Optimizing Probabilistic Models for Relational Sequence Learning

Nicola Di Mauro, Teresa M.A. Basile, Stefano Ferilli, and Floriana Esposito

Department of Computer Science, LACAM laboratory
University of Bari "Aldo Moro", Via Orabona,4, 70125 Bari, Italy
{ndm,basile,ferilli,esposito}@di.uniba.it

Abstract. This paper tackles the problem of relational sequence learning selecting relevant features elicited from a set of labelled sequences. Each relational sequence is firstly mapped into a feature vector using the result of a feature construction method. The second step finds an optimal subset of the constructed features that leads to high classification accuracy, by adopting a wrapper approach that uses a stochastic local search algorithm embedding a Bayes classifier. The performance of the proposed method on a real-world dataset shows an improvement compared to other sequential statistical relational methods, such as Logical Hidden Markov Models and relational Conditional Random Fields.

1 Introduction

Sequential data may be found in many contexts of everyday life, and in many computer science applications such as video understanding, planning, computational biology, user modelling and speech recognition. Different methodologies have been proposed to face the problem of sequential learning. Some environments involve very complex components and features, and hence classical existing approaches have been extended to the case of relational sequences [1] to exploit a more powerful representation formalism. Sequential learning techniques may be classified according to the language they adopt to describe sequences. On the one hand there are methods adopting a propositional language, such as Hidden Markov Models (HMMs), allowing both a simple model representation and an efficient algorithm; on the other hand (Sequential) Statistical Relational Learning (SRL) [2] techniques, such as Logical Hidden Markov Models (LoHMMs) [3] and relational Conditional Random Fields [4, 5] are able to elegantly handle complex and structured descriptions for which a flat representation could make the problem intractable to propositional techniques. The goal of this paper is to propose a new probabilistic method for relational sequence learning [1].

A way to tackle the task of inferring discriminant functions in relational learning is to reformulate the problem into an attribute-value form and then apply a propositional learner [6]. The reformulation process may be obtained adopting a *feature construction* method, such as mining frequent patterns that can then be successfully used as new Boolean features [7–9]. Since, the effectiveness of learning algorithms strongly depends on the used features, a *feature*

selection task is very desirable. The aim of feature selection is to find an optimal subset of the input features leading to high classification performance, or, more generally, to carry out the classification task optimally. However, the search for a variable subset is a NP-hard problem. Therefore, the optimal solution cannot be guaranteed to be reached except when performing an exhaustive search in the solution space. Using *stochastic local search* procedures [10] allows one to obtain good solutions without having to explore the whole solution space.

In this paper we propose an algorithm for relational sequence learning, named `Lynx`[1], that works in two phases. In the first phase it adopts a feature construction approach that provides a set of probabilistic features. In the second step, `Lynx` adopts a wrapper feature selection approach, that uses a stochastic local search procedure, embedding a naïve Bayes classifier to select an optimal subset of the features constructed in the previous phase. In particular, the optimal subset is searched using a Greedy Randomised Search Procedure (GRASP) [11] and the search is guided by the predictive power of the selected subset computed using a naïve Bayes approach. The focus of this paper is on combining probabilistic feature construction and feature selection for relational sequence learning.

Related works may be divided into two categories. The former includes works belonging to the Inductive Logic Programming (ILP) [12] area, that reformulate the initial relational problem into an attribute-value form, by using frequent patterns as new Boolean features, and then applying propositional learners. The latter category includes all the systems purposely designed to tackle the problem of relational sequence analysis falling into the more specific SRL area where probabilistic models are combined with relational learning.

This work may be related to that in [9], where the authors presented one of the first ILP feature construction methods. They firstly build a set of features adopting a declarative language to constrain the search space and find discriminant features. Then, these features are used to learn a classification model with a propositional learner. In [13] are presented a logic language for mining sequences of logical atoms and an inductive algorithm, that combines principles of the level-wise search algorithm with the version space in order to find all patterns that satisfy a given constraint. These ILP works, however, take into account the feature construction problem only. In this paper, on the other hand, we want to optimise the predictive accuracy of a probabilistic model built on an optimal set of the constructed features.

