Mining Sequence Classifiers for Early Prediction *

Zhengzheng Xing
School of Computing Science
Simon Fraser University
zxing@cs.sfu.ca

Jian Pei
School of Computing Science
Simon Fraser University
jpei@cs.sfu.ca

Guozhu Dong
Department of Computer Science and Engineering
Wright State University
gdong@cs.wright.edu

Philip S. Yu
Department of Computer Science
University of Illinois at Chicago
psyu@cs.uic.edu

Abstract

Supervised learning on sequence data, also known as sequence classification, has been well recognized as an important data mining task with many significant applications. Since temporal order is important in sequence data, in many critical applications of sequence classification such as medical diagnosis and disaster prediction, *early prediction* is a highly desirable feature of sequence classifiers. In early prediction, a sequence classifier should use a prefix of a sequence as short as possible to make a reasonably accurate prediction. To the best of our knowledge, early prediction on sequence data has not been studied systematically.

In this paper, we identify the novel problem of mining sequence classifiers for early prediction. We analyze the problem and the challenges. As the first attempt to tackle the problem, we propose two interesting methods. The sequential classification rule (SCR) method mines a set of sequential classification rules as a classifier. A so-called early-prediction utility is defined and used to select features and rules. The generalized sequential decision tree (GSDT) method adopts a divide-and-conquer strategy to generate a classification model. We conduct an extensive empirical evaluation on several real data sets. Interestingly, our two methods achieve accuracy comparable to that of the state-of-the-art methods, but typically need to use only very short prefixes of the sequences. The results clearly indicate that early prediction is highly feasible and effective.

1 Introduction

Supervised learning on sequence data, also known as sequence classification, has been well recognized as an important data mining task with many applications [4]. Sequence classification can be tackled by the general classification strategies using feature extraction [13]. That is, (short) subsequences can be extracted as features. A sequence is transformed into a set of features. Then, a general classification method such as support vector machines (SVM) and artificial neural networks can be applied on the transformed data set.

A unique characteristic of sequence data is that the order is essential. In many applications of sequence classification, the temporal order plays a critical role. As an example, consider the application of disease diagnosis using medical record sequences. For a patient, the symptoms and the medical test results are recorded as a sequence. Diagnosis can be modeled as a problem of classification of the medical record sequences.

Obviously, the longer the sequence for a patient, the more information is available about the patient and the more accurate a classification decision can be made. However, to make the diagnosis useful, an early prediction is always desirable. A reasonably accurate prediction using an as short as possible prefix of a patient's medical record sequence is highly valuable. Such an early prediction can lead to early treatment of diseases. For example, the survival rate of many types of cancer is high if the tumors can be detected in an early stage.

Early prediction is also strongly needed in disaster prediction. For example, early prediction of earthquakes and tsunamis is extremely important since a few minutes ahead may save thousands of lives. In applications of system management including intrusion detection, an event sequence generated from a complex system is used to determine the underlying problem. Early prediction will save time and provide opportunity to take early action and prevent catastrophic failures.

*We thank the anonymous reviewers for their insightful comments which help to improve the quality of this paper. Z. Xing's and J. Pei's research is supported in part by an NSERC Discovery Grant. Z. Xing's research is also supported in part by an SFU CTEF graduate fellowship. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

Surprisingly, early prediction has not been studied systematically. The existing work on sequence classification only focuses on improving the accuracy of classification. The existing methods extract features from the whole sequences and use those features to construct classification models. None of them explore the utility of features in early prediction.

In this paper, we study the important problem of sequence classification towards early prediction. We make the following contributions. First, we identify the problem of sequence classification towards early prediction. We analyze the problem and the challenges. Second, as the first attempt to tackle the problem, we propose two interesting methods. The sequential classification rule (SCR) method mines a set of sequential classification rules as a classifier. A so-called early-prediction utility is defined and used to select features and rules. The generalized sequential decision tree (GSDT) method adopts a divide-and-conquer strategy to generate a classification model. Last, we conduct an extensive empirical evaluation on several real data sets. Interestingly, our two methods achieve accuracy comparable to that of the state-of-the-art methods, but typically need to use only very short prefixes of the sequences. The results clearly indicate that early prediction is feasible and effective.

The rest of the paper is organized as follows. We describe the problem of sequence classification for early prediction and review the related work in Section 2. The sequential classification rule (SCR) method is developed in Section 3. Section 4 presents the generalized sequential decision tree (GSDT) method. We report a systematic empirical evaluation in Section 5. The paper is concluded in Section 6.

2 Problem Description and Related Work

2.1 Sequence Classification and Early Prediction Let Ω be a set of symbols which is the alphabet of the sequence database in question. $s = a_1 \cdots a_l$ is a *sequence* if $a_i \in \Omega$ $(1 \leq i \leq l)$. $len(s) = l$ is the *length* of the sequence.

Let $\mathcal{L}$ be a finite set of *class labels*. A *sequence database* SDB is a set of tuples (s, c) such that $s \in \Omega^*$ is a sequence and $c \in \mathcal{L}$ is a class label.

A *sequence classifier* is a function $C : \Omega^* \rightarrow \mathcal{L}$. That is, for each sequence $s \in \Omega^*$, C predicts the class label of s as $C(s)$.

There are many possible ways to construct a sequence classifier. We review some existing methods in Section 2.2. In order to make early prediction, in this paper, we are particularly interested in serial sequence classifiers, a special type of sequence classifier. Roughly speaking, a serial classifier reads a sequence from left to right, and makes prediction once it is confident about the class label of the input sequence based on the prefix read so far.

Formally, for sequence $s = a_1 \cdots a_l$, sequence $s' = a_1 \cdots a_{l'}$ $(1 \leq l' \leq l)$ is a *prefix* of s. We write $s' = s[1, l']$. A sequence classifier C is *serial* if for any (potentially infinite) sequence $s = a_1 a_2 \cdots$, there exists a positive integer l_0 such that $C(s[1, l_0]) = C(s[1, l_0 + 1]) = C(s[1, l_0 + k]) = C(s)$ $(k \geq 0)$. In other words, C predicts based on only the prefix $s[1, l_0]$. One may generalize the definition to require C to give consistent predictions most of the time. The length l_0 of the prefix that C checks in order to make the prediction is called the *cost* of the prediction, denoted by $Cost(C, s) = l_0$. Clearly, $Cost(C, s[1, l_0]) = Cost(C, s[1, l_0 + 1]) = \cdots = Cost(C, s)$. Trivially, $Cost(C, s) \leq \|s\|$ for any finite sequence s.

Given a sequence database SDB, the *accuracy* of a serial classifier C is $Accuracy(C, SDB) = \frac{\|\{C(s)=c \mid (s,c) \in SDB\}\|}{\|SDB\|}$.

