sliu1@uvm.edu
Sida Liu

1 Learning Optimization: RL-based Optimizers for Gradient Descent

1.1 Structure

The deliverable will be a piece of open-source software.

1.2 Topic

Reinforcement Learning (RL) was developed in the 90s. The recent boost of RL was benefited from the development of Deep Learning (DL), especially the backpropagation and sophisticated optimization algorithms based on Gradient Descent (GD). However, currently used optimizers, such as Adam, are still made of hand-designed rules. It might be possible to produce optimizers that are more adaptive in complex tasks.

In this work, I will propose RL-based Optimizers. We can call them AlphaOpts. There will be two loops in this design: the outer loop is the Deep Learning task we want to solve; the inner loop is an RL algorithm inside the AlphaOpt.

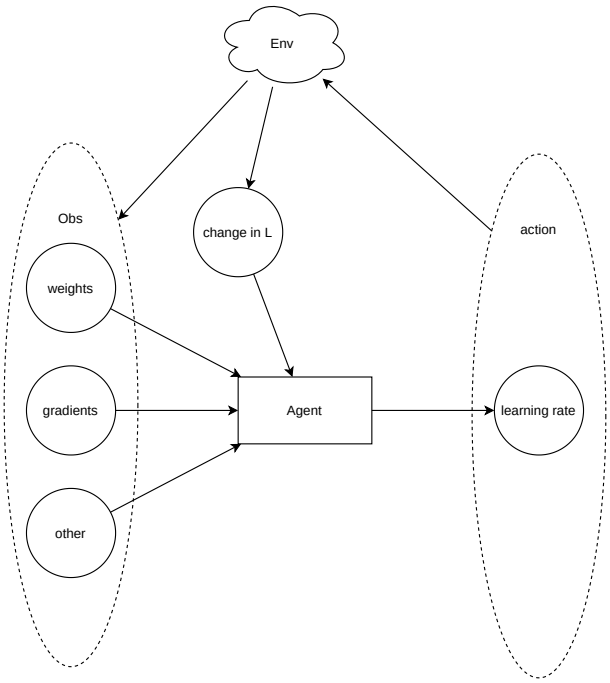

Figure 1: The outer loop of an AlphaOpt

The core of an AlphaOpt is a standard Deep RL agent. The goal of an AlphaOpt is to minimize loss function as fast as possible, $\min_{\theta'} \mathcal{L}$, where θ' is the inner parameters. The

reward function is the change of the value of the loss function, $-\Delta\mathcal{L}$. The observation includes current outer parameters, θ, outer gradients, $\nabla_\theta\mathcal{L}$, and optionally, other additional information. The action will be the learning rate for each parameter at that step, α_θ.

Though it is possible to use recursive AlphaOpts, in this work, an AlphaOpt itself will use conventional optimizers, such as Adam, to adjust its parameters.

1.3 Plan

The basic principle is *Dream big, Start small*.

1.3.1 2-D task

The first task will be a toy task. The purpose of this task is to see the AlphaOpt work in action. In this task, there will be no Deep Learning in the outer loop.

There are only two outer parameters, so we can visualize the parameter landscape. Those landscapes are hand-designed, such as a convex landscape as the simplest case. We can see the learning process of the AlphaOpt and eventually the parameters of the outer loop reached the minimum.

1.3.2 MNIST

The second task is to recogonize MNIST hand-written digits. This is a well known scenario that we can use a CNN architecture.

I will modify an existing algorithm, replacing the optimizer with an AlphaOpt, and see if it can solve the problem.

1.3.3 Omniglot

The third task is to recognize Omniglot letters in a meta-learning setting. There are many sub-dataset in Omniglot. Our learning algorithm will first train on the meta-training set, and then test on the meta-testing set. During meta-training, the outer parameters and the inner parameters can change, but during meta-testing, only outer parameters can change, and the inner parameter of AlphaOpts will keep the same. Thus we can see how pre-trained optimizer can facilitate the few-shot learning in this setting.

1.4 Related Works

1.4.1 Learning to Optimize

https://bair.berkeley.edu/blog/2017/09/12/learning-to-optimize-with-rl/

Li et al. proposed this idea in 2016 [1], which is very similar to my idea. In Li et al., 2016, the observation contains: (1) Current parameters, θ, (2) The $\Delta\mathcal{L}$ in recent H time steps, (3) The $\nabla_\theta\mathcal{L}$ in recent H time steps. (I changed the notation to be aligned with this document.) The action contains: (1) the change to make, $\Delta\theta = \alpha\nabla_\theta\mathcal{L}$. The goal is to minimize the total area under the learning curve at meta-test time. The reward function (cost function) is the negative value of outer loss function, $-\mathcal{L}$. They chose Guided Policy Search (GPS) as the RL algorithm for the optimizer.

Li et al. 2017 provides more details on implementation [2], though source code is not available. The learned optimizer then tested on several image dataset, which outperforms Adam, AdaGrad, RMSprop in the experiments.

In general, I think Li's implementation can be improved somehow. At least I should be able produce a reusable package that can be easily used in PyTorch.

1.4.2 Learning to learn by gradient descent by gradient descent

Another highly cited paper Andrychowicz et al. 2016 [3]. Not using RL but other methods.

References

[1] Ke Li and Jitendra Malik. Learning to optimize, 2016.

[2] Ke Li and Jitendra Malik. Learning to optimize neural nets, 2017.

[3] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent, 2016.