Sparse Networks and Differential Privacy

Zachary Ross

February 2, 2022

Contents

In the following research, we make use frequently of the set of parameters θ. A couple things to note about how this is used: it is common practice to reference the set of parameter in a multitude of ways, none of which is extrapolated upon in any papers I've come across and has been entirely frustrating to decipher on its own. θ is often referred to a set of parameters rather than a collection, yet is often assigned an ordering of arbitrary dimensionality. I will attempt to disclose my understanding of this dimensionality and these short-hands briefly

1. θ_i is access to some arbitrary element of θ,
2. $\theta_i^{(l)}$ is the i^{th} row in the matrix of parameters at layer l,
3. and $\theta_{i,j}^{(l)}$ is the element in the i^{th} row and j^{th} column of the matrix of parameters at layer l. This is periodically shorthanded to $\theta_{i,j}$ when authors assume an arbitrary layer.

Unless otherwise mentioned, these will be used throughout.

1 Dated Findings

February 2, 2022

In the work presented to me, Dr. Lee primarily uses pruning for gradient masking rather than parameter masking, although most studies I've read seem to have done the opposite. I initially assumed that given the standard linear layer function with masking

$$\boldsymbol{z} = (\mathbf{M} \odot \mathbf{W})\boldsymbol{h} + \boldsymbol{b} \tag{1}$$

the effect of the mask would still result in non-zero values in the gradient of the matrix. This is falsified by the derivation

$$\frac{\partial L}{\partial W_{i,j}} = \frac{\partial L}{\partial z_i}\frac{\partial z_i}{\partial W_{ij}} = M_{ij}\frac{\partial L}{\partial z_i}h_j \tag{2}$$

or in vectorized form we have that

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{M} \odot \left(\frac{\partial L}{\partial \boldsymbol{z}}(\boldsymbol{h})^{\intercal}\right) \tag{3}$$

which implies that the use of a forward mask guarantees that of a backward. The method used instead in Dr. Lee's examples was $\boldsymbol{z} = \mathbf{W}\boldsymbol{h} + \boldsymbol{b}$ with masking only applied during the gradient (i.e. that of Equation 3). So what needs to be tested is whether or not we can apply masking solely during the forward phase i.e. implement Equation 1 with builds its gradient without the mask.

2 Pruning neural networks without any data by iteratively conserving synaptic flow [2]

This paper, while theoretically captivating, is lackluster with providing mathematical commentary on the concepts and proofs. It may be worthwhile to go back and either notate the more mathematical definitions of the paper, or derive my own definitions and reprove their claims using a more solid theoretical foundation. This may even set a better groundwork for future analysis.

2.1 Pruning Algorithms

Pruning algorithms are generally defined by

1. scoring parameters by some metric, and
2. masking parameters according to their scores.

The latter is generally done by removing the parameters (e.g. making the value zero via Hadamard product). This can either be done via global masking or layer-masking, with global-masking performing better but suffering from *layer-collapse*, which is when an entire layer is masked.

Let θ be a collection of network parameters and θ_{prune} be the parameters remaining after pruning. The *compression ratio* ρ of a pruning algorithm is

$$\rho = \frac{|\theta|}{|\theta_{\text{prune}}|} \tag{4}$$

. We define $\rho_{\max}$ as the maximal possible compression ratio for a *network* that doesn't lead to layer collapse and the ρ_{cr} as the maximal compression ratio a given *algorithm* can achieve without inducing layer collapse. Note that the distinction in the two is $\rho_{\max}$ is maximal for a network while ρ_{cr} is maximal for an algorithm. These definitions motivate the following axiom:

Axiom 1 (Maximal Critical Compression)**.** *For any pruning algorithm and any network, we should always have $\rho_{cr} = \rho_{\max}$.*

2.2 Synaptic Saliency

Synaptic saliency is a class of score metrics defined by

$$\mathcal{S}(\theta) = \frac{\partial \mathcal{R}}{\partial \theta} \odot \theta \tag{5}$$

where $\mathcal{R}$ is a scalar loss function of the output y of a feed-forward network parameterized by θ. This metric satisfies two conservation laws:

Theorem 1 (Neuron-wise Conservation of Synaptic Saliency)**.** *For a feedforward neural network with continuous, homogeneous activation functions $\phi(x) = \phi'(x)x$, let j be the index of a hidden neuron in layer i. The sum of the synaptic saliency for the incoming parameters to a hidden neuron $\left(\mathcal{S}_j^{(l)_{in}} = \langle \frac{\partial \mathcal{R}}{\partial \theta_j^{(l)}}, \theta_j^{(l)} \rangle\right)$ is equal to the sum of the synaptic saliency for the outgoing parameters from the hidden neuron $\left(\mathcal{S}_j^{(l)_{out}} = \langle \frac{\partial \mathcal{R}}{\partial \theta_{:,j}^{(l+1)}}, \theta_{:,j}^{(l+1)} \rangle\right)$.*

