Intelligent Information Systems 9999
ISBN 666-666-666, pages 1–14

Modelling (Bio)Logical Sequences through Markov Logic Networks

Marenglen Biba, Stefano Ferilli and Floriana Esposito

Department of Computer Science, University of Bari, Italy
Via E. Orabona, 4, 70125, Bari, Italy
{biba,ferilli,esposito}@di.uniba.it

Abstract

Markov Logic Networks are a powerful representation that combines first-order logic and probability by attaching weights to first-order formulas and using these as templates for features of Markov networks. In this paper we propose a simple temporal extension of MLN that is able to deal with sequences of logical atoms. We also propose iterated robust tabu search (IRoTS) for MAP inference and Markov Chain-IRoTS (MC-IRoTS) for conditional inference in the new framework. We show how MC-IRoTS can also be used for discriminative weight learning. As application domain, we describe how sequences of protein secondary structure can be modelled through the proposed language and show through some preliminary experiments the promise of our approach for the problem of protein fold recognition from these sequences.

1 Introduction

Many real-world application domains are characterized by both uncertainty and complex relational structure. Statistical learning focuses on the former, and relational learning on the latter. Statistical Relational Learning [10] aims at combining the power of both. One of the representation formalisms in this area is Markov Logic which subsumes both finite first-order logic and probabilistic graphical models as special cases [26]. Upon this formalism, Markov Logic Networks (MLNs) can be built serving as templates for constructing Markov Networks (MNs). In Markov Logic a weight is attached to each clause and learning an MLN consists in structure learning (learning the clauses) and weight learning (setting the weight of each clause).

Dealing with sequential data has become an important application area of machine learning. Such data are frequently found in computational biology, speech recognition, activity recognition, information extraction, etc. One of the main problems in this area of machine learning is assigning labels to sequences of objects. This class of problems has been called *sequential supervised learning* [8]. Probabilistic graphical models and in particular Hidden Markov Models (HMM) have been quite successful in modeling sequential phenomena. However, the main weaknesses for this model are: 1. It handles sequences of flat alphabets only (i.e.

sequential objects have no structure) and 2. It is hard to express dependencies in the input data. Recently, to overcome the first problem, the work in [18], introduced Logical Hidden Markov Models (LoHMM), an extension of HMM to handle sequences of logical atoms. However, the second problem still remains for LoHMM. For this reason, conditional random fields (CRFs) [14] have been proposed. CRFs are discriminatively trained graphical models instead of generatively trained such as HMMs. CRFs can easily handle non-independent input features and represent the conditional probability distribution $P(Y|X)$ where X represents elements of the input space and Y of the output space. For many tasks in computational biology, information extraction or user modeling CRF have outperformed HMMs

One of the problems where sequences exhibit internal structure is modeling sequences of protein secondary structure. These sequences can be seen as sequences of logical atoms (details about logic can be found in [9]). For example the following sequence of the TIM beta/alpha-barrel protein:

$$st('SB', null, medium), st('SB', plus, medium), he(h(right, alpha), long),$$

$$st('SB', plus, medium), he(h(right, alpha), medium), ...$$

represents a sequence of logical atoms. Helices and strands are represented respectively by *he(type,length)* and *st(orientation, length)*. Traditional HMMs or CRFs would ignore the structure of the symbols in the sequence loosing therefore the structure that each symbol implies or would take into account all the possible combinations (of orientation and length) into account that could lead to a combinatiorial explosion of the number of parameters.

The first approach to dealing with sequences of logical atoms by extending CRFs is that of [12] where the authors propose TildeCRF that uses relational regression trees in the gradient tree boosting approach [8] to make relational abstraction through logical variables and unification. The authors showed that TildeCRF outperformed previous approaches based on LoHMMs such as [12, 7].

In this paper we propose Stochastic MLNs, a simple model based on MLNs that is able to deal with sequences of logical atoms. We also propose two algorithms for inference and learning in SMLNs based on the iterated robust tabu search metaheuristic. Finally, we model in SMLNs the problem of protein fold recognition from sequences of protein secondary structure and show through some preliminary experiments the promise of our approach.

The paper is organized as follows: in Section 2 we introduce MNs and MLNs, in Section 3 we describe existing learning approaches for MLNs, Section 4 introduces Stochastic MLNs, Section 5 introduces the Iterated Local Search (ILS) and robust tabu search (RoTS) metaheuristic and the satisfiability solver Iterated Robust Tabu Search (IRoTS) that combines both and describes how IRoTS can be used for MAP inference in MLNs instead of the Viterbi algorithm, Section 6 introduces Markov Chain IRoTS (MC-IRoTS) for performing inference in MLNs, Section 7 describes how protein sequences can be modeled in SMLNs. Section 8 presents some preliminary experiments and Section 9 concludes with future work.

