Robust Multiagent AI for Contested Environments

Hoda Bidkhori, University of Pittsburgh
with Vahid Tarokh, Duke University
DOD applications

Many Air Force problems, such as protecting communications networks against adversaries can be formulated as dynamic games between adversaries and defenders.

- Defender wants to protect certain assets and areas with limited resources.
- The adversary wants to reduce (increase) the utility of the defender as much as possible.
Robust Multiagent AI

- Dynamically allocating resources for defending assets.
- Combating attacks and misinformation propagated by adversaries.
Problem Formulation

One time period, perfect information

\[ \min_{s \in S} \max_{a \in A} f(s, a) \]

\[ f(s, a) = u_0(s) + u_1(a, s), \quad u_0(s) \text{ is utility of defender and } u_1(a, s) \]

is how much adversary can change it by action \( a \).

- Defender has set of known strategies \( S \). There is a set of \( m \) targets, defender can only protect \( k \) of these targets.
- Attacker observes \( s \) and decides for an action \( a \). Action \( a \) could be disrupting at most \( k' \) assets/facilities.
- The objective is \( f(s, a) \). Defender wants minimize it with strategy \( s \). The attacker wants to maximize it with an action \( a \).
Simple Example (zero sum game)

Defender: deploy resources to protect targets.
Attacker: which target to attack.

- Defender receives utility $w_i$ if she protects the asset $i$.
- If attacker attacks the protected asset $i$, the overall utility increases by $k_i$. If she attacks the unprotected asset, she increases the overall utility by $\hat{k}_i$, where $k_i \leq \hat{k}_i$.
Randomized version

Both defender and attacker randomize their strategies and actions according to the distributions $p_s$ and $p_a$. $p_s$ is a probability distribution over possible $s \in S$ and $p_a$ is a probability distribution over possible $a \in A$.

$$\min_{p_s} \max_{p_a} \mathbb{E}_{p_s,p_a}[f(s,a)]$$

- The defender wants to choose a distribution $p_s$ over set of strategies to minimize the expected utility.
- The attacker wants to choose the distribution $p_a$ over set of actions to maximize the expected utility.
More general model

Attacker and defender in real-time.

At time $t$

$$\min_{s_t \in S^t} \max_{a_t \in A^t} f^t(s_t, a_t) \text{ where } f^t(s_t, a_t) = u^t_0(s_t) + u^t_1(a_t, s_t)$$

- The set of strategies is $S^t$. This set changes over time and could depend on various factors including the previous strategies and actions and the context of the problem.
- The set of actions is $A^t$. This set changes over time and could depend on various factors including the previous strategies and actions and the context of the problem.
- The function $f^t(s_t, a_t)$ also depends on the previous actions and strategies.
Challenges and variations

\[
\min_{s_t \in S^t} \max_{a_t \in A^t} f^t(s_t, a_t)
\]

- Defender knows \(A^t\) and the functions \(f^t(s_t, a_t)\), \(u^t_0(s_t)\) and \(u^t_1(a_t, s_t)\). Attacker knows defender utility and the set of strategies \(S^t\).
- Defender partially knows \(A^t\) and the function \(f^t\). Attacker partially knows defender utility and strategies \(S^t\).
- Presence of noise: Knowledge influenced by the noise. (Noise can come from system or adversary)
- The attacker and defender knowledge comes from incomplete data sets. (Some of the data are not observable: missing data)
- The environment and data can be highly non-stationary. The decisions need to be made in real-time.
- The objective \(f^t(s_t, a_t)\) has a complex form.
Uncertain knowledge

\[
\min_{s \in S} \max_{p_{a,s} \in P_{a,s}} \mathbb{E}_{p_{a,s}} [f(s, a)]
\]

\[
\min_{p_s} \max_{p_{a,s} \in P_{a,s}} \mathbb{E}_{p_s} [\mathbb{E}_{p_{a,s}} [f(s, a)]]
\]

- Defender chooses the strategy \( s \). (According to the distribution \( p_s \)). Attacker observes \( s \) and knows the function \( f(s, a) \).
- Attacker chooses the action \( a \) for observation \( s \) according to the distribution \( p_{a,s} \).
- Defender does not know the exact distribution \( p_{a,s} \) but knows that \( p_{a,s} \in P_{a,s} \).
- Defender chooses the best strategy with respect to the worst-case distribution.
Defender chooses the strategy $s$. (According to the distribution $p_s$). Attacker partially observes some dimensions of $s$ denoted by $s_{obs}$, and knows the function $f(s, a)$.

