Machine Learning for Behavioral Game Theory

Modeling Strategic Behavior as a Machine Learning Problem

Kevin Leyton-Brown

University of British Columbia
Canada CIFAR AI Chair, Amii

James R. Wright

University of Alberta
Canada CIFAR AI Chair, Amii

THE UNIVERSITY OF BRITISH COLUMBIA

UNIVERSITY OF ALBERTA

Lecture Overview

Recap: Behavioral Game Theory

- **Descriptive** models, not normative
- **QRE:** All agents quantally best respond to each other
- **CH:** Level-0 agents do something (uniform?), level-1 agents best respond to level-0, level-2 agents best respond to mix of level-0 and level-1, ...
- **QCH:** Level-0 agents do something (uniform?), level-1 agents **quantally** best respond to level-0, level-2 agents **quantally** best respond to mix of level-0 and level-1, ...
- **Linear4:** One story about the "something" that level-0 do: linear combination of simple rules.

Recap: Behavioral Game Theory

- **Descriptive** models, not normative
- **QRE:** All agents quantally best respond to each other
- **CH:** Level-0 agents do something (uniform?), level-1 agents best respond to level-0, level-2 agents best respond to mix of level-0 and level-1, ...
- **QCH:** Level-0 agents do something (uniform?), level-1 agents **quantally** best respond to level-0, level-2 agents **quantally** best respond to mix of level-0 and level-1, ...
- **Linear4:** One story about the "something" that level-0 do: linear combination of simple rules.
- Every model has parameters that need to be set:
  - QRE, QCH: Precision parameter λ
  - CH, QCH: Distribution of levels $\alpha_0, \ldots, \alpha_K$
  - Linear4: Rule weights $w_{\text{unif}}, \ldots, w_{\text{maxmax}}$

Lecture Overview

Nice story—but are these models any good?

Let's say we pay a bunch of people to play games against each other, and gather some data. Now we'd like to know how good a job our (e.g., QRE) model does. How would we do that?

Two issues:

- have to set the model's **parameter** (λ) to use it at all;
- must ensure that we do this in a way that **generalizes** to new play by the same people.

Generalization

- We don't want to predict behavior in just a single, known game
- If that's all we wanted, we wouldn't need a model at all; we could just treat it like a multinomial prediction problem
- Instead, we want to choose a model that performs well on the games in our dataset, *and also* on new, unseen games

Generalization

- We don't want to predict behavior in just a single, known game
- If that's all we wanted, we wouldn't need a model at all; we could just treat it like a multinomial prediction problem
- Instead, we want to choose a model that performs well on the games in our dataset, *and also* on new, unseen games
- **Question:** Why would we care about predicting behavior in new, unseen games?

Supervised Learning Approach (MLE)

One approach:

1. Gather a bunch of gameplay data for multiple games (**why multiple games?**)
2. Treat each action by a participant as an i.i.d. draw from s predicted by the model
3. Optimize model parameters on the training set

Comparing Models (MLE)

- Randomly partition our data into different sets: $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$
- Choose parameter value(s) that **maximize the likelihood** of the training data:

$$\vec{\theta}^* = \arg\max_{\vec{\theta}} \Pr(\mathcal{D}_{\text{train}} \mid \mathcal{M}, \vec{\theta})$$

  where $\Pr(\mathcal{D}_{\text{train}} \mid \vec{\theta}) = \prod_{p=1}^{n} s_i(a_i^{(p)})$.
- Score the performance of a model by the likelihood of the **test data**:

$$\Pr(\mathcal{D}_{\text{test}} \mid \mathcal{M}, \vec{\theta}^*).$$

- To reduce variance, repeat this process multiple times with different random partitions and average the results

Log Likelihood is Annoying

- Good news about LL
  1. Obvious probabilistic interpretation
  2. Proper scoring rule
  3. Locality
- Bad news about LL: Everything else
  - If I tell you that the test-set accuracy of a model was 0.9998, that is good!
  - If I tell you that the test-set log-likelihood of a model is -936, that is... ??

