Inducing Sparse Programs for Learning Modulo Theories

Stefano Teso TESO@DISI.UNITN.IT
DISI, University of Trento, Italy

Andrea Passerini PASSERINI@DISI.UNITN.IT
DISI, University of Trento, Italy

Abstract

The ability to learn hybrid Boolean-numerical concepts is crucial in "learning to design" tasks, that is, learning applications where the goal is to learn from examples how to perform automatic de novo design of novel objects. Recently Learning Modulo Theories, an extension of Structured Output SVMs leveraging state-of-the-art logical-numerical optimization techniques, has been proposed as a viable approach to performing inference and parameter learning in this setting. Like other statistical-relational learning methods, LMT presupposes the availability of a set of *constraints* that act as feature functions over the objects. Designing such constraints by hand requires a great deal of domain knowledge and may not always be practical. In this paper we tackle the challenging problem of automatically inducing the constraints from data. We cast constraint learning as a (non-trivial) sparse ranking problem, and sketch an approximate solution strategy using convex programming.

1. Introduction

Constructive learning encompasses a number of "learning to design" tasks, i.e. applications where the goal is to learn from examples how to perform automatic design of novel objects. In many cases such tasks involve hybrid structured objects composed by a mixture of Boolean and numerical variables. Prototypical examples include automated/interactive layout synthesis (Yang et al., 2013; Hausner, 2001), where the task is to find an optimal 2D layout constrained by logical and spatial relations between building blocks; procedural generation of game content and automated level design (Hendrikx et al., 2013); and many synthetic biology problems such as automated design and optimization of chemical reaction networks (Fagerberg et al., 2012; Yordanov et al., 2013).

Researchers in automated reasoning and formal verification have developed logical languages and reasoning tools that allow for *natively* reasoning over mixtures of Boolean *and* numerical variables (or even more complex structures). These languages are grouped under the umbrella term of *Satisfiability Modulo Theories* (SMT) (Barrett et al., 2009). Each such language corresponds to a decidable fragment of First-Order Logic augmented with an additional background theory $\mathcal{T}$, like linear arithmetic over the rationals $\mathcal{LRA}$ or over the integers $\mathcal{LIA}$. SMT is a decision problem, which consists in finding an assignment to both Boolean and theory-specific variables making an SMT formula true. Recently, researchers have leveraged SMT from decision to optimization. The most general framework is that of *Optimization Modulo Theories* (*OMT*) (Sebastiani & Tomasi, 2015), which consists in finding a *model* for a formula which minimizes the value of some (arithmetical) cost function defined over the variables in the formula.

We recently proposed *Learning Modulo Theories* (LMT) as a viable approach to learning in hybrid constructive problems (Teso et al., 2014). LMT is an instance of Structured-output Support Vector Machines (Tsochantaridis et al., 2005) that operates directly over combinations of Boolean and rational variables. The OMT machinery is employed by LMT to enable efficient inference and parameter learning.

In this paper we consider the problem of learning the *structure* of LMT problems from data. We formulate the corresponding optimization problem as a non-trivial sparse ranking problem, and outline both exact and heuristic learning strategies.

Preliminary work. Under review by the Constructive Machine Learning workshop @ ICML 2015. Do not distribute.

Symbol	Meaning
$\boldsymbol{x} := (\boldsymbol{x}^B, \boldsymbol{x}^C)$	Complete object
$\boldsymbol{x}^B \in \{\top, \bot\}^\ell$	Boolean variables
$\boldsymbol{x}^C \in \mathbb{Q}^m$	Rational variables
$\phi_k := x_k^B$	Boolean atomic constraints
$\phi_k := \boldsymbol{a}_k^\top \boldsymbol{x}^C + b_k$	Rational atomic constraints
$\{C_i, C_{ij}, C_{ijk}\}$	Term constraints
$\boldsymbol{\psi}(\boldsymbol{x})$	Feature representation of $\boldsymbol{x}$
$A, \boldsymbol{b}$	Learned atomic hyperplanes
$\boldsymbol{w}$	Learned term weights

Table 1. Explanation of the notation used throughout the text.

