ADAGRAD

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 Adagrad.ai. Deadline for the challenge is one week after you receive the document. You may make the submission before the deadline if you wish so.

Rules

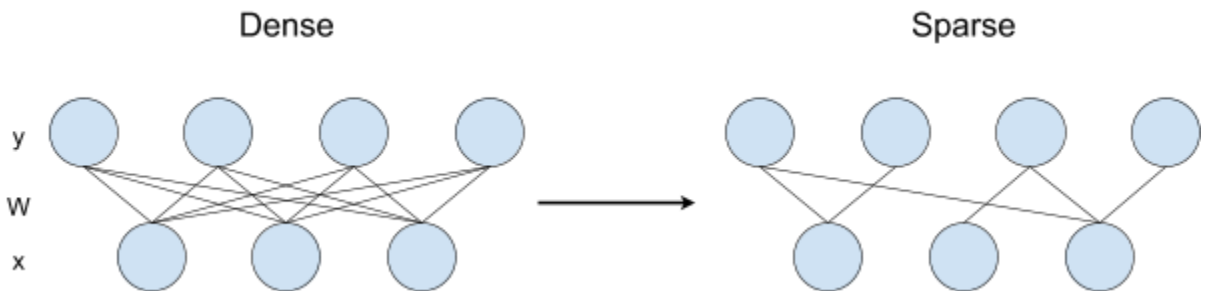
1. Please use Tensorflow or PyTorch (it's what we use for all the projects).
2. While this assignment is relatively simple and doesn't require specialized hardware, you might want to use Colab or Kaggle for access to free GPUs/TPUs which would speed up your productivity as you can run experiments faster than on your laptop/PC.
3. **Bonus**: try solving the challenge **without** using high-level APIs like Keras, FastAI, sklearn.
4. Please do not borrow code from any sources on the internet. **A poor solution is better than a stolen one**. **Plagiarism accounts for immediate rejection.**
5. You should **keep your code simple and focus on the 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. Use proper variable names for functions, variables, and classes. All functions used should have a doc string. Please follow Google's Python Style Guide. We take readability of code very seriously.**
6. You should make your submission via email, please refrain from uploading the code on Github since there may be others who are trying to solve the same challenge.
7. You are expected to learn something new after you complete the challenge :)

Assessment

1. Your solution will be assessed according to the criteria above and we will respond within a week with your results.
2. Primarily, you should focus on getting the output correct, other elements are irrelevant if you fail to solve the task correctly.
3. Please create properly labelled charts/graphs (with labels, legends, and titles) using any charting library of your choice (matplotlib, seaborn, plotly, etc.)
4. Once you're done with solving the challenge, spend some time on optimizing the code. For instance: can this for loop be avoided using a vectorized numpy function?
5. You have an entire week, there are **no bonus points** for submitting the solution before the deadline. A well thought solution is better than a hastily submitted one.

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 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), 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 and delete the smallest k%.

Do not retrain the network after you have pruned it, the entire point in pruning the network is to make many values zero and take advantage of the sparsity.

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.

Here are the steps you should go through:

1. Read the Rules!
2. 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.
3. Train your network on MNIST or Fashion-MNIST (your choice, whatever is easier)
4. 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.
5. 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).
6. Make your code clean and readable. Add comments where needed.
7. Follow the submission guidelines

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 TensorFlow 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 / PyTorch models. Report the increased speed. For example: the performance improved by 25% after using this.

Submission

Please email a zip file with the following contents:

1. Your Jupyter Notebook
2. A word document with the results you observed. 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)? Write a description of roughly 200 words.
3. Include the following header in the results word document mentioned above:
   Name:
   College/graduating year/GPA:
   Phone number:
   Email ID:
4. Name of the ZIP file should be firstname_collegename.zip. For instance, if my name is John and I'm from IIT Bombay then the submission file should be called john_iitbombay.zip

Email should be sent to niranjan@adagrad.ai and CC'ed to mayur@adagrad.ai with subject as "Internship Application: <your full name here>"

Godspeed,

Adagrad AI Team