AJCSE
ASIAN JOURNAL OF COMPUTER SCIENCE ENGINEERING

www.ajcse.info

Asian Journal of Computer Science Engineering 2016; 2(2):15-18

RESEARCH ARTICLE

Evaluation of Three Classifiers on the Letter Image Recognition Dataset

[1]Ridam Singhal*, [2]Sumit

[1, 2]Information Technology students at Maharaja Agrasen Institute of Technology*

Received on: 29/10/2016, Revised on: 31/12/2016, Accepted on: 15/03/2017

ABSTRACT:

This report presents the Data Mining case study of the Letter Image Recognition Dataset available in UCI Machine learning repository. The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters (26 classes) in the English alphabet. Three different versatile classifiers namely Naïve Bayes, Decision tree C4.5 (J48) and Random Forest were used to mine the data. Data mining open source tool WEKA 3.8.1 is used for the preprocessing and the mining purposes.

INTRODUCTION:

This report presents the Data Mining [2] [6] case study of the Letter Image Recognition Dataset [8] available in UCI Machine learning repository. The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters (26 classes) in the English alphabet. Three different versatile classifiers namely Naïve Bayes, Decision tree C4.5 (J48) and Random Forest were used to mine the data. Data mining [2] [6] open source tool WEKA 3.8.1 [1] is used for the preprocessing and the mining purposes. We present the experimental results in terms of TP Rate, FP Rate, Precision, Recall, F-Measure and ROC Area [7] for each of the classes. Finally, we propose certain new dimensions to improve the efficiency up to a certain extent.

CLASSIFICATION

Classification is a data mining [2], [6] function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.

THE DATASET

The Letter Image Recognition Dataset [8] found in the UCI machine learning repository was donated by David J. Slate of Odesta Corporation; 1890 Maple Ave; Suite 115; Evanston, IL 60201. It was used in [10] that investigated the ability of several variations of Holland-style adaptive classifier systems to learn to correctly guess the letter categories associated with vectors of 16 integer attributes extracted from raster scan images of the letters. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes containing statistical moments and edge counts which can be found in [10] which were then scaled to fit into a range of integer values from 0 through 15.

The detailed description of the various features can be found in [10] and not included in here.

TOOLS USED

The tool chosen for implementation of algorithms was Weka 3.8.1 [1]. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. WEKA is open source software issued under the GNU General Public License. WEKA is helpful in learning the basic concepts of Data Mining [2] [6] where we can apply different options and analyze the output that is being produced. Details of WEKA can be found in [1].

OVERVIEW OF CLASSIFIERS

Naïve Bayes Classifier

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common

***Corresponding Author:** Ridham Singhal**, Email**: ridamsinghal11@gmail.com

principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable.

For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector {\displaystyle \mathbf {x} =(x_{1},\dots ,x_{n})}representing some n features (independent variables), it assigns to this instance probabilities for each of k possible outcomes or classes {\displaystyle C_{k}}

$$p(C_k \mid x_1, \ldots, x_n)$$

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.
Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

Constructing a classifier from the probability model
The Naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the maximum a posteriori or MAP decision

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

rule. The corresponding classifier, a Bayes classifier, is the function that assigns a class label for some k is as follows:

Decision Tree C4.5 Classifier
C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan [3] [4] C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier.

Algorithm
C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy. The training data is a set S = s1, s2… {\displaystyle S= {s_ {1}, s_ {2}...}} of already classified samples. Each sample is {\displaystyle s_{i}}consists of a p-dimensional vector (x1,i,x2,i,…,xp,i){\displaystyle(x_{1,i},x_ {2,i},...,x_{p,i})}, where the xj {\displaystyle x_{j}}represent attribute values or features of the sample, as well as the class in which is {\displaystyle s_{i}} falls.
At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain (difference in entropy). The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sub lists.
This algorithm has a few base cases.

- All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.
- None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.
- Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

Pseudo code
In pseudo code, the general algorithm for building decision trees is:

1. Check for the above base cases.
2. For each attribute a, find the normalized information gain ratio from splitting on a.
3. Let a_best be the attribute with the highest normalized information gain.
4. Create a decision node that splits on a_best.
5. Recur on the sub lists obtained by splitting on a_best, and add those nodes as children of node.

Decision Forest Classifier
Random forest (or random forests) is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees. The algorithm for

inducing a random forest was developed by Leo Breiman [5] and Adele Cutler, and "Random Forests" is their trademark.

Each tree is constructed using the following algorithm:

Let the number of training cases be N, and the number of variables in the classifier be M.

We are told the number m of input variables to be used to determine the decision at a node of the tree; m should be much less than M.

Choose a training set for this tree by choosing n times with replacement from all N available training cases (i.e. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.

For each node of the tree, randomly choose m variables on which to base the decision at that node. Calculate the best split based on these m variables in the training set.

Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).

For prediction, a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction.

MINING RESULTS

After we tested several classifiers and parameters on the training and testing data, we found the following three classifiers attracting our attention. We present in this section the various output models and the results that we got after these three classifiers were trained and tested. We give a few evaluation parameters like the precision, recall, errors. We also analyzed the graph based evaluation with ROC (Receiver Operating Characteristics) curves [7] and the confusion matrices [9] which are not included here due to space shortage.

```
=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 17.96 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       19278               96.3948 %
Incorrectly Classified Instances       721                3.6052 %
Kappa statistic                          0.9625
Mean absolute error                      0.0131
Root mean squared error                  0.0623
Relative absolute error                 17.6643 %
Root relative squared error             32.4171 %
Total Number of Instances            19999
```

Naïve Bayes Classifier

So, we can see from the results that the classifier is not up to the mark of a good algorithm to use in this case. We move forward with the other classifiers.

