Knowl Inf Syst (2015) 45:731–749
DOI 10.1007/s10115-014-0817-0

REGULAR PAPER

# Mining sequential patterns for classification

**Dmitriy Fradkin · Fabian Mörchen**

Received: 15 May 2014 / Revised: 8 October 2014 / Accepted: 26 December 2014 /
Published online: 3 January 2015

**Abstract** While a number of efficient sequential pattern mining algorithms were developed over the years, they can still take a long time and produce a huge number of patterns, many of which are redundant. These properties are especially frustrating when the goal of pattern mining is to find patterns for use as features in classification problems. In this paper, we describe BIDE-Discriminative, a modification of BIDE that uses class information for direct mining of predictive sequential patterns. We then perform an extensive evaluation on nine real-life datasets of the different ways in which the basic BIDE-Discriminative can be used in real multi-class classification problems, including 1-versus-rest and model-based search tree approaches. The results of our experiments show that 1-versus-rest provides an efficient solution with good classification performance.

**Keywords** Sequential pattern mining · Sequence classification · Information gain

## 1 Introduction

Temporal data mining exploits temporal information in data sources in the context of data mining tasks such as clustering or classification. Many scientific and business data sources are dynamic and thus promising candidates for application of temporal mining methods. For an overview of methods to mine time series, sequence, and streaming data, see [13,20].

One particular type of temporal data consists of sequences of (sets of) discrete items associated with time stamps. Examples of such data include medical records [4,5], histories of transactions of customers in an online shop, log messages emitted by machines or telecommunication equipment during operation [43,49], and discretized or abstracted time series or sensor data. A common task is to mine for local regularities in this data by looking for

D. Fradkin (✉)
Siemens Corporate Technology, 755 College Rd East, Princeton, NJ 08540, USA
e-mail: dmitriy.fradkin@siemens.com

F. Mörchen
Amazon, Seattle, WA, USA

sequential patterns [2] that represent a sequence of itemsets, possibly with gaps, embedded in the observation sequences. Another common task is sequence classification—given a set of example sequences belonging to two or more classes, to learn a predictive model capable of correctly assigning class labels to previously unseen sequences. In this paper, we

- propose BIDE-Discriminative—a modification to the highly efficient BIDE algorithm [46] for direct predictive pattern mining of sequential data that rely on pruning statistical measures of feature predictiveness.
- propose approaches (**BIDE-D** and **BIDE-DC**) for using BIDE-Discriminative for direct predictive pattern mining in multi-class problems.
- propose a combination of BIDE-Discriminative and model-based tree (MbT) classifier [15] for creating a sequence tree-based classifier (**SMBT**) or for an alternative direct pattern mining approach (**SMBT-FS**).
- conduct an extensive experimental evaluation of all of the proposed methods against regular BIDE as well as an obvious alternative of separately mining patterns for each class (**BIDE-C**). Our experiments show that the proposed methods are faster and produce fewer patterns than indirect methods and lead to at least comparable accuracy. They also indicate possible trade-offs in mining speed and accuracy between different methods.

## 2 Related work

Two types of approaches for classification on itemset and symbolic sequential data have been proposed. These can be thought of as *indirect* and *direct* approaches [11]. The indirect approaches consist of three stages. First, candidate pattern is mined in an unsupervised fashion or separately for each class, and then feature selection [19] is applied, usually using criteria such as information gain (IG) or $\chi^2$ [51]. Finally, a classifier such as SVM is built on the remaining features. Many papers utilize such indirect approaches for classification, e.g., [8,10].

The drawbacks of indirect approaches are clear and have been noted in multiple publications [11]: (i) Generation of all candidate patterns can be computationally very expensive, and (ii) the overwhelming majority of the patterns thus found turn out to be useless for classification or interpretation. Thus, feature selection is still necessary as an additional step before classification to avoid loss in predictive performance. Note that utilizing more restrictive definitions of patterns, such as closed or margin-closed itemset [32,34] or sequential [17,46] patterns, or reducing the number of patterns by other means [6,27], ameliorates to some extent the first of these problems, but does not address the second.

Direct methods address both of these problems by utilizing class label information in the pattern mining stage, and possibly combining/interleaving the steps of pattern mining and pattern evaluation, thereby generating fewer but better patterns. Furthermore, they can be faster due to pruning of the pattern search space.

### 2.1 Classification with itemset data

Multiple indirect pattern mining algorithms exist for itemset data, staring with Apriori [1]. Examples include CHARM [53], FP-tree [21], and FPclose [18]. Any of these can be used to generate features for classification.

In [10], a minimum support threshold is set based on information gain bounds for mining discriminative itemsets. FPclose is used to find all itemsets with support greater than this threshold. Maximal Marginal Relevance [9] is then applied to further reduce the set of itemsets to be used as features. SVM and decision trees are used as classifiers. This approach can be seen as falling between indirect and direct approaches.

There have also been some real direct approaches. In [11], a branch-and-bound search is performed for discriminative itemset patterns. These patterns are generated sequentially on a shrinking FP-tree [22] by eliminating training instances that are sufficiently covered, as a way to reduce redundancy in the set. This requires multiple runs of the mining algorithm to generate one pattern at a time. Model-based search tree ($M^bT$) approach is proposed in [15] and evaluated on itemset and on graph datasets. Here, a single most discriminative pattern is selected, and the dataset is split into two, based on whether instances have this pattern. Each set is then recursively processed, until all instances in a set belong to the same class, or the set is too small. In this way, a tree for classifying unseen examples is constructed.

None of these approaches have been evaluated on sequential data, which is what we do in the present paper. Our approaches SBMT and SMBT-FS are analogous to those of [11] except for sequential patterns, but we also consider mining top $k$ informative patterns in a single run (BIDE-D and BIDE-DC approaches).

## 2.2 Classification with sequential data

Just as for itemsets, there are many indirect sequential pattern mining methods. GSP [44], SPADE [52], PrefixSpan [41], GSpan [50], and BIDE [46] are just some of them. Examples of their use can be found in [8] and [31]. In [8], time series are converted into symbolic sequences using Symbolic Aggregate Approximation [29]. Sequential patterns (motifs) are then mined using GSP-/Apriori-like approach with taxonomy. These patterns (motifs) are used as features for construction of SVMs and Bayesian Networks. In [31], sequential patterns in software traces are mined using a custom algorithm, then feature selection based on Fisher score is applied, and finally, a classifier is applied.

We focus below on more direct approaches.

Text and biologic sequences can be considered sequential data, so we mention an approach that was described in [23,24]. Logistic regression models are built in the space of all possible n-grams (of words or symbols) by directly constructing predictive n-grams. Gaps of predefined maximum size (i.e., n-grams with wildcards) can also be handled. This approach can therefore be thought of as discovering predictive sequential patterns, though it is quite different from other sequential pattern mining work.

In [28], the approach of [10] (discussed above in Sect. 2.1) was used to set a minimum support threshold based on information gain bound for mining sequential patterns. Partial sequential patterns, i.e., of limited length, were mined, and then feature selection using F-score was performed before applying classification methods. This approach is thus more of an indirect than a direct method.

In [30], dyadic sequential patterns are mined using a BIDE-like [46] algorithm. The instances are pairs of sequences, and the task is to determine whether elements in a pair match or not. While the high-level idea is similar to BIDE-Discriminative that we propose here, the data, the patterns, and the problem are different.

In [7], multiple pattern languages, including sequences, trees, and graphs, were compared for their usefulness in a classification task. The patterns were mined using branch-and-bound methods like those described in Sect. 2.3 with a modification of GSpan [50] (no details provided) and $\chi^2$ as the discriminative measure. In our work, we focus on evaluating alternative approaches to finding discriminative sequential patterns, rather than evaluating different pattern languages. Also, we base our discriminative pattern mining algorithm on BIDE, rather than on GSpan.

