Learning with Combinatorial Optimization Layers: a Probabilistic Approach

Guillaume Dalle, Léo Baty, Louis Bouvier, Axel Parmentier
presented by Francesco Demelas

Combinatorial Optimization Layers

We want introduce a way to use two kinds of layers:

- Machine Learning layers,
- Combinatorial Optimization layers.

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

Here we find **two main challenges**:

- Transform a C.O. problem in an useful layer (by defining meaningful derivatives),
- Find a good loss.

$$\xrightarrow{\text{Input } x} \boxed{\text{ML layer } \varphi_w} \xrightarrow{\text{Objective } \theta = \varphi_w(x)} \boxed{\text{CO oracle } f} \xrightarrow{\text{Solution } y = f(\theta)} \boxed{\text{Loss function } \mathcal{L}}$$

Setting

$$f : \theta \rightarrow \arg\max_{v \in \nu} \theta^T v$$

where:

- $\theta \in \mathbb{R}^d$ is the *objective direction*.
- $\nu \subseteq \mathbb{R}^d$ is a finite set of the *feasible solutions*.

Any MILP and LP can be written in this way.
Similar arguments can be made for:

$$f : \theta \rightarrow \arg\max_{v \in \nu} \theta^T g(v)$$

for any $g : \nu \rightarrow \mathbb{R}^d$.
We consider this formulation as the objective function is linear ans so it makes no difference to optimize over ν or $conv(\nu)$.

Setting

$$f : \theta \to \arg\max_{v \in \nu} \theta^T v$$

where:

- $\theta \in \mathbb{R}^d$ is the *objective direction*.
- $\nu \subseteq \mathbb{R}^d$ is a finite set of the *feasible solutions*.

Any MILP and LP can be written in this way.
Similar arguments can be made for:

$$f : \theta \to \arg\max_{v \in \nu} \theta^T g(v)$$

for any $g : \nu \to \mathbb{R}^d$.
We consider this formulation as the objective function is linear ans so it makes no difference to optimize over ν or $conv(\nu)$.

Motivating Example

Motivating Example - Warcraft

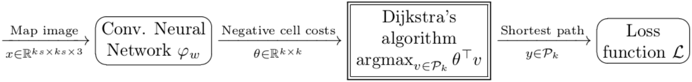
Shortest path on a map.

$$\xrightarrow[x \in \mathbb{R}^{ks \times ks \times 3}]{\text{Map image}} \boxed{\text{Conv. Neural Network } \varphi_w} \xrightarrow[\theta \in \mathbb{R}^{k \times k}]{\text{Negative cell costs}} \boxed{\text{Dijkstra's algorithm } \operatorname{argmax}_{v \in \mathcal{P}_k} \theta^\top v} \xrightarrow[y \in \mathcal{P}_k]{\text{Shortest path}} \boxed{\text{Loss function } \mathcal{L}}$$

Motivating Example

Motivating Example - Warcraft

Shortest path on a map.




```julia
using Flux, Metalhead, Statistics

resnet18 = ResNet(
    18; pretrain=false, nclasses=1
)

warcraft_encoder = Chain(
    resnet18.layers[1][1:4],
    AdaptiveMaxPool((12, 12)),
    x -> mean(x; dims=3),
    x -> dropdims(x; dims=(3, 4)),
    x -> -softplus.(x)
)
```

Code sample 1: CNN encoder for Warcraft

```julia
using Graphs, GridGraphs, LinearAlgebra

function warcraft_maximizer(theta)
    g = GridGraph(-theta)
    path = grid_dijkstra(g, 1, nv(g))
    y = path_to_matrix(g, path)
    return y
end

function warcraft_cost(y; theta_ref)
    return dot(y, theta_ref)
end
```

Code sample 2: Dijkstra optimizer for Warcraft

```julia
x, theta_ref, y_ref = images[1], cells[1], paths[1]
theta = warcraft_encoder(x)
y = warcraft_maximizer(theta)
c = warcraft_cost(y; theta_ref=theta_ref)
```

Code sample 3: Full pipeline for Warcraft shortest paths

Lack of useful derivatives

In order to handle with the lack of useful derivatives we consider **approximate derivatives**.
Let be:

- $p(v|\theta) = \delta_{f(\theta}(v)$,
- $\hat{p}$ the *smooth* and *differentiable* (w.r.t. θ) approximation of p,
- We define the **probabilistic layer** as

$$\hat{f}(\theta) = \mathbb{E}_{\hat{p}(\cdot|\theta)}[V] = \sum_{v \in \nu} v \hat{p}(v|\theta).$$

