Attend, Adapt and Transfer: Attentive Deep Architecture for Adaptive Transfer from multiple sources in the same domain

Janarthanan Rajendran, Aravind S. Lakshminarayanan, Mitesh M. Khapra, P Prasanna, Balaraman Ravindran

University of Michigan, Indian Institute of Technology Madras, McGill University
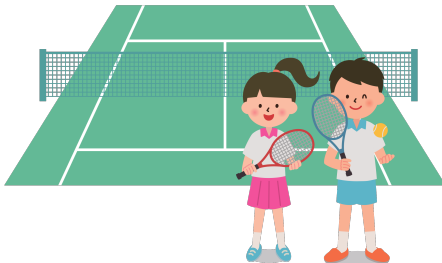
ICLR 2017
Presenter: Jack Lanchantin

Outline

Knowledge Transfer

Knowledge Transfer

- N source tasks with $K_1, K_2, ..., K_N$ being the solutions of the source tasks (e.g. tennis coaches)
- K_B is the base solution for the target task which starts learning from scratch (tennis student's initial knowledge)
- K_T is the solution we want to learn for target task T (tennis student's final skills)

Knowledge Transfer

- N source tasks with $K_1, K_2, ..., K_N$ being the solutions of the source tasks (e.g. tennis coaches)
- K_B is the base solution for the target task which starts learning from scratch (tennis student's initial knowledge)
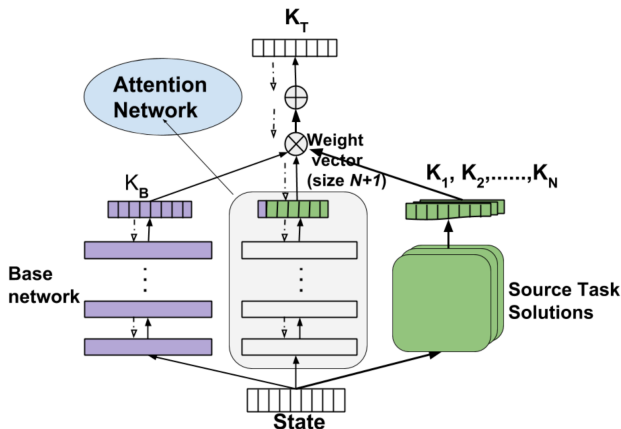- K_T is the solution we want to learn for target task T (tennis student's final skills)

This paper: Using combination of the solutions to obtain K_T

$$K_T(s) = w_{N+1,s}K_B(s) + \sum_{i=1}^{N} w_{i,s}K_i(s) \tag{1}$$

$w_{i,s}$ is the weight of solution i at state s (learned by a separate network)

Attention Network for Selective Transfer (A2T)

$$K_T(s) = w_{N+1,s} K_B(s) + \sum_{i=1}^{N} w_{i,s} K_i(s) \tag{1}$$

Outline

Background: Reinforcement Learning

- S: finite set of states

Background: Reinforcement Learning

- S: finite set of states
- A: finite set of actions

Background: Reinforcement Learning

- S: finite set of states
- A: finite set of actions
- P: state transition probability matrix,
  $P_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$

Background: Reinforcement Learning

- S: finite set of states
- A: finite set of actions
- P: state transition probability matrix,
  $P_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$
- $r(s, a)$: reward function

Background: Reinforcement Learning

- S: finite set of states
- A: finite set of actions
- P: state transition probability matrix,
  $P_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$
- $r(s, a)$: reward function
- R_t: return, sum of rewards over the agent's trajectory: $R_t = r_t + r_{t+1} + r_{t+2} + ... + r_T = \sum_{k=0:\infty} \gamma^k r_{t+k}$

Background: Reinforcement Learning

- S: finite set of states
- A: finite set of actions
- P: state transition probability matrix,
  $P^a_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$
- $r(s, a)$: reward function
- R_t: return, sum of rewards over the agent's trajectory: $R_t = r_t + r_{t+1} + r_{t+2} + ... + r_T = \sum_{k=0:\infty} \gamma^k r_{t+k}$
- π: policy function, distribution over actions given states: $\pi(a, s) = \mathbb{P}[A_t = a | S_t = s]$

