Centered Weight Normalization in Accelerating Training of Deep Neural Networks

北京航空航天大學 BEIHANG UNIVERSITY

Lei Huang, XiangLong Liu, Yang Liu, Bo Lang, Dacheng Tao[#]

State Key Lab of Software Development Environment, Beihang University, Beijing, China

[#]UBTECH Sydney AI Centre, School of IT, FEIT, The University of Sydney, Australia

{huanglei, xlliu, blonster, langbog}@nlsde.buaa.edu.cn, dacheng.tao@sydney.edu.au

ICCV 17 International Conference on Computer Vision 2017

1. Introduction

Optimization in Deep Model

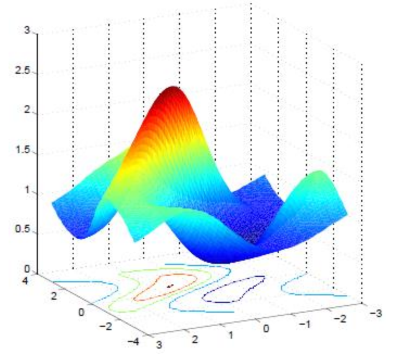
- Goal: $\theta^* = \arg\min_{\theta} \mathbb{E}_{(\mathbf{x},\mathbf{y})\in D}[\mathcal{L}(\mathbf{y}, f(\mathbf{x};\theta))]$
- Update Iteratively: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha^{(t)}\nabla^{(t)}$
- Challenge: Non-convex, ill conditioning

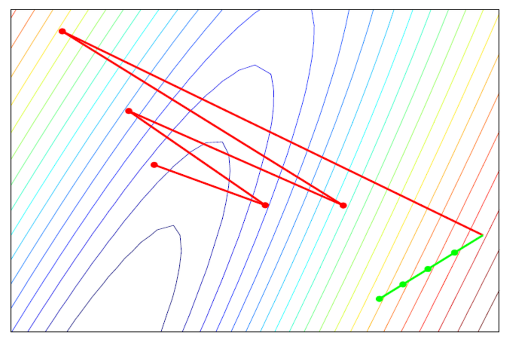
$$\mathbf{s}^l = \mathbf{W}^l\mathbf{h}^{l-1} + \mathbf{b}^l \qquad \mathbf{h}^l = \varphi(\mathbf{s}^l)$$

Stochastic gradient descent

- Gradient is averaged by the sampled examples

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{1}{m}\Sigma_{i=1}^{m}\frac{\partial \mathcal{L}(\mathbf{y}_i, f(\mathbf{x}_i;\theta))}{\partial \theta}$$





Estimate curvature or scale

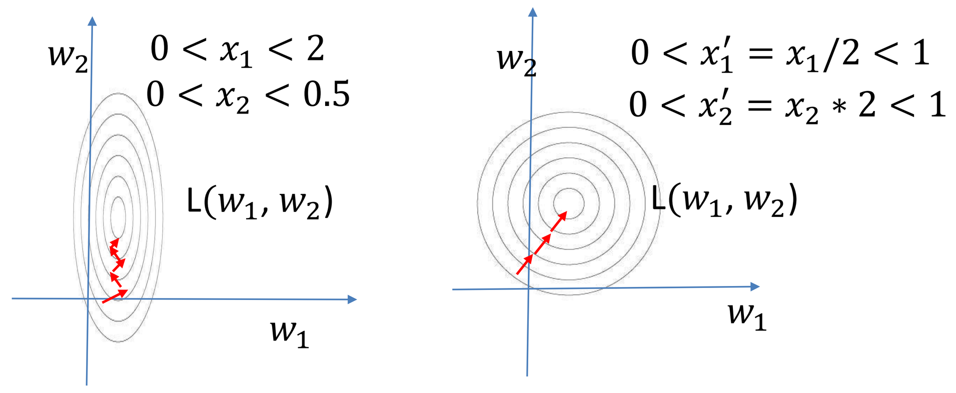
- Quadratic optimization: Newton, quasi-Newton, Natural Gradient
- Estimate the scale: AdaGrad, Rmsprop, Adam

