Deterministic Calibration with Simple
Rules

Dean Foster
University of Pennsylvania

and

Sham Kakade
University of Pennsylvania

January 31, 2005

The problem: Learning Nash equilibr

Current methods are slow and involve exhaustive sear

Can a fast method be found?

How about for special form games?

Measuring complexity

Two definitions of speed of convergence:

- total CPU used
- number of rounds of play

History

	Forecast probability	**Forecas**
Blackwell	**CE** Calibration (F. and Vohra, '97)	**CE** No re (F. and V (Hart and Ma
Exhaustive search	**NE** Hypothesis testing (F. and Young '03)	**NE** Regret (F. and Yo (Germano &
Public methods	**NE** Weak calibration yesterday's talk (Kakade and F. '04)	**NE** Weak utility today' (Kakade ar

	Forecast probability	**Forecas**
Blackwell ($\to$ CE)	$(1/\epsilon)^{a^n}$	$(a/\epsilon$
Exhaustive search ($\to$ Nash)	$\gg (1/\epsilon)^{a^n}$	$\gg (1/$
Public methods ($\to$ Nash)	$(1/\epsilon)^{a^n}$	$(1/\epsilon$
	$2^{\vert\mathcal{I}\vert}$	$\vert\mathcal{I}\vert^{\log\log\vert\mathcal{I}\vert}$ (wit

$$
\begin{aligned}
n &= \text{number of players} \\
a &= \text{number of actions per play} \\
\epsilon &= \text{desired accuracy} \\
|I| = a^n &= \text{input size } (a \text{ is fixed})
\end{aligned}
$$

(CE: Blackwell gives fast approx algo. NE: slow, few
results known.)

- X_t sequence to be forecast by p_t

- Weak calibration, means

$$\sum_{t=1}^{T} (X_t - p_t)\, w(p_t) \to 0$$

  - $w()$ is any smooth function.

  - What Sham talked about yesterday.

- Today's twist: Use other testing functions. Eg

$$\sum_{t=1}^{T} (X_t - p_t)\, w(p_t, X_{t-1}) \to 0$$

  Would test for Markov patterns.

Individual vs Public calibration

- Game setting for calibration
  - $X_{i,t}$ is the observable that player i cares about
  - $p_{i,t}$ is a forecast of $X_{i,t}$

- Individual calibration:

$$(\forall i) \qquad \sum_{t=1}^{T} (X_{i,t} - p_{i,t})\, w(p_{i,t}) \to 0$$

- Public calibration:

$$(\forall i) \qquad \sum_{t=1}^{T} (X_{i,t} - p_{i,t})\, w(\vec{p}_t) \to 0$$

The game model

- Player i uses $p_{i,t}$ to predict the round t

- Player i then use smooth decision rule $s_i(p_{i,t})$ to p probability of their play in round t.

- Player i then randomly action S_i from this distrib

Observables

- Game setup:
  - Take $X_i = S_{-i}$ (i.e. all actions but player i)
  - $p_{i,t}$ is forecast of $X_{i,t}$

- Individual calibration:

$$(\forall i) \qquad \sum_{t=1}^{T} (X_{i,t} - p_{i,t})\, w(p_{i,t}) \to 0$$

- Public calibration:

$$(\forall i) \qquad \sum_{t=1}^{T} (X_{i,t} - p_{i,t})\, w(\vec{p}_t) \to 0$$

- Suppose players play a smooth best reply to forec
  - Traditional calibration $\rightarrow$ correlated equilibria
  - Public calibration $\rightarrow$ Nash equilibria

- Speed of convergence is related to dimension of t
space" of the testing functions
  - For individual: dimension $(1/\epsilon)^{a^n}$
  - For public: dimension is $(1/\epsilon)^{na^n}$
  - Hence convergence is slow in both cases.

- Need lower dimensional space, but what can be c

- Truth $\approx$ prediction
  - via calibration

- Truth is independent
  - Given $\vec{p}$ each player is in fact playing independe

- ϵ-rationality
  - ϵ-BR to prediction
  - p_i includes information about what all other pl

- Independence $+$ ϵ-rationality $=$ ϵ-NE.

What can be changed?

