Normalization Techniques

Devansh Arpit

Table of Contents

1. Introduction
2. Motivation
3. Batch Normalization
4. Normalization Propagation
5. Weight Normalization
6. Layer Normalization
7. Conclusion

Introduction

- Task:

  Train deep networks using gradient descent based methods

Introduction

- Task:

  Train deep networks using gradient descent based methods

- Goal:

  Improve optimization for faster convergence

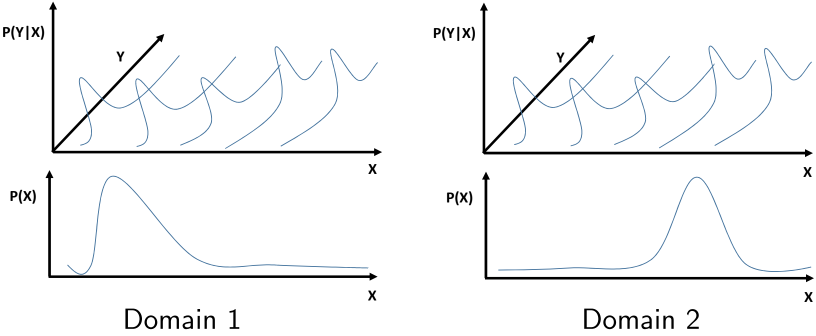
Table of Contents

1. Introduction
2. Motivation
3. Batch Normalization
4. Normalization Propagation
5. Weight Normalization
6. Layer Normalization
7. Conclusion

MILA

Motivation 1/2– Covariate Shift (Shimodaira, 2000)

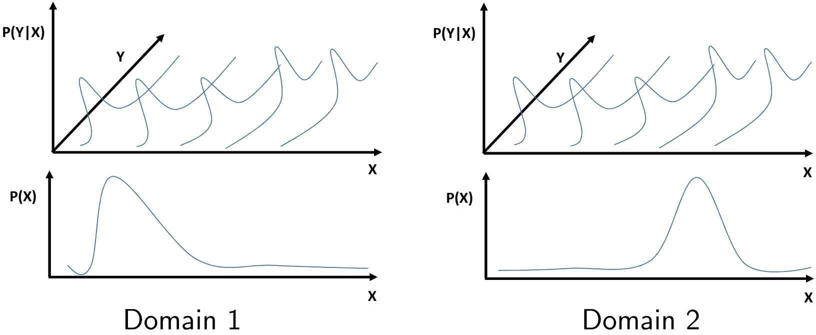
- Transfer mapping from domain 1 to domain 2:
  $\mathbf{X} \xrightarrow{P(\mathbf{Y}|\mathbf{X})} \mathbf{Y}$



- $P(\mathbf{Y}|\mathbf{X})$ identical for both Domain
- Learn best approximation $P_\theta(\mathbf{Y}|\mathbf{X}) \approx P(\mathbf{Y}|\mathbf{X})$

Motivation 1/2– Covariate Shift (Shimodaira, 2000)

- Transfer mapping from domain 1 to domain 2:
  $\mathbf{X} \xrightarrow{P(\mathbf{Y}|\mathbf{X})} \mathbf{Y}$

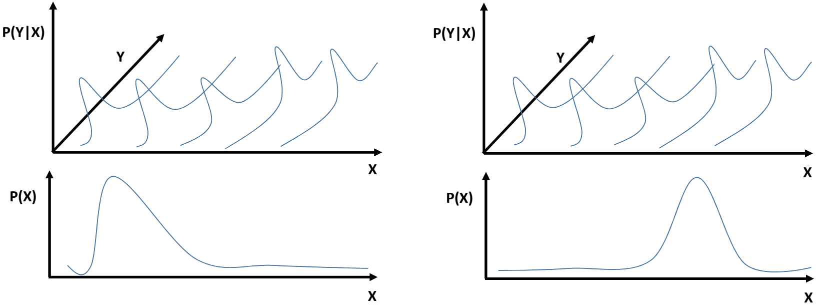


- $P(\mathbf{Y}|\mathbf{X})$ identical for both Domain
- Learn best approximation $P_\theta(\mathbf{Y}|\mathbf{X}) \approx P(\mathbf{Y}|\mathbf{X})$
- Maximum Likelihood focuses on high density regions of $\mathbf{X}$

Motivation 1/2– Covariate Shift (Shimodaira, 2000)

- Transfer mapping from domain 1 to domain 2:

$$\mathbf{X} \xrightarrow{P(\mathbf{Y}|\mathbf{X})} \mathbf{Y}$$

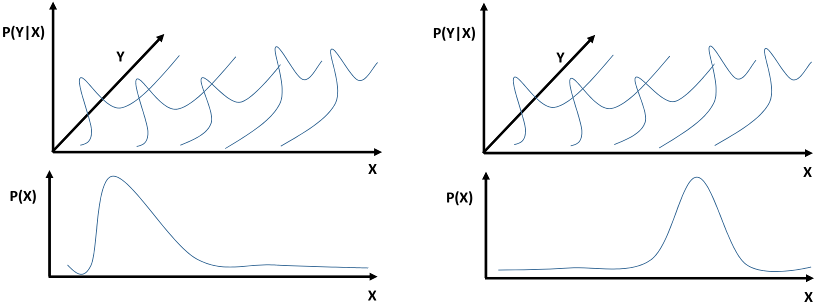


Domain 1 Domain 2

- $P(\mathbf{Y}|\mathbf{X})$ identical for both Domain
- Learn best approximation $P_\theta(\mathbf{Y}|\mathbf{X}) \approx P(\mathbf{Y}|\mathbf{X})$
- Maximum Likelihood focuses on high density regions of $\mathbf{X}$
- Low density regions of $\mathbf{X}$ are artifacts in $P_{\theta*}(\mathbf{Y}|\mathbf{X})$

Motivation 1/2– Covariate Shift (Shimodaira, 2000)

- Transfer mapping from domain 1 to domain 2:
  $\mathbf{X} \xrightarrow{P(\mathbf{Y}|\mathbf{X})} \mathbf{Y}$



Domain 1

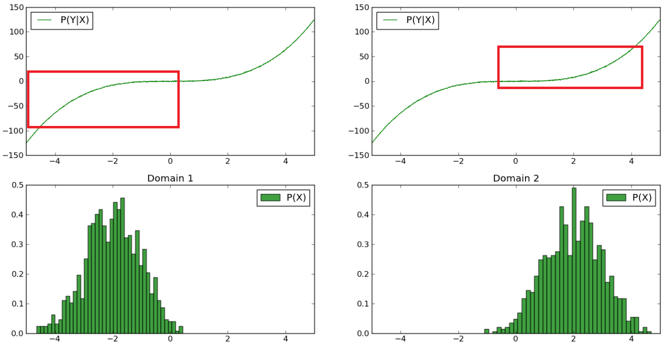
Domain 2

- $P(\mathbf{Y}|\mathbf{X})$ identical for both Domain
- Learn best approximation $P_\theta(\mathbf{Y}|\mathbf{X}) \approx P(\mathbf{Y}|\mathbf{X})$
- Maximum Likelihood focuses on high density regions of $\mathbf{X}$
- Low density regions of $\mathbf{X}$ are artifacts in $P_{\theta*}(\mathbf{Y}|\mathbf{X})$
- $P_{\theta*}(\mathbf{Y}|\mathbf{X})$ for Domain 1 $\neq P_{\theta*}(\mathbf{Y}|\mathbf{X})$ for Domain 2

