Opponent Modeling in Imperfect-Information EFGs

Ranges, Probabilistic Policies, and Posterior-Predictive Exploitation

Intelligent Agents: Computational Game Solving

November 13, 2025

Why Opponent Modeling? Where Are We Going?

Setting: Imperfect-information extensive-form games (poker-like). Opponents are *not* necessarily playing Nash.

What we want:

- Use observed actions to infer *how* the opponent plays (policy) and *what* they likely hold (range).
- Turn beliefs into *posterior-predictive* control: choose actions that increase EV against current opponent.
- Do this *online*, repeatedly, as the public state evolves.

Where we will get to (today):

- Formalize public belief states (PBS), ranges, and probabilistic policies.
- Fit opponent models (Dirichlet/tabular; softmax/parametric) from action data.
- Update ranges in PBS and compute best response in the current subgame to the *posterior-predictive* opponent.

Next (Lecture 2): *Safe* exploitation: Restricted Nash Response, robust blending, and re-solving constraints (control risk of model error).

Context: Robust Blueprint vs. Live Exploitation

Headline systems (Libratus, DeepStack, Pluribus): did *not* perform per-opponent modeling or targeted live exploitation.

- **Blueprint:** self-play CFR/LCFR to near-equilibrium in an abstraction.
- **Ranges:** Bayesian narrowing from public history (standard), not fitting opponent policies.
- **Real-time:** subgame re-solving (2p) or depth-limited search (6p) with safe/robust constraints.

Why avoid live exploitation:

- **Risk:** model error $\Rightarrow$ increased exploitability (especially in multiplayer).
- **Data sparsity/drift:** few observations per infoset; opponent behavior changes.
- **Complexity:** 6-player "best response" notions are subtle; robustness wins.

Where exploitation appears:

- Controlled 2-player settings; **Restricted Nash Response (RNR)** and robust variants.
- Offline analysis/personalization; not typically used mid-match in production AIs.

Takeaway: We study opponent modeling to understand adaptation and build exploiters in 2p scenarios.

Roadmap

Today:

1. Setup: public states, ranges, infosets (recap EFG notation)
2. Two opponent model types: nonparametric (Dirichlet) and parametric (softmax)
3. Why these models? (Bayesian smoothing; maximum entropy)
4. Bayesian updates from observed actions
5. Hidden private states: marginalizing over opponent's hidden cards
6. Range updates in public belief states (PBS)
7. Using the model: posterior-predictive control and exploitation

Goal: From observations $\Rightarrow$ beliefs $\Rightarrow$ exploitative decisions (with safety next lecture).

Setup: Extensive-Form Notation (Recap)

- Histories $h \in \mathcal{H}$, terminals $z \in Z$, utility $u_i(z)$.
- Infosets $\mathcal{I}_i$ for player i; actions $A(I)$ at I.
- Strategy profile $\sigma = (\sigma_1, \sigma_2)$ (behavioral strategies).
- Reach prob. to h: $\pi^\sigma(h) = \pi^\sigma_c(h) \cdot \pi^\sigma_1(h) \cdot \pi^\sigma_2(h)$.
- Counterfactual values $v_i^\sigma(I, a)$, $v_i^\sigma(I)$ (CFR context).

New terminology for modeling:

- **Public state S:** shared observable history (public cards, bets, pot, stacks).
- **Private state x:** hidden info (e.g., opponent's hole cards).
- **Range $\rho_{-i}(x \mid S)$:** our belief distribution over opponent's private states at S.

Public Belief States (PBS)

Definition: Public state S partitions histories by shared observations.
Within S, each player holds a *range* over private states:

$$\rho_i(x \mid S) = \text{probability player } i \text{ holds private state } x \text{ at } S$$

Example (poker):

- S: Flop is K♠ 7♡ 2♣; opponent has bet; pot $= 40$
- $\rho_{-i}(x \mid S)$: distribution over opponent's hole card pairs $x \in \{$AA, KK, QQ, ..., 72o$\}$
- Some x more likely given betting (e.g., opponent unlikely to hold 72o after betting)

Range evolution:

$$\rho_{-i}(x \mid S_{\text{new}}) \propto \rho_{-i}(x \mid S_{\text{prev}}) \cdot \Pr(\text{observed action} \mid x, S, \sigma_{-i})$$

Use: Conditioning subgames (re-solving), counterfactual evaluation, exploitative play.