Moreover, the *cost of the prediction* is $Cost(C, SDB) = \frac{\sum_{(s,c) \in SDB} Cost(C,s)}{\|SDB\|}$.

Generally, for early prediction, we want to reduce the prediction cost and retain the prediction accuracy at a satisfactory level. Theoretically, we define the following optimization problem.

Problem Definition The problem of *sequence classification for early prediction* is to construct a serial classifier C such that C has an expected accuracy p_0 and minimizes the expected prediction cost, where p_0 is a user specified parameter. ∎

2.2 Related Work To the best of our knowledge, [1] is the only existing study mentioning early prediction. In [1], early classification is to classify a case before all features required by the classifier are present. It uses linear combinations of features in classification, allowing one to make a prediction when some features are missing, although the accuracy may deteriorate. The problem addressed by [1] is very different from the problem discussed in this paper. The method in [1] is not entirely dedicated to early prediction. In [33], progressive confidence rules are proposed. Our work is different from [33] because we focus on using prefix to do early prediction but [33] considers progressive confidence rules as patterns to increase the accuracy of prediction but does not deal with early classification.

Recently, [18] studies the problem of phrase prediction as suggesting the highly possible phrases given a prefix. The phrase prediction problem is different from the early prediction problem in this paper. Phrase prediction can suggest multiple phrases for a prefix, and can revise the prediction as the prefix grows.

Generally, classification on sequence data has been an extensively studied problem. Most previous studies tackle the sequence classification problem by combining some general classification methods and some sequence feature selection methods. In [13], criteria for selecting features for sequence classification is proposed. [20, 9, 25, 27] study the prediction of outer membrane proteins from protein sequences by combining the support vector machines (SVM) [29] and several feature selection methods. Particularly, [20] uses five different types of features, namely amino acids, amino acid pairs, one-gapped amino acid pairs (allowing exactly one gap between two amino acids), two-gapped amino acid pairs, and three-gapped amino acid pairs. [9] uses gapped amino acid pairs as the features. [25] used frequent subsequences as the features. Several kernel functions (e.g., the simple linear kernel, the polynomial kernel, and the RBF kernel) are used in those studies.

Some other studies use artificial neural networks (ANN) for sequence classification. For example, [31] uses ANN for protein sequence family classification. Sequences are mapped to vectors of k-gram frequencies. The vectors are used as input to the ANN. It reduces the dimensionality of the vectors by mapping amino acids into amino-acid equivalence groups, together with the SVD method. Moreover, [17] uses neural networks and expectation maximization to classify *E. Coli* promoters.

The methods using SVM and ANN are often whole-sequence based, where all features from all parts of the sequences can be used.

Early prediction is different from general whole-sequence based prediction, since it desires to use features as near the beginning of the sequences as possible.

Hidden Markov models (HMM) [24, 5] and variants are popular classification methods for the so-called "site-based" classification problem, where one is interested in whether a sequence contains a site of interest and the actual position of the site in the sequence if it exists. This problem is closely related to the motif finding problems. Here, one usually uses features before/around the site. Examples of site-based classification include the gene transcription start site problem [11, 16], the splicing site problem [3, 26], the transcription factor binding sites area [21], etc. Examples of using HMM and variants include [12, 7]. Early prediction is different from site-based prediction since there might be no obvious sites in the sequences under consideration.

3 The Sequential Classification Rule Method

In this section, we develop a sequential classification rule (SCR) method. The major idea is that we learn a set of classification rules from a training data set. Each rule hypothetically represents a set of sequences of the same class and sharing the same set of features.

3.1 Sequential Classification Rules A *feature* is a short sequence $f \in \Omega^*$. A feature $f = b_1 \cdots b_m$ *appears* in a sequence $s = a_1 \cdots a_l$, denoted by $f \sqsubseteq s$, if there exist $1 \leq i_0 \leq l - m + 1$ such that $a_{i_0} = b_1$, $a_{i_0+1} = b_2$, $\ldots$, $a_{i_0+m-1} = b_m$. For example, feature $f = bbd$ appears in sequence $s = acbbdadbbdca$. When a feature f appears in a sequence s, we can write $s = s'fs''$ such that $s' = a_1 \cdots a_{i_0-1}$ and $s'' = a_{i_0+m} \cdots a_l$. Generally, a feature may appear multiple times in a sequence. The *minimum prefix* of s where feature f appears is denoted by $minprefix(s, f)$. When $f \not\sqsubseteq s$, $minprefix(s, f) = s$, which means f does not appear in any prefix of s.

A *sequential classification rule* (or simply a *rule*) is in the form of $R : f_1 \rightarrow \cdots \rightarrow f_n \Rightarrow c$ where $f_1, \ldots, f_n$ are features and $c \in \mathcal{L}$ is a class label. For the sake of simplicity, we also write a rule as $R : F \Rightarrow c$ where F is the shorthand of a series of features $f_1 \rightarrow \cdots \rightarrow f_n$. The class label in a rule R is denoted by $\mathcal{L}(R) = c$.

A sequence s is said to *match* a sequential classification rule $R : f_1 \rightarrow \cdots \rightarrow f_n \Rightarrow c$, denoted by $R \sqsubseteq s$, if $s = s'f_1s_1f_2 \cdots s_{n-1}f_ns''$. That is, the features in R appear in s in the same order as in R. The *minimum prefix* of s matching rule R is denoted by $minprefix(s, R)$. Particularly, when $R \not\sqsubseteq s$, $minprefix(s, R) = s$, which means any prefix of s does not match R.

Given a sequence database SDB and a sequential classification rule R, the *support* of R in SDB is defined as $sup_{SDB}(R) = \frac{\|\{s|s\in SDB, R\sqsubseteq s\}\|}{\|SDB\|}$.

Moreover, the *confidence* of rule R on SDB is given by $conf_{SDB}(R) = \frac{\|\{(s,c)|(s,c)\in SDB, R\sqsubseteq s, c=\mathcal{L}(R)\}\|}{\|\{(s,c)|(s,c)\in SDB, R\sqsubseteq s\}\|}$.

Let $\mathcal{R} = \{R_1, \ldots, R_n\}$ be a set of sequential classification rules. For early prediction, a sequence tries to match a rule using a prefix as short as possible. Therefore, the *action rule* is the rule in $\mathcal{R}$ which s has the shortest minimum prefix of matching, that is, $actionR(s, \mathcal{R}) = \arg\min_{R_i\in\mathcal{R}} \|minprefix(s, R_i)\|$.