Motivation 1/2– Covariate Shift (Shimodaira, 2000)

- Simulation:

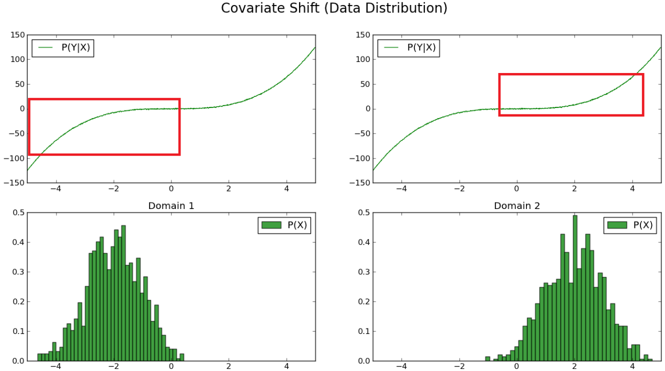
Covariate Shift (Data Distribution)



- Design: $P(X)$ is different but $P(Y|X)$ is identical

Motivation 1/2– Covariate Shift (Shimodaira, 2000)

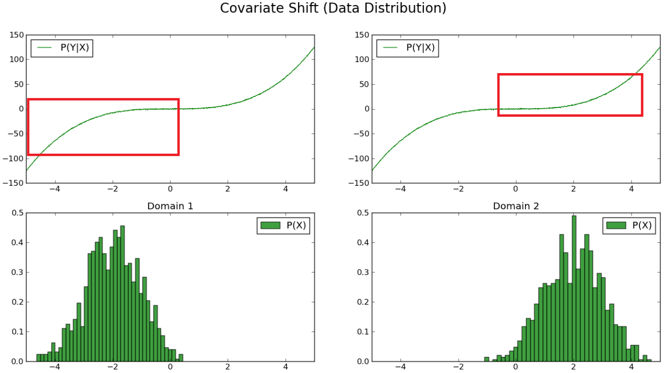
- Simulation:



- Design: $P(X)$ is different but $P(Y|X)$ is identical
- Sample $x \sim P(X)$ separately from the 2 domains, then generate y for each x using $P(Y|X)$

Motivation 1/2– Covariate Shift (Shimodaira, 2000)

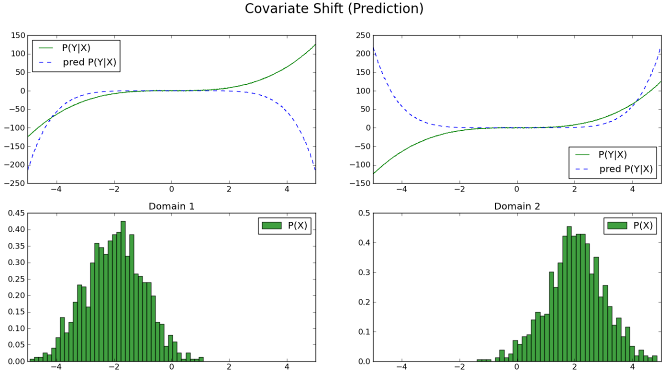
- Simulation:



- Design: $P(X)$ is different but $P(Y|X)$ is identical
- Sample $x \sim P(X)$ separately from the 2 domains, then generate y for each x using $P(Y|X)$
- Sampled points for each domain are along the curve contained in the red box

Motivation 1/2– Covariate Shift (Shimodaira, 2000)

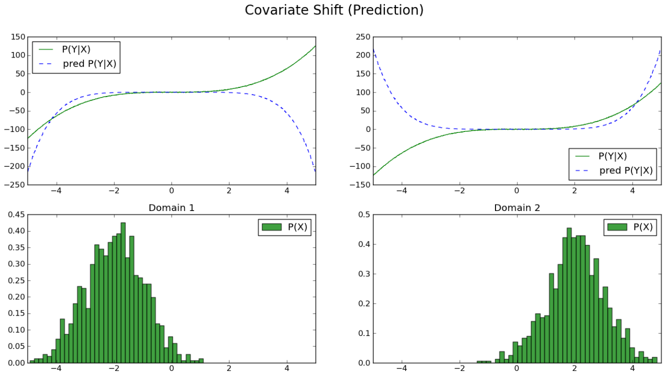
- Simulation:



- Learn $P_\theta(Y|X)$ separately using the sampled (x,y) pair

Motivation 1/2– Covariate Shift (Shimodaira, 2000)

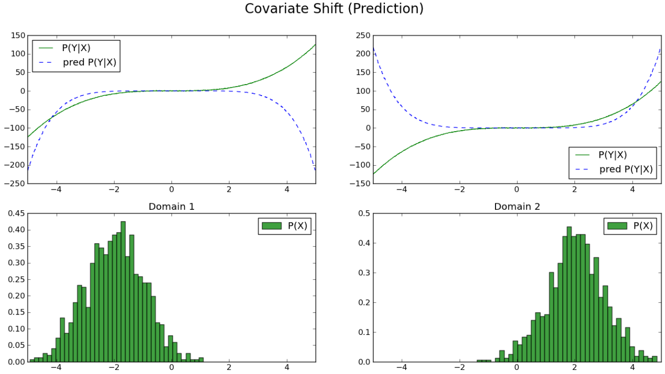
- Simulation:



- Learn $P_\theta(Y|X)$ separately using the sampled (x,y) pair
- Learned $P_{\theta*}(Y|X)$ for both domains are shown by dotted curves

Motivation 1/2– Covariate Shift (Shimodaira, 2000)

- Simulation:



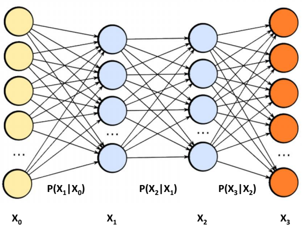
- Learn $P_\theta(Y|X)$ separately using the sampled (x,y) pair
- Learned $P_{\theta*}(Y|X)$ for both domains are shown by dotted curves
- Optimal $P_{\theta*}(Y|X)$ are different $\implies$ $P_{\theta*}(Y|X)$ not transferable

Motivation 1/2– Covariate Shift

- Internal Covariate Shift (Ioffe and Szegedy, 2015)

  $\mathbf{X}_i \xrightarrow{P(\mathbf{X}_{i+1}|\mathbf{X}_i)} \mathbf{X}_{i+1}$

- Multi-layer end-to-end learning model



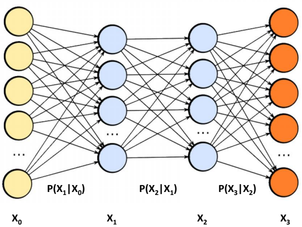
- Learn the best approximation $P_\theta(\mathbf{X}_{i+1}|\mathbf{X}_i) \approx P(\mathbf{X}_{i+1}|\mathbf{X}_i)$

Motivation 1/2– Covariate Shift

- Internal Covariate Shift (Ioffe and Szegedy, 2015)

  $\mathbf{X}_i \xrightarrow{P(\mathbf{X}_{i+1}|\mathbf{X}_i)} \mathbf{X}_{i+1}$

- Multi-layer end-to-end learning model



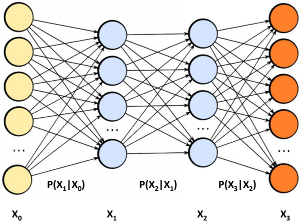
- Learn the best approximation $P_\theta(\mathbf{X}_{i+1}|\mathbf{X}_i) \approx P(\mathbf{X}_{i+1}|\mathbf{X}_i)$
- During SGD updates, hidden layer $P(\mathbf{X}_i)$ keeps shifting