Background: Reinforcement Learning

- S: finite set of states
- A: finite set of actions
- P: state transition probability matrix,
  $P_{ss'}^a = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a]$
- $r(s,a)$: reward function
- R_t: return, sum of rewards over the agent's trajectory: $R_t = r_t + r_{t+1} + r_{t+2} + ... + r_T = \sum_{k=0:\infty} \gamma^k r_{t+k}$
- π: policy function, distribution over actions given states: $\pi(a,s) = \mathbb{P}[A_t = a|S_t = s]$
- $V(s)$: state value function, the expected return of a policy π, for every state: $V_\pi(s) = \mathbb{E}_\pi[R_t|S_t = s]$

Outline

Policy Transfer

- Transferring source task policies
- We have $K_1, ..., K_N, K_B, K_T \leftarrow \pi_i, ..., \pi_N, \pi_B, \pi_T$
- The agent acts in the target task by sampling actions from the target distribution π_T, obtained from:

$$K_T(s) = w_{N+1,s} K_B(s) + \sum_{i=1}^{N} w_{i,s} K_i(s) \tag{1}$$

Policy Transfer using REINFORCE

REINFORCE

Direct policy search by making weight adjustments along the gradient of expected reinforcement.

$$\theta_a \leftarrow \theta_a + \alpha_{\theta_a}(r - b)\frac{\partial \sum_{t=1}^{M} log(\pi_T(s_t, a_t))}{\partial \theta_a} \tag{2}$$

$$\theta_b \leftarrow \theta_b + \alpha_{\theta_b}(r - b)\frac{\partial \sum_{t=1}^{M} log(\pi_B(s_t, a_t))}{\partial \theta_b} \tag{3}$$

where α is learning rate, r is return obtained in the episode, b is a reinforcement baseline, M is the length of the episode

Policy Transfer in Actor-Critic

Actor-Critic

Temporal Difference (TD) method where the actor proposes a policy and the critic estimates the value function to critique the actors policy. The updates to the actor happens through TD-error
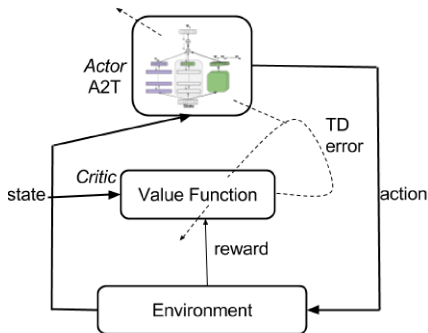
Outline

Value Transfer

- Transferring source task's action-value functions (Q functions):

$$Q_\pi(s, a) = \mathbb{E}_\pi[R_t | S_t = s, A_t = a] \tag{4}$$

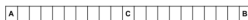- The Q function is used to guide the agent to selecting the optimal action a at a state s.

Value Transfer

Q-learning

One way to learn optimal policies for an agent is to estimate the optimal $Q(s, a)$ for the task. Q-learning is an off-policy learning algorithm that estimates the Q function (e.g. using a deep neural net).

Outline

Selective Transfer with Policy Function

(a) Chain World

- Task LT is to start in A or B with uniform probability and end up in C in the least number of steps.
- Two source tasks, $L1$ and $L2$ are available. $L1$ has learned to reach A from B and $L2$ has learned to reach B from A.
- Model learns to solve LT using REINFORCE

(a) The weights given by the attention network. Selective transfer in REINFORCE

Selective Transfer with Policy Function
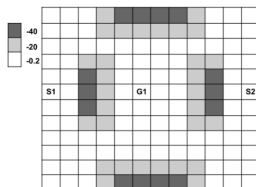
(c) Puddle World 2

- Task *LT* is to start in S1 or S2 and end up in G1 in the least number of steps
- *L*1 has learned to reach G1 from S1 and *L*2 has learned to reach G1 from S2
- Model learns to solve *LT* using Actor-Critic

