Associative Classification using Common Instances among Conflicting Discriminative Patterns

Kentaro Maeda[1] and Yoshitaka Kameya[1]

[1]Department of Information Engineering, Meijo University

***Abstract*—This paper proposes a white-box, associative classifier that uses discriminative patterns mined from a dataset including numeric values. In the proposed model, when there exist conflicting patterns for a test instance, we take into account the training instances covered in common by them as the "neighbors" of the test instance. By this design, we can accurately capture the space around the test instance, and as a result, it is observed in our experiments that the predictive performance is improved from some simpler methods. We also show another advantage of the proposed classifier by inspecting its interpretability/explanability.**

***Index Terms*—Associative Classifiers, Discriminative Patterns, Interpretability, Explanability**

I. Introduction

Frequently we wonder why or how our predictive model generates the outputs we have obtained when taking a machine learning approach. As an example, let us consider the case of automated loan approval in a bank. In such a case, the bank judges whether it is acceptable to lend money to a customer. With a black-box classifier, the customer would not be able to trust the result of the prediction even if it shows a good predictive performance, since the classifier does not show the reason for the prediction. In contrast, with an interpretable classifier, the customer would trust the prediction since he/she can tell the degree of credibility to his/her knowledge, and futhermore, customers way accept that the bank has decided not to lend money.

From the motivation above, in this paper, we aim to build a white-box, associative classifier [1], called the ECHO classifier, which uses discriminative patterns [2], [3] mined by an algorithm we call *E̲xhaustive C̲overing in H̲ybrid Do̲mains* (ECHO) [4]. Discriminative patterns exhibit the differences among classes and can be directly used as if-then rules from which users easily understand the reason for each prediction. One advantage of ECHO is that it can directly deal with a dataset including numeric values. There also exist methods that make black-box models, including deep neural networks, interpretable [5]. However, at present, developing such interpretation methods are still under study (e.g. [6]). Instead, we prefer to build a classifier without an extra tool for interpretation. In this paper, for simplicity, we focus on binary classification.

When making a prediction using discriminative patterns, the classes they predict often conflict, and the way of resolving such a conflict would severely affect the accuracy of classification. Some associative classifiers simply return the class predicted by the discriminative pattern with the highest performance metric [1]. However, such patterns cannot always capture the space around the test instance and often lead to erroneous prediction for test instances. For this reason, in this paper, we propose to take into account the training instances covered *in common* by conflicting patterns in prediction for a test instance. Such common instances work as the "neighbors" of the test instance, and therefore we can accurately capture the space around the test instance. In addition, we conducted some experiments that compare the proposed method with some simpler methods, and show that the predictive performance of the proposed method is improved from such simpler methods. We also inspect the interpretability/explanability of discriminative patterns used in the ECHO classifier and show its another advantage.

The rest of this paper is outlined as follows. First, we will describe ECHO in Section II. Next, we introduce the proposed classifier in Section III and show the results of our experiments in Section IV. Finally, we conclude this paper in Section V.

II. Background

In this section, we will give a brief description on ECHO. Please refer to [4] for details.

A. Datasets, Instances, Items, and Patterns

Let us start with introducing some notations and related concepts. Throughout the paper, we consider a database $\mathcal{D} = \{t_1, t_2, ..., t_N\}$ of size N, where each t_i is called an instance, and is a set of attribute-value pairs of the form $\langle A, v\rangle$, where A is called an attribute, and v is called a value. Here we have two types of attributes, i.e. symbolic attributes and numeric attributes. If an attribute A is symbolic, v is some symbolic value chosen from A's own domain, and if A is numeric, v is some real number. In classification and discriminative pattern mining, we assume that each t_i in $\mathcal{D}$ is associated with a class c_i, which is one of pre-defined classes $\mathcal{C}$.