Motivation 1/2– Covariate Shift

- Internal Covariate Shift (Ioffe and Szegedy, 2015)

  $\mathbf{X}_i \xrightarrow{P(\mathbf{X}_{i+1}|\mathbf{X}_i)} \mathbf{X}_{i+1}$

- Multi-layer end-to-end learning model



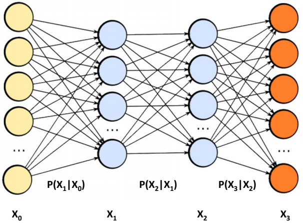
- Learn the best approximation $P_\theta(\mathbf{X}_{i+1}|\mathbf{X}_i) \approx P(\mathbf{X}_{i+1}|\mathbf{X}_i)$
- During SGD updates, hidden layer $P(\mathbf{X}_i)$ keeps shifting
- $\Longrightarrow$ target $P_{\theta*}(\mathbf{X}_{i+1}|\mathbf{X}_i)$ keeps shifting

MILA

Motivation 1/2– Covariate Shift

- Internal Covariate Shift (Ioffe and Szegedy, 2015)
  $$\mathbf{X}_i \xrightarrow{P(\mathbf{X}_{i+1}|\mathbf{X}_i)} \mathbf{X}_{i+1}$$

- Multi-layer end-to-end learning model



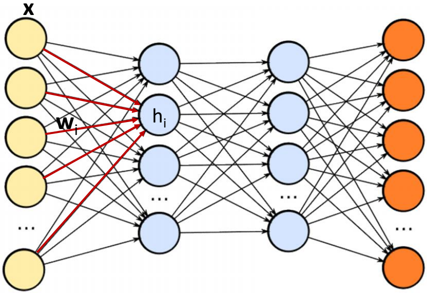
- Learn the best approximation $P_\theta(\mathbf{X}_{i+1}|\mathbf{X}_i) \approx P(\mathbf{X}_{i+1}|\mathbf{X}_i)$
- During SGD updates, hidden layer $P(\mathbf{X}_i)$ keeps shifting
- $\implies$ target $P_{\theta*}(\mathbf{X}_{i+1}|\mathbf{X}_i)$ keeps shifting
- Learning $P_\theta(\mathbf{X}_{i+1}|\mathbf{X}_i)$ using SGD is slow

Motivation 2/2– Optimization Perspective (Lecun, 1998)

What happens if input samples are not 0 mean?

- Let: $h_i = \sigma(a_i)$, where $a_i = \mathbf{w}_i^T \mathbf{x} + b_i$
- Consider the SGD weight update equation:

$$\begin{aligned}\mathbf{w}_i^{t+1} &= \mathbf{w}_i^t - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} \\ &= \mathbf{w}_i^t - \eta \frac{\partial \mathcal{L}}{\partial a_i} \frac{\partial a_i}{\partial \mathbf{w}_i} \\ &= \mathbf{w}_i^t - \eta \delta \mathbf{x}\end{aligned}$$

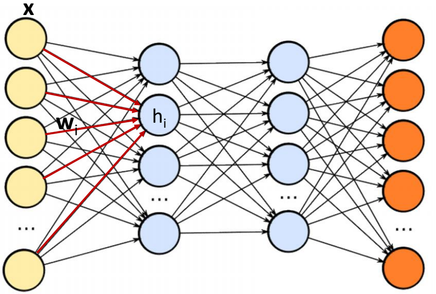


Motivation 2/2– Optimization Perspective (Lecun, 1998)

What happens if input samples are not 0 mean?

- Let: $h_i = \sigma(a_i)$, where $a_i = \mathbf{w}_i^T \mathbf{x} + b_i$
- Consider the SGD weight update equation:

$$\begin{aligned}\mathbf{w}_i^{t+1} &= \mathbf{w}_i^t - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} \\ &= \mathbf{w}_i^t - \eta \frac{\partial \mathcal{L}}{\partial a_i} \frac{\partial a_i}{\partial \mathbf{w}_i} \\ &= \mathbf{w}_i^t - \eta \delta \mathbf{x}\end{aligned}$$



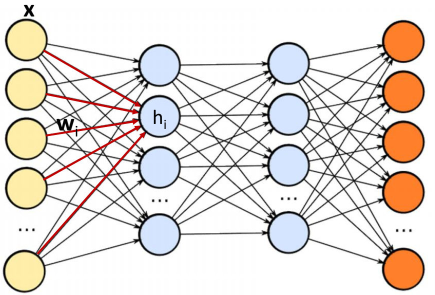
- If all $\mathbf{x}$ are positive $\implies$ all the weight units get updated with the same sign!

Motivation 2/2– Optimization Perspective (Lecun, 1998)

What happens if input samples are not 0 mean?

- Let: $h_i = \sigma(a_i)$, where $a_i = \mathbf{w}_i^T \mathbf{x} + b_i$
- Consider the SGD weight update equation:

$$\begin{aligned}\mathbf{w}_i^{t+1} &= \mathbf{w}_i^t - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} \\ &= \mathbf{w}_i^t - \eta \frac{\partial \mathcal{L}}{\partial a_i} \frac{\partial a_i}{\partial \mathbf{w}_i} \\ &= \mathbf{w}_i^t - \eta \delta \mathbf{x}\end{aligned}$$

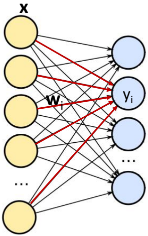


- If all **x** are positive $\implies$ all the weight units get updated with the same sign!
- Any bias in **x** introduces bias in weight updates– bad!
- Learning slows down– path to optimal $\mathbf{w}^*$ becomes longer

Motivation 2/2– Optimization Perspective (Lecun, 1998)

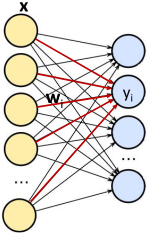
What happens if input samples are not 0 mean and unit variance?
(A more concrete special case)

- Consider loss:
  $\mathcal{L}(\mathbf{W}) = (1/2)\mathbb{E}_{\mathbf{x},\mathbf{y}}[\|\mathbf{y} - \mathbf{W}\mathbf{x}\|^2]$
- Update rule: $\mathbf{W}_i^{t+1} = \mathbf{W}_i^t - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_i}$



Motivation 2/2– Optimization Perspective (Lecun, 1998)

What happens if input samples are not 0 mean and unit variance?
(A more concrete special case)



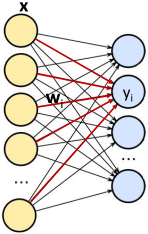
- Consider loss: $\mathcal{L}(\mathbf{W}) = (1/2)\mathbb{E}_{\mathbf{x},\mathbf{y}}[\|\mathbf{y} - \mathbf{Wx}\|^2]$
- Update rule: $\mathbf{W}_i^{t+1} = \mathbf{W}_i^t - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_i}$

