30/11/2010
Lecture: 10

Name: Dimitrios Georgiou

Potential Games

A Game is called Potential Game, if we can find a Potential Function.

Game with N Players.

Symbolism: a_i : action of player i

$\vec{a}$: vector of all Stategies

$u_i(.)$: Utility Function for each Player.

Definition: - We define a function $(\Phi : A \longrightarrow R)$, the function has domain all actions $\{A\}$, get a vector of Strategies and return a real number $\{R\}$.
The function Φ is ~~define~~ for all Players.
(Where : $A = A_1 \times A_2 \times \ldots \times A_n$ and $A_i \triangleq$ action Space for Player i)

- We define exact Potential Function $(\Phi : A \longrightarrow R)$ if $\forall \vec{a} \in A$, $\forall i \in N$ and $\forall b_i \in A_i$ is valid $\Delta\Phi = \Delta u$.

$\Delta\Phi = \Delta u \iff$
$\Phi(b_i, \vec{a}_{-i}) - \Phi(a_i, \vec{a}_{-i}) = u_i(b_i, \vec{a}_{-i}) - u_i(a_i, \vec{a}_{-i})$

- We define Weighted Potential Function $(\Phi : A \longrightarrow R)$ if $\forall \vec{a} \in A$ ~~and~~ $\forall a_i, b_i \in A_i$ is valid $\Delta\Phi = w_i \Delta u_i$,
w_i: (different for each Player) weight for Player i.

$\Delta\Phi = \Delta u \iff$
$\Phi(b_i, \vec{a}_{-i}) - \Phi(a_i, \vec{a}_{-i}) = w_i \left[u_i(b_i, \vec{a}_{-i}) - u(a_i, \vec{a}_{-i}) \right]$

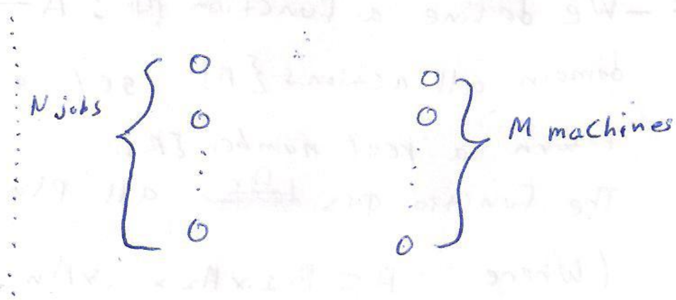
- We define ordinal Potential Function $(\Phi : A \to R)$ if.
$\forall \vec{a} \in A$, $\forall a_i, b_i \in A_i$ we have $\Delta u_i < 0 \Rightarrow \Delta\Phi < 0$

Theorem: Each Potential Game has at least one pure N.E.

- If we have a Potential Game, the Best Respone of the game it Converges to. N.E.

Example:

I have N sobs and M machines and I want to assign jobs to machines



w_i: time needed to execute job i

a_i: the machine who choose user i

$$L_j(\vec{a}) = \sum_{i: a_{ij} = j} w_i \quad \text{- Load on the machine } j$$

$$u_i(\vec{a}) = L_{a_i}(\vec{a}) \quad \text{: Utility of the job } i$$

$$\Phi(\vec{a}) = \frac{1}{2} \sum_{j=1}^{m} L_j^2(\vec{a}) \quad \text{: Potential Function}$$

We assume that the sob i, move from machine M_1 to the machine M_2

$$\Delta u_i = u_i(M_2, \vec{a}_{-i}) - u_i(M_1, \vec{a}_{-i})$$

$$= \underbrace{L_2(\vec{a})}_{\text{Load of } M_2 \text{ Before}} + \underbrace{w_i}_{\text{Load of the new user on Machine 2}} - \underbrace{L_1(\vec{a})}_{\text{Load } M_1 \text{ Before move}}$$

$$\Delta\Phi = \underbrace{\Phi(M_2, \vec{a}_{-i})}_{\text{Load after Move}} - \underbrace{\Phi(M_1, \vec{a}_{-i})}_{\text{Load Before Move}} = \frac{1}{2}\left[(L_1(\vec{a}) - w_i)^2 + (L_2(\vec{a}) + w_i)^2\right] +$$

