Extremely Low Bit Neural Network: Squeeze the Last Bit Out with ADMM

2018-11-14

Overview

- AAAI 2018
- N-bit量化weight，scale factor为2^N（移位运算）
- ADMM优化方法

ADMM

分三步优化：

the method of multipliers. The algorithm solves problems in the form:

$$\begin{aligned} \min \quad & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{s.t.} \quad & A\mathbf{x} + B\mathbf{z} = \mathbf{c} \end{aligned} \tag{1}$$

with variables $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathbb{R}^m$, where $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$ and $\mathbf{c} \in \mathbb{R}^p$.

The augmented Lagrangian of Eq.(1) can be formed as:

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(A\mathbf{x} + B\mathbf{z} - \mathbf{c}) + (\rho/2)\|A\mathbf{x} + B\mathbf{z} - \mathbf{c}\|_2^2 \tag{2}$$

增广拉格朗日形式

where $\mathbf{y}$ is the Lagrangian multipliers, and ADMM consists of three step iterations:

$$\begin{aligned} \mathbf{x}^{k+1} &:= \arg\min_{\mathbf{x}} L_\rho(\mathbf{x}, \mathbf{z}^k, \mathbf{y}^k) \\ \mathbf{z}^{k+1} &:= \arg\min_{\mathbf{z}} L_\rho(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{y}^k) \\ \mathbf{y}^{k+1} &:= \mathbf{y}^k + \rho(A\mathbf{x}^{k+1} + B\mathbf{z}^{k+1} - \mathbf{c}) \end{aligned}$$

交替优化
（y为拉格朗日乘子）

The Proposed Method

- Objective function

$$\min_{W} \; f(W) \qquad \text{s.t.} \;\; W \in \mathcal{C} = \{-1, 0, +1\}^d$$

将离散值权重的神经网络训练定义成一个离散约束优化问题

Since the weights are restricted to be zero or powers of two, we have constraints of this form

$$\mathcal{C} = \{-2^N, \cdots, -2^1, -2^0, 0, +2^0, +2^1, \cdots, +2^N\}$$

- 加入scale factor之后：

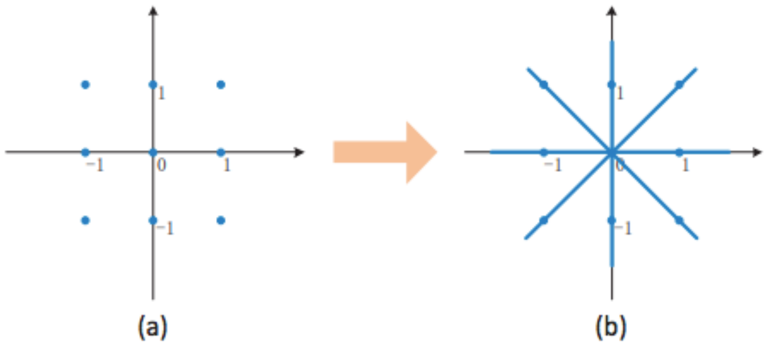

Figure 1: In ternary neural network, scaling factor expands the constrained space from (a) nice discrete points to (b) four lines in the space (two dimensional space as an example).

The Proposed Method

$$\min_{W} \quad f(W) + I_{\mathcal{C}}(W) \tag{4}$$

where $I_{\mathcal{C}}(W) = 0$ if $W \in \mathcal{C}$, otherwise $I_{\mathcal{C}}(W) = +\infty$.

Ic(W)：指示函数

$$\begin{aligned} \min_{W,G} \quad & f(W) + I_{\mathcal{C}}(G) \\ \text{s.t.} \quad & W = G \end{aligned} \tag{5}$$

The Proposed Method

lems of such form can be conveniently solved with ADMM. The augmented Lagrange of Eq.(5), for parameter $\rho > 0$, can be formulated as:

$$L_\rho(W, G, \mu) = f(W) + I_C(G) + \frac{\rho}{2}\|W - G\|^2 + \langle \mu, W - G \rangle \tag{6}$$

where μ denotes the Lagrangian multipliers and $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors. With some basic collection of terms and a change of variable $\lambda = (1/\rho)\mu$, Eq.(6) can be equivalently formed as:

$$L_\rho(W, G, \lambda) = f(W) + I_C(G) + \frac{\rho}{2}\|W - G + \lambda\|^2 - \frac{\rho}{2}\|\lambda\|^2 \tag{7}$$

转化为可以用ADMM优化求解的增广拉格朗日方程

the method of multipliers. The algorithm solves problems in the form:

$$\begin{aligned} \min \quad & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{s.t.} \quad & A\mathbf{x} + B\mathbf{z} = \mathbf{c} \end{aligned} \tag{1}$$

with variables $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathbb{R}^m$, where $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$ and $\mathbf{c} \in \mathbb{R}^p$.

The augmented Lagrangian of Eq.(1) can be formed as:

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(A\mathbf{x} + B\mathbf{z} - \mathbf{c}) + (\rho/2)\|A\mathbf{x} + B\mathbf{z} - \mathbf{c}\|_2^2 \tag{2}$$

where $\mathbf{y}$ is the Lagrangian multipliers, and ADMM consists of three step iterations:

$$\begin{aligned} \mathbf{x}^{k+1} &:= \arg\min_{\mathbf{x}} L_\rho(\mathbf{x}, \mathbf{z}^k, \mathbf{y}^k) \\ \mathbf{z}^{k+1} &:= \arg\min_{\mathbf{z}} L_\rho(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{y}^k) \\ \mathbf{y}^{k+1} &:= \mathbf{y}^k + \rho(A\mathbf{x}^{k+1} + B\mathbf{z}^{k+1} - \mathbf{c}) \end{aligned}$$

