Causal Dynamics Learning for Task-Independent State Abstractions

Peter Stone

Learning Agents Research Group (LARG)
Department of Computer Science
The University of Texas at Austin

(also Executive Director of Sony AI America)

UT Austin: Exciting Times!

- NSF Institute for Foundations of Machine Learning (IFML)
  - **Machine Learning Laboratory** (MLL)

UT Austin: Exciting Times!

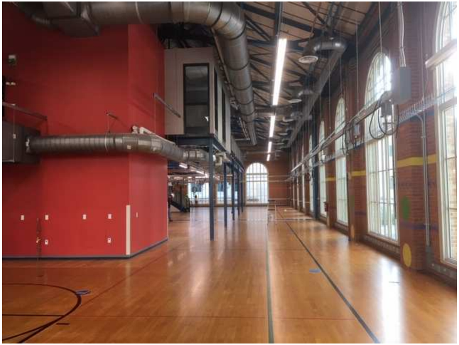
- NSF Institute for Foundations of Machine Learning (IFML)
  - **Machine Learning Laboratory** (MLL)
- Bridging Barriers: **Good Systems**
  - Inspired by **One Hundred Year Study on AI (AI100)**

UT Austin: Exciting Times!

- NSF Institute for Foundations of Machine Learning (IFML)
  - **Machine Learning Laboratory** (MLL)
- Bridging Barriers: **Good Systems**
  - Inspired by **One Hundred Year Study on AI (AI100)**
- Director of **Texas Robotics**
  - Ribbon cutting on new building in 2021

UT Austin: Exciting Times!

- NSF Institute for Foundations of Machine Learning (IFML)
  - **Machine Learning Laboratory** (MLL)
- Bridging Barriers: **Good Systems**
  - Inspired by **One Hundred Year Study on AI (AI100)**
- Director of **Texas Robotics**
  - Ribbon cutting on new building in 2021



CORE FACULTY

TEXAS Robotics

My Research Question

To what degree can autonomous intelligent agents learn in the presence of teammates and/or adversaries in real-time, dynamic domains?

My Research Question

To what degree can autonomous intelligent agents learn in the presence of teammates and/or adversaries in real-time, dynamic domains?

Research Areas

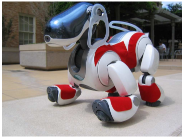
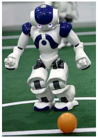
- Autonomous agents
- Robotics
- Machine learning
  - **Reinforcement learning**
- **Multiagent systems**

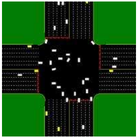
My Research Question

To what degree can autonomous intelligent agents learn in the presence of teammates and/or adversaries in real-time, dynamic domains?

Research Areas

- Autonomous agents
- Robotics
- Machine learning
  - **Reinforcement learning**
- **Multiagent systems**

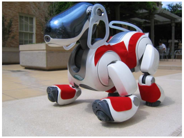
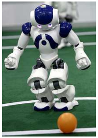



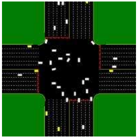
My Research Question

To what degree can autonomous intelligent agents learn in the presence of teammates and/or adversaries in real-time, dynamic domains?

Research Areas

- Autonomous agents
- Robotics
- Machine learning
  - **Reinforcement learning**
- **Multiagent systems**



GT Sophy

State Abstraction Discovery for Generalization

State Abstraction Discovery for Generalization

- Learn which state variables to ignore (and when)

State Abstraction Discovery for Generalization

- Learn which state variables to ignore (and when)
  - based on policy irrelevance (IJCAI 2005)

State Abstraction Discovery for Generalization

- Learn which state variables to ignore (and when)
  - based on policy irrelevance (IJCAI 2005)
  - based on causal dynamics (ICML 2022)

Causal Dynamics Learning for Task-Independent State Abstractions

Peter Stone

Learning Agents Research Group (LARG)
Department of Computer Science
The University of Texas at Austin

(also Executive Director of Sony AI America)

State Abstraction Discovery from Irrelevant State Variables

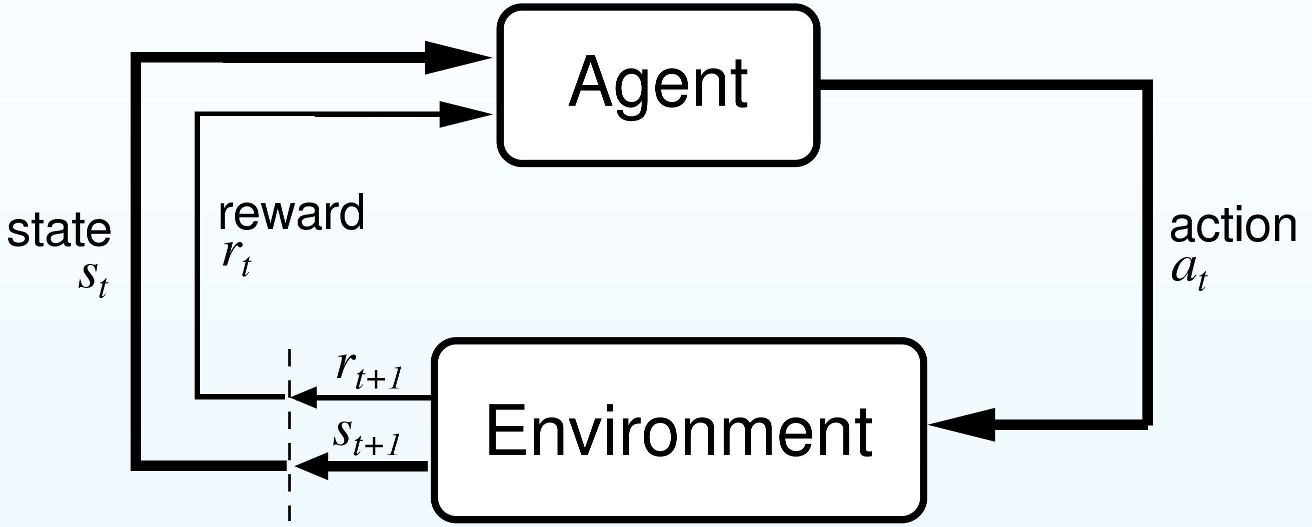
Nicholas K. Jong and Peter Stone

{nkj,pstone}@cs.utexas.edu.

Department of Computer Sciences

The University of Texas at Austin

Reinforcement Learning



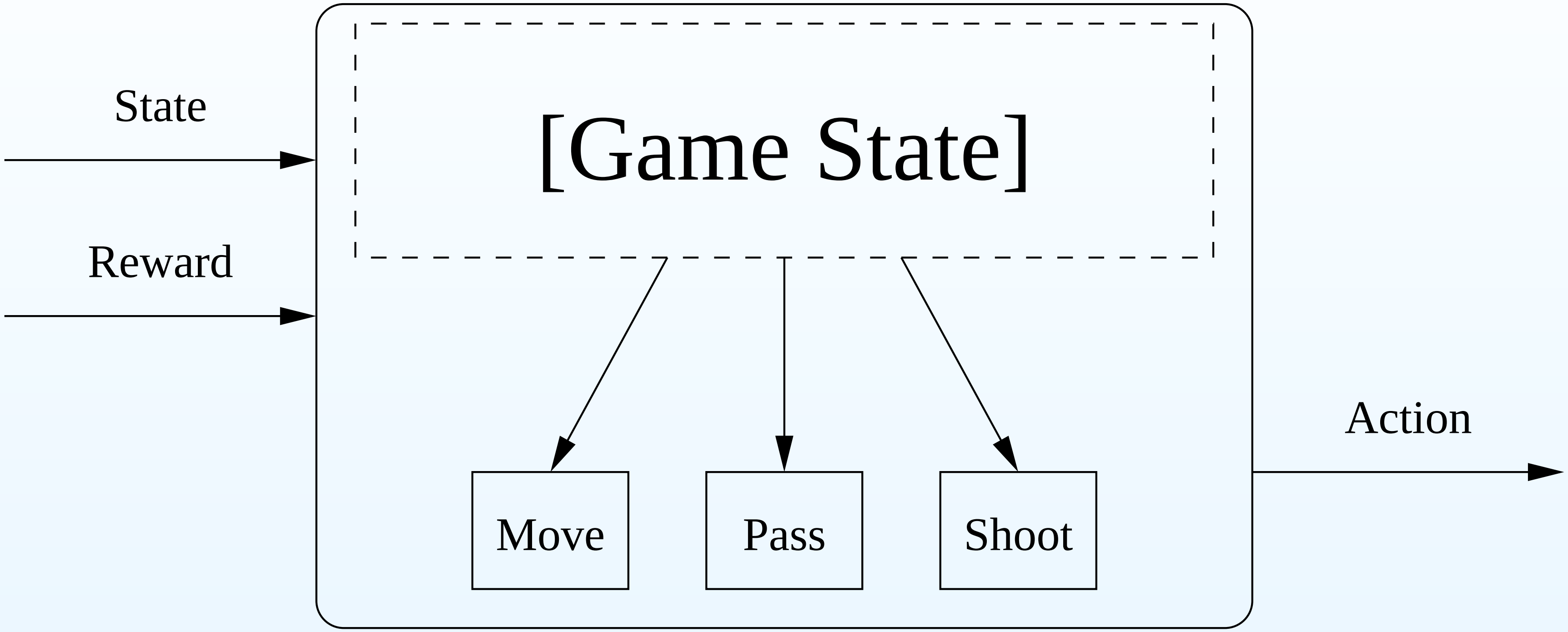
- Task: Maximize rewards in an unknown environment
- Only given: the state-action interface
- Much research: learn policies given an arbitrary interfaces
- Our research: discover interfaces that are easier to learn

Value-Based RL

Learn: a control policy

“What action should I choose in each state?”

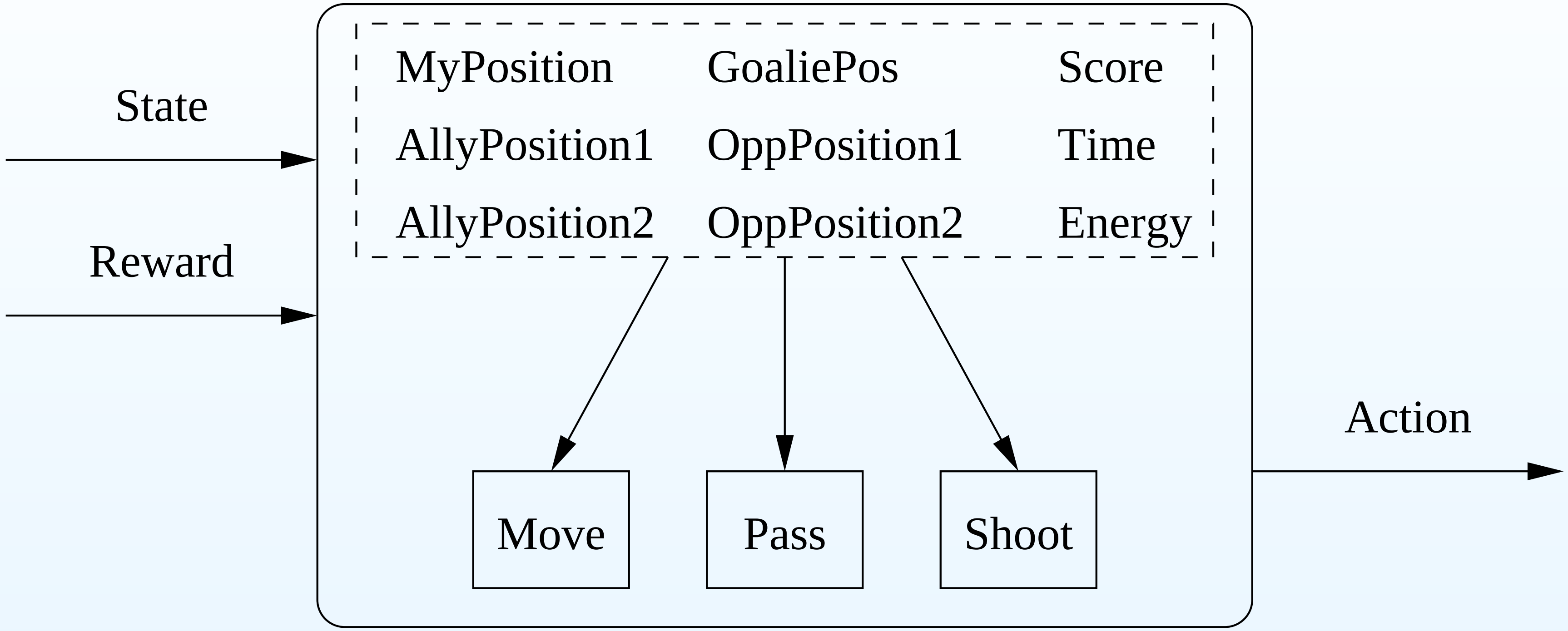
Value-Based RL



Learn: $Q : S \times A \to \mathbb{R}$

"How much reward can I earn starting at s by choosing a?"

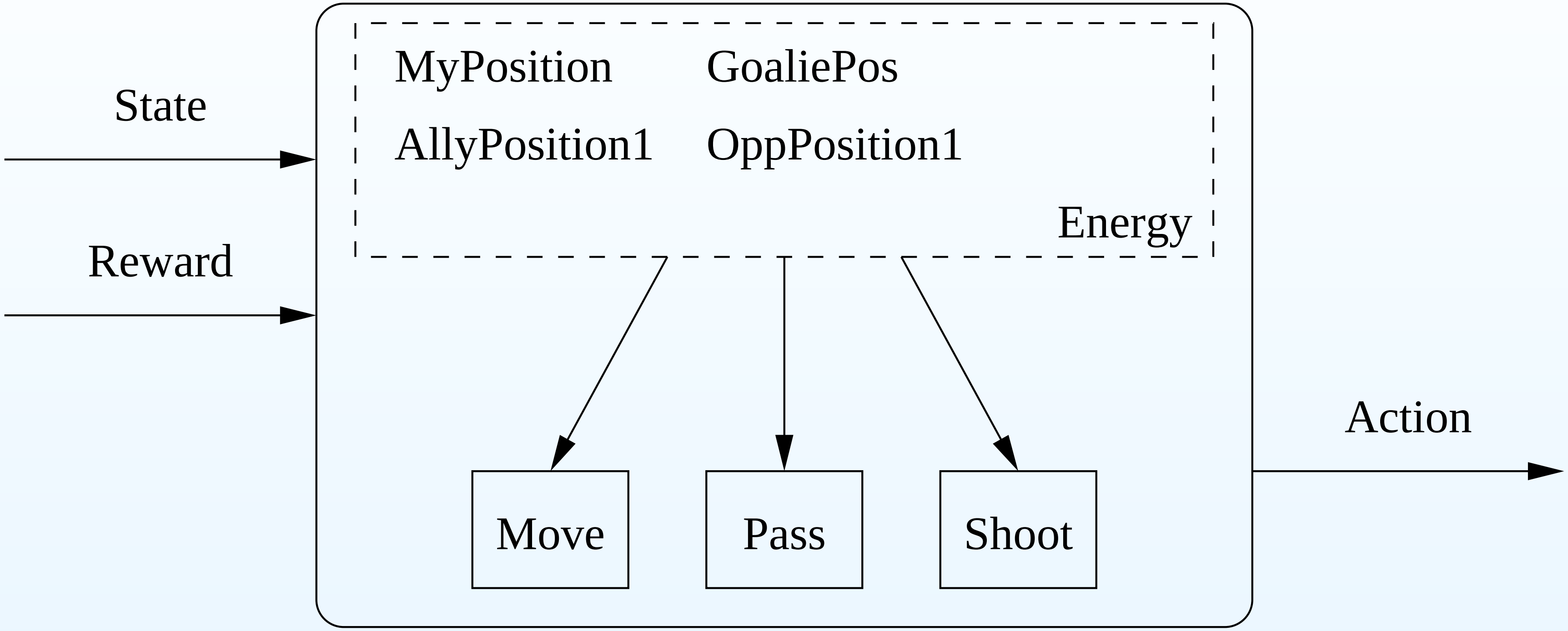
Value-Based RL



Learn: $Q : F_1 \times F_2 \times F_3 \times F_4 \times F_5 \times F_6 \times F_7 \times F_8 \times F_9 \times A \rightarrow \mathbb{R}$

In practice: high-dimensional state spaces

Value-Based RL



Learn: $Q : F_1 \times F_2 \times F_4 \times F_5 \times F_9 \times A \longrightarrow \mathbb{R}$

State abstraction: ignore the irrelevant dimensions

State abstraction as qualitative knowledge

- Traditional sources of abstraction
  - Prior knowledge from a human
  - Computation from a given model

State abstraction as qualitative knowledge

- Traditional sources of abstraction
  - Prior knowledge from a human
  - Computation from a given model
- Automatic discovery?
  - But discovering structure is harder than learning policies

State abstraction as qualitative knowledge

- Traditional sources of abstraction
  - Prior knowledge from a human
  - Computation from a given model
- Automatic discovery?
  - But discovering structure is harder than learning policies
  - Our approach: knowledge transfer

1. Discover abstractions in easy domains
2. Transfer abstractions to hard domains

Policy irrelevance: A new basis for state abstraction

When should we ignore a feature?

- Prior work
  - *... if the states share the same abstract one-step model.*
  - Requires the true model of the environment
  - Depends on the global abstraction

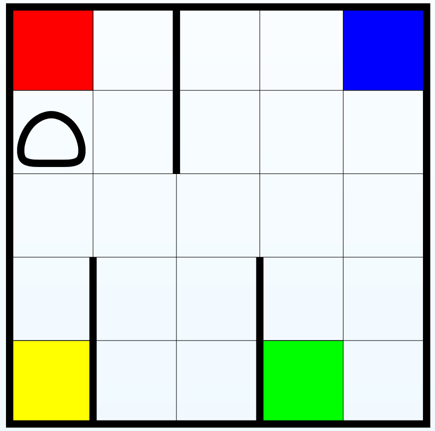
Policy irrelevance: A new basis for state abstraction

When should we ignore a feature?

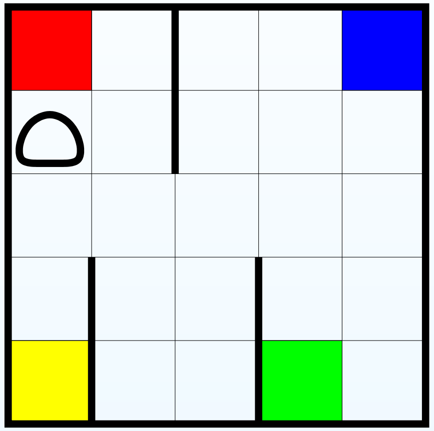
- Prior work
  - *... if the states share the same abstract one-step model.*
  - Requires the true model of the environment
  - Depends on the global abstraction
- Our work
  - *... if the states share the same optimal action.*
  - Requires a learned policy for the environment
  - Independent of abstraction at other states

The Taxi domain

- Four features
  - Taxi x coordinate
  - Taxi y coordinate
  - Current passenger location
  - Passenger destination

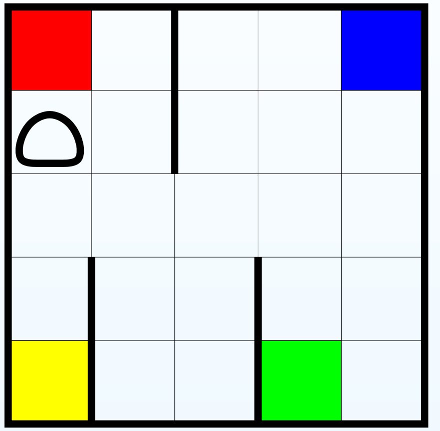


The Taxi domain



- Four features
  - Taxi x coordinate
  - Taxi y coordinate
  - Current passenger location
  - Passenger destination
- Six actions: North, South, East, West, Pick Up, Put Down

The Taxi domain



- Four features
  - Taxi x coordinate
  - Taxi y coordinate
  - Current passenger location
  - Passenger destination
- Six actions: North, South, East, West, Pick Up, Put Down
- Optimal policy:
  - Navigate to the passenger's location
  - Pick up the passenger
  - Navigate to the passenger's destination
  - Put down the passenger

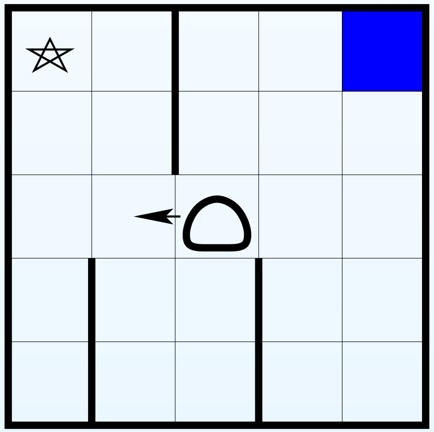
Policy irrelevance in the Taxi domain

Relevance of the passenger destination. . .

Policy irrelevance in the Taxi domain

Relevance of the passenger destination. . .

