Learning Interestingness of Streaming Classification Rules

Tolga Aydın and Halil Altay Güvenir

Department of Computer Engineering, Bilkent University
06800 Ankara, Turkey
{atolga,guvenir}@cs.bilkent.edu.tr

Abstract. Inducing classification rules on domains from which information is gathered at regular periods lead the number of such classification rules to be generally so huge that selection of interesting ones among all discovered rules becomes an important task. At each period, using the newly gathered information from the domain, the new classification rules are induced. Therefore, these rules stream through time and are so called streaming classification rules. In this paper, an interactive rule interestingness-learning algorithm (IRIL) is developed to automatically label the classification rules either as "interesting" or "uninteresting" with limited user interaction. In our study, VFP (Voting Feature Projections), a feature projection based incremental classification learning algorithm, is also developed in the framework of IRIL. The concept description learned by the VFP algorithm constitutes a novel approach for interestingness analysis of streaming classification rules.

1 Introduction

Data mining is the efficient discovery of patterns, as opposed to data itself, in large databases [1]. Patterns in the data can be represented in many different forms, including classification rules, association rules, clusters, sequential patterns, time series, contingency tables, and others [2]. However, for example, inducing classification rules on domains from which information is gathered at regular periods lead the number of such classification rules to be generally so huge that selection of interesting ones among all discovered rules becomes an important task. At each period, using the newly gathered information from the domain, the new classification rules are induced. Therefore, these rules stream through time and are so called streaming classification rules.

In this paper, an interactive rule interestingness-learning algorithm (IRIL) is developed to automatically label the classification rules either as "interesting" or "uninteresting" with limited user interaction. In our study, VFP (Voting Feature Projections), a feature projection based incremental classification learning algorithm, is also developed in the framework of IRIL. The concept description learned by the VFP algorithm constitutes a novel approach for interestingness analysis of streaming classification rules. Being specific to our concerns, VFP takes the rule interestingness factors as features and is used to learn the rule interestingness concept and to classify the newly learned classification rules.

C. Aykanat et al. (Eds.): ISCIS 2004, LNCS 3280, pp. 62–71, 2004.

Section 2 describes the interestingness issue of patterns. Section 3 is devoted to the knowledge representation used in this study. Sections 4 and 5 are related to the training and classifying phases of the VFP algorithm. IRIL is explained in the following section. Giving the experimental results in Section 7, we conclude.

2 Interestingness Issue of Patterns

The interestingness issue has been an important problem ever since the beginning of data mining research [3]. There are many factors contributing to the interestingness of a discovered pattern [3-5]. Some of them are coverage, confidence, completeness, action ability and unexpectedness. The first three factors are objective, action ability is subjective and unexpectedness is sometimes regarded as subjective [6-8] and sometimes as objective [9,10]. Objective interestingness factors can be measured independently of the user and domain knowledge. However, subjective interestingness factors are not user and domain knowledge independent. The measurement of a subjective interestingness factor may vary among users analyzing a particular domain, may vary among different domains that a particular user is analyzing and may vary even for the same user analyzing the same domain at different times.

An objective interestingness measure is constructed by combining a proper subset of the objective interestingness factors in a suitable way. For example, objective interestingness factor x can be multiplied by the square of another objective interestingness factor y to obtain an objective interestingness measure of the form xy^2. It is also possible to use an objective interestingness factor x alone as an objective interestingness measure (e.g. *Confidence*). Discovered patterns having *Confidence* ≥ *threshold* are regarded as "interesting". Although the user determines the threshold, this is regarded as small user intervention and the interestingness measure is still assumed to be an objective one.

The existing subjective interestingness measures in the literature are constructed upon unexpectedness and action ability factors. Assuming the discovered pattern to be a set of rules induced from a domain, the user gives her knowledge about the domain in terms of fuzzy rules [8], general impressions [7] or rule templates [6]. The induced rules are then compared with user's existing domain knowledge to determine subjectively unexpected and/or actionable rules.

