General Research Notes

Zachary Ross

February 24, 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.

February 9, 2022

Thorough testing has disproved our previous hypothesis. Strangely enough, the gradient masking is much more effective than weight masking for some odd reason.

February 23, 2022

In the following discussion, I use beats per minute (BPM) to refer to heart rate. I only do it this way because I've already written it out and don't feel like changing it.

I've been pondering a bit recently on a hypothesis I've had, that being the perception of time is relative to the individual based on several factors. At first I believed that it's connected to BPM, but this was disproved by my addiction to coffee. I've noticed this relativity even in my day to day, some days seem faster than others while certain activities seem to be slower. For example, under the influence of coffee I feel that my day goes by much faster, which leads me to believe the effects of caffeine are misunderstood (which I will go into later). After or during an intense workout, my perception of time seems to slow way the hell down. I notice this because I'll be listening to music, and maybe it has to do with my percussionist background with intense focus on steady timing, but the songs pass by much slower. This is incredibly curious.

On other days where I don't quite feel myself, I notice instead that the music tends to be much quicker, often times leading me to feel like I'm missing alot.

For the coffee thing, maybe its affect on BPM does slow down perception of time, but caffeine could effectively and periodically cut out clips of time, making us believe that time is moving faster. I believe this is affects productivity because it allows us to move from task to task seemingly "quicker". I don't believe it fully alters our productivity; rather, it seems like it allows us to reach the reward much quicker and thereby reinforce the thought process that we are "getting shit done". This activity/reward behavior most likely becomes a self-fulfilling cycle as the momentum continues, motivating us to continue doing tasks.

I've also considered how we would test these hypotheses because it's rather elusive. Say we tried to get a set of people to listen to a meteronome prior to an energy intensive activity and post said activity and asked them to compare the two for which one is faster. For one, how would we get reliable information? If we were just to pick random people from a population, they may not even be capable of discerning between two tempos. I suppose we could do a preliminary exam where we vet out those who could not tell the difference even between minor changes in tempo. We could pick out trained musicians, but then the study is limited to musicians techinically. I guess we would just have to do the vetting procedure and only select the upper echelon of people which can tell extremely minute differences in tempo after a select period of time.

Thoughts I had while working out, lol.

2 Algorithms and Theory

3 Neural Network Pruning

3.1 Pruning neural networks without any data by iteratively conserving synaptic flow [4]

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.

3.1.1 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 $\rho_{cr} = \rho_{\max}$.*

3.1.2 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 $\langle \frac{\partial \mathcal{R}}{\partial x}, x \rangle = \langle \frac{\partial \mathcal{R}}{\partial y}, y \rangle$.*