On the other hand, we consider two types of items. A symbolic items takes the form $\langle A, v\rangle$, and we can equate it with an attribute-value pair in an instance. An interval item takes the form $\langle A, [u, v)\rangle$ and further has two types. A base interval item $\langle A, [v_i, v_{i+1})\rangle$ corresponds to one of n base intervals obtained from $n-1$ cut-points $\Delta_A = \{v_1, v_2, \ldots, v_{n-1}\}$ and two extreme points $v_0 = -\infty$ and $v_n = \infty$. An upper

Corresponding Author: Yoshitaka Kameya, ykameya@meijo-u.ac.jp

interval item is then defined as $\langle A, [v_i, v_j)\rangle$ where $0 \le i \le n$, $0 \le j \le n$, and $i < j$. We hereafter replace every attribute-value pair $\langle A, v_{\mathrm{raw}}\rangle$ in $\mathcal{D}$ in advance with a basic interval item $\langle A, [v_i, v_{i+1})\rangle$ where $v_i \le v_{\mathrm{raw}} < v_{i+1}$, and think of each instance t_i as a set of symbolic items and basic interval items. In a default setting, we *adaptively* decide the cut-points Δ_A for each attribute A based on previous work [7], [8].

A pattern is basically defined as a set of items. In a logical context, on the other hand, we sometimes consider each item as an atomic proposition, and each pattern $\boldsymbol{x}$ as a conjunction of items in it, i.e. $\bigwedge_{x \in \boldsymbol{x}} x$. One may see that interval items form a concept lattice. Indeed, for two interval items $x = \langle A, [u, v)\rangle$ and $y = \langle A, [u', v')\rangle$ where $u \le u'$ and $v' \le v$, x is more general than y, or x *subsumes* y, which is written as $x \succeq y$. This subsumption relation $x \succeq y$ also holds for two symbolic items $x = \langle A, u\rangle$ and $y = \langle A, v\rangle$ where $u = v$. We can finally consider the subsumption relation among patterns. That is, for two patterns $\boldsymbol{x}$ and $\boldsymbol{y}$, $\boldsymbol{x} \succeq \boldsymbol{y}$ holds when, for each $x \in \boldsymbol{x}$, there exists $y \in \boldsymbol{y}$ such that $x \succeq y$. As a special case, we say that a pattern $\boldsymbol{x}$ *covers* an instance t_i when $\boldsymbol{x} \succeq t_i$.

B. Classification Metrics

Let c be the class of our current interest. Then, discriminative patterns are the patterns that distinguish the instances in class c (*positive* instances), and the instances *not* in c (*negative* instances). One may directly use a discriminative pattern $\boldsymbol{x}$ for class c as a classification rule $\boldsymbol{x} \Rightarrow c$, which predicts class c for an instance covered by $\boldsymbol{x}$. Thus, in order to measure the quality of a discriminative pattern $\boldsymbol{x}$, we borrow the metrics often used in performance evaluation of a classifier. These metrics are computed based on the empirical joint probabilities $p(c, \boldsymbol{x}) = N(c, \boldsymbol{x})/N$ from the database, where $N(c, \boldsymbol{x}) = \#\{i \mid c_i = c, \boldsymbol{x} \succeq t_i, 1 \le i \le N\}$, i.e. the number of instances in class c covered by pattern $\boldsymbol{x}$. Here $\#S$ indicates the number of elements of set S.

Precision and recall are frequently used as classification metrics of a classifier. Using the notation above, confidence is written as $p(c \mid \boldsymbol{x})$ and positive support as $p(\boldsymbol{x} \mid c)$. We may call positive support simply "support." F-value is then defined as the harmonic mean of them: $\mathrm{F}_c(x) = 2p(c \mid \boldsymbol{x})p(\boldsymbol{x} \mid c)/(p(c \mid \boldsymbol{x}) + p(\boldsymbol{x} \mid c))$. A normalized version of χ^2-value is defined based on a binary relation between pattern $\boldsymbol{x}$ and class c: $\chi^2_c(\boldsymbol{x}) = \sum_{c' \in \{c, \neg c\}, \boldsymbol{x}' \in \{\boldsymbol{x}, \neg \boldsymbol{x}\}} \tau(c', \boldsymbol{x}')$, where

$$\tau(c', \boldsymbol{x}') = \frac{(p(c', \boldsymbol{x}') - p(c')p(\boldsymbol{x}'))^2}{p(c')p(\boldsymbol{x}')}. \quad (1)$$

Our implementation of ECHO allows us to specify the minimum confidence and the minimum positive support. From now on, however, we just consider intuitive settings, i.e. 0.5 and $1/N$, respectively.