Theorem 2 (Network-wise Conservation of Synaptic Saliency)**.** *The sum of the synaptic saliency across any set of parameters that exactly separates the input neurons x from the output neurons y of a feedforward neural network with homogeneous activation functions equals $\langle \frac{\partial \mathcal{R}}{\partial x}, x \rangle = \langle \frac{\partial \mathcal{R}}{\partial y}, y \rangle$.*

2.3 Algorithm

For the following theorem, we define the *prune size* to be the total score for the parameters pruned at any iteration and the *cut size* to be the total score for an entire layer.

Theorem 3. *If a pruning algorithm with global-masking assigns positive scores that respect layer-wise conservation, and if the prune size is strictly less than the cut size whenever possible, then the algorithm satisfies the Maximal Critical Compression axiom.*

The pruning algorithm defined in this paper uses a loss function

$$\mathcal{R}_{\text{SF}} = \mathbb{1}^\intercal \left(\prod_{l=1}^{L} |\theta^{(l)}| \right) \mathbb{1} \tag{6}$$

so that for a fully connected network $\left(f_\theta(x) = \theta^{(L)} \dots \theta^{(1)} \boldsymbol{x}\right)$ the Synaptic Flow score for a parameter $\theta_{i,j}^{(l)}$ is

$$\mathcal{S}_{\text{SF}}(\theta_{i,j}^{(l)}) = \left[\mathbb{1}^\intercal \prod_{k=l+1}^{L} |\theta^{(k)}| \right]_i |\theta_{i,j}^{(l)}| \left[\prod_{k=1}^{l-1} |\theta^{(k)}| \mathbb{1} \right]_j \tag{7}$$

i.e. the portion of the l_1-path norm the network has through this parameter. This contributes to the development of Algorithm 1.

Algorithm 1 Iterative Synaptic Flow Pruning (SynFlow)

```
Require: network $f_\theta$, compression ratio $\rho$, iteration steps $n$
  $\mu = \mathbb{1}$                                                   ▷ Initialize binary mask
  for $k$ in $[1, \dots, n]$ do
    $\theta_\mu \leftarrow \mu \odot \theta_0$                          ▷ Mask parameters
    $\mathcal{R} \leftarrow \mathbb{1}^\intercal \left(\prod_{l=1}^{L} |\theta_\mu^{(l)}|\right) \mathbb{1}$    ▷ Evaluate SynFlow objective
    $\mathcal{S} = \frac{\partial \mathcal{R}}{\partial \theta_\mu} \odot \theta_\mu$    ▷ Compute SynFlow score
    $\tau \leftarrow \left(1 - \rho^{-k/n}\right)$ percentile of $\mathcal{S}$    ▷ Find threshold
    $\mu \leftarrow (\tau < \mathcal{S})$                               ▷ Update mask
  end for
  return $f_{\mu \odot \theta}$                                         ▷ Return masked network
```

2.4 Commentary

A thought I had: the loss function defined in Equation 6 seems to almost indicate a just-past-the-minimum metric for the predefined network. It effectively is just the sum of the entries in the product of the network space. Although I agree with the data agnostic approach, I feel there should be a more suitable loss function which should seek to preserve this score rather than minimize it, or at the very least, should compare this score against some optimum, i.e. the optimum score for that network under those parameters. I also feel that the prune size and cut size should have a greater impact over the outcome of the resulting network, such that the compression ratio is fairly equal for all layers.

Research Question 1. *Does SynFlow, or other pruning methods for that matter, produce the same network given different initial parameterizations? Furthermore, might we be able to use a genetic algorithm to find the "lottery ticket"?*

One thing that needs to be considered with this question is that it's entirely possible that isomorphic networks could be produced, so there may need to be some kind of check that occurs post pruning that confirms whether or not this network has been discovered already.

3 The Lottery Ticket Hypothesis: Training Pruned Neural Networks [1]

References

[1] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Training Pruned Neural Networks". In: *CoRR* abs/1803.03635 (2018). arXiv: `1803.03635`. URL: `http://arxiv.org/abs/1803.03635`.

[2] Hidenori Tanaka et al. "Pruning neural networks without any data by iteratively conserving synaptic flow". In: *CoRR* abs/2006.05467 (2020). arXiv: `2006.05467`. URL: `https://arxiv.org/abs/2006.05467`.