  - Using Taylor's expansion around a minima $\mathbf{W}_i^*$: $\mathcal{L}(\mathbf{W}_i) = \mathcal{L}(\mathbf{W}_i^*) + (\mathbf{W}_i - \mathbf{W}_i^*)^T \frac{\partial^2 \mathcal{L}}{\partial \mathbf{w}_i^2} (\mathbf{W}_i - \mathbf{W}_i^*)$

MILA

Motivation 2/2– Optimization Perspective (Lecun, 1998)

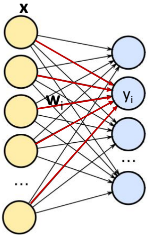
What happens if input samples are not 0 mean and unit variance?
(A more concrete special case)



- Consider loss: $\mathcal{L}(\mathbf{W}) = (1/2)\mathbb{E}_{\mathbf{x},\mathbf{y}}[\|\mathbf{y} - \mathbf{W}\mathbf{x}\|^2]$
- Update rule: $\mathbf{W}_i^{t+1} = \mathbf{W}_i^t - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_i}$
  - Using Taylor's expansion around a minima $\mathbf{W}_i^*$: $\mathcal{L}(\mathbf{W}_i) = \mathcal{L}(\mathbf{W}_i^*) + (\mathbf{W}_i - \mathbf{W}_i^*)^T \frac{\partial^2 \mathcal{L}}{\partial \mathbf{w}_i^2} (\mathbf{W}_i - \mathbf{W}_i^*)$
  - Taking derivative on both sides: $\frac{\partial \mathcal{L}(\mathbf{W}_i)}{\mathbf{W}_i} = (1/2) \frac{\partial^2 \mathcal{L}}{\partial \mathbf{w}_i^2} (\mathbf{W}_i - \mathbf{W}_i^*)$ $\left(\frac{\partial^2 \mathcal{L}}{\partial \mathbf{w}_i^2} = \mathbb{E}_{\mathbf{x}}[\mathbf{x}\mathbf{x}^T] \right)$

Motivation 2/2– Optimization Perspective (Lecun, 1998)

What happens if input samples are not 0 mean and unit variance?
(A more concrete special case)

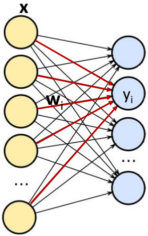


- Consider loss:
  $\mathcal{L}(\mathbf{W}) = (1/2)\mathbb{E}_{\mathbf{x},\mathbf{y}}[\|\mathbf{y} - \mathbf{W}\mathbf{x}\|^2]$
- Update rule: $\mathbf{W}_i^{t+1} = \mathbf{W}_i^t - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_i}$

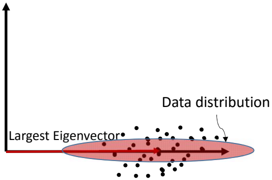
  - Using Taylor's expansion around a minima $\mathbf{W}_i^*$:
    $\mathcal{L}(\mathbf{W}_i) = \mathcal{L}(\mathbf{W}_i^*) + (\mathbf{W}_i - \mathbf{W}_i^*)^T \frac{\partial^2 \mathcal{L}}{\partial \mathbf{w}_i^2} (\mathbf{W}_i - \mathbf{W}_i^*)$
  - Taking derivative on both sides:
    $\frac{\partial \mathcal{L}(\mathbf{W}_i)}{\mathbf{W}_i} = (1/2) \frac{\partial^2 \mathcal{L}}{\partial \mathbf{w}_i^2} (\mathbf{W}_i - \mathbf{W}_i^*)$ $\left(\frac{\partial^2 \mathcal{L}}{\partial \mathbf{w}_i^2} = \mathbb{E}_{\mathbf{x}}[\mathbf{x}\mathbf{x}^T] \right)$
  - Thus update rule is:
    $\mathbf{W}_i^{t+1} = \mathbf{W}_i^t - \eta \mathbb{E}_{\mathbf{x}}[\mathbf{x}\mathbf{x}^T](\mathbf{W}_i^t - \mathbf{W}_i^*)$

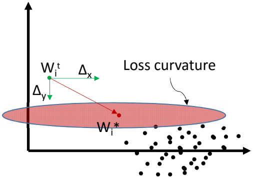
Motivation 2/2– Optimization Perspective (Lecun, 1998)

What happens if input samples are not 0 mean and unit variance? (A more concrete special case)



- Consider loss: $\mathcal{L}(\mathbf{W}) = (1/2)\mathbb{E}_{\mathbf{x},\mathbf{y}}[\|\mathbf{y} - \mathbf{W}\mathbf{x}\|^2]$
- Update rule: $\mathbf{W}_i^{t+1} = \mathbf{W}_i^t - \eta\mathbb{E}_{\mathbf{x}}[\mathbf{x}\mathbf{x}^T](\mathbf{W}_i^t - \mathbf{W}_i^*)$

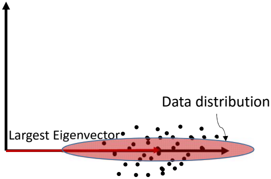


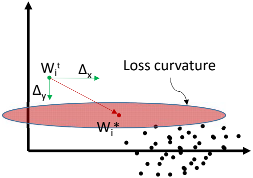


$$\mathbf{W}_i^{t+1} = \mathbf{W}_i^t - \eta \begin{pmatrix} \lambda_{\text{max}} & & \\ & \ddots & \\ & & \lambda_{\text{min}} \end{pmatrix} (\mathbf{W}_i^t - \mathbf{W}_i^*)$$

Motivation 2/2– Optimization Perspective (Lecun, 1998)

What happens if input samples are not 0 mean and unit variance? (A more concrete special case)





$$\mathbf{W}_i^{t+1} = \mathbf{W}_i^t - \eta \begin{pmatrix} \lambda_{\text{max}} & & \\ & \ddots & \\ & & \lambda_{\text{min}} \end{pmatrix} (\mathbf{W}_i^t - \mathbf{W}_i^*)$$

- Large mean and unequal variance $\implies \lambda_{\text{max}} >> \lambda_{\text{min}}$
- Each dimension of $\mathbf{W}_i^t$ needs a different learning rate
- Hence overall learning rate η is capped by $\frac{1}{\lambda_{\text{max}}}$

(λ_{max} is the largest eigenvalue of $\mathbb{E}_{\mathbf{x}}[\mathbf{x}\mathbf{x}^T]$)

Motivation

Take home message:

- At least have zero mean (avoid single large eigenvalue)
- Equal variance across dimensions is better (not very useful if mean $! = 0$)
- Uncorrelated (spherical) representations are best (but expensive!)

Motivation

Take home message:

- At least have zero mean (avoid single large eigenvalue)
- Equal variance across dimensions is better (not very useful if mean $! = 0$)
- Uncorrelated (spherical) representations are best (but expensive!)
- Maintain these properties at all hidden layers during training!

Table of Contents

1. Introduction
2. Motivation
3. Batch Normalization
4. Normalization Propagation
5. Weight Normalization
6. Layer Normalization
7. Conclusion

Batch Normalization (Ioffe and Szegedy, 2015)

- Avoid covariate shift
    - Normalize the entire distribution at every layer at every iteration!

Batch Normalization (Ioffe and Szegedy, 2015)

- Avoid covariate shift
  - Normalize the entire distribution at every layer at every iteration!
  - Simply infeasible!!!

Batch Normalization (Ioffe and Szegedy, 2015)

- Avoid covariate shift
    - Normalize the entire distribution at every layer at every iteration!
    - Simply infeasible!!!
- Alleviate it by normalizing the 1^{st} two orders of statistics?
    - Needs whitening over the entire dataset at every iteration!

