La méthode des poids multiplicatifs : un méta-algorithme d'approximation pour l'apprentissage et l'optimisation

Richard Lassaigne
IMJ/Logique mathématique
CNRS-Université Paris Diderot

- Un exemple (parmi beaucoup d'autres) d'application
  Respect **différentiel** de la vie privée

- Présentation unifiée de différents **algorithmes**
  d'**approximation** dans des domaines d'application divers

- Algorithmes **en ligne** de prise de **décison**
  avec **récompenses** (ou coûts)

- La méthode de mise à jour des **poids multiplicatifs**
  Correction de l'algorithme MWU

- Une méthode d'**approximation probabiliste**
  pour casser la barrière de la **complexité**

- Requête du type :

"**Combien** d'enregistrements dans la BD
satisfont la **propriété** P"
(P prédicat **booléen** sur l'univers D)

- Idée : La **probabilité** que le résultat communiqué soit R est presque la même, indépendamment du fait que l'on fournit ou non une **information personnelle**

$$\frac{\mathbf{Prob}[\mathbf{R} \mid \text{univers} = \mathbf{D_I}]}{\mathbf{Prob}[\mathbf{R} \mid \text{univers} = \mathbf{D_{I \pm i}}]} \leq \exp(\varepsilon)$$

pour tous I, i, R

- Définition (Dwork, Mc Sherry, Nisssim et Smith, 2006) :
  Un algorithme **probabiliste** M est ε-**différentiellement privé** si pour tous ensembles de données D, D' **voisins** et tout évènement S

$$Prob[M(D) \in S] \leq \exp(\varepsilon).Prob[M(D') \in S]$$

- Deux ensembles de données D, D' sont **voisins** s'ils ne diffèrent que par un seul élément

- Algorithme :
  Calculer la **vraie réponse** à la requête $q(D)$
  Ajouter un **bruit aléatoire** suivant une distribution satisfaisant

$$\forall z, z' \mid z - z' \mid \leq 1 \Rightarrow Prob[z] \leq \exp(\varepsilon).Prob[z']$$

- Exemple : Distribution de **Laplace**
  $p(z) = \frac{1}{2b} \exp(-\frac{|z|}{b})$ avec $b = \frac{1}{\varepsilon}$

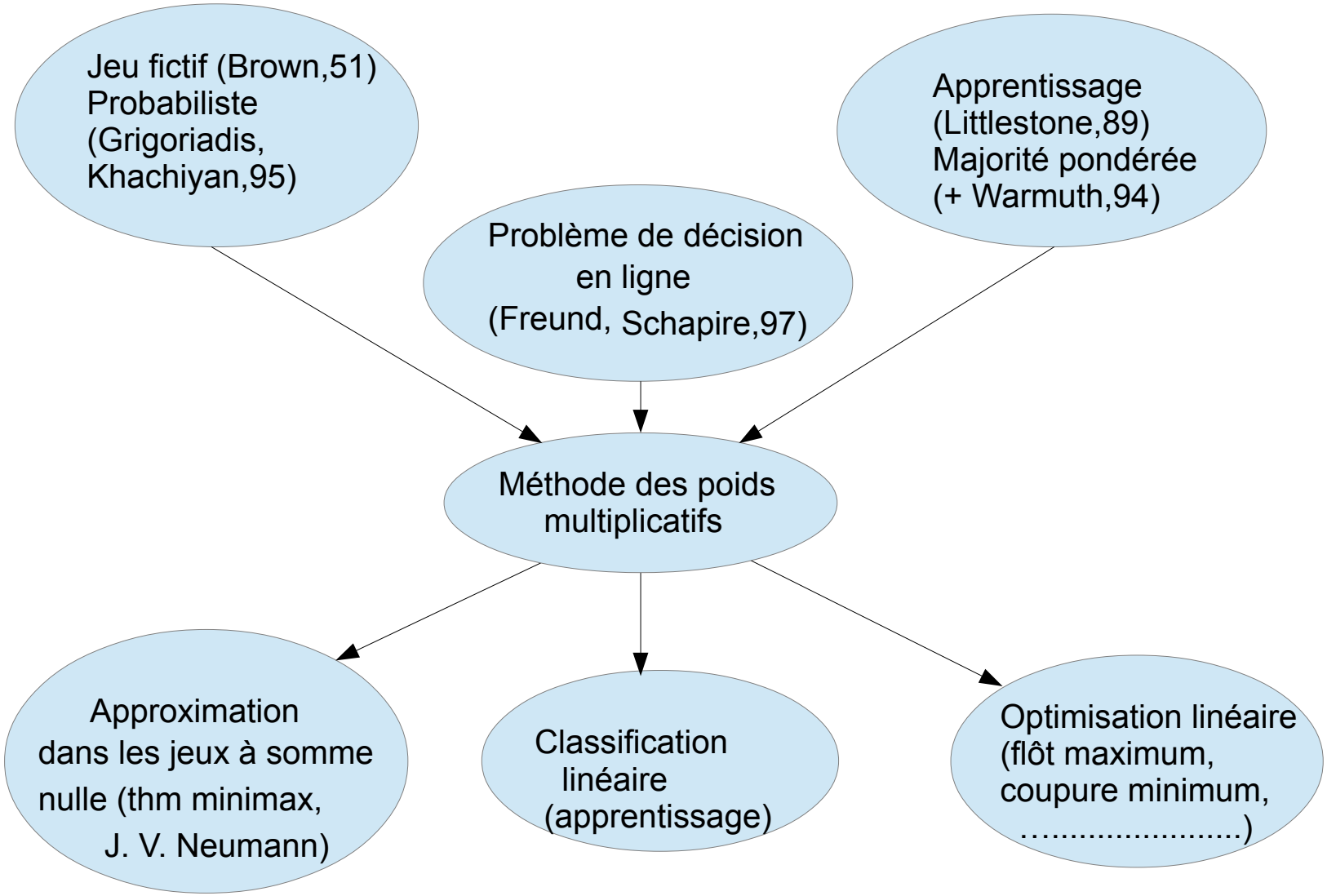
Problème : complexité de l'**erreur** cumulée

