Hybrid SRL with Optimization Modulo Theories

Stefano Teso **Roberto Sebastiani** **Andrea Passerini**
Department of Information Engineering and Computer Science
University of Trento
Povo, Trento I-38123, Italy
{teso,roberto.sebastiani,passerini}@disi.unitn.it

Abstract

Generally speaking, the goal of constructive learning could be seen as, given an example set of structured objects, to generate *novel* objects with similar properties. From a statistical-relational learning (SRL) viewpoint, the task can be interpreted as a constraint satisfaction problem, i.e. the generated objects must obey a set of soft constraints, whose weights are estimated from the data. Traditional SRL approaches rely on (finite) First-Order Logic (FOL) as a description language, and on MAX-SAT solvers to perform inference. Alas, FOL is unsuited for constructive problems where the objects contain a mixture of Boolean and numerical variables. It is in fact difficult to implement, e.g. linear arithmetic constraints within the language of FOL. In this paper we propose a novel class of hybrid SRL methods that rely on Satisfiability Modulo Theories, an alternative class of formal languages that allow to describe, and reason over, mixed Boolean-numerical objects and constraints. The resulting methods, which we call *Learning Modulo Theories*, are formulated within the structured output SVM framework, and employ a weighted SMT solver as an optimization oracle to perform efficient inference and discriminative max margin weight learning. We also present a few examples of constructive learning applications enabled by our method.

1 Introduction

Traditional statistical-relational learning (SRL) methods allow to reason and make inference about relational objects characterized by a set of soft constraints [1]. Most methods rely on some form of (finite) First-Order Logic (FOL) to encode the learning problem, and define the constraints as weighted logical formulae. In this context, maximum a posteriori inference is often interpreted as a (partial weighted) MAX-SAT problem, i.e. finding a truth value *assignment* of all predicates that maximizes the total weight of the satisfied formulae; moreover, MAX-SAT plays a role in maximum likelihood inference as well. In order to solve this problem, SRL methods may rely on one of the many efficient, approximate solvers available. One issue with these approaches is that First-Order Logic is not suited for reasoning over hybrid variables. The propositionalization of an n-bit integer variable requires n distinct binary predicates, which account for 2^n distinct states, making naïve translation impractical. In addition, FOL offers no efficient mechanism to describe simple operators between numerical variables, like comparisons (e.g. "less-than", "equal") and arithmetical operations (e.g. summation), limiting the range of realistically applicable constraints to those based solely on logical connectives.

In order to side-step these limitations, researchers in automated reasoning and formal verification have developed more appropriate logical languages that allow to *natively* reason over mixtures of Boolean *and* numerical variables (or more complex algebraic structures). These languages are grouped under the umbrella term of *Satisfiability Modulo Theories* (SMT) [2]. Each such language corresponds to a decidable fragment of First-Order Logic augmented with an additional background theory $\mathcal{T}$. There are many such background theories, including those of linear arithmetic over the

rationals $\mathcal{LA}(\mathbb{Q})$ and integers $\mathcal{LA}(\mathbb{Z})$, among others [2]. In SMT, a formula can contain Boolean variables (i.e. logical predicates) and connectives, mixed with symbols defined by the theory $\mathcal{T}$, e.g. rational variables and arithmetical operators. For instance, the SMT($\mathcal{LA}(\mathbb{Q})$) syntax allows to write constraints such as:

$$\texttt{HasProperty}(\mathbf{x}) \Rightarrow ((a + b) > 1024 \cdot c)$$

where the variables are Boolean (the truth value of `HasProperty`) and rational (a, b, and c). More specifically, SMT is the decision problem of finding a variable assignment that makes all logical and theory-specific formulae true, and is analogous to SAT. Recently, researchers have leveraged SMT for optimization [3]. In particular, MAX-SMT requires to maximize the *total weight* of the satisfied formulae; Optimization Modulo Theories, or OMT, requires to maximize the *amount of satisfaction* of all weighted formulae, and strictly subsumes MAX-SMT. Most important for the scope of this paper is that there are high quality MAX-SMT (and OMT) solvers, which (at least for the $\mathcal{BV}$ and $\mathcal{LA}(\mathbb{Q})$ theories) can handle problems with a large number of hybrid variables.

