Blended Coarse Gradient Descent for Full Quantization of Deep Neural Networks

Jack Xin

Department of Mathematics
University of California, Irvine.

Collaborators and Acknowledgements

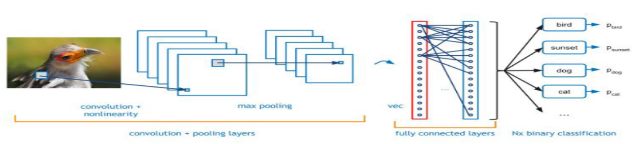
- Jiancheng Lyu (UC Irvine).
- Penghang Yin, Stan Osher (UCLA).
- Shuai Zhang, Yingyong Qi (Qualcomm AI Research, San Diego).
- Partially supported by NSF Big Data Program.

Outline

- Deep Neural Networks (DNN) and Quantization Problem.
- Why Coarse Derivative ?
- Blended Coarse Gradient Descent and Properties.
- Numerical Experiments.
- Analysis of Coarse Gradient Descent.
- Conclusion and Future Work.

Deep Learning

- DNNs drive the recent AI advances suppassing human performance (image/speech recognition, Alpha-Go) and big data research across all scientific disciplines.



$$\mathbf{O} := \mathbf{w}_{L+1} * \sigma(\mathbf{w}_L * \cdots \sigma(\mathbf{w}_1 * \mathbf{I}) \cdots), \ \ \sigma = \max(\cdot, 0), \text{ activation.}$$

- Require hundreds of megabytes of memory to store full-precision floating-point weights ($\mathbf{w}_1, \mathbf{w}_2, \cdots$), and billions of FLOPs (floating point operations per second) on a single forward pass ($\mathbf{I}$ to $\mathbf{O}$).

Quantization

- A challenge to run DNN on mobile devices or other platforms with limited resources.
- An effective complexity reduction method is quantization:

  Restrict weight and activation values to discrete and finite subsets.
- Network retraining is needed to maintain the same level of accuracy.
- Resolving conflict:

  discreteness of quantization

  vs.

  continuous nature of stochastic gradient descent (SGD).

Weight Quantization

- Entries of weight matrix $\mathbf{w}_l$ of dimension $N(l)$ in each layer l are constrained to values from the set:

$$\mathbf{Q} := \mathbb{R}_+ \times \{\pm q_1, \pm, q_2, \cdots, \pm q_m\}^{N(l)}$$

  a float precision scaling factor times signed quantized values:

$$0 \le q_1 < q_2 < \cdots < q_m.$$

- 1-bit (binarization): $m = 1$, $q_1 = 1$.
- 2-bit (ternarization): $m = 2$, $q_1 = 0$, $q_2 = 1$.
- 4-bit linear quantization: $m = 8$, $q_j = \frac{j}{8}$, $j = 0, 1, \cdots, 8$.

Projection: solving least squares problem

- Given matrix W, find:

$$\mathrm{proj}_{\mathbf{Q}}(W) = \mathrm{argmin}_{z \in \mathbf{Q}} \|z - W\|^2 = s_+ \cdot q_*$$

$$(s_+, q_*) = \mathrm{argmin}_{s \geq 0, s \cdot q \in \mathbf{Q}} \|s \cdot q - W\|^2.$$

- Binarization (Rastegari, et al, 2016; complexity $O(N)$):

$$s_+ = \|W\|_1 / \mathrm{dim}(W),\ q_* = \mathrm{sign}(W),\ \mathrm{sign}(0) := 1.$$

- Ternarization (Yin, Zhang, Qi, X, 2016; complexity $O(N \log N)$):

$$s_+ = \|W_{[t^*]}\|_1 / t^*,\ q_* = \mathrm{sign}(W_{[t^*]}),\ \mathrm{sign}(0) = 0,$$

$$t^* := \mathrm{argmax}_{1 \leq t \leq \mathrm{dim}(W)} \|W_{[t]}\|_1^2 / t,$$

$W_{[t]}$: keep t largest entries in magnitude, zero out the rest.