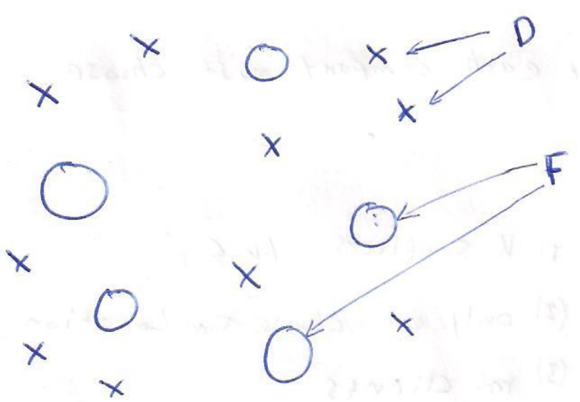
$$+ \frac{1}{2}\left[L_1^2(\vec{a}) + L_2^2(\vec{a})\right]$$

$$\Delta\varphi = w_i\left[L_2(\vec{a}) - L_1(\vec{a})\right] + w_i^2 \cancel{w_i}$$
$$= w_i\left[L_2(\vec{a}) - L_1(\vec{a}) + w_i\right]$$
$$= w_i \Delta u_i$$

Facility Location Game

First: Facility Location Problem without Game

We assume that we have possible places to build a serer.
and ther is a set of clients waiting to be sered.
We want to find a subset of locations to build ~~servers~~



F: possible places to build a shop (ie. sever)

D: set of clients.

$d(i,j)$: distance function, gives the cost to serve ~~at the~~ machine i the client j.

F Vector: $(f_1, f_2, \dots f_N)$: Cost for build (where f_i cost to build shop i)

We want to find a subset(s) of locations to build shops, $S \subseteq F$.

$$\boxed{\begin{array}{l}\text{output} \\ S \subseteq F \\ \text{s.t.}\ \ \min \sum_S f_i + \sum_{j\in D} \min d(i,j)\end{array}}$$

⟸ Says that each client will serve from the nearest sever.

<u>Special case</u>: K-median Problem (Appears in the field of data mined.))

- There is no cost to build a shop. ($f_i = 0$)
- Should choose K places to build ($|S| = K$)

- We want to find at least K places where will minimize the sum of distance between the clients and ~~servers~~ nearest servers of them.

(Algorithm : Lloyd, K-means)

<u>Other version of the promlem</u> (Capacitated Facility Location)

Which have the restriction, where says that at each server (F_i) can not ~~connect~~ be served more than (K_i) users.

<u>Now the Game</u>

Players are competitive companies that each company must choose only one potential location.

We assume that each company: have; (1) K suppliers, $L_n \subseteq F$

(2) only one choose for Location

(3) m. Clients

π_i: Utility of client i, when served from the nearest server.

d_{ij}: The cost needed to serve client i, from the location j.

$$0 \leq d_{ij} \leq \pi_i$$

$$\text{if } d_{ij} \geq \pi_i \text{ then } d_{ij} = \pi_{ij}$$

Client i will serve from $\sigma(i) = \arg\min d_{ij}$ (The nearest server where serve client i)

The strategy of user j : (a) To choose the location
(b) How much costed services

- How much costed services ?
- Will charge the customer i, price equal to the second smallest.

$$P_i = \min_{j \neq \sigma(i)} \lambda_{ij}$$

Benefit of client i: $\pi_i - P_i$

~~Then~~ Benefit of the supplier, where is the nearest to the client i: $P_i - \underbrace{\lambda_{i\sigma(i)}}_{\text{cost of service customer } i}$

$$\text{Total Benefit} = \overset{\text{Benefit}}{\text{Customers}} + \overset{\text{Benefit}}{\text{Suppliers}}$$

$$= \sum_{i\,\text{seres}} (\pi_i - P_i) + \sum_{i:\,\text{sered}} (P_i - \lambda_{i\sigma(i)})$$

$$= \sum_i (\pi_i - \lambda_{i\sigma(i)})$$

Questions

(1) $\exists$ N.E ?

(2) PoA ?

Demonstrated that the Facility Location Game is a Potential Game.

where Potential function $\Phi = \sum_i \lambda_i \sigma(i)$.

Answers

(1) Yes, there is N.E (From the theorem of Potential games.)

