Introduction to Meta Reinforcement Learning

Gwanwoo Choi
Department of Computer Science and Engineering
UNIST
cgw1999@unist.ac.kr

Abstract

Reinforcement learning is a learning process in which an agent and environment interact through information such as action, state, and reward to achieve a given goal. Meta-learning, known as "learn to learn," means adjusting the model to quickly adapt to a new environment through some training examples. And meta reinforcement learning is a field that aims to efficiently learn an agent using interrelated tasks drawn from same distribution so that agent can adapt unseen task quickly. In order to achieve this goal, there are a lot of meta reinforcement learning algorithms distinct from typical RL. In this paper, I will explain these basic methods of Meta Reinforcement Learning in details and classify these according to the characteristics of each methods.

1 Introduction

In this paper I will introduce about Meta Reinforcement learning that is considered to effectively learn on related tasks from same distribution. It can also address shortcomings of Deep Reinforcement Learning and typical Reinforcement Learning. I will explain some basic Meta Reinforcement Learning algorithms with categorizing to RNN based and Gradient based. Before introducing methods there is a summarize about meta learning and purpose of meta reinforcement learning.

1.1 Meta Learning

The goal of meta learning is to train a model on a variety of learning tasks, so that it can solve new tasks using prior knowledge to the distribution of tasks obtained by training the tasks from that distribution. This method is devised to overcome limitation of deep learning. Deep learning only performs well on huge amount of avaliable data. To reduce dependency on the quantity of data meta learning is proposed. Typically, well-learned meta learning model adapts quickly or generalizes well to new tasks or environments that were not encountered in the training session. Usually in meta-test time, new tasks are selected from similar or same distribution of training tasks.

1.2 Meta Reinforcement Learning

Reinforcement learning is a learning process in which an agent and environment interact through information such as action, state, and reward to maximize expected rewards. Meta reinforcement learning is simply, applying meta-learning to reinforcement learning. In meta reinforcement learning, by using interrelated Markov Decision processes (MDP) as training environments, agent can adapt fastly unseen new MDPs which has similar or same distribution with training MDPs. Meta-RL (meta reinforcement learning) aims to make agent learn a policy which can maximize the discounted cumulative reward for new tasks from distribution $\mathcal{T}$ as efficiently as possible. Usually training tasks and test tasks are different with each other but test tasks are from same distribution $\mathcal{T}$ or slightly different distribution $\mathcal{T}'$.

There are some basic meta reinforcement learning algorithms which can be classified by RNN-based meta learner and Gradient descent based meta learner. So I will briefly explain each category followed specific methods. Also there is explanation other famous methods which don't include in RNN-based meta-learner category and Gradient-based meta-learner category.

2 RNN Based Meta Learner

This type of methods utilizes recurrent neural network to memorize past history in RNN cells (hidden states) [4,19]. It is largely proceeded with the following three processes. First, train a recurrent neural network by using deep RL algorithm. Second, Use training set which includes a series of interrelated tasks. Third, give action selected and reward received in the previous time interval as input to network. By utilizing recurrent neural network agent can draw proper bias over interrelated tasks. The reason utilizing RNN network is to gain information of past states, actions, and rewards, $\mathcal{H}_t = \{x_0, a_0, r_0, , , , .x_{t-1}, a_{t-1}, r_{t-1}, x_t\}$, from hidden state.

RNN based meta reinforcement learning method appears after development of Deep Reinforcement Learning [5]. Deep Reinforcement Learning presented a breakthrough to enable tricky tasks such as Atari [5] and Go [20]. The main key point of Deep Reinforcement Learning is adapting non-linear function approximation to Reinforcement learning. However there are obvious limitations although this deep RL algorithm achieves more than or equal to human-level performance in some tasks. First, deep RL requires huge amount of training data, unlike human can obtain proper performance with comparatively little experience. Second, deep RL shows good performance for only one task at a time, Whereas human learners show satisfactory performance for several tasks in a short period of time [6]. To address these problem, RNN based meta reinforcement learning method is proposed.

2.1 RL^2

RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning [2], basically wants to utilize prior knowledge from training session when it solves new tasks. Deep reinforcement learning had successfully achievement in several tasks such as Atari game [4]. But it takes extremely long time to training model compared to human. [2] saw the reason of this as lack of prior knowledge. So they utilize several reinforcement learning experiences (history), which of each is obtained from individual MDP excuted n times. These learning experiences, states, actions, rewards and termination flags, are stacked to Gated Feedback Recurrent Neural Network (GRU) [4]. The input of GRU is embedded feature represented by $\phi(s, a, r, d)$ where s is state, a is action, r is reward and d is termination flags. And the agent, who is able to bring information from GRU, is learned by Trust Region Policy Optimization [3] algorithm which is one of the popular policy optimization algorithm.