Opponent Modeling: What We Need

Two intertwined problems:

1. **Policy modeling:** Infer $\sigma_{-i}(I, a)$ from observed actions
   - At each infoset I, what's the probability opponent plays action a?
   - Data: observed (I_t, a_t) pairs
2. **Range tracking:** Update $\rho_{-i}(x \mid S)$ as actions are observed
   - Bayesian update: actions reveal information about private state x
   - Uses the policy model from (1) as the likelihood function

Output: Posterior-predictive opponent policy $\hat{\sigma}_{-i}$ and updated range ρ_{-i}

Decision rule: Compute our best response to $\hat{\sigma}_{-i}$ in current subgame

Two Approaches to Policy Modeling

Approach 1: Nonparametric (Dirichlet-Multinomial)

- Treat each infoset I independently
- Place a Dirichlet prior over action probabilities
- Update with observed counts: closed-form Bayesian posterior
- Pro: Simple, exact, no hyperparameters (except prior α)
- Con: No sharing across infosets; data-hungry

Approach 2: Parametric (Softmax/Log-Linear)

- Model $\sigma_{-i}(I, a)$ with features $f(I, a)$ and shared parameters θ
- Fit θ via maximum likelihood (gradient descent)
- Pro: Generalizes across infosets; data-efficient
- Con: Requires feature engineering; potential mis-specification

Often combined: Use parametric for well-observed infosets, Dirichlet for rare ones

Nonparametric Approach: Dirichlet Prior

Goal: Model opponent's action distribution at infoset I without features or parameters

Bayesian framework:

1. **Prior:** Opponent's policy $p(I, \cdot)$ at I is a probability vector over $A(I)$
2. Place a distribution over this probability vector: $p(I, \cdot) \sim \text{Dirichlet}(\alpha(I, \cdot))$
3. $\alpha(I, a) > 0$ are hyperparameters (one per action)

The Dirichlet distribution:

- A distribution over the probability simplex: $\sum_a p(a) = 1$, $p(a) \geq 0$
- Parameterized by $\alpha = (\alpha_1, \ldots, \alpha_K)$ where $K = |A(I)|$
- Prior mean: $\mathbb{E}[p(a)] = \alpha(a)/\alpha_0$ where $\alpha_0 = \sum_{a'} \alpha(a')$
- Concentration: larger α_0 = stronger prior (more "pseudo-counts")

Intuition: $\alpha(I, a)$ acts like "pseudo-counts" — imaginary observations before seeing real data

Dirichlet Posterior After Observations

Observed data at infoset I: Counts $n(I, a)$ for each action

Bayesian update (conjugacy):

- Prior: $p(I, \cdot) \sim \text{Dirichlet}(\alpha(I, \cdot))$
- Likelihood: multinomial with counts $n(I, \cdot)$
- Posterior: $p(I, \cdot) \mid n \sim \text{Dirichlet}(\alpha(I, \cdot) + n(I, \cdot))$

Posterior predictive policy (what we use for decisions):

$$\hat{\sigma}_{-i}(I, a) = \frac{\alpha(I, a) + n(I, a)}{\alpha_0 + N}$$

where $\alpha_0 = \sum_{a'} \alpha(I, a')$ and $N = \sum_{a'} n(I, a')$

Convex combination view:

$$\hat{\sigma}_{-i}(I, a) = \underbrace{\frac{\alpha_0}{\alpha_0 + N}}_{\text{prior weight}} \cdot \underbrace{\frac{\alpha(I, a)}{\alpha_0}}_{\text{prior mean}} + \underbrace{\frac{N}{\alpha_0 + N}}_{\text{data weight}} \cdot \underbrace{\frac{n(I, a)}{N}}_{\text{empirical freq}}$$

The Role of α (Prior Hyperparameters)

What does α do?