Both types of interestingness measures have some drawbacks. A particular objective interestingness measure is not sufficient by itself [8]. They are generally used as a filtering mechanism before applying a subjective measure. On the other hand, subjective measures are sometimes used without prior usage of an objective one. In the case of subjective interestingness measures, user may not be well in expressing her domain knowledge at the beginning of the interestingness analysis. It'd be better to automatically learn this knowledge based on her classification of some presented rules as "interesting" or "uninteresting". Another drawback of a subjective measure is that the induced rules are compared with the domain knowledge that addresses the unexpectedness and/or action ability issues. Interestingness is assumed to depend on these two issues. That is, if a rule is found to be unexpected, it is automatically regarded as an interesting rule. However, it would be better if we

learned a concept description that dealt with the interestingness issue directly and if we benefited from unexpectedness and action ability as two of the factors used to express the concept description. That is, interestingness of a pattern may depend on factors other than unexpectedness and action ability issues.

The idea of a concept description that is automatically determined and directly related with the interestingness issue motivated us to design IRIL algorithm. The concept description learned by the VFP algorithm, which was also developed in this framework, constitutes a novel approach for interestingness analysis of classification rules.

To ensure that the concept description is directly related to the rule interestingness issue, some existing and newly developed interestingness factors that have the capability to determine the interestingness of rules were used instead of the original attributes of the data set. Current implementation of IRIL does not incorporate unexpectedness and action ability factors, leading to no need for domain knowledge. Although the interestingness factors are all of type objective in the current version of IRIL, the thresholds of the objective factors are learned automatically rather than expressing them manually at the beginning. The values of these thresholds are based upon the user's classification results of some presented rules. So, although in the literature subjectivity is highly related to the domain knowledge, IRIL differs from them. IRIL's subjectivity is not related with the domain knowledge. IRIL makes use of objective factors (actually the current version makes use of only objective factors) but for each such a factor, it subjectively learns what ranges of factor values (what thresholds) lead to interesting or uninteresting rule classifications if only that factor is used for classification purposes. That is, IRIL presents a hybrid interestingness measure.

IRIL proceeds interactively. An input rule is labeled if the learned concept description can label the rule with high certainty. If the labeling or classification certainty factor is not of sufficient strength, user is asked to classify the rule manually. The user looks at the values of the interestingness factors and labels the rule accordingly. In IRIL, concept description is learned or updated incrementally by using the interestingness labels of the rules that are on demand given either as "interesting" or "uninteresting" by the user.

3 Knowledge Representation

We think of a domain from which information is gathered at regular periods. For each period p, classification rules are induced from the gathered information and these streaming rules' interestingness labeling seems to be an important problem. This labeling problem is modeled as a new classification problem and a *rule set* is produced for these rules. Each instance of the rule set is represented by a vector whose components are the interestingness factors having the potential to determine the interestingness of the corresponding rule and the interestingness label of the rule.

The classification rules used in this study are probabilistic and have the following general structure:

If (A_1 *op* $value_1$) AND (A_2 *op* $value_2$) AND …AND (A_n *op* $value_n$) THEN
($Class_1$: $vote_1$, $Class_2$: $vote_2$,…,$Class_k$: $vote_k$)
A_i's are the features, $Class_i$'s are the classes and *op* ∈ {=, ≠, <, ≤, >, ≥}.

The instances of the rule set have either "interesting" or "uninteresting" as the interestingness label, and have the interestingness factors shown in Table 1. In this new classification problem, these factors are treated as determining features, and interestingness label is treated as the target feature (class) of the rule set.