Projection: solving least squares problem

- At bit-width $b_w \geq 3$, exact solutions are too expensive computationally.
- Iterative solutions by Lloyd algorithm: alternating between s-update and q-update.
- s-update:

$$\frac{(\mathbf{q}^{(i)})^\top W}{\|\mathbf{q}^{(i)}\|^2} = \mathrm{argmin}_{s\in\mathbb{R}} \|s\,\mathbf{q}^{(i)} - W\|^2.$$

- $\mathbf{q}$-update: minimize component by component.
- Practically, one step Lloyd: initialize $s = \frac{2}{2^{b_w}-1}\|W\|_\infty$; find $\mathbf{q} \in \mathbf{Q}$ componentwise to the least squares problem.
- Errors in quantization can be corrected during network retraining.

Activation Quantization (AQ)

- Uniform AQ:

$$\sigma(x,\alpha) = \begin{cases} 0, & \text{if} \quad x \le 0, \\ k\alpha, & \text{if} \quad (k-1)\,\alpha < x \le k\alpha,\ k = 1 : 2^{b_a} - 1, \\ \left(2^{b_a} - 1\right)\alpha, & \text{if} \quad x > \left(2^{b_a} - 1\right)\alpha, \end{cases}$$

  x the scalar input, $\alpha > 0$ the resolution, $b_a \in \mathbb{Z}_+$ the bit-width of activation, k the quantization level.

- 4-bit (4A): $b_a = 4$ and $k = 1, 2, \cdots, 15$.
- Sample loss function for training input $\mathbf{Z}$ and label u:

  $$\ell(\mathbf{w}, \alpha; \{\mathbf{Z}, u\}) := \ell(\mathbf{w}_{L+1} * \sigma(\mathbf{w}_L * \cdots * \mathbf{w}_2 * \sigma(\mathbf{w}_1 * \mathbf{Z}, \alpha_1) \cdots, \alpha_L); u)$$

  $\mathbf{w}_j$: weights in j-th linear (fully-connected or convolutional) layer. * $=$ matrix-vector product or convolution. The j-th quantized ReLU $\sigma(\mathbf{x}_j, \alpha_j)$ acts element-wise on output $\mathbf{x}_j$ from previous linear layer, with a trainable scalar $\alpha_j > 0$.

Minimize Piecewise Constant Functions in High Dim

- Given N training samples, minimize empirical risk with quantized ReLU:

$$\min_{\mathbf{w},\boldsymbol{\alpha}} f(\mathbf{w},\boldsymbol{\alpha}) := \frac{1}{N}\sum_{i=1}^{N} \ell(\mathbf{w},\alpha;\{\mathbf{Z}^{(i)},u^{(i)}\})$$

- Gradient calculated by chain rule involves: $\frac{\partial\sigma}{\partial x}$ ($= 0$ a.e.) and $\frac{\partial\sigma}{\partial\alpha}$.

$$\frac{\partial\ell}{\partial\mathbf{w}_L} = \sigma(\mathbf{x}_{L-1},\alpha_{L-1}) \circ \frac{\partial\sigma}{\partial x}(\mathbf{x}_L,\alpha_L) \circ \mathbf{w}_{L+1}^\top \circ \nabla\ell(\mathbf{x}_{L+1};u)$$

$$\frac{\partial\ell}{\partial\alpha_{L-1}} = \frac{\partial\sigma}{\partial\alpha}(\mathbf{x}_{L-1},\alpha_{L-1}) \circ \mathbf{w}_L^\top \circ \frac{\partial\sigma}{\partial x}(\mathbf{x}_L,\alpha_L) \circ \mathbf{w}_{L+1}^\top \circ \nabla\ell(\mathbf{x}_{L+1};u)$$

$\mathbf{x}_j := \mathbf{w}_j * \boldsymbol{\sigma}(\mathbf{x}_{j-1},\alpha_{j-1})$.

- Zero gradients a.e. of ℓ in $\{\mathbf{w}_j\}_{j=1}^{L}$ and $\{\alpha_j\}_{j=1}^{L-1}$.
- Zero gradients by auto-diff on Pytorch, causing SGD to stagnate.

