Derivative expressions for some normalization operations based on BatchNorm

Kadulin V.

October 18, 2022

Let's remind BatchNorm for 2-D case: $Y = \gamma \hat{X} + \beta$, $\hat{X} \in \mathbb{R}^{N \times D}$, $\gamma \in \mathbb{R}^D$, $\beta \in \mathbb{R}^D$, $\hat{X} = \frac{X-\mu}{\sigma}$, $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$, $v = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$, $\sigma = \sqrt{v+\epsilon}$, $\epsilon \in \mathbb{R}$ is a small scalar to omit zero division.

Let $f : \mathbb{R}^{N \times D} \to \mathbb{R}$ is a differentiable scalar function, and $\frac{\partial f}{\partial Y}$ is known.

BatchNorm operation is applied columnwise in 2-D case. So, in case of k-th column the derivative expressions looks as follows:

$$\frac{\partial f}{\partial \gamma} = \sum_{i=1}^{N} \frac{\partial f}{\partial y_i}\frac{\partial y_i}{\partial \gamma} = \sum_{i=1}^{N} \frac{\partial f}{\partial y_i}\hat{x}_i$$

$$\frac{\partial f}{\partial \beta} = \sum_{i=1}^{N} \frac{\partial f}{\partial y_i}\frac{\partial y_i}{\partial \beta} = \sum_{i=1}^{N} \frac{\partial f}{\partial y_i}$$

$$\frac{\partial f}{\partial x_j} = \sum_{i=1}^{N} \frac{\partial f}{\partial y_i}\frac{\partial y_i}{\partial \hat{x}_i}\frac{\partial \hat{x}_i}{\partial x_j} =$$

$$\gamma(v+\epsilon)^{-\frac{1}{2}}\left(-\frac{1}{N}\sum_{i=1}^{N}\frac{\partial f}{\partial y_i} - (x_j - \mu)(v+\epsilon)^{-1}\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)\frac{\partial f}{\partial y_i} + \frac{\partial f}{\partial y_j}\right)$$

where $y, x, \hat{x} \in \mathbb{R}^N$ represents the k-th column of $Y, X, \hat{X}$ respectively, $\gamma, \beta, \mu, v, \sigma \in \mathbb{R}$ represents the k-th values of the same named vectors.

Let's derive the same expressions for the following normalization methods: LayerNorm, spatial BatchNorm, GroupNorm, InstanceNorm.

LayerNorm

Differences from BatchNorm:

- x is a row of X
- $\mu = \frac{1}{D}\sum_{i=1}^{D} x_i$
- $v = \frac{1}{D}\sum_{i=1}^{D}(x_i - \mu)^2$

This implies the following changes:

- γ is a $\mathbb{R}^D$ vector
- $\frac{\partial y_i}{\partial \hat{x_i}} = \gamma_i$
- $\frac{\partial \mu}{\partial x_j} = \frac{1}{D}$

So, the equation for $\frac{\partial f}{\partial x_j}$ in case of LayerNorm looks as follows:

$$\frac{\partial f}{\partial x_j} = (v+\epsilon)^{-\frac{1}{2}}\left(-\frac{1}{D}\sum_{i=1}^{D}\gamma_i\frac{\partial f}{\partial y_i} - (x_j - \mu)(v+\epsilon)^{-1}\frac{1}{D}\sum_{i=1}^{D}\gamma_i(x_i - \mu)\frac{\partial f}{\partial y_i} + \gamma_j\frac{\partial f}{\partial y_j}\right)$$

$\frac{\partial f}{\partial \gamma}$, $\frac{\partial f}{\partial \beta}$ are not changed.

Other normalization methods

Now we know two normalization methods for 2-D case: BatchNorm and LayerNorm. But there are some details in case of images, which have a spatial structure. Consider a batch of images of shape $\mathbb{R}^{N\times C\times H\times W}$, where N - batch size, C - number of channels (or feature maps), H - height, W - width.

We only need equations $\frac{\partial f}{\partial \gamma}$, $\frac{\partial f}{\partial \beta}$ for BatchNorm and $\frac{\partial f}{\partial x_j}$ for LayerNorm to get all expressions for all these kinds of normalisations. Differences will be in input tensors dimentions, axes to accumulate $\frac{\partial f}{\partial \gamma}$, $\frac{\partial f}{\partial \beta}$, axes to accumulate intermediate values for $\frac{\partial f}{\partial x_j}$. Table below summarises these changes:

name	per	over	norm
BatchNorm	D	N	D
LayerNorm	N	D	D
Spatial BatchNorm	C	N, H, W	C
GroupNorm	N, G	C / G, H, W	C
InstanceNorm	N, C	H, W	C

- *per* - independent input elements
- *over* - axes to group for computing μ and v
- *norm* - scale and shift axis

For example, derivatives of Spatial BatchNorm are as follows:

$$\frac{\partial f}{\partial \gamma_j} = \sum_{\substack{1 \le i \le N \\ 1 \le k \le H \\ 1 \le m \le W}} \frac{\partial f}{\partial Y_{ijkm}} \hat{X}_{ijkm}$$

$$\frac{\partial f}{\partial \beta_j} = \sum_{\substack{1 \le i \le N \\ 1 \le k \le H \\ 1 \le m \le W}} \frac{\partial f}{\partial Y_{ijkm}}$$

$$\frac{\partial f}{\partial X_{ijkm}} = \gamma_j (v_j + \epsilon)^{-\frac{1}{2}} \Bigg(-\frac{1}{NHW} \sum_{\substack{1 \le i' \le N \\ 1 \le k' \le H \\ 1 \le m' \le W}} \frac{\partial f}{\partial Y_{i'jk'm'}} -$$

$$(X_{ijkm} - \mu_j)(v_j + \epsilon)^{-1} \frac{1}{NHW} \sum_{\substack{1 \le i \le N \\ 1 \le k \le H \\ 1 \le m \le W}} (X_{i'jk'm'} - \mu_j) \frac{\partial f}{\partial Y_{i'jk'm'}} + \frac{\partial f}{\partial Y_{ijkm}} \Bigg)$$

Some notes about each method:

- BatchNorm – normalizes each feature independently. Quality of moments depends on batch size - higher is better.
- LayerNorm – computes statistics accross whole features. This fact makes it more preferable in case of small batches. Assumes equal contribution of each feature.
- Spatial BatchNorm – normalizes each feature map independently. Makes statistics consistent accross different images and image regions.
- GroupNorm – LayerNorm analogue for images, where aggregation is also done per channel groups: hypothesis is that feature maps are grouped by some factors like frequency, shapes, illumination, textures (examples from original paper). If so, each group might have different moments. Parametrized by G - number of groups of feature maps.
- InstanceNorm – special case of GroupNorm where $G = C$.