1. **Smoothing:** Prevents zero probabilities for unobserved actions
   - If $n(I,a) = 0$ but $\alpha(I,a) = 1$, then $\hat{\sigma}_{-i}(I,a) > 0$
   - Avoids division-by-zero in range updates; regularizes estimates
2. **Prior beliefs:** Encodes baseline policy before seeing data
   - Uniform prior: $\alpha(I,a) = c$ for all a (e.g., $c = 1$) — no prior knowledge
   - Informative prior: $\alpha(I,a) = c \cdot \pi_0(a)$ — encode equilibrium or baseline policy
3. **Adaptation rate:** Controls how quickly posterior moves from prior to data
   - Larger $\alpha_0 = \sum_a \alpha(a) \to$ slower adaptation (prior dominates longer)
   - Smaller $\alpha_0 \to$ faster adaptation (data dominates sooner)
   - Think of α_0 as "equivalent sample size"

Not a restriction: α parameterizes a prior over *all* valid probability distributions

Example: Dirichlet Update at an Infoset

Scenario: Infoset I with actions $A(I) = \{\text{check}, \text{bet}\}$

Prior: $\alpha(I, \text{check}) = 1$, $\alpha(I, \text{bet}) = 1$ (uniform; $\alpha_0 = 2$)

Observations: check, bet, bet, check, bet $\rightarrow$ counts $n(\text{check}) = 2$, $n(\text{bet}) = 3$, $N = 5$

Posterior predictive:

$$\hat{\sigma}_{-i}(I, \text{check}) = \frac{1+2}{2+5} = \frac{3}{7} \approx 0.43$$

$$\hat{\sigma}_{-i}(I, \text{bet}) = \frac{1+3}{2+5} = \frac{4}{7} \approx 0.57$$

Compare to:

- Pure empirical frequency: check $= 2/5 = 0.40$, bet $= 3/5 = 0.60$
- Prior mean: check $= 1/2 = 0.50$, bet $= 1/2 = 0.50$
- Posterior is a blend (prior weight $= 2/7 \approx 0.29$, data weight $= 5/7 \approx 0.71$)

Dirichlet: Prior Fades with More Data

As observations accumulate, data dominates:

N (total obs)	Prior weight $\frac{\alpha_0}{\alpha_0+N}$	Data weight $\frac{N}{\alpha_0+N}$	Effect
0	$2/2 = 1.0$	$0/2 = 0.0$	Pure prior
5	$2/7 \approx 0.29$	$5/7 \approx 0.71$	Prior visible
50	$2/52 \approx 0.04$	$50/52 \approx 0.96$	Nearly empirical
500	$2/502 \approx 0.004$	$500/502 \approx 0.996$	empirical freq

Key insight: α matters most when data is sparse (early observations, rare infosets)

Practical choices:

- Symmetric $\alpha = 1$: Laplace smoothing (add 1 to all counts)
- Informative $\alpha \propto \pi_{\text{NE}}$: start with equilibrium baseline
- Larger α_0: conservative (slow to trust new patterns)

Why Dirichlet? (Conjugacy Property)

Conjugacy: Prior and posterior are in the same family

Bayesian update:

- **Prior:** $p \sim \text{Dirichlet}(\alpha)$
- **Likelihood:** Multinomial counts $n = (n_1, \ldots, n_K)$
- **Posterior:** $p \mid n \sim \text{Dirichlet}(\alpha + n)$

Benefits:

1. **Closed-form updates:** No numerical optimization needed
2. **Fast online learning:** Add observed a to $n(I, a)$ in $\mathcal{O}(1)$
3. **Uncertainty quantification:** Posterior variance reflects confidence
4. **Sequential updating:** Observe data incrementally, update on-the-fly

Alternative (non-conjugate): If we used different prior (e.g., Gaussian on logits), would need MCMC or variational inference → much slower

Parametric Approach: Softmax over Features

Motivation: Share information across infosets via features

Key idea:

- Define features $f(I, a)$ capturing properties of (infoset, action) pair
- Model action probability as softmax of linear score: $\text{score}(I, a) = \theta^\top f(I, a)$
- Fit global parameter θ from all observed (I_t, a_t) pairs

Benefits over Dirichlet:

- Pro: Generalizes to unobserved infosets (via features)
- Pro: Data-efficient (share statistical strength)
- Con: Requires feature engineering
- Con: Model mis-specification risk

When to use:

- Many infosets with sparse data per infoset
- Clear feature structure (hand strength, pot odds, position)

Softmax Model: Definition

Model:

$$\sigma_{-i}(I, a; \theta) = \frac{\exp(\theta^\top f(I, a))}{\sum_{a' \in A(I)} \exp(\theta^\top f(I, a'))}$$

Components:

