

Expanding the Training Data Space Using Emerging Patterns and Genetic Methods

Hamad Alhammady and Kotagiri Ramamohanarao

Department of Computer Science and Software Engineering
The University of Melbourne
Parkville, Victoria 3010, Australia
Email: {hhammady, rao}@cs.mu.oz.au

Abstract. Classification is a major problem in machine learning. Many classifiers have been developed recently. However, the performance of these classifiers is proportional to the knowledge obtained from the training data. As a result, traditional classifiers can not perform very well when the training data space is very limited. In this paper, we propose a new approach to expand the training data space (ETDS) using emerging patterns (EPs) [4] and genetic methods (GMs) [7]. EPs are those itemsets whose supports in one class are significantly higher than their supports in the other classes. GMs are evolutionary methods that incorporate computational techniques inspired by biology [8]. We combine the power of EPs and GMs to expand the training data space before applying standard classifiers. The expansion process is performed by generating more training instances using four techniques. An extensive experimental evaluation carried out on a number of datasets shows that our approach has a great impact on the performance of many traditional classifiers.

Keywords: emerging patterns, genetic methods

1 Introduction

The problem of classification is one of the most important research topics in KDD (Knowledge Discovery in Databases). Different classifiers have been proposed recently to solve this problem. These classifiers are based on one general idea; they obtain knowledge from the training dataset, and then they use this knowledge in a certain way to classify a test instance. Generally, traditional classifiers differ on the way they use the knowledge obtained from the training dataset. They have no control over the amount of knowledge which has a great impact on the classification performance, and which depends on the number of available instances in the training dataset.

In this paper, we employ emerging patterns (EPs) [4] and genetic methods (GMs) [7] to expand the data space of the training sets. EPs are a new kind of patterns introduced recently [4]. EPs can be used in many applications such as rare-class classification [1] [2].

They are defined as itemsets whose supports increase significantly from one class to another. The discriminating power of EPs can be measured by their growth rates. The growth rate of an EP is the ratio of its support in a certain class over that in another class. Usually the discriminating power of an EP is proportional to its growth rate.

2 Emerging Patterns and Classification

Let $obj = \{a_1, a_2, a_3, \dots, a_n\}$ is a data object following the schema $\{A_1, A_2, A_3, \dots, A_n\}$. $A_1, A_2, A_3, \dots, A_n$ are called *attributes*, and $a_1, a_2, a_3, \dots, a_n$ are *values* related to these attributes. We call each pair (attribute, value) an *item*.

Let I denote the set of all items in an encoding dataset D . *Itemsets* are subsets of I . We say an instance Y contains an itemset X , if $X \subseteq Y$.

Definition 2.1. Given a dataset D , and an itemset X , the support of X in D , $s_D(X)$, is defined as

$$s_D(X) = \frac{count_D(X)}{|D|} \quad (1)$$

where $count_D(X)$ is the number of instances in D containing X .

Definition 2.2. Given two different classes of datasets D_1 and D_2 . Let $s_i(X)$ denotes the support of the itemset X in the dataset D_i . The growth rate of an itemset X from D_1 to D_2 , $GR_{D_1 \rightarrow D_2}(X)$, is defined as

$$GR_{D_1 \rightarrow D_2}(X) = \begin{cases} 0, & \text{if } s_1(X)=0, s_2(X)=0 \\ \infty, & \text{if } s_1(X)=0, s_2(X) \neq 0 \\ \frac{s_2(X)}{s_1(X)}, & \text{otherwise} \end{cases} \quad (2)$$

Definition 2.3. Given a growth rate threshold $p > 1$, an itemset X is said to be a *p-emerging pattern* (*p-EP* or simply EP) from D_1 to D_2 if $GR_{D_1 \rightarrow D_2}(X) \geq p$.

Let $C = \{c_1, \dots, c_m\}$ is a set of *class labels*. A *training dataset* is a set of data objects such that, for each object *obj*, there exists a class label $c_{obj} \in C$ associated with it. A *classifier* is a function from attributes $\{A_1, A_2, A_3, \dots, A_n\}$ to class labels $\{c_1, \dots, c_m\}$, that assigns class labels to unseen examples.

3 Expanding the Training Data Space

3.1 Overview

We propose a new approach to improve the classification performance. The key idea is to expand the training data space by generating more training instances. We present four methods to generate additional training instances. These methods involve using emerging patterns (EPs) and genetic methods (GMs). We use an existing algorithm [3] to mine EPs. The generation methods are presented in subsections 3.2, 3.3, 3.4, and 3.5.