2 Markov Networks and Markov Logic Networks

A Markov Network (or Markov random field) represents a model for the joint distribution of a set of variables X = (X_1,X_2,...,X_n) $\in \chi$ [5] (in this paper we deal only with discrete features and variables). The model has an undirected graph G and a set of potential functions. There is a node for each variable and a potential function ϕ_k for each clique in the graph (a clique in an undirected graph G, is a set of vertices V, such that for every two vertices in V, there exists an edge connecting the two). A potential function is a non-negative real-valued function of the state of the corresponding clique. The joint distribution defined by a MN is given by the following:

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}) \tag{1}$$

where $x_{\{k\}}$ is the state of the kth clique (i.e., the state of the variables that appear in that clique). Z is known as the *partition function* and is given by:

$$Z = \sum_{x \in \chi} \prod_k \phi_k(x_{\{k\}}) \tag{2}$$

MNs are often represented as log-linear models, by replacing each clique potential with an exponentiated weighted sum of features of the clique state. This replacement gives the following:

$$P(X = x) = \frac{1}{Z} \exp(\sum_j w_j f_j(x)) \tag{3}$$

A feature f may be any real-valued function of the state. The focus of this paper is on binary features, $f_j \in \{0, 1\}$. Thus, if we translate from the potential-function form, the model will have one feature for each possible state x_k of each clique and its weight will be $\log(\phi(x_{\{k\}})$. This representation is exponential in the size of the cliques, but however, we can specify a much smaller number of features in a more compact representation than the potential-function form. This is the case when large cliques are present and MLNs try to take advantage of this.

A first-order knowledge base (KB) can be considered as a set of hard constraints on a set of possible worlds: if a world violates a single formula, it will have zero probability. The idea in Markov Logic is to soften these constraints: when a world violates a formula in the KB it will be less probable, but not impossible. The fewer formulas a world will violate, the more probable it will be. Each formula has an attached weight that represents how hard a constraint it is. A higher weight of a formula means there is a greater difference in log probability between a world that satisfies that formula and one that does not, all other things being equal.

An MLN [26] T is a set of pairs $(F_i; w_i)$, where F_i is a formula in FOL and w_i is a real number. Together with a finite set of constants $C = \{c_1, c_2, \ldots, c_p\}$ it defines a MN $M_{T;C}$ as follows:

1. There is a binary node in $M_{T;C}$ for each possible grounding of each predicate appearing in T and the value of the node is 1 if the ground predicate is true,

and 0 otherwise.

2. There is one feature in $M_{T;C}$ for each possible grounding of each formula F_i in T and the value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight w_i of the formula F_i in T becomes the weight of this feature. There is an edge between two nodes of $M_{T;C}$ iff the corresponding ground predicates appear together in at least one grounding of a formula in T.

An MLN can be viewed as a template for constructing MNs. The probability distribution over possible worlds x defined by the ground MN $M_{T;C}$ is given by:

$$P(X = x) = \frac{1}{Z} \exp(\sum_{i=1}^{F} w_i n_i(x)) \tag{4}$$

where F is the number of formulas in T and $n_i(x)$ is the number of true groundings of F_i in x. When formula weights increase, an MLN will resemble a purely logical KB and in the limit of all infinite weights it becomes equivalent to it.

The focus of this paper is on MLNs with function-free clauses assuming domain closure in order to ensure that the MNs generated will be finite. In this case, the groundings of a formula are formed by replacing the variables with constants in all possible ways.

3 Learning Approaches for MLNs

The first attempt to learn MLNs structure was that in [26], where the authors used an ILP system to learn the clauses and then learned the weights by maximizing pseudo-likelihood [1]. In [19] another method was proposed that combines ideas from ILP and feature induction of Markov networks. This algorithm, that performs a beam or shortest first search in the space of clauses guided by a weighted pseudo-log-likelihood (WPLL) measure, outperformed that of [26]. Recently, in [16] a bottom-up approach was proposed in order to reduce the search space. This algorithm uses a propositional Markov network learning method to construct template networks that guide the construction of candidate clauses. In this way, it generates fewer candidates for evaluation. Recently a structure learning algorithm based on ILS was proposed in [2] and it was shown to improve over those in [16, 19]. For every candidate structure, in all these algorithms the parameters that optimize the WPLL are set through L-BFGS that approximates the second-derivative of the WPLL by keeping a running finite-sized window of previous first-derivatives. Another algorithm that works in a discriminative fashion was proposed in [3], that scores structures by conditional likelihood and learns parameters by maximum likelihood.

