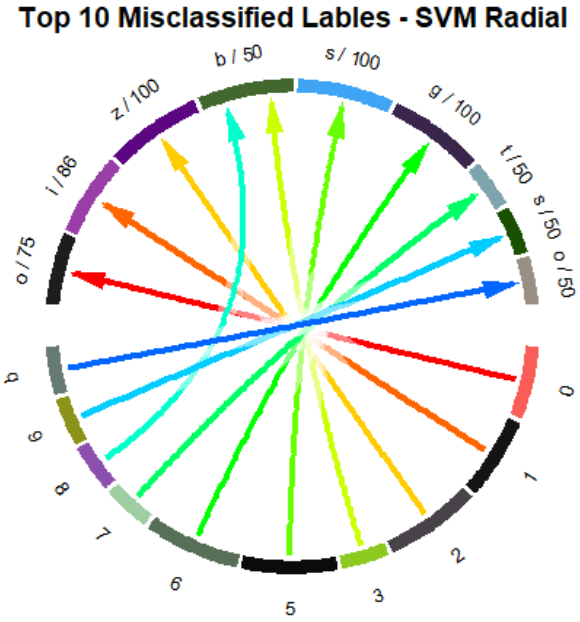

Figure 14: Chord Diagram Showing Actual - Predicted Misclassifications

These misclassified labels are mostly being misclassified with a similar looking label. i.e. '0' as 'o', 1 as 'i' , 2 as 'z'. Not only similarity of characters, but less numeric classes in the dataset can be a cause for less accuracy of numerics. Techniques such as oversampling can be used to deal with this. Also, use of sophisticated techniques such as Convolutional Neural Networks which are particularly efficient in computer vision can be explored. However, we limit the scope of our effort to the classifiers we built and enthusiastic about learning much more.

References

[1] Pradeep, J., Srinivasan, E., & Himavathi, S. (2011). Diagonal based feature extraction for handwritten character recognition system using neural network. 2011 3rd International Conference on Electronics Computer Technology. doi:10.1109/icectech.2011.5941921

[2] James, G., Witten, D., Hastie, T. J., & Tibshirani, R. J. (2017). An introduction to statistical learning: With applications in R. New York: Springer.