Sequential Decision Making

Lecture 1 : From Batch to Sequential Learning

Emilie Kaufmann

M2 Data Science, 2021/2022

Who am I ?

- a CNRS researcher in the CRIStAL computer science lab
- a member of the Inria team Scool
  (Sequential COntinual Online Learning)
- Contact : emilie.kaufmann@univ-lille.fr

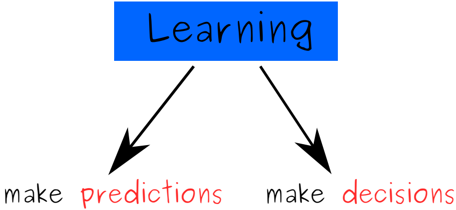
Practical information :

- Evaluation : final homework + paper reading
- Webpage of the class : https://emiliekaufmann.github.io/SDM.html

Sequential Decision Making

Sequential Decision Making <u>vs.</u> Supervised Learning

- sequential learning : the data needs to be processed sequentially (= one by one) online learning



- decisions can influence the data collection process

➜ collect data in a smart way in order to optimize some criterion [e.g., in *Reinforcement Learning* maximize some *cumulated reward*]

Outline of the SDM course

1. Online Learning, Adversarial Bandits
2. Stochastic Mutli-Armed Bandits
3. Contextual Bandits
4. Introduction to Markov Decision Processes (MDP)
5. Solving a known MDP : Dynamic Programming
6. Solving an unknown MDP : RL algorithms
7. Reinforcement Learning with Function Approximation
8. Bandit tools for Reinforcement Learning

Outline of the SDM course

1. Online Learning, Adversarial bandits
2. Stochastic Mutli-Armed Bandits
3. Contextual Bandits
4. Introduction to Markov Decision Processes (MDP)
5. Solving a known MDP : Dynamic Programming
6. Solving an unknown MDP : RL algorithms
7. Reinforcement Learning with Function Approximation
8. Bandit tools for Reinforcement Learning

Supervised Learning

We observe a database containing features ($\boldsymbol{X}$) and labels ($\boldsymbol{Y}$)

$$\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1,\dots n} \in \mathcal{X} \times \mathcal{Y}$$

("labeled examples")

Typically $\mathcal{X} = \mathbb{R}^d$ (features are represented by vectors) and

- $\mathcal{Y} = \{0, 1\}$: binary classification
- $3 \leq |\mathcal{Y}| < \infty$: multi-class classification
- $\mathcal{Y} = \mathbb{R}$: regression

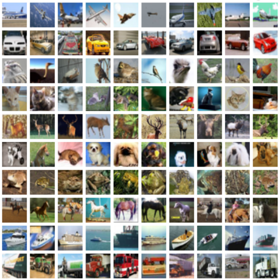
The goal is to build a **predictor** $\hat{g}_n : \mathcal{X} \to \mathcal{Y}$, which is a function that depends on the data $\mathcal{D}_n$, such that for a new observation $(\boldsymbol{X}, \boldsymbol{Y})$

$$\hat{g}_n(\boldsymbol{X}) \simeq \boldsymbol{Y}.$$

➜ smart prediction by means of generalization from examples

Examples

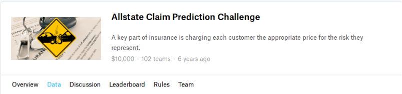
Image classification :

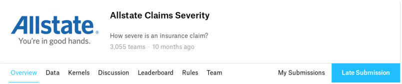


Features : pixel values
Label : type of image

(classification)

Personalized marketing :





Features : customer information
Label : yearly claim

(regression)

Mathematical formalization

Modelling assumption : $\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1,\dots n}$ contains **i.i.d samples** whose distribution is that of a random vector

$$(\boldsymbol{X}, \boldsymbol{Y}) \sim \mathbb{P}.$$

Goal

Given a loss function ℓ, build a predictor with small risk

$$R(g) = \mathbb{E}_{(\boldsymbol{X},\boldsymbol{Y})\sim\mathbb{P}}\left[\ell(g(\boldsymbol{X}), \boldsymbol{Y})\right]$$