2. Background on LMT

In the LMT setting each object $\boldsymbol{x} := (\boldsymbol{x}^B, \boldsymbol{x}^C)$ is encoded as a vector of Boolean and rational variables:

$$\boldsymbol{x}^B \in \{\top, \bot\}^\ell \qquad \boldsymbol{x}^C \in \mathbb{Q}^m$$

The Boolean variables encode the truth value of predicates while the rational variables represent the continuous components of the object. The feature representation of an object is determined by a finite set of (soft) *constraints* $\{\varphi_i\}$, each constraint φ_i being either a Boolean- or linear algebraic formula on the variables of the object $\boldsymbol{x}$. An example constraint may look like:

$$\texttt{fat}(\boldsymbol{x}) \iff ((width(\boldsymbol{x}) > 2 \times height(\boldsymbol{x})) \lor (weight(\boldsymbol{x}) > 200))$$

These constraints are typically constructed using the background knowledge available for the domain. For each Boolean-valued constraint φ_i, we denote its *indicator function* as $\mathbb{1}_i(\boldsymbol{x})$, which evaluates to 1 if the constraint is satisfied and to 0 otherwise. Similarly, we refer to the *indicator function* of a rational-valued constraint φ_i as $c_i(\boldsymbol{x}) \in \mathbb{Q}$: more specifically, the costs c_i are linear functions of the rational variables $\boldsymbol{x}^C$. The feature vector $\boldsymbol{\psi}(\boldsymbol{x})$ is obtained by concatenating indicator and cost functions of Boolean and rational constraints respectively.

The LMT score associated to an object $\boldsymbol{x}$ is defined as a linear function of the feature representation of $\boldsymbol{x}$, $f(\boldsymbol{x}) := \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x})$; the weight vector $\boldsymbol{w}$ is to be learned. Given a partially observed object $\boldsymbol{x} = (\boldsymbol{I}; \boldsymbol{O})$, where the variables in $\boldsymbol{I}$ are observed and those in $\boldsymbol{O}$ are not, inference amounts to finding the value of $\boldsymbol{O}$ that maximizes the total score: $\boldsymbol{O}^* := \mathrm{argmax}_{\boldsymbol{O}}\, \boldsymbol{w}^\top \boldsymbol{\psi}((\boldsymbol{I}; \boldsymbol{O}))$. Parameter learning is formulated in a max-margin structured output setting (Tsochantaridis et al., 2005) and solved approximately using a Cutting Plane (CP) algorithm (Joachims et al., 2009). The sub-problems generated by the CP procedure can be cast as optimization modulo $\mathcal{LRA}$ problems and solved with an appropriate tool (we use the OPTIMATHSAT solver[1]). Please see (Teso et al., 2014) for more details.

3. Inducing Sparse LMT Programs

Our aim is to automatically induce the set of Boolean and linear algebraic constraints $\{\varphi_i\}$ from a collection of positive and negative labelled objects. In this paper we focus on inducing weighted MAX-SMT programs, i.e. programs where the feature function $\boldsymbol{\psi}$ includes only indicator terms. This formulation is general enough to capture a number of interesting hybrid constructive problems. The assumption underlying our method is that although the search space may include arbitrarily complex programs, we seek to learn a *sparse* program: only few variables and constraints are relevant for discriminating between good and bad quality objects.

We assume an upper bound n on the number of linear inequalities in the problem[2], each represented as $\boldsymbol{a}_i^\top \boldsymbol{x}^C + b_i \geq 0$, $i = 1, \ldots, n$, where $\boldsymbol{x}^C$ is the rational part of the object $\boldsymbol{x}$. The Boolean atomic constraints are defined as $\phi_k \iff x_k^B$, $k = 1, \ldots, \ell$, while the atomic constraints on the rational variables are $\phi_k \iff \boldsymbol{a}_k^\top \boldsymbol{x}^C + b_k \geq 0$, $k = \ell + 1, \ldots, \ell + n$.

