Counterexample-Guided Strategy Improvement for POMDPs Using Recurrent Neural Networks (IJCAI'19)

Weizhi Feng

January 5, 2021

Outline

- Motivation & Contribution
- Formal Problem Statement
- Synthesis Procedure
- Experimental Results
- Conclusion

Motivation

- Autonomous agents that make decisions under uncertainty and incomplete information can be mathematically represented as POMDPs.

- It obtains observations and infers the likelihood of the system being in a certain state, known as the belief state.

- Traditional POMDP problems typically seek to compute a strategy that maximizes a cumulative reward over a finite horizon.

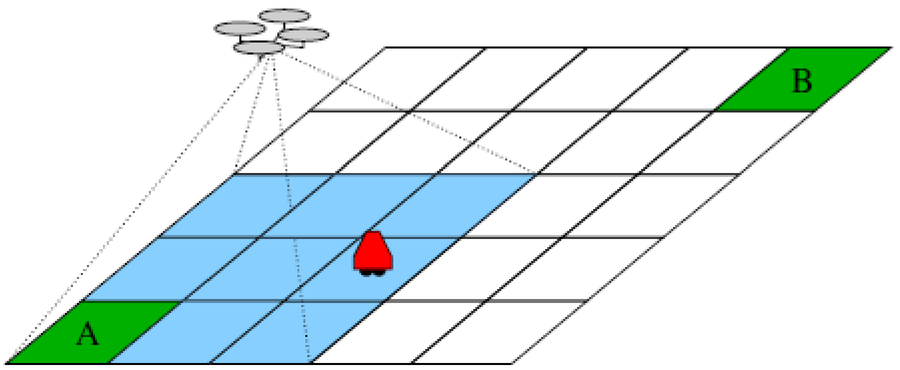
- But, the agent's behavior is often required to obey more complicated specifications.

Motivation

- Strategy synthesis for POMDPs is a difficult problem.

- Example – drone surveillance
  - The UAV wants to survey regions labeled with A and B, while avoiding the ground agent.
  - LTL formula:

$$\square \lozenge A \;\wedge\; \square \lozenge B \;\wedge\; \square \neg \mathsf{Detected}.$$



Key questions

- How to generate a good strategy in the first place;
- How to improve a strategy if verification refutes the specification.

- Machine learning and formal verification techniques address these questions separately.

Contribution

- This paper propose a novel method that combines from machine learning and formal verification to handle strategy synthesis problem.

- 1) They train RNN (Recurrent Neural Network) to encode POMDP strategies.

- 2) They restrict the RNN-based strategy on a specific POMDP. For the resulting finite Markov chain, formal verification provides guarantees against temporal logic specifications.

- 3) If not satisfied, counterexample supply diagnostic information. The information is then used to improve the strategy by iteratively training the RNN.

Preliminaries – (PO)MDPs

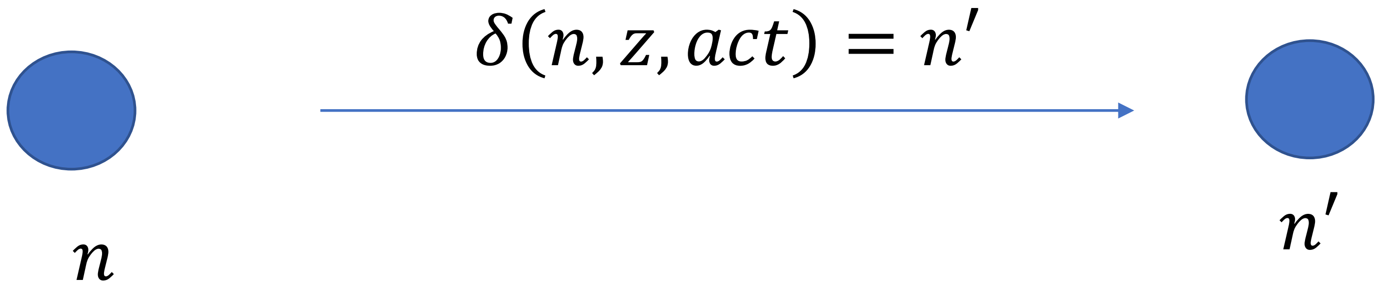
- MDP: $M = (S, Act, P, r)$
- $P: S \times Act \rightarrow Distr(S)$
- $r: S \times a \rightarrow \mathbb{R}$
- A finite *path* π is a sequence of states and actions.
- The set of finite paths of M is $Paths_{fin}^{M}$.
- A strategy γ for an MDP M is a function:
- $\gamma: Paths_{fin}^{M} \rightarrow Distr(Act)$

Preliminaries – (PO)MDPs

- $POMDP: M = (M, Z, O)$
- Z: a finite set of observations;
- $O: S \times Z$ the observation function;
- $ObsSeq_{fin}^{M}$: the set of all finite observation-action sequences for a POMDP.
  - $z_0 \xrightarrow{a_0} z_1 \dots z_n$
- POMDP Strategy:
  - A function $\gamma \in \Gamma_z^M: ObsSeq_{fin}^{M} \rightarrow Distr(Act)$.

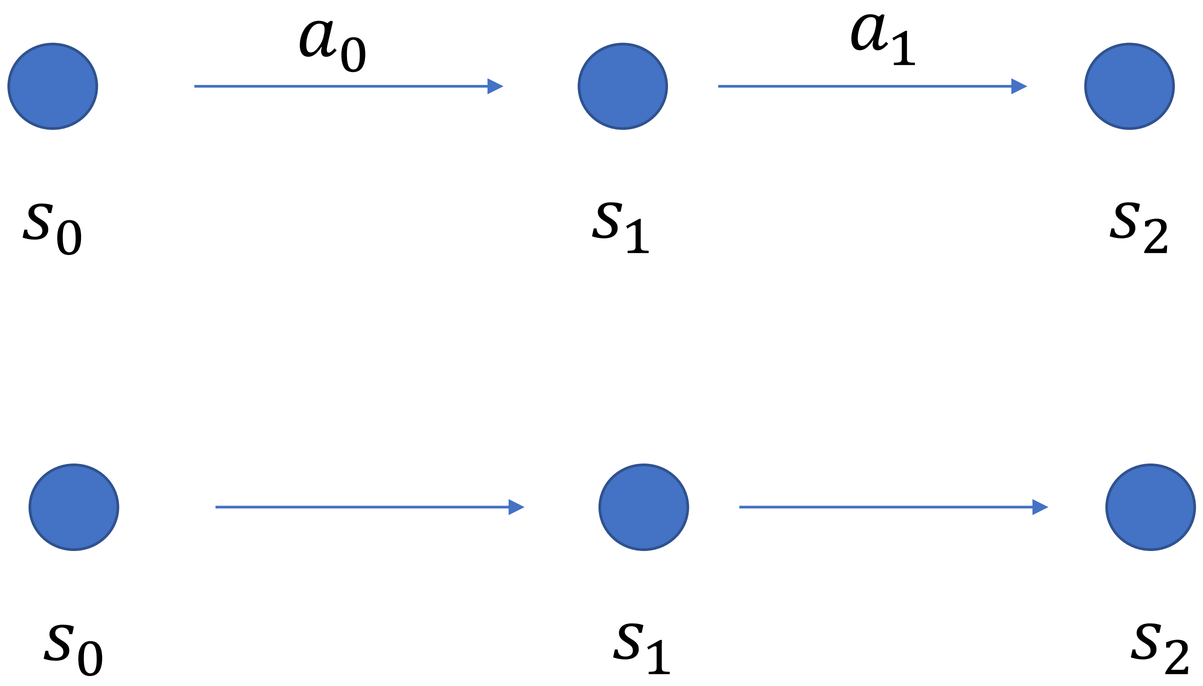
Preliminaries – FSC (Finite-State Controllers)

- $A\ k - FSC$ for a POMDP is a tuple $A = (N, n_I, \gamma_\alpha, \delta)$ where N is a finite set of k memory nodes.



