Using Hidden Markov Models for Opponent Modeling

A review and python implementation, by Sina Azartash

Overview

Opponent Modeling (OM)

Opponent does not have to be an adversary. Can be another player.

- What is OM and what is its purpose?
- Ethical concerns
- Advantages provided by OM
- OM architectures paradigms
- OM Algorithms & then Hidden Markov Models
- Application to the Iterated Prisoner's Dilemma
- Algorithm Foundations
- Experimentation Python Implementation
- Statistical Significance testing

Purpose

What → To learn the strategy of another player in a game:

- ❖ Uses Prior Knowledge and/or observed actions
- ❖ Why:
  - ➢ To predict behavior
  - ➢ To exploit predictions
  - ➢ To defend against exploitation
- ❖ Intelligent Software that can predict your chess strategy

Friday, 1 February, 2019

How a Computer Plays Chess: an excerpt from "Playing Smart"

by Julian Togelius

The approach almost all Chess-playing programs take is to use some variant of the *minimax* algorithm. This is actually a very simple algorithm. It works with the

Ethical Concerns

- ❖ OM enables learning of preferences and strategies of individuals
- ❖ Teams of Scientists & Engineers are employed to use OM to devise new strategies to influence users.
- ❖ Can use multiple information sources to create user profile eroding online privacy

Four Advantages to OM

1. Exploit risk of Opponent
    a. Identify where strategy has taken risk
    b. Identify where opponent strategy deviates from the long-term standard
2. Faster strategy detection
    a. Can detect and then respond to strategies even before other player finishes executing their moves
    b. Can use the extra time to deploy a counter strategy
3. Identify Opponent Weakness
    a. Play the strategy that incurs the highest likelihood of causing the opponent to struggle
    b. Use most effective strategy personalized to opponent
4. Avoid risk being Exploited by Opponent
    a. Increase player safety and reduce uncertainty of opponent strategy
    b. Identify risks opponent is least likely to detect

OM Architectures Paradigm

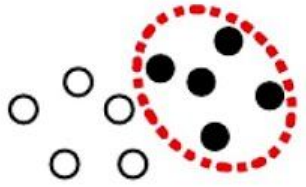
1. Data collection method
    a. Extracting and observing behavior
    b. Preprocessing & Data Structure
    c. Connecting actions to specific agent
    d. Expert Knowledge, Incorporating Domain Knowledge
2. Learning Algorithm
    a. Game Theory Algorithms
    b. Statistics
    c. Machine Learning Algorithms: Support Vector Machines, Decision Trees, Neural Networks, and Multi-Agent Reinforcement Learning
3. Decision Making Abstraction
    a. low -level decision = best interest of single agent
    b. High-level decision = best interest of entire population
    c. Mid-level decisions = best interest of group

OM Algorithm Types

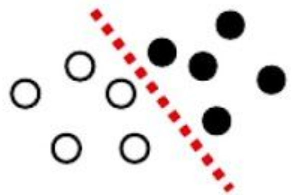
1. Discriminative Role or Strategy Classification
    a. Supervised learning
    b. Support vector machines, Case-based reasoning, Expert Systems, Game Theory Algorithms
2. Goal Based Generative Models
    a. Compare likelihood of actions with probability of strategy
    b. Hidden Markov Models, Bayesian Networks, Neural Networks, Expert Systems
3. Policy approximation
    a. Reinforcement learning: Model the problem sequential decision making in state action pairs
    b. Calculate a policy as an approximation of the strategy
    c. Abstract Markov Models, Deterministic Automata, Deep Neural Networks
    d. Partially Observable Markov Decision Process

Generative vs. Discriminative

- Generative:
  - probabilistic "model" of each class
  - decision boundary:
    - where one model becomes more likely
  - natural use of unlabeled data



- Discriminative:
  - focus on the decision boundary
  - more powerful with lots of examples
  - not designed to use unlabeled data
  - only supervised tasks



Hidden Markov Models

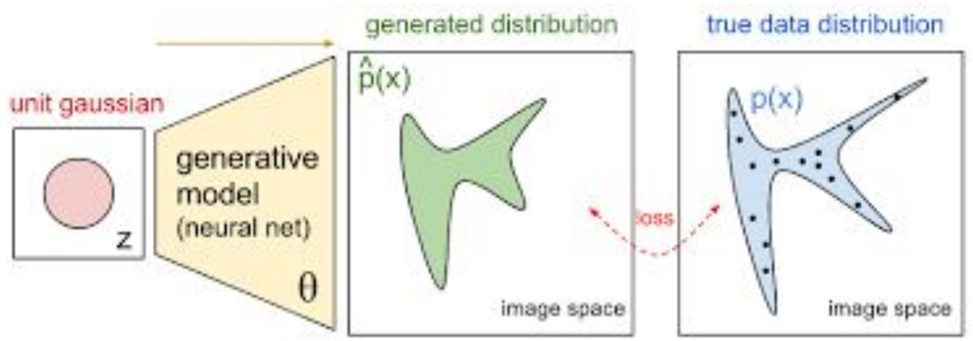


Image by https://openai.com/blog/generative-models/

- ❖ Generative Model
  - ➢ Iteratively updates conditional probability distributions
  - ➢ Generates samples of each strategy
  - ➢ Compares generated distribution(inferred strategy) with true data distribution(true strategy)

1. Likelihood Computation
   a. Find how likely an action is given several different strategies
2. Decoding
   a. Find which strategy most likely produced the actions
3. Learning
   a. Correct mistakes and improve predictions overtime with more samples

OM Markov Modeling

- Markov Property
  - The current state depends only on the previous state
- Hidden State
  - The intention of a player at a hidden time
  - Guided by the strategy
- Observed State
  - The action the player has taken
  - Results from the hidden state
- Transition Matrix
  - Hidden state → hidden state
  - Describes probability of switching to a different state or staying on current state
- Emission Matrix
  - Hidden state → actions
  - Describes probability of actions aligning with hidden state