Given a sequence database SDB, the *cost of prediction* can be measured as the average cost per sequence, that is,

$$Cost(\mathcal{R}, SDB) = \frac{\sum_{s\in SDB} \|minprefix(s, action(s, \mathcal{R}))\|}{\|SDB\|}$$

To make the classifier accurate, we can confine that only sequential rules of confidence at least p_0 are considered, where p_0 is a user-specified accuracy expectation. Now, the problem is how to mine a set of rules $\mathcal{R}$ such that each rule is accurate (i.e., of confidence at least p_0) and the cost $Cost(\mathcal{R}, SDB)$ is as small as possible.

3.2 Feature Selection To form sequential classification rules, we need to extract features from sequences in the training data set. Particularly, we need to extract effective features for classification.

3.2.1 Utility Measure for Early Prediction We consider three characteristics of features for early prediction. First, *a feature should be relatively frequent*. A frequent feature in the training set may indicate that it is applicable to many sequences to be classified in the future. On the other hand, an infrequent feature may overfit a small number of training samples. Second, *a feature should be discriminative between classes*. Discriminative features are powerful in classification. Last, *we consider the earliness of features*. We prefer features to appear early in sequences in the training set.

Based on the above consideration, we propose a utility measure of a feature for early prediction. We consider a finite training sequence database SDB and a feature f.

The *entropy* of SDB is given by $E(SDB) = -\sum_{c\in\mathcal{L}} p_c \log p_c$ where $p_c = \frac{\|\{(s,c)\in SDB\}\|}{\|SDB\|}$ is the probability that a sequence is in class c in SDB.

Let $SDB_f = \{(s,c)|(s,c) \in SDB, f \sqsubseteq s\}$ be the subset of sequences in SDB where feature f appears. The difference of entropy in SDB and SDB_f, $E(SDB) - E(SDB_f)$, measures the discriminativeness of feature f.

To measure the frequency and the earliness of a feature f, we can use a weighted frequency of f. That is, for each sequence s where f appears, we use the minimum prefix of s where f appears to weight the contribution of s to the support of f. Technically, we have $wsup_{SDB}(f) = \frac{\sum_{f\sqsubseteq s, s\in SDB} \frac{1}{\|minprefix_{(s,f)}\|}}{\|SDB\|}$. Then, the *utility measure* of f is defined as

$$(3.1) U(f) = (E(SDB) - E(SDB_f))^w wsup_{SDB}(f)$$

In the formula, we use a parameter $w \geq 1$ to determine the relative importance of information gain versus earliness and popularity. This parameter carries the same spirit of those used in some previous studies such as [19].

3.2.2 Top-k Feature Selection Many existing rule-based classification methods such as [15, 14, 32] set some thresholds on feature quality measures like support, confidence, and discriminativeness, so that only those high quality (i.e., relatively frequent and discriminative) rules are selected for classifier construction. In our case, we may take a similar approach to set a utility threshold and mine all features passing the threshold from the training database.

However, we argue that such a utility threshold method is ineffective in practice. The utility values of effective features may differ substantially in various data sets. It is very hard for a user to guess the right utility threshold value. On the one hand, a too high utility threshold may lead to too few features which are insufficient to generate an accurate classifier. On the other hand, a too low utility threshold may lead to too many features which are costly to mine.

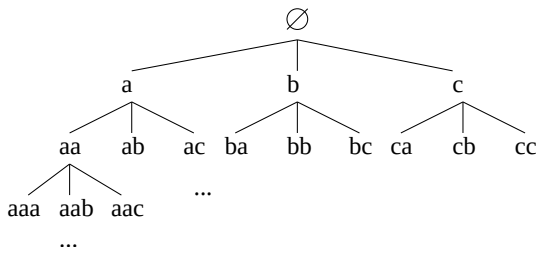

Figure 1: A sequence enumeration tree.

To overcome the problem, we propose a progressive approach. We first find top-k features in utility, and build a set of rules using the top-k features. If the rules are insufficient in classification, we mine the next k features. The progressive mining procedure continues until the resulting set of sequential classification rules are sufficient. As verified by our experimental results, on real data sets, we often only need to find a small number of rules, ranging from 20 on small data sets to 100 on large data sets, to achieve an accurate classifier.

Now, the problem becomes how to mine top-k features effectively for sequential classification rule construction.

Given a finite alphabet set Ω, all possible features as sequences in Ω^* can be enumerated using a sequence enumeration tree $T(\Omega)$. The root of the tree represents the empty feature $\emptyset$. Each symbol $x \in \Omega$ is a child of the root node. Generally, a length-l sequence s is the parent of a length-$(l+1)$ sequence s' if $s' = sx$ where $x \in \Omega$. Figure 1 shows a sequence enumeration tree of alphabet $\Omega = \{a, b, c\}$.

To find the top-k features, we search a sequence enumeration tree. As the first step, we select a set *Seed* of k features as the seeds.

To obtain the initial set *Seed* of seed features, we first select $k' < k$ length-1 features. That is, we compute the utility values of all length-1 features, and select the best k' features into *Seed*. Then, for each of those length-1 features f in *Seed*, we search the length-2 children of f, and select the best k' children as candidates. Among all the k'^2 length-2 candidate features, we select the best k' features and insert them into *Seed*. We use an auxiliary support threshold *min_sup* to prune features. A feature is not considered

if its support is lower than *min_sup*. The selection procedure continues iteratively level by level until no longer features can be added into the seed set. As the last step, we choose the best k features in the set *Seed*, and remove the rest.

Once we obtain a set of k seed features, we can use the seeds to prune the search space. Let $U_{lb} = \min_{f \in Seed} U(f)$ be the lower bound of the utility values of the features in the set *Seed*. For a feature f in the sequence enumeration tree, if the utility of the descendants of f can be determined no greater than U_{lb}, then the subtree of f can be pruned.

Then, for a feature f in the sequence enumeration tree, how can we determine whether a descendant of f may have a utility value over U_{lb}?

THEOREM 3.1. (UTILITY BOUND) *Let SDB be the training data set. For features f and f' such that f is a prefix of f',*

$$U(f') \leq \frac{E(SDB)^w}{\|SDB\|} \sum_{s \in SDB, f \sqsubseteq s} \frac{1}{minprefix(s,f)+1}$$

Proof. In the best case, all sequences in $SDB_{f'}$ belong to the same class. In such a case, $E(SDB_{f'}) = 0$. Thus, the gain in entropy is no greater than $E(SDB)^w$.