Batch Normalization (Ioffe and Szegedy, 2015)

- Avoid covariate shift
  - Normalize the entire distribution at every layer at every iteration!
  - Simply infeasible!!!
- Alleviate it by normalizing the 1^{st} two orders of statistics?
  - Needs whitening over the entire dataset at every iteration!
  - Still very expensive!!!

Batch Normalization (Ioffe and Szegedy, 2015)

- Avoid covariate shift
  - Normalize the entire distribution at every layer at every iteration!
  - Simply infeasible!!!
- Alleviate it by normalizing the 1^{st} two orders of statistics?
  - Needs whitening over the entire dataset at every iteration!
  - Still very expensive!!!
- Key idea:
  - Compute statistics over mini-batch
  - Compute statistics unit-wise (notice mean is still optimal per batch)

Batch Normalization (Ioffe and Szegedy, 2015)

- Avoid covariate shift
  - Normalize the entire distribution at every layer at every iteration!
  - Simply infeasible!!!
- Alleviate it by normalizing the 1^{st} two orders of statistics?
  - Needs whitening over the entire dataset at every iteration!
  - Still very expensive!!!
- Key idea:
  - Compute statistics over mini-batch
  - Compute statistics unit-wise (notice mean is still optimal per batch)
    Finally we can do something about it :)

Batch Normalization (Ioffe and Szegedy, 2015)

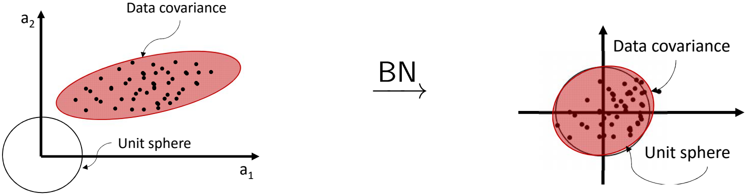
- Consider any hidden unit's pre-activation a_i
- Then normalized pre-activation under BN is given by:

$$\mathrm{BN}(a_i) = \frac{(\mathbf{a}_i - \mathbb{E}_{\mathcal{B}}[\mathbf{a}_i]))}{\sqrt{\mathrm{var}_{\mathcal{B}}(\mathbf{a}_i)}}$$

Batch Normalization (Ioffe and Szegedy, 2015)

- Consider any hidden unit's pre-activation a_i
- Then normalized pre-activation under BN is given by:

$$\mathrm{BN}(a_i) = \frac{(\mathbf{a}_i - \mathbb{E}_{\mathcal{B}}[\mathbf{a}_i]))}{\sqrt{\mathrm{var}_{\mathcal{B}}(\mathbf{a}_i)}}$$



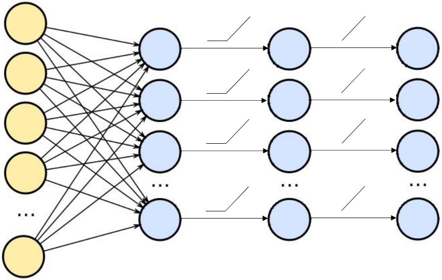
- Pretty close!

(approximation is worst when principal components are maximally away from axis)

Batch Normalization (Ioffe and Szegedy, 2015)

- A traditional vs. Batch Normalized (BN) ReLU layer:

Traditional: $\mathbf{x}$ $\quad a_i = \mathbf{W}_i^T\mathbf{x} + \beta_i$ $\quad \mathbf{h} = \mathrm{ReLU}(\mathbf{a})$ $\quad \mathbf{o} = \mathbf{h}$

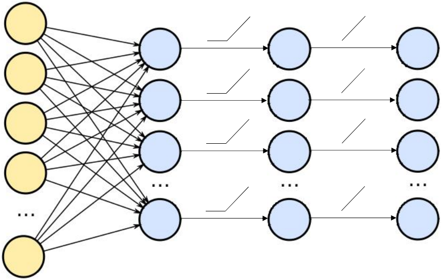


BN: $\mathbf{x}$ $\quad \hat{a}_i = \frac{\gamma_i(\mathbf{W}_i^T(\mathbf{x} - \mathbb{E}_{\mathcal{B}}[\mathbf{x}]))}{\sqrt{\mathrm{var}_{\mathcal{B}}(\mathbf{W}_i^T\mathbf{x})}} + \beta_i$ $\quad \mathbf{h} = \mathrm{ReLU}(\hat{\mathbf{a}})$ $\quad \mathbf{o} = \mathbf{h}$

Batch Normalization (Ioffe and Szegedy, 2015)

- A traditional vs. Batch Normalized (BN) ReLU layer:

Traditional: $\mathbf{x}$ $\quad a_i = \mathbf{W}_i^T\mathbf{x} + \beta_i$ $\quad \mathbf{h} = \mathrm{ReLU}(\mathbf{a})$ $\quad \mathbf{o} = \mathbf{h}$



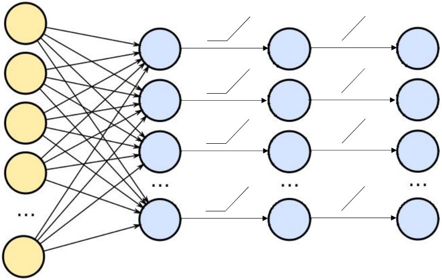
BN: $\mathbf{x}$ $\quad \hat{a}_i = \frac{\gamma_i(\mathbf{W}_i^T(\mathbf{x}-\mathbb{E}_{\mathcal{B}}[\mathbf{x}]))}{\sqrt{\mathrm{var}_{\mathcal{B}}(\mathbf{W}_i^T\mathbf{x})}} + \beta_i$ $\quad \mathbf{h} = \mathrm{ReLU}(\hat{\mathbf{a}})$ $\quad \mathbf{o} = \mathbf{h}$

- γ_i and β_i are learnable scale and bias parameters

Batch Normalization (Ioffe and Szegedy, 2015)

- A traditional vs. Batch Normalized (BN) ReLU layer:

Traditional: $\mathbf{x}$ $a_i = \mathbf{W}_i^T \mathbf{x} + \beta_i$ $\mathbf{h} = \mathrm{ReLU}(\mathbf{a})$ $\mathbf{o} = \mathbf{h}$



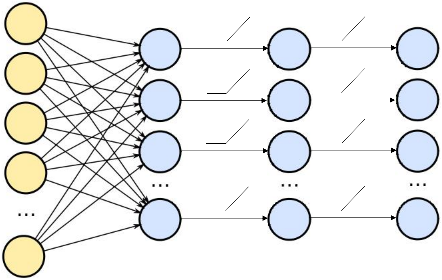
BN: $\mathbf{x}$ $\hat{a}_i = \frac{\gamma_i(\mathbf{W}_i^T(\mathbf{x} - \mathbb{E}_{\mathcal{B}}[\mathbf{x}]))}{\sqrt{\mathrm{var}_{\mathcal{B}}(\mathbf{W}_i^T \mathbf{x})}} + \beta_i$ $\mathbf{h} = \mathrm{ReLU}(\hat{\mathbf{a}})$ $\mathbf{o} = \mathbf{h}$

- γ_i and β_i are learnable scale and bias parameters
- Notice bias β_i is outside the normalization (why?)