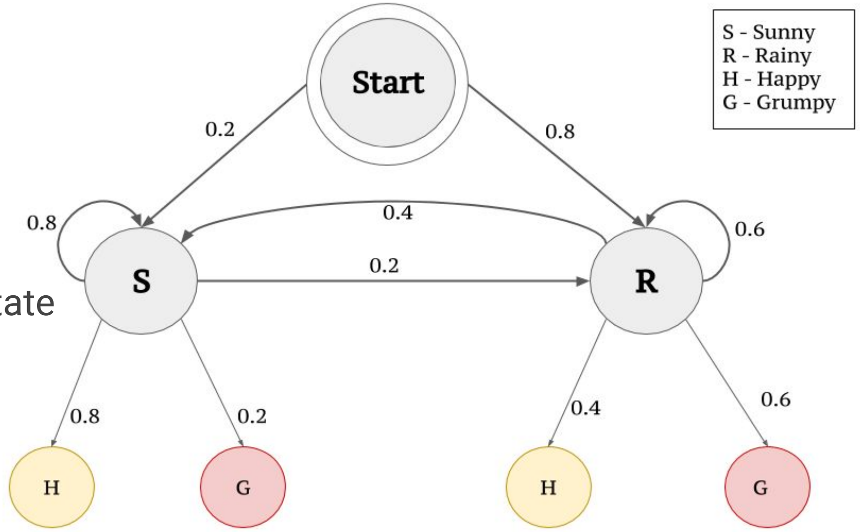
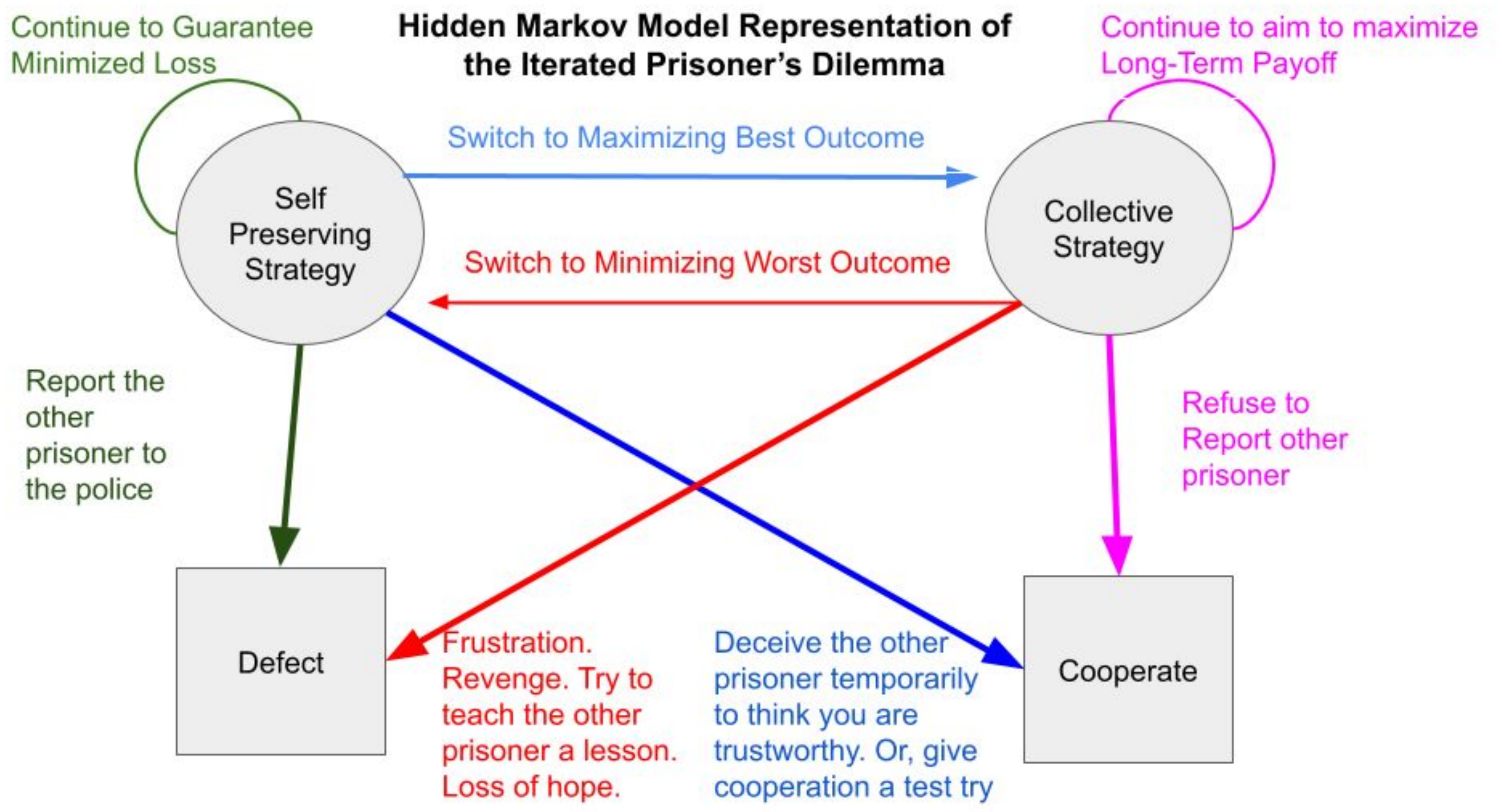


Image by Vivek Vinushanth Christopher

HMM Modeling Of the Iterated Prisoner Dilemma


Hidden Markov Model Representation of the Iterated Prisoner's Dilemma
Continue to Guarantee Minimized Loss
Continue to aim to maximize Long-Term Payoff
Switch to Maximizing Best Outcome
Self Preserving Strategy
Collective Strategy
Switch to Minimizing Worst Outcome
Report the other prisoner to the police
Refuse to Report other prisoner
Defect
Frustration. Revenge. Try to teach the other prisoner a lesson. Loss of hope.
Deceive the other prisoner temporarily to think you are trustworthy. Or, give cooperation a test try
Cooperate