$y = w_1x_1 + w_2x_2$+b

L=$(y-\hat{y})$^2

Normalize input/activation

- Normalize explicitly: batch normalization
- Normalize implicitly(constrain weights): weight normalization



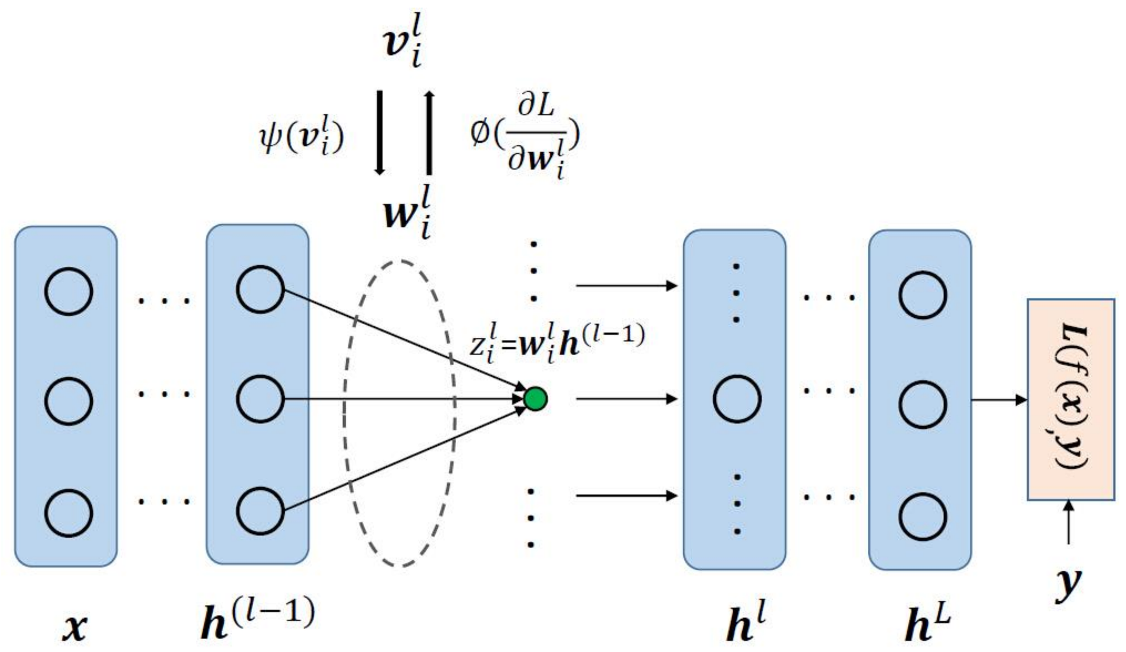
2. Motivation

Initialization methods

- Random, Xavier, MSRInit: Zero mean, stable-variance
- Keep desired characters during training

Formulation

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{(\mathbf{x},\mathbf{y})\in D}[\mathcal{L}(\mathbf{y}, f(\mathbf{x};\theta))]$$
$$s.t. \quad \mathbf{w}^T\mathbf{1} = 0 \ and \ \|\mathbf{w}\| = 1$$



3. Method

Solution by re-parameterization

- Proxy parameter v:

$$\mathbf{w} = \frac{\mathbf{v} - \frac{1}{d}\mathbf{1}(\mathbf{1}^T\mathbf{v})}{\|\mathbf{v} - \frac{1}{d}\mathbf{1}(\mathbf{1}^T\mathbf{v})\|}$$

- Adjustable scale: $z = g\,\mathbf{w}^T\mathbf{h} + b$
- Back-propagated Gradient