Preliminaries – Specifications

- If φ is satisfied in a POMDP M under γ, we write $M^{\gamma} \vDash \varphi$, that is, the specification is satisfied in the induced MC.



Fix a strategy γ

Induced MC

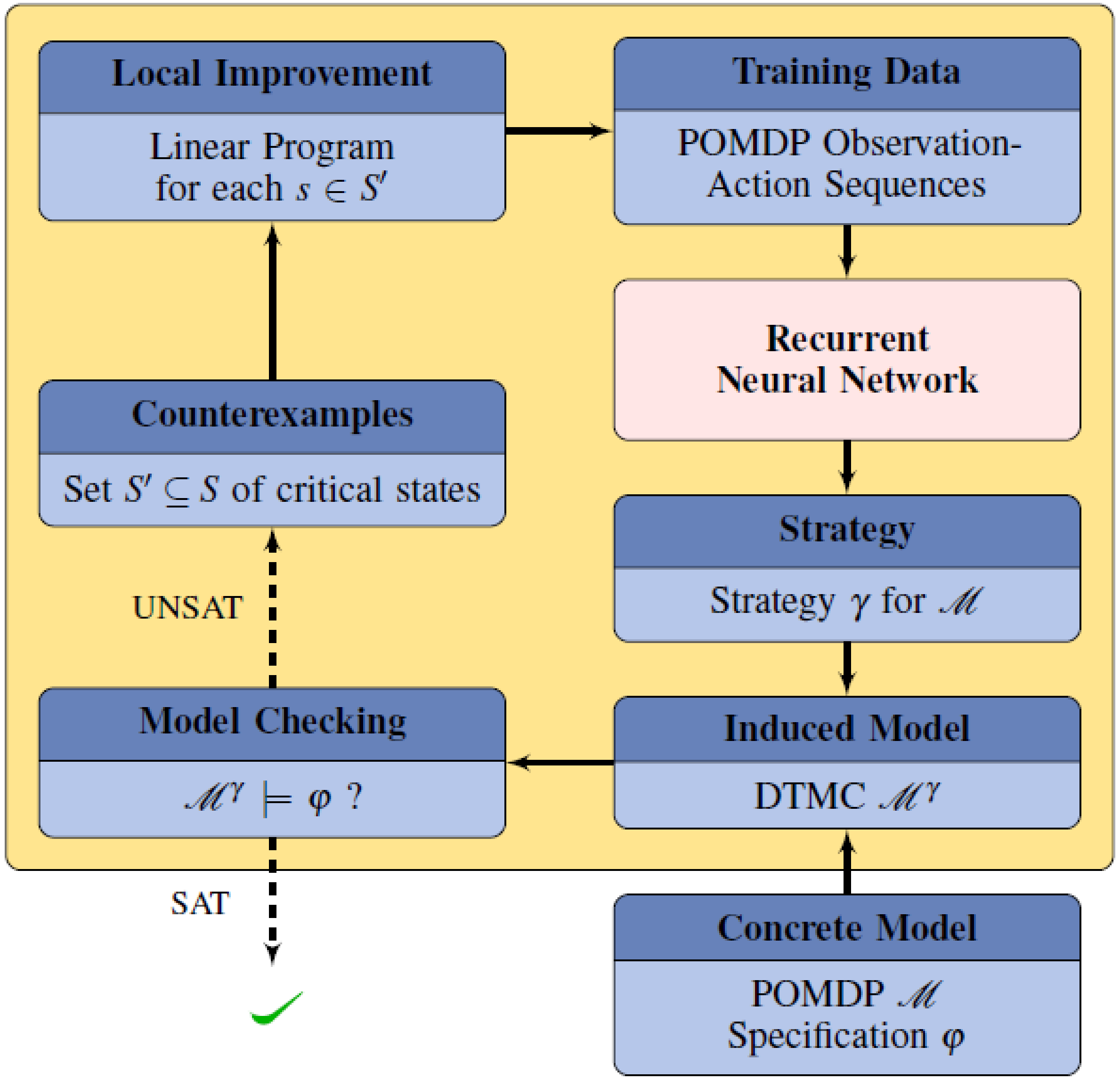
Formal Problem Statement

- For a POMDP M and a specification φ, where either $\varphi = \mathbb{P}_{\sim\lambda}(\psi)$ with ψ an LTL formula, or $\varphi = \mathbb{E}_{\sim\lambda}(\diamond\, a)$, the problem is to determine a finite-memory strategy $\gamma \in \Gamma_z^M$ such that $M^\gamma \vDash \varphi$.

- $\varphi = \mathbb{P}_{\sim\lambda}(\psi)$: the probability of satisfying an LTL-property respects a given bound.
- $\mathbb{E}_{\sim\lambda}(\diamond\, a)$: undiscounted expected reward properties, require that the expected accumulated cost until reaching a state satisfying a.

Workflow

- Flowchart of the RNN-based refinement loop:
- 1) train an RNN using observation-action sequences generated from an initial strategy.
- 2) the strategy network extract a strategy and we obtain the induced MC.
- 3) Model checking of this MC evaluates whether the φ is satisfied or not.



Preliminaries : RNN and LSTM

- Recurrent Neural Network
  - In traditional NN, it is assumed that every input is independent each other.
  - But with sequential data, input in current time step is highly likely depends on input in previous time step.
  - We need some additional structure that can model dependencies of inputs over time.
  - Given fixed input and target from data, RNN is to learn intermediate association between them and also the real-valued vector representation.

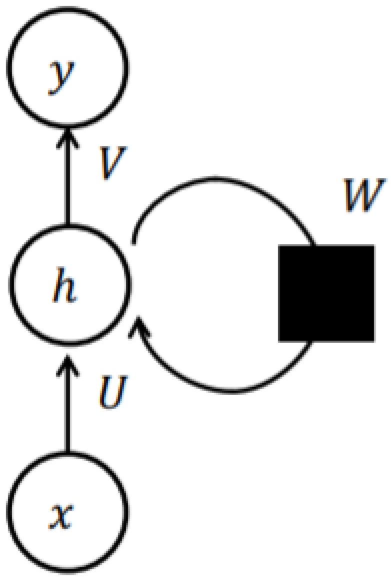
Preliminaries : RNN and LSTM

- Recurrent Neural Network
  - Input, output and internal representation (hidden states):
  - x_t: input vector;
  - $\hat{y}$: output vector;
  - h_t: hidden states;
  - (U, W, V): parameter matrices;

  - $h_t = \tanh(Ux_t + Wh_{t-1})$.
  - $\hat{y} = \lambda(Vht)$.