Since f is a prefix of f', $\|f'\| \geq \|f\| + 1$. Thus,

$$wsup_S DB(f') \leq \frac{\sum_{s \in SDB, f \sqsubseteq s} \frac{1}{minprefix(s,f)+1}}{\|SDB\|}$$

Both the gain in entropy and the weighted support are non-negative. Thus, using Equation 3.1, we have the upper bound in the theorem. ∎

If the descendants of f cannot be pruned by Theorem 3.1, we need to search the subtree of f. Once a feature whose utility value is greater than U_{lb} is found, we insert it into *Seed*, the set of seed features, and remove the feature in *Seed* whose utility value is the lowest. After inserting a better feature into *Seed* and removing a worse one from *Seed*, the lower bound U_{lb} of the top-k utility values is increased. The new lower bound U_{lb} is used to prune the search space.

When there are multiple nodes in the sequence enumeration tree whose utility values are over U_{lb} and whose subtrees need to be searched, we conduct a *best-first* search. That is, we first search the subtree of the node of the highest utility value, since heuristically it may have a good chance to provide good features. The search continues until all branches are pruned.

Since we search the sequence enumeration tree, which is the complete space of all possible features, and our pruning method guarantees no feature which is promising in the top-k list in utility is discarded, we have the following claim.

THEOREM 3.2. (COMPLETENESS OF TOP-k FEATURES) *The top-k feature selection procedure described in this section selects the top-k features in utility values.* ∎

3.3 Mining Sequential Classification Rules

Given a set of features $F = \{f_1, \ldots, f_k\}$, all possible rules using the features in F can be enumerated using a rule enumeration tree similar to a feature enumeration tree in spirit. The root of the tree is the empty set. The children of the root are the single features $f_i \in F$. The node f_i represents a set of rules $f_i \Rightarrow c$ ($c \in \mathcal{L}$). Generally, a node $R_1 : f_1 \rightarrow \cdots \rightarrow f_l$ represents a set of rules $f_1 \rightarrow \cdots \rightarrow f_l \Rightarrow c$ ($c \in \mathcal{L}$). Node $R_2 : f_1 \rightarrow \cdots \rightarrow f_l \rightarrow f_{l+1}$ is a child of R_1.

To construct a set of sequential classification rules of low prediction cost, we conduct a best-first search on the rule enumeration tree. A node in the rule enumeration tree can be in one of the five status: *inactive*, *active*, *chosen*, *pruned*, or *processed*. At the beginning, all nodes are inactive.

We consider rules with smaller prediction cost before rules with larger cost. We start with all nodes in the enumeration tree of single feature. For each node, we calculate the dominant class. For a node of feature f_i, if the dominant class in the sequences in SDB_{f_i} is c, then $f_1 \Rightarrow c$ is the rule at the node. We also calculate the prediction cost for each rule. All those nodes of single features are set to active.

Among the active nodes, we select a rule R of the lowest cost. If its confidence is at least p_0, where p_0 is the user-specified accuracy expectation, then the rule is chosen and added to the rule set $\mathcal{R}$. The status of R is set to chosen, and all descendants of R in the rule enumeration tree are set to pruned.

If the confidence of R is less than p_0, we consider all children of R by adding one feature at the end of R. We calculate the dominant class, the support and the prediction cost for each child node. The support of R' that is a child of R is defined as $sup(R') = \frac{\|\{s | s \in SDB, R' \sqsubseteq s, \nexists R'' \in \mathcal{R} \text{ st. } minprefix(s,R'') \leq minprefix(s,R')\}\|}{\|SDB\|}$. It means if a sequence s can be matched by rule R', and the sequence cannot be matched by any rule R'' already in the rule set with a lower cost, s contributes one vote for the support of R'. Otherwise, if a s matches both R' and R'', and $minprefix(s, R'') \leq minprefix(s, R')$, then s should not contribute to the support of R'.

A child node is set to active if its support is at least *min_sup* (the auxiliary support threshold). Otherwise, the child node and its descendants are set to pruned. After the expansion, node R is set to processed.

Once a rule R is chosen and added into the rule set $\mathcal{R}$, we need to adjust as follows the support and the confidence of the other active rules in the rule

enumeration tree. If a sequence s uses another rule R' as the action rule before R is chosen, and uses R as the action rule because s has the shortest minimum prefix matching R, then the support of R' is decreased by 1. After the adjustment, a node R' and its descendants may be pruned if the support of R' is lower than min_sup.

The above best-first search will terminate due to two facts. First, as the active rules grow to longer rules, the supports monotonically decrease. Second, once a rule is found, the supports of some active rules will also decrease. The number of sequences in the training data set is finite. Thus, the search procedure will terminate.

After the best-first search where a set of rules $\mathcal{R}$ is selected, there still can be some sequences in SDB which do not match any rules in $\mathcal{R}$. We need to find new features and new rules to cover them. We consider the subset $SDB_{\bar{\mathcal{R}}} \subseteq SDB$ which is the set of sequences not matching any rules in $\mathcal{R}$. We select features and mine rules on $SDB_{\bar{\mathcal{R}}}$ using the feature selection procedure and the rule mining procedure as described before. The only difference is that, when computing the utility of features, we still use $(E(SDB) - E(SDB_f))$ as the entropy gain, but use $wsup_{SDB_{\bar{\mathcal{R}}}}(f)$ as the weighted support. The reason is that the feature selected should be discriminative on the whole data set. Otherwise, some biased or overfitting features may be chosen if we use $(E(SDB_{\bar{\mathcal{R}}}) - E(SDB_f))$ as the entropy gain.

Iteratively, we conduct the feature selection and rule mining until each sequence in the training data set SDB matches at least one rule in $\mathcal{R}$. The rule set $\mathcal{R}$ is used as the classifier.

When a set of rules $\mathcal{R}$ is used in classification, a sequence is matched against all rules in $\mathcal{R}$ simultaneously, until the first rule is matched completely. This earliest matched rule gives the prediction.

3.4 Summary In this section, we give a sequential classification rule approach towards early prediction. The major idea is to mine a set of sequential classification rules as the classifier. Discriminative features are selected to construct rules. Earliness is considered in both feature selection (through weighted support) and rule mining (through prediction cost). An auxiliary support threshold is used to avoid overfitting. We conduct best-first search for the sake of efficiency. The search of features is complete and the search of rules is greedy.

4 The Generalized Sequential Decision Tree Method

The decision tree model [22, 23] is popularly used for classification on attribute-value data which does not consider any sequential order of attributes. It is well recognized that decision trees have good understandability. Moreover, a decision tree is often easy to construct.

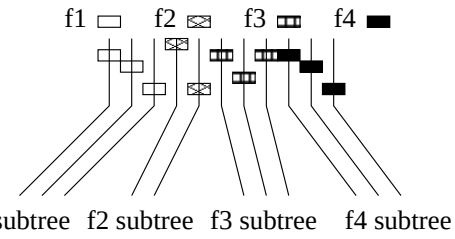

Figure 2: The idea of using a set of features as an attribute in GSDT.

