Toward Interpretable Deep Reinforcement Learning with Linear Model U-Trees

Guiliang Liu, Oliver Schulte, Wang Zhu and Qingcan Li
Simon Fraser University, BC, Canada

SIMON FRASER UNIVERSITY

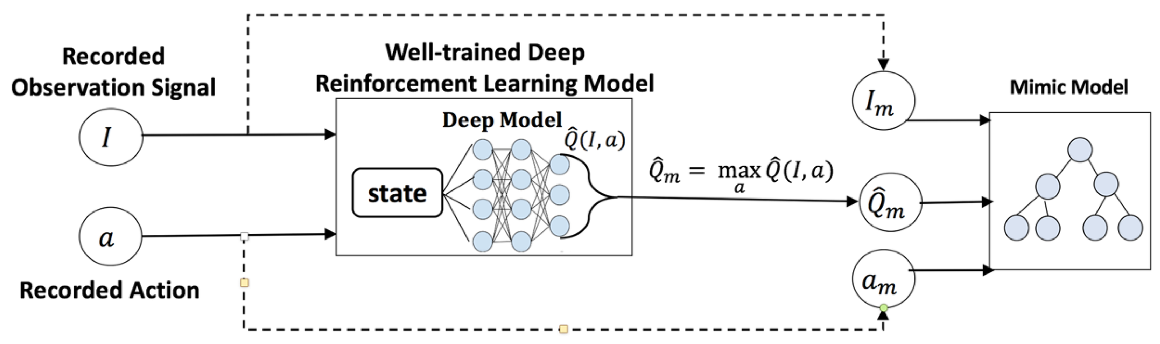
ECML PKDD Dublin, Ireland 10-14 SEPT 2018

Overview

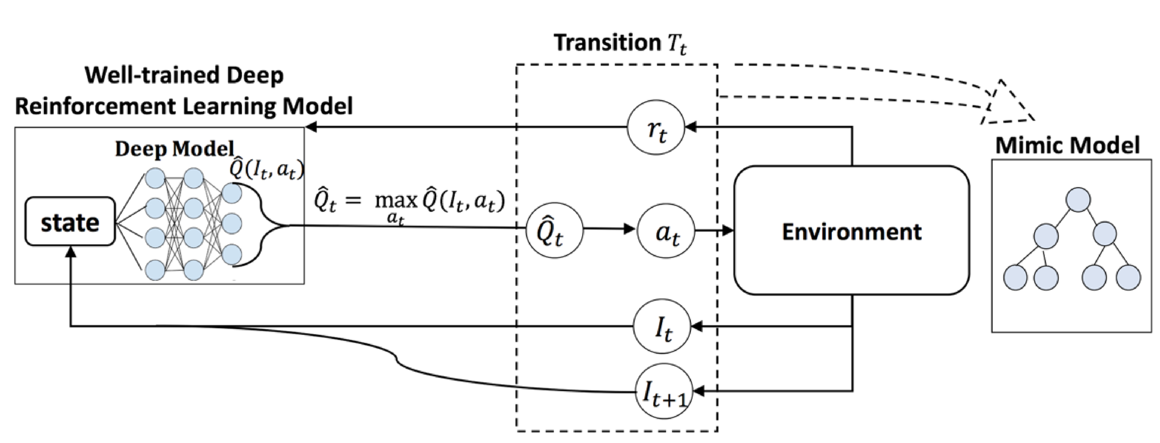
- **Goal**: Understand the knowledge learned by Deep Reinforcement Learning (DRL) Model.
- **Motivation:** Despite excellent performance of DRL, the knowledge remains implicit in neural networks.
- **Our Work:** The contribution of our work includes:
  - To our best knowledge, the first work that extends interpretable mimic learning to Reinforcement Learning.
  - We define the on-line learning algorithm for LMUT, which is a novel model tree to mimic a DRL model.
  - We show how to interpret a DRL model by analyzing the knowledge stored in the tree structure of LMUT.

Mimic learning for Deep Reinforcement Learning

- **Experience Training** Setting:
  - Recording observation signals I and actions a during the training process of Deep Reinforcement Learning (DRL).
  - Input them to mature DRL model and obtain soft output $\hat{Q}(I, a)$.
  - Generates samples for Experience Training Dataset *(for batch training)*.