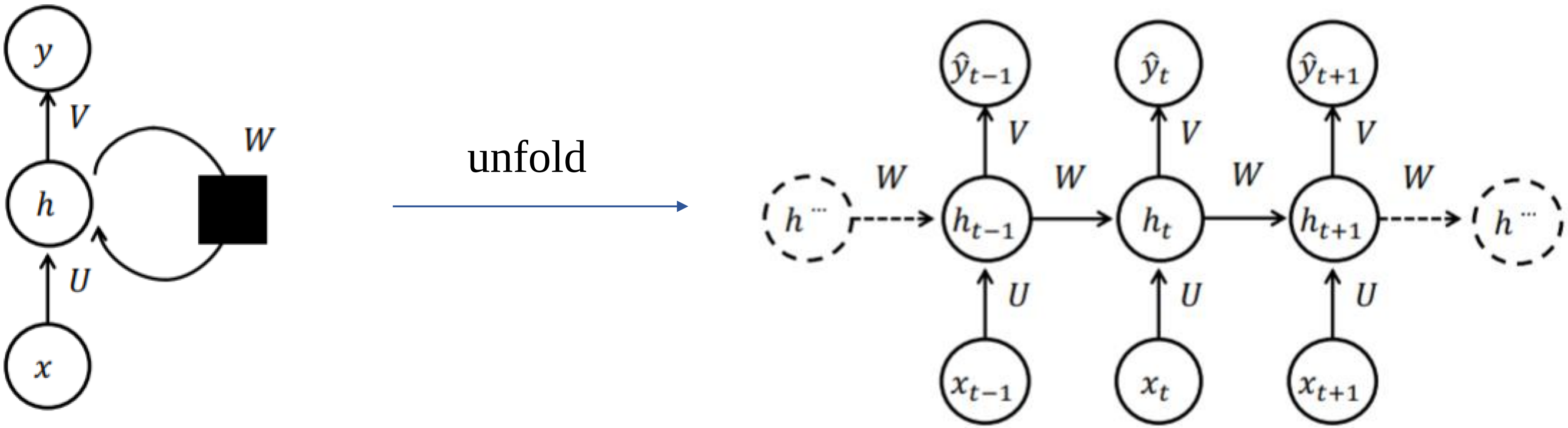
Preliminaries : RNN and LSTM

- Recurrent Neural Network
  - A type of a neural network that has a recurrence structure.
  - The recurrence structure allows us to operate over a sequence of vectors.



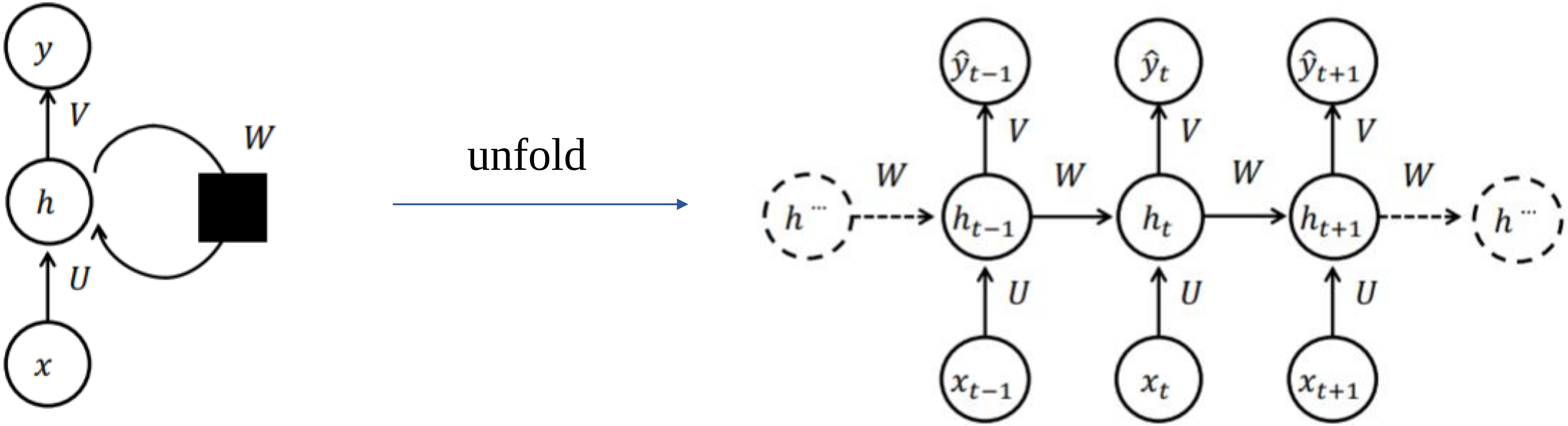
Preliminaries : RNN and LSTM

- Recurrent Neural Network
  - A type of a neural network that has a recurrence structure.
  - The recurrence structure allows us to operate over a sequence of vectors.



Preliminaries : RNN and LSTM

- Recurrent Neural Network
  - $h_t = \tanh(Ux_t + Wh_{t-1})$.
  - $\hat{y} = \lambda(Vht)$.



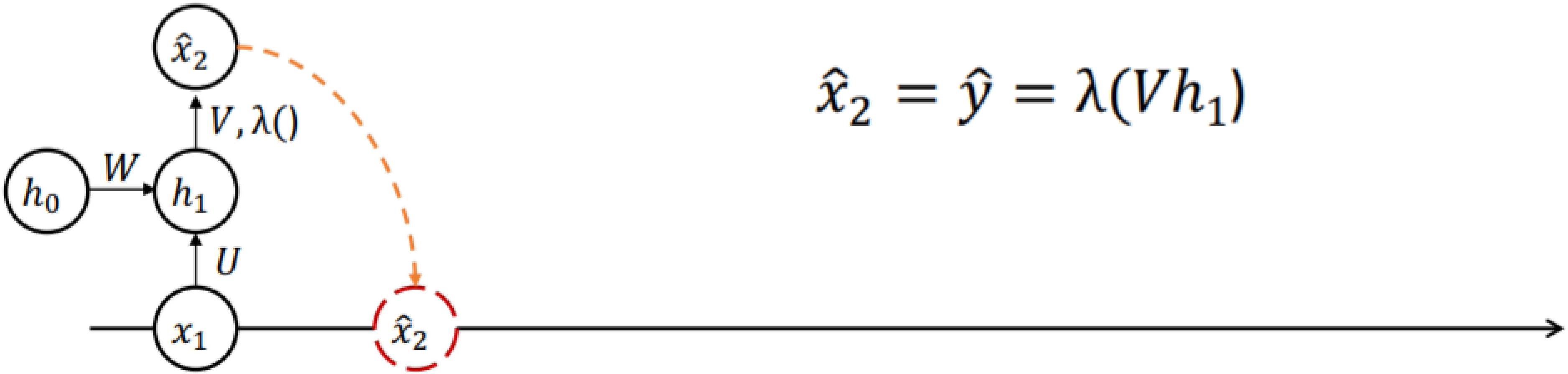
Preliminaries : RNN and LSTM

- Make a prediction
  - Initial hidden state h_0.
  - Assume we currently have observation x_1 and want to predict x_2.
  - First we compute hidden states h_1.

$$h_1 = \tanh(Ux_1 + Wh_0)$$

Preliminaries : RNN and LSTM

- Make a prediction
  - Then we generate prediction: $\widehat{x_2} = \hat{y}$.