## 2.3 Branch-and-bound methods

How can discriminative patterns be discovered without examining all the candidates? The solution, utilized in some form in all of the above-mentioned direct approaches, is to use branch-and-bound search in the space of patterns while maintaining an upper bound on the utility of the search subtree at a pattern.

Pruning based on statistical metrics such as IG or $\chi^2$ was proposed in [37] for finding predictive itemsets. An Apriori-type algorithm was used to traverse the search space, and upper bound on the feature quality was used for pruning. In [39], such approach was used for discriminative subgraph mining, in [30]—for classification of dyadic sequences.

Relation between a pattern's support and discriminative power was discussed in [10], but only in the context of setting appropriate minimum support for frequent itemset pattern mining.

## 3 Upper bounds on discriminative measures

Suppose we have a dataset $D$ of labeled examples $(x_i, y_i), i = 1, \dots, N$ where $y_i \in C$ and $C$ is a set of $k$ class labels: $C = \{c_j\}, j = 1, \dots, k$. Let $s_j(P)$ be support of pattern $P$ in class $c_j$—in other words, it is the number of examples of class $c_j$ that contains/matches pattern $P$. Also, let $s(P)$ be the total support of $P$—the sum of support over all classes $c \in C$. Table 1 shows the relationships between these values.

Information gain (IG) has long been used for feature selection in classification [51]. For convenience, let $l(p) = p\log_2(p)$. Then, IG is defined (for a fixed dataset) as:

$$\begin{aligned} IG(P) = \sum_{c\in C} l\left(\frac{|c|}{N}\right) - \frac{s(P)}{N}\sum_{c\in C} l\left(\frac{s_c(P)}{s(P)}\right) \\ - \frac{N-s(P)}{N}\sum_{c\in C} l\left(\frac{|c|-s_c(P)}{N-s(P)}\right) \end{aligned} \quad (1)$$

Note that the first term in Eq. 1 is fixed for any given dataset, since it depends only on class distribution. Therefore, for a fixed dataset $IG(P)$ is a function of conditional distribution $p(c|x)$ which can be fully represented by a vector of class supports $s_c(P)$. We can write $IG(P) = IG(s_1(P), \dots, s_k(P))$.

For a two-class problem, IG is a convex function. Morishita and Sese [37] have come up with an upper bound on it:

$$IG_{ub}(r, q) = max(IG(r, 0), IG(0, q)) \quad (2)$$

where $r = s_+(P)$ and $q = s_-(P)$.

Analysis in [37] shows that for two-class problems other discriminative measures such as chi-square and correlation can also be bounded in a similar fashion. For multi-class problems, bounds (in the same form as above) have been found for chi-squared [38] and inter-class

**Table 1** Co-occurrence matrix for pattern $P$ and classes $C$

| | Class $c_1$ | $\cdots$ | Class $c_k$ | Total |
|---|---|---|---|---|
| $P$ present | $s_1(P)$ | $\cdots$ | $s_k(P)$ | $s(P)$ |
| $P$ absent | $\|c_1\| - s_1(P)$ | $\cdots$ | $\|c_k\| - s_k(P)$ | $N - s(P)$ |
| | $\|c_1\|$ | $\cdots$ | $\|c_k\|$ | $N$ |

variance [42]. Both papers used these bounds in analysis of itemset data. Note, however, that these results apply to any pattern language, such as itemsets, sequential patterns, and graphs.

We are not aware of similar upper bounds for IG in multi-class problems. Theorem 7.4 in [12] states that mutual information (aka IG) $I(X, C)$ is a concave function of $p(x)$ for a fixed $p(c|x)$, and a concave function of $p(c|x)$ for a fixed $p(x)$. However, in our case neither one is fixed (extending a pattern changes both), and so IG appears to be neither convex nor concave for more than 2 classes.

Let us state clearly what existence of such upper bounds implies. Given a labeled dataset, patterns $P$ and $P'$ such that $P$ occurs in $P'$ (or $P'$ matches or extends $P$—we will define this formally in Sect. 2.3) and a discriminativeness measure F with a known upper bound, if $F_{ub}(P) = x$, then $F(P') \leq x$. This enables construction of branch-and-bound algorithms for pattern mining, such as those we mentioned in Sect. 2.3 and those we present in this paper, using any discriminative measure with a defined upper bound. Basically, if we already observe a pattern with discriminative score $x$, and $F_{ub}(P) = y < x$, then we know that no extension of $P$ is going to have a higher discriminative score, and this part of pattern space does not need to be explored. The discriminative measure itself does not need to be (and usually is not) anti-monotone.

In this paper, we focus on IG, while noting that the same methods are applicable to chi-square, correlation, and other discriminativeness measures where an upper bound can be formulated.

## 4 BIDE Family of algorithms

### 4.1 Definitions

An **event sequence** over a set of events $\Sigma$ is a sequence of pairs $(t_i, s_i)$ of **event sets** $s_i \subseteq \Sigma, \forall i = 1, \ldots, n$ and time stamps $t_i \in \Re^+$. The ordering is based on time, i.e., $\forall i < j : t_i \leq t_j$. The **length** of the event sequence is $n$.

For the present discussion (and in much of the sequential pattern mining literature), the exact values of the time stamps are not as important as the ordering that they impose. We will therefore treat event sequences as just an ordered set of event sets $S = \{s_i\}$.

A **sequence database**, SDB, of size $N$ is a collection of event sequences $S_i$, $i = 1, \ldots, N$.

A sequence $S = \{s_i\}$, $i = 1, \ldots, k$ **matches** a sequential pattern $P = \{p_j\}$, $j = 1, \ldots, m$ (or a pattern $P$ **occurs** in the sequence $S$) iff $\exists i_1, \ldots, i_m$ with $p_j \subseteq s_{i_j}$ for $j = 1, \ldots, m$, such that $\forall 1 \leq j, k \leq m$: $p_j \prec p_k$ implies $i_j < i_k$. We will denote such a **match** by $m_{i_1, i_m}(P, S)$. A match $m_{i_1, i_m}(P, S)$ is the **earliest match** iff for any other match $m_{j_1, j_m}(P, S) i_k \leq j_k, \forall k = 1, \ldots, m$,

Support of a pattern, support$(P)$, is the number of sequences where the pattern occurs. A pattern is **frequent** if its support is above some predefined minimum support $\mu$. A pattern is **closed** if none of the patterns that include it have the same support.

In order to describe the algorithms in the following sections, we need to introduce the notion of **projected databases**, which is extremely useful in constructing efficient algorithms for sequential pattern mining.

Given a pattern $P = \{p_i\}, i = 1, \ldots, |P|$ and a sequence $S = \{s_j\}, j = 1, \ldots, |S|$, with the earliest match $m_{k_1, k_m}(P, S)$, a projection of $S$ on $P$ results in a **projected sequence** $S|P = \{s_t\}$, where $t = k_m + 1, \ldots, |S|$. We refer to $k_m$ as an offset.

Given a pattern $P = \{p_i\}, i = 1, \ldots, |P|$ and an SDB $D = \{S_j\}, j = 1, \ldots, |D|$, a projection of $D$ on $P$ is a **projected database** $D|P$, consisting of projected sequences $S_j|P$, obtained by projecting $S_j$ onto $P$. Note that if a sequence does not match a pattern, it does not appear in the projected database. Projected database $D|P$ can be efficiently represented with a list of pairs of indices $(j, t)$, where $j$ refers to $S_j|P$ and $t$ is the corresponding offset.

## 4.2 BIDE

Details of BIDE implementation can be found in [17,46,47]. BIDE is initially called with the full sequential database $D$, minimum support $\mu$, and an empty pattern $P = \emptyset$. It returns a list of frequent closed sequential patterns. BIDE operates by recursively extending patterns, and while their frequency is above the minimum support, checking closure properties of the extensions.

