ONLINE MULTI-TASK LEARNING USING ACTIVE SAMPLING

Sahil Sharma & Balaraman Ravindran
Department of Computer Science & Engineering
Indian Institute of Technology, Madras
Chennai, 600036, India
{sahil,ravi}@cse.iitm.ac.in

ABSTRACT

One of the long-standing challenges in Artificial Intelligence for goal-directed behavior is to build a *single agent* which can solve multiple tasks. Recent progress in multi-task learning for goal-directed sequential tasks has been in the form of *distillation based learning* wherein a single student network learns from multiple task-specific expert networks by mimicking the task-specific policies of the expert networks. While such approaches offer a promising solution to the multi-task learning problem, they require supervision from large task-specific (expert) networks which require extensive training. We propose a simple yet efficient multi-task learning framework which solves multiple goal-directed tasks in an *online* or *active learning* setup without the need for expert supervision.

1 INTRODUCTION

Models learned by Deep Reinforcement Learning (DRL) algorithms (Mnih et al., 2015; 2016; Schaul et al., 2015; Lillicrap et al., 2015) tend to be task-specific. The inability of the AI agents to generalize across tasks gives rise to the field of multi-task learning (MTL) which seeks to find a single agent (in the case of DRL algorithms, a single deep neural network) which can perform well on all the tasks. Successful DRL approaches to the goal-directed MTL problem are of the following kind. They seek to condense the prowess of multiple task-specific teacher networks into a single student network. The Policy Distillation framework (Rusu et al., 2015) and Actor-Mimic Networks (Parisotto et al., 2015) fall into this category of approaches. One trains k task-specific teacher networks and then *distills* the individual task-specific policies learned by the teacher networks into a single student network which is trained using supervised learning. These approaches require teacher networks which are task-specific Deep Q-Networks (Mnih et al., 2015). Such a training of all the teacher networks tends to be extremely resource intensive and often infeasible with growing k. In this work we present the first successful truly on-line deep reinforcement learning approach towards multi-task learning on tasks which have very different state spaces. Our approach never stops learning on any of the tasks and does not require any form of teacher networks' supervision. Our approach learns from the raw video stream of pixel-level data and scalar rewards generated by the environments of the multiple tasks that it is trained to solve. We present empirical evidence that our approach significantly outperforms the baselines we have considered. While our approach can be combined with any DRL algorithm, we show results on one particular instantiation of our approach by combining it with A3C (Mnih et al., 2016) algorithm and showing results on the Atari 2600 domain. A more complete version of this manuscript can be found at (Sharma & Ravindran, 2017)

2 MODEL DEFINITION

We first describe a baseline multi-tasking agent (MTA) (called BA3C) and then describe our approach (Active-sampling A3C - A4C) as a modification of BA3C.
The BA3C MTA is a single A3C network which learns to perform k tasks in an online learning fashion. At the end of every episode, the BA3C agent decides uniformly at random, the identity of the task on which it will train next, for 1 episode. The training algorithm for BA3C is given as Algorithm 1.

Algorithm 1 Baseline Multi-Task Learning

function BASELINEMULTITASKING(SetOfTasks T)
  $k \leftarrow |T|$
  $t \leftarrow$ Total number of training steps for the algorithm
  bmta $\leftarrow$ the naive baseline multi-tasking agent
  for i in $\{1, \cdots, k\}$ **do**
    $p_i \leftarrow \frac{1}{k}$
  for train_steps:0 **to** t **do**
    $j \leftarrow j \sim p_i$. Identity of next task to train on
    score$_j \leftarrow$ bsmta.train_for_one_episode(T_j)

Our method is based on the abstract machine learning principle of active learning (Settles, 2010; Prince, 2004; Zhu, 2005). The core hypothesis which drives active learning in the usual machine learning contexts (such as classification problems) is that a machine learning algorithm can achieve better performance with fewer labeled training examples if it is *allowed to choose* the data from which it learns (Settles, 2010). We present the first successful online multi-tasking deep reinforcement learning algorithm. This means that our method *does not* require access to the hidden features (Rusu et al., 2016) or the action-value predictions (Rusu et al., 2015; Parisotto et al., 2015) of multiple task-specific experts for being able to solve multiple tasks using a single machine learning agent. Our method is also data-efficient. All of our agents are trained on half the training data that would be required to train all the task-specific expert networks. The training algorithm for A4C is stated as Algorithm 2.

