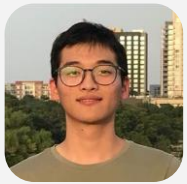
Causal Dynamics Learning for Task-Independent State Abstraction

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





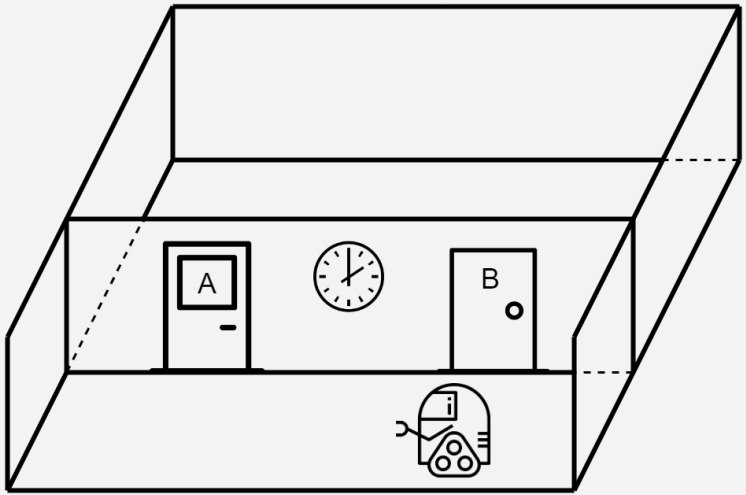
LARG Learning Agents Research Group The University of Texas at Austin | RPL UT Robot Perception & Learning Lab | Sony AI

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:

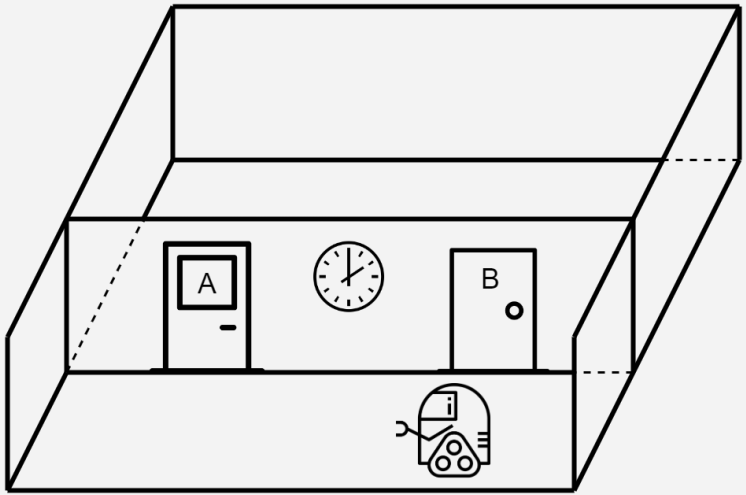


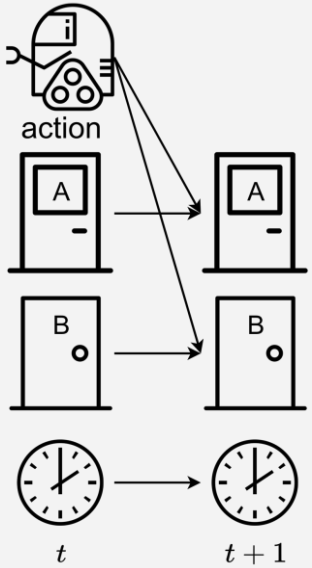
Motivation

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:

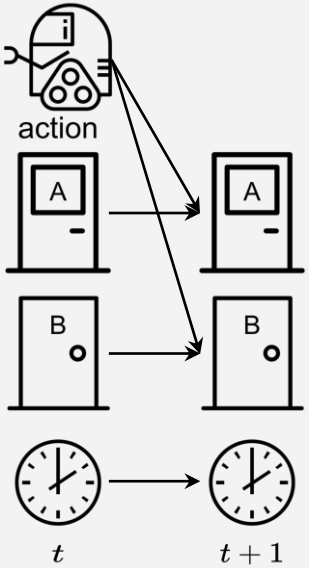




sparse real-world dynamics

Motivation

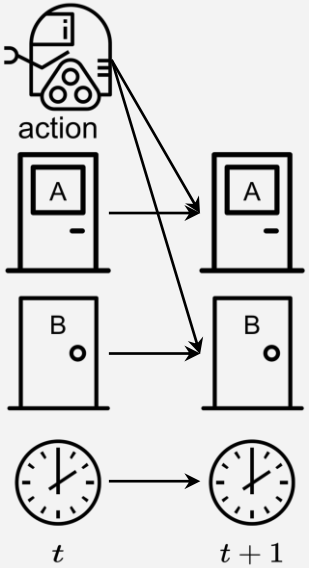
But most model-based RL work uses dense dynamics models (fully-connected networks).



sparse real-world dynamics

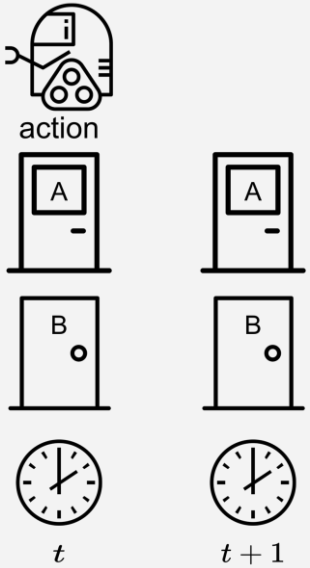
Motivation

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



sparse real-world dynamics

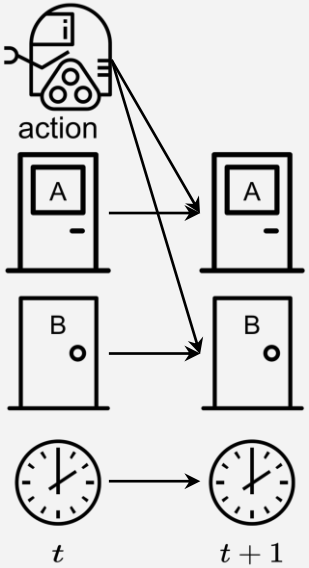
VS



dense dynamics model

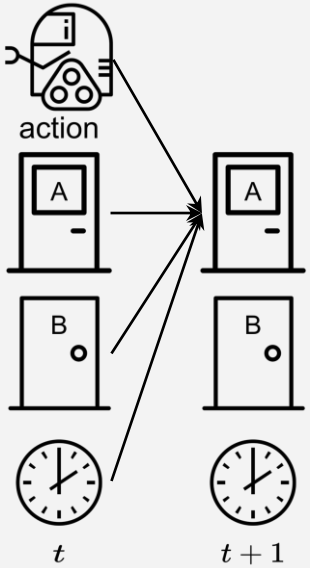
Motivation

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



sparse real-world dynamics

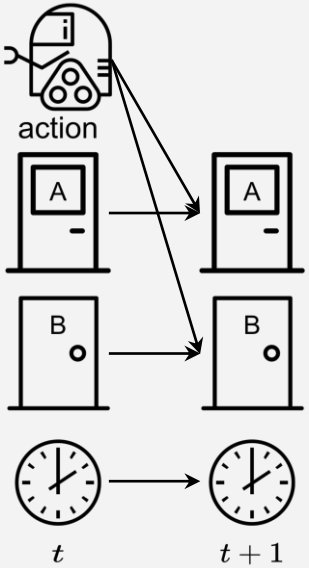
VS



dense dynamics model

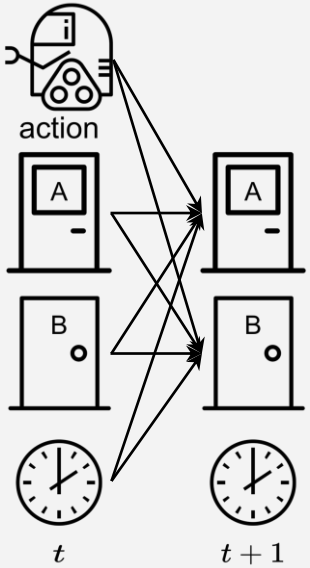
Motivation

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



sparse real-world dynamics

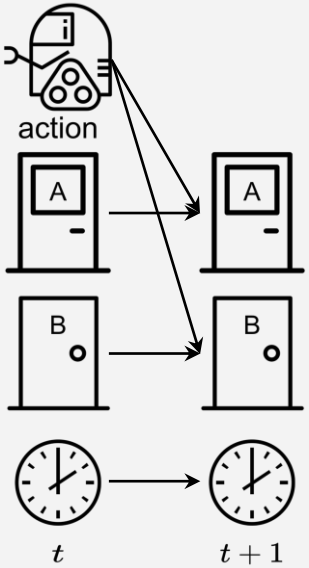
VS



dense dynamics model

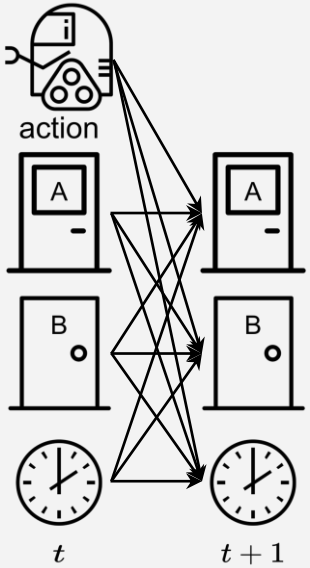
Motivation

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



sparse real-world dynamics

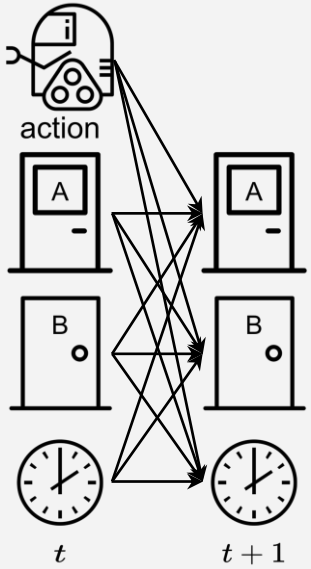
VS



dense dynamics model

Motivation

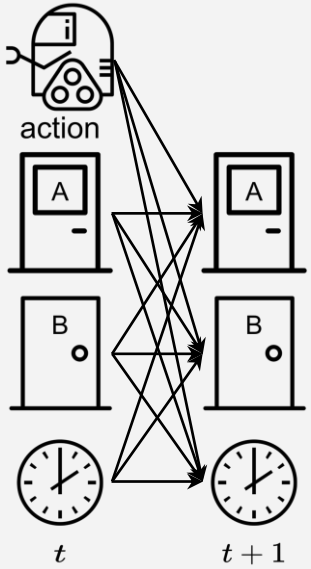
dense dynamics model



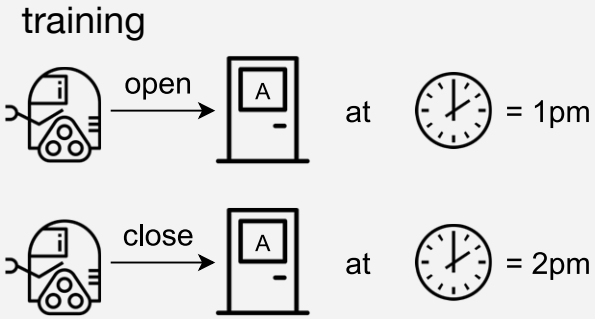
generalizes badly
due to spurious correlation

Motivation

dense dynamics model



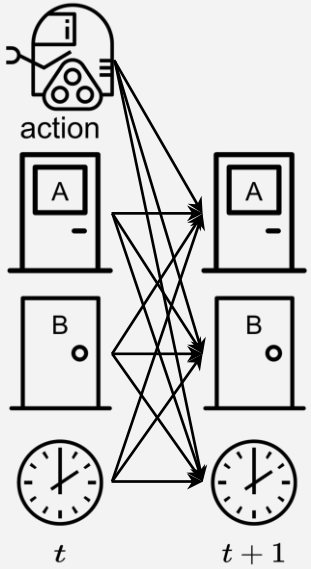
generalizes badly
due to spurious correlation

