梯度累计场景下 BN 的替代策略

朱梦

初稿于 2025-06-26，修改于 2025-06-28

在博客《优化算法的分析及改进（七）：基于梯度累积的优化算法》、《优化算法的分析及改进（八）：隐藏在动量/一阶矩/二阶矩中的梯度累积》中：提出梯度累积算法来缓解由于有限显存而导致的较小批处理大小。但是，如果模型中包含了 Batch Normalization（BN）层，那么想要准确达到较大批处理大小的效果，唯一的方法就是加显存/显卡，因为 BN 算法在梯度下降的时候必须使用整个批处理数据来计算均值和方差。

既然 BN 层无法适配梯度累计算法，那么替换掉 BN 层，原模型中最佳替代算法无疑为 Group Normalization（GN）算法。可是，任何事物都有双面性。相比于 BN 算法，GN 算法额外引入了一个超参数，即分组数。

1. BN 算法

算法 1　BN 算法

输入数据: 批处理数据 $\boldsymbol{X} \in \mathbb{R}^{d\times s\times b}$。
输入权重: 缩放向量 $\boldsymbol{\gamma} \in \mathbb{R}^{d}$，平移向量 $\boldsymbol{\beta} \in \mathbb{R}^{d}$，均值滑动平均统计量 $\bar{\boldsymbol{\mu}} \in \mathbb{R}^{d}$，方差滑动平均统计量 $\bar{\mathbf{var}} \in \mathbb{R}^{d}$。
输入超参数: 防除零极小量 $\epsilon = 1e-7$，$\beta = 0.9$。
输出: $\boldsymbol{Y} \in \mathbb{R}^{d\times s\times b}$。

如果 训练模式 **则**
　计算均值和方差：$\mu_i = \frac{1}{sb}\sum_{j=1}^{s}\sum_{n=1}^{b} x_{i,j,n}$，$\sigma_i^2 = \frac{1}{sb}\sum_{j=1}^{s}\sum_{n=1}^{b}(x_{i,j,n}-\mu_i)^2$;
　均值滑动平均统计量：$\bar{\mu}_i \leftarrow \beta\bar{\mu}_i + (1-\beta)\mu_i$;
　方差滑动平均统计量：$\bar{\text{var}}_i \leftarrow \beta\bar{\text{var}}_i + (1-\beta)\sigma_i^2$;
否则
　$\mu_i = \bar{\mu}_i$，$\sigma_i^2 = \bar{\text{var}}_i$;
标准化：$\widehat{x}_{i,j,n} = \frac{x_{i,j,n}-\mu_i}{\sqrt{\sigma_i^2+\epsilon}}$;
变换重构：$y_{i,j,n} = \widehat{x}_{i,j,n}\gamma_i + \beta_i$;

为了研究滑动平均统计量和迭代次数之间的关系，加入下标 t 并迭代均值

滑动平均统计量，有：

$$\bar{\mu}_{i,t} = \beta^t \bar{\mu}_{i,0} + (1-\beta)\sum_{m=1}^{t}\beta^{t-m}\mu_{i,m} = (1-\beta)\sum_{m=1}^{t}\beta^{t-m}\mu_{i,m} \approx \mathbb{E}_t[\mu_{i,t}] \quad (1)$$

推荐 β 满足：

$$\beta \geq 1 - \frac{1}{T} \quad (2)$$

其中,T 表示每个 epoch 中的迭代次数。PyTorch 用户注意：nn.BatchNorm2d(momentum=0.1) 中的 0.1 对应 $1-\beta$（即 $\beta = 0.9$）。

2. GN 算法

算法 2 GN 算法

输入数据: 批处理数据 $\boldsymbol{X} \in \mathbb{R}^{b\times d\times s}$。
输入权重: 缩放向量 $\boldsymbol{\gamma} \in \mathbb{R}^d$，平移向量 $\boldsymbol{\beta} \in \mathbb{R}^d$。
输入超参数: 防除零极小量 $\epsilon = 1e-7$，分组数量 g。
输出: $\boldsymbol{Y} \in \mathbb{R}^{b\times d\times s}$。

形状变换：$\boldsymbol{X} \in \mathbb{R}^{b\times d\times s}$ 变换为 $\boldsymbol{Z} \in \mathbb{R}^{b\times g\times (d/g)\times s}$;

计算均值和方差：$\mu_{i,j} = \frac{1}{(d/g)\times s}\sum_{k=1}^{d/g}\sum_{n=1}^{s} z_{i,j,k,n}$，
$\sigma_{i,j}^2 = \frac{1}{(d/g)\times s}\sum_{k=1}^{d/g}\sum_{n=1}^{s}(z_{i,j,k,n}-\mu_{i,j})^2$;

标准化：$\widehat{z}_{i,j,k,n} = \frac{z_{i,j,k,n}-\mu_{i,j}}{\sqrt{\sigma_{i,j}^2+\epsilon}}$;

形状逆变换：$\widehat{\boldsymbol{Z}} \in \mathbb{R}^{b\times g\times (d/g)\times s}$ 变换为 $\widehat{\boldsymbol{X}} \in \mathbb{R}^{b\times d\times s}$;

变换重构：$y_{i,k,n} = x_{i,k,n}\gamma_k + \beta_k$。

若分组不足，即 g 过小，例如：当 $g=1$ 时，样本内特征的相对关系被破坏，计算量增加，从而导致过归一化。若分组过多，即 g 过大，例如：当 $g=d$ 时，GN 退化为 Instance Normalization，从而导致归一化不足。

3. 结论及其反思

在梯度累积/小批量训练/动态批处理大小等场景下，GN 算法都是 BN 算法的最优替代。可是，相比于 BN 算法，GN 算法额外引入了一个超参数，即分组数。因此，需要精细调节分组数。