Unfortunately, the classical decision tree construction framework cannot be applied straightforwardly to sequence data for early prediction. There are not natural attributes in sequences since sequences are not attribute-value data. Thus, the first challenge is how to construct some "attributes" using features in sequences. One critical difference between attribute-value data and features in sequences is that, while an attribute is defined in every record of an attribute-value data set such as a relational table, a feature may appear in only a relatively small number of sequences. Thus, a feature cannot be used directly as an attribute.

Moreover, in order to achieve early prediction, when a feature is selected as an attribute in a sequential decision tree, we cannot wait to scan the whole sequence to determine whether the feature appears or not. Instead, we need to aggressively integrate the early prediction utility in attribute composition.

In this section, we develop a generalized sequential decision tree (GSDT for short) method.

4.1 The GSDT Framework As the critical idea of attribute construction in GSDT, we use a set of features A as an attribute such that at least one feature in A likely appears in a sequence to be classified. In classification of a sequence s, once a feature in A is matched by a minimum prefix in s, s can be moved to the corresponding child of the node for further classification. Figure 2 illustrates the intuition.

Ideally, we can choose a set of features A as an attribute such that each sequence to be classified has one and only one feature in A. Such an attribute mimics a node in a classical decision tree perfectly. Unfortunately, such an ideal case may not happen in practice. Due to the existence of noise and the incompleteness of training data, a sequence may contain none, one, or multiple features in A.

To tackle the problem, in GSDT, we allow a sequence in the training set to contain more than one feature from A, and simultaneously ensure that each se-

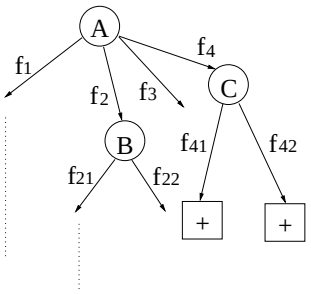

Figure 3: A GSDT.

quence in the training set contains at least one feature in A. By covering the training data set well, we hope that when the resulting decision tree is applied for classification of a unseen sequence, the sequence may likely have at least one feature from A.

Figure 3 shows a GSDT example. At the root node, a set of features $\{f_1, f_2, f_3, f_4\}$ are chosen to form an attribute A. As will be explained in Section 4.2, we ensure that every sequence in the training set contains at least one feature from A, and allow more than one feature to appear in a sequence.

Attribute A divides the sequences in the training set into 4 subsets: SDB_{f_1}, SDB_{f_2}, SDB_{f_3}, and SDB_{f_4}. Each feature and the corresponding subset form a branch. Different from a classical decision tree, the four subsets are not disjoint. A sequence s in the training set is in both SDB_{f_1} and SDB_{f_2} if both features f_1 and f_2 appear in s. Therefore, GSDT allows redundancy in covering the training data set.

The redundancy in covering the training set in fact improves the robustness of GSDT. Due to the existence of noise, a feature may randomly appear or disappear in a sequence by a low probability. If a sequence is captured by more than one feature and thus more than one branch in GSDT, the opportunity that the sequence is classified correctly by GSDT is improved.

Although we select the features at the root node in a way such that each sequence in the training set contains at least one feature at the root node, it is still possible that an unseen sequence to be classified in the future does not contain any feature at the root node. To handle those sequences, we also store the majority class (i.e., the class of the largest population) in the training data set at the root node. If a sequence does not match any feature at the root node, the GSDT predicts its class using the majority class.

Once the root node is constructed, the subtrees of the branches of the root node can be constructed recursively. For example, in Figure 3, the branch of feature f_2 is further partitioned by choosing a set of features $B = \{f_{21}, f_{22}\}$.

If a branch is pure enough, that is, the majority class has a population of at least p_0 where p_0 is the user specified parameter to express accuracy expectation, a leaf node carrying the label of the majority class is added. In Figure 3, the branch f_4 is further split by attribute $C = \{f_{31}, f_{32}\}$, and two leaf nodes are obtained.

To avoid overfitting, similar to the classical decision tree construction, we stop growing a branch if the training subset in the branch has less than min_sup sequences, where min_sup is the minimum support threshold.

4.2 Attribute Composition Now, the only remaining issue is how to select a set of features as an attribute.

Consider a training set SDB, we need to find a set of features which have high early prediction utility and cover all sequences in SDB. We also want the set of features as small as possible to avoid overfitting.

We can extract features of high early prediction utility using the top-k feature selection method described in Section 3.2.2. However, finding a minimal set of features covering all sequences in the training set is NP-hard due to the minimum cover problem [6].

Here, we adopt a greedy approach. We set $A = \emptyset$ and $SDB' = SDB$ at the beginning. We consider the top-k features of the highest early prediction utility extracted using the method in Section 3.2.2. If a sequence in SDB' matches a feature in A, then the sequence is removed from SDB'. We iteratively add to A the feature which has the largest support in SDB'. Such a feature matches the largest number of sequences in the training set that do not have any feature in the current A. The iteration continues until SDB' is empty. If the k features are used up but SDB' is not empty yet, another k features are extracted.

In the classical decision tree construction, at each node, an attribute of the best capability to discriminate different classes is chosen. In GSDT, importantly only the features of high early prediction utility are considered for attribute composition. As explained in Section 3.2.1, the utility measure defined in Equation 3.1 captures the capability of discriminating different classes and the earliness of prediction simultaneously. Using only the features of high utility in attribute composition ensures the effectiveness of GSDT in classification accuracy and earliness.

4.3 Summary In this section, we develop GSDT, a generalized sequential decision tree method for early prediction. The central idea is to use a (small) set of features having high early prediction utility to compose

an attribute. Redundancy is allowed in covering the sequences in the training set to improve the robustness of GSDTs. A GSDT is easy to construct.

Similar to the existing decision tree construction methods, the hypothesis space of the GSDT method is complete, i.e., all possible GSDTs are considered. The GSDT method maintains only one current hypothesis and does not backtrack. In each step, the GSDT method uses all training examples and is insensitive to errors in individual training examples.

Importantly, GSDT has good understandability. The features appearing at the nodes closer to the root are those likely appearing early in sequences and effective for classification. This kind of information is not captured by the sequential classification rule (SCR) approach.

5 Empirical Evaluation

In this section, we report an empirical study on our two methods using 3 real data sets and 1 synthetic data set across different application domains. All the experiments were conducted using a PC computer with an AMD 2.2GHz CPU and 1GB main memory. The algorithms were implemented in Java using platform JDK^{TM} 6.

5.1 Results on DNA Sequence Data Sets DNA promoter data sets are suitable for testing our feature extraction techniques for early prediction, because there are explicit motifs on promoters. Also, the sequential nature of DNA sequences can be used to simulate the temporal orders. We use two promoter data sets of prokaryotes and eukaryotes respectively in our experiments.