There is relatively little previous work on *hybrid* SRL methods. Most current approaches are direct generalizations of existing SRL methods [1]. Hybrid Markov Logic networks [4] extend Markov Logic by including continuous variables, and allow to embed numerical comparison operators (namely $\neq$, $\geq$ and $\leq$) into the constraints by defining an *ad hoc* translation of said operators to a continuous form amenable to numerical optimization. Inference relies on an MCMC procedure that interleaves calls to a MAX-SAT solver and to a numerical optimization procedure. This results in a very expensive iterative process, which can hardly scale with the size of the problem. Conversely, MAX-SMT and OMT are specifically designed to tightly integrate a theory-specific and a SAT solver, and we expect them to perform very efficiently. Some probabilistic-logical methods, e.g. ProbLog [5] and PRISM [6], have also been modified to deal with continuous random variables. These models, however, rely on probabilistic assumptions that make it difficult to implement fully expressive constraints in, e.g. linear arithmetic, in their formalism. While there are other interesting hybrid and continuous approaches in the literature, we skip over them due to space restrictions.

In this paper we propose *Learning Modulo Theories* (LMT), a class of novel hybrid statistical relational learning methods. By combining the flexibility of structured output Support Vector Machines [7] and the expressivity and Satifiability Modulo Theories, LMT is able to perform learning and inference in mixed Boolean-numerical domains. Thanks to the efficiency of the underlying OMT solver, and of the discriminative max-margin weight learning procedure we propose, we expect LMT to scale to large constructive learning problems. Furthermore, LMT is *generic*, and can in principle be applied to any of the existing SMT background theories. In the following two sections we give a short overview of SMT and detail how it can be employed with the structured output SVM framework, then we describe a few applications that can be tackled with our approach.

2 Satisfiability Modulo Theories

Propositional satisfiability, or SAT, is the problem of deciding whether a logical formula over Boolean variables and logical connectives can be satisfied by some truth value assignment of the variables. Satisfiability Modulo Theories, or SMT, generalize SAT problems by considering the satisfiability of a formula with respect to a *background theory* $\mathcal{T}$ [2]. The latter provides the meaning of predicates and function symbols that would otherwise be difficult to describe, and reason over, in classical logic. SMT is fundamental in mixed Boolean domains, which require to reason about equalities, arithmetic operations and data structures. Popular theories include, e.g. those of linear arithmetic over the rationals $\mathcal{LA}(\mathbb{Q})$ or integers $\mathcal{LA}(\mathbb{Z})$, bit-vectors $\mathcal{BV}$, strings $\mathcal{ST}$, and others. Most current SMT solvers are based on a very efficient *lazy* procedure to find a satisfying assignment of the Boolean and the theory-specific variables: the search process alternates calls to an underlying SAT procedure and a specialized theory-specific solver, until a solution satisfying both solvers is retrieved, or the problem is found to be unsatisfiable. Recently, researchers have developed methods to solve the SMT equivalent of MAX-SAT and more complex optimization problems [8]. In particular, MAX-SMT requires to maximize the *total weight* of the satisfied formulae; Optimization Modulo Theories, or OMT, requires to maximize the *amount of satisfaction* of all formulae, modulated by the formulae weights. Clearly, OMT is strictly more expressive than MAX-SMT. There are a number of very efficient MAX-SMT packages available, specialized for a subset of the available theories, such as MathSAT 5 [9], Yices [10], Barcelogic [11], which can deal with industrial grade problems. MAX-SMT solvers have been previously exploited to perform e.g., formal microcode verification

at Intel [9] and large-scale circuit analysis in synthetic biology [12]. Most important for the goal of this paper, the MathSAT 5 solver also supports full-fledged OMT problems in the $\mathcal{LA}(\mathbb{Q})$ theory of linear arithmetic [8].