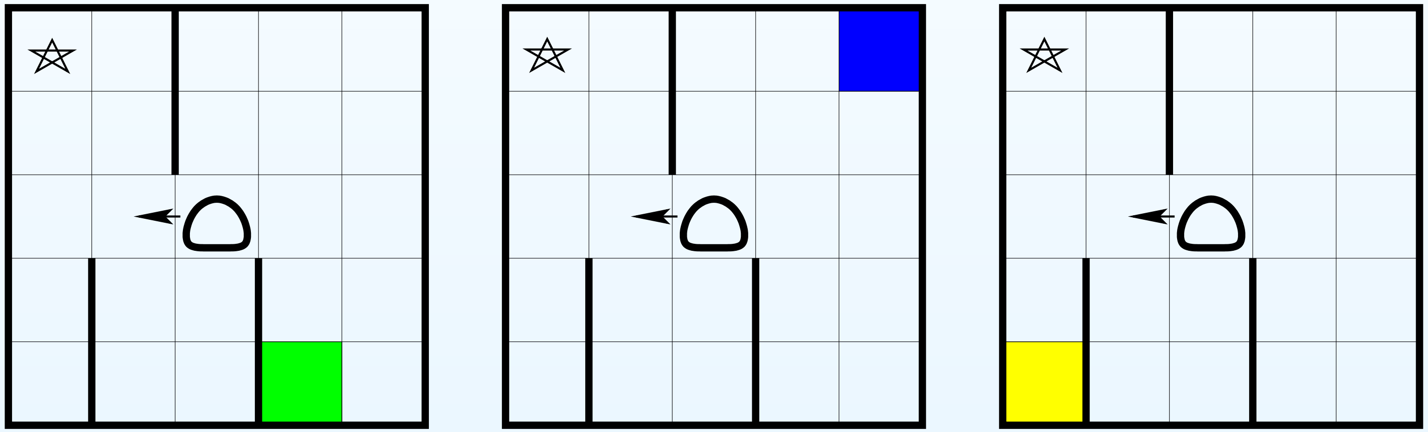
- When the passenger is not inside the taxi



Policy irrelevance in the Taxi domain

Relevance of the passenger destination. . .

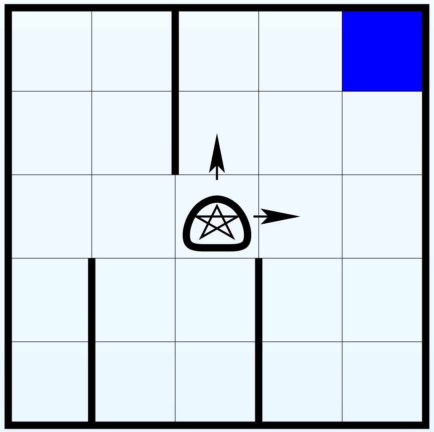
- When the passenger is not inside the taxi



Policy irrelevance in the Taxi domain

Relevance of the passenger destination. . .

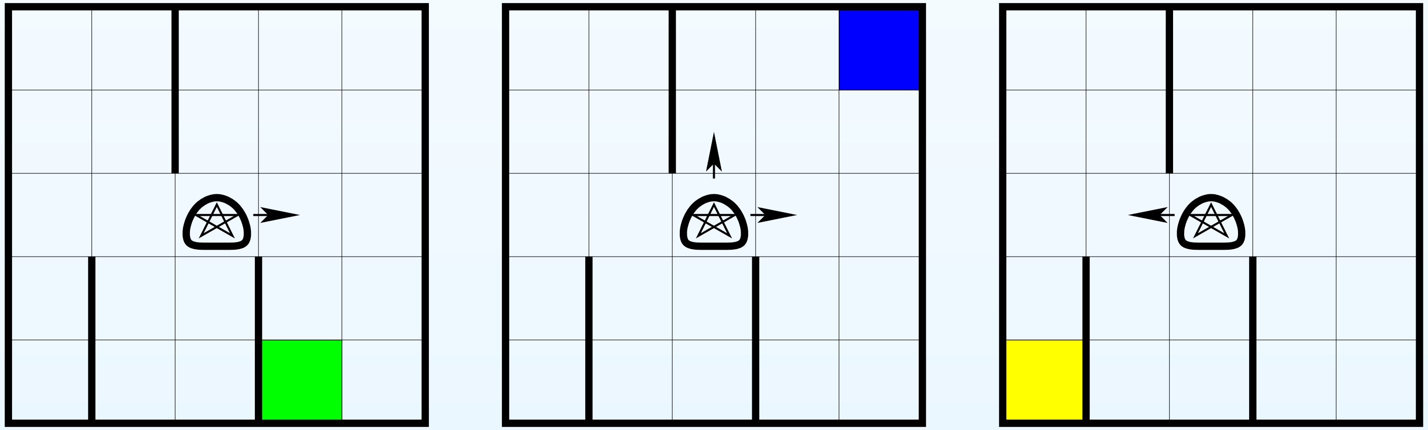
- When the passenger is not inside the taxi
- When the passenger is inside the taxi



Policy irrelevance in the Taxi domain

Relevance of the passenger destination. . .

- When the passenger is not inside the taxi
- When the passenger is inside the taxi



Policy irrelevance with real data

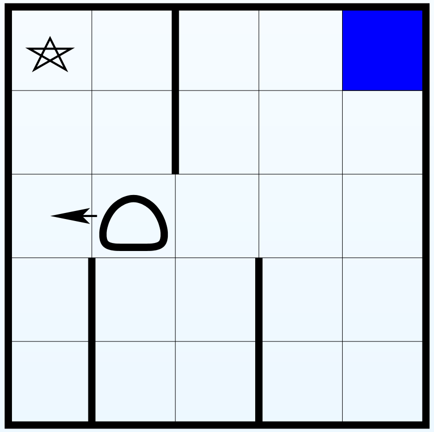
Relevance of the passenger destination...

- When the policy is learned from data

Policy irrelevance with real data

Relevance of the passenger destination. . .

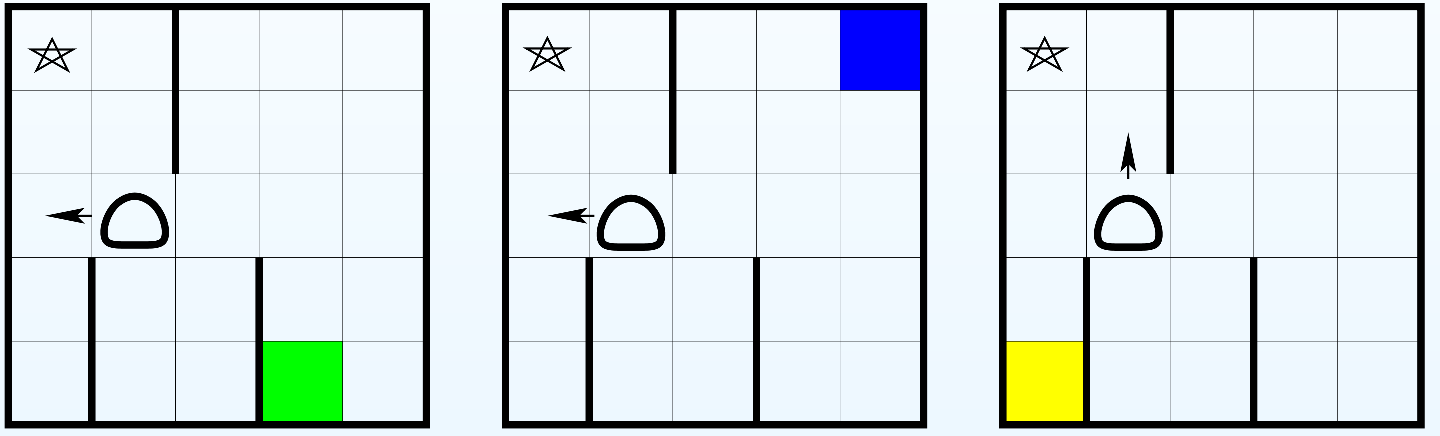
- When the policy is learned from data



Policy irrelevance with real data

Relevance of the passenger destination. . .

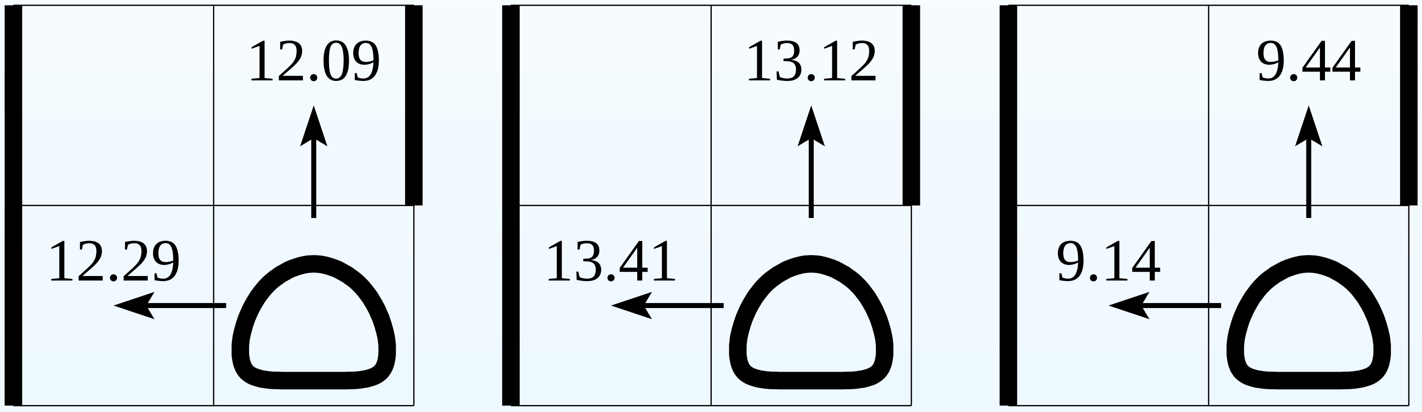
- When the policy is learned from data



Policy irrelevance with real data

Relevance of the passenger destination. . .

- When the policy is learned from data



Policy irrelevance from action-value comparisons

$$Q(s', a) \geq Q(s', a')$$

When should we ignore a set of features F at a state s?

Policy irrelevance from action-value comparisons

$$Q(s', a) \geq Q(s', a')$$

- Action a is better than action a' at state s'

When should we ignore a set of features F at a state s?

Policy irrelevance from action-value comparisons

$$\forall a' \; Q(s', a) \geq Q(s', a')$$

- Action a is better than action a' at state s'
- Action a is optimal at state s'

When should we ignore a set of features F at a state s?

Policy irrelevance from action-value comparisons

$$\forall s' \in [s]_F \; \forall a' \; Q(s', a) \geq Q(s', a')$$

- Action a is better than action a' at state s'
- Action a is optimal at state s'
- Action a is optimal at every state $s' \in [s]_F$

When should we ignore a set of features F at a state s?

($[s]_F$ is the set of states obtained from s by varying over F)

Policy irrelevance from action-value comparisons

$$\exists a \, \forall s' \in [s]_F \, \forall a' \; Q(s', a) \geq Q(s', a')$$

- Action a is better than action a' at state s'
- Action a is optimal at state s'
- Action a is optimal at every state $s' \in [s]_F$
- Some action is optimal at every $s' \in [s]_F$

When should we ignore a set of features F at a state s?

($[s]_F$ is the set of states obtained from s by varying over F)

Policy irrelevance from action-value comparisons

$$\exists a \; \forall s' \in [s]_F \; \forall a' \; Q(s', a) \geq Q(s', a')$$

- Action a is better than action a' at state s'
- Action a is optimal at state s'
- Action a is optimal at every state $s' \in [s]_F$
- Some action is optimal at every $s' \in [s]_F$
- Features F are policy irrelevant at s

When should we ignore a set of features F at a state s?

($[s]_F$ is the set of states obtained from s by varying over F)

Robust action-value comparison via sampling

$$\boxed{Q(s', a) \overset{?}{\geq} Q(s', a')}$$

Robust action-value comparison via sampling

$$\boxed{Q(s', a) \overset{?}{\geq} Q(s', a')}$$

- Compare samples of estimates, not individual estimates!

Robust action-value comparison via sampling

$$\boxed{Q(s', a) \overset{?}{\geq} Q(s', a')}$$

- Compare samples of estimates, not individual estimates!
- Method 1: Statistical hypothesis testing
  - Solve task repeatedly with a value-based RL algorithm
  - Low computational but high sample complexity

Robust action-value comparison via sampling

$$\boxed{Q(s', a) \overset{?}{\geq} Q(s', a')}$$

- Compare samples of estimates, not individual estimates!
- Method 1: Statistical hypothesis testing
  - Solve task repeatedly with a value-based RL algorithm
  - Low computational but high sample complexity
- Method 2: Monte Carlo simulation
  - Construct a Bayesian model from an experience trace
  - Low sample but high computational complexity

Partial state abstractions

$\boxed{\text{Are features } F \text{ relevant at state } s\text{?}}$

$$\downarrow$$

$\boxed{\text{At what states is each set of features relevant?}}$

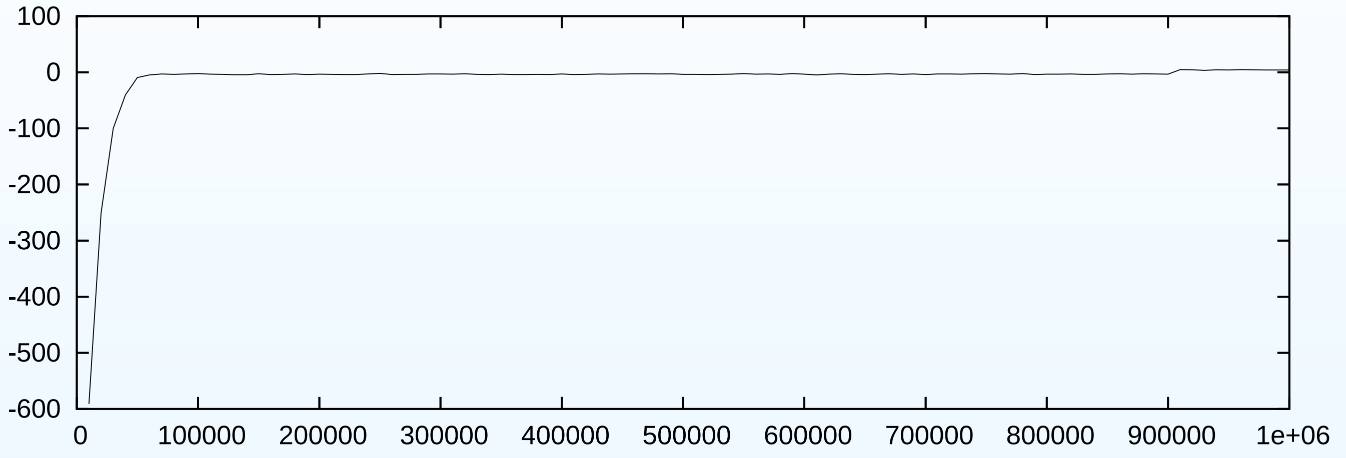
- Train a binary classifier for certain sets of features
- Learn when each set of features is irrelevant
- Naive application: ignore F at classified states

Transferring abstractions to novel domains

- Sources of error for straightforward state aggregation
  - Statistical testing error
  - Generalization error of the learned classifiers
  - Novelty in the transfer domain
  - Disruption of value-function semantics!

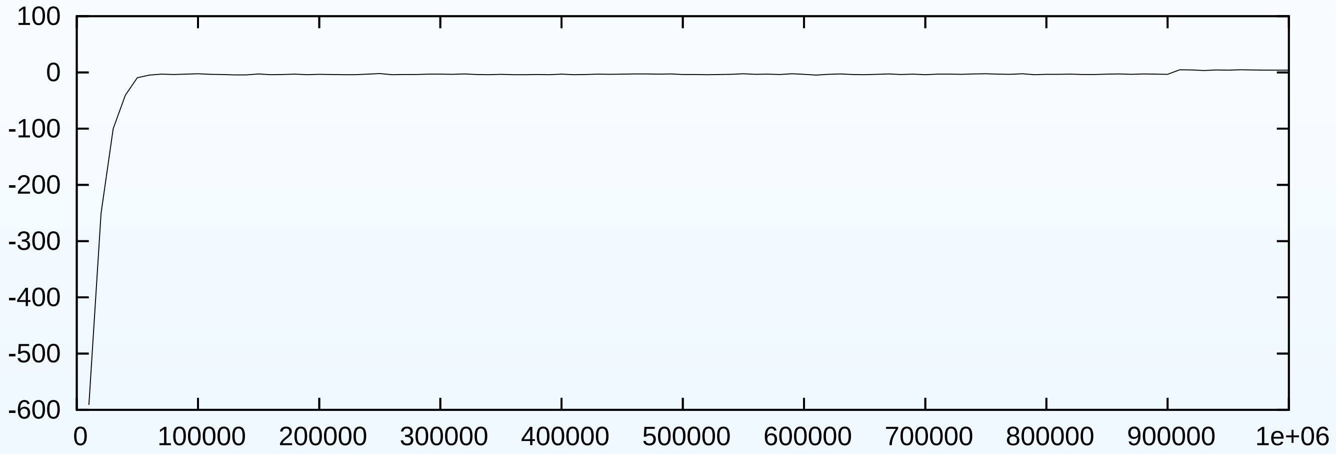
Results in the Taxi domain

- Original 5×5 domain

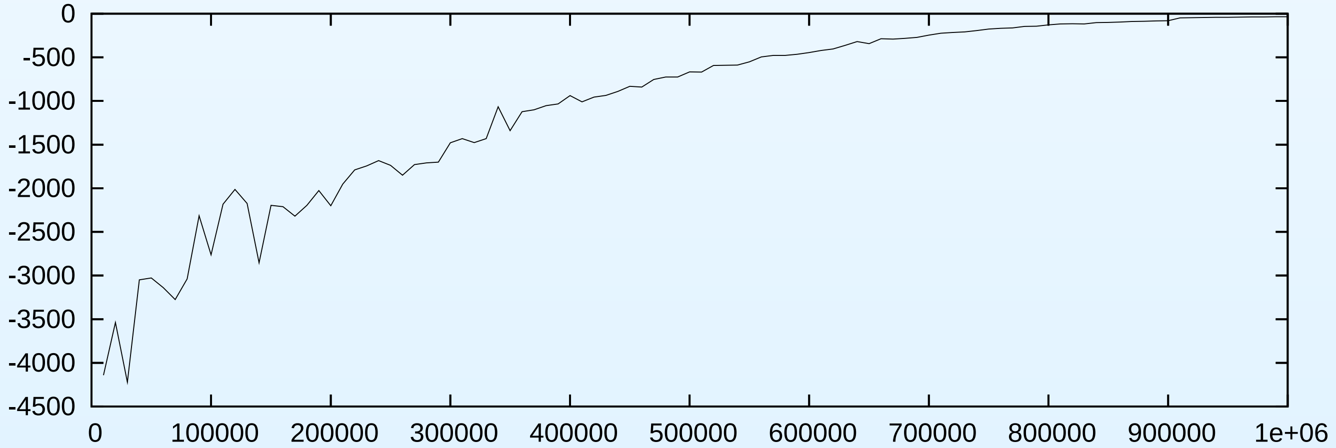


Results in the Taxi domain

- Original 5×5 domain

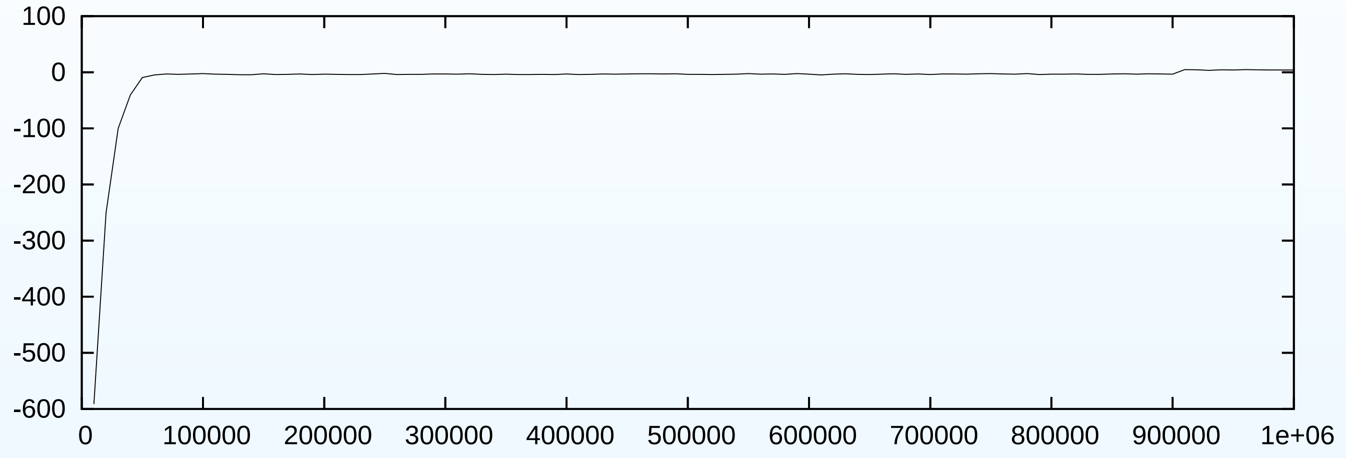


- Randomly generated 10×10 domain

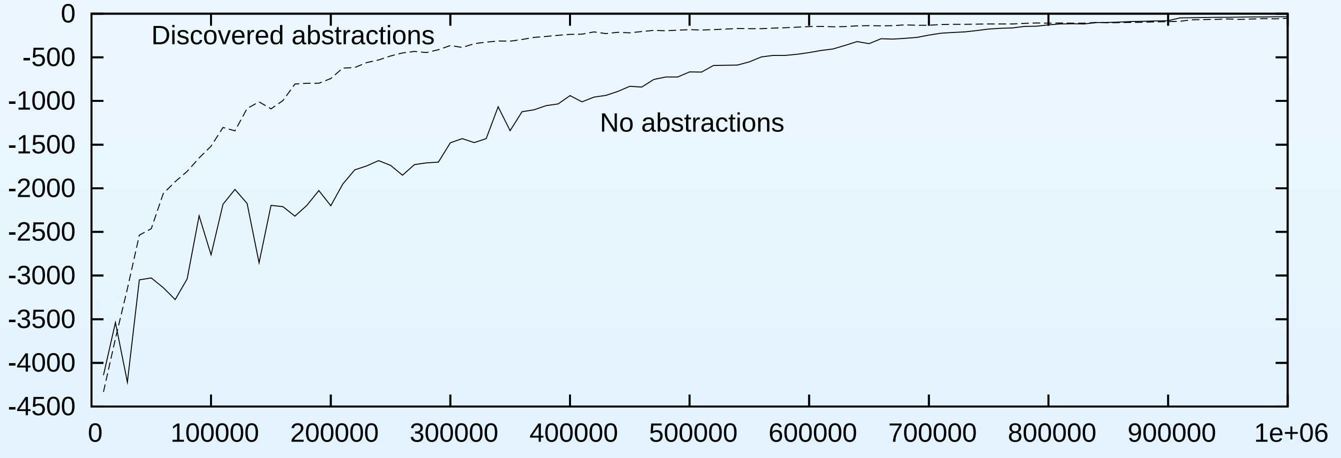


Results in the Taxi domain

- Original 5×5 domain



- Randomly generated 10×10 domain



Conclusions

- Abstraction discovery as problem reformulation
- A new basis for state abstraction: policy irrelevance
  - Statistical testing methods
  - Trajectory-based discovery algorithm
- Safe transfer of state abstractions to novel domains
  - Encapsulation inside temporal abstractions
  - Synergy of temporal and state abstractions

Causal Dynamics Learning for Task-Independent State Abstraction

Zizhao Wang, Xuesu Xiao, Zifan Xu, Yuke Zhu, and Peter Stone





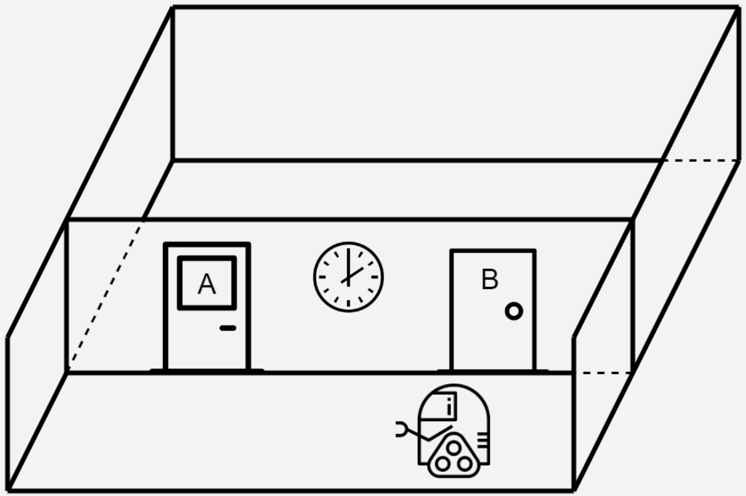
LARG Learning Agents Research Group The University of Texas at Austin | RPL UT Robot Perception & Learning Lab | Sony AI

Motivation

Real-world dynamics are usually ***sparse***.