Walking Down a Hill of Terraces ?

Differentiate a Staircase Function over Large Scale

- "Large scale" derivative of quantized σ (a staircase):

$$\frac{\partial \sigma}{\partial x}(x, \alpha) \approx \begin{cases} 0, & \text{if} \quad x \leq 0, \\ 1, & \text{if} \quad 0 < x \leq \left(2^{b_a} - 1\right)\alpha, \\ 0, & \text{if} \quad x > \left(2^{b_a} - 1\right)\alpha \end{cases}$$

  a non-zero value in the middle to reflect the overall variation of σ. Or the derivative of the step-back view of σ in x.

- Same as the derivative of the clipped ReLU (a ramp):

$$\tilde{\sigma}(x, \alpha) = \begin{cases} 0, & \text{if} \quad x \leq 0, \\ x, & \text{if} \quad 0 < x \leq (2^{b_a} - 1)\alpha, \\ \left(2^{b_a} - 1\right)\alpha, & \text{if} \quad x > \left(2^{b_a} - 1\right)\alpha. \end{cases}$$

When a.e. partial derivative exists

- but not the best in either classification accuracy or computational cost:

$$\frac{\partial \sigma}{\partial \alpha}(x,\alpha) = \begin{cases} 0, & \text{if} \quad x \le 0, \\ k, & \text{if} \quad (k-1)\,\alpha < x \le k\alpha, \quad k = 1 : 2^{b_a} - 1; \\ 2^{b_a} - 1, & \text{if} \quad x > \left(2^{b_a} - 1\right)\alpha. \end{cases}$$

- 3-valued "coarse" partial derivative in α:

$$\frac{\partial \sigma}{\partial \alpha}(x,\alpha) \approx \begin{cases} 0, & \text{if} \quad x \le 0, \\ 2^{b_a-1}, & \text{if} \quad 0 < x \le \left(2^{b_a} - 1\right)\alpha, \\ 2^{b_a} - 1, & \text{if} \quad x > \left(2^{b_a} - 1\right)\alpha. \end{cases}$$

The middle value 2^{b_a-1} is the arithmetic mean of the intermediate k values of the a.e. partial derivative above.

Coarse Gradients

- 2-valued coarse partial derivative proposed in PACT '18 (using straight-through estimator of Hinton '12, Bengio et al, '13), or simply zero out all nonzero values except their maximum in a.e. $\frac{\partial \sigma}{\partial \alpha}(x, \alpha)$

$$\frac{\partial \sigma}{\partial \alpha}(x, \alpha) \approx \begin{cases} 0, & \text{if} \quad x \leq \left(2^{b_a} - 1\right)\alpha, \\ 2^{b_a} - 1, & \text{if} \quad x > \left(2^{b_a} - 1\right)\alpha, \end{cases}$$

- exactly $\frac{\partial \text{ clipped ReLU}}{\partial \alpha}(x, \alpha)$.

- Coarse gradients in action: substitute

  1) coarse partials for α partial derivative,

  2) clipped ReLU in **x** partial derivative

  of the quantized σ in the chain rule expressions of gradients.

Full Quantization Problem

- Layer-wise weight and activation quantization problem is:

$$\min_{\mathbf{w},\boldsymbol{\alpha}} f(\mathbf{w},\boldsymbol{\alpha}) \text{ subject to } \mathbf{w} \in \mathbf{Q} = \mathbf{Q}_1 \times \mathbf{Q}_2 \cdots \times \mathbf{Q}_{L+1},$$

weight in j-th linear layer is constrained as $\mathbf{w}_j = \delta_j \mathbf{q}_j \in \mathbf{Q}_j$ for some adjustable scalar $\delta_j > 0$. Each entry of $\mathbf{q}_j$ is optimally drawn from the quantization set given by $\{\pm \frac{k}{2^{b_w-1}} : k = 0, 1, \cdots, 2^{b_w-1} - 1\}$ for $b_w \geq 2$ and $\{\pm 1\}$ for $b_w = 1$. Here $b_w \in \mathbb{Z}_+$ is the bit-width for weight quantization, δ_j a floating (32-bit) real number.