A learning algorithm : Empirical risk minimization

Given a class $\mathcal{G}$ of possible predictors, one can compute/approximate

$$\hat{g}_n^{\mathrm{ERM}} \in \underset{g\in\mathcal{G}}{\operatorname{argmax}} \left[\frac{1}{n}\sum_{i=1}^{n} \ell(g(X_i), Y_i)\right]$$

Many supervised learning algorithms

Some of them can be related to an ERM :

- linear regression (Gauss, 1795)
- logistic regression (1950s)
- k-nearest neighbors (1960s)
- Decision Trees (CART, 1984)
- Support Vector Machines (1995)
- Boosting algorithms (Adaboost, 1997)
- Random Forest (2001)
- Neural Networks (1960s-80s, Deep Learning 2010s)

...

Example : Linear Regression

$\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, 1\}$ (binary classification).

Linear regression

$\hat{g}_n(x) = \mathrm{sgn}\left(\langle x | \hat{\theta}_n \rangle\right)$ where

$$\hat{\theta}_n \in \underset{\theta \in \mathbb{R}^d}{\mathrm{argmin}} \sum_{i=1}^{n} \left(Y_i - \langle X_i, \theta \rangle\right)^2$$

Links with the ERM with

- $\mathcal{G} = \{\text{linear functions}\}$
- square loss : $\ell(u, v) = (u - v)^2$

Example : Logistic Regression

$\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, 1\}$ (binary classification).

Logistic regression

$\hat{g}_n(x) = \mathrm{sgn}\left(\langle x|\hat{\theta}_n\rangle\right)$ where

$$\hat{\theta}_n \in \underset{\theta\in\mathbb{R}^d}{\mathrm{argmin}} \sum_{i=1}^{n} \ln\left(1 + e^{-Y_i\langle X_i,\theta\rangle}\right)$$

Links with the ERM with

- $\mathcal{G} = \{\text{linear functions}\}$
- logistic loss : $\ell(u, v) = \ln\left(1 + e^{-uv}\right)$

Batch versus Online

Supervised Learning :
Based on a large database (batch), predict the label of new data (e.g., a test set).

Online Learning :
Data is collected sequentially, and we have to predict their label one-by-one (online), after which the true label is revealed.

Examples :

- predict the value of a stock
- predict electricity consumption for the next day
- predict the behavior of a customer

...

Can existing methods be (efficiently) adapted to the online setting ?

- **Linear regression** : not at first sight...

Closed-form expression for the least-square estimate :

$$\hat{\theta}_n = \left(X_{(n)}^\top X_{(n)}\right)^{-1} X_{(n)}^\top Y_{(n)}$$

where

$$X_{(n)} = \begin{pmatrix} X_1^\top \\ X_2^\top \\ \cdot \\ X_n^\top \end{pmatrix} \in \mathbb{R}^{n \times d} \quad \text{and} \quad Y_{(n)} = \begin{pmatrix} Y_1 \\ Y_2 \\ \cdot \\ Y_n \end{pmatrix} \in \mathbb{R}^n$$

design matrix   vector of labels

➜ need to invert a $d \times d$ matrix depending on $\mathcal{D}_n$ in each round $n+1$

➜ need to store a growing matrix and vector

Can existing methods be (efficiently) adapted to the online setting ?