Log Likelihood is Annoying

- Good news about LL
  1. Obvious probabilistic interpretation
  2. Proper scoring rule
  3. Locality
- Bad news about LL: Everything else
  - If I tell you that the test-set accuracy of a model was 0.9998, that is good!
  - If I tell you that the test-set log-likelihood of a model is -936, that is... ??
- Log-likelihood depends on **entropy of underlying data**
  - Higher-entropy distributions are harder to predict
- Log-likelihood depends on **size of dataset**
  - Larger datasets will have worse log-likelihoods

Coping Strategies

Question: How can we deal with these issues?

Coping Strategies

Question: How can we deal with these issues?

1. Baselines
   - Subtract the log-likelihood that would have been achieved by a uniform prediction
   - Interpretation: How many times more likely is the data according to the model than according to the uniform prediction?
   - Sort of deals with the entropy issue (can combine high-entropy and low-entropy data)
   - Deals a little bit with the dataset size issue

Coping Strategies

Question: How can we deal with these issues?

1. Baselines
   - Subtract the log-likelihood that would have been achieved by a uniform prediction
   - Interpretation: How many times more likely is the data according to the model than according to the uniform prediction?
   - Sort of deals with the entropy issue (can combine high-entropy and low-entropy data)
   - Deals a little bit with the dataset size issue
2. "Completeness" evaluation
   - [Fudenberg et al., 2021]: "Measuring the Completeness of Economic Models"
   - Use a baseline (as above)
   - Additionally, use a topline:
     - Performance of empirical frequencies
     - Performance of uninterpretable, high-capacity ML model (next section)
   - Normalize so that baseline performance is 0, topline performance is 1

Coping Strategies

Question: How can we deal with these issues?

1. Baselines
   - Subtract the log-likelihood that would have been achieved by a uniform prediction
   - Interpretation: How many times more likely is the data according to the model than according to the uniform prediction?
   - Sort of deals with the entropy issue (can combine high-entropy and low-entropy data)
   - Deals a little bit with the dataset size issue
2. "Completeness" evaluation
   - [Fudenberg et al., 2021]: "Measuring the Completeness of Economic Models"
   - Use a baseline (as above)
   - Additionally, use a topline:
     - Performance of empirical frequencies
     - Performance of uninterpretable, high-capacity ML model (next section)
   - Normalize so that baseline performance is 0, topline performance is 1
   - Benefit: Completely interpretable, portable between datasets
   - Drawback: I'm not sure I buy that simple division is the right normalization for log-likelihood

Significance Testing

- When you are comparing models, you don't want to just compare average test-set performance (**why?**)

Significance Testing

- When you are comparing models, you don't want to just compare average test-set performance (**why?**)
- Test performance can depend on the random division of training set and test set
- Repeating division and averaging can get more samples from this distribution
- But we want to have confidence intervals that give us a sense for how much variance is in this distribution
- How can we compute confidence intervals?

Significance Testing

- When you are comparing models, you don't want to just compare average test-set performance (**why?**)
- Test performance can depend on the random division of training set and test set
- Repeating division and averaging can get more samples from this distribution
- But we want to have confidence intervals that give us a sense for how much variance is in this distribution
- How can we compute confidence intervals?
- My favourite approach is to assume a t-**distribution**:
  1. I have an average of several performances
  2. What is the 95% confidence interval for the "true" average from this distribution?
  3. t-distribution because I usually don't have enough samples to use Gaussian approximation

10-fold Cross-Validation

I usually use **10 repetitions of 10-fold cross-validation.**

I.e., repeat the following 10 times:

1. Partition full dataset into 10 roughly-equal-sized "folds"
   - Input to the model is really the game, so make sure all the data for a given game goes into a single fold
2. For $j \in \{1, \ldots, 10\}$, training set is all folds but jth, test set is j fold
3. Train on training set, test on test set
4. Performance of this repetition is the total test performance over all 10 folds

