FOR.ai Technical Challenge - Pruning

Purpose

This technical challenge is for candidates to demonstrate their problem solving and learning abilities. All tasks in this challenge are based on real-life examples of what you could work on at FOR.ai. There's no right or wrong way to do this challenge, so don't stress out too much about it. The purpose is to facilitate a conversation about machine learning and best practices. We want to see how you think and what your opinions are.

Rules

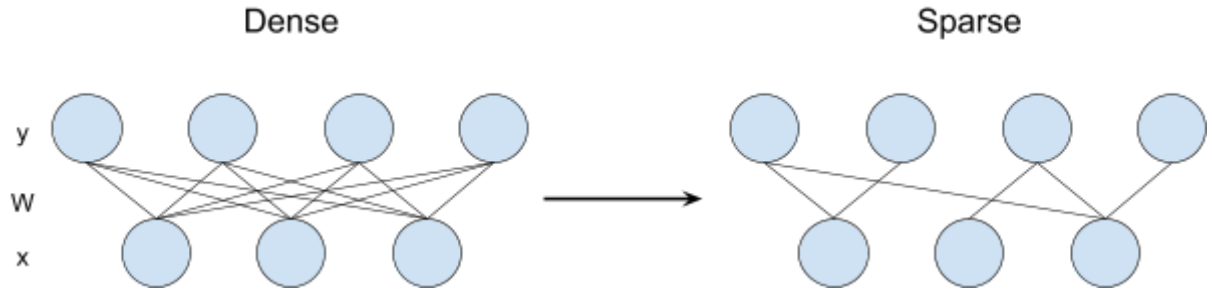
1. Please use TensorFlow (it's what we use for all of our projects). Consider using colab for access to free GPUs/TPUs.
2. You may use frameworks or libraries as you see fit. If you borrow code please include proper attribution **and have a clear separation between the code you borrowed and the code you wrote yourself.**
3. You should **keep your code simple and focus on readability of your code**. Include any instructions for running and reading your code in a README file. **We value thoughtfully written clean, and communicative code so other contributors can easily understand and build on top of it.**
4. You may skip any parts of the challenge if you get stuck or don't have relevant experience. However, we encourage you to learn and demonstrate newly acquired skills.
5. You should check your solution into **GitHub** or **Colab** and provide basic instructions on how to reproduce your results.
6. You are free to spend as little or as much time as you want on this challenge.
7. You are expected to learn something new after you complete the challenge :)

Assessment

Your solution will be assessed according to the criteria above and we will respond within the week with your results.

The Challenge

TL;DR: Train a very large neural network, then make it very small.



Networks generally look like the one on the left: every neuron in the layer below has a connection to the layer above; but this means that we have to multiply a lot of floats together (12 in the example). Ideally, we'd only connect each neuron to a few others and save on doing some of the multiplications; this is called a "sparse" network.

Given a layer of a neural network $ReLU(xW)$ are two well-known ways to prune it:

- Weight pruning: set individual weights in the weight matrix to zero. This corresponds to deleting connections as in the figure above.
    - Here, to achieve sparsity of k% we rank the individual weights in weight matrix W according to their magnitude (absolute value) $|w_{i,j}|$, and then set to zero the smallest k%.
- Unit/Neuron pruning: set entire columns to zero in the weight matrix to zero, in effect deleting the corresponding output neuron.
    - Here to achieve sparsity of k% we rank the columns of a weight matrix according to their L2-norm $|w| = \sqrt{\sum_{i=1}^{N} (x_i)^2}$ and delete the smallest k%.

Naturally, as you increase the sparsity and delete more of the network, the task performance will progressively degrade. What do you anticipate the degradation curve of sparsity vs. performance to be? Your assignment will be to plot this curve for both weight and unit pruning. Compare the curves and make some observations and hypotheses about the observations.

NOTE: your neural network will have 5 weight matrices in total, but the final weight matrix leading to the logits should not be pruned. Also, for simplicity, do not use biases in your network.

For more on pruning see: this post

Here are the steps you should go through:

1. Read the Rules!
2. Install Tensorflow
3. Construct a ReLU-activated neural network with four hidden layers with sizes [1000, 1000, 500, 200]. Note: you'll have a fifth layer for your output logits, which you will have 10 of.
4. Train your network on MNIST or Fashion-MNIST (your choice, whatever is easier)
5. Prune away (set to zero) the k% of weights using weight and unit pruning for k in [0, 25, 50, 60, 70, 80, 90, 95, 97, 99]. Remember not to prune the weights leading to the output logits.
6. Create a table or plot showing the percent sparsity (number of weights in your network that are zero) versus percent accuracy with two curves (one for weight pruning and one for unit pruning).
7. Make your code clean and readable. Add comments where needed.
8. Analyze your results. What interesting insights did you find? Do the curves differ? Why do you think that is/isn't? Do you have any hypotheses as to why we are able to delete so much of the network without hurting performance (this is an open research question)?

Bonus: See if you can find a way to use your new-found sparsity to speed up the execution of your neural net! Hint: ctrl + f "sparse" in the TF docs, or use unit level sparsity (which deletes entire rows and columns from weight matrices). This can be tricky but is a worthwhile engineering lesson in the optimization of Tensorflow models.

Q & A

1. Can I use keras? Yes
2. Can I use PyTorch? Yes
3. Is there a deadline? No
4. Can I borrow some code from other repo? Yes, as long as you borrowed your entire solution and as long as you clearly cite which part in your code is borrowed.