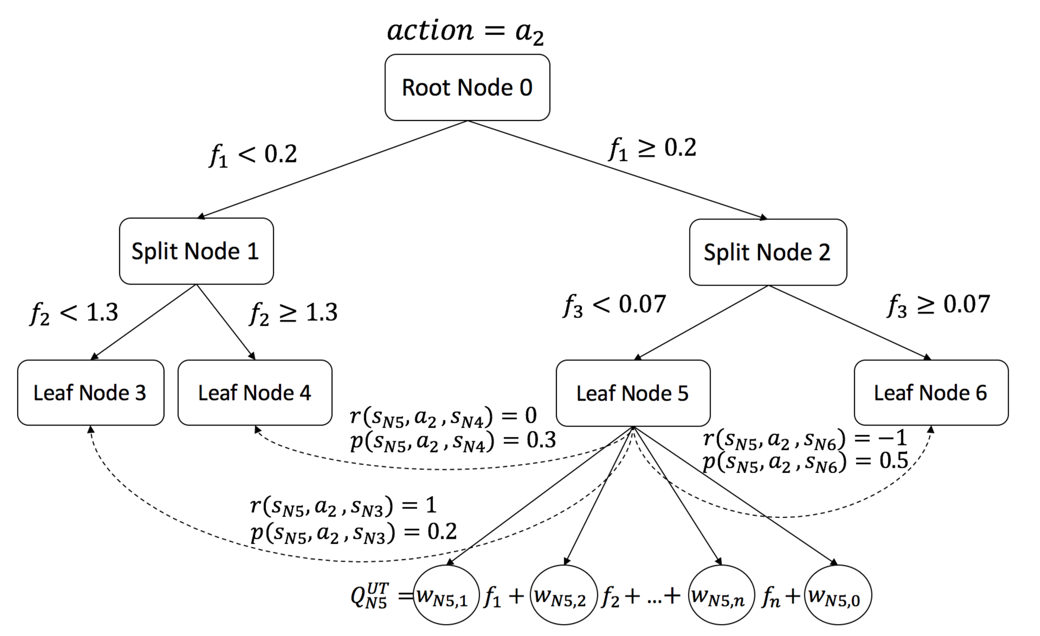


- **Active Play** Setting:
  - Apply a mature DRL model to interact with the environment.
  - At time step t, record a labelled transition $T_t =< I_t, a_t, r_t, I_{t+1}, \hat{Q}(I_t, a_t) >$.
  - Repeat until we have training data for the active learner to finish sufficient updates over mimic model (can apply online learning).
  - Compared to Experience Training, Active Play does not require recording data during the training process of DRL models. This is important because: (1) Many mimic learners have access only to the trained deep models. (2) Training a DRL model often generates a large amount of data, which requires much memory and is computationally challenging to process. (3) The Experience Training data includes frequent visits to sub-optimal states, which makes it difficult for the mimic learner to obtain an optimal return.



Model: Linear Model U-Trees (LMUT)

- **Introduction** to LMUT:
  - **U-tree** learning was developed as an online reinforcement learning algorithm with a tree structure representation.
  - To extend the *generalization ability*, we allow its leaf nodes to contain a linear model.
  - LMUT can also approximate a continuous function arbitrarily closely, with typically with many *fewer leaves.*



- **Training process** of LMUT contian two phases:
  - **Data Gathering Phase** collects transitions on leaf nodes and prepares for fitting linear models and splitting nodes.
  - **Node Splitting Phase** is where LMUT scans the leaf nodes and updates their linear model with Stochastic Gradient Descent (SGD). If SGD achieves insufficient improvement, LMUT determines a new split and adds leaves to the current partition cell.*(For more details check Algorithm 1 on the paper.)*

Empirical Evaluation

- **Evaluation Environment** includes Mountain Car, Cart Pole (benchmark tasks for reinforcement learning) and Flappy Bird (a mobile game that controls a bird to fly between pipes.).
- **Baseline Methods** includes Classification and Regression Tree (CART), M5 Regression-Tree (M5-RT), M5 Model-Tree (M5-MT) and Fast Incremental Model Tree (FIMT). We experiment the advanced version of FIMT with Adaptive Filters on leaf nodes (FIMT-AF)
- **Fidelity: Regression Performance**
  - We evaluate how well our LMUT approximates the soft output ($\hat{Q}$ values) from Q function in a Deep Q-Network (DQN).
  - Compared to the online learning methods (FIMT and FIMT-AF), LMUT achieves a better fit to the neural net predictions with a smaller tree.
- Fidelity results:

Method		Evaluation Metrics		
		MAE	RMSE	Leaves
Experience Training	CART	0.284	0.548	1772.4
	M5-RT	0.265	0.366	779.5
	M5-MT	**0.183**	**0.236**	240.3
	FIMT	3.766	5.182	4012.2
	FIMT-AF	2.760	3.978	3916.9
	LMUT	0.467	0.944	620.7
Active Play	FIMT	3.735	5.002	1020.8
	FIMT-AF	2.312	3.704	712.4
	LMUT	0.475	1.015	453.0