Consider a frequent pattern $P = \{p_i\}, i = 1, \ldots, n$. There are two ways to extend pattern $P$ *forward* with item $j$:

- Appending the set $p_{n+1} = \{j\}$ to $P$ obtaining $P' = p_1 \ldots p_n p_{n+1}$, called a forward-S(equence)-extension.
- Adding $j$ to the last itemset of $P$: $P' = p_1 \ldots p'_n$, with $p'_n = p_n \cup j$, assuming $j \notin p_n$, called a forward-I(tem)-extension.

Similarly, a pattern can be extended *backward*

- Inserting the set $p_x = \{j\}$ into $P$ anywhere before the last set obtaining $P' = p_1 \ldots p_i p_x p_{i+1} \ldots p_n$, for some $0 \leq i \leq n$, called a backward-S(et)-extension.
- Adding $j$ to any set in $P$ obtaining $P' = p_1 \ldots p'_i \ldots p_n$, with $p'_i = p_i \cup j$, assuming $j \notin p_n$, $1 \leq i \leq n$, called a backward-I(tem)-extension.

According to Theorem 3 of [47], a pattern is closed if there exists no forward-S-extension item, forward-I-extension item, backward-S-extension item, nor backward-I-extension item with the same support.

Furthermore, if there is a backward extension item, then the resulting extension and all of its future extensions are explored in a different branch of the recursion, meaning that it can be pruned from current analysis. These insights are combined in BIDE, leading to a very memory-efficient algorithm, because the patterns found do not need to be kept in memory while the algorithm is running.

**Algorithm 1** BIDE Algorithm

```
Require: Sequential Pattern $P = \{p_i\}$, projected database $D|P$, minimum support $\mu$
1: $F$ - set of frequent closed patterns (global variable)
2: $l = |P|$
3: $Ls = sStepFrequentItems(P, D|P, \mu)$;
4: $Li = iStepFrequentItems(P, D|P, \mu)$;
5: if $!(freqCheck(Ls, P)||freqCheck(Li, P))$ then
6:    if $backscan(P, D, true)$ then
7:       $F = F \cup P$
8: for itemset $p \in Ls$ do
9:    $P' = p_1, .., p_l, p$
10:   if $backscan(P', D|P', false)$ then
11:      $bide(P', D|P', \mu)$;
12: for itemset $p \in Li$ do
13:   $P' = p_1, .., p_{l-1}, p_l \cup p$
14:   if $backscan(P', D|P', false)$ then
15:      $bide(P', D|P', \mu)$;
16: return $F$
```

Specifically, consider pseudo-code for BIDE (Algorithm 1). In Lines 3–4, items that can be used in forward extension of the current pattern are found. If there is no forward extension with the same support (Line 5), the backward closure is checked (Line 6) using function backscan. If the pattern is also backward-closed, it can be added to the set of closed frequent patterns (Line 7).

Then, we check every item in forward S and I extensions (in the two for-loops) to see whether it is explored in a different branch of recursion, again via backscan function (Lines 10, 14). If not, then we project the database on the extension and call BIDE recursively on the extension and the new projected database. We will omit additional details as they are not relevant to our discussion.

## 4.3 BIDE-Discriminative

In order to select only discriminative patterns, we need to keep track of the class label information. We assume that each sequence in dataset $D$ is associated with a class label, and thus we know class distribution in the dataset. When we search for potential S and I extensions, we also keep track of class distribution of sequences where they occur. This gives us all the necessary information for determining discriminative scores and their upper bounds for any new pattern that we discover. Let $d(P, c)$ denotes discriminative score for pattern $P$ for class $c$, and let $d_{UB}(P, c)$ denotes the upper bound on discriminative score for any extension $P'$' of $P$ for class $c$. When discussing two-class problems, $c$ can be omitted from the notation.

The pseudo-code for BIDE-Discriminative is given in Algorithm 2. We have a new user-specified parameter $k$—the number of patterns to extract. We introduce variable $dt$, a threshold on discriminative score, which is initially set to 0.

**Algorithm 2** BIDE-Discriminative Algorithm

```
Require: Sequential Pattern $P = \{p_i\}$, projected database $D|P$, minimum support $\mu$, $k$ - number of patterns
    to be selected
1: $F$ - set of discriminative patterns (global variable)
2: $dt = 0$ - minimal threshold for discriminative score of a pattern (global variable)
3: if $d_{UB}(P) < dt$ then
4:    return
5: $l = |P|$
6: $Ls = sStepFrequentItems(P, D|P, \mu)$;
7: $Li = iStepFrequentItems(P, D|P, \mu)$;
8: if $d(P) \geq dt$ then
9:    if $!(freqCheck(Ls, P)||freqCheck(Li, P))$ then
10:       if $backscan(P, D, true)$ then
11:          $F = F \cup P$
12:          if $|F| > k$ then
13:             $F = F - argmin_{X \in F} d(X)$
14:             $dt = \min_{X \in F} d(X)$
15: for itemset $p \in Ls$ do
16:    $P' = p_1, .., p_l, p$
17:    if $backscan(P', D|P', false)$ then
18:       $BIDEDiscriminative(P', D|P', \mu)$;
19: for itemset $p \in Li$ do
20:    $P' = p_1, .., p_{l-1}, p_l \cup p$
21:    if $backscan(P', D|P', false)$ then
22:       $BIDEDiscriminative(P', D|P', \mu)$;
23: return $F$
```

In Lines 3–4, we check the upper bound of pattern $P$. If the upper bound is below the threshold $dt$, then the pattern and all of its extensions can be pruned—the function returns. Otherwise, algorithm proceeds. In Line 8, the score of the pattern itself is checked against the threshold, to determine if it should be added to the list. If the size of the set $F$ exceeds $k$, the pattern with the lowest score is removed, and threshold $dt$ is updated accordingly (Lines 12–14). Regardless of whether pattern itself is added to the set, its extensions are still examined like in regular BIDE, since the upper bound was above the threshold.

Note that with minor modifications, we can have BIDE-Discriminative output, all patterns with discriminative score above some value. All that is needed is to remove parameter $k$ and make $dt$ a fixed parameter instead.

## 4.4 Handling multi-class problems

As mentioned before, the upper bounds for IG or $\chi^2$ hold for two-class problems only. Multi-class problems can be handled in two ways.

**BIDE-D:** Run BIDE-Discriminative on the whole training data once, but redefine discriminative scores and upper bounds of patterns to be maximums over all binary one-versus-rest problems. Specifically:

$$d'(P) = max_{c \in C} d(P, c) \tag{3}$$

$$d'_{UB}(P) = max_{c \in C} d_{UB}(P, c) \tag{4}$$

**BIDE-DC:** Mine discriminative patterns on the whole training data separately for each of $|C|$ one-versus-rest binary problems. (This straightforward idea has been mentioned in [38] in context of itemset pattern mining, but not implemented). The sets of top-$k$ patterns produced for each class can be merged, resulting in at most $k|C|$ patterns. Note that this could be done in a single run, by maintaining separate sets of patterns and thresholds for each class and expanding a pattern as long as it could be informative for at least one class (**BIDE-DC-1R**). Alternatively, the same set of patterns can be discovered by performing $|C|$ one-versus-rest runs of two-class discriminative BIDE (**BIDE-DC-CR**).

These direct approaches can be contrasted with two indirect approaches:

**BIDE:** Run regular BIDE on all training data and use all closed patterns found. This approach is fully unsupervised.

**BIDE-C:** Run BIDE separately on training data from each class and merge the sets of patterns found. Note that this is the only approach of the four that does mining on subsets of the whole dataset. This approach is indirect but does make some use of class labels.

## 4.5 Computational efficiency

A single run of BIDE-Discriminative has the same complexity as BIDE, since it may still potentially generate all frequent closed sequential patterns, in exactly the same fashion. However, the pruning, depending on the values of $k$ or $dt$, may remove a significant number of patterns and their extensions from consideration. The cost involved in computing the upper bounds for each pattern is independent of the size of the data (at most proportional to number of classes) with appropriate bookkeeping and so does not affect computational complexity.