Terms are formed by conjunctions of up to d atomic constraints, including negations, e.g. for $d = 3$:

$$\begin{aligned} &\forall i, j, k \in [1, 2(\ell + n)] \\ &\qquad C_i \iff \phi_i \\ &\qquad C_{i,j} \iff \phi_i \land \phi_j \\ &\qquad C_{i,j,k} \iff \phi_i \land \phi_j \land \phi_k \end{aligned}$$

where if $i > \ell + n$ then $\phi_i = \neg \phi_{i-\ell-n}$. The choice of d affects the degree of "non-linearity" of the learned program, and therefore the difficulty of the learning problem[3] The feature function $\boldsymbol{\psi}$ is the concatenation of all term indicators:

$$\boldsymbol{\psi}(\boldsymbol{x}) := (\mathbb{1}(C_1), \ldots, \mathbb{1}(C_{2(\ell+n),2(\ell+n),2(\ell+n)})) \qquad (1)$$

The corresponding LMT score function can be seen as a "soft" DNF over the terms $\{C_i\} \cup \{C_{i,j}\} \cup \{C_{i,j,k}\}$ and weights $\boldsymbol{w}$. The role of the Boolean variables $\boldsymbol{x}^B$ is that of selecting the subsets of term constraints that are enabled/disabled for the example $\boldsymbol{x}$. A summary of the notation introduced so far can be found in Table 1.

[1] http://optimathsat.disi.unitn.it/

[2] An iterative approach progressively increasing this number until a satisfactory solution is found could be conceived.

[3] As for the number of linear inequalities n, d could be increased in an iterative fashion to learn progressively more complex programs.

Now let the matrix A be the result of vertically stacking the rows $\boldsymbol{a}_i^\top$, and $\boldsymbol{b} := (b_1, \ldots, b_n)$. Our goal is to learn an LMT problem that constructs good quality (positive) objects, so we require each positive object to score higher than any negative one. This insight can be formalized as:

$$\begin{aligned} \min_{\boldsymbol{w}, A, \boldsymbol{b}} \quad & \|\boldsymbol{w}\|_1 + \lambda \|A\|_\star \\ \text{s.t.} \quad & \forall \boldsymbol{x} \in \text{pos}, \boldsymbol{x}' \in \text{neg} \\ & \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}) > \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}') + \Delta(\boldsymbol{x}, \boldsymbol{x}') \end{aligned} \tag{2}$$

where the constraints impose that objects are ranked correctly and Δ is a structured loss quantifying the dissimilarity between $\boldsymbol{x}$ and $\boldsymbol{x}'$, such as the Hamming loss in instance space:

$$\Delta(\boldsymbol{x}, \boldsymbol{x}') := \sum_{i=1}^{\ell} \mathbb{1}(\boldsymbol{x}_i^B \neq \boldsymbol{x}_i'^B) + \sum_{i=1}^{m} |\boldsymbol{x}_i^C - \boldsymbol{x}_i'^C|$$

The ℓ_1 regularization on $\boldsymbol{w}$ limits the number of active term constraints (i.e. with non-zero weight), while the $\|\cdot\|_\star$ norm on A is chosen as to encourage learning sparse decision hyperplanes (see next section for details).

A naive method to solve the above minimization problem is to encode it as an OMT($\mathcal{LRA}$) problem and solve it accordingly. OMT solvers however focus on finding *globally optimal* solutions, which is intractable for general LMT structure learning problems. In the next section we sketch a tractable approximate solution technique.