10-fold Cross-Validation

I usually use **10 repetitions of 10-fold cross-validation.**

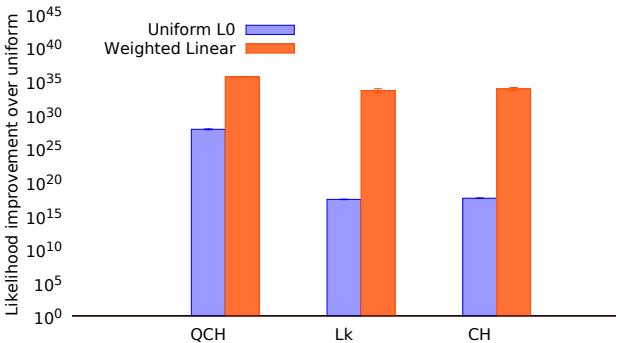
I.e., repeat the following 10 times:

1. Partition full dataset into 10 roughly-equal-sized "folds"
   - Input to the model is really the game, so make sure all the data for a given game goes into a single fold
2. For $j \in \{1, \ldots, 10\}$, training set is all folds but jth, test set is j fold
3. Train on training set, test on test set
4. Performance of this repetition is the total test performance over all 10 folds

Benefits:

- Every datapoint gets used as a test point exactly once, regardless of the partition
- So all test log-likelihoods will be on exactly the same scale
- If you just randomly hold out $x\%$ of your data each time, the log-likelihoods will all be on a different scale (due to different-entropy distributions of test data)

Example: Model Comparison



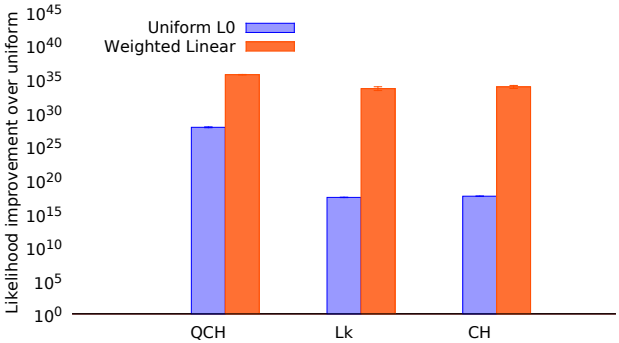
Two **level-0 meta-models**:

1. Uniform L0
2. Weighted Linear

Three **iterative models**:

1. Quantal Cognitive Hierarchy
2. Level-k
3. Cognitive Hierarchy

Example: Model Comparison



- Linear4 model for level-0 agents dramatically improved the performance of **all three** iterative models.
  - Almost erases the difference between the models themselves.

Multiple Observations (Panel Data)

- If you have multiple observations from each participant, you can treat each **set of actions** by a participant as an i.i.d. sample from a more complicated distribution.
- The exact distribution depends on which parameters are assumed to be global vs. agent-specific

Multiple Observations (Panel Data)

- If you have multiple observations from each participant, you can treat each **set of actions** by a participant as an i.i.d. sample from a more complicated distribution.
- The exact distribution depends on which parameters are assumed to be global vs. agent-specific
- Example: Level-k model, n observations each from m participants

$$\Pr(\mathcal{D}_{\text{train}} \mid \vec{\alpha}) = \prod_{p=1}^{m} \sum_{k=1}^{K} \alpha_k \prod_{j=1}^{n} \pi_{i,k}(a_i^{(p,j)})$$

- Assumption: Every agent has a stable level, $\vec{\alpha}$ give distributions of levels in population

Multiple Observations (Panel Data)

- If you have multiple observations from each participant, you can treat each **set of actions** by a participant as an i.i.d. sample from a more complicated distribution.
- The exact distribution depends on which parameters are assumed to be global vs. agent-specific
- Example: Level-k model, n observations each from m participants

$$\Pr(\mathcal{D}_{\text{train}} \mid \vec{\alpha}) = \prod_{p=1}^{m} \sum_{k=1}^{K} \alpha_k \prod_{j=1}^{n} \pi_{i,k}(a_i^{(p,j)})$$

- Assumption: Every agent has a stable level, $\vec{\alpha}$ give distributions of levels in population
- **Question:** What would the likelihood be if we instead assumed that agents did *not* have a stable level, and re-sampled from $\vec{\alpha}$ every time?