The Forward Algorithm

Minimizes probability of a sequence of actions given hidden strategy of opponent

- ❑ $P(Y_0, Y_1, Y_0) \rightarrow$ probability of a defect action, then cooperate, then defect
- ❑ $P(Y_0 \mid X_0) * P(X_0) \rightarrow$ posterior probability of a defect action given a selfish strategy multiplied by the probability of a selfish strategy

- ❑ $P(Y_0 \mid X_0) * P(X_0) * P(X_1 \mid X_0) * P(Y_1 \mid X_1) * P(X_1 \mid X_1) * P(Y_1 \mid X_1)$
  - ❑ Posterior probability of a defect given selfish state * prob of selfish strategy
  - ❑ Probability of transitioning to cooperative hidden state
  - ❑ Posterior probability of cooperation given cooperative state
  - ❑ Probability of staying on cooperative state
  - ❑ Posterior probability of going against state and choosing to defect instead given cooperative strategy

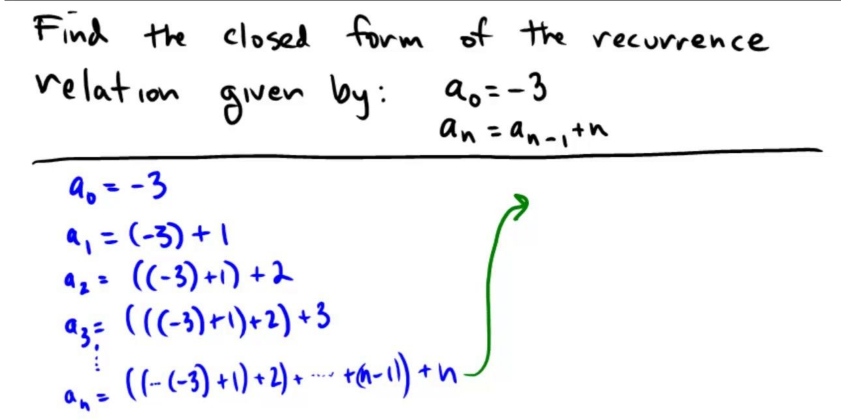
Recurrence Relations

The nth term of a sequence can be based on the n-1 state (enable Markov Property)

$$\alpha_t(X_i) = \sum_{j=0}^{n-1} \boxed{\alpha_{t-1}(X_j)}\,\boxed{P(X_i|X_j)}\,\boxed{P(Y^t|X_i)}$$

Probability of a sequence of actions:

1. Yellow → dependent on previous state
2. Green → probability of hidden state transition
3. Red → conditional probability of action state given hidden state



Find the closed form of the recurrence relation given by: $a_0 = -3$
$a_n = a_{n-1} + n$

$a_0 = -3$

$a_1 = (-3) + 1$

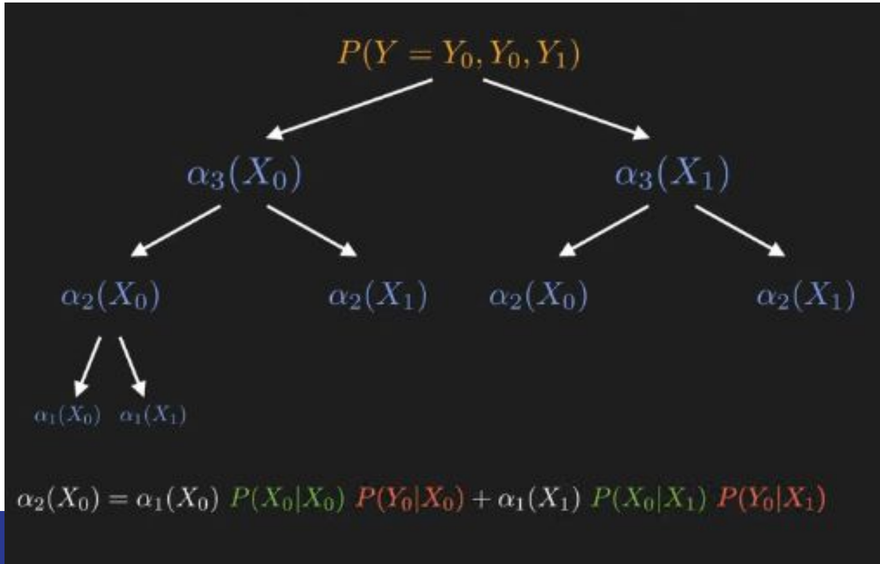
$a_2 = ((-3)+1) + 2$

$a_3 = (((-3)+1)+2) + 3$

$\vdots$

$a_n = ((\ldots(-3)+1)+2) + \cdots + (n-1)) + n$

Image screenshot of video "Finding a solution to a recurrence relation", *by Joshua Helston*



$$\begin{tikzcd}
 & & & P(Y = Y_0, Y_0, Y_1) \arrow[dll] \arrow[drr] & & & \\
 & \alpha_3(X_0) \arrow[dl] \arrow[dr] & & & & \alpha_3(X_1) \arrow[dl] \arrow[dr] & \\
\alpha_2(X_0) \arrow[d] \arrow[dr] & & \alpha_2(X_1) & & \alpha_2(X_0) & & \alpha_2(X_1) \\
\alpha_1(X_0) & \alpha_1(X_1) & & & & &
\end{tikzcd}$$

$$\alpha_2(X_0) = \alpha_1(X_0)\ P(X_0|X_0)\ P(Y_0|X_0) + \alpha_1(X_1)\ P(X_0|X_1)\ P(Y_0|X_1)$$

Image screenshot of video "Markov Chains Explained", https://www.youtube.com/watch?v=i3AkTO9HLXo

5 Strategies Studied in Prisoner’s Dilemma

Always Defect	New Strategy		
Old Strategy		Selfish	Collective
	Selfish	1.0	0.0
	Collective	1.0	0.0

