Batch Normalization Preconditioning for Neural Network Training

Qiang Ye

(Joint work with Susanna Lange and Kyle Helfrich)
University of Kentucky

Precond 24 - Atlanta

Works Supported by NSF

Outline

1. Neural Networks Training
2. Batch Normalization Preconditioning (BNP)
3. Experiments

Supervised Learning

Supervised Learning

Given a labeled data set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N} \subset \mathbf{R}^m \times \mathbf{R}^n$, fit a parametric family of functions $y = f(\mathbf{x}, \theta) \in \mathbf{R}^m \times \mathbf{R}^p \to \mathbf{R}^n$ to the data;

Supervised Learning

Supervised Learning

Given a labeled data set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N \subset \mathbf{R}^m \times \mathbf{R}^n$, fit a parametric family of functions $y = f(\mathbf{x}, \theta) \in \mathbf{R}^m \times \mathbf{R}^p \to \mathbf{R}^n$ to the data;

- Use a neural network for $f(x, \theta)$
- Choose a loss function $L(f(x_i, \theta), y_i)$
- find $\theta \in \mathbf{R}^p$ by minimizing $\mathcal{L}(\theta) := \frac{1}{N}\sum_{i=1}^N L(f(x_i, \theta), y_i)$

Neural Network

- Composition function:

$$f(\mathbf{x}, \theta) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$$

- hidden variables at ℓ-th layer:

$$\begin{aligned} h^{(\ell)} &= f^{(\ell)}(h^{(\ell-1)}) \\ &:= g(W^{(\ell)} h^{(\ell-1)} + b^{(\ell)}) \end{aligned}$$

- $g(t)$: an elementwise nonlinear activation function, e.g. ReLU:

$$g(t) = \max\{t, 0\}$$

Optimization/Training

Gradient descent:

$$\theta \leftarrow \theta - \lambda \nabla \mathcal{L}(\theta)$$

- $\lambda > 0$ - learning rate
- Mini-batch training: sample a mini-batch $\{x_{i_1}, x_{i_2}, \cdots, x_{i_N}\}$ and train with

$$\nabla \mathcal{L}(\theta) = \frac{1}{N} \sum_{j=1}^{N} \nabla L(f(x_{i_j}, \theta), y_{i_j})$$

- Accelerations: Momentum, Adagrad, RMSProp, Adams, Batch normalization (BN), Layer normalization (LN), ...

Convergence of Gradient Descent

Gradient Descent:

$$\theta_{k+1} \leftarrow \theta_k - \alpha \nabla_\theta L(\theta)$$

Let θ^* be a local minimizer.

Convergence of Gradient Descent

Gradient Descent:

$$\theta_{k+1} \leftarrow \theta_k - \alpha \nabla_\theta L(\theta)$$

Let θ^* be a local minimizer.

- Local convergence if $\alpha < 2/\|\nabla_\theta^2 L(\theta^*)\|$.
- Optimal convergence rate:

$$r = \frac{\kappa - 1}{\kappa + 1} + \epsilon$$

where $\kappa = \kappa(\nabla_\theta^2 \mathcal{L}(\theta^*))$ is the condition number.

Preconditioned Gradient Descent

Consider a change of variable: $\theta = Pz$ and $L = L(\theta) = L(Pz)$.

Preconditioned Gradient Descent

Consider a change of variable: $\theta = Pz$ and $L = L(\theta) = L(Pz)$.

- Gradient Descent in z:

$$z_{k+1} = z_k - \alpha P^T \nabla_\theta L\left(Pz_k\right)$$

Preconditioned Gradient Descent

Consider a change of variable: $\theta = Pz$ and $L = L(\theta) = L(Pz)$.

- Gradient Descent in z:

$$z_{k+1} = z_k - \alpha P^T \nabla_\theta L\left(Pz_k\right)$$

  Equivalently

$$\theta_{k+1} = \theta_k - \alpha PP^T \nabla_\theta L\left(\theta_k\right).$$

Preconditioned Gradient Descent

Consider a change of variable: $\theta = Pz$ and $L = L(\theta) = L(Pz)$.

