

Fraud Detection Using Opponent Modeling

A Passion Project, by Sina Azartash
sina.azartash@gmail.com

Summary

- ❖ We implemented Opponent Modeling using Hidden Markov Models
- ❖ We compared the hidden strategy of users and detected fraudulent accounts
- ❖ Our model was functional at 100 samples and statistically credible at 900 samples
- ❖ Our OM model is useful when there is a lot of samples. If there are less samples, humans could learn much faster, but our OM model could assist the human by flagging potential unusual accounts
- ❖ We leveraged the HMMLearn python library to achieve a functional model on a variety of open source fraud detection datasets

Overview of Opponent Modeling (OM)

- ❖ Adversarial Environments
 - Professional Fraudsters will use sophisticated technology to attempt to deceive fraud detection software
- ❖ Advantages provided by OM
 - Compares strategy of normal users against fraudster strategy
- ❖ Overview of OM architectures and algorithms
 - Hidden Markov Models (A generative model)
- ❖ Example to Iterated Prisoner's Dilemma
 - Simple example using sequential data
- ❖ Explanation of algorithm fundamentals
- ❖ Results with statistical significance testing



Adversarial Environments

What → To learn the strategy of credit card users:

- ❖ Uses Prior Knowledge and/or observed actions
- ❖ Why:
 - To predict behavior
 - To exploit predictions
 - To defend against exploitation
- ❖ Fraudster and Fraud detection software become involved in a cat and mouse game

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

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

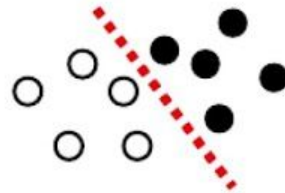
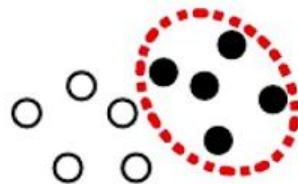
- a. Compare likelihood of actions with probability of strategy
- b. Hidden Markov Models, Bayesian Networks, Neural Networks, Expert Systems

3. Policy approximation

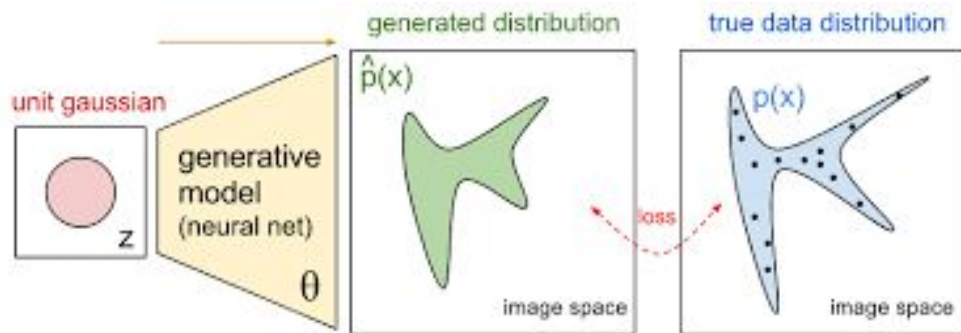
- a. Reinforcement learning: Model the problem sequential decision making in state action pairs
 - b. Calculate a policy as an approximation of the strategy
 - c. Abstract Markov Models, Deterministic Automata, Deep Neural Networks
 - d. Partially Observable Markov Decision Process
- 

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



❖ Generative Model

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

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

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 \rightarrow hidden state
 - ❑ Describes probability of switching to a different state or staying on current state
- ❑ Emission Matrix
 - ❑ Hidden state \rightarrow actions
 - ❑ Describes probability of actions aligning with hidden state