(b) Selective transfer in Actor-Critic
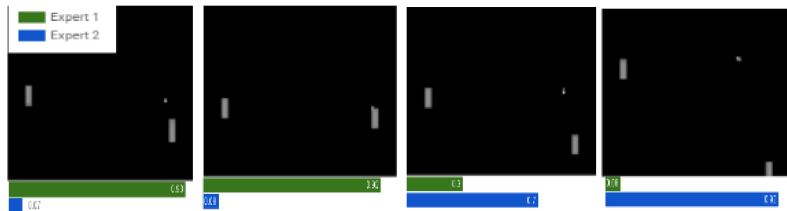
Selective Transfer with Value Function

Figure 4: Visualisation of the attention weights in the Selective Transfer with Attention Network

- L1 performs poorly on upper right quadrant
- L2 performs poorly on lower right quadrant
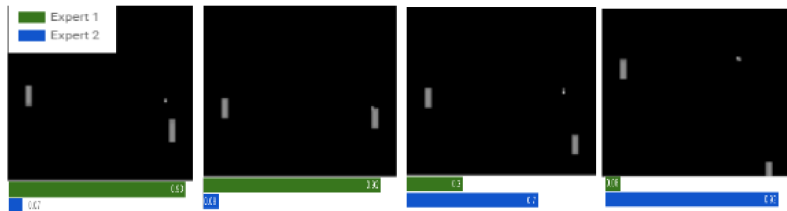
Figure 4: Visualisation of the attention weights in the Selective Transfer with Attention Network

- L1 performs poorly on upper right quadrant
- L2 performs poorly on lower right quadrant
- L1 score of 9.2, L2 score of 8, LT score of 17.2 ([-21,21])

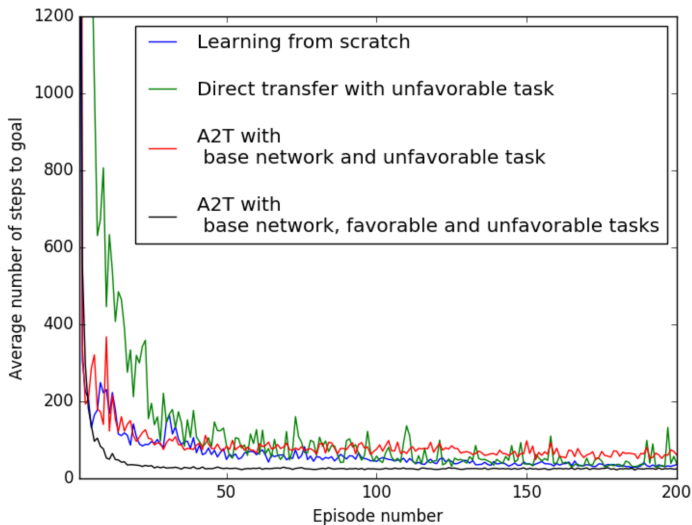# Selective Transfer with Value Function
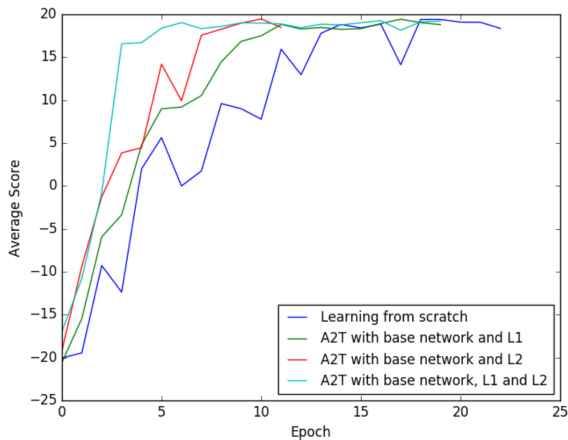
Outline