Learning MLNs in a discriminative fashion has produced for predictive tasks much better results than generative approches as the results in [28] show. In this work the voted-perceptron algorithm was generalized to arbitrary MLNs by replacing the Viterbi algorithm with a weighted satisfiability solver. The new algorithm is gradient descent with an MPE approximation to the expected sufficient statistics (true clause counts). These could vary widely between clauses, causing the learning problem to be highly ill-conditioned, and making gradient descent very slow. In [17] a preconditioned scaled conjugate gradient approach is shown to outperform the algorithm in [28] in terms of learning time and prediction accuracy. This algorithm is based on the scaled conjugate gradient method and very good results are obtained with a simple approach: per-weight learning weights, with the weight's learning rate being the global one divided by the corresponding clause's empirical number of true groundings.

4 Temporal Extension of Markov Logic

In stochastic processes the world evolves and a ground predicate's truth value depends on the time step t. In order to model in MLNs the evolution of objects and relations we need to represent time. To achieve this we introduce the concept of *temporal predicates* which have an additional time argument. Time is represented as a non-negative integer variable and all predicates are of the form $P(x_1, ..., x_n, t)$ where t denotes time.

We define Stochastic Markov Logic Networks (SMLNs) as a set of MLN formulas defined on the *temporal predicates*. In addition, we borrow from Linear Temporal Logic [21] the concept of *temporal operator* and introduce *succ*, a *time predicate* (similar to the operator *next* in LTL) that represents the successor of a time step t, i.e., succ(1,2), succ(2,3) and so on.

In principle, the structure of SMLNs could be learned in a similar fashion as for MLNs. This task could be made easier by imposing further restrictions such as the Markovian assumption, or through the use of *declarative bias*. We plan to perform this in the future by using the structure learning algorithms proposed in [2, 3]. In this paper, we assume the structure of the model to be fixed and try to learn optimal parameters for discriminative training of SMLNs for the problem of protein fold recognition from sequences of secondary structure.

5 MAP inference using Iterated Robust Tabu Search

In logic, one of the main problems is determining if a KB is satisfiable (SAT problem), i.e., if there is a truth assignment to ground atoms that make the KB true. This problem is NP-complete. However, Stochastic Local Search (SLS) methods have made great progress towards efficiently solving SAT problems with hundreds of thousands of variables in a few minutes. The optimization variant of SAT is MAX-SAT where the problem is to find a truth assignment that maximizes the number of satisfied clauses (unweighted MAX-SAT) or if clauses have an associated weight, maximize the total weight of the satisfied clauses (weighted MAX-SAT). Both, unweighted and weighted MAX-SAT are NP-complete problems. First-order problems can also be successfully solved through SLS methods by performing first a propositionalization and then applying a SAT solver.

The currently best performing SLS algorithms for MAX-SAT are Tabu Search (TS) algorithms, Dynamic Local Search and Iterated Local Search (ILS). Here we will describe a combination of a variant of TS with ILS to build a hybrid SAT solver that we will use for inference and learning in SMLNs.

5.1 Iterated Local Search and Robust Tabu Search

Many widely known and high-performance local search algorithms make use of randomized choice in generating or selecting candidate solutions for a given combinatorial problem instance. These algorithms are called SLS algorithms [13] and represent one of the most successful and widely used approaches for solving hard combinatorial problem. Many "simple" SLS methods come from other search methods by just randomizing the selection of the candidates during search, such as Randomized Iterative Improvement (RII), Uniformed Random Walk, etc. Many other SLS methods combine "simple" SLS methods to exploit the abilities of each of these during search. These are known as Hybrid SLS methods [13]. ILS is one of these metaheuristics because it can be easily combined with other SLS methods.