- par combinaison de **requêtes**
- dans le contexte **interactif** de requêtes adaptatives

Théorème (Hardt et Rothblum, 2010) : Algorithme (PMW)
Le *Private Multiplicative Weights Mechanism*
est un méchanisme **différentiellement privé**
de réponse à k **requêtes de comptage** dans le
contexte **interactif** avec **erreur** maximum
polynomiale en $\sqrt{\log(k)/n}$ où $n = \mid \mathrm{BD} \mid$
Le temps de réponse à chaque requête est **polynomial**.

Remarque : la dépendance en k et n est **optimale**

Algorithme d'**apprentissage** sous-jacent :
Méthode de mise à jour des **poids multiplicatifs**

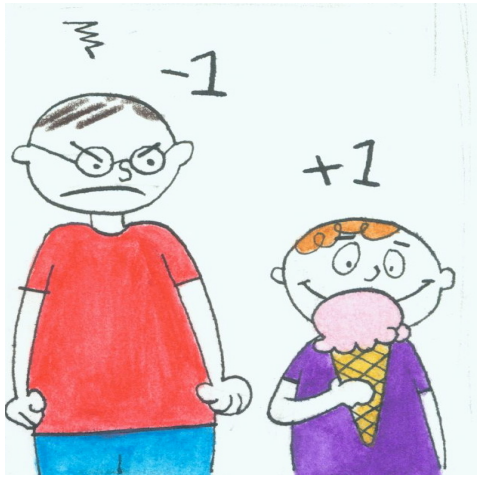

Jeu fictif (Brown,51)
Probabiliste
(Grigoriadis,
Khachiyan,95)
Apprentissage
(Littlestone,89)
Majorité pondérée
(+ Warmuth,94)
Problème de décision
en ligne
(Freund, Schapire,97)
Méthode des poids
multiplicatifs
Approximation
dans les jeux à somme
nulle (thm minimax,
J. V. Neumann)
Classification
linéaire
(apprentissage)
Optimisation linéaire
(flôt maximum,
coupure minimum,
…….)

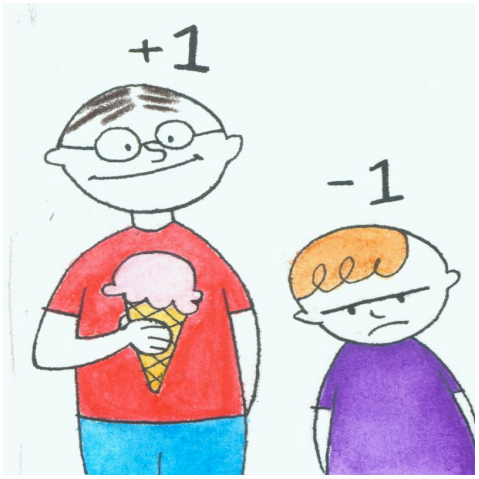
Problèmes de décision en ligne :

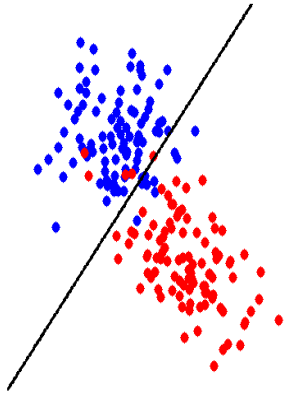
- L'entrée arrive **élément par élément** à chaque étape
- L'algorithme prend une **décision** à chaque étape