- BinaryConnect weight update (Courbariaux et al, '15):

$$\mathbf{w}_f^{t+1} = \mathbf{w}_f^t - \eta \nabla f(\mathbf{w}^t),\ \mathbf{w}^{t+1} = \mathrm{proj}_{\mathbf{Q}}(\mathbf{w}_f^{t+1}),$$

where $\{\mathbf{w}^t\}$ is the sequence of quantized weights, $\{\mathbf{w}_f^t\}$ is an auxiliary sequence of floating weights.

Blended Gradient Descent

- Blend BinaryConnect and classical projected GD (for smooth constraint):

$$PGD: \ \mathbf{w}_f^{t+1} = \mathbf{w}^t - \eta \nabla f(\mathbf{w}^t), \ \mathbf{w}^{t+1} = \mathrm{proj}_{\mathbf{Q}}(\mathbf{w}_f^{t+1})$$

$$BGD: \ \mathbf{w}_f^{t+1} = (1-\rho)\,\mathbf{w}_f^t + \rho\,\mathbf{w}^t - \eta \nabla f(\mathbf{w}^t), \ \mathbf{w}^{t+1} = \mathrm{proj}_{\mathbf{Q}}(\mathbf{w}_f^{t+1})$$

  with parameter $\rho \ll 1$.

- If objective function f has L-Lipschitz gradient, then for $\rho \in (0,1)$, at small enough learning rate $\eta > 0$, BGD satisfies the sufficient descent property (SDP):

$$f(\mathbf{w}^{t+1}) - f(\mathbf{w}^t) \leq -c\,\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2,$$

  for some positive constant $c > 0$, while BinaryConnect does not.

Blended Coarse Gradient Descent

- In fully quantized network training ($\tilde{}$ = coarse):

$$\begin{aligned}\boldsymbol{\alpha}^{t+1} &= \boldsymbol{\alpha}^t - \eta_\alpha\, \tilde{\nabla}_{\boldsymbol{\alpha}}\, f(\mathbf{w}^t, \boldsymbol{\alpha}^t),\\ \mathbf{w}_f^{t+1} &= (1-\rho)\,\mathbf{w}_f^t + \rho\,\mathbf{w}^t - \eta_w\, \tilde{\nabla}_{\mathbf{w}}\, f(\mathbf{w}^t, \boldsymbol{\alpha}^t),\\ \mathbf{w}^{t+1} &= \mathrm{proj}_{\mathbf{Q}}(\mathbf{w}_f^{t+1})\end{aligned}$$

- Two scale learning: $\eta_\alpha = 0.01\,\eta_w$.
  $\boldsymbol{\alpha}$-learning much slower than $\mathbf{w}$-learning.
- Blending parameter: $\rho = 10^{-5}$.
- Initialization: $\alpha^1 = 1/(2^{b_a} - 1)$, $\eta_\alpha = 10^{-4}$.
  Decay factor of learning rates: 0.1.
- Image Datasets: CIFAR-10, ImageNet.
- PyTorch on 4 Nvidia GeForce GTX 1080 Ti GPUs.

Experiment: a) Blending improves accuracy especially at low precision. b) No need of a.e. derivative.

Network	Float	32W4A	1W4A	2W4A	4W4A
VGG-11 + BC	92.13	91.74	88.12	89.78	**91.51**
VGG-11+BCGD			**88.74**	**90.08**	91.38
ResNet-20 + BC	92.41	91.90	89.23	90.89	91.53
ResNet-20+BCGD			**90.10**	**91.15**	91.56

Table: CIFAR-10 validation accuracies in % with the a.e. α derivative.

Network	Float	32W4A	1W4A	2W4A	4W4A
VGG-11 + BC	92.13	92.08	89.12	90.52	**91.89**
VGG-11+BCGD			**89.59**	**90.71**	91.70
ResNet-20 + BC	92.41	92.14	89.37	91.02	91.71
ResNet-20+BCGD			**90.05**	91.03	**91.97**

Table: CIFAR-10 validation accuracies with the 3-valued α derivative.