Attacker can identify $S_{obs}$ which is the set of $s$ compatible with $s_{obs}$. Attacker makes an inference about the possibilities of distributions of $s$: $p_{s_{obs}}$.

Attacker could infer the uncertainty set of possible distributions: $p_{s_{obs}} \in P_{s_{obs}}$. In that case, defender needs to consider the worst case.

$$\min_{s \in S} \max_a \max_{p_{s_{obs}} \in P_{s_{obs}}} \mathbb{E}_{p_{a,s}}[f(s, a)]$$
## Challenges in real-world setting

### Solving the min max optimization (equivalently max min optimization)

- Full information is not available. We observe real-time data.
- Actions and strategies are not fully known or observable.
- Strategies and actions are changing in real-time. Need to update the decisions based on limited observations.
- Need to make decisions to take care of both worst case and average case performance.

### Proposed work

- Data-driven robust optimization framework for missing data.
- Learning and optimization framework to handle real-time non-stationary data.
- Robust algorithms that perform well in most scenarios.
## Challenges in complex environment

### More complex scenarios
- The state space and the action space are unknown.
- The utility functions are unknown.

### Proposed work
- Reinforcement learning frameworks to learn the utilities from real-time data.
- Deep reinforcement learning methods to learn the state space and action space.
- GAN framework to learn the distributions.
Research plan

**RO1: Data-driven decision-making frameworks to handle missing data**

Data-driven robust/stochastic optimization frameworks for missing data.

**RO2: Online data-driven decision-making models for real-time non-stationary data**

Learning and optimization framework to handle real-time non-stationary data.

**RO3: Reinforcement learning and neural networks for more complex settings**

Reinforcement learning frameworks to learn the utilities from real-time data (Missing and non-stationary data). Deep reinforcement learning methods to learn the state space and action space.
RO1: data-driven decision-making facing missing data

- Missing data in some dimensions.
- The sharing of data is limited among dimensions.
- Different sizes of data in different dimensions.

- Integrated frameworks for data-driven decision making facing incomplete joint data. K. Ren* and H. Bidkhor
Data assumptions

The complete data points are $I$ dimensional and denoted by $s$. $s \in S$ and $S$ is finite. $s_{obs}$ denotes a partially observe data.

$$S(s_{obs}) = \{ s \in S | s \circ \phi_{obs} = s_{obs} \}$$

Independence Assumption

The missing probability of a missed value is independent of the value itself.

$$P(s_{obs} | s) = P(s_{obs} | s'), \quad s, s' \in S(s_{obs}) \quad (1)$$
Stochastic programming with missing data

Our decision-making problems can formulated as stochastic programs:

\[
\min_{x \in \mathcal{X}} \mathbb{E}_s[Q(x, s)]
\]

Properties of the traditional model

- \(Q(x, s)\): represents a cost function with respect to decision \(x\), and \(\mathcal{X}\) is the feasible region of \(x\)
- Estimate the data distribution via data imputation and solve the optimization subsequently.
- Does not lead to a good performance.
Proposed method: DRO framework with missing data

\[
\min_{x \in \mathcal{X}} \max_{P \in \mathcal{P}} \sum_{s \in S} P(s) Q(x, s).
\]

The properties of the model

- Choices of the ambiguity set \( \mathcal{P} \).
- Theoretical performance: statistical consistency
- Computational tractability.
- Experimental results and evidence.
Statistical consistency

Our model is as good as having complete information when the available data size $N$ is large enough.

**Theorem: Consistency**

Suppose $N \to \infty$ and $\lim_{N \to \infty} \tau(N) = 0$. The objective value for $\min_{x \in \mathcal{X}} \max_{P' \in \mathcal{P}'} \mathbb{E}[Q(x, s)]$, $\hat{O}_N$ under the incomplete data set of size $N$ converges the objective value of $\min_{x \in \mathcal{X}} \mathbb{E}_{P^*}[Q(x, s)]$ under the true unknown joint distribution. $\hat{O}_N \xrightarrow{p} O^*$. If the maximizer $x^*$ is unique, then $\hat{x}_N \xrightarrow{p} x^*$.

**Idea of the proof**

As $N$ goes to infinity, the center of the ambiguity set $\hat{P}$ converges to $P^*$.
Out of sample performance and tractability

The optimal value of DRO model serves as a probabilistic upper bound for the out-of-sample performance.

**Theorem: Finite sample guarantee**

Suppose $P^*$ represents the true joint distribution, and $\hat{x}_N$ and $\hat{O}_N$ are one optimal solution and the objective value of DRO Model with $N$ data. Then $P\left[\mathbb{E}_{P^*}[Q(\hat{x}_N, s)] \leq \hat{O}_N \right] \geq \alpha$, for a determined $\alpha$.