Method		Evaluation Metrics		
		MAE	RMSE	Leaves
Experience Training	CART	15.973	34.441	55531.4
	M5-RT	25.744	48.763	614.9
	M5-MT	19.062	37.231	155.1
	FIMT	43.454	65.990	6626.1
	FIMT-AF	31.777	50.645	4537.6
	LMUT	**13.825**	**27.404**	658.2
Active Play	FIMT	32.744	62.862	2195.0
	FIMT-AF	28.981	51.592	1488.9
	LMUT	14.230	43.841	416.2

(MAE=Mean Absolute Error, RMSE=Root Mean Square Error. The results of Flappy Bird are omitted due to space limit.)

- **Matching Game Playing Performance**
  - We evaluate how well a model mimics Q functions in DQN by playing the games with them and computing the Average Reward Per Episode (APER).
  - We find that among all mimic methods, LMUT achieves the Game Play Performance APER closest to the DQN.

Model		Game Environment		
		Mountain Car	Cart Pole	Flappy Bird
Deep Model	*DQN*	*-126.43*	*175.52*	*123.42*
Basic Model	CUT	-200.00	20.93	78.51
Experience Training	CART	-157.19	100.52	79.13
	M5-RT	-200.00	65.59	42.14
	M5-MT	-178.72	49.99	78.26
	FIMT	-190.41	42.88	N/A
	FIMT-AF	-197.22	37.25	N/A
	LMUT	-154.57	145.80	97.62
Active Play	FIMT	-189.29	40.54	N/A
	FIMT-AF	-196.86	29.05	N/A
	LMUT	**-149.91**	**147.91**	**103.32**

Interpretability

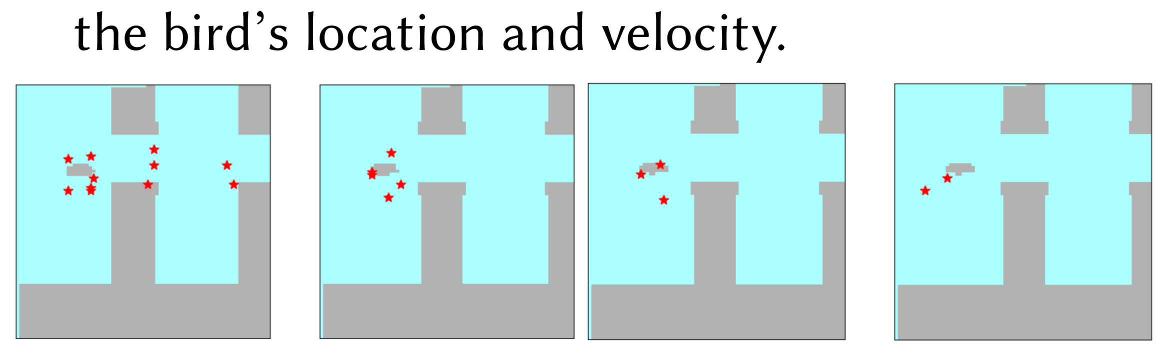
- **Feature Influence**: We evaluate the influence of a splitting feature by the total variance reduction of the Q values:

$$Inf_f^N = \left(1 + \frac{|w_{Nf}|^2}{\sum_{j=1}^{J} |w_{Nj}|^2}\right)\left(var_N - \sum_{c=1}^{C} \frac{Num_c}{\sum_{i=1}^{C} Num_i} var_c\right)$$

(where w_{Nf} is the weight of feature f on node N, Num_c is the number of Q values on node c and var_N is the variance of Q values on node N.)

	Feature	Influence
Mountain Car	Velocity	376.86
	Position	171.28
Cart Pole	Pole Angle	30541.54
	Cart Velocity	8087.68
	Cart Position	7171.71
	Pole Velocity At Tip	2953.73

- **Rule Extraction**: calculate the importance of features and extract rules for typical examples of agent behavior (check our paper for more details).
- **Super-pixel Explanation:**
  - The pixels that have large influence in input images are highlighted with red color to illustrate the key regions.
  - We find most splits are made on the first image which reflects the importance of the most recent image input
  - The first image is often used to locate the pipes (obstacles) and the bird, while the remaining three images provide further information about the bird's location and velocity.



Flappy Bird input images, The input order of four consecutive images is left to right and top to bottom