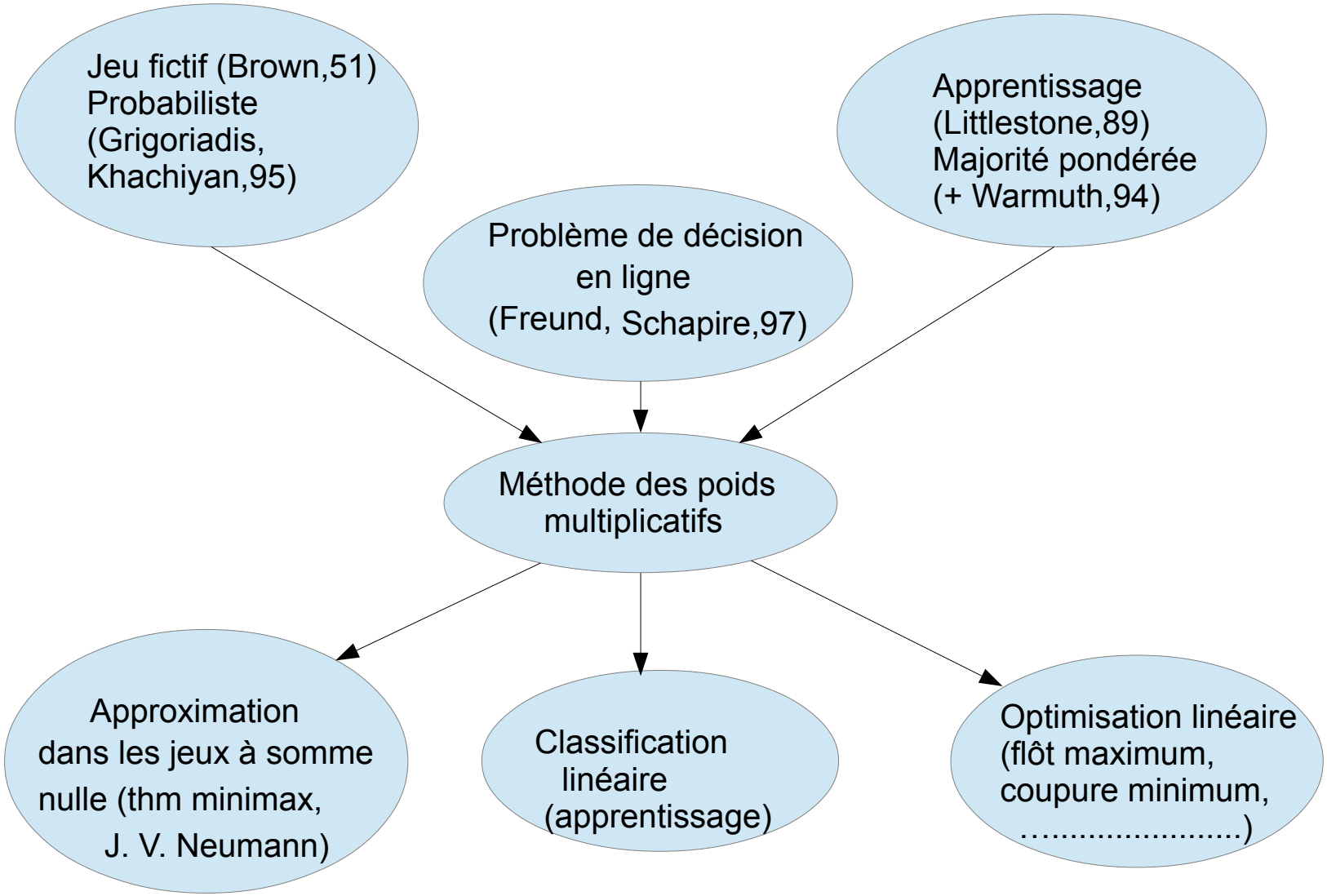
La méthode des poids multiplicatifs : un méta-algorithme d'approximation pour l'apprentissage et l'optimisation

Richard Lassaigne
IMJ/Logique mathématique
CNRS-Université Paris Diderot

- Présentation unifiée de différents **algorithmes** d'**approximation** dans des domaines d'application divers
- Algorithmes **en ligne** de prise de **décison** avec **récompenses** (ou coûts)
- Version constructive de la **dualité** en programmation linéaire et du théorème **minimax** dans les jeux à somme nulle
- Algorithmes d'**apprentissage** (exemple : **classification linéaire**)
- Algorithmes d'**approximation** de programmes linéaires
- Efficacité et nécessité des algorithmes **probabilistes**


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,
…….)