Batch Normalization (Ioffe and Szegedy, 2015)

- A traditional vs. Batch Normalized (BN) ReLU layer:

Traditional: $\mathbf{x}$ $a_i = \mathbf{W}_i^T\mathbf{x} + \beta_i$ $\mathbf{h} = \text{ReLU}(\mathbf{a})$ $\mathbf{o} = \mathbf{h}$



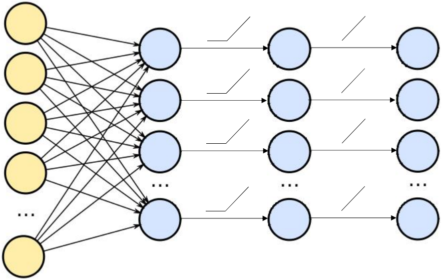
BN: $\mathbf{x}$ $\hat{a}_i = \frac{\gamma_i(\mathbf{W}_i^T(\mathbf{x}-\mathbb{E}_{\mathcal{B}}[\mathbf{x}]))}{\sqrt{\text{var}_{\mathcal{B}}(\mathbf{W}_i^T\mathbf{x})}} + \beta_i$ $\mathbf{h} = \text{ReLU}(\hat{\mathbf{a}})$ $\mathbf{o} = \mathbf{h}$

- γ_i and β_i are learnable scale and bias parameters
- Notice bias β_i is outside the normalization (why?)
- Why not apply BN post-activation ? (hint: Gaussianity)

Batch Normalization (Ioffe and Szegedy, 2015)

- A traditional vs. Batch Normalized (BN) ReLU layer:

Traditional: $\mathbf{x}$ $a_i = \mathbf{W}_i^T\mathbf{x} + \beta_i$ $\mathbf{h} = \text{ReLU}(\mathbf{a})$ $\mathbf{o} = \mathbf{h}$

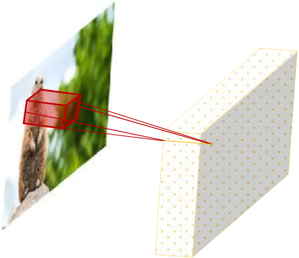


BN: $\mathbf{x}$ $\hat{a}_i = \frac{\gamma_i(\mathbf{W}_i^T(\mathbf{x}-\mathbb{E}_{\mathcal{B}}[\mathbf{x}]))}{\sqrt{\text{var}_{\mathcal{B}}(\mathbf{W}_i^T\mathbf{x})}} + \beta_i$ $\mathbf{h} = \text{ReLU}(\hat{\mathbf{a}})$ $\mathbf{o} = \mathbf{h}$

- γ_i and β_i are learnable scale and bias parameters
- Notice bias β_i is outside the normalization (why?)
- Why not apply BN post-activation ? (hint: Gaussianity)
- During test time– use running average estimates of mean and standard deviation

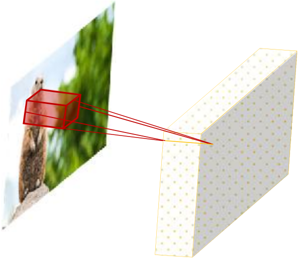
Batch Normalization (Ioffe and Szegedy, 2015)

- How to extend to convolutional layers?
  Simply treat each depth vector as a separate sample



Batch Normalization (Ioffe and Szegedy, 2015)

- How to extend to convolutional layers?
  Simply treat each depth vector as a separate sample



- Back-propagate through the normalization
  Otherwise normalizing representations changes prediction

Batch Normalization (Ioffe and Szegedy, 2015)

Effects of batch normalization:

- Parameter scaling:
  - Representations become scale invariant
    $\text{BN}(c\mathbf{Wx}) = \text{BN}(\mathbf{Wx})$
  - Gradients become inversely proportional to parameter scale
    $\frac{\partial \text{BN}(c\mathbf{Wx})}{\partial c\mathbf{W}} = (1/c)\frac{\partial \text{BN}(\mathbf{Wx})}{\partial \mathbf{W}}$
    Allows for large learning rates!

Batch Normalization (Ioffe and Szegedy, 2015)

Effects of batch normalization:

- Parameter scaling:
  - Representations become scale invariant
    $\mathrm{BN}(c\mathbf{Wx}) = \mathrm{BN}(\mathbf{Wx})$
  - Gradients become inversely proportional to parameter scale
    $\frac{\partial \mathrm{BN}(c\mathbf{Wx})}{\partial c\mathbf{W}} = (1/c)\frac{\partial \mathrm{BN}(\mathbf{Wx})}{\partial \mathbf{W}}$
    Allows for large learning rates!
- Regularization:
  - Each sample gets a different representation depending on mini-batch

Batch Normalization (Ioffe and Szegedy, 2015)

Effects of batch normalization:

- Parameter scaling:
  - Representations become scale invariant
    $\mathrm{BN}(c\mathbf{W}\mathbf{x}) = \mathrm{BN}(\mathbf{W}\mathbf{x})$
  - Gradients become inversely proportional to parameter scale
    $\frac{\partial \mathrm{BN}(c\mathbf{W}\mathbf{x})}{\partial c\mathbf{W}} = (1/c)\frac{\partial \mathrm{BN}(\mathbf{W}\mathbf{x})}{\partial \mathbf{W}}$
    Allows for large learning rates!
- Regularization:
  - Each sample gets a different representation depending on mini-batch

Awesome trick!
But also puzzling at the same time–
What if 2 different samples in two different batches get the same representation after BN?

Batch Normalization (Ioffe and Szegedy, 2015)

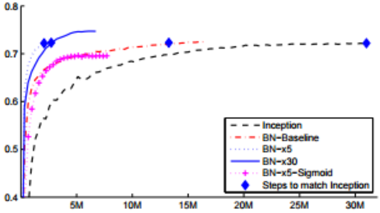
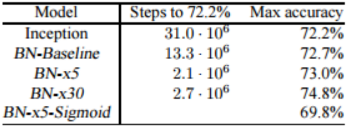


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*



Model	Steps to 72.2%	Max accuracy
Inception	$31.0 \cdot 10^6$	72.2%
BN-Baseline	$13.3 \cdot 10^6$	72.7%
BN-x5	$2.1 \cdot 10^6$	73.0%
BN-x30	$2.7 \cdot 10^6$	74.8%
BN-x5-Sigmoid		69.8%

Figure 3: *For Inception and the batch-normalized variants, the number of training steps required to reach the maximum accuracy of Inception (72.2%), and the maximum accuracy achieved by the network.*

Figures taken from Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." ICML (2015).