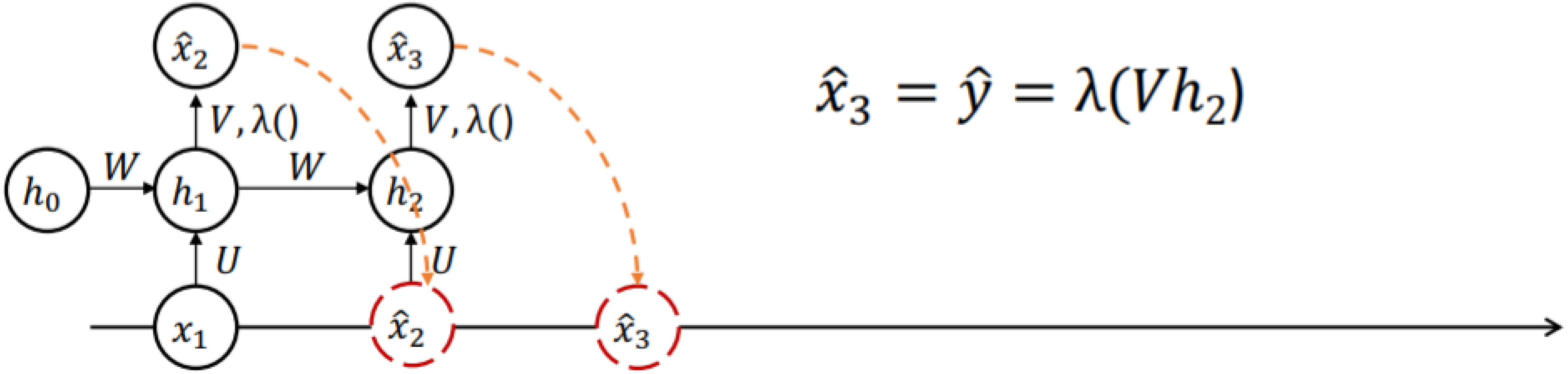


$$h_1 = \tanh(Ux_1 + Wh_0)$$

$$\hat{x}_2 = \hat{y} = \lambda(Vh_1)$$

Preliminaries : RNN and LSTM

- Make a prediction – multiple steps
  - Predicted value $\widehat{x_2}$ from previous step is considered as input x_2 at time step 2.

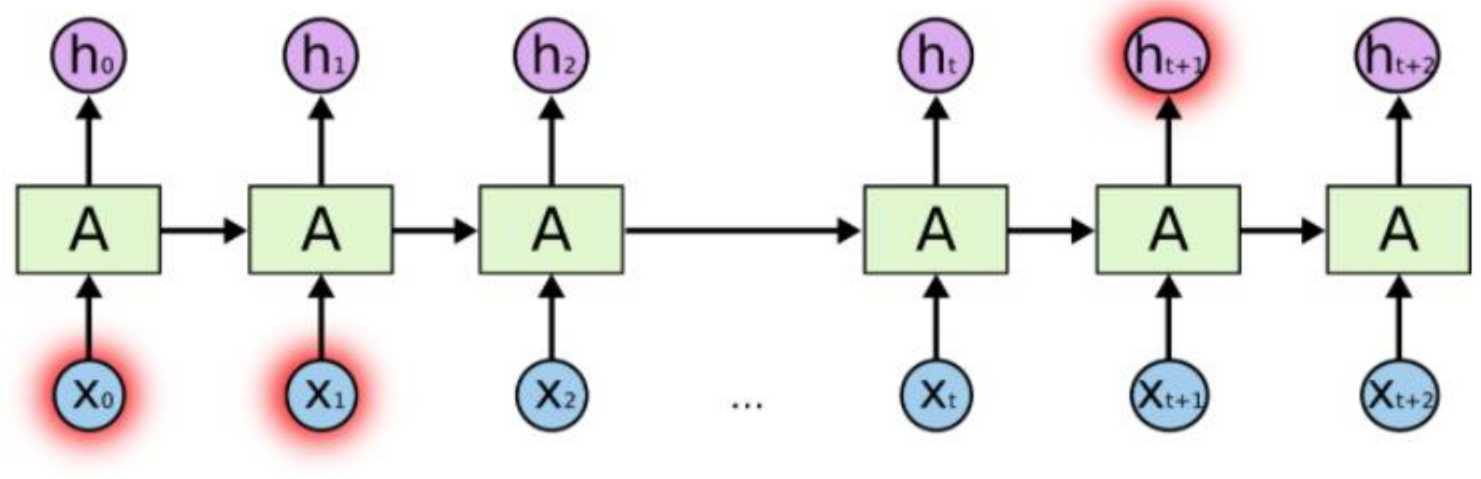


$$h_2 = \tanh(U\hat{x}_2 + Wh_1)$$

$$\hat{x}_3 = \hat{y} = \lambda(Vh_2)$$

Preliminaries : RNN and LSTM

- Problem of Long-Term Dependencies
  - Sometimes we need long-term information.
  - Example:
    - “ I grew up in France… I speak fluent *French*.”



