

Can Contrastive Learning Use Gradient Accumulation?

Jianlin Su

June 17, 2021

In a previous article, ["Trading Time for Quality: Keras Gradient Accumulation Optimizer"](#), we introduced "gradient accumulation," a technique to achieve the effect of a large `batch_size` under limited VRAM. Generally, gradient accumulation is applicable to scenarios where the loss is independent and identically distributed (i.i.d.). In other words, the loss is calculated for each sample individually, and the total loss is the average or sum of all individual losses. However, not all tasks satisfy this condition. For instance, in the currently popular field of contrastive learning, the loss for each sample is related to other samples in the batch.

So, in the context of contrastive learning, can we still use gradient accumulation to achieve the effect of a large `batch_size`? This article analyzes this question.

1 Introduction

In general, the loss for contrastive learning can be written as:

$$\mathcal{L} = - \sum_{i,j=1}^b t_{i,j} \log p_{i,j} = - \sum_{i,j=1}^b t_{i,j} \log \frac{e^{s_{i,j}}}{\sum_j e^{s_{i,j}}} = - \sum_{i,j=1}^b t_{i,j} s_{i,j} + \sum_{i=1}^b \log \sum_{j=1}^b e^{s_{i,j}} \quad (1)$$

Here, b is the `batch_size`; $t_{i,j}$ is a pre-given label satisfying $t_{i,j} = t_{j,i}$, which is a one-hot matrix where each row/column has only one 1 and the rest are 0; and $s_{i,j}$ is the similarity between sample i and sample j , satisfying $s_{i,j} = s_{j,i}$. Usually, there is also a temperature parameter, but here we assume the temperature parameter is already integrated into $s_{i,j}$ to simplify notation. The model parameters exist within $s_{i,j}$, denoted as θ .

It can be verified that, in general:

$$- \sum_{i,j=1}^{2b} t_{i,j} \log p_{i,j} \neq - \sum_{i,j=1}^b t_{i,j} \log p_{i,j} - \sum_{i,j=b+1}^{2b} t_{i,j} \log p_{i,j} \quad (2)$$

Therefore, directly accumulating the gradients of contrastive learning with a small `batch_size` is not equivalent to contrastive learning with a large `batch_size`. A similar problem exists in models with Batch Normalization (BN).

2 Gradient

Note that while we just stated that conventional simple gradient accumulation is not equivalent, a slightly more complex accumulation scheme might exist. To this end, let us analyze the

gradient of Equation (1):

$$\begin{aligned}
\nabla_{\theta} \mathcal{L} &= - \sum_{i,j=1}^b t_{i,j} \nabla_{\theta} s_{i,j} + \sum_{i=1}^b \nabla_{\theta} \log \sum_{j=1}^b e^{s_{i,j}} \\
&= - \sum_{i,j=1}^b t_{i,j} \nabla_{\theta} s_{i,j} + \sum_{i,j=1}^b p_{i,j} \nabla_{\theta} s_{i,j} \\
&= \nabla_{\theta} \sum_{i,j=1}^b \left(p_{i,j}^{(sg)} - t_{i,j} \right) s_{i,j}
\end{aligned} \tag{3}$$

Where $p_{i,j}^{(sg)}$ indicates that the gradient of $p_{i,j}$ with respect to θ is not required, which is the **stop_gradient operator** in deep learning frameworks. The above equation shows that if we use a gradient-based optimizer, using Equation (1) as the loss is completely equivalent to using $\sum_{i,j=1}^b \left(p_{i,j}^{(sg)} - t_{i,j} \right) s_{i,j}$ as the loss (because the calculated gradients are identical).

3 Inner Product

Next, consider the calculation of $\nabla_{\theta} s_{i,j}$. Generally, it takes the form of a vector inner product, i.e., $s_{i,j} = \langle h_i, h_j \rangle$, where the parameter θ is inside h_i and h_j . In this case:

$$\nabla_{\theta} s_{i,j} = \langle \nabla_{\theta} h_i, h_j \rangle + \langle h_i, \nabla_{\theta} h_j \rangle = \nabla_{\theta} \left(\langle h_i, h_j^{(sg)} \rangle + \langle h_i^{(sg)}, h_j \rangle \right) \tag{4}$$