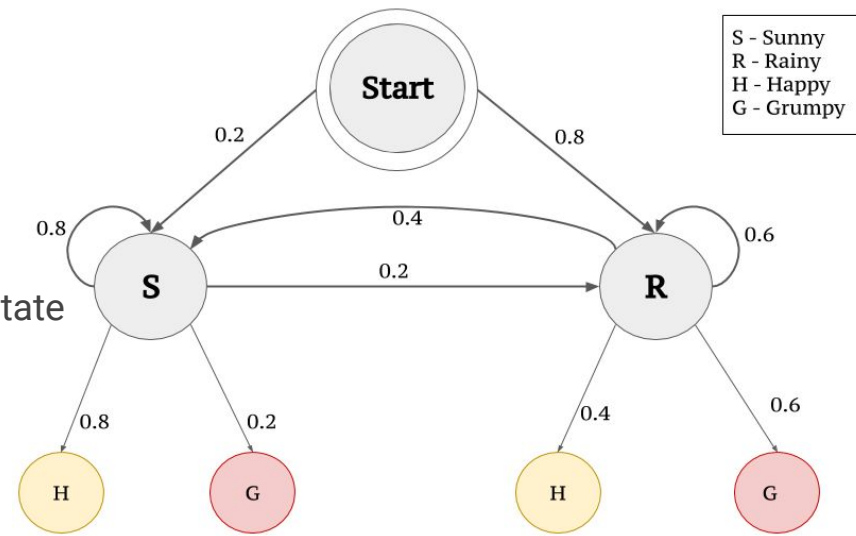
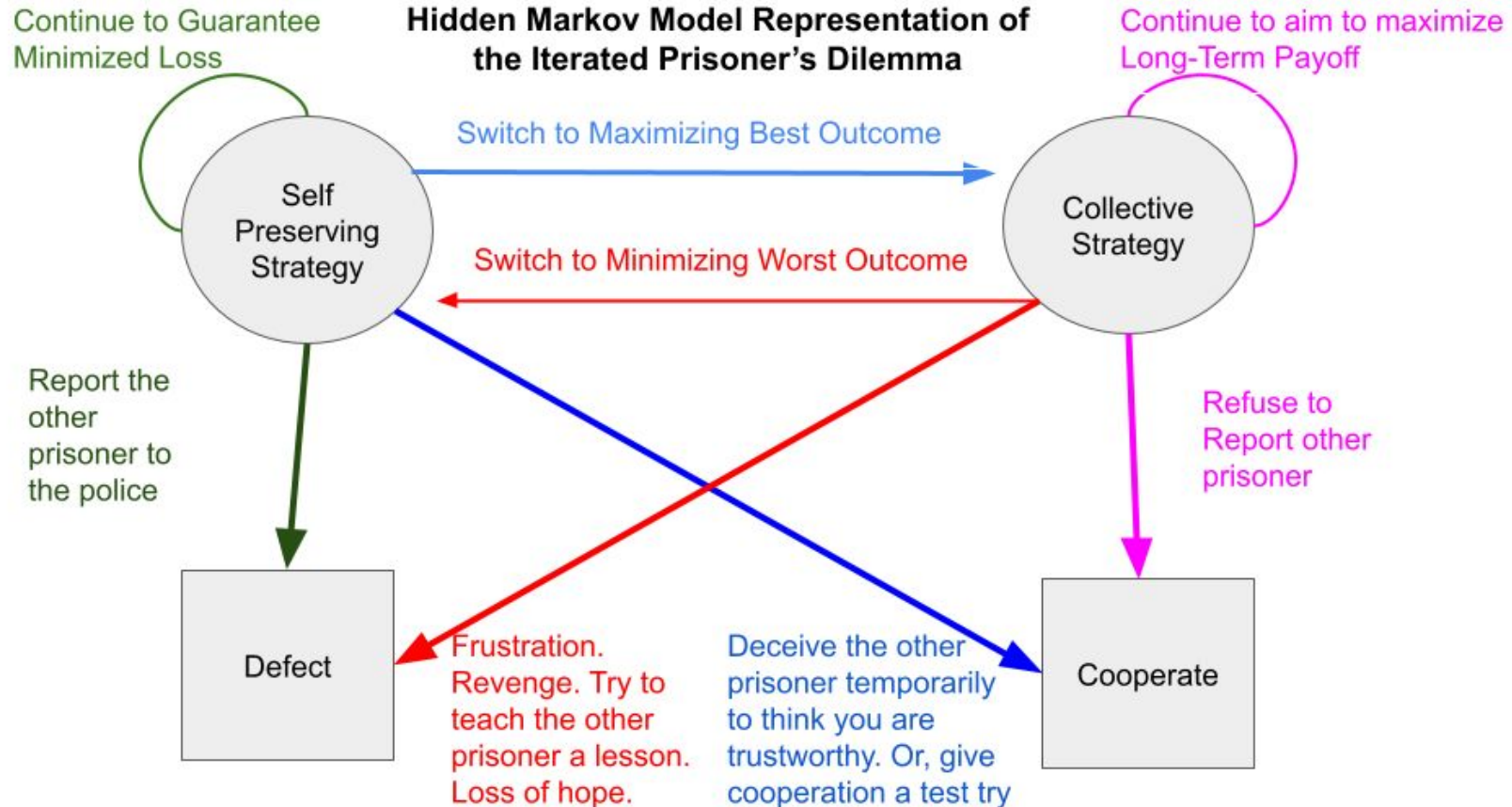


