

L03 (partb)

# Online Optimization and Learning: Applications

50.579 Optimization for Machine Learning

Ioannis Panageas

ISTD, SUTD

# Multiplicative Weights Update (recap)

**Algorithm (MWUA).** *We define the following algorithm:*

1. Initialize  $w_i^0 = 1$  for all  $i \in [n]$ .
2. **For**  $t=1 \dots T$  **do**
3.     **Choose** action  $i$  with probability proportional to  $w_i^{t-1}$ .
4.     **For** each action  $i$  **do**
5.          $w_i^t = (1 - \epsilon)^{c_i^t} w_i^{t-1}$ .
6.     **End For**
7. **End For**

Remarks:

- $\epsilon := \sqrt{\frac{\log n}{T}}$
- We choose  $i$  with probability  $p_i^t = \frac{w_i^{t-1}}{\sum_j w_j^{t-1}}$ .
- $c_i^t$  is the cost of action  $i$  at time  $t$  chosen by the adversary.

# MWUA general setting

**Theorem (MWUA).** Let  $OPT = \min_i \sum_{t=1}^T c_i^t$

$$\mathbb{E}[cost_{MWUA}] \leq OPT + \epsilon T + \frac{\log n}{\epsilon}.$$

*Proof.* Let's define the **potential** function  $\phi_t = \sum_i w_i^t$ .

Let best action in hindsight be  $i^*$  then,  
we have

$$\phi_T > w_{i^*}^T = (1 - \epsilon)^{OPT}.$$

$$\text{Now } \phi_{t+1} = \sum w_i^{t+1} = \sum w_i^t (1 - \epsilon)^{c_i^t}$$

# MWUA general setting

**Theorem (MWUA).** Let  $OPT = \min_i \sum_{t=1}^T c_i^t$

$$\mathbb{E}[\text{cost}_{MWUA}] \leq OPT + \epsilon T + \frac{\log n}{\epsilon}.$$

*Proof.* Let's define the **potential** function  $\phi_t = \sum_i w_i^t$ .

Let best action in hindsight be  $i^*$  then,  
we have

$$\phi_T > w_{i^*}^T = (1 - \epsilon)^{OPT}.$$

$$\begin{aligned} \text{Now } \phi_{t+1} &= \sum w_i^{t+1} = \sum w_i^t (1 - \epsilon)^{c_i^t} \\ &= \sum \phi_t p_i^{t+1} (1 - \epsilon)^{c_i^t} \end{aligned}$$

# MWUA general setting

**Theorem (MWUA).** Let  $OPT = \min_i \sum_{t=1}^T c_i^t$

$$\mathbb{E}[\text{cost}_{MWUA}] \leq OPT + \epsilon T + \frac{\log n}{\epsilon}.$$

*Proof.* Let's define the **potential** function  $\phi_t = \sum_i w_i^t$ .

Let best action in hindsight be  $i^*$  then,  
we have

$$\phi_T > w_{i^*}^T = (1 - \epsilon)^{OPT}.$$

$$\begin{aligned} \text{Now } \phi_{t+1} &= \sum w_i^{t+1} = \sum w_i^t (1 - \epsilon)^{c_i^t} \\ &= \sum \phi_t p_i^{t+1} (1 - \epsilon)^{c_i^t} \\ &= \phi_t \sum p_i^{t+1} (1 - \epsilon)^{c_i^t} \end{aligned}$$

# MWUA general setting

*Proof cont.* Therefore

$$\phi_{t+1} = \phi_t \sum p_i^{t+1} (1 - \epsilon)^{c_i^t}$$

# MWUA general setting

*Proof cont.* Therefore

$$\begin{aligned}\phi_{t+1} &= \phi_t \sum p_i^{t+1} (1 - \epsilon)^{c_i^t} \\ &\leq \phi_t \sum p_i^{t+1} (1 - \epsilon \cdot c_i^t)\end{aligned}$$

Note  $(1 - \epsilon)^x \leq 1 - \epsilon x$  for  $x \in [0, 1], \epsilon \in [0, 1/2]$ .

# MWUA general setting

*Proof cont.* Therefore

$$\begin{aligned}\phi_{t+1} &= \phi_t \sum p_i^{t+1} (1 - \epsilon)^{c_i^t} \\ &\leq \phi_t \sum p_i^{t+1} (1 - \epsilon \cdot c_i^t) \\ &= \phi_t (1 - \epsilon \cdot \mathbb{E}[\text{cost}(t)_{\text{MWUA}}])\end{aligned}$$

Note  $(1 - \epsilon)^x \leq 1 - \epsilon x$  for  $x \in [0, 1], \epsilon \in [0, 1/2]$ .



