Constructive Learning Modulo Theories

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

Roberto Sebastiani ROBERTO.SEBASTIANI@UNITN.IT
DISI, University of Trento, Italy

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

Abstract

1. Introduction

Modelling problems containing a mixture of Boolean and numerical variables is a long-standing interest of Artificial Intelligence. However, performing inference and learning in hybrid domains is a particularly daunting task. The ability to model these kind of domains 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. Apart from hybrid Bayesian Networks, for which efficient inference is limited to conditional Gaussian distributions, there is relatively little previous work on *hybrid* methods. The few existing attempts (Goodman et al., 2008; Wang & Domingos, 2008; Närman et al., 2010; Gutmann et al., 2011; Choi & Amir, 2012; Islam et al., 2012) impose strong limitations on the type of constraints they can handle. Inference is typically run by approximate methods, based on variational approximations or sampling strategies. Exact inference, support for hard numeric (in addition to Boolean) constraints and combination of diverse theories are out of the scope of these approaches. In order to overcome these limitations, we focused on the most recent advances in automated reasoning over hybrid domains. 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.

In this paper we present Learning Modulo Theories (LMT), a max-margin approach for learning in hybrid domains based on Satisfiability Modulo Theories, which allows to combine Boolean reasoning and optimization over continuous linear arithmetical constraints. The main idea is to combine the discriminative power of structured-output SVMs (Tsochantaridis et al., 2005) with the reasoning capabilities of SMT technology. Training structured-output SVMs requires a separation oracle for generating counterexamples and updating the parameters, while an inference oracle is required at prediction stage to generate the highest scoring candidate structure for a certain input. Both tasks can be accomplished by a generalized Satisfiability Modulo Theory solver. We show the potential of LMT on an artificial layout synthesis scenario.

2. LMT as structured-output learning

Structured-output SVMs (Tsochantaridis et al., 2005) generalize max-margin methods to the prediction of output structures by learning a scoring function over joint input-output pairs $f(\boldsymbol{I}, \boldsymbol{O}) = \boldsymbol{w}^T \boldsymbol{\psi}(\boldsymbol{I}, \boldsymbol{O})$. Given an input $\boldsymbol{I}$, the predicted output will be the one maximizing the scoring function:

$$\boldsymbol{O}^* = \operatorname*{argmax}_{\boldsymbol{O}} f(\boldsymbol{I}, \boldsymbol{O}) \tag{1}$$

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

and the problem boils down to finding efficient procedures for computing the maximum. Efficient exact procedures exist for some special cases. In this paper we are interested in the case where inputs and ouputs are combinations of discrete and continuous variables, and we'll leverage SMT techinques to perform efficient inference in this setting. Max-margin learning is performed by enforcing a margin separation between the score of the correct structure and that of any possible incorrect structure, possibly accounting for margin violations to be penalized in the objective function. A cutting plane (CP) algorithm allows to deal with the exponential number of candidate structures by iteratively adding the most violated constraint for each example pair given the current scoring function, and refining it by solving the augmented quadratic problem with a standard SVM solver. 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 $\boldsymbol{\psi}$ of input-output pairs (and consequently a scoring function f); (ii) an oracle to perform inference, i.e. to solve Equation (1); and (iii) an oracle to retrieve the most violated constraint of the QP, i.e. to solve the *separation* problem:

$$\underset{\boldsymbol{O}'}{\operatorname{argmax}}\, \boldsymbol{w}^T \boldsymbol{\psi}(\boldsymbol{I}_i, \boldsymbol{O}') + \Delta(\boldsymbol{I}_i, \boldsymbol{O}_i, \boldsymbol{O}') \qquad (2)$$

where $\Delta(\boldsymbol{I}_i, \boldsymbol{O}_i, \boldsymbol{O}')$ is a *loss function* quantifying the penalty incurred when predicting $\boldsymbol{O}'$ instead of the correct output $\boldsymbol{O}$. In the following we will detail how these components are provided within the LMT framework.

input-output feature map Recall that in our setting each object $(\boldsymbol{I}, \boldsymbol{O})$ is represented as a set of Boolean and rational variables:

$$(\boldsymbol{I}, \boldsymbol{O}) \in \underbrace{(\{\top, \bot\} \times \ldots \times \{\top, \bot\})}_{\text{Boolean part}} \times \underbrace{(\mathbb{Q} \times \ldots \times \mathbb{Q})}_{\text{rational part}}$$