One of the simplest and most intuitive ideas for addressing the fundamental issue of escaping local optima is to use two types of SLS steps: one for reaching local optima as efficiently as possible, and the other for effectively escaping local optima. ILS methods [13, 15] exploit this key idea, and essentially use two types of search steps alternatingly to perform a walk in the space of local optima w.r.t the given evaluation function. The algorithm works as follows: The search process starts from a randomly selected element of the search space. From this initial candidate solution, a locally optimal solution is obtained by applying a subsidiary local search procedure. Then each iteration step of the algorithm consists of three major steps: first a perturbation method is applied to the current candidate solution s; this yields a modified candidate solution *s'* from which in the next step a subsidiary local search is performed until a local optimum *s"* is obtained. In the last third step, an acceptance criterion is used to decide from which of the two local optima s or *s'* the search process is continued. The algorithm can terminate after some steps have not produced improvement or simply after a certain number of steps. The choice of the components of the ILS has a great impact on the performance of the algorithm.

Robust Tabu Search (RoTS) [31] is a special case of Tabu Search. In each search step, the RoTS algorithm for MAX-SAT flips a non-tabu variable that achieves a maximal improvement in the total weight of the unsatisfied clauses (the size of this improvement is also called score) and declares it tabu for the next tt steps. The parameter tt is called the tabu tenure. An exception to this tabu rule is made if a more recently flipped variable achieves an improvement over the best solution seen so far (this mechanism is called aspiration). Furthermore, whenever a variable has not been flipped within a certain number of search steps (we use 10n), it is forced to be flipped. This implements a form of long-term memory and helps prevent stagnation of the search process. The tabu status of variables is determined by comparing the number of search steps that have been performed since the most recent flip of a given variable with the current tabu tenure.

5.2 Iterated Robust Tabu Search

The original version of IRoTS for MAX-SAT was proposed in [30]. Algorithm 1 starts by independently (with equal prabability) initializing the truth values of the atoms. Then it performs a local search to efficiently reach a local optimum

Algorithm 1 Iterated Robust Tabu Search

```
Input: C: set of weighted clauses in CNF, BestScore: current best score)
CL$_C$ = Random initialization of truth values for atoms in C;
CL$_S$ = $LocalSearch_{RoTS}$(CL$_S$);
BestAssignment = CL$_S$;
BestScore = Score(CL$_S$);
repeat
  CL'$_C$ = $Perturb_{RoTS}$(BestAssignment);
  CL'$_S$ = $LocalSearch_{RoTS}$(CL'$_C$);
  if Score(CL'$_S$) $\geq$ BestScore then
    BestScore = Score(CL'$_S$)
  end if
  BestAssignment = accept(BestAssignment,CL'$_S$);
until two consecutive steps have not produced improvement
Return BestAssignment
```

CL_S by using RoTS. At this point, a perturbation method based again on RoTS is applied leading to the neighbor CL'_C of CL_S and then again a local search based on RoTS is applied to CL'_C to reach another local optimum CL'_S . The *accept* function decides whether the search must continue from the previous local optimum or from the last found local optimum CL'_S (*accept* can perform random walk or iterative improvement in the space of local optima).

Careful choice of the various components of Algorithm 1 is important to achieve high performance. For the tabu tenure we refer to the parameters used in [30] that have proven to be highly performant accross many domains. At the begining of each local search and perturbation phase, all variables are declared non-tabu. The clause perturbation operator (flipping the atoms truth value) has the goal to jump in a different region of the search space where search should start with the next iteration. There can be strong or weak perturbations which means that if the jump in the search space is near to the current local optimum the subsidiary local search procedure *LocalSearch*$_{RoTS}$ may fall again in the same local optimum and enter regions with the same value of the objective function called *plateau*, but if the jump is too far, *LocalSearch*$_{RoTS}$ may take too many steps to reach another good solution. In our algorithm we use a fixed number of RoTS steps 9n/10 with tabu tenure n/2 where n is the number of atoms (in future work we intend to dynamically adapt the nature of the perturbation). Regarding the procedure *LocalSearch*$_{RoTS}$, it performs RoTS steps until no improvement is achieved for $n^2/4$ steps with a tabu tenure $n/10 + 4$. The *accept* function always accepts the best solution found so far. The difference of our algorithm with that in [30] is that we do not dynamically adapt the tabu tenure and do not use a probabilistic choice in *accept*.

5.3 MAP Inference using IRoTS

Maximum *a posteriori* (MAP) inference in Markov Networks means finding the most likely state of a set of output variables given the state of the input variables.