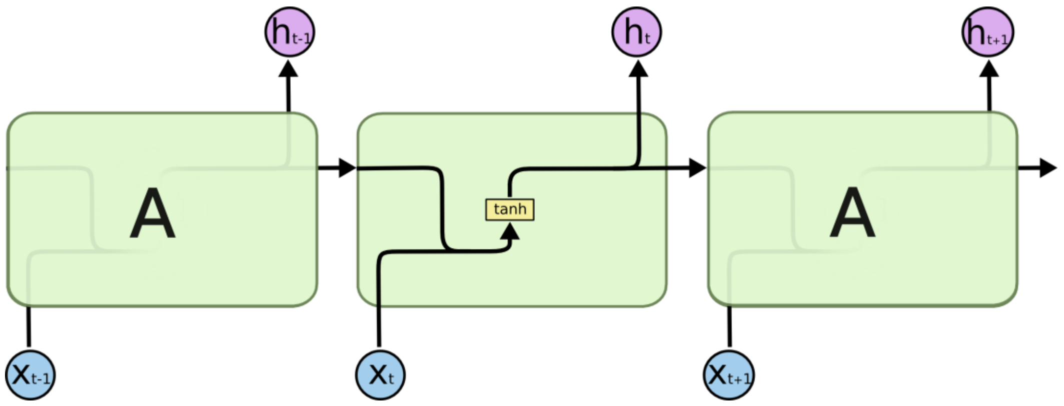
Preliminaries : RNN and LSTM

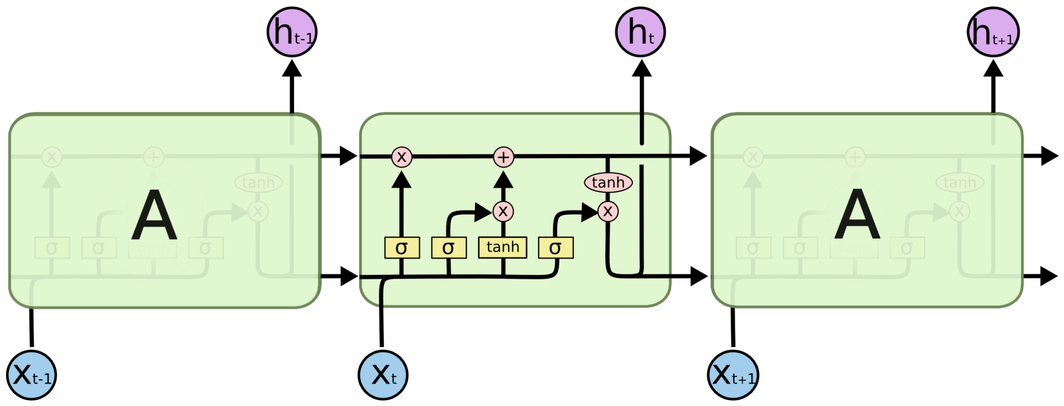
- Gating mechanism
  - Add gates to produce paths where gradients can flow more constantly in longer-term without vanishing nor exploding.

- Long Short-term Memory (LSTM)
  - Memory cells;
  - Gates.

Preliminaries : RNN and LSTM

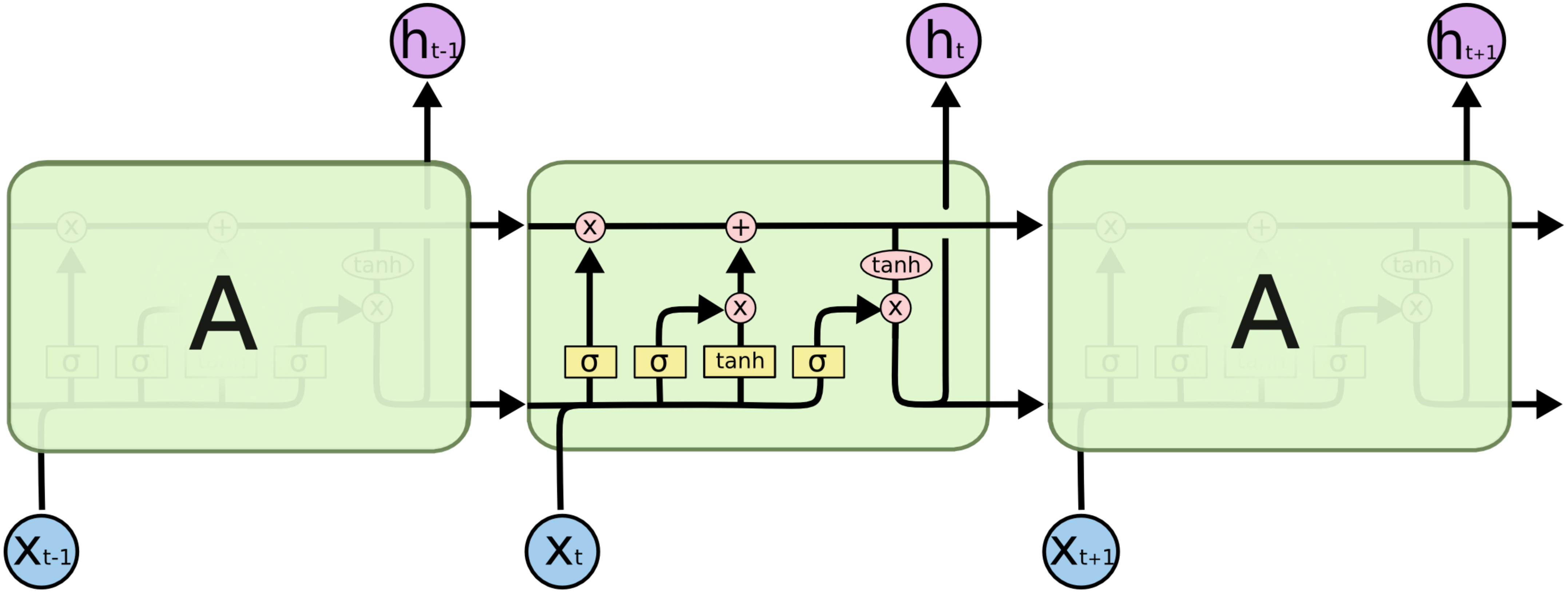
- Long Short-term Memory (LSTM)
  - RNN:
    - $h_t = \tanh(Ux_t + Wh_{t-1})$
  - LSTM:
    - Instead of having a single NN layer, there are four, interacting in a very special way.





Preliminaries : RNN and LSTM

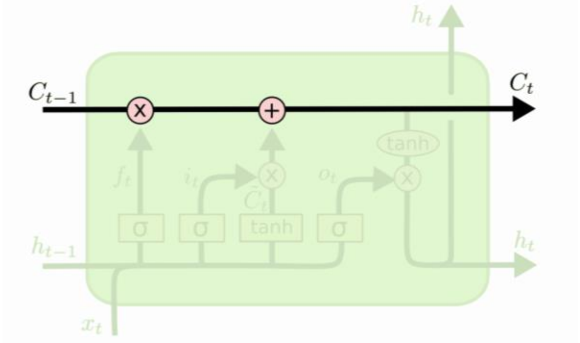
- Long Short-term Memory (LSTM)



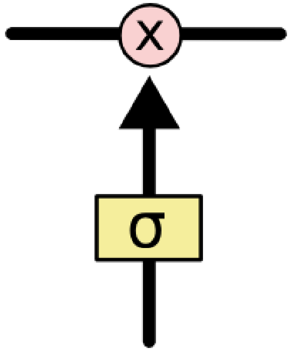
Preliminaries : RNN and LSTM

- Long Short-term Memory (LSTM)
  - The horizontal line:

    \- It is easy for information to just flow along it unchanged.

