Optimizers^3 Reflection

Alexander Brown (abrown53), Dichuan Gao (dgao7),
Chace Hayhurst (chayhurs), Alexander Young (ayoung38)

November 2020

1 Introduction

A *model* is a parameterized function

$$f_\theta : X \to Y$$

where X is the space of all possible inputs (say, all possible images), and Y is the space of all possible outputs (say, a binary set where the items indicate whether or not an input image contains a cat); and θ is the parameter on f. We denote Θ to be the parameter space, i.e. the set of all possible values θ can take on.

A *loss-function* is a function $l : \Theta \to \mathbb{R}$ measuring the performance of the model: the lower the value of l, the better the model. Most machine learning tasks boil down to minimizing the loss-function; that is, we seek

$$\theta^* = \arg\min_{\theta \in \Theta} l(\theta)$$

The idea of supervised learning is to successively approximate the optimal value θ^* by repeatedly updating θ so as to decrease the value of $l(\theta)$ gradually. This successive update requires an *update rule* $\theta_t \mapsto \theta_{t+1}$, or in other words, an *optimizer*, specifying how θ is to be updated. For example, gradient descent is the update rule

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla_\theta \, l(\theta)$$

In this course, all the optimizers we worked with are hand-crafted. The update rule is explicitly written in closed mathematical form.

But it is clear that an optimizer is itself a model. It takes as input the current parameter θ_t, together with some information about the loss-function (say, the gradient $\nabla_\theta \, l(\theta)$), and it returns as output a new parameter θ_{t+1}; or formally, an optimizer is a model

$$g : \Theta \times \Lambda \to \Theta$$

where Λ is a space of some information about the loss-function. Why, then, should we not make this optimizer itself trainable?

We will implement the 2016 paper "Learning to Learn by Gradient Descent" [**?**]. The idea is to parameterize the optimizer itself, so that it is itself a trainable model

$$g_\phi : \Theta \times \Lambda \to \Theta$$

where Λ is simply the space of possible gradients $\nabla_\theta\, l(\theta)$, and Φ is the parameter space for the optimizer, i.e. the set of all possible values for ϕ. We will create an RNN architecture for this optimizer, define a reasonable loss-function $\mathcal{L} : \Phi \to \mathbb{R}$ for the optimizer, and train it to become a good optimizer for several different supervised models.

In this sense, a model (say a model for categorizing MNIST data) will be an *optimizee.* Our RNN will be an *optimizer.* Our program, which trains the *optimizer*, will be an *optimizer optimizer.* Thus, we, the team of programmers writing this program, are *optimizer optimizer optimizers.* We are optimizers cubed.

2 Challenges

The most challenging part of the project thus far has to be actually coding the model itself. It was rather difficult to implement some of the portions of the optimizee model, as it essentially required us to recreate the LSTM layer from scratch in order for it to have the behavior we want. Furthermore, we are having issues writing the loss function for the rnn_optimizer class that we hope to get help with during the next checkin. Other than this, the project has been moving along smoothly.

3 Insights

There are no concrete results for our project at this point (aside from the code we have written itself), as we are not finished creating the model and thus cannot test it on the datasets we have planned. This is exactly where we expected our model to be at this point, and functionally is limited but the parts that are written do work correctly.

4 Plan

We are on track with this project, as we are nearing a completed model, which was where we wanted to be at this point in time. We will need to dedicate more time to model creation and possibly have several others help with this process, as we foresee several issues with coding the loss function for the RNN optimizer, and the RNN optimizer in general. We don't currently plan to change anything with our model, just continue coding it until we can perform our experiments.