Learning In Distributed Systems: A Game
Theoretic View

Malik Magdon-Ismail
Rensselaer Polytechnic Institute

November 9, 2004.
Example

Sensor Network Admission Control

(Joint work with Carothers, Yener)
Sensors – A Simple Model

Sensor Capabilities:
- Can **sense** some particular event $X$ and **transmit**.
- Can **receive** a neighbor transmission and **re-transmit**.
- No buffer to store events.
- Small CPU, small RAM (can store and execute a small program).
- Has access to a random number generator.
- Low power FM antenna to receive (centralized) information.
A Simple Learning Task

A sensor senses an event and/or receives a transmission.

- Transmit sensed event (local job)
- XOR
- Transmit received event (remote job)

\[ p_{route} = \text{probability to process remote job} \]

If no remote job exists, only option is local job.
Feedback

Events have a value $X$ (cookie).

Individual sensors have “bank accounts” (bank).

$\text{ROUTE (remote job): } \text{bank } \epsilon.$

(immediate reward)

$\text{SENSE (local job): } \text{bank } \beta X.$

(potential delayed reward)

- Delayed reward only if job arrives at destination.
- Bank is communicated by central authority.
- Sensors occasionally check their “bank accounts”
A Simple Learning Algorithm

Learning Task (for each sensor): Determine $p_{route}$.

Gradient Based algorithm:

1: Initialize: $p_{route}(0)$;
2: Obtain **change** in bank after $\tau$ periods: $\Delta B(0)$;
3: **while** Alive **do**
4: Perturb: $p_{route}(t + 1) \leftarrow p_{route}(t) + \Delta p$;
5: Obtain **change** in bank after $\tau$ periods: $\Delta B(t+1)$;
6: **if** $\Delta B(t + 1) > \Delta B(0)$ **then**
    7: Keep search direction;
8: **else**
9: Change search direction: $\Delta p \leftarrow -\Delta p$;
10: **end if**
11: **end while**
A Simple Learning Algorithm

Learning Task (for each sensor): Determine $p_{route}$.

Gradient Based algorithm:

1: Want to learn $p_{route}$.
2: Start somewhere (say) $p_{route} = 0.5$.
3: How good is $p_{route}$?
4: Change $p_{route}$ a little.
5: Is it better?
6: if yes then
7: Keep going.
8: else
9: Reverse direction.
10: end if
11: Repeat until “convergence”.

A Simulation

– Each sensor has a continuous stream of sensing events.

– Cookies drawn from a stationary distribution.

– $\epsilon, \beta$ fixed.
Convergence?

Given: \( \epsilon, \beta \) and traffic (cookie) distribution \((X)\),

– convergence to a “self organized” **steady state**.

What are the properties of the steady state?
Reinforcement Learning?

Similar to setup of Reinforcement Learning:

- There is an environment \((\epsilon, \beta, X)\).
- Node performs action (picks a particular \(p_{route}\)).
- Node receives reward (increase in bank account).
Except . . .

. . . there are other “reinforcement learners” that affect your reward.

– Adversarial.
Two Node System

\[ D_2 \quad S_1 \quad S_2 \quad D_1 \]

\[ x_1 \quad x_2 \]
Deterministic Actions

$S_1$: route $R$ or sense $S$.

$S_2$: route $R$ or sense $S$.

4 state system

$(R, R), (R, S), (S, R), (S, S)$.
Payoffs

Sensor 1

Sensor 2

<table>
<thead>
<tr>
<th></th>
<th>( R )</th>
<th>( S )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R )</td>
<td>( \frac{\beta_1 X_1 + \epsilon_1}{2} )</td>
<td>( \epsilon_1 )</td>
</tr>
<tr>
<td>( S )</td>
<td>( \epsilon_2 )</td>
<td>( 0 )</td>
</tr>
</tbody>
</table>
Non-Cooperative Games
Two Player Matrix Games

\[ R_1(s_1, s_2) \] is reward to player 1 when \((s_1, s_2)\) is played.

\[ R_2(s_1, s_2) \] is reward to player 2 when \((s_1, s_2)\) is played.
Nash Equilibrium

$(s_1, s_2)$ is a Nash Equilibrium (NE) if

neither player wishes to switch strategies:

$R_1(s_1, s_2) \geq R_1(u, s_2)$ for all other possible plays $u$ of player 1.

