Project Proposal Addendum

Clement Lee

October 24, 2016

Organized by major points.

1 Weight Updates and Training

My intent is for the weights to be specified by standard packages provided in most major deep learning packages; that is, built on a base of backpropogation but also including optimizations for momentum, magnitude adjustments, and more. In the past I have personally used Adam [2] to good succcess, but it will also be interesting to see how different weight updating methodologies react to these algorithms.

2 Architecture

It was recently brought to my attention that Deep Residual Networks [1] might be the most interesting architecture to play with for this project. Basically, where most neural networks transform an input layer by layer, encoding the entire layer (let's call each layer l_i), such that $l_0 \to l_1 \to \cdots \to l_n$. Residual networks instead have each layer compute the difference between layers ($l_i - l_{i-1}$), and then add the previous output, like

$$l_0 \to l_0 + (l_1 - l_0) \to l_1 + (l_2 - l_1) \to \cdots$$

This means that each layer of the neural network only has to compute "new information" between two layers, and not re-encode the original information, which has been shown to be very good in helping create deeper networks.

For my own project, this is primarily interesting because it means that it's easier to play with adding and removing layers. In this setup, a layer with all weights initialized to zero will produce 0 residual ($l_i - l_{i-1} = 0$), so it is as if the layer doesn't exist at all. This means that I can produce more dynamic architectures

without having to worry about generating significantly different structures for every iteration.

3 Weight Deletion/Addition

I'm not quite sure what the complexity of the optimal brain damage algorithm is yet, or if there are any practical concerns preventing it from being used, so I'll have to explore how it works precisely. This will also mean that I'll be seeing how possible it is to integrate it into standard libraries, or whether it will be dramatically more feasible just to do most of my experimenting with a heuristic-based deletion algorithm (e.g. delete all small weights).

In terms of the other goal I had in mind of addition, I think the residual network will help with adding layers, and in terms of adding width to the network (making a single layer bigger), I think the first way of doing this will be to just start with an overcapacity; that is, make the layers bigger to begin with and set the other unused values to 0. The decision of adding layers just needs to be limited for memory concerns; one of the key points of interest for residual networks is that they can be incredibly deep anyway, so adding layers can be done relatively lightly.

The difficult part of this is algorithmically determining when a layer needs to be expanded, which is something I've been actively thinking about. I've been toying with an idea that also builds on the residual network described above. Basically, since each layer passes on a residual, there's some ability to model how small changes affect the overall performance of the network; for example, if in layer 1, we modify some weights, we can see how these propagate in terms of error in the network as a ripple effect. If these changes are drastic, then the network may be underprovisioned (needs more capacity), in which case we expand it; if these changes don't do anything, then the network can be reduced in capacity iteratively.

4 Validation

There are some pretty popular basic results indicating what can be done with standard networks and training methodologies. I'd consider it a win if I can reduce the number of hyperparameters needed in general, either by taking an existing network and modifying it or by generating a new network entirely. This is of course going to come at the cost of training time, but I don't see that as a necessary point of optimization, unless the time taken is entirely unreasonable.

At the same time, I'd like to achieve at least as good or better performance on common datasets. I don't believe that there's much to be gained from testing

other datasets (since the networks won't have been optimized for it), but I think it'd be definitely interesting to compare if I have enough time, as a side note on generalizability of the algorithm.

5 Next Steps and General Timeline

I'm going to be working on better understanding weight deletion algorithm over fall break. Beyond that, here are some timelines I'm hoping to hit:

Mid November Finish example weight deletion on basic neural network

Early December Test out heuristic or optimal brain damage weight deletion on residual network

Mid December Begin "adding" layers by using preinitialized 0 layers

Winter Break Use more advanced weight deletion methods

References

[1] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385* (2015).

[2] Kingma, D., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).