3.2 Method 1: Generation by Superimposing EPs

This method was first introduced in our earlier work [2]. The main difference is that our work in [2] is dedicated to deal with rare-class datasets. Hence the generation is performed for the rare class only. In this paper we deal with balanced datasets, and the generation is performed for all classes. Given a training dataset and a set of mined EPs, the following steps are repeated for every class in the dataset:

- The attribute values (considered as itemsets consisting of one attribute) that have the highest growth rates in the current class are found.
- The set of EPs related to current class is divided into a number of groups such that EPs in each group have attribute values for most of the elements in the attribute set.
- The new instances are generated by combining the EPs in each group. If a value for an attribute is missing from a group, it is substituted by the value that has the highest growth rate for the same attribute (found in step 1).

Figure 1 shows an example of this process. Suppose we have a dataset consisting of 7 attributes $\{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$. The values that have the highest growth rate for these attributes in the current class are $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. Table a in figure 1 represents a group of three EPs. These EPs are $e_1\{(A_1=1), (A_2=x_1)\}$, $e_2\{(A_5=y_1), (A_6=2), (A_7=3)\}$, and $e_3\{(A_2=x_2), (A_3=4), (A_5=y_2)\}$. Notice that none of these EPs has a value for attribute A_4 . As a result, the value

that has the highest growth rate for attribute A_4 , v_4 , will be used as described earlier. The three EPs and the value chosen for attribute A_4 are combined to generate four new instances for the current class. The intersected values (x_1 and x_2 for attribute A_2 , and y_1 and y_2 for attribute A_5) are used one after the other to generate the new instances. The four generated instances are shown in table b in figure 1.

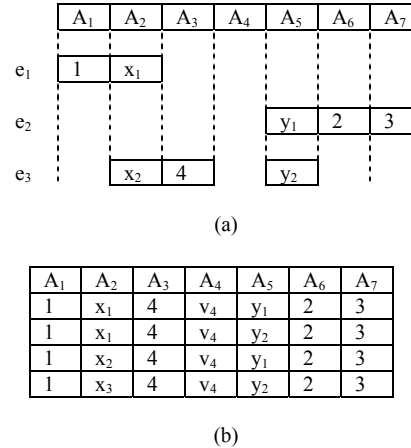


Figure 1: Example of generation by superimposing EPs

3.3 Method 2: Generation by Crossover

Consider two instances and a randomly chosen breaking point. The values of these instances are switched before the breaking point. Figure 2 shows an example of this process. Suppose that we have two instances $I_1 \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$ and $I_2 \{b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$, and the breaking point is randomly chosen to be between the third and fourth attributes. Then the two generated instances are $I_3 \{b_1, b_2, b_3, a_4, a_5, a_6, a_7\}$ and $I_4 \{a_1, a_2, a_3, b_4, b_5, b_6, b_7\}$.

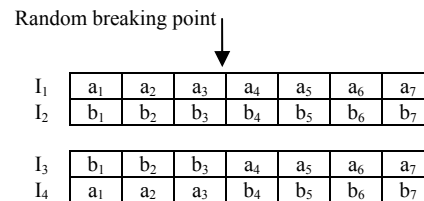


Figure 2: Example of generation by crossover

3.4 Method 3: Generation by Mutation

In this method, we mutate an instance with the highest-growth-rate values. The first step is to find the attribute values that have the highest growth rates in the current

class. A random binary number is chosen and overlapped over the instance. All attribute values in the instance that match 1's in the random binary number are replaced by the values that have the highest growth rates. Figure 3 shows an example of this method. Suppose that we have an instance $I_1 \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$. The highest-growth-rate values are $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. The random binary number is (1001011). The values in I_1 that match 1's in the random binary number (namely, the first, fourth, sixth, and seventh attributes) are replaced with the highest-growth-rate values that they match.

I_1	a_1	a_2	a_3	a_4	a_5	a_6	a_7
Random binary number	1	0	0	1	0	1	1
Highest-growth-rate values	v_1	v_2	v_3	v_4	v_5	v_6	v_7
I_2	v_1	a_2	a_3	v_4	a_5	v_6	v_7

Figure 3: Example of generation by mutation

3.5 Method 4: Generation by Mutation and EPs

In this method, we mutate an instance with an EP. All values in the instance are replaced with their matched values in the EP. Figure 4 shows an example of this method. Suppose that we have an instance $I_1 \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$. We want to mutate this instance with an EP $e \{e_2, e_3, e_6\}$. We replace the values in I_1 (namely, the second, third, and sixth values) with their matched values in e .

