

Online Multi-task Learning with Hard Constraints

Gábor Lugosi¹, Omiros Papaspiliopoulos¹, and Gilles Stoltz²

¹Universitat Pompeu Fabra – ICREA

²CNRS – École normale supérieure – HEC Paris



M **classification tasks** are to be performed **simultaneously**, in a repeated fashion.

To each round t and to each task $j = 1, \dots, M$ is associated a loss $\ell_{j,t}$, given by a hinge loss: $\ell_{j,t} = (1 - y_{j,t} \mathbf{w}_{j,t} \cdot \mathbf{x}_{j,t})_+$.

These individual losses then lead to a **global loss** through a given (e.g., L_2 or L_∞) **norm**:

$$\ell_t = \left\| (\ell_{1,t}, \dots, \ell_{M,t}) \right\|$$

The goal is to minimize $\ell_1 + \dots + \ell_n$ with respect to the best constant choice of a M -tuple of weight vectors $\mathbf{w}_1, \dots, \mathbf{w}_M$.

M tasks are to be performed but only **once at a round**. N actions are available for each task.

To each task $j = 1, \dots, M$ are associated an outcome space \mathcal{Y}_j and a loss function $\ell^{(j)} : \{1, \dots, N\} \times \mathcal{Y}_j \rightarrow [0, 1]$.

At each round t , the forecaster is **informed** of the index I_t of the **active task** and chooses a **single action** K_t .

Nature picks an outcome y_t and the forecaster suffers the loss $\ell_t = \ell^{(I_t)}(K_t, y_t)$.

The goal is to minimize $\ell_1 + \dots + \ell_n$ with respect to

$$\min \left\{ \sum_{t=1}^n \ell^{(I_t)}(k_{I_t}, y_t) : (k_1, \dots, k_M) \text{ s.t. } |\{k_1, \dots, k_M\}| \leq m \right\}$$

for a given parameter $1 \leq m \leq M$.



Multiple tasks with hard constraints: Overview

As in the setting of Dekel, Long, and Singer, in our setting we have to play **all tasks** at each round.

In the setting of Abernethy, Bartlett, and Rakhlin, constraints modeling the relatedness between tasks are considered **only** on the **comparison class**.

There, no constraint is enforced on the way the forecaster predicts.

In our setting, we will consider constraints on the actions chosen by the forecaster (which we call **hard constraints**). These constraints will account for the relatedness between the tasks.



Multiple tasks with hard constraints: Setting

M tasks are to be performed **simultaneously**, in a repeated fashion.

For simplicity, we consider a finite common action space

$$\mathcal{X} = \{x_1, \dots, x_N\} \subset \mathbb{R}.$$

To each task $j = 1, \dots, M$ are associated an outcome space \mathcal{Y}_j and a loss function $\ell^{(j)} : \mathcal{X} \times \mathcal{Y}_j \rightarrow [0, 1]$.

At each round $t = 1, \dots, n$, the forecaster selects a vector of **simultaneous** actions

$$\mathbf{X}_t = (x_{K_{1,t}}, \dots, x_{K_{M,t}})$$

while Nature chooses a vector of outcomes

$$\mathbf{y}_t = (y_{1,t}, \dots, y_{M,t}).$$

No assumption is made on the sequence of the $\mathbf{y}_1, \mathbf{y}_2, \dots$ (the so-called **individual sequence** framework).



Multiple tasks with hard constraints: Setting

We consider a set $\mathcal{A} \subset \mathcal{X}^M$ of **legal M -tuples** of actions.

We impose the constraint that the vectors of simultaneous actions \mathbf{X}_t taken by the forecaster have to lie in \mathcal{A} .

The losses ℓ_t associated to each round t are given by the sum of the individual losses on each task:

$$\ell_t = \ell(\mathbf{X}_t, \mathbf{y}_t) = \sum_{j=1}^M \ell^{(j)}(x_{K_{j,t}}, y_{j,t})$$

The goal is to minimize $\ell_1 + \dots + \ell_n$ with respect to

$$\min_{\mathbf{x} \in \mathcal{A}} \sum_{t=1}^n \ell(\mathbf{x}, \mathbf{y}_t)$$



Escalation constraint: when tasks are consumers, actions are conditions offered to the consumers, and both tasks and conditions are ranked in a natural order

$$\mathcal{A} = \left\{ (x_{k_1}, \dots, x_{k_M}) : \forall j \leq M-1, x_{k_j} \leq x_{k_{j+1}} \right\} .$$

Budget constraint: same setting, with the interpretation that x_k is the cost of choosing action k ; for a budget $B \geq 0$,

$$\mathcal{A} = \left\{ (x_{k_1}, \dots, x_{k_M}) : \sum_{j=1}^M x_{k_j} \leq B \right\} .$$

Constancy constraint: for a parameter $1 \leq m \leq M$,

$$\mathcal{A} = \left\{ (x_{k_1}, \dots, x_{k_M}) : \sum_{j=1}^{M-1} \mathbb{I}_{\{k_j \neq k_{j+1}\}} \leq m \right\} .$$

A forecaster and a performance bound

We perform a standard **exponentially weighted average** on the elements of \mathcal{A} .