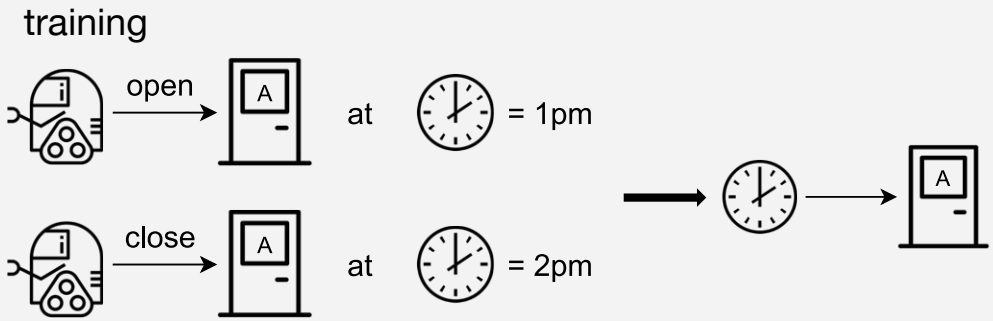


overfit to data noise, etc

Motivation

dense dynamics model



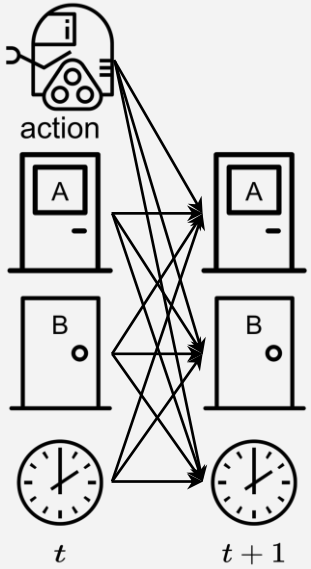


generalizes badly
due to spurious correlation

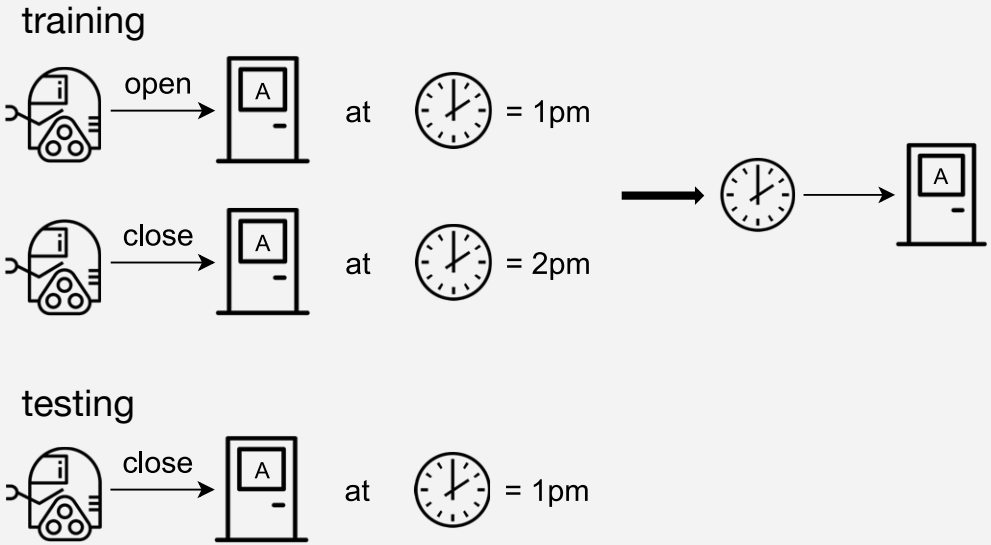
overfit to data noise, etc

Motivation

dense dynamics model

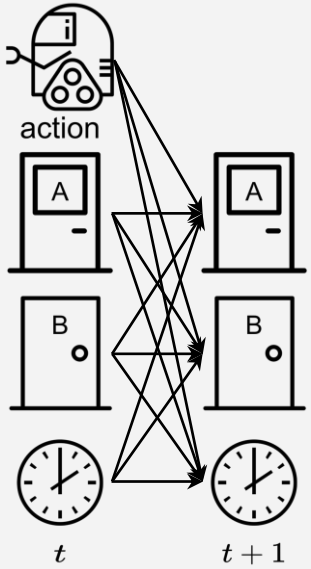


generalizes badly
due to spurious correlation



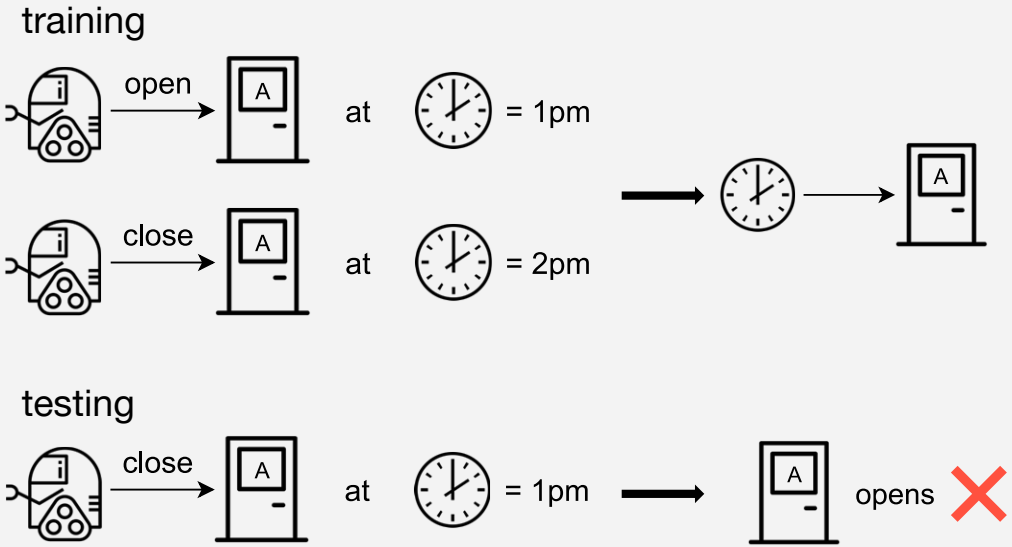
overfit to data noise, etc

Motivation



generalizes badly
due to spurious correlation

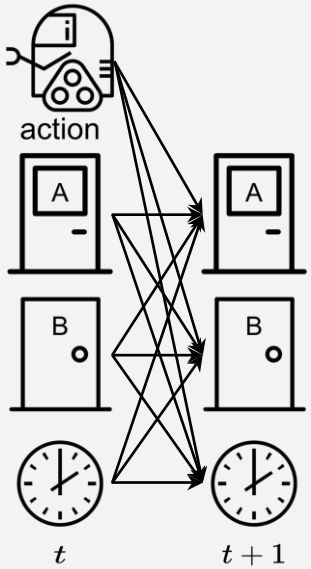
dense dynamics model



overfit to data noise, etc

Motivation

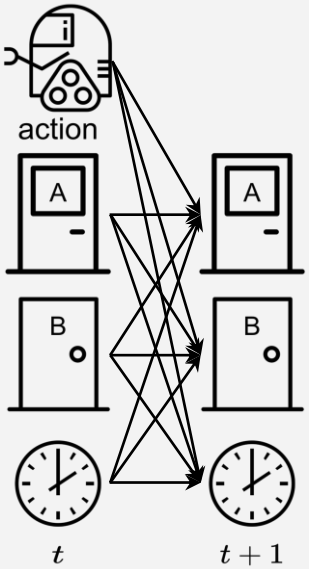
dense dynamics model



generalizes badly
due to spurious correlation

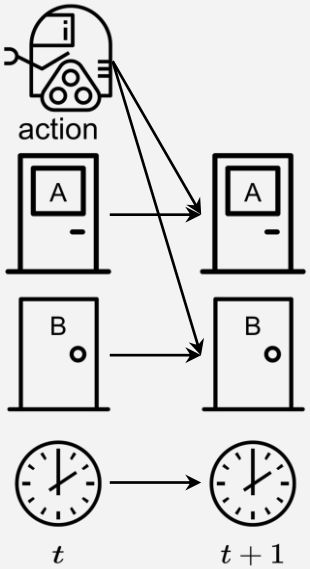
Motivation

dense dynamics model



generalizes badly
due to spurious correlation

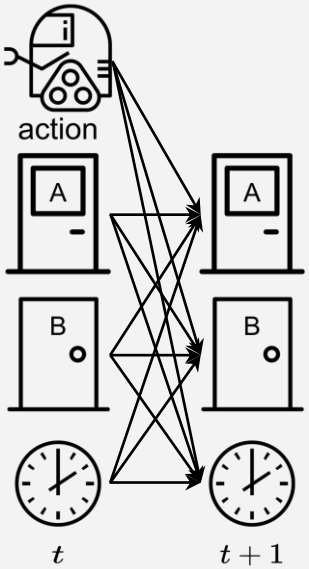
causal dynamics learning (CDL)



only keep causal edges, robust to outliers,

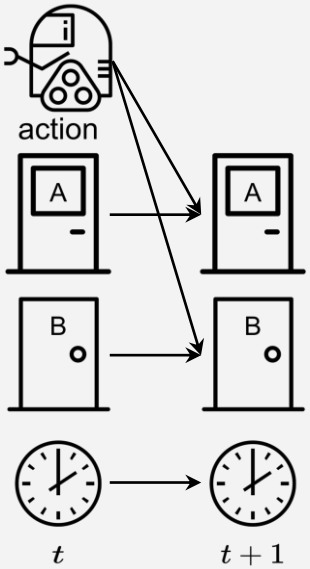
Motivation

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

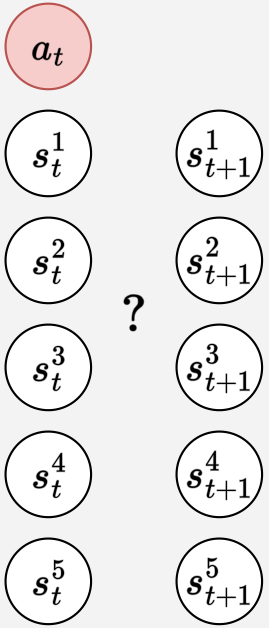
Problem Setup

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

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)



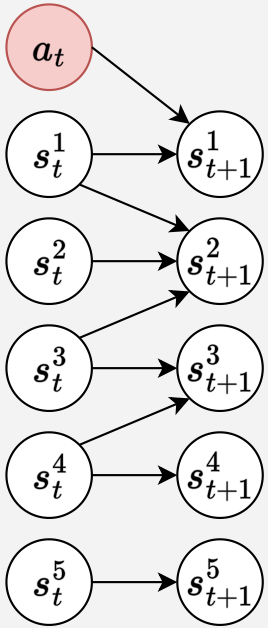
Problem Setup

Goals

1. Learn a causal dynamics model from transition data

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

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



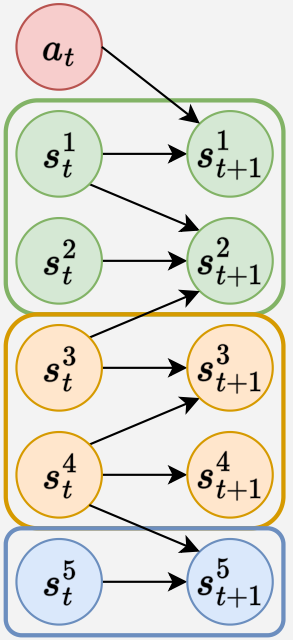
Problem Setup

Goals

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

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

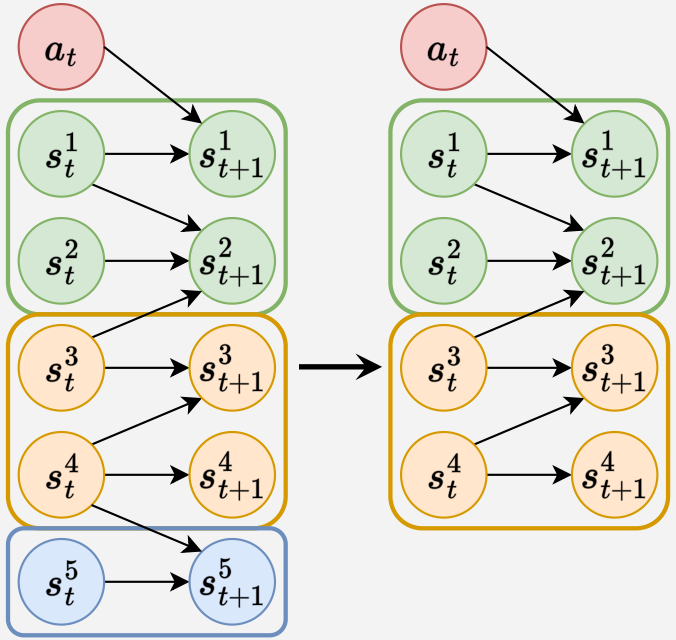
S^c: space of controllable state variables
S^r: space of action-relevant state variables
S^i: space of action-irrelevant state variables



Problem Setup

Goals

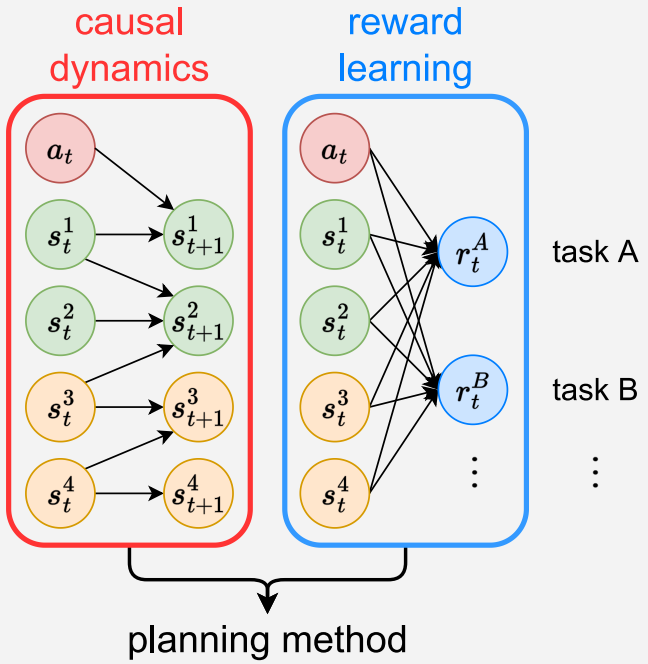
1. Learn a causal dynamics model from transition data
2. Split state variables into three categories
3. Derive a state abstraction by omitting action-irrelevant state variables



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

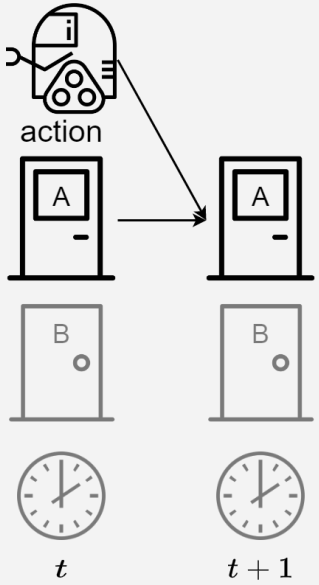
$$R(x, a) = R(x', a),$$
$$\sum_{x'' \in \phi^{-1}(s)} P(x''|x, a) = \sum_{x'' \in \phi^{-1}(s)} P(x''|x', a)$$

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

Related Work

Compared to CDL,

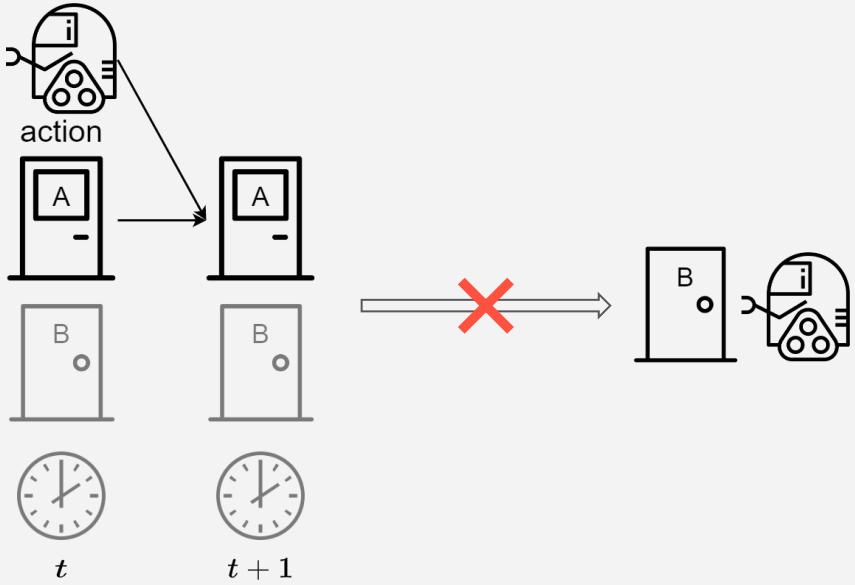
- Bisimulation is reward-specific (applicable to limited tasks).
  e.g., the bisimulation abstraction learned from “opening door A” can’t be used for “opening door B”.



Related Work

Compared to CDL,

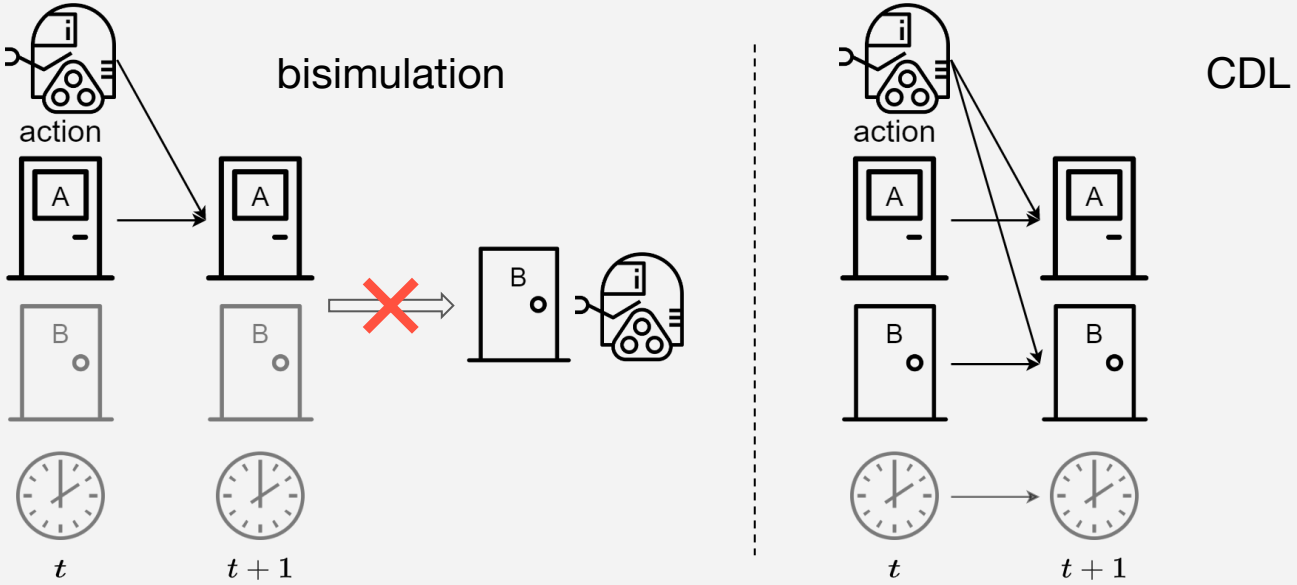
- Bisimulation is reward-specific (applicable to limited tasks).
  e.g., the bisimulation abstraction learned from “opening door A” can’t be used for “opening door B”.



Related Work

Compared to CDL,

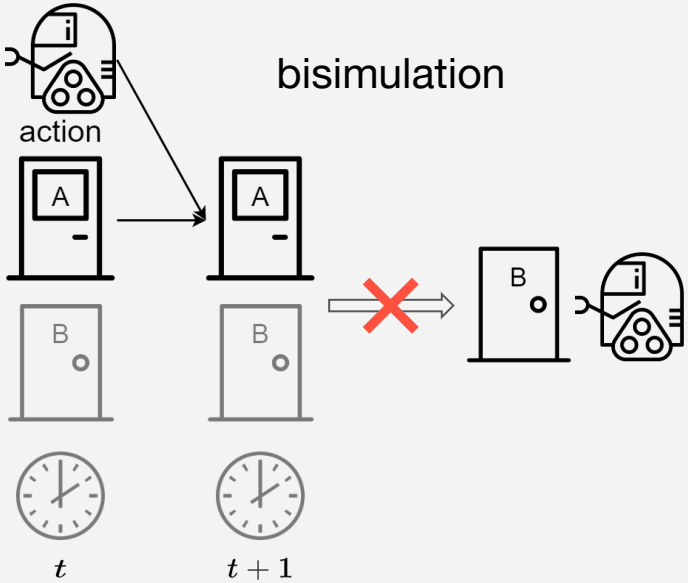
- Bisimulation is reward-specific and thus applicable to **limited** tasks.
  In contrast, CDL's abstraction can be applied to a larger range of tasks.

