A Temporal Abstraction Framework for Classifying Clinical Temporal Data

Iyad Batal[1] **Lucia Sacchi[2]** **Riccardo Bellazzi[2]** **Milos Hauskrecht[1]**
iyad@cs.pitt.edu **lucia.sacchi@unipv.it** **riccardo.bellazzi@unipv.it** **milos@cs.pitt.edu**
[1] Department of Computer Science, University of Pittsburgh, PA; [2] Dipartimento di Informatica e Sistemistica, University of Pavia, Italy.

Abstract

The increasing availability of complex temporal clinical records collected today has prompted the development of new methods that extend classical machine learning and data mining approaches to time series data. In this work, we develop a new framework for classifying the patient's time-series data based on temporal abstractions. The proposed STF-Mine algorithm automatically mines discriminative temporal abstraction patterns from the data and uses them to learn a classification model. We apply our approach to predict HPF4 test orders from electronic patient health records. This test is often prescribed when the patient is at the risk of Heparin induced thrombocytopenia (HIT). Our results demonstrate the benefit of our approach in learning accurate time series classifiers, a key step in the development of intelligent clinical monitoring systems.

Introduction

Advances in data collection and electronic health record technologies have led to the emergence of complex clinical datasets, where data instances consist of sequences of clinical findings, lab values, measurements and medications. Such multivariate time series data provide us with a complex temporal characterization of the patient case.

There is an enormous utility of using these temporal clinical datasets to learn a variety of classification models. By ignoring the temporal aspect of data, the patient case can be relatively easily described using the most recent set of values, e.g. "a low blood pressure", or "a high white blood cells count". However, this information may be limited in describing the patient case. For example, the information important for a proper assessment may include simple trends, such as "an increase in blood pressure", or more complex temporal patterns such as "a low blood pressure following the prescription of a certain medication". Clearly, incorporating more complex temporal information may help improving our ability to classify the patient case.

Unfortunately, learning accurate classification models from temporal clinical datasets poses a number of challenges. The main challenge is that the number of temporal features (patterns) one may generate to characterize the multivariate time series data can be enormous. Typically, only a very small fraction of these may turn out to be useful for the classification task. Hence, it is very important to develop techniques capable of identifying temporal features useful for classification.

The objective of this work is to develop a framework capable of automatically generating temporal features for classifying clinical multivariate time series data. We use the temporal abstractions framework[1] to obtain a qualitative description of time series using trend and value abstractions. These basic abstractions are then combined using temporal logic relations to form complex temporal patterns. Our *STF-Mine* (Segmented Time series Feature Mine) algorithm extends the *Apriori* algorithm[2], used in frequent pattern mining, to mine frequent temporal patterns from abstracted time series. After identifying the most frequent temporal patterns for each class, *STF-Mine* selects those patterns that are highly discriminative for the target classes, and uses them to define a new feature space for representing the data.

We test our method by predicting orders of the Heparin Platelet Factor 4 antibody test (HPF4) from electronic patient health records. This test is prescribed when the patient is at the risk of Heparin induced thrombocytopenia (HIT). We show that our method is capable of automatically identifying temporal patterns useful for detecting HPF4 orders, hence the risk of HIT. We demonstrate that using the *STF-Mine* patterns lead to classification models that outperform models that ignore temporal information and rely solely on the recent set of lab values.

Methodology

Our work deals with multivariate time series (MTS) data extracted from patients' medical records, where each data instance is formed by sequences of observations (time-series) from multiple clinical variables. A key characteristic of such time series data is that they are irregularly sampled in time.

The objective of our work is to develop methods suitable for classifying these medical records. We

reduce this complex MTS data into a fixed-length feature vector representation. The features correspond to temporal patterns important for the classification task. Briefly, our *STF-Mine* algorithm (Figure 1) takes as an input training (labeled) time series training data and it outputs a set of frequent and discriminative temporal patterns. It consists of the following steps:

1. Segment the time series using temporal abstractions.
2. Generate the frequent patterns of each class (category) from the abstracted series.
3. Select the frequent patterns that are highly discriminative between the classes and use them as classification features.