This problem is NP-hard. For discriminative training, the voted perceptron is a special case in which tractable inference is possible using the Viterbi algorithm [4]. In [28] the voted perceptron was generalized to MLNs by replacing the Viterbi algorithm with a weighted SAT solver. This algorithm is gradient descent and computing the gradient of the conditional log-likelihood (CLL) requires the computation of the number of true groundings for each clause. This can be performed by finding the MAP state which can be computed by dynamic programming methods. Since for MLNs, the MAP state is the state that maximizes the sum of the weights of the satisfied ground clauses, this state can be efficiently found using a weighted MAX-SAT solver. The authors in [28] use the MaxWalkSat solver [27]. In this paper we propose to use IRoTS as a MAX-SAT solver and show how this algorithm can be applied not only to MLNs but also to the proposed extension SMLNs. IRoTS is one the best weighted MAX-SAT solvers and we want to show also how MAP inference in SMLNs can benefit from it.

Given a SMLN in the form of clauses based on *temporal predicates*, including a time predicate and a set of evidence atoms, the KB to be used as input for IRoTS is formed by constructing all groundings of clauses in the SMLNs involving query atoms. Then the evidence atoms are replaced by their true values followed by semplification. Once the SMLN has been propositionalized it is a natural input to IRoTS.

6 Markov Chain Iterated Robust Tabu Search

In this section we describe how IRoTS can be combined with Markov Chain Monte Carlo (MCMC) in order to uniformly sample from the space of satisfying assignments of a clause. We show how the proposed algorithm MC-IRoTS can be used for inference and learning in SMLNs.

6.1 Conditional Inference through MC-IRoTS

Conditional inference in graphical models involves computing the distribution of the query variables given the evidence and it has been shown to be #P-complete. The most widely used approach to approximate inference is by using MCMC methods and in particular Gibbs sampling. One of the problems that arises in real-world applications, is that an inference method must be able to handle probabilistic and deterministic dependencies that might hold in the domain. MCMC methods are suitale for handling probabilistic dependencies but give poor results when deterministic or near deterministic dependencies characterize a certain domain. On the other side logical ones, like satisfiability testing cannot be applied to probabilistic dependencies. One approach to deal with both kinds of dependencies is that of [24] where the authors use SampleSAT [32] in a MCMC algorithm to uniformly sample from the set of satisfying solutions. As pointed out in [32], SAT solvers find solutions very fast but they may sample highly non-uniformly. On the other side, MCMC methods may take exponential time, in terms of problem size, to reach the stationary distribution. For this reason, the authors in [32] proposed to use a hybrid strategy ny combining random walk steps with MCMC steps, and in particular with Metropolis transitions. This permits to efficiently jump between isolated or near-isolated regions of non-zero probability, while preserving detailed balance.

We use the same approach as the authors did in [24], but instead of SampleSAT, for MC-IRoTS we propose to use SampleIRoTS, which performs with probability p a RoTS step and with probability $1 - p$ a simulated annealing (SA) step. We used fixed temperature annealing (i.e., Metropolis) moves. The goal is to reach as fast as possible a first solution through IRoTS and then exploit the ability of SA to explore a cluster of solutions. A cluster of solutions is usually a set of connected solutions, so that any two solutions within the cluster can be connected through a series of flips without leaving the cluster. In many domains of interest, solutions exist in clusters and it is highly useful to explore such clusters without leaving them. SA has good properties in exploring a connected space, therefore it samples near-uniformly and often explores all the neighboring solutions.

Through MC-IRoTS we can perform conditional inference given evidence to compute probabilites for query predicates. These probabilites can be used to make predictions from the model.

6.2 Discriminative Learning by Sampling with MC-IRoTS

Discriminative approaches to weight learning try to optimize the conditional log-likelihood (CLL). Preconditioned Scaled Conjugate Gradient (PSCG) is the state-of-the-art discriminative training algorithm for MLN and it was shown in [17] to outperform the voted perceptron. PSCG is a *conjugate gradient* method that uses samples from MC-SAT to approximate the Hessian for MLNs instead of the line search to choose a step size. This approach is also known as *scaled conjugate gradient* and was originally proposed in [23] for training neural networks. PSCG, in each iteration, takes a step in the diagonalized Newton direction (for details see [17]). Here we propose to use MC-IRoTS to sample for approximating the Hessian for SMLNs. The goal is to use samples from MC-IRoTS that can serve as good estimates for computing the Hessian.

7 Modelling Protein Sequences in SMLNs

In this section we describe how sequences of protein secondary structure can be modelled in SMLNs, how to learn model parameters from the data and how to make predictions from the model.

7.1 Model Construction and Weight Learning

The approach we follow is quite simple: we write a few formulas that represent the structure of the domain and then from the training sequences we learn the weights of these formulas.