We assume that the expectation must be tractable, e.g. with Monte Carlo.
We assume also that all the computations must only require calls to the CO oracle f.
$\hat{f}$ is *differentiable* as

$$J_\theta \hat{f}(\theta) = J_\theta \mathbb{E}_{\hat{p}(\cdot|\theta)}[V] = \sum_{v \in \nu} v \nabla_\theta \hat{p}(v|\theta)^T$$

Lack of useful derivatives

In order to handle with the lack of useful derivatives we consider **approximate derivatives**.
Let be:

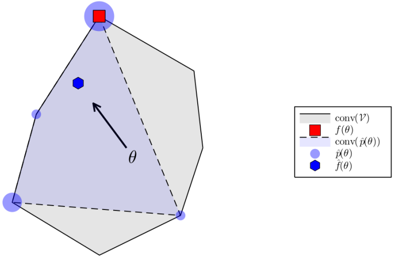
- $p(v|\theta) = \delta_{f(\theta}(v)$,
- $\hat{p}$ the *smooth* and *differentiable* (w.r.t. θ) approximation of p,
- We define the **probabilistic layer** as

$$\hat{f}(\theta) = \mathbb{E}_{\hat{p}(\cdot|\theta)}[V] = \sum_{v \in \nu} v\hat{p}(v|\theta).$$

We assume that the expectation must be tractable, e.g. with Monte Carlo.
We assume also that all the computations must only require calls to the CO oracle f.
$\hat{f}$ is *differentiable* as

$$J_\theta \hat{f}(\theta) = J_\theta \mathbb{E}_{\hat{p}(\cdot\ \theta)}[V] = \sum_{v \in \nu} v \nabla_\theta \hat{p}(v|\theta)^T$$

Effect of CO layer



Example

If $\hat{p}(e_i|\theta) = e^{\theta^T e_i} = e^{\theta_i}$ then the **probabilistic layers** is:

$$\hat{f}(\theta) = \mathbb{E}_{\hat{p}(\cdot|\theta)}[V] = \sum_{i=1}^{d} \frac{e^{\theta_i}}{\sum_{j=1}^{d} e^{\theta_j}} e_i = softmax(\theta)$$

Distributions and regularization

We consider a regularized version of the CO problem:

$$\hat{f}_\Omega : \theta \to \arg\max_{\mu \in dom(\Omega)} \theta^t \mu - \Omega(\mu)$$

with $\Omega : \mathbb{R}^d \to \mathbb{R}$ *smooth* and *convex* penalization function, and $\mu \in dom(\Omega) \subseteq conv(\nu)$.

It is easy to see that any feasible μ is the expectation of some distribution over ν, hence *regularization* is just another way to define *probability distributions*.

Distributions and regularization

We consider a regularized version of the CO problem:

$$\hat{f}_\Omega : \theta \to \arg\max_{\mu \in dom(\Omega)} \theta^t \mu - \Omega(\mu)$$

with $\Omega : \mathbb{R}^d \to \mathbb{R}$ *smooth* and *convex* penalization function, and $\mu \in dom(\Omega) \subseteq conv(\nu)$.
It is easy to see that any feasible μ is the expectation of some distribution over ν, hence *regularization* is just another way to define *probability distributions*.

How compute $\hat{p}$ and $\hat{f}$

We will see *three possibilities*:

- *Additive* Perturbation
- *Multiplicative* Perturbation
- General *Regularization*

Additive Perturbation

$$\hat{f}_\epsilon^+(\theta) = \mathbb{E}[\arg\max_{v\in\nu}(\theta+\epsilon Z)^T v] = \mathbb{E}[f(\theta+\epsilon Z)] = \sum_{v\in\nu} v\hat{p}_\epsilon^+(v|\theta)$$

where $\hat{p}_\epsilon^+(v|\theta) = \mathbb{P}(f(\theta+\epsilon Z) = v)$.