Table of Contents

1 Introduction

2 Motivation

3 Batch Normalization

4 Normalization Propagation

5 Weight Normalization

6 Layer Normalization

7 Conclusion

Normalization Propagation (Arpit et al, 2016)

Key idea:

- Instead of recomputing statistics at every layer, exploit normalization in data by propagating it to hidden layers

Coherence $= \max_{\mathbf{w}_i, \mathbf{w}_j, i \neq j} \frac{|\mathbf{w}_i^T \mathbf{w}_j|}{\|\mathbf{w}_i\|_2 \|\mathbf{w}_j\|_2}$

Incoherence $\implies$ pairwise angle between vectors is large

Normalization Propagation (Arpit et al, 2016)

Key idea:

- Instead of recomputing statistics at every layer, exploit normalization in data by propagating it to hidden layers
- Trick: If $\mathbf{x}$ has $\mathbf{0}$ mean $\mathbf{I}$ covariance, then $\mathbf{W}\mathbf{x}$ also has $\mathbf{0}$ mean $\approx \mathbf{I}$ covariance if $\|\mathbf{W}_i\|_2 = 1$ and $\mathbf{W}$ is incoherent

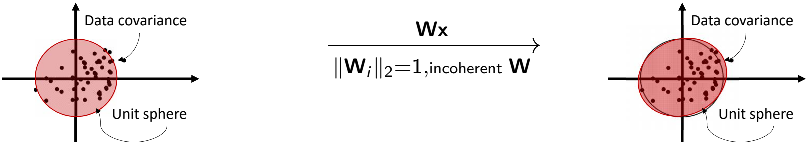
Coherence $= \max_{\mathbf{w}_i, \mathbf{w}_j, i \neq j} \frac{|\mathbf{w}_i^T \mathbf{w}_j|}{\|\mathbf{w}_i\|_2 \|\mathbf{w}_j\|_2}$

Incoherence $\implies$ pairwise angle between vectors is large

Normalization Propagation (Arpit et al, 2016)

Key idea:

- Instead of recomputing statistics at every layer, exploit normalization in data by propagating it to hidden layers
- Trick: If $\mathbf{x}$ has $\mathbf{0}$ mean $\mathbf{I}$ covariance, then $\mathbf{Wx}$ also has $\mathbf{0}$ mean $\approx \mathbf{I}$ covariance if $\|\mathbf{W}_i\|_2 = 1$ and $\mathbf{W}$ is incoherent



Data covariance
Unit sphere
$\xrightarrow[\|\mathbf{W}_i\|_2=1,\text{incoherent } \mathbf{W}]{\mathbf{Wx}}$
Data covariance
Unit sphere

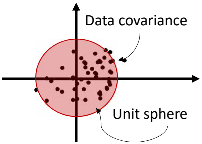
Coherence $= \max_{\mathbf{w}_i,\mathbf{w}_j, i\neq j} \frac{|\mathbf{w}_i^T \mathbf{w}_j|}{\|\mathbf{w}_i\|_2 \|\mathbf{w}_j\|_2}$

Incoherence $\implies$ pairwise angle between vectors is large

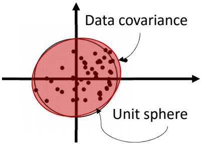
Normalization Propagation (Arpit et al, 2016)

Key idea:

- Instead of recomputing statistics at every layer, exploit normalization in data by propagating it to hidden layers
- Trick: If $\mathbf{x}$ has $\mathbf{0}$ mean $\mathbf{I}$ covariance, then $\mathbf{Wx}$ also has $\mathbf{0}$ mean $\approx \mathbf{I}$ covariance if $\|\mathbf{W}_i\|_2 = 1$ and $\mathbf{W}$ is incoherent



$$\xrightarrow[\|\mathbf{W}_i\|_2=1,\text{incoherent } \mathbf{W}]{\mathbf{Wx}}$$



- Assumption: pre-activations $\mathbf{Wx}$ are approximately Gaussian $\sigma(\mathbf{W}_i^T\mathbf{x})$ has fixed mean and variance

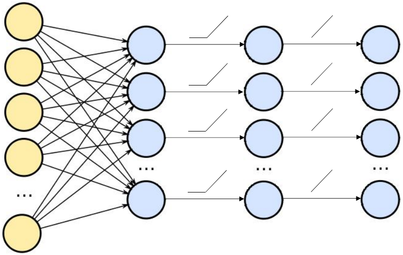
Coherence $= \max_{\mathbf{w}_i,\mathbf{w}_j,i\neq j} \frac{|\mathbf{w}_i^T\mathbf{w}_j|}{\|\mathbf{w}_i\|_2\|\mathbf{w}_j\|_2}$

Incoherence $\implies$ pairwise angle between vectors is large

Normalization Propagation (Arpit et al, 2016)

- A traditional vs. NormProp ReLU layer:

Traditional: $\mathbf{x}$ $\quad a_i = \mathbf{W}_i^T\mathbf{x} + \beta_i$ $\quad \mathbf{h} = \text{ReLU}(\mathbf{a})$ $\quad \mathbf{o} = \mathbf{h}$

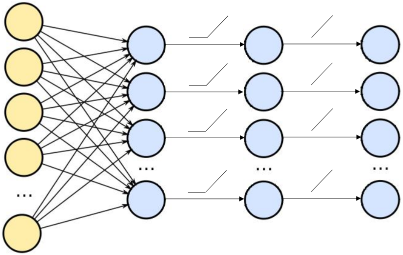


NormProp: $\mathbf{x}$ $\quad \hat{a}_i = \frac{\gamma_i(\mathbf{W}_i^T\mathbf{x})}{\|\mathbf{W}_i\|_2} + \beta_i$ $\quad \mathbf{h} = \text{ReLU}(\hat{\mathbf{a}})$ $\quad \mathbf{o} = \frac{\left[\mathbf{h} - \sqrt{\frac{1}{2\pi}}\right]}{\sqrt{\frac{1}{2}\left(1 - \frac{1}{\pi}\right)}}$

Normalization Propagation (Arpit et al, 2016)

- A traditional vs. NormProp ReLU layer:

Traditional: $\mathbf{x}$ $a_i = \mathbf{W}_i^T \mathbf{x} + \beta_i$ $\mathbf{h} = \text{ReLU}(\mathbf{a})$ $\mathbf{o} = \mathbf{h}$



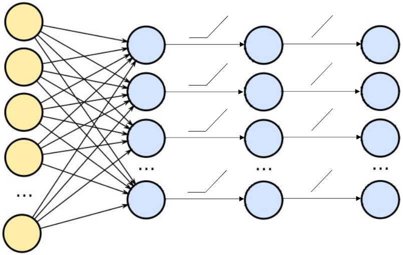
NormProp: $\mathbf{x}$ $\hat{a}_i = \frac{\gamma_i(\mathbf{W}_i^T \mathbf{x})}{\|\mathbf{W}_i\|_2} + \beta_i$ $\mathbf{h} = \text{ReLU}(\hat{\mathbf{a}})$ $\mathbf{o} = \frac{\left[\mathbf{h} - \sqrt{\frac{1}{2\pi}}\right]}{\sqrt{\frac{1}{2}\left(1 - \frac{1}{\pi}\right)}}$

- γ_i and β_i are learnable scale and bias parameters

Normalization Propagation (Arpit et al, 2016)

- A traditional vs. NormProp ReLU layer:

Traditional: $\mathbf{x}$ $\quad a_i = \mathbf{W}_i^T\mathbf{x} + \beta_i$ $\quad \mathbf{h} = \text{ReLU}(\mathbf{a})$ $\quad \mathbf{o} = \mathbf{h}$



NormProp: $\mathbf{x}$ $\quad \hat{a}_i = \frac{\gamma_i(\mathbf{W}_i^T\mathbf{x})}{\|\mathbf{W}_i\|_2} + \beta_i$ $\quad \mathbf{h} = \text{ReLU}(\hat{\mathbf{a}})$ $\quad \mathbf{o} = \frac{\left[\mathbf{h} - \sqrt{\frac{1}{2\pi}}\right]}{\sqrt{\frac{1}{2}\left(1 - \frac{1}{\pi}\right)}}$