Always Cooperate	New Strategy		
Old Strategy		Selfish	Collective
	Selfish	0.0	1.0
	Collective	0.0	1.0

- 1 indicates 100% probability
- These strategy tables represent the transition matrix of switching between hidden states

- Strategy → (transition matrix) → hidden state → (emission matrix) → action
- Observed Action → inferred opponent hidden state → inferred opponent strategy

5 Strategies Studied in Prisoner's Dilemma cont

Stubborn	New Strategy		
Old Strategy		Selfish	Collective
	Selfish	0.95	0.05
	Collective	0.05	0.95

Ambivalent	New Strategy		
Old Strategy		Selfish	Collective
	Selfish	0.65	0.35
	Collective	0.35	0.65

Average	New Strategy		
Old Strategy		Selfish	Collective
	Selfish	0.85	0.15
	Collective	0.15	0.85

- Emotional inertia:
  - People are more likely to continue on their strategy than switch to a new strategy
  - 85% chance on staying current strategy is considered as average for a player

Predicting Future Actions

$\text{argmax}(P_1, P_2, P_3, P_4)$

$P_1 = P(Y_0, Y_0, Y_1, Y_0, Y_0 \mid \text{ambivalent})$ $P_2 = P(Y_0, Y_0, Y_1, Y_1, Y_0 \mid \text{ambivalent})$

$P_3 = P(Y_0, Y_0, Y_1, Y_0, Y_1 \mid \text{ambivalent})$ $P_4 = P(Y_0, Y_0, Y_1, Y_1, Y_1 \mid \text{ambivalent})$

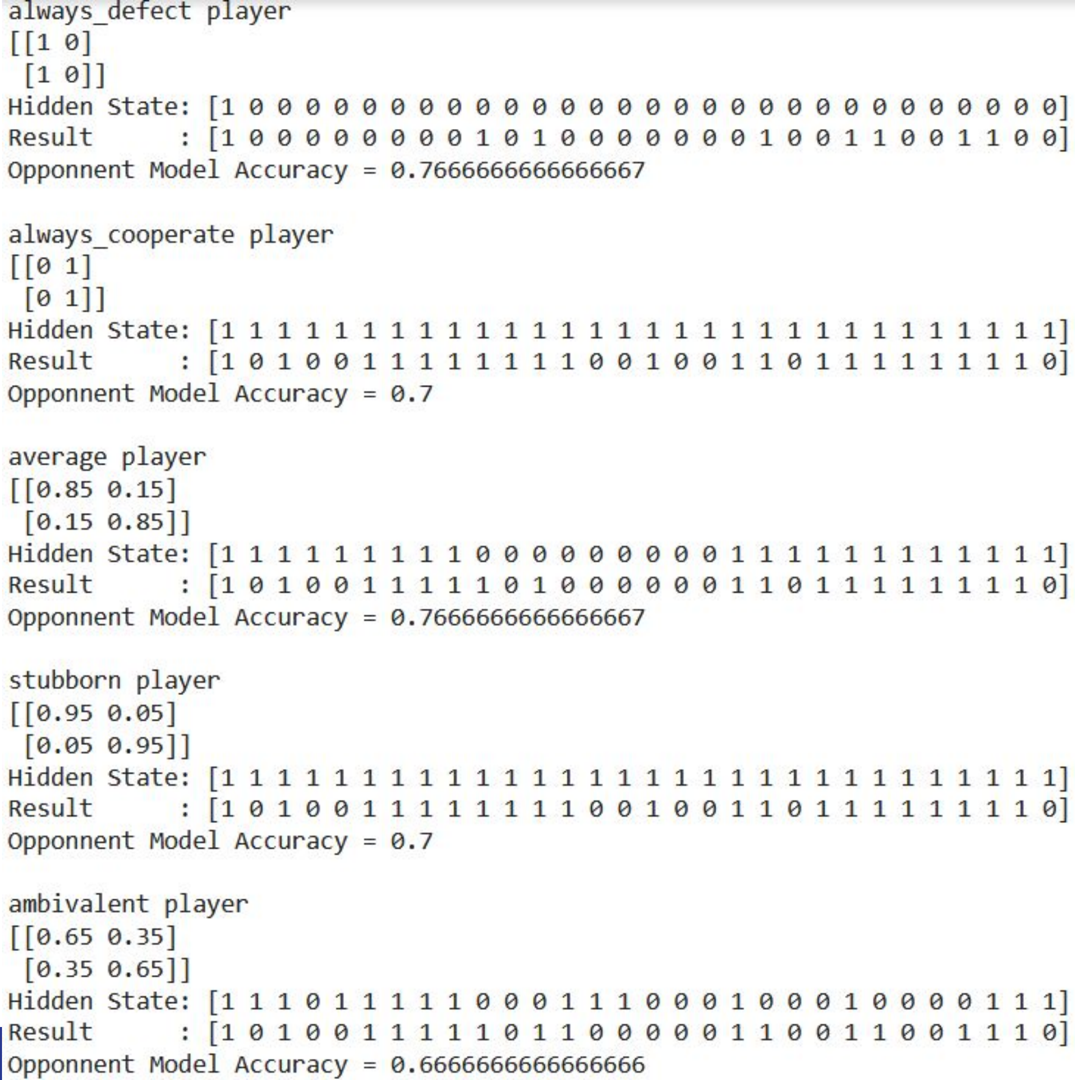
- As the number of future actions increase, the number of possibilities that need to be calculated grow exponentially