Decision Tree C4.5

The J48 operator was used to model the decision tree for the training set. Results are shown in Figure. So, we can see that the decision tree is giving much better results than the Naïve Bayes classifier.

```
Number of Leaves  :     1226

Size of the tree :      2451


Time taken to build model: 4.86 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       17594               87.9744 %
Incorrectly Classified Instances      2405               12.0256 %
Kappa statistic                          0.8749
Mean absolute error                      0.0106
Root mean squared error                  0.0903
Relative absolute error                 14.2796 %
Root relative squared error             46.9441 %
Total Number of Instances            19999
```

Random Forest Classifier

Random forest of 10 trees, each constructed while considering 5 random features. The results are shown in Figure

ANALYZING THE RESULTS

The output of the WEKA [1] tool contained the detailed views of prediction, the model, evaluation parameters, ROC curves [7], Confusion matrices [9] and all, some of which we could not accommodate here. By looking into the various result outputs and studying them thoroughly, we came to the following main conclusions.

1. The Decision tree and Random forest classifiers outpaced the famous Naïve Bayes classifier in terms of accuracy up to a large extent. This happened probably because of the assumption of feature independency we did in the case of Naïve Bayes, as some of the attributes here in our dataset were related to each other, violating the assumption.
2. In the random forest, among the 721 wrong predictions, we observed that most of them were similar pairs, like confusion between “Q” and “O”, between “H” and “K”, between “J” and “I” etc. (Confusion pairs). Which are very common even for

human eyes also if the handwriting is not proper. Another interesting fact that we observed was like for the confusion pairs, both of them have similar prediction probability distribution and they are the only two (most cases) letter classes having positive probability, others being zero. That means, if we have an "O", but we predicted it as "Q", then it's likely that probability of the test image being "Q" is only slightly greater than "O", leading to the wrong result. At the same time, probability of the test image being other letters is mostly zero, which defines the accuracy of the classifier.

3. Another small but important observation is the training time. We see that training time is increasing as the efficiency is increasing, which means that more the detailed analysis we do to make the model, more is the classifier accuracy.
4. We also tried testing with some instances from the training data itself; we had accuracy of 64.10% for Naïve Bayes, 87.97% for J48 and 96.39% for Random Forest. It proves that if we ask the classifier to predict something from the training data itself, it can predict with almost 100% accuracy which is a great achievement. This result becomes very useful when we use the algorithm in real time auto learning intelligent systems.

CONCLUSION

We have seen that the task of letter recognition can be easily handled with the help of Data Mining [2] [6] techniques. Among the three classifiers we used, Random Forest gave the best results as 93.8% accuracy on unseen data and 99.98% accuracy on seen data, which is far better than those found in the original paper [10]. Thus, we can think of implementing these algorithms in practical intelligent state-of-the-art systems for better performance.

SOLUTIONS FOR IMPROVING ACCURACY

There are two ways we can improve the accuracy:

1. By introducing new features to the dataset to take into account the more detailed description of the letter images.
2. We can handle the few confusion pairs we found in the random forest results with the help of special learning and testing blocks. For example, if we know that we have a confusion in prediction between "C" and "G" (which we can easily find out from probability distributions), then we can use the 6th and 8th attribute of the corresponding classes to remove the ambiguity, because those features say about some geometrical dimensions which are more relevant for "C" and "G" only. This task is may be assumed as a temporary reduction of the feature set to reduce the effect of unwanted features.

ACKNOWLEDGMENT

We express our deep gratitude to Dr. M.L. Sharma, Head of Department, Department of Information Technology, and Maharaja Agrasen Institute of Technology for his valuable guidance and suggestion throughout the research work.

I am also thankful to M.L. Goyal, Director General, Maharaja Agrasen Institute of Technology for providing us the facilities to carry out the research work.

REFERENCES

1. WekaManual-3-6-5V Data Mining: Concepts and techniques by Jiawei Han and Micheline Kamber.
2. J. R. Quinlan, C4.5: Programs for machine learning, Morgan Kaufmann, San Mateo, CA, 1993.
3. J. R. Quinlan, Improved use of continuous attributes in C4.5," Journal of Artificial Intelligence Research, Vol. 4, 1996, pp. 77-90.
4. L. Breiman. Random forests. Machine Learning, 45(1): 5-32, 2001.
5. Data Mining Techniques by Arun Pujari.
6. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers by Tom Fawcett
7. http://archive.ics.uci.edu/ml/datasets.html.
8. http://www2.cs.uregina.ca/~hamilton/courses/831/notes/confusion_matrix/confusion_martix.html.
9. P. W. Frey and D. J. Slate Machine Learning Vol. 6 #2 March 91): "Letter Recognition Using Holland-style Adaptive Classifiers".
10. https://en.wikipedia.org/wiki/Naive_Bayes_classifier.
11. https://en.wikipedia.org/wiki/C4.5_algorithm.