IST 707 Applied Machine Learning

HW7: kNN, SVM, and Random Forest for handwriting recognition

In this homework you will use kNN, SVMs, and Random Forest algorithms for handwriting recognition and compare their performance with the naïve Bayes and decision tree models you built for the last assignment. *Use the same data sets used in HW6, digit-train.csv and digit-test.csv.*

**Report structure:**

*Section 1: Introduction*

Briefly describe the classification problem, the general data pre-processing steps, and the

chosen evaluation method and measure(s) (accuracy is enough in this case). Note that some data preprocessing steps maybe specific to a particular algorithm. Report those steps under each algorithm section.

INTRODUCTION:

We are given 2 dataset – Digits.train and Digits.test. Each row of both the data sets (4198 Rows and 785 Columns) contains information about an image of a numeric digit (from 0-9) and the intensity level for each of 785 pixels (785) that form that image. We want to predict the labels in the test dataset after we have trained our model with RNN, SVM and Random Forest. The data sets also contains information about what the digit in the image is in the label column.

I realized it was really time consuming for me to run models in with this dataset, therefore I cut my training dataset and only took first 1000 rows (or images) from my original dataset. Not only that, I also went with normal Cross Validation method for SVM instead of repeated cross validation to save computational time. By doing this I was able to save a lot of time in training models. Before, it took 30-40 mins for my model to run and now it only takes 5-7 minutes.

Steps for Preprocessing in all 3 models:

* Set the appropriate working directory
* Store the csv files of our digit train and digit test dataset
* Cut the data and only take first 1000 rows from train set
* Convert label column to factor type (label contains what kind of digit, that row represents)

*Section 2: kNN*

Build a kNN model. Tune parameters and report test performance.

* For KNN, I used my expand grid from 5 to 25 nearest neighbors with a step of 2 and check which model with certain k worked the best. I used odd number of nearest neighbor in case of conflict when classes will have equal neighbors.
* For KNN I used repeated cross validation with 10 repeats and 3 folds.
* While training our training set, I got 84.9% accuracy.
* With our trained model, I try to predict the labels for test set. I was able to obtained 87.78% accuracy. This means that I was able to predict 87.78% of the labels of our test set correctly with KNN.

*Section 3: SVM*

Build a SVM model. Tune parameters and report test performance.

* For SVM I used linear kernel to conclude my prediction findings. I tried radial kernal but it was taking too much time.
* In linear SVM, I folded my set 3 times with cross validation method. This CV is done without repeat
* For tuning my SVM model I used C parameter to tune my train model which ranged from 0.1 to 2 with 20 evenly spaced values
* After running my analysis SVM gave me 88% accuracy for training set.
* For predicting, I was able to predict 88.33 % of label in our test data.

*Section 4: Random Forest*

Build a random forest model. Tune parameters and report test performance.

* For Random forest I used expand.grid parameter to great range (grid) of values for mtry parameter which is used for tuning.
* I used cross validation (repeatedcv) with 3 folds and 10 repeats.
* I checked top 5 models with highest accuracy and found that my trained model has 87.8% as highest accuracy
* When using this trained model to predict test set, I found 89.88% accuracy with random forest.

*Section 5: Algorithm performance comparison*

Compare the results of the three algorithms. Which performed best? Use your knowledge of the theory behind these algorithms to explain whether the algorithm performance differences make sense or not.

Provide your code in a separate script.

I trained my digits dataset with Naiyes Bayes, Decision tree, KNN, SVM, and random forest. Of all I found random forest had the highest accuracy with 89.88%.

|  |  |  |
| --- | --- | --- |
| Model | Training Accuracy | Prediction Accuracy |
| Naïve Bayes | 42.2% | 42.38% |
| Decision Tree | 59.03% | 60.74% |
| KNN | 84.9% | 87.78% |
| SVM | 88% | 88.33 % |
| Random Forest | 87.8% | 89.88% |

Whereas the lowest accuracy was of Naïve Bayes and it was only able to predict 42.38% of the labels in test set. This might be because although Naïve Bayes is good when working with high dimensional data, it assumes that features are independent of each other which can be problematic in this Dataset. For example, for a image depicting ‘1’ a pixel which contains the ink (black pixel) is definitely related to the pixel which is to its immediate south of it as 1 is mostly a straight line.

Whereas random forest, which combines multiple decision tree to predict the label, is better for high dimensional data. It is also better when our data has outliers. Suppose in our image, there is a digit and a unintentional dot, our RF model would detect and ignore the dot as it is an outlier.

RF is smoother to tune and works well as compared to SVM in high dimensional data. As I tried different variation of SVM -Linear and Radial. I either had to abort training for elongated training time or not use repeated CV in order to save time. As compared to KNN, RF can be a better option if the data is high dimensional. KNN can also be affected by the number of nearest neighbor we chose. Although we used odd neighbors as K ranging from 5-25, It was not enough to surpass the accuracy of random forest model.