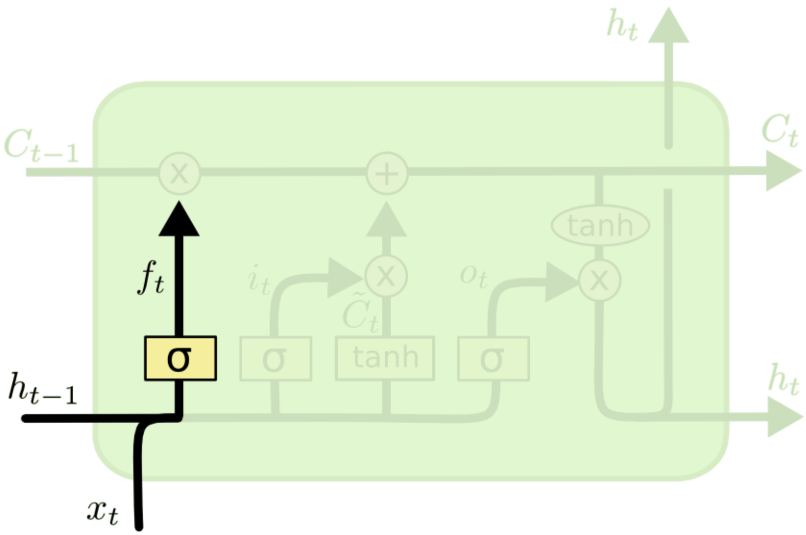


  - Gates are used to remove or add information to the cell state.



Preliminaries : RNN and LSTM

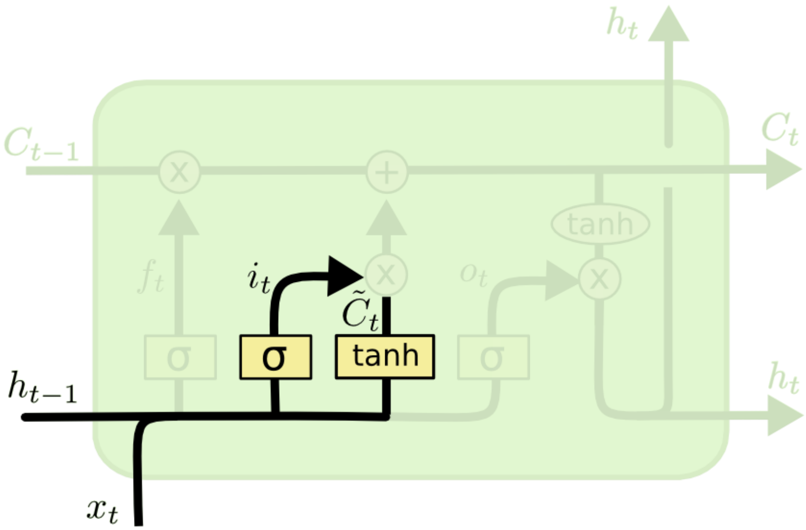
- Long Short-term Memory (LSTM)
  - The “forget gate layer”;
    - Output a number between 0 and 1;



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Preliminaries : RNN and LSTM

- Long Short-term Memory (LSTM)
  - The “input gate layer” and a tanh layer.
    - Decide what new information we’re going to store in the cell state.

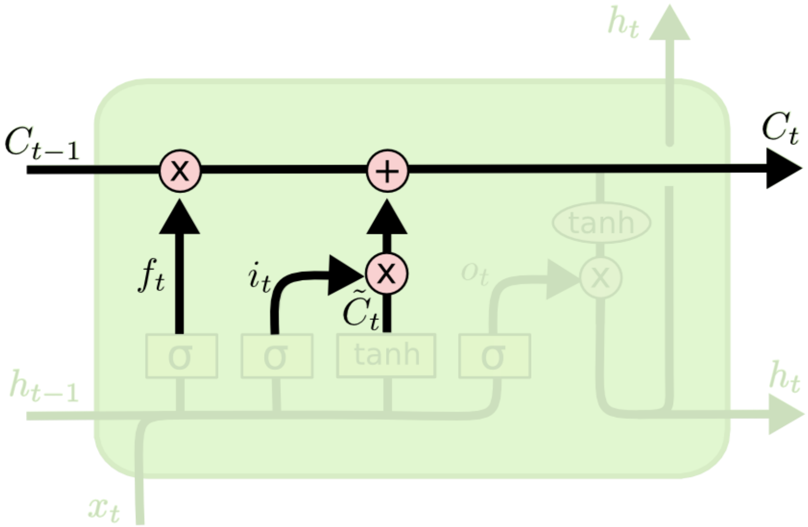


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Preliminaries : RNN and LSTM

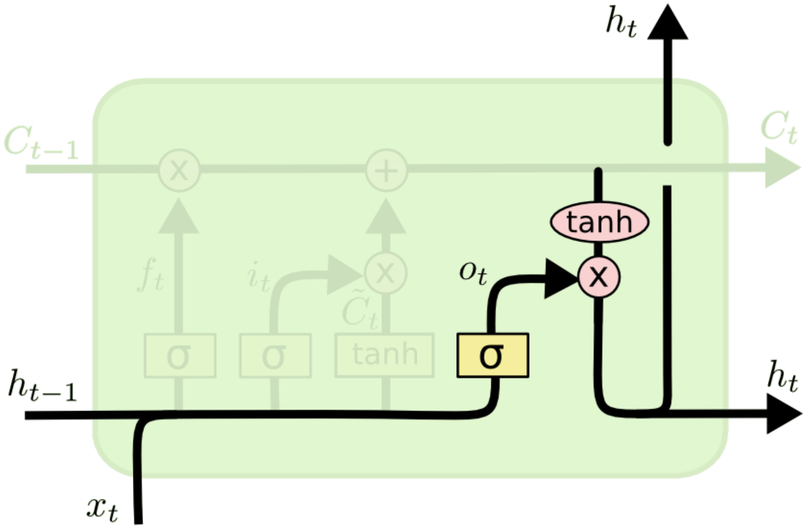
- Long Short-term Memory (LSTM)
  - The "input gate layer" and a tanh layer.
    - Decide what new information we're going to store in the cell state.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Preliminaries : RNN and LSTM

- Long Short-term Memory (LSTM)
  - The “output gate layer” and a tanh layer.
    - Decide what we are going to output.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Learning Strategies with RNNs

- Policy gradient algorithms are not well suited for POMDPs.
- Constructing the strategy network. (LSTM architecture)
  - $\hat{\gamma}: ObsSeq_{fin}^{M} \rightarrow Distr(Act)$.
  - For a given observation-action sequence from $ObsSeq_{fin}^{M}$, the model learns a strategy $\hat{\gamma}$. The output is a discrete probability distribution over the actions Act.

RNN training

- For a POMDP M and a specification φ:
- First compute a strategy γ of the underlying MDP M that satisfies φ.
- Then sample uniformly over all states of the MDP and generate finite paths of the induced MC M^{γ}, thereby creating multiple trajectory trees.