- $f(I, a) \in \mathbb{R}^d$: feature vector (hand strength, pot size, action type, ...)
- $\theta \in \mathbb{R}^d$: weight vector (parameters to learn)
- $\theta^\top f(I, a)$: linear score (higher score $\rightarrow$ higher probability)
- Softmax: converts scores to valid probability distribution

Properties:

- Always produces valid probabilities: $\sigma_{-i}(I, a) \in [0, 1]$, $\sum_a \sigma_{-i}(I, a) = 1$
- Smooth: differentiable everywhere
- Temperature can be added: $\exp(\beta\theta^\top f)$ where β controls randomness

Why Softmax? Three Justifications

1. **Multinomial logistic regression (statistical)**
   - Standard GLM for categorical outcomes with covariates
   - Convex negative log-likelihood $\to$ efficient optimization
2. **Maximum entropy (information-theoretic)**
   - Among all distributions matching $\mathbb{E}[f] = \bar{f}$, softmax has max entropy
   - Makes minimal additional assumptions beyond feature constraints
3. **Random utility / Quantal response (decision-theoretic)**
   - Assume opponent chooses action maximizing: $U(I, a) = \theta^\top f(I, a) + \varepsilon_a$
   - If ε_a are i.i.d. Gumbel noise, then $\Pr[a] = \text{softmax}(\theta^\top f)$
   - Interpretation: opponent is "noisy-rational" (higher-value actions more likely)

Bottom line: Softmax is a principled, tractable default for modeling action probabilities

Example Features $f(I, a)$ for Poker

Common features for (infoset, action) pairs:

Hand/Board context:

- Hand strength proxy (if observable or estimated): EHS, pair indicator, draw indicator
- Position (button, big blind, small blind)
- Street (preflop=0, flop=1, turn=2, river=3)

Action properties:

- Action type: fold=0, check=0, call=1, bet=2, raise=3 (one-hot or ordinal)
- Bet size (if raise): fraction of pot, or absolute chips
- Aggression indicator: is this action aggressive (bet/raise) vs. passive (check/call)?

Pot/stack context:

- Pot odds: ratio of call amount to pot
- Stack-to-pot ratio (SPR): remaining stack / current pot

Result: $f(I, a) \in \mathbb{R}^{10-50}$, typically normalize to $[0,1]$ for stability

Fitting the Softmax Model: Maximum Likelihood

Data: Observed action sequences $\mathcal{D} = \{(I_1, a_1), (I_2, a_2), \ldots, (I_T, a_T)\}$

Log-likelihood:

$$\ell(\theta) = \sum_{t=1}^{T} \log \sigma_{-i}(I_t, a_t; \theta)$$

Gradient (for SGD):

$$\nabla_\theta \ell(\theta) = \sum_t \left[f(I_t, a_t) - \sum_{a' \in A(I_t)} \sigma_{-i}(I_t, a'; \theta) \, f(I_t, a') \right]$$

Interpretation:

- Observed action feature: $f(I_t, a_t)$ (push θ toward this)
- Expected feature under model: $\sum_{a'} \sigma_{-i}(I_t, a'; \theta) f(I_t, a')$ (pull back)
- Gradient = observed expected (match feature expectations)

Regularization: Add $-\lambda\|\theta\|^2$ to prevent overfitting; optimize via SGD/Adam

Softmax: Optimization Details

Algorithm: Stochastic Gradient Descent (SGD)

```
Initialize θ^(0) (e.g., zeros or small random)
for epoch = 1 ... E do
    for batch B ⊂ D do
        g ← Σ_{(I,a)∈B} ∇_θ log σ_{-i}(I, a; θ) − λθ
        θ ← θ + η g                                   ▷ learning rate η
    end for
end for
```

Practical notes:

- Negative log-likelihood is convex in $\theta \rightarrow$ converges to global optimum
- Online learning: refit θ periodically as new (I, a) observed
- Feature normalization: scale $f(I, a)$ to similar ranges for numerical stability

Comparison: Dirichlet vs. Softmax

	Dirichlet (Nonparametric)	**Softmax (Parametric)**
Model	Independent per infoset	Shared θ across infosets
Updates	Closed-form (add counts)	Gradient descent (iterative)
Data needs	Requires observations at each ***I***	Generalizes via features
Smoothing	Via prior α	Via regularization λ
Pros	Simple; exact Bayesian; no features needed	Data-efficient; generalizes
Cons	No sharing; sparse-data issues	Requires feature engineering
Best for	Small games; well-observed infosets	Large games; sparse infosets