Table 1. Features of the rule set

Feature	**Short description and/or formula**
Major Class	$Class_i$ that takes the highest vote
Major Class Frequency	Ratio of the instances having $Class_i$ as the class label in the data set
Rule Size	Number of conditions in the antecedent part of the rule
Confidence with respect to Major Class	$\|Antecedent \,\&\, Class_i\| / \|Antecedent\|$
Coverage	$\|Antecedent\| / \|N\|$
Completeness with respect to Major Class	$\|Antecedent \,\&\, Class_i\| / \|Class_i\|$
Zero Voted Class Count	Number of classes given zero vote
Standard Deviation of Class Votes	Standard deviation of the votes of the classes
Major Class Vote	Maximum vote value distributed
Minor Class Vote	Minimum vote value distributed
Decisive	True if Std.Dev.of Class.Votes > s_{min}

Each feature carries information of a specific property of the corresponding rule. For instance, letting $Class_i$ to take the highest vote makes it the *Major Class* of that classification rule. If we shorten the representation of any rule as "If *Antecedent* THEN $Class_i$" and assume the data set to consist of N instances, we can define *Confidence*, *Coverage* and *Completeness* as in Table 1. Furthermore, a rule is decisive if the standard deviation of the votes is greater than s_{min}, whose definition is given in the following equation:

$$s_{min} = \frac{1}{(Class\ Count - 1)\sqrt{Class\ Count}} \tag{1}$$

If a rule distributes its vote, '1', evenly among all classes, then the standard deviation of the votes becomes zero and the rule becomes extremely indecisive. This is the worst vote distribution that can happen. The next worst vote distribution happens if exactly one class takes a zero vote, and the whole vote is distributed evenly among the remaining classes. The standard deviation of the votes that will occur in such a scenario is called s_{min}.

4 Training in the VFP Algorithm

VFP (Voting Feature Projections) is a feature projection based classification-learning algorithm developed in this study. It is used to learn the rule interestingness concept and to classify the unlabeled rules in the context of modeling rule interestingness problem as a new classification problem.

The training phase of VFP, given in Figure 3, is achieved incrementally. On a nominal feature, concept description is shown as the set of points along with the numbers of instances of each class falling into those points. On the other hand, on a numeric feature, concept description is shown as the gaussian probability density functions for each class. Training can better be explained by looking at the sample data set in Figure 1, and the associated learned concept description in Figure 2.