Image by [Vivek Vinushanth Christopher](#)

HMM Modeling Of the Iterated Prisoner Dilemma



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
- ❑ $P(Y_0 | X_0) * P(X_0) \rightarrow$ posterior probability of a defect action given a selfish strategy multiplied by the probability of a selfish strategy

- ❑ $P(Y_0 | X_0) * P(X_0) * P(X_1 | X_0) * P(Y_1 | X_1) * P(X_1 | X_1) * P(Y_1 | X_1)$
 - ❑ 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 n th term of a sequence can be based on the $n-1$ state (enable Markov Property)

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

Probability of a sequence of actions:

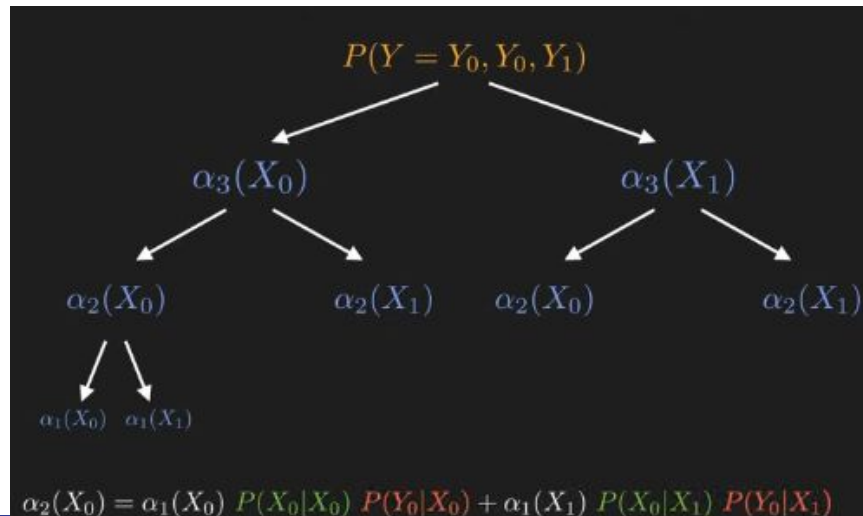
1. Yellow \rightarrow dependent on previous state
2. Green \rightarrow probability of hidden state transition
3. Red \rightarrow conditional probability of action state given hidden state

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

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

$$\begin{aligned} a_0 &= -3 \\ a_1 &= (-3) + 1 \\ a_2 &= ((-3) + 1) + 2 \\ a_3 &= (((-3) + 1) + 2) + 3 \\ &\vdots \\ a_n &= ((-3) + 1) + 2 + \dots + (n-1) + n \end{aligned}$$

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



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 \rightarrow (transition matrix) \rightarrow hidden state \rightarrow (emission matrix) \rightarrow action
- ❖ Observed Action \rightarrow inferred opponent hidden state \rightarrow 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

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

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

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

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

Black \rightarrow previous history action Blue \rightarrow future actions



Generating Examples of Each Strategy

1. Top row \rightarrow name of strategy
2. Underneath strategy name \rightarrow transition matrix
3. Hidden State \rightarrow intentions of player
4. Results \rightarrow Observed Actions
5. OM Accuracy \rightarrow 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]
```