Hybrid approach: Use softmax for common infosets, Dirichlet for rare ones

Hidden Private Information: The Challenge

Key difficulty in imperfect-information games:
We observe opponent's actions $(a_1, a_2, \dots)$ but **not** their private state x (hole cards)

Why this matters:

- Opponent's infoset $I(x)$ depends on their private state x
- Different private states $\rightarrow$ different infosets $\rightarrow$ potentially different action distributions
- We don't know which x they have, so we don't know which $I(x)$ they're at!

Example (poker):

- Public state S: Flop is K♠ 7♡ 2♣, opponent bets
- We observe: opponent chose "bet"
- But we don't see: opponent's hole cards (AA? KQ? 98s?)
- Their infoset $I(x)$ differs for each holding x
- Each infoset may have different policy $\sigma_{-i}(I(x), \cdot)$

Question: How do we update our models when observations are partial?

Likelihood with Hidden Private States

Observation model: We observe action a_{obs} at public state S, but not opponent's x

Likelihood (marginalize over hidden x):

$$\Pr(a_{\text{obs}} \mid S, \theta) = \sum_{x \in \mathcal{X}(S)} \rho_{-i}(x \mid S) \cdot \sigma_{-i}(I(x), a_{\text{obs}}; \theta)$$

where:

- $\mathcal{X}(S)$ = set of opponent private states consistent with public state S
- $\rho_{-i}(x \mid S)$ = our current range (belief over opponent's private state)
- $I(x)$ = opponent's infoset given private state x and public state S
- $\sigma_{-i}(I(x), a_{\text{obs}}; \theta)$ = probability they play a_{obs} from infoset $I(x)$

Interpretation: Weighted average of action probabilities across possible private states

This is the likelihood for updating θ (parametric) or counts (Dirichlet) when x is latent

Two-Step Update Process

Step 1: Update policy model (from observed actions)
Dirichlet:

- Which infoset was opponent at? Depends on hidden x
- **Exact:** Add fractional counts weighted by $\rho_{-i}(x \mid S)$ to each $n(I(x), a_{\text{obs}})$
- **Approximate:** If infoset doesn't strongly depend on x, treat as single I and add full count

Softmax:

- Likelihood: $\ell(\theta) = \sum_t \log \sum_x \rho_{-i}(x \mid S_t) \cdot \sigma_{-i}(I_t(x), a_t; \theta)$
- Optimize via EM or approximate gradient

Step 2: Update range $\rho_{-i}(x \mid S)$ using Bayes rule with the policy model

In practice: These steps are interleaved (online learning)

Range Update from Observed Action

Bayes rule for range:

$$\rho_{-i}(x \mid S, a_{\text{obs}}) \;\propto\; \rho_{-i}(x \mid S) \cdot \hat{\sigma}_{-i}(I(x), a_{\text{obs}})$$

Normalize over all $x \in \mathcal{X}(S)$ so probabilities sum to 1.

Step-by-step:

1. Start with prior range $\rho_{-i}(x \mid S)$ (from previous state)
2. Observe action a_{obs} at public state S
3. For each possible private state x:
   - Determine infoset $I(x)$ opponent would be at
   - Look up $\hat{\sigma}_{-i}(I(x), a_{\text{obs}})$ from policy model
   - Multiply: $\rho_{-i}(x) \times \hat{\sigma}_{-i}(I(x), a_{\text{obs}})$
4. Normalize to get posterior range

Interpretation: Private states that make a_{obs} more likely gain probability mass

Range Update Example

Scenario: Poker flop, opponent bets. What hands might they have?