# MWUA general setting

*Proof cont.* Therefore

$$\begin{aligned}\phi_{t+1} &= \phi_t \sum p_i^{t+1} (1 - \epsilon)^{c_i^t} \\ &\leq \phi_t \sum p_i^{t+1} (1 - \epsilon \cdot c_i^t) \\ &= \phi_t (1 - \epsilon \cdot \mathbb{E}[\text{cost}(t)_{\text{MWUA}}]) \\ &\leq \phi_t e^{-\epsilon \mathbb{E}[\text{cost}(t)_{\text{MWUA}}]}\end{aligned}$$

Note  $(1 - \epsilon)^x \leq 1 - \epsilon x$  for  $x \in [0, 1], \epsilon \in [0, 1/2]$ .

# MWUA general setting

*Proof cont.* Therefore

$$\begin{aligned}\phi_{t+1} &= \phi_t \sum p_i^{t+1} (1 - \epsilon)^{c_i^t} \\ &\leq \phi_t \sum p_i^{t+1} (1 - \epsilon \cdot c_i^t) \\ &= \phi_t (1 - \epsilon \cdot \mathbb{E}[\text{cost}(t)_{\text{MWUA}}]) \\ &\leq \phi_t e^{-\epsilon \mathbb{E}[\text{cost}(t)_{\text{MWUA}}]}\end{aligned}$$

Telescopic product gives

$$\phi_T \leq \phi_1 e^{-\epsilon \mathbb{E}[\text{cost}_{\text{MWUA}}]}.$$

Therefore  $(1-\epsilon)^{OPT} \leq e^{-\epsilon \mathbb{E}[\text{cost}_{\text{MWUA}}]} n$ , or  $OPT(-\epsilon - \epsilon^2) \leq \log n - \epsilon \mathbb{E}[\text{cost}_{\text{MWUA}}]$ .

# MWUA general setting

*Proof cont.* Therefore

Plugging in  $\epsilon = \sqrt{\frac{\log n}{T}}$ , gives  $\frac{1}{T}(\mathbb{E}[\text{cost}_{\text{MWUA}}] - \text{OPT}) \leq 2\sqrt{\frac{\log n}{T}}$

$$\leq \phi_t e^{-\epsilon \mathbb{E}[\text{cost}(t)_{\text{MWUA}}]}$$

Telescopic product gives

$$\phi_T \leq \phi_1 e^{-\epsilon \mathbb{E}[\text{cost}_{\text{MWUA}}]}.$$

Therefore  $(1-\epsilon)^{\text{OPT}} \leq e^{-\epsilon \mathbb{E}[\text{cost}_{\text{MWUA}}]} n$ , or  $\text{OPT}(-\epsilon - \epsilon^2) \leq \log n - \epsilon \mathbb{E}[\text{cost}_{\text{MWUA}}]$ .

# Solving Linear Programs

**Problem (Linear Program).** Suppose we are given a linear program in the standard form

$$\begin{aligned} Ax &\geq b \\ \text{s.t. } x &\geq 0. \end{aligned}$$

**Goal (Check feasibility).** Compute a vector  $x^* \geq 0$  such that for some  $\epsilon > 0$  we get

$$\alpha_i^\top x^* \geq b_i - \epsilon, \text{ for all } i.$$

**Oracle access:** Given a vector  $c$  and scalar  $d$ , does there exist a  $x \geq 0$  such that  $c^\top x \geq d$ .

Remark: Using the above and **binary search**, you can solve any linear program!

# Solving Linear Programs

**Problem (Linear Program).** Suppose we are given a linear program in the standard form

$$\begin{aligned} Ax &\geq b \\ \text{s.t. } x &\geq 0. \end{aligned}$$

**Goal (Check feasibility).** Compute a vector  $x^* \geq 0$  such that for some  $\epsilon > 0$  we get

$$\alpha_i^\top x^* \geq b_i - \epsilon, \text{ for all } i.$$

**Oracle access:** Given a vector  $c$  and scalar  $d$ , does there exist a  $x \geq 0$  such that  $c^\top x \geq d$ .

Remark: Using the above and **binary search**, you can solve any linear program!

Use MWUA, what are the actions/costs?