- The transition of each state variable only depends on a few state variables.

For example, for an environment with a robot, two doors and a clock on the wall:

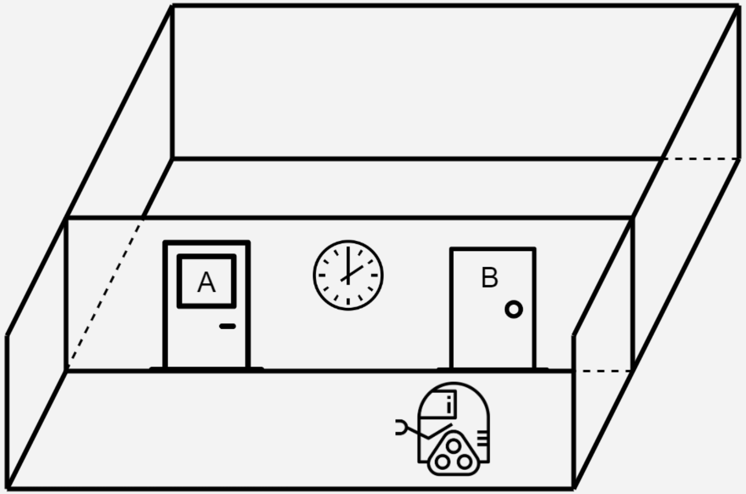


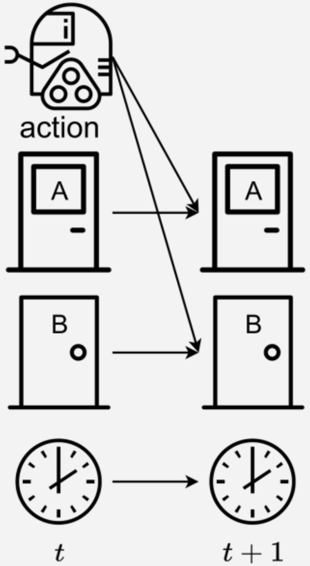
Motivation

Real-world dynamics are usually ***sparse***.

- The transition of each state variable only depends on a few state variables.

For example, for an environment with a robot, two doors and a clock on the wall:

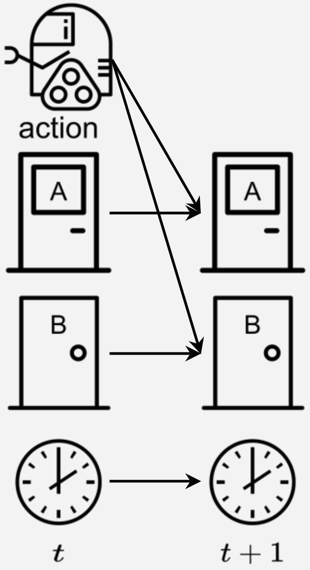




sparse real-world dynamics

Motivation

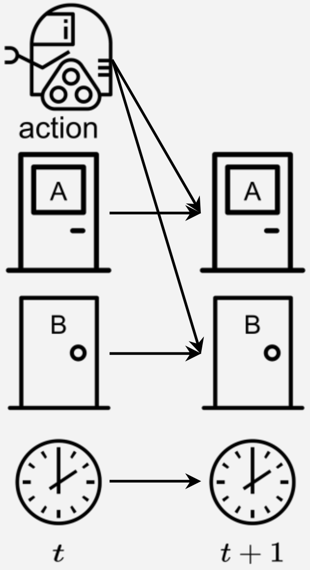
But most model-based RL work uses dense dynamics models (fully-connected networks).



sparse real-world dynamics

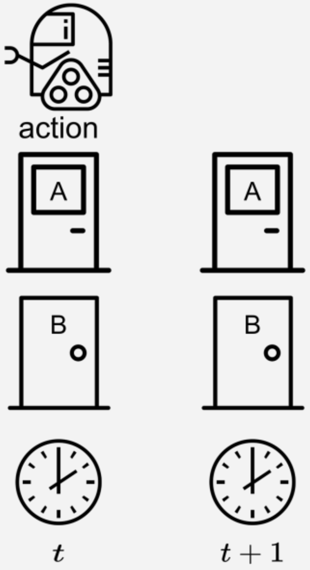
Motivation

But most model-based RL work uses dense dynamics models (fully-connected networks).



sparse real-world dynamics

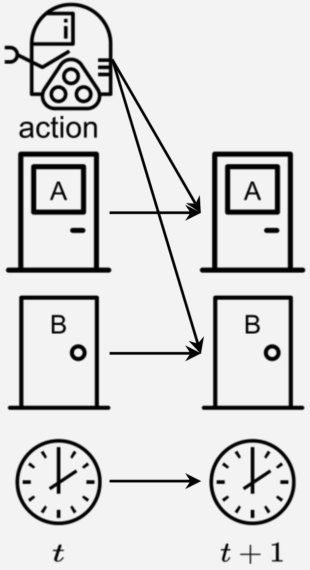
VS



dense dynamics model

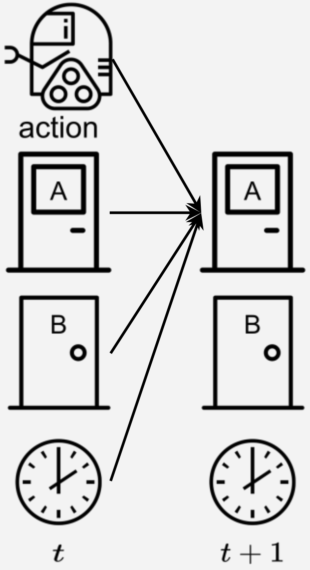
Motivation

But most model-based RL work uses dense dynamics models (fully-connected networks).



sparse real-world dynamics

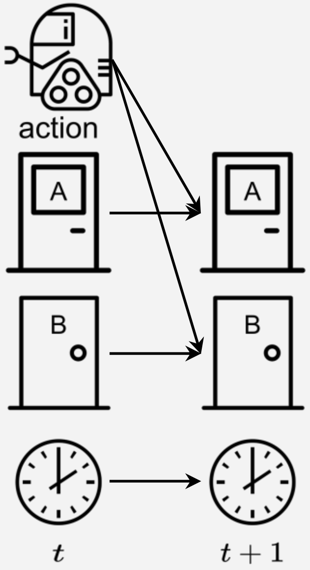
VS



dense dynamics model

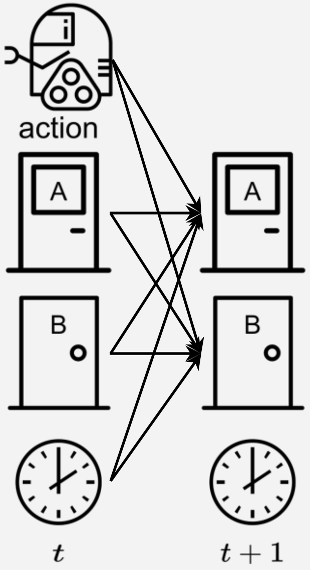
Motivation

But most model-based RL work uses dense dynamics models (fully-connected networks).



sparse real-world dynamics

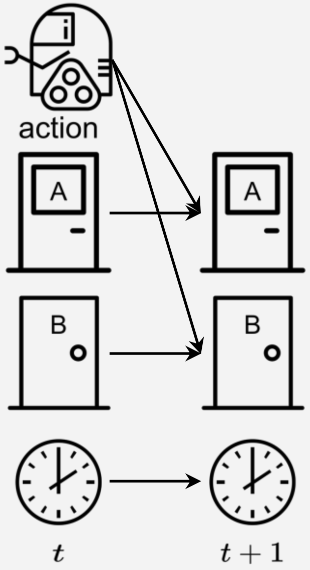
VS



dense dynamics model

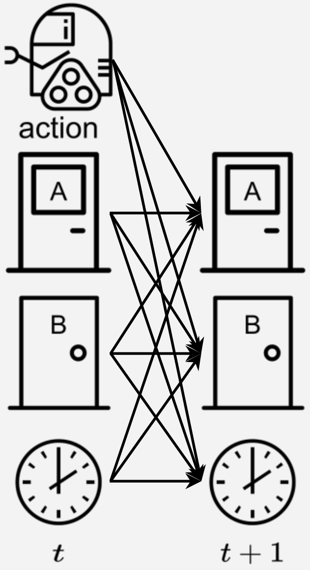
Motivation

But most model-based RL work uses dense dynamics models (fully-connected networks).



sparse real-world dynamics

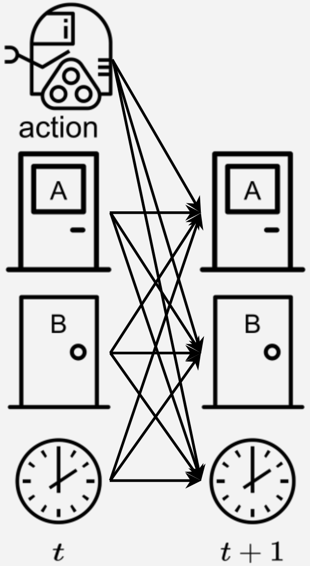
VS



dense dynamics model

Motivation

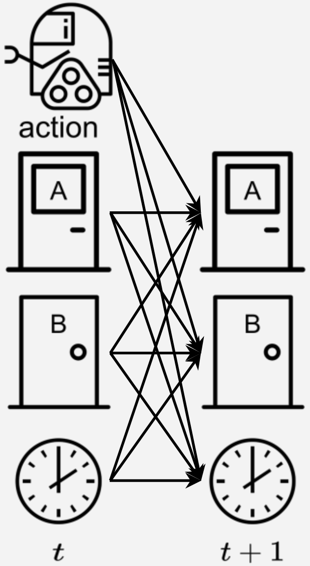
dense dynamics model



generalizes badly
due to spurious correlation

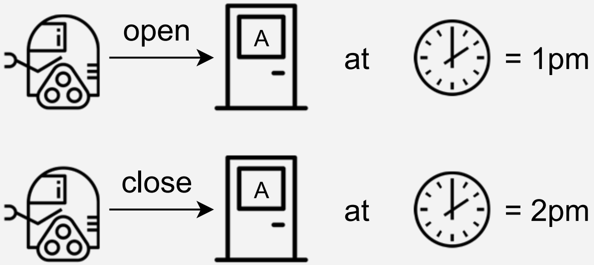
Motivation

dense dynamics model



generalizes badly
due to spurious correlation

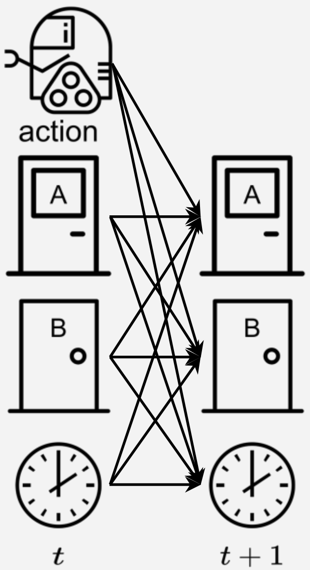
training



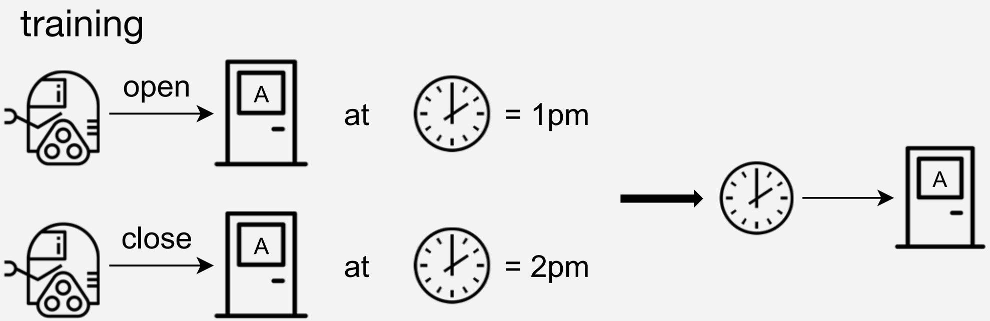
overfit to data noise, etc

Motivation

dense dynamics model



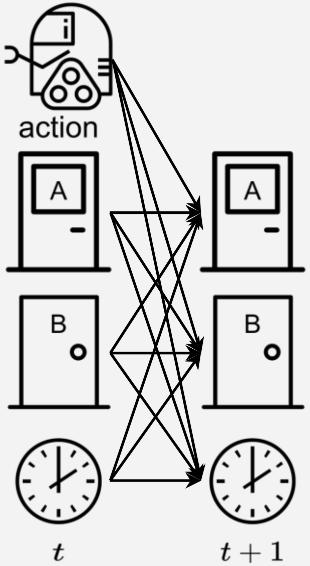
generalizes badly
due to spurious correlation



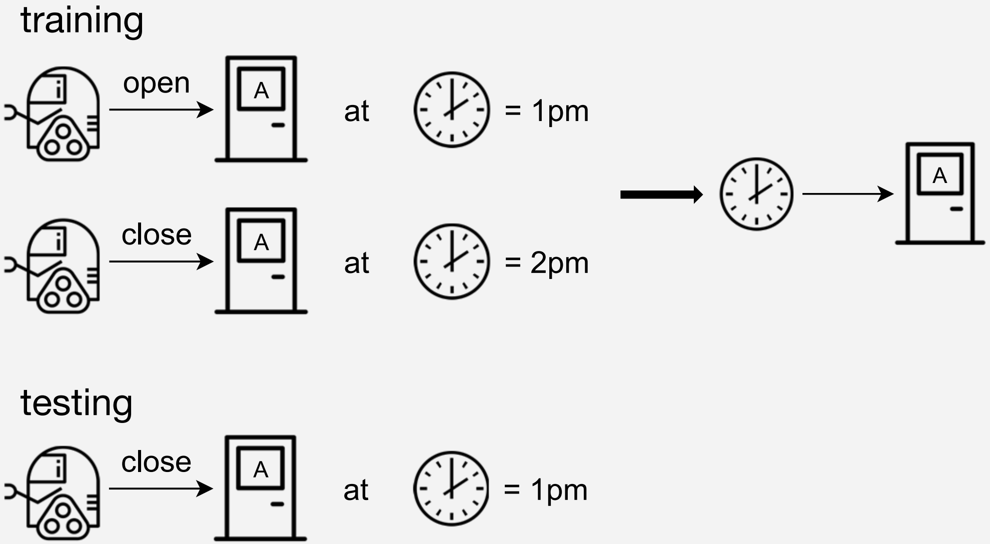
overfit to data noise, etc

Motivation

dense dynamics model



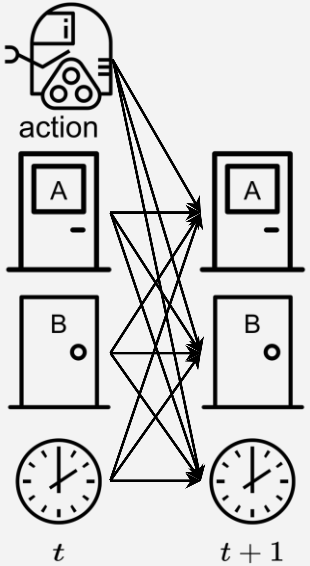
generalizes badly
due to spurious correlation



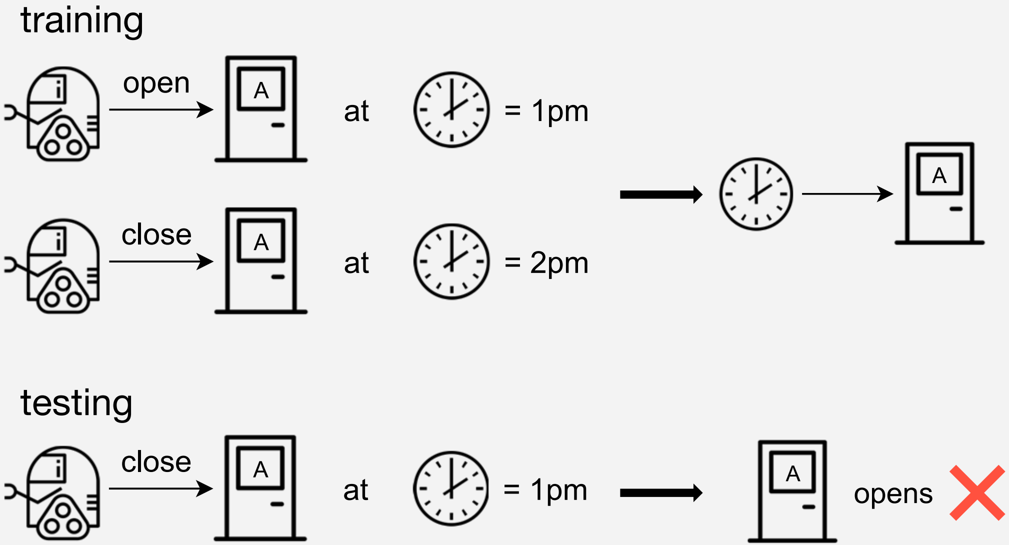
overfit to data noise, etc

Motivation

dense dynamics model



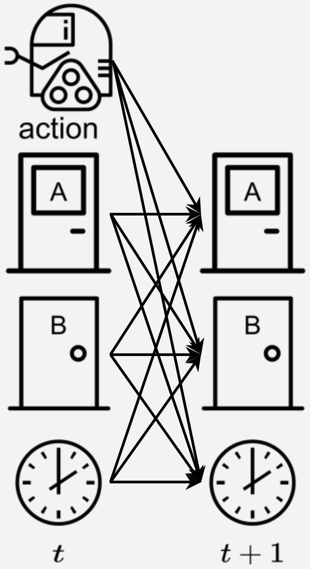
generalizes badly
due to spurious correlation



overfit to data noise, etc

Motivation

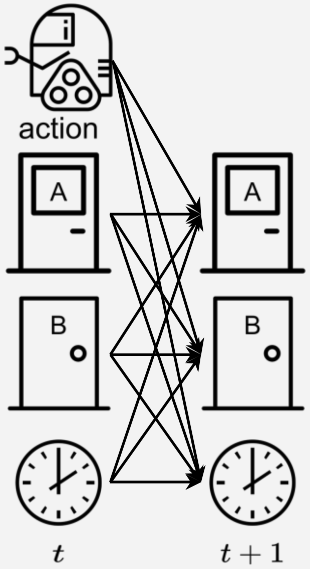
dense dynamics model



generalizes badly
due to spurious correlation

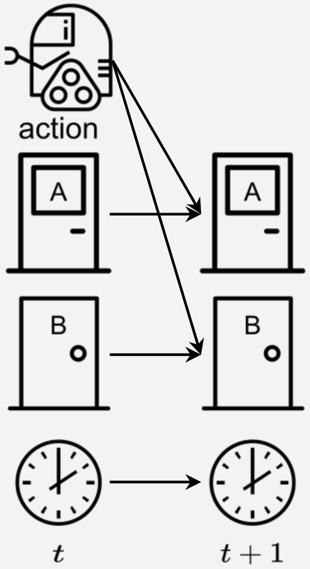
Motivation

dense dynamics model



generalizes badly
due to spurious correlation

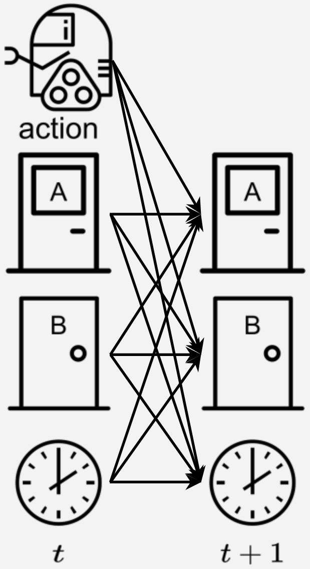
causal dynamics learning (CDL)



only keep causal edges, robust to outliers,

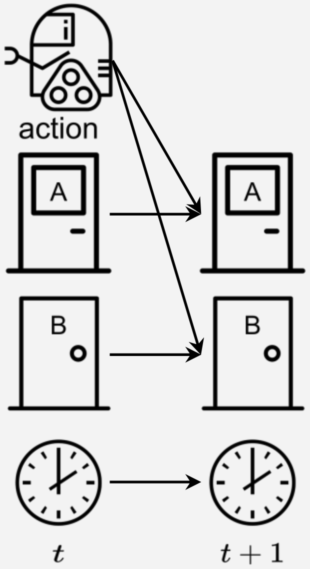
Motivation

dense dynamics model



generalizes badly
due to spurious correlation

causal dynamics learning (CDL)



only keep causal edges, robust to outliers,
e.g., clock outliers won't affect door A & B prediction

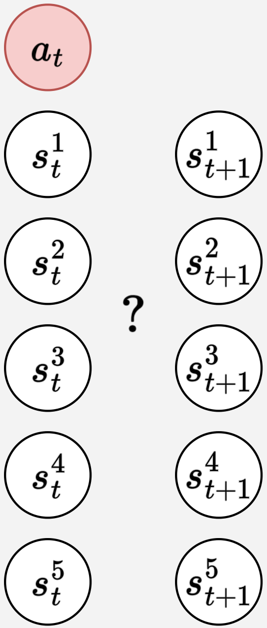
Problem Setup

$$< \mathcal{S}, \mathcal{A}, \mathcal{P} >$$

S: state space (known, *high-level* variables are given)
We leave handling low-level, partially-observable state space (e.g., images) as future work.