```
Result      : [1 0 0 0 0 0 0 0 0 1 0 1 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

$$[[0 \ 1]$$
$$\begin{bmatrix} 0 & 1 \end{bmatrix}$$

Hidden State: [1 1]

```
Result      : [1 0 1 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 1 0 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 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]

```
Result      : [1 0 1 0 0 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

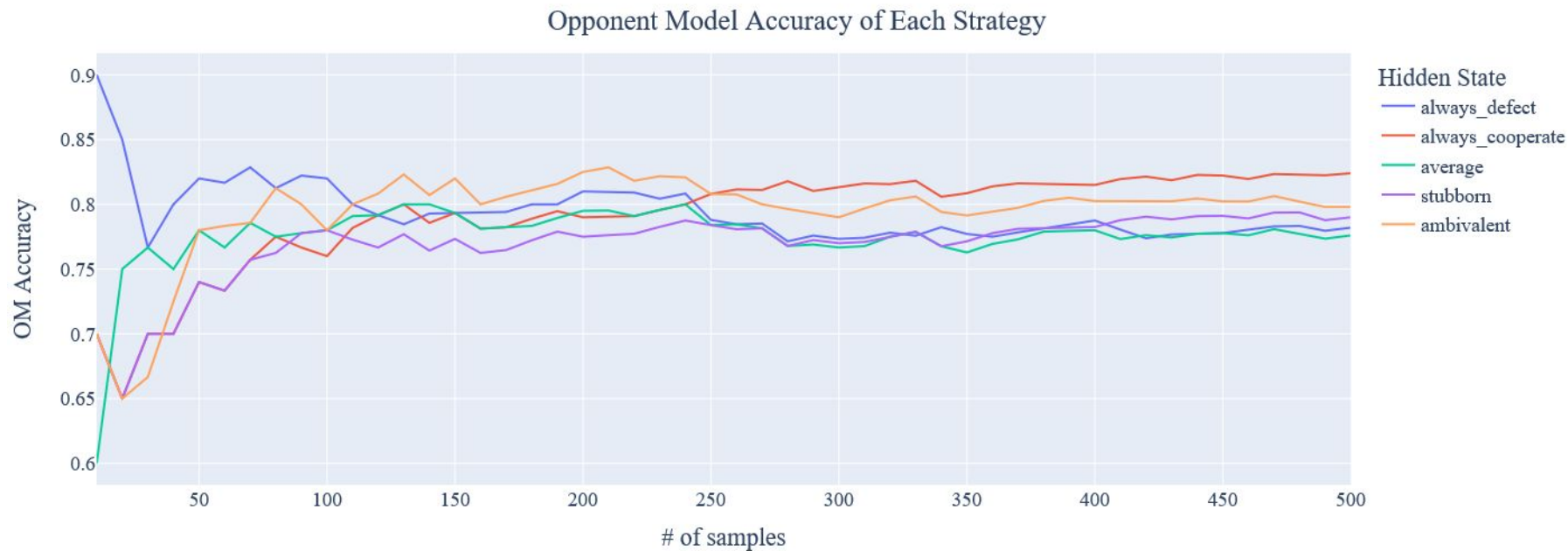
$$[[0.65 \ 0.35]]$$
$$[0.35 \ 0.65]]$$

Hidden State: [1 1 1 0 1 1 1 1 0 0 0 1 1 1 0 0 0 1 0 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