The dataset we refer to is that used in [12]. The data consists of logical sequences of the secondary structure of protein domains:

$beginSequence.$
$strand('SB', null, medium).$

$strand('SB', plus, medium).$
$helix(right, alpha, long).$
...
...
$strand('SB', plus, medium).$
$helix(right, alpha, medium).$
$strand('SB', plus, short).$
$endSequence.$

A simple SMLN that can be used to model these data can be the following:

//predicates
Helix(*wing*, *typeh*, *length*, *time*)
Strand(*types*, *pm*, *length*, *time*)
Succ(*time*, *time*)
//rules

$$Helix(Right, +ty1, +l1, +t1) \wedge Succ(t2, t1) \wedge Helix(Right, +ty2, +l2, t2)$$

$$Helix(Right, +ty1, +l1, +t1) \wedge Succ(t2, t1) \wedge Strand(typ1, +p1, +le1, t2)$$

$$Strand(typ1, +p1, +le1, +t1) \wedge Succ(t2, t1) \wedge Helix(Right, +ty2, +l2, t2)$$

$$Strand(typ1, +p1, +le1, +t1) \wedge Succ(t2, t1) \wedge Strand(typ1, +p2, +le2, t2)$$

$$Strand(typ1, +p1, +le1, +t1) \wedge Succ(t2, t1) \wedge Strand(typ2, +p2, +le2, t2)$$

The first three expressions, represent the *temporal predicates* (Helix and Strand) and the *time predicate* Succ. The rules express temporal relations between Helix and Strand. The first rule expresses that a helix is followed by another helix, the second rules states that helix is followed by a strand and so on. The last two rules differ in that one states that strand is followed by another strand of the same type, while the other states that two strands of different types appear in the sequence one following the other. The + operator is used to express that the argument should be grounded, so that a weight is learned for each grounded formula. If multiple variables are preceded by +, a weight is learned for each combination of their values.

It must be noted that this model is not the only one. We have stated some regularities that in general are true but we would like to learn weights that can express how strong each rule it is. A reasonable model would also be that of learning a rule for each grounding of the argument that expresses the type of strand, i.e., $Strand(+typ1, p1, le1, t1)$ We plan to experiment this in the future.

7.2 Making Predictions from the Learned SMLNs

There are several ways in which a SMLNs can be used to make predictions regarding sequences. One way is to predict the sequence with highest probability (this is performed in other models by the Viterbi algorithm). In SMLNs this means performing MAP inference given the evidence. For example, given a certain sequence and a query predicate (helix or strand), we can perform MAP inference through IRoTS and return the positive query atoms. Then we can count how many of the returned positive query atoms are true in the sequence and consider these atoms *correctly classified.* If there are several models (one for each protein fold) a majority vote can be used to assign the sequence, i.e., the sequence belongs to the model that gives the highest number of *correctly classified* atoms for that sequence.

Another way to get a classifier is to get the probability of query atoms given evidence through MC-IRoTS. In this case CLL (conditional log-likelihood averaged over all the groundings of the query predicate where predicted probability p is summed for positive atoms and 1 - p for negative ones) can be used to assign the sequence to the fold that produces the highest CLL.

8 Experiments

Through some preliminary experiments we want to answer the following questions:

(Q1) Are SMLNs suitable for modeling and learning with sequences of logical atoms?

(Q2) Can we use SMLNs to make predictions for sequences and are these predictions reasonably good?

8.1 Dataset and Systems

We implemented the algorithms IRoTS and MC-IRoTS as extensions of the alchemy system [20] and used the implementation of PSCG in this package to learn weights for the SMLN.

The protein fold classification task is to predict one of the five most populated SCOP folds of alpha and beta proteins (a/b). We will use for our experiments only two folds from the dataset in [12] with a subset of the whole sequences. (Our goal here is to show how SMLNs can be used for sequence modeling/learning and not to boost performance or compare with other methods. For this reason we did not optimize any parameters for the learning and inference algorithms). We randomly extracted 80 sequences (totally 160) from each of the folds TIM beta/alpha-barrel (c1) and NAD(P)-binding Rossmann-fold (c2). We divided this set in the training set (60 sequences) and test set (20 sequences). The classification problem is a multi-class one. We will learn for each fold a model based on the SMLN presented in the previous section and will test both models on the 40 test sequences (thus for each model we have 20 positive and 20 negative testing sequences)

In order to learn weights for the SMLN presented in the previous section we used PSCG with MC-IRoTS as a sampler. We performed two weight learning experiments, one for each protein fold giving in input the same SMLN. We used 100 iterations for PSCG with 50 samples for MC-IRoTS. We used the lazy version of inference in alchemy [25]. (all these parameters could be greatly optimized and we plan to do this in the future for more complete experiments).

