Report on

Evaluation of three classifiers on the Letter Image Recognition Dataset and analysis of the results along with new solutions for the improvement of the classification accuracy

(Data Mining term project under Mrs. Ujwala Baruah, Asst. Professor, CSE, NIT Silchar)

Submitted by:
Himangshu Ranjan Borah, 7th Semester, CSE, NIT Silchar (08-15-013)
Himkalyan Bordoloi, 7th Semester, CSE, NIT Silchar (08-15-006)
Raj Nandan Sharma, 7th Semester, CSE, NIT Silchar (08-15-028)

Abstract

This report presents the Data Mining 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 open source tool WEKA 3.6.5 and MATLAB were 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 for each of the classes. Finally we propose certain new dimensions to improve the efficiency up to a certain extent.

Introduction

Data mining[2][6], a relatively young and interdisciplinary field of computer science is the process of discovering new patterns from large data sets involving methods at the intersection of artificial intelligence, statistics and database systems. We can train the machine by mining into a training dataset and by extracting some rules for the output prediction of the test samples. Data mining consists of basically five kinds of tasks namely Anomaly detection, Association rule learning, clustering, Classification and regression. Among the above tasks, clustering and classification and regression plays a great role in the machine learning models.

Classification: Our field of study

Classification is a data mining 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 and the aim of the Mining

The Letter Image Recognition Dataset found in the UCI machine learning repository [8] 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 (namely \texttt{lettr} (Class label); \texttt{x-box} (horizontal position); \texttt{y-box} (vertical position); \texttt{width} (width of box); \texttt{high} (height of box); \texttt{onpix} (total # of on pixels); \texttt{x-bar} (mean x of onpix); \texttt{y-bar} (mean y of onpix); \texttt{x2bar} (mean x var); \texttt{y2bar} (mean y var); \texttt{xybar} (mean xy correlation); \texttt{x2ybr} (mean of x*x*y); \texttt{xy2br} (mean of x*y*y); \texttt{x-ege} (mean edge count); \texttt{xegvy} (correlation of x-ege with y); \texttt{y-ege} (mean edge count bottom to top); \texttt{yegvx} (correlation of y-ege with x)) which were then scaled to fit into a range of integer values from 0 through 15. Two rows from the data set are shown below…

T,2,8,3,5,1,8,13,0,6,6,10,8,0,8,0,8  
I,5,12,3,7,2,10,5,5,4,13,3,9,2,8,4,10

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

**Prior Work on the Data Set:**

The dataset was previously used in [10] for an experiment on letter image recognition. The research for this article 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 best accuracy obtained was a little over 80%. It would be interesting to see how well other methods do with the same data.

**Data Preprocessing:**

1. **Data cleaning**

Since the dataset downloaded is almost corrected already, it doesn't contain any noisy data or inconsistent values. Hence the only treatment we took was for the missing data which might be present for some attributes. We used the attribute mean to fill the missing values, implemented in MATLAB.

2. **Data Transformation**

The tools we used for the mining process accepts ARFF (Attribute-Relation File Format) files only [1]. Since the downloaded data was in .data format, so we had to convert it. An ARFF file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files have two distinct sections, the first section is the **Header** information, which is followed the **Data** information.

3. **Dividing into training and testing set ::**

   We analyzed the raw data and found that the classes were evenly distributed throughout the dataset, which contains 20,000 data samples. That’s why we selected the first 18,000 samples as the training set and rest 2000 as the testing set for evaluating the mining process.

**Tools Used**
The tools chosen for implementation of algorithms were **Weka 3.6.5** and **MATLAB**. 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 where we can apply different options and analyze the output that is being produced. Details of WEKA can be found in [1].

**Overview of the used classifiers:**

**Naive Bayes classifier:**

A Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' Theorem with strong (naive) independence assumptions [2]. If each data sample is represented by an n-dimensional feature vector, \( X = (x_1, x_2, \ldots, x_n) \), depicting the values of \( n \) attributes, \( A_1, A_2, \ldots, A_n \) and there are \( m \) classes, \( C_1, C_2, \ldots, C_m \), given an unknown data sample, \( X \), the classifier will predict that \( X \) belongs to the class having the highest posterior probability, conditioned on \( X \).

\[
P(C_i|X) > P(C_j|X) \text{ for } 1 \leq j \leq m; j \neq i
\]

i.e. we need to maximize

\[
P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}
\]

We assume that the values of the attributes are independent (That’s why the name). Thus

\[
P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)
\]

where \( P(x_k|C_i) = \frac{s_{ik}}{s_i} \), where \( s_{ik} \) is the number of training samples of class \( C_i \) having the value \( x_k \) for \( A_k \), and \( s_i \) is the number of training samples belonging to \( C_i \).

If \( A_k \) is continuous valued, then the attribute is typically assumed to have a Gaussian or any other suitable probability distribution.

**Decision tree C 4.5 (Implemented as J48 in WEKA)**

**C4.5** [QUI92] 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 for classification. 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 \( S = s_1, s_2, \ldots \) of already classified samples. Each sample \( s_i = x_1, x_2, \ldots \) is a vector where \( x_1, x_2, \ldots \) represent attributes or features of the sample. The training data is augmented with a vector \( C = c_1, c_2, \ldots \) where \( c_1, c_2, \ldots \) represent the class to which each sample belongs. At each node of the tree, C4.5 chooses one attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. Its criterion is the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. The C4.5 algorithm then recurses on the smaller sublists. **J48** is an open source Java implementation of the C4.5 algorithm in the WEKA data mining tool.

**The Random Forest**

**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:

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

2. 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 $M$.

3. 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.

4. 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.

5. 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.

**Data modeling and 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 which are not included here due to space shortage.

**Naïve Bayes Classifier:**

First classifier we selected was the famous Naïve Bayes classifier. We provided training and testing set separately. The training took 0.3 seconds. After the classifier was tested on the test set, the following results were produced.

- Correctly Classified Instances: 1228 (61.4%)
- Incorrectly Classified Instances: 772 (38.6%)
- Kappa statistic: 0.5985
- Mean absolute error: 0.034
- Root mean squared error: 0.144

The confusion matrix or ROC curves [9] for the results are impracticable to include here, so we just provide the weighted average of the various evaluation parameters for all the 26 classes that corresponds to the alphabets.

<table>
<thead>
<tr>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>ROC Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.614</td>
<td>0.015</td>
<td>0.64</td>
<td>0.614</td>
<td>0.613</td>
<td>0.951</td>
</tr>
</tbody>
</table>
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 Induction C4.5 (Implemented in WEKA as J48 operator):**

The J48 operator was used to model the decision tree for the training set. The training time was 4.35 seconds. Results are as follows…

| Correctly Classified Instances | 1742       | 87.1 % |
| Incorrectly Classified Instances | 258       | 12.9 % |

Kappa statistic: 0.8658
Mean absolute error: 0.011
Root mean squared error: 0.0936

Evaluation averages are…

<table>
<thead>
<tr>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>ROC Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.871</td>
<td>0.005</td>
<td>0.875</td>
<td>0.871</td>
<td>0.872</td>
<td>0.95</td>
</tr>
</tbody>
</table>

So we can see that the decision tree is giving much better results than the Naïve Bayes classifier. We will discuss the various factors that might be affecting the improved performance later on.

**Random Forest classifier:**

Training time was 5.58 seconds. Random forest of 10 trees, each constructed while considering 5 random features. Out of bag error: 0.1752. The results are as follows…

| Correctly Classified Instances | 1877       | 93.85 % |
| Incorrectly Classified Instances | 123      | 6.15 % |

Kappa statistic: 0.936
Mean absolute error: 0.0129
Root mean squared error: 0.0683

Evaluation parameters are…

<table>
<thead>
<tr>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>ROC Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.939</td>
<td>0.002</td>
<td>0.941</td>
<td>0.939</td>
<td>0.939</td>
<td>0.995</td>
</tr>
</tbody>
</table>

We see that using Random Forest gives a complete different level of prediction accuracy. Only 123 instances out of 2000 testing samples were predicted wrongly. This result seems to be quite acceptable.

**Analysis of results:**

This section is the heart of the whole Data Mining process. The output of the WEKA tool contained the detailed views of prediction, the model, evaluation parameters, ROC curves, Confusion matrices 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 123 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 64.65% for Naïve Bayes, 96.24% for J48 and 99.98% 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.

Solutions for improvement of 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 says about some geometrical dimensions which are more relevant for “C” and “G” (description of the features in [10]) only. This task is may be assumed as a temporary reduction of the feature set to reduce the effect of unwanted features.

Conclusions:

We have seen that the task of letter recognition can be easily handled with the help of data mining 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.
References:


[2] Data Mining: Concepts and techniques by Jiawei Han and Micheline Kamber.


[6] Data Mining Techniques by Arun Pujari

[7] ROC Graphs: Notes and Practical Considerations for Data Mining Researchers by Tom Fawcett


    "Letter Recognition Using Holland-style Adaptive Classifiers".


[12] www.google.com