3 Method Overview

Structured output SVMs [7] are a very flexible framework that generalizes max-margin methods to the case of multi-label classification with exponentially many classes. In this setting, the association between inputs $\mathbf{x} \in \mathcal{X}$ and outputs $\mathbf{y} \in \mathcal{Y}$ is controlled by a so-called *compatibility function* $f(\mathbf{x}, \mathbf{y}) \equiv \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$, defined as a linear combination of the joint feature space representation $\mathbf{\Psi}$ of the input-output pair and a vector of learned weights $\mathbf{w}$. Inference reduces to finding the most compatible output $\mathbf{y}^*$ for a given input $\mathbf{x}$:

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{x}, \mathbf{y}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}) \tag{1}$$

Performing inference is non-trivial, since the maximization ranges over an exponential number of possible outputs.

In order to learn the weights from a training set of n examples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$, we need to define a non-negative *loss function* $\Delta(\mathbf{y}_i, \mathbf{y})$ that quantifies the penalty incurred when predicting $\mathbf{y}$ instead of the correct output $\mathbf{y}_i$. Weight learning can then be expressed, following the *margin rescaling* formulation [7], as finding the weights $\mathbf{w}$ that jointly minimize the training error $\boldsymbol{\xi}$ and the model complexity:

$$\operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^d, \boldsymbol{\xi} \in \mathbb{R}^n} \|\mathbf{w}\|_2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i \tag{2}$$

$$s.t. \quad \mathbf{w}^T \left(\mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}') \right) \geq \Delta(\mathbf{y}_i, \mathbf{y}') - \xi_i, \qquad \forall\, i = 1, \ldots, n;\ \mathbf{y}' \neq \mathbf{y}_i$$

Here the constraints require that the compatibility between $\mathbf{x}_i$ and the correct output $\mathbf{y}_i$ is always higher than that with all wrong outputs $\mathbf{y}'$, with ξ_i playing the role of per-instance violations. Weight learning is a quadratic program, and can be solved very efficiently with a cutting-plane algorithm [7]. Since in Eq 2 there is an exponential number of constraints, it is infeasible to naïvely account for all of them during learning. Based on the observations that the constraints obey a subsumption relation, the CP algorithm [13] sidesteps the issue by keeping a working set of active constraints: at each iteration, it augments the working set with the most violated constraint, and then solves the corresponding reduced quadratic program. The procedure is guaranteed to find an ϵ-approximate solution to the QP in a polynomial number of iterations, independently of the cardinality of $\mathcal{Y}$ and the number of examples n [7].

The CP algorithm is generic, meaning that it can be adapted to any structured prediction problem as long as it is provided with: i) a joint feature space representation $\mathbf{\Psi}$ of input-output pairs (and consequently a compatibility function f); ii) an oracle to perform inference, i.e. Equation 1; iii) an oracle to retrieve the most violated constraint of the QP, i.e. solve the *separation* problem:

$$\operatorname*{argmax}_{\mathbf{y}'} \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}') + \Delta(\mathbf{y}_i, \mathbf{y}') \tag{3}$$

The oracles are used as sub-routines during the optimization procedure. Efficient implementations of the oracles are fundamental for the prediction to be tractable in practice. For a more detailed exposition, please refer to [7]. In the following we provide exactly the three ingredients required to apply the structured output SVM framework for predicting hybrid boolean-continuous possible worlds.

We first define the LMT joint feature space of possible words $\mathbf{z} = (\mathbf{x}, \mathbf{y})$. Our definition is grounded on the concept of *violation* or *cost* incurred by $\mathbf{z}$ with respect to a set of SMT formulae. Given m formulae $F = \{f_j\}_{j=1}^m$, we define the feature vector $\mathbf{\Psi}_F(\mathbf{z}) \equiv (\psi_{f_1}(\mathbf{z}), \ldots, \psi_{f_m}(\mathbf{z}))$ as the collation of m per-formula cost functions $\psi_f(\mathbf{z})$. In the simplest case, the individual components ψ_f are indicator functions, termed *boolean costs*, that evaluate to 0 if $\mathbf{z}$ satisfies f, and to 1 otherwise. The LMT compatibility function, written as $f(\mathbf{z}) \equiv \mathbf{w}^T \mathbf{\Psi}_F(\mathbf{z})$, represents the *total cost* incurred by a possible world: each unsatisfied formula f_j contributes an additive factor w_j to $\mathbf{\Psi}_F$, while

satisfied formulae carry no contribution. Two possible worlds $\mathbf{z}$ and $\mathbf{z}'$ are therefore close in feature space if they satisfy/violate similar sets of constraints.