Differentiate

- $\nabla_\theta \hat{p}_\epsilon^+(v|\theta) = \frac{1}{\epsilon}\mathbb{E}[\delta_{f(\theta+\epsilon Z)=v} Z]$
- $J_\theta \hat{f}_\epsilon^+(\theta) = \frac{1}{\epsilon}\mathbb{E}[f(\theta+\epsilon Z) Z^T]$

Associated regularization

Let be $F_\epsilon^+(\theta) = \mathbb{E}[\max_{v\in\nu}(\theta+\epsilon Z)^T v]$ and $\Omega_\epsilon^+(\nu) = (F_\epsilon^+)^*(\nu) = \sup_\theta(\theta^T v - F_\epsilon^+(\theta))$.
$\Omega_\epsilon^+(\nu)$ is convex, $dom(\Omega_\epsilon^+(\nu)) \subseteq conv(\nu)$ and $\hat{f}_\epsilon^+(\theta) = \arg\max_{\mu\in conv(\nu)} \theta^T\mu - \Omega_\epsilon^+(\nu)$

Additive Perturbation

$$\hat{f}^+_\epsilon(\theta) = \mathbb{E}[\arg\max_{v\in\nu}(\theta+\epsilon Z)^T v] = \mathbb{E}[f(\theta+\epsilon Z)] = \sum_{v\in\nu} v\hat{p}^+_\epsilon(v|\theta)$$

where $\hat{p}^+_\epsilon(v|\theta) = \mathbb{P}(f(\theta+\epsilon Z)=v)$.

Differentiate

- $\nabla_\theta \hat{p}^+_\epsilon(v|\theta) = \frac{1}{\epsilon}\mathbb{E}[\delta_{f(\theta+\epsilon Z)=v} Z]$
- $J_\theta \hat{f}^+_\epsilon(\theta) = \frac{1}{\epsilon}\mathbb{E}[f(\theta+\epsilon Z)Z^T]$

Associated regularization

Let be $F^+_\epsilon(\theta) = \mathbb{E}[\max_{v\in\nu}(\theta+\epsilon Z)^T v]$ and
$\Omega^+_\epsilon(\nu) = (F^+_\epsilon)^*(\nu) = \sup_\theta(\theta^T v - F^+_\epsilon(\theta))$.
$\Omega^+_\epsilon(\nu)$ is convex, $dom(\Omega^+_\epsilon(\nu)) \subseteq conv(\nu)$ and
$\hat{f}^+_\epsilon(\theta) = \arg\max_{\mu\in conv(\nu)}\theta^T\mu - \Omega^+_\epsilon(\nu)$

Additive Perturbation

$$\hat{f}_\epsilon^+(\theta) = \mathbb{E}[\arg\max_{v\in\nu}(\theta+\epsilon Z)^T v] = \mathbb{E}[f(\theta+\epsilon Z)] = \sum_{v\in\nu} v\hat{p}_\epsilon^+(v|\theta)$$

where $\hat{p}_\epsilon^+(v|\theta) = \mathbb{P}(f(\theta+\epsilon Z) = v)$.

Differentiate

- $\nabla_\theta \hat{p}_\epsilon^+(v|\theta) = \frac{1}{\epsilon}\mathbb{E}[\delta_{f(\theta+\epsilon Z)=v} Z]$
- $J_\theta \hat{f}_\epsilon^+(\theta) = \frac{1}{\epsilon}\mathbb{E}[f(\theta+\epsilon Z)Z^T]$

Associated regularization

Let be $F_\epsilon^+(\theta) = \mathbb{E}[\max_{v\in\nu}(\theta+\epsilon Z)^T v]$ and
$\Omega_\epsilon^+(\nu) = (F_\epsilon^+)^*(\nu) = \sup_\theta(\theta^T v - F_\epsilon^+(\theta))$.
$\Omega_\epsilon^+(\nu)$ is convex, $dom(\Omega_\epsilon^+(\nu)) \subseteq conv(\nu)$ and
$\hat{f}_\epsilon^+(\theta) = \arg\max_{\mu\in conv(\nu)} \theta^T\mu - \Omega_\epsilon^+(\nu)$

Multiplicative Perturbation

$$\hat{f}_\epsilon^{\odot} = \mathbb{E}[\arg\max_{v \in \nu} (\theta \odot e^{\epsilon Z - \epsilon^2/2})^T v] = \mathbb{E}[f(\theta \odot e^{\epsilon Z - \epsilon^2/2}] = \sum_{v \in \nu} v \hat{p}_\epsilon^{\odot}(v|\theta)$$

where $\hat{p}_\epsilon^{\odot}(v|\theta) = \mathbb{P}(f(\theta \odot e^{\epsilon Z - \epsilon^2/2}) = v)$.