- Take $X_{i,t}$ to be the vector of potential payoffs
  - $\vec{S}_{-i}$ is the vector of everyone else's play
  - $u_{i,t}(k) = u_i(k, \vec{S}_{-i,t})$
  - $X_{i,t} = (u_{i,t}(1), \ldots, u_{i,t}(a))$

- Utility model
  - $p_{i,t}$ is an estimate of $X_{i,t}$ made at time $t-1$
  - For CE we need

$$(\forall i) \qquad \sum_{t=1}^{T} (X_{i,t} - p_{i,t})\, w(p_{i,t}) \rightarrow$$

  - For NE we need

$$(\forall i) \qquad \sum_{t=1}^{T} (X_{i,t} - p_{i,t})\, w(\vec{p}_t) \rightarrow$$

Speed of convergence of utility estima

- For CE: number of rounds is $O((n/\epsilon)^a)$
- For NE: number of rounds is $O((n/\epsilon)^{an})$
- Looks almost polynomial in length of input
  - $|I| = a^n =$ input size (a is fixed)
  - number of rounds is $O(|\mathcal{I}|^{\log\log|\mathcal{I}|})$
  - "pseudo Poly".
- Although exp in a, little known computationally.

Graphical Models for Game Theor

- Undirected graph capturing local (strategic) inter
(Kearns, Littman, & Singh)
  - Each "player" represented by a vertex
  - Payoff to i, is only a function of neighbors act
  - Compact (yet general) representation of game
  - Assume max degree is d, then representation is
of $O(a^n)$.

- Can graphical games be learned faster than gener

- $X_{i,t}$ need only capture plays of neighbors
  - $N(i)$ is the set of neighbors of i (assume $|N(i)$
  - $S_{N(i)-i}$ is actions of all neighbors excluding sel
  - $u_{i,t} = u_i(S_{i,t}, S_{N(i)-i})$
  - $p_{i,t}$ is forecast of $X_{i,t}$

- Same proof as before shows that for a NE we nee

$$(\forall i) \qquad \sum_{t=1}^{T} (X_{i,t} - p_{i,t})\, w(\vec{p}_t) \to 0$$

- But we desire to to better for structured games.

  (This is $(1/\epsilon)^{na^d}$, while the representation of a gr na^d.)

- We don't need to check $w(\vec{p}_t)$
- Instead we can check only

$$(\forall i) \qquad \sum_{t=1}^{T} (X_{i,t} - p_{i,t})\, w(\vec{p}_{N(i),t}) \rightarrow$$

where $\vec{p}_{N(i),t}$ is a vector of all the p's of all the ne

- Since this is all that matters in $u_i()$, rationality ag rationality against the entire $\vec{p}$.
- Complexity: $n(1/\epsilon)^{a^{2d}}$
- The complexity is $|\mathcal{I}|$.
- NOTE TO SELF: No matter how excited you are complexity, never, write it as $|\mathcal{I}|$!

A even smaller observable set

- X_i = personal utility

- p_i = forecast of personal utility

- $w()$ is local:

$$(\forall i) \qquad \sum_{t=1}^{T} (X_{i,t} - p_{i,t})\, w(\vec{p}_{N(i),t}) \rightarrow$$

- Converges to NE.

- Complexity: $n(1/\epsilon)^{a^d}$

- $X_i =$ action taken

- $p_i =$ forecast of own action

- decisions are made based on other peoples forecas

- $w()$ is local:

$$(\forall i) \qquad \sum_{t=1}^{T} (X_{i,t} - p_{i,t})\, w(\vec{p}_{N(i),t}) \rightarrow$$

- Converges to NE.

- Complexity: $n(1/\epsilon)^{a^d}$

- Violations can cause the system to crumble

Speed of convergence:

- Complexity: $n(1/\epsilon)^{da^d}$
- Recall, game representation is na^d
- Hence, the max degree is the bottleneck!
- Can get better results with utility forecasts: $n(1/$

CPU time:

- For tree games, fast per round computation
- Total CPU time comparable to NashProp
- For general graphs, could be hard to make foreca

Future directions

- Analyze the CPU complexity
  - Have we just pushed the difficulty back to the
    step?

- Look at other games with simple structure

- Look at linear weightings rather than local weight

See reverse side of handout for related re