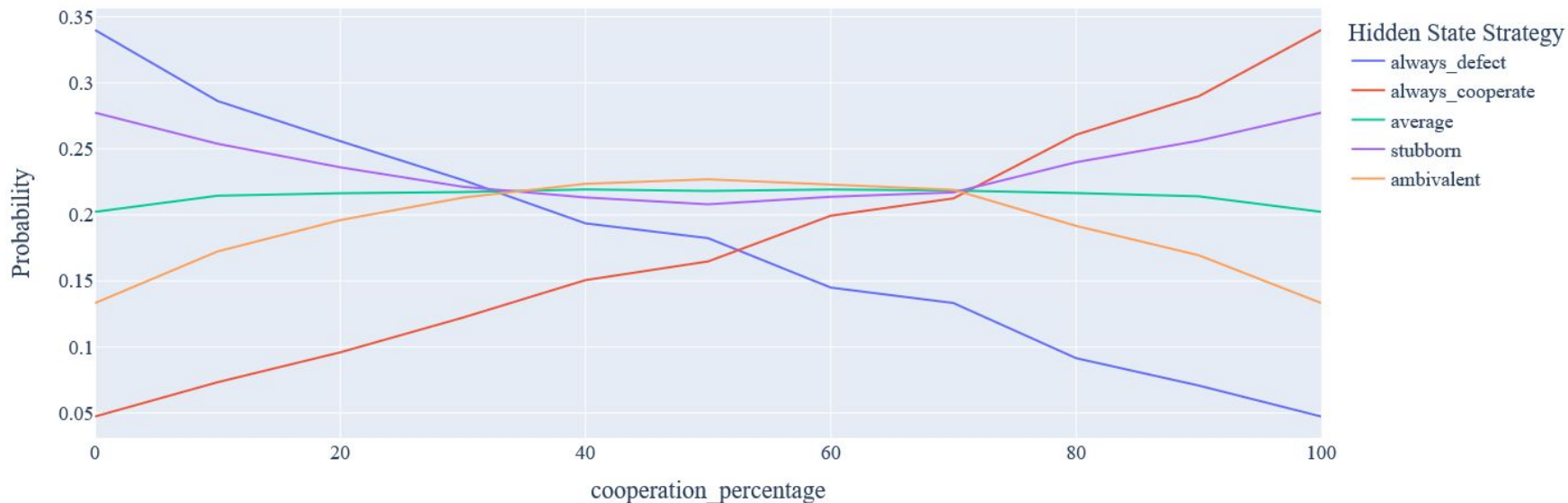
OM Accuracy of each strategy matches emission probability with asymptotic increase in number of samples



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

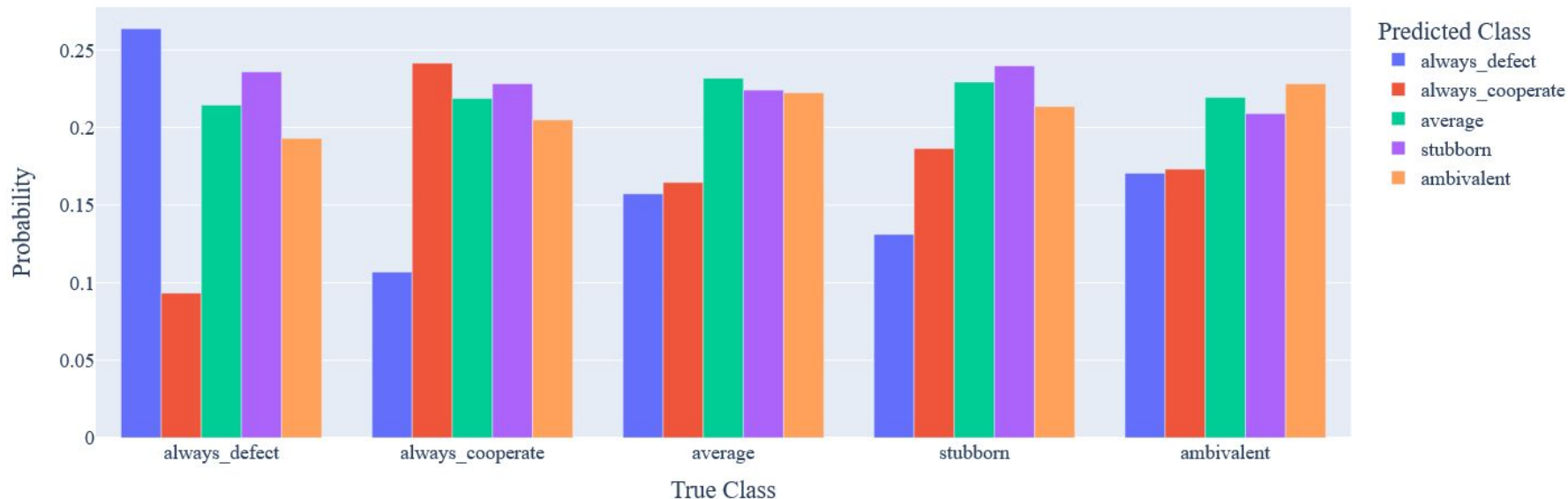
Opponent Model Probability of Random Test Data



