Extending DeEPs, An Instance-Based Lazy Discovery And Classification System

Onur Onder
oonder@cs.bilkent.edu.tr
Computer Science Department
Bilkent University
Ankara - TURKEY

April 7, 2005

Abstract

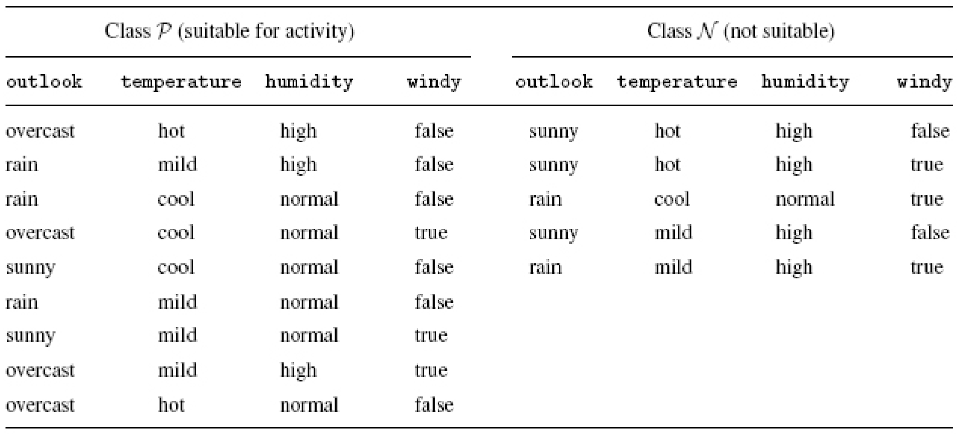
DeEPs concentrates on the frequency of emerging patterns that can be computed between the classes. One of the features of this classification system is to use the infinite frequency-change rate between the emerging patterns. Instead of infinite rate, a finite rate can be used, which would include the discovery of attribute subsets that are found in both classes.

Keywords: DeEPs, instance-based, lazy discovery, classification, finite frequency-change rate

1 Introduction

1.1 DeEPs Definition

DeEPs is defined in [1], is the acronym of "Decision making by Emerging Patterns". It uses pattern information on the training set to find the class of a test instace. The DeEPs technique described in the paper does the classification by calculating the frequency of patterns found in each class. The frequency-change rate between the classes creates a base for the idea and only the infinite rates are considered between the classes. DeEPs is a lazy learning algoritm and to



Class $\mathcal{P}$ (suitable for activity)				Class $\mathcal{N}$ (not suitable)			
`outlook`	`temperature`	`humidity`	`windy`	`outlook`	`temperature`	`humidity`	`windy`
overcast	hot	high	false	sunny	hot	high	false
rain	mild	high	false	sunny	hot	high	true
rain	cool	normal	false	rain	cool	normal	true
overcast	cool	normal	true	sunny	mild	high	false
sunny	cool	normal	false	rain	mild	high	true
rain	mild	normal	false				
sunny	mild	normal	true				
overcast	mild	high	true				
overcast	hot	normal	false				

Figure 1: Whole training set

make sure that it does compete with the other lazy learning classifiers some data reduction methods are described by the authors.

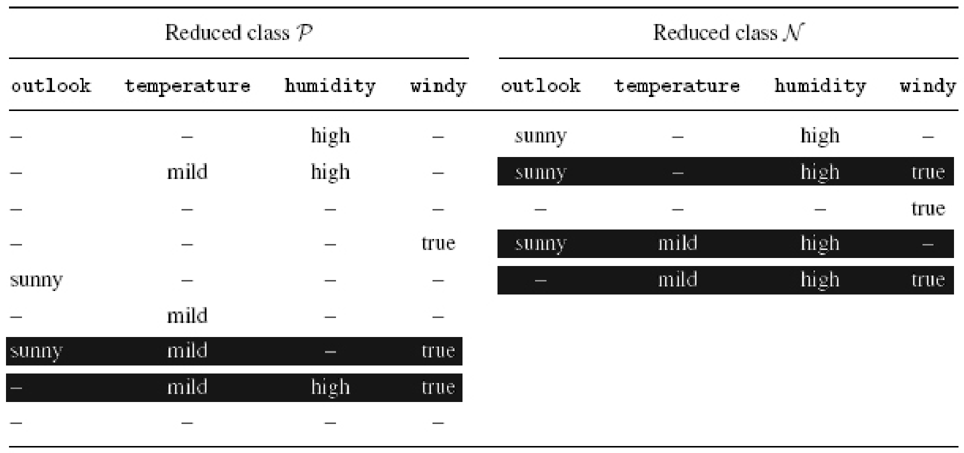
1.2 Data Set Reduction

1.2.1 Eliminating Irrevelant Data

Since the lazy learning approaches has no (or little) previous learning phase, the training data must be evaluated at the testing phase. Considering efficiency, the training data must be simplified as much as possible. For this purpose, initially the irrevelant data is removed or filtered out from the data set. This can be done by eliminating the data that does not match the test instance attributes. This operation will decrease the data set in horizontal dimention.