- **Linear regression** : ... but yes thanks to online least-squares

Another way to write the least-square estimate

$$\hat{\theta}_n = \left(\sum_{t=1}^{n} X_t X_t^\top\right)^{-1} \left(\sum_{t=1}^{n} Y_t X_t\right)$$

Hence

$$\hat{\theta}_{n+1} = \left(\sum_{t=1}^{n} X_t X_t^\top + X_{n+1} X_{n+1}^\top\right)^{-1} \left(\sum_{t=1}^{n} Y_t X_t + Y_{n+1} X_{n+1}\right)$$

→ easy online update thanks to the Sherman-Morisson formula :

$$\left(A + uv^\top\right)^{-1} = A^{-1} - \frac{A^{-1} u v^\top A^{-1}}{1 + v^\top A^{-1} u}$$

→ only requires to store a $d \times d$ matrix and a vector in $\mathbb{R}^d$

Can existing methods be (efficiently) adapted to the online setting ?

- **Logistic regression** : not so clear...

The optimization problem

$$\hat{\theta}_n = \underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} \ln\left(1 + e^{-Y_i \langle X_i, \theta \rangle}\right)$$

has no closed-form solution...

➜ no hope for an explicit only update

➜ online version of the optimization algorithms used ?

Online Learning : general framework

Online Learning

At every time step $t = 1, \dots, T$,

1. observe (features) $x_t \in \mathcal{X}$
2. predict (label) $\hat{y}_t \in \mathcal{Y}$
3. y_t is revealed and we suffer a loss $\ell(y_t, \hat{y}_t)$.

Goal : Minimize the cumulated loss

$$\sum_{t=1}^{T} \ell(y_t, \hat{y}_t)$$

We can compare our performance to :

- ➜ that of the best predictor in a family $\mathcal{G}$
- ➜ that of ("black-box") experts that propose predictions

Learning the Best Predictor Online

Let $\mathcal{G}$ be a class of predictors.

A particular Online Learning problem

A each time step $t = 1, \ldots, T$,

1. choose a predictor $g_t \in \mathcal{G}$
2. observe $x_t \in \mathcal{X}$ and predict $\hat{y}_t = g_t(x_t)$
3. observe y_t and suffer a loss $\ell(y_t; \hat{y}_t)$.

- **Goal :** minimize regret

Regret of a prediction strategy $(g_t)_{t \in \mathbb{N}}$

The regret is the difference between the cumulative loss of the **strategy** and the cumulative loss of the best predictor in $\mathcal{G}$:

$$R_T = \sum_{t=1}^{T} \ell(y_t; \hat{y}_t) - \min_{g \in \mathcal{G}} \sum_{t=1}^{T} \ell(y_t; g(x_t)).$$

Learning the Best Predictor Online

Let $\mathcal{G}$ be a class of predictors.

A particular Online Learning problem

A each time step $t = 1, \dots, T$,

1. choose a predictor $g_t \in \mathcal{G}$ **(based on previous observation)**
2. observe $x_t \in \mathcal{X}$ and predict $\hat{y}_t = g_t(x_t)$
3. observe y_t and suffer a loss $\ell(y_t; \hat{y}_t)$.

- **Goal :** minimize regret

Regret of a prediction strategy $(g_t)_{t\in\mathbb{N}}$

The regret is the difference between the cumulative loss of the **strategy** and the cumulative loss of the best predictor in $\mathcal{G}$:

$$R_T = \sum_{t=1}^{T} \ell(y_t; \hat{y}_t) - \min_{g\in\mathcal{G}} \sum_{t=1}^{T} \ell(y_t; g(x_t)).$$

Example : Online Logistic Regression

Let $\mathcal{G}$ be a class of predictors.

A particular Online Learning problem

A each time step $t = 1, \dots, T$,

1. choose a predictor $g_t \in \mathcal{G}$
2. observe $x_t \in \mathcal{X}$ and predict $\hat{y}_t = g_t(x_t)$
3. observe y_t and suffer a loss $\ell(y_t; \hat{y}_t)$.

Example : $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$ (can be converted to prediction in $\{-1, 1\}$).