Bayesian Parameter Analysis

- All of the BGT models discussed so far have intuitive meanings
- You might be interested in the values of the parameters for their own sake
- But you probably should *not* just interpret the MLE-fitted parameters on their own (**why?**)

Bayesian Parameter Analysis

- All of the BGT models discussed so far have intuitive meanings
- You might be interested in the values of the parameters for their own sake
- But you probably should *not* just interpret the MLE-fitted parameters on their own (**why?**)
- One alternative: Estimate **posterior distribution** of the parameters

$$\Pr(\vec{\theta} \mid \mathcal{D}) = \frac{\Pr(\mathcal{D} \mid \vec{\theta}) \Pr(\vec{\theta})}{\int_{\Theta} \Pr(\mathcal{D} \mid \vec{\theta}) \Pr(\vec{\theta}) \, d\vec{\theta}}$$

Bayesian Parameter Analysis

- All of the BGT models discussed so far have intuitive meanings
- You might be interested in the values of the parameters for their own sake
- But you probably should *not* just interpret the MLE-fitted parameters on their own (**why?**)
- One alternative: Estimate **posterior distribution** of the parameters

$$\Pr(\vec{\theta} \mid \mathcal{D}) = \frac{\Pr(\mathcal{D} \mid \vec{\theta}) \Pr(\vec{\theta})}{\int_{\Theta} \Pr(\mathcal{D} \mid \vec{\theta}) \Pr(\vec{\theta}) \, d\vec{\theta}}$$

- Requires specification of a **prior** over the parameters

Bayesian Parameter Analysis

- All of the BGT models discussed so far have intuitive meanings
- You might be interested in the values of the parameters for their own sake
- But you probably should *not* just interpret the MLE-fitted parameters on their own (**why?**)
- One alternative: Estimate **posterior distribution** of the parameters

$$\Pr(\vec{\theta} \mid \mathcal{D}) = \frac{\Pr(\mathcal{D} \mid \vec{\theta}) \Pr(\vec{\theta})}{\int_{\Theta} \Pr(\mathcal{D} \mid \vec{\theta}) \Pr(\vec{\theta}) \, d\vec{\theta}}$$

- Requires specification of a **prior** over the parameters
- Integral is often non-tractable
  - But can use Monte Carlo approximation with standard tools (e.g., `pymc3`)
- Multi-dimensional visualization is hard, but often the marginals are informative

Example: Level Parameter Analysis

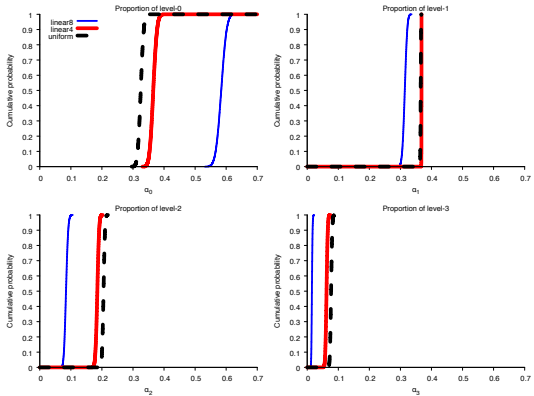


Figure 4: Marginal cumulative posterior distributions of levels of reasoning in the ALL10 dataset, for Poisson-QCH with `linear8`, `linear4`, and uniform specifications.

- Narrow width of the CDFs indicates that data argue strongly for a specific value
- But different models back out qualitatively different parameter values (**why?**)

Lecture Overview

Something Completely Different

- What if we don't care about peering inside people's heads?
- **Question:** Can we just throw a neural network at this? (**why or why not?**)

Direct Application of a Feedforward Network

You can construct a multilayer feedforward network for normal-form games:

- One input node per payoff
- One output node per action of player being predicted
- Softmax over outputs to get predicted distribution

Direct Application of a Feedforward Network

You can construct a multilayer feedforward network for normal-form games:

- One input node per payoff
- One output node per action of player being predicted
- Softmax over outputs to get predicted distribution

BUT

- You need to pick a maximum number of actions for each player (and a maximum number of players!)
- Learning about one game doesn't tell you anything about a strategically identically game with permuted action labels
  - E.g., need to learn concept of dominance separately for each pair of actions, potentially
- This kind of model can have a very large number of parameters, even for very small games

