Temporal Rule Discovery using Genetic Programming and Specialized Hardware

Magnus Lie Hetland
IDI, NTNU
Sem Sælands vei 9, NO–7491 Trondheim, Norway
magnus@hetland.org

Pål Sætrom
Interagon AS
Medisinsk-teknisk senter, NO–7489 Trondheim, Norway
paalsat@interagon.com

Abstract: *Discovering association rules is a well-established problem in the field of data mining, with many existing solutions. In later years, several methods have been proposed for mining rules from sequential and temporal data. This paper presents a novel technique based on genetic programming and specialized pattern matching hardware. The advantages of this method are its flexibility and adaptability, and its ability to produce intelligible rules of considerable complexity.*

Keywords: Time series, sequence mining, rule discovery, genetic programming, pattern matching hardware

1 Introduction

Discovering association rules is a well-established problem in the field of data mining, with many existing solutions. In later years, several methods have been proposed for mining rules from sequential and temporal data (see, for example, [1, 5, 10]). Quite a few of these methods are based on the common premise of counting the occurrences of viable rules or patterns. While this approach has the advantage of finding all highly frequent patterns, it constrains the set of possible solutions.

One promising alternative is to use evolutionary algorithms, as described in [6]. While this approach places fewer restrictions on the form of patterns and rules that can be discovered, the performance of the method relies heavily on a speedy evaluation of each candidate rule, and such an evaluation typically involves examining the entire training data set. When mining rules in relational databases, existing indexing methods makes it possible to efficiently calculate the fitness of each rule. When mining sequences for complex patterns, this evaluation is not quite as straightforward. Efficient indexing methods for some forms of patterns exist, (for example, using Patricia trees, as in [2], or multigram indices, as in [4]), but in this paper we use a specialized co-processor designed to perform very advanced, high volume pattern matching.

The paper is structured as follows: Section 1.1 describes the problem we are trying to solve in more formal terms; Section 1.2 gives a brief overview of some related work; Section 2 describes the specifics of our method; Section 3 contains some empirical results; and, finally, Section 4 concludes the paper.

1.1 Problem Definition

Our problem is as follows: Given a sequence s, a predicate p over all the indices of s, and a delay δ, find a rule that estimates the value of $p(i+\delta)$ from the prefix $s_1, \ldots, s_i$ of s. The estimated predicate is written $\hat{p}$. In the terminology of [11] this is a problem of sequence recognition, although by letting the predicate p represent a type of event that may or may not occur at time

$i + \delta$, the rules can also be used to make predictions. Note that we take all available history into consideration by using a full prefix of s, rather than a fixed-width sliding window.

1.2 Related Work

A survey of association rule mining algorithms can be found in [8]. The underlying principle in these algorithms is exploiting the lattice structure of the pattern space when searching for frequent patterns and rules. This principle has also been applied to sequence mining [1, 5, 10]. The assumptions about the pattern space makes this general approach unsuitable for mining more complex patterns and rules.

Even though the problem tackled in this paper is closer to that of sequence prediction than that of sequence mining (see [11] for several sequence prediction algorithms), the goal of our method is to find rules that are readable and understandable by a human expert. Since this is one of the fundamental goals of data mining and knowledge discovery, we have chosen to classify our method as a rule discovery method.

A problem similar to ours is tackled in [7], where Giles et al. use recurrent neural networks to predict fluctuations in foreign exchange rates. In addition to the prediction task, their method encompasses the extraction of deterministic finite state automata, which are equivalent to regular expressions. Like most current sequence learning methods, the algorithms works with a fixed-width sliding window. We have tested our method on the same data sets as [7] in Section 3.3.

2 Method

To evolve our predictor rules we use genetic programming with the rule encoding scheme referred to in [6] as the Michigan approach, that is, each individual in the population represents a single rule. Since the consequent is a part of the problem definition, each rule is represented by its antecedent, an expression in a general query language that can be interpreted by our pattern matching hardware (see Section 2.1).

The training data consist of a discrete sequence s with elements from some finite alphabet, and predicate p, represented by the set of indices for which p is true. The rules are evaluated by having the hardware interpret their antecedents as queries, and comparing the returned hit positions, that is $\hat{p}$, with the correct positions given by p.

