Inverse Reinforcement Learning from Non-Stationary Learning Agents

Kavinayan Sivakumar, Yi Shen, and Michael M. Zavlanos, Duke University

Zachary Bell, AFRL

Duke
UNIVERSITY

Outline

Problem Definition

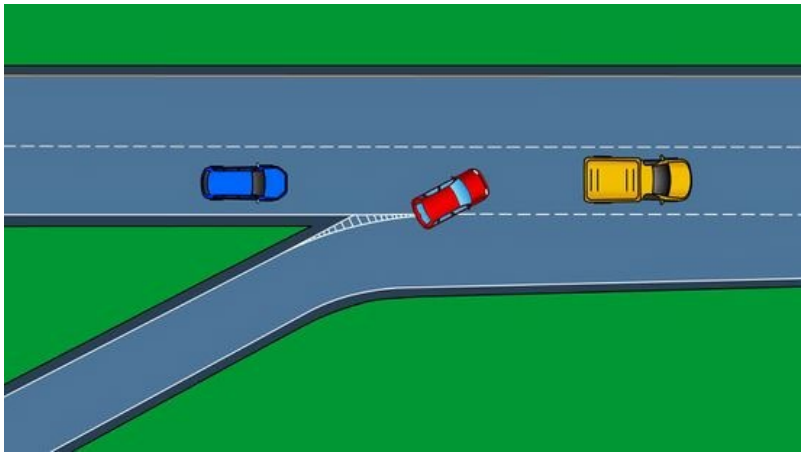
Bundle Behavior Cloning

Complexity Analysis

Empirical Results

Duke
UNIVERSITY

Understanding an Agent's Intentions



Duke
UNIVERSITY

Learner Agent and Observer Agent

The learner agent:

- Objective is to maximize its accumulated reward

$$\max_{\theta} J(\theta) = \mathop{\mathbb{E}}_{s_0 \sim \rho^u}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1})]$$

- Learns a policy

$$\theta_{t+1} = \theta_t + \alpha\gamma^t R(s_{t+1}) \nabla \ln \pi(a_t | s_t, \theta)$$

The observer agent:

- Has access to the trajectory data of the learner agent as it learns,
- Does not know the reward function of the learning agent

Problem formulation:

Learn the reward function of a learning agent using only trajectories of state action pairs generated as the agent interacts with its environment to learn its optimal policy that solves the objective function

Assumptions:

- 1: Learning agent updates its policy using the stochastic gradient descent (SGD) update,
- 2: Learning rate of the learning agent α is known and is sufficiently small
- 3: All states and actions of the learning agent in the environment can be observed

Duke UNIVERSITY

Why is this problem difficult?

We don't know the policy weights of the learning agent as it learns:

- If policy weights are known during learning, distribution of agent actions at any given state can also be estimated
- However, this is not ideal in practice as learning agent must communicate this after every update

Trajectories generated by learning agent are not generated from a stationary expert policy:

- A common assumption in IRL literature is the expert policy is fixed
- This assumption does not hold when trajectory data are collected as the agent interacts with its environment to learn an optimal policy

Related Literature

“Inverse reinforcement learning from a gradient-based learner” – Ramponi, Drappo, Restelli

- Learn an agent’s reward function while learner policy is being updated with assumption of SGD updates
- Requires that reward function of the learner agent is approximated by a set of linear feature functions

Limitations

- In practice, selecting an expressive enough set of feature functions is challenging
- In the case of radial basis functions that are commonly used as feature functions, choosing the proper centers and bandwidth parameters is not straightforward
- If a proper feature space is not chosen correctly, it is difficult to ensure each state action pair has a unique feature and that no state action pairs are favored over others
- Moreover, the learned reward function may be skewed towards the state action pairs that are favored more

Outline

Problem Definition

Bundle Behavior Cloning

- **Why standard behavior cloning does not work here**
- **Idea behind Bundle Behavior Cloning**
- **Full algorithm description**

Complexity Analysis

Empirical Results

Duke
UNIVERSITY

Standard Behavior Cloning

What is behavior cloning?