More similar to our approach are sequential statistical relational techniques that combine a probabilistic model with a relational description belonging to the SRL area, such as Logical Hidden Markov Models (LoHMMs) [3] and relational Conditional Random Fields [4] that are purposely designed for relational sequence learning. In [3] the authors proposed an algorithm for selecting LoHMMs from logical sequences. The proposed logical extension of HMMs overcomes their weakness on flat symbols by handling sequences of structured symbols by means of a probabilistic ILP framework. In [14] the authors presented a method to compute the gradient of the likelihood with respect to the parameters of a LoHMM.

[1] `Lynx` is public available at `http://www.di.uniba.it/~ndm/lynx/`.

They overcome the predictive accuracy of the generative model of LoHMMs using a Fisher Kernel. Finally, in [4] an extension of Conditional Random Fields (CRFs) to logical sequences has been proposed. CRFs are undirected graphical models that, instead of learning a generative model as in HMMs, learn a discriminative model designed to handle non-independent input features. In [4], the authors lifted CRFs to the relational case representing the potential functions as a sum of relational regression trees.

2 Lynx: a Relational Pattern-based Classifier

This section firstly briefly reports the framework for mining relational sequences introduced in [15] and used in Lynx due to its general logic formalism. Over that framework Lynx implements a probabilistic pattern-based classifier. After introducing the representation language, the Lynx system will be presented, along with its feature construction capability, the adopted pattern-based classification model, and the feature selection approach.

As a representation language we used first-order logic.A first-order *alphabet* consists of a set of *constants*, a set of *variables*, a set of *function symbols*, and a non-empty set of *predicate symbols*. Both function symbols and predicate symbols have a natural number (its *arity*) assigned to it. A *term* is a constant symbol, a variable symbol, or an n-ary function symbol f applied to n terms $t_1, t_2, \ldots, t_n$. An atom $p(t_1, \ldots, t_n)$ is a predicate symbol p of arity n applied to n terms t_i. Both l and its negation $\bar{l}$ are said to be (resp., positive and negative) *literals* whenever l is an atom. Literals and terms are said to be *ground* whenever they do not contain variables. A *substitution* θ is defined as a set of bindings $\{X_1 \leftarrow a_1, \ldots, X_n \leftarrow a_n\}$ where $X_i, 1 \leq i \leq n$ are variables and $a_i, 1 \leq i \leq n$ are terms. A substitution θ is applied to an expression e, obtaining the expression $(e\theta)$, by replacing all variables X_i with their corresponding term a_i.

Lynx adopts the relational framework, and the corresponding pattern mining algorithm, reported in [15], that here we briefly recall. Considering a sequence as an ordered succession of events, fluents have been used to indicate that an atom is true for a given event. A *multi-dimensional relational sequence* may be defined as a set of atoms, concerning n dimensions, where each event may be related to another event by means of the $<_i$ operators, $1 \leq i \leq n$. In order to represent multi-dimensional relational patterns, the following dimensional operators have been introduced. Given a set $\mathcal{D}$ of dimensions, $\forall i \in \mathcal{D}$: $<_i$ indicates the direct successor on the dimension i; $\lhd_i$ encodes the transitive closure of $<_i$; and $\bigcirc_i^n$ calculates the n-th direct successor. Hence, a *multi-dimensional relational pattern* may be defined as a set of atoms, regarding n dimensions, in which there are non-dimensional atoms and each event may be related to another event by means of the operators $<_i$, $\lhd_i$ and $\bigcirc_i^n$, $1 \leq i \leq n$. In order to compute the frequency of a pattern over a sequence it is important to define the concept of sequence subsumption. Given $\Sigma = \mathcal{B} \cup U$, where U is the set of atoms in a sequence S, and $\mathcal{B}$ is a background knowledge. A pattern P *subsumes* a sequence S ($P \subseteq S$),

iff there exists an SLD_{OI}-deduction of P from Σ. An SLD_{OI}-deduction is an SLD-deduction under Object Identity [16].

2.1 Feature Construction via Pattern Mining

The first step of `Lynx` carries out a feature construction process by mining frequent patterns from sequences with an approach similar to that reported in [9]. The algorithm for frequent multi-dimensional relational pattern mining is based on the same idea as the generic level-wise search method, known in data mining from the Apriori algorithm [17]. The level-wise algorithm performs a breadth-first search in the lattice of patterns ordered by a specialization relation $\preceq$. Generation of the frequent patterns is based on a top-down approach. The algorithm starts with the most general patterns. Then, at each step it tries to specialise all the candidate frequent patterns, discarding the non-frequent patterns and storing those whose length is equal to the user specified input parameter `maxsize`. For each new refined pattern, semantically equivalent patterns are detected, by using the θ_{OI}-subsumption relation [16], and discarded. In the specialization phase the specialization operator, basically, adds atoms to the pattern.