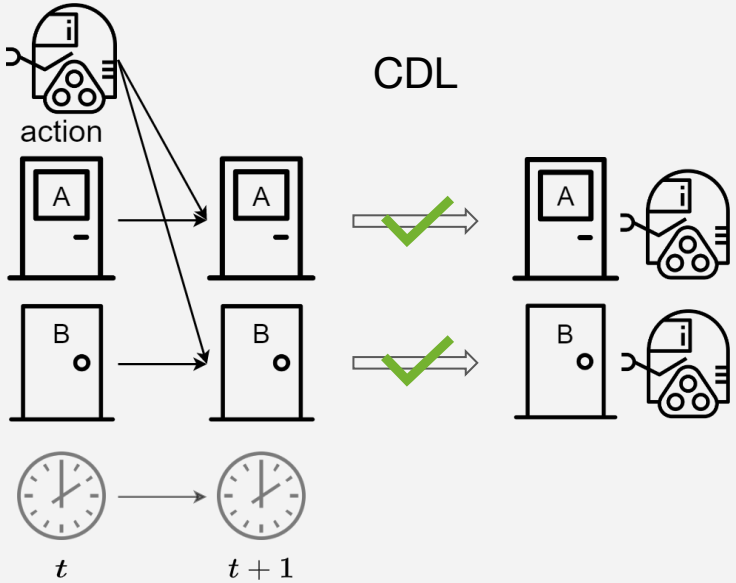


Related Work

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.

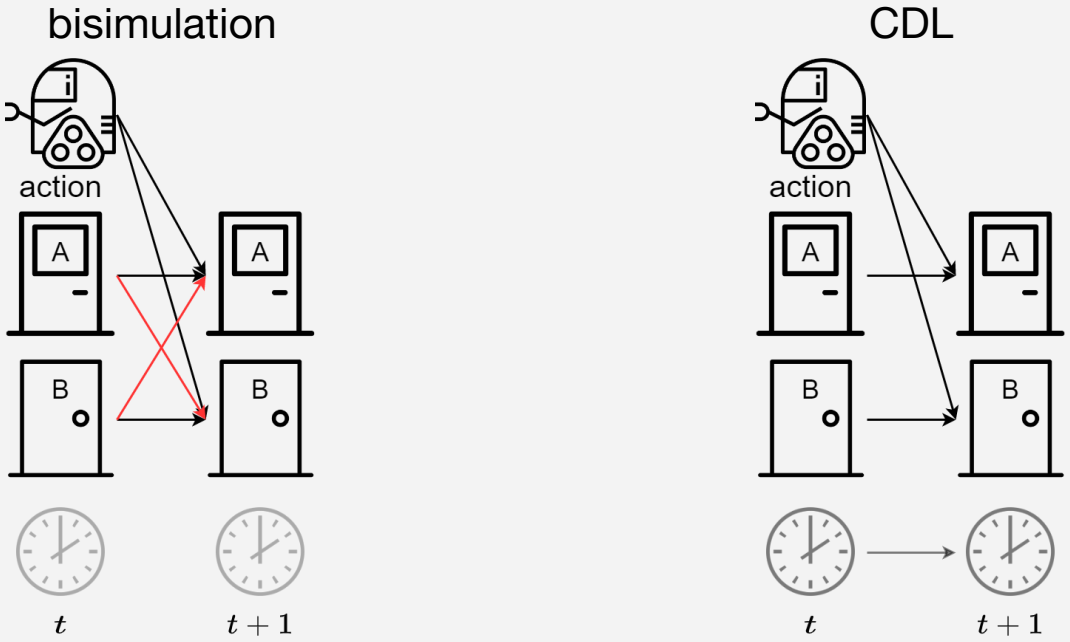




Related Work

Compared to CDL,

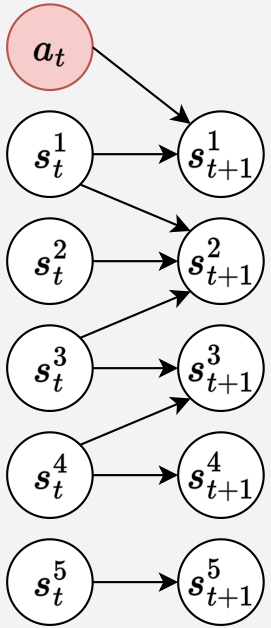
- Bisimulation is reward-specific and thus applicable to **limited** tasks.
- Most bisimulation work still uses dense dynamics, leading to poor generalization.



Method

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

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

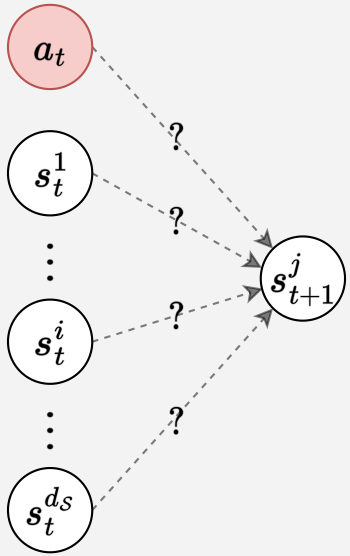


Method

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

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

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 $s_t^i \rightarrow s_{t+1}^j$ exists with a conditional independence test.

Skipping assumptions and proofs,

Method

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

Skipping assumptions and proofs,

Theorem 1

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

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

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

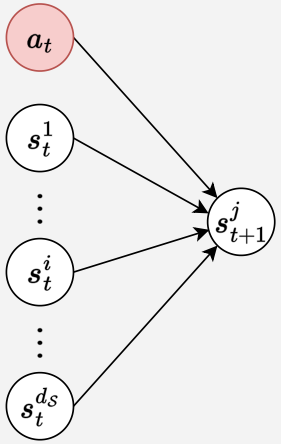
In other words, is s_t^i needed to predict s_{t+1}^j?

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

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

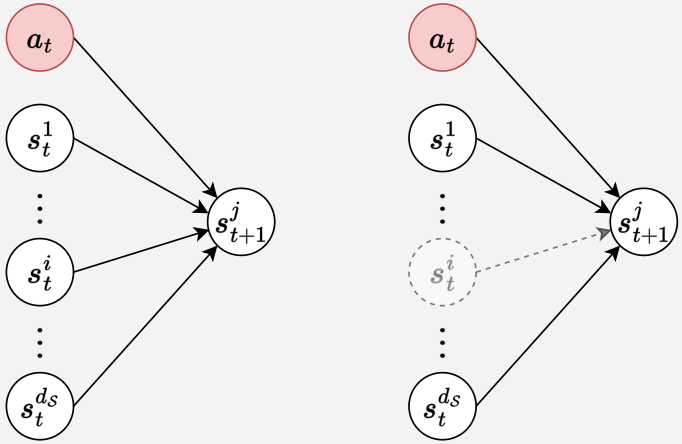


Method

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

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

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

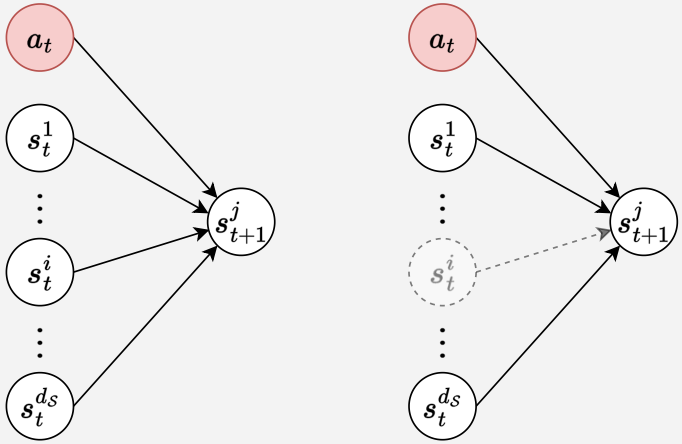


Method

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?



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

Method

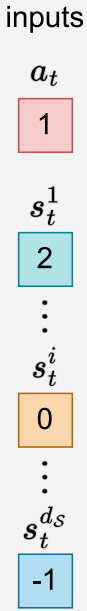
Learn and predict $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ using generative models, but there will be $d^2_{\mathcal{S}}$ models to train...

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 a mask M_j and an element-wise maximum module, one network can represent all generative models in the form of $p(s^j_{t+1}|\cdot)$.