C. Reducing Redundancy among Patterns

ECHO employs the closedness constraint [9] and the best-covering constraint [10] to reduce redundancy among patterns. The closedness constraint leaves only the most specific pattern among the patterns that cover the same set of positive instances. This helps ECHO to find more specific and hopefully more discriminative patterns. Inversely, when we disable the closedness constraint, the obtained patterns will become more general. The best-covering constraint says that each output pattern $\boldsymbol{x}$ must have some positive transaction covered by $\boldsymbol{x}$ with the highest classification metric. It is shown in [10] that the best-covering constraint is tighter than the productivity constraint [11], often used in discriminative pattern mining.

D. Exhaustive Covering

For finding meaningful discriminative patterns, ECHO performs a depth-first search with branch-and-bound pruning for patterns of higher quality, under the closedness constraint and the best-covering constraint. This search strategy is called *exhaustive covering* [10]. At the algorithmic level, we built ECHO on top of FP-growth [12], a standard frequent pattern mining algorithm, by extending FP-trees based on concept lattices over interval items. In the search process, appropriate interval items will be chosen from concept lattices, which means that discretization that considers multiple attributes simultaneously is conducted inside ECHO. Several conventional associative classifiers basically require discretization in advance [1]. Discretization in each individual attribute may lead to a considerable information loss, so the way of handling numeric values above is one notable advantage of ECHO. Another advantage also comes from its exhaustiveness. Rule-based classifiers such as decision trees can also deal with numeric values, but they are greedy algorithms relying on a heuristic function whose effect is often unclear. In contrast, the patterns found by ECHO have declarative meaning in terms of classification metrics and the constraints among patterns, which would lead to higher interpretability/explanability of associative classifiers using discriminative patterns.

Of course, the computational burden of exhaustive search is not ignorable in many real-world datasets. To mitigate this, ECHO adopts dynamic re-ordering [13] at branches in the depth-first search, and attribute-wise binning based on Jensen-Shannon divergence. By dynamic re-ordering, ECHO is controlled to find promising patterns earlier, and hence would output patterns of higher quality within a limited amount of time. In our implementation, we specify $T_{\max}$ as an upper limit of running time of ECHO. Similarly, in our attribute-wise binning, we specify $I_{\max}$ as an upper limit of the number of base intervals for each attribute. With smaller $I_{\max}$, we have only to consider fewer base intervals, and thus the search space can be drastically reduced. In this binning, we merge base intervals so as to maximize the difference between the probability distributions of positive instances and negative instances. The difference is measured by Jensen-Shannon divergence, and the merge operation is conducted in a dynamic programming fashion similarly to optimal histogram construction [14]. It should be noted here that, in this binning, we just intend to limit the number of base intervals, and not to find an optimal set of base intervals for each attribute, like

[15]. Rather, as said above, appropriate interval items will be chosen in the later search process of ECHO.

III. Proposed Method

Here we describe our proposed method, the ECHO classifier, which predicts the class of a test instance using discriminative patterns obtained by ECHO. In most cases, there are some patterns that cover the test instance, and we need to decide which patterns are used for prediction. A simple approach to exploiting discriminative patterns is that we rank the patterns according to a performance metric (e.g. F-value or χ^2-value). However, this method may not be suitable when some discriminative patterns overly cover many instances, and they tend to be selected as the best pattern even for an almost irrelevant instance. This method can also be problematic since it relies on a single rule. Therefore in this study, we propose a method that exploits multiple, conflicting discriminative patterns rules using the training instances covered in common by them. Then, we predict a class based on a weighted class ratio in the common instances.

To be more specific, we first consider a set $\mathcal{X}$ of discriminative patterns obtained by ECHO. We then pick up the subset $\mathcal{X}^*$ from $\mathcal{X}$ that covers the test instance of interest. Note here that the classes predicted by the patterns in $\mathcal{X}^*$ can be conflicting. Also we introduce a set $I_{\boldsymbol{x}}$ of training instances covered by a pattern $\boldsymbol{x}$, and a set of common training instances $I^* = \bigcap_{\boldsymbol{x} \in \mathcal{X}^*} I_{\boldsymbol{x}}$ covered by the patterns in $\mathcal{X}^*$. Finally, we predict c^* as a class for the test instance, where

$$c^* = \operatorname*{argmax}_c \frac{1}{p(c)} p\left(c \,\middle|\, \bigwedge_{\boldsymbol{x} \in \mathcal{X}^*} \boldsymbol{x}\right) \tag{2}$$

$$p\left(c \,\middle|\, \bigwedge_{\boldsymbol{x} \in \mathcal{X}^*} \boldsymbol{x}\right) = \frac{\#\{i \mid t_i \in I^*, c_i = c, 1 \le i \le N\}}{\# I^*}. \tag{3}$$