Since we want the formulae to *hold* in the predicted output, we want to *minimize* the total cost of the unsatisfied rules, or equivalently maximize its opposite: $\text{argmax}_{\mathbf{y}\in\mathcal{Y}} -\mathbf{w}^T\mathbf{\Psi}(\mathbf{x},\mathbf{y})$. The resulting optimization problem is identical to the original inference problem in Equation 1, as the minus at the RHS can be absorbed into the learned weights. By defining an appropriate loss function, such as the Hamming loss $\Delta(\mathbf{y},\mathbf{y}') \equiv \sum_j \mathbb{I}(\mathbf{y}_j \neq \mathbf{y}'_j)$, it turns out that both Eq. 1 and Eq.3 can be interpreted as MAX-SMT problems. This observation enables us to use a MAX-SMT solver to implement the two oracles required by the CP algorithm, and thus to efficiently solve the learning task. Note also that *hard* constraints, i.e. formulae with infinite weight, can also be included in the SMT problem.

The above definition of per-formula *boolean cost* ψ_f is only the simplest option. A more refined alternative, applicable to formulae with only numerical variables, is to employ a *linear cost* of the assignment $\mathbf{z}$ and the constants appearing in the formula f, as follows:

$$\psi_{y<c}(y) \equiv \max(y-c,0) \qquad \psi_{y>c}(y) \equiv \max(c-y,0) \qquad \psi_{y=c} \equiv |y-c|$$

For instance, given $f = x + y < 5$ and $\mathbf{z} = (x,y) = (4,3)$, the amount of violation would be $\psi_f(\mathbf{z}) = \max((x+y)-5,0) = 2$, while for $\mathbf{z} = (1,5)$ the cost would be 0 (since f is satisfied). Applying linear costs has two consequences. First, they allow to enrich the feature space with information about the *amount* of violation of any linear formula f: an unsatisfied formula contributes $w_j \cdot \psi_{f_j}(\mathbf{z})$ to $\mathbf{\Psi}_F$. Second, since the cost of unsatisfied constraints depends on the value of the numerical variables involved, the resulting inference and separation oracles can not be solved using MAX-SMT, but require a full-fledged OMT solver. More complex cost functions can be developed for mixed boolean-numerical formulae (consider e.g. $(y > 10) \Rightarrow (a \vee b)$), for instance by summing the violations of the individual clauses. One issue with this formulation is that, since the cost of continuous clauses is unbounded, inference may have a bias towards satisfying continuous clauses rather than the Boolean ones; this problem however is shared by all hybrid satisfaction-based models.

4 Applications

There are a number of applications involving both Boolean and numerical constraints, such as environment learning for robot planning [4] and the modeling of gene expression data [14]. Here we describe two of them, to illustrate the flexibility and expressive power of LMT. We postpone a formal definition of these problems to a future publication, due to space restrictions.

Activity recognition [15] is the problem of determining which human activities $\mathbf{y}_t$ have produced a given set of sensor observations $\mathbf{x}_t$ at each time instant t. Here the *activities* are understood to be common everyday tasks such as "having breakfast", "watching TV" or "taking a shower". The observations are taken from sensors deployed in a smart environment (e.g. an instrumented home/hospital), and may include different sensory channels such as video, audio, the agent's position, posture, heartbeat, *etc.* Activity recognition is typically cast as a *tagging* problem in discrete time, and tackled by means of probabilistic temporal models. In real-world scenarios the activities are often concurrent and inter-related, in which case Factorial versions of Hidden Markov Models or Conditional Random Fields are used. Unfortunately, training these models is intractable. With LMT we take a rather different route, and cast activity recognition as a form of data-driven *scheduling* in continuous time. Allen's *interval temporal logic* (ITL) [16] is an intuitive formal language to express relations between temporal events. ITL provides primitives such as $\texttt{before}(a,b)$, $\texttt{after}(a,b)$, $\texttt{overlaps}(a,b)$, $\texttt{during}(a,b)$, $\texttt{equal}(a,b)$. These predicates can be straightforwardly translated to linear arithmetic constraints, and therefore easily implemented in LMT. The combination of ITL and FOL allows to express concurrent, interdependent, nested and hierarchical activities, and to specify the likely duration of activities and intervals between them. Consider for instance constraints such as "breakfast occurs within an hour after waking up", and "cooking a dish involves interacting with at least three ingredients, in a specific order". Using similar constraints, LMT would be able to generate a scheduling of the activities that is consistent with respect to the observations and with the (soft) constraints.