Experiment: a) 3-valued better than 2-valued $d\sigma/d\alpha$. b) Network at (4W,4A) within 1 % of float network precision.

Network	Float	32W4A	1W4A	2W4A	4W4A
VGG-11 + BC	92.13	92.08	89.12	90.52	**91.89**
VGG-11+BCGD			**89.59**	**90.71**	91.70
ResNet-20 + BC	92.41	92.14	89.37	91.02	91.71
ResNet-20+BCGD			**90.05**	91.03	**91.97**

Table: CIFAR-10 validation accuracies with the 3-valued α derivative.

Network	Float	32W4A	1W4A	2W4A	4W4A
VGG-11 + BC	92.13	91.66	88.50	89.99	91.31
VGG-11+BCGD			**89.12**	90.00	91.31
ResNet-20 + BC	92.41	91.73	89.22	90.64	91.37
ResNet-20+BCGD			**89.98**	**90.75**	**91.65**

Table: CIFAR-10 validation accuracies with the 2-valued α derivative.

Experiment: Blending effects during training on ImageNet.

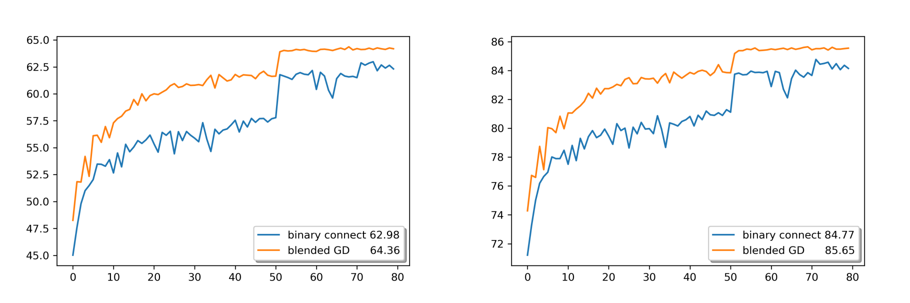


Figure: ImageNet validation accuracies vs. number of epochs using 3-valued α-derivative on 1W4A ResNet-18; with (orange) and without (blue) blending. Top-1: left. Top-5: right.

Experiment: all convolution layers quantized on ImageNet.

	Float	1W4A		4W4A		4W8A	
		3 val	2 val	3 val	2 val	3 val	2 val
top-1	69.64	**64.36**	63.37	**67.36**	66.97	**68.85**	68.83
top-5	88.98	**85.65**	84.93	**87.76**	87.41	88.71	**88.84**

Table: ImageNet validation accuracies (%) with BCGD on ResNet-18. The 3-valued α-derivatives improve more on 2-valued in low bit regime. Quantized 4W8A network accurate within 1% of float precision network.

- ImageNet: 1.2 million images for training and 50,000 for validation, 1,000 classes. Mini-batch size 256 and 80 epochs of training with learning rate decay at epoch #50 and #70.
- CIFAR-10: 60,000 small color images of 10 classes. Split into 50,000 training and 10,000 validation. Mini-batch size 128 and 200 epochs of training with learning rate decay at epoch #80 and #140.

Two-Layer Neural Network Regression Problem

- Two-layer neural network model with binarized ReLU activation:

$$\sigma(x) = \begin{cases} 0 & \text{if } x \leq 0, \\ 1 & \text{if } x > 0. \end{cases}$$

  Sample loss function:

$$\ell(\mathbf{v}, \mathbf{w}; \mathbf{Z}) := \frac{1}{2}\Big(\mathbf{v}^\top \sigma(\mathbf{Z}\mathbf{w}) - (\mathbf{v}^*)^\top \sigma(\mathbf{Z}\mathbf{w}^*)\Big)^2$$

  $\mathbf{v}^* \in \mathbb{R}^m$ and $\mathbf{w}^* \in \mathbb{R}^n$: prescribed nonzero 'teacher parameters' in 2nd and 1st layers. Gaussian input data: entries of $\mathbf{Z} \in \mathbb{R}^{m\times n}$, i.i.d. sampled from unit Gaussian $\mathcal{N}(0, 1)$.