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

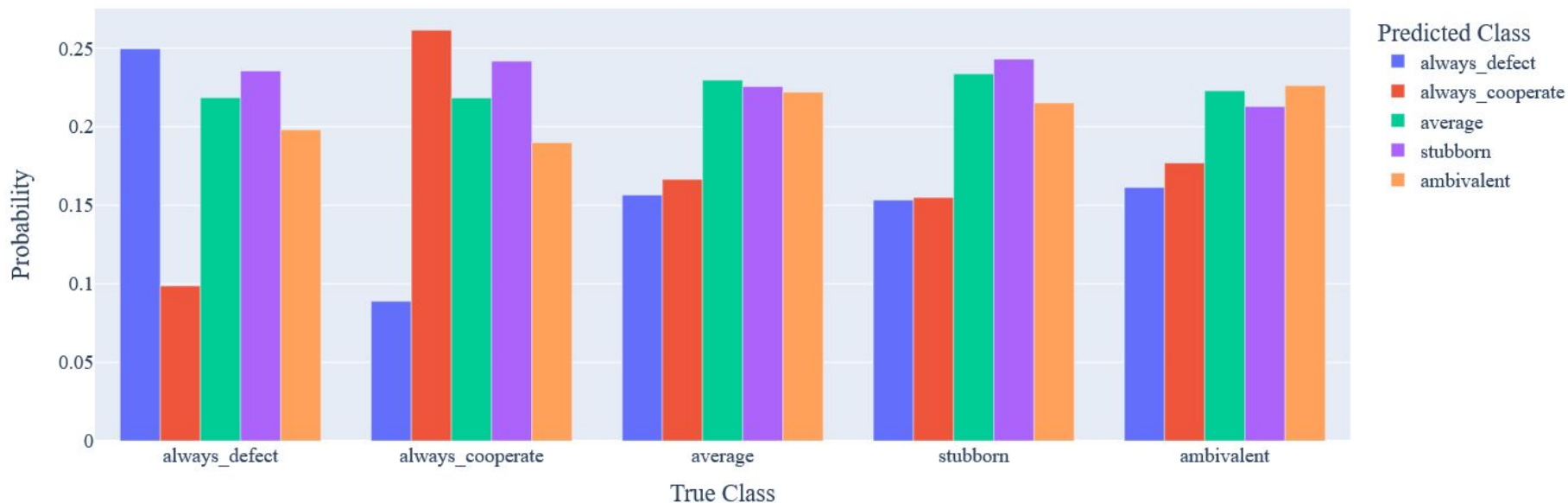
Probability of Detecting Generated Test Data with 100 samples



More certainty with 1000 actions?