Quelques précurseurs

J.V. Neumann

G.W. Brown

Yoav Freund

R.E. Schapire

- Problématique de la prise de décision **en ligne**
  Notion de **regret** par rapport à la meilleure suite d'actions

- Méthode de mise à jour des **poids multiplicatifs**
  Nécessité d'un algorithme **probabiliste**
  pour une majoration efficace du **regret**

- Application aux **jeux** à somme nulle

- Un exemple d'algorithme d'**apprentissage**

- Approximation pour la **programmation linéaire**
  (problèmes de **flot maximum, couverture**, etc...)

Quelques acteurs d'une présentation unifiée

Sanjeev Arora

Satyen Kale

Christos Papadimitriou

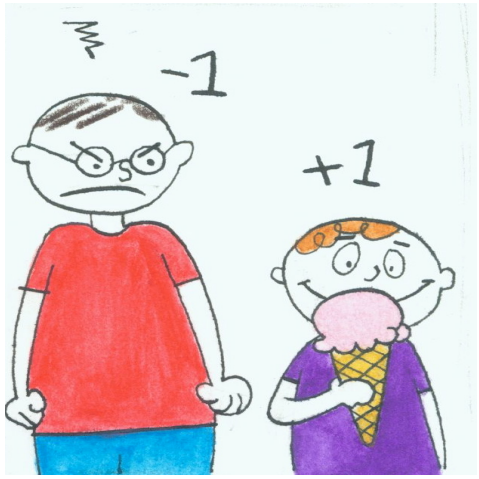
Tim Roughgarden

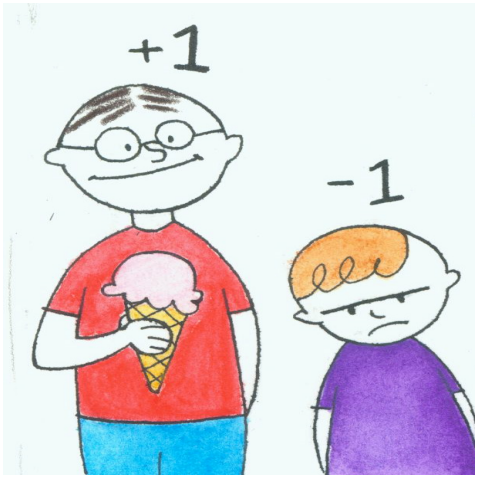
Problèmes de décision en ligne :

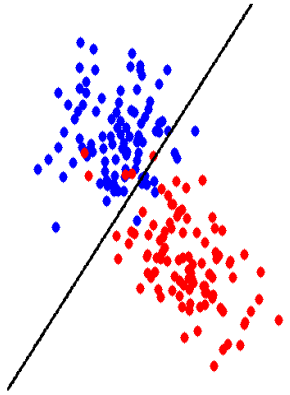
- L'entrée arrive **élément par élément** à chaque étape
- L'algorithme prend une **décision** à chaque étape

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 l'ordre de $2T$
  (les récompenses $\in [-1, 1]$)
  On considère comme mauvais un regret de cet ordre $\Omega(T)$

- Des algorithmes simples et naturels d'**apprentissage** remplissent cet objectif

- Cet objectif n'est pas évident (impossible pour un algorithme **déterministe**)

- Remplir cet objectif est suffisant pour de nombreuses **applications**

Algorithme de décision en ligne naturel **suivre le leader** :
à l'étape t, choisir l'action a avec la récompense cumulée $\sum_{u=1}^{t-1} r_u(a)$ maximum

- Les algorithmes **déterministes** ne sont pas bons :
  à chaque étape t, l'algorithme choisit une **seule action** a_t
  une stratégie évidente pour l'**adversaire**
  $r_t(a_t) = 0$ et $r_t(a) = 1$ pour toute autre action
  récompense totale de l'algorithme $= 0$
  récompense totale de la meilleure action $\geq T(1 - 1/n)$
  si $n = 2$, le regret dans le pire des cas est $\geq T/2$

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 $\sqrt{T}$

- 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 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 :