Here we can interpret $p(c \mid \bigwedge_{\boldsymbol{x} \in \mathcal{X}^*} \boldsymbol{x})$ as the estimated class distribution inside the space limited by $\bigwedge_{\boldsymbol{x} \in \mathcal{X}^*} \boldsymbol{x}$ around the test instance, and $1/p(c)$ is the weight that cancels the influence from the imbalance in the entire class distribution.

The outline of the prediction procedure is as follows:

```
1 Load the discriminative patterns that have been found
  *with* the closedness constraint.
2 Predict a class according to the number of patterns
  covering a test instance:
    – 0 → Go to Step 3.
    – 1 → Return the class of the covering pattern.
    – 2 or more → Is there any common training
      instance covered by the patterns? (Case 1)
        ∗ Yes → Return the class predicted by Eq. 2.
        ∗ No → Go to Step 3.
3 Load the discriminative patterns that have been found
  *without* the closeness constraint.
4 Predict a class according to the number of patterns
  covering a test instance:
    – 0 → Return the majority class.
    – 1 → Return the class of the covering pattern.
    – 2 or more → Is there any common training
      instance covered by the patterns? (Case 1)
        ∗ Yes → Return the class predicted by Eq. 2.
        ∗ No → Return the prediction by $k$-NN.
          (Case 2)
```

Note here that we refer to discriminative patterns *with* the closedness constraint first, and then refer to those *without* the closedness constraint. This is because, as mentioned in Section II-C, we tend to have more specific and more discriminative patterns under the closedness constraint. In addition, in Case 1, we need to consider common instances only when the classes predicted by the patterns covering the test instance are conflicting. Moreover, in Case 2, we call the k-nearest neighbor classifier (k-NN) when there is no common instance covered by conflicting discriminative patterns. In the cases other than Case 2, we can predict the class of a test instance in an interpretable way using discriminative patterns. k-NN is primarily adopted for guaranteeing the predictive performance, but we can still think of k-NN methods as interpretable since it is possible to present (a summary of) k-neighbor instances as the reason for the prediction [16].

IV. Experimental Results

In this section, we report the results of our experiments. First, we describe the experimental settings, and review the other classifiers in comparison. Second, we show some results on predictive performance. Lastly, we make a simple analysis on the interpretability/explanability of the proposed method.

A. Experimental Settings

1) Datasets: Datasets were selected from UCI Machine Learning Repository. All datasets are provided primarily for binary classification. We show the datasets below, where the name of each dataset is followed by the number of positive instances and the number of negative instances.

- Both symbolic and numeric attributes contained (hybrid):
  - credit-german (bad : good = 300 : 700)
  - heart-cleveland ('>50_1' : '<50' = 138 : 165)
  - hepatitis (DIE : LIVE = 32 : 123)
  - sick (sick : negative = 231 : 3541)
- Only continuous attributes contained (numeric):
  - breast-w (malignant : benign = 241 : 458)
  - diabetes (test_positive : test_negative = 268 : 500)
  - ionosphere (b : g = 126 : 225)
- Only discrete attributes contained (symbolic):
  - kr-vs-kp (nowin : won = 1527 : 1669)
  - mushroom (e : p = 4062 : 4062)
  - vote (republican : democrat = 168 : 267)

The dataset files are converted from the ARFF files provided in Weka's website (https://www.cs.waikato.ac.nz/ml/weka/).