By using TRPO [3] policy optimization to agent who has RNN based policy, RL^2 achieved relatively better or similar performance on Multi-Armed Bandits problem and Tabular MDPs problem compared to algorithms designed to only solve Multi-Amred Bandits and Tabular MDP problems. The performance of RL^2 is meaningful in the sense that it doesn't know which task is it in. To show that it can work well not only these simple problem but also high dimensional task, Duan, Y. et al [2] show the experimental results of visual navigation in which agent should find exit of maze generated in Minecraft.

In summary the slow reinforcement learning is typical reinforcement learning process and slow reinforcement learning is the meta learning phase that learns agent with RNN network which utilizes states, actions, rewards and termination flags of slow reinforcement learning histories as inputs .

2.2 Learning to Reinforcement Learn

This method, Learning to Reinforcement Learn [6], is inspired by an approach introduced by Hochreiter and colleges [7] and takes similar approach with above one [2]. In [7], They use LSTM cell [19] to train a network over series of interrelated tasks only for supervised learning. At the training time, [6] gives LSTM cell auxilary input y_{t-1} as well as typical input x_t so that make model recognize the interrelation of tasks. And in this paper, this idea was extended from supervised learning to reinforcement learning. Like [7] which utilizes y_{t-1} as auxiliary input in supervised circumstance, Learning to Reinforcement Learn [6] utilizes previous action and reward as auxiliary inputs.

Also [6] adapts Advantage Actor-Critic structure with one or more LSTM cells. Advantage Actor-Critic algorithm is explained in [8,9] and simply called A2C and A3C. In order to proof whether model shows good performance on several Markov Decision Processes drawn from same or slightly different distribution of training tasks, [6] use total seven tasks to experiments. This model also achieves higher performances than typical multi-arm bandit algorithms and had a good result in other Markov Decision Process tasks.

3 Gradient Descent Based Meta Learner

Gradient descent based meta learning basically aims to learn common structures of tasks by updating its parameters with gradient descent method. This approach has simpler structure than RNN based model because it does not acquire additional memorable parameters of models, such as RNN network. This is method initializing parameter closely to optimal parameters of training tasks. To do so, model can be easily fine-tuned to testing tasks. Note that testing tasks have same or similar distribution with training tasks.

3.1 MAML

MAML [1] is the beginning of the gradient descent based meta reinforcement learner. MAML is an abbreviation of "Model-Agnostic Meta-Learning for fast Adaptationi of Deep Networks". MAML is a general purpose method so it can used in both supervised learning and reinforcement learning. Actually it can be applied to any model which uses gradient descent method to optimize its own parameter. "Model-Agnostic" means this flexibility and generality.

MAML claims that only with a small amount of training datas from same distribution produce good generalization performance. In another words, this algorithm make model sensitive to loss function of unseen tasks with respect to the parameters. So if the model is well trained, small changes to the parameters draw out the huge improving of loss and one can easily fine-tune model to new tasks. By this property, with only few gradient updates, model trained by MAML algorithm can show good generalization performance on a new task.

When considering general model that can be represented by f_θ, after gradient update, parameter can be represented as $\theta' = \theta - \alpha\nabla_\theta\mathcal{L}(f_\theta)$. And when model weight θ is updated on specific task $\mathcal{T}_i$ driven from $\mathcal{T}$, it can be represented similarly as $\theta_i' = \theta - \alpha\nabla_\theta\mathcal{L}_i(f_\theta)$. $\mathcal{T}$ represents the set of tasks with same distribution, $\mathcal{L}$ represents loss and α represents hyperparameter. In typical training session MAML collects updated parameters $\theta_1, \theta_2, ..., \theta_i$ over several tasks $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_i$. After training session, meta-learning phase is started. In meta-learning phase, meta-objective is finding $\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ and MAML finds parameter $\bar{\theta}$ that minimize meta-objective to update parameter θ as $\bar{\theta}$. By doing so, MAML updates model's parameter θ close to the optimal parameters $\theta_1^*, \theta_2^*, ..., \theta_i^*$ of tasks $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_i$. The typical learning phase and meta learning phase are runned alternately until model converges. The reason utilize this kind of algorithm is drawing some important internal representations on interrelated tasks.

In typical learning session gradient update is computed using vanilla policy gradient (REINFORCEMENT) [10] and in meta learning session gradient update is computed by trust-region policy optimization (TRPO) [3]. In the experiments MAML achieves much higher performance than other baseline models : pretrained model and random model. Not only did it show excellent performance in relatively easy 2D navigation, but it also showed good performance in Locomotion, a high-level task , by using MuJoCo simulator [11].

3.2 Reptile

Reptile [18] is an approximation method to MAML [1] but more simpler compared to MAML. The training process is only slightly different with MAML. Reptile only uses first-order derivatives which is driven by ignoring second-order gradient term in MAML. In Reptile, tasks are sampled repeatedly. It trains on each task and move the initialization weight to updated point on that task.

Nichol, A. et al [18] shows that both result applying tayler series to gradient update of MAML and Reptile has almost same result in condition of small αk, where is k is the number of gradient updates

per one task and α is the stepsize. This is the reason Reptile works well exhibiting almost same result and lower time complexity to MAML.