Convolutional Neural Nets

These are many of the same problems that CNNs solve for image recognition:

- Want to allow inputs of varying sizes without retraining the model
- Want to allow generalization across a set of symmetries (Images: translation equivariance; Games: permutation equivariance)
- Want to exploit symmetries to reduce the number of parameters that need to be learned

GameNet [Hartford et al., 2016, 2018]

Can construct a particular kind of CNN for learning behavior in 2-player NFGs:

GameNet [Hartford et al., 2016, 2018]

Can construct a particular kind of CNN for learning behavior in 2-player NFGs:

1. Inputs: Payoff matrices for each player

GameNet [Hartford et al., 2016, 2018]

Can construct a particular kind of CNN for learning behavior in 2-player NFGs:

1. Inputs: Payoff matrices for each player

   Multiple iterations of convolution and pooling:
2. Convolutions: 1×1 convolution filters
   - Effectively: scale each payoff matrix by a single value

GameNet [Hartford et al., 2016, 2018]

Can construct a particular kind of CNN for learning behavior in 2-player NFGs:

1. Inputs: Payoff matrices for each player

Multiple iterations of convolution and pooling:

2. Convolutions: 1×1 convolution filters
   - Effectively: scale each payoff matrix by a single value
3. Pooling: Max/sum over columns, rows
   - Then duplicate out to get to same dimensions

GameNet [Hartford et al., 2016, 2018]

Can construct a particular kind of CNN for learning behavior in 2-player NFGs:

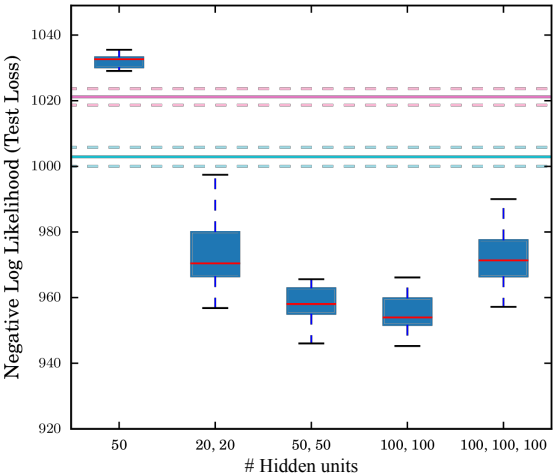
1. Inputs: Payoff matrices for each player

Multiple iterations of convolution and pooling:

2. Convolutions: 1×1 convolution filters
   - Effectively: scale each payoff matrix by a single value
3. Pooling: Max/sum over columns, rows
   - Then duplicate out to get to same dimensions

4. Output layer: weighted combination of final pooled vectors
   - Dimension will change depending on dimensions of inputs

Example: GameNet Performance



- Pink line is QCH with uniform level-0
- Blue line is QCH with linear4 level-0
- Just two layers of 50 filters each is sufficient to significantly outperform cognitive-based models
- Good: Didn't need any access to hand-crafted features, models, etc.
- Bad: Requires > 2500 parameters!

Cognitive Models vs. Opaque Models

Cognitive models (QRE, QCH, etc.):

- Many fewer parameters
- Empirical content is easier to interpret
- Stronger assumptions may lead to more robust generalization

GameNet:

- Best known prediction performance
- Data-driven generalization
  - No need to intuit/introspect/hand-craft features
- Many parameters (although fewer than standard neural models)
- "Black box" model
- What are the assumptions/empirical content of the architecture?

Summary

- Parameterized behavioral game theory models can be fitted and compared using standard supervised learning techniques
- Parameters of cognitively-inspired models can be interesting for their own sake
- Black-box ML models (CNNs) do an even better job of predicting NFG behavior than BGT models
  - Some special domain-specific issues
  - Cognitive models and black-box models each have benefits and drawbacks
- Next time: Examples of going beyond the normal form
  - Repeated play (BGT)
  - Repeated play: No-regret as a behavioral assumption
  - Bayesian games