Median Batch Normalization

Tyler King

July 2022

1 Mean Batch Normalization

With classical mean batch normalization, the feedforward transform is defined by the following functions over tunable parameters γ and β and a mini-batch:

$$\mu_\beta \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_\beta^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\beta)^2$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta$$

For training, backpropagating the gradient through the transformation is a necessity. Ioffe and Szegedy [1] compute these gradients of loss ℓ via chain rule:

$$\frac{\partial \ell}{\partial \hat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_\beta^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i}(x_i - \mu_\beta) \cdot \frac{-1}{2}(\sigma_\beta^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_\beta} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} \frac{-1}{\sqrt{\sigma_\beta^2 + \epsilon}}\right) + \frac{\partial \ell}{\partial \sigma_\beta^2} \frac{\sum_{i=1}^{m} -2(x_i - \mu_\beta)}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \hat{x}_i} \frac{1}{\sqrt{\sigma_\beta^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_\beta^2} \frac{2(x_i - \mu_\beta)}{m} + \frac{\partial \ell}{\partial \mu_\beta} \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

Thus, to compute the forward and backpropagation components of the proposed median batch normalization, we replace these sets of equations.

2 Forward Computation for Median Batch Normalization

The key difference between the transform that Ioffe and Szegedy [1] proposed and the median batch normalization we propose comes in the computation of the μ_β. Instead of utilizing μ_β (mini-batch mean), we use M_β (median of mini batch). This can be represented as the 1-dimensional case of the geometric median, defined as

$$\operatorname*{arg\,min}_{y \in \mathbb{R}^n} \sum_{i=1}^{m} \|x_i - y\|_2$$

where each x_i represents a value of the mini-batch and the argument y that minimizes this function is M_β.

Remark. *While the 1-dimensional case represents a way to compute the median, we plan to use* `torch.median` *for forward computation (due to lower overhead costs).*

All other equations from the feedforward function remain the same, but with μ_β being replaced with M_β.

3 Backpropagation for Median Batch Normalization

Starting with the easiest derivations,

$$\begin{aligned}\frac{\partial \ell}{\partial \hat{x}_i} &= \frac{\partial \ell}{\partial y_i} \frac{\partial y_i}{\partial \hat{x}_i} \\ &= \frac{\partial \ell}{\partial y_i} \cdot \gamma\end{aligned}$$

The derivatives of loss with respect to tunable parameters also do not change:

$$\begin{aligned}\frac{\partial \ell}{\partial \beta} &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \frac{\partial y_i}{\partial \beta} \\ &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}\end{aligned}$$

and

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \frac{\partial y_i}{\partial \gamma}$$

$$= \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \hat{x}_i$$

Next, we compute $\frac{\partial \ell}{\partial \sigma_\beta^2}$ and $\frac{\partial \ell}{\partial M_\beta}$ as follows.

$$\begin{aligned}\frac{\partial \ell}{\partial \sigma_\beta^2} &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} \frac{\partial \hat{x}_i}{\partial \sigma_\beta^2} \\ &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} \cdot (x_i - M_\beta) \cdot \frac{-1}{2} (\sigma_\beta^2 + \epsilon)^{-3/2}\end{aligned}$$

$$\begin{aligned}\frac{\partial \ell}{\partial M_\beta} &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} \frac{\partial \hat{x}_i}{\partial M_\beta} + \frac{\partial \ell}{\partial \sigma_\beta^2} \frac{\partial \sigma_\beta^2}{\partial M_\beta} \\ &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} \frac{-1}{\sqrt{\sigma_\beta^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_\beta^2} \frac{\sum_{i=1}^{m} -2(x_i - M_\beta)}{m}\end{aligned}$$

These equations remain mostly the same for both classical batch normalization and median batch normalization, with the small caveat that the mean μ_β is replaced with the median M_β. These prior computations also allow us to calculate $\frac{\partial \ell}{\partial x_i}$.