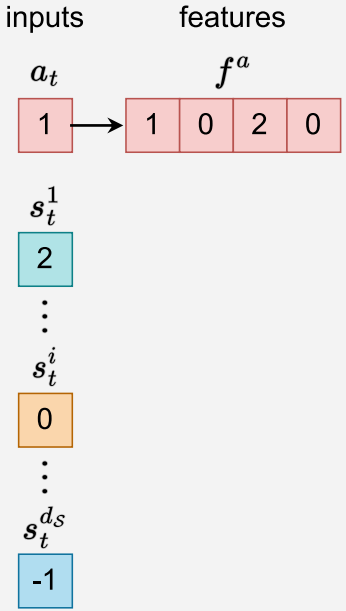
Method

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



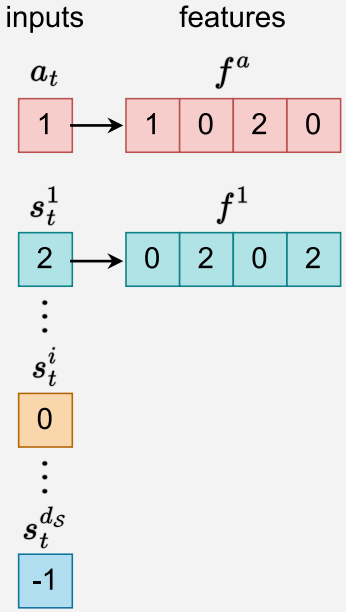
Method

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



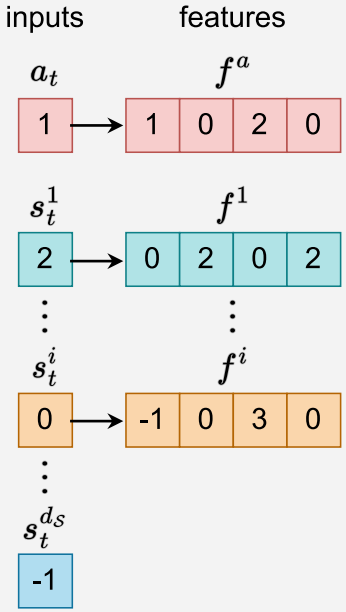
Method

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



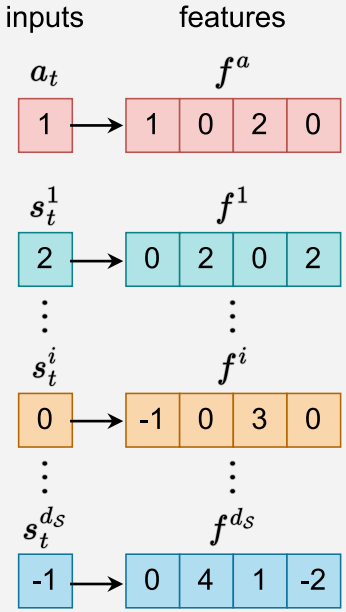
Method

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



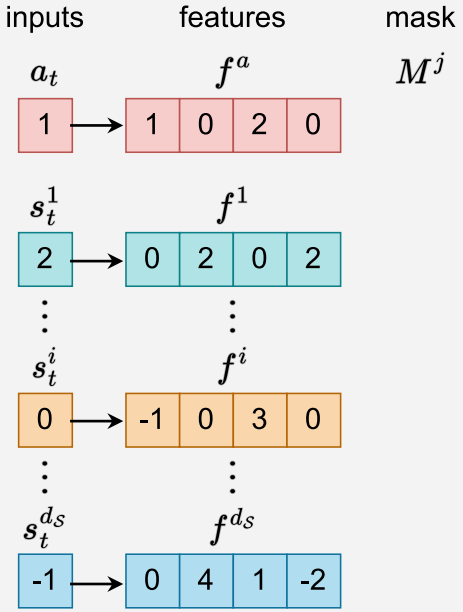
Method

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



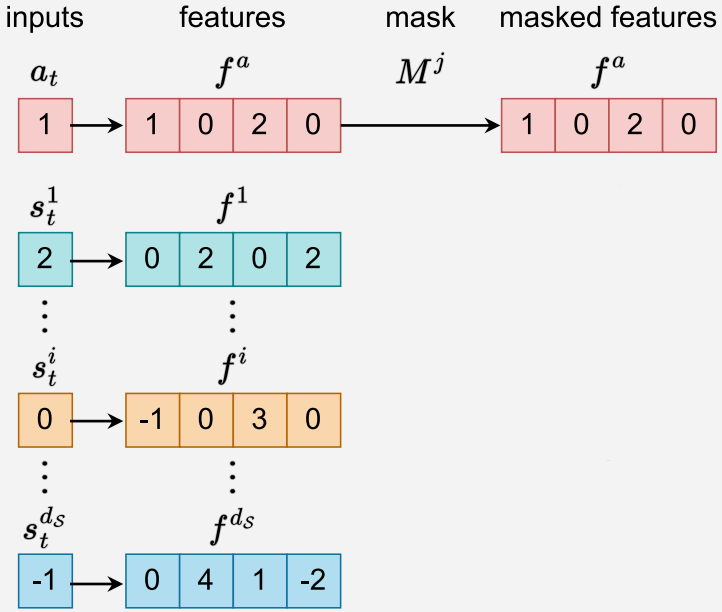
Method

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



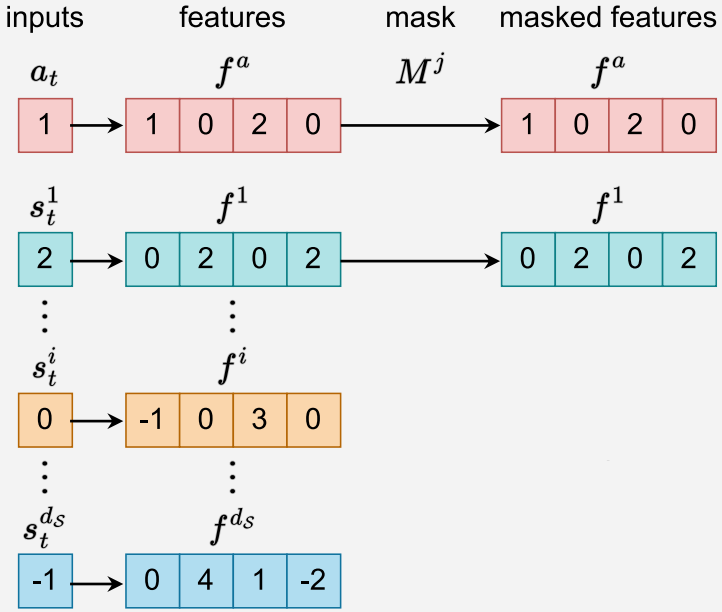
Method

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



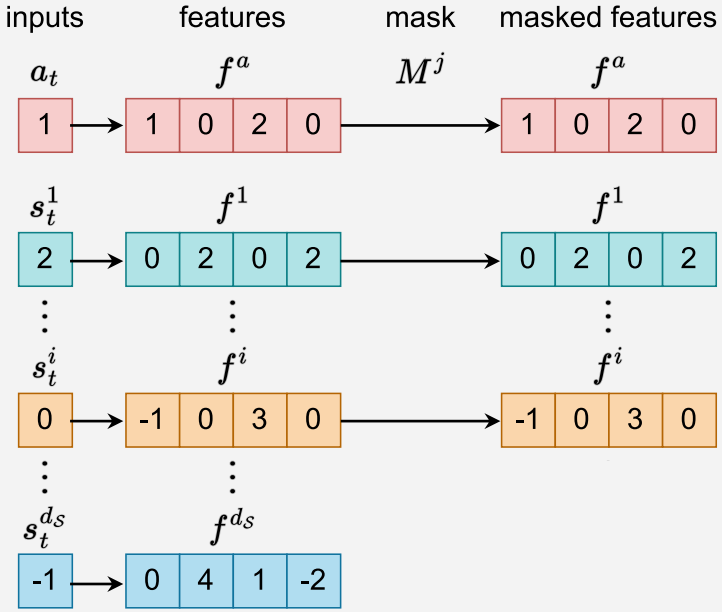
Method

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



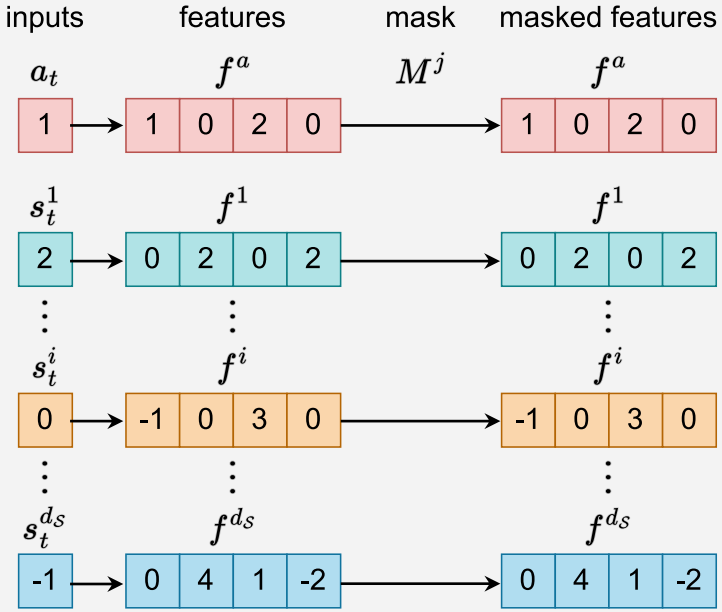
Method

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



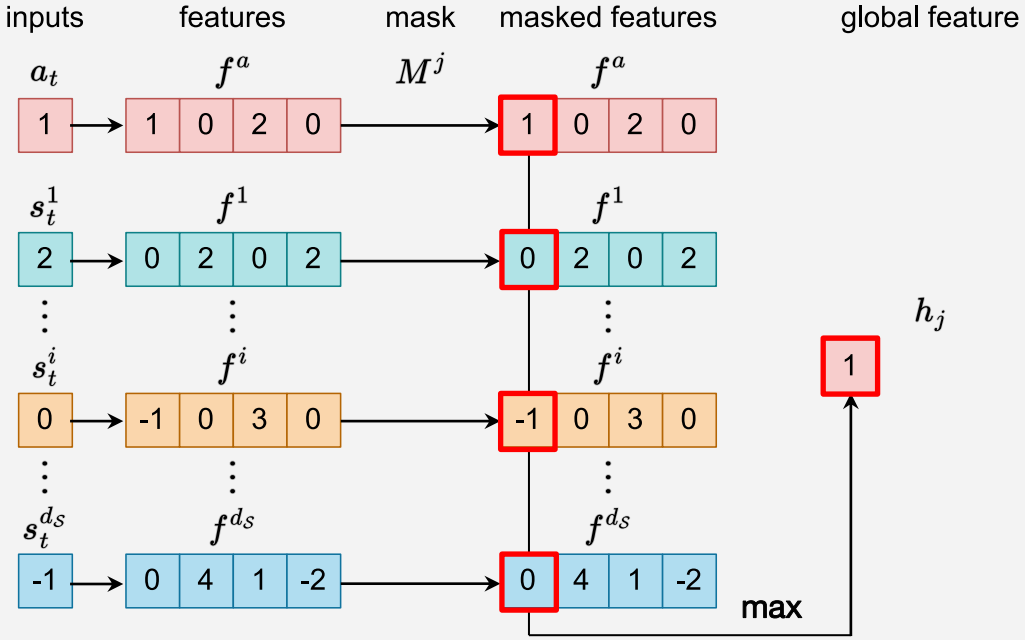
Method

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



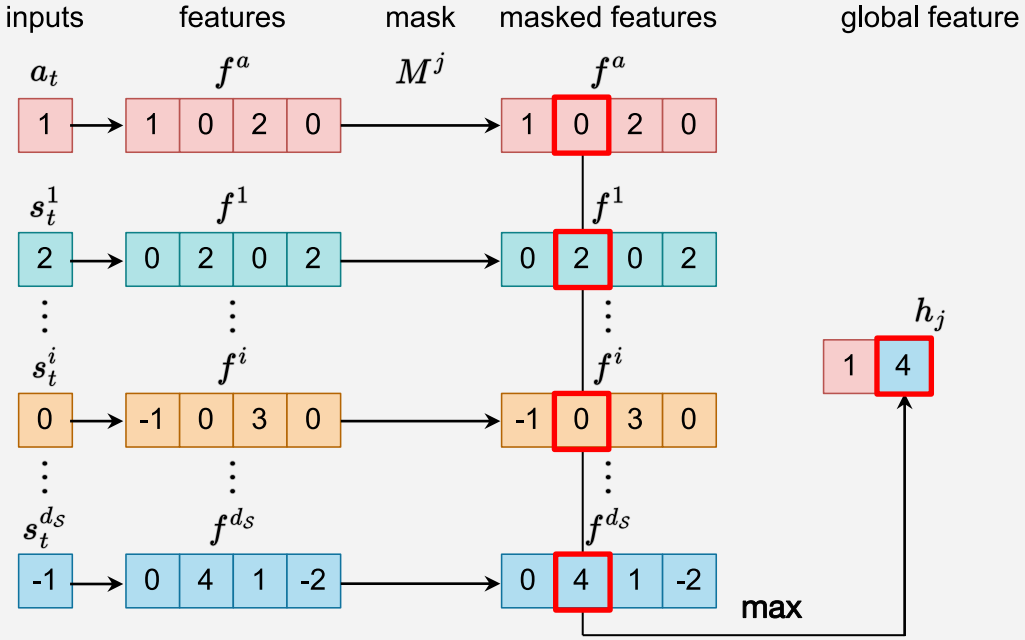
Method

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



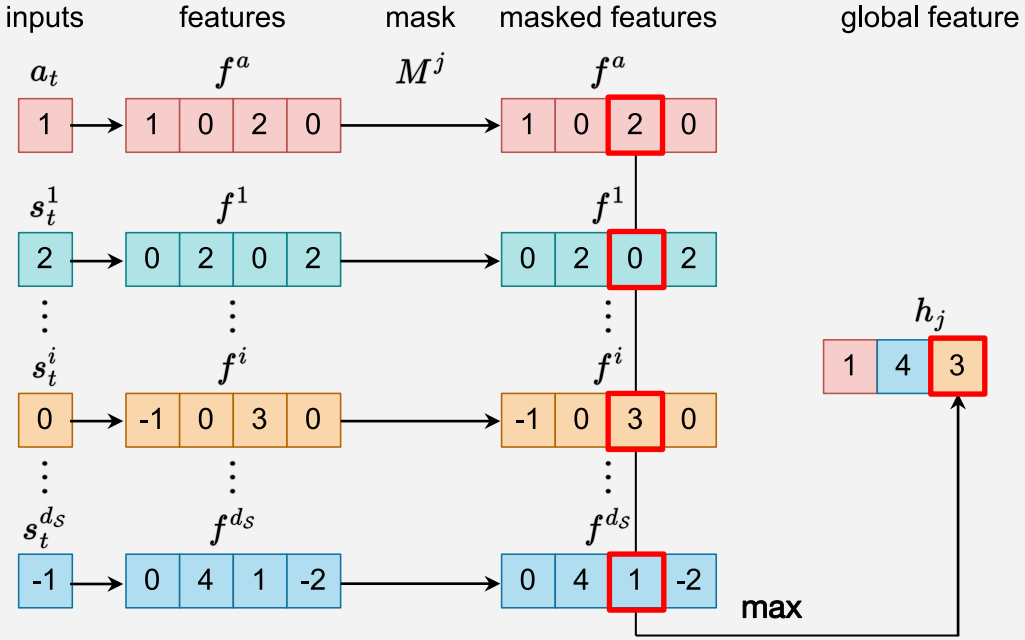
Method

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



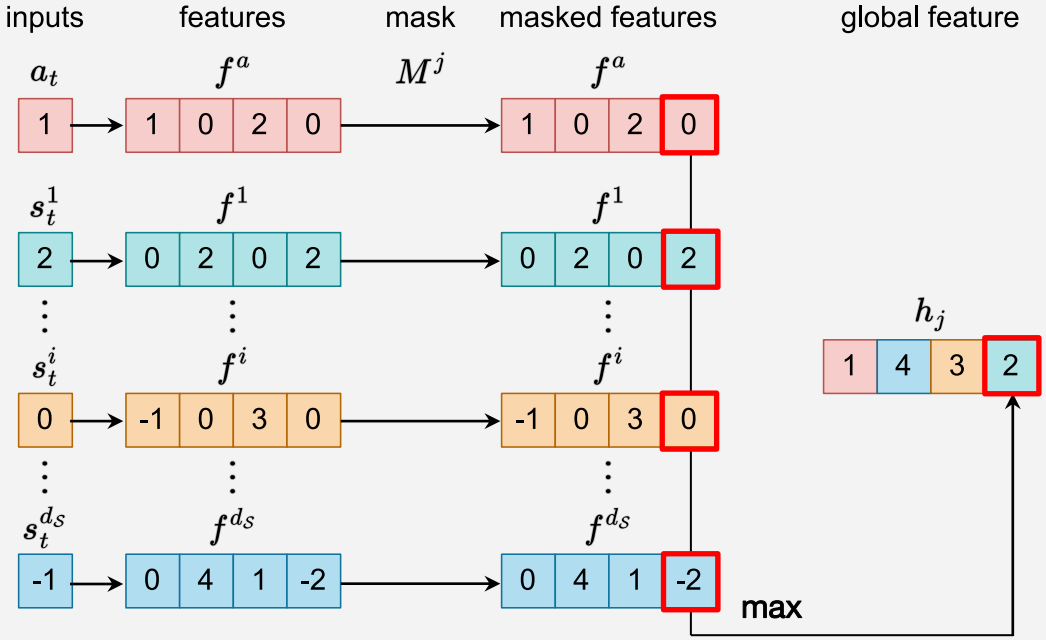
Method

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



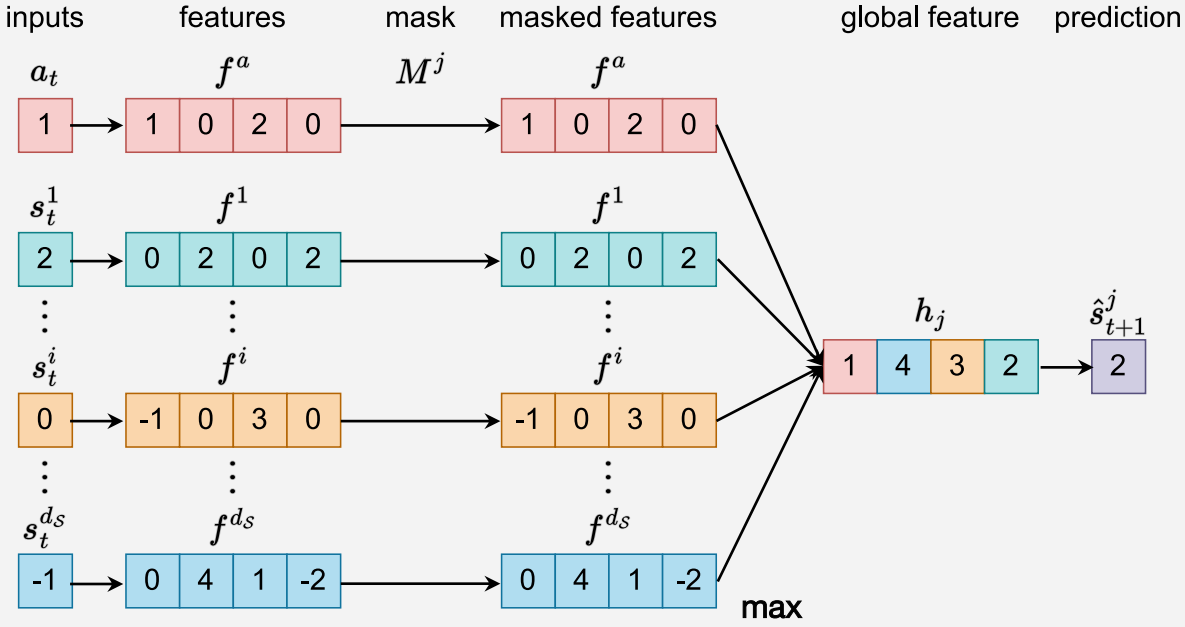
Method

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



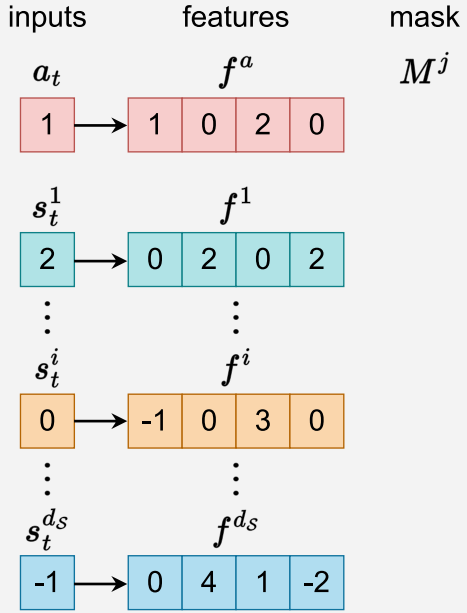
Method

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



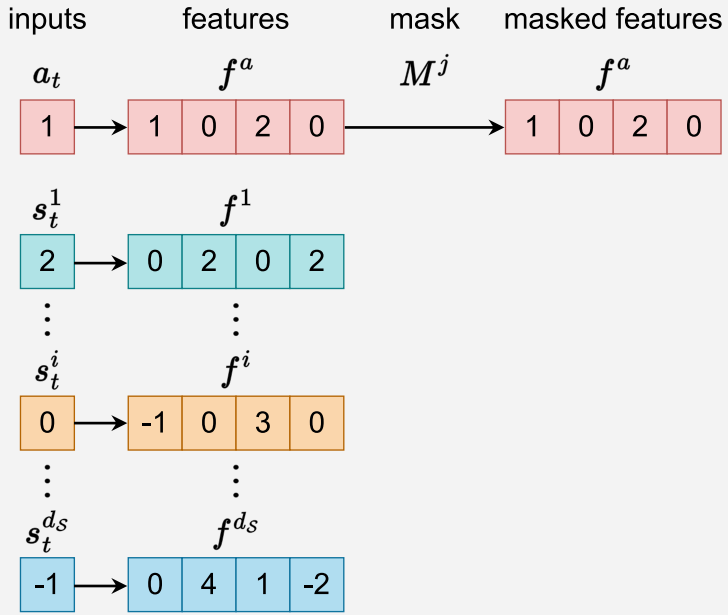
Method

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



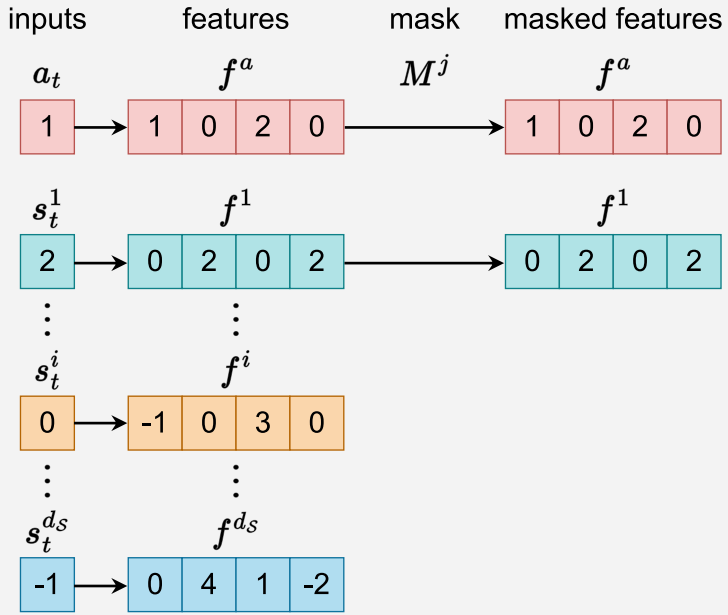
Method

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 $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