# Solving Linear Programs

**Setting.** Consider every *constraint*  $a_i^\top x - b_i$  as an *action*.

- Choose  $c_i(x) = \frac{a_i^\top x - b_i}{\rho}$  with  $\rho$  chosen so that  $|c_i| \leq 1$ .
- Initialization  $w_i^0 = 1$  (uniform distribution).
- For each  $t = 1, \dots, T$ , ask oracle if there exists a point  $x \geq 0$  such that  $c^\top x \geq d$  where

$$c = \sum p_i^t a_i, \quad d = \sum p_i^t b_i.$$

# Solving Linear Programs

**Setting.** Consider every *constraint*  $a_i^\top x - b_i$  as an *action*.

- Choose  $c_i(x) = \frac{a_i^\top x - b_i}{\rho}$  with  $\rho$  chosen so that  $|c_i| \leq 1$ .
- Initialization  $w_i^0 = 1$  (uniform distribution).
- For each  $t = 1, \dots, T$ , ask oracle if there exists a point  $x \geq 0$  such that  $c^\top x \geq d$  where

$$c = \sum p_i^t a_i, \quad d = \sum p_i^t b_i.$$

If the answer is no, linear problem *infeasible*!

# Solving Linear Programs

**Setting.** Consider every *constraint*  $a_i^\top x - b_i$  as an *action*.

- Choose  $c_i(x) = \frac{a_i^\top x - b_i}{\rho}$  with  $\rho$  chosen so that  $|c_i| \leq 1$ .
- Initialization  $w_i^0 = 1$  (uniform distribution).
- For each  $t = 1, \dots, T$ , ask oracle if there exists a point  $x \geq 0$  such that  $c^\top x \geq d$  where

$$c = \sum p_i^t a_i, \quad d = \sum p_i^t b_i.$$

If the answer is no, linear problem *infeasible*!

If the answer is yes (returns a  $x^t$ ), each action *suffers cost*  $c_i^t = c_i(x^t)$ .



# Solving Linear Programs

From our theorem we get that

$$0 \leq \sum_t \sum_i p_i^t (a_i^\top x_i^t - b_i) \leq \sum_t \sum_i p_i^* (a_i^\top x_i^t - b_i) + 2\rho \sqrt{\frac{\log m}{T}}.$$

where  $p^*$  is the optimal hindsight. But the RHS is at most (for all  $i$ )

$$\sum_t a_i^\top x_i^t - b_i + 2\rho \sqrt{\frac{\log m}{T}} = a_i^\top \sum_t x_i^t - T b_i + 2\rho \sqrt{\frac{\log m}{T}}.$$

# Solving Linear Programs

From our theorem we get that

$$0 \leq \sum_t \sum_i p_i^t (a_i^\top x_i^t - b_i) \leq \sum_t \sum_i p_i^* (a_i^\top x_i^t - b_i) + 2\rho \sqrt{\frac{\log m}{T}}.$$

where  $p^*$  is the optimal hindsight. But the RHS is at most (for all  $i$ )

$$\sum_t a_i^\top x_i^t - b_i + 2\rho \sqrt{\frac{\log m}{T}} = a_i^\top \sum_t x_i^t - T b_i + 2\rho \sqrt{\frac{\log m}{T}}.$$

Therefore, by choosing  $T = \frac{4\rho^2 \log m}{\epsilon^2}$ ,  $\tilde{x} = \frac{1}{T} \sum_t x^t$  we get that

$$a_i^\top \tilde{x} - b_i + \epsilon \geq 0 \text{ for all } i.$$

# MWUA and Zero-sum games

**Definition.** Consider a matrix  $A$  (called *payoff*).  $A_{ij}$  denotes the amount of money player  $x$  pays to player  $y$ . Example (Rock-Paper-Scissors):

$$A = \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}.$$

# MWUA and Zero-sum games

**Definition.** Consider a matrix  $A$  (called *payoff*).  $A_{ij}$  denotes the amount of money player  $x$  pays to player  $y$ . Example (Rock-Paper-Scissors):

$$A = \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}.$$

**Definition (Nash Equilibrium).** A vector  $(x^*, y^*)$  is called a NE if

$$x^{*\top} A y^* \geq x^{*\top} A \tilde{y} \text{ for all } \tilde{y} \in \Delta \text{ and } x^{*\top} A y^* \leq \tilde{x}^\top A y^* \text{ for all } \tilde{x} \in \Delta.$$