Compared to existing methods for discovering sequential rules, our method has two main advantages: (1) It can produce rules in a very rich rule language, which can be customized to each application as needed, and (2) each rule has the entire history of its sequence available when making a prediction. This means that the method is quite flexible, and that restrictions on the rule format are mainly dictated by the problem, rather than by the method itself.

A similar approach was originally used to evolve patterns for classifying human genomes. This is the subject of an upcoming paper.[1] The hardware and the process of fitness evaluation are discussed in the following subsections.

2.1 The Pattern Matching Chip

To make it possible to perform a full search in the training data for each fitness calculation, we used a specialized pattern matching chip (PMC). The PMC is a special purpose co-processor designed for locating complex patterns in unstructured data [9]. A detailed description is the subject of an upcoming paper;[2] a brief description will be given here.

[1] Svingen, Sætrom, Hetland: "Pattern Evolution" (manuscript in preparation)

[2] Svingen, Halaas, Birkeland, Nedland: "The Pattern Matching Chip"(manuscript in preparation)

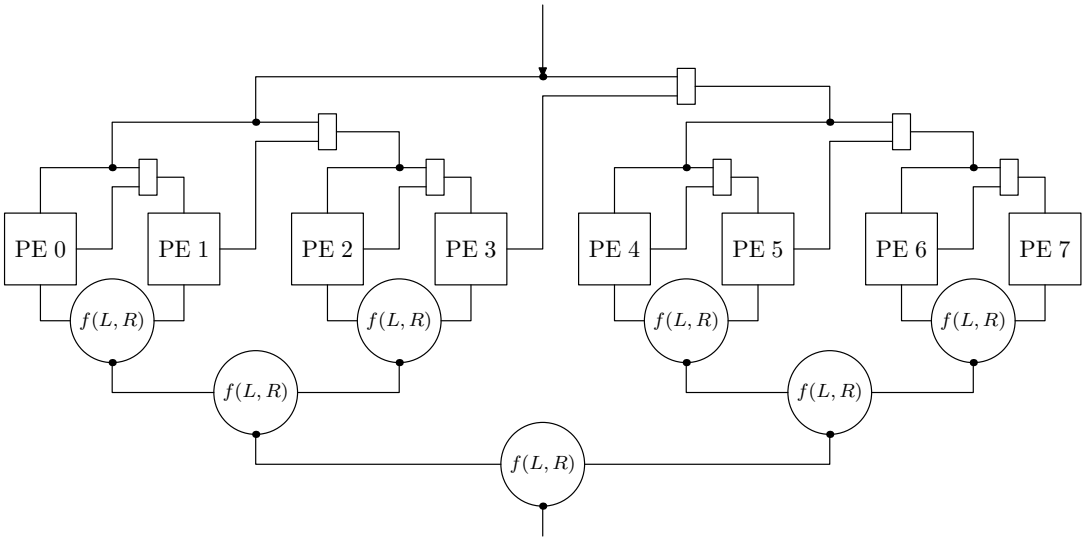

Figure 1: A data distribution tree with eight leaf nodes (PEs), and the corresponding result gathering tree with $f(L, R)$ calculated from the above left (L) and right (R) results.

The PMC consists of three functional units, as illustrated in Figure 1: A data distribution tree (top), a set of processing elements, or PEs (middle), and a result gathering tree (bottom). The PEs monitor the data flowing through the chip. They receive the data from the data distribution tree, which can be configured so that single PEs and groups of PEs receive data either sequentially or in parallel. Each PE is configured with one byte (character) and a comparison operator, which it uses to look for bytes in the data stream. Matches are reported to the result gathering tree, which combines them and produces the final result, a Boolean value representing a hit or miss for the entire pattern. If the result is a hit, the hit location is reported back to the host computer.

The PMC is capable of performing searches at the rate of 100 MB/s and can handle several complex queries in parallel (from 1 to 64 depending on query complexity.)[3] The interface to the PMC is the special purpose query language IQL. This language supports such functionality as regular expressions, latency (distance), alpha-numerical comparisons, and generalized Boolean operations. The regular expression syntax is similar to that of UNIX tools such as `grep`. A detailed description of the language is available online[4] and is the subject of an upcoming paper.[5]

2.2 Rule Evaluation

In evolutionary algorithms (such as genetic programming) each individual of the population must be assigned a fitness score, which describes how well the individual solves the problem at hand. In rule mining there are several possible quality measures, including various measures of interestingness. In this paper we focus on predictive power, because the events to be predicted (given by p) are part of the problem definition, which makes most interestingness measures unsuitable.