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

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 $T = \frac{4M^2(\ln\ n)}{\varepsilon^2}$

Spécification : **matrice** $\mathbf{A} = (a_{ij})$ de taille $m \times n$ à valeurs réelles

Joueur 1 : joue une **ligne** ; Joueur 2 : joue une **colonne**

a_{ij} gain du joueur 1 lorsque 1 joue la ligne i et 2 la colonne j

Dans ce cas, le gain du joueur 2 est $-a_{ij}$ (jeu à somme nulle)

Exemple (simple, mais représentatif) :

	Pierre	Papier	Ciseaux
Pierre	0	-1	1
Papier	1	0	-1
Ciseaux	-1	1	0

Gain espéré du joueur 2 (stratégies **mixtes**) :

la stratégie (distribution) sur les colonnes est $\mathbf{x}$

la stratégie (distribution) sur les lignes est $\mathbf{y}$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} Prob[sortie(i,j)]a_{ij} = \sum_{i=1}^{m}\sum_{j=1}^{n} Prob[i\ choisi]Prob[j\ choisi]a_{ij} = \mathbf{x}^T\mathbf{A}\mathbf{y}$$

Préférez-vous engager une stratégie mixte

avant ou **après** l'autre joueur ?

Intuitivement : le 2e joueur peut s'adapter à celle du 1er

Théorème **minimax** (John von Neumann, 1928) :
Pour tout jeu à 2 joueurs à somme nulle $\mathbf{A}$

$$\max_{\mathbf{x}}(\min_{\mathbf{y}}\mathbf{x}^T\mathbf{A}\mathbf{y}) = \min_{\mathbf{y}}(\max_{\mathbf{x}}\mathbf{x}^T\mathbf{A}\mathbf{y})$$

La preuve originelle de von Neumann utilisait un résultat de **point fixe** de Brouwer.
Dans les années 1940, il a donné une 2e preuve utilisant des arguments équivalents à la **dualité**.

Soit un jeu à somme nulle $\mathbf{A}$ et un paramètre d'**erreur** $\varepsilon > 0$

A chaque étape $t = 1, 2, \ldots, T = \frac{4M^2(\ln\, n)}{\varepsilon^2}$:

- le joueur 1 choisit une stratégie **mixte** $\mathbf{p}_t$
  suivant l'algorithme **MW** (les actions sont les lignes)
- le joueur 2 répond de manière **optimale**
  avec la stratégie **déterministe** $\mathbf{q}_t$
- si le joueur 2 choisit la colonne j, alors prendre comme
  récompense $r_t(i) = a_{ij}$ pour toute ligne i
  et alimenter l'algorithme **MW** avec le vecteur $\mathbf{r}_t$

Résultat 1 :

Le gain espéré **en moyenne** du joueur 2 est **au plus** $\max_{\mathbf{p}}(\min_{\mathbf{q}} \mathbf{p}^T \mathbf{A}\mathbf{q})$

Soit $\hat{\mathbf{p}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{p}_t$ la stratégie **ligne** en moyenne
et $\mathbf{q}^*$ une réponse optimale à $\hat{\mathbf{p}}$

$$\max_{\mathbf{p}}(\min_{\mathbf{q}} \mathbf{p}^T \mathbf{A}\mathbf{q}) \geq \min_{\mathbf{q}} \hat{\mathbf{p}}^T \mathbf{A}\mathbf{q} = \hat{\mathbf{p}}^T \mathbf{A}\mathbf{q}^* = \frac{1}{T}\sum_{t=1}^{T} (\mathbf{p}_t)^T \mathbf{A}\mathbf{q}^*$$

$$\max_{\mathbf{p}}(\min_{\mathbf{q}} \mathbf{p}^T \mathbf{A}\mathbf{q}) \geq \frac{1}{T}\sum_{t=1}^{T} (\mathbf{p}_t)^T \mathbf{A}\mathbf{q}_t$$

Remarque : $\mathbf{q}_t$ est une réponse **optimale** à $\mathbf{p}_t$ pour chaque t

Résultat 2 :

Le gain espéré **en moyenne** du joueur 2 est **au moins**
$\min_{\mathbf{q}}(\max_{\mathbf{p}} \mathbf{p}^T \mathbf{A}\mathbf{q}) - \varepsilon$