The algorithm uses a background knowledge $\mathcal{B}$ containing the sequences and a set of constraints, similar to that defined in SeqLog [13], that must be satisfied by the generated patterns. In particular, some of the constraints in $\mathcal{B}$ are (see [15] for more details): `maxsize(M)`, maximal pattern length; `minfreq(m)`, the frequency of the patterns must be greater than m; `type(p) and mode(p)`, denote, respectively, the type and the input/output mode of the predicate's arguments `p`, used to specify a language bias; `negconstraint(`$[p_1, p_2, \ldots, p_n]$`)` specifies a constraint that the patterns must not fulfil; `posconstraint(`$[p_1, p_2, \ldots, p_n]$`)` specifies a constraint that the patterns must fulfil; `atmostone(`$[p_1, p_2, \ldots, p_n]$`)` discards all the patterns that make true more than one predicate among p_1, p_2,..., p_n; `key(`$[p_1, p_2, \ldots, p_n]$`)` specifies that each pattern must have one of the predicates $p_1, p_2, \ldots p_n$ as a starting literal.

Given a set of relational sequences D defined over a set of classes C, the *frequency* of a pattern p, $\text{freq}(p, D)$, corresponds to the number of sequences $s \in D$ such that p subsumes s. The *support* of a pattern p with respect to a class $c \in C$, $\text{supp}_c(p, D)$ corresponds to the number of sequences $s \in D$ whose class label is c. Finally, the *confidence* of a pattern p with respect to a class $c \in C$ is defined as $\text{conf}_c(p, D) = \text{supp}_c(p, D)/\text{freq}(p, D)$.

The refinement of patterns is obtained by using a refinement operator ρ that maps each pattern to a set of its specializations, i.e. $\rho(p) \subset \{p' | p \preceq p'\}$ where $p \preceq p'$ means that p is more general than p' or that p subsumes p'. For each specialization level, before starting the next refinement step, `Lynx` records all the obtained patterns. Hence, it might happen that the final set includes a pattern p that subsumes many other patterns in the same set. However, the subsumed patterns may have a different support, contributing in different way to the classification model.

2.2 Pattern-based Classification

After identifying the set of frequent patterns, the next question is how to use them as features in order to correctly classify unseen sequences. Let $\mathcal{X}$ be the input space of relational sequences, and $\mathcal{Y} = \{1, 2, \ldots, Q\}$ denote the finite set of possible class labels. Given a training set $D = \{(X_i, Y_i) | 1 \leq i \leq m\}$, where $X_i \in \mathcal{X}$ is a single relational sequence and $Y_i \in \mathcal{Y}$ is the label associated to X_i, the goal is to learn a function $h : \mathcal{X} \to \mathcal{Y}$ from D that predicts the label for each unseen instance. Let $\mathcal{P}$, with $|\mathcal{P}| = d$, be the set of constructed features obtained in the first step of the `Lynx` system (the patterns mined from D), as reported in Section 2.1. For each sequence $X_k \in \mathcal{X}$ we can build a d-component vector-valued $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ random variable where each $x_i \in \mathbf{x}$ is 1 if the pattern $p_i \in \mathcal{P}$ subsumes sequence X_k, and 0 otherwise, for each $1 \leq i \leq d$.

Using the Bayes' theorem, if $p(Y_j)$ describes the prior probability of class Y_j, then the posterior probability $p(Y_j|\mathbf{x})$ can be computed from $p(\mathbf{x}|Y_j)$ as

$$p(Y_j|\mathbf{x}) = \frac{p(\mathbf{x}|Y_j)p(Y_j)}{\sum_{i=1}^{Q} p(\mathbf{x}|Y_i)p(Y_i)}.$$

