Minitutorial on
User Modeling and Temporal Pattern Discovery via Symbolic Learning

Ryszard S. Michalski

Machine Learning and Inference Laboratory
George Mason University
Fairfax, VA, USA

michalski @ gmu.edu

www.mli.gmu.edu/michalski.html

PLAN

- Introduction
- Fundamental concepts
- Sequential patterns in transactional databases
- User modeling for intrusion detection
- Conclusion

INTRODUCTION
World is Dynamic

- Most research on pattern discovery in databases has dealt with static data—data about the world at one point in time.
- Typical research topics include:
  - determination of concept dependencies
  - discovery of strong patterns
  - splitting data into clusters
  - selecting representative data points
  - selecting most relevant attributes
  - coping with data deficiencies (inconsistency, missing values, and noise).
- The world changes in time and the dimension is important for many applications
- Problems similar to those above can be posed also for the dynamic world, but their complexity is significantly higher.

Application Domains for Temporal Pattern Mining

- Customer transactions (e.g., finding common patterns in sets of transactions; Agrawal and Srikant, 1995))
- User modeling (e.g., finding user interaction patterns for computer intrusion detection; Bace, 2000)
- Stock market (e.g., modeling market activities; Kirkland and Senator, 1999)
- Business decision making (e.g., predicting changes in product demand)
- Event prediction (e.g., predicting next event/object in a sequence; Dietterich and Michalski, 1985)
- Bioinformatics (e.g., finding similarities among genetic sequences; Waterman, 1998)
- Finding text patterns (e.g., similar text sequences; Wu and Manber, Comm. of the ACM, 35, Oct., 1992.)
- Medicine (e.g., disease development, therapy effectiveness prediction)
- Agricultural prediction (e.g., crop damage prediction due to insects)
- Global change (e.g., global warming, climate change)

Areas Concerned with Deriving Knowledge from Data

- *Exploratory Data Analysis*---the use of statistical and analytic methods to generate and test hypotheses about data.
- *Machine Learning*---a general field that studies and implements methods for all forms of knowledge creation from data and/or other knowledge.
- *Data Mining*---hypothesizing simple patterns in massive datasets (adapts many ideas from machine learning)
- *Knowledge Discovery*---the term is often used to characterize the entire process of deriving final, tested regularities from initial, raw data
- ***Emerging new direction:*** *Knowledge mining---synthesizing human-oriented knowledge from databases and prior knowledge.*

Fundamental Concepts

Symbol
Attribute
Data
Information
Description
Generalization
Abstraction
Knowledge

- *Symbol:* a physical object, its state, or its behavior that is used to convey a choice from a predefined set of choices; the choice is called the *meaning* or *interpretation* of the symbol. The choices may refer to any entities (physical or abstract objects), or to their properties.
- *Attribute:* a symbol that is used to characterize some aspect of an entity by specifying the *attribute value* for that entity. An attribute has a name, a domain (value set), and a type.
- *Data:* a recorded set of symbols characterizing a set of entities.
- *Information:* interpreted data; data whose symbols have been assigned meaning.
- *Description:* an expression in some language that characterizes a set of entities. The set being described is called the *reference set* of the description.
- *Abstraction:* a process of reducing information about a reference set, or its result (often by employing less informative terms or linguistic constructs)
- *Generalization:* a process of extending the reference set of a description; or its result.
- *Knowledge:* a system of abstracted and generalized descriptions of a set of entities.