**Idea of the proof**

The deviation between the true distribution and the nominal distribution can be approximated by a normal distribution asymptotically.

**Computational tractability**

We have shown tractable reformulations for the uncertainty sets with the L1 norm, L2 norm, Wasserstein distance and f-divergence.
Computational experiments: Portfolio optimization

\[ \min_{x \in \mathcal{X}} \mathbb{E}[-\xi, x] + \rho \text{CVaR}_\alpha(-\xi, x) \]

- 57 pairs of the training set and test set.
RO2: Learning and optimization framework to handle real-time non-stationary data.

The data in practice are often unknown and non-stationary. Need to make decisions sequentially based on the observed data.

- Real-time decisions based on the data that revealed in a dynamic sequential fashion.
- We need to update the decision based on limited data. A reliable forecast is not available. The former works mainly rely on simulations.
Main model and contributions

• The data $d_1, \cdots, d_n, \cdots$ are sequentially arriving everyday. At day $n$, the decision-maker needs to solve the optimization problem:

$$\min_{x_1^n, \cdots, x_s^n} C(x_1^n, \cdots, x_s^n, d_1, \cdots, d_n)$$

• Data season changes over time. The current distribution is unknown and time-varying, and the data is revealed over time.
• Assume the function $C$ and the decisions depend on the current data distribution.
• We propose four approaches and provide comparisons.

Choosing the right data-driven inventory policy: what to learn from the data? With K. Ren* and Z.J.M. Shen. 2020. Other ongoing works.
Choose the policy that achieves the lowest expected costs.

**Algorithm**

- Step 1: Estimating the possibilities of the data distributions.
- Step 2: Evaluating the costs of different policies under different data distributions.
- Step 3: Evaluating the expected future costs of different policies based on the results of Steps 1 and 2.
- Step 4: Choose the best policy.
A separated analysis: Lasso based approaches (parametric and nonparametric)

A separated analysis framework

First, determine the distributions and then derive the policies.
- Step 1: Determine the data distributions from the observed data.
- Step 2: Derive the decision policies based on Step 1.

Lasso framework to identify the current pattern

\[
\min_{\lambda_1, \ldots, \lambda_n} \frac{1}{2} \sum_{t=1}^{n} (d_t - \lambda_t)^2 + \beta \sum_{t=s+1}^{n} |\lambda_t - \lambda_{t-1}|
\]

A trade-off between the noise and the number of patterns.

Solving the DRO model based on the data of the current season
## Benchmark and Performance

<table>
<thead>
<tr>
<th>I-PA</th>
<th>S-NPA</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.06</td>
<td>1.06</td>
<td>1</td>
</tr>
<tr>
<td>1.29</td>
<td>1.16</td>
<td>1</td>
</tr>
<tr>
<td>1.04</td>
<td>1.06</td>
<td>1</td>
</tr>
<tr>
<td>1.29</td>
<td>1.17</td>
<td>1</td>
</tr>
<tr>
<td>1.04</td>
<td>1.06</td>
<td>1</td>
</tr>
<tr>
<td>1.16</td>
<td>1.14</td>
<td>1</td>
</tr>
</tbody>
</table>

### High bias/variance
- I-PA
- I-NPA

### Low bias/variance
- S-PA
- S-NPA

![Graph showing daily demand over days](image-url)
RO3: Reinforcement learning and neural networks for more complex settings

- Reinforcement learning (RL): How intelligent agents learn and take actions in an environment in order to maximize the notion of cumulative reward.
Q-learning

- At each step $s$, choose the action $a$ which maximizes the function $Q(s, a)$. $Q$ is the estimated utility function.
- $Q(s, a)$ is an immediate reward for making an action and the utility $Q$ for the resulting state.

### Q-learning

$$Q(s, a) = r(s, a) + \gamma \max_{a'}(Q(s', a'))$$

- $r(s, a) =$ Immediate reward.
- $\gamma$ is relative value of delayed and immediate reward.
- $s'$ the new state after action $a'$
- $a, a'$ are the actions in states $s, s'$
Deep Reinforcement Learning

- Neural networks are function approximators. They can be used for reinforcement learning when the state space or action space are too large to be completely known. They can be used to approximate a value function, or a policy function.
- In reinforcement learning, convolutional networks can be used to recognize an agent’s state when the input is visual.
Training NN when we have missing data and non-stationary environment

We will use and combine the ideas we have developed previously for training neural networks in complex environments.

Prior work: missing data.

Missing data in non-stationary environment

We will investigate trading from missing data to missing data in non-stationary environment and we will use LSTM and RNNs.