- $\mathcal{G}$ is the set of linear functions : $\mathcal{G} = \left\{g(x) = \langle x, \theta \rangle, \theta \in \mathbb{R}^d\right\}$

→ there exists $\theta_t \in \mathbb{R}^d$ such that $g_t(x) = \langle \theta_t, x \rangle$

- ℓ is the logistic loss : $\ell\left(y_t; \hat{y}_t\right) = \ln\left(1 + e^{-y_t \langle \theta_t, x_t \rangle}\right)$

Example : Online Logistic Regression

Let $\mathcal{G}$ be a class of predictors.

A particular Online Learning problem

A each time step $t = 1, \ldots, T$,

1. choose a predictor $g_t \in \mathcal{G}$
2. observe $x_t \in \mathcal{X}$ and predict $\hat{y}_t = g_t(x_t)$
3. observe y_t and suffer a loss $\ell(y_t; \hat{y}_t)$.

Goal : the regret that we should minimize rewrites

$$R_T = \underbrace{\sum_{t=1}^{T} \ln\left(1 + e^{-y_t\langle\theta_t, x_t\rangle}\right)}_{\substack{\text{loss obtained by updating} \\ \text{our predictor in an online fashion}}} - \underbrace{\min_{\theta \in \mathcal{R}^d} \sum_{t=1}^{T} \ln\left(1 + e^{-y_t\langle\theta, x_t\rangle}\right)}_{\substack{\text{loss obtained by the} \\ \text{logistic regression predictor} \\ \text{trained with the whole dataset}}}$$

Example : Online Logistic Regression

$\mathcal{G}$ is a parametric class of predictors : $\mathcal{G} = \{g_\theta, \theta \in \mathbb{R}^d\}$

A particular Online Learning problem

A each time step $t = 1, \dots, T$,

1. choose a vector $\theta_t \in \mathbb{R}^d$
2. a loss function is observed : $\ell_t(\theta) = \ln\left(1 + e^{-y_t\langle\theta, x_t\rangle}\right)$
3. we suffer a loss $\ell_t(\theta_t)$.

Goal : the regret that we should minimize rewrites

$$R_T = \underbrace{\sum_{t=1}^{T} \ln\left(1 + e^{-y_t\langle\theta_t, x_t\rangle}\right)}_{\substack{\text{loss obtained by updating} \\ \text{our predictor in an online fashion}}} - \underbrace{\min_{\theta\in\mathcal{R}^d} \sum_{t=1}^{T} \ln\left(1 + e^{-y_t\langle\theta, x_t\rangle}\right)}_{\substack{\text{loss obtained by the} \\ \text{logistic regression predictor} \\ \text{trained with the whole dataset}}}$$

Example : Online Logistic Regression

$\mathcal{G}$ is a parametric class of predictors : $\mathcal{G} = \{g_\theta, \theta \in \mathbb{R}^d\}$

A particular Online Learning problem

A each time step $t = 1, \dots, T$,

1. choose a vector $\theta_t \in \mathbb{R}^d$
2. a loss function is observed : $\ell_t(\theta) = \ln\left(1 + e^{-y_t\langle\theta, x_t\rangle}\right)$
3. we suffer a loss $\ell_t(\theta_t)$.

Goal : the regret that we should minimize rewrites

$$R_T = \underbrace{\sum_{t=1}^{T} \ell_t(\theta_t)}_{\substack{\text{loss obtained by updating} \\ \text{our predictor in an online fashion}}} - \underbrace{\min_{\theta \in \mathcal{R}^d} \sum_{t=1}^{T} \ell_t(\theta)}_{\substack{\text{loss obtained by the} \\ \text{logistic regression classifier} \\ \text{trained with the whole dataset}}}$$

➜ fits the framework of Online Convex Optimization

Online Convex Optimization

Online Convex Optimization

A each time step $t = 1, \ldots, T$,

1. choose $\theta_t \in \mathcal{K}$, a **convex set**
2. a **convex loss function** $\ell_t(\theta)$ is observed
3. we suffer a loss $\ell_t(\theta_t)$.