After eliminating the irrelevant data, the maximal instances can be chosen to further make the data set sparse in the vertical dimension.

For instance, the figures 1 and 2 show the original training set, then the removed irrelevant data and the maximal instances.



Reduced class $\mathcal{P}$				Reduced class $\mathcal{N}$			
outlook	temperature	humidity	windy	outlook	temperature	humidity	windy
–	–	high	–	sunny	–	high	–
–	mild	high	–	sunny	–	high	true
–	–	–	–	–	–	–	true
–	–	–	true	sunny	mild	high	–
sunny	–	–	–	–	mild	high	true
–	mild	–	–				
sunny	mild	–	true				
–	mild	high	true				
–	–	–	–				

Figure 2: Reduced data set, the highlighted instances are the maximal ones

1.2.2 Borders for Efficiency

The implementation of DeEPs classifier searches the subsets of a test instance attributes in the classes. However finding and evaluating all the subsets is not feasible. For this purpose, the concept of "Borders" is given.

A border is denoted by $< L, R >$, is defined as an ordered pair of two bounds L and R such that L and R are two anti-chains satisfying:

- $\forall X \in L, \exists Y \in R$ such that $X \subseteq Y$
- $\forall Y \in R, \exists X \in L$ such that $Y \supseteq X$

For instance, a border $< \{\{a\}, \{b\}\}, \{\{a, b, c\}, \{b, c, d\}\} >$ will describe the collection of $[L, R]$ that is $\{\{a\}, \{b\}, \{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}, \{c, d\}, \{a, b, c\}, \{b, c, d\}\}$.

Borders for a test instance can be found by the equations

$$[\{\emptyset\}, max\_Rp] - [\{\emptyset\}, max\_Rn]$$

$$[\{\emptyset\}, max\_Rn] - [\{\emptyset\}, max\_Rp]$$

where max_Rp and max_Rn defines the maximal items that are gathered from the classes in the previous section.

These difference operations can be efficiently implemented with the JepProducer algorithm given in [1]. Also the cases where there are more then two class are explained in [1].

1.3 DeEPs Classification Handling

After the data reduction operations, a score for each class is calculated and compared for the classification of test instance. For these scores to work some representatives from the boundaries has to be selected and used in the computation. The authors from [1] suggest that we should select hte left boundaries (L from $< L, R >$) because we know that they are the most general of the collection they represent, and also they cannot be further simplified. So after selecting these boundaries as representatives of their classes, they are put in a formulation to find the score. The method called "Compact Summation" has the following formula;

$$compactScore(c) = \frac{count_{D_c}(SEP)}{\mid D_c \mid}$$

where SEP is the collection of EPs (the collection represented by the boundary) and D_c is the set of training instances for class c. By this way, a score for all the classes can be computed and the one that has the highest score will be predicted as the class of the test instance.

2 Extending DeEPs

2.1 Main Idea: Finite Frequency-Change Rate

One of the enchantments that can be made on the DeEPs classifier is to let it handle the patterns that might occur on more then one class, that is the frequency-change rate of a pattern between any two class is not infinite.

For this case to work, there are some modifications that has to be done on the original system. For instance, the boundaries must cover some subsets that are found in more than one class. Another change would include how to include the fractions of patterns that are found in more then one class to the overall score. For these changes, firstly an example will be given to understand the idea of the extended system.

2.2 Implementation Details

Consider an example where the saturday morning sports activity is suitable or not for a given day[1]. Considering the figure 1, the table describes the training set for the two classes, Class P (that is the activity is suitable) and Class N (that is the activit is not suitable). And the test instance $T = \{sunny, mild, hightrue\}$ is given.

Using this information and continuing with the initial steps of DeEPs system, we need to eliminate irrelevant data and select the maximal items. The results are already shown in the figure 2.

At this point, the original DeEPs would calculate the borders, however, the common subsets of the test instance in both of the classes will be found. For

[1]The example information taken from [1]

the current example, the list of common subsets can be described with;

$$[\{\emptyset\}, \{\{s,m,t\}, \{m,h,t\}\}] \cap [\{\emptyset\}, \{\{s,h,t\}, \{s,m,h\}, \{m,h,t\}\}]$$

which are

$$commonS = \{\{\emptyset\}, \{s\}, \{m\}, \{h\}, \{t\}, \{s,m\}, \{s,t\}, \{m,t\}, \{m,h\}, \{h,t\}, \{m,t,h\}\}$$

Then the frequency changes of these subsets between each class has to be calculated. For this purpose, the following formula will give the frequency change between two classes;

$$freqChange(c_1, c_2) = \sum_{x \in commonS} \frac{count_{c_1}(x)}{count_{c_2}(x)}$$

where c_1 and c_2 are the two classes and the result of this formula will give a score for the c_1 class.

Continuing from the example, the following scores are found;

$$freqChange(P, N) = 10.91$$

$$freqChange(N, P) = 10.83$$

which would result in predicting the class P as the test instance's class.