5.1.1 Results on the E.Coli Promoters Data Set The *E. Coli* Promoters data set was obtained from the UCI machine learning repository [2]. The data set contains 53 *E. Coli* promoters instances and 53 non-promoter instances. Every instance contains 57 sequential nucleotide ("base-pair") positions.

Along with the data set, the results of some previous classification methods [28] on it are also provided. The accuracy of those results are obtained by using the "leave-one-out" methodology. For the comparison purpose, we also use "leave-one-out" in our experiments. In methods SCR and GSDT, we set the maximal length of a feature to 20. In each round of feature extraction, we extract the top-30 features with the highest utility scores. The results in Table 1 are obtained by setting the accuracy parameter as $p_0 = 94\%$, minimal support threshold as $s_0 = 5\%$, and the weight in Equation 3.1 as $\omega = 3$. Those three parameters are user-specified instead of being learned automatically from data. Parameters selection will be discussed in session 5.1.3.

Method	Error rate	Comments
KBANN	$\frac{4}{106}$	a hybrid machine learning system
SCR	$\frac{7}{106}$	Average prefix length for prediction: $\frac{21}{57}$
BP	$\frac{8}{106}$	standard artificial neural network with back-propagation using one hidden layer
GSDT	$\frac{10}{106}$	Average prefix length for prediction: $\frac{20}{57}$
O'Neill	$\frac{12}{106}$	ad hoc technique from the bioinformatics literature
3-NN	$\frac{13}{106}$	a nearest-neighbor method
ID3	$\frac{19}{106}$	[22]

Table 1: The accuracy of several methods on the *E. Coli* Data set. Except for SCR and GSDT, all methods use the entire sequences in classification.

In Table 1, the results from our methods and the previous methods are listed in the error rate ascending order. Our two methods can obtain competitive prediction accuracy. Except for our two methods, the other methods all use every nucleotide position as a feature and thus cannot give early prediction. Our two methods use very short prefixes (up to 36.8% of the whole sequence on average) to give early prediction.

Please note that, when counting the error rate of SCR and GSDT in Table 1, if a testing sequence cannot be classified using a sequential classification rule or a path in the GSDT (i.e., the sequence does not match any feature in a node of a path), we treat the case as an error. In other words, we test the exact effectiveness of the sequential classification rules generated by SCR and the decision tree generated by GSDT. Thus, the error rate reported in Table 1 is the upper bound. In practice, the two methods can make prediction on an unmatched sequence using the majority class.

Since the data set is small, the running time of GSDT and SCR is 294 and 340 milliseconds, respectively. SCR is slightly more accurate than GSDT but uses a slightly longer prefix on average.

Some rules generated by SCR and some paths generated in GSDT are interesting. For example, rules (paths) $TATAA \Rightarrow$promoter and $ATAAT \Rightarrow$promoter generated by both SCR and GSDT are highly meaningful in biology, since $TATAAT$ is a highly conserved

Method	Accuracy	Avg. prefix len	Runtime
SCR	87%	83.56 /300	142 seconds
GSDT	86%	85.64 /300	453 seconds

Table 2: Results on the drosophila promoters data set.

region on *E. Coli* promoters [8].

5.1.2 Results on the Drosophila Promoters Data Set The Berkeley Drosophila Genome Project provides a large number of drosophila promoter sequences and non-promoter coding sequences (CDS). We use a set of promoter sequences of 327 instances and a set of CDS of 527 instances provided in year 2002 by the project to test our two methods. The data set is obtained from `http://genomebiology.com/2002/3/12/research/0087.1`.

The length of each sequence is 300 nucleotide base pairs. Compared to the *E. Coli* data set, this data set has longer sequences and more instances. *Drosophila* (fruit fly) is a more advanced species than *E. Coli* (bacteria), which is expected to have more complex mechanics on promoters. Those differences make the feature extraction in this data set more challenging than that in the *E. Coli* data set.

10-fold cross validation is used to test the two methods. When setting the accuracy parameter as $p_0 =$ 90% and 95%, respectively, for SCR and GSDT, the minimal support threshold as $s_0 = 10\%$, and $\omega = 8$, the accuracy and the runtime of SCR and GSDT obtained are shown in Table 2.

SCR and GSDT reach similar accuracy, and both use a short prefix in prediction, around 28% of the whole sequence on average. GSDT is much slower than SCR because the rules discovered for drosophila promoters are more complex than those in the *E. Coli* data set. SCR only needs to conduct feature extraction once, but GSDT has to extract features for each node in the decision tree. Moreover, GSDT has to select features to build attributes.

5.1.3 Parameters selection On the *E. Coli* data set, we test the effect of parameter ω in Equation 3.1 on the accuracy and the early prediction of SCR and GSDT using the setting described before except for varying ω. Figure 4 shows the accuracy of the two methods with respect to ω. When $\omega = 3$, both methods reach the highest accuracy. When $\omega > 3$, the accuracy is insensitive to ω.

Figure 5 shows the average length of prefix with respect to ω. SCR and GSDT use the shortest prefix length on average in early prediction when ω is 2 and 3, respectively. When $\omega > 3$, the average length of prefix used in prediction is insensitive to ω. On the drosophila promoters data set, similar experiments are conducted, and the results are shown in Figures 6 and 7. We observe that the best prediction accuracy is achieved by SCR when $\omega = 8$, and by GSDT when $\omega = 10$.

On the drosophila promoters data set, we test the effect of accuracy parameter p_0 on accuracy and average length of prefix in prediction. The results are shown in Figures 8 (for SCR) and 9 (for GSDT).

The figures show the accuracy, the average length of prefix in percentage, the unmatch rate (i.e., the percentage of the test sequences which do not match any rule in SCR or any path in GSDT, and are assigned to the majority class). Figure 8 also shows the rule error rate which is the percentage of test sequences incorrectly classified by some sequential classification rules. Correspondingly, Figure 9 shows the tree error rate which is the percentage of the test sequences incorrectly classified by some leaf nodes in the decision tree. The figures are obtained by using the default settings described before except for varying the p_0 value.

When p_0 increases, the rule error rate of SCR and the tree error rate of GSDT decrease. The reason is that the accuracy of each rule in SCR and each path in GSDT increases. However, the unmatch rates in both methods increase, since the number of qualified rules and that of tree paths decrease. The overall effect is that the accuracy of SCR and GSDT is high when choosing a modest p_0. The accuracy is not good when p_0 is either too low or too high. Choosing $p_0 = 0.9$ as a starting point and adjusting it to find the optimal result is the way we used in all the experiments.