Given a set of discriminant functions $g_i(\mathbf{x})$, $i = 1, \ldots, Q$, a classifier is said to assign the vector $\mathbf{x}$ to class Y_j if $g_j(\mathbf{x}) > g_i(\mathbf{x})$ for all $j \neq i$. Taking $g_i(\mathbf{x}) = P(Y_i|\mathbf{x})$, the maximum discriminant function corresponds to the *maximum a posteriori* (MAP) probability. For minimum error rate classification, the following discriminant function will be used

$$g_i(\mathbf{x}) = \ln p(\mathbf{x}|Y_i) + \ln P(Y_i). \tag{1}$$

We are considering a multi-class classification problem involving discrete features. In this problem the components of vector $\mathbf{x}$ are binary-valued and conditionally independent. In particular, let the component of vector $\mathbf{x} = (x_1, \ldots, x_d)$ be binary valued (0 or 1). We define

$$p_{ij} = \text{Prob}(x_i = 1|Y_j)_{\substack{i=1,\ldots,d \\ j=1,\ldots,Q}}$$

with the components of $\mathbf{x}$ being statistically independent for all $x_i \in \mathbf{x}$. In this model each feature x_i gives a yes/no answer about pattern p_i. However, if $p_{ik} > p_{it}$ we expect the i-th pattern to subsume a sequence more frequently when its class is Y_k than when it is Y_t. The factors p_{ij} can be estimated by frequency counts on the training examples, as $p_{ij} = \text{support}_{Y_j}(p_i)$. In this way, the constructed features p_i may be viewed as *probabilistic features* expressing the relevance for pattern p_i in determining classification Y_j.

By assuming conditional independence we can write $P(\mathbf{x}|Y_i)$ as a product of the probabilities of the components of $\mathbf{x}$. Given this assumption, a particularly convenient way of writing the class-conditional probabilities is as follows: $P(\mathbf{x}|Y_j) = \prod_{i=1}^{d} (p_{ij})^{x_i} (1-p_{ij})^{1-x_i}$. Hence, Eq. 1 yields the discriminant function

$$g_j(\mathbf{x}) = \ln p(\mathbf{x}|Y_j) + \ln p(Y_j) = \sum_{i=1}^{d} x_i \ln \frac{p_{ij}}{1-p_{ij}} + \sum_{i=1}^{d} \ln(1-p_{ij}) + \ln p(Y_j). \tag{2}$$

The factor corresponding to the prior probability for class Y_j can be estimated from the training set as $p(Y_i) = \frac{|\{(X,Y)\in D \text{ s.t. } Y=Y_i\}|}{|D|}, 1 \leq i \leq Q$. The minimum probability of error is achieved by the following decision rule: decide Y_k, $1 \leq k \leq Q$, if $\forall j, 1 \leq j \leq Q \wedge j \neq k : g_k(\mathbf{x}) \geq g_j(\mathbf{x})$, where $g_i(\cdot)$ is defined as in Eq. 2. Note that this discriminant function is linear in x_i, and thus we can write $g_j(\mathbf{x}) = \sum_{i=1}^{d} \alpha_i x_i + \beta_0$, where $\alpha_i = \ln(p_{ij}/(1-p_{ij}))$, and $\beta_0 = \sum_{i=1}^{d} \ln(1-p_{ij}) + \ln p(Y_j)$. The magnitude of the weight α_i in $g_j(\mathbf{x})$ indicates the relevance of a subsumption for pattern p_i in determining classification Y_j. This is the probabilistic characteristic of the features obtained in the feature construction phase, as opposed to the classical Boolean feature approach.

2.3 Feature Selection with Stochastic Local Search

After having constructed a set of features, and presented a method to use those features to classify unseen sequences, now the problem is how to find a subset of these features that optimises prediction accuracy. The optimization problem of selecting a subset of features (patterns) with a superior classification performance may be formulated as follows. Let $\mathcal{P}$ be the constructed original set of patterns, and let $f : 2^{|\mathcal{P}|} \rightarrow \mathbb{R}$ be a function scoring a selected subset $X \subseteq \mathcal{P}$. The problem of feature selection is to find a subset $\widehat{X} \subseteq \mathcal{P}$ such that $f(\widehat{X}) = \max_{Z\subseteq\mathcal{P}} f(Z)$. An exhaustive approach to this problem would require examining all $2^{|\mathcal{P}|}$ possible subsets of the feature set $\mathcal{P}$, making it impractical for even small values of $|\mathcal{P}|$. The use of a stochastic local search procedure [10] allows to obtain *good* solutions without having to explore the whole solution space.