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

Probability of Detecting Generated Test Data with 1000 samples



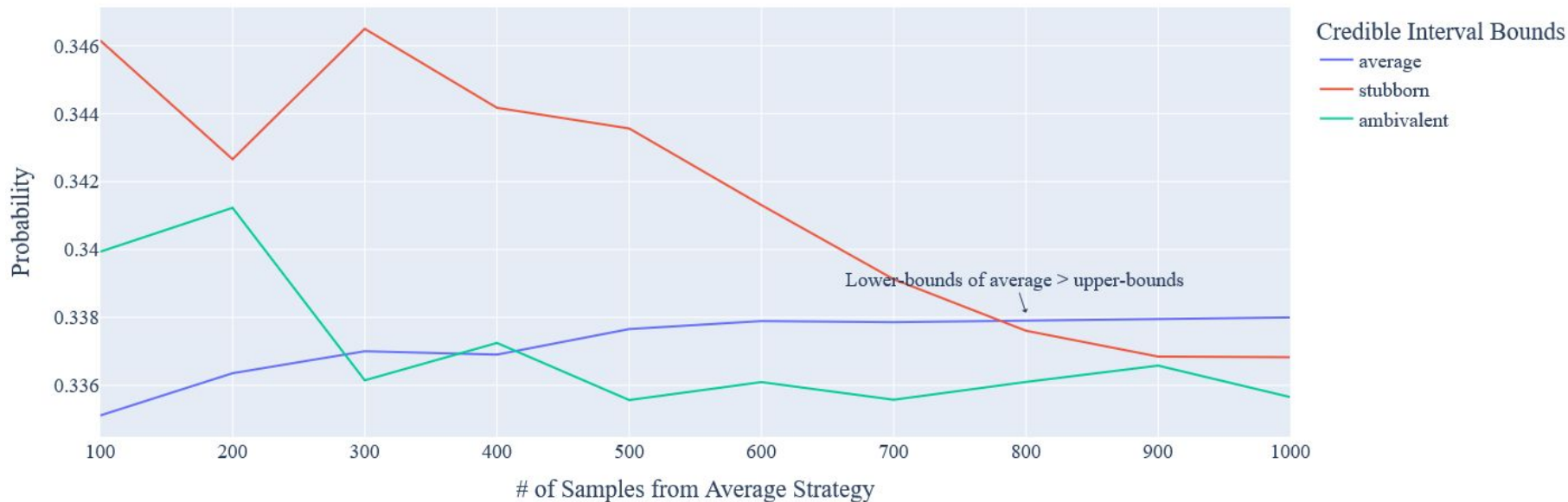
Statistical Testing

- 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



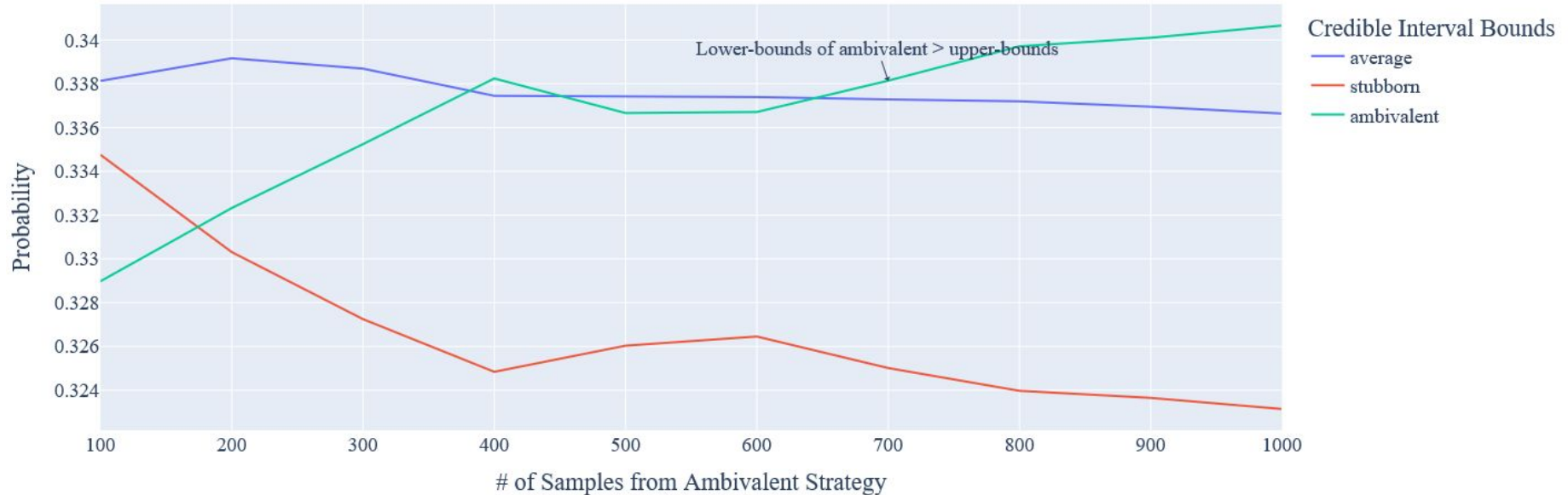
Average Strategy Detection 95% credibility requires 800 actions