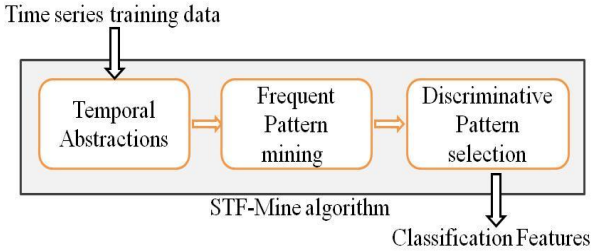

Figure 1: the flowchart of the STF-Mine algorithm

The features generated by *STF-Mine* let us map each MTS instance into a Boolean feature vector. After this transformation, we can apply any of the classical classification techniques to learn and predict the class of future MTS examples. In the following, we explain these steps in more depth.

Qualitative Time Series Abstraction: Our first step is to obtain a qualitative description of the raw time series data. Given an MTS instance, the time series of each variable is converted into a sequence of abstract states s_i, where each state represents a property that holds during an interval $[b_i, e_i]$. The alphabet Σ represents all possible values for the states.

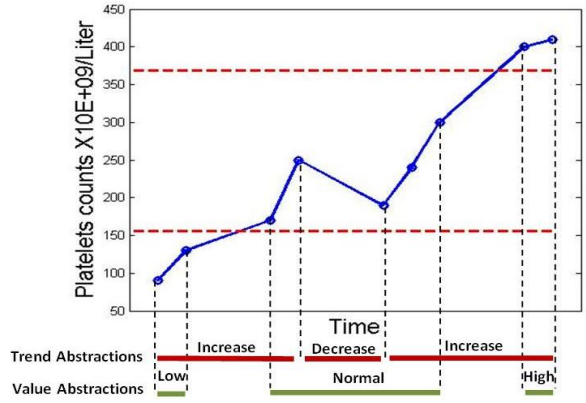

Figure 2: An example illustrating trend and value abstractions.

In this work, we use two types of abstractions: (1) *trend abstractions*, which represent the time series in terms of its local trends: Σ = {*increasing*, *steady*, *decreasing*}.
(2) *value abstractions*: Σ = { *high*, *normal*, *low*}.

To obtain trend abstractions, we segment the series by using the sliding window method[3], which keeps expanding each segment until its error exceeds a user-specified threshold. The trend is then defined according to the slope of the fitted segment.

Figure 2 shows an example of a time-series for platelet counts (PLT) and its corresponding trend and value abstractions; the two horizontal dashed lines represent the normal range for PLT.

Temporal Patterns: The basic temporal abstractions (extracted from each variable) can be combined to form more complex temporal patterns. To construct these complex patterns, we need to define the temporal relations between state intervals. Allen's temporal logic[4] describes temporal relations for any pair of intervals using 13 possible relationships. However, the majority of these require equality of two or more interval end points. The time information in clinical datasets is only rarely precise enough to be aligned to a specific time point, hence expressing patterns using all Allen's interval relations may not be the best solution and may hinder the discovery process. To alleviate the problem, we choose to model only two relationships: *before* and *overlaps*, which we redefine as follows:

Given two states, *A* and *B* with intervals $[a_1,a_2]$ and $[b_1,b_2]$

- *A before B* iff $a_2 <= b_1$
- *A overlaps B* iff $a_1 <= b_1$ and $a_2 > b_1$, i.e. *A* starts before *B* and their intervals overlap.

We define the temporal pattern as a sequence of states (intervals) related using temporal relationships. We denote each pattern by a pair of vectors (*S*, *R*), where S_i corresponds to the i^{th} state of the pattern and R_i to the temporal relation between states S_i and S_{i+1}, i.e., $R_i = rel(S_i, S_{i+1})$: $rel \in \{before, overlaps\}$.