Black → previous history action Blue → future actions

Generating Examples of Each Strategy

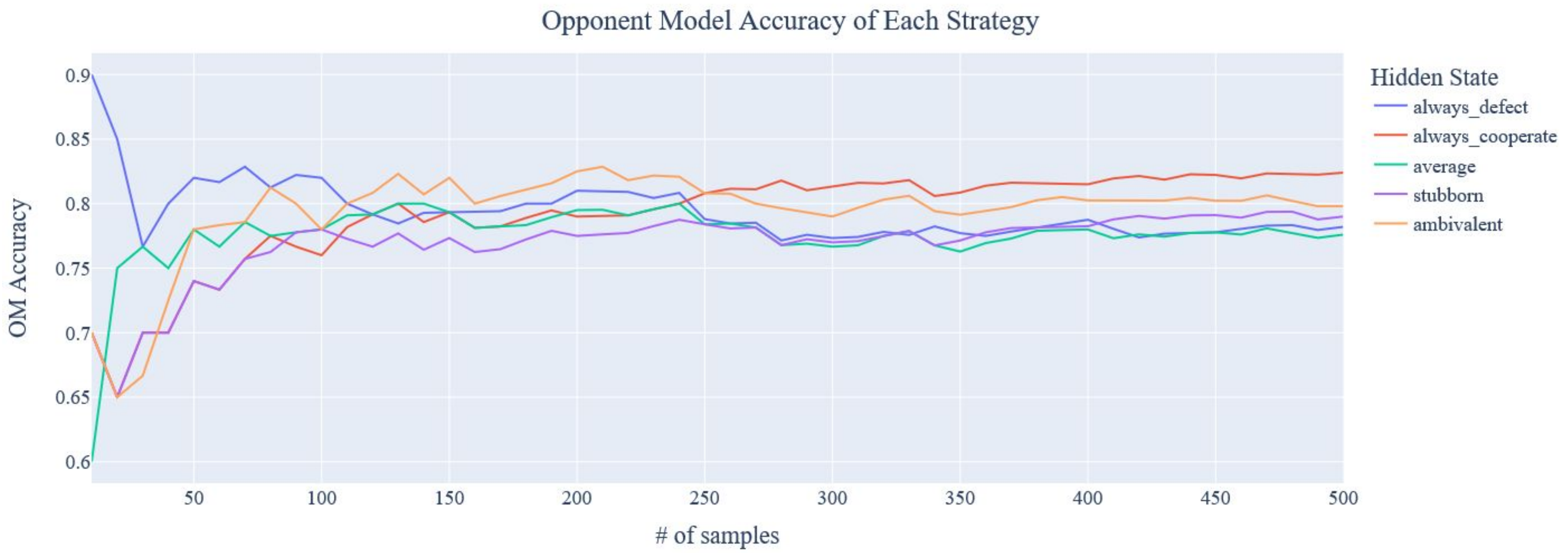
1. Top row → name of strategy
2. Underneath strategy name → transition matrix
3. Hidden State → intentions of player
4. Results → Observed Actions
5. OM Accuracy → agreement between Hidden State and Results

*Note, we set emission probability to 0.8. → players have a 80% probability of following through on their intention



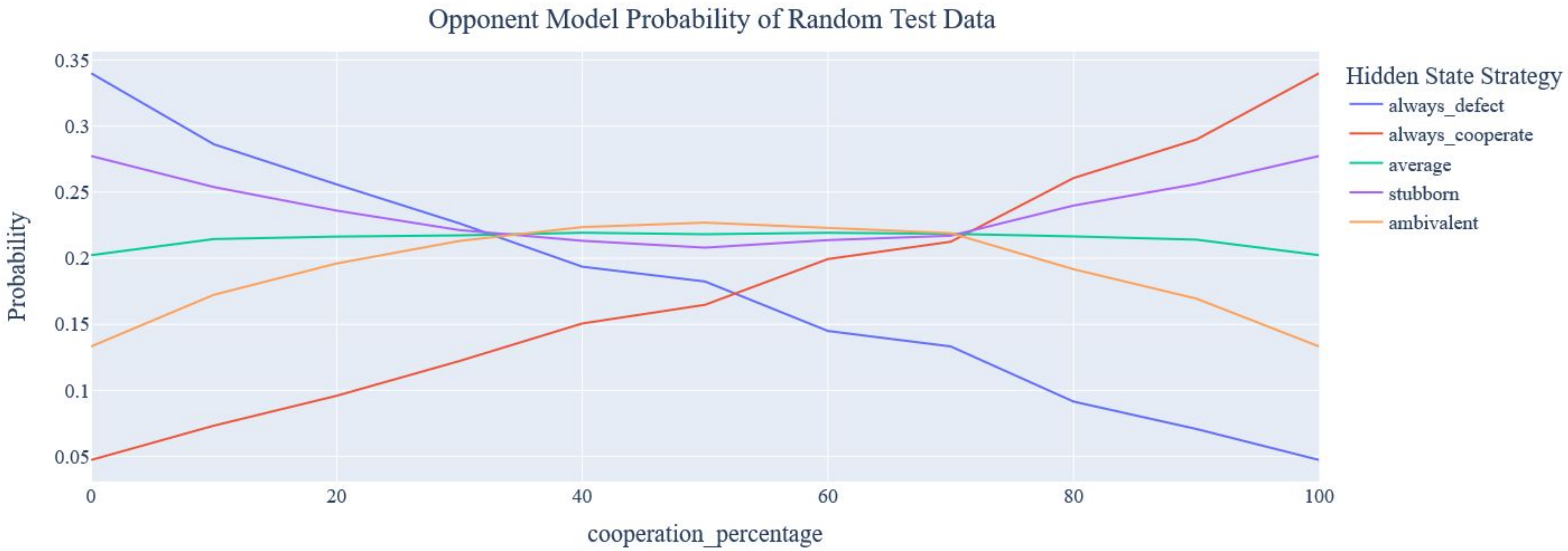
```
always_defect player
[[1 0]
 [1 0]]
Hidden State: [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
Result      : [1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 1 0 0]
Opponent Model Accuracy = 0.7666666666666667

always_cooperate player
[[0 1]
 [0 1]]
Hidden State: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
Result      : [1 0 1 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 0]
Opponent Model Accuracy = 0.7

average player
[[0.85 0.15]
 [0.15 0.85]]
Hidden State: [1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1]
Result      : [1 0 1 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 1 1 0 1 1 1 1 1 1 1 1 0]
Opponent Model Accuracy = 0.7666666666666667

stubborn player
[[0.95 0.05]
 [0.05 0.95]]
Hidden State: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
Result      : [1 0 1 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 0]
Opponent Model Accuracy = 0.7

ambivalent player
[[0.65 0.35]
 [0.35 0.65]]
Hidden State: [1 1 1 0 1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 0 0 1 0 0 0 0 1 1 1]
Result      : [1 0 1 0 0 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 0 0 1 1 0 0 1 1 1 0]
Opponent Model Accuracy = 0.6666666666666666
```