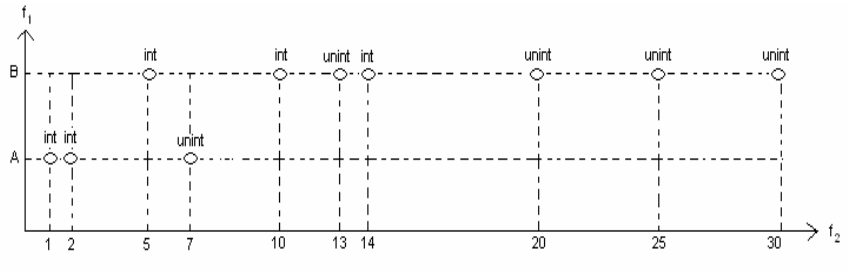

Fig. 1. Sample data set

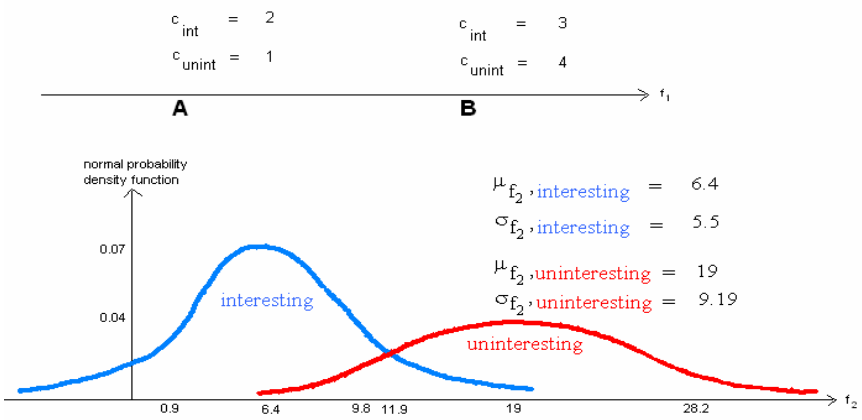

Fig. 2. Concept description learned for the sample data set

The example data set consists of 10 training instances, having nominal f_1 and numeric f_2 features. f_1 takes two values: 'A' and 'B', whereas f_2 takes some integer values. There are two possible classes: "interesting" and "uninteresting". f_2 is assumed to have gaussian probability density functions for both classes.

```
VFP_train (t)        /* t: newly added training instance */
begin
   let c be the class of t
   let others be the remaining classes
   if training set = {t}
      for each class s
         class_count[s] = 0
   class_count[c]++

   for each feature f

      if f is nominal
         p = find_point(f,t_f)
         if such a p exists
            point_class_count [f,p,c] ++
         else  /* add new point for f */
            add a new p' point
            point_class_count [f,p',c] = 1
            point_class_count [f,p',others] = 0

      else if f is numeric
        if training set = {t}
           μ_f,c = t_f        ,      μ_f,others = 0
           μ²_f,c = t_f²      ,      μ²_f,others = 0
           σ_f,c = Undefined
           norm_density_func._f,c = Undefined
        else
           n = class_count[c]
           μ_f,c = (μ_f,c * (n-1) + t_f) / n      /*update*/
           μ²_f,c = (μ²_f,c * (n-1) + t_f²) / n   /*update*/
           σ_f,c = sqrt( n/(n-1) (μ²_f,c − (μ_f,c)²) )

           norm_density_func._f,c = 1/(σ_f,c √(2π)) e^( −(x−μ_f,c)² / (2σ²_f,c) )

   return { For numeric features:
            norm_density_func._f,c (∀f, c)
            For nominal features:
            point_class_count[f, p, c] (∀f, p, c)
end.
```

Fig. 3. Incremental train in VFP

In Figure 3 for a nominal feature *f*, *find_point* (f, t_f) searches t_f, the new training instance's value at feature *f*, in the *f* projection. If t_f is found at a point *p*, then *point_class_count* [*f*, *p*, *c*] is incremented, assuming that the training instance is of class *c*. If t_f is not found, then a new point *p'* is constructed and *point_class_count* [*f*, *p'*, *class*] is initialized to 1 for *class* = *c*, and to 0 for *class* = *others*. In this study,

features used in VFP are the interestingness factor values computed for the classification rules, and the classes are "interesting" and "uninteresting".

For a numeric feature *f*, if a new training instance *t* of class *c* is examined, the previous training instances of class *c* is let to construct a set *P* and $\mu_{f,c}$ and $\sigma_{f,c}$ are let to be the mean and the standard deviation of the *f* feature projection values of the instances in *P*, respectively. $\mu_{f,c}$ and $\sigma_{f,c}$ are updated incrementally. Updating $\sigma_{f,c}$ incrementally requires $\mu^2_{f,c}$ to be updated incrementally, as well.

5 Classification in the VFP Algorithm

Classification in VFP is shown in Figure 4. The query instance is projected on all features If a feature is not ready to querying, it gives zero, otherwise normalized votes. Normalization ensures each feature to have equal power in classifying the query instances. For a feature to be ready to querying, it requires to have at least two different values for each class.

The classification starts by giving zero votes to classes on each feature projection. For a nominal feature *f*, *find_point* (f, q_f) searchs whether q_f exists in the *f* projection. If q_f is found at a point *p*, feature *f* gives votes as given in equation 2, and then these votes are normalized to ensure equal voting power among features.

$$feature\_vote\,[f, c] = \frac{point\_class\_count\,[\,f, p, c\,]}{class\_count\,[\,c\,]} \tag{2}$$

In equation 2, the number of class *c* instances on point *p* of feature projection *f* is divided by the total number of class *c* instances to find the class conditional probability of falling into the *p* point. For a linear feature *f*, each class gets the vote given in equation 3. Normal probability density function values are used as the vote values. These votes are also normalized.