Some measures of predictive power (precision, true positive rate, true negative rate, and accuracy rate, as well as some combinations) are described in [6, pp. 129–134]. In this paper we have used another measure, the correlation coefficient of the set of data points given by $(p(i), \hat{p}(i))$, for $1 \leq i \leq n$ (where n is the size of the training sequence). This can be interpreted as the cosine

[3]The prototype used in our experiments runs at 33 MB/s and handles up to 4 parallel queries.

[4]`http://www.interagon.com/pub/whitepapers/IQL.reference-latest.pdf`

[5]Svingen, Halaas, Hetland, Fjelstad: "The Interagon Query Language" (manuscript in preparation)

of the angle between two n-dimensional vectors p and $\hat{p}$, which means that we get a fitness of $+1.0$ for perfect prediction and -1.0 for completely erroneous prediction.

By classifying the hits reported by a given rule as true or false positives (correct or incorrect hits), the correlation can be expressed as

$$r(p, \hat{p}) = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TN} + \text{FN})(\text{TN} + \text{FP})(\text{TP} + \text{FN})(\text{TP} + \text{FP})}}. \quad (1)$$

Here TP and FP are the number of true and false positives, respectively. The number of true and false negatives (TN and FN) can easily be calculated from TP, FP, and the total number of positives and negatives (directly available from the training data).

3 Experiments

The method was tested on three data sets: Synthetic data consisting of uniformly random symbols where certain positions were flagged according to predetermined rules; DNA sequence data, where the task was to predict the location of intron/exon splice sites; and foreign exchange rates, where the goal was to predict the trend for the next day (as in [7]). In the first two experiments we were mainly interested in the expressive power of our rule format, while in the last experiment we focused on predictive power.

For the first two data sets we used a "fuzzy" version of the problem definition from Section 1.1: For this fuzzy problem, the prediction predicate is the disjunction $p(i+\delta) \vee \cdots \vee p(i+\delta+\varepsilon)$ for some fixed window size ε. For these two data sets we used $\varepsilon = 10$. The original definition, which was used with the currency data, corresponds to $\varepsilon = 1$. For all the data sets we used $\delta = 1$.

For the last two data sets the technique of early stopping was used to prevent overfitting. This simply means that, in addition to a training set and a test set, we used a validation set, and the fitness calculated for this data set was used to select the best rules.

3.1 Synthetic Data

The synthetic data were constructed by repeatedly drawing symbols from the lowercase Latin alphabet (`a`–`z`) with a uniform probability distribution. The hit positions (representing the predicate p) were then found according to some predetermined antecedent patterns.

Two different antecedents types were used:

1. The regular expression `o[^n]*n`.
2. The letters `a`, `b`, `c`, `d` and `e` occurring in any order within a window of width 10.

The two sets consisted of 1 MB of sequence data with about 20000 and 1600 occurrences of antecedent type 1 and 2, respectively.

For the second antecedent type we wanted to simulate a search for fixed-width window rules, so we introduced a new operator into the language, composed from existing operators. The function of this operator was to match any number of arbitrary symbols (specified as parameters to the operator) that were all found within a window of width 10.

The system was able to generate expressions that recognized all occurences of both rules. Table 1 lists two of the expressions generated. As can be seen from this table, both problems were solved with a perfect correlation (100% prediction rate) for the test set. This rate is certainly a result of the data set being particularly well suited for our rule format.

Table 1: Results on synthetic data set.

Type	Antecedent Expression	Corr.
1	`([^n]+.+o[^n]+)n`	1.0
2	`{{a:d=9},{b:d=9},{c:d=9},{d:d=9},{e:d=9}:p=5}`	1.0

Table 2: Results on DNA data set.