A: action space (known)

P: transition probability (not known)



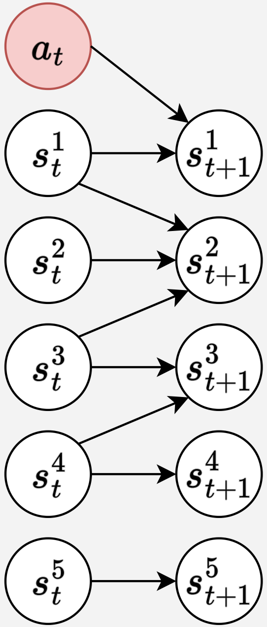
Problem Setup

Goals

1. Learn a causal dynamics model from transition data

$$\mathcal{P}(s_{t+1} | s_t, a_t) = \prod_{i=1}^{d_S} \mathcal{P}(s^i_{t+1} | \mathbf{PA}_{s^i})$$

   $\mathbf{PA}_{s^i}$ are parents of s^i during the data generation process.



Problem Setup

Goals

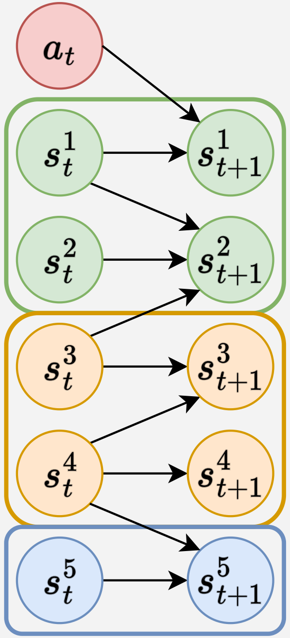
1. Learn a causal dynamics model from transition data
2. Split state variables into three categories

$$\mathcal{S} = \mathcal{S}^c \times \mathcal{S}^c \times \mathcal{S}^i$$

S^c: space of controllable state variables

S^r: space of action-relevant state variables

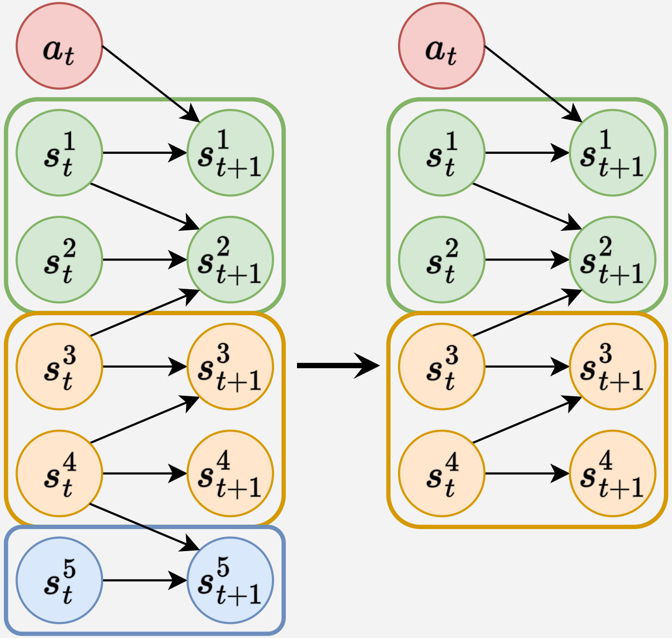
S^i: space of action-irrelevant state variables



Problem Setup

Goals

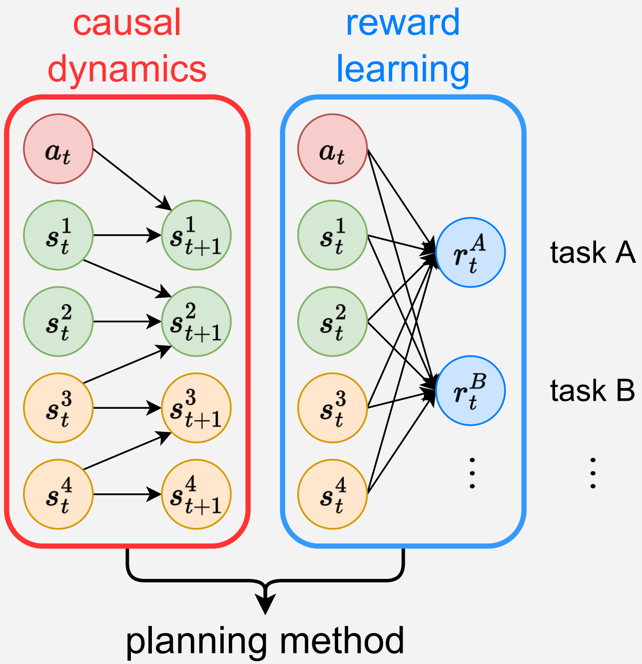
1. Learn a causal dynamics model from transition data
2. Split state variables into three categories
3. Derive a state abstraction by omitting action-irrelevant state variables



Problem Setup

Goals

1. Learn a causal dynamics model from transition data
2. Split state variables into three categories
3. Derive a state abstraction by omitting action-irrelevant state variables
4. Use the abstracted causal dynamics to learn (many) downstream tasks



Related Work

Bisimulation[1] ϕ: bisimulation considers two states the same $\phi(x) = \phi(x')$ if

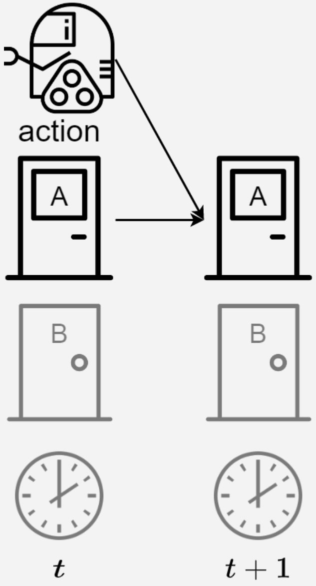
$$R(x, a) = R(x', a),$$
$$\sum_{x'' \in \phi^{-1}(s)} P(x''|x, a) = \sum_{x'' \in \phi^{-1}(s)} P(x''|x', a)$$

[1] Ravindran, B., 2004; Li, L., 2009

Related Work

Compared to CDL,

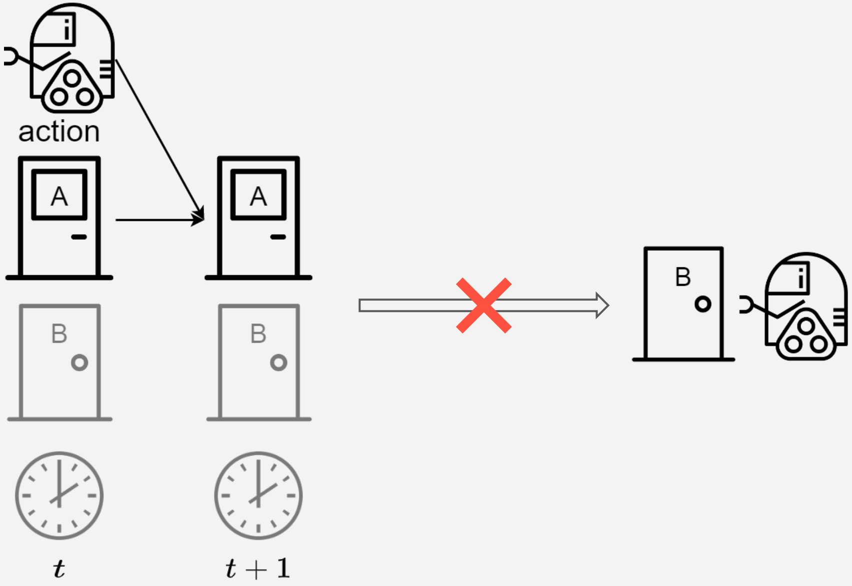
- Bisimulation is reward-specific (applicable to limited tasks).
  e.g., the bisimulation abstraction learned from “opening door A” can’t be used for “opening door B.



Related Work

Compared to CDL,

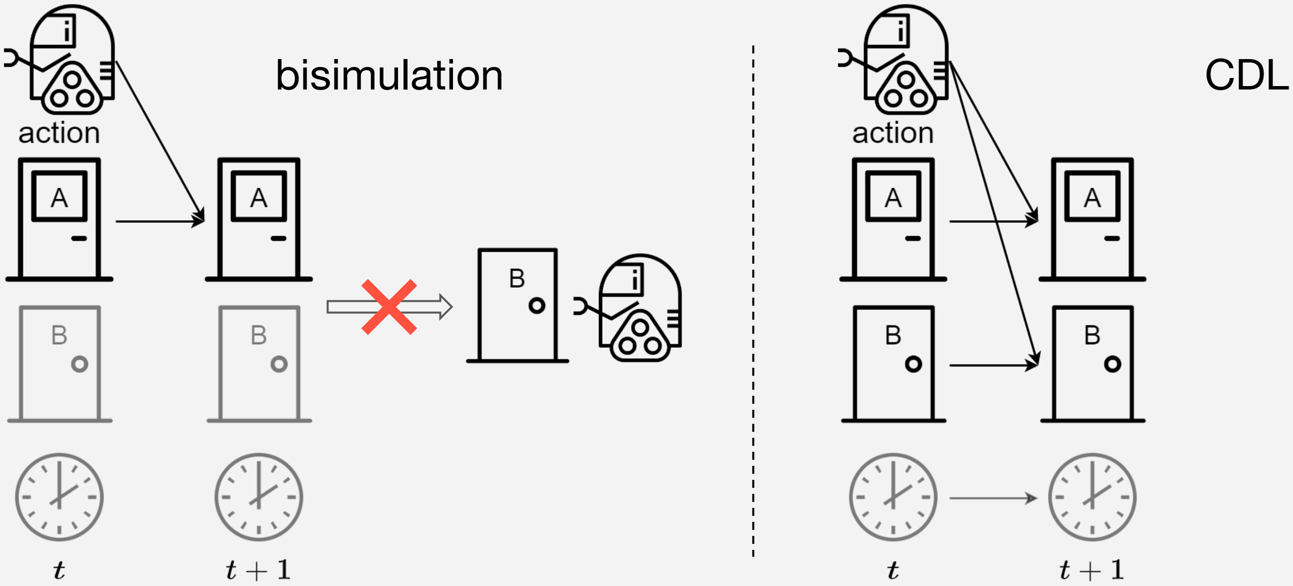
- Bisimulation is reward-specific (applicable to limited tasks).
  e.g., the bisimulation abstraction learned from “opening door A” can’t be used for “opening door B.



Related Work

Compared to CDL,

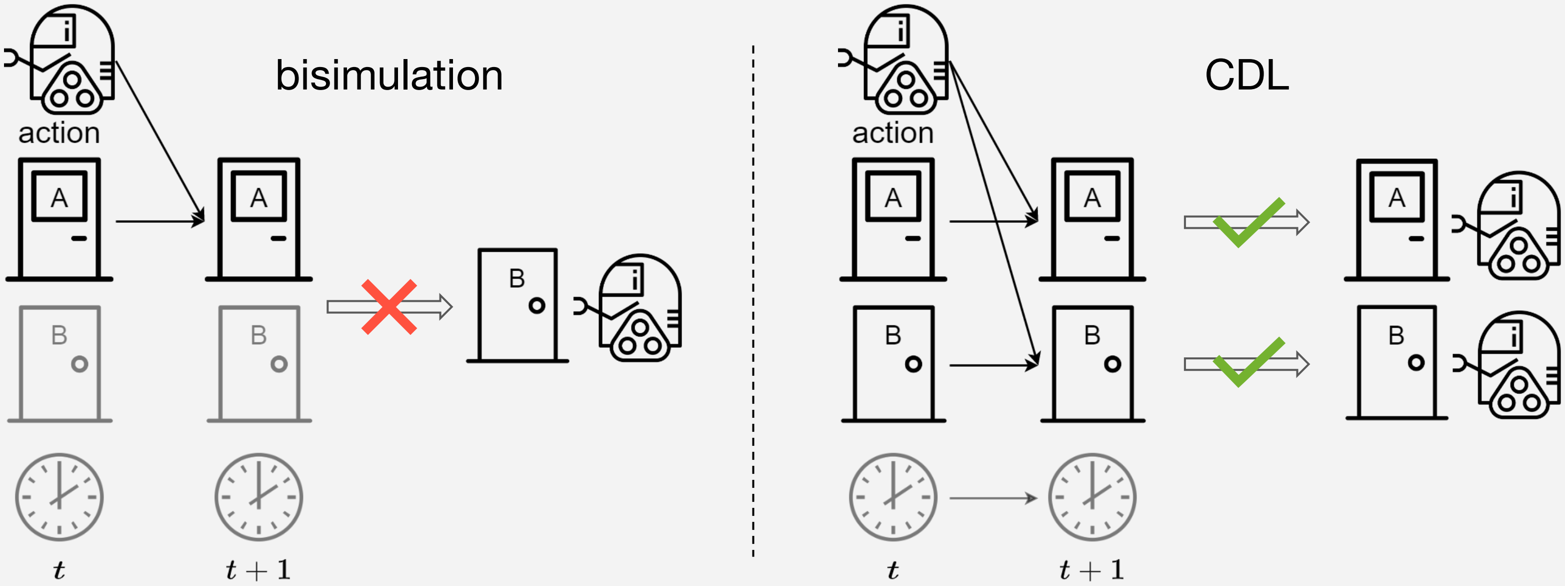
- Bisimulation is reward-specific and thus applicable to **limited** tasks.
  In contrast, CDL's abstraction can be applied to a larger range of tasks.



Related Work

Compared to CDL,

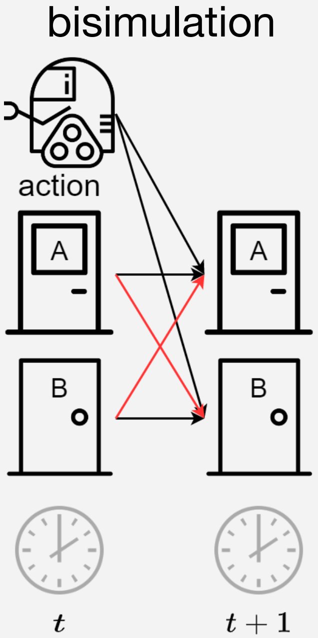
- Bisimulation is reward-specific and thus applicable to **limited** tasks.
  In contrast, CDL's abstraction can be applied to a larger range of tasks.

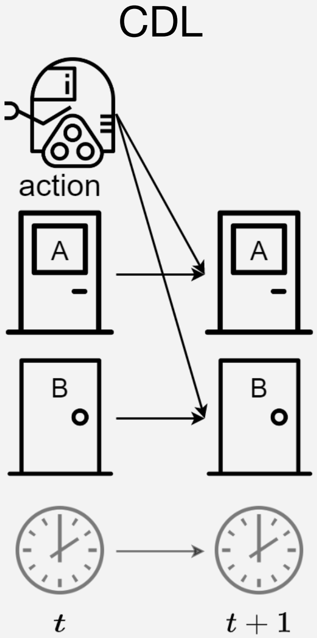


Related Work

Compared to CDL,

- Bisimulation is reward-specific and thus applicable to **limited** tasks.
- Most bisimulation work still uses dense dynamics, leading to poor generalization.

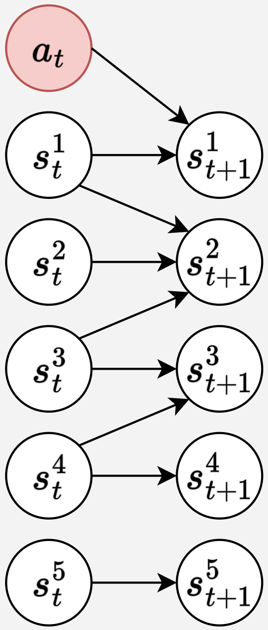




Method

So far, the key of CDL is to learn a causal dynamics model.

$$\mathcal{P}(s_{t+1} | s_t, a_t) = \prod_{i=1}^{d_{\mathcal{S}}} \mathcal{P}(s^i_{t+1} | \mathbf{PA}_{s^i})$$

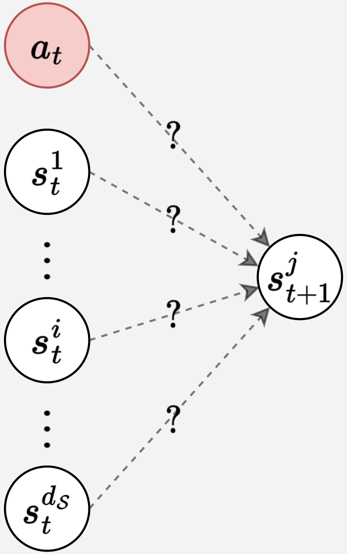


Method

So far, the key of CDL is to learn a causal dynamics model.

$$\mathcal{P}(s_{t+1} | s_t, a_t) = \prod_{i=1}^{d_{\mathcal{S}}} \mathcal{P}(s_{t+1}^i | \mathbf{PA}_{s^i})$$

Specifically, for each state variable s^j, how to determine if a state variable s^i is one of its parents?



Method

Key idea: determine if the causal edge $s_t^i \to s_{t+1}^j$ exists with a conditional independence test.

Skipping assumptions and proofs,

Method

Key idea: determine if the causal edge $s_t^i \to s_{t+1}^j$ exists with a conditional independence test.

Skipping assumptions and proofs,

Theorem 1

$$\text{If } s_t^i \not\perp\!\!\!\perp s_{t+1}^j \,|\, \{s_t / s_t^i, a_t\}, \text{ then } s_t^i \to s_{t+1}^j \,.$$

Method

Key idea: determine if the causal edge $s_t^i \to s_{t+1}^j$ exists with a conditional independence test.

Theorem 1 If $s_t^i \not\perp s_{t+1}^j | \{s_t / s_t^i, a_t\}$, then $s_t^i \to s_{t+1}^j$.

Method

Key idea: determine if the causal edge $s_t^i \rightarrow s_{t+1}^j$ exists with a conditional independence test.

Theorem 1 If $s_t^i \not\perp s_{t+1}^j \,|\{s_t / s_t^i, a_t\}$, then $s_t^i \rightarrow s_{t+1}^j$.

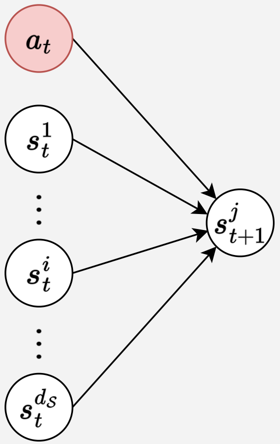
In other words, is s_t^i needed to predict s_{t+1}^j?

Method

Key idea: determine if the causal edge $s_t^i \rightarrow s_{t+1}^j$ exists with a conditional independence test.

Theorem 1 If $s_t^i \not\perp s_{t+1}^j | \{s_t / s_t^i, a_t\}$, then $s_t^i \rightarrow s_{t+1}^j$.

In other words, is s_t^i needed to predict s_{t+1}^j?

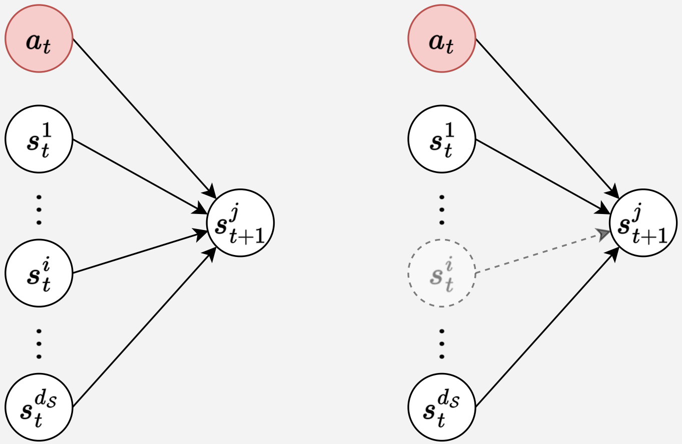


Method

Key idea: determine if the causal edge $s_t^i \to s_{t+1}^j$ exists with a conditional independence test.

Theorem 1 If $s_t^i \not\perp\!\!\!\perp s_{t+1}^j | \{s_t / s_t^i, a_t\}$, then $s_t^i \to s_{t+1}^j$.