- Gradient Descent in z:

$$z_{k+1} = z_k - \alpha P^T \nabla_\theta L\,(Pz_k)$$

  Equivalently

$$\theta_{k+1} = \theta_k - \alpha P P^T \nabla_\theta L\,(\theta_k)\,.$$

- Convergence rate dependent on

$$\kappa' = \kappa(\nabla_z^2 L\,(Pz^*)) = \kappa(P^T \nabla_\theta^2 L\,(\theta^*)\,P).$$

Preconditioned Gradient Descent

Consider a change of variable: $\theta = Pz$ and $L = L(\theta) = L(Pz)$.

- Gradient Descent in z:

$$z_{k+1} = z_k - \alpha P^T \nabla_\theta L\,(Pz_k)$$

  Equivalently

$$\theta_{k+1} = \theta_k - \alpha P P^T \nabla_\theta L\,(\theta_k)\,.$$

- Convergence rate dependent on

$$\kappa' = \kappa(\nabla_z^2 L\,(Pz^*)) = \kappa(P^T \nabla_\theta^2 L\,(\theta^*)\,P).$$

- Preconditioning: choose P such that $P^T \nabla_\theta^2 L\,(\theta^*)\,P$ has a better condition number

Neural Networks Loss

Consider one weight and bias for layer ℓ. Recall

$$h^{(\ell)} = g\left(W^{(\ell)} h^{(\ell-1)} + b^{(\ell)}\right) \in \mathbb{R}^n$$

Let ${w_i^{(\ell)}}^T \in \mathbb{R}^{1\times m}$ be the ith row of $W^{(\ell)}$ and $b_i^{(\ell)}$ be the ith entry of $b^{(\ell)}$. Let

$$a_i^{(\ell)} = {w_i^{(\ell)}}^T h^{(\ell-1)} + b_i^{(\ell)} = \widehat{w}^T \widehat{h} \in \mathbb{R}$$

where

$$\widehat{w}^T = \left[b_i^{(\ell)}, {w_i^{(\ell)}}^T\right] \in \mathbb{R}^{1\times(m+1)},\ \widehat{h} = \begin{bmatrix} 1 \\ h^{(\ell-1)} \end{bmatrix} \in \mathbb{R}^{(m+1)\times 1},$$

Neural Network Loss Hessian

Theorem 1

Consider a loss function L *and write* $L = L\left(a_i^{(\ell)}\right) = L\left(\widehat{w}^T\widehat{h}\right)$*. When training over a mini-batch of* N *inputs, let* $\{h_1^{(\ell-1)}, h_2^{(\ell-1)}, \ldots, h_N^{(\ell-1)}\}$ *be the associated* $h^{(\ell-1)}$ *and let* $\widehat{h}_j = \begin{bmatrix} 1 \\ h_j^{(\ell-1)} \end{bmatrix} \in \mathbb{R}^{(m+1)\times 1}$*. Let* $\mathcal{L} = \mathcal{L}(\widehat{w}) := \frac{1}{N}\sum_{j=1}^{N} L\left(\widehat{w}^T\widehat{h}_j\right)$*.*

Then,

$$\nabla^2_{\widehat{w}}\mathcal{L}(\widehat{w}) = \widehat{H}^T S \widehat{H}$$

where

$$\widehat{H} = \begin{bmatrix} 1 & h_1^{(\ell-1)^T} \\ \vdots & \vdots \\ 1 & h_N^{(\ell-1)^T} \end{bmatrix} \text{ and } S = \frac{1}{N}\begin{bmatrix} L''\left(\widehat{w}^T\widehat{h}_1\right) & & \\ & \ddots & \\ & & L''\left(\widehat{w}^T\widehat{h}_N\right) \end{bmatrix},$$

Batch Normalization Preconditioning (BNP)

Precondition $\widehat{H} = [e,\ H]$:

- $\widehat{w} = Pz$, where

$$P := UD, \quad U := \begin{bmatrix} 1 & -\mu_A^T \\ 0 & I \end{bmatrix}, \quad D := \begin{bmatrix} 1 & 0 \\ 0 & \operatorname{diag}(\sigma_A) \end{bmatrix}^{-1},$$

where

$$\mu_A := \frac{1}{N}\sum_{j=1}^{N} h_j^{(\ell-1)}, \quad \text{and} \quad \sigma_A^2 := \frac{1}{N}\sum_{j=1}^{N}(h_j^{(\ell-1)} - \mu_A)^2$$

Preconditioned Gradient Descent

Theorem 2

The preconditioned Hessian matrix is

$$\nabla_z^2 \mathcal{L} = P^T(\widehat{H}^T S \widehat{H})P = \widehat{G}^T S \widehat{G}.$$

where $\widehat{G} := \widehat{H}P$, *i.e.*

$$\widehat{G} = \begin{bmatrix} 1 & g_1^T \\ \vdots & \vdots \\ 1 & g_N^T \end{bmatrix} = \begin{bmatrix} 1 & h_1^{(\ell-1)^T} \\ \vdots & \vdots \\ 1 & h_N^{(\ell-1)^T} \end{bmatrix} \begin{bmatrix} 1 & -\mu_A^T \\ 0 & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & diag(\sigma_A) \end{bmatrix}^{-1}, \qquad (1)$$

and $g_j = (h_j^{(\ell-1)} - \mu_A)/\sigma_A$ *is* $h_j^{(\ell-1)}$ *normalized to have zero mean and unit variance.*

Preconditioned Gradient Descent

$\widehat{G} = \widehat{H}UD$ improves conditioning:

Preconditioned Gradient Descent

$\widehat{G} = \widehat{H}UD$ improves conditioning:

$$\widehat{H}U = \begin{bmatrix} 1 & {h_1^{(\ell-1)}}^T - \mu_A^T \\ \vdots & \vdots \\ 1 & {h_N^{(\ell-1)}}^T - \mu_A^T \end{bmatrix}$$

and

$$e^T(H - e\mu_A^T) = 0.$$

Preconditioned Gradient Descent

$\widehat{G} = \widehat{H}UD$ improves conditioning:

$$\widehat{H}U = \begin{bmatrix} 1 & h_1^{(\ell-1)^T} - \mu_A^T \\ \vdots & \vdots \\ 1 & h_N^{(\ell-1)^T} - \mu_A^T \end{bmatrix}$$

and

$$e^T(H - e\mu_A^T) = 0.$$

Theorem 3

$$\kappa(\widehat{H}U) \leq \kappa(\widehat{H})$$

and (by a theorem of van der Sluis)

$$\kappa(\widehat{G}) \leq \sqrt{m+1} \min_{D_0 \text{ is diagonal}} \kappa(\widehat{H}UD_0).$$

Balancing the norms of Hessians

G's entries has mean 0 and variance 1. By a theorem of Seginer:

$$\mathbb{E}[\|G\|] \leq C \max\{\sqrt{m}, \sqrt{N}\}$$

Balancing the norms of Hessians

G's entries has mean 0 and variance 1. By a theorem of Seginer:

$$\mathbb{E}[\|G\|] \leq C \max\{\sqrt{m}, \sqrt{N}\}$$

Scale $\nabla_z^2 L(\theta^*)$ by $q = \max\{\sqrt{m/N}, 1\}$:

$$(1/q)\mathbb{E}[\|\widehat{G}\|] \leq C'\sqrt{N}$$

so that the norms are independent of m, which varies with layers.

Balancing the norms of Hessians

G's entries has mean 0 and variance 1. By a theorem of Seginer:

$$\mathbb{E}[\|G\|] \leq C \max\{\sqrt{m}, \sqrt{N}\}$$

Scale $\nabla_z^2 L(\theta^*)$ by $q = \max\{\sqrt{m/N}, 1\}$:

$$(1/q)\mathbb{E}[\|\widehat{G}\|] \leq C'\sqrt{N}$$

so that the norms are independent of m, which varies with layers.