Private x	Prior $\rho(x)$	$\hat{\sigma}(I(x), \text{bet})$	Unnorm. posterior	Normalized
AA (overpair)	0.20	0.90	$0.20 \times 0.90 = 0.18$	$0.18/0.27 = 0.67$
KQ (top pair)	0.30	0.60	$0.30 \times 0.60 = 0.18$	$0.18/0.27 = 0.67$
98s (draw)	0.30	0.40	$0.30 \times 0.40 = 0.12$	$0.12/0.27 = 0.44$
72o (nothing)	0.20	0.05	$0.20 \times 0.05 = 0.01$	$0.01/0.27 = 0.04$
		Sum	0.49	—

Interpretation:

- AA and KQ gain probability (betting is consistent with strong hands)
- 72o loses probability (betting is unlikely with trash)
- Draw (98s) moderately consistent with betting (semi-bluff)

Updated range: $\rho_{-i}(x \mid S, \text{bet observed})$ is now more concentrated on strong hands

Sequential Range Updates

Multiple observations across public states: $S_0 \rightarrow S_1 \rightarrow \ldots \rightarrow S_T$

Recursive Bayes rule:

$$\rho_{-i}(x \mid S_t) \propto \rho_{-i}(x \mid S_{t-1}) \cdot \hat{\sigma}_{-i}(I_t(x), a_t)$$

Chained form:

$$\rho_{-i}(x \mid S_T) \propto \rho_{-i}(x \mid S_0) \cdot \prod_{t=1}^{T} \hat{\sigma}_{-i}(I_t(x), a_t)$$

Interpretation:

- Each observed action is a "filter" on the range
- Private states consistent with all actions gain mass
- Private states inconsistent with observed play lose mass
- By showdown (river), range is highly concentrated

Use case: At decision point, use updated ρ_{-i} and $\hat{\sigma}_{-i}$ to compute BR in current subgame

Practical Simplification: PBS-Conditional Models

Full latent-variable treatment is expensive (EM iterations, soft counts)

Approximation: Decouple policy learning and range tracking

1. **Policy model:** Fit $\sigma_{-i}(I, a)$ treating each observation as hard assignment to one infoset
   - Dirichlet: add full count $n(I, a_{\text{obs}}) \leftarrow n(I, a_{\text{obs}}) + 1$
   - Softmax: use observed (I_t, a_t) pairs directly in MLE
2. **Range tracking:** Update $\rho_{-i}(x \mid S)$ separately using the learned policy

$$\rho_{-i}(x \mid S_{\text{new}}) \propto \rho_{-i}(x \mid S_{\text{old}}) \cdot \hat{\sigma}_{-i}(I(x), a_{\text{obs}})$$

When this works well:

- Infosets don't vary dramatically across private states within a PBS
- Or: use PBS-conditional models $\sigma_{-i}(I, a \mid S)$ that adapt to public context

Advantage: Simple, fast, modular (update policy and range independently)

EM-Style Updates (Optional Depth)

For completeness: Full latent-variable approach when $I(x)$ strongly depends on x

E-step: Infer posterior over x given data and current $\theta^{(k)}$:

$$q^{(k)}(x) \;\propto\; \rho_{-i}(x \mid S) \cdot \prod_t \sigma_{-i}(I_t(x), a_t; \theta^{(k)})$$

M-step: Maximize expected log-likelihood:

$$\theta^{(k+1)} = \arg\max_\theta \sum_x q^{(k)}(x) \sum_t \log \sigma_{-i}(I_t(x), a_t; \theta) - \lambda \|\theta\|^2$$

Dirichlet version: Add fractional counts $n(I(x), a_t) \leftarrow n(I(x), a_t) + q^{(k)}(x)$

In practice: Often approximated by PBS-conditional updates (previous slide) or ignored when infoset dependence on x is weak

Posterior Predictive Opponent Policy

After updating from data, what policy do we use?

Dirichlet model:

$$\hat{\sigma}_{-i}(I, a) = \frac{\alpha(I, a) + n(I, a)}{\sum_{a'} (\alpha(I, a') + n(I, a'))}$$

Parametric model: Use fitted $\hat{\theta}$ (MLE or MAP):

$$\hat{\sigma}_{-i}(I, a) = \sigma_{-i}(I, a; \hat{\theta}) = \frac{\exp(\hat{\theta}^\top f(I, a))}{\sum_{a'} \exp(\hat{\theta}^\top f(I, a'))}$$

This is the opponent model we use for:

- Computing best response in current subgame
- Updating ranges: $\rho_{-i}(x) \propto \rho_{-i}(x) \cdot \hat{\sigma}_{-i}(I(x), a_{\text{obs}})$
- Exploitative decision-making (maximizing EV vs. $\hat{\sigma}_{-i}$)

Using the Model: Posterior-Predictive Control

Decision rule: Compute our best response in current subgame to $\hat{\sigma}_{-i}$