Static vs. Dynamic Data Mining

Static: Database consists of records (object descriptions or ODs) characterizing objects in terms of attributes that do not change over time, or relate to an object at a given time instance:

$OD = (x_1, x_2, x_3, \ldots, x_n)$

Dynamic: Database is a collection of records characterizing objects at different time instances:

$OD_t = (x_{t1}, x_{t2}, x_{t3}, \ldots x_{tn})$

where $t = t_0, t_1, t_2, \ldots, t_L$

Problems and Approaches

- Determine relationships among attributes
  - Association rules
  - Decision trees
  - Probabilistic networks
  - Attributional dependencies
- Select representative datapoints
- Improve representation space (constructive induction)
  - Attribute discretization/abstraction
  - Attribute selection
  - Attribute generation
- Determine clusters in data
  - Similarity-based clusters
  - Conceptual clusters

Adding the time dimension requires an extension of knowledge representation so that is can characterize events as time sequences. Among well-known approaches:

- Time-series analysis
- N-grams

Statistical vs. Symbolic Methods

- Statistical methods employ concepts and tools of statistics to characterize patterns as numerical functions.They produce pattern descriptions in terms of correlations, probability distributions, regression equations, etc.
- Symbolic learning methods, developed in the field of machine learning, employ concepts of logic and inference, and produce logic-style descriptions.

Mining Temporal Patterns

As mentioned before, a temporal database contains information about objects (processes) changing in time. Records can be viewed as event descriptions (EDs) that characterize events (e.g., transactions, sales, activities, states of a system, instances of interaction of a user with a computer system, states of a patient, etc.) at different moments of time:

$$ED_t = (x_{1t}, x_{2t}, x_{3t}, \ldots, x_{nt})$$

where $t = t_0, t_1, t_2, \ldots, t_L$

The time moments $t_0, t_1, t_2, \ldots, t_L$ may occur at equal time intervals, or at random intervals.

An important problem of interest is to determine patterns in the given data that would help to predict likely values of selected attributes in the future.

Mining Sequential Patterns in Customer Transactions

Definitions

- Let D be a database of customer transactions, where a transaction, T_i, is defined by a three-tuple:
  T_i = *<customer-id, transaction-time, itemset>*
  where *itemset* is a set of items purchased during the given transaction
- An itemset is represented by a sequence of integers, p_i, corresponding to the items/products included in the set: IS = $\{p_1, p_2, p_3, \ldots, p_n\}$
- An *itemset-sequence* is denoted IS = < IS_1 , IS_2 , ..., IS_k >
- A *customer-transaction-sequence* is a sequence of transactions by a given customer ordered by time: CTS = < T_1, T_2, ..., T_k >, where T_i is a transaction
- A *customer-sequence* is a sequence of itemsets in the customer-transaction sequence: CS = < $IS(T_1)$, $IS(T_2)$,... , $IS(T_k)$ >, where $IS(T_i)$ is the itemset in transaction T_i

Problem Statement
Finding Inter-transaction Patterns

(Agrawal, R. and Srikant, R., 1995)

- An itemset sequence, $< A_1, A_2, \ldots, A_n >$ is *contained in* an itemset sequence $< B_1, B_2, \ldots, B_m >$, if there exist integers $i_1 > i_2 > \ldots > i_n$ such that $A_1 \subseteq B_{i1}, A_2 \subseteq B_{i2}, \ldots, A_n \subseteq B_{in}$
- For example, the sequence <{2,3}, {4,5}, {6,7,8}> is contained in the sequence <{1,3,6}, {2,3,6}, {3,5}, {4,5,9}, {1,4,7}, {2,6,7,8,11}
- A customer-sequence *supports* (*covers*) an itemset-sequence IS, if IS is contained in it.
- The support of an itemset sequence is the percentage of customers who support that item sequence.
- Given a database, D, of customer transactions, the problem is to determine maximal itemset sequences, called *sequential patterns*, among all sequences that have a user-specified minimum support.
- Unlike a static data mining problem of finding itemsets bought together (intra-transaction patterns), here the problem is to find inter-transaction patterns.