- Learn a policy through expert demonstrations
- Can try to match the distribution of actions taken at each state

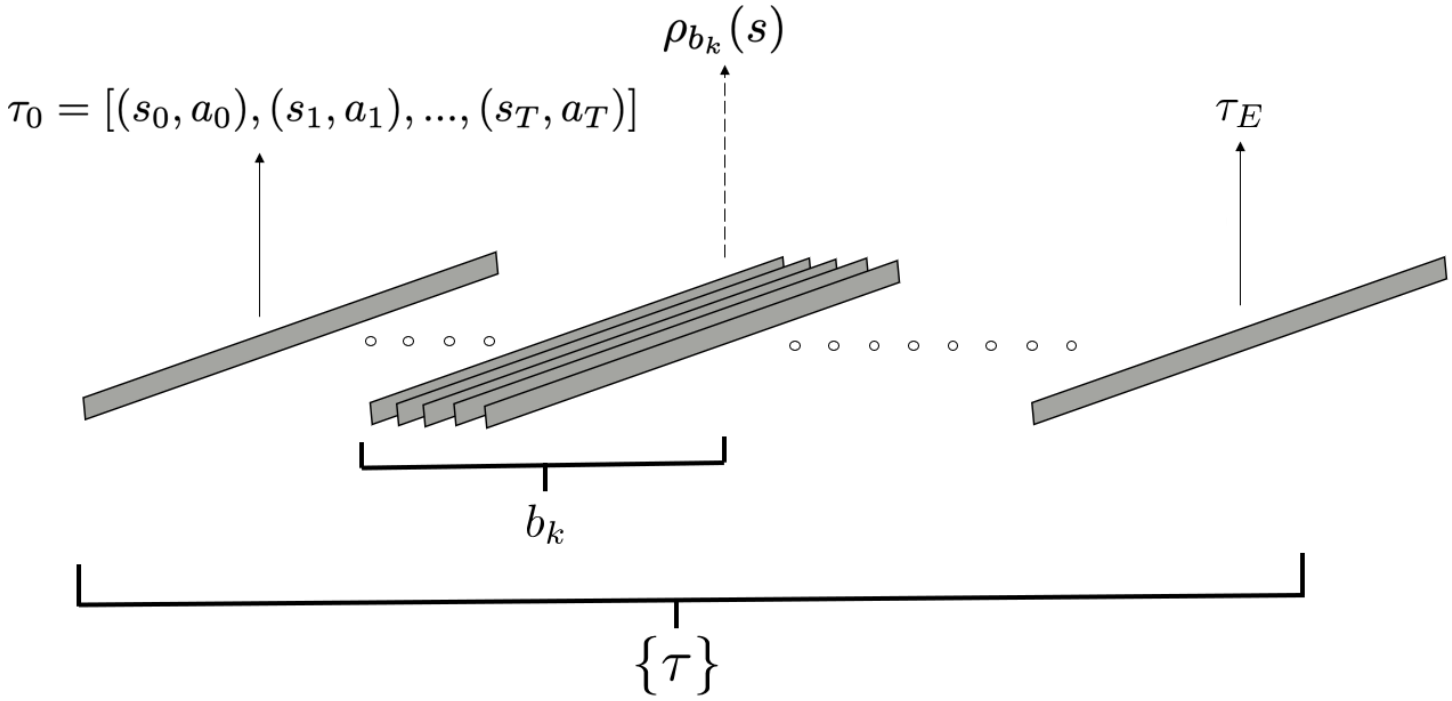
Key condition for standard behavior cloning:

- Policy is fixed
- Expert demonstrations are all sampled from the same policy

Bundle Behavior Cloning

Key Idea: Adjacent Trajectories Sampled from Similar Policies

Due to the assumption that the learning rate α is small, we know θ_t and θ_{t+1} will be close to each other.



$$\text{loss}(\psi_k) = \sum_{s \in S} \text{MSE}(\pi_\psi^k, \rho_{b_k}(s))$$

Duke UNIVERSITY

Learning Rewards with Cloned Bundles

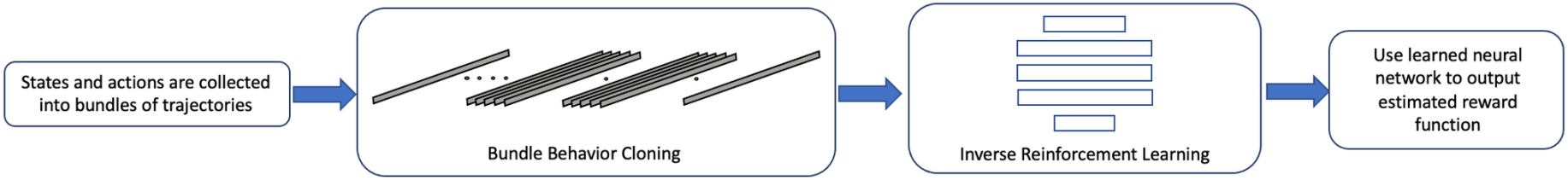
Training a Neural Network to Learn Reward Function