Goal : minimize the regret

$$R_T = \underbrace{\sum_{t=1}^{T} \ell_t(\theta_t)}_{\text{loss obtained by updating } \theta \text{ in an online fashion}} - \underbrace{\min_{\theta \in \mathcal{R}^d} \sum_{t=1}^{T} \ell_t(\theta)}_{\text{loss obtained by the best static choice of } \theta}$$

Online Gradient Descent

Online (Projected) Gradient Descent

$$\begin{cases} \theta_1 & \in & \mathcal{K} \\ \theta_{t+1} & = & \Pi_{\mathcal{K}}\left(\theta_t - \eta \nabla \boldsymbol{\ell_t}(\theta_t)\right) \end{cases}$$

where $\Pi_{\mathcal{K}}(x) = \text{argmin}_{u \in \mathcal{K}} ||x - u||$ is the projection on $\mathcal{K}$.

Theorem [e.g., Theorem 3.2 in Bubeck 2015]

Assume $||\nabla \ell_t(\theta)|| \leq L$ and $\mathcal{K} \subseteq B(\theta_1, R)$. Then

$$R_T = \max_{\theta \in \mathcal{K}} \sum_{t=1}^{T} (\ell_t(\theta_t) - \ell_t(\theta)) \leq \frac{R^2}{2\eta} + \frac{\eta L^2 T}{2}$$

Proof : 

Online Gradient Descent

Online (Projected) Gradient Descent

$$\left\{\begin{array}{lcl}\theta_1 & \in & \mathcal{K} \\ \theta_{t+1} & = & \Pi_{\mathcal{K}}\left(\theta_t - \eta\nabla\boldsymbol{\ell_t}(\theta_t)\right)\end{array}\right.$$

where $\Pi_{\mathcal{K}}(x) = \mathrm{argmin}_{u\in\mathcal{K}}||x-u||$ is the projection on $\mathcal{K}$.

Theorem [e.g., Theorem 3.2 in Bubeck 2015]

Assume $||\nabla\ell_t(\theta)|| \leq L$ and $\mathcal{K} \subseteq B(\theta_1, R)$. Then

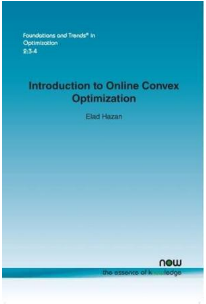
$$R_T = \max_{\theta\in\mathcal{K}} \sum_{t=1}^{T} (\ell_t(\theta_t) - \ell_t(\theta)) \leq \frac{R^2}{2\eta} + \frac{\eta L^2 T}{2}$$

Corollary : for the choice $\eta_T = \frac{R}{L\sqrt{T}}$, we obtain $R_T \leq RL\sqrt{T}$

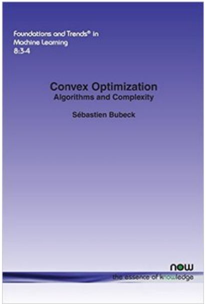
... and beyond

- smaller regret for more regular functions (smooth, strongly convex)
- second order methods (e.g. online version of Newton's algorithm)

References :



[The OCO book]



[Introduction to Online Optimization]

Prediction with expert advice

- we want to sequentially predict some phenomenon (market, weather, energy cunsumption...)
- no probabilistic hypothesis is made about this phenomenon
- we rely on experts (black boxes) $\pm$ good
- we want to be at least as good as the best expert

A prediction game

K experts. Prediction space $\mathcal{Y}$. Loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$.

Prediction with Expert Advice

At each time step $t = 1, \dots, T$,

1. each expert k makes a prediction $z_{k,t} \in \mathcal{Y}$ (that we observe)
2. we predict $\hat{y}_t \in \mathcal{Y}$
3. y_t is revealed and we suffer a loss $\ell(\hat{y}_t, y_t)$.
   Expert k suffers a loss $\ell(z_{k,t}, y_t)$.