TABLE I
EXPERIMENTAL RESULTS ON CLASSIFICATION

Classifier				Dataset										Average
#	Settings			Hybrid				Numeric			Symbolic			
	Method	$T_{\max}$	$I_{\max}$	credit-g	heart-c	hepatitis	sick	breast-w	diabetes	ionosphere	kr-vs-kp	mushroom	vote	
1	top-ranker	1	unlimited	0.498	0.726	0.552	0.282	0.903	*0.664	0.894	0.866	*0.913	0.932	0.723
2	no-k-NN	1	unlimited	0.537	0.779	***0.581**	0.297	0.924	0.653	0.858	*0.915	0.881	0.932	0.736
3	full	1	unlimited	0.540	0.787	0.516	0.679	*0.929	0.644	0.863	*0.915	0.887	0.933	0.769
4	full	3	unlimited	0.559	0.797	0.500	0.703	0.926	0.647	0.870	*0.915	0.887	*0.942	0.775
5	full	6	unlimited	*0.571	*0.799	0.525	0.721	0.926	0.647	0.876	*0.915	0.887	0.938	0.781
6	full	1	20	0.555	0.797	0.484	***0.778**	0.926	0.637	*0.899	*0.915	0.887	0.939	*0.782
7	full	1	50	0.559	0.793	0.492	0.765	0.926	0.642	0.870	*0.915	0.887	0.933	0.778
Random Forests				**0.594**	**0.807**	0.512	0.721	**0.956**	**0.695**	**0.909**	0.959	**0.950**	**0.948**	**0.805**
Decision Trees				0.582	0.773	0.460	0.736	0.895	0.668	0.841	**0.963**	**0.950**	0.935	0.780

2) Settings for ECHO: We implemented the ECHO classifier in a scikit-learn (https://scikit-learn.org/) compatible fashion, in order to easily compare with existing classifiers with standard evaluation methods. The learning procedure (i.e. the "fit" function) includes the routine that finds the discriminative patterns by ECHO. As the prediction procedure (i.e. the "predict" function), on the other hand, the procedure described in Section III was implemented.

For rigorous comparison, we introduced grid search and stratified cross-validation [17]. Grid search is one of the standard hyper-parameter tuning methods, finds the best combination of hyper-parameters from all combinations. In stratified cross-validation, the number of folds is set as 10. The performance metric we use is F-value for the minority class. This is because, in general, the problem of identifying the minority class is more difficult and important. Grid search was not performed for ECHO since ECHO has only a few hyper-parameters to be tuned. This is another advantage of using ECHO, as more complex models can be more difficult to tune their hyper-parameters. We also conducted a grid search for tuning k-NN, whose hyper-parameter is the number of neighbors $k \in \{1, 3, 5, 9, 19, 29\}$. The candidates for k were chosen as odd for avoiding ties. Furthermore, in the grid search, 10-fold stratified cross-validation is applied to k-NN. The processor we used is Core i7-8700 (3.2GHz).

The ECHO classifiers use two hyper-parameters $T_{\max}$ and $I_{\max}$ defined in Section II-D, and they were configured in two ways. One way is that we vary $T_{\max}$ using 1, 3, and 6 hours with $I_{\max}$ being unlimited. Another way is that we vary $I_{\max}$ using 20 and 50 with $T_{\max}$ being 1 hour. Based on a preliminary experiment, we chose χ^2-value as the classfication metric in finding discriminative patterns by ECHO.

To see the effect of the operations in the ECHO classifiers described in Section III, we consider two simplified variants. The first variant, called the "top-ranker" variant, just ranks discriminative patterns according to χ^2-value *without* using common instances covered by conflicting patterns. The second one, the "no-k-NN" variant, *does* use such common instances, but returns the majority class without using k-NN.

3) Settings for Conventional Classifiers: We also chose random forests and decision trees as conventional classification models in comparison. Random forests are an ensemble model known for its high predictive accuracy thanks to the majority decision of multiple trees. We performed imputation for missing values and one-hot encoding for symbolic attributes as preprocessing. Missing values for an attribute are imputed by randomly generated values from the other instances having observed values for the attribute.

In hyper-parameter tuning of these conventional classifiers, we conducted grid search. In learning random forests, we tuned three hyper-parameters: the number of trees to be generated ('n_estimators'), the minimum number of supports each leaf holds ('min_samples_leaf'), and whether to consider class weights ('class_weight'). The candidates of hyper-parameters are as follows.

'n_estimators': 50, 100, 200, 500, and 1000
'min_samples_leaf': 1, 2, 5, 10, and 20
'class_weight':
"balanced according to the class distribution" and "uniform"

In learning decision trees, we conducted grid search for 'min_samples_leaf' and 'class_weight' over the same candidates as those in random forests. We run 10-fold stratified cross-validation using the tuned hyper-parameters. Similarly to the ECHO classifiers, we evaluated the conventional classifiers by F-value for the minority class. Random forests and decision trees we use are implemented in scikit-learn.