Best response computation: Backward induction (single-agent DP on game tree)

- At terminal nodes: return payoff $u_i(z)$
- At chance nodes: expected value over chance distribution
- At opponent nodes: expected value w.r.t. $\hat{\sigma}_{-i}(I, \cdot)$

$$v_i(h) = \sum_{a \in A(I)} \hat{\sigma}_{-i}(I, a) \cdot v_i(h \cdot a)$$

- At our nodes: maximize over actions

$$v_i(h) = \max_{a \in A(I)} v_i(h \cdot a), \quad \mathrm{BR}(I) =_a v_i(h \cdot a)$$

Output: Pure (deterministic) action per infoset (or mixed if desired)

Time complexity: $\mathcal{O}(|Z_{\text{subgame}}|)$ (one tree traversal)

Preview: Robust Blending (Safety for Next Lecture)

Issue: Model error $\Rightarrow$ over-exploitation risks making us exploitable

Blend opponent model with equilibrium baseline:

$$\sigma_{-i}^{\text{blend}}(I,a) = \lambda\,\hat{\sigma}_{-i}(I,a) + (1-\lambda)\,\sigma_{-i}^{\text{NE}}(I,a), \quad \lambda \in [0,1]$$

where:

- $\hat{\sigma}_{-i}$: learned opponent model (Dirichlet or softmax)
- σ_{-i}^{NE}: equilibrium policy (from blueprint or known Nash)
- λ: exploitation parameter ($0 =$ pure equilibrium, $1 =$ pure exploitation)

Use: Compute BR to $\sigma_{-i}^{\text{blend}}$ for risk-controlled exploitation

Next lecture: Restricted Nash Response (RNR), safety constraints, robust optimization

Online Update Pipeline (Runtime)

At each decision point:

1. **Observe:** Public state S and opponent action a_{obs} at some infoset
2. **Update policy model:**
   - Dirichlet: increment $n(I, a_{\text{obs}})$ (possibly fractional if latent x)
   - Softmax: add (I, a_{obs}) to dataset; refit θ periodically
3. **Update range:** Apply Bayes rule

$$\rho_{-i}(x \mid S_{\text{new}}) \propto \rho_{-i}(x \mid S) \cdot \hat{\sigma}_{-i}(I(x), a_{\text{obs}})$$

4. **Form posterior-predictive policy:** $\hat{\sigma}_{-i}$ across all infosets in current subgame
5. **Compute decision:** Best response to $\hat{\sigma}_{-i}$ in subgame (or blended policy)
6. **Execute action:** Play chosen action; update public state

Pseudocode: Online Opponent Modeling + Decision

function UPDATEANDDECIDE(S, a_{obs}, model)
    // Step 1: Update policy model
    if model.type == Dirichlet **then**
        Infer likely infosets from range and S
        $n(I, a_{\text{obs}}) \leftarrow n(I, a_{\text{obs}}) + 1$ ▷ or fractional
        $\hat{\sigma}_{-i}(I, \cdot) \leftarrow (\alpha + n)/(\alpha_0 + N)$
    else ▷ Softmax
        Add (I, a_{obs}) to $\mathcal{D}$
        if time to refit **then** ▷ periodic, e.g., every 100 obs
            $\theta \leftarrow$ FitSoftmax($\mathcal{D}$) via SGD
        end if
        $\hat{\sigma}_{-i}(I, \cdot) \leftarrow \text{softmax}(\theta^\top f(I, \cdot))$
    end if
    // Step 2: Update range via Bayes rule
    for each $x \in \mathcal{X}(S)$ **do**
        $\rho_{-i}(x) \leftarrow \rho_{-i}(x) \cdot \hat{\sigma}_{-i}(I(x), a_{\text{obs}})$
    end for
    Normalize ρ_{-i}
    // Step 3: Compute exploitative action

Evaluating Opponent Models

How do we know if our model is good?