Remark : experts may exploit some underlying feature vector $x_t \in \mathcal{X}$

Goal : minimize regret

The regret of a **prediction strategy** is

$$R_T = \underbrace{\sum_{t=1}^{T} \ell(\hat{y}_t, y_t)}_{\substack{\text{cumulative loss} \\ \text{of our prediction strategy}}} - \underbrace{\min_{k \in K} \left[\sum_{t=1}^{T} \ell(z_{k,t}, y_t) \right]}_{\substack{\text{cumulative loss} \\ \text{of the best expert}}}$$

A prediction game

K experts. Prediction space $\mathcal{Y}$. Loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$.

Prediction with Expert Advice

At each time step $t = 1, \dots, T$,

1. each expert k makes a prediction $z_{k,t} \in \mathcal{Y}$ (that we observe)
2. we predict $\hat{y}_t \in \mathcal{Y}$ **(using past observation + current predictions)**
3. y_t is revealed and we suffer a loss $\ell(\hat{y}_t, y_t)$.
   Expert k suffers a loss $\ell(z_{k,t}, y_t)$.

Remark : experts may exploit some underlying feature vector $x_t \in \mathcal{X}$

Goal : minimize regret

The regret of a **prediction strategy** is

$$R_T = \underbrace{\sum_{t=1}^{T} \ell(\hat{y}_t, y_t)}_{\substack{\text{cumulative loss} \\ \text{of our prediction strategy}}} - \underbrace{\min_{k \in K} \left[\sum_{t=1}^{T} \ell(z_{k,t}, y_t) \right]}_{\substack{\text{cumulative loss} \\ \text{of the best expert}}}$$

Weighted (Average) Prediction

Idea

Assign a weight $w_{k,t}$ for expert k at round t and predict a "weighted average" of the experts' predictions.

- **First idea :**

$$\hat{y}_t = \frac{\sum_{k=1}^{K} w_{k,t} z_{k,t}}{\sum_{k=1}^{K} w_{k,t}} = \sum_{k=1}^{K} \left(\frac{w_{k,t}}{\sum_{i=1}^{K} w_{i,t}} \right) z_{k,t}.$$

→ the prediction of experts with large weights matter more

→ we should assign larger weights to "good" experts

Weighted (Average) Prediction

Idea

Assign a weight $w_{k,t}$ for expert k at round t and predict a "weighted average" of the experts' predictions.

- **First idea :**

$$\hat{y}_t = \frac{\sum_{k=1}^{K} w_{k,t} z_{k,t}}{\sum_{k=1}^{K} w_{k,t}} = \sum_{k=1}^{K} \left(\frac{w_{k,t}}{\sum_{i=1}^{K} w_{i,t}} \right) z_{k,t}.$$

➜ the prediction of experts with large weights matter more

➜ we should assign larger weights to "good" experts

⚠ $\hat{y}_t$ might not be in $\mathcal{Y}$ if $\mathcal{Y}$ is not convex...

Weighted (Average) Prediction

Idea

Assign a weight $w_{k,t}$ for expert k at round t and predict a "weighted average" of the experts' predictions.

- **Second idea :**

➜ compute the probability vector $p_t = (p_{1,t}, \ldots, p_{K,t})$ where

$$p_{k,t} := \frac{w_{k,t}}{\sum_{i=1}^K w_{i,t}},$$

➜ select an expert $k_t \sim p_t$, i.e. $\mathbb{P}(k_t = k) = p_{k,t}$

➜ predict $\hat{y}_t = z_{k_t,t} \in \mathcal{Y}$

How to choose the weights ?

The weights should depend on the quality of the expert in the past.

- cumulative loss of expert k at time t : $L_{k,t} = \sum_{s=1}^{t} \ell(z_{k,s}, y_s)$
- "good expert" at time t = expert with a small loss

A natural weight selection

$w_{k,t} = F\left(L_{k,t-1}\right)$ for some decreasing function F.