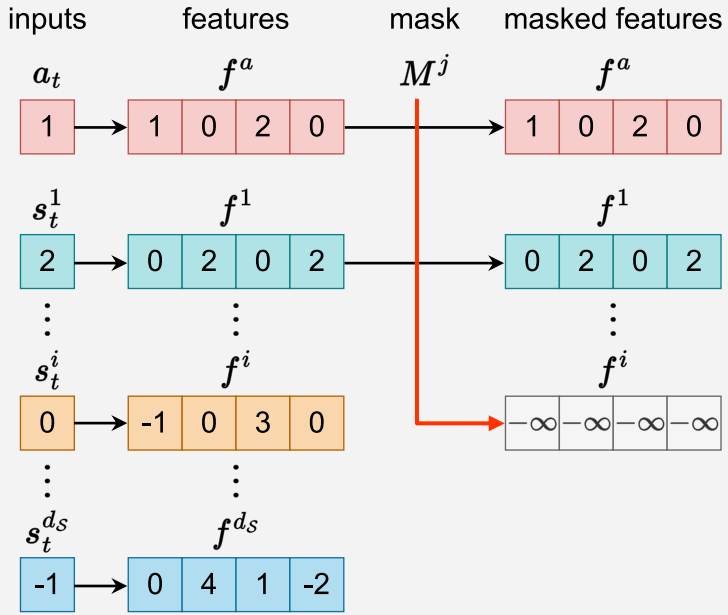
Method

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 $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



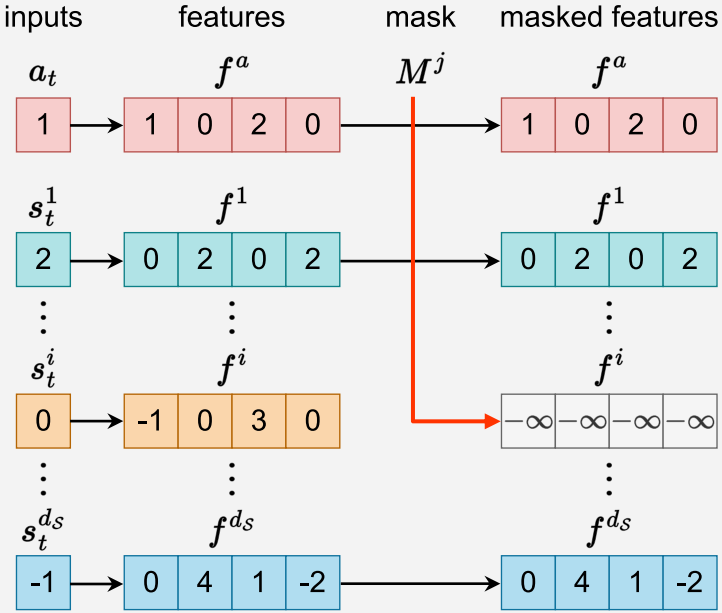
Method

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



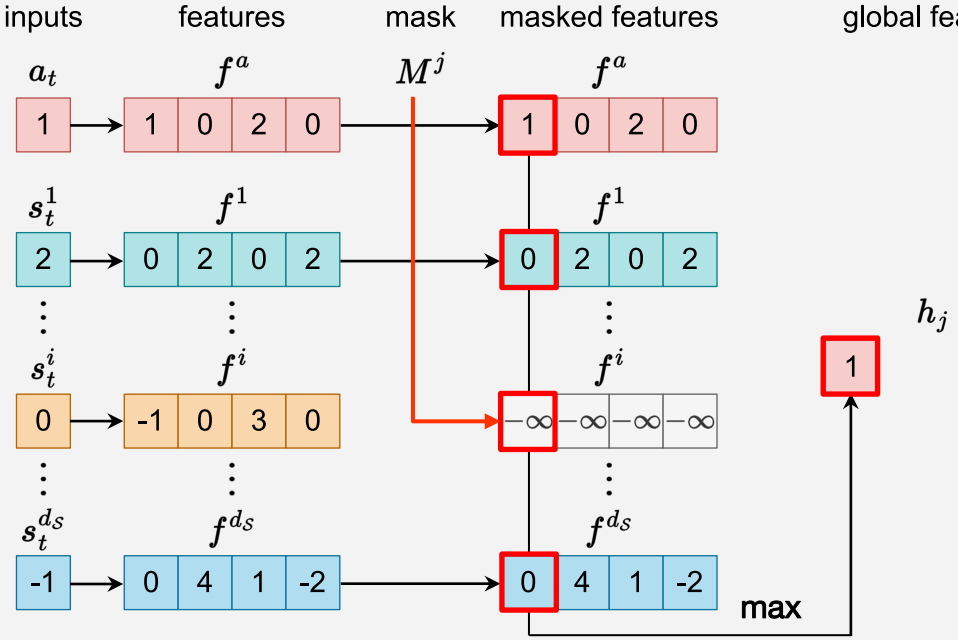
Method

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 $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



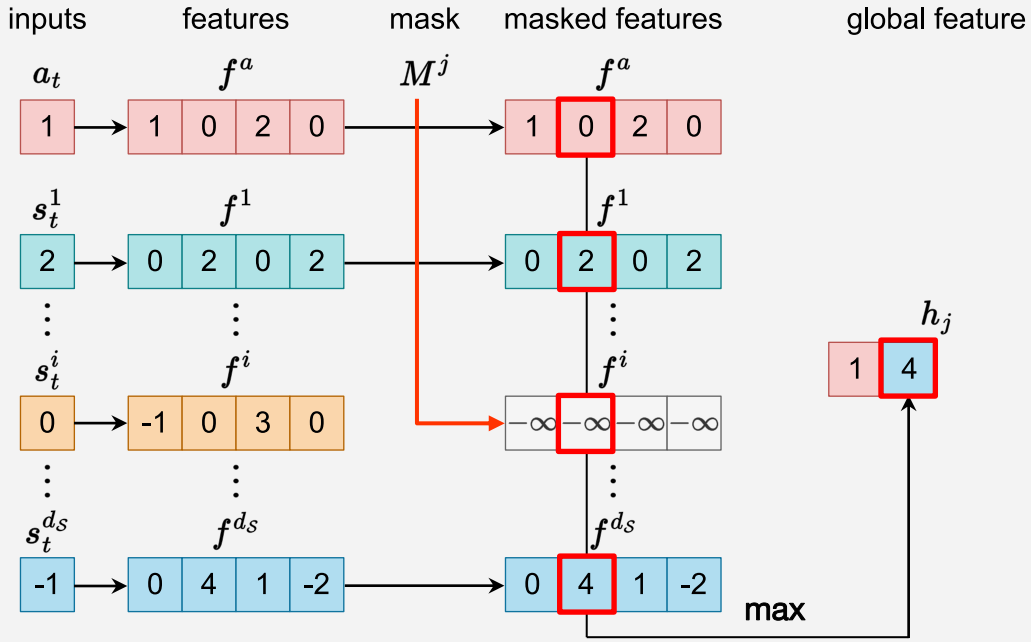
Method

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



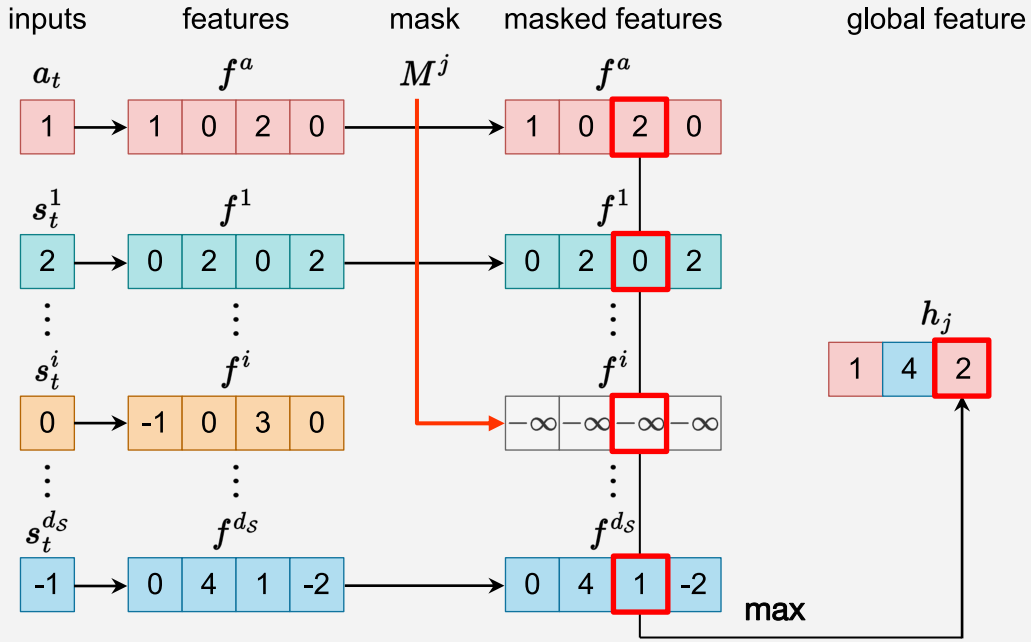
Method

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 $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



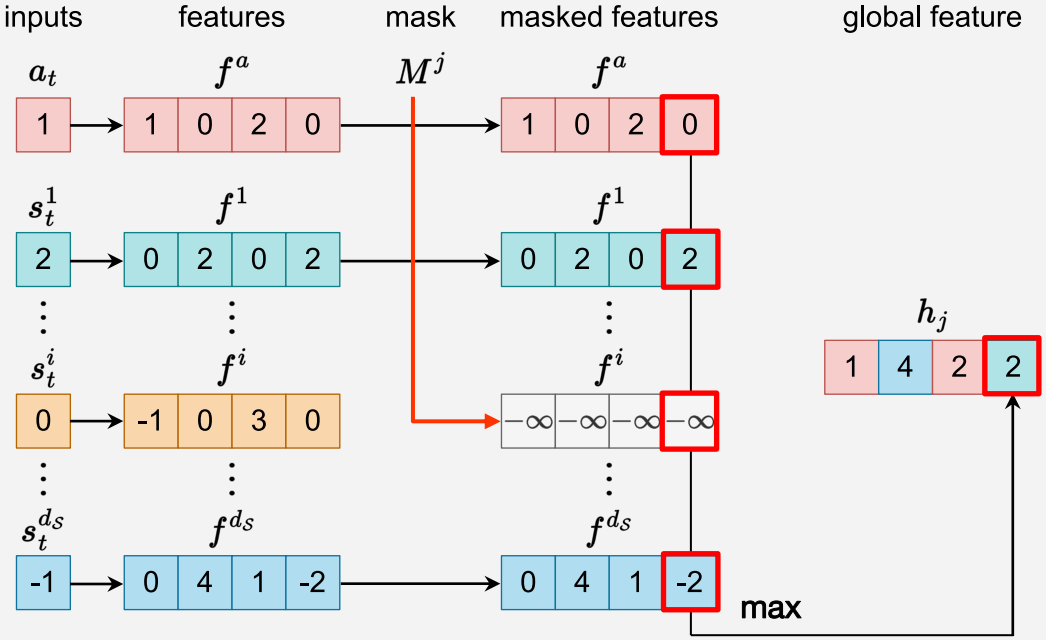
Method

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



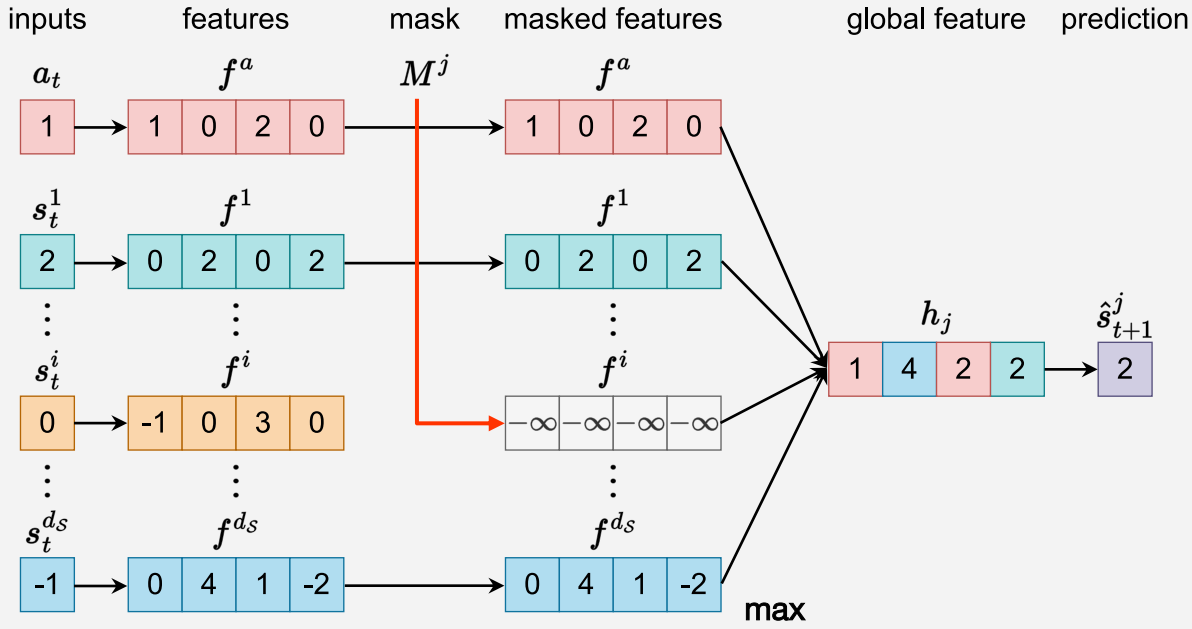
Method

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 $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



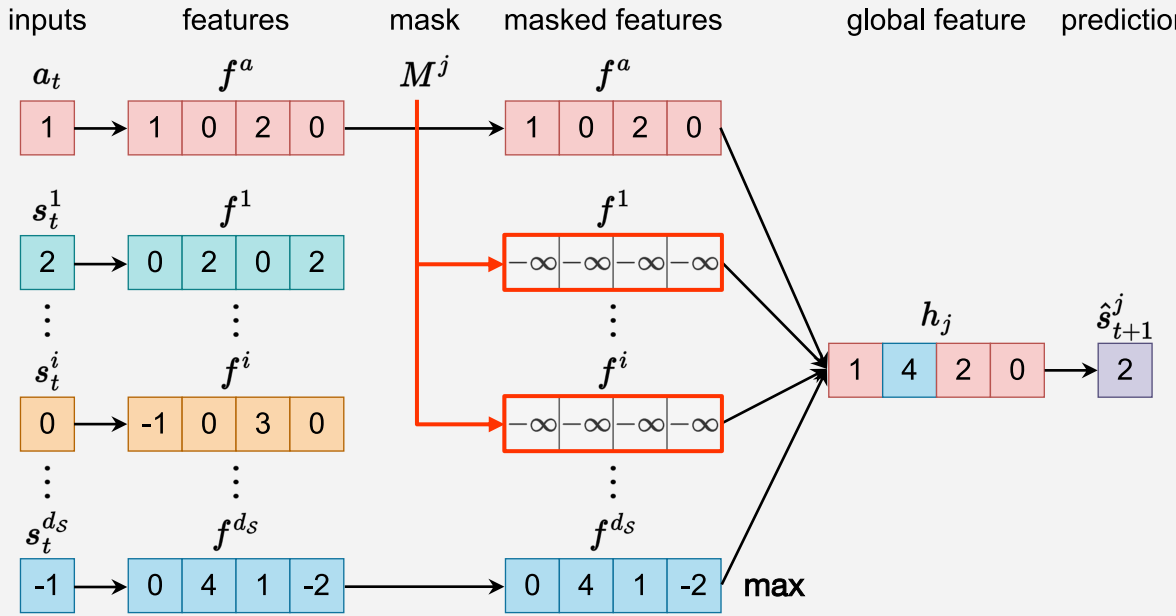
Method

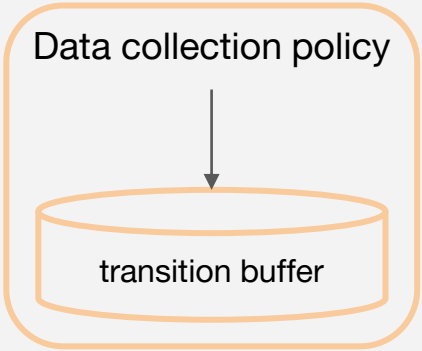
Learning $p(s^j_{t+1}|s_t, a_t)$ & $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$ needs to train d_S^2 models.
With M_j and an element-wise maximum module, one network can represent all models.
For example, to represent $p(s^j_{t+1}|\{s/s^i\}_t, a_t)$,



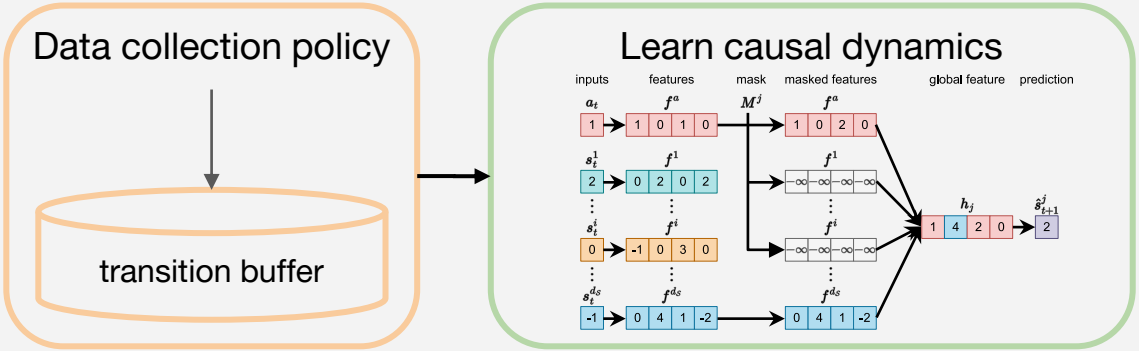
Method

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.

