Learning with Combinatorial Optimization Layers: a Probabilistic Approach

Guillaume Dalle[1], Léo Baty[1], Louis Bouvier[1], Axel Parmentier[1]

CERMICS, Ecole des Ponts, Marne-la-Vallée, France

{guillaume.dalle, leo.baty, louis.bouvier, axel.parmentier}@enpc.fr

Keywords : *combinatorial optimization, machine learning, automatic differentiation, Julia programming language.*

Machine learning (ML) and combinatorial optimization (CO) are two essential ingredients of modern industrial processes. While ML extracts meaningful information from noisy data, CO enables decision-making in high-dimensional constrained environments. But in many situations, combining both of these tools is necessary: for instance, we might want to generate predictions from data, and then use those predictions to make optimized decisions. To do that, we need *pipelines* that contain two types of *layers*: ML layers and CO layers.

Let us discuss a generic hybrid ML-CO pipeline, which includes a CO oracle amid several ML layers:

$$\xrightarrow{\text{Input } x} \boxed{\text{ML layers}} \xrightarrow{\text{Objective } \theta} \boxed{\boxed{\text{CO oracle}}} \xrightarrow{\text{Solution } y} \boxed{\text{More ML layers}} \xrightarrow{\text{Output}} \tag{1}$$

The inference problem consists in predicting an output from a given input. It is solved online, and requires the knowledge of the parameters (weights) for each ML layer. On the other hand, the learning problem aims at finding parameters that lead to "good" outputs during inference. It is solved offline based on a training set that contains several inputs, possibly complemented by target outputs.

In Equation (1), we use the term *CO oracle* to emphasize that any algorithm may be used to solve the optimization problem, whether it relies on an existing solver or a handcrafted implementation. In our hybrid ML-CO pipelines, we consider CO oracles f that solve the following kind of problem:

$$f : \theta \longmapsto \operatorname*{argmax}_{v \in \mathcal{V}} \theta^\top v \tag{2}$$

Here, the input $\theta \in \mathbb{R}^d$ is the *objective direction.* Meanwhile, $\mathcal{V} \subset \mathbb{R}^d$ (for *vertices*) denotes a finite set of feasible solutions – which may be exponentially large in d – among which the optimal solution $f(\theta)$ shall be selected.

Due to their many possible applications, hybrid ML-CO pipelines currently attract a lot of research interest. The recent reviews by [1] and [3] are excellent resources on this topic. Unfortunately, relevant software implementations are scattered across paper-specific repositories, with few tests, minimal documentation and sporadic code maintenance. Not only does this make comparison and evaluation difficult for academic purposes, it also hurts practitioners wishing to experiment with such techniques on real use cases. Furthermore, they are seldom compatible with modern ML libraries, which provide a wealth of basic building blocks called layers that allow users to assemble and train complex pipelines. Our goal is to provide a Julia package that address these issues.

We face two main mathematical challenges. First, if we want to use a CO oracle as a layer, we must be able to compute meaningful derivatives using automatic differentiation. Since it may call black box subroutines, an arbitrary CO oracle is seldom compatible with automatic differentiation. And even when it is, its derivatives are zero almost everywhere, which gives us no exploitable slope information. Second, standard ML losses are ill-suited to our setting, because they often ignore the underlying optimization problem. Our goal is to remove these difficulties.

Our general approach is to turn the CO oracle f into a probability distribution $p(\cdot|\theta)$ on $\mathcal{V}$. The naive choice would be the Dirac mass $p(v|\theta) = \delta_{f(\theta)}(v)$, but it shares the lack of differentiability of the oracle itself. Thus, our goal is to spread out the distribution p into an approximation $\widehat{p}$, such that the probability mapping $\theta \longmapsto \widehat{p}(\cdot|\theta)$ becomes smooth with respect to θ. If we can do that, then the expectation mapping

$$\widehat{f} : \theta \longmapsto \mathbb{E}_{\widehat{p}(\cdot|\theta)}[V] = \sum_{v \in \mathcal{V}} v \widehat{p}(v|\theta), \tag{3}$$

where it is understood that $V \sim \widehat{p}(\cdot|\theta)$, will be just as smooth. Furthermore, using expectations of the regret or the non-optimality of the imitated solution leads to natural losses that take into account the structure of the CO layer.

Based on these ideas, we introduce `InferOpt.jl`[1], a Julia package which 1) can turn any CO oracle into a layer with meaningful derivatives, and 2) provides structured loss functions that work well with the resulting layers. It contains several state-of-the-art methods that are fully compatible with Julia's automatic differentiation and ML ecosystem, making CO layers as easy to use as any ML layer. To describe the available methods in a coherent manner, we leverage the unifying concept of probabilistic CO layer, hence the name of our package.

The talk will introduce these probabilistic CO layers, and show how they can be used to build practically efficient algorithms for several combinatorial optimization problems from the literature. We will also illstrate how easy it is to build such pipelines with `InferOpt.jl`. Further details can be found in the following preprint [2].

References

[1] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. 290(2):405–421.

[2] Guillaume Dalle, Léo Baty, Louis Bouvier, and Axel Parmentier. Learning with Combinatorial Optimization Layers: A Probabilistic Approach, July 2022.

[3] James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. End-to-End Constrained Optimization Learning: A Survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4475–4482. International Joint Conferences on Artificial Intelligence Organization.

[1]`https://github.com/axelparmentier/InferOpt.jl`