Therefore, $s_{i,j}$ in the loss can be replaced by $\langle h_i, h_j^{(sg)} \rangle + \langle h_i^{(sg)}, h_j \rangle$ without changing the effect:

$$\begin{aligned}
\nabla_{\theta} \sum_{i,j=1}^b \left(p_{i,j}^{(sg)} - t_{i,j} \right) s_{i,j} &= \nabla_{\theta} \sum_{i,j=1}^b \left(p_{i,j}^{(sg)} - t_{i,j} \right) \left(\langle h_i, h_j^{(sg)} \rangle + \langle h_i^{(sg)}, h_j \rangle \right) \\
&= 2 \nabla_{\theta} \sum_{i,j=1}^b \left(\overline{p_{i,j}^{(sg)}} - t_{i,j} \right) \langle h_i, h_j^{(sg)} \rangle \\
&= \nabla_{\theta} \sum_{i=1}^b \left\langle h_i, 2 \sum_{j=1}^b \left(\overline{p_{i,j}^{(sg)}} - t_{i,j} \right) h_j^{(sg)} \right\rangle
\end{aligned} \tag{5}$$

Where $2\overline{p_{i,j}^{(sg)}} = p_{i,j}^{(sg)} + p_{j,i}^{(sg)}$. The second equality stems from swapping the summation indices i, j for the $\langle h_i^{(sg)}, h_j \rangle$ term, which does not change the result of the sum.

4 Process

Equation (5) actually provides the final scheme, which can be divided into two steps. The first step is the calculation of the vector:

$$\tilde{h}_i = 2 \sum_{j=1}^b \left(\overline{p_{i,j}^{(sg)}} - t_{i,j} \right) h_j^{(sg)} \tag{6}$$

This step does not require gradient calculation and is purely a prediction process, so the **batch_size** can be relatively large. The second step is to pass $\tilde{h}_i$ into the model as a "label" and optimize the model using $\langle h_i, \tilde{h}_i \rangle$ as the loss for a single sample. This step requires gradient calculation, but it has been transformed into the form of a sum of gradients for each sample, so conventional gradient accumulation can be used here.

Assuming the maximum `batch_size` for backpropagation is b , and the maximum `batch_size` for forward propagation is nb , the formatted process for achieving the effect of a `batch_size` of nb in contrastive learning via gradient accumulation is as follows:

1. Sample a batch of data $\{x_i\}_{i=1}^{nb}$, with the corresponding label matrix $\{t_{i,j}\}_{i,j=1}^{nb}$, and initialize the accumulated gradient as $g = 0$;
2. Perform a forward pass of the model to obtain the encoding vectors $\{h_i\}_{i=1}^{nb}$ and the corresponding probability matrix $\{p_{i,j}\}_{i,j=1}^{nb}$;
3. Calculate the label vectors $\{\tilde{h}_i\}_{i=1}^{nb}$ according to Equation (6);
4. For $k = 1, 2, \dots, n$, execute:

$$g \leftarrow g + \nabla_{\theta} \sum_{i=(k-1)b+1}^{kb} \langle h_i, \tilde{h}_i \rangle$$

5. Use g as the final gradient to update the model, then restart from Step 1.

Overall, in terms of computation, this requires one extra forward pass compared to conventional gradient accumulation. Of course, if the maximum `batch_size` for forward propagation still does not meet our needs, we can perform forward passes in batches, because we only need to calculate and store each $\{h_i\}_{i=1}^{nb}$, and $\{p_{i,j}\}_{i,j=1}^{nb}$ can be calculated based on $\{h_i\}_{i=1}^{nb}$.

Finally, it should be noted that the above process is only equivalent to a large `batch_size` model during optimization. That is, the gradient of $\langle h_i, \tilde{h}_i \rangle$ is equivalent to the gradient of the original loss, but its value is not equal to the value of the original loss. Therefore, $\langle h_i, \tilde{h}_i \rangle$ cannot be used as a loss to evaluate the model; it is not necessarily monotonic, nor necessarily non-negative, and it does not have a strict correlation with the original loss.

5 Problems

The above process has the same problem as the "recomputation technique" introduced in "[Keras Version of Memory-Saving Recomputation Technique is Here](https://kexue.fm/archives/8471)": it is incompatible with Dropout. This is because each update involves multiple forward passes, and each forward pass has a different Dropout mask. This means that the h_i used when calculating the label vector $\tilde{h}_i$ is not the same as the h_i used when calculating the gradient, resulting in a gradient that is not the most reasonable one.

There is no great solution for this; the simplest and most effective method is to remove Dropout from the model. This is not a major issue for Computer Vision (CV), as CV models basically no longer use Dropout. For Natural Language Processing (NLP), the first consequence that comes to mind is that [SimCSE](#) cannot use gradient accumulation, because Dropout is the foundation of SimCSE.

6 Summary

This article analyzes the gradient accumulation method for contrastive learning. The results show that contrastive learning can indeed use gradient accumulation, albeit with an extra forward pass and the requirement to remove Dropout from the model. The same logic in this article can also be used to analyze how BN can use gradient accumulation; interested readers might want to try it out.

*When reprinting, please include the original address of this article: <https://kexue.fm/archives/8471>
For more detailed reprinting matters, please refer to: "Scientific Space FAQ"*