Review of Batch Normalization

Guannan Hu

April 13, 2018

Batch Normalization allows us to use much higher learning rates and be less careful about initialization, and it also acts as a regularizer, in some cases eliminating the need for Dropout.

Algorithm 1 Batch Normalizing Transform, applied to activation x over a mini-batch

---

Input: Values of x over a mini-batch: $\mathcal{B} = x_{1\ldots m}$; Paramters to be learned: γ, β

Output: $y_i = \mathrm{BN}_{\gamma,\beta}(x_i)$

$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum\limits_{i=1}^{m} x_i$ ▷ mini-batch mean

$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum\limits_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$ ▷ mini-batch variance

$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$ ▷ normalize

$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$ ▷ Scale and shift

---

During training we beed to backpropagate the gradient of loss l through this transformation, as well as compute the gradients with respect to the paramters of the BN transform. We use chain rule, as follows:

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

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

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

$\frac{\partial l}{\partial x_i} = \frac{\partial l}{\partial \hat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}} + \epsilon}} + \frac{\partial l}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial l}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$

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

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

Algorithm 2 Training a Batch-Normalized Network

Input: Network N with trainable paramters Θ;

Output: Batch-normalized network for ingerence, $N_{\mathrm{BN}}^{\mathrm{inf}}$

$N_{\mathrm{BN}}^{\mathrm{tr}} \leftarrow N$ ▷ Training BN network

for $k = 1 \ldots K$ **do**

  Add transformation $y(k) = \mathrm{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\mathrm{BN}}^{\mathrm{tr}}$ (Alg.1)

  Modify each layer in $N_{\mathrm{BN}}^{\mathrm{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead

end for

Train $N_{\mathrm{BN}}^{\mathrm{tr}}$ to optimize the parameters $\Theta \bigcup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$

$N_{\mathrm{BN}}^{\mathrm{inf}} \leftarrow N_{\mathrm{BN}}^{\mathrm{tr}}$ ▷ Inference BN network with frozen paramters

for $k = 1 \ldots K$ **do**

  //For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.

  Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\begin{aligned} \mathrm{E}[x] &\leftarrow \mathrm{E}_{\mathcal{B}}[\mu_{\mathcal{B}}] \\ \mathrm{Var}[x] &\leftarrow \frac{m}{m-1}\mathrm{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2] \end{aligned}$$

  In $N_{\mathrm{BN}}^{\mathrm{inf}}$, replace the transform $y = \mathrm{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\mathrm{Var}[x]+\epsilon}} \cdot x + (\beta - \frac{\gamma \mathrm{E}(x)}{\sqrt{\mathrm{Var}[x]+\epsilon}})$

end for