Given a subset $P \subseteq \mathcal{P}$, for each sequence $X_j \in \mathcal{X}$ we let the classifier find the MAP hypothesis $\widehat{h}_P(X_j) = \arg\max_i g_i(\mathbf{x}_j)$ by adopting the discriminant function reported in Eq. 1, where $\mathbf{x}_j$ is the feature based representation of sequence X_j obtained using patterns in P. Hence the initial optimization problem corresponds to minimise the expectation $\mathrm{E}[\mathbf{1}_{\widehat{h}_P(X_j)\neq Y_j}]$ where $\mathbf{1}_{\widehat{h}_P(X_j)\neq Y_j}$ is the characteristic function of training example X_j returning 1 if $\widehat{h}_P(X_j) \neq Y_j$, and 0 otherwise. Finally, given D the training set with $|D| = m$ and P a set of features (patterns), the number of classification errors made by the Bayesian model is

$$err_D(P) = m\mathrm{E}[\mathbf{1}_{\widehat{h}_P(X_j)\neq Y_j}]. \tag{3}$$

GRASP$^{\text{FS}}$ Consider a *combinatorial optimisation* problem, where one is given a discrete set X of solutions and an objective function $f : X \rightarrow \mathbb{R}$ to be minimised, and seek a solution $x^* \in X$ such that $\forall x \in X : f(x^*) \leq f(x)$. A method to find high-quality solutions for a combinatorial problem consists of a two-step approach made up of a greedy construction phase followed by a perturbative local search [10]. The greedy construction method starts the process from an empty candidate solution and at each construction step adds the best ranked component according to a heuristic selection function. Then, a perturbative local search algorithm, searching a local *neighbourhood*, is used to improve the candidate

solution thus obtained. Advantages of this search method are a much better solution quality and fewer perturbative improvement steps needed to reach the local optimum.

GRASP [11] solves the problem of the limited number of different candidate solutions generated by a greedy construction search method by randomising the construction method. GRASP is an iterative process combining at each iteration a construction and a local search phase. In the construction phase a feasible solution is built, and then its neighbourhood is explored by the local search. Algorithm 1 reports the $\texttt{GRASP}^{\texttt{FS}}$ procedure included in the Lynx system to perform the feature selection task. In each iteration, it computes a solution $S \in \mathcal{S}$ by using a randomised constructive search procedure and then applies a local search procedure to S yielding an improved solution. The main procedure is made up of two components: a constructive phase and a local search phase.

Algorithm 1 $\texttt{GRASP}^{\texttt{FS}}$

Input: D: the training set; $\mathcal{P}$: a set of patterns (features); *maxiter*: maximum number of iterations; $err_D(P)$: the evaluation function (see Eq. 3)
Output: solution $\widehat{S} \subseteq \mathcal{P}$

1: $\widehat{S} = \emptyset$, $err_D(\widehat{S}) = +\infty$
2: iter = 0
3: **while** iter < maxiter **do**
4: α = rand(0,1)
5: $S = \emptyset$; $i = 0$
6: **while** $i < n$ **do**
7: $\mathcal{S} = \{S' | S' = add(S, A)\}$ for each component $A \in \mathcal{P}$ s.t. $A \notin S$
8: $\overline{s} = \max\{err_D(T) | T \in \mathcal{S}\}$
9: $\underline{s} = \min\{err_D(T) | T \in \mathcal{S}\}$
10: RCL $= \{S' \in \mathcal{S} | err_D(S') \leq \underline{s} + \alpha(\overline{s} - \underline{s})\}$
11: select the new S, at random, from RCL
12: $i \leftarrow i + 1$
13: $\mathcal{N} = \{S' \in neigh(S) | err_D(S') < err_D(S)\}$
14: **while** $\mathcal{N} \neq \emptyset$ **do**
15: select $S \in \mathcal{N}$
16: $\mathcal{N} \leftarrow \{S' \in neigh(S) | err_D(S') < err_D(S)\}$
17: **if** $err_D(S) < err_D(\widehat{S})$ **then**
18: $\widehat{S} = S$
19: iter = iter + 1
20: **return** $\widehat{S}$

The constructive search algorithm (lines 4-12) used in $\texttt{GRASP}^{\texttt{FS}}$ iteratively adds a solution component by randomly selecting it, according to a uniform distribution, from a set, named *restricted candidate list* (RCL), of highly ranked solution components with respect to a greedy function $g : \mathcal{S} \rightarrow \mathbb{R}$. The probabilistic component of $\texttt{GRASP}^{\texttt{FS}}$ is characterised by a random choice of one of the best candidates in the RCL. In our case the greedy function g corresponds to the error