B. Results

We show the results in Table I, where the names of some datasets are abbreviated due to the space limit. Also, for easy reference, the variants of the ECHO classifiers are numbered as in the first column of the table. For each dataset, the best performance metric among all classifiers is written in bold. In addition, the best performance among the ECHO classifiers (#1–7) is annotated with the '$*$' symbol.

First of all, we compare the "top-ranker" variant (#1) and the "no-k-NN" variant (#2). The "no-k-NN" variant has a higher average F-value than the "top-ranker" variant. However, the F-value for the sick dataset remains low. This result implies that the stability of these simplified variants seems low. Next, we compare the "no-k-NN" variant (#2) and the full ECHO classifier (#3). The full ECHO classifier has a higher average F-value than the "no-k-NN" variant. In particular, in

TABLE II
STATISTICS ON DISCRIMINATIVE PATTERNS

Classifier	Statistics	Closedness constraint	Dataset									
			Hybrid				Numeric			Symbolic		
			credit-g	heart-c	hepatitis	sick	breast-w	diabetes	ionosphere	kr-vs-kp	mushroom	vote
ECHO classifier	Avg. number of patterns	Yes	82	32	23	57	18	70	23	11	10	11
		No	69	31	27	50	24	68	20	11	12	11
	Avg. pattern length	Yes	4.02	5.36	4.62	3.04	6.01	3.71	6.83	5.65	4.44	1.19
		No	3.11	3.95	4.08	2.16	4.93	3.20	2.73	3.97	2.22	1.19
Decision Trees	Number of leaves		31	18	10	61	19	23	9	34	12	13
	Maximum depth		10	6	5	30	7	7	5	11	6	5

TABLE III
A PART OF DISCRIMINATIVE PATTERNS OBTAINED BY ECHO FOR THE CREDIT-GERMAN DATASET

Test instance (class = ‘bad’)	age = 28, checking_status = ‘0 $\leq$X<200’, credit_amount = 5234, duration = 30, housing = ‘own’ purpose = ‘new_car’, other_payment_plans = ‘none’, savings_status = ‘<100’, ...
Covering patterns for ‘bad’	$15.50 \leq$ duration < 66.00, savings_status = ‘<100’
	$19.50 \leq$ age < 29.50, $653.0 \leq$ credit_amount < 15270, $8.500 \leq$ duration < 66.00, purpose = ‘new_car’
	checking_status = ‘0 $\leq$X<200’, $14.50 \leq$ duration
Covering patterns for ‘good’	$23.50 \leq$ age, credit_amount < 7973, duration < 46.00, housing = ‘own’, other_payment_plans = ‘none’
	$23.50 \leq$ age, credit_amount < 7845, duration < 46.00, other_payment_plans = ‘none’
	$26.50 \leq$ age, credit_amount < 12860, duration < 66.00, housing = ‘own’

the sick dataset the full ECHO classifier showed a significant improvement in average F-value.

There are also several results with other settings for the full ECHO classifier. First, from some cases in which the upper limit of running time ($T_{\max}$) for finding discriminative patterns for each class is raised (#3, 4, and 5), one may find that the performance is improved as the quality of patterns gets higher. When the upper limit is 6 hours, it is higher than decision trees on average. It should also be noted that ECHO's exhaustive search for discriminative patterns finished within one hour in hepatitis, breast-w, kr-vs-kp, mushroom, and vote, and thus there was only little difference among cases #3, 4, and 5.[1] Next, we see some cases in which we change the upper limit of the number of intervals ($I_{\max}$) for numeric attributes in ECHO (#3, 6, and 7). From these cases, we see that performance is improved by limiting the number of base intervals to some extent. In particular, we find that F-value was improved for the sick dataset. With the upper limit being set as 20, the ECHO classifier shows a better performance than decision trees.

C. Discussion

As said before, the ECHO classifier is expected to have advantages as a white-box classifier. In this section, we discuss the interpretability/explanability of the ECHO classifier.