The reduction in the number of patterns produced can also lead to savings in I/O time and space/memory to store them. Furthermore, since the discriminative scores for the generated patterns are already known, there is no need to separately compute them, as in indirect approaches.

The two variants of BIDE-DC allow us to compare the costs of deeper expansion and of maintaining multiple pattern sets, against the costs of doing $|C|$ runs of BIDE-Discriminative.

Meanwhile, BIDE-C requires $|C|$ runs of BIDE, but each is only on a fraction of the dataset, which is likely to be much faster than a single run on the full dataset. We explore these trade-offs in the experimental section.

## 5 Model-based tree algorithms

A method for direct construction of a tree-based predictive model (model-based tree or MbT) was proposed in [15] and evaluated on itemset and on graph data. We combine BIDE-Discriminative algorithm with the MbT idea for efficiently building a predictive model for sequential data. We will refer to this as **SMBT**. The pseudo-code is given in Algorithm 4. BIDE-Discriminative (Line 4) is used to find the most informative pattern. The database is then split into two sets of sequences: those that match this pattern, and those that don't. The two sets are processed recursively in the same fashion until they become too small or until the class purity of a node exceeds a predefined threshold.

The resulting tree can be used to make prediction on test data in a straightforward fashion. Starting with the root node, we check if a node is a leaf (in which case we return the class label for the node). If the node is not a leaf, we check if the node pattern occurs in the sequence and depending on the result descent to the left or to the right child.

We expect the SMBT to include only a small fraction of the patterns that would have been produced by BIDE-Discriminative alone, significantly easing interpretation of the classifier and of the extracted patterns.

**Algorithm 3** BIDE-DC-1R: Discriminative Algorithm for Multiclass Problems

```
Require: Sequential Pattern $P = \{p_i\}$, projected database $D|P$, minimum support $\mu$, $k$ - number of patterns
    to be selected per class
1: $F_c$ - a set of sets of discriminative patterns for each class (global variable)
2: $dt_c = 0$ - a vector of minimal thresholds for discriminative scores of a pattern for each class (global
    variable)
3: if $\forall c \in C$: $d_{UB}(P, c) < dt_c$ then
4:   return
5: $l = |P|$
6: $Ls = sStepFrequentItems(P, D|P, \mu)$;
7: $Li = iStepFrequentItems(P, D|P, \mu)$;
8: if $!(freqCheck(Ls, P)||freqCheck(Li, P))$ then
9:   if $backscan(P, D, true)$ then
10:      for class $c \in C$ do
11:        if $d(P, c) \geq dt_c$ then
12:          $F_c = F_c \cup P$
13:          if $|F_c| > k$ then
14:            $F_c = F_c - argmin_{X \in F} d(X, c)$
15:            $dt_c = \min_{X \in F_c} d(X, c)$
16: for itemset $p \in Ls$ do
17:   $P' = p_1, .., p_l, p$
18:   if $backscan(P', D|P', false)$ then
19:     $BIDE - DC - 1R(P', D|P', \mu)$;
20: for itemset $p \in Li$ do
21:   $P' = p_1, .., p_{l-1}, p_l \cup p$
22:   if $backscan(P', D|P', false)$ then
23:     $BIDE - DC - 1R(P', D|P', \mu)$;
24: return $\cup F_c$
```

**Algorithm 4** SMBT: MBT Algorithm for sequential data

```
Require: Projected database $D$, minimum support $\mu$, maximum purity $p$, minimum leaf size $z$
1: if $(|D| < z)||(purity(D) > p)$ then
2:     Create and return a leaf node $N$ with label of majority class of $D$
3: Create an empty tree node $N$
4: $P = BIDEDiscriminative(D, \mu, k = 1)$
5: $N.P = P$ - assign pattern $P$ to the node
6: $N.left = MBT(S|P \notin S, \mu)$
7: $N.right = MBT(S|P \in S, \mu)$
8: return $N$
```

While SMBT approach directly constructs a classifier, it can also be viewed as a form of feature selection (**SMBT-FS**): The patterns in the SMBT can be treated as individual features, to be combined using methods such as support vector machines or neural nets.

## 6 Experiments

So far, we have described the following approaches:

– two indirect (i.e., unsupervised) sequential pattern mining approaches (BIDE and BIDE-C)
– two variants of BIDE for directly mining discriminative patterns, specifically BIDE-D and BIDE-DC (with subvariants BIDE-DC-1R and BIDE-DC-CR)
– a sequential model-based tree approach, SMBT, which is a direct mining method and a complete classifier in itself
– SMBT-FS, which uses SMBT purely as a feature Selection/direct mining step and builds a classifier such as SVM with the patterns that SMBT extracts as features

The goals of our experiments are to examine and compare the behavior of the above methods, under different parameter choices, in terms of (i) accuracy, (ii) number of patterns produced, and (iii) time required to mine the patterns. Our expectations are that direct mining methods produce fewer pattern and require less time than indirect and unsupervised methods without sacrificing accuracy. Specifically, because of their respective designs, we expect BIDE-D and BIDE-DC to produce fewer patterns and run faster than BIDE, while SMBT will be faster still with even fewer patterns. The behavior of BIDE-C is more difficult to predict.

We use information gain as the discriminativeness measure with all of these algorithms.

### 6.1 Data

While unlabeled sequential data are relatively common, e.g., web log dataset [3], labeled data needed for our evaluation are much harder to come by. The nine datasets used in our experiments are summarized in Table 2. Two (Unix and WW3D) are simple sequential datasets. The remaining seven, while technically databases of intervals, can be interpreted as sequential databases by treating start and end boundaries of an interval as separate events [48]. Specifically, each symbolic interval, a triple $(t_s, t_e, \sigma)$ with event $\sigma \in \Sigma$ and time stamps $t_s \leq t_e$, is converted into two symbolic time points $(t_s, \sigma^+)$ and $(t_e, \sigma^-)$, and then all time points with the same time stamp are aggregated into itemsets, resulting in a standard event sequence.

**Table 2** Datasets used in experiments

| Data | Intervals | Labels | Sequences | Classes |
|---|---|---|---|---|
| ASL-BU [40] | 18,250 | 154 | 441 | 7 |
| ASL-GT [45] | 89,247 | 47 | 3,493 | 40 |
| Auslan2 [25] | 900 | 12 | 200 | 10 |
| Blocks [16] | 1,207 | 8 | 210 | 8 |
| Context [33] | 12,916 | 54 | 240 | 5 |
| Pioneer [3] | 4,883 | 92 | 160 | 3 |
| Skating [36] | 18,953 | 41 | 530 | 6 (7) |
| Unix [3] | 147,504 | 1,598 | 7,762 | 9 |
| WW3D [26] | 11,597 | 107 | 157 | 6 |

The advantage of this collection is that class labels are available for each sequence. This allows an automated evaluation of patterns using a classifier, while the categorical sequential data available in the UCI Machine Learning Repository [3], such as web log data, are largely unlabeled.

**ASL-BU**[1] The intervals are transcriptions from videos of American Sign Language expressions provided by Boston University [40]. It consists of observation interval sequences with labels such as *head mvmt: nod rapid* or *shoulders forward* that belong to one of 7 classes like *yes–no question* or *rhetorical question*.

**ASL-GT** The intervals are derived from 16-dimensional numerical time series with features derived from videos of American Sign Language expressions [45]. The numerical time series were discretized into 2–4 states, each using Persist [35]. Each sequence represents one of the 40 words such as *brown* or *fish*.

**Auslan2** The intervals were derived from the high-quality Australian Sign Language dataset in the UCI repository [3] donated by Kadous [25]. The $x, y, z$ dimensions were discretized using Persist with 2 bins, and five dimensions representing the fingers were discretized into 2 bins using the median as the divider. Each sequence represents a word such as *girl* or *right*.