We define the size of pattern *P* to be the number of states it contains. If size(*P*)= *k*, we say that *P* is a *k-pattern*. In our notation, the symbol between brackets refers to the variable (time series) from which the state has been extracted. To give an example, assume each data instance contains two variables *X* and *Y*. An example of a temporal *2-pattern* is: *P= increase*[*X*] *before decrease*[*Y*], which corresponds to an increase in variable *X* followed by a decrease in variable *Y*.

Interesting patterns are usually limited in their temporal extension. Hence, we allow the user to

specify the maximum pattern duration that is of interest (w). This parameter serves as the width of the sliding window used in the frequent pattern mining algorithm. The algorithm only considers the patterns that can be observed within this window. If the user does not specify w, the algorithm treats the whole time series as a single window. This case could be of interest if the time series are short.

Let T denotes an MTS instance after being converted into a sequence of states and E denotes the states of T visible within window w. We say that pattern P (S, R) holds in T within w, denoted as $P \in (T,w)$, if there is an injective mapping π_i from the states of P to the states of T such that:

$$\forall i \in \{1..size(p)-1\}: S_i = E_{\pi(i)}\ and$$
$$S_{i+1} = E_{\pi(i+1)}\ and\ R_i = rel(E_{\pi(i)}, E_{\pi(i+1)})$$

Now, we define the support of pattern P in a time series database D using window size w to be the number of instances from D where P holds within any position of the sliding window.

Frequent Temporal Pattern Mining: In this section, we describe the algorithm that mines the frequent temporal patterns from the multivariate state sequences (obtained from the temporal abstractions).

The algorithm is based on the *Apriori* algorithm[2] and is applied to each class of examples separately. We rely on the *Apriori property* to reduce the search space. This property states that the support of a pattern is always less than or equal to the support of any of its subpatterns.

The frequent mining algorithm takes three inputs: 1) D_{ci}: the set of all abstracted multivariate instances belonging to a specific class (c_i), 2) *min_sup*: a user specified threshold on the support of frequent patterns and 3) w: the sliding window width (the maximum pattern duration). The algorithm outputs all temporal patterns P_j that are frequent in class c_i, i.e., which satisfy: $sup(P_j, D_{ci}, w) >= min\_sup$.

Candidate Generation: A candidate *(k+1)-pattern* is generated by joining two frequent *k-patterns* which share the same *k*-1 states as a prefix. Assume we are joining frequent patterns P_1 and P_2 to form the next level candidates. Let us denote the last states of P_1 and P_2 as a and b, respectively. The candidates generated from P_1 and P_2 are completely specified by the relation between a and b. Since we can define 4 possible relations between the two states (*a before b*, *a overlaps b*, *b before a* and *b overlaps a*), joining P_1 and P_2 can potentially generate 4 different candidates.

However, we do not have to generate all four candidates in every join. Furthermore, we can speed up the algorithm by pruning the candidates that contain infrequent subpatterns. The following example illustrates candidate generation and pruning.

Example (Figure 3): Let *I*, *D*, and *S* denote the trend abstractions: *increasing*, *decreasing* and *steady*, respectively. Let *b* and *o* denote the *before* and *overlaps* relations.

Assume we have the following frequent *2-patterns*: P_1=*I*[*X*] *b D*[*Y*], P_2=*I*[*X*] *o I*[*Z*] and P_3=*I*[*Z*] *o D*[*Y*]. Since only P_1 and P_2 share a common prefix (*I*[*X*]), they can be joined to generate candidate *3-patterns*. However, we know that *D*[*Y*] should appear after *I*[*Z*] in the generated patterns (since *I*[Z] overlaps *I*[*X*], while *D*[*y*] is after *I*[*X*]). Thus, we only consider the two candidates C_1=*I*[*X*] *o I*[*Z*] *o D*[*Y*] and C_2=*I*[*X*] *o I*[*Z*] *b D*[*Y*]. Because the subpattern *I*[*Z*] *b D*[*Y*] contained in C_2 is not frequent, C_2 cannot be a frequent *3-pattern* (the *Apriori property*). Hence, C_1 is the only candidate that survives the pruning.