Conf.	Lynx	Folds										Mean
		1	2	3	4	5	6	7	8	9	10	
0.95	w/o GRASP^{FS}	0.84	0.88	0.83	0.83	0.85	0.76	0.85	0.81	0.82	0.80	0.826
	w GRASP^{FS}	0.88	0.92	0.88	0.88	0.89	0.84	0.93	0,87	0.90	0.93	**0.878**
1.0	w/o GRASP^{FS}	0.89	0.94	0.84	0.92	0.94	0.88	0.91	0.89	0.88	0.87	0.896
	w GRASP^{FS}	0.94	0.97	0.93	0.95	0.95	0.93	0.93	0.97	0.90	0.94	**0.942**

Table 1. Cross-validated accuracy on 10 folds of the data of Lynx with and without feature selection.

function $err_D(P)$ previously reported in Eq. 3. In particular, given $err_D(P)$, the heuristic function, and $\mathcal{S}$, the set of feasible solutions, $\underline{s} = \min\{err_D(S)|S \in \mathcal{S}\}$ and $\overline{s} = \max\{err_D(S)|S \in \mathcal{S}\}$ are computed. Then the RCL is defined by including in it all the components S such that $err_D(S) \geq \underline{s} + \alpha(\overline{s} - \underline{s})$.

To improve the solution generated by the construction phase, a local search is used (lines 13-16). It works by iteratively replacing the current solution with a better solution taken from the neighbourhood of the current solution while such a better solution exists. Given $\mathcal{P}$ the set of patterns, in order to build the neighbourhood *neigh(S)* of a solution $S = \{p_1, p_2, \ldots, p_t\} \subseteq \mathcal{P}$, the following operators are exploited:

add: $S \rightarrow S \cup \{p_i\}$ where $p_i \in \mathcal{P} \setminus S$;
replace: $S \rightarrow S \setminus \{p_i\} \cup \{p_k\}$ where $p_i \in S$ and $p_k \in \mathcal{P} \setminus S$.

In particular, given a solution $S \in \mathcal{S}$, the elements of the neighbourhood $neigh(S)$ of S are those solutions that can be obtained by applying an elementary modification (add or replace) to S. Local search starts from an initial solution $S^0 \in \mathcal{S}$ and iteratively generates a series of improving solutions $S^1, S^2, \ldots$. At the k-th iteration, $neigh(S^k)$ is searched for an improved solution S^{k+1} such that $err_D(S^{k+1}) < err_D(S^k)$. If such a solution is found, it becomes the current solution. Otherwise, the search ends with S^k as a local optimum.

3 Experiments

Experiments were conducted on protein fold classification, an important problem in biology. The dataset, already used in [14, 3, 4], is made up of logical sequences of the secondary structure of protein domains. The task is to predict one of the five most populated SCOP folds of alpha and beta proteins (a/b): TIM beta/alpha-barrel (c1), NAD(P)-binding Rossmann-fold domains (c2), Ribosomal protein L4 (c23), Cysteine hydrolase (c37), and Phosphotyrosine protein phosphatases I-like (c55). Overall, the class distribution is 721 sequences for class c1, 360 for c2, 274 for c23, 441 for c37 and 290 for c55. As in [4], we used a round robin approach, treating each pair of classes as a separate classification problem, and the overall classification of an example instance is the majority vote among all pairwise classification problems.

Table 1 reports the experimental results of a 10-fold cross-validated accuracy of Lynx. Two experiments have been run choosing confidence levels 0.95 and

System	Accuracy
`LoHMMs` [3]	75%
`Fisher kernels` [14]	84%
`TildeCRF` [4]	92.96%
`Lynx`	**94.15**%

Table 2. Cross-validated accuracy of LoHHMs, Fisher kernels, TildeCRF and `Lynx`

1.0. For each experiment, `Lynx` was applied on the same data with and without feature selection. In particular, we run classification on the test instances without applying $\texttt{GRASP}^{\texttt{FS}}$ in order to have a baseline accuracy value. Indeed, it turns out that accuracy grows when $\texttt{GRASP}^{\texttt{FS}}$ optimises the feature set, proving the validity of the method adopted for the feature selection task. Furthermore, the accuracy level grows up when we mine patterns with a confidence level equal to 1.0 which corresponds to saving *jumping emerging patterns*[2] only. This proves that jumping patterns have a discriminative power greater than *emerging patterns* (when the confidence level is equal to 0.95).