Typical choice : $F(x) = \exp(-\eta x)$.

➜ leads to an easy multiplicative update

Exponentially Weighted Forecaster

Parameter : $\eta > 0$.
Initialization : for all $k \in \{1, \dots, K\}, w_{k,1} = \frac{1}{K}$.
For $t = 1, \dots, T$

1. Observe the experts' predictions : $(z_{k,t})_{1 \leq k \leq K}$
2. Compute the probability vector $p_t = (p_{1,t}, \dots, p_{K,t})$ where

$$p_{k,t} = \frac{w_{k,t}}{\sum_{i=1}^{K} w_{i,t}} \quad \text{(normalize the weights)}$$

3. Select an expert $k_t \sim p_t$, i.e., $\mathbb{P}(k_t = k) = p_{k,t}$
4. Predict $\hat{y}_t = z_{k_t,t}$ and observe the losses

$$\ell_{k,t} = \ell(z_{k,t}, y_t) \quad \text{for all} \quad k \in \{1, \dots, K\}$$

5. Update the weights : $\forall k \in \{1, \dots, K\},\ w_{k,t+1} = w_{k,t} \exp(-\eta \ell_{k,t})$.

$\mathrm{EWF}(\eta)$ algorithm (or Hedge)

Analysis of EWF

As the algorithm is randomized, we consider the expected regret

$$\mathbb{E}[R_T] = \mathbb{E}\left[\sum_{t=1}^{T} \ell_{k_t,t} - \min_{k\in\{1,\dots,K\}} \sum_{t=1}^{T} \ell_{k,t}\right].$$

Theorem (e.g., Cesa-Bianchi and Lugosi 06)

Assume that

- the losses $\ell_{k,t} = \ell(z_{k,t}, y_t)$ are fixed in advance (oblivious case)
- for all k, t, $0 \leq \ell_{k,t} \leq 1$

Then for all $\eta > 0$ and $T \geq 0$, $\mathrm{EWF}(\eta)$ satisfies

$$\mathbb{E}[R_T] \leq \frac{\ln(K)}{\eta} + \frac{\eta T}{8}.$$

Proof : 

Analysis of EWF

Theorem

Choosing $\eta_T = \sqrt{\frac{8\ln(K)}{T}}$, $\mathrm{EWF}(\eta_T)$ satisfies

$$\mathbb{E}[R_T] \leq \sqrt{\frac{T\ln(K)}{2}}$$

Remarks :

- η can also be chosen without the knowledge of the "horizon" T with similar regret guarantees (up to a constant factor) :

$$\eta_t = \sqrt{\frac{8\ln(K)}{t}}$$

- if $\mathcal{Y}$ is convex, one can replace randomization by actual average, with the same regret guarantees
  - ➜ Exponentially Weighted Average (EWA)

Exponentially Weighted Average

Parameter : $\eta > 0$.
Initialization : for all $k \in \{1, \dots, K\}, w_{k,1} = \frac{1}{K}$.
For $t = 1, \dots, T$

1. Observe the experts' predictions : $(z_{k,t})_{1 \leq k \leq K}$
2. Compute the probability vector $p_t = (p_{1,t}, \dots, p_{K,t})$ where

$$p_{k,t} = \frac{w_{k,t}}{\sum_{i=1}^{K} w_{i,t}} \quad \text{(normalize the weights)}$$

3. Predict $\hat{y}_t = \sum_{k=1}^{K} p_{k,t} z_{k_t,t}$ and observe the losses

$$\ell_{k,t} = \ell(z_{k,t}, y_t) \quad \text{for all} \quad k \in \{1, \dots, K\}$$

4. Update the weights : $\forall k \in \{1, \dots, K\}, \; w_{k,t+1} = w_{k,t} \exp\left(-\eta \ell_{k,t}\right)$.

$\mathrm{EWA}(\eta)$ algorithm

From full information to partial information

Prediction with Expert Advice

At each time step $t = 1, \dots, T$,

1. each expert k makes a prediction $z_{k,t} \in \mathcal{Y}$ (that we observe)
2. we predict $\hat{y}_t \in \mathcal{Y}$
3. y_t is revealed and we suffer a loss $\ell_{k,t} := \ell(\hat{y}_t, y_t)$.