3.3 MAESN

MAESN [12] means model agnostic exploration with structured noise, which enable model to learn exploration strategies from prior experience. In reinforcement, the balance between exploit and exploration is quite important and in previous models we had seen above there are some lackages for exploring. MAESN starts from the idea that prior tasks can be utilized to teach agent how exploration should be performed in new tasks. Unlike RL methods without learned exploration strategies or task-agnostic exploration methods, MAESN aims to offer general exploration strategy that is able to apply over any tasks.

Both RNN-based meta-learners and gradient descent based meta-learner have drawbacks, they can't efficiently explore in high-level tasks. The more complex tasks are, the more sophisticated exploration strategies are needed. RNN based learner simply find optimal policies from training tasks and this is essentially different with learning good exploration strategies. So RNN based learner easy to fail exploration. And in case of gradient descent based meta learners we had seen, they appeared to find it difficult to effectively learn exploring since they don't have specific exploration strategy. Gradient descent based meta learner only relies on gradient update method. The reason that past two types of models achieve such well working in its own experiment is they are tested on too simple tasks where only a few random trials are need to gain the desired goals. So in the tasks with challenging exploration components, MAESN [12] can adapt quickly while others can't.

MAESN adopt structure of MAML [1] as basic foundation. It has structure which is combined structured stochasticity with gradient update of MAML. Unlike typical stochastic policy which utilizes action distributions $\pi_\theta(a|s)$ inependet for each time step, MAESN adds conditional variables $z \sim q_w(z)$ to π_θ, where q is specific distribution parametrized by w in order to adapt temporally coherent stochasticity to policy π_θ. So the policy used in MAESN is represented as $\pi_\theta(a|s, z)$. The purpose of training agent in MAESN is mainly finding coherent exploration strategy from training tasks and producting quick adaptation to new tasks by structured exploration strategy. In order to achieve that purpose, MAESN jointly learn not only policy parameters θ, but also latent space distribution parameters w. By doing so, MAESN can achieve to goals, both structured exploration and also quick adaptation.

4 Other methods

There are several other Meta-RL methods. Meta-Gradient RL [13] meta-learns hyperparameters, discount factor r and bootstrapping parameter λ. High-dimensional continuous control using generalized advantage estimation [14] and Evolved Policy Gradient [15] is the method of meta-learning the loss. In addition, there exists Meta-RL methods through random rewards [16] and methods through Evolutionary Algorithm on Environment Generation [17].

References

[1] Finn, C., Abbeel, P., & Levine, S. (2017, July). Model-agnostic meta-learning for fast adaptation of deep networks. In International Conference on Machine Learning (pp. 1126-1135). PMLR.

[2] Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., & Abbeel, P. (2016). RL ^{2}: Fast reinforcement learning via slow reinforcement learning. arXiv preprint arXiv:1611.02779.

[3] Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015, June). Trust region policy optimization. In International conference on machine learning (pp. 1889-1897). PMLR.

[4] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2015, June). Gated feedback recurrent neural networks. In International conference on machine learning (pp. 2067-2075). PMLR.

[5] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

[6] Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., ... & Botvinick, M. (2016). Learning to reinforcement learn. arXiv preprint arXiv:1611.05763.

[7] Hochreiter, S., Younger, A. S., & Conwell, P. R. (2001, August). Learning to learn using gradient descent. In International Conference on Artificial Neural Networks (pp. 87-94). Springer, Berlin, Heidelberg.

[8] Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., & Kavukcuoglu, K. (2016). Reinforcement learning with unsupervised auxiliary tasks. arXiv preprint arXiv:1611.05397.

[9] Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A. J., Banino, A., ... & Hadsell, R. (2016). Learning to navigate in complex environments. arXiv preprint arXiv:1611.03673.

[10] Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). PMLR.

[11] Todorov, E., Erez, T., & Tassa, Y. (2012, October). Mujoco: A physics engine for model-based control. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 5026-5033). IEEE.

[12] Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., & Levine, S. (2018). Meta-reinforcement learning of structured exploration strategies. In *Neural Information Processing Systems*, NIPS.

[13] Xu, Z., van Hasselt, H., & Silver, D. (2018). Meta-gradient reinforcement learning. In *Neural Information Processing Systems*, NIPS.

[14] Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, ICLR.

[15] Houthooft, R., Chen, R. Y., Isola, P., Stadie, B. C., Wolski, F., Ho, J., & Abbeel, P. (2018). Evolved policy gradients. In *Neural Information Processing Systems*, NIPS.

[16] Gupta, A., Eysenbach, B., Finn, C., & Levine, S. (2018). Unsupervised meta-learning for reinforcement learning. arXiv preprint arXiv:1806.04640.

[17] Wang, R., Lehman, J., Clune, J., & Stanley, K. O. (2019). Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. arXiv preprint arXiv:1901.01753.

[18] Nichol, A., Achiam, J., & Schulman, J. (2018). On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999.

[19] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[20] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Chen, Y. (2017). Mastering the game of Go without human knowledge. Nature, 550(7676), 354.