- Using bundle behavior cloning, we can estimate the learning agent's policy parameters θ at various points in time during learning
- We can use it to estimate the gradient $\nabla_\psi \tilde{J}$ using the gradient of the cloned policy associated with bundle
- Given the cloned policy parameters and gradients, we can train a neural network $\beta(s_{t+1})$ to approximate the agent's reward function

$$\text{loss} = \sum_{Z} (\beta(s_{t+1}) \nabla \pi_\psi^k(s_t, a_t) \gamma^t - (\psi_{k+1} - \psi_k))$$

- Z is the batch size of existing state action pairs from the forward case
- $\nabla \pi_\psi^k(a_t | s_t)$ is the estimated gradient of the cloned policy from the $k\text{th}$ bundle corresponding to the state action pair (s_t, a_t)
- $(\psi_{k+1} - \psi_k)$ is the difference in policy parameters between the $k\text{th}$ bundle and the $k+1$ bundle

Entire Pipeline of Algorithm



Duke UNIVERSITY

Outline

Problem Definition

Bundle Behavior Cloning

Complexity Analysis

- **Supporting Lemmas**
- **Main Theorem**

Empirical Results

Duke UNIVERSITY

Main Lemmas

Assumption 4: Bound on consecutive policies' distributions

$$||\pi^t(s) - \pi^{t+1}(s)||_{\text{tv}} \leq \epsilon$$

Lemma 1: Bound on policy distribution for standard behavior cloning

$$\mathbb{E}_{s\sim d^\pi} ||\hat{\pi}(\cdot|s) - \pi(\cdot|s)||^2_{\text{tv}} \leq \frac{2\ln(|\Pi|/\delta)}{T}$$

Lemma 2: Bound on state visitation distribution for standard behavior cloning

$$||d^{\pi'} - d^\pi||_1 \leq \frac{2\gamma}{1-\gamma}\mathbb{E}_{s\sim d^\pi} ||\pi'(\cdot|s) - \pi(\cdot|s)||_{\text{tv}}$$

Lemma 3: Bound on policies' weights and state visitation distributions within bundle

$$||\pi^i - \pi^j||_{\text{tv}} \leq (B-1)\epsilon$$

$$||d^{\pi^i} - d^{\pi^j}||_1 \leq \frac{2\gamma(B-1)\epsilon}{1-\gamma}$$

Duke UNIVERSITY

Main Theorem

Theorem 1: Guarantee on tighter bound achieved through bundle behavior cloning than standard behavior cloning

Find a bound for:

$$\mathbb{E}_{s\sim d^{\pi^1}} ||\hat{\pi}^{1:B}(\cdot|s) - \pi^1(\cdot|s)||^2_{\text{tv}}$$

Rewriting lefthand side and using inequality $||x-z||^2 \leq 2||x-y||^2 + 2||y-z||^2$

$$\mathbb{E}_{s\sim d^{\pi^1}} ||\hat{\rho}^B - \rho^B + \rho^B - \rho^1||^2_{\text{tv}} \leq 2\mathbb{E}_{s\sim d^{\pi^1}} ||\hat{\rho}^B - \rho^B||^2_{\text{tv}} + 2\mathbb{E}_{s\sim d^{\pi^1}} ||\rho^B - \rho^1||^2_{\text{tv}}$$

Using Lemma 3 for policies π^1 and $\pi^{1:B}$ to obtain an upper bound on the righthand side :

$$\leq 2[\mathbb{E}_{s\sim d^{\pi^1}} ||\hat{\rho}^B - \rho^B||^2_{\text{tv}} - \mathbb{E}_{s\sim d^{\pi^B}} ||\hat{\rho}^B - \rho^B||^2_{\text{tv}}] + \frac{4\ln(|\Pi|/\delta)}{BT} + 2\epsilon^2(B-1)^2$$