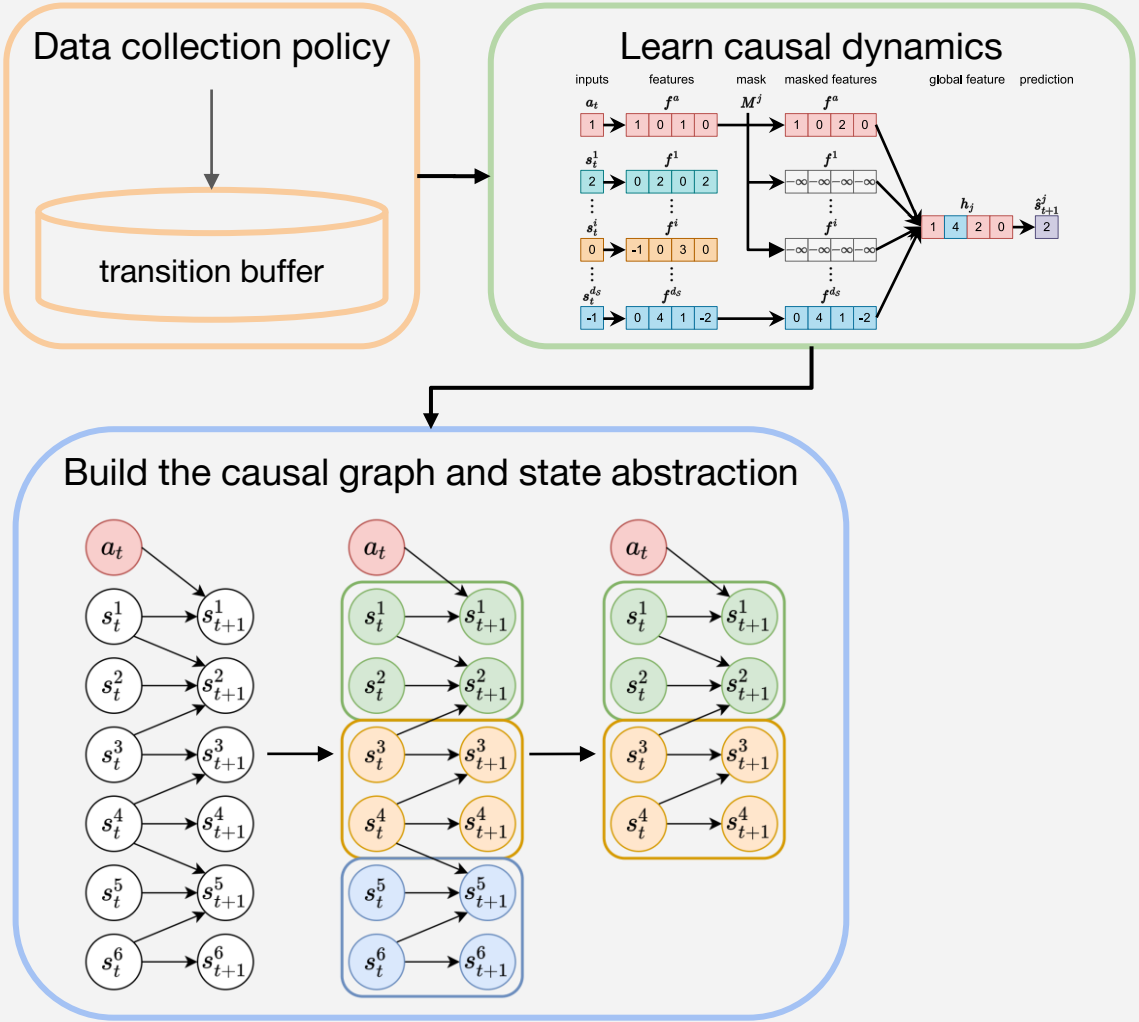




Causal Dynamics Learning (CDL)

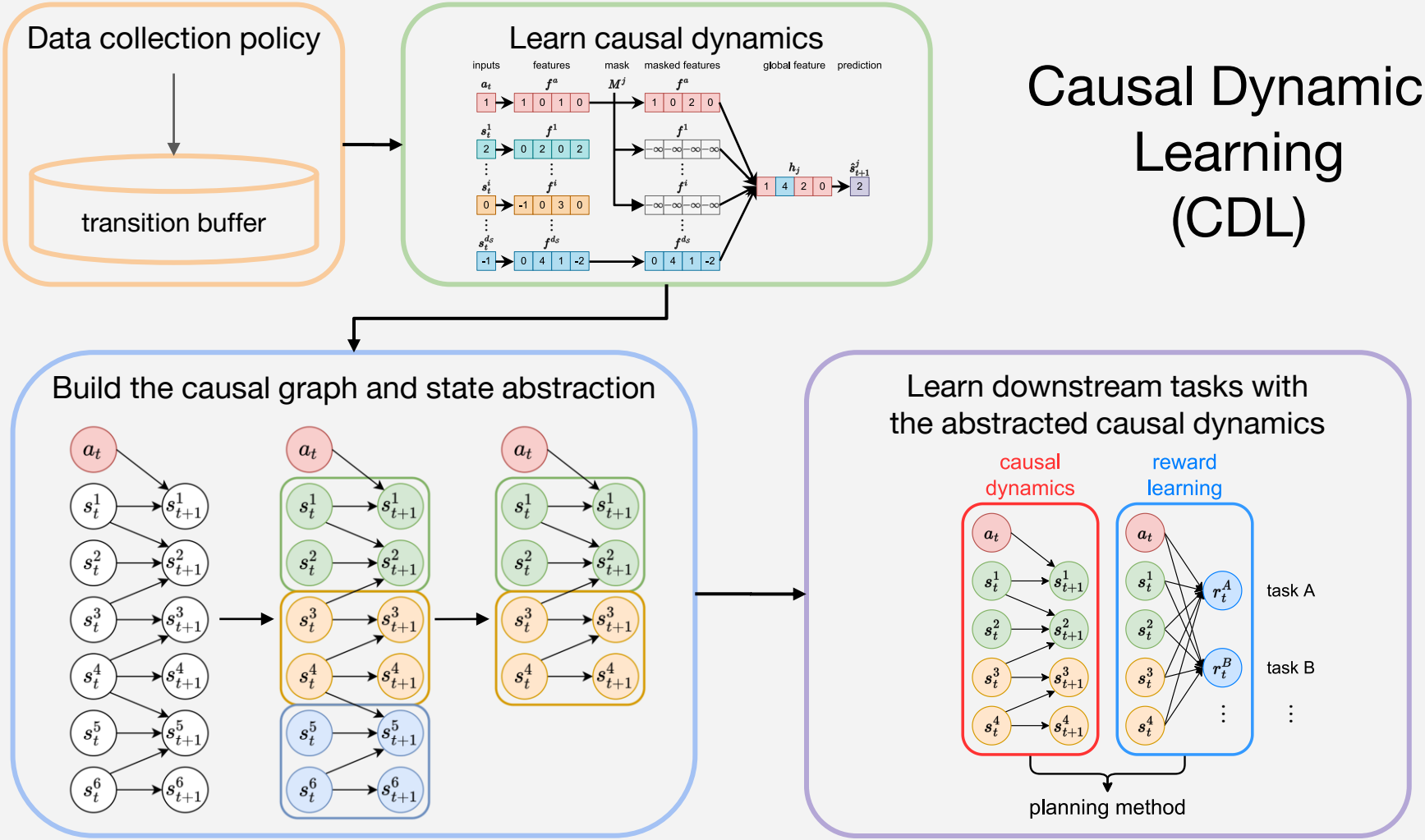


Causal Dynamics Learning (CDL)



Causal Dynamics Learning (CDL)

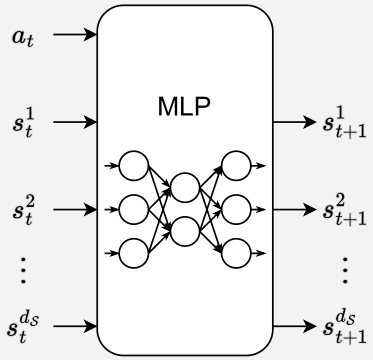
Causal Dynamics Learning (CDL)



Experiments

Baselines

Monolithic

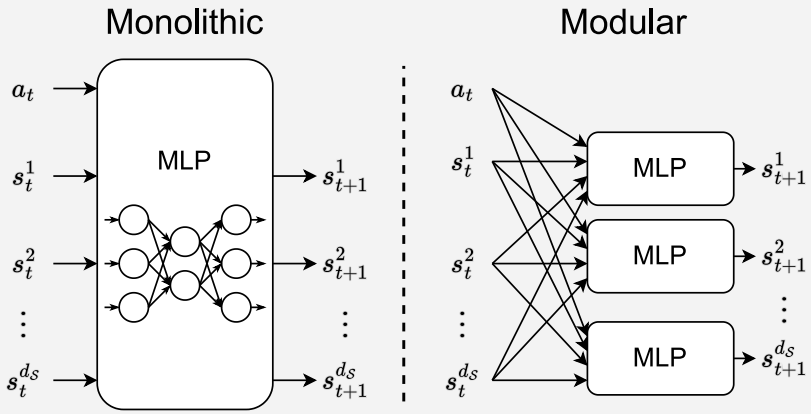


MLP: multi-layer perceptron

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

Experiments

Baselines

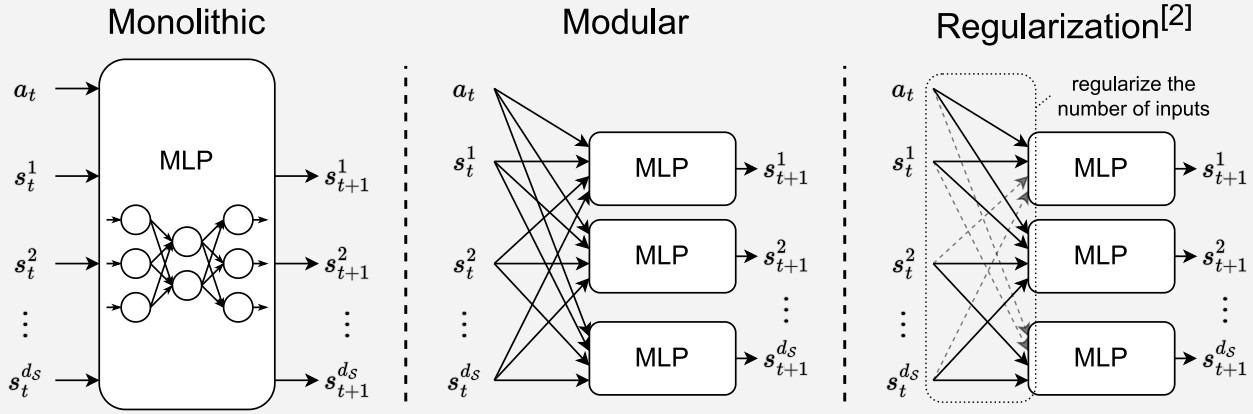


MLP: multi-layer perceptron

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

Experiments

Baselines

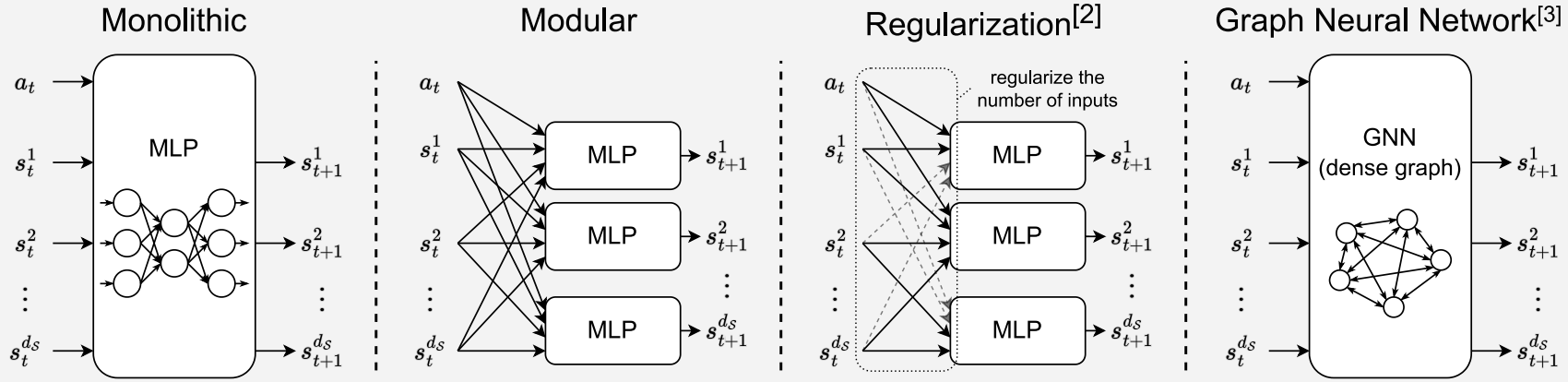


MLP: multi-layer perceptron

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

Experiments

Baselines

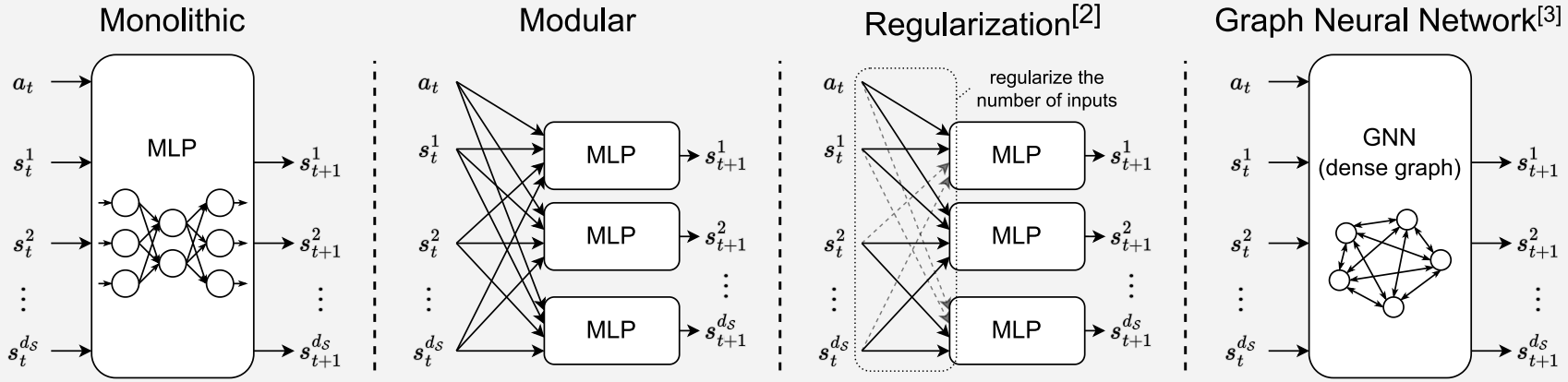


MLP: multi-layer perceptron

[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?