**Blocks**[2] The intervals describe visual primitives obtained from videos of a human hand stacking colored blocks provided by [16]. The interval labels describe which blocks touch and the actions of the hand (*contacts blue red*, *attached hand red*). Each sequence represents one of 8 different scenarios from atomic actions (*pick-up*) to complete scenarios (*assemble*).

**Context**[3] The intervals were derived from categoric and numeric data describing the context of a mobile device carried by humans in different situations [33]. Numeric sensors were discretized using 2–3 bins chosen manually based on exploratory data analysis. Each sequence represents one of five scenarios such as *street* or *meeting*.

**Pioneer** The intervals were derived from the Pioneer-1 datasets in the UCI repository [3]. The numerical time series were discretized into 2–4 bins by choosing thresholds manually based on exploratory data analysis. Each sequence describes one of three scenarios: *gripper*, *move*, or *turn*.

**Skating** The intervals were derived from 14-dimensional numerical time series describing muscle activity and leg position of six professional in-line speed skaters during controlled tests at seven different speeds on a treadmill [36]. The time series were discretized into 2–3

[1] http://www.bu.edu/asllrp/.

[2] http://ftp.ecn.purdue.edu/qobi/ama.tar.Z.

[3] http://www.cis.hut.fi/jhimberg/contextdata/index.shtml.

bins using Persist and manually chosen thresholds. Each sequence represents a complete movement cycle and is labeled by skater or speed.

**Unix** The dataset consists of sanitized command histories of 9 users [3].

**WW3D** This dataset was collected from Wubble World 3D (ww3d), a virtual environment with simulated physics in which softbots, called wubbles, interact with objects [26].

## 6.2 Experiment setup

For each method and parameter setting, we repeated fivefold cross-validation 3 times, each time with a different random split. Thus, all the measures (accuracy, number of patterns, run times, etc) are averages taken over $3 * 5 = 15$ test sets. Each experimental run consisted of two parts: pattern mining and classification. Pattern mining is done on the training folds. Then, sequences in both training and test folds are converted to binary vectors based on the presence of the patterns. These vectors are written to files. (This part is programmed in Java). The vectors from training folds are then used to build classifiers, which are then applied to the test folds.

For direct BIDE approaches (BIDE-D and both variants of BIDE-DC), we considered $k = 10, 20, 30, 40, 50, 70, 90$, where $k$ is the number of patterns per class.

In order to compare all the approaches, we also had to find a way of setting minimum support similarly for all of them. For most datasets, we used the following strategy: parameter $\nu = 0.2, 0.3, 0.4, 0.5, 0.6$ specified the fraction of the size of the smallest class in the training data that would be used as minimum support. For example, if the training set has 100 instances of 5 classes, but the smallest class has 10 instances, then with $\nu = 0.5$, the minimum support would be set to 5 for all methods.

It turned out that for some datasets (ASL-GT, Context, Skating, and Unix), this approach did not work—pattern mining was taking too long for all the approaches. For these datasets, we changed the way minimum support was computed: fraction $\nu$ of the whole training set size was used. BIDE-C cannot be used in such situations, but the other three methods ran successfully. In these cases, $\nu$ is equivalent to $\mu$, minimum support. For Context and Skating, $\nu = 0.7, 0.8, 0.9$. For ASL-GT dataset, $\nu = 0.2, 0.3, 0.4, 0.5, 0.6$; while for Unix dataset, $\nu = 0.1, 0.15, 0.2$.

The way we set minimum support ensures that all four methods are exploring the same pattern space and could conceivably produce exactly the same set of patterns, i.e., a set of closed frequent patterns for minimum support derived using $\nu$. It follows that the set of patterns found by BIDE is going to be a superset of sets of patterns found by the other approaches for the same value of $\nu$.

For SMBT experiments, we kept the same values of $\nu$. However, we had to additionally specify values for minimum leaf size $z$ and leaf purity $p$. We experimented with $p = 0.8, 0.9$ and $z = 1, 3, 5$.

Classification was performed in MATLAB, using LIBLINEAR [14] (a fast implementation of linear SVM) with options "-B 1 -S 5", i.e., L1-regularized L2-loss support vector classification. The value for parameter C was selected from the set $10^{-3,\ldots,3}$ using threefold cross-validation on the training set.

## 6.3 Experimental results

Here, we describe the highlights of our evaluation. More details are presented in the "Appendix."[4]

[4] https://sites.google.com/site/dfradkin/kais2014-separateAppendix.pdf.

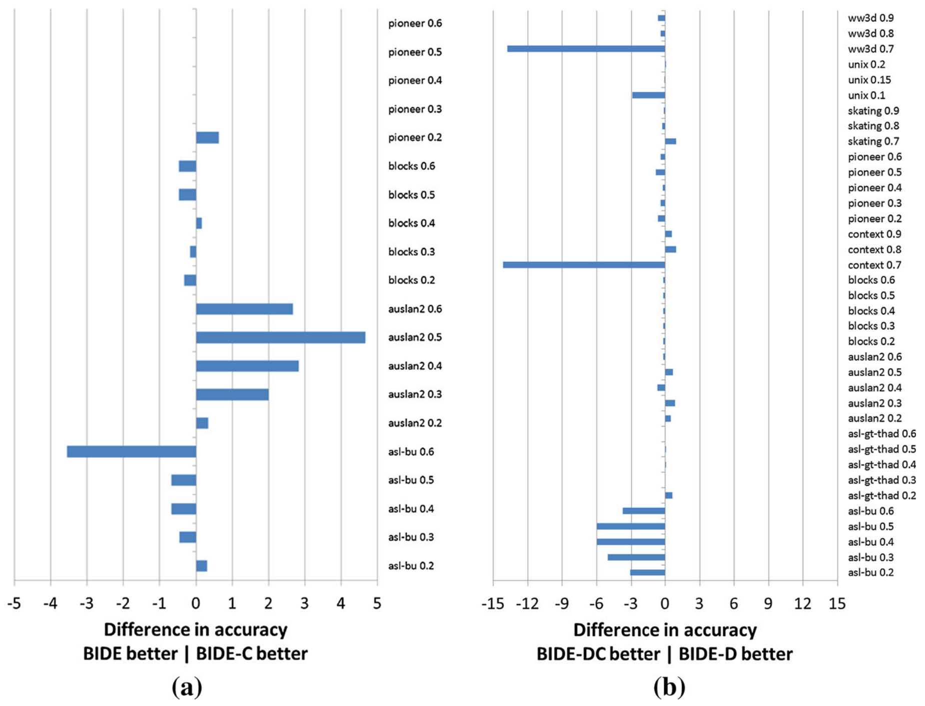


**Fig. 1** Comparison of accuracy of BIDE approaches

**Comparison of two BIDE-DC variants:** The two variants, BIDE-DC-1R and BIDE-DC-CR, will produce identical sets of patterns, so the only meaningful comparison between them is in speed. Not surprisingly, using a single run (BIDE-DC-1R) is always significantly faster than doing separate runs (BIDE-DC-CR), despite some additional overhead and deeper search required for the former. (These results are shown in "Appendix"). Thus, in the rest of the paper, we will use only BIDE-DC-1R and, for simplicity, will refer to it as BIDE-DC.

**Effect of** $k$: We experimented with $k = 10, 20, 30, 40, 50, 70, 90$, as mentioned in Sect. 6.2. Performance of BIDE-DC seems to be less sensitive to value of $k$ and of $\nu$ than that of BIDE-D, likely because it distributes patterns evenly across classes, while BIDE-D may suffer from redundant patterns that are all predictive for the same class. Higher values of $k$ lead to better results where the differences are observed. Thus in the rest of the experiments, we will keep $k$ fixed at 90. (The plots showing accuracy versus $\nu$ for different values of $k$ are shown in "Appendix").

**Effects of purity and leaf size:** SMBT and SMBT-FS are not particularly sensitive to choices of minimum leaf size and purity (see "Appendix"). Thus we focus on results with leaf size 5 and purity of 0.9.