Differentiate

- $\nabla_\theta \hat{p}_\epsilon^{\odot}(v|\theta) = \frac{1}{\epsilon}\theta \odot \mathbb{E}[\delta_{f(\theta \odot e^{\epsilon Z - \epsilon^2/2}) = v} Z]$
- $J_\theta \hat{f}_\epsilon^{\odot}(\theta) = \frac{1}{\epsilon\theta}\mathbb{E}[f(\theta \odot e^{\epsilon Z - \epsilon^2/2}) Z^T]$

Associated regularization

Let be $F_\epsilon^{\odot}(\theta) = \mathbb{E}[\max_{v \in \nu}(\theta \odot e^{\epsilon Z - \epsilon^2/2})^T v]$ and $\Omega_\epsilon^{\odot}(\nu) = (F_\epsilon^{\odot})^*(\nu)$.
$\Omega_\epsilon^{\odot}(\nu)$ is convex and satisfies
$\hat{f}_\epsilon^{\odot}(\theta) = \arg\max_{\mu \in conv(\nu)} \theta^T \mu - \Omega_\epsilon^{\odot}(\nu) = \hat{f}_{\Omega_\epsilon^{\odot}(\theta)}$ but
$dom(\Omega_\epsilon^{\odot}(\nu)) \not\subseteq conv(\nu)$.

Multiplicative Perturbation

$$\hat{f}_\epsilon^{\odot} = \mathbb{E}[\arg\max_{v \in \nu} (\theta \odot e^{\epsilon Z - \epsilon^2/2})^T v] = \mathbb{E}[f(\theta \odot e^{\epsilon Z - \epsilon^2/2}] = \sum_{v \in \nu} v \hat{p}_\epsilon^{\odot}(v|\theta)$$

where $\hat{p}_\epsilon^{\odot}(v|\theta) = \mathbb{P}(f(\theta \odot e^{\epsilon Z - \epsilon^2/2}) = v)$.

Differentiate

- $\nabla_\theta \hat{p}_\epsilon^{\odot}(v|\theta) = \frac{1}{\epsilon}\theta \odot \mathbb{E}[\delta_{f(\theta \odot e^{\epsilon Z - \epsilon^2/2}) = v} Z]$
- $J_\theta \hat{f}_\epsilon^{\odot}(\theta) = \frac{1}{\epsilon\theta}\mathbb{E}[f(\theta \odot e^{\epsilon Z - \epsilon^2/2}) Z^T]$

Associated regularization

Let be $F_\epsilon^{\odot}(\theta) = \mathbb{E}[\max_{v \in \nu}(\theta \odot e^{\epsilon Z - \epsilon^2/2})^T v]$ and $\Omega_\epsilon^{\odot}(\nu) = (F_\epsilon^{\odot})^*(\nu)$.
$\Omega_\epsilon^{\odot}(\nu)$ is convex and satisfies
$\hat{f}_\epsilon^{\odot}(\theta) = \arg\max_{\mu \in conv(\nu)} \theta^T \mu - \Omega_\epsilon^{\odot}(\nu) = \hat{f}_{\Omega_\epsilon^{\odot}(\theta)}$ but
$dom(\Omega_\epsilon^{\odot}(\nu)) \not\subseteq conv(\nu)$.

Multiplicative Perturbation

$$\hat{f}_\epsilon^{\odot} = \mathbb{E}[\arg\max_{v \in \nu} (\theta \odot e^{\epsilon Z - \epsilon^2/2})^T v] = \mathbb{E}[f(\theta \odot e^{\epsilon Z - \epsilon^2/2}] = \sum_{v \in \nu} v \hat{p}_\epsilon^{\odot}(v|\theta)$$

where $\hat{p}_\epsilon^{\odot}(v|\theta) = \mathbb{P}(f(\theta \odot e^{\epsilon Z - \epsilon^2/2}) = v)$.