- A full information game :
  we assumed to observe the losses of **all** experts
- Partial information game : we only observe a **subset of the** $(\ell_{k,t})_k$
- Bandit information : we predict $\hat{y}_t = z_{k_t,t}$ and only observe the **loss of the chosen expert**, $\ell_{k_t,t}$

Bandit information : Our prediction strategy has consequences on the loss received but also on the information gathered.

Can we use EWF ?

The Bandit game

At each time step $t = 1, \dots, T$,

1. nature picks a loss vector $\ell_t = (\ell_{1,t}, \dots, \ell_{K,t})$ [*unobserved*]
2. the learner selects an action $k_t \in \{1, \dots, K\}$
3. the learner receives (and observes) the loss of the chosen action $\ell_{k_t,t}$

- **EWF update :**

$$\forall k \in \{1, \dots, K\}, \ \ w_{k,t+1} = w_{k,t} \exp\left(-\eta \ell_{k,t}\right)$$

➜ not possible for $k \neq k_t$...

EWF becomes EXP3

Parameter : $\eta > 0$.
Initialization : for all $k \in \{1, \dots, K\}, w_{k,1} = \frac{1}{K}$.
For $t = 1, \dots, T$

1. Observe the experts' predictions : $(z_{k,t})_{1 \leq k \leq K}$
2. Compute the probability vector $p_t = (p_{1,t}, \dots, p_{K,t})$ where

$$p_{k,t} = \frac{w_{k,t}}{\sum_{i=1}^{K} w_{i,t}} \quad \text{(normalize the weights)}$$

3. Select an expert $k_t \sim p_t$, i.e., $\mathbb{P}(k_t = k) = p_{k,t}$
4. Predict $\hat{y}_t = z_{k_t,t}$ and observe $\ell_{k_t,t}$
5. Compute estimates of the unobserved losses : $\tilde{\ell}_{k,t} = \frac{\ell_{k,t}}{p_{k,t}} \mathbb{1}_{(k_t = k)}$
6. Update the weights : $\forall k, \quad w_{k,t+1} = w_{k,t} \exp\left(-\eta \tilde{\ell}_{k,t}\right)$.

EXP3 (Explore, Exploit and Exponential Weights)

Theoretical guarantees for EXP3

Why does it work ?

$$\tilde{\ell}_{k,t} = \frac{\ell_{k,t}}{p_{k,t}} \mathbb{1}_{(k_t = k)} \quad \text{is an unbiaised estimate of} \quad \ell_{k,t}$$

Theorem [Auer et al., 02]

For the choice

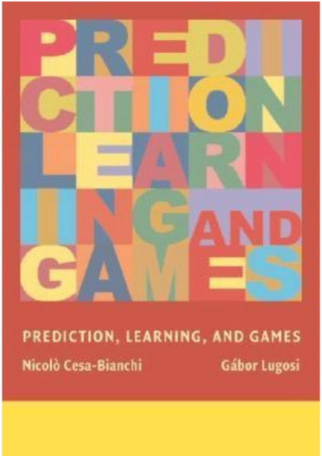
$$\eta_T = \sqrt{\frac{\log(K)}{KT}}$$

EXP3(η_T) satisfies

$$\mathbb{E}[\mathcal{R}_T] \leq \sqrt{2\ln(K)}\sqrt{KT}$$

- ➜ regret in $\sqrt{T}$ for both EWF and EXP3
- ➜ worse dependency in the number of “arms” K for EXP3

Reference



[Prediction, Learning and Games]