Application dans des contextes variés :

- Théorie des **jeux**
- **Apprentissage** automatique
- **Optimisation** linéaire







Modèle de décision en ligne :

Ensemble A de $n \geq 2$ actions, **horizon** de temps $T \geq 1$

A chaque étape $t = 1, 2, \ldots, T$

- un preneur de décision choisit une **distribution** $\mathbf{p_t}$ sur A
- un adversaire choisit un vecteur de **récompenses** $\mathbf{r_t} : A \longrightarrow [-1, 1]$
- une action a_t est choisie suivant la **distribution** $\mathbf{p_t}$ et le preneur de décision reçoit la **récompense** $r_t(a_t)$
- le preneur de décision apprend le vecteur de récompenses $\mathbf{r_t}$

Algorithme en ligne :

- Pour chaque t, la **distribution** $\mathbf{p_t}$ est une fonction des vecteurs $r_1, \ldots, r_{t-1}$ et des actions $a_1, \ldots, a_{t-1}$
- Pour chaque t, le **vecteur** $\mathbf{r_t}$ est une fonction des distributions $p_1, \ldots, p_t$ et des actions $a_1, \ldots, a_{t-1}$

- Récompense **totale espérée** proche de la récompense de la meilleure **suite d'actions** ?
  L'objectif $\sum_{t=1}^{T} max_{a \in A} r_t(a)$ est trop fort

- Objectif de la meilleure **action fixée** : $max_{a \in A} \sum_{t=1}^{T} r_t(a)$
  Mesure de l'efficacité de l'algorithme :
  minimisation du **regret**

$$max_{a \in A} \sum_{t=1}^{T} r_t(a) - \sum_{t=1}^{T} r_t(a_t)$$

  Le regret dans le pire des cas est de $2T$
  (les récompenses $\in [-1, 1]$)
  On considère comme mauvais un regret de cet ordre $\Omega(T)$

Les algorithmes **probabilistes** sont meilleurs
avec une borne inférieure sur le **regret** de l'ordre de $\sqrt{T \ln n}$

- si $n = 2$, on choisit chaque vecteur de récompenses r_t
  indépendamment et de manière **uniforme**
  entre $(1, -1)$ et $(-1, 1)$
  à chaque étape, la récompense espérée est égale à 0
  la récompense totale espérée est aussi nulle
  la récompense totale espérée de la **meilleure action** est
  proportionnelle à $\sqrt{T}$, donc le **regret** est en $O(\sqrt{T})$

- avec n actions, avec un argument similaire, on obtient que
  le **regret espéré** d'un algorithme de décision en ligne
  ne peut croître plus lentement que $b\sqrt{T \ln n}$
  où b est une constante indépendante de n et T

- Théorème : Existence d'un algorithme de décision en ligne qui, pour tout adversaire, a un **regret espéré** $\leq 2\sqrt{T \ \ln \ n}$

- Corollaire : Existence d'un algorithme de décision en ligne qui, pour tout adversaire et tout $\varepsilon > 0$, a un regret espéré **en moyenne** $\leq \varepsilon$ après au plus $(4 \ \ln \ n)/\varepsilon^2$ étapes

Conséquence :

- Le nombre d'étapes nécessaire pour rendre le regret espéré en moyenne aussi petit que l'on veut est **logarithmique** dans le nombre d'actions
- C'est cette forme de garantie qui est utilisée dans les applications

Principes de conception :

- La probabilité de choix d'une action croît avec la **récompense cumulée**
- La probabilité de choix d'une action peu performante doit décroître de manière **exponentielle**

Algorithme **MW** :

Initialisation : $w_1(a) = 1$ pour tout $a \in A$

Pour toute étape $t = 1, 2, \ldots, T$

- utiliser la distribution $\mathbf{p_t} = \mathbf{w_t}/\Gamma_t$ sur les actions où $\Gamma_t = \sum_{a \in A} w_t(a)$ est la somme des poids
- étant donné le vecteur de récompenses $\mathbf{r_t}$, pour toute action $a \in A$, mettre à jour le poids $w_{t+1}(a) = w_t(a)(1 + \eta r_t(a))$

Le paramètre η est compris entre 0 et $1/2$ et est choisi à la fin de la preuve du théorème.

1e étape : Comment la **somme des poids** évolue avec le temps ?