Differentiate

- $\nabla_\theta \hat{p}_\epsilon^{\odot}(v|\theta) = \frac{1}{\epsilon}\theta \odot \mathbb{E}[\delta_{f(\theta \odot e^{\epsilon Z - \epsilon^2/2}) = v} Z]$
- $J_\theta \hat{f}_\epsilon^{\odot}(\theta) = \frac{1}{\epsilon\theta}\mathbb{E}[f(\theta \odot e^{\epsilon Z - \epsilon^2/2}) Z^T]$

Associated regularization

Let be $F_\epsilon^{\odot}(\theta) = \mathbb{E}[\max_{v \in \nu} (\theta \odot e^{\epsilon Z - \epsilon^2/2})^T v]$ and $\Omega_\epsilon^{\odot}(\nu) = (F_\epsilon^{\odot})^*(\nu)$.
$\Omega_\epsilon^{\odot}(\nu)$ is convex and satisfies
$\hat{f}_\epsilon^{\odot}(\theta) = \arg\max_{\mu \in conv(\nu)} \theta^T \mu - \Omega_\epsilon^{\odot}(\nu) = \hat{f}_{\Omega_\epsilon^{\odot}(\theta)}$ but
$dom(\Omega_\epsilon^{\odot}(\nu)) \not\subseteq conv(\nu)$.

General regularization

Starting from an *explicit regularization* Ω smooth and convex we can obtain an *approximate* $\hat{f}_\Omega(\theta)$ using the **Frank-Wolfe algorithm**.

The Frank-Wolfe algorithm is interesting for two reasons:

- Requires only the access to the C.O. oracle f and the gradient of Ω.
- The algorithm provides both a solution $\hat{f}_\Omega(\theta)$, but also a sparse probability distribution $\hat{p}_\Omega^{FW}$ (this one is not uniquely specified by Ω).

Differentiate

$$J_\theta \hat{f}_\Omega(\theta) = \sum_{v \in \nu} v \nabla_\theta \hat{p}_\Omega^{FW}(v|\theta)^T$$

General regularization

Starting from an *explicit regularization* Ω smooth and convex we can obtain an *approximate* $\hat{f}_\Omega(\theta)$ using the **Frank-Wolfe algorithm**.
The Frank-Wolfe algorithm is interesting for two reasons:

- Requires only the access to the C.O. oracle f and the gradient of Ω.
- The algorithm provides both a solution $\hat{f}_\Omega(\theta)$, but also a sparse probability distribution $\hat{p}_\Omega^{FW}$ (this one is not uniquely specified by Ω).

Differentiate

$$J_\theta \hat{f}_\Omega(\theta) = \sum_{v \in \nu} v \nabla_\theta \hat{p}_\Omega^{FW}(v|\theta)^T$$

General regularization

Starting from an *explicit regularization* Ω smooth and convex we can obtain an *approximate* $\hat{f}_\Omega(\theta)$ using the **Frank-Wolfe algorithm**.
The Frank-Wolfe algorithm is interesting for two reasons:

- Requires only the access to the C.O. oracle f and the gradient of Ω.
- The algorithm provides both a solution $\hat{f}_\Omega(\theta)$, but also a sparse probability distribution $\hat{p}_\Omega^{FW}$ (this one is not uniquely specified by Ω).

Differentiate

$$J_\theta \hat{f}_\Omega(\theta) = \sum_{v \in \nu} v \nabla_\theta \hat{p}_\Omega^{FW}(v|\theta)^T$$

Example: Quadratic Penality

If we consider the *quadratic penalty* $\Omega(\mu) = \frac{1}{2}\|\mu\|^2$ we find the **Sparse Map Method** with:

$$\hat{f}_\Omega = \arg\max_{\mu \in conv(\nu)} \{\theta^T \mu - \frac{1}{2}\|\mu\|^2\} = \arg\min_{\mu \in conv(\nu)} \|\mu - \theta\|^2.$$

Example: Quadratic Penality

If we consider the *quadratic penalty* $\Omega(\mu) = \frac{1}{2}\|\mu\|^2$ we find the **Sparse Map Method** with:

$$\hat{f}_\Omega = \arg\max_{\mu \in conv(\nu)} \{\theta^T \mu - \frac{1}{2}\|\mu\|^2\} = \arg\min_{\mu \in conv(\nu)} \|\mu - \theta\|^2.$$

```
using InferOpt

perturbed_add = PerturbedAdditive(
    warcraft_maximizer;
    epsilon=0.5, nb_samples=10
)

perturbed_mult = PerturbedMultiplicative(
    warcraft_maximizer;
    epsilon=0.5, nb_samples=10
)
```

Code sample 5: Probabilistic CO layers defined by perturbation

```
using InferOpt

regularized = RegularizedGeneric(
    warcraft_maximizer;
    omega=y -> 0.5 * sum(y .^ 2),
    omega_grad=y -> y
)
```