Soit $\hat{\mathbf{q}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{q}_t$ la stratégie **colonne** en moyenne
La garantie de l'algorithme **MW** assure que le gain espéré
en moyenne du joueur 2 est au moins, à ε près,
le gain espéré en moyenne obtenu par une stratégie **mixte fixée** $\mathbf{p}$

$$\frac{1}{T}\sum_{t=1}^{T}(\mathbf{p}_t)^T \mathbf{A}\mathbf{q}_t \geq \max_p(\frac{1}{T}\sum_{t=1}^{T}\mathbf{p}^T \mathbf{A}\mathbf{q}_t) - \varepsilon = \max_p \mathbf{p}^T \mathbf{A}\hat{\mathbf{q}} - \varepsilon$$

$$\frac{1}{T}\sum_{t=1}^{T}(\mathbf{p}_t)^T \mathbf{A}\mathbf{q}_t \geq \min_{\mathbf{q}}(\max_{\mathbf{p}} \mathbf{p}^T \mathbf{A}\mathbf{q}) - \varepsilon$$

- Entrée : m données étiquetées positives $\mathbf{x}^1, \ldots, \mathbf{x}^m \in \mathbb{R}^d$
  m' données étiquetées négatives $\mathbf{y}^1, \ldots, \mathbf{x}^{m'} \in \mathbb{R}^d$

- Sortie : Une fonction **linéaire** $h : \mathbb{R}^d \longrightarrow \mathbb{R}$ t. q.
  $h(\mathbf{z}) = \sum_{j=1}^{d} a_j z_j + b$, $h(\mathbf{x}^i) > 0$ et $h(\mathbf{y}^i) < 0$

Pré-calcul : Mise en forme du problème

- on force $b = 0$ en ajoutant une $d+1$-ième variable avec $a_{d+1} = b$
  tout donnée a une nouvelle cordonnée égale à 1
- on multiplie les coordonnées des données négatives $\mathbf{y}^i$ par -1
  le système de contraintes s'écrit alors $h(\mathbf{x}^i) > 0$ et $h(\mathbf{y}^i) > 0$
- on peut demander que tous les coefficients a_j soient positifs
  en faisant 2 copies $\mathbf{x}^i_j$ et $-\mathbf{x}^i_j$ de chaque coordonnée $\mathbf{x}^i_j$ et
  en interprétant les 2 coefficients correspondants a'_j et a''_j
  avec $a_j = a'_j - a''_j$ (on double la dimension $d+1$)

Oracle : le problème est **faisable** avec une **marge** $\varepsilon > 0$

Il existe un vecteur de coefficients $\mathbf{a}^* \in \mathbb{R}^d$ t.q.
$\sum_{j=1}^{d} a_j^* = 1$ et $\sum_{j=1}^{d} a_j^* x_j^i > \varepsilon$ pour toute donnée $\mathbf{x}^i$

Soit M t.q. $|x_j^i| \leq M$ pour tout i, j et
$A = \{1, 2, \ldots, d\}$ l'ensemble des actions.

Algorithme : Pour $t = 1, 2, \ldots, T = \frac{4M^2(\ln\ n)}{\varepsilon^2}$

- utiliser l'algorithme **MW** pour engendrer une **distribution de probabilités** $\mathbf{a}^t \in \mathbb{R}^d$
- si $\sum_{j=1}^{d} a_j^t x_j^i > 0$ pour tout $\mathbf{x}^i$, alors stop et retourner $\mathbf{a}^t$
- sinon choisir une **donnée** $\mathbf{x}^i$ avec $\sum_{j=1}^{d} a_j^t x_j^i < 0$ et définir un vecteur de **récompenses** $\mathbf{r}^t$ avec $r^t(j) = x_j^i$
- fournir le vecteur $\mathbf{r}^t$ à l'algorithme **MW**

Analyse :

- Les vecteurs de récompenses sont tels que, à chaque étape t,
  le produit scalaire de $\mathbf{a}^t$ et $\mathbf{r}^t$ soit négatif .
  En considérant $\mathbf{a}^t$ comme une distribution de probabilités,
  la **récompense espérée** est négative à chaque étape et
  la récompense espérée **en moyenne** est au plus 0
- D'après la supposition, il existe une distribution de
  probabilités $\mathbf{a}^*$ sur A t. q., à chaque étape t,
  la **récompense espérée** en jouant $\mathbf{a}^*$ aurait été
  $\sum_{j=1}^{d} a_j^* r^t(j) \geq \min_{i=1}^{m} \sum_{j=1}^{d} a_j^* x_j^i > \varepsilon$
- Tant que l'algorithme n'a pas trouvé de solution,
  le **regret en moyenne** de la procédure **MW** est $> \varepsilon$
  L'algorithme **MW** garantit qu'après $T = \frac{4M^2(\ln\ n)}{\varepsilon^2}$ étapes,
  le regret en moyenne est **au plus** ε