In other words, is s_t^i needed to predict s_{t+1}^j?

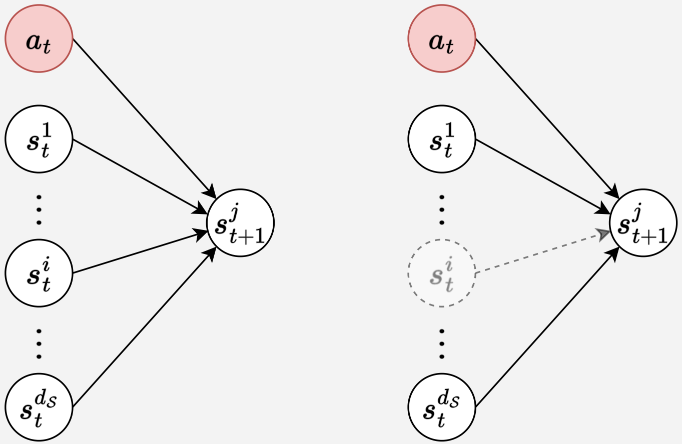


Method

Key idea: determine if the causal edge $s_t^i \to s_{t+1}^j$ exists with a conditional independence test.

Theorem 1 If $s_t^i \not\perp\!\!\!\perp s_{t+1}^j | \{s_t / s_t^i, a_t\}$, then $s_t^i \to s_{t+1}^j$.

In other words, is s_t^i needed to predict s_{t+1}^j?



$$p(s_{t+1}^j | s_t, a_t) \stackrel{?}{=} p(s_{t+1}^j | \{s_t / s_t^i, a_t\})$$

Method

Learn and predict $p(s_{t+1}^j | s_t, a_t)$ & $p(s_{t+1}^j | \{s/s^i\}_t, a_t)$ using generative models, but there will be d_S^2 models to train...

Method

Learning $p(s_{t+1}^{j}|s_t, a_t)$ & $p(s_{t+1}^{j}|\{s/s^i\}_t, a_t)$ needs to train $d_{\mathcal{S}}^2$ models.
With a mask M_j and an element-wise maximum module, one network can represent all generative models in the form of $p(s_{t+1}^{j}|\cdot)$.

Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train $d^2_{\mathcal{S}}$ models.

With M_j and an element-wise maximum module, one network can represent all models.

For example, to represent $p(s^j_{t+1}|s_t, a_t)$,

inputs

a_t

1

s^1_t

2

$\vdots$

s^i_t

0

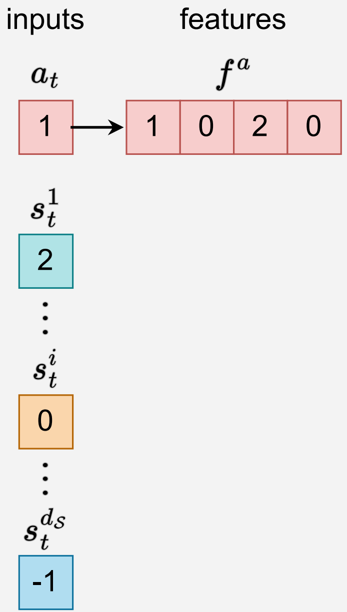
$\vdots$

$s^{d_{\mathcal{S}}}_t$

-1

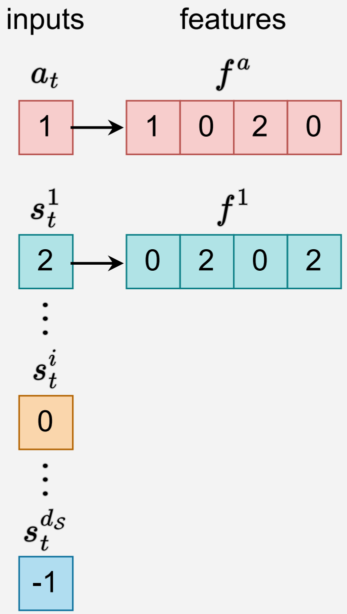
Method

Learning $p(s_{t+1}^j|s_t, a_t)$ & $p(s_{t+1}^j|\{s/s^i\}_t, a_t)$ needs to train d_S^2 models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s_{t+1}^j|s_t, a_t)$,



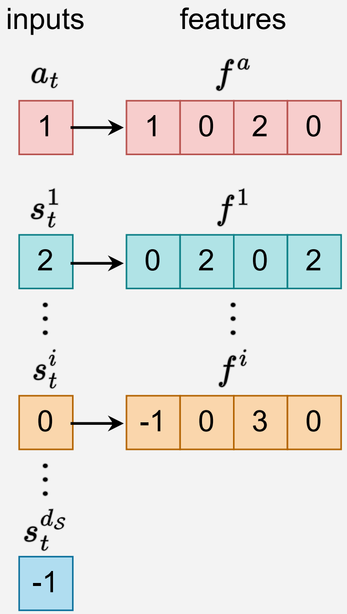
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train $d^2_{\mathcal{S}}$ models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|s_t, a_t)$,



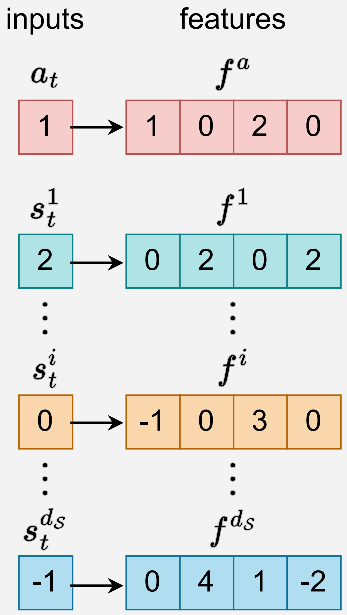
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d^2_S models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|s_t, a_t)$,



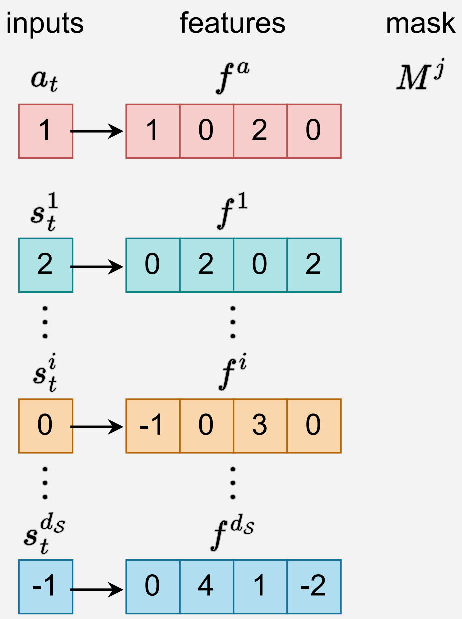
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d^2_S models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|s_t, a_t)$,



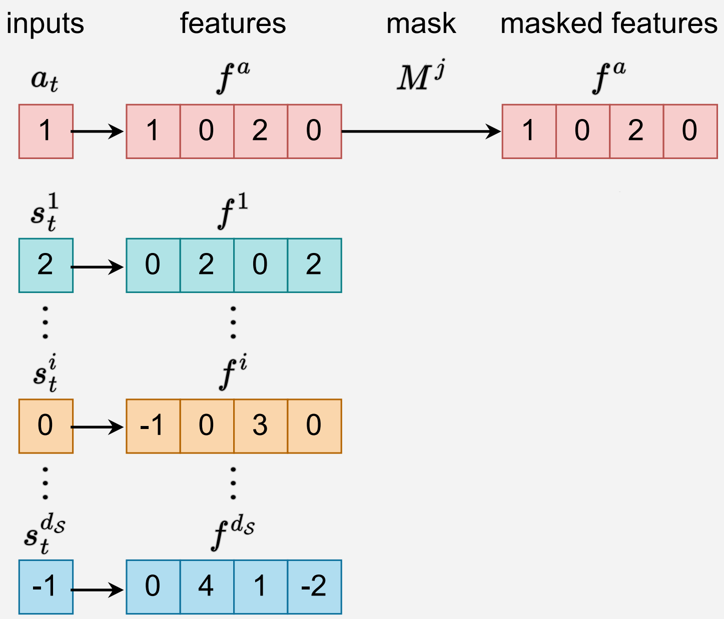
Method

Learning $p(s_{t+1}^j | s_t, a_t)$ & $p(s_{t+1}^j | \{s/s^i\}_t, a_t)$ needs to train d_S^2 models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s_{t+1}^j | s_t, a_t)$,



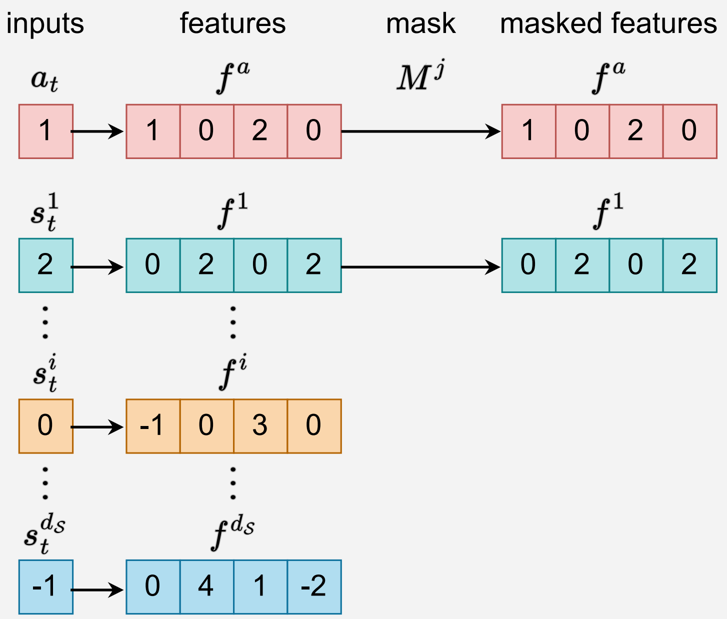
Method

Learning $p(s_{t+1}^j | s_t, a_t)$ & $p(s_{t+1}^j | \{s/s^i\}_t, a_t)$ needs to train d_S^2 models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s_{t+1}^j | s_t, a_t)$,



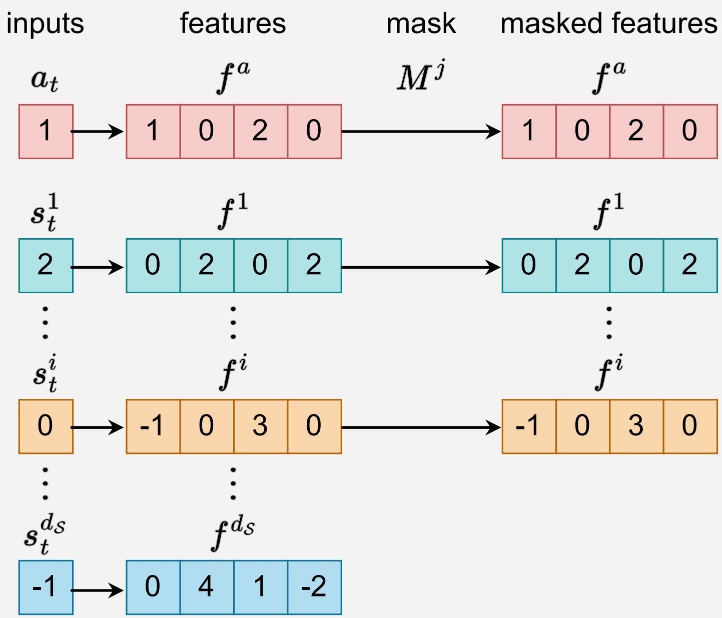
Method

Learning $p(s_{t+1}^j | s_t, a_t)$ & $p(s_{t+1}^j | \{s/s^i\}_t, a_t)$ needs to train d_S^2 models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s_{t+1}^j | s_t, a_t)$,



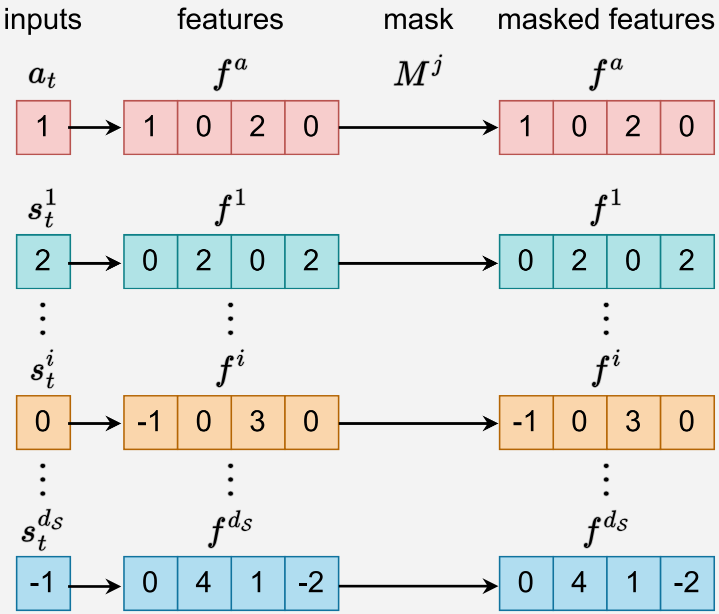
Method

Learning $p(s_{t+1}^{j}|s_t, a_t)$ & $p(s_{t+1}^{j}|\{s/s^i\}_t, a_t)$ needs to train d_S^2 models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s_{t+1}^{j}|s_t, a_t)$,



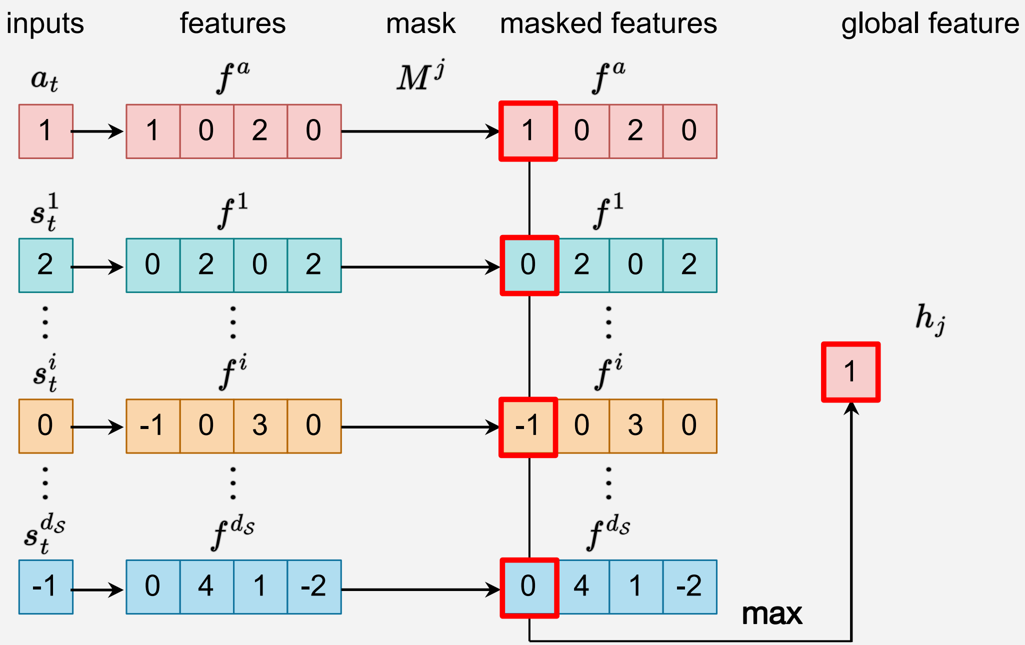
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d^2_S models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|s_t, a_t)$,



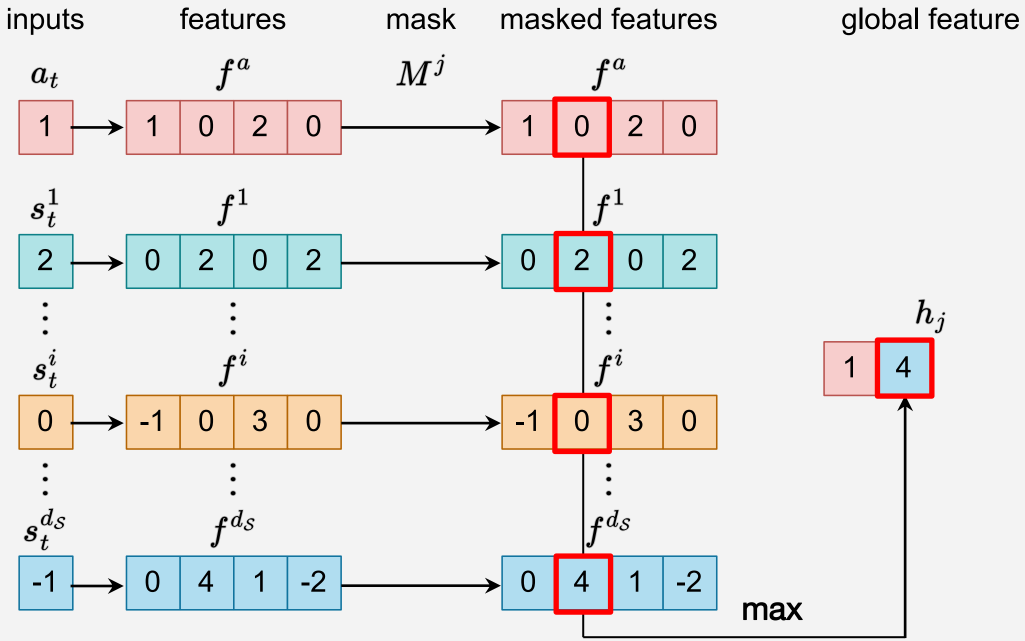
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d^2_S models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|s_t, a_t)$,



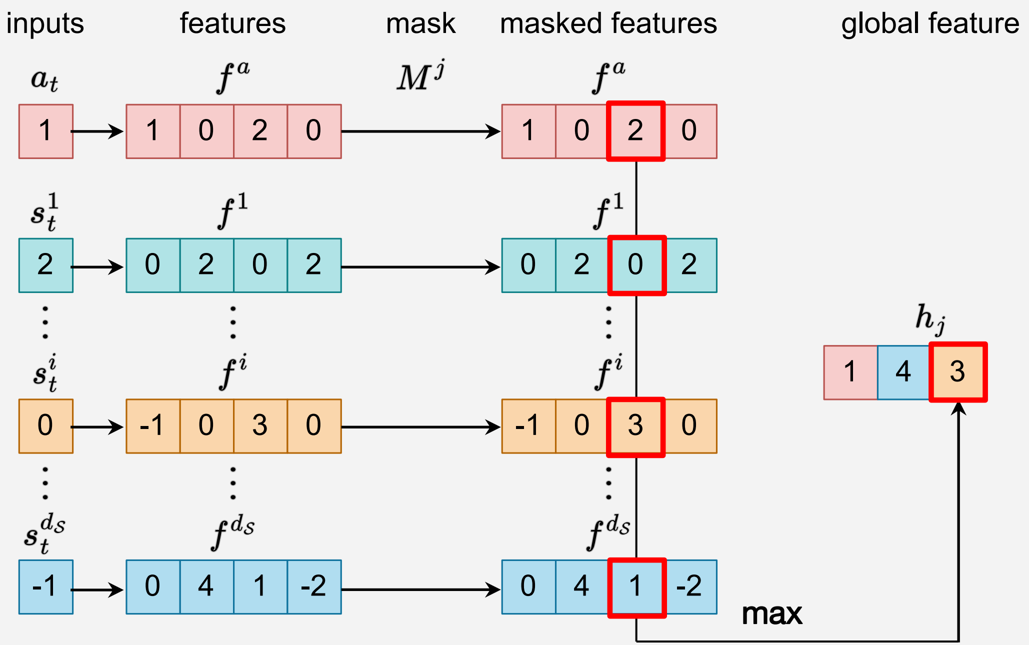
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d_S^2 models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|s_t, a_t)$,



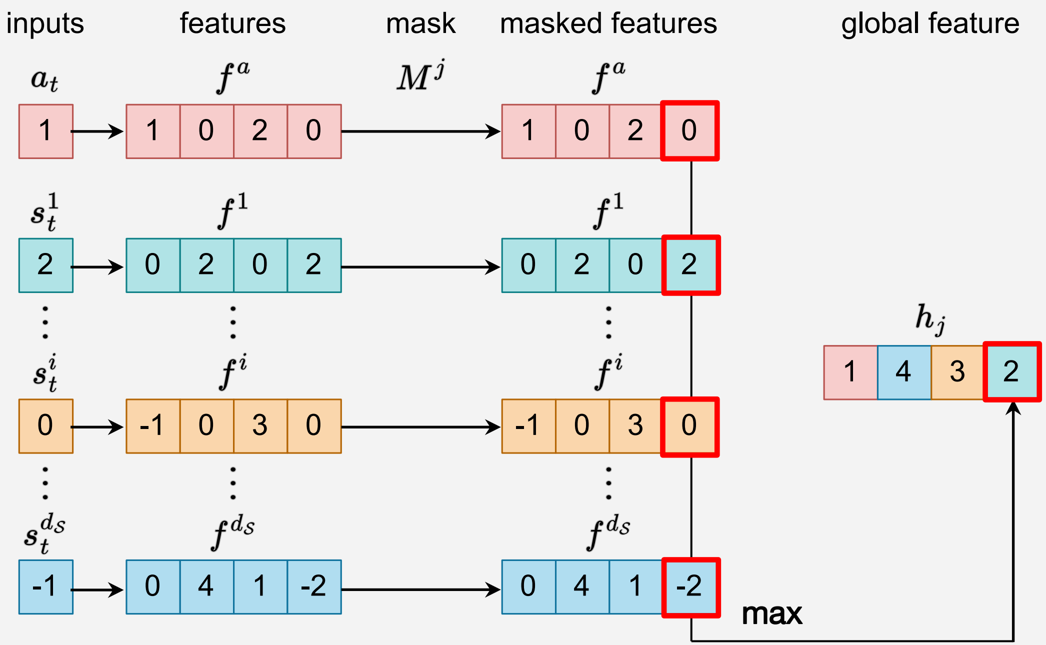
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d_S^2 models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|s_t, a_t)$,