$R_2(s_1, s_2) \geq R_1(s_1, v)$ for all other possible plays $v$ of player 1.
Some Two Person Games

Prisoners Dilemma

<table>
<thead>
<tr>
<th></th>
<th>Quiet</th>
<th>Rat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quiet</td>
<td>(-3,-3)</td>
<td>(-10,-1)</td>
</tr>
<tr>
<td>Rat</td>
<td>(-1,-10)</td>
<td>(-7,-7)</td>
</tr>
</tbody>
</table>

**NE:** (Rat,Rat)

Matching Pennies

<table>
<thead>
<tr>
<th></th>
<th>Heads</th>
<th>Tails</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heads</td>
<td>(1,-1)</td>
<td>(-1,1)</td>
</tr>
<tr>
<td>Tails</td>
<td>(-1,1)</td>
<td>(1,-1)</td>
</tr>
</tbody>
</table>

**NE:** None

If probabilistic choices are allowed (mixed strategies), there is always a Nash Equilibrium [Nash, 1951].
Repeated Games
(Non-Perfect Information)

Players play multiple times (not necessarily against the same players).

Varying degrees of information (feedback):
  Know the strategy spaces;
  Experience only reward;
  ...

Convergence: for a large class of repeated games, and imperfect information settings, gradient based (learning) methods converge to a Nash Equilibrium [Arslan & Shamma, 2003,2004].
Two Node Sensor Game

Sensor 1

Sensor 2

\[
\begin{array}{cccc}
R & S \\
\hline
R & \beta_2 X_2 + \epsilon_2 / 2 & \beta_2 X_2 \\
S & \beta_1 X_1 + \epsilon_1 / 2 & \epsilon_1 & 0 \\
\end{array}
\]
Large Sensing Reward

Nash Equilibria: \((S, R), (R, S)\)
Large Routing Reward

Nash Equilibria: \((R, R)\)
Mixed Reward

Nash Equilibria: \((S, R)\)
Summary of NE

OSAMA vs. osama
  One will make it through.

Zebra vs. Zebra
  Both zebra’s make it through half time.

Osama vs. Zebra
  Osama makes it through.
Why NE

System settles in a NE.

NE is a function of control parameters.

Control problem:
- Have some global criterion to optimize.
- Selfish learners reach some steady state.
- How bad is this steady state?
The Price of Anarchy
[Koutsoupias & Papadimitriou, 1999]

Anarchy: A bunch of selfish “individuals”.

Price of Anarchy: How bad is where the selfish individuals settle compared with the optimal for the “society”.

\[
\text{Price of Anarchy} = \frac{\text{“Cost” of worst NE}}{\text{“Cost” of optimal}}
\]
A Reasonable Cost Function

\[1 - \Pr(X_1)\Pr(X_2)\]

Theorem. Price of Anarchy \(\leq \frac{4}{3}\).

Learning in this distributed system is not too bad, with respect to this cost function.
Wrapping Up

– Many situations where learning in distributed systems occurs:
  Learning gives adaptivity.

– What happens in such systems: Nash Equilibria
  Taylor the individual objective so that desired NE are “good”. Control parameters can also affect NE.

– Learning can be bad.
  There can be bad NE: that’s the price of anarchy.
Some Context

[Wardrop, Beckman, 1950’s], Nash Equilibria in selfish traffic flow.

[Korilis & Lazar]
[Nisan & Ronen]
[Koutsoupias, Papadimitriou]
[Roughgarden & Tardos]
[Shenker]

Internet protocols

(selfish routing of flows)


[Bush & Magdon-Ismail], Network games for load distribution (in progress).
[Magdon-Ismail et al.], Sensor network admission control (in progress).
Thank You!