In those two methods, the support of a rule is not required high. Usually, s_0 is set between 5% and 10%.

5.2 Results on Time Series Data Set Early prediction is important for the sequences capturing temporal relationships. In this subsection, we use two time series data sets to test the efficacy of our methods.

The data sets we used contain continuous data. Since our methods are designed for categorical sequences, we preprocess the time series as follows. We apply k-means discretization on the training data set to get the discretization threshold, and transform the training data set into discrete sequences. The test data set is discretized using the same threshold learned from the training data set. In classification, the discretization is performed online. The results in the following session are obtained by setting $k = 3$ for the discretization.

5.2.1 Results on the Synthetic Control Data set The synthetic control data set from the UCI machine

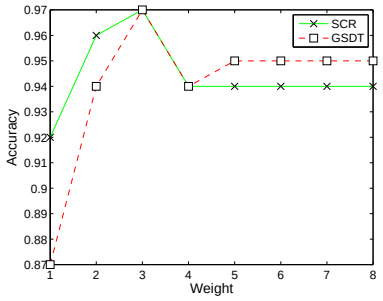

Figure 4: Accuracy with respect to weight on the *E. Coli* data set.

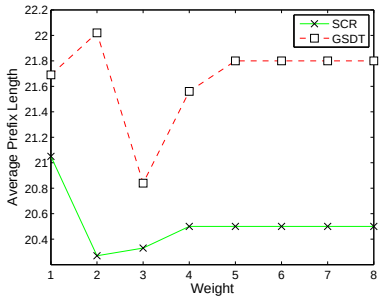

Figure 5: Average length of prefix with respect to weight on the *E. Coli* data set.

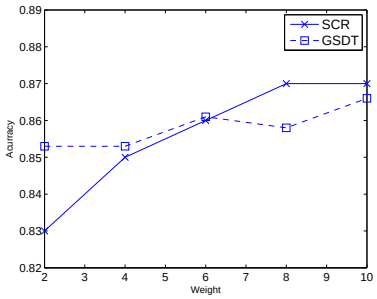

Figure 6: Accuracy with respect to weight on the Drosophila data set.

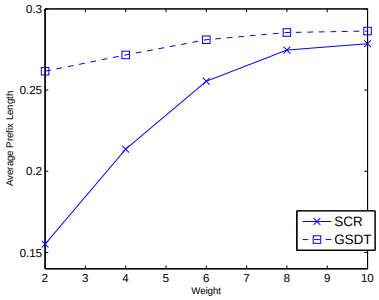

Figure 7: Average length of prefix with respect to weight on the Drosophila data set.

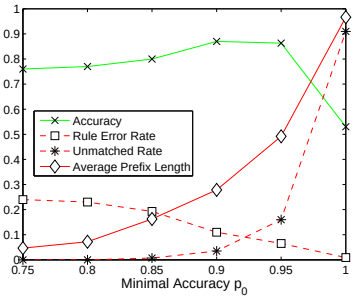

Figure 8: Performance of SCR with respect to p_0.

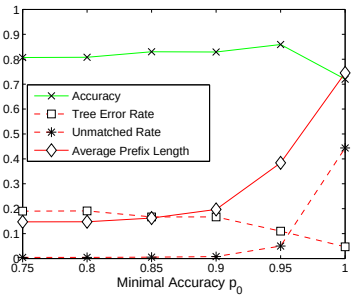

Figure 9: Performance of GSDT with respect to p_0.

learning repository [2] contains 600 examples of control charts synthetically generated by the process. There are six different classes of control charts: normal, cyclic, increasing trend, decreasing trend, upward shift, and downward shift. Every instance contains 60 time points.

We randomly split the data set into two subsets of the same size, i.e., 300 instances as the training examples, and the other 300 instances as the testing examples. We set $p_0 = 0.95$, $w = 3$, $s_0 = 10\%$, $k = 30$, and the maximal length of features as 20 to obtain the results in Table 3. [10] shows that, for time series classification, one-nearest-neighbor (1NN) with Euclidean distance is very difficult to beat. In Table 3, the results of 1NN classification with Euclidean distance using the whole sequences and using prefixes with different lengthes are shown for comparison.

From the results of 1NN classification, we can see that there is a big accuracy drop from using prefixes of length 40 to using prefixes of length 30. It indicates that the prefixes till length 40 contain the important features for classification. In GSDT and SCR, the average length used in prediction is 27 and 33. It demonstrates that our feature selection procedure can capture the key features within suitable length. GSDT and SCR perform better on sequences in classes normal, cyclic, increasing trend and decreasing trend than in class upward shift and downward shift. The reason is that upward shift and downward shift sequences are very similar to increasing and decreasing sequences, respectively, especially after discretization. To correctly classify the upward and downward shift sequences, a classifier has to capture the shift regions accurately. However, the shift regions are similar to some noise regions in other classes, which make the early prediction inaccurate.

SCR and GSDT do not perform as well as 1NN on this data set for early prediction. One major reason is that SCR and GSDT are designed for categorical sequences.

If we remove the sequences in the classes of upward shift and downward shift, the performance of the two methods, as shown in Table 4, improve substantially. The accuracy is improved further and the average length

Method	Normal	Cyclic	Increasing	Decreasing	Upward shift	Downward shift	Avg. prefix len.
SCR	0.96	0.96	0.80	0.76	0.36	0.46	33/60
GSDT	0.98	0.88	0.92	0.96	0.42	0.36	27/60
1NN	1	1	1	0.98	0.94	0.9	60/60
1NN (on prefix)	0.48	1	1	1	0.54	0.54	30/60
1NN (on prefix)	0.98	1	0.98	0.98	0.82	0.74	40/60
1NN (on prefix)	1	1	1	0.98	0.9	0.84	50/60

Table 3: Results on the Synthetic Control data set.

Method	Normal	Cyclic	Increasing	Decreasing	Avg. prefix len.
SCR	0.96	0.90	0.98	1.00	13/60
GSCT	0.96	0.92	0.98	1.00	15/60
1NN	1	1	1	1	60/60
1NN (on prefix)	0.84	1	0.92	0.96	20/60
1NN (on prefix)	1	1	1	1	30/60

Table 4: Results on the Synthetic Control data set without upward/downward shift sequences.

of prefix is shortened remarkably. Our two method cannot beat 1NN when the prefix is longer than 30. But when the prefix is reduced to 20, SCR and GSDT outperform $1NN$ by using average prefixes of 13 and 15 respectively.

The features at the root node of a GSDT have good utility in early prediction. In the complete synthetic control data set (i.e., containing sequences of all 6 classes), on average, a sequence matches one of the features at the root node with a prefix of length 14.54, which is substantially shorter than the length of the whole sequence (60). Moreover, when the sequences in classes upward/downward shift are excluded, the average length of prefix matching one of the features in the root node is further reduced to 11.885.