4. An Approximate Solution Strategy

We approximate the problem by decoupling the optimization of $\boldsymbol{w}$ and $(A, \boldsymbol{b})$. In the first step we attempt to determine a *sparse* set of *non-redundant* hyperplanes that separate positive from negative objects. This can be done by solving the following *convex* problem:

$$\begin{aligned} \min_{A, \boldsymbol{b}} \quad & \lambda_2 \|A\|_{1,2} + \lambda_1 \|A\|_{1,1} \\ \text{s.t.} \quad & \forall \boldsymbol{x} \in \text{pos} \;\; A\boldsymbol{x} + \boldsymbol{b} \geq \boldsymbol{0} \\ & \forall \boldsymbol{x} \in \text{pos}, \boldsymbol{x}' \in \text{neg} \\ & \sum_i (A\boldsymbol{x} + \boldsymbol{b})_i > \sum_i (A\boldsymbol{x}' + \boldsymbol{b})_i \end{aligned} \tag{3}$$

The group lasso $\|A\|_{1,2} = \sum_{i=1}^n ||\boldsymbol{a}_i||_2$ encourages sparsity over the *rows* of A, i.e. over the *set* of atoms/hyperplanes, and $\|A\|_{1,1} = \sum_i \sum_j |a_{ij}|$ encourages sparsity of the individual weights. This regularization functional is called *sparse group lasso* (Friedman et al., 2010), and encourages the hyperplanes to be non-redundant (at the atomic constraint level). Learning $\boldsymbol{a}_i = \boldsymbol{0}$ essentially discards the ith atomic constraint.

The hyperplanes determined in the first step can be used to precompute the feature representation $\boldsymbol{\psi}(\boldsymbol{x})$ (see Eq. 1) for all examples $\boldsymbol{x}$. Finding $\boldsymbol{w}$ equates to solving the following linear problem:

$$\begin{aligned} \min_{w} \quad & \|w\|_1 \\ \text{s.t.} \quad & \forall \boldsymbol{x} \in \text{pos}, \boldsymbol{x}' \in \text{neg} \\ & \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}) > \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}') + \Delta(\boldsymbol{x}, \boldsymbol{x}') \end{aligned}$$

where the number of constraints is quadratic in the number of examples.

Note that the first step only deals with linear combinations of atomic constraints, which are non-linearly combined in the second step only. As a consequence, this approach fails to identify atomic constraints having an exclusively non-linear role. In order to lift this limitation we plan to learn hyperplanes whose *combination* is discriminative with respect to the objects, e.g. by replacing the comparison in Eq 3 with a non-linear one such as:

$$\begin{aligned} & \forall \boldsymbol{x} \in \text{pos} \;\; \boldsymbol{f}(A\boldsymbol{x} + \boldsymbol{b}) \geq \boldsymbol{0} \\ & \forall \boldsymbol{x} \in \text{pos}, \boldsymbol{x}' \in \text{neg} \\ & \sum_i (\boldsymbol{f}(A\boldsymbol{x} + \boldsymbol{b}))_i > \sum_i (\boldsymbol{f}(A\boldsymbol{x}' + \boldsymbol{b}))_i \end{aligned}$$

where $\boldsymbol{f}$ is an inhomogeneous (multi-valued) polynomial. The resulting mathematical problem however is much harder to handle.

The quality of the solution depends crucially on the quality of the hyperplanes found in the first step. While intuitively looking for diverse hyperplanes should provide a good starting point, we currently have no guarantees on the quality of said hyperplanes with respect to the original objective function (Eq. 2). We leave the required detailed analysis to future work.

5. Related Work

Structure learning of statistical-relational models is frequently cast to Inductive Logic Programming (see e.g. (Huynh & Mooney, 2008)) followed by a separate weight learning stage; or solved via custom methods. Purely logical approaches however are not directly applicable to hybrid Boolean-continous problems. As a matter of fact, little attention has been given to learning the structure of hybrid statistical-relational models. To the best of our knowledge, the only study tackling this challenging problem is (Ravkic et al., 2015). Ravkic and colleagues propose a technique for learning the structure of hybrid relational dependency networks. Assuming that the candidate structure scoring function is decomposable, the procedure of (Ravkic et al., 2015) allows to learn both the dependency structure and the conditional probability table for each predicate in the network using a clever decomposition technique. However, unlike LMT, hybrid relational dependency networks do not allow to express linear arithmetical constraints over the continuous variables.