# MWUA and Zero-sum games

**Definition.** Consider a matrix  $A$  (called *payoff*).  $A_{ij}$  denotes the amount of money player  $x$  pays to player  $y$ . Example (Rock-Paper-Scissors):

$$A = \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}.$$

**Definition (Nash Equilibrium).** A vector  $(x^*, y^*)$  is called a NE if

$$x^{*\top} A y^* \geq x^{*\top} A \tilde{y} \text{ for all } \tilde{y} \in \Delta \text{ and } x^{*\top} A y^* \leq \tilde{x}^\top A y^* \text{ for all } \tilde{x} \in \Delta.$$

**How to compute NE? Let them run MWUA!**

# MWUA and Zero-sum games

**Algorithm (MWUA).** We define the following algorithm for zero sum games:

1. Initialize  $p_{i,x}^0 = 1/n$ ,  $p_{i,y}^0 = 1/n$  for all  $i$  (both players, uniform).
2. **For**  $t=1 \dots T$  **do**
3.     Player  $x$  chooses  $i$  with probability  $p_{i,x}^t$  and  $y$  with  $p_{i,y}^t$  respectively.
4.     **For** each action  $i$  **do**
5.         
$$p_{i,x}^t = p_{i,x}^{t-1} \frac{(1-\epsilon)^{(Ap_y^{t-1})_i}}{Z_x}$$
6.         
$$p_{i,y}^t = p_{i,y}^{t-1} \frac{(1+\epsilon)^{(A^\top p_x^{t-1})_i}}{Z_y}$$
7.     **End For**
8. **End For**

Remarks:

- $\epsilon := \sqrt{\frac{\log n}{T}}$
- $c_i^t := (Ap_y^{t-1})_i$  is the (expected cost) of action  $i$  at time  $t$  for player  $x$ .
- For player  $y$  is the expected utility...

# MWUA and Zero-sum games

**Theorem (MWUA).** Let  $\tilde{x} = \frac{1}{T} \sum_t p_x^t$  and  $\tilde{y} = \frac{1}{T} \sum_t p_y^t$ . Assume that  $A$  has entries in  $[-1, 1]$  and  $T = \Theta\left(\frac{\log n}{\epsilon^2}\right)$ . It holds  $(\tilde{x}, \tilde{y})$  is an  $\epsilon$ -approximate NE that is

$$\tilde{x}^\top A \tilde{y} \leq x'^\top A \tilde{y} + \epsilon \text{ and } \tilde{x}^\top A \tilde{y} \geq x^\top A y' - \epsilon.$$

# MWUA and Zero-sum games

**Theorem (MWUA).** Let  $\tilde{x} = \frac{1}{T} \sum_t p_x^t$  and  $\tilde{y} = \frac{1}{T} \sum_t p_y^t$ . Assume that  $A$  has entries in  $[-1, 1]$  and  $T = \Theta\left(\frac{\log n}{\epsilon^2}\right)$ . It holds  $(\tilde{x}, \tilde{y})$  is an  $\epsilon$ -approximate NE that is

$$\tilde{x}^\top A \tilde{y} \leq x'^\top A \tilde{y} + \epsilon \text{ and } \tilde{x}^\top A \tilde{y} \geq x^\top A y' - \epsilon.$$

*Proof.* **Exercise 6!**



# MWUA and Zero-sum games

**Theorem (MWUA).** Let  $\tilde{x} = \frac{1}{T} \sum_t p_x^t$  and  $\tilde{y} = \frac{1}{T} \sum_t p_y^t$ . Assume that  $A$  has entries in  $[-1, 1]$  and  $T = \Theta\left(\frac{\log n}{\epsilon^2}\right)$ . It holds  $(\tilde{x}, \tilde{y})$  is an  $\epsilon$ -approximate NE that is

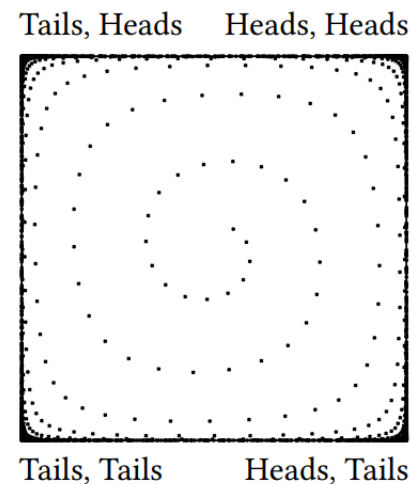
$$\tilde{x}^\top A \tilde{y} \leq x'^\top A \tilde{y} + \epsilon \text{ and } \tilde{x}^\top A \tilde{y} \geq x^\top A y' - \epsilon.$$

*Proof.* **Exercise 6!**

Remark: The result above is not true for last iterate  $p_x^T, p_y^T$ .