Using Lemma 2:

$$\leq \frac{4\gamma(B-1)\epsilon}{1-\gamma} + \frac{4\ln(|\Pi|/\delta)}{BT} + 2\epsilon^2(B-1)^2$$

Bound for standard behavior cloning with same probability $(1-\delta)^2$:

$$\mathbb{E}_{s\sim d^{\pi^1}} ||\hat{\pi}^1(\cdot|s) - \pi^1(\cdot|s)||^2_{\text{tv}} \leq \frac{2\ln(|\Pi|/(2\delta-\delta^2))}{T}$$

Duke UNIVERSITY

Outline

Problem Definition

Bundle Behavior Cloning

Complexity Analysis

Empirical Research

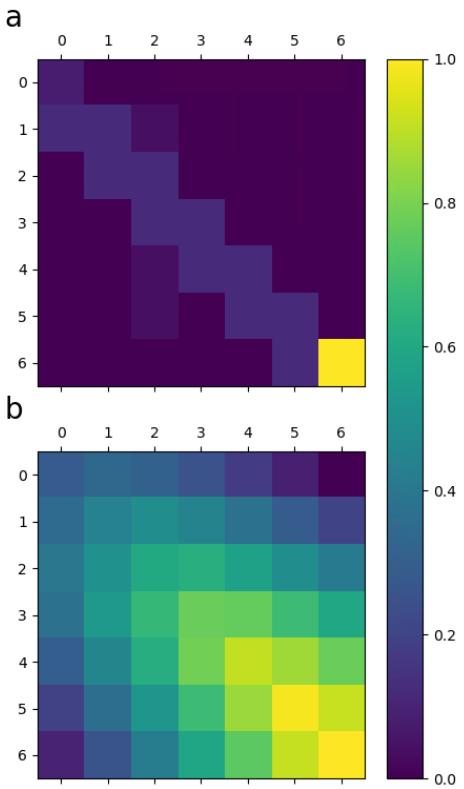
- **Learning Rewards on Gridworld Environment**
- **Independent Policies and Testing Different Neural Network Structures**
- **Learning While Learning**
- **Deviation of Cloned Policies in Between Bundles**

Duke
UNIVERSITY

Learning Rewards

Gridworld Environment

- Grid is 7 by 7, with agent starting at the top left square
- Has three possible actions, moving to the right, to the left, and down
- Agent receives a maximum reward of 20 when it reaches the goal state, located at the bottom right square
- Different squares on the grid result in different, negative rewards ranging from -5 to -2