Forme générale : $\exists \mathbf{x} \in \mathcal{P} \quad \mathbf{Ax} \geq \mathbf{b}$ (1)
où $\mathbf{A} \in \mathbb{R}^{m \times n}$ est une matrice et $\mathcal{P}$ est un ensemble **convexe** de $\mathbb{R}^n$

- l'ensemble $\mathcal{P}$ représente les contraintes **faciles** à satisfaire
- la matrice $\mathbf{A}$ les contraintes **difficiles** à satisfaire

Objectif : Un algorithme qui, étant donné un paramètre d'erreur ε,

- soit résout le problème avec une erreur **additive** ε,
  c.à.d. trouve un $\mathbf{x} \in \mathcal{P}$ t.q. pour tout i, $\mathbf{A}_i\mathbf{x} \geq b_i - \varepsilon$,
- ou sinon prouve que le système n'est **pas résoluble**

On suppose l'existence d'un algorithme, appelé **Oracle**, qui,
étant donné un vecteur de **probabilités** $\mathbf{p}$ sur les m contraintes,
résout le problème : $\exists \mathbf{x} \in \mathcal{P} \quad \mathbf{p}^T\mathbf{Ax} \geq \mathbf{p}^T\mathbf{b}$ (2)

Une manière d'implanter cette procédure est de **maximiser**
$\mathbf{p}^T\mathbf{Ax}$ sur $\mathbf{x} \in \mathcal{P}$ (résoudre **une contrainte** plutôt que m)

Remarque :

- si le problème (1) a une solution $\mathbf{x}^*$, alors la même solution satisfait aussi (2) pour **tout** vecteur de **probabilités** $\mathbf{p}$
- s'il existe un vecteur $\mathbf{p}$ t.q. **aucun** $\mathbf{x} \in \mathcal{P}$ ne satisfait (2), alors le problème originel n'est **pas résoluble**

Oracle M-borné : Algorithme qui, étant donné un vecteur $\mathbf{p}$ résout le problème (2) avec pour tout i, $\mathbf{A}_i\mathbf{x} - b_i \in [-M, M]$ ($M \geq 0$)

Théorème : Soit $\varepsilon > 0$ un paramètre d'erreur.
On suppose qu'il existe un oracle M-borné et que $M \geq \varepsilon/2$.
Alors il existe un algorithme

- qui soit **trouve** un $\mathbf{x} \in \mathcal{P}$ t.q. pour tout i, $\mathbf{A}_i\mathbf{x} \geq b_i - \varepsilon$,
- soit conclut que le système n'est **pas résoluble**.

L'algorithme fait $O(\frac{M^2(\log(m)}{\varepsilon^2})$ **appels** à l'oracle,
avec un temps additionnel de $O(m)$ par appel

- 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

- Y. Freund and R. E. Schapire. *A decision-theoretic generalization of on-line learning and an application to boosting*. JCSS , 55(1), p.119–139, 1997.

- N. Garg and J. Koneman. *Faster and simpler algorithms for multicommodity flow and other fractional packing problems*. SIAM Journal of Computing 37, p.630–652, 2007

- M. Grigoriadis and L. Khachiyan. *A sublinear-time randomized approximation algorithm for matrix games*. Operations Research Letters, vol. 18, p. 53–58, 1995.

- N. Littlestone and M. K. Warmuth. *The weighted majority algorithm*. Information and Computation, 108(2), p.212–261, 1994.

- S. A. Plotkin, D. B. Shmoys, and Eva Tardos. *Fast approximation algorithm for fractional packing and covering problems*. Proc. 32nd IEEE Symposium on Foundations of Computer Science, p.495-504, 1991

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

- U. Vazirani. *Combinatorial Algorithms and Data Structures Lectures 4 and 5*. Dept. of Computer Science, Berkeley, 2011

Merci pour votre attention