OM Accuracy of each strategy matches emission probability with asymptotic increase in number of samples



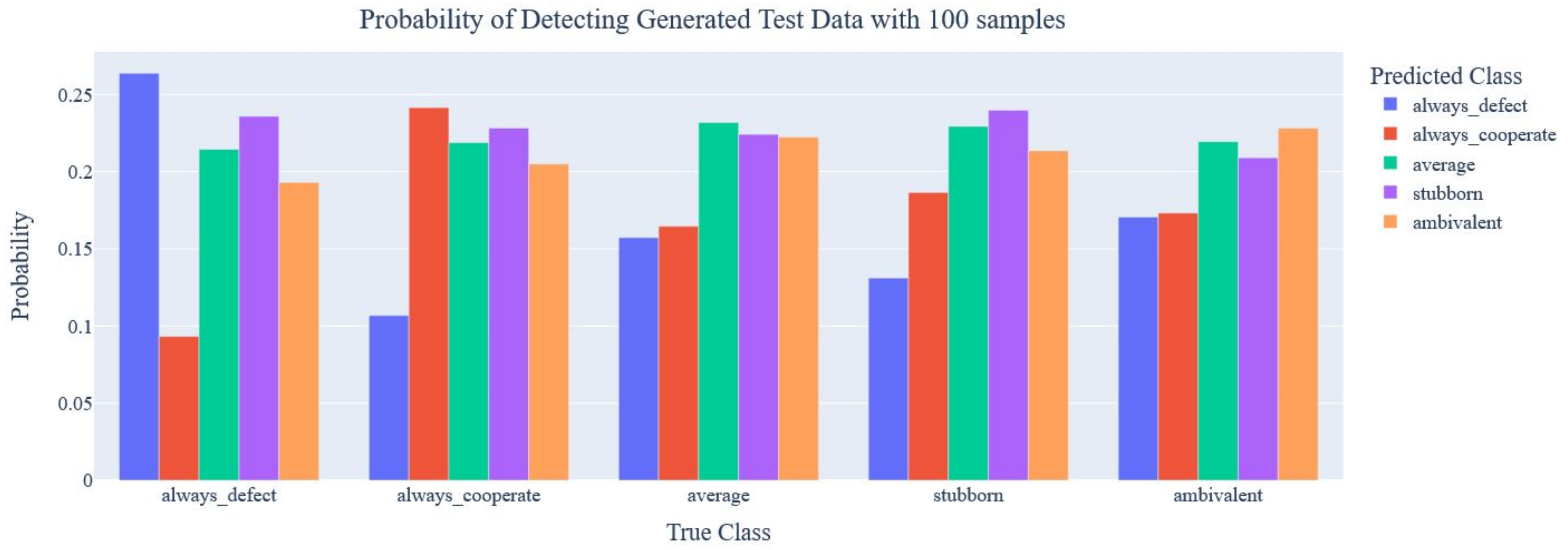
Detect Hidden Strategy on randomly generated test data

- Each strategy was run on a sample size of 100 trials from the iterated prisoner's dilemma
- Test data only differed in percentage of cooperation, sequential information not encoded
- Always_defect, always_cooperate → easiest to detect & occur at extreme disproportionate datasets
- Average, stubborn, ambivalent → more difficult to detect & occur at more balanced datasets



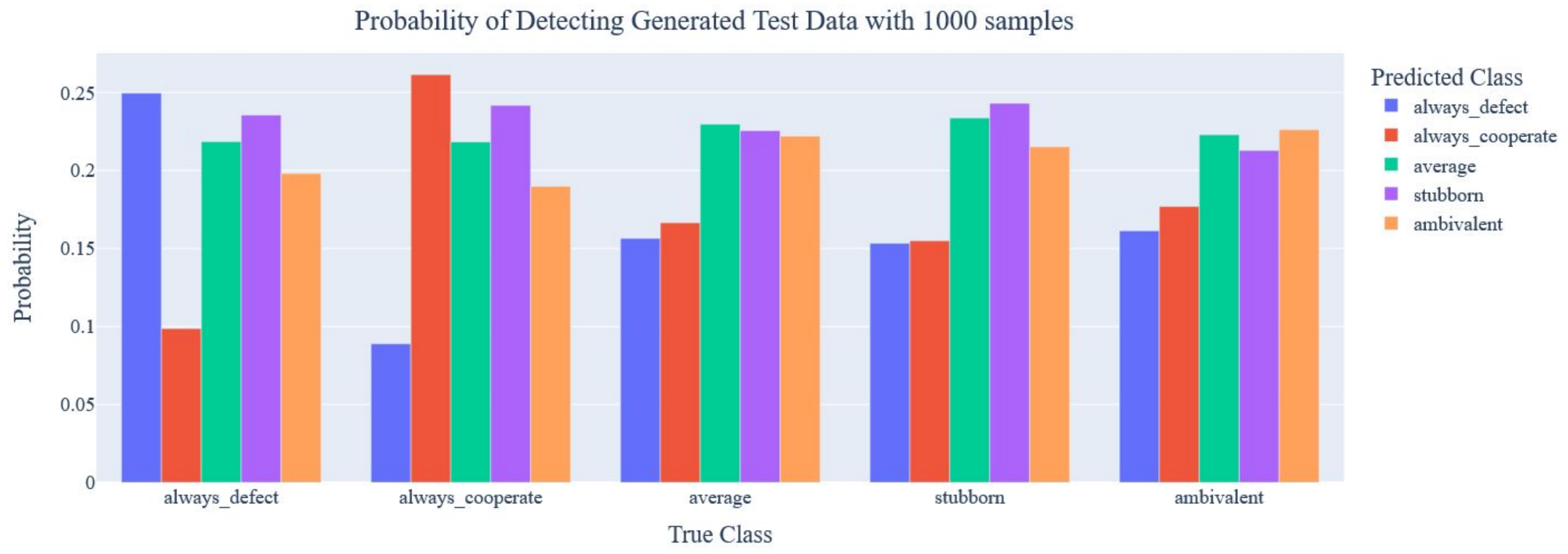
Predicting strategy generated from unknown random strategy