Strategy

induced MC

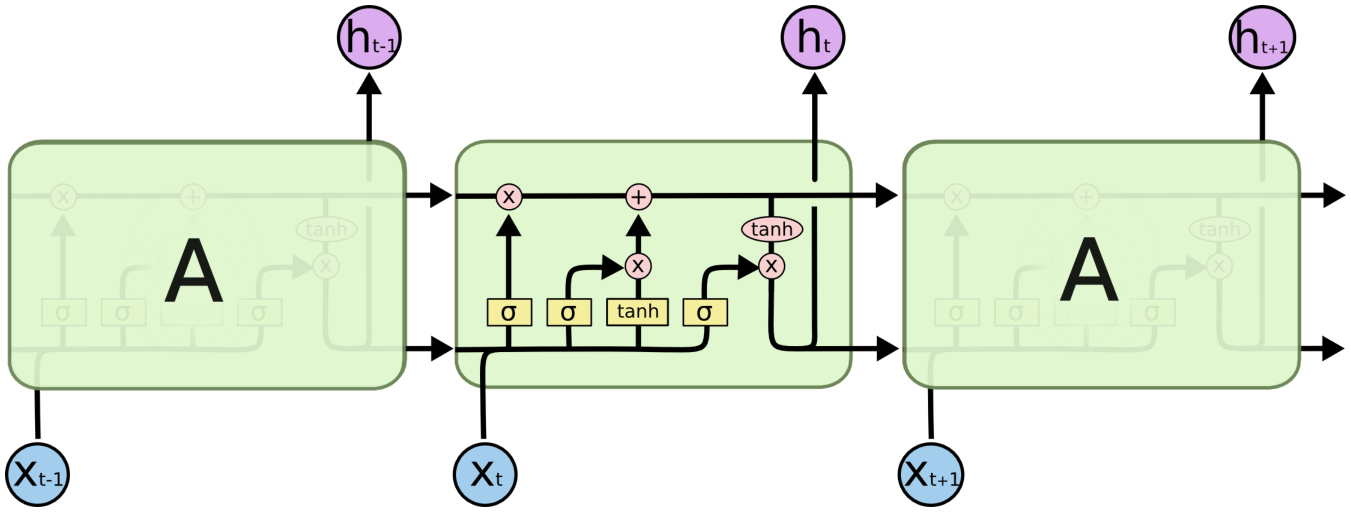
RNN training

- For a POMDP M and a specification φ:
- First compute a strategy γ of the underlying MDP M that satisfies φ.
- Then sample uniformly over all states of the MDP and generate finite paths of the induced MC M^{γ}, thereby creating multiple trajectory trees.
- For each finite path π, generate one possible observation-action sequence:$\pi_z = z_0, a_0, z_1, a_1, \dots., a_{n-1}, z_n, z_i = O(\pi[i])$.
- Form the training set D from a number of m observation-action sequences with observations as input and actions ad output labels.

Strategy Extraction and Evaluation

- How to extract a memoryless strategy from the strategy network.
  - Given a POMDP M, we use the trained strategy network $\hat{\gamma}: ObsSeq_{fin}^{M} \to Distr(Act)$ directly as observation-based strategy.

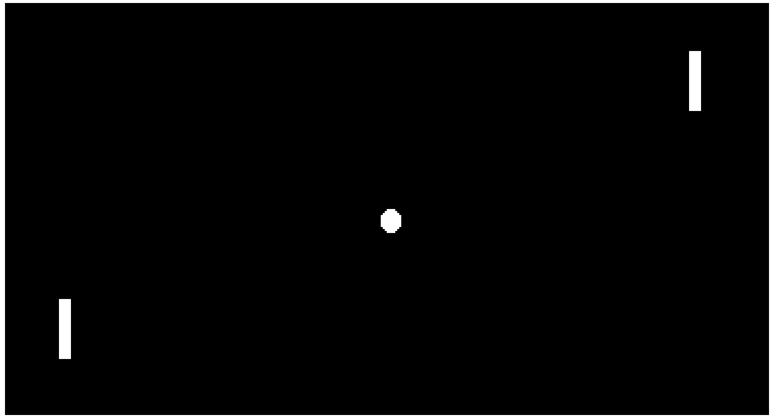
$actions$



obs_i

Extension to FSCs (N, n_I, γ, δ)

- LTL specifications as well as observation-dependencies in POMDPs require memory.
- Assume the FSC is in memory node n_k at position i of π_z, we define $\delta(n_k, z_i, a_i) = n_{k+1}$, if $\pi_z[i] = (z_i, a_i)$.



$$n_k \xrightarrow{\delta(n, z, act) = n_{k+1}} n_{k+1}$$

Extension to FSCs (N, n_I, γ, δ)

- Once δ has been defined, we compute a product POMDP $M \times A$.
- Generate observation-node-action sequence: $(z_0, n_0), a_0, \ldots$
- In this case, the RNN is learning the mapping of observation and memory node to the Distr(Act) as an FSC strategy network.

Improving the Represented Strategy

- A local improvement for a strategy that does not satisfy the specification.
- POMDP M, φ, the strategy γ and $M^{\gamma} \not\models \varphi$.
- Create diagnostic information on why the specification φ is not satisfied.

Improving the Represented Strategy

- Critical Decision
  - $\varphi = \mathbb{P}_{\leq\lambda}(\psi)$, for some threshold $\lambda' \in [0,1]$, a state s is critical iff $Pr^*(s) > \lambda'$.
  - The set of critical decision serves as a counterexample, generated by the set of critical states and the strategy γ.
- For each observation with a critical decision, we construct an optimization problem that minimizes the number of critical actions the strategy chooses per observation class.

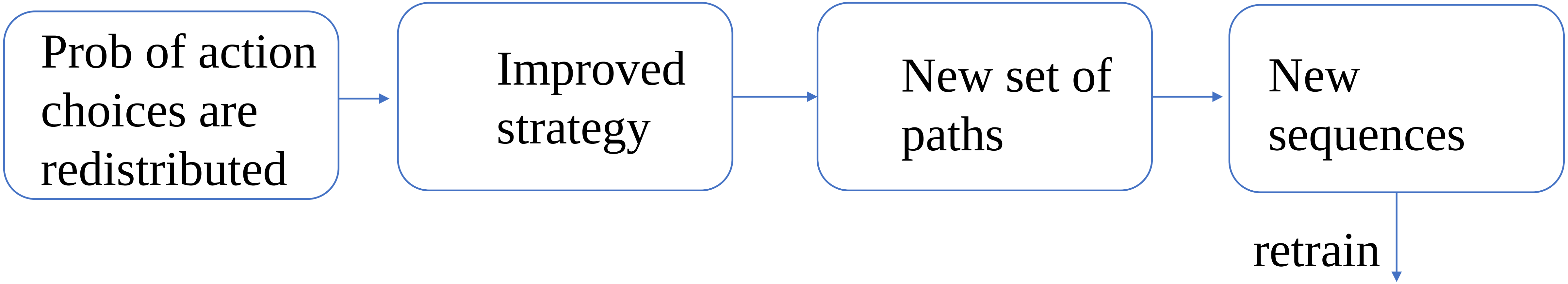
$$\max_{\gamma(z)(a), a \in Act} \min_{s \in S} p_s \tag{1}$$

subject to

$$\forall s \in O^{-1}(z). \quad p_s = \sum_{a \in Act} \gamma(z)(a) \cdot \sum_{s' \in S} \mathscr{P}(s,a,s') \cdot p^*(s')$$

Improving the Represented Strategy

- The probabilities of action choices under γ are redistributed such that the critical choices are minimized.
- From the resulting improved strategy, we generate a new set of paths starting from the critical states.