As a second experiment we compared `Lynx` on the same data to other SRL systems. Cross-validated accuracy is summarised in Table 2. `LoHHMs` [3] were able to achieve a predictive accuracy of 75%, `Fisher kernels` [14] achieved an accuracy of about 84%, `TildeCRF` [4] reached an accuracy value of 92.96%, while `Lynx` obtained an accuracy of 94.15%. We can conclude that `Lynx` performs better than established methods on this real-world dataset.

4 Conclusions

In this paper we considered the problem of relational sequence learning using relevant patterns discovered from a set of labelled sequences. We firstly apply a feature construction method in order to map each relational sequence into a feature vector. Then, a feature selection algorithm to find an optimal subset of the constructed features leading to high classification accuracy is applied. The performance of the proposed method on a real-world dataset shows an improvement when compared to other sequential statistical relational techniques.

Acknowledgment

This work is partially founded by the MBLab Italian MIUR-FAR project: "The Molecular Biodiversity LABoratory Initiative" (DM19410).

References

1. Kersting, K., De Raedt, L., Gutmann, B., Karwath, A., Landwehr, N.: Relational sequence learning. In De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S.,

[2] An *emerging pattern* is a pattern whose support in one class differs from its support in others. A *jumping emerging* pattern is a pattern with non-zero support on a class and zero support on all other classes, i.e. with confidence equal to 1.

eds.: Probabilistic Inductive Logic Programming. Volume 4911 of LNCS. Springer (2008) 28–55
2. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press (2007)
3. Kersting, K., De Raedt, L., , Raiko, T.: Logical hidden markov models. Journal of Artificial Intelligence Research **25** (2006) 425–456
4. Gutmann, B., Kersting, K.: TildeCFR: Conditional random fields for logical sequences. In Fürnkranz, J., Scheffer, T., Spiliopoulou, M., eds.: Proceedings of the 15th European Conference on Machine Learning. Volume 4212 of LNAI. Springer (2006) 174–185
5. Antanas, L., Gutmann, B., Thon, I., Kersting, K., De Raedt, L.: Combining video and sequential statistical relational techniques to monitor card games. In Thurau, C., Driessens, K., Missura, O., eds.: ICML Workshop on Machine Learning and Games. (2010)
6. Kramer, S., Lavrac, N., Flach, P.: Propositionalization approaches to relational data mining. In Dzeroski, S., Lavrac, N., eds.: Relational Data Mining. Springer (2001) 262–291
7. Dehaspe, L., Toivonen, H., King, R.: Finding frequent substructures in chemical compounds. In Agrawal, R., Stolorz, P., Piatetsky-Shapiro, G., eds.: 4th International Conference on Knowledge Discovery and Data Mining. AAAI Press. (1998) 30–36
8. King, R.D., Srinivasan, A., DeHaspe, L.: Warmr: A data mining tool for chemical data. Journal of Computer-Aided Molecular Design **15**(2) (2001) 173–181
9. Kramer, S., De Raedt, L.: Feature construction with version spaces for biochemical applications. In: Proceedings of the 18th International Conference on Machine Learning. Morgan Kaufmann Publishers Inc. (2001) 258–265
10. Hoos, H., Stützle, T.: Stochastic Local Search: Foundations & Applications. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2004)
11. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization **6** (1995) 109–133
12. Muggleton, S., De Raedt, L.: Inductive logic programming: Theory and methods. Journal of Logic Programming **19/20** (1994) 629–679
13. Lee, S., De Raedt, L.: Constraint based mining of first order sequences in SeqLog. In Meo, R., Lanzi, P., Klemettinen, M., eds.: Database Support for Data Mining Applications. Volume 2682 of LNCS. Springer (2004) 154–173
14. Kersting, K., Gärtner, T.: Fisher kernels for logical sequences. In Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D., eds.: Proceedings of the 15th European Conference on Machine Learning. Volume 3201 of LNCS. Springer (2004) 205–216
15. Esposito, F., Di Mauro, N., Basile, T., Ferilli, S.: Multi-dimensional relational sequence mining. Fundamenta Informaticae **89**(1) (2008) 23–43
16. Ferilli, S., Di Mauro, N., Basile, T., Esposito, F.: θ-subsumption and resolution: A new algorithm. In Zhong, N., Raś, Z.W., Tsumoto, S., Suzuki, E., eds.: Foundations of Intelligent Systems. Volume 2871 of LNCS., Springer Verlag (2003) 384–391
17. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the International Conference on Data Engineering. (1995) 3–14