Optimization for Training Deep Models

CMPT 498/898 Deep Learning and Applications

Najeeb Khan
najeeb.khan@usask.ca

Slides based on Deep Learning, Goodfellow et al. 2016 (Ch. 8)

University of Saskatchewan

Winter 2020

Review

opt vs Training.

SGD:

θ = get_initial_params();

ϵ = 0.001

while stopping criterion not met {

x, y = get_mini_batch();

Compute $\nabla_\theta J$

$$\theta \leftarrow \theta - \epsilon \nabla$$

Parameter Initialization

Algorithms are strongly affected by the choice of initialization

Parameter Initialization

Algorithms are strongly affected by the choice of initialization
No two units should have the same weights

Parameter Initialization

Algorithms are strongly affected by the choice of initialization
No two units should have the same weights
Random initialization from a high-entropy distribution

Parameter Initialization

Algorithms are strongly affected by the choice of initialization
No two units should have the same weights
Random initialization from a high-entropy distribution

- Computationally cheaper

Parameter Initialization

Algorithms are strongly affected by the choice of initialization
No two units should have the same weights
Random initialization from a high-entropy distribution

- Computationally cheaper
- Unlikely to assign any units to compute the same function as each other

Scale of the distribution?

Parameter Initialization

Algorithms are strongly affected by the choice of initialization
No two units should have the same weights
Random initialization from a high-entropy distribution

- Computationally cheaper
- Unlikely to assign any units to compute the same function as each other

Scale of the distribution?
Smaller initial weights

Parameter Initialization

Algorithms are strongly affected by the choice of initialization
No two units should have the same weights
Random initialization from a high-entropy distribution

- Computationally cheaper
- Unlikely to assign any units to compute the same function as each other

Scale of the distribution?
Smaller initial weights $\rightarrow$ vanishing gradient

Parameter Initialization

Algorithms are strongly affected by the choice of initialization
No two units should have the same weights
Random initialization from a high-entropy distribution

- Computationally cheaper
- Unlikely to assign any units to compute the same function as each other

Scale of the distribution?
Smaller initial weights $\rightarrow$ vanishing gradient
Larger initial weights

Parameter Initialization

Algorithms are strongly affected by the choice of initialization
No two units should have the same weights
Random initialization from a high-entropy distribution

- Computationally cheaper
- Unlikely to assign any units to compute the same function as each other

Scale of the distribution?
Smaller initial weights $\rightarrow$ vanishing gradient
Larger initial weights $\rightarrow$ exploding values during forward/backward propagation

Parameter Initialization

Algorithms are strongly affected by the choice of initialization
No two units should have the same weights
Random initialization from a high-entropy distribution

- Computationally cheaper
- Unlikely to assign any units to compute the same function as each other

Scale of the distribution?
Smaller initial weights $\rightarrow$ vanishing gradient
Larger initial weights $\rightarrow$ exploding values during forward/backward propagation

Xavier initialization:

$$W_{i,j} = U\Big(-\frac{\sqrt{6}}{\sqrt{m+n}}, \frac{\sqrt{6}}{\sqrt{m+n}}\Big)$$

Batch Normalization

Gradient based learning:

Batch Normalization

Gradient based learning:

> Parameter updates under the assumption that the other layers do not change

Batch Normalization

Gradient based learning:

- Parameter updates under the assumption that the other layers do not change
- Gradient updates are computed and applied simultaneously

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g}$$

Batch Normalization

Gradient based learning:

Parameter updates under the assumption that the other layers do not change

Gradient updates are computed and applied simultaneously

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g}$$

This is problematic for **deep** neural networks

Problem

Problem

Solution: Batch Normalization

Let $\mathbf{H}$ be a mini-batch of activations of the layer
Replace it with

$$\mathbf{H}' = \frac{\mathbf{H} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$$

Solution: Batch Normalization

Normalizing a unit can reduce the expressive power of the neural network containing that unit

Solution: Batch Normalization

Normalizing a unit can reduce the expressive power of the neural network containing that unit

Make the mean and standard deviation a trainable parameter

$$\boldsymbol{\gamma} \boldsymbol{H}' + \boldsymbol{\beta}$$