We denote by

$$L_{t-1}(\mathbf{x}) = \sum_{\tau=1}^{t-1} \ell(\mathbf{x}, \mathbf{y}_\tau)$$

the cumulative loss of a **legal action** $\mathbf{x} \in \mathcal{A}$ suffered just before round t .

For each round $t \geq 2$, the vector of actions \mathbf{X}_t is drawn at random according to the distribution \mathbf{p}_t on \mathcal{A} : for all $\mathbf{x} \in \mathcal{A}$,

$$\mathbf{p}_t(\mathbf{x}) = \frac{\exp(-\eta L_{t-1}(\mathbf{x}))}{\sum_{\mathbf{a} \in \mathcal{A}} \exp(-\eta L_{t-1}(\mathbf{a}))},$$

where $\eta > 0$ is a parameter to be tuned.



A forecaster and a performance bound

The regret of the previous forecaster (for a suitable choice of η) is bounded as follows, with probability at least $1 - \delta$:

$$\sum_{t=1}^n \ell(\mathbf{x}_t, \mathbf{y}_t) - \min_{\mathbf{x} \in \mathcal{A}} \sum_{t=1}^n \ell(\mathbf{x}, \mathbf{y}_t) \leq M \sqrt{\frac{n}{2} \log \frac{|\mathcal{A}|}{\delta}}$$

The issue is to reduce the **computational complexity**, which is proportional to $|\mathcal{A}|$ for a naive implementation.

For example, in the case of the **constancy constraint** (fewer than m shifts), $|\mathcal{A}|$ is of the order of $(MN)^m / m!$.

To do so, we perform a reduction to an **online shortest path** problem.



Reduction to an online shortest path problem

For the reduction to be efficient, we need a good description of the legal M -tuples.

To that end, we consider an **additional state space** \mathcal{S} and its extension $\mathcal{S}^* = \mathcal{S} \cup \{\star\}$. The symbol \star will mark illegal vectors of actions.

To each j -tuple $(x_{k_1}, \dots, x_{k_j})$ of length $j \leq M$ we associate a state denoted by

$$S((x_{k_1}, \dots, x_{k_j})) \in \mathcal{S}^*$$

We assume that the **application** S implicitly defined above satisfies a **Markovian assumption**:

$S((x_{k_1}, \dots, x_{k_j}))$ only depends

- on the value of x_{k_j} and
- on the state $S((x_{k_1}, \dots, x_{k_{j-1}}))$.



Reduction to an online shortest path problem

We also introduce a **transition function** T from $\mathcal{X} \times \mathcal{S}^*$ to the **subsets** of $\mathcal{X} \times \mathcal{S}^*$.

It keeps track of the legal transitions from a given j -tuple $(x_{k_1}, \dots, x_{k_j})$ **summarized** by its **state** to some $(j + 1)$ -tuples $(x_{k_1}, \dots, x_{k_j}, x_{k_{j+1}})$.

We now describe \mathcal{S} , S , and T on the example dealing with the **constancy constraint** (fewer than m shifts).



State and transition functions for the constancy constraint

The **state function** S counts the number of shifts seen so far,

$$S\left((x_{k_1}, \dots, x_{k_j})\right) = \begin{cases} S'\left((x_{k_1}, \dots, x_{k_j})\right) & \text{if } S'\left((x_{k_1}, \dots, x_{k_j})\right) \leq m, \\ \star & \text{otherwise,} \end{cases}$$

where
$$S'\left((x_{k_1}, \dots, x_{k_j})\right) = \sum_{i=1}^{j-1} \mathbb{I}_{\{k_i \neq k_{i+1}\}}.$$

The **transition function** T is described by the following relations:
for all $x \in \mathcal{X}$,

- $T((x, \star)) = \mathcal{X} \times \{\star\}$
- for all $s = 0, \dots, m-1$,
$$T((x, s)) = \{(x, s)\} \cup \left((\mathcal{X} \setminus \{x\}) \times \{s+1\} \right)$$
- $T((x, m)) = \{(x, m)\} \cup \left((\mathcal{X} \setminus \{x\}) \times \{\star\} \right)$



Reduction to an online shortest path problem

We are now ready to describe the **graph of the paths**.

It contains $MN|\mathcal{S}|$ vertices, indexed by $\{1, \dots, M\} \times \mathcal{X} \times \mathcal{S}$.

Two vertices $v = (j, x_k, s)$ and $v' = (j', x_{k'}, s')$ are **connected** with an edge if and only if $j' = j + 1$, and $(x_{k'}, s') \in T((x_k, s))$, that is, $(x_k, s) \rightarrow (x_{k'}, s')$ is a legal transition between tasks j and $j + 1$.