- Learning rate: $\alpha < 2/\|\nabla_z^2 L(\theta^*)\|$.
- A large $\|\nabla_z^2 L(\theta^*)\|$ at one layer will require a smaller α;

BNP Algorithm

Preconditioning implicitly by modifying gradients.

BNP Gradients on $W^{(\ell)}, b^{(\ell)}$

Input: $A = \{h_1^{(\ell-1)}, h_2^{(\ell-1)}, \ldots, h_N^{(\ell-1)}\} \subset \mathbb{R}^m$ and the parameter gradients: $G_w \leftarrow \frac{\partial \mathcal{L}}{\partial W^{(\ell)}} \in \mathbb{R}^{n\times m}$, $G_b \leftarrow \frac{\partial \mathcal{L}}{\partial b^{(\ell)}} \in \mathbb{R}^{1\times n}$

1. Compute μ_A, σ_A^2;
2. Compute: $\mu \leftarrow \rho\mu + (1-\rho)\mu_A$, $\sigma^2 \leftarrow \rho\sigma^2 + (1-\rho)\sigma_A^2$;
3. Set $\tilde{\sigma}^2 = \sigma^2 + \epsilon_1 \max\{\sigma^2\} + \epsilon_2$ and $q^2 = \max\{m/N, 1\}$;
4. Update: $G_w \leftarrow \frac{1}{q}(G_w - \mu G_b)/\tilde{\sigma}^2$; $G_b \leftarrow \frac{1}{q} G_b - \mu^T G_w$;

Relation to Batch Normalzation (BN)

BN - `Ioffe and Szegedy (2015)` :

- For a mini-batch of inputs $\{x_1, x_2, \ldots, x_N\}$, the corresponding $\{h_1^{(\ell-1)}, h_2^{(\ell-1)}, \ldots, h_N^{(\ell-1)}\}$ has mean μ_A and variance σ_A^2.
- Normalize $h^{(\ell-1)}$:

$$h^{(\ell)} = g\left(W^{(\ell)}\mathcal{B}_{\beta,\gamma}\left(h^{(\ell-1)}\right) + b^{(\ell)}\right)$$

where $\mathcal{B}_{\beta,\gamma}\left(h^{(\ell-1)}\right) = \gamma\frac{h^{(\ell-1)}-\mu_A}{\sigma_A} + \beta$

Relation to Batch Normalzation (BN)

BN - `Ioffe and Szegedy (2015)` :

- For a mini-batch of inputs $\{x_1, x_2, \ldots, x_N\}$, the corresponding $\{h_1^{(\ell-1)}, h_2^{(\ell-1)}, \ldots, h_N^{(\ell-1)}\}$ has mean μ_A and variance σ_A^2.
- Normalize $h^{(\ell-1)}$:

$$h^{(\ell)} = g\left(W^{(\ell)} \mathcal{B}_{\beta,\gamma}\left(h^{(\ell-1)}\right) + b^{(\ell)}\right)$$

where $\mathcal{B}_{\beta,\gamma}\left(h^{(\ell-1)}\right) = \gamma \frac{h^{(\ell-1)} - \mu_A}{\sigma_A} + \beta$

Theorem 4

One step of gradient descent training of BN with $\mathcal{B}_{0,1}(\cdot)$ in $\{W^{(\ell)}, b^{(\ell)}\}$ without passing the gradient through μ_A, σ_A is equivalent to one step of BNP training of the vanilla network with parameter $\widehat{W}^T, \widehat{b}$.

Equivalence holds for one training step only.

Experiments

- Dataset: MNIST, CIFAR10
- Networks: Fully-Connected Neural Network (three hidden layers of size 100 each)