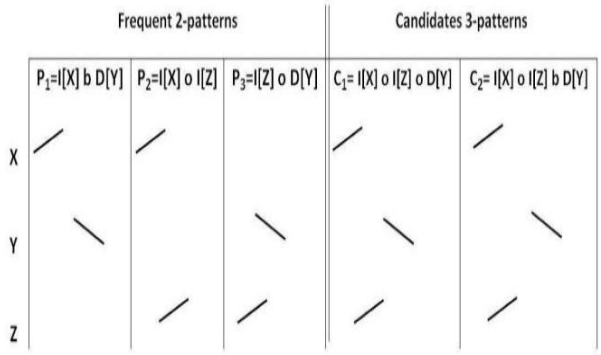

Figure 3: An example illustrating candidate generation and pruning.

Discriminative Pattern Selection: Our ultimate goal is to select temporal patterns that discriminate well among target classes to be the new features. So far our algorithm has extracted frequent patterns for every class of time-series examples. However, if we keep all these frequent patterns as features, the high dimensionality can easily cause the classifier to overfit the training data. To alleviate the problem, we want to select only a small number of temporal patterns which are good predictors of the class label. To do this, we use the chi-square (χ^2) statistics, which measures the correlation between the pattern and class variables. We define χ^2 for pattern P_k as:

$$\chi^2(P_k) = \sum_{c \in \{C_1..C_j\}, p \in \{P_k, \overline{P_k}\}} \frac{(Pr(p,c) - Ex(p,c))^2}{Ex(p,c)}$$

$$\Pr(p,c) = \frac{\#(p)}{N},\ \ Ex(p,c) = \frac{\#(p)}{N} * \frac{\#(c)}{N}$$

In the equation, N is the total number of instances in the dataset. The symbol # represents counts. For example, #(P_k, C_j) is the number of instances from

class C_j that contain pattern P_k and #($\overline{P_k}$, C_j) is the number of instances of C_j that do not contain P_k. Notice that the entries which contribute the most to the χ^2 value are those whose actual probabilities are very different from the expected probability under the independence assumption.

Finally, we rank all frequent patterns according to their χ^2 values and we select a small set of these patterns be the features of our classifier.

Building the classifier: The purpose of the previous steps is to convert the multivariate time series data into a feature-vector format that can be used with standard classification algorithms. Briefly, we map every MTS instance T_i into a Boolean vector V_i of size equal to the number of the temporal features extracted by the *STF-Mine* algorithm. . Each element $V_{i,j}$ in V_i corresponds to a specific pattern P_j and its value is set to 1 if P_j is present in T_i; otherwise, it is set to 0. These Boolean vectors are then fed to the classifier to build the model.

Experimental evaluation

In this section, we test and present results of our approach on clinical data by predicting the orders of the Heparin Platelet Factor 4 antibody (HPF4) test. This test is important for detecting and confirming Heparin-induced thrombocytopenia (HIT)[5]. **HIT** is a pro-thrombotic disorder induced by Heparin exposure with subsequent thrombocytopenia and associated thrombosis. HIT is a life-threatening condition if it is not detected and managed properly.

Our objective is to automatically learn from the data when an HPF4 test should be ordered for a patient on Heparin. In other words, given a specific point in time (which we call the *anchor point*) for a specific patient, we want to detect whether this patient starts to exhibit the HIT symptoms, which requires an order of HPF4.

