A Quantitative Analysis of the Effect of Batch Normalization on Gradient Descent

Yongqiang Cai[1], Qianxiao Li[1,2], Zuowei Shen[1]

9-15 June 2019 (ICML), Long Beach, CA, USA

[1]Department of Mathematics, National University of Singapore, Singapore

[2]Institute of High Performance Computing, A*STAR, Singapore

Batch Normalization

A vanilla fully-connected layer

$$z = \sigma(Wu + b).$$

With batch normalization (Ioffe & Szegedy 2015):

$$z = \sigma(\gamma \mathsf{N}(Wu) + \beta), \qquad \mathsf{N}(\xi) := \frac{\xi - \mathbb{E}[\xi]}{\sqrt{\mathsf{Var}[\xi]}}.$$

Batch normalization works well in practice, e.g. allows stable training with large learning rates, works well in high dimensions or ill-conditioned problems

Related work on BN [Ma & Klabjan (2017); Kohler et al. (2018); Arora et al. (2019)]

Batch Normalization

A vanilla fully-connected layer

$$z = \sigma(Wu + b).$$

With batch normalization (Ioffe & Szegedy 2015):

$$z = \sigma(\gamma \mathsf{N}(Wu) + \beta), \qquad \mathsf{N}(\xi) := \frac{\xi - \mathbb{E}[\xi]}{\sqrt{\mathsf{Var}[\xi]}}.$$

Batch normalization works well in practice, e.g. allows stable training with large learning rates, works well in high dimensions or ill-conditioned problems

Related work on BN [Ma & Klabjan (2017); Kohler et al. (2018); Arora et al. (2019)]

Question: Can we quantify the precise effect of BN on gradient descent (GD)?

Batch Normalization on Ordinary Least Squares

Linear regression model:

Input: $x \in \mathbb{R}^d$ Label: $y \in \mathbb{R}$ Model: $y = x^T w^* + \text{noise}$

Batch Normalization on Ordinary Least Squares

Linear regression model:

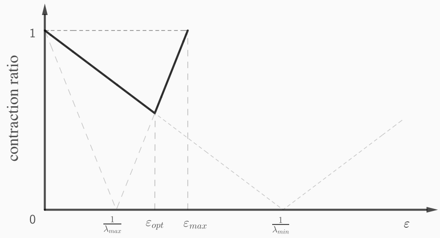
Input: $x \in \mathbb{R}^d$ Label: $y \in \mathbb{R}$ Model: $y = x^T w^* + \text{noise}$

OLS regression without BN

Optimization problem: $\min_w J_0(w) := \mathbb{E}_{x,y}[\frac{1}{2}(y - x^T w)^2]$

Gradient descent dynamics: $w_{k+1} = w_k - \varepsilon \nabla_w J_0(w_k) = w_k + \varepsilon(g - H w_k)$,

where $H := \mathbb{E}[xx^T]$, $g := \mathbb{E}[xy]$, $c := \mathbb{E}[y^2]$.



Batch Normalization on Ordinary Least Squares

Linear regression model:

Input: $x \in \mathbb{R}^d$ Label: $y \in \mathbb{R}$ Model: $y = x^T w^* + \text{noise}$

OLS regression with BN

Optimization problem: $\min_{a,w} J(a,w) = \mathbb{E}_{x,y}\left[\frac{1}{2}\left(y - a\ \mathsf{N}(x^T w)\right)^2\right]$

Gradient descent dynamics:

$$\begin{cases} a_{k+1} = a_k - \varepsilon_a \nabla_a J(a_k, w_k) = a_k + \varepsilon_a \left(\dfrac{w_k^T g}{\sqrt{w_k^T H w_k}} - a_k \right), \\ w_{k+1} = w_k - \varepsilon \nabla_w J(a_k, w_k) = w_k + \dfrac{\varepsilon a_k}{\sqrt{w_k^T H w_k}} \left(g - \dfrac{w_k^T g}{w_k^T H w_k} H w_k \right). \end{cases}$$

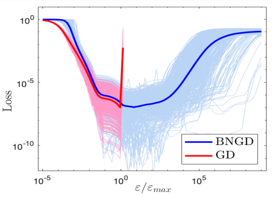
How does this compare with the GD case?

$$w_{k+1} = w_k - \varepsilon \nabla_w J_0(w_k) = w_k + \varepsilon (g - H w_k)$$

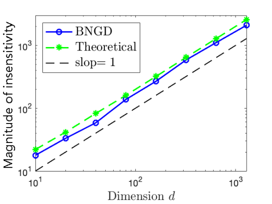
Properties of interest: convergence, robustness

Summary of Theoretical Results

Property	Gradient Descent	Gradient Descent with BN
Convergence	only for small ε	arbitrary ε provided $\varepsilon_a \leq 1$
Convergence Rate	linear	linear (can be faster)
Robustness to Learning Rates	small range of ε	wide range of ε
Robustness to Dimensions	no effect	the higher the better



(a) Loss of GD and BNGD($d = 100$)

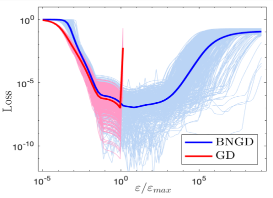


(b) Effect of dimension on BNGD

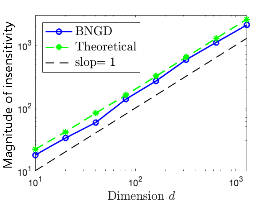
Summary of Theoretical Results

Property	Gradient Descent	Gradient Descent with BN
Convergence	only for small ε	arbitrary ε provided $\varepsilon_a \leq 1$
Convergence Rate	linear	linear (can be faster)
Robustness to Learning Rates	small range of ε	wide range of ε
Robustness to Dimensions	no effect	the higher the better

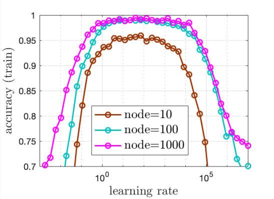
- Those properties are also observed in neural network experiments.



(a) Loss of GD and BNGD($d = 100$)



(b) Effect of dimension on BNGD



(c) Accuracy of BNGD on MNIST

Poster: Pacific Ballroom #54