Soit γ_t la récompense espérée à l'étape t :

$$\gamma_t = \sum_{a \in A} p_t(a) r_t(a) = \sum_{a \in A} \frac{w_t(a)}{\Gamma_t} r_t(a)$$

On exprime Γ_{t+1} en fonction de Γ_t :

$$\Gamma_{t+1} = \sum_{a \in A} w_{t+1}(a) = \sum_{a \in A} w_t(a)(1 + \eta r_t(a))$$

$$\Gamma_{t+1} = \Gamma_t(1 + \eta \gamma_t)$$

Borne supérieure à chaque étape : $\Gamma_{t+1} \leq \Gamma_t e^{\eta \gamma_t}$

Borne supérieure finale :

$$\Gamma_{T+1} \leq \Gamma_1 \prod_{t=1}^{T} e^{\eta \gamma_t} = n e^{\eta \sum_{t=1}^{T} \gamma_t}$$

On obtient ainsi une borne inférieure de la **récompense espérée** comme une fonction relativement simple de la quantité Γ_{T+1}

2e étape : S'il existe une **bonne** action fixée, alors son poids montre que la valeur finale Γ_{T+1} est plutôt grande

Soit $opt = \sum_{t=1}^{T} r_t(a^*)$ la récompense cumulée de la **meilleure** action fixée a^* :

$$\Gamma_{T+1} \geq w_{T+1}(a^*) = w_1(a) \prod_{t=1}^{T} (1 + \eta r_t(a^*))$$

On utilise alors : $ln(1+x) \geq x - x^2$ pour $|x| \leq 1/2$
Borne inférieure obtenue ($\eta \leq 1/2$ et $|r_t(a^*)| \leq 1$) :

$$\Gamma_{T+1} \geq \prod_{t=1}^{T} e^{\eta r_t(a^*) - \eta^2 (r_t(a^*))^2}$$

$$\Gamma_{T+1} \geq e^{\eta opt - \eta^2 T}$$

En combinant les deux bornes obtenues :

$$n e^{\eta \sum_{t=1}^{T} \gamma_t} \geq \Gamma_{T+1} \geq e^{\eta opt - \eta^2 T}$$

$$\sum_{t=1}^{T} \gamma_t \geq opt - \eta T - \frac{\ln\, n}{\eta}$$

On choisit le paramètre η de manière à rendre égaux les deux termes d'erreur : $\eta = \sqrt{(\ln\, n)/T}$

Le **regret espéré** de l'algorithme **MW** est alors :

$$opt - \sum_{t=1}^{T} \gamma_t \leq 2\sqrt{T\, \ln\, n}$$

Conséquence : le regret espéré **en moyenne** est

$$\frac{1}{T}(opt - \sum_{t=1}^{T} \gamma_t) \leq 2\sqrt{\frac{\ln\, n}{T}} \leq \varepsilon \text{ si } T \geq \frac{4\, \ln\, n}{\varepsilon^2}$$

Regret espéré :

$$max_{a \in A} \sum_{t=1}^{T} r_t(a) - \sum_{t=1}^{T} r_t(a_t)$$

L'espérance est sur le choix aléatoire de l'**action** à chaque étape

La garantie du théorème peut être étendue à la **meilleure distribution de probabilités** fixée : la meilleure action fixée est au moins aussi bonne que la meilleure distribution fixée

Autre extension utile pour les applications :
l'ensemble des valeurs de **récompenses** est $[-M, M]$
Changement d'échelle pour obtenir un regret **en moyenne** $\leq \varepsilon$:

- diviser toutes les récompenses par M
- appliquer l'algorithme **MW** pour que le regret $\leq \varepsilon/M$
- choisir l'horizon de temps $T = \frac{4M^2(\ln\ n)}{\varepsilon^2}$

- La méthode des poids multiplicatifs fournit un cadre **général** pour divers algorithmes d'**approximation**
- La **mise à jour** des poids dans la distribution de probabilités sur les actions permet de diminuer de manière **exponentielle** l'influence des mauvaises stratégies
- L'horizon de temps ne dépend que de manière **logarithmique** du nombre d'actions
- L'utilisation d'algorithmes probabilistes permet souvent de réduire de manière importante la complexité en **espace**

- S. Arora, E. Hazan and S. Kale. *The Multiplicative Weights Update Method : A Meta-Algorithm and Applications*. Theory of Computing, vol. 8, p.121-164, 2012

- C. Dwork, Frank McSherry, K.Nissim and A. Smith. *Calibrating noise to sensitivity in private data analysis*. 3rd Theory of Cryptography Conference, 2006.

- M. Hardt and G. Rothblum. *A multiplicative weights mechanism for interactive privacy-preserving data analysis*. Proc. Foundations of computer Science, 2010.

- T. Roughgarden. *A second course in Algorithms. Lectures 11 and 12*. Dept. of Computer Science, Stanford, 2016

Merci pour votre attention