Dataset: For our experiments, we use a database of 4460 records of post cardiac surgical patients treated at one of the University of Pittsburgh Medical Center (UPMC) hospitals. From this database, we choose 220 patients for which the HPF4 test was ordered and set the anchor point to be the time HPF4 was ordered. We then choose 220 other patients treated by Heparin, but who did not have an HPF4 test and set their anchor points randomly by the arrival of a new platelet result, a key feature used in HIT detection. We mix the cases (HPF4 orders) and controls, and test if *STF-Mine* can help us deciding whether HPF4 should be ordered or not for a specific patient at a specific point in time.

For every patient, we consider the platelet counts (PLT) and the Hemoglobin (Hgb) series and we use trend and value abstractions of each of the two series. This results in 4 state sequences that can be combined to form the temporal patterns. The normal range for PLT is [156-369]*10^9 and for Hgb is [12.9-16.9].

Since most recent values and patterns are likely the most important for classification, we consider the last 5 days of the patient's data prior to the anchor point. We set the window size for the *STF-Mine* algorithm to be 5 days. We set the *min_sup* parameter to 10% the number of instances in each class, and select the top 10 patterns to be used for classification.

We test the benefit of the *STF-Mine* features using two classifiers: linear kernel support vector machines (SVM)[6] and naïve Bayes (NB). We use the maximum likelihood approach for learning the parameters of the NB model[7]. All experiments follow a 20 fold cross validation scheme. To eliminate any classification bias, the temporal features are extracted only from the training sets, i.e., features used in different folds may be different. Because the vast majority of time series classification algorithms[8,9,10,11,12] can only handle regularly sampled time series, we compare our approach against baseline classifiers that use only the last two values of PLT and Hgb.

Results: Figure 4 compares the receiver operating characteristic (ROC) curves of SVM obtained using the *STF-Mine* features with the baseline SVM classifier. This ROC is the average of all ROCs obtained from the 20 folds. We can see that, using our features, SVM achieves 44% sensitivity (true positive rate) for a 95% specificity (true negative rate) level. For significance test on the differences in area under the ROC curve (AUC) values, we run 3 rounds of 20-fold cross validation. The p-value was p=0.0083 for the independent t-test and p=0.12 for the cross-validation corrected t-test[13].

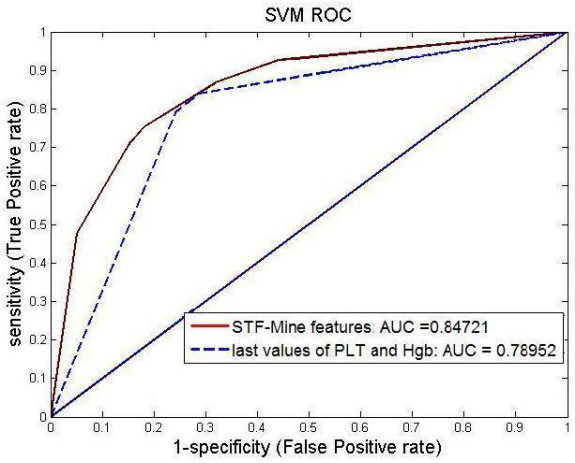

Figure 4: Comparing the ROC of SVM using the *STF-Mine* features (AUC=0.847) and using the last two values of PLT and Hgb (AUC=0.789).

We also tried the same task using the Naïve Bayes classifier. The AUC we get using the baseline features was 0.76, while using the *STF-Mine* features was 0.84. These experiments suggest that using the frequent and discriminative abstracted patterns as classification features is beneficial, regardless of the classification algorithm used.

Analysis of temporal patterns: Table 1 shows the top five discriminative temporal patterns indentified by our *STF-Mine* algorithm. The second and third columns show the number of times the pattern occurs in the cases (patients with HPF4 test) and the controls (patients without HPF4 test), respectively. The last column shows the pattern's χ^2 score.