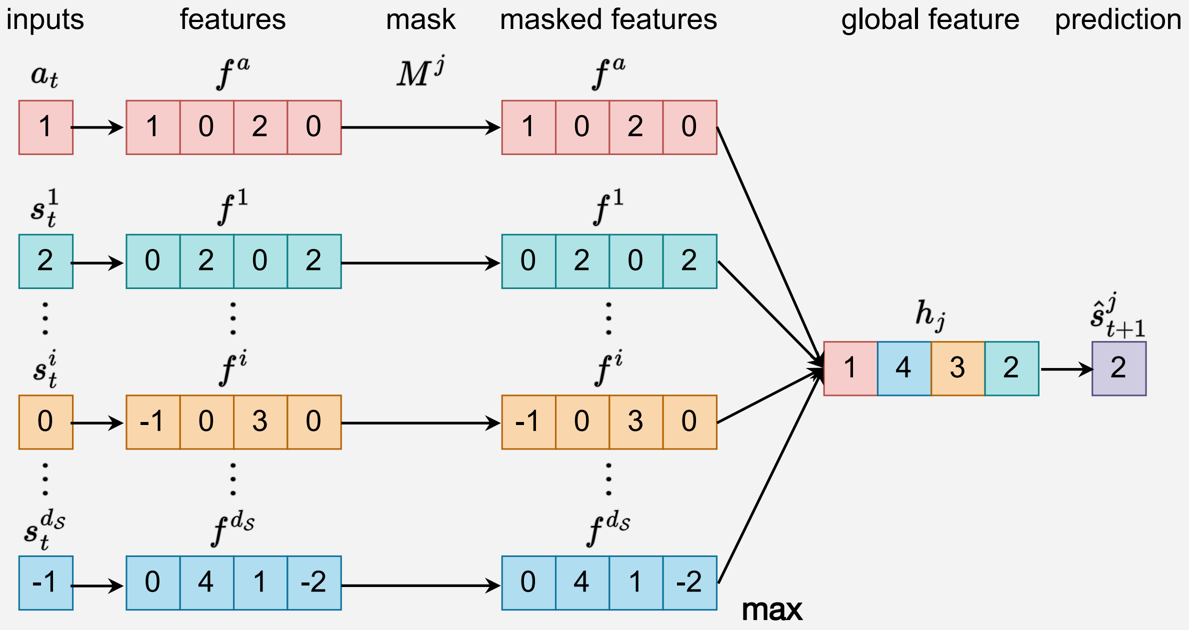
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d^2_S models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|s_t, a_t)$,



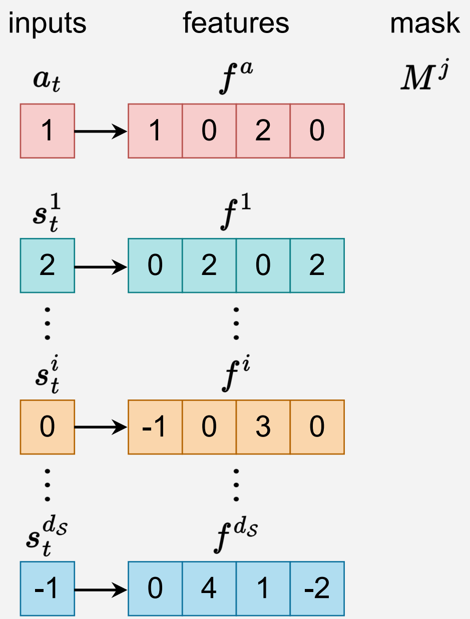
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d^2_S models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|s_t, a_t)$,



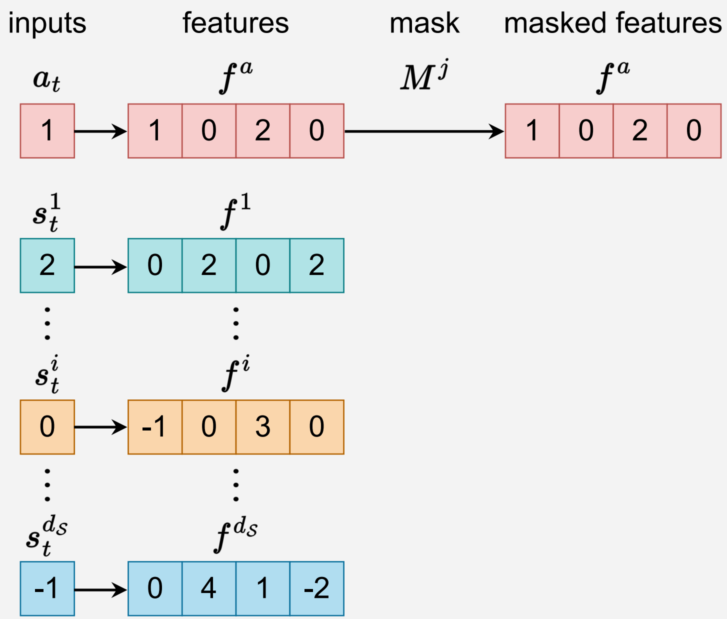
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d^2_S models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



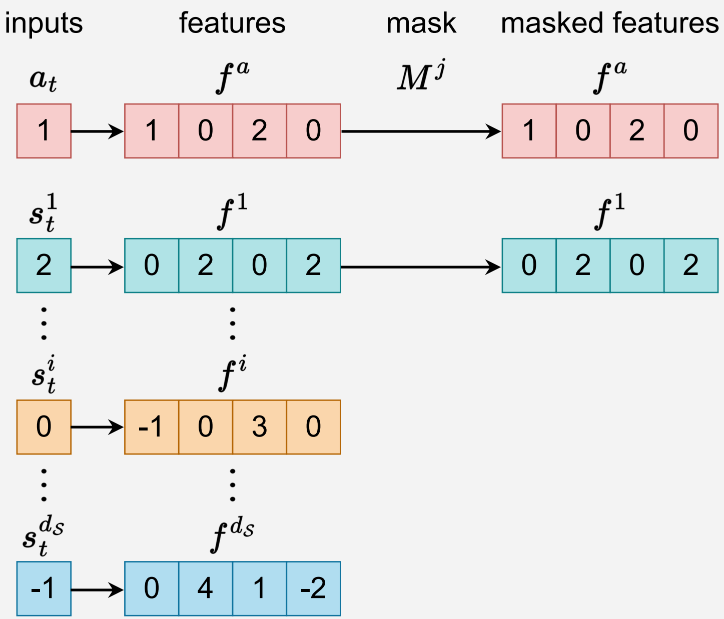
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d^2_S models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



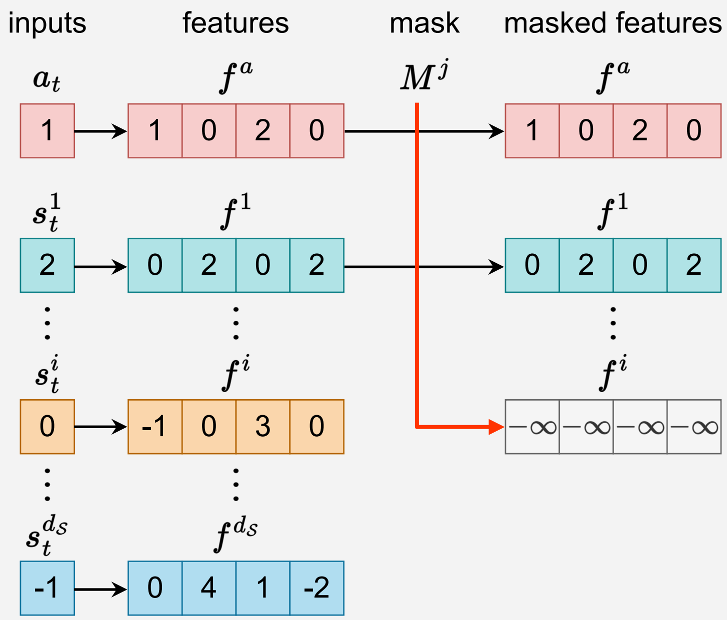
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d_S^2 models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



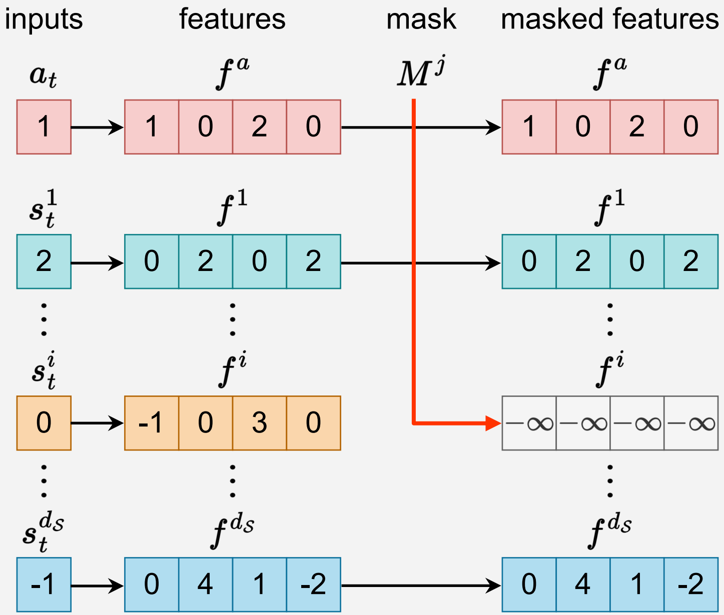
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d_S^2 models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



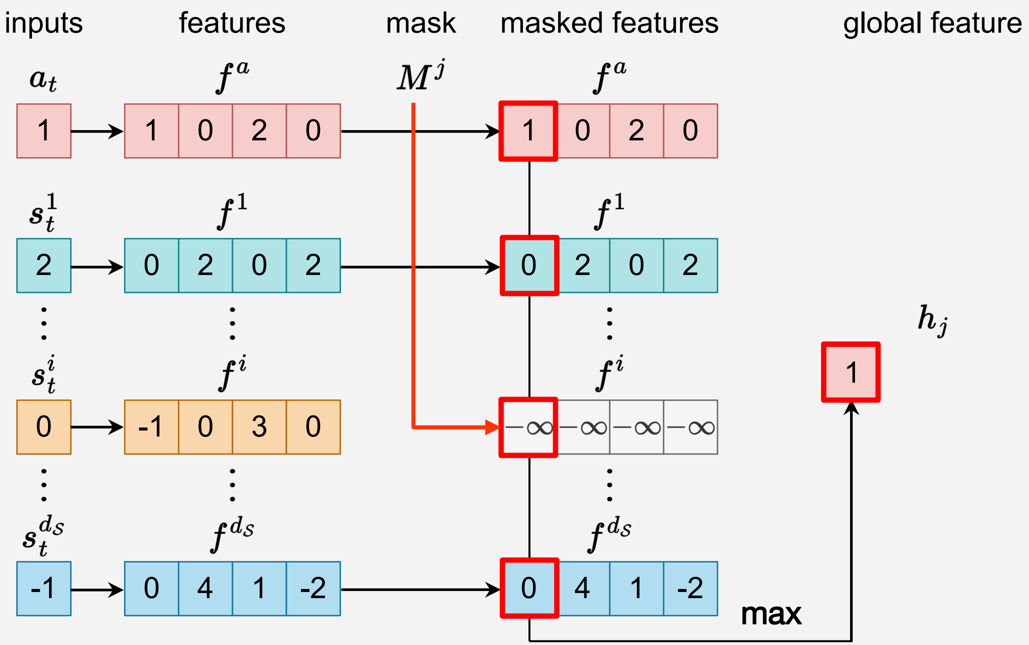
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d^2_S models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



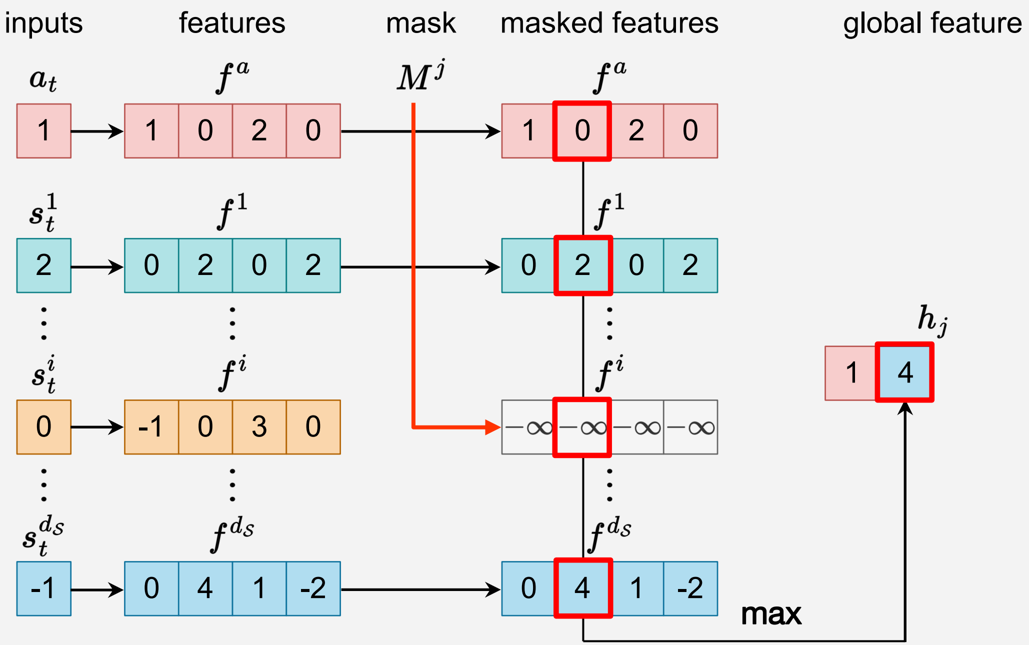
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d_S^2 models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



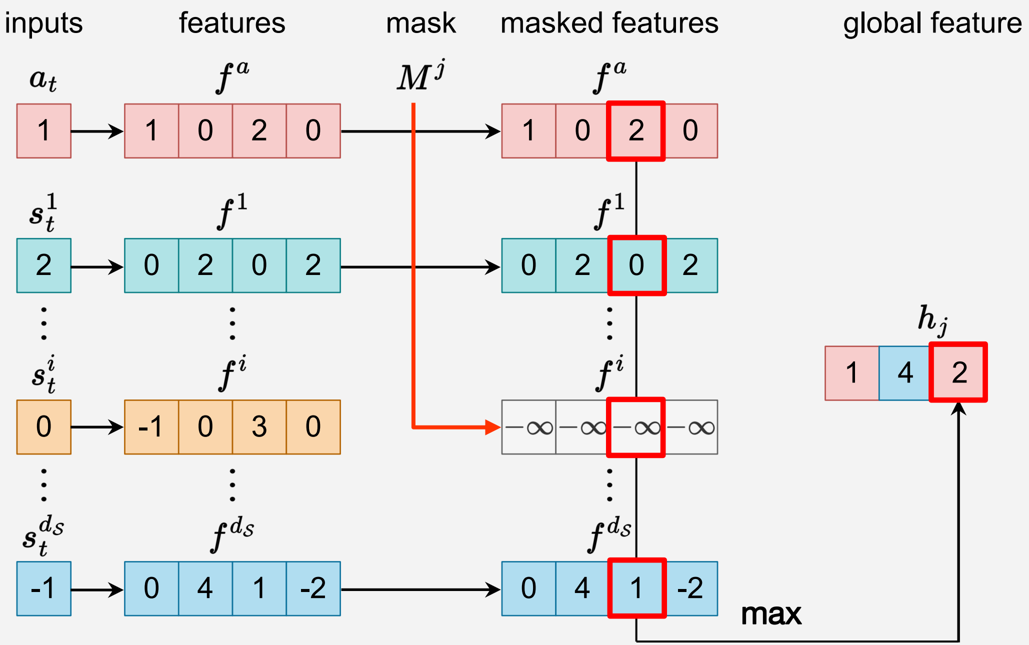
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d_S^2 models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



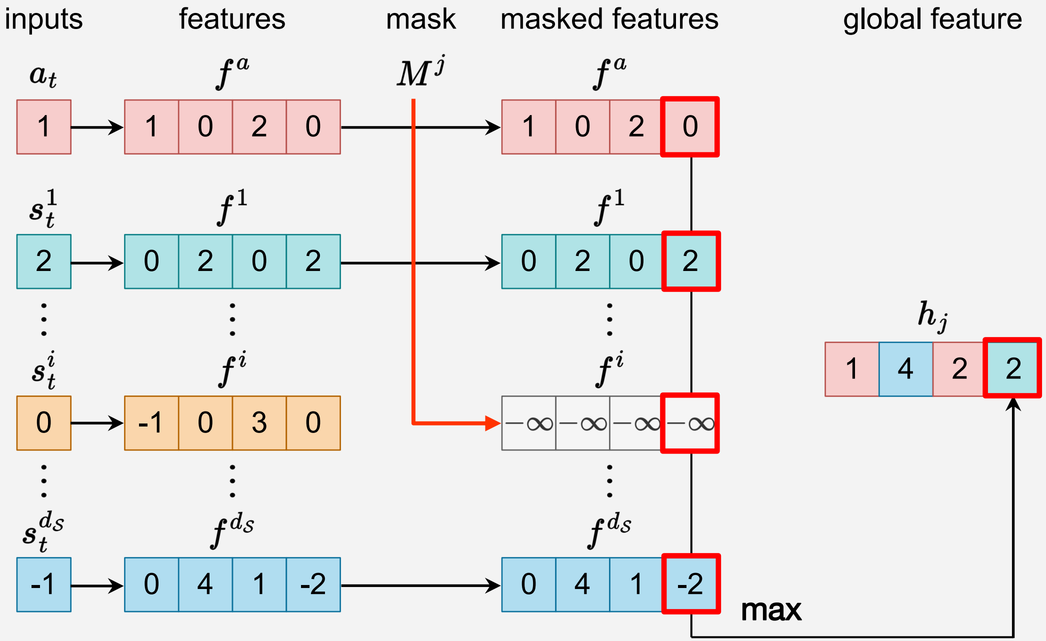
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d^2_S models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



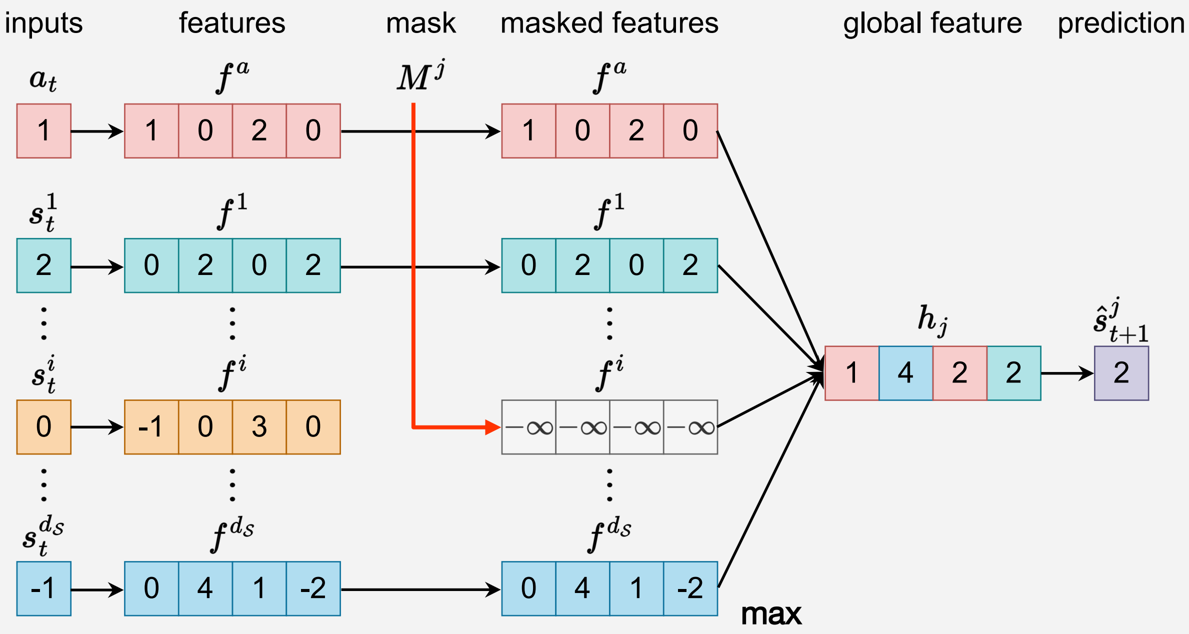
Method

Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d^2_S models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



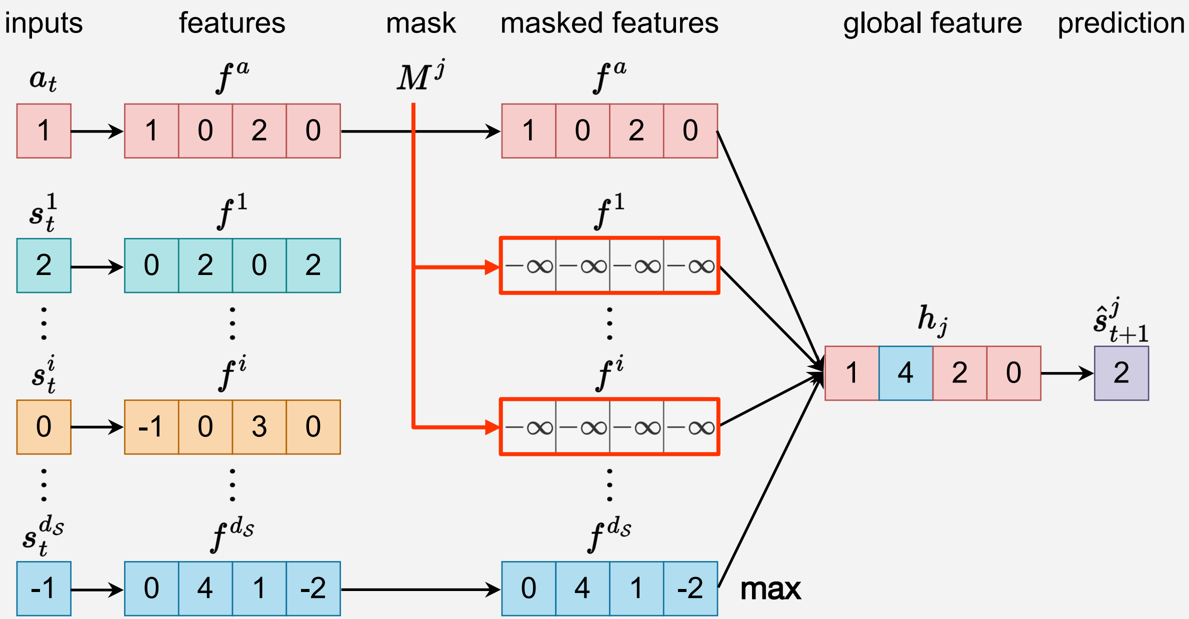
Method

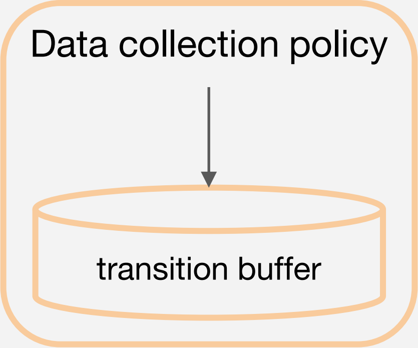
Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d^2_S models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



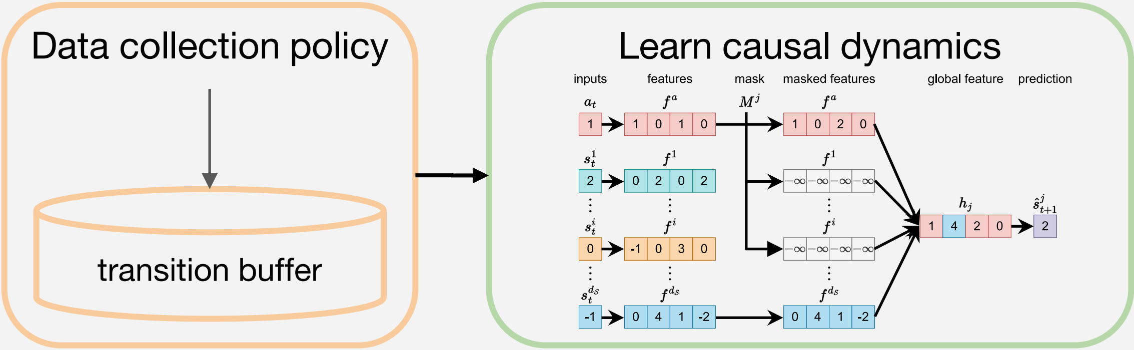
Method

After training, to represent the causal model $p(s_{t+1}^j | \mathbf{PA}_t^j)$, we can adjust the mask to select causal parents of s^j only.