I_1	a_1	a_2	a_3	a_4	a_5	a_6	a_7
e		e_2	e_3			e_6	
I_2	a_1	e_2	e_3	a_4	a_5	e_6	a_7

Figure 4: Example of generation by mutation and EPs

3.6 Generation Constraints and Fitness Function

Using the generation methods explained in subsections 3.2, 3.3, 3.4, and 3.5, a large number of training instances can be generated. Our aim is to generate a reasonable number of high-quality training instances. We propose four constraints to achieve this aim. These constraints are:

1. Instead of using the whole range of EPs, we use a certain percentage (α) of the best-quality EPs in method 1 (generation by superimposing EPs) and method 4 (generation using mutation and EPs).

2. Instead of using the whole range of the training data, we use a certain percentage (β) of the best-quality instances in method 2 (generation using crossover), method 3 (generation using mutation), and method 4 (generation using mutation and EPs).
3. Adding the best-quality generated instances to the training dataset, and discarding the bad-quality generated instances.
4. Stopping the generating process when the training dataset is increased by a certain percentage (χ).

The quality of EPs and instances (constraints 1, 2, and 3) can be measured by their fitness functions. A fitness function is defined as a measure to evaluate the elements (EPs or data). For the EPs (constraint 1), the fitness function can be measured by the growth rate. The growth rate of an EP is proportional to its discriminating power. Hence, it indicates how good this EP is. The fitness function of an EP (growth rate) is defined by equation 2.

The fitness function of a data instance can be measured by the average support of the attribute values in this instance. Suppose that we have an instance $I \{a_1, a_2, a_3, \dots, a_n\}$. We first find the supports of all the attribute values (from a_1 to a_n). Then we average these supports to obtain a measure that tells how good the instance is. This fitness function is defined by equation 3.

$$Fitness(I) = \frac{\sum_{i=1}^n s(a_i)}{n} \quad (3)$$

We use this function to choose the best-quality instances to be used in the generating process (constraint 2). Moreover, we use it to evaluate (keep or discard) the instances generated by our proposed methods (constraint 3).

4 Experimental Evaluation

In order to investigate the performance of our proposed system, we carry out a number of experiments. We use 30 datasets from UCI repository of machine learning databases [5]. We also use C4.5, boosting (C4.5), and SVM [6] as base learners. The testing method is stratified 10-cross-validation. Each experiment is carried out as follows:

- Run a base learner $\{M\}$.
- Apply method 1 (generation by superimposing EPs) and run the base learner $\{M1\}$.

- Apply method 2 (generation by crossover) and run the base learner {M2}.
- Apply method 3 (generation by mutation) and run the base learner {M3}.
- Apply method 4 (generation by mutation and EPs) and run the base learner {M4}.
- Apply the union of the four methods and run the base learner {M*}.

The last method (M*) is explained as follows. We use our four methods (presented in section 3) to generate more training instances. We use the fitness function (presented in subsection 3.6) to evaluate the generated instances and choose the best instances resulted from the four methods.

The results of our experiments are shown in tables 2, 3, and 4. Taking into consideration that no classification method can outperform all others in all datasets, and that different classifiers deal differently with a dataset, we can draw some interesting points from results as follows:

- Using C4.5, M3 wins on 8 datasets, M* wins on 7 datasets, M1 wins on 6 datasets, M4 wins on 4 datasets, M2 wins on 3 datasets, and M wins on 2 datasets.
- Using boosting (C4.5), M1 wins on 8 datasets, M* wins on 6 datasets. M3 wins on 5 datasets, both M2 and M4 win on 4 datasets, and M wins on 3 datasets.
- Using SVM, M* wins on 8 datasets, Both M1 and M3 win on 6 datasets, M2 wins on 5 datasets, M4 wins on 3 datasets, and M wins on 2 datasets.
- Considering the three classifiers, M*, M1, and M3 are the most powerful methods as they win on 21, 20, and 19 cases, respectively. M2 wins on 12 cases, M4 wins on 11 cases, and M wins on 7 cases.
- Our approach as a whole (considering the winning method for each dataset) outperforms the base learner in 28 datasets out of the 30 datasets used in our experiments. This significant result is achieved in the three sets of experiments (C4.5 experiments, boosting experiments, and SVM experiments).
- In terms of the average accuracy, M* outperforms all other methods using the three classifiers.
- Our proposed methods increase the accuracy significantly in some cases. For example, the increase in C4.5 experiments is up to 5.6% (Hayes-roth dataset). In boosting experiments, the increase is up to 5% (Pima dataset). In SVM experiments, the increase is up to 4.1% (Iono dataset).