We indicate Boolean variables using predicates such as $\texttt{touching}(i, j)$, and write rational variables as lower-case letters, e.g. $cost$, $distance$, x, y. Features are represented in terms of (soft) *constraints* $\{\varphi_k\}_{k=1}^m$, each constraint φ_k being either a Boolean- or rational-valued function of the object $(\boldsymbol{I}, \boldsymbol{O})$. These constraints are constructed using the background knowledge available for the domain. For each Boolean-valued constraint φ_k, we denote its *indicator function* as $\mathbb{1}_k(\boldsymbol{I}, \boldsymbol{O})$, which evaluates to 1 if the constraint is satisfied and to -1 otherwise. Similarly, we refer to the *cost* of a rational-valued constraint φ_k as $c_k(\boldsymbol{I}, \boldsymbol{O}) \in \mathbb{Q}$. The feature vector $\boldsymbol{\psi}(\boldsymbol{I}, \boldsymbol{O})$ is obtained by concatenating indicator and cost functions of Boolean and rational constraints respectively.

inference oracle Given the feature vector $\boldsymbol{\psi}(\boldsymbol{I}, \boldsymbol{O})$, the scoring function $f(\boldsymbol{I}, \boldsymbol{O})$ is a linear combination of indicator and cost functions. Since $\boldsymbol{\psi}$ can be expressed in terms of SMT($\mathcal{LRA}$) formulas, the resulting maximization problem can be readily cast as an OMT($\mathcal{LRA}$) problem and inference is performed by an OMT-solver (we use the OPTIMATHSAT solver [1]). Given that OMT-solvers are conceived to *minimize* cost functions rather than maximize scores, we actually run it on the negated scoring function.

separation oracle The separation problem amounts at maximizing the sum of the scoring function and a *loss* function over output pairs (see eq.(2)). We observe that by picking a loss function expressible as an OMT($\mathcal{LRA}$) problem, we can readily use the same OMT solver used for inference to also solve the separation oracle. This can be achieved by selecting a loss function such as the Hamming loss in feature space:

$$\Delta(\boldsymbol{I}, \boldsymbol{O}, \boldsymbol{O}') := \|\boldsymbol{\psi}(\boldsymbol{I}, \boldsymbol{O}) - \boldsymbol{\psi}(\boldsymbol{I}, \boldsymbol{O}')\|_1$$

This loss function is piecewise-linear, and as such satisfies the desideratum. While this is the loss which was used in all experiments, LMT can work with any loss function that can be encoded as an SMT formula.

3. A Stairway to Heaven

We show the potential of the LMT framework on a toy constructive problem which consists in learning how to assemble different kinds of *stairways* from examples. A stairway is made up of a collection of m blocks (rectangles) in a 2D unit-sized bounding box. Figure 1 shows examples of different types of stairways (a-c), varying in terms of *orientation* and *proportions* between block heights and widths, and a configuration which is not a stairway (d).

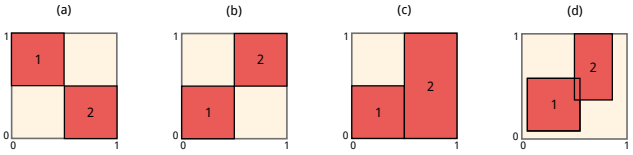

Figure 1. (a) A *left ladder* 2-stairway. (b) A *right ladder* 2-stairway. (c) A *right pillars* 2-stairway. (d) not a stairway.

Each block is identified by a tuple (x, y, dx, dy), consisting of bottom-left coordinates, width and height. An output $\boldsymbol{O}$ is given by the set of tuples identifying the blocks. Input is assumed to be empty in the following, but non-empty inputs can used to model for instance partially observed scenes.

Learning amounts to assigning appropriate weights to a set of soft-constraints in order to maximize the score of stair-

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

ways of the required type, with respect to non-stairways and stairways of other types. Inference amounts to generating configurations (i.e. variables assignments to all blocks) with maximal score, and can be easily encoded as an OMT($\mathcal{LRA}$) problem. As a first step, we define a set of hard rules to constrain the space of admissible block assignments. We require that all blocks fall within the bounding box, and that blocks do not overlap:

$$\begin{aligned} \forall i \quad & 0 \leq x_i, dx_i, y_i, dy_i \leq 1 \\ \forall i \quad & 0 \leq (x_i + dx_i) \leq 1 \ \wedge \ 0 \leq (y_i + dy_i) \leq 1 \\ \forall i \neq j \quad & (x_i + dx_i \leq x_j) \vee (x_j + dx_j \leq x_i) \vee \\ & (y_i + dy_i \leq y_j) \vee (y_j + dy_j \leq y_i) \end{aligned}$$