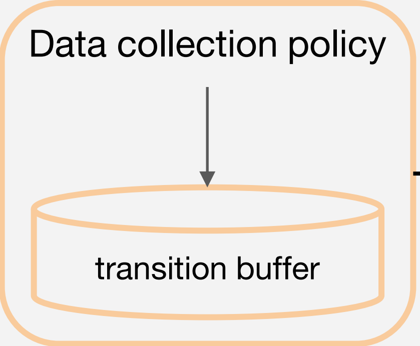


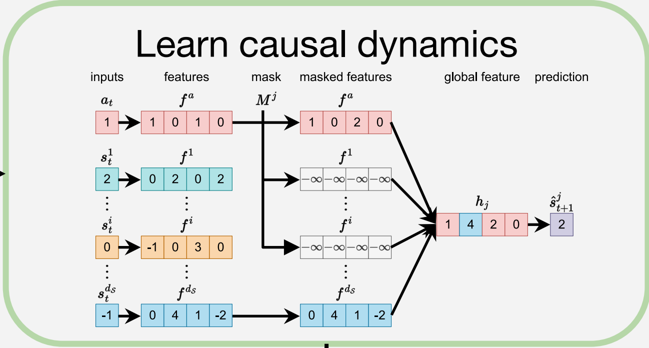


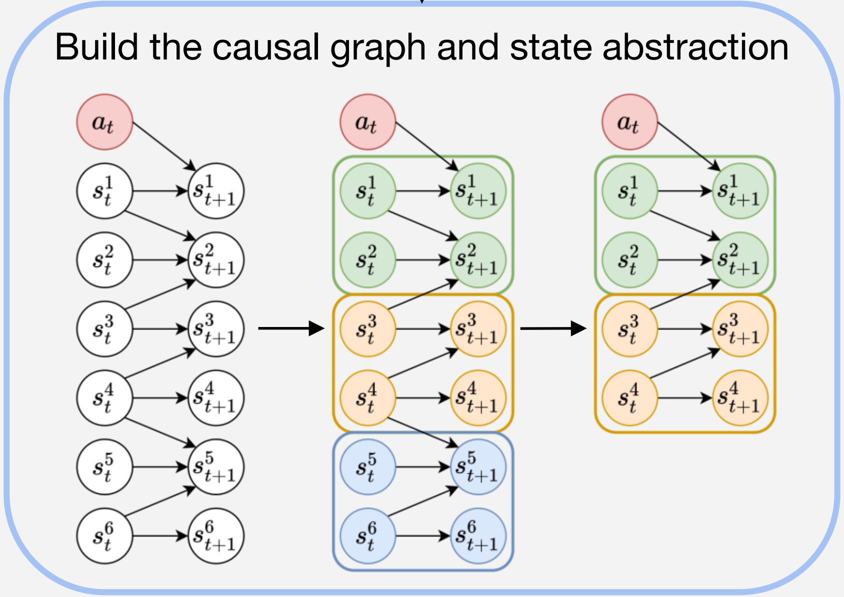
Causal Dynamics Learning (CDL)



Causal Dynamics Learning (CDL)

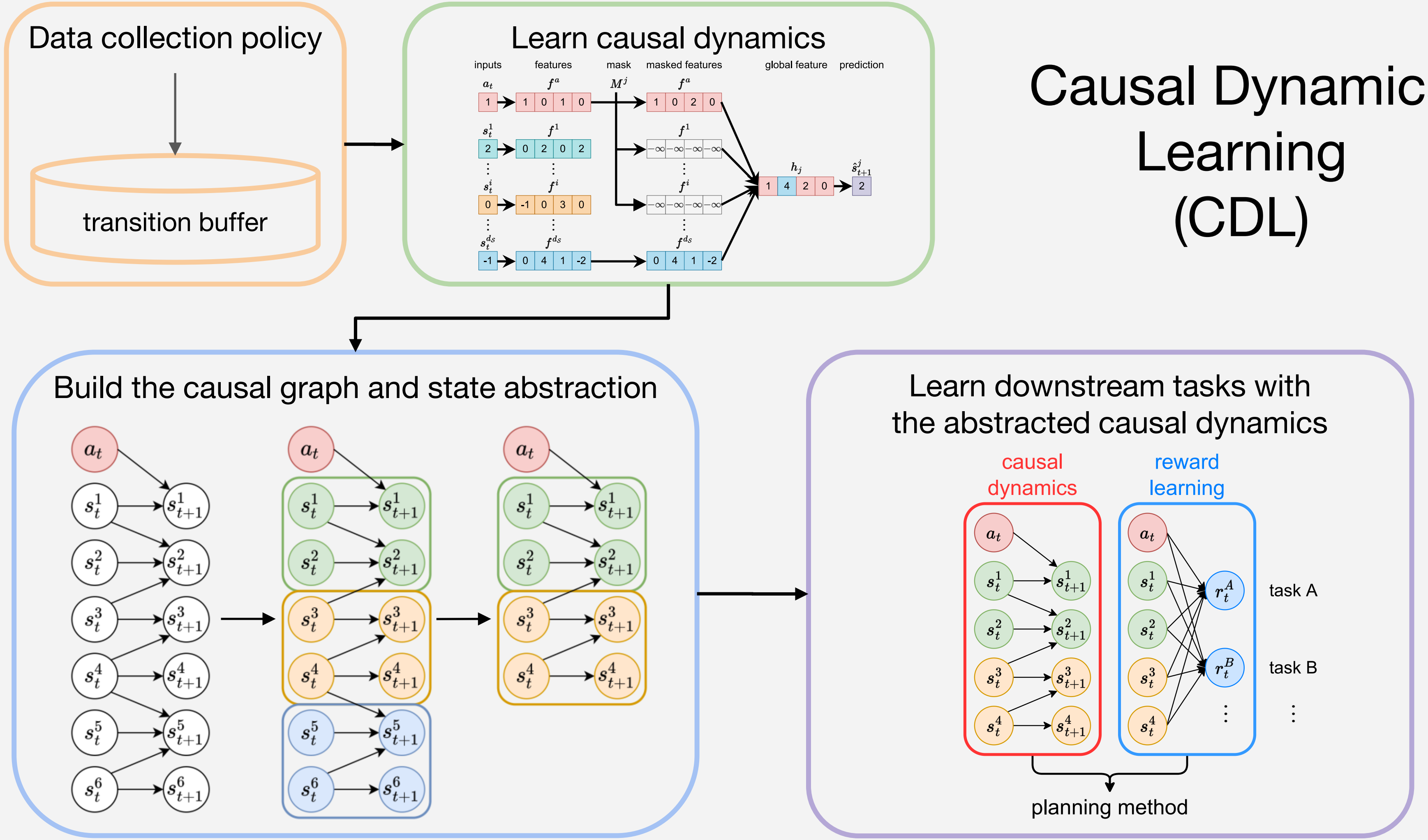






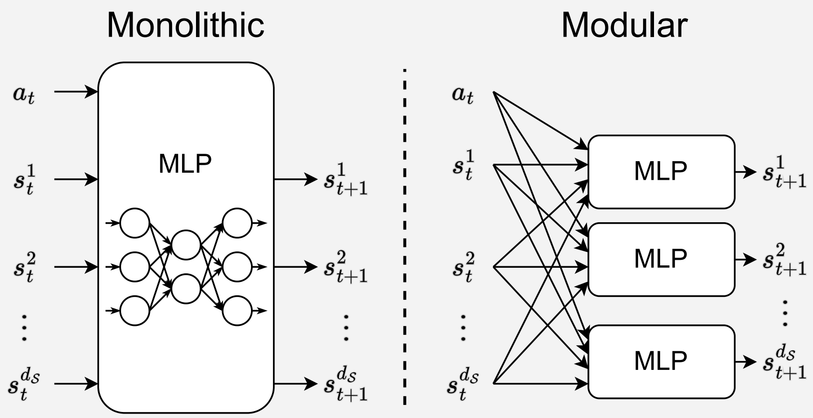
Causal Dynamics Learning (CDL)

Causal Dynamics Learning (CDL)



Experiments

Baselines

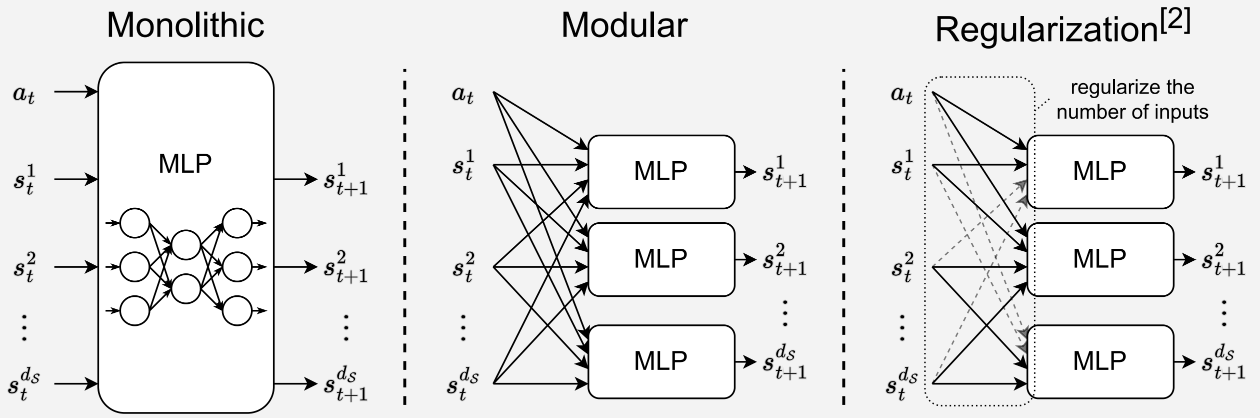


MLP: multi-layer perceptron

[2] Wang et al., Neurips 2021. [3] Kipf et al., ICLR 2020

Experiments

Baselines

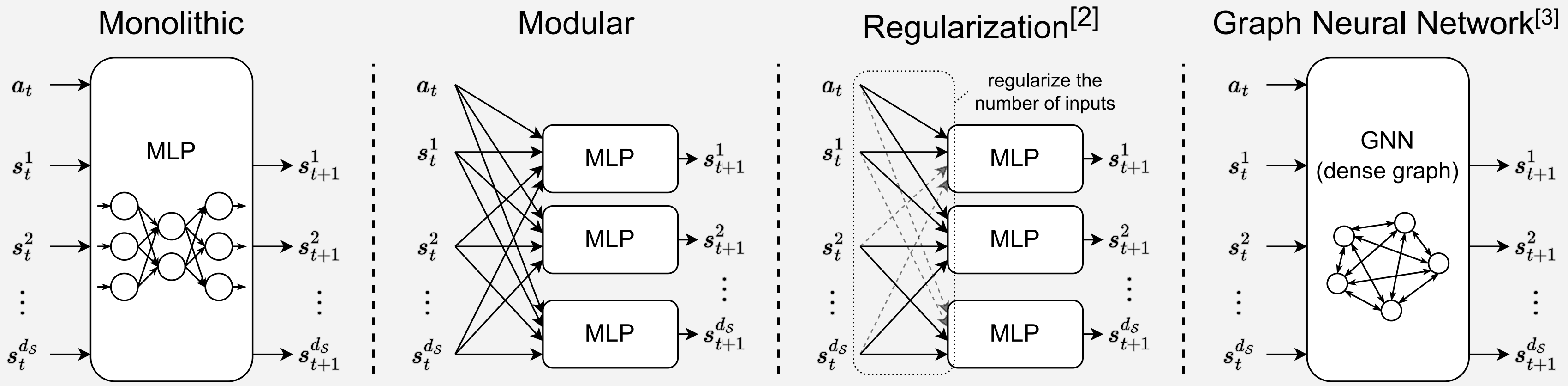


MLP: multi-layer perceptron

[2] Wang et al., Neurips 2021. [3] Kipf et al., ICLR 2020

Experiments

Baselines

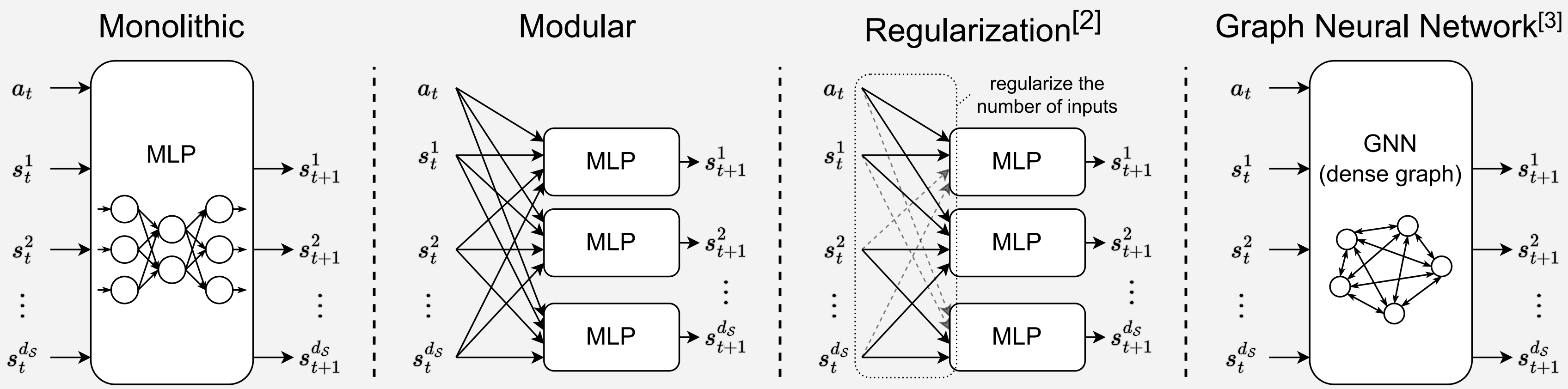


MLP: multi-layer perceptron

[2] Wang et al., Neurips 2021. [3] Kipf et al., ICLR 2020

Experiments

Baselines



MLP: multi-layer perceptron

Does each baseline learn a causal model?

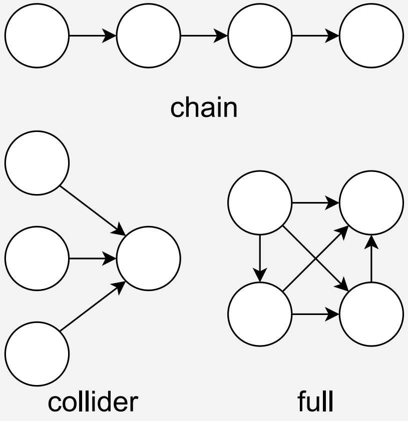
✗ ✗ ✓ ✗

[2] Wang et al., Neurips 2021. [3] Kipf et al., ICLR 2020

Experiments

Chemical Environment[4]

Synthesized environment
– with different underlying graphs

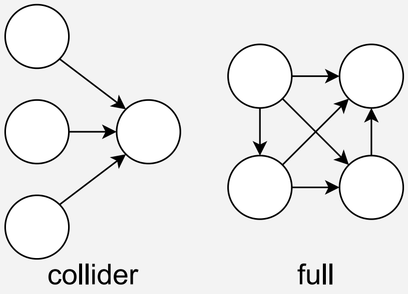


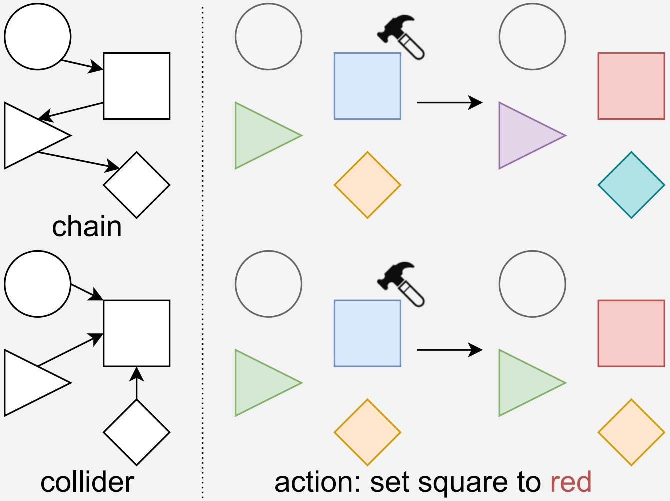
[4] Ke et al., Neurips 2021.

Experiments

Chemical Environment[4]

Synthesized environment
- with different underlying graphs
- as action changes the color of one node, colors of all its descendants will also change.





[4] Ke et al., Neurips 2021.

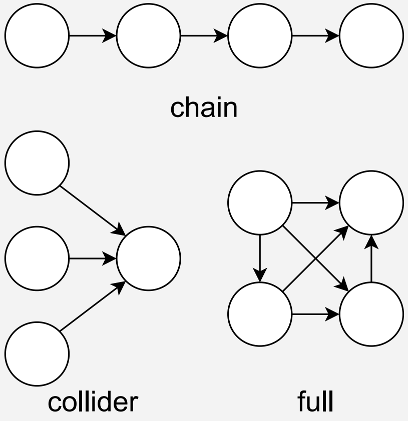
Experiments

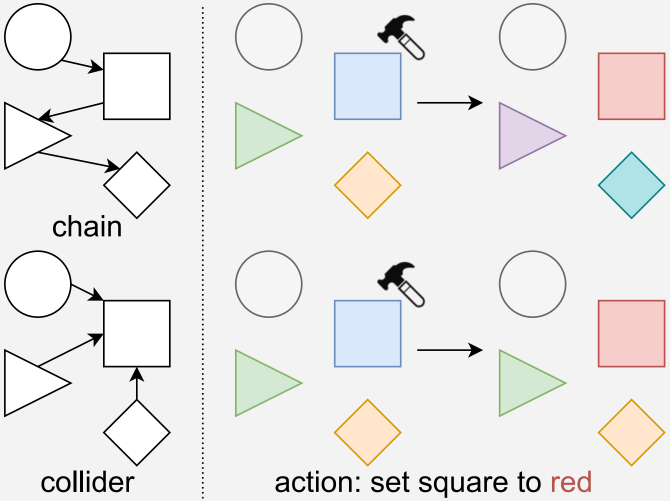
Chemical Environment[4]

Synthesized environment

- with different underlying graphs
- as action changes the color of one node, colors of all its descendants will also change.

Action-irrelevant variables: positions sampled from $N(0, 0.01)$.