5 Conclusions

In this paper, we propose a new approach to expand the training data space. Our approach, called ETDS, introduces the idea of generating new training instances using emerging patterns (EPs) and genetic methods (GMs). It uses the advantage of EPs, which have a strong discriminating power, and GMs, which have a great reproduction ability, to expand the training data space by generating new instances for the training set. We introduce four methods to achieve this aim. We experimentally demonstrate that our approach has a great impact on improving the results of other classification methods such as C4.5, boosting, and SVM. Furthermore, our approach can be thought of as an avenue to extend the applications of EPs and GMs to cover a wide range of problems in data mining.

Table 2: Results of C4.5 experiments

Dataset	C4.5					
	M	M1	M2	M3	M4	M*
Adult	86.1	88.3	87.2	87.7	86.4	87.8
Australian	84.3	84.9	85.7	84.4	83.9	86.9
Breast	94.6	96.3	94.6	95.1	95.9	95.9
Cleve	73.8	72.6	73.2	74.9	74.6	74.8
Credit card	85.3	86.4	85.2	86.7	87.2	86.9
Crx	85.3	85.8	85.1	83.7	84.6	85.5
Diabetes	73.4	71.6	71.8	72.8	72.2	74.7
Flags	57.5	59.3	59.7	57.9	57.1	59.3
German	69.6	69.9	70.2	71.3	70.6	72.6
Glass	64.7	66.2	64.9	64.1	65.3	66.1
Hayes-roth	70.2	70.7	72.5	75.3	75.8	75.5
Heart	80.6	81.1	80.4	81.3	80.9	81.1
Hepatitis	81.8	80.9	80.2	81.7	82.6	82.4
Horse	85.2	85.4	85.4	85.9	85.7	85.6
Hypo	99.3	97.2	98.1	96.8	98.4	99.4
Iono	89.4	88.6	89.5	89.1	89.8	91.1
Labor	76.9	76.2	77.3	77.8	76.9	77.5
Liver	58.1	59.3	59.8	58.5	61.3	60.6
Machine	87	87.5	89.5	89.2	88.4	89.2
Pima	74	76.2	74	73.4	74.7	76.1
Segment	93.5	93.6	92.7	94.8	94.1	94.4
Sick	98.7	97.5	97.7	98.9	98.1	98.6
Sonar	75.3	76.8	76.3	77.6	75.4	77.4
Staimage	85.2	84.7	85.5	86.1	84.9	85.8
Tic-tac-toe	84.2	86.5	88.1	84.6	87.1	87.8
Vehicle	71.2	72.8	69.5	70.2	70.7	72.9
Votes	77.8	78.9	77.4	78.5	78.1	78.8
Wine	84.2	85.6	83.1	84.8	85.2	85.9
Yeast	49.9	48.1	48.5	49.4	48.6	49.7
Zoo	93	91	89	91	90	92
Average	79.67	79.99	79.73	80.11	80.15	82.50

Table 3: Results of boosting experiments

Dataset	Boosting(C4.5)					
	M	M1	M2	M3	M4	M*
Adult	86.6	88.7	86.9	86.1	87.3	88.6
Australian	82.3	83.4	81.9	84.1	83.1	83.8
Breast	95.9	97.1	96.3	95.7	96.3	96.9
Cleve	80.8	79.9	81.2	79.2	83.1	82.7
Credit card	84.9	86.4	84.4	85.6	85.1	85.7
Crx	82.8	83.2	83.7	82.8	82.4	83.5
Diabetes	72	72	72.8	72.3	72.2	72.4
Flags	54.9	57.8	55.1	55.3	57.2	57.7
German	71.8	73.1	71.2	73.3	72.4	72.8
Glass	74.1	74.9	76.5	74.3	75.2	76.4
Hayes-roth	78.3	79.5	78.5	78.5	78.1	79.7
Heart	76.2	77.4	76.5	79.2	77.2	78.9
Hepatitis	79.2	79.7	80.5	81.3	80.1	82.4
Horse	82.2	82	81.9	82.2	84.3	84.1
Hypo	98.9	98.1	98.5	98.1	97.3	98.8
Iono	92	89.8	91.5	87.7	89.2	92.7
Labor	82	82.6	84.1	84.8	82	84.6
Liver	61	61.5	60.7	60.4	61.3	61.4
Machine	90.2	92.6	90.9	90	90.9	92.4
Pima	71.5	75.2	73.3	75	73.7	76.5
Segment	94.8	94.2	95.2	94.9	96.3	97.7
Sick	98.7	98.6	98.1	98.9	98.4	98.7
Sonar	78.2	78.7	79.2	78.1	79.5	79.4
Stainage	85.9	86.2	86.5	85.4	86.7	86.5
Tic-tac-toe	95.4	97.8	94.9	94.4	95.3	97.5
Vehicle	73	73.5	74.2	73.5	73.7	75.3
Votes	77.9	79.1	76.8	78.6	78.2	78.8
Wine	91.5	90.1	92.7	91.8	91.8	92.6
Yeast	50.2	49.3	48.6	48.3	48.2	48.9
Zoo	98	95	95	91	93	97
Average	81.37	81.91	81.58	81.36	81.65	82.81