Avoiding Negative Transfer and Ability to Transfer from Favorable Task (policy transfer in puddle world)
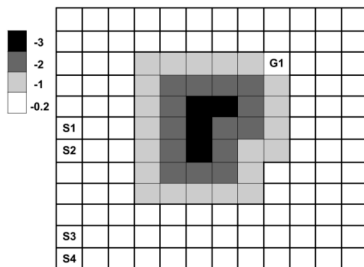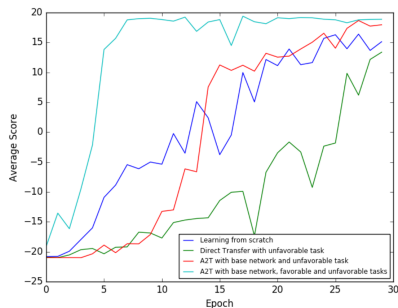

(b) Puddle World 1

- Reach goal state by starting in S1,S2,S3,S4
- L1 is favorable (good) model
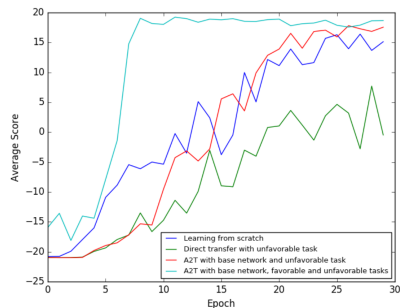- L2 is unfavorable (inverse output weights of L1)

Avoiding Negative Transfer and Ability to Transfer from Favorable Task (policy transfer in puddle world)
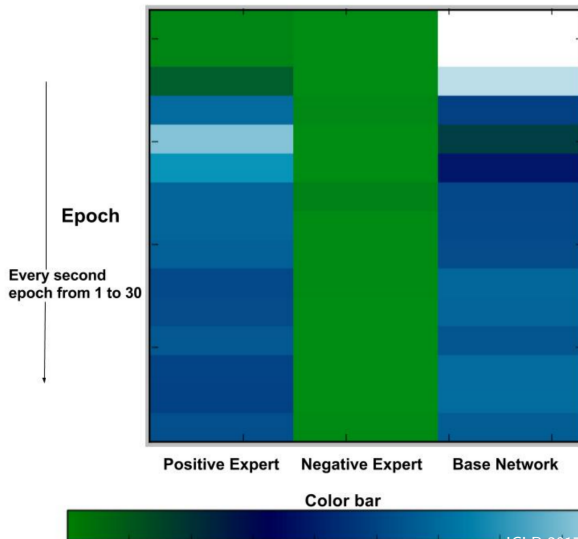
(a) Avoiding negative transfer(Pong) and transferring from a favorable task

(b) Avoiding negative transfer(Freeway) and transferring from a favorable task

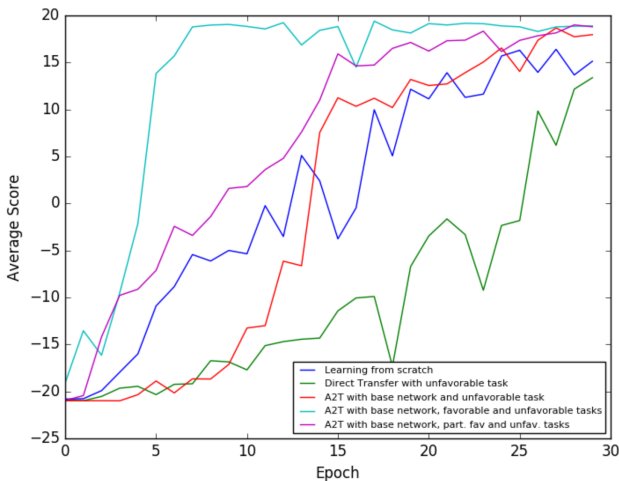# Attention Map for Favorable/Unfavorable Sources (value transfer in pong)

Outline

- Pong with partially favorable and unfavorable source tasks

Conclusions

- General deep neural network architecture, A2T, for transfer learning
- A2T avoids negative transfer while enabling selective transfer from multiple source tasks in the same domain