$$feature\_vote\,[f, c] = \lim_{\Delta x \to 0} \int_{q_f}^{q_f + \Delta x} \frac{1}{\sigma_{f,c}\sqrt{2\pi}} e^{-\frac{(q_f - \mu_{f,c})^2}{2\sigma_{f,c}^2}} dx \tag{3}$$

Final vote for any class *c* is the sum of all votes given by the features. If there exists a class *c* that uniquely gets the highest vote, then it is predicted to be the class of the query instance. The certainty factor of the classification is computed as follows:

$$C_f = \frac{final\ vote[c]}{\sum_{i=1}^{\#Classes} final\ vote[i]} \tag{4}$$

VFP$_{query}$**(*q*)** /* *q*: query instance*/
begin
  for each feature *f* and class *c*
    feature_vote[*f*,*c*] = 0
    if *feature_ready_for_query_process*(*f*)
      if *f* is nominal
        p = *find_point*(*f*,q_f)
        if such a *p* exists
          for each class *c*

$$feature\_vote\ [f,c] = \frac{point\_class\_count[f,p,c]}{class\_count[c]}$$

          normalize_feature_votes (*f*)
      else if *f* is numeric
        for each class *c*
          feature_vote [*f*,*c*]=

$$\lim_{\Delta_x \to 0} \int_{q_f}^{q_f+\Delta x} \frac{1}{\sigma_{f,c}\sqrt{2\pi}} e^{-\frac{(q_f-\mu_{f,c})^2}{2\sigma_{f,c}^2}} dx$$

        normalize_feature_votes (*f*)
  for each class *c*

$$final\_vote\ [c] = \sum_{f=1}^{\#Features} feature\_vote\ [f,c]$$

  if $\min_{i=1}^{\#Classes} final\_vote[i] <$ *final_vote* [*k*] $= \max_{i=1}^{\#Classes} final\_vote[i]$
    classify *q* as "*k*" with a certainty factor C_f
    return C_f
  else return -1
end.

Fig. 4. Classification in VFP

6 IRIL Algorithm

IRIL algorithm, shown in Figure 5, needs two input parameters: R_p (The set of streaming classification rules of period *p* and $MinC_t$ (Minimum Certainty Threshold). It tries to classify the rules in R_p. If $C_f \geq MinC_t$ for a query rule *r*, this rule is inserted into the successfully classified rules set (R_s). Otherwise, two situations are possible: either the concept description is not able to classify *r* (C_f = -1), or the concept description's classification (prediction of *r*'s interestingness label) is not of sufficient strength. If $C_f < MinC_t$, rule *r* is presented, along with its computed eleven interestingness factor values such as *Coverage*, *Rule Size*, *Decisive* etc., to the user for classification. This rule or actually the instance holding the interestingness factor values and the recently determined interestingness label of this rule is then inserted into the training rule set R_t and the concept description is reconstructed incrementally.

All the rules in R_p are labeled either automatically by the classification algorithm, or manually by the user. User participation leads rule interestingness learning process to be an interactive one. When the number of instances in the training rule set increases, the concept description learned tends to be more powerful and reliable.

IRIL executes on classification rules of all the periods and finally concludes by presenting the labeled rules in R_s.

```
IRIL (R_p, MinC_t)
  begin
     R_t ← ∅,    R_s ← ∅
     if p is the 1st period          //Warm-up Period
        for each rule r ∈ R_p
                 ask the user to classify r
                 set C_f of this classification to 1
                 insert r into R_t
                 VFP_train (r)
     else
        for each rule r ∈ R_p
           C_f ←VFP_query (r)
              if C_f < MinC_t
                 ask the user to classify r
                 set C_f of this classification to 1
                 insert r into R_t
                 VFP_train (r)  //Update Concept Description
              else
                 insert r into R_s
     return rules in R_s
  end.
```