Table 4: Results of SVM experiments

Dataset	SVM					
	M	M1	M2	M3	M4	M*
Adult	87.1	86.7	85.3	86.2	85.3	87.5
Australian	85.1	86.8	84.6	85.9	86.2	86.6
Breast	95.9	94.6	95.5	95.1	94.3	95.7
Cleve	84.4	86.3	85.1	85.8	84.2	86.7
Credit card	84.6	85.1	84.3	85.9	85.1	85.8
Crx	84.7	85.2	84.2	84.9	84.9	84.9
Diabetes	72.8	73.6	72.8	72.9	73.2	73.3
Flags	60.6	59.2	60.1	59.2	57.7	59.9
German	74.6	74.7	74.1	75.9	75.1	75.7
Glass	75.7	76.1	75.1	75.8	76.9	77.8
Hayes-roth	84.7	84.1	85.6	86.1	85.2	86.9
Heart	85.1	86.7	84.3	84.6	84.3	86.5
Hepatitis	83.3	82.9	85.2	82.9	83.7	84.8
Horse	86.9	86.4	86.3	87.7	87.5	87.5
Hypo	98.9	97.7	98.9	98.9	99.2	98.9
Iono	88.5	89.3	91.1	89.5	88.7	92.6
Labor	97.4	97.6	97.1	97.9	96.4	97.6
Liver	66.8	67.1	66.5	66.5	65.8	66.8
Machine	93.7	93.5	93.1	92.8	92.4	93.8
Pima	73.6	74.2	75.8	74.5	73.9	75.7
Segment	95.5	95.9	97.1	95.9	95.6	96.9
Sick	93.9	95.4	94.3	94.3	93.2	96.6
Sonar	77.7	78.3	78.2	78.9	77.3	78.6
Stainage	86.7	87.3	88.8	87.7	86.7	88.2
Tic-tac-toe	98.3	98.8	98.3	98.1	97.4	98.7
Vehicle	68.5	68.8	69.9	68.8	66.1	69.8
Votes	64.7	64.1	64.7	65.6	65.4	65.9
Wine	95.4	94.2	95.1	97.1	95.9	96.9
Yeast	52.7	51.5	52.1	52.7	52.9	52.7
Zoo	97	93	92	92	98	97
Average	83.16	83.17	83.18	83.33	82.95	84.21

References

- [1] H. Alhammady, and K. Ramamohanarao. The Application of Emerging Patterns for Improving the Quality of Rare-class Classification. In Proceedings of 2004 Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '04), Sydney, Australia.
- [2] H. Alhammady, and K. Ramamohanarao. Using Emerging Patterns and Decision Trees in Rare-class Classification. In Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM '04), Brighton, UK.
- [3] H. Fan, and K. Ramamohanarao. An Efficient Single-Scan Algorithm For Mining Essential Jumping Emerging Patterns for Classification. In Proceedings of 2002 Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '02), Taipei, Taiwan.
- [4] G. Dong, and J. Li. Efficient Mining of Emerging Patterns: Discovering Trends and Differences. In Proceedings of 1999 International Conference on Knowledge Discovery and Data Mining (KDD '99), San Diego, CA, USA.
- [5] C. Blake, E. Keogh, and C. J. Merz. UCI repository of machine learning databases. Department of Information and Computer Science, University of California at Irvine, CA, 1999. <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
- [6] I. H. Witten, E. Frank. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Mateo, CA., 1999.
- [7] Z. Michalewicz. Genetic Algorithms + Data Structure = Evolution Programs. Springer-Verlag, Berlin, 1996.
- [8] M. Obitko, and P. Slavik. Visualization of Genetic Algorithms in a Learning Environment. In Proceedings of 1999 Spring Conference on Computer Graphics, (SCCG '99). Bratislava, Slovakia.