- $\ell(\mathbf{v}, \mathbf{w}; \mathbf{Z}) = \ell(\mathbf{v}, \mathbf{w}/c; \mathbf{Z})$, $\forall\, c > 0$. WLOG: $\|\mathbf{w}^*\| = 1$.

Two-Layer Neural Network Regression Problem

- Empirical risk minimization:

$$\min_{\mathbf{v}\in\mathbb{R}^m,\mathbf{w}\in\mathbb{R}^n} \; \frac{1}{N}\sum_{i=1}^{N}\ell(\mathbf{v},\mathbf{w};\mathbf{Z}^{(i)})$$

piecewise constant objective.

- Population loss minimization:

$$\min_{\mathbf{v}\in\mathbb{R}^m,\mathbf{w}\in\mathbb{R}^n} \; f(\mathbf{v},\mathbf{w}) := \mathbf{E}_{\mathbf{Z}}\left[\ell(\mathbf{v},\mathbf{w};\mathbf{Z})\right]$$

smoother objective ($\theta(\cdot,\cdot)$ = angle between dots):

$$\begin{aligned} 8\,f(\mathbf{v},\mathbf{w}) &= \mathbf{v}^\top\left(\mathbf{I}+\mathbf{1}\mathbf{1}^\top\right)\mathbf{v} - 2\mathbf{v}^\top\left(\left(1-\frac{2}{\pi}\theta(\mathbf{w},\mathbf{w}^*)\right)\mathbf{I}+\mathbf{1}\mathbf{1}^\top\right)\mathbf{v}^* \\ &+ (\mathbf{v}^*)^\top\left(\mathbf{I}+\mathbf{1}\mathbf{1}^\top\right)\mathbf{v}^*. \end{aligned}$$

Two-Layer Neural Network Regression Problem

- Lipschitz continuous gradients:

$$\frac{\partial f}{\partial \mathbf{v}}(\mathbf{v},\mathbf{w}) = \frac{1}{4}\left(\mathbf{I} + \mathbf{1}\mathbf{1}^\top\right)\mathbf{v} - \frac{1}{4}\left(\left(1 - \frac{2}{\pi}\theta(\mathbf{w},\mathbf{w}^*)\right)\mathbf{I} + \mathbf{1}\mathbf{1}^\top\right)\mathbf{v}^*$$

$$\frac{\partial f}{\partial \mathbf{w}}(\mathbf{v},\mathbf{w}) = -\frac{\mathbf{v}^\top\mathbf{v}^*}{2\pi\|\mathbf{w}\|}\frac{\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^\top}{\|\mathbf{w}\|^2}\right)\mathbf{w}^*}{\left\|\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^\top}{\|\mathbf{w}\|^2}\right)\mathbf{w}^*\right\|}, \quad \forall\,\theta(\mathbf{w},\mathbf{w}^*) \in (0,\pi).$$

- Possible locations for non-trivial local minimizers are:
  1. Critical points where the gradients defined above vanish simultaneously (may not be possible in general)

$$\mathbf{v}^\top\mathbf{v}^* = 0 \text{ and } \mathbf{v} = \left(\mathbf{I} + \mathbf{1}\mathbf{1}^\top\right)^{-1}\left(\left(1 - \frac{2}{\pi}\theta(\mathbf{w},\mathbf{w}^*)\right)\mathbf{I} + \mathbf{1}\mathbf{1}^\top\right)\mathbf{v}^*.$$

  2. Non-differentiable points where $\theta(\mathbf{w},\mathbf{w}^*) = 0$ and $\mathbf{v} = \mathbf{v}^*$ (global minimizer), or $\theta(\mathbf{w},\mathbf{w}^*) = \pi$ and $\mathbf{v} = \left(\mathbf{I} + \mathbf{1}\mathbf{1}^\top\right)^{-1}(\mathbf{1}\mathbf{1}^\top - \mathbf{I})\mathbf{v}^*$.