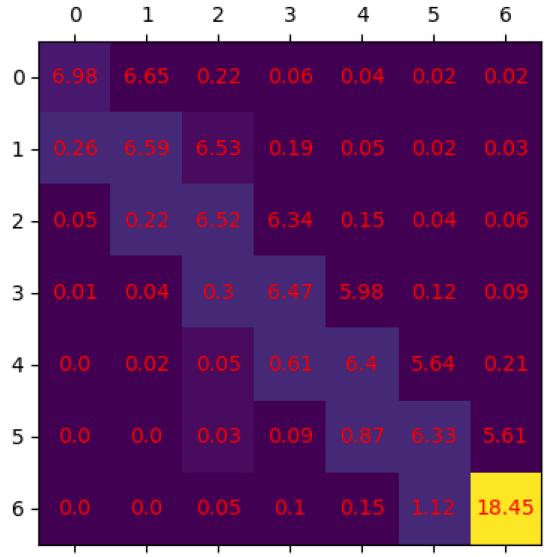


Duke UNIVERSITY

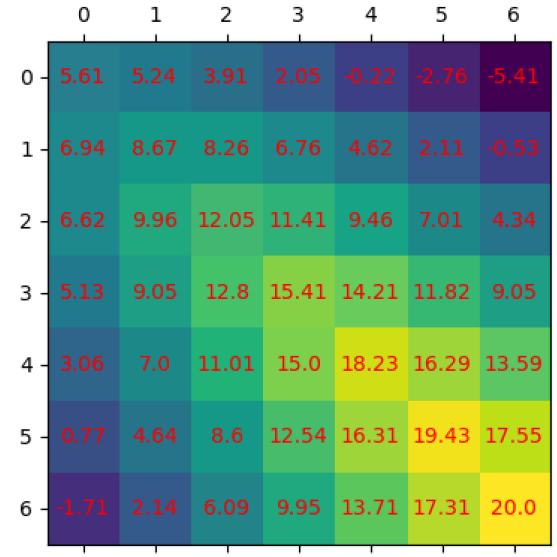
Learning Rewards

Lower and upper bound reward functions for 95% confidence interval

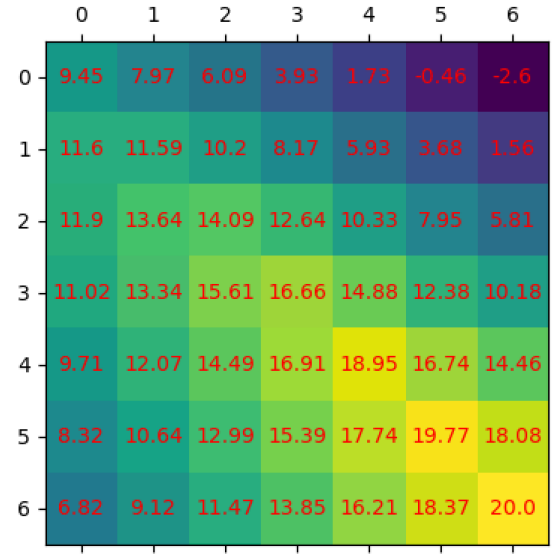
a



b



c



Duke UNIVERSITY

Testing Robustness of Algorithm

Independent Policies

- In the forward case, future policy weights depend on previous ones, so assumption that policies that generate the trajectories are independent in the theoretical analysis does not hold here
- Can simulate independent policies by doubling the size of the bundle but skipping over every other state action pair during sampling

Testing Different Neural Network Structures

- Wanted to measure robustness of algorithm when true neural network structure of learning agent is not known
- Assume the activation functions used in between layers is ReLU

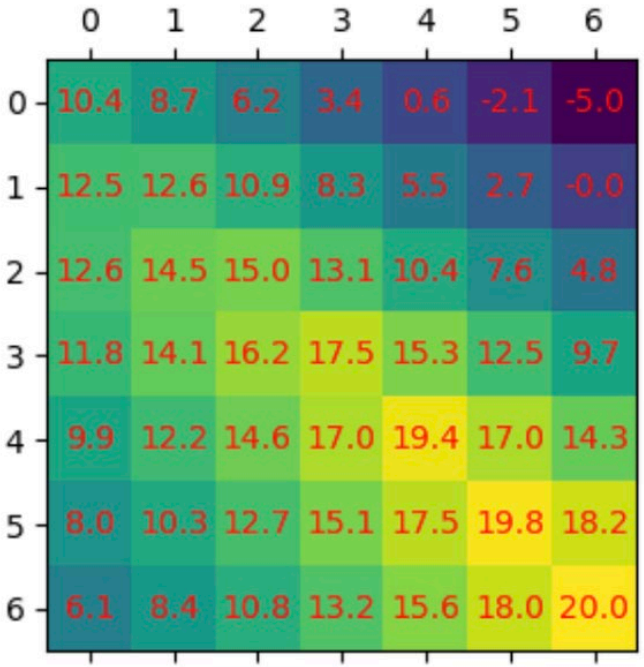
Description	Average Reward
Forward Policy: 2 layer, 16 hidden nodes	**57.461 ± 4.199**
BBC: 2 layer, 16 hidden nodes (same structure)	**57.840 ± 0.946**
BBC: 2 layer, 16 hidden nodes, independent policies	**54.551 ± 1.777**
BBC: 2 layer, 8 hidden nodes	**57.923 ± 1.248**
BBC: 2 layer, 24 hidden nodes	**57.912 ± 1.262**
BBC: 2 layer, 32 hidden nodes	46.855 ± 3.865
BBC: 3 layer, 16 hidden nodes	**52.190 ± 3.037**
BBC: 5 layer, 16 hidden nodes	**54.934 ± 1.304**