Related decoupling strategies have been proposed for learning the structure of Bayesian Networks. Under the condition that the structure scoring function decomposes over the parent sets of the network, structure learning can be cast as an (exponentially large but extremely sparse) constrained integer linear programming instance (Cussens & Bartlett, 2013), where the constraints ensure that the learned structure is acyclic. Efficient optimization schemas for this kind of problem have been devised, see (Cussens & Bartlett, 2013) for a review. These techniques however do not support parameter and structure learning *jointly*, as is the case in our problem.

Another related line of research is that of global parameter learning for linear decision trees, i.e. decision trees with linear classifiers at each node. Such DTs can be seen as a simple instance of SMT program in disjunctive normal form. In (Bennett, 1994) Bennet proposes a global (non-greedy) parameter learning technique for such models. Bennet defines the total loss as a polynomial of local hinge losses over individual nodes, which turns parameter learning into a non-convex multilinear program. The technique assumes the structure of the DT to be known in advance. While it could in principle used as a sub-procedure within an iterative LMT constraint learner, it is not immediately clear how it would compare in terms of solution quality and runtime against our proposed approximate procedure.

6. Conclusions

In this paper we presented a formulation of LMT constraint learning for the case of MAX-SMT programs by casting it as a sparse ranking problem. We also outlined a decoupled greedy procedure that allows to find approximate solutions, and highlighted some methods that may allow to alleviate the greediness of the current approximate strategy. Further work on the subject involves of course validating the proposed approach on real-world data, and evaluating alternative approximate solution strategies.

References

Barrett, C., Sebastiani, R., Seshia, S. A., and Tinelli, C. *Satisfiability Modulo Theories*, chapter 26, pp. 825–885. Frontiers in Artificial Intelligence and Applications. IOS Press, February 2009. ISBN 978-1-58603-929-5.

Bennett, Kristin P. Global tree optimization: A non-greedy decision tree algorithm. *Computing Science and Statistics*, pp. 156–156, 1994.

Cussens, James and Bartlett, Mark. Advances in bayesian network learning using integer programming. *arXiv preprint arXiv:1309.6825*, 2013.

Fagerberg, R., Flamm, C., Merkle, D., and Peters, P. Exploring chemistry using smt. In *CP'12*, pp. 900–915, 2012.

Friedman, J., Hastie, T., and Tibshirani, R. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

Hausner, Alejo. Simulating decorative mosaics. In *SIGGRAPH '01*, pp. 573–580, 2001.

Hendrikx, Mark, Meijer, Sebastiaan, Van Der Velden, Joeri, and Iosup, Alexandru. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 9(1):1, 2013.

Huynh, Tuyen N and Mooney, Raymond J. Discriminative structure and parameter learning for markov logic networks. In *Proceedings of the 25th international conference on Machine learning*, pp. 416–423. ACM, 2008.

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

Ravkic, Irma, Ramon, Jan, and Davis, Jesse. Learning relational dependency networks in hybrid domains. *Machine Learning*, pp. 1–38, 2015.

Sebastiani, Roberto and Tomasi, Silvia. Optimization Modulo Theories with Linear Rational Costs. *ACM Transactions on Computational Logics*, 16, 2015.

Teso, Stefano, Sebastiani, Roberto, and Passerini, Andrea. Structured learning modulo theories. *To appear in Artificial Intelligence Journal*, 2014. URL `http://arxiv.org/abs/1405.1675`.

Tsochantaridis, Ioannis, Joachims, Thorsten, Hofmann, Thomas, and Altun, Yasemin. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, December 2005.

Yang, Yong-Liang, Wang, Jun, Vouga, Etienne, and Wonka, Peter. Urban pattern: Layout design by hierarchical domain splitting. *ACM Trans. Graph.*, 32(6):181:1–181:12, November 2013.

Yordanov, Boyan, Wintersteiger, Christoph M., Hamadi, Youssef, and Kugler, Hillel. Smt-based analysis of biological computation. In *NASA Formal Methods Symposium 2013*, pp. 78–92, May 2013.