3.1.3 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
```

3.1.4 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.

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

4 Gradients

4.1 Efficient Per-Example Gradient Computations [2]

An important note about this paper is that it is *not* a derivation of the per-example gradient itself. Rather, it "describes an efficient technique for computing the *norm* of the gradient" with respect to the loss function.

Let each layer of a neural network be defined by the standard transformations

$$\begin{aligned} \boldsymbol{z}^{(i)} &= \mathbf{W}^{(i)}\boldsymbol{h}^{(i-1)} \\ \boldsymbol{h}^{(i)} &= \phi^{(i)}\left(\boldsymbol{z}^{(i)}\right) \end{aligned} \tag{8}$$

with the bias assumed to be an extra row/column of $\mathbf{W}$ and loss function $\mathcal{L}$. Let $\mathcal{B} \subset \mathcal{D}$ be a minibatch of the input dataset $\mathcal{D}$, and let $\mathcal{B}^{(j)}$ be the jth example in the minibatch with $\mathcal{L}^{(j)}$ corresponding to the loss of $\mathcal{B}^{(j)}$ and cost function $C = \sum_{j=1}^{n} \mathcal{L}^{(j)}$. The per example gradient norm is computed with respect both to the example and the layer, such that the gradient norm for layer i and example j is

$$\left\| \frac{d\mathcal{L}^{(j)}}{d\mathbf{W}^{(i)}} \right\|^2 = \sum_{k,l} \left(\frac{\partial \mathcal{L}^{(j)}}{\partial W_{k,l}^{(i)}} \right)^2, \tag{9}$$

which is just the frobenius norm. Note that this equation can be used to calculate the L_2 norm of the parameter gradient for an example or the L_2 norm of the gradient for an individual weight matrix by summing over j or i, respectively.

The proposed method makes use of $\mathbf{H}^{(i)}$ whose jth row contains the activation layer $\boldsymbol{h}^{(i)}$ corresponding to $\mathcal{B}^{(j)}$ i.e. $\mathbf{H}_j^{(i)} = \phi^{(i)}\left(\mathbf{W}^{(i)}\phi^{(i-1)}\left(\dots\left(\mathbf{W}^{(1)}\mathcal{B}^{(j)}\right)\dots\right)\right)$. Likewise, define $\mathbf{Z}^{(i)}$ for each $\boldsymbol{z}^{(i)}$ and compute $\bar{\mathbf{Z}} = \nabla_{\mathbf{Z}} C$, which can be computed in a single pass using standard backpropagation. This gives us

$$\left\| \frac{\partial \mathcal{L}^{(j)}}{\partial \mathbf{W}^{(i)}} \right\|^2 = \left(\sum_k \left(\bar{Z}_{j,k}^{(i)} \right)^2 \right) \left(\sum_k \left(H_{j,k}^{(i-1)} \right)^2 \right) = \left\| \bar{\boldsymbol{Z}}_j^{(i)} \right\|^2 \left\| \boldsymbol{H}_j^{(i-1)} \right\|^2. \tag{10}$$

4.1.1 Commentary

This method works well with gradient clipping due to the efficient computation of gradient norms. This calculation allows for simple re-scaling of the gradients used in backwards propagation.

4.2 Understanding gradient clipping in incremental gradient methods [3]

This paper considers the class of problems dealing with the minimization of

$$\min_{x\in\mathbb{R}^n} f(x) \qquad \text{where} \qquad f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x), \tag{11}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ are differentiable, non-convex functions. As a side note, this differs from notation that I'm used to, in that x is considered to be the set of parameters while i, or more importantly f_i, is a function of the parameters using the i^{th} component of the dataset to measure the loss. That is, the function $f_i(x)$ determines the loss accumulated when using x as the set of parameters on input i. f is said to be the *objective function* while each f_i is called a *component function*.

The incremental method for SGD is

$$x_{k+1} = x_k - \alpha_k \nabla f_{i_k}(x_k) \quad k = 0, 1, \dots \tag{12}$$

where x_k is the values of the parameters at ieration k, α_k is the learning rate, and f_{i_k} is a uniform randomly chosen component function $i_k \in \{1, 2, \dots, m\}$ of the dataset. The incremental method for IGC is

$$x_{k,i} = x_{k,i-1} - \alpha_k \nabla f_i(x_{k,i-1}) \quad i = 1, \dots, m \quad k = 0, 1, \dots \tag{13}$$

where the recursion is defined by base case $x_{0,0}$ and iterative step $x_{k,0} = x_{k-1,m}$ and is more cyclical by nature.

The clipping function $\mathcal{C} : \mathbb{R}^n \to \mathbb{R}^n$ with clipping threshold parameter $\eta > 0$ is defined by

$$\mathcal{C}(g;\eta) = \min\left\{1, \frac{\eta}{\|g\|}\right\} \cdot g. \tag{14}$$

Our iterative methods can take advantage of clipping by clipping the gradient so that SGD with clipping is

$$x_{k+1} = x_k - \alpha_k \mathcal{C}(\nabla f_{i_k}(x_k);\eta), \tag{15}$$