8.2 Results

After learning a SMLN on each training set of 60 sequences we performed MAP inference through IRoTS with both learned models on the two testing sets. For each sequence, every model produced positive query atoms and we checked the number of predicted positive atoms that were true in the sequence. The sequence was assigned to the model that infered the highest number of correct predictions. From the 40 test sequences, 31 sequences were correctly assigned to the belonging fold, 4 sequences were predicted equally by both models and 5 sequences were classified incorrectly. For the equally predicted sequences we decided to use MC-IRoTS to perform inference over these sequence in order to get CLL for each one. Surprisingly, using the probability for each atom in the sequence, all these cases were correctly classified, thus we achieved a classification rate of 35 out of 40.

The results respond to questions **(Q1)** and **(Q2)**. Since the scope of the experiments was only to confirm these questions, we did not perform cross-validation for accuracy evaluation or parameters optimization to increase perfomance. We plan, howefer, to perform extensive experiments and compare with state-of-the-art methods such that in [12].

9 Conclusions and Future Work

Markov Logic Networks are a powerful representation that combines first-order logic and probability by attaching weights to first-order formulas and viewing these as templates for features of Markov networks. In this paper we have proposed Stochastic Markov Logic Networks, (SMLNs), a simple extension of MLN that is able to deal with sequences of logical atoms. We also propose iterated robust tabu search (IRoTS) for MAP inference in SMLNs and Markov Chain-IRoTS (MC-IRoTS) for conditional inference in SMLNs. We show how MC-IRoTS can be also used for discriminative weight learning in SMLNs. As application domain, we have described how sequences of protein secondary structure can be modelled in SMLNs and have shown through some preliminary experiments the promise of our approach.

Future work regarding the algorithms proposed includes: comparing performace of IRoTS toward MaxWalkSat regarding MAP inference in MLNs and SMLNs; comparing conditional inference results of MC-IRoTS towards MC-SAT; comparing learning results with PSCG by using MC-SAT and MC-IRoTS; compare sampling of SampleSAT and SampleIRoTS regarding uniformity.

Regarding protein sequence modelling in SMLNs we intend to: perform extensive experiments with SMLNs for all the five folds of the protein secondary structure problem; optimize the parameters of IRoTS and MC-IRoTS for learning and inference; evaluate the best way to make predictions from SMLNs by comparing the use of MAP inference with CLL results; verify in inference if the use of AUC (area under curve) for precision-recall curves can help improve classification accuracy; finally investigate if structure learning algorithms such as that in [2, 3] can be used to learn the rules from the sequences and if this improves over a fixed structure.

Regarding SMLNs in general, we would like to apply this simple extension of

MLNs to more complex domains where stochastic relational problems must be handled. Natural application domains for SMLNs are areas such as Systems Biology and Gene Regulatory Networks where networks involved in stochastic processes need to be modeled.

Acknowledgements. This work was partially funded by DM19410 – Laboratorio di Bioinformatica per la Biodiversità Molecolare. We would like to thank Pedro Domingos for helpful discussions on MLNs, Marc Sumner for his help on alchemy, Daniel Lowd for his help on PSCG and Hoifung Poon for discussions on MC-SAT. We thank Kristian Kersting for kindly providing us the dataset on proteins sequences. Finally, we also thank the anonymous reviewers for their useful comments.

References

[1] Besag, J. Statistical analysis of non-lattice data. *Statistician*, 24:179–195, 1975.

[2] Biba, M., Ferilli, S., and Esposito, F. Structure Learning of Markov Logic Networks through Iterated Local Search. In *Proc. 18th ECAI*, 2008. To appear.

[3] Biba, M. and Ferilli, S. and Esposito, F. Discriminative Structure Learning of Markov Logic Networks . In Proceedings of 18th International Conference on Inductive Logic Programming, (ILP 2008), LNCS 5194, (pp. 59-76), Springer, 2008.

[4] Collins, M. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In Proc. of the 2002 Conference on Empirical Methods in Natural Language Processing, 18. Philadelphia, PA: ACL, 2002.

[5] Della Pietra, S., Pietra, V. D., and Laferty, J. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–392, 1997.

[6] De Raedt, L., and Dehaspe, L. Clausal discovery. *Machine Learning*, 26:99–146, 1997.