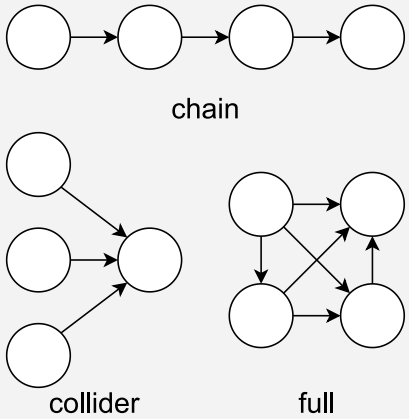
✗ ✗ ✓ ✗

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

Experiments

Chemical Environment[4]

Synthesized environment
– with different underlying graphs

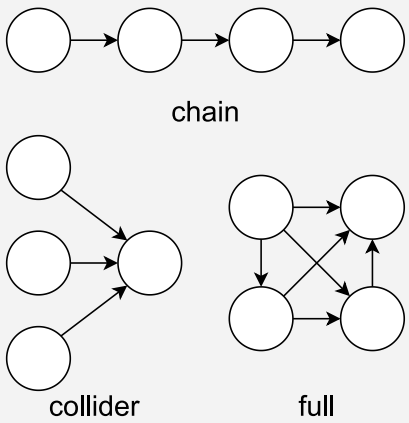


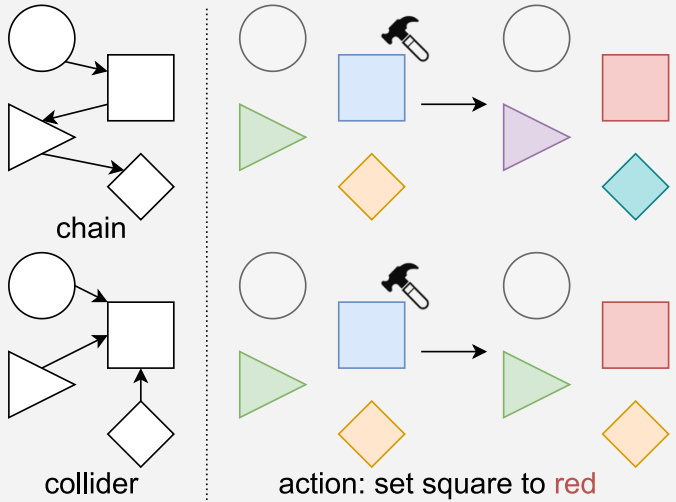
[4] Ke et al., Neurips 2021.

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.





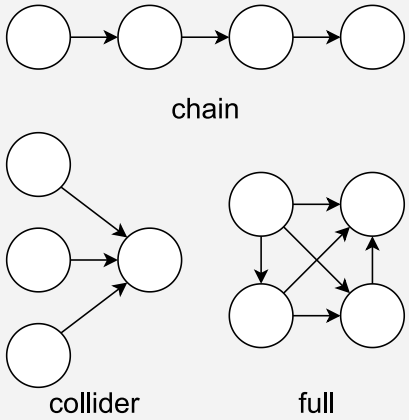
[4] Ke et al., Neurips 2021.

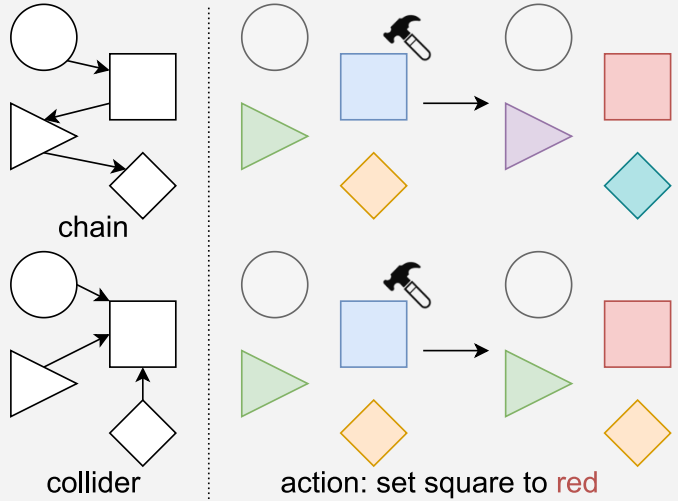
Experiments

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 $N(0, 0.01)$.



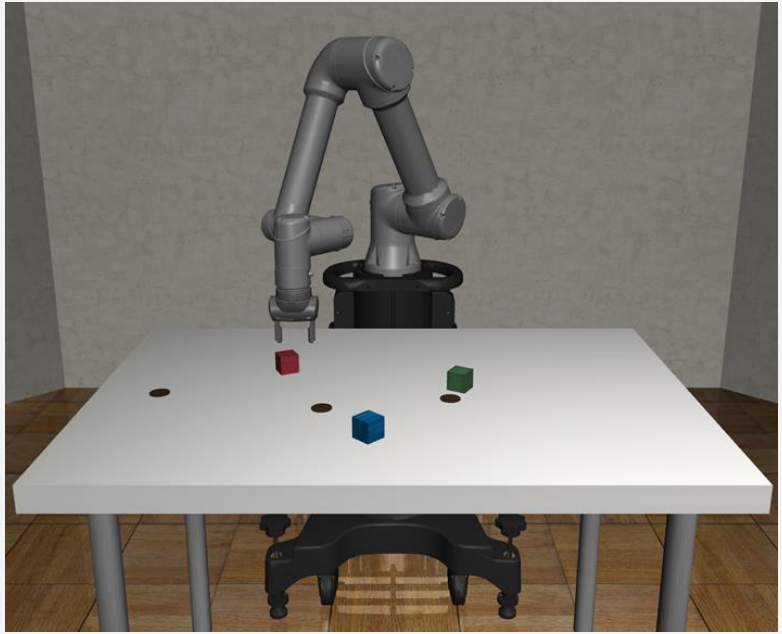


[4] Ke et al., Neurips 2021.

Experiments

Manipulation Environment

State Variables:



Experiments

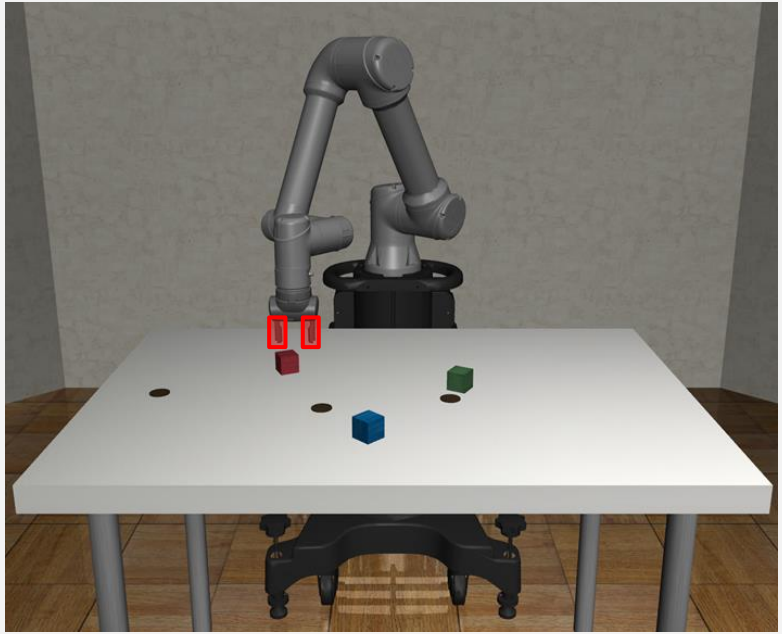
Manipulation Environment

State Variables:
- end-effector (eef)

Experiments

Manipulation Environment

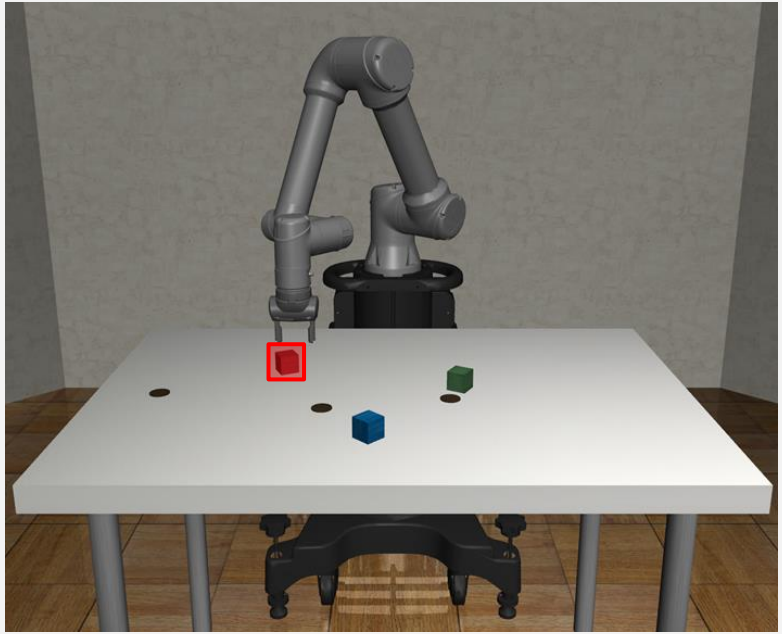
State Variables:
- end-effector (eef)
- gripper (grp)



Experiments

Manipulation Environment

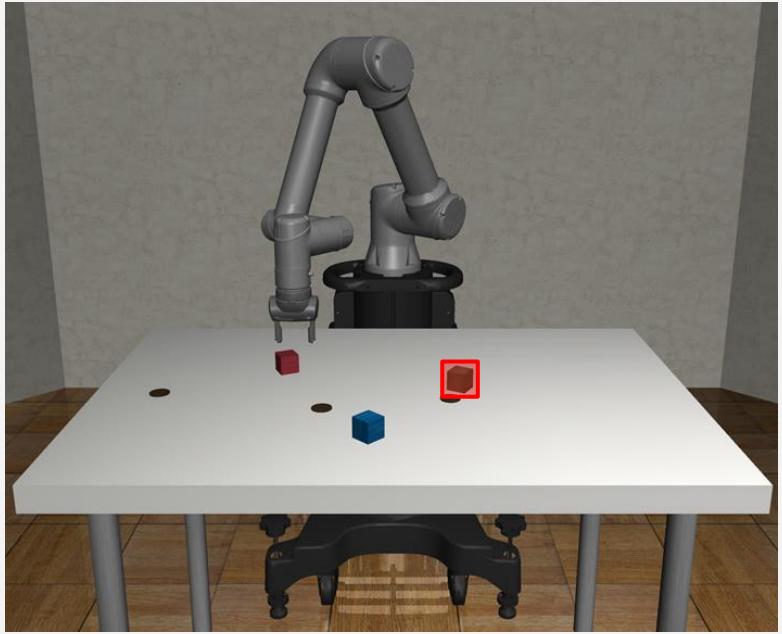
State Variables:
- end-effector (eef)
- gripper (grp)
- the movable object (mov)



Experiments

Manipulation Environment

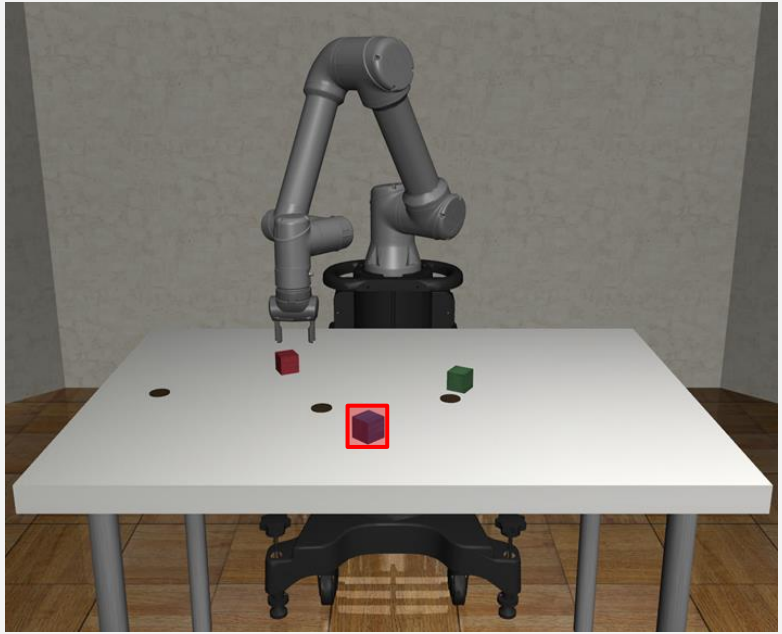
State Variables:
- end-effector (eef)
- gripper (grp)
- the movable object (mov)
- the unmovable object (unm)



Experiments

Manipulation Environment

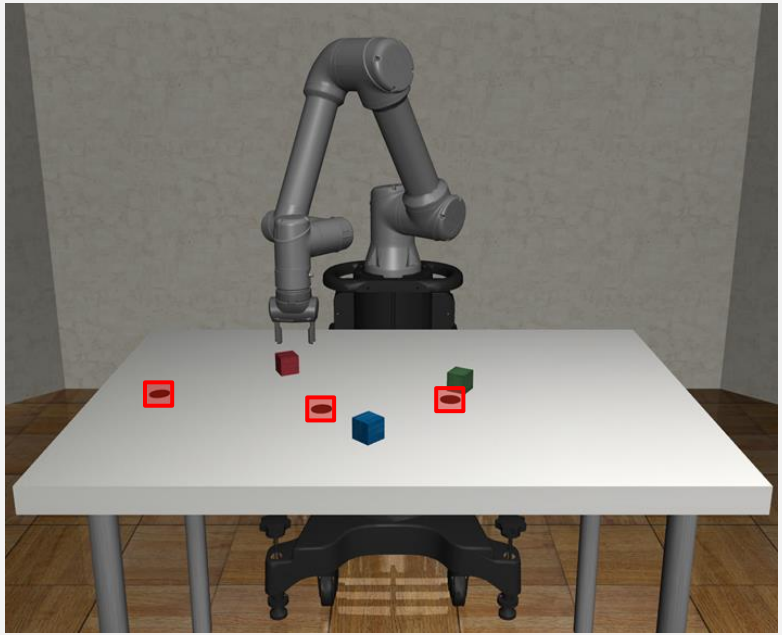
State Variables:
- end-effector (eef)
- gripper (grp)
- the movable object (mov)
- the unmovable object (unm)
- the randomly moving object (rand)



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^{1-3})

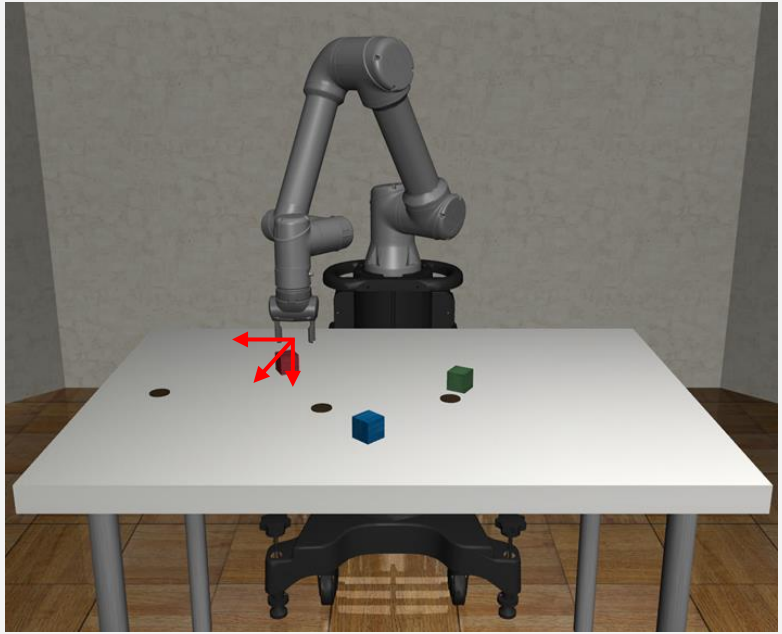


Experiments

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^{1-3})