and IGC with clipping is

$$x_{k,i} = x_{k,i-1} - \alpha_k \mathcal{C}(\nabla f_i(x_{k,i-1});\eta). \tag{16}$$

For the theoretical understanding of the clipping function's effects on the convergence of both SGD and IGC, we make the following assumptions:

Bounded below there exists an f^* such that for all $x \in \mathbb{R}^n$, $f(x) \geq f^*$;

Relaxed smoothness there exist constants $L_0, L_1 \in \mathbb{R}^+$ such that $\|\nabla^2 f(x)\| \leq L_0 + L_1\|\nabla f(x)\|$.

The second condition becomes Lipschitz smoothness when $L_1 = 0$. Let $f_k = f(x_k)$ and assume for simplicity from here on out that SGD and IGC have a constant step size α and that IGC's notation for $x_{k,i}$ can be simplified for $x_{k,0}$ to x_k. The paper shows that convergence for the algorithm can be ensured if there exists a constant C such that

$$\langle \sum_{i=1}^{m} \mathcal{C}(\nabla f_i(x_k);\eta), \nabla f_k\rangle \geq C\|\nabla f_k\|^2 \tag{17}$$

i.e. if the magnitude of the clipped gradient projected in the direction of the regular gradient is bounded below by the magnitude of the regular gradient. This value is simplified further as follows.

Let $g_{ki} = \nabla f_i(x_k)$ and note the following properties of g_{ki}

$$g_{ki} = \beta_{ki}\nabla f_k + t_{ki} \quad \text{where} \quad \begin{array}{l} \beta_{ki} = \dfrac{\langle g_{ki}, \nabla f_k\rangle}{\|\nabla f_k\|^2} = \dfrac{\|g_{ki}\|}{\|\nabla f_k\|}\cos\theta_{ki} \\ \\ t_{ki} \in \nabla f_k^{\perp} = \{z | \langle z, \nabla f_k\rangle = 0\} \end{array}. \tag{18}$$

Since $\frac{1}{m}\sum_i^m g_{ki} = \nabla f_k$, it is a necessary condition that $\sum_i^m \beta_{ki} = m$ and $\sum_i^m t_{ki} = 0$. Let $\gamma_{ki} = \min\{1, \eta/\|g_{ki}\|\}$ and note that $\mathcal{C}(g_{ki};\eta) = \gamma_{ki} g_{ki}$. The paper then shows that

$$\langle \sum_{i=1}^{m} \mathcal{C}(\nabla f_i(x_k);\eta), \nabla f_k \rangle = \sum_{i=1}^{m} \gamma_{ki}\beta_{ki}\|\nabla f_k\|^2. \tag{19}$$

so that Equation 17 becomes

$$\sum_{i=1}^{m} \gamma_{ki}\beta_{ki} \geq C, \tag{20}$$

that is, *the necessary condition for convergence is that the sum in Equation 20 is greater than some positive scalar C.*

5 Finance

5.1 An Introduction to Quantitative Finance

5.1.1 Interest Rates

Definition 1. A *notional* or *principal* is an initial deposit or value N.

Definition 2. An *interest rate* r is the rate at which a value is increased according to a specified frequency of time.

Suppose we have an account with interest rate r and notional N. The value of the account at time T is

$$N_T = Ne^{rT} \tag{21}$$

Throughout finance, there exist discrete and continuous analogs for most equations due the discrete case being more realistic while the continuous case is far more mathematically efficient. That being said, discrete analogs will be used sparsely and we will primarily depend on continuous equations. The discrete versions can be derived from the continuous versions by noting that

$$e^{rT} = \left(1 + \frac{r_m}{m}\right)^{mT} \tag{22}$$

where T is commonly denoted as the time in years, m is the frequency the interest is compounded, and r_m is the equivalent interest rate with the discrete frequency. r_m can then be recovered from r using

$$r_m = m\left(e^{\left(\frac{r}{m}\right)} - 1\right). \tag{23}$$

Definition 3. A *money market account* is an account compounded continuously at rate r with notional 1. I.e. $M_0 = 1$ and $M_t = e^{rt}$.

Definition 4. A *zero coupon bond* (ZCB) with maturity T is an asset that pays 1 at time T (and nothing else). The value of a ZCB at time t for a continuously compounded rate r is denoted as

$$Z(t,T) = e^{-r(T-t)} \tag{24}$$

where $Z(T,T) = 1$ by definition. The values $Z(t,T)$ with $0 \leq t \leq T$ are known as *discount factors* or *present values.*

The intuition behind the present values for a ZCB derives from the fact that if we have two portfolios, one a ZCB with maturity T and another that contains cash which will accumulate interest to be worth 1 at time T, then both will be worth the same price at each t leading up to T and thereafter.

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] Ian Goodfellow. *Efficient Per-Example Gradient Computations.* 2015.

[3] Jiang Qian et al. “Understanding gradient clipping in incremental gradient methods”. In: *International Conference on Artificial Intelligence and Statistics.* PMLR. 2021, pp. 1504–1512.

[4] 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.