Code sample 6: Probabilistic CO layer defined by regularization

Choice of the loss

Two main paradigms:

- Learning by experience
- Learning by imitation

Learning by Experience and Reinforcement learning learning

Similar to reinforcement learning, but there are few **differences**, as in RL:

- is based on *Markov decision processes*,
- the available actions are *elementary*,
- the *policy update is local* (depends to the state and to the action),
- The *Bellman fixed point equation is used explicitly* to derive parameter updates

Learning by experience

Learning Problem

$$\min_w \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\varphi_w(x^{(i)}))$$

For simplicity we consider a single point input x and the gradient for $\theta = \varphi_w(x)$.

We can assume that exists $c : \nu \to \mathbb{R}$ a **cost function** for the problem. A *natural loss* could be $R(\theta) = c(f(\theta))$ and so we can have the impulse to take $\hat{R}(\theta) = c(f(\theta))$ but also this is **not** *smooth* and *it could be defined only in* ν.

Solution : Expected regret

$$R_{\hat{p}}(\theta) = \mathbb{E}_{\hat{p}(\cdot|\theta)}[c(V)]$$

this is as smooth as the probability mapping $\hat{p}(\cdot|\theta)$.

Learning by experience

Learning Problem

$$\min_w \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\varphi_w(x^{(i)}))$$

For simplicity we consider a single point input x and the gradient for $\theta = \varphi_w(x)$.
We can assume that exists $c : \nu \to \mathbb{R}$ a **cost function** for the problem. A *natural loss* could be $R(\theta) = c(f(\theta))$ and so we can have the impulse to take $\hat{R}(\theta) = c(f(\theta))$ but also this is **not** *smooth* and *it could be defined only in* ν.

Solution : Expected regret

$$R_{\hat{p}}(\theta) = \mathbb{E}_{\hat{p}(\cdot|\theta)}[c(V)]$$

this is as smooth as the probability mapping $\hat{p}(\cdot|\theta)$.

Learning by experience

Learning Problem

$$\min_w \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\varphi_w(x^{(i)}))$$

For simplicity we consider a single point input x and the gradient for $\theta = \varphi_w(x)$.
We can assume that exists $c : \nu \to \mathbb{R}$ a **cost function** for the problem. A *natural loss* could be $R(\theta) = c(f(\theta))$ and so we can have the impulse to take $\hat{R}(\theta) = c(f(\theta))$ but also this is **not** *smooth* and *it could be defined only in* ν.

Solution : Expected regret

$$R_{\hat{p}}(\theta) = \mathbb{E}_{\hat{p}(\cdot|\theta)}[c(V)]$$

this is as smooth as the probability mapping $\hat{p}(\cdot|\theta)$.

Derivative for regret - Learning by experience

- $\nabla_\theta R_{\hat{p}^+_\epsilon(\theta)} = \frac{1}{\epsilon}\mathbb{E}[(c \circ f)(\theta + \epsilon Z)Z]$
- $\nabla_\theta R_{\hat{p}^\odot_\epsilon(\theta)} = \frac{1}{\epsilon}\mathbb{E}[(c \circ f)(\theta + e^{\epsilon Z - \epsilon^2/2})Z]$
- $\nabla_\theta R_{\hat{p}^{FW}_\epsilon(\theta)} = \sum_{v \in \nu} c(v)\nabla_\theta \nabla_\theta \hat{p}^{FW}_\Omega(v \mid \theta)$

```julia
using InferOpt

regret_pert = Pushforward(
    perturbed_add, warcraft_cost
)
regret_reg = Pushforward(
    regularized, warcraft_cost
)
```

Code sample 7: Expected regrets associated with probabilistic CO layers

```julia
using Zygote

R = regret(theta)
Zygote.gradient(regret, theta)
```

Code sample 8: Supported operations for an expected regret

```julia
using Flux, InferOpt

gradient_optimizer = ADAM()
parameters = Flux.params(warcraft_encoder)
data = images

function pipeline_loss(x)
    theta = warcraft_encoder(x)
    return regret(theta)
end

for epoch in 1:1000
    train!(pipeline_loss, parameters, data, gradient_optimizer)
end
```

Code sample 9: Learning with an expected regret

Learning by imitation

Learning Problem

$$\min_w \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(\varphi_w(x^{(i)})), \bar{t}^{(i)})$$