Action dimensions:
- end-effector target

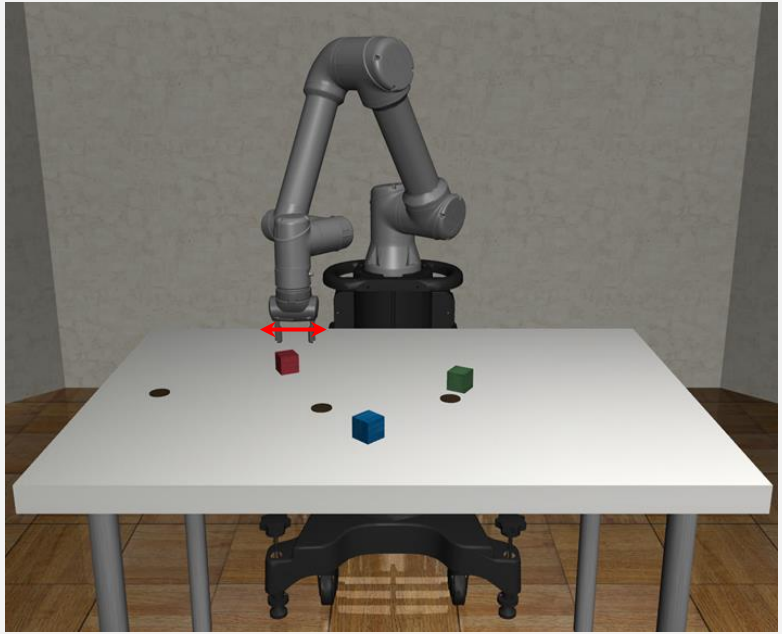


Experiments

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^{1-3})

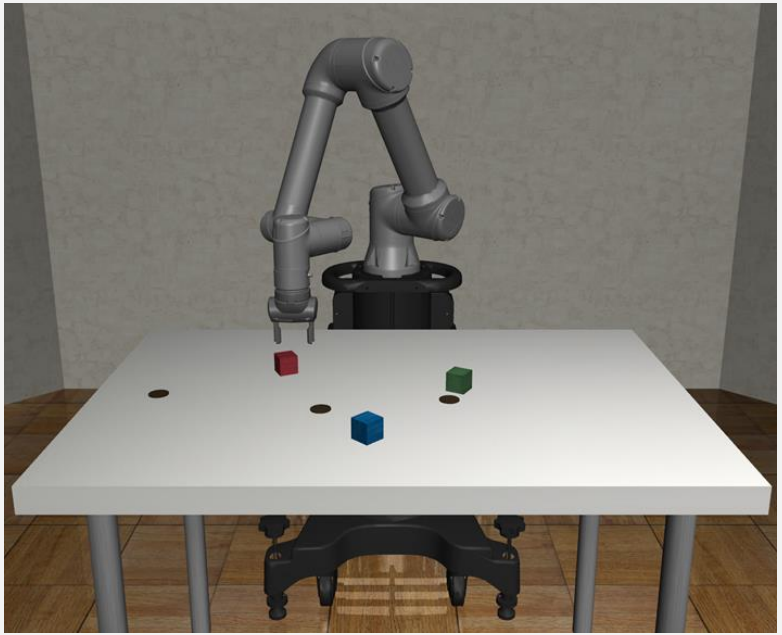
Action dimensions:
- end-effector target
- gripper open/close

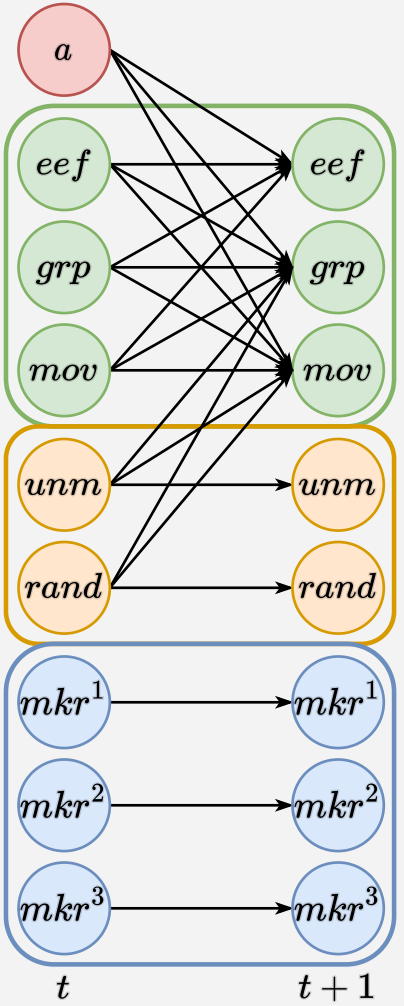


Results

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.

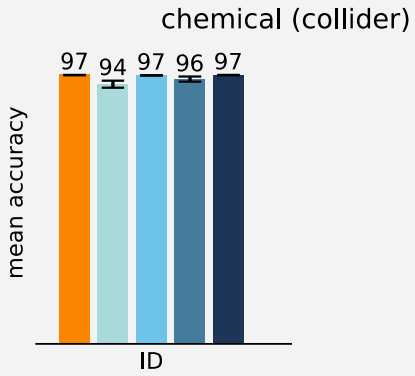
Results

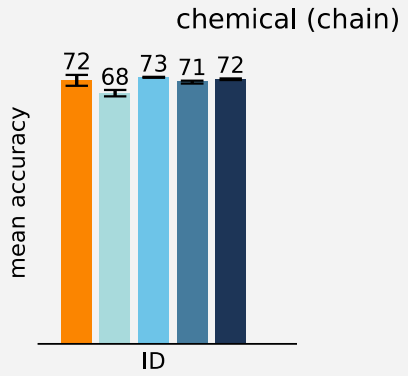
Dynamics Generalization

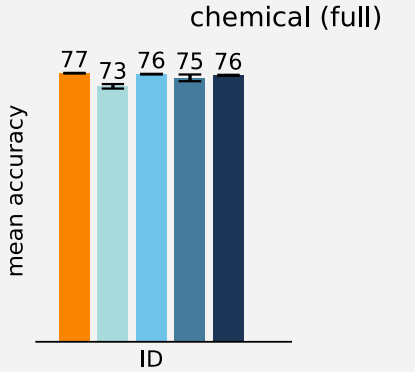
Causal dynamics generalizes best in unseen states.

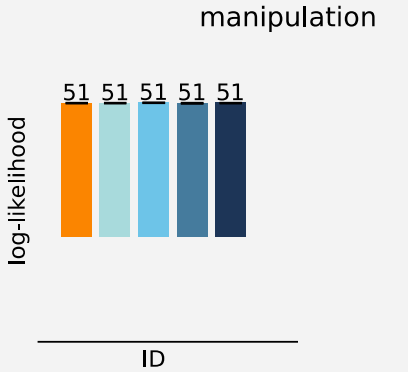
- Causal Dynamics Learning (Ours)
- Regularization
- Graph Neural Network
- Modular
- Monolithic

ID: in-distribution states









Results

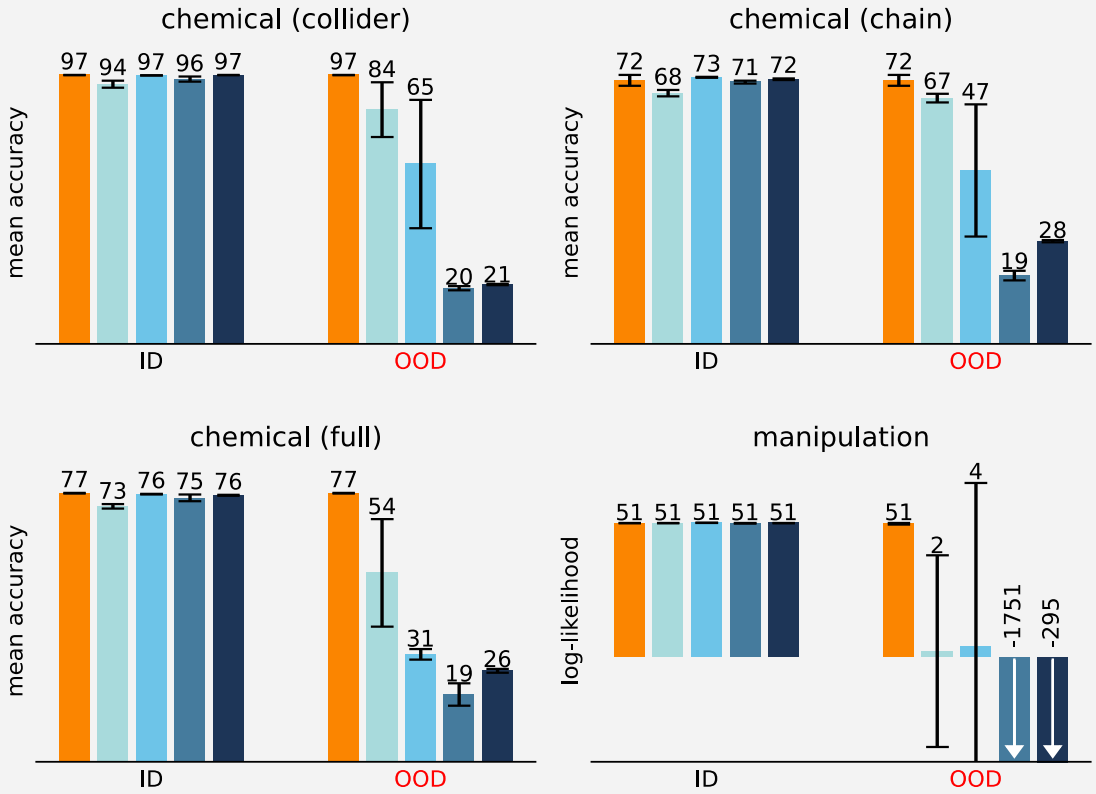
Dynamics Generalization

Causal dynamics generalizes best in unseen states.

- Causal Dynamics Learning (Ours)
- Regularization
- Graph Neural Network
- Modular
- Monolithic

ID: in-distribution states

OOD: out-of-distribution states



Results

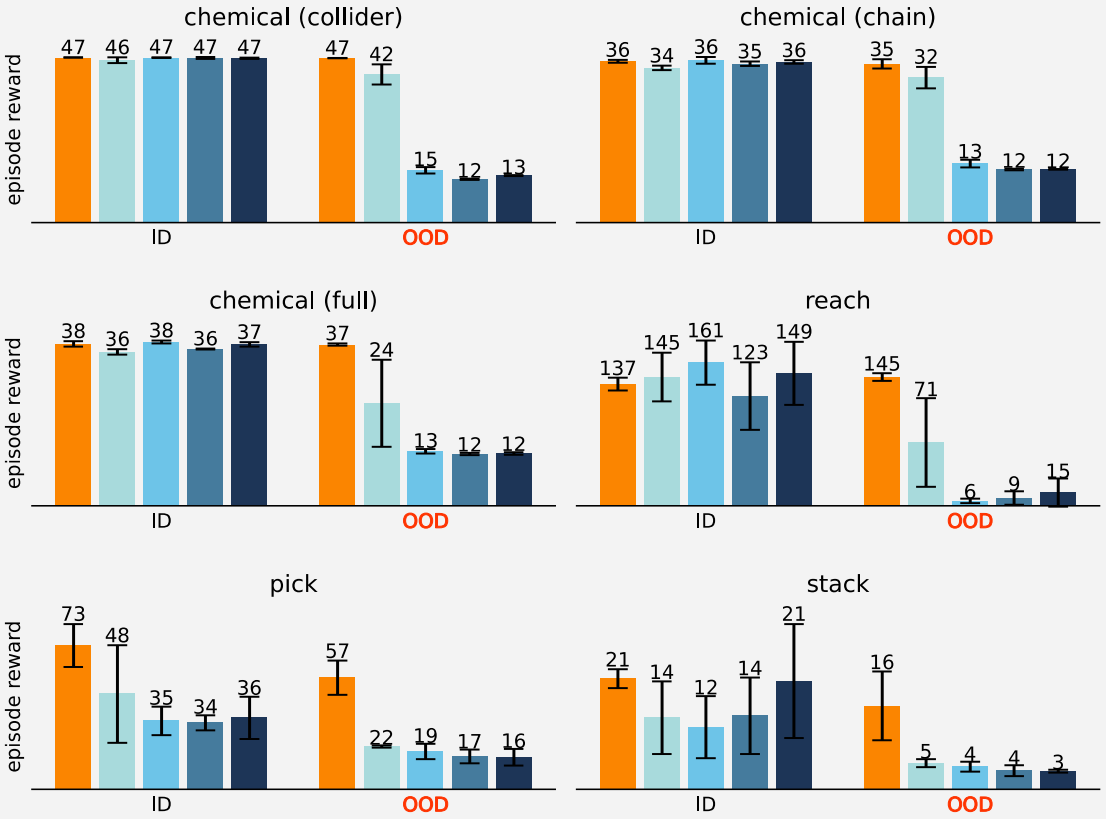
Task Generalization

Causal dynamics generalizes best in unseen states.

- Causal Dynamics Learning (Ours)
- Regularization
- Graph Neural Network
- Modular
- Monolithic

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

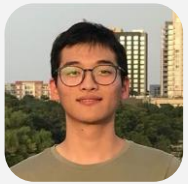
Causal dependencies are learned globally only.

- 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

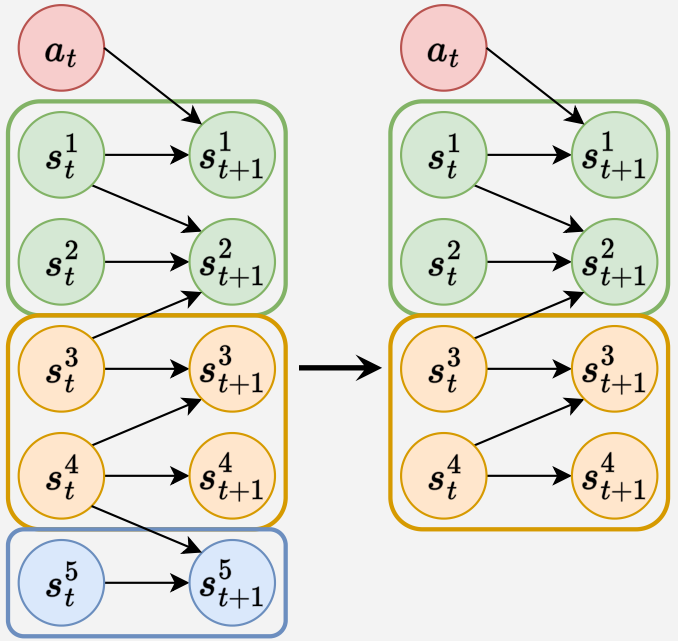
LARG Learning Agents Research Group The University of Texas at Austin | RPL UT Robot Perception & Learning Lab | Sony AI

Problem Setup

CDL's **state abstraction** omits action-irrelevant variables.
What tasks can this state abstraction solve?

- ✓ Tasks whose rewards are defined by controllable and action-relevant state variables
- ✗ Tasks with rewards involving action-irrelevant state variables

Solving any task (learning any reward) means no abstraction.

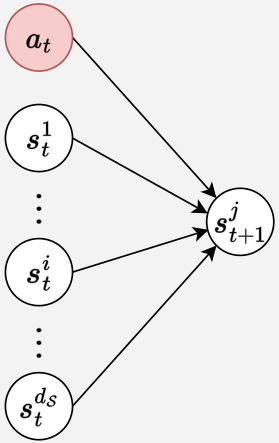


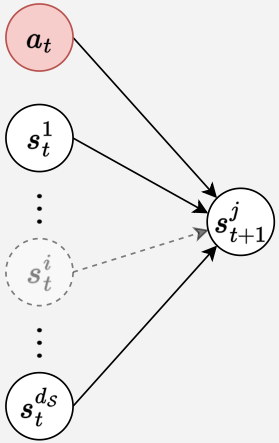
Method

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

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

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





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

$$\Updownarrow$$

$$\text{CMI}^{ij} = \mathbb{E}[\log \frac{p(s_{t+1}^j | s_t, a_t)}{p(s_{t+1}^j | \{s / s^i\}_t, a_t)}] \geq \epsilon$$