Fig. 5. IRIL algorithm

7 Experimental Results

IRIL algorithm was tested to classify 1555 streaming classification rules induced from a financial distress domain between years 1989 and 1998. Each year has its own data and classification rules induced by using a benefit maximizing feature projection based rule learner proposed in [11]. The data set of the financial distress domain is a comprehensive set consisting of 25632 data instances and 164 determining features (159 numeric, 5 nominal). There are two classes: "DeclareProfit" and "DeclareLoss". The data set includes some financial information about 3000 companies collected during 10 years and the class feature states whether the company declared a profit or loss for the next three years. Domain expert previously labeled all the 1555 induced rules by an automated process to make accuracy measurement possible. Rules of the first year are selected as the warm-up rules to construct the initial concept description.

The results for $\text{MinC}_t = 51\%$ show that 1344 rules are classified automatically with $C_f > \text{MinC}_t$. User participation is 13% in the classification process. In the classification process, it is always desired that rules are classified automatically, and user participation is low.

The accuracy values generally increase in proportion to the $MinC_t$. Because higher the $MinC_t$, higher the user participation is. And higher user participation leads to learn a more powerful and predictive concept description.

Table 2. Results for IRIL

	$MinC_t$ 51%	$MinC_t$ 53%	$MinC_t$ 55%	$MinC_t$ 57%
Number of rules	1555	1555	1555	1555
Number of rules classified automatically with high certainty	1344	1286	1196	1096
User participation	13%	17%	23%	29%
Overall Accuracy	80%	82%	86%	88%

8 Conclusion

IRIL feature projection based, interactive rule interestingness learning algorithm was developed and gave promising experimental results on streaming classification rules induced on a financial distress domain. The concept description learned by the VFP algorithm, also developed in the framework of IRIL, constitutes a novel approach for interestingness analysis of classification rules. The concept description differs among the users analyzing the same domain. That is, IRIL determines the important rule interestingness factors for a given domain subjectively.

References

1. Fayyad, U., Shapiro, G., Smyth, P.: From data mining to knowledge discovery in databases. AI Magazine **17**(3) (1996) 37–54
2. Hilderman, R.J., Hamilton, H.J.: Knowledge discovery and interestingness measures: a survey. Technical Report, Department of Computer Science, University of Regina (1999)
3. Frawely, W.J., Piatetsky-Shapiro, G., Matheus, C.J.: Knowledge discovery in databases: an overview. Knowledge Discovery in Databases. AAAI/MIT Press (1991) 1–27
4. Major, J.A., Mangano, J.J.: Selecting among rules induced from a hurricane database. In: Proceedings of AAAI Workshop on Knowledge Discovery in Databases. (1993) 30–31
5. Piatetsky-Shapiro, G., Matheus, C.J.: The interestingness of deviations. In: Proceedings of AAAI Workshop on Knowledge Discovery in Databases. (1994) 25–36
6. Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., Verkamo, A.I.: Finding interesting rules from large sets of discovered association rules. In: Proceedings of the 3rd Int. Conf. on Information and Knowledge Management. (1994) 401–407
7. Liu, B., Hsu, W., Chen, S.: Using general impressions to analyze discovered classification rules. In: Proceedings of the 3rd Int. Conf. on KDD (1997) 31–36
8. Liu, B., Hsu, W.: Post-analysis of learned rules. AAAI (1996) 828–834
9. Hussain, F., Liu, H., Suzuki, E., Lu, H.: Exception rule mining with a relative interestingness measure. In: Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining. (2000) 86–97
10. Dong, G., Li, J.: Interestingness of discovered association rules in terms of neighborhood-based unexpectedness. In: Proceedings of the 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining. (1998) 72–86
11. Güvenir, H.A.: Benefit maximization in classification on feature projections. In: Proc. 3rd IASTED Int. Conf. on Artificial Intelligence and Applications (AIA 2003). (2003) 424–429