Furthermore, we require (without loss of generality) blocks to be ordered from left to right, $\forall i \ x_i \leq x_{i+1}$. Note that stairway conditions (apart from non-overlap between blocks) are not included in these hard rules and will be implicitly modelled as soft constraints. To this aim we introduce a set of useful predicates. We use four predicates to encode the fact that a block i may touch one of the four corners of the bounding box, e.g.:

$$\texttt{bottom\_right}(i) := (x_i + dx_i) = 1 \wedge y_i = 0$$

We also define predicates to describe the relative positions of two blocks i and j, such as $\texttt{left}(i, j)$:

$$\begin{aligned} \texttt{left}(i,j) := \ & (x_i + dx_i) = x_j \wedge ((y_j \leq y_i \leq y_j + dy_j) \vee \\ & (y_j \leq y_i + dy_i \leq y_j + dy_j)) \end{aligned}$$

that encodes the fact that block i is touching block j from the left. Similarly, we also define $\texttt{below}(i, j)$ and $\texttt{over}(i, j)$. Finally, we combine the above predicates to define the concept of *step*:

$$\begin{aligned} \texttt{left\_step}(i,j) := & (\texttt{left}(i,j) \wedge (y_i + dy_i) > (y_j + dy_j)) \vee \\ & (\texttt{over}(i,j) \wedge (x_i + dx_i) < (x_j + dx_j)) \end{aligned}$$

We define right_step(i,j) in the same manner. This background knowledge allows to encode the property of being a stairway of a certain type as a conjunction of predicates. However, our inference procedure does not have access to this knowledge. We rather encode an appropriate set of *soft rules* (costs) which, along with the associated weights, should bias the optimization towards block assignments that form a stairway of the correct type. Our cost model is based on the observation that it is possible to discriminate between the different stairway types using only four factors: minimum and maximum step size, and amount of horizontal and vertical *material*. For instance, in the cost we account for both the maximum step height of all left steps (a good stairway should not have too high steps) and the minimum step width of all right steps (good stairways should have sufficiently large steps):

$$maxshl = m \times \max_{i \in [1, m-1]} \begin{cases} (y_i + dy_i) - (y_{i+1} + dy_{i+1}) \\ \quad \text{if } i, i+1 \text{ form a left step} \\ 1 \quad \text{otherwise} \end{cases}$$

$$minswr = m \times \min_{i \in [1, m-1]} \begin{cases} (x_{i+1} + dx_{i+1}) - (x_i + dx_i) \\ \quad \text{if } i, i+1 \text{ form a right step} \\ 0 \quad \text{otherwise} \end{cases}$$

The value of these costs depends on whether a pair of blocks actually forms a left step, a right step, or no step at all. Finally, we write the average amount of vertical material as $vmat = \frac{1}{m} \sum_i dy_i$. All the other costs can be written similarly. Putting all the pieces together, the complete cost is:

$$\begin{aligned} cost \ := \ & (maxshl, minshl, maxshr, minshr, \\ & maxswl, minswl, maxswr, minswr, \\ & vmat, hmat) \ \boldsymbol{w} \end{aligned}$$

The actual weights are learned, allowing the learnt model to reproduce whichever stairway type is present in the training data.

To test the stairway scenario, we generated "perfect" stairways of 2 to 6 blocks for each stairway type to be used as training instances. We then learned a model using all training instances up to a fixed number of blocks (e.g from 2 to 4) and asked the learnt models to generate configurations with a *larger* number of blocks (up to 10) than those in the training set. The results can be found in Figure 2.

The experiment shows that LMT is able to solve the stairway construction problem, and can learn appropriate models for all stairway types, as expected. Some imperfections can be seen when the training set is too small (e.g., only two training examples; first row of each table), but already with four training examples the model is able to generate perfect 10-block stairways of the given type, for all types.

4. Conclusions

Albeit simple, the stairway application showcases the ability of LMT to handle learning in hybrid Boolean-numerical domains characterized by complex combinations of hard and soft constraints, whereas other formalisms are not suited for the task[2].The application is a simple instance

[2]The stairway problem can be easily encoded in the Chuch probabilistic programming language. However, the sampling strategies used for inference are not conceived for performing optimization with hard continuous constraints. Even the simple task of generating two blocks, conditioned on the fact that they form a step, is prohibitively expensive. A more comprehensive discussion on alternative approaches and their limitations can be found in the journal version of this work.

