David Gao, Alex Young, Chace Hayhurst, Alex Brown

Optimizer^3

Based on 'Learning to learn by gradient descent by gradient descent'

Introduction

Typically, optimizers for neural networks are hand designed by programmers (ADAM, SGD, etc...). As we have seen, these optmizers work well, but are not always perfect for every application. What if we could use a neural network to learn an optimizing function? In theory this network would be able to generalize better than rtraditional learning algorithms.

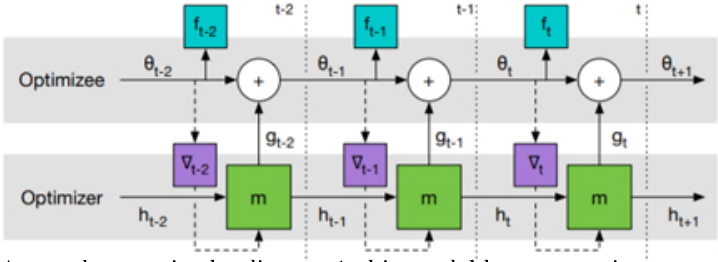
A neural network that trains another neural network! This is the focus of our project, based on the 2016 paper 'Lea`rning to learn by gradient descent by gradiebt descent'[1].

Dataset

For this project, we used the MNIST dataset in the form that was given to us in project 1. Furthermore, we also generate our own data to experiment with in the form of points drawn from a synthetic quadratic function that we hope to be able to learn using a neural network.

Model

Our project was built using the architecture outlined in the original paper:

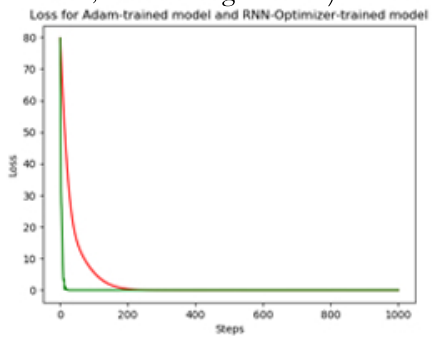


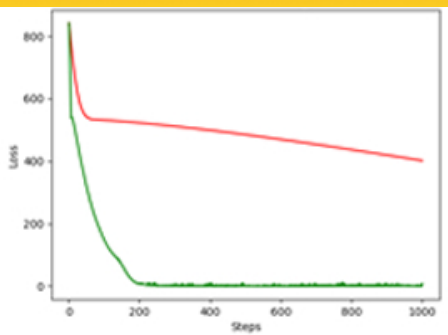
As can be seen in the diagram[1], this model has two main componets. An optimizee, and an optimizer. The optimizee is a neural network that generates outputs, and gets updates to its parameters by feeding gradients into the optimizer network. The optimizer is an LSTM based network that will calculate the updates to the optimizee parameters using its LSTM cell.

In practice, we also use what the paper termed a 'coordinatewise LSTM optimizer', to act on each parameter individually to prevent a massive hidden state being passed into the optimizer on every iteration. To train these networks, we use a loop over all the examples, calculate the gradient w.r.t. every parameter in the optimizee, and then use the optimizer to get the updates to the optimizer. The optimizer is trained to minimize the loss caused by its updates.

Results

The following demonstrate our results for training on the quadratic functions and MNIST respectively (the green line is our model, red is using ADAM):





Discussion

As you can see, our model converges much faster than ADAM, meaning our project is sucessful! Throughout this project, we learned to always carefully select the library you are going to use, as accessing the computation graph is much easier in pytorch, however we opted to go with tensor-flow, which made this project much more difficult. This project is limited in the sense that we only explored meta-learning on two datasets, which is fine as a proof of concept, but doesn't give any information on how this construction would generalize to different situations.
As for future work, one could test this implementation with more complex datasets to see if there is some point at which the model breaks down. Furthermore, they could test against other optimizers (SGD, etc...)to see how they compare to this model we have constructed.

Sources

1. Andrychowicz, Marcin, Denil, Misha, Colmenarejo, Sergio Gomez, Hoffman, Matthew W., Pfau, David, Schaul, Tom, and de Freitas, Nando.
Learning to learn by gradient descent by gradient descent. CoRR, 2016.