A Method for Finding Inter-transaction Patterns

Below is a description of a method developed by Agrawal and Srikant (1995):

1 Sort phase: Sort database, with customer-id as the major key, and transaction-time as the minor key

2 Large-itemset phase: Find all large itemsets, defined as itemsets that have a sufficient support

3 Transformation phase: Each transaction is replaced by the set of all large-itemsets in this transaction

4 Sequence phase: Use the set of large-itemsets to find sequences of large-itemsets (candidate sequences):

- 1 Count-all algorithm: counts all large sequences, including non-maximal ones (*AprioriAll*)
- 2 Count-some algorithm: Avoid counting sequences included in other sequences (*AprioriSome*)

5 Maximal sequence phase: Find maximal itemset sequences among the set of large sequences.

6 **Experimental results**: both AprioriAll and AprioriSome have comparable performance and scale-up linearly with the number of transactions. The second is a little better for the lower values of the minimum number of customers to support a pattern.

Example

Consider a database of transactions:

Customer ID	Transaction Date	Itemsets
1 1	Jan 10, 01 Feb 12, 01	{23, 45, 67} {25,69,95}
2 2 2	Jan 13, 01 Feb 18, 01 March 2, 01	{10, 20, 48} {23,80} {40,50,89}
3 3	Jan 1, 01 Feb 1, 01	{23, 45, 67} {50,94,95}
4 4 4	Jan 10, 01 Feb 12, 01 Feb 18, 01	{23, 33, 45, 67} {23, 80} {40, 50, 69, 95}

Sequential patterns with support 75% (3 customers)
<{23}, {50}>
<{45,67}, {95}>

Mining Sequential Patterns For User Modeling

TASK:

Given records measuring various characteristics of the process of interaction between users and computers create models of the users' behavior.

Such models can be used for:

- Computer intrusion detection
- Predicting needs for different resources of individual users
- Intelligent tutoring—adapting the tutoring to individual users
- Intelligent advertisement

Computer Intrusion Detection Based on User Models

- Problem specification
  The problem is how to detect a computer intruder who broke into a system by finding out a legitimate user's password.
- Approach
  A way to approach this problem is to build models of the user interactions with the computer. Such models are then matched against future user computer interactions. A high match between the behavior of a user and the corresponding model confirms the legitimate use of the computer system, and a low match may indicate an intrusion.

The LUS Method

- The LUS method (Learning User Style) applies symbolic machine learning (AQ) to induce general and consistent patterns in the interactions (episodes) between users and computers, and then uses these patterns (user signatures) to validate the legitimate use of the computer or detect an illegitimate use (Michalski and Kaufman, 2001).
- An important advantage of this method is that due to the application of *natural induction*, generated user models are easy to interpret, and can be potentially improved/edited by an expert.

Using Process Table and Mode Attribute For Creating User Models

Process table

Assume that records measuring various characteristics of the interaction between users and computers are based on the *process table*, which contains information about processes used by a given user at a given time, the parent and spawn processes, and other information. Based on this information, one can compute the *mode of user activity*, which specifies the type of activity the user is involved in, such as editing, word processing, writing an email, accessing a file system, reading website, programming, compiling a program, etc.

Mode sequences

A session of interaction between the user and the computer (from login to logout) can thus be represented as a sequence of modes.

User i: $<m_1, m_2, m_3, \ldots\ldots, m_n>$

N-gram Representation of Sequences

- One effective method to characterize a sequence for the purpose of building a model is to transform the sequence into a set of *n-grams.*
- An n-gram is a sequence of n consecutive elements extracted from a sequence. By extracting n-grams starting from each element of the sequence, the sequence can be represented as a set of n-tuples.
- For example, a sequence of modes:
  <M4,M5,M2,M80, M39, M3, M1, M40, M5>
  can be represented by a set of 3-grams:
  {(M4,M5,M2), (M5,M2,M80), (M80, M39, M3),.., (M1, M40, M5)}

Terminology

A *session* is a sequence of events characterizing a user's interaction with the computer from the login to the logoff.

An *episode* is a sequence of consecutive events extracted from a session; it may contain just a few events, or all of the events in a session.

In the *training* phase, it is generally desirable to use long episodes, or even whole sessions.

In the *testing* (or execution) phase, it is desirable to use short episodes, at it is desirable to identify a user from as little information as possible.

Training Phase

In training phase, sets of n-grams characterizing training episodes of each user are supplied to a learning program (AQ).