Fully Connected Network/MNIST

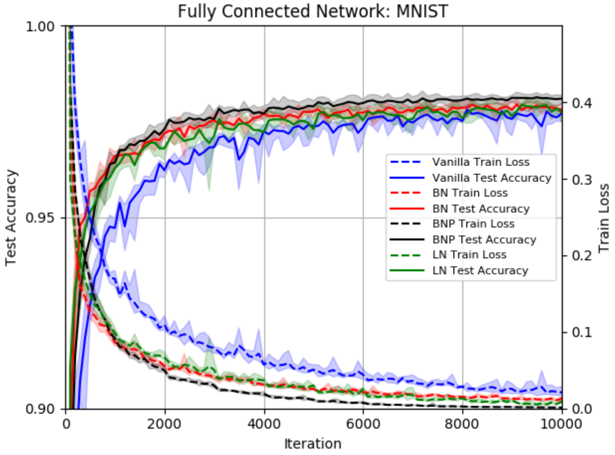

Figure: Mini-batch size = 60. Training loss (dashed lines) and test accuracy (solid lines)

Fully Connected Network/CIFAR 10

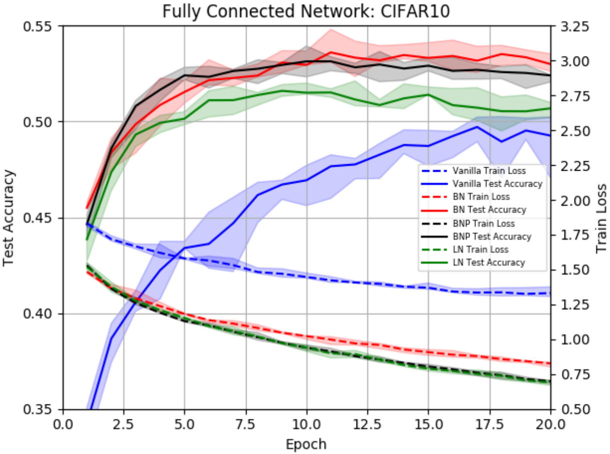

Figure: Mini-batch size = 60. Training loss (dashed lines) and test accuracy (solid lines)

Fully Connected Network/CIFAR 10

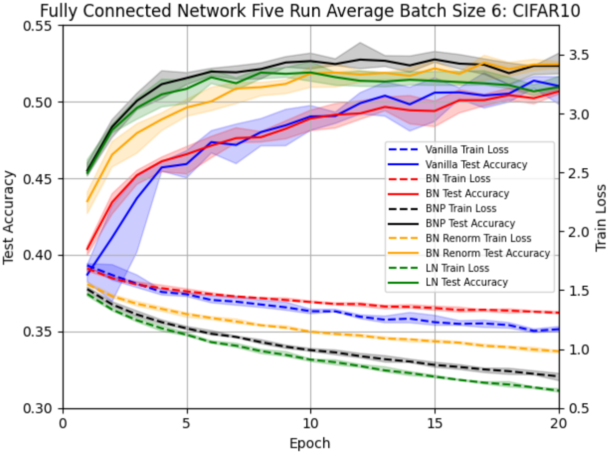

Figure: Mini-batch size = 6.

Fully Connected Network and CNN/CIFAR 10

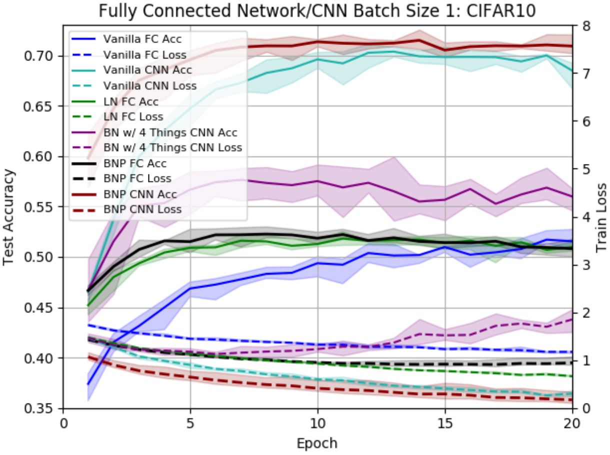

Figure: Mini-batch size = 1.

Conclusions

- Preconditioning framework applicable to a variety of networks.
- Outperform BN for small mini-batches.
- Provide partial theoretical justifications for BN.

Ref:

- Susanna Lange, Kyle Helfrich, and Qiang Ye, Batch Normalization Preconditioning for Neural Network Training, Journal of Machine Learning Research, 23(72):1-41, 2022. .