Pattern	Occurrence in cases	Occurrence in controls	χ^2 Score
P1=low[PLT]	87%	32%	135.8
P2=normal[PLT]	9%	58%	116.9
P3=low[Hgb] overlaps low[PLT]	63%	15%	103.2
P4=decrease[PLT] overlaps low[PLT]	51%	10%	84.6
P5=increase[PLT]	7%	40%	63.9

Table 1: Top 5 discriminative temporal abstraction patterns according to the $\chi 2$ score.

The first two patterns P1 and P2 are simple platelet value abstractions. Low platelet values (P1) are more frequent in cases, while normal platelets (P2) are more frequent in controls. Patterns P3 and P4 are complex temporal patterns which combine two abstractions via the *overlaps* operator. Both P3 and P4 are more frequent in cases. P3 combines low hemoglobin and low platelet values. We believe this pattern covers patient cases that are on heparin for a longer period of time and experience some bleeding complications. It takes at least four to five days to develop HIT[12], so the incidence of HPF4 orders for this group is likely to go up. P4 combines low and decreasing platelets, which is the key pattern used for detecting the HIT risk. Finally, P5 is a simple trend pattern and indicates that HPF4 is less likely to be ordered if the platelet lab exhibits an increasing trend.

Pattern	P1	P2 *	P3	P4	P5 *
PPV	12.3%	10%	17.4%	20%	7.3%
NPV	99%	99.2%	97.8%	97.2%	99%

Table 2: Positive and negative predictive values of the temporal patterns in Table 1. The prior for cases and controls where estimated from all 4460 patients. The asterisk denotes that we use the negated pattern (absence of the pattern) to predict the case.

To better illustrate the benefit of these patterns for predicting HPF4 test orders, Table 2 shows the positive predictive value (PPV) and the negative predictive value (NPV) for each pattern. An asterisk next to the pattern indicates that we use the negated pattern to predict the case. Notice that refining pattern P1 in both P3 and P4 helps increasing the PPV value of the pattern.

Conclusion

The methodology developed in this work integrates two powerful data mining paradigms: frequent patterns mining and inductive classification to solve the complex multivariate time series classification problem. We believe the proposed method has wide applicability in the medical field and can be used to better predict clinically important outcomes and events from data. We currently investigate applying our approach in the design of clinical monitoring and alerting systems, which traditionally rely on features and rules defined a priori by an expert.

References

1. Shahar Y. A framework for knowledge-based temporal abstraction. Artificial Intelligence. 1997; 90:79-133.
2. Agrawal R, Srikant R. Fast algorithms for mining association rules in large databases. VLDB; 1994.
3. Keogh E, Chu S, Hart D, Pazzani M. Segmenting time series: a survey and novel approach. Data Mining in Time Series Databases, World Scientific Pub.; 2004.
4. Allen F. Towards a general theory of action and time. Artificial Intelligence; 1984; 23:123-154.
5. Warkentin T. Heparin-induced thrombocytopenia: pathogenesis and management. British Journal of Haematology; 2003; 121:535-555.
6. Vapnik V. The nature of statistical learning theory. Springer-Verlag New York; 1995.
7. Pedro D, Pazzani M.. On the Optimality of the simple Bayesian Classifier under zero-one Loss. Machine Learning; 1997.
8. Rabiner LR. A tutorial on hidden markov models and selected applications in speech recognition. IEEE; 1989.
9. Mandic D, Chambers J. Recurrent neural networks for prediction: learning algorithms, architectures and stability. John Wiley & Sons; 2001.
10. Xi X , Keogh E, Shelton C , Wei L, Ratanamahatana C. Fast time series classification using numerosity reduction. Proc. of ICML; 2006.
11. Chaovalitwongse W, Pardalos P, Prokoyev O. Electroencephalogram (EEG) time series classification: Applications in epilepsy. Annals of Operations Research. 2006: 148:227–250.
12. Weng X, Shen J. Classification of multivariate time series using two-dimensional singular value decomposition. Knowledge-Based System. 2008; 21: 535-539.
13. Nadeau C, Bengio Y. Inference for the generalization error. Machine Learning. 2003.