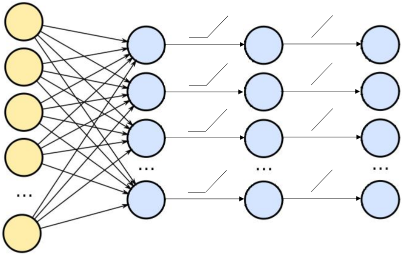
- γ_i and β_i are learnable scale and bias parameters
- Data can be normalized globally or batch-wise

Normalization Propagation (Arpit et al, 2016)

- A traditional vs. NormProp ReLU layer:

Traditional:

$\mathbf{x}$ $\quad a_i = \mathbf{W}_i^T\mathbf{x} + \beta_i$ $\quad \mathbf{h} = \text{ReLU}(\mathbf{a})$ $\quad \mathbf{o} = \mathbf{h}$



NormProp:

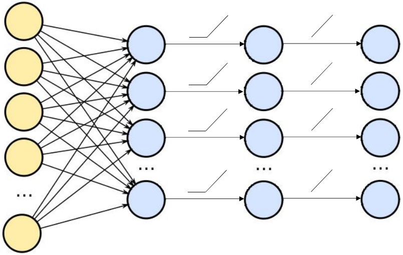
$\mathbf{x}$ $\quad \hat{a}_i = \frac{\gamma_i(\mathbf{W}_i^T\mathbf{x})}{\|\mathbf{W}_i\|_2} + \beta_i$ $\quad \mathbf{h} = \text{ReLU}(\hat{\mathbf{a}})$ $\quad \mathbf{o} = \frac{\left[\mathbf{h} - \sqrt{\frac{1}{2\pi}}\right]}{\sqrt{\frac{1}{2}\left(1 - \frac{1}{\pi}\right)}}$

- γ_i and β_i are learnable scale and bias parameters
- Data can be normalized globally or batch-wise
- train and test normalizations identical (for global data normalization)

Normalization Propagation (Arpit et al, 2016)

- A traditional vs. NormProp ReLU layer:

Traditional: $\mathbf{x}$ $\quad a_i = \mathbf{W}_i^T \mathbf{x} + \beta_i \quad$ $\mathbf{h} = \text{ReLU}(\mathbf{a}) \quad$ $\mathbf{o} = \mathbf{h}$



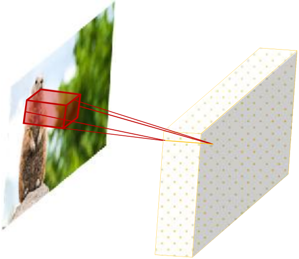
NormProp: $\mathbf{x}$ $\quad \hat{a}_i = \frac{\gamma_i (\mathbf{W}_i^T \mathbf{x})}{\|\mathbf{W}_i\|_2} + \beta_i \quad$ $\mathbf{h} = \text{ReLU}(\hat{\mathbf{a}}) \quad$ $\mathbf{o} = \frac{\left[\mathbf{h} - \sqrt{\frac{1}{2\pi}}\right]}{\sqrt{\frac{1}{2}\left(1 - \frac{1}{\pi}\right)}}$

- γ_i and β_i are learnable scale and bias parameters
- Data can be normalized globally or batch-wise
- train and test normalizations identical (for global data normalization)
- Applicable with batch-size 1

Normalization Propagation (Arpit et al, 2016)

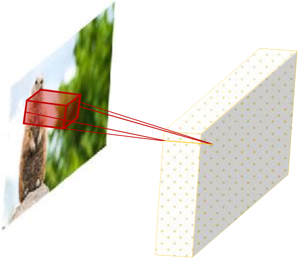
- How to extend to convolutional layers?

$$\hat{a}_i = \frac{\gamma_i(\mathbf{W}_i * \mathbf{x})}{\|\mathbf{W}_i\|_F} + \beta_i \qquad (i^{th} featuremap)$$



Normalization Propagation (Arpit et al, 2016)

- How to extend to convolutional layers?
  $\hat{a}_i = \frac{\gamma_i(\mathbf{W}_i * \mathbf{x})}{\|\mathbf{W}_i\|_F} + \beta_i \qquad (i^{th} featuremap)$



- Back-propagate through the normalization of weight scale
  Otherwise normalizing representations changes prediction

Normalization Propagation (Arpit et al, 2016)

Analysis:

- Singular values of $\mathbf{J} = \frac{\partial \mathbf{o}}{\partial \mathbf{x}} \approx 1$ prevents gradient problems

  (Saxe et al. 2014)

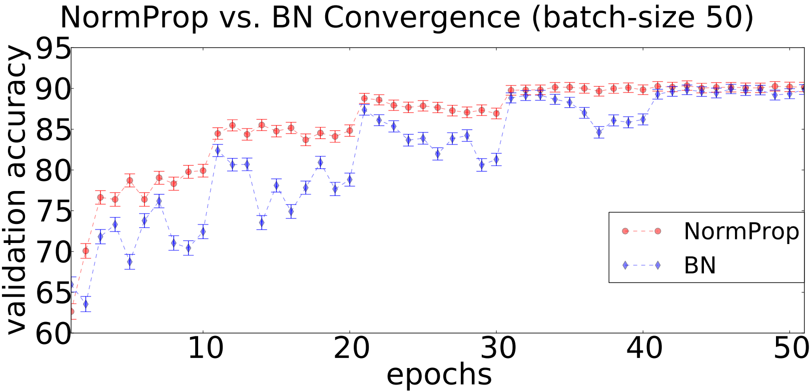
- For ReLU layer:
  - $\mathbb{E}_{\mathbf{x}}[\mathbf{J}\mathbf{J}^T] \approx 1.47\mathbf{I} \implies$ Singular values of $\mathbf{J} \approx 1.2$

Extension to other Activation functions (σ):

- $\mathbf{o}_i = \frac{1}{c_1}\left[\sigma\left(\frac{\gamma_i(\mathbf{W}_i * \mathbf{x})}{\|\mathbf{W}_i\|_F} + \beta_i\right) - c_2\right]$
- $c_1 = \sqrt{\text{var}(\sigma(Y))}$, $c_2 = \mathbb{E}[\sigma(Y)]$
- Y has Standard Normal distribution

Identical parameter scaling and regularization effects as BN

Normalization Propagation (Arpit et al, 2016)



Methods	Test Error (%)
CIFAR-10 with data augmentation	
NormProp	7.47
Batch Normalization	**7.25**
NIN + ALP units	7.51
NIN	8.81
DSN	7.97
Maxout	9.38

Table of Contents

1. Introduction
2. Motivation
3. Batch Normalization
4. Normalization Propagation
5. Weight Normalization
6. Layer Normalization
7. Conclusion

Weight Normalization (Salimans and Kingma, 2016)

Key idea:

- Decouple the scale and direction of weights

Weight Normalization (Salimans and Kingma, 2016)

Key idea:

- Decouple the scale and direction of weights
- Initialize parameters to have 0 mean unit variance pre-activations

Weight Normalization (Salimans and Kingma, 2016)

Key idea:

- Decouple the scale and direction of weights
- Initialize parameters to have 0 mean unit variance pre-activations
- Back-propagate through the normalization

Weight Normalization (Salimans and Kingma, 2016)

- Consider any hidden unit's pre-activation a_i
- Then weight normalized pre-activation is given by:

$$\text{WeighNorm}(a_i) = \frac{\gamma_i(\mathbf{W}_i^T \mathbf{x})}{\|\mathbf{W}_i\|_2} + \beta_i$$