The program returns a set of attributional rules characterizing each user.

To make the rules easier to interpret, they are transformed to generalized n-tuples. Such n-tuples contain groups of values in the rule area connected by the internal disjunction, and an * if a variable is not present in a rule.

In the *training* phase, it is generally desirable to use long episodes, or even whole sessions, as this helps to generate better user models.

In the *testing* (or execution) phase, it is desirable to use short episodes, so that a user can be identified from as little information as possible.

Attributional Calculus

- Attributional calculus combines elements of propositional calculus, many-valued logic, and predicate calculus.
- The core form is a propositiaonal calculus in which *literals* (symbols representing propositions or their negations) are replaced by *attributional conditions*
- Attributional conditions represent relational statements binding one or more *typed* variables with their values or other variables
- Each attribute (or a variable) is assigned a *domain* and a *type*.
  - The domain is the set of legal values of the attribute
  - The type characterizes the structure of the domain
- An attribute is of type
  - *nominal*, if its domain is an unordered set (e. g., blood type)
  - *linear* , if its domain is a totally ordered se (e.g., length, height, temperature)
  - *structured*, if its domain is a hierarchically ordered set (e.g., plants, animals, geometrical figures).

  Note: Linear attributes include attributes measured on the rank interval, ratio and absolute scales.
- Depending on the attribute type, different operators apply to the attribute
- Attributional calculus is based on the variable-valued logic one (VL1), a many-valued logic system introduced by Michalski (1972)

AQ LEARNING
Progressive Covering or "Separate and Conquer"

- The AQ learning stands for a class of methods that combine the star algorithm A^q with a symbolic object representation for the purpose of machine learning (e.g., Michalski, 1969, 1972; Michalski and Kaufman, 2000).
- The central concept of the algorithm is that of a *star* defined as a set of alternative generalizations of a selected event (called seed) that satisfy the assumed constraints (e.g., do not cover negative examples)
- AQ learning pioneered Progressive Covering or "Separate and Conquer" approach to rule learning
- AQ learners (e.g., AQ19 or INDUCE) employ richer than conventional rule representation languages in order to produce descriptions that are more compact and easy to understand (*attributional calculus* or *annotated predicate calculus).* For references see www.mli.gmu.edu/pubs.html.

Testing/Execution Phase

- Testing of descriptions (rulesets) created by an AQ learning program involves matching single events with the learned descriptions. This is done by ATEST -- a module integrated within the program.
- In the case of matching user models learned from episodes, ATEST is inadequate, because ones needs to score not just a single event, but an episode. To this end, a method for matching episodes with attributional rulesets generated by AQ learning was developed and implemented in EPIC (Episode Classifier).
- EPIC is a module within AQ20 that runs on top of the ATEST program.

Matching Episodes with User Models

Algorithm:
Calculate_predicted_classes(episode, classes)

- The algorithm returns the set of classes whose degree of match with the episode is above a certain threshold.
- Suppose the task is to classify an episode, which is a sequence of *z* events: e_1, e_2, …, e_z, into one of *m* decision classes (in this case, computer users): C_1, C_2, …, or C_m. Each decision class is described by a ruleset $R_i = \{R_{ij}\}$, where *j* can range from 1 to *s(i)*, the number of rules in the ruleset (signature) of class *i*.
- The EPIC module calls the ATEST module for each of the distinct events in the episode, and for each decision class determines a degree of match between the corresponding ruleset R_i and the episode. For each rule R_{ij} in ruleset *i*, it calculates a degree of match, c_{ijk}, between event *k* and rule R_{ij}, using the ATEST method.

The Degree of Match Between an Event and a Ruleset

The degree of match between an event *k* and a ruleset R_i, called an *event score for class i and event k*, and denoted $\boldsymbol{EV}_{ik}$, is defined:

$$\boldsymbol{EV}_{ik} = \text{Max}_{j=1...s(i)} (c_{ijk} \times t_{ij}) \quad (1)$$

where c_{ijk} is a degree of match between event *k* and rule R_{ij}, calculated by the ATEST method (Michalski and Kaufaman, 2000), and t_{ij} is the number of training examples satisfying R_{ij}.