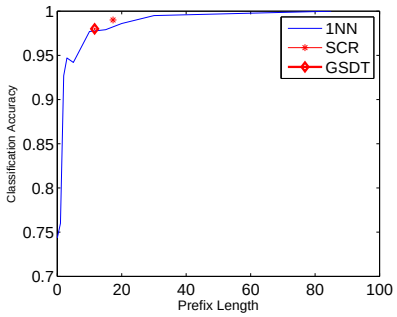

Figure 10: Results Comparison on ECG Data.

5.2.2 Results on Physiology ECG Data Set PhysioBank (`http://physionet.org/physiobank/`) provides a large amount of time series physiology data. A set of ECG data from physioBank is normalized and used in [30]. We downloaded the data set from `http://www.cs.ucr.edu/~wli/selfTraining/`. The data set records the ECG data of abnormal people and normal people. Each instance in the normalized data set contains 85 time points. As same as in [30], we use 810 instances as the training examples and $1,216$ instances as the test examples.

In Figure 10, the prediction accuracy by using various lengths of prefixes in 1NN is shown by the curve. From the result, we can see this data set has highly distinguishing feature around the very beginning of the sequence. When $p_0 = 99\%$, $w = 3$, $k = 30$, and $s_0 = 10\%$, SCR reaches a prediction accuracy as 0.99 using average prediction prefix of length 17.3 out of 85, which is the asterisk point in the figure. Under the same setting, GSDT reaches an accuracy of 0.98 by using an average prefix of length 11.6, which is the diamond point in the figure. Both methods can make prediction with competitive accuracy compared to 1NN and using a very short prefix on average. SCR takes 6.799 seconds for training and predication, while GSDT takes 491.147 seconds.

5.3 Summary In this section, we report an empirical study on DNA sequence data sets and time series data sets. The results clearly show that SCR and GSDT can obtain competitive prediction accuracy using an often remarkably short prefix on average. Early prediction is highly feasible and effective. SCR tends to get a better accuracy, and GSDT often achieves earlier prediction. SCR is often faster than GSDT.

6 Conclusions

In this paper, we identify the novel problem of mining sequence classifiers for early prediction, which has several important applications. As the first attempt to tackle the challenging problem, we propose two interesting methods: the sequential classification rule (SCR) method and the generalized sequential decision tree (GSDT) method. A so-called early prediction utility is used to select features to form rules and decision trees. An extensive empirical evaluation clearly shows that our two methods can achieve accuracy comparable to the state-of-the-art methods, but often make prediction using only very short prefixes of sequences. The results clearly indicate that early prediction is highly feasible and effective.

Although we made some good initial progress in early prediction, the problem is still far from being solved. For example, more accurate methods should be developed and methods on continuous time series are needed.

References

[1] C. J. Alonso and J. J. Rodríguez. Boosting interval based literals: Variable length and early classification. In *Data Mining in Time Series Databases*. World Scientific, 2004.

[2] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[3] S. M. Berget. Exon recognition in vertebrate splicing. *Journal of Biological Chemistry*, 270(6):2411–2414, 1995.

[4] G. Dong and J. Pei. *Sequence Data Mining*. Springer, USA, 2007.

[5] R. Durbin *et al. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[6] M. Garey and D. Johnson. *Computers and Intractability: a Guide to The Theory of NP-Completeness*. Freeman and Company, New York, 1979.

[7] W. N. Grundy *et al.* Meta-MEME: motif-based hidden Markov models of protein families. *Computer Applications in the Biosciences*, 13(4):397–406, 1997.

[8] C. B. Harley and R. P. Reynolds. Analysis of *E. Coli* promoter sequences. *Nucleic Acids Res.*, 15(5):2343-2361, 1987.

[9] S. H. Huang *et al.* Prediction of Outer Membrane Proteins by Support Vector Machines Using Combinations of Gapped Amino Acid Pair Compositions. In *BIBE'05*.

[10] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. In *KDD'02*.

[11] M. Kozak et al. Compilation and analysis of sequences upstream from the translational start site in eukaryotic mRNAs. *Nucleic Acids Res*, 12(2):857–872, 1984.

[12] A. Krogh *et al.* Hidden Markov models in computational biology. Applications to protein modeling. *J Mol Biol*, 235(5):1501–31, 1994.

[13] N. Lesh *et al.* Mining features for sequence classification. In *KDD '99*.

[14] W. Li *et al.* CMAR: Accurate and efficient classification based on multiple class-association rules. In *ICDM'01*.

[15] B. Liu *et al.* Integrating classification and association rule mining. In *KDD'98*.

[16] A. V. Lukashin and M. Borodovsky. GeneMark.hmm: new solutions for gene finding. *Nucleic Acids Research*, 26(4):1107–1115, 1998.

[17] Q. Ma *et al.* DNA sequence classification via an expectation maximizationalgorithm and neural networks: a case study. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 31(4):468–475, 2001.

[18] A. Nandi and H. V. Jagadish Effective phrase prediction. In *VLDB'07*.

[19] M. Núnez. The use of background knowledge in decision tree induction. *Mach. Learn.*, 6(3):231–250, 1991.

[20] K. J. Park and M. Kanehisa. Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs. *Bioinformatics*, 19(13):1656–1663, 2003.

[21] D. S. Prestridge. Predicting Pol II promoter sequences using transcription factor binding sites. *Journal of Molecular Biollogy*, 249(5):923–32, 1995.

[22] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[23] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[24] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Readings in speech recognition*, Morgan Kaufmann, 1990.

[25] R. She *et al.* Frequent-Subsequence-Based Prediction of Outer Membrane Proteins. In *KDD'03*.

[26] C. W. Smith and J. Valcarcel. Alternative pre-mRNA splicing: the logic of combinatorial control. *Trends Biochem. Sci*, 25(8):381–388, 2000.

[27] S. Sonnenburg *et al.* Learning interpretable SVMs for biological sequence classification. *RECOMB'05*.

[28] Geofrey G. Towell *et al.* Refinement of approximate domain theories by knowledge based neural network. In *AAAI'90*.

[29] V. N. Vapnik. *Statistical learning theory*. Wiley, 1998.

[30] Li Wei and E. Keogh. Semi-supervised time series classification. In *KDD'06*.

[31] C. Wu *et al.* Neural networks for full-scale protein sequence classification: Sequence encoding with singular value decomposition. *Machine Learning*, 21(1):177–193, 1995.

[32] X. Yin and J. Han. CPAR: Classification based on predictive association rules. In *SDM'2003*.

[33] M. Zhang *et al.* Mining progressive confident rules. In *KDD'06*.