- We generated 100 actions from the iterated prisoner's dilemma game using a specific true class strategy
- Then we ran OM to see if our model can find out what strategy it was
- At 100 samples, we were able to successfully detect all strategies



More certainty with 1000 actions?

An exponential increase of samples is required to produce a small linear increase in accuracy.

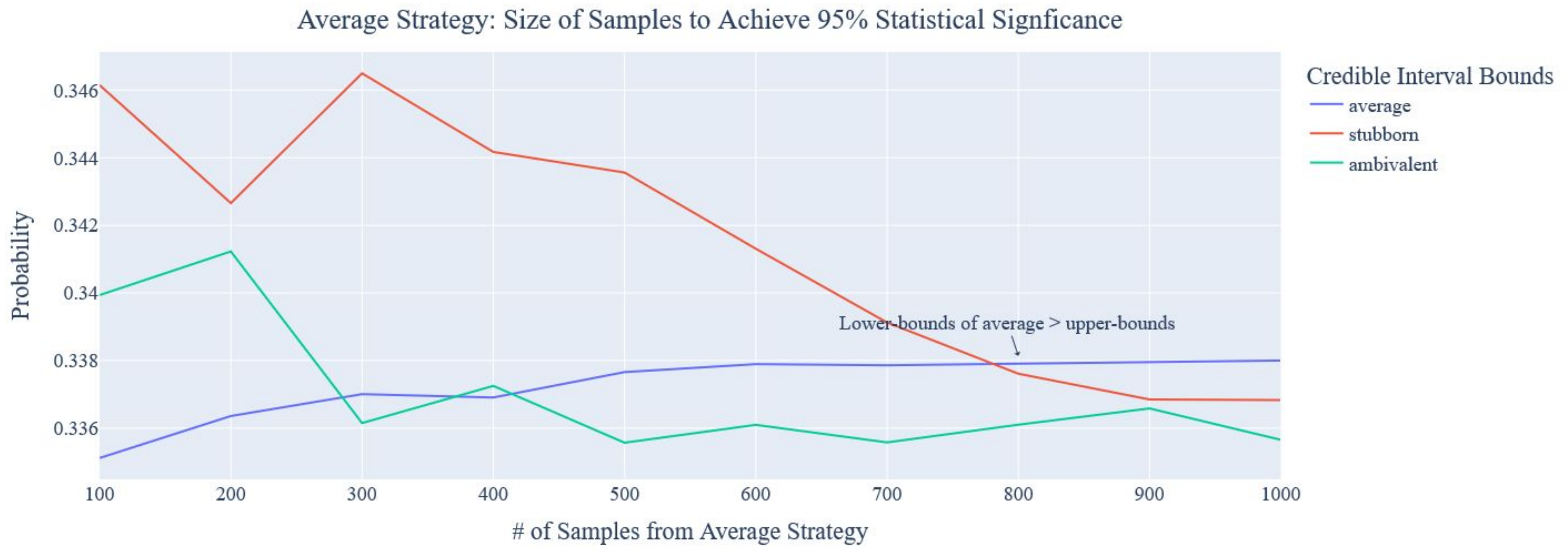


Statistical Testing

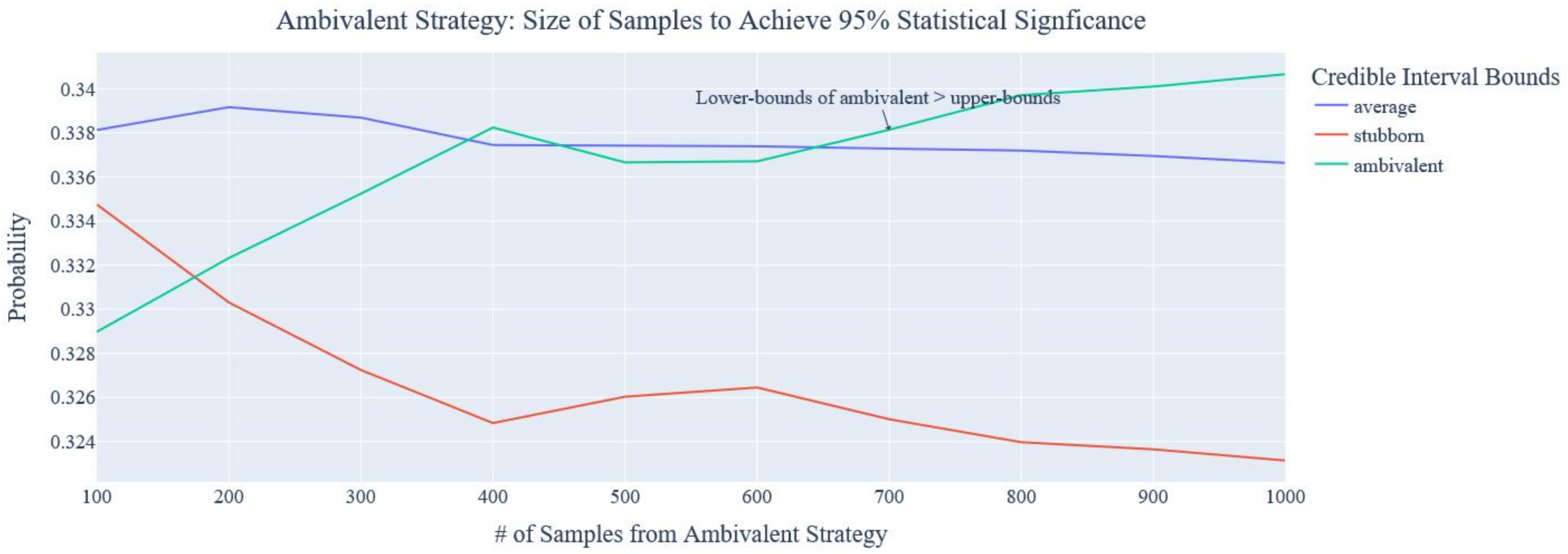
- But how reliable is our OM model?
- Could the previous results be attributed to random chance or luck?