**Accuracy comparisons:** We start by comparing the accuracies of the most similar approaches: BIDE versus BIDE-C, BIDE-D versus BIDE-DC, and SMBT versus SMBT-FS. We then compare performance of 'winners' to each other. The "Appendix" shows complete results and discusses them in more detail.

BIDE leads to better results than BIDE-C on ASL-BU and Blocks, and worse ones on Auslan2 dataset. On the other datasets, however, BIDE-C is difficult to apply due to the need to set minimum support $\mu$ separately for each class. Thus, BIDE is to be generally preferred between these two (Fig. 1a).

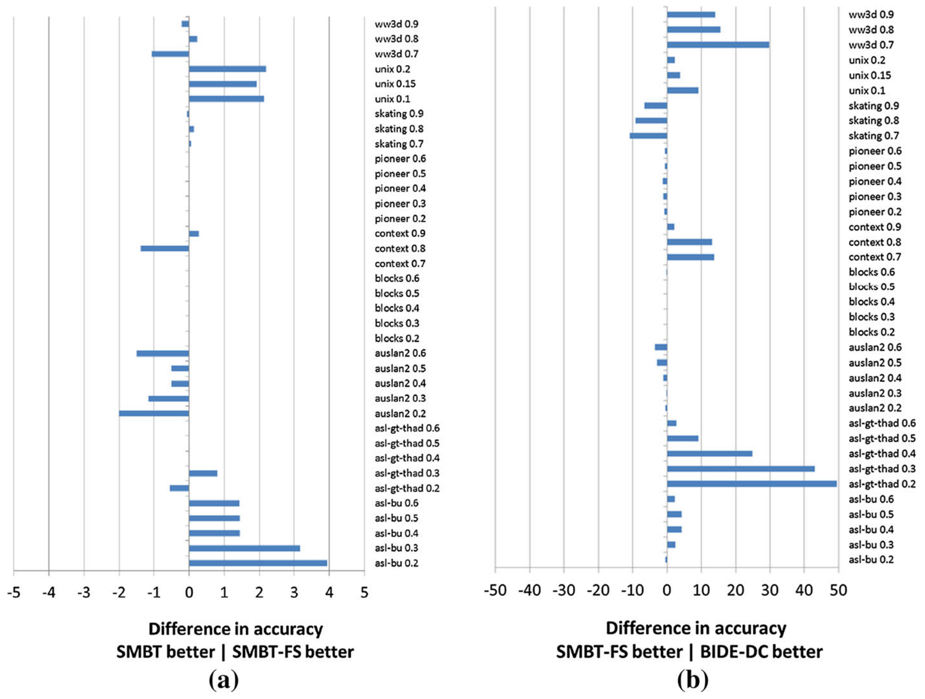


**Fig. 2** Comparison of accuracy of SMBT-FS with SMBT and BIDE-DC

BIDE-DC outperforms BIDE-D on ASL-BU, Context, Pioneer, and WW3D while having comparable accuracy on the other datasets. (Fig. 1b).

SMBT-FS noticeably outperforms SMBT on ASL-BU and Unix, is slightly worse on Auslan2, Context, and WW3D, and is comparable on the other datasets. As can be seen in "Appendix," SMBT-FS also performs better against other methods (i.e., SMBT does better on datasets where both methods perform poorly). Thus, SMBT-FS is a better method.

We now compare the 'winners' from each pair. Comparison between BIDE-DC and SMBT-FS favors the former, which performs much better on most datasets, only slightly worse on Skating and Auslan2 (Fig. 2b). SMBT-FS also is clearly worse than BIDE, again excepting Skating and Auslan2 (Fig. 3a). BIDE-DC and BIDE are closely matched (Fig. 3b), with the differences in accuracy less than 2 % in either direction in almost all the cases.

We can thus conclude that BIDE-DC is the best and the most stable of the direct methods in terms of accuracy and gives performance comparable to using all the patterns mined in an unsupervised fashion, i.e., BIDE. SMBT-FS can occasionally perform somewhat better than BIDE-DC, but also frequently produces much worse results.

**Number of patterns and mining speed:**
Having compared the accuracies of different approaches, we turn our attention to the number of patterns mined and computational efficiency. Figure 4a shows, on log scale, the ratio of patterns produced by BIDE to that produced by BIDE-DC. It is obvious that on most datasets, BIDE-DC uses just a small fraction of frequent closed patterns that BIDE generates. Figure 4b shows reduction in pattern mining time from using BIDE-DC, compared to using BIDE. BIDE-DC is faster, sometimes by a lot, except on ASL-GT and Auslan2 datasets.