[7] Dick, U., Kersting, K. Fisher Kernels for Relational Data. In J. Fuernkranz, T. Scheffer, M. Spiliopoulou, editors, Proc. 17th ECML, pages 114-125, 2006.

[8] Dietterich, T., Ashenfelter, A., Bulatov. Y. Training conditional random fields via gradient tree boosting. ICML 2004.

[9] Genesereth, M. R., and Nilsson, N. J. *Logical foundations of articial intelligence.* San Mateo, CA: Morgan Kaufmann, 1987.

[10] Getoor, L., and Taskar, B. *Introduction to statistical relational learning.* MIT Press, 2007.

[11] Grossman, D., and Domingos, P. Learning bayesian network classiers by maximizing conditional likelihood. In *Proc. 21st Int'l Conf. on Machine Learning*, pages 361–368, Banf, Canada: ACM Press, 2004.

[12] Gutmann, B., and Kersting, K. TildeCRF: Conditional Random Fields for Logical Sequences. In J. Fuernkranz, T. Scheffer, M. Spiliopoulou, editors, Proc. of the 17th ECML, pages 174-185, 2006.

[13] Hoos, H. H., and Stutzle, T. *Stochastic local search: Foundations and applications.* Morgan Kaufmann, San Francisco, 2005.

[14] Laferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th Int'l Conf. on Machine Learning*, pages 282–289, 2001.

[15] Loureno, H.R., Martin, O., and Stutzle, T. Iterated local search. In *Handbook of Metaheuristics*, pages 321–353, Kluwer Academic Publishers, 2002.

[16] Mihalkova, L., and Mooney, R. J. Bottom-up learning of markov logic network structure. In *Proc. 24th Int'l Conf. on Machine Learning*, pages 625–632, 2007.

[17] Lowd, D., and Domingos, P. Efficient weight learning for markov logic networks. In *Proc. of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages, 200–211, 2007.

[18] Kersting, K., De Raedt, L., and Raiko, T. Logial Hidden Markov Models. Journal of Articial Intelligence Research (JAIR), 25:425456, 2006.

[19] Kok, S., and Domingos, P. Learning the structure of markov logic networks. In *Proc, 22nd Int'l Conf. on Machine Learning*, pages 441–448, 2005.

[20] Kok, S., Singla, P., Richardson, M., and Domingos, P.: The alchemy system for statistical relational ai (Technical Report). Department of Computer Science and Engineering, University of Washington, Seattle, WA, http://alchemy.cs.washington.edu/, 2005.

[21] Manna Z., and Pnueli, A. The Temporal Logic of Reactive and Concurrent Systems. Springer, 1992.

[22] McCallum, A. Efficiently inducing features of conditional random fields. In *Proc. 19th Conf. on Uncertainty in Artificial Intelligence*, pages 403–410, 2003.

[23] M. Moller. A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks, 6:525533, 1993.

[24] Poon, H., and Domingos, P. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proc. 21st Nat'l Conf. on Articial Intelligence*, pages 458–463, AAAI Press, 2006.

[25] Poon, H., Domingos, P., and Sumner, M. A General Method for Reducing the Complexity of Relational Inference and its Application to MCMC. In *Proc. 23rd Nat'l Conf. on Articial Intelligence*, 2008, Chicago, IL: AAAI Press. To appear.

[26] Richardson, M., and Domingos, P. Markov logic networks. *Machine Learning*, 62:107–236, 2006.

[27] Selman, B., Kautz, H., and Cohen, B. Local search strategies for satisability testing. In Johnson, D. S., and Trick, M. A., eds., Cliques, Coloring, and Satisability: Second DIMACS Implementation Challenge, American Mathematical Society. 521532, 1996.

[28] Singla, P., and Domingos, P. Discriminative training of markov logic networks. In *Proc. 20th Nat'l Conf. on Articial Intelligence*, pages 868–873, AAAI Press, 2005.

[29] Sha, F., and Pereira, F. Shallow parsing with conditional random fields. In *Proc. HLT-NAACL*, pages 134–141, 2003.

[30] Smyth, K., Hoos, H., Sttzle, T. Iterated Robust Tabu Search for MAX-SAT. Canadian Conference on AI, 129-144, 2003.

[31] Taillard. E.D. Robust taboo search for the quadratic assignment problem. Parallel Computing, 17:443455, 1991.

[32] Wei, W., Erenrich, J., Selman, B. Towards efficient sampling: Exploiting random walk strategies. In AAAI-04.