Another interesting application is the *housing problem* [17], which is just one instance of a class of weighted constraint satisfaction problems that routinely occur in logistics. Consider a customer

planning to build her own house and judging potential housing locations provided by a real estate company. There are different locations available, characterized by different housing values, prices, constraints about the design of the building (e.g a minimum distance to other buildings), *etc.* A description of the customer preferences and requirements may be given in SMT, in order to express them with both Boolean and numerical constraints, e.g., the crime rate, distance from downtown, location-based taxes, public transit service quality, maxiumum walking or cycling distances to the closest facilities. The underlying optimization problem is clearly an instance of MAX-SMT, and LMT can be used to efficiently learn the formula weights from user-provided data. We have already developed a MAX-SMT-based prototype to solve the housing problem in an active learning setting, by using an interactive preference elicitation mechanism to learn the relative importance of the various constraints for the customer which has shown encouraging results [17].

References

[1] Lise Getoor and Ben Taskar. *Introduction to statistical relational learning*. The MIT press, 2007.

[2] Clark W Barrett, Roberto Sebastiani, Sanjit A Seshia, and Cesare Tinelli. Satisfiability modulo theories. *Handbook of satisfiability*, 185:825–885, 2009.

[3] Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. A modular approach to maxsat modulo theories.

[4] Jue Wang and Pedro Domingos. Hybrid markov logic networks. In *AAAI*, volume 8, pages 1106–1111, 2008.

[5] Bernd Gutmann, Manfred Jaeger, and Luc De Raedt. Extending problog with continuous distributions. In *Inductive Logic Programming*, pages 76–91. Springer, 2011.

[6] Muhammad Asiful Islam, CR Ramakrishnan, and IV Ramakrishnan. Parameter learning in prism programs with continuous random variables. *arXiv preprint arXiv:1203.4287*, 2012.

[7] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pages 1453–1484, 2005.

[8] Roberto Sebastiani and Silvia Tomasi. Optimization in smt with $\mathcal{LA}(\mathbb{Q})$ cost functions. In *Automated Reasoning*, pages 484–498. Springer, 2012.

[9] Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. The mathsat5 smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 93–107. Springer, 2013.

[10] Bruno Dutertre and Leonardo De Moura. The yices smt solver. *Tool paper at http://yices. csl. sri. com/tool-paper. pdf*, 2:2, 2006.

[11] Miquel Bofill, Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell, and Albert Rubio. The barcelogic smt solver. In *Computer Aided Verification*, pages 294–298. Springer, 2008.

[12] Boyan Yordanov, Christoph M Wintersteiger, Youssef Hamadi, Andrew Phillips, and Hillel Kugler. Functional analysis of large-scale dna strand displacement circuits. In *DNA Computing and Molecular Programming*, pages 189–203. Springer, 2013.

[13] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[14] Ondřej Kuželka, Andrea Szabóová, Matěj Holec, and Filip Železnỳ. Gaussian logic for predictive classification. In *Machine Learning and Knowledge Discovery in Databases*, pages 277–292. Springer, 2011.

[15] Tim Van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 1–9. ACM, 2008.

[16] James F Allen and George Ferguson. Actions and events in interval temporal logic. *Journal of logic and computation*, 4(5):531–579, 1994.

[17] Paolo Campigotto, Andrea Passerini, and Roberto Battiti. Active learning of combinatorial features for interactive optimization. In *Learning and Intelligent Optimization*, pages 336–350. Springer, 2011.