Average Strategy: Size of Samples to Achieve 95% Statistical Significance



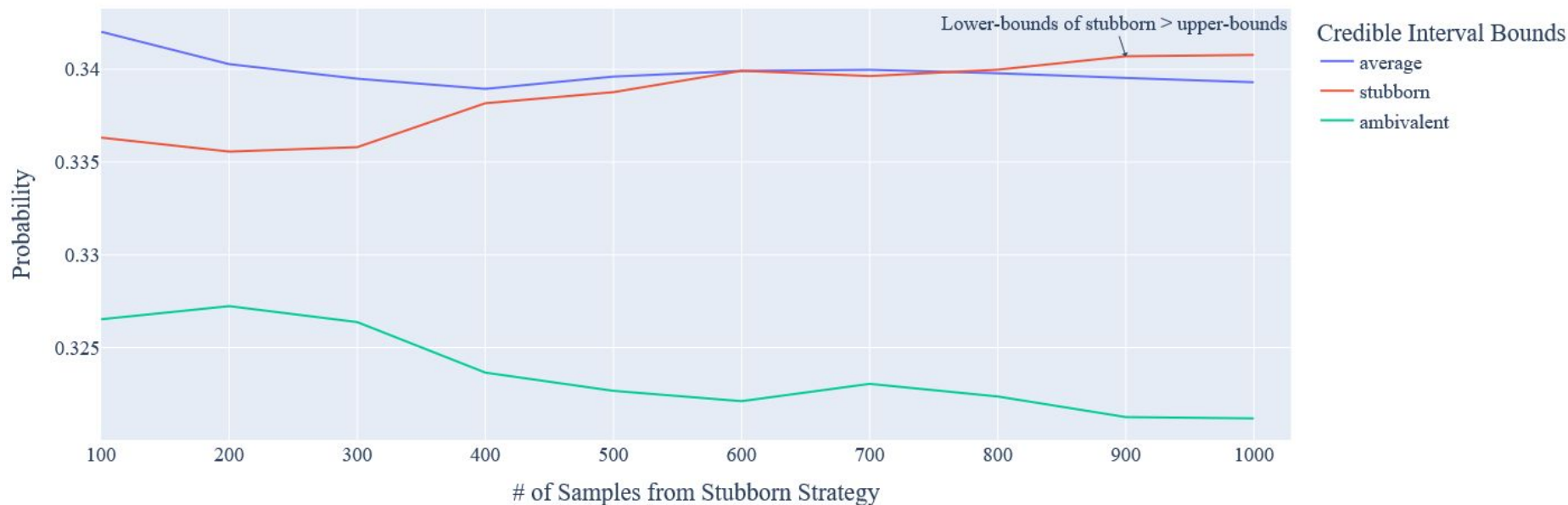
Ambivalent Strategy Detection 95% credibility requires 700 actions

Ambivalent Strategy: Size of Samples to Achieve 95% Statistical Significance



Stubborn Strategy Detection 95% credibility requires 900 actions

Stubborn Strategy: Size of Samples to Achieve 95% Statistical Significance



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
- 

Python Implementation Link

https://github.com/soazarta/Portfolio/blob/main/Multi-Agent%20Systems/azarta_sh_sina.ipynb



Work In Progress:

- ❖ Demonstrate predict future actions to understand which resources to block and where to add more security
- ❖ Generate deceptive examples to fool our fraud detection software
- ❖ Teach fraud detection software to learn from deceptive examples and teach fraudster to learn to generate more intelligent examples of fraud
- ❖ Further refine the model by:
 - More hyper-parameterization
 - Algorithm options, different parameters, and regularizers
 - Data preprocessing
- ❖ Discuss advantages and disadvantages of additional tuned ML algorithms
- ❖ Use more realistic data and include more uncertainty to represent real life situations

By Sina Azartash

Sazarta1@alumni.jh.edu

sina.azartash@gmail.com