Two-Layer Neural Network Regression Problem

- Accessible gradients in training are finite sample approximations of:

$$\mathbf{E_Z}\left[\frac{\partial \ell}{\partial \mathbf{v}}(\mathbf{v}, \mathbf{w}; \mathbf{Z})\right] \text{ and } \mathbf{E_Z}\left[\frac{\partial \ell}{\partial \mathbf{w}}(\mathbf{v}, \mathbf{w}; \mathbf{Z})\right].$$

- Formally by chain rule (gradient to **w** is a.e. 0):

$$\frac{\partial \ell}{\partial \mathbf{v}}(\mathbf{v}, \mathbf{w}; \mathbf{Z}) = \sigma(\mathbf{Z}\mathbf{w})\left(\mathbf{v}^\top \sigma(\mathbf{Z}\mathbf{w}) - (\mathbf{v}^*)^\top \sigma(\mathbf{Z}\mathbf{w}^*)\right).$$

$$\frac{\partial \ell}{\partial \mathbf{w}}(\mathbf{v}, \mathbf{w}; \mathbf{Z}) = \mathbf{Z}^\top \left(\sigma'(\mathbf{Z}\mathbf{w}) \odot \mathbf{v}\right)\left(\mathbf{v}^\top \sigma(\mathbf{Z}\mathbf{w}) - (\mathbf{v}^*)^\top \sigma(\mathbf{Z}\mathbf{w}^*)\right)$$

- Replace σ' by (sub)derivative μ' of regular ReLU function $\mu(x) := \max(x, 0)$, and define coarse gradient:

$$\mathbf{g}(\mathbf{v}, \mathbf{w}; \mathbf{Z}) := \mathbf{Z}^\top \left(\mu'(\mathbf{Z}\mathbf{w}) \odot \mathbf{v}\right)\left(\mathbf{v}^\top \sigma(\mathbf{Z}\mathbf{w}) - (\mathbf{v}^*)^\top \sigma(\mathbf{Z}\mathbf{w}^*)\right).$$

Two-Layer Neural Network Regression Problem

- Coarse gradient descent with weight normalization:

$$\begin{cases} \mathbf{v}^{t+1} = \mathbf{v}^t - \eta\, \mathbf{E}_\mathbf{Z}\left[\frac{\partial \ell}{\partial \mathbf{v}}(\mathbf{v}^t, \mathbf{w}^t; \mathbf{Z})\right] \\ \mathbf{w}^{t+\frac{1}{2}} = \mathbf{w}^t - \eta\, \mathbf{E}_\mathbf{Z}\left[\mathbf{g}(\mathbf{v}^t, \mathbf{w}^t; \mathbf{Z})\right] \\ \mathbf{w}^{t+1} = \frac{\mathbf{w}^{t+1/2}}{\left\|\mathbf{w}^{t+1/2}\right\|} \end{cases}$$

- Expected coarse gradient:

$$\mathbf{E}_\mathbf{Z}\left[\mathbf{g}(\mathbf{v}, \mathbf{w}; \mathbf{Z})\right] = \frac{h(\mathbf{v}, \mathbf{v}^*)}{2\sqrt{2\pi}} \frac{\mathbf{w}}{\|\mathbf{w}\|} - \cos\left(\frac{\theta(\mathbf{w}, \mathbf{w}^*)}{2}\right) \frac{\mathbf{v}^\top \mathbf{v}^*}{\sqrt{2\pi}} \frac{\frac{\mathbf{w}}{\|\mathbf{w}\|} + \mathbf{w}^*}{\left\|\frac{\mathbf{w}}{\|\mathbf{w}\|} + \mathbf{w}^*\right\|}$$

$$h(\mathbf{v}, \mathbf{v}^*) := \|\mathbf{v}\|^2 + (\mathbf{1}^\top \mathbf{v})^2 - (\mathbf{1}^\top \mathbf{v})(\mathbf{1}^\top \mathbf{v}^*) + \mathbf{v}^\top \mathbf{v}^*.$$

- Critical point conditions and global minimizer are the same as those of the population loss.