We have two kinds of target:

- a **good quality solution** $t = \bar{y}$
- true **objective direction** $\bar{\theta}$

Method	Notation	Target	Base loss	Regul.	Loss formula
S-SVM	$\mathcal{L}_\ell^{\text{S-SVM}}$	$\bar{y}$	$\ell(y, \bar{y})$	No	$\max_y \ell(y, \bar{y}) + \theta^\top (y - \bar{y})$
SPO+	$\mathcal{L}^{\text{SPO+}}$	$(\bar{\theta}, \bar{y})$	$\bar{\theta}^\top (\bar{y} - y)$	No	$\max_y \bar{\theta}^\top (\bar{y} - y) + 2\theta^\top (y - \bar{y})$
FY	$\mathcal{L}_\Omega^{\text{FY}}$	$\bar{y}$	0	Yes	$\max_y \theta^\top (y - \bar{y}) - (\Omega(y) - \Omega(\bar{y}))$
Generic	$\mathcal{L}^{\text{gen}}$	$\bar{t}$	$\ell(y, \bar{t})$	Yes	$\max_y \ell(y, \bar{t}) + \theta^\top (y - \bar{y}) - (\Omega(y) - \Omega(\bar{y}))$

Table 2: A common decomposition for loss functions in imitation learning

Generic Imitation Loss

Let be

$$\mathcal{L}^{aux}(\theta, \bar{t}, y) = l(y, \bar{t}) + \theta^T (y - \bar{y}) - (\Omega(y) - \Omega(\bar{y}))$$

Generic Loss

$$\mathcal{L}^{gen}(\theta, \bar{t}) = \max_{y \in dom(\Omega)} \mathcal{L}^{aux}(\theta, \bar{t}, y)$$

$\mathcal{L}^{gen}$ is convex w.r.t. θ and

$$\arg\max_{y \in dom(\Omega)} \mathcal{L}^{aux}(\theta, \bar{t}, y) - \bar{y} \in \partial_\theta \mathcal{L}^{gen}(\theta, \bar{t})$$

The generic loss is a cross-over between the Fenchel-Young loss and the problem specific base loss.

Generic Imitation Loss

Let be

$$\mathcal{L}^{aux}(\theta, \bar{t}, y) = l(y, \bar{t}) + \theta^T (y - \bar{y}) - (\Omega(y) - \Omega(\bar{y}))$$

Generic Loss

$$\mathcal{L}^{gen}(\theta, \bar{t}) = \max_{y \in dom(\Omega)} \mathcal{L}^{aux}(\theta, \bar{t}, y)$$

$\mathcal{L}^{gen}$ is convex w.r.t. θ and

$$\arg\max_{y \in dom(\Omega)} \mathcal{L}^{aux}(\theta, \bar{t}, y) - \bar{y} \in \partial_\theta \mathcal{L}^{gen}(\theta, \bar{t})$$

The generic loss is a cross-over between the Fenchel-Young loss and the problem specific base loss.

```julia
using InferOpt

fyl_pert = FenchelYoungLoss(perturbed_add)
fyl_reg = FenchelYoungLoss(regularized)
spol = SPOPlusLoss(warcraft_maximizer)
```

Code sample 10: Example imitation losses

```julia
using Zygote

L = loss(theta, y_ref)
Zygote.gradient(loss, theta, y_ref)
```

Code sample 11: Supported operations for an imitation loss

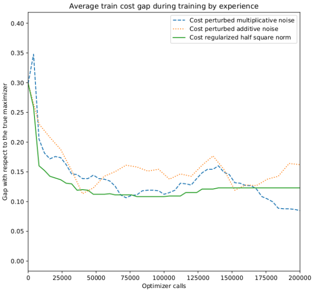
```julia
using Flux, InferOpt

gradient_optimizer = ADAM()
parameters = Flux.params(warcraft_encoder)
data = zip(images, paths)

function pipeline_loss(x, y)
    theta = warcraft_encoder(x)
    return loss(theta, y)
end

for epoch in 1:1000
    train!(pipeline_loss, parameters, data, gradient_optimizer)
end
```

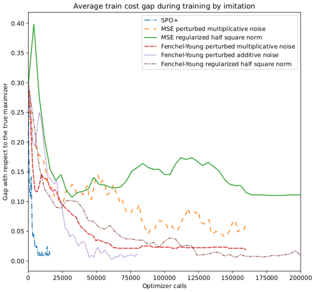
Code sample 12: Learning with an imitation loss