**Definition.** *Matching Pennies:*

$$A = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \Rightarrow$$



# General Family of no-regret Algorithms

**Definition (Follow the Leader).** Let  $f_k : \mathbb{R}^n \rightarrow \mathbb{R}$  be convex functions for all  $k$ , differentiable in some convex set  $\mathcal{K}$ . FTL is defined:

Initialize at some  $x_0$ .

For  $t:=1$  to  $T$  do

1. Choose  $x_t = \operatorname{argmin}_{x \in \mathcal{K}} \sum_{k=0}^{t-1} f_k(x)$ .

Remark: The above can perform really **poorly**! Why?

# General Family of no-regret Algorithms

**Definition (Follow the Leader).** Let  $f_k : \mathbb{R}^n \rightarrow \mathbb{R}$  be convex functions for all  $k$ , differentiable in some convex set  $\mathcal{K}$ . FTL is defined:

Initialize at some  $x_0$ .

For  $t:=1$  to  $T$  do

1. Choose  $x_t = \operatorname{argmin}_{x \in \mathcal{K}} \sum_{k=0}^{t-1} f_k(x)$ .

Remark: The above can perform really **poorly**! Why?

Consider  $n = 2$ ,  $\mathcal{K} = \Delta_2$ ,  $x_0 = (1/2, 1/2)$  and  $f_k(x) = x^\top \ell_k$ .

- $\ell_0 = (0, 1/2)$

# General Family of no-regret Algorithms

**Definition (Follow the Leader).** Let  $f_k : \mathbb{R}^n \rightarrow \mathbb{R}$  be convex functions for all  $k$ , differentiable in some convex set  $\mathcal{K}$ . FTL is defined:

Initialize at some  $x_0$ .

For  $t:=1$  to  $T$  do

1. Choose  $x_t = \operatorname{argmin}_{x \in \mathcal{K}} \sum_{k=0}^{t-1} f_k(x)$ .

Remark: The above can perform really **poorly**! Why?

Consider  $n = 2$ ,  $\mathcal{K} = \Delta_2$ ,  $x_0 = (1/2, 1/2)$  and  $f_k(x) = x^\top \ell_k$ .

- $\ell_0 = (0, 1/2)$
- Thus  $x_1 = (1, 0)$

# General Family of no-regret Algorithms

**Definition (Follow the Leader).** Let  $f_k : \mathbb{R}^n \rightarrow \mathbb{R}$  be convex functions for all  $k$ , differentiable in some convex set  $\mathcal{K}$ . FTL is defined:

Initialize at some  $x_0$ .

For  $t:=1$  to  $T$  do

1. Choose  $x_t = \operatorname{argmin}_{x \in \mathcal{K}} \sum_{k=0}^{t-1} f_k(x)$ .

Remark: The above can perform really **poorly**! Why?

Consider  $n = 2$ ,  $\mathcal{K} = \Delta_2$ ,  $x_0 = (1/2, 1/2)$  and  $f_k(x) = x^\top \ell_k$ .

- $\ell_0 = (0, 1/2)$
- Thus  $x_1 = (1, 0)$
- $\ell_1 = (1, 0)$

# General Family of no-regret Algorithms

**Definition (Follow the Leader).** Let  $f_k : \mathbb{R}^n \rightarrow \mathbb{R}$  be convex functions for all  $k$ , differentiable in some convex set  $\mathcal{K}$ . FTL is defined:

Initialize at some  $x_0$ .

For  $t:=1$  to  $T$  do

1. Choose  $x_t = \operatorname{argmin}_{x \in \mathcal{K}} \sum_{k=0}^{t-1} f_k(x)$ .

Remark: The above can perform really **poorly**! Why?

Consider  $n = 2$ ,  $\mathcal{K} = \Delta_2$ ,  $x_0 = (1/2, 1/2)$  and  $f_k(x) = x^\top \ell_k$ .