Algorithm 2 Active Sampling based Multi-Task Learning

function MULTITASKING(SetOfTasks T)
  $k \leftarrow |T|$
  $b_i \leftarrow$ Baseline score in task T_i. This could be based on expert human performance or even published scores from other technical works
  $n \leftarrow$ Number of episodes used for estimating current average performance in any task T_i
  $l \leftarrow$ Number of training steps for which uniformly random policy is executed for task selection. At the end of l training steps, the agent must have learned on $\geq n$ episodes $\forall$ tasks $T_i \in T$
  $t \leftarrow$ Total number of training steps for the algorithm
  $s_i \leftarrow$ list of the last n scores that the multi-tasking agent scored during training on task T_i.
  $p_i \leftarrow$ probability of training on an episode of task T_i next.
  bsmta $\leftarrow$ the biased sampling multi-tasking agent
  $\tau \leftarrow$ Temperature hyper-parameter of the softmax operation
  for i in $\{1, \cdots, k\}$ **do**
    $p_i \leftarrow \frac{1}{k}$
  for train_steps:0 **to** t **do**
    if train_steps $\geq l$ **then**
      for i in $\{1, \cdots, k\}$ **do**
        $a_i \leftarrow s_i$.average()
        $m_i \leftarrow \frac{b_i - a_i}{b_i \times \tau}$
        $p_i \leftarrow \frac{e^{m_i}}{\sum_{q=1}^{k} e^{m_q}}$
    $j \leftarrow j \sim p$. Identity of next task to train on
    score$_j \leftarrow$ bsmta.train_for_one_episode(T_j)
    s_j.enqueue(score$_j$)
    if s_j.length() $> n$ **then**
      s_j.dequeue()

3 EXPERIMENTAL SETUP AND RESULTS

We tested our approach on 3 different multi-tasking problems. MT_1 involved solving the set of games {Space Invaders, Seaquest, Crazy Climber, Demon Attack, Name this game, Star Gunner}. MT_2 involved solving the set of games {Asterix, Alien, Assault, Bank Heist, Gopher, Tutankhamun}. MT_3 involved solving the set of games {Breakout, Centripede, Kung Fu, Frostbite, Q*-Bert, Wizard Of Wor}.

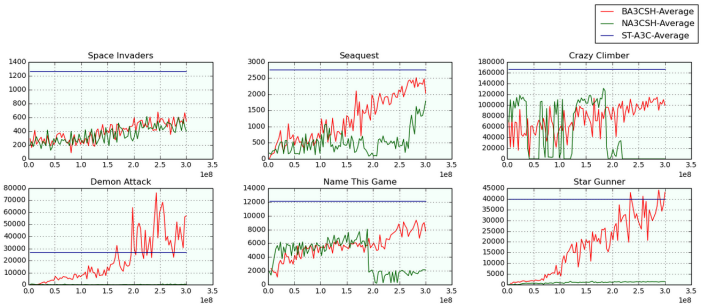

Figure 1: Comparison of performance of A4CSH agent with BA3CSH agent as well as task-specific A3C agents for 6 tasks (multi tasking instance MT_1)

Since the original A3C publication Mnih et al. (2016) contains a different metric (human starts) for evaluation of performance as compared to raw task scores (which we would like to optimize for), we took the published baseline A3C scores from Sharma et al. (2017) as baseline scores. All hyper-parameters throughout this work are tuned on MT_1.

In most previous works such as Parisotto et al. (2015), the arithmetic mean of the performance of the multi-tasking agent on the different tasks is considered as the performance metric based on which the multi-tasking agent is evaluated. This metric is not robust enough because it becomes impossible to distinguish a good multi-tasking agent (latter) from a bad one (former) using this metric of average performance.

To alleviate this problem, we define the following performance metric: $q_{am} = \left(\sum_{i=1}^{k} \min\left(\frac{a_i}{b_i}, 1\right)\right) \Big/ k$. We can similarly define geometric-mean and harmonic-mean based metrics. The evaluation procedure followed throughout our work is as follows. The multi-tasking network is trained for 300 million steps. Every 3 million steps, it is evaluated on each of the 6 tasks for 10 episodes. These 100 evaluations are stored in a log file. After the completion of training, the best model according to each metric is selected separately and offline by directly maximizing for the metric. The performance is reported in Tables 1. In our MTAs, the behavior policy of the agent has a constant size equal to 18, the maximum possible number of actions in the Atari 2600 emulator. We call the A4C version of these agents A4CSH and the BA3C version of these agents BA3CSH.

Table 1: Comparison of performance of A4CSH agents to BA3CSH agents according to metrics q_{am}, q_{gm} and q_{hm} on multi-tasking instances MT_1, MT_2 and MT_3

Name	Agent	q_{am}	q_{gm}	q_{hm}
MT_1	A4CSH	**0.799**	**0.782**	**0.678**
MT_1	BA3CSH	0.244	0.131	0.063
MT_2	A4CSH	**0.601**	**0.580**	**0.515**
MT_2	BA3CSH	0.372	0.343	0.297
MT_3	A4CSH	**0.646**	**0.617**	**0.536**
MT_3	BA3CSH	0.337	0.047	0.008

It is clear from table 1 that A4C out-performs BA3C, specially if one would like to optimize for the performance of the multi-tasking agent on all the tasks (q_{hm}).

REFERENCES

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, February 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

Michael Prince. Does active learning work? a review of the research. *Journal of engineering education*, 93(3):223–231, 2004.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *4th International Conference on Learning Representations*, 2015.

Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

Sahil Sharma, Aravind S. Lakshminarayanan, and Balaraman Ravindran. Learning to repeat: Fine grained action repetition for deep reinforcement learning. *To appear in 5th International Conference on Learning Representations*, 2017.

Sahil A Sharma and Balaraman Ravindran. Online multi-task learning using active sampling. *arXiv preprint arXiv:1702.06053v2*, 2017.

Xiaojin Zhu. Semi-supervised learning literature survey. 2005.