- We calculate 95% credible interval using bootstrapping
  - We run our OM implementation on samples sizes of 100,200,300,400,500,600,700,800,900,1000
  - At each sample size, we ran our experiment 30 times with a different random seed
  - If the 5% lower bound of our strategy prediction was higher than the 95% upper bound of other strategies, then we say that 95% of the time, our model correctly predicts the true strategy

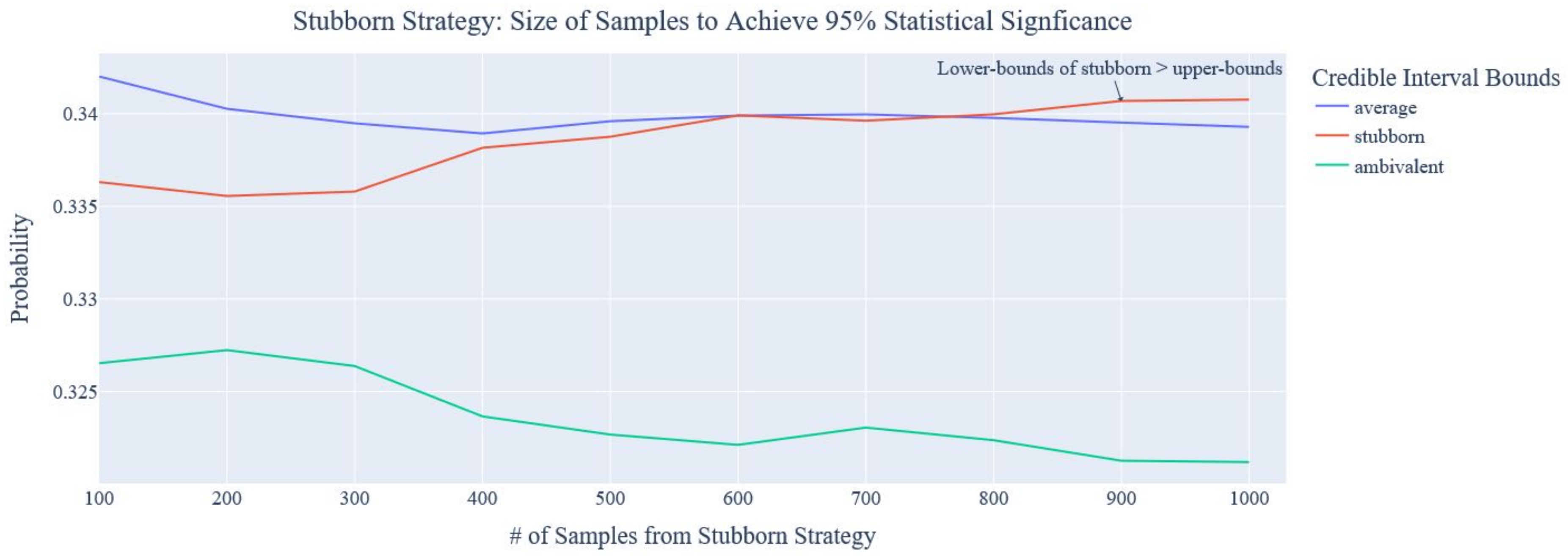
Average Strategy Detection 95% credibility requires 800 actions



Ambivalent Strategy Detection 95% credibility requires 700 actions



Stubborn Strategy Detection 95% credibility requires 900 actions



Challenges and Limitations

- Noise in data
  - Difficulty in categorizing individuals
  - Difficulty connecting individual strategy to a category
- Uncertainty & Human Error
  - opponent's actions do not always reflect their intentions(strategy)
  - Opponents are often not aware why their actions do not align with their intentions
- Large Samples Size Required
  - Our python implementation required 900 trials of the iterated prisoner's dilemma to accurately predict the opponent's strategy within a 95% credible interval
  - Can detect a strategy within as little as 50 samples, but the number of samples required exponentially to achieve a linear increase in accuracy
- Feature Interdependence
  - The strategy being investigated can be very nuanced and contextual
  - Strategy deployed depends on previous strategies and opponent strategies
  - Unraveling the sequential patterns within a strategy can be too difficult
  - May not adhere to the markov property

If you liked this presentation…

I just graduated from my MS and I am looking for a ML engineer / data science job

If you know anyone offering a job, or can offer job hunting tips, I would greatly appreciate talking to you.

My email is sazarta1@jh.edu

Python implementation code →
https://github.com/soazarta/Portfolio/blob/main/Multi-Agent%20Systems/azartash_sina.ipynb

Conclusion:

- ❖ We used Hidden Markov Models to implement an Opponent Modeling System using HMMLearn python
- ❖ Our objective was to detect the hidden strategy of an opponent and predict future actions
- ❖ Our model was functional at 100 samples
- ❖ Statistically credible at 900 samples
- ❖ Our OM model would be useful when there is a lot of samples
- ❖ A human could likely estimate the strategy of an opponent in a much smaller sample size, but may not have the patience to
- ❖ Our OM can be scaled to include more strategies and handle more complex games in which a human may struggle

By Sina Azartash
Sazarta1@jh.edu