1) Statistics on Discriminative Patterns: We first confirm whether discriminative patterns are interpretable. For that, we collect two statistics on discriminative patterns. One statistic is the average number of discriminative patterns obtained by ECHO for one class, in 10-folds stratified cross-validation. We consider two cases where the closedness constraint is enabled and disabled. Another statistic is the average of lengths of patterns and we present them in Table II. In Table III, we show a part of discriminative patterns in the credit-german dataset. In this example, the length of the first pattern is 2, that of the second pattern is 4, and so on. Then, we take their average over 10-folds. We also consider the cases where the closedness constraint is enabled and disabled.

Table II shows the average number of discriminative patterns and the average of pattern lengths observed in our experiments. The ECHO classifier uses the same setting as that of #5 in Table I. In Table II, we also show the number of leaves and the maximum depth in decision trees. One may see that the number of patterns is considerably large in Table II, since ECHO searches for discriminative patterns exhaustively. This tendency can also be observed in the datasets where it takes a long time for finding discriminative patterns. In contrast, each pattern short and intuitive, as illustrated in Table III, where each interval item $\langle A, [v_i, v_j)\rangle$ is expressed as $v_i \leq A < v_j$. We can interpret patterns thanks to this shortness of patterns. Decision trees are understandable at a glance if they are small, but they can easily be too complicated to write down.

2) Tracing the Classification Process: Next we show an example of how the ECHO classifier predicts the class of a test instance. The ECHO classifier finds 40 patterns for positive class ‘bad’ and 43 patterns for negative class ‘good’ using the credit-german dataset (Step 1 in Section III). Among those patterns, there are 17 patterns for ‘bad’ and 27 patterns for ‘good’ covering the test instance shown in Table III, and Table III also shows a part of patterns covering the test instance. We then extract the common instance covered by these covering patterns (Step 2-2). In this example, the number of the common training instances for ‘bad’ is 1 and

[1]Some variation in F-value arose due to the test instances to which we applied k-NN.

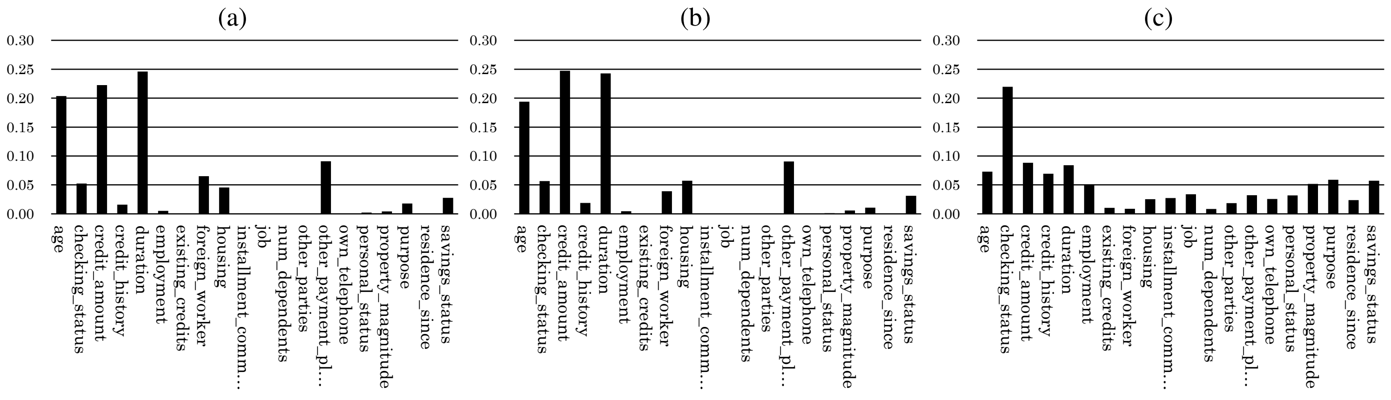

Fig. 1. (a) Relative frequency of attributes in ECHO *with* the closedness constraint. (b) Relative frequency of attributes in ECHO *without* the closedness constraint. (c) Importance of attributes in random forests.

for 'good' is 0. In this way, we can make an *instance-specific*, detailed explanation of how the ECHO classifier works in the classification process.