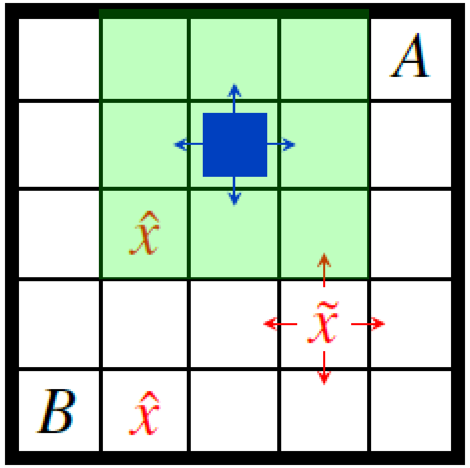
Implementation

- PRISM
  - LTL model checking

- STORM
  - undiscounted expected rewards

Experimental results

- PRISM-POMDP
- The point-based solver SolverPOMDP



(a)

Navigation with moving obstacles – an agent and a single stochastically moving obstacle. The agent task is to maximize the probability to navigate to a goal state A while not colliding with obstacles (both static and moving): $\varphi_1 = \mathbb{P}_{\max}(\neg X \cup A)$ with $x = \hat{x} \cup \tilde{x}$,

Delivery without obstacles – an agent and static objects (landmarks). The task is to deliver an object from A to B in as few steps as possible: $\varphi_2 = \mathbb{E}_{\min}(\Diamond(A \wedge \Diamond B))$.

Slippery delivery with static obstacles – an agent where the probability of moving perpendicular to the desired direction is 0.1 in each orientation. The task is to maximize the probability to go back and forth from locations A and B without colliding with the static obstacles $\hat{x}$: $\varphi_3 = \mathbb{P}_{\max}(\Box\Diamond A \wedge \Box\Diamond B \wedge \neg\Diamond X)$, with $x = \hat{x}$,

Experimental results

- PRISM-POMDP

			RNN-based Synthesis		PRISM-POMDP	
Problem	States	Type, φ	Res.	Time (s)	Res.	Time (s)
Navigation (3)	333	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.74	**14.16**	**0.84**	73.88
Navigation (4)	1088	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.82	**22.67**	**0.93**†	1034.64
Navigation (4) [2-FSC]	13373	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.91	47.26	–	–
Navigation (4) [4-FSC]	26741	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.92	59.42	–	–
Navigation (4) [8-FSC]	53477	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	**0.92**	85.26	–	–
Navigation (5)	2725	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.91	**34.34**	MO	MO
Navigation (5) [2-FSC]	33357	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.92	115.16	–	–
Navigation (5) [4-FSC]	66709	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.92	159.61	–	–
Navigation (5) [8-FSC]	133413	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	**0.92**	250.91	–	–
Navigation (10)	49060	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.79	**822.87**	MO	MO
Navigation (10) [2-FSC]	475053	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.83	1185.41	–	–
Navigation (10) [4-FSC]	950101	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	**0.85**	1488.77	–	–
Navigation (10) [8-FSC]	1900197	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.81	1805.22	–	–
Navigation (15)	251965	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	**0.91**	**1271.80***	MO	MO
Navigation (20)	798040	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	**0.96**	**4712.25***	MO	MO
Navigation (30)	4045840	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	**0.95**	**25191.05***	MO	MO
Navigation (40)	–	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	TO	TO	MO	MO
Delivery (4) [2-FSC]	80	$\mathbb{E}^{\mathcal{M}}_{\min}, \varphi_2$	6.02	35.35	**6.0**	**28.53**
Delivery (5) [2-FSC]	125	$\mathbb{E}^{\mathcal{M}}_{\min}, \varphi_2$	8.11	**78.32**	**8.0**	102.41
Delivery (10) [2-FSC]	500	$\mathbb{E}^{\mathcal{M}}_{\min}, \varphi_2$	**18.13**	**120.34**	MO	MO
Slippery (4) [2-FSC]	460	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_3$	0.78	67.51	**0.90**	**5.10**
Slippery (5) [2-FSC]	730	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_3$	0.89	84.32	**0.93**	**83.24**
Slippery (10) [2-FSC]	2980	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_3$	**0.98**	**119.14**	MO	MO
Slippery (20) [2-FSC]	11980	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_3$	**0.99**	**1580.42**	MO	MO

Experimental results

- PRISM-POMDP
- The point-based solver SolverPOMDP

Problem	$\lvert S\rvert$	$\lvert Act\rvert$	$\lvert Z\rvert$
Navigation (c)	c^4	4	256
Delivery (c)	c^2	4	256
Slippery (c)	c^2	4	256
Maze(c)	$3c+8$	4	7
Grid(c)	c^2	4	2
RockSample[4,4]	257	9	2
RockSample[5,5]	801	10	2
RockSample[7,8]	12545	13	2

Problem	Type	RNN-based Synthesis States	RNN-based Synthesis Res	RNN-based Synthesis Time (s)	PRISM-POMDP Res	PRISM-POMDP Time (s)	pomdpSolve Res	pomdpSolve Time (s)
Maze (1)	$\mathbb{E}^{\mathcal{M}}_{\min}$	68	4.31	31.70	**4.30**	**0.09**	4.30	0.30
Maze (2)	$\mathbb{E}^{\mathcal{M}}_{\min}$	83	5.31	46.65	5.23	2.176	**5.23**	**0.67**
Maze (3)	$\mathbb{E}^{\mathcal{M}}_{\min}$	98	8.10	58.75	7.13	38.82	**7.13**	**2.39**
Maze (4)	$\mathbb{E}^{\mathcal{M}}_{\min}$	113	11.53	58.09	8.58	543.06	**8.58**	**7.15**
Maze (5)	$\mathbb{E}^{\mathcal{M}}_{\min}$	128	14.40	**68.09**	13.00†	4110.50	**12.04**	132.12
Maze (6)	$\mathbb{E}^{\mathcal{M}}_{\min}$	143	22.34	**71.89**	MO	MO	**18.52**	1546.02
Maze (10)	$\mathbb{E}^{\mathcal{M}}_{\min}$	203	100.21	**158.33**	MO	MO	MO	MO
Grid (3)	$\mathbb{E}^{\mathcal{M}}_{\min}$	165	2.90	38.94	2.88	2.332	**2.88**	**0.07**
Grid (4)	$\mathbb{E}^{\mathcal{M}}_{\min}$	381	4.32	79.99	4.13	1032.53	**4.13**	**0.77**
Grid (5)	$\mathbb{E}^{\mathcal{M}}_{\min}$	727	6.62	91.42	MO	MO	**5.42**	**1.94**
Grid (10)	$\mathbb{E}^{\mathcal{M}}_{\min}$	5457	**13.63**	**268.40**	MO	MO	MO	MO
RockSample[4,4]	$\mathbb{E}^{\mathcal{M}}_{\max}$	2432	17.71	35.35	N/A	N/A	**18.04**	**0.43**
RockSample[5,5]	$\mathbb{E}^{\mathcal{M}}_{\max}$	8320	18.40	**43.74**	N/A	N/A	**19.23**	621.28
RockSample[7,8]	$\mathbb{E}^{\mathcal{M}}_{\max}$	166656	20.32	**860.53**	N/A	N/A	**21.64†**	20458.41