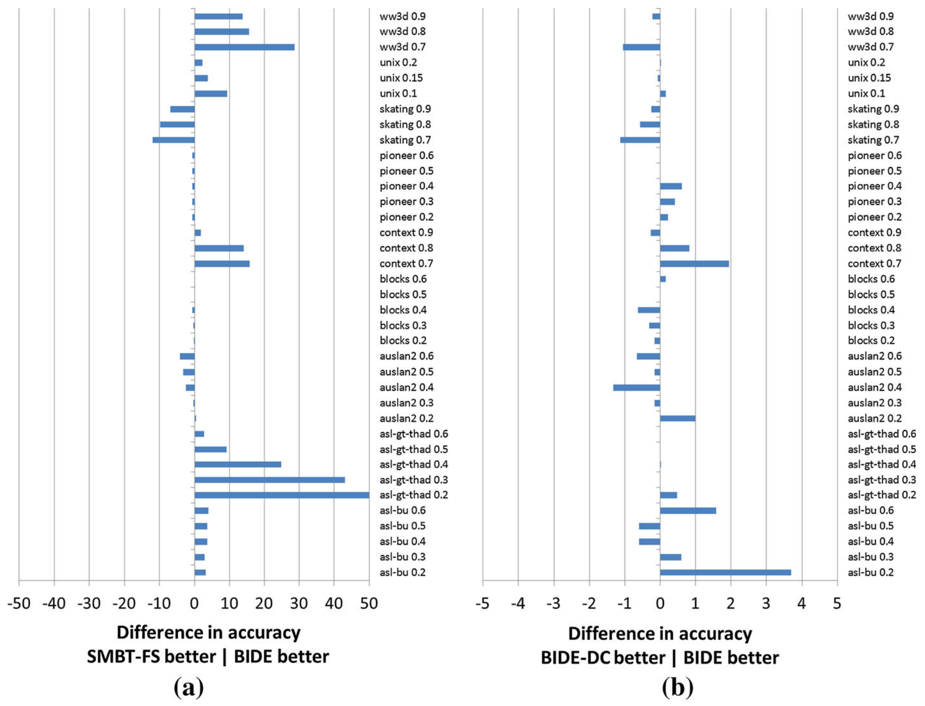


**Fig. 3** Comparison of accuracy of SMBT-FS, BIDE-DC, and BIDE

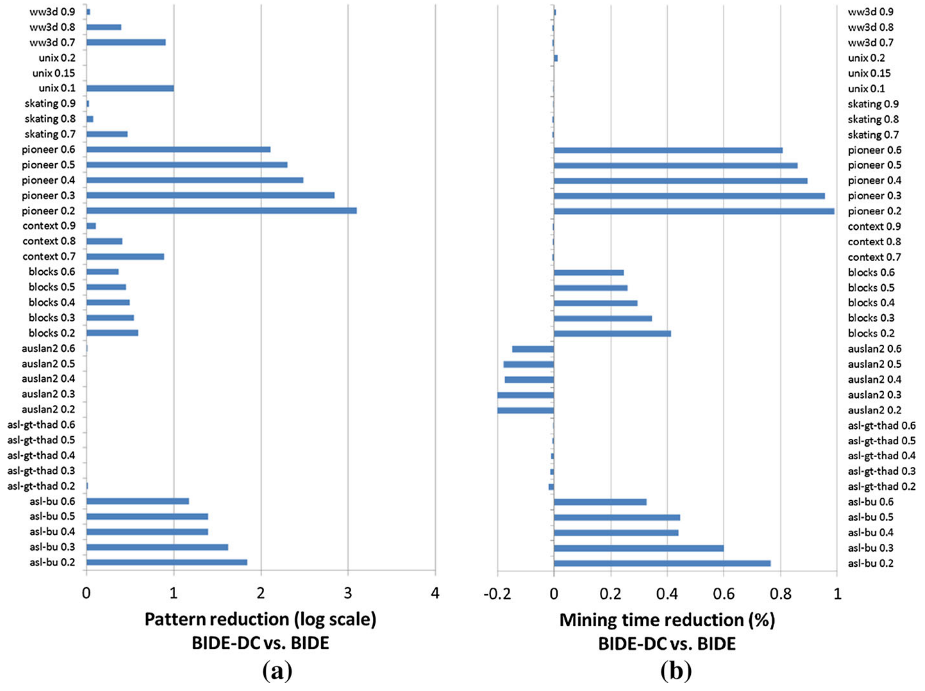


**Fig. 4** Comparison of number of patterns and mining times of BIDE-DC and BIDE

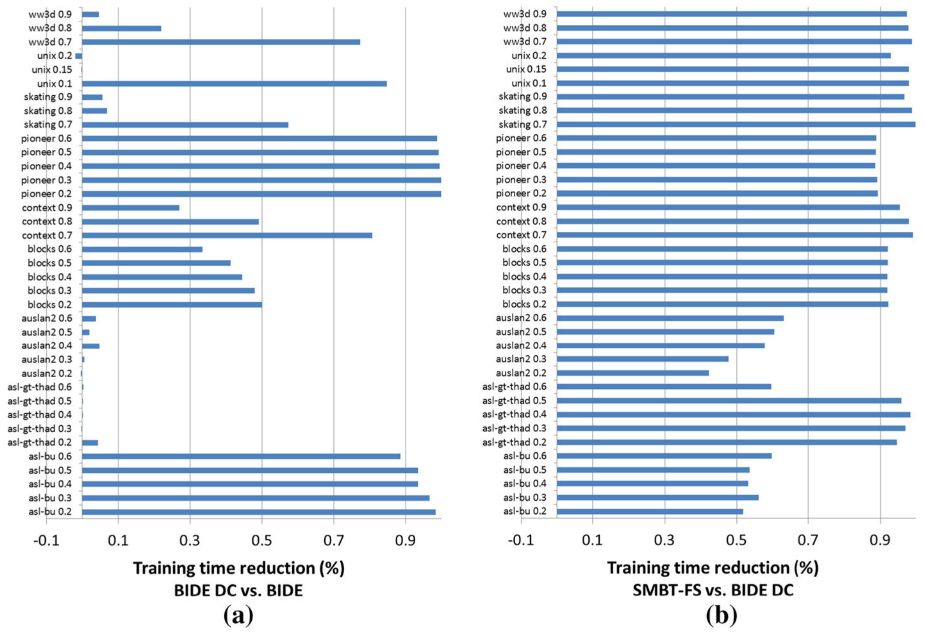


**Fig. 5** Comparison of training times of SMBT-FS, BIDE-DC, and BIDE

Note that these are the same datasets where the number of patterns produced by BIDE-DC is comparable or same as that produced by BIDE. The explanation for these results is that on some datasets, for certain values of $\nu$ few patterns exist, and BIDE finds them all faster than BIDE-DC due to smaller overhead. However, when the number of potential patterns is high, BIDE-DC is faster.

Finally, we demonstrate that using only the top discriminative features leads to improved training time for a classifier, such as SVM. Figure 5a compares training times with patterns produced by BIDE and BIDE-DC, with latter consistently faster except on ASL-GT and Auslan, as discussed.

We also note that SMBT-FS can sometime produce a lot fewer patterns and be several times faster in mining time and training time (Fig. 5b) than BIDE-DC, but again, the cost in terms of accuracy can be high. Detailed results are presented in the “Appendix.”

## 7 Conclusions

We have described a direct sequential pattern mining approach, BIDE-Discriminative, and how it can be utilized for discriminative sequential pattern mining in multi-class problems. We have evaluated approaches for discriminative sequential pattern mining in multi-class problems (BIDE-D, BIDE-DC, SMBT, and SMBT-FS) against unsupervised approaches (BIDE and BIDE-C). Our experiments suggest that BIDE-DC is usually the best option, as it efficiently generates a small number of predictive patterns leading to comparable classification performance while potentially saving the user order of magnitude in memory and noticeable amount of time both during the pattern mining stage and when training a classifier. The slight advantage in accuracy obtained by BIDE-DC over BIDE-D is likely due to former extracting fewer redundant patterns.

SMBT and SMBT-FS can sometimes match accuracy of BIDE-DC with a lot fewer patterns, but can also result in significantly worse performance. Also, while in some cases they were faster than the other methods, in others the reverse was true. This unpredictability makes caution against these approaches, though in some application domains where very small models are required, the benefits may outweigh the risks.

Our evaluation was performed on a collection of nine real-world datasets and is the first evaluation of sequential discriminative pattern mining methods.

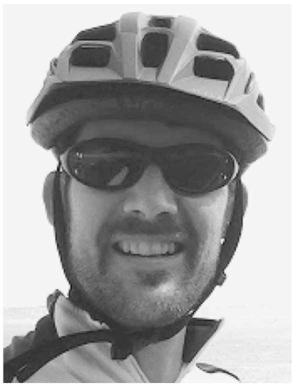
## References

1. Agrawal R, Imielinski T, Swami AN (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD international ACM Press, conference on management of data, pp 207–216
2. Agrawal R, Srikant R (1995) Mining sequential patterns. In: ICDE. IEEE Press, pp 3–14
3. Asuncion A, Newman D (n.d.) UCI Machine Learning Repository
4. Batal I, Fradkin D, Harrison J, Moerchen F, Hauskrecht M (2012) Mining recent temporal patterns for event detection in multivariate time series data. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining, pp 280–288. doi:10.1145/2339530.2339578
5. Batal I, Valizadegan H, Cooper GF, Hauskrecht M (2011) A pattern mining approach for classifying multivariate temporal data. In: Proceedings of the 2011 IEEE international conference on bioinformatics and biomedicine, pp 358–365. doi:10.1109/BIBM.2011.39
6. Bringmann B, Zimmermann A (2008) One in a million: picking the right patterns. Knowl Inf Syst 18(1):61–81
7. Bringmann B, Zimmermann A, Raedt L, Nijssen S (2006) Dont be afraid of simpler patterns. In: Frnkranz J, Scheffer T, Spiliopoulou M (eds) Knowledge discovery in databases: PKDD 2006, vol 4213 of LNCS. Springer, Berlin, pp 55–66. doi:10.1007/11871637_10
8. Buza K, Schmidt-Thieme L (2010) Motif-based classification of time series with bayesian networks and svms. In: Fink A, Lausen B, Seidel W, Ultsch A (eds) Advances in data analysis, data handling and business intelligence. Studies in classification, data analysis, and knowledge organization. Springer, Berlin, pp 105–114. doi:10.1007/978-3-642-01044-6_9
9. Carbonell J, Coldstein J (1998) The use of MMR, diversity-based reranking for reordering documents and producing summaries. In: Proceedings of SIGIR, p 335336
10. Cheng H, Yan X, Han J, Hsu C-W (2007) Discriminative frequent pattern analysis for effective classification. In: Proceedings of the IEEE ICDE
11. Cheng H, Yan X, Han J, Yu PS (2008) Direct discriminative pattern mining for effective classification. In: ICDE, pp 169–178
12. Cover TM, Thomas JA (2006) Elements of information theory, 2nd edn. Wiley, New York
13. Dong G, Pei J (2007) Sequence data mining. Morgan Kaufmann, Burlington
14. Fan R-E, Chang K-W, Hsieh C-J, Wang X-R, Lin C-J (2008) Liblinear: a library for large linear classification. J Mach Learn Res 9:1871–1874
15. Fan W, Zhang K, Cheng H, Gao J, Yan X, Han J, Yu PS, Verscheure O (2008) Direct mining of discriminative and essential frequent patterns via model-based search tree. In: KDD, pp 230–238
16. Fern A (2004) Learning models and formulas of a temporal event logic. PhD thesis, Purdue University, West Lafayette, IN, USA
17. Fradkin D, Moerchen F (2010) Margin-closed frequent sequential pattern mining. KDD workshop on useful patterns. ACM, New York, NY, USA, pp 45–54
18. Grahne G, Zhu J (2003) Efficiently using prefix-trees in mining frequent itemsets. In: ICDM workshop on frequent itemset mining implementations
19. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182
20. Han J, Kamber M (2006) Data mining: concepts and techniques, 2nd edn. Morgan Kaufmann, Burlington
21. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proceedings of the ACM SIGMOD international conference on management of data. ACM Press, pp 1–12
22. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: SIGMOD, pp 1–12
23. Ifrim G, Bakir GH, Weikum G (2008) Fast logistic regression for text categorization with variable-length n-grams. In: KDD, pp 354–362