3) Usage of Attributes in Discriminative Patterns: Additionally, we may make a *model-specific* explanation by inspecting usage of attributes in discriminative patterns. That is, in the ECHO classifier, we counted the attributes appearing in discriminative patterns. For example, in Table III, 'duration' appears more frequently for six times and 'age' for four times. We also consider two cases where the closedness constraint is enabled and disabled. Here random forests are considered as a target for comparison. The scikit-learn implementation of random forests is able to compute the importance of each attribute. The target dataset is the credit-german dataset. The settings for ECHO are the same as the ones in Section IV-C1. We show the relative frequency of each attributes in the patterns obtained by ECHO in Figs. 1 (a) and (b) and the importance of attributes in random forests presented in Fig. 1 (c). Fig 1 (a) (resp. Fig. 1 (b)) shows the results when the closedness constraint is enabled (resp. disabled).

First, by comparing Figs. 1 (a) and (b), we have almost the same relative frequencies. In the results related to ECHO (Fig. 1 (a) or (b)), it is seen that 'age,' 'credit_amount,' and 'duration' are quite frequently used. Besides, a comparison between Fig. 1 (a) (or Fig. 1 (b)) and Fig. 1 (c), one may see that the ECHO classifier only uses fewer attributes in prediction than random forests. Note here that, at present, random forests give us no simple way for making instance-specific explanations, like the one described in Section IV-C2.

V. Conclusion

In this paper, we proposed a white-box, associative classifier that uses the training instances covered in common by conflicting discriminative patterns. We also introduced k-NN for the test instance not covered by interpretable discriminative patterns in order to guarantee the accuracy of prediction. Using this classifier, we achieved at least the same performance metric as that of decision trees. We also investigated the interpretability/explanability of the proposed classifier. Currently, we are planning to shorten the running time of ECHO by parallelization, and to compare the proposed classifier with other associative classifiers such as CBA [18].

References

[1] F. Thabtah, "A review of associative classification mining," *Knowledge Engineering Review*, vol. 22, no. 1, pp. 37–65, 2007.

[2] G. Dong and J. Bailey, Eds., *Contrast Data Mining: Concepts, Algorithms, and Applications*. CRC Press, 2012.

[3] P. Kralj Novak, N. Lavrač, and G. I. Webb, "Supervised descriptive rule discovery: a unifying survey of contrast set, emerging pattern and subgroup mining," *J. of Machine Learning Research*, vol. 10, pp. 377–403, 2009.

[4] Y. Kameya, "Towards efficient discriminative pattern mining in hybrid domains," in *Proc. of the 17th Forum on Information Technology (FIT-18)*, 2018, an English version available as arXiv:1908.06801.

[5] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Computing Surveys*, vol. 51, no. 5, pp. 93:1–93:42, 2019.

[6] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," in *Proc. of the 32th Annual Conf. on Neural Information Processing Systems (NeurIPS-18)*, 2018.

[7] S. Brin, R. Rastogi, and K. Shim, "Mining optimized gain rules for numeric attributes," *IEEE Trans. on Knowledge and Data Engineering*, vol. 15, no. 2, pp. 324–338, 2003.

[8] H. Grosskreutz and S. Rüping, "On subgroup discovery in numerical domains," *Data Mining and Knowledge Discovery*, vol. 19, no. 2, pp. 210–226, 2009.

[9] N. Pasquier, Y. Bastide, R. Taouli, and L. Lakhal, "Discovering frequent closed itemsets for association rules," in *Proc. of ICDT-99*, 1999, pp. 398–416.

[10] Y. Kameya, "An exhaustive covering approach to parameter-free mining of non-redundant discriminative itemsets," in *Proc. of DaWaK-16*, 2016, pp. 143–159.

[11] R. Bayardo, R. Agrawal, and D. Gunopulos, "Constraint-based rule mining in large, dense databases," *Data Mining and Knowledge Discovery*, vol. 4, pp. 217–240, 2000.

[12] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proc. of SIGMOD-00*, 2000, pp. 1–12.

[13] Y. Kameya and K. Ito, "Dynamic re-ordering in mining top-k productive discriminative patterns," in *Proc. of TAAI-17*, 2017, pp. 172–177.

[14] P. Kontkanen and P. Myllymäki, "MDL histogram density estimation," in *Proc. of AISTATS-07*, 2007, pp. 219–226.

[15] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *Proc. of IJCAI-93*, 1993, pp. 1022–1029.

[16] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems: An Introduction*. Cambridge University Press, 2010.

[17] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.

[18] B. Liu, W. Hsu, and Y. Ma, "Integrating classification and association rule mining," in *Proc. of KDD-98*, 1998, pp. 80–86.