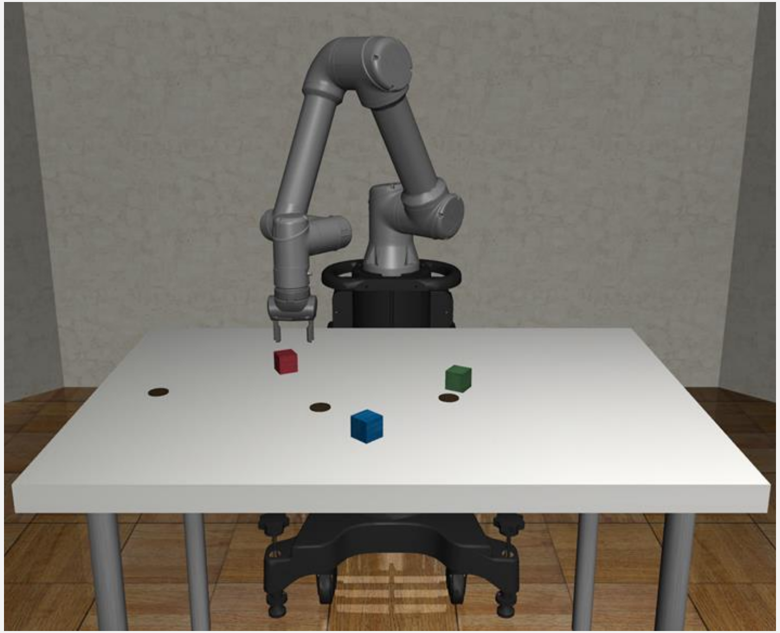


[4] Ke et al., Neurips 2021.

Experiments

Manipulation Environment

State Variables:



Experiments

Manipulation Environment

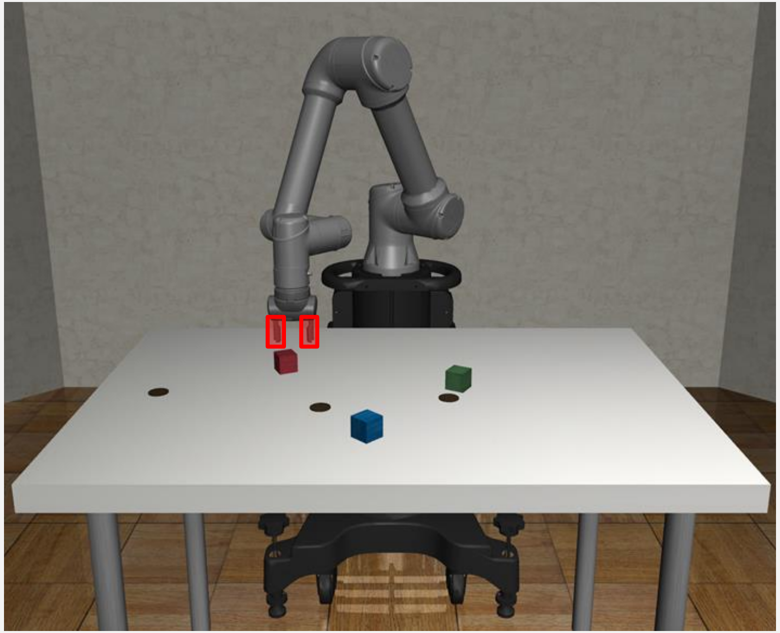
State Variables:

- end-effector (eef)

Experiments

Manipulation Environment

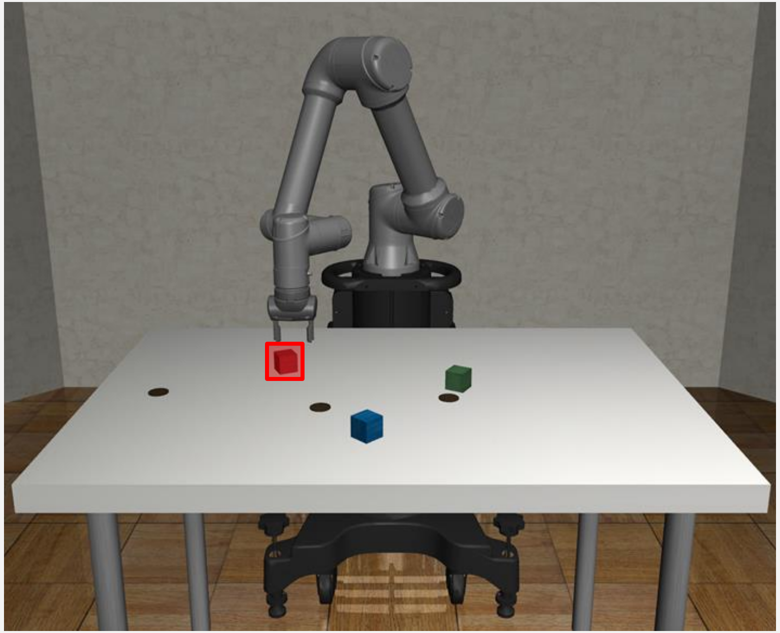
State Variables:
- end-effector (eef)
- gripper (grp)



Experiments

Manipulation Environment

State Variables:
- end-effector (eef)
- gripper (grp)
- the movable object (mov)

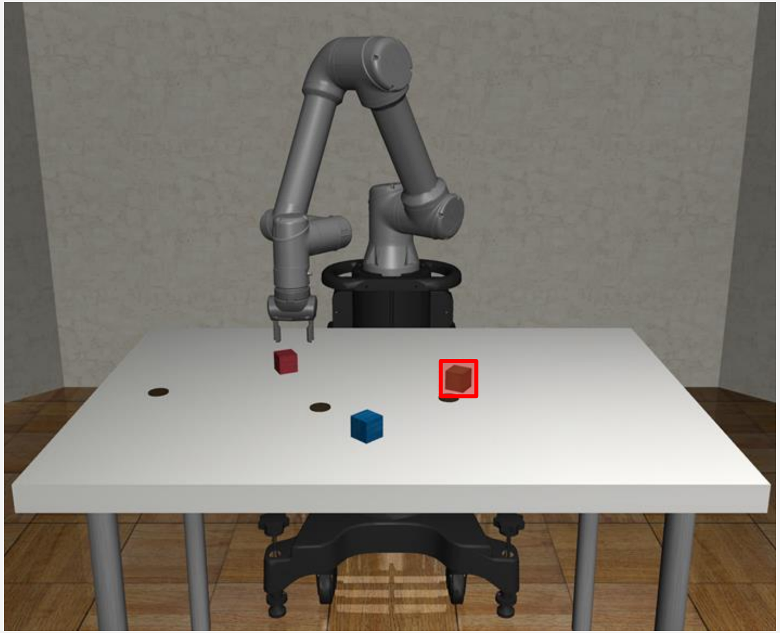


Experiments

Manipulation Environment

State Variables:

- end-effector (eef)
- gripper (grp)
- the movable object (mov)
- the unmovable object (unm)

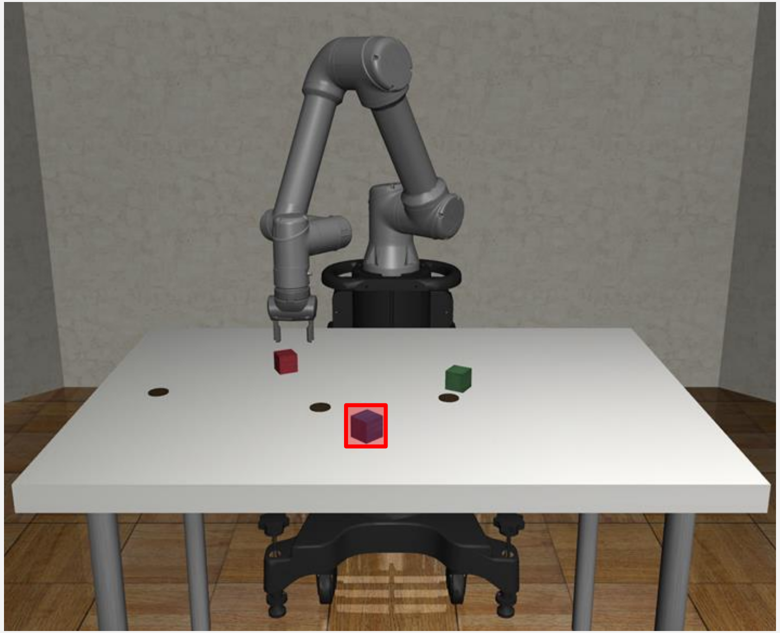


Experiments

Manipulation Environment

State Variables:

- end-effector (eef)
- gripper (grp)
- the movable object (mov)
- the unmovable object (unm)
- the randomly moving object (rand)

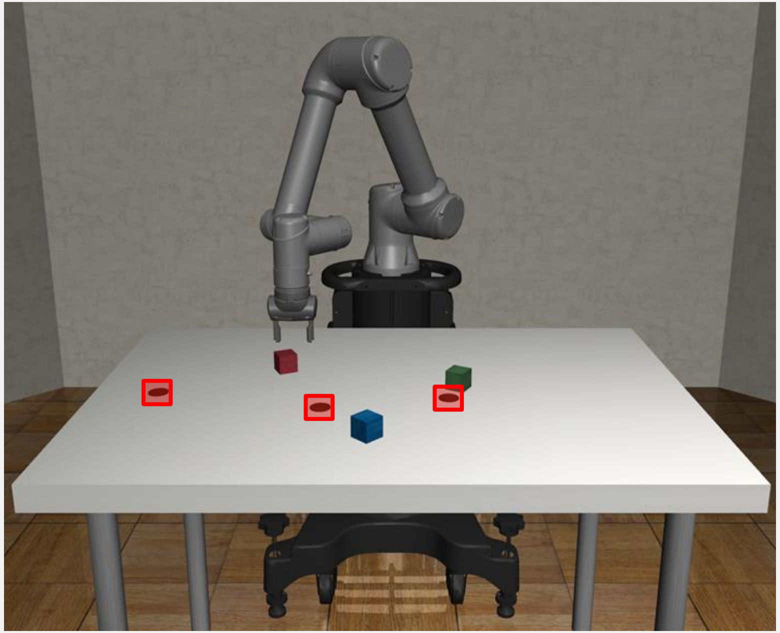


Experiments

Manipulation Environment

State Variables:

- end-effector (eef)
- gripper (grp)
- the movable object (mov)
- the unmovable object (unm)
- the randomly moving object (rand)
- non-interactable markers (mkr$^{1\text{-}3}$)



Experiments

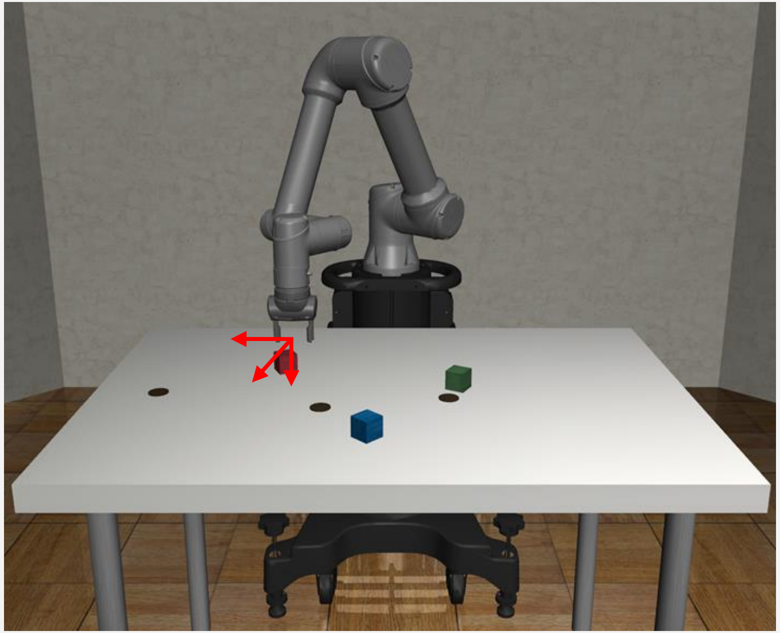
Manipulation Environment

State Variables:

- end-effector (eef)
- gripper (grp)
- the movable object (mov)
- the unmovable object (unm)
- the randomly moving object (rand)
- non-interactable markers ($mkr^{1\text{-}3}$)

Action dimensions:

- end-effector target



Experiments

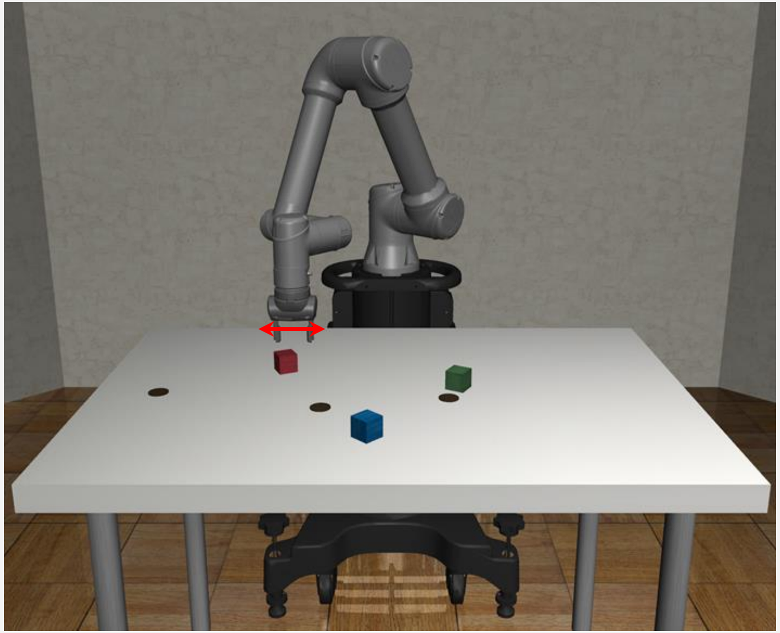
Manipulation Environment

State Variables:

- end-effector (eef)
- gripper (grp)
- the movable object (mov)
- the unmovable object (unm)
- the randomly moving object (rand)
- non-interactable markers ($mkr^{1\text{-}3}$)

Action dimensions:

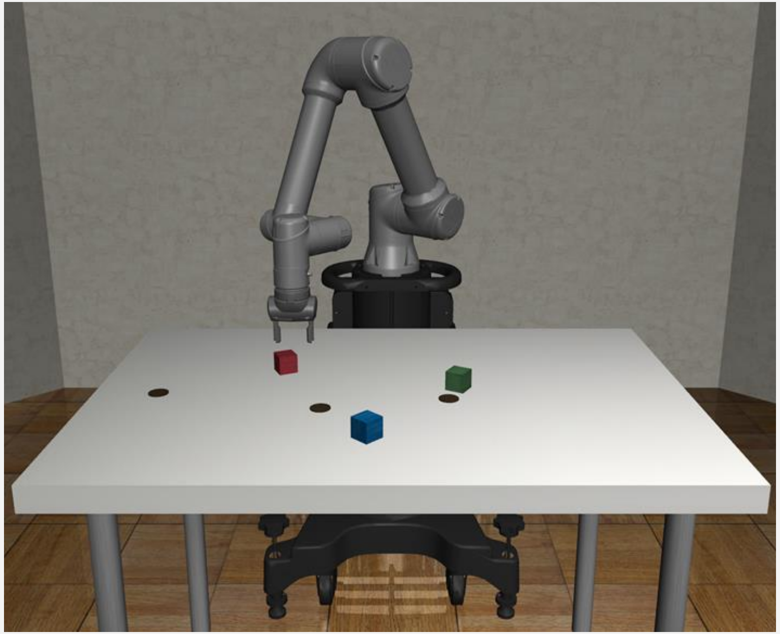
- end-effector target
- gripper open/close

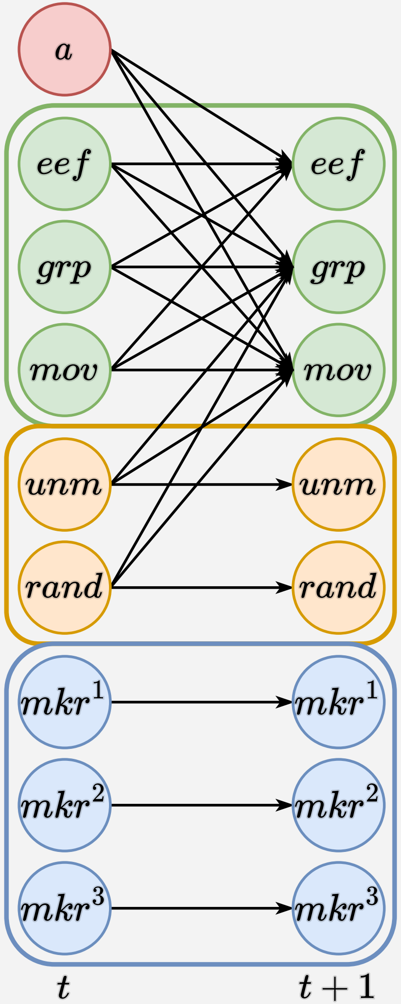


Results

Causal Graph Accuracy

At the object level, the learned dependence is (subjectively) reasonable.





Results

Causal Graph Accuracy

Table 1. Causal Graph Accuracy (in %) for CDL and Reg

Environment	CDL (Ours)	**Reg**
Chemical (Collider)	**100.0** $\pm$ 0.0	99.4 $\pm$ 0.4
Chemical (Chain)	**100.0** $\pm$ 0.1	99.7 $\pm$ 0.1
Chemical (Full)	**99.1** $\pm$ 0.1	97.7 $\pm$ 0.4
Manipulation	**90.2** $\pm$ 0.3	84.4 $\pm$ 0.5

Results

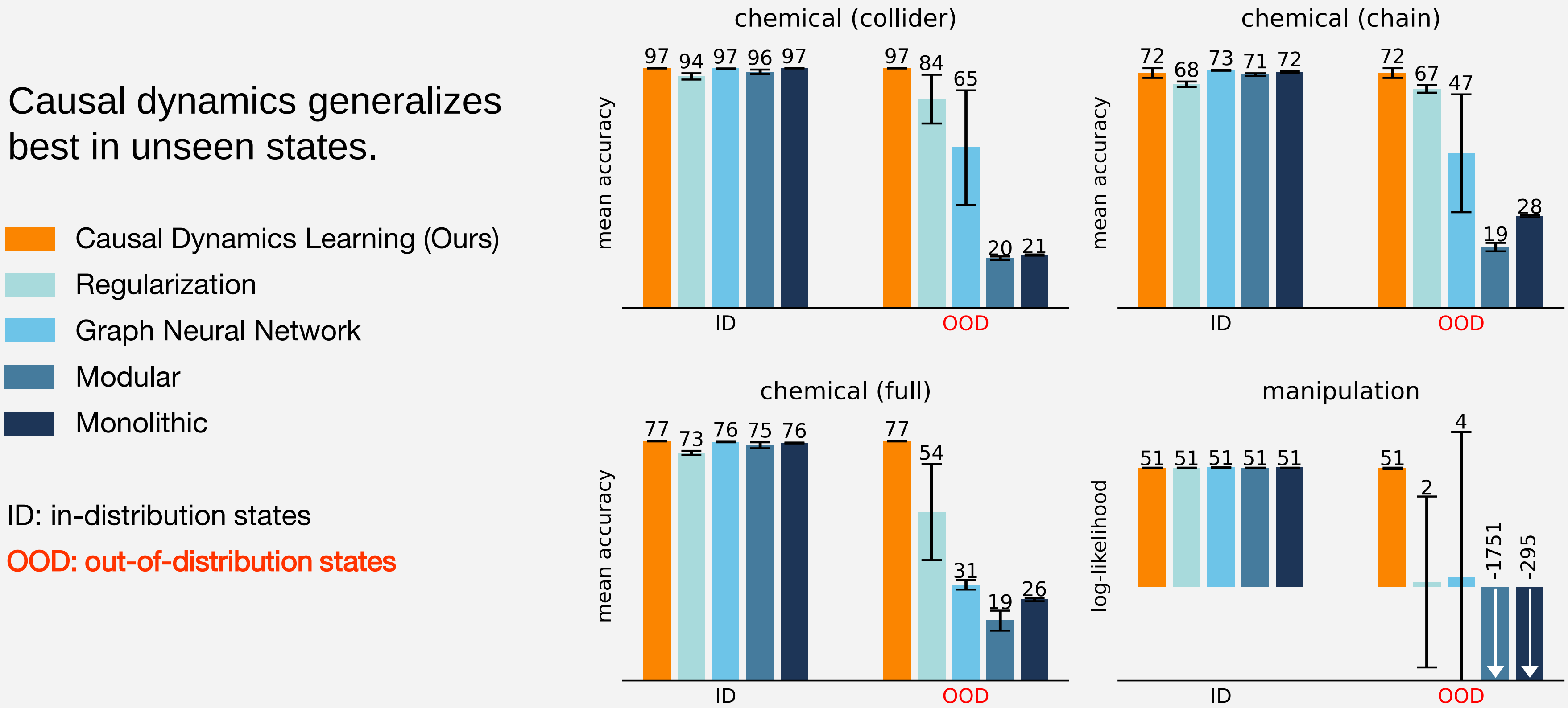
Dynamics Generalization

Causal dynamics generalizes best in unseen states.

Results

Dynamics Generalization

Causal dynamics generalizes best in unseen states.



ID: in-distribution states

OOD: out-of-distribution states

Limitations and Future Directions

Scale to high-dimensional observations (e.g. images)?

- Learn disentangled representations, then learn dynamics in the representation space

Causal dependencies are learned globally only.

- Learning local independencies to further sparsify the dynamics.

Causal Dynamics Learning for Task-Independent State Abstraction

Zizhao Wang, Xuesu Xiao, Zifan Xu, Yuke Zhu, and Peter Stone





Contact Information:
Zizhao Wang: zizhao.wang@utexas.edu

Link to the Paper: https://arxiv.org/pdf/2206.13452.pdf

Scan to read the paper

LARG Learning Agents Research Group The University of Texas at Austin | RPL UT Robot Perception & Learning Lab | Sony AI

State Abstraction Discovery for Generalization

- Learn which state variables to ignore (and when)
  - based on policy irrelevance (IJCAI 2005)
  - based on causal dynamics (ICML 2022)

Causal Dynamics Learning for Task-Independent State Abstractions

Peter Stone

Learning Agents Research Group (LARG)
Department of Computer Science
The University of Texas at Austin

(also Executive Director of Sony AI America)