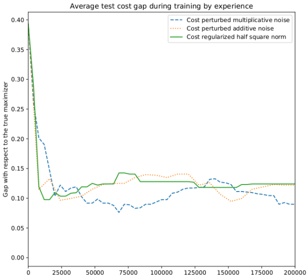
Setting for Warcraft

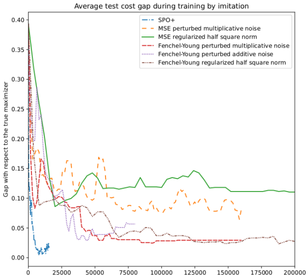
Name	**CO problem (CO oracle)**	**Probabilistic CO layer**	**Exp./Imit. Target**	**Loss**
Cost perturbed multiplicative noise	SP with non-negative costs (Dijkstra)	Multiplicative perturbation	Experience No target	Perturbed cost
Cost perturbed additive noise	SP on an extended acyclic graph (Ford-Bellman)	Additive perturbation	Experience No target	Perturbed cost
Cost regularized half square norm	SP on an extended acyclic graph (Ford-Bellman)	Half square norm	Experience No target	Regularized cost
SPO+	SP on an extended acyclic graph (Ford-Bellman)	No regularization	Imitation Cost and path	SPO+ loss
MSE perturbed multiplicative noise	SP with non-negative costs (Dijkstra)	Multiplicative perturbation	Imitation Path	Mean squared error
MSE regularized half square norm	SP on an extended acyclic graph (Ford-Bellman)	Half square norm	Imitation Path	Mean squared error
Fenchel-Young perturbed multiplicative noise	SP with non-negative costs (Dijkstra)	Multiplicative perturbation	Imitation Path	Fenchel-Young
Fenchel-Young perturbed additive noise	SP on an extended acyclic graph (Ford-Bellman)	Additive perturbation	Imitation Path	Fenchel-Young
Fenchel-Young regularized half square norm	SP on an extended acyclic graph (Ford-Bellman)	Half square norm	Imitation Path	Fenchel-Young

Results


Average train cost gap during training by experience
Cost perturbed multiplicative noise
Cost perturbed additive noise
Cost regularized half square norm
Gap with respect to the true maximizer
Optimizer calls


Average train cost gap during training by imitation
SPO+
MSE perturbed multiplicative noise
MSE regularized half square norm
Fenchel-Young perturbed multiplicative noise
Fenchel-Young perturbed additive noise
Fenchel-Young regularized half square norm
Gap with respect to the true maximizer
Optimizer calls


Average test cost gap during training by experience
Cost perturbed multiplicative noise
Cost perturbed additive noise
Cost regularized half square norm
Gap with respect to the true maximizer


Average test cost gap during training by imitation
SPO+
MSE perturbed multiplicative noise
MSE regularized half square norm
Fenchel-Young perturbed multiplicative noise
Fenchel-Young perturbed additive noise
Fenchel-Young regularized half square norm
Gap with respect to the true maximizer

Conclusions

Contributions:

- Implemented **InferOpt.jl**
- New **perturbation technique** that allows to accept objective vectors with a certain sign.
- Probabilistic regularization allows to **differentiate through large class of C.O. layers**, combining the Frank-Wolfe algorithm with implicit differentiation.
- **Generic decomposition framework** for imitation losses.

Conclusions

Contributions:

- Implemented **InferOpt.jl**
- New **perturbation technique** that allows to accept objective vectors with a certain sign.
- Probabilistic regularization allows to **differentiate through large class of C.O. layers**, combining the Frank-Wolfe algorithm with implicit differentiation.
- **Generic decomposition framework** for imitation losses.

Conclusions

Contributions:

- Implemented **InferOpt.jl**
- New **perturbation technique** that allows to accept objective vectors with a certain sign.
- Probabilistic regularization allows to **differentiate through large class of C.O. layers**, combining the Frank-Wolfe algorithm with implicit differentiation.
- **Generic decomposition framework** for imitation losses.

Conclusions

Contributions:

- Implemented **InferOpt.jl**
- New **perturbation technique** that allows to accept objective vectors with a certain sign.
- Probabilistic regularization allows to **differentiate through large class of C.O. layers**, combining the Frank-Wolfe algorithm with implicit differentiation.
- **Generic decomposition framework** for imitation losses.