The degree of match between an episode and the ruleset R_i, called the *episode score for class i* , and denoted EP_i, is defined:

$$\boldsymbol{EP}_i = \sum (k=1,...z)\ \boldsymbol{EV}_{ik} \quad (2)$$

EPIC classifies the episode to the class which receives the highest episode score. if the episode score is above the *score acceptance threshold* (SATH), and the difference between the highest and the next highest episode scores is greater than the *score acceptance tolerance (SATO)*. If the difference between the highest and the next highest episode scores are too small, the program classifies an episode as "unknown."

An Illustration of LUS Results

- Training examples in the form of collections of 4-grams were extracted from a training session involving four computer users. There were two sets of data, S1 and S2, each containing all 4-grams in the 20 training and 30 testing sessions for each of the four users. The files were relatively large, e.g., S1 occupied about 80 Mb, and S2 occupied about 60 Mb; the input file for S contained close to 2,000,000 events. Given these examples, AQ20 generated user models in the form of attributional rulesets.
- AQ20 parameters:
  - *ambiguity = ignore* *(treat ambiguous events as "don't care")*
  - *mode = Theory Formation* *(Generates complete and consistent rulesets)*
  - *maxstar = 2 & maxrule = 5* *(They control the beam search. The first parameter specifies the maximum number of intermediate hypotheses maintained at any one time, and the second specifies the maximum number of rules selected from each star (Cervone, Panait and Michalski, 2001).*
  - ***LEF =** (MinNumSelectors, 0.3) (MaxNewPositives, 0.1) (specifies the preference criterion for rule selection: minimize number of selectors with tolerance 0.3, and maximize new examples covered with tolerance 0.1)*

A Selection of Rules from User Models

[**user = A**]
- ← < debug, *, *, * > (p=76, u=13)
- ← < *, graphics, x, * > (p=31, u=16)
- ← < *, ghostview, *, * > (p=29, u=14)
- ← < x, calendar, filesystem, * > (p=11, u=7)

.....

[**user = B**]
- ← < timesheet, *, *, news > (p=15, u=2)
- ← < *, *, x, xrolo > (p=14, u=9)
- ← < *, x, timesheet, *reading* > (p=11, u=6)
- ← < sqt, news, *, * > (p=10, u=6)

.....

[**user = C**]
- ← < *, *, *, emacs > (p=376, u=126)
- ← < benchmark, *, *, * > (p=31, u=8)
- ← < userprog, *, *shells*, * > (p=30, u=12)
- ← < shell, edit, *, * > (p=21, u=13)

.....

[**user = D**]
- ← < xemacs, shell, *, * > (p=22, u=13)
- ← < xemacs, *, *reading*, * > (p=19, u=11)
- ← < *, mathematica, *, * > (p=19, u=11)
- ← < shell, news, *, * > (p=10, u=5)

.....

An Example of Testing Results

Table 1. Confusion matrix for classifying episodes in the dataset S1-test

Predicted class →	A	B	C	D
True Class = A	30	0	0	0
True Class = B	0	30	0	0
True Class = C	0	0	30	0
True Class = D	1	0	0	29

Table 2. Confusion matrix for classifying episodes in the dataset S2-test

Predicted class →	A	B	C	D
True Class = A	30	0	0	0
True Class = B	0	30	0	0
True Class = C	0	0	30	0
True Class = D	0	0	0	30

Summary of Current Methods

Advantages:
- Relatively simple to implement
- Easy to execute
- Tend to be general-purpose oriented
- Can scale-up

Limitations:
- Use very simple knowledge representation
- Utilize little background knowledge
- Typically work as an add-on to a DB---are not closely integrated with DBs
- Patterns may be difficult to interpret

For more information see:

www.mli.gmu.edu

or contact:

Ryszard Michalski
michalski@gmu.edu

or
Ken Kaufman
kaufman@gmu.edu