Metrics:

- **Log-likelihood** on held-out action sequences

$$\text{Test-LL} = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(I,a)\in\mathcal{D}_{\text{test}}} \log \hat{\sigma}_{-i}(I, a)$$

  Higher is better (model assigns high probability to observed actions)

- **Cross-entropy / Perplexity:**

$$\text{Perplexity} = \exp(-\text{Test-LL})$$

  Lower is better; measures "surprise" per decision

- **Calibration:** Compare predicted $\hat{\sigma}_{-i}(I, a)$ to empirical frequencies
  - Plot predicted prob vs. actual frequency (should lie on diagonal)

Diagnostic: High variance across PBS $\Rightarrow$ use PBS-conditional models

Hierarchical Priors and Parameter Sharing

Motivation: Many infosets have little data; need to share information

Sharing schemes:

1. **Group by PBS:**
   - Use same prior α for all infosets within a public state S
   - Or same θ (softmax) with PBS-specific features
2. **Feature-based sharing (softmax):**
   - Single global θ; features $f(I, a)$ adapt to context
   - Automatically shares across similar (infoset, action) pairs
3. **Hierarchical Bayesian (Dirichlet):**
   - Hyperprior on α: $\alpha \sim \text{Gamma}(\ldots)$ or similar
   - Partial pooling: infosets "borrow strength" from each other
   - Requires inference (MCMC or variational)

Trade-off: More sharing $\rightarrow$ more robust with sparse data, but less flexible per infoset

Complexity and Data Requirements

Computational cost:

- **Dirichlet update:** $\mathcal{O}(1)$ per observation (just increment counts)
- **Softmax fit:** $\mathcal{O}(|\mathcal{D}| \cdot d \cdot |A|)$ per SGD epoch (d = feature dim)
- **Range update:** $\mathcal{O}(|\mathcal{X}(S)| \cdot |A|)$ (enumerate private states)
- **Best response:** $\mathcal{O}(|Z_{\text{subgame}}|)$ per decision

Data requirements:

- Sparse infosets: need strong priors (α) or parameter sharing (θ)
- PBS-conditional aggregation reduces sparsity
- Rule of thumb: $\sim$10–100 observations per infoset for stable Dirichlet; fewer for softmax with good features

Memory:

- Dirichlet: store $\alpha(I, a)$ and $n(I, a)$ for each (I, a) pair
- Softmax: store $\theta \in \mathbb{R}^d$ (much smaller if $d \ll |\mathcal{I}| \times |A|$)

Caveats and Pitfalls

1. **Concept drift:** Opponent changes strategy over time
   - Use forgetting factors: decay old counts/observations
   - Sliding window: only use recent k observations
   - Detect shifts: monitor likelihood or regret; reset model if needed
2. **Exploration vs. exploitation:** Need data to learn, but gathering data may be costly
   - Occasionally probe with unusual actions (exploration)
   - Safe blending (next lecture): hedge against model error
3. **Model mis-specification:**
   - Wrong features (softmax) or wrong infoset grouping (Dirichlet)
   - Diagnostics: poor test log-likelihood, miscalibration
   - Remedies: better features, PBS conditioning, non-linear models (neural nets)
4. **Abstraction mismatch:**
   - Infoset mapping differs between training and deployment
   - Test cross-PBS and cross-abstraction robustness

Summary

Opponent modeling in imperfect-information EFGs:

Policy models:

- **Dirichlet:** Bayesian per-infoset; α provides smoothing and prior beliefs; fast closed-form updates
- **Softmax:** Feature-based sharing; θ learned via MLE; generalizes across infosets

Hidden private information:

- Observations reveal actions but not opponent's private state x
- Marginalize over x (exact EM) or approximate (PBS-conditional)

Range tracking:

- Update $\rho_{-i}(x \mid S)$ via Bayes rule using policy model
- Sequential filtering: $\rho \propto \rho_{\text{prev}} \times \hat{\sigma}_{-i}(I(x), a_{\text{obs}})$

Posterior-predictive control:

- Use $\hat{\sigma}_{-i}$ and ρ_{-i} to compute best response in current subgame
- Exploitative play vs. learned opponent

Next lecture: Safe exploitation (RNR, robust blending, constraints)

References

- **Billings et al. (1998, 2002):** Opponent modeling in poker (foundational work)
- **Johanson & Bowling (2009):** “Data Biased Robust Counter Strategies” (AISTATS) — introduces Restricted Nash Response
- **Ganzfried & Sandholm (2011):** “Game Theory-Based Opponent Modeling in Large Imperfect-Information Games” (AAMAS)
- **Brown & Sandholm (2017):** “Safe and Nested Subgame Solving” (NIPS) — re-solving context
- **OpenSpiel documentation:** PBS, ranges, best response implementations