Hybrid Learning Scheme for Data Mining Applications

T.Ravindra Babu, M.Narasimha Murty
Department of CSA, Indian Institute of Science
Bangalore-560012
trbabu,mnm@csa.iisc.ernet.in

V.K.Agrawal
ISRO Satellite Centre
Bangalore-560017
agrawal@isac.ernet.in

Abstract

Classification of large datasets is a challenging task in Data Mining. In the current work, we propose a novel method that compresses the data and classifies the test data directly in its compressed form. The work forms a hybrid learning approach integrating the activities of data abstraction, frequent item generation, compression, classification and use of rough sets.

1. Introduction

In Data Mining, scalability, high dimensionality, speed and predictive accuracy are important issues. In the current large scale pattern classification application, we encounter large data sets in the form of training and test datasets [3]. Each of these datasets forms a pattern matrix [6]. The pattern matrix can be viewed as a transaction database. Each pattern in the data under consideration consists of binary features. Thus each pattern forms a transaction with a set of items.

Data compression has been one of the enabling technologies for multimedia communication in the recent years [9, 7]. Based on the requirements of reconstruction, data compression is divided into two broad classes, viz., lossless compression and lossy compression schemes. For optimal performance, nature of data and the context influence the choice of the compression technique. One of the objectives of current work is to compress the data and then classify. Thus, it is necessary that the compressed form of the training and test data should be tractable and should require minimum space and time.

We use two concepts of association rule mining viz., support and frequent item set [1, 5]. Use of frequent items that are above the given support will avoid less frequent input features and provide better abstraction.

In the current paper, we propose an algorithm that summarizes the given data using a *single scan*, in the form of distinct subsequences. Less frequent subsequences are further pruned by more frequent, nearest neighbours. This leads to a *compact* or *compressed* representation of data. The test data is classified in its *compressed form* without sacrificing the classification accuracy. The scheme results in a *significant compression* of the input data. The *classification of data directly in the compressed form* provides an additional advantage. The compression leads to highly accurate classification of test data because of improvement in generalization.

Rough classification [4, 8] based on dissimilarity limit of the test patterns is carried out. The data summary thus generated requires significantly less storage as compared to rough set based schemes with similar classification accuracy [2].

2. Methodology

Consider a training data set consisting of 'm' patterns. Let each pattern consist of 'p' binary features. Let 'ϵ' be minimum support [1, 5] for any feature of a pattern to be considered for the study.

Definition 1 ***Sequence***. *A sequence of integer numbers $\{s_1, s_2..\}$ is a function from **I** to $I^{'}$ where **I** and $I^{'}$ are two sets of positive integers.*

Definition 2 ***Subsequence***. *A subsequence is a subset of **sequence**.*

Definition 3 ***Length of a subsequence***. *Number of elements of a subsequence is the length of a subsequence.*

Definition 4 ***Block, Block Length***. *A finite number of binary digits forms a block. Number of bits in a block is called the **block length**.*

Definition 5 ***Value of a Block***. *Decimal equivalent of a block is the value of block*

Table-1 provides the list of parameters used in the current work. In the current implementation, all the parameters are integers.

Proceedings of the Fourth International Conference on Hybrid Intelligent Systems (HIS'04)
0-7695-2291-2/04 $ 20.00 IEEE

Parameter	Description
m	Number of patterns or transactions
p	Number of features or items prior to identification of frequent items
n	Number of binary features that makes one block
q	Number of blocks in a pattern
r	Length of a subsequence
ϵ	Minimum support
ψ	Minimum frequency for pruning a subsequence
η	Dissimilarity threshold for identifying nearest neighbour to a subsequence

Table 1. List of parameters used

With this background of parameter definition, the entire method can be summarized in the following steps.

1. Initialization
2. Frequent Item generation
3. Generation of coded training data
4. Subsequence identification and frequency generation
5. Pruning of subsequences
6. Generation of coded test data
7. Classification using distance based Rough concept
8. Classification using *k*-NNC

We elaborate each of the above steps in the following subsections.

2.1. Initialization

With the given data, number of training patterns, *m* and number of features, *p* are known. Based on a priori domain knowledge of the data as well as through preliminary analysis on the training data, the parameters of (a) minimum support, ϵ for frequent item generation, (b) block length, *n*, (c) minimum frequency for subsequence pruning, ψ and (d) dissimilarity threshold for identifying nearest neighbours to the pruned subsequences, η are initialized to non-zero values.