24. Ifrim G, Wiuf C (2011) Bounded coordinate-descent for biological sequence classification in high dimensional predictor space. In: KDD
25. Kadous MW (2002) Temporal classification: extending the classification paradigm to multivariate time series. PhD thesis, University of New South Wales
26. Kerr W, Cohen P, Chang Y-H (2008) Learning and playing in wubble world. In: Proceedings of the fourth artificial intelligence and interactive digital entertainment conference, pp 66–71
27. Knobbe AJ, Ho EKY (2006) Pattern teams. In: PKDD, pp 577–584
28. Lee J-G, Han J, Li X, Cheng H (2011) Mining discriminative patterns for classifying trajectories on road networks. IEEE Trans Knowl Data Eng 23(5):713–726
29. Lin J, Keogh E, Lonardi S, Chiu B (2003) A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 2003 ACM SIGMOD workshop on research issues in data mining and knowledge discovery. ACM Press, pp 2–11. URL:http://citeseer.ist.psu.edu/583097.html
30. Lo D, Cheng H, Cia L (2011) Mining closed discriminative dyadic sequential patterns. In: EDBT
31. Lo D, Han J, Cheng H, Khoo S-C, Sun C (2009) Classification of software behaviros for failure detection: a discriminative pattern mining approach. In: Proceedings of KDD
32. Lucchese C, Orlando S, Perego R (2006) Fast and memory efficient mining of frequent closed itemsets. IEEE Trans Knowl Data Eng 18(1):21–36
33. Mäntyjärvi J, Himberg J, Kangas P, Tuomela U, Huuskonen P (2004) Sensor signal data set for exploring context recognition of mobile devices. In: Proceedings of PERVASIVE. Springer, pp 18–23
34. Moerchen F, Thies M, Ultsch A (2011) Efficient mining of all margin-closed itemsets with applications in temporal knowledge discovery and classification by compression. Knowl Inf Syst 29:55–80. doi:10.1007/s10115-010-0329-5
35. Mörchen F, Ultsch A (2005) Optimizing time series discretization for knowledge discovery. In: Proceedings of the ACM SIGKDD. ACM Press, pp 660–665
36. Mörchen F, Ultsch A (2007) Efficient mining of understandable patterns from multivariate interval time series. Data Min Knowl Discov 15(2):181–215. doi:10.1007/s10618-007-0070-1
37. Morishita S, Sese J (2000) Traversing itemset lattice with statistical metric pruning. In: PODS, pp 226–236
38. Nijssen S, Kok J (2006) Multi-class correlated pattern mining. In: Bonchi F, Boulicaut J-F (eds) Knowledge discovery in inductive databases, vol 3933 of LNCS. Springer, Berlin, pp 165–187. doi:10.1007/11733492_10
39. Ohara K, Hara M, Takabayashi K, Motoda H, Washio T (2008) Pruning strategies based on the upper bound of information gain for discriminative subgraph mining. In: PKAW'08, pp 50–60
40. Papaterou P, Kollios G, Sclaroff S, Gunopoulos D (2005) Discovering frequent arrangements of temporal intervals. In: ICDM, pp 354–361
41. Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu M-C (2001) PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the IEEE ICDE. IEEE Press, pp 215–224
42. Sese J, Morishita S (2004) Itemset classified clustering, In: Boulicaut J-F, Esposito F, Giannotti F, Pedreschi D (eds) Knowledge discovery in databases: PKDD 2004, vol 3202 of LNCS. Springer, Berlin, pp 398–409. doi:10.1007/978-3-540-30116-5_37
43. Sipos R, Fradkin D, Moerchen F, Wang Z (2014) Log-based predictive maintenance, In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1867–1876. doi:10.1145/2623330.2623340
44. Srikant R, Agrawal R (1996) Mining sequential patterns: generalizations and performance improvements. In: Proceedings of the 5th international conference on extending database technology (EDBT). Springer, pp 3–17. URL:http://citeseer.ist.psu.edu/article/srikant96mining.html
45. Starner T, Weaver J, Pentland A (1998) Real-time American sign language recognition using desk and wearable computer-based video. IEEE Trans Pattern Anal Mach Intell 20(12):1371–1375. doi:10.1109/34.735811
46. Wang J, Han J (2004) BIDE: Efficient mining of frequent closed sequences. In: ICDE. IEEE Press, pp 79–90
47. Wang J, Han J, Li C (2007) Frequent closed sequence mining without candidate maintenance. IEEE Trans Knowl Data Eng 19(8):1042–1056
48. Wu S-Y, Chen Y-L (2007) Mining nonambiguous temporal patterns for interval-based events. IEEE Trans Knowl Data Eng 19(6):742–758
49. Xu W, Huang L, Fox A, Patterson D, Jordan M (2008) Mining console logs for large-scale system problem detection. In: Proceedings of the 3rd workshop on tackling computer systems problems with machine learning techniques
50. Yan X, Han J (2002) gspan: Graph-based substructure pattern mining. In: ICDM

51. Yang Y, Pedersen J (1997) A comparative study on feature selection in text categorization. In: ICML, pp 412–420
52. Zaki M (2001) Spade: an efficient algorithm for mining frequent sequences. Mach Learn 42:31–60
53. Zaki MJ, Hsiao C-J (2002) CHARM: an efficient algorithm for closed itemset mining. In: Proceedings of the 2nd SIAM international conference on data mining (SDM), SIAM, pp 457–473

**Dmitriy Fradkin** is a Senior Scientist at Siemens Corporate Technology, Princeton, NJ. He received B.A. in Mathematics and Computer Science from Brandeis University, Waltham, MA in 1999 and PhD from Rutgers, The State University of New Jersey in 2006. Before joining Siemens, in 2007 he has worked at Ask.com. His research is in applying data mining and machine learning techniques to solve real-world problems in areas of predictive maintenance, health care, and text analytics. Dr. Fradkin is a member of the ACM SIGKDD and a reviewer for several data mining journals.



**Fabian Mörchen** graduated with a PhD in 2006 from the Philipps University of Marburg, Germany, with summa cum laude. His thesis contributed novel methods in mining temporal pattern from interval data. From 2006 to 2012, he was worked at Siemens Corporate Research leading data mining projects with applications in predictive maintenance, text mining, health care, and sustainable energy. He continued his research in temporal data mining in the context of industrial and scientific problems and served the community as a reviewer, organizer of workshops, and presenter of tutorials. Since 2012, he is leading a data science team at Amazon to improve customer experience using machine learning and big data analytics.