Type	Antecedent Expression	Corr.
$5'$	`({t[ag]:d=52}{c:c=21})({(<=c):d=3}gg)t[ag]ag`	0.266
$3'$	`({[ct][ct][ct][cgt][ct][ct][ct][ct]:d=9}[ct]ag)&ag&(>=atctgt)`	0.177

3.2 DNA Sequence Data

The goal of this experiment was to find rules predicting intron/exon splice sites i DNA sequences. In addition to testing for predictive power, the rules were informally validated by comparing them to well-known splice site patterns.

The DNA sequences and exon locations were retrieved from NCBI.[6] The combined data set consisted of more than 6000 DNA sequences totalling 34 MB and about 20000 splice sites (of types $5'$ and $3'$, representing the beginning and end of an intron, respectively). The first 10 MB of this set were used in the training process (8 MB for training and 2 MB for early stopping.) The rest of the set was used for testing the generated rules.

Table 2 lists the rules produced for the $5'$ and $3'$ splice sites. The results are comparable to previously published splice site patterns (see, for example, [3]).

3.3 Foreign Exchange Rates

This experiment was performed on 5 sets of foreign exhange rates.[7] Because the number of data points was quite small (fewer than 4000 in total), tenfold cross-validation was used on each data set. The training data were discretized using the clustering method described in [5], and the resulting clusters were used to discretize the test and evaluation sets.

The average correlation and average prediction rate for each of the five currencies is listed in table 3. Giles et al. [7] report a prediction rate of 52.9% on the same data set.

Due to the sequental nature of the data, the use of cross-validation is problematic. Therefore, we performed the same experiment on a larger data set,[8] with about 8000 data points, using the

[6] `http://www.ncbi.nlm.nih.gov`

[7] Available from `http://www-psych.stanford.edu/~andreas/Time-Series/Data/Exchange.Rates.Daily`

[8] The exchange rate of BP to USD from 1971 to 2002, available from `http://www.federalreserve.gov/Releases/H10/Hist`

Table 3: Results on currency data sets.

Set	Avg. corr.	Avg. pred.	Max. pred.	Min. pred.
1	0.0465	54.5%	58.7%	50.5%
2	0.0446	54.3%	60.9%	45.1%
3	0.0845	54.8%	61.4%	48.4%
4	0.0527	56.1%	62.0%	52.2%
5	−0.0164	50.6%	57.6%	43.5%

first half as the training set. The experiment was repeated five times, giving prediction rates of 52.4%, 64.8%, 59.4%, 52.9%, and 65.1%.

4 Summary and Conclusions

The experiments performed so far seem promising: The method produces rules with good predictive power that are also readable by humans. For the synthetic data, we were able to reproduce the original rules, which contained regular expressions of varying complexity, without restriction on sequence history. For the DNA sequence data, our method produced rules with relatively good predictive power, which resemble well-known intron/exon splice site motifs. For the foreign exchange rates we were able to achieve a prediction rate comparable to that of Giles et al. in [7]. In each case, the rules were fairly easy to interpret, which means that it should be possible for a domain expert to find some structure in them, and perhaps modify them to a simpler or more general form, and to test them in real time, using the pattern matching hardware.

References

[1] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In Philip S. Yu and Arbee S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

[2] Richardo A. Baeza-Yates and Gaston H. Gonnet. Fast text searching for regular expressions or automaton searching on tries. *Journal of the ACM*, 43(6):915–936, 1996.

[3] Christopher B. Burge, Thomas Tuschl, and Phillip A. Sharp. Splicing of precursors to mRNAs by the spliceosomes. In *The RNA World (Second edition)*, pages 525–560. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York, 1999.

[4] Junghoo Cho and Sridhar Rajagopalan. A fast regular expression indexing engine. In *ICDE*, 2002.

[5] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule discovery from time series. In *Knowledge Discovery and Data Mining*, pages 16–22, 1998.

[6] Alex A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms.* Springer-Verlag, 2002.

[7] C. Lee Giles, Steve Lawrence, and Ah Chung Tsoi. Noisy time series prediction using a recurrent neural network and grammatical inference. *Machine Learning*, 44(1/2):161–183, July/August 2001.

[8] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining – a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, July 2000.

[9] Interagon AS. European patent specification EP1125216B1 titled "Digital processing device," deriving from international published patent application WO 00/22545.

[10] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[11] Ron Sun and C. Lee Giles, editors. *Sequence Learning : Paradigms, Algorithms, and Applications.* Number 1828 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 2000.