Again referencing Ioffe and Szegedy [1], we know that

$$\begin{aligned}\frac{\partial \ell}{\partial x_i} &= \frac{\partial \ell}{\partial \hat{x}_i} \frac{\partial \hat{x}_i}{\partial x_i} + \frac{\partial \ell}{\partial \sigma_\beta^2} \frac{\partial \sigma_\beta^2}{\partial x_i} + \frac{\partial \ell}{\partial M_\beta} \frac{\partial M_\beta}{\partial x_i} \\ &= \frac{\partial \ell}{\partial \hat{x}_i} \frac{1}{\sqrt{\sigma_\beta^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_\beta^2} \frac{2(x_1 - M_\beta)}{m} + \frac{\partial \ell}{\partial M_\beta} \frac{\partial M_\beta}{\partial x_i}\end{aligned}$$

3.1 Differentiating through the Geometric Median

To compute $\frac{\partial M_\beta}{\partial x_i}$, we reference Gould et al. [2] and their approach to differentiation through the arg min. Using similar notation, we call M_β the median of the set of points $\{h_i(x)\}_{i=1}^m$. We then define a function $g(x)$ such that

$$g(x) = \arg\min_y f(x, y)$$

where

$$f(x, y) = \sum_{i=1}^{m} \|x_i - y\|$$

In the 1-dimensional case, an analytical solution exists: $g(x) = M_\beta$.

Applying Lemma 3.1 from Gould et al. [2], we know that for some $g(x) = \arg\min_y f(x, y)$, that

$$g'(x) = -\frac{f_{XY}(x, g(x))}{f_{YY}(x, g(x))}$$

First, we compute $f_{YY}(x, g(x))$:

$$\begin{aligned}
f_Y &= \frac{\partial}{\partial y}\left(\sum_{i=1}^{m} \|h_i(x) - y\|\right) \\
&= \sum_{i=1}^{m} -\frac{1}{\|h_i(x) - y\|} \\
f_{YY} &= \sum_{i=1}^{m} -\frac{-(-1)}{2\|h_i(x) - y\|^3} \\
&= \sum_{i=1}^{m} -\frac{1}{2\|h_i(x) - y\|^3}
\end{aligned}$$

Then we compute $f_{XY}(x, g(x))$: Starting with

$$f_Y = \sum_{i=1}^{m} -\frac{1}{\|h_i(x) - y\|}$$

from earlier, we compute f_{XY}:

$$f_{XY} = \sum_{i=1}^{m} \frac{h_i'(x)}{2\|h_i(x) - y\|^3}$$

This gives us

$$g'(x) = \frac{\sum_{i=1}^{m} \frac{h_i'(x)}{2\|h_i(x)-y\|^3}}{-\sum_{i=1}^{m} \frac{1}{2\|h_i(x)-y\|^3}}$$

which is functionally equivalent to

$$\frac{\partial M_\beta}{\partial x} = -\frac{\sum_{i=1}^{m} \frac{h_i'(x)}{\|h_i(x)-M_\beta\|^3}}{\sum_{i=1}^{m} \frac{1}{\|h_i(x)-M_\beta\|^3}}$$

Since we are only looking for the derivative with respects to a single $h_i(x)$ (i.e. we are only looking at a single index k) we can state that $h_i'(x) = 0$ (since it is a derivative of a constant with respect to $h_k(x)$) for all $i \neq k$ and $h_i'(x) = 1$ for $i = k$. More formally,

$$\begin{aligned}
h_k'(x) &= \lim_{h \to 0} \frac{h_k(x+h) - h_k(x)}{h} \\
&= \lim_{h \to 0} \frac{x + h - x}{h} \\
&= 1
\end{aligned}$$

This gives us

$$\frac{\partial M_\beta}{\partial x_k} = \frac{-\frac{1}{\|h_k(x)-M_\beta\|^3}}{\sum_{i=1}^{m} \frac{1}{\|h_i(x)-M_\beta\|^3}}$$

which means that our loss derivative with respect to a particular x_i is

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \hat{x}_i} \frac{1}{\sqrt{\sigma_\beta^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_\beta^2} \frac{2(x_1 - M_\beta)}{m} - \frac{\partial \ell}{\partial M_\beta} \frac{\frac{1}{\|h_i(x)-M_\beta\|^3}}{\sum_{j=1}^{m} \frac{1}{\|h_j(x)-M_\beta\|^3}}$$

While this is computationally expensive, it is worth noting that the denominator of $\frac{\partial M_\beta}{\partial x_i}$ only has to be computed once and then can be reused for all x_i (since it is a constant), so the overhead is not terrible.

References

[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[2] Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *CoRR*, abs/1607.05447, 2016.