(2) $PoA \leq 2$

To maximize the total Benefit, need to minimize $\sum_i \lambda_i \sigma(i)$,

but it is just the Potential function.

Mechanisms of Learning

- Regret Mechanism
- Fictitious Play

We consider ~~for~~ a Player, has made a series of N Strategies.

$A = \{a_1, a_2, \dots, a_n\}$

As times passes (step t): $\ell_i(t)$ (cost if make strategy a_i, $\ell_i(t) \in [0,1]$)

General Strategy of Player i: $\underline{P}(t) = (\underbrace{P_1(t)}, \dots, P_m(t))$

The probability to choose Strategy a_1 at time t.

$\underline{\ell} = (\ell_1(t), \dots, \ell_m(t))$ — loss.

Average loss ~~$\underline{\ell}(t) = (\ell_1(t), \dots, \ell_m(t))$~~
at time t : $\sum_{i=1}^{m} P_i(t) \cdot \ell_i(t) = \overline{P}(t)$

Συνολικό loss in T periods: $L_T = \sum_{t=1}^{t} \bar{l}(t) = \sum_{t=1}^{T} \left(\underline{p}(t)^T \cdot \underline{l}(t) \right)$

We want to compare the losses; so we can see how well is the user

We compare the loss of ~~user~~ a user i from strategy a_i with the loss of a user i if apply the optimal strategy.

$$OPT = \sum_{t=1}^{T} \min_i l_i(t)$$

$$Regret^{(1)} = L_T - OPT$$

$$Regret^{(2)} : \min \{0, L - L_T^{Best}\}$$

$$L_T^{Best} = \min_i \left(\sum_{t=1}^{T} l_i(t) \right)$$

There is 3 Algorithm who help to change strategy at every step

(1) Gready Algorithm

$l_i(t) \in \{0, 1\}$

- $t = 1$ choose randomly strategy a_1
- $t > 1$ choose the best action until now: $a_t = L_{t-1}^{i}$

$$\boxed{\text{where } L_t^i = \sum_{t=1}^{t} l_i(t) \,;\ \forall\ i \text{ strategy}}$$

~~For this strategy~~ :

For the Gready Algorithm : $\boxed{L_T^{Gready} \leq m L_T^{best} + m - 1}$

where m : number of strategies and

$$L_T^{best} = \min_i \left(\sum_{t=1}^{T} l_i(t) \right)$$

the Algorithm says

I choose every time the strategy, ~~that~~ which until that time has given me less losses.

(2) Random Gready (RG)

$$S_t = \{ i : L_t^i = L_t^{best} \}$$

- $t=1$: choose $a_i^1 \in \{a_1, \dots a_m\}$ with probability $\frac{1}{m}$
- $t>1$: Choose $a_i(t)$ random from those with smaller losses, as at that time.

$$P_i(t) = \begin{cases} \frac{1}{|S_{t-1}|} & , \text{if } i \in S_{t-1} \\ 0 & , \text{else.} \end{cases}$$

$$\boxed{L_T^{RG} \leq (\ln m + 1) L_T^{Best} + \ln m}$$

(3) Random Weighted Majority

Can choose and strategies were not the best so far.

$$w_i(t) = (1-\varepsilon)^{L_{t-1}} \quad , \quad \varepsilon \in [0,1]$$

weight $\forall$ strategy and $\forall$ time t.

as increasing losses, reduce the weight.

$w_i(0) = 1$

A Strategy i, at each step, chosen with probability: $P_i(t)$

$$P_i(t) = \frac{w_i(t)}{\sum_{i=1}^{m} w_i(t)}$$

- $t = 1$, $w_i(1) = 1$, $P_i(1) = \frac{1}{m}$

- $t \geq 2$; $w_i(t) = (1-\beta)^{L^i_{t-1}}$, $l_i(t-1) = 1$

where $L^i_{t-1} \begin{cases} L^i_{t-2} + 1 \\ L^i_{t-2} \end{cases}$

total weight : $W(t) = \sum_{i=1}^{m} w_i(t)$

Probability : $P_i(t) = \frac{w_i(t)}{W(t)}$

$$\boxed{\beta \leq \frac{1}{2} \cdot L_T^{RWM} \leq (\beta+1) L_T^{best} + \frac{\ln m}{\beta}}$$