The Proposed Method

lems of such form can be conveniently solved with ADMM. The augmented Lagrange of Eq.(5), for parameter $\rho > 0$, can be formulated as:

$$L_\rho(W, G, \mu) = f(W) + I_C(G) + \frac{\rho}{2}\|W - G\|^2 + \langle \mu, W - G \rangle \tag{6}$$

where μ denotes the Lagrangian multipliers and $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors. With some basic collection of terms and a change of variable $\lambda = (1/\rho)\mu$, Eq.(6) can be equivalently formed as:

$$L_\rho(W, G, \lambda) = f(W) + I_C(G) + \frac{\rho}{2}\|W - G + \lambda\|^2 - \frac{\rho}{2}\|\lambda\|^2 \tag{7}$$

转化为可以用ADMM优化求解的增广拉格朗日方程

$$W^{k+1} := \arg\min_W L_\rho(W, G^k, \lambda^k) \tag{8}$$

$$G^{k+1} := \arg\min_G L_\rho(W^{k+1}, G, \lambda^k) \tag{9}$$

$$\lambda^{k+1} := \lambda^k + W^{k+1} - G^{k+1} \tag{10}$$

The Proposed Method

- 与其他算法不同，在实数空间和离散空间分别求解，然后通过拉格朗日乘子的更新将两组解联系起来

$$
\begin{aligned}
W^{k+1} &:= \arg\min_{W} L_\rho(W, G^k, \lambda^k) && (8) \\
G^{k+1} &:= \arg\min_{G} L_\rho(W^{k+1}, G, \lambda^k) && (9) \\
\lambda^{k+1} &:= \lambda^k + W^{k+1} - G^{k+1} && (10)
\end{aligned}
$$

The Proposed Method

$$W^{k+1} := \arg\min_{W} L_\rho(W, G^k, \lambda^k) \tag{8}$$

$$G^{k+1} := \arg\min_{G} L_\rho(W^{k+1}, G, \lambda^k) \tag{9}$$

$$\lambda^{k+1} := \lambda^k + W^{k+1} - G^{k+1} \tag{10}$$

$$L_\rho(W, G^k, \lambda^k) = f(W) + \frac{\rho}{2}\|W - G^k + \lambda^k\|^2$$

$$\partial_W L = \partial_W f + \rho(W - G^k + \lambda^k)$$

由于普通的梯度更新收敛过于缓慢，所以采用以下方法更新

$$W^{(p)} := W - \beta_p \partial_W L(W),$$
$$W^{(c)} := W - \beta_c \partial_W L(W^{(p)})$$

beta为学习率

The Proposed Method

$$W^{k+1} := \arg\min_{W} L_\rho(W, G^k, \lambda^k) \tag{8}$$

$$G^{k+1} := \arg\min_{G} L_\rho(W^{k+1}, G, \lambda^k) \tag{9}$$

$$\lambda^{k+1} := \lambda^k + W^{k+1} - G^{k+1} \tag{10}$$

$$\begin{aligned} \min_{G_i, \alpha_i} \quad & \|V_i - G_i\|^2 \\ \text{s.t.} \quad & G_i \in \{0, \pm\alpha_i, \pm 2\alpha_i, \cdots, \pm 2^N \alpha_i\}^{d_i} \end{aligned} \tag{12}$$

Taking the scaling factor away from the constraints, the objective can be equivalently formulated as:

$$\begin{aligned} \min_{Q_i, \alpha_i} \quad & \|V_i - \alpha_i \cdot Q_i\|^2 \\ \text{s.t.} \quad & Q_i \in \{0, \pm 1, \pm 2, \cdots, \pm 2^N\}^{d_i} \end{aligned} \tag{13}$$

迭代优化求解

with Q_i fixed, the problem becomes an univariate optimization. The optimal α_i can be easily obtained as

$$\alpha_i = \frac{V_i^T Q_i}{Q_i^T Q_i} \tag{14}$$

With α_i fixed, the optimal Q_i is actually the projection of $\frac{V_i}{\alpha_i}$ onto $\{0, \pm 1, \pm 2, \cdots, \pm 2^N\}$, namely,

$$Q_i = \Pi_{\{0, \pm 1, \pm 2, \cdots, \pm 2^N\}} \left(\frac{V_i}{\alpha_i} \right) \tag{15}$$

Experiment

	Accuracy	Binary	BWN	Ternary	TWN	{-2, +2}	{-4, +4}	Full Precision
AlexNet	Top-1	**0.570**	0.568	**0.582**	0.575	0.592	0.600	0.600
	Top-5	**0.797**	0.794	**0.806**	0.798	0.818	0.822	0.824
VGG-16	Top-1	**0.689**	0.678	**0.700**	0.691	0.717	0.722	0.711
	Top-5	**0.887**	0.881	**0.896**	0.890	0.907	0.909	0.899

	Accuracy	Binary	BWN	Ternary	TWN	{-2, +2}	{-4, +4}	Full Precision
Resnet-18	Top-1	**0.648**	0.608	**0.670**	0.618	0.675	0.680	0.691
	Top-5	**0.862**	0.830	**0.875**	0.842	0.879	0.883	0.890
Resnet-50	Top-1	**0.687**	0.639	**0.725**	0.656	0.739	0.740	0.753
	Top-5	**0.886**	0.851	**0.907**	0.865	0.915	0.916	0.922

	Accuracy	Binary	BWN	Ternary	TWN	{-2, +2}	{-4, +4}	Full Precision
GoogLeNet	Top-1	**0.603**	0.590	**0.631**	0.612	0.659	0.663	0.687
	Top-5	**0.832**	0.824	**0.854**	0.841	0.873	0.875	0.889

{-2, -1, 0, 1, 2 }

{-4, -2, -1, 0, 1, 2, 4 }