2.3 Multi-class Case

When there exists more then two classes to decide, the *freqChange* formula can be applied to all of the other classes and the results will be added up to form that class' overall score. For instance, if we have k classes, say D_1, D_2, ..., D_k, then the score for D_i will be calculated as;

$$score(D_i) = \sum_{j \in [1,k], j \neq i} freqChange(D_i, D_j)$$

2.4 Non-discrete Valued Attributes

As defined in [1], a similar approach will be taken for the continuous valued attributes. For instance is an attribute has a continuous value, a small ϵ will be chosen for an interval, which will help us to make the comparisons with the other values. Let's say if there is some value like 1.34 and the $\epsilon = 0.02$ then the values in the interval

$$[x - \epsilon, x + \epsilon] = [1.32, 1.36]$$

will be viewed as the same data and for the intersection process they will be counted as an intersection point.

2.5 Efficiency

Finding an efficient algorithm for the intersection operation is required in this process since the evaluation of the whole subset collection will be very expensive. For this intersection operation, the following algorithm will be used;

```
commonS($< \{A_1, A_2, ..., A_{k_1}\} >$, $< \{B_1, B_2, ..., B_{k_2}\} >$

1. $diff \leftarrow JepProducer(< \{\emptyset\}, \{A_1, A_2, ..., A_{k_1}\} >, < \{\emptyset\}, \{B_1, B_2, ..., B_{k_2}\} >$
   $)$
2. $commonS \leftarrow JepProducer(< \{\emptyset\}, \{A_1, A_2, ..., A_{k_1}\} >, diff)$
3. return $commonS$
```

JepProducer algorithm taken from [1] and included here in figure 3.

```
JEPPRODUCER($\langle\{\emptyset\}, \{A_1, \ldots, A_{k_1}\}\rangle, \langle\{\emptyset\}, \{B_1, \ldots, B_{k_2}\}\rangle$)
    ;; return $\langle\mathcal{L}, \mathcal{R}\rangle$ such that $[\mathcal{L}, \mathcal{R}] = [\{\emptyset\}, \{A_1, \ldots, A_{k_1}\}] - [\{\emptyset\}, \{B_1, \ldots, B_{k_2}\}]$
    1) $\mathcal{L} \leftarrow \{\}$; $\mathcal{R} \leftarrow \{\}$;
    2) for $j$ from 1 to $k_1$ do
    3)     if some $B_{k_i}$ is a superset of $A_j$ then continue;
    4)     border = BORDER-DIFF($\langle\{\emptyset\}, \{A_j\}\rangle, \langle\{\emptyset\}, \{B_1, \ldots, B_{k_2}\}\rangle$);
    5)     $\mathcal{R} = \mathcal{R} \cup$ the right bound of border;
    6)     $\mathcal{L} = \mathcal{L} \cup$ the left bound of border;
    7) return $\langle\mathcal{L}, \mathcal{R}\rangle$;

BORDER-DIFF($\langle\{\emptyset\}, \{U\}\rangle, \langle\{\emptyset\}, \{S_1, S_2, \ldots, S_k\}\rangle$)
;; return border of $[\{\emptyset\}, \{U\}] - [\{\emptyset\}, \{S_1, S_2, \ldots, S_k\}]$
1) initialize $\mathcal{L}$ to $\{\{x\} \mid x \in U - S_1\}$;
2) for $i = 2$ to $k$ do
3)     $\mathcal{L} \leftarrow \{X \cup \{x\} \mid X \in \mathcal{L}, x \in U - S_i\}$;
4)     remove all $Y$ in $\mathcal{L}$ that are not minimal;
5) return $\langle\mathcal{L}, \{U\}\rangle$;
```

Figure 3: The JepProducer algorithm

3 Conclusion

Throughout the previous sections, the some of the steps of DeEPs classifier is redefined for the case where the patterns are seached within more then one class. After that the scores are calculated based on these pattern frequencies in each class and a prediction is made by comparing these scores between the classes.

As a future work, the performance evaluation of this extended system can be made. Comparing the results with the normal DeEPs and other lazy discovery classifiers. Since the algorithm uses the JepProduces routine twice as the normal DeEPs, the speed might be decreased, but the accuracy of the system should be increased.

Another option might be to unite the results of the normal DeEPs system with the extended one and make a prediction based on the both results. Based on the data set, some predefined biases might be placed to balance the results of the classifiers.

This extended DeEPs classifier might be slower than the other lazy discovery classifiers, but on the accuracy part, it will probablt be able to at least compete with the others. The system will probably result in be better outcomes in error-critical jobs instead of time-critical ones.

References

[1] J. Li , G. Dong , K. Ramamohanarao and L. Wong, "DeEPs: A New Instance-Based Lazy Discovery and Classification System," *Machine Learning*, vol. 54, no. 2, pp. 99 - 124, 2004

[2] Hongjian Fan "Efficient Mining of Interesting Emerging Patterns and Their Effective Use in Classification," *PhD Thesis*, University Of Melbourne, 2004

[3] J. Li , G. Dong , K. Ramamohanarao, "Instance-Based Classification by Emerging Patterns," *Principles of Data Mining and Knowledge Discovery*, 2000

[4] G. Dong, J. Li, "Efficient Mining of Emerging Patterns: Discovering Trends and Differences," *Knowledge Discovery and Data Mining*, 1999