Duke UNIVERSITY

Learning While Learning

Ability to Recover Shape of Reward Function without Needing Optimal Policy

- Test our method with just the first 50 bundles rather than the full set of 333 bundles
- This corresponds to only 750 episodes from forward learning, while it took 5000 episodes for the forward case policy to converge
- With just a sixth of the total episodes from forward learning, we can start to recover the shape of the reward function, showing the bundle behavior cloning does not require an optimal policy to imitate

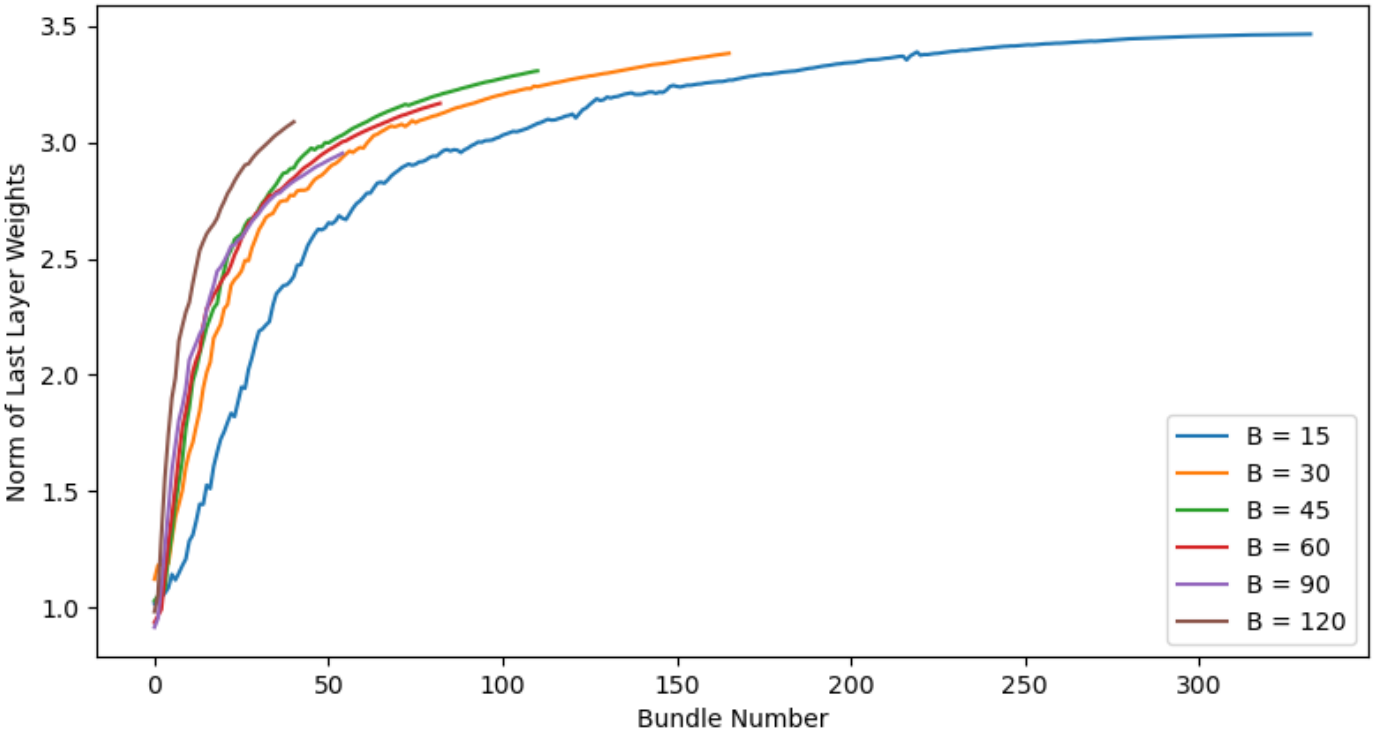


Duke UNIVERSITY

Deviation of Cloned Policies in Between Bundles

Norms of Last Layer Weights vs. Different Bundle Sizes

- Empirically test the effect of bundle choice on performance
- Note: norm of weights is not a substitute for the distribution of the policy itself
- Graph shows adjusting bundle size can help satisfy theoretical requirement in Assumption 4 as it results in policy networks that are closer to each other across consecutive bundles



Duke UNIVERSITY

Thank You

Duke
UNIVERSITY