The **loss** associated with such an **edge** equals the cumulative loss of action $x_{k'}$ in task j' in the previous time rounds,

$$L_{t-1}^{(j')}(x_{k'}) = \sum_{\tau=1}^{t-1} \ell^{(j')}(x_{k'}, y_{j', \tau})$$

We add a **source node** and connect all vertices of the form $(1, x, s)$ to it, with an associated loss equal to $L_{t-1}^{(1)}(x)$.

Finally, we connect to a **sink node** all nodes of the form (M, x, s) and for which there exists a path from the source.



Reduction to an online shortest path problem

Note that in the graph thus constructed, the paths from the source node to the sink node correspond exactly to the legal M -tuples of actions.

The sum of the losses over the edges of a path equals the cumulative loss of the corresponding legal M -tuples of actions.

Choosing such a path according to an exponentially weighted average can be done in an **efficient** way, with a complexity bounded by the **number of edges**.

This is done by **dynamic programming**, see Takimoto and Warmuth; JMLR'04 or György, Linder, and Lugosi; COLT'05.



We now work out the complexity bounds in the example with the **constancy constraint**.

There are fewer than MN^2m edges in the associated graph.

The complexity of the implementation of the reduction is thus of the order of MN^2m , to be compared to the complexity of the most straightforward implementation, which is proportional to $|\mathcal{A}|$, that is, to $(MN)^m/m!$.

This is interesting since in the applications we have in mind $m \ll N \ll M$.



Five extensions can be considered:

- **Tracking** the best simultaneous action in $|\mathcal{A}|$
- A **bandit** case where only the total loss of the chosen action $\mathbf{X}_t \in \mathcal{A}$ is observed at each round
- A setting where the forecaster is only **evaluated** on a **subset** of fixed size M' of the tasks (which he chooses)
- Other than additive measures of the global loss: **Markovian global losses**, like the min loss or the max loss
- Dealing with a **continuum of tasks**

We worked out all extensions but we present **only** the **last one** in this short talk.



Extension to a continuum of tasks

Tasks are now indexed by $g \in [0, 1]$.

The action space is still finite, $\mathcal{X} = \{x_1, \dots, x_N\} \subset \mathbb{R}$.

A **simultaneous action** is given by a measurable **function**

$$I : g \in [0, 1] \mapsto I(g) \in \mathcal{X}$$

Nature's choice of the outcome is modeled by a measurable loss function $\psi : [0, 1] \times \mathcal{X} \rightarrow [0, 1]$.

The incurred loss is then

$$\ell(I, \psi) = \int_{[0,1]} \psi(g, I(g)) \, dg = \sum_{x \in \mathcal{X}} \int_{\{I=x\}} \psi(g, x) \, dg$$



Extension to a continuum of tasks

We only discuss one particular **hard constraint**: the extension of the constancy constraint.

The set \mathcal{A} of legal actions is formed by those functions I which are right-continuous and have only **m shifts** (remember that the I take values in a finite set).

The forecaster is bound to choose repeatedly simultaneous actions $I_t \in \mathcal{A}$ and his regret is then defined as

$$R_n = \sum_{t=1}^n \ell(I_t, \psi_t) - \inf_{I \in \mathcal{A}} \sum_{t=1}^n \ell(I, \psi_t)$$

Motivation: Alternative model when the number M of tasks is large. E.g., the decision maker has to determine actions for each individual in a large population, which he does by setting some thresholds.



Extension to a continuum of tasks

Each element of \mathcal{A} is described by an element of \mathcal{X}^m and an element of the simplex of order $m + 1$, two sets on which it is easy to define a uniform distribution.

Hence, a **uniform distribution** μ can be defined on \mathcal{A} and an adaptive mixture (à la **Cover**) can be defined on \mathcal{A} as the distribution with density

$$d\mathbf{p}_t(I) = \frac{\exp\left(-\eta \sum_{s=1}^{t-1} \ell(I, \psi_s)\right)}{\int_{\mathcal{A}} \exp\left(-\eta \sum_{s=1}^{t-1} \ell(J, \psi_s)\right) d\mu(J)} d\mu(I)$$

Drawing I_t according to \mathbf{p}_t leads to a **regret** bounded with probability $1 - \delta$ as

$$R_n = O\left(\sqrt{nm \ln \frac{n}{\delta}}\right)$$



In the case of the **continuum of tasks**:

We lack an efficient implementation of the proposed forecaster.
Sequential Monte Carlo methods (particle filtering) may be useful.

In the case of a **finite** number M of tasks:

For the time being, we have not been able to deal efficiently enough with the constraint of picking only m actions out of the N actions to form the vector of simultaneous actions (a Bousquet–Warmuth type constraint).

This is because this constraint is hard to represent with a small hidden state space.