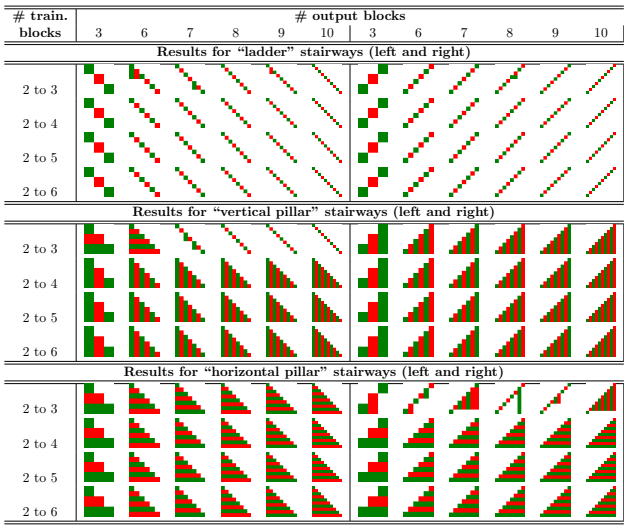

Figure 2. Results for the stairway construction problem. From top to bottom: results for the *ladder*, *horizontal pillar*, and *vertical pillar* cases. Number of blocks in training and generated images are reported in rows and columns respectively.

of a layout problem, where the task is to find an optimal layout subject to a set of constraints. Automated or interactive layout synthesis has a broad range of potential applications, including urban pattern layout (Yang et al., 2013), decorative mosaics (Hausner, 2001) and furniture arrangement (Yu et al., 2011). Note that many spatial constraints can be encoded in terms of relationships between blocks (Yu et al., 2011). Existing approaches typically design an energy function to be minimized by stochastic search. Our approach suggests how to automatically identify the relevant constraints and their respective weights, and can accomodate hard constraints and exact search. This is especially relevant for *water-tight* layouts (Peng et al., 2014), where the whole space needs to be filled (i.e. no gaps or overlaps) by deforming basic elements from a predetermined set of templates (as in residential building layout (Merrell et al., 2010)). Generally speaking, the LMT framework allows to introduce a learning stage in all application domains where SMT and OMT approaches have shown their potential, ranging, e.g., from engineering of chemical reactions (Fagerberg et al., 2012) to synthetic biology (Yordanov et al., 2013).

References

Barrett, C., Sebastiani, R., Seshia, S. A., and Tinelli, C. Satisfiability modulo theories. In *Handbook of Satisfiability*, chapter 26, pp. 825–885. IOS Press, 2009.

Choi, Jaesik and Amir, Eyal. Lifted relational variational inference. In *UAI'12*, pp. 196–206, 2012.

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

Goodman, Noah D., Mansinghka, Vikash K., Roy, Daniel M., Bonawitz, Keith, and Tenenbaum, Joshua B. Church: a language for generative models. In *UAI*, pp. 220–229, 2008.

Gutmann, B., Jaeger, M., and De Raedt, L. Extending problog with continuous distributions. In *Inductive Logic Programming*, volume 6489, pp. 76–91. 2011.

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

Islam, M. A., Ramakrishnan, C. R., and Ramakrishnan, I. V. Inference in probabilistic logic programs with continuous random variables. *Theory Pract. Log. Program.*, 12(4-5):505–523, September 2012. ISSN 1471-0684.

Merrell, Paul, Schkufza, Eric, and Koltun, Vladlen. Computer-generated residential building layouts. *ACM Trans. Graph.*, 29(6):181:1–181:12, December 2010.

Närman, P., Buschle, M., König, J., and Johnson, P. Hybrid probabilistic relational models for system quality analysis. In *EDOC*, pp. 57–66. IEEE Computer Society, 2010.

Peng, Chi-Han, Yang, Yong-Liang, and Wonka, Peter. Computing layouts with deformable templates. *ACM Trans. Graph.*, 33(4):99:1–99:11, July 2014.

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

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.

Wang, Jue and Domingos, Pedro. Hybrid markov logic networks. In *AAAI'08*, pp. 1106–1111, 2008.

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.

Yu, Lap-Fai, Yeung, Sai-Kit, Tang, Chi-Keung, Terzopoulos, Demetri, Chan, Tony F., and Osher, Stanley J. Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graph.*, 30(4):86:1–86:12, July 2011.