Two-Layer Neural Network Regression Problem

- Coarse partial gradient $\mathbf{E}_{\mathbf{Z}}\left[\mathbf{g}(\mathbf{v},\mathbf{w};\mathbf{Z})\right] = \mathbf{0}$ is well-defined at global minimizer, $\mathbf{v} = \mathbf{v}^*$, $\theta(\mathbf{w},\mathbf{w}^*) = 0$, of the population loss. In contrast, the true gradient $\frac{\partial f}{\partial \mathbf{w}}(\mathbf{v},\mathbf{w})$ does not exist.
- Coarse gradient is positively correlated with the true gradient.

Theorem (Positive Correlation)

If $\theta(\mathbf{w},\mathbf{w}^*) \in (0,\pi)$, and $\|\mathbf{w}\| \neq 0$, the inner product between the coarse and true gradients w.r.t. $\mathbf{w}$:

$$\left\langle \mathbf{E}_{\mathbf{Z}}\left[\mathbf{g}(\mathbf{v},\mathbf{w};\mathbf{Z})\right], \frac{\partial f}{\partial \mathbf{w}}(\mathbf{v},\mathbf{w}) \right\rangle = \frac{\sin\left(\theta(\mathbf{w},\mathbf{w}^*)\right)}{2(\sqrt{2\pi})^3\|\mathbf{w}\|}(\mathbf{v}^\top\mathbf{v}^*)^2 \geq 0.$$

Two-Layer Neural Network Regression Problem

- Minus coarse gradient is a descent direction.

Theorem (Coarse Gradient Descent and Convergence to Global Minimizer)

Given the initialization $(\mathbf{v}^0, \mathbf{w}^0)$ *with* $\|\mathbf{w}^0\| = 1$*, and let* $\{(\mathbf{v}^t, \mathbf{w}^t)\}$ *be the sequence generated by the normalized coarse gradient descent algorithm. There exists* $\eta_0 > 0$*, such that for any step size* $\eta < \eta_0$*,* $\{f(\mathbf{v}^t, \mathbf{w}^t)\}$ *is monotonically decreasing, both* $\left\|\mathbf{E}_\mathbf{Z}\left[\frac{\partial \ell}{\partial \mathbf{v}}(\mathbf{v}^t, \mathbf{w}^t; \mathbf{Z})\right]\right\|$ *and* $\|\mathbf{E}_\mathbf{Z}[\mathbf{g}(\mathbf{v}^t, \mathbf{w}^t; \mathbf{Z})]\|$ *converge to 0, as* $t \to \infty$*.*

Morover, if the initialization $(\mathbf{v}^0, \mathbf{w}^0)$ *satisfies geometric conditions:* $\theta(\mathbf{v}^0, \mathbf{v}^*) < \frac{\pi}{2}$*,* $\theta(\mathbf{w}^0, \mathbf{w}^*) < \frac{\pi}{2}$*, and* $(\mathbf{1}^\top \mathbf{v}^*)(\mathbf{1}^\top \mathbf{v}^0) \leq (\mathbf{1}^\top \mathbf{v}^*)^2$*, then* $\{(\mathbf{v}^t, \mathbf{w}^t)\}$ *converges to the global minimizer* $(\mathbf{v}^*, \mathbf{w}^*)$*.*

- Same sufficient conditions required for convergence of gradient descent to global minimizer in the 2-layer model with regular ReLU activation (Du, Lee, Tian, Póczos, Singh, '18).

Conclusion and Future Work

- Coarse gradients are simple and effective for SGD training of fully quantized deep neural networks.
- Blending enhances classification accuracy in the low bit-width regime.
- Proved positive correlation between expected coarse gradient and true gradient, and convergence of a coarse gradient descent algorithm in 2-layer neural network regression model with Gaussian input data.
- Further understanding of coarse gradient descent for large scale optimization problems with no or vanishing gradients, and non-Gaussian data.

Thank You !

Questions ?