- $\ell_0 = (0, 1/2)$
- Thus  $x_1 = (1, 0)$
- $\ell_1 = (1, 0)$
- Thus  $x_2 = (0, 1)$

# General Family of no-regret Algorithms

**Definition (Follow the Leader).** Let  $f_k : \mathbb{R}^n \rightarrow \mathbb{R}$  be convex functions for all  $k$ , differentiable in some convex set  $\mathcal{K}$ . FTL is defined:

Initialize at some  $x_0$ .

For  $t:=1$  to  $T$  do

1. Choose  $x_t = \operatorname{argmin}_{x \in \mathcal{K}} \sum_{k=0}^{t-1} f_k(x)$ .

Remark: The above can perform really **poorly**! Why?

Consider  $n = 2$ ,  $\mathcal{K} = \Delta_2$ ,  $x_0 = (1/2, 1/2)$  and  $f_k(x) = x^\top \ell_k$ .

- $\ell_0 = (0, 1/2)$
- Thus  $x_1 = (1, 0)$
- $\ell_1 = (1, 0)$
- Thus  $x_2 = (0, 1)$

**Regret  $T/2$  hence average  
Regret not vanishing!**

# General Family of no-regret Algorithms

**Definition** (**Follow the Regularized Leader**). Let  $f_k : \mathbb{R}^n \rightarrow \mathbb{R}$  be convex for all  $k$ , differentiable in some convex set  $\mathcal{K}$ . Moreover, let  $R$  be a strongly convex function. FTRL is defined:

Initialize at some  $x_0$ .

For  $t:=1$  to  $T$  do

1. Choose  $x_t = \operatorname{argmin}_{x \in \mathcal{K}} \{ \epsilon_{t-1} \cdot \sum_{k=0}^{t-1} f_k(x) + R(x) \}$ .

What happens when  $R(x) = \frac{1}{2} \|x\|^2$  and  $f_k(x) = x^T c_k$  (linear in  $x$ )?



# General Family of no-regret Algorithms

**Definition** (**Follow the Regularized Leader**). Let  $f_k : \mathbb{R}^n \rightarrow \mathbb{R}$  be convex for all  $k$ , differentiable in some convex set  $\mathcal{K}$ . Moreover, let  $R$  be a strongly convex function. FTRL is defined:

Initialize at some  $x_0$ .

For  $t:=1$  to  $T$  do

1. Choose  $x_t = \operatorname{argmin}_{x \in \mathcal{K}} \{ \epsilon_{t-1} \cdot \sum_{k=0}^{t-1} f_k(x) + R(x) \}$ .

What happens when  $R(x) = \frac{1}{2} \|x\|^2$  and  $f_k(x)$

**Online GD!**

# General Family of no-regret Algorithms

**Definition** (**Follow the Regularized Leader**). Let  $f_k : \mathbb{R}^n \rightarrow \mathbb{R}$  be convex for all  $k$ , differentiable in some convex set  $\mathcal{K}$ . Moreover, let  $R$  be a strongly convex function. FTRL is defined:

Initialize at some  $x_0$ .

For  $t:=1$  to  $T$  do

1. Choose  $x_t = \operatorname{argmin}_{x \in \mathcal{K}} \{ \epsilon_{t-1} \cdot \sum_{k=0}^{t-1} f_k(x) + R(x) \}$ .

What happens when  $R(x) = \frac{1}{2} \|x\|^2$  and  $f_k(x)$

**Online GD!**

What happens when  $R(x) = -\sum x_i \log x_i$  (entropy) and  $f_k(x) = x^T c_k$  (linear in  $x$ )?

# General Family of no-regret Algorithms

**Definition** (**Follow the Regularized Leader**). Let  $f_k : \mathbb{R}^n \rightarrow \mathbb{R}$  be convex for all  $k$ , differentiable in some convex set  $\mathcal{K}$ . Moreover, let  $R$  be a strongly convex function. FTRL is defined:

Initialize at some  $x_0$ .

For  $t:=1$  to  $T$  do

1. Choose  $x_t = \operatorname{argmin}_{x \in \mathcal{K}} \{ \epsilon_{t-1} \cdot \sum_{k=0}^{t-1} f_k(x) + R(x) \}$ .

What happens when  $R(x) = \frac{1}{2} \|x\|^2$  and  $f_k(x)$

**Online GD!**

What happens when  $R(x) = -\sum x_i \log x_i$  (entropy) a

**MWUA!**

**Exercise 7! (MWUA)**

# Conclusion

- Introduction to Online Optimization and Learning.
  - Applications of MWUA.
  - Introduction to FTRL
- Next week we will talk about **non-convex optimization!**