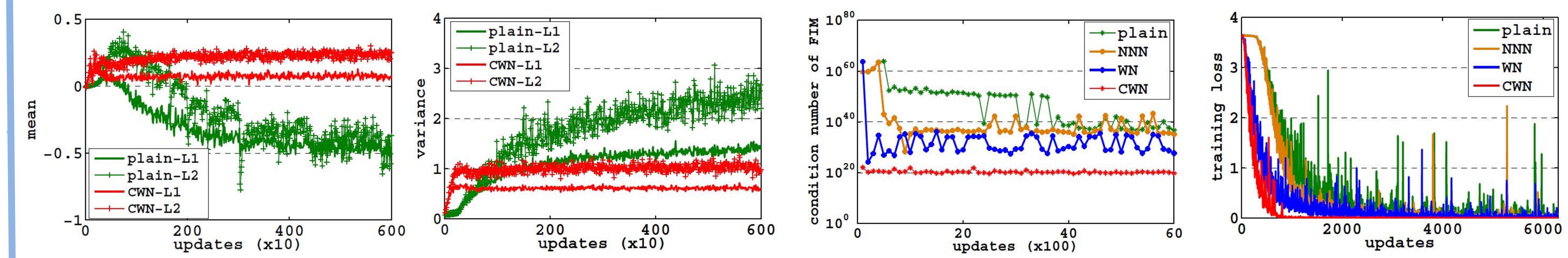
$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = \frac{1}{\|\hat{\mathbf{v}}\|}[\frac{\partial \mathcal{L}}{\partial \mathbf{w}} - (\frac{\partial \mathcal{L}}{\partial \mathbf{w}}\mathbf{w})\mathbf{w}^T - \frac{1}{d}(\frac{\partial \mathcal{L}}{\partial \mathbf{w}}\mathbf{1})\mathbf{1}^T]$$

Beneficial Properties

- Stabilize the distributions
- Better Conditioning of Hessian

$$\frac{\partial L}{\partial v}\cdot \mathbf{1}=0$$

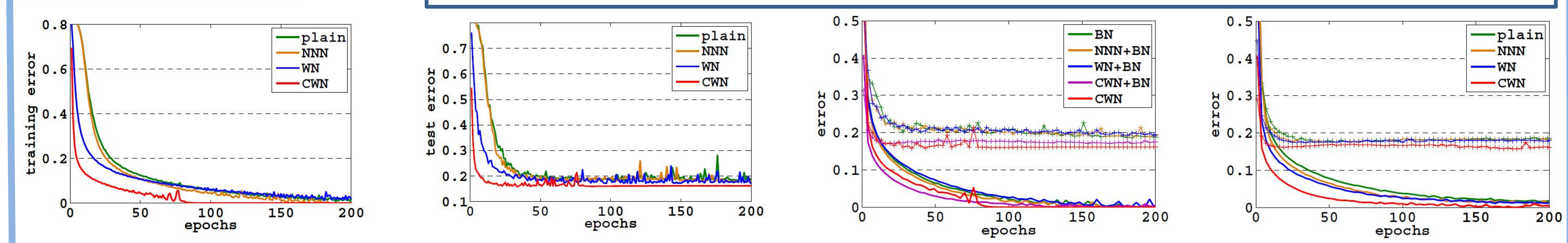
Proposition 1. *Let $z = \mathbf{w}^T\mathbf{h}$, where $\mathbf{w}^T\mathbf{1} = 0$ and $\|\mathbf{w}\| = 1$. Assume $\mathbf{h}$ has Gaussian distribution with the mean: $\mathbb{E}_{\mathbf{h}}[\mathbf{h}] = \mu\mathbf{1}$, and covariance matrix: $cov(\mathbf{h}) = \sigma^2\mathbf{I}$, where $\mu \in \mathbb{R}$ and $\sigma^2 \in \mathbb{R}$. We have $\mathbb{E}_z[z] = 0$, $var(z) = \sigma^2$.*



4. Experiments

Simply replace Linear/Conv module with CWN module!

MLP; SVHN



SGD: training | SGD: test | SGD+BN | Adam

CNN architecture

- BN-Inception

	Cifar-10	Cifar-100
Plain	6.14 ±0.04	25.52 ±0.15
WN	6.18 ±0.34	25.49 ±0.35
WCBN	**6.01 ±0.16**	**24.45 ±0.54**

- 56 layers residual network

	Cifar-10	Cifar-100
Plain	7.34 ±0.52	29.38 ±0.14
WN	7.58 ±0.40	29.85 ±0.66
WCBN	**6.85 ±0.25**	**29.23 ±0.14**

- BN-Inception, ImageNet 2012

Methods	Top-1 error	Top-5 error
plain	30.78	11.14
WN	28.64	9.7
CWN	**26.1**	**8.35**

Code: https://github.com/huangleiBuaa/CenteredWN