2.2. Frequent Item Generation

The input data encountered in practice is noisy and contain features that are not frequent. Also the number of effective features differs from pattern to pattern. While generating an abstraction of entire data, it is necessary to smooth out the noisy behaviour, which otherwise may lead to improper abstraction. This can be visualized by considering data such as handwritten digit data or a sales transaction data. Thus features of support above 'ϵ' are only considered for the study. The support of each feature across the training data is computed. The items whose support is above ϵ are considered for the study. It should be noted here that the feature dimension, *p* is kept unchanged, even though a feature gets eliminated across all the patterns with the chosen ϵ.

2.3. Generation of Coded Training Data

At this stage, the training data consists only of frequent items. Considering n-binary items at a time, decimal value is computed. The value of *n* is a result of preliminary analysis. Thus, *n*-bit block values are generated. The *p* binary features of each pattern are now represented as *q* decimal values, where *p*=*q***n*.

2.4. Subsequence Identification and Frequency Computation

The sequence of decimal values are further compacted as subsequences of decimal values. In a large data generated by a common process (say, sales transaction), there would be a good amount of similarity among the different patterns. The length of subsequence is a result of preliminary analysis of the training data. It is essentially a trade-off between minimum representativeness of the data and maximum compactness of representation.

A set of *r* decimal values is considered as a subsequence. The frequency of each unique subsequence is computed. The number of distinct subsequences depend on ϵ, also.

2.5. Pruning of Subsequences

Among the large set of distinct subsequences, say, m_1, the least frequent subsequences are identified. The least repeating subsequences indicate that they may not significantly contribute to classification activity; on the other hand, they would add to computational burden.

Depending on the value of ψ, all such subsequences, whose frequency is less than ψ, are identified. Each of such subsequences is replaced by its nearest neighbour to those that remained within a pre-chosen dissimilarity threshold, η. The value of η is chosen to be as as small as possible for meaningful generalization.

It should be noted here that by discarding low-frequent subsequences, we actually improve the classification activity, as the generalization improves. Further, we operate with a least number of distinct subsequences.

The number of distinct subsequences that remain after this step depend on the value of ψ. All the remaining distinct subsequences are numbered, say, from 1 to m_2. It should be noted that $m_2 \leq m_1$.

At the end of this step, the training data consists of just m_2 unique id's, numbered from 1 to m_2.

2.6. Generation of Coded Test Data

The test data is an independent set from that of training data. Proceeding on the similar lines as above, *n*-bit decimal codes are generated for the test data. However, ϵ is not made use of. With pruned, ordered subsequences of the training data as input, those distinct sequences are in the test data are identified.

However, it is likely that, (a) many of the subsequences in the test data are not present in the ordered subsequences and (b) some of the previously discarded subsequences could be available in test data. In such a case, each new subsequence found in the test pattern is replaced by its neighbour in the set of m_2 subsequences generated using training set.

2.7. Classification using dissimilarity based Rough concept

Rough set theory is used here for classification. A given class A is approximated, using rough set terminology [4, 8], by two sets, viz., a *lower approximation* of A and an *upper approximation* of A. The lower approximation of A consists of all of the samples that are certain to belong to A, unambiguously. The upper approximation of A consists of all samples that cannot be described as not belonging to A.

In the current work, the decision rule is chosen based on dissimilarity threshold.

The upper approximation of A consists of all those training patterns which are neighbours by means of ordered distance without a limit on dissimilarity. The lower approximation of A consists of set of training patterns that are below the chosen dissimilarity threshold. As the patterns falling within lower approximation are classified unambiguously, the patterns falling between lower and upper approximation are chosen to be rejected as unclassifiable.

The following section provides dissimilarity computation procedure between compressed patterns. Same procedure is used in the entire work, whenever dissimilarity was required, such as, pruning, nearest neighbour computation, classification, etc.

2.7.1. Dissimilarity Computation between compressed patterns Dissimilarity computation between compressed patterns, in both the classification schemes, is based on the unique id's of the subsequences.

Each subsequence consists of known decimal codes. With a chosen block length, the range of decimal codes is known. Thus, storing an upper triangular matrix containing distances between any two decimal codes would be sufficient for dissimilarity computation. For example, for 4-bit blocks, the range of decimal codes is 0 to 15, corresponding upper triangular distance matrix would consist of (16*17)/2=136 values.

Since the data is inherently binary, Hamming distance is used for dissimilarity computation.

2.8. Classification using *k*-Nearest Neighbour Classifier

In this approach, each of the compressed test patterns, consisting of pruned subsequences is considered and dissimilarity is computed with all training patterns. The first *k*-neighbours are identified depending on the dissimilarity value. Based on majority voting, the test pattern is assigned the class label. The classification accuracy depends on the value of k.

3. Preliminary Data Analysis

The proposed method is implemented on a set of large Handwritten digit data. The current section provides a brief description of the data and preliminary analysis carried out.

The Handwritten data that is under consideration consists of 192 binary featured patterns. In all there are 10003 patterns, which are divided into 6670 training patterns and 3333 test patterns. The data consists of 10-class data with class labels 0 to 9.

Each of the training and test data sets is subdivided into almost equal number of patterns per class. Each of the 192 featured labeled patterns is represented as a readable pattern by means of 16X12 matrix. Figure 1 provides a set of typical and atypical patterns of the given training data.

Each pattern is characterized by 192 binary features. Block lengths of 3-bits and 4-bits are considered for Huffman coding of the input training data.

The 4-bit blocks result in 16 distinct codes, viz., 0000 to 1111. The 3-bit blocks result in 8 codes. The 4-bit block codes and 3-bit block codes respectively result in 48 and 64 sequences of values for each 192 featured pattern.

With Huffman coding for 4-bit blocks, the original data of size 2574620 bits is compressed to 1910142 bits, leading to 25% saving in space.

4. Implementation of the Proposed Scheme

The current section discusses each step of the proposed scheme in the context of considered data.

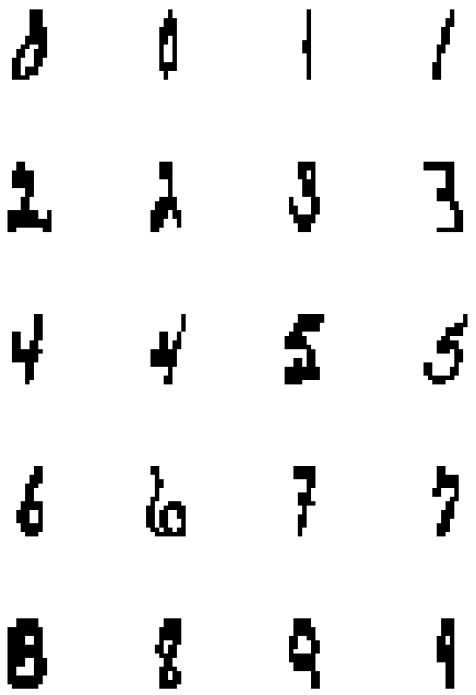

Figure 1. A set of typical and atypical patterns of Handwritten data

4.1. Choice of parameters

Values of minimum support for frequent item generation, ϵ, minimum frequency for pruning of subsequences, ψ are dealt as experimental parameters with initial values as 1. After initial experimentation the value of maximum dissimilarity limit, η, is chosen to be within a value of 3. All the experiments are carried out on the training data.

For the number of bits for blocking, n, two values of 3 and 4 are considered. The 4-bit block values of two typical training patterns of classes 0 and 1 respectively are provided in Table-2. There are 48 decimal equivalent codes(block values) for each pattern.

Table-3 contains a sample of 3-bit coded training data. There are 64 block values for each pattern.

The proposed scheme makes use of the subsequences of blocked values, which is explained in the following section. In the paper, after elaborate exercises, 4-bit blocks are chosen. However, a brief mention about 3-bit blocks is made wherever necessary in order to bring home subtlety of the concepts.

Label	Data
0	0 3 8 0 3 8 0 3 8 0 3 8 0 3 12 0 3 12 0 7 12 0 7 12 0 12 12 1 9 12 1 9 12 3 1 8 3 1 8 3 7 0 3 7 0 3 12 0
1	0 2 0 0 2 0 0 2 0 0 2 0 0 2 0 0 2 0 0 2 0 0 6 0 0 6 0 0 6 0 0 2 0 0 2 0 0 2 0 0 2 0 0 2 0 0 2 0 0

Table 2. Sample of 4-bit Coded Training Data

Label	Data
0	0 0 7 0 0 0 7 0 0 0 7 0 0 0 7 0 0 0 7 4 0 0 7 4 0 1 7 4 0 1 7 4 0 3 1 4 0 6 3 4 0 6 3 4 1 4 3 0 1 4 3 0 1 5 6 0 1 5 6 0 1 7 0 0
1	0 0 4 0 0 0 4 0 0 0 4 0 0 0 4 0 0 0 4 0 0 0 4 0 0 0 4 0 0 4 0 0 0 4 0 0 1 4 0 0 1 4 0 0 1 4 0 0 0 4 0 0 0 4 0 0 4 0 0 0 4 0 0 0 4 0 0 0 4 0 0 0 4 0 0 4 0

Table 3. Sample of 3-bit Coded Training Data

4.2. Frequent Items and subsequences

The minimum support value, ψ is changed starting from 1. Increasing this value results in less number of distinct subsequences. For example, from Table-2 and Table-3, observe repeating subsequences. In Table-2, first pattern contains following subsequences of length 3 (r=3), viz., (0,3,8), (0,3,12), (0,7,12), (0,12,12), (1,9,12), (3,1,8), (3,7,0), (3,12,0). In Table-3, observe subsequences of length 4(r=4), viz., (0,0,7,0), (0,0,7,4), (0,1,7,4), (0,3,1,4), (0,6,3,4,), (1,4,3,0), (1,5,6,0), (1,7,0,0).

The choice of minimum support value, ψ influences number of distinct subsequences. Figure 2 depicts reduction in number of distinct subsequences with increasing support value, ϵ.

It should be observed from the figure that number of distinct subsequences reduced from 690 at input support value of 1 to 395 with an input support value of 100.

4.3. Compressed Data and Pruning of Subsequences

The distinct subsequences are numbered continuously. This forms compressed data. Table-4 contains typical compressed training data.

The subsequences are pruned further by discarding infrequent subsequences. This is carried out by choosing the value of ψ. Increase in ψ reduces number of distinct subsequences.

Figure 3 contains distinct subsequences and the corresponding frequency.

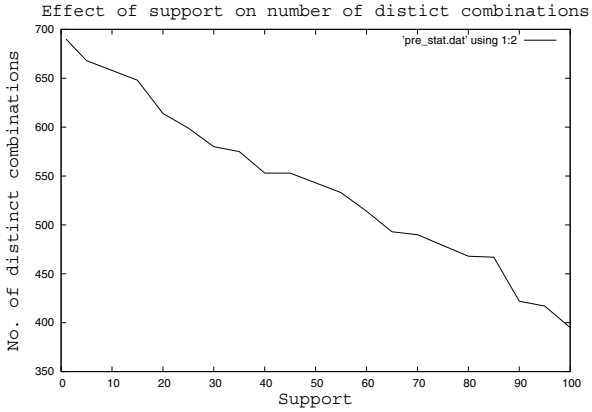

Figure 2. Distinct subsequences as a function of support value(ϵ)

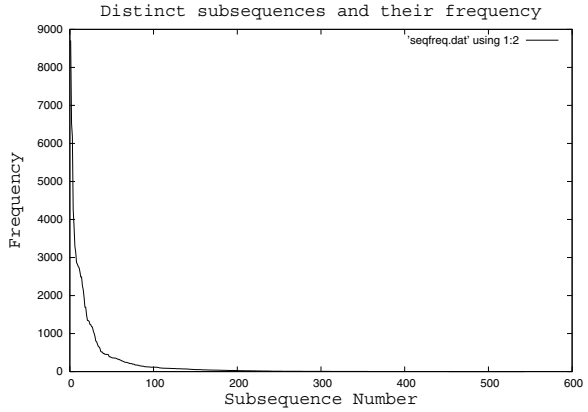

Figure 3. Distinct subsequences and their frequency

Label	Compressed Data
0	23 23 23 23 51 51 39 39 81 96 96 25 25 44 44 30
1	17 17 17 17 17 17 17 30 30 30 17 17 17 17 17 17
2	30 235 235 185 185 18 18 202 168 168 195 195 250 250 1
3	6 6 65 65 12 12 12 12 58 58 25 25 5 5 7 7

Table 4. Typical Compressed Training Data

Figure 4 consists of effect of frequency limit for pruning after their nearest neighbour(NN) assignments(replacements). The data is generated for a specific value of ϵ, viz., 50.

After reducing the number of distinct subsequences subject to ψ, the subsequences are re-numbered. For example, 452 distinct subsequences are reduced to 106 distinct subsequences with reducing threshold value. This forms the input for regeneration of compressed training and test data.

4.4. Generation of compressed training and test data

Based on pruned distinct subsequence list, say, 1..106, the training data is regenerated by replacing distinct subsequences with these coded numbers. Each of those subsequences of the training data which are not available among the distinct subsequence list is replaced with its nearest

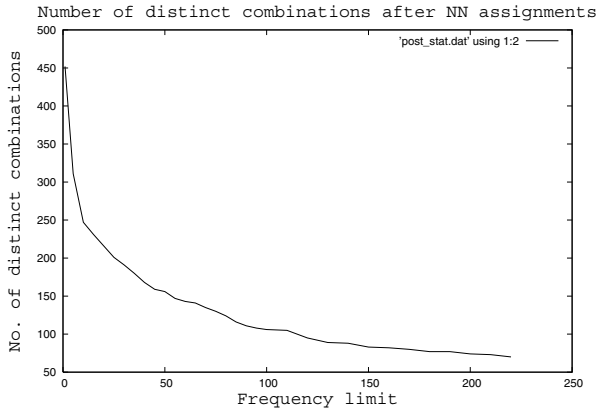

Figure 4. No. of distinct subsequences

Neighbour among the distinct sequences, within the dissimilarity threshold, η. Similarly, after generating subsequences in the test data, compressed test data is generated.

5. Summary of results

A number of case studies were carried out, by varying the values of ϵ, η, ψ.

With "rough set" approach, the *best classification accuracy obtained is 94.13%*, by classifying 94.54% of the test

patterns and rejecting 182 out of 3333 test patterns unclassified.

The results obtained by *k*-NNC approach, are provided in Figure 5. *It should be noted from the figure that 543 distinct subsequences are reduced to 70, without drastically reducing the classification accuracy*. The best *classification accuracy obtained using k-NNC is 93.3%* for ϵ=50, η=3 and ψ =1. It made use of 452 distinct subsequence. However, it should be noted that classification accuracy is not significantly different even for increasing ψ. For example, with *106 distinct subsequences, the classification accuracy obtained is 92.92%*.

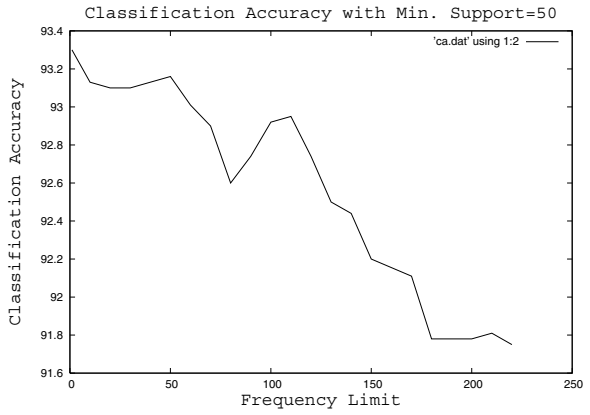

Figure 5. Classification Accuracy for ϵ=50

Another significant result of the method is that the space required for storing 106 distinct subsequences, each of length of 3 is given by 106*3*4=1272 bits. The original data size is 6670*192=1,280,640 bits. This is *superior to the compression achieved using Rough PC Tree approach on the same data* [2].

6. Conclusions

Large Handwritten data is compressed by means of a novel method.

It combines the concepts from supervised learning, frequent item sets, data abstraction and compression, rough set and k-NNC classification.

Such a lossy compression of data leads to a very *significant summarization* of data of 1,280,640 bits using 106, 12-bit strings. The data is classified in its *compressed form directly*. The classification accuracy obtained is 93.3% with *k*-NNC and 94.13% with the rough set approach. The activity combines more than one learning technique, leading to hybrid learning methodology.

Further work is proposed in the direction of run length code generation of compressed data(Table-4).

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data(SIGMOD'93)*, pages 207–216, 1993.

[2] V. Ananthanarayana and M. N. Murty. Tree structure for efficient data mining using rough sets. *Pattern Recognition Letters*, 24(6):851–862, March 2003.

[3] T. R. Babu and M. N. Murty. Comparison of genetic algorithm based prototype selection schemes. *Pattern Recognition*, 34(3):523–525, March 2001.

[4] J. Deogun, V. Raghavan, and H. Sever. Rough set based classification methods for extended decision tables. In *Proceedings of International Workshop on Rough Sets and Soft Computing*, pages 302–309, 1994.

[5] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data(SIGMOD'00)*, pages 1–12, 2000.

[6] A. Jain and R. Dubes. *Algorithm for clustering data*. Prentice Hall, New Jersey, 1988.

[7] D. Lelewer and D. Hirshberg. Data compression. *ACM Computing Surveys*, (9):261–296, September 1987.

[8] Z. Pawlak, J. Grzymala-Busse, R. Slowinski, and W. Ziarko. Rough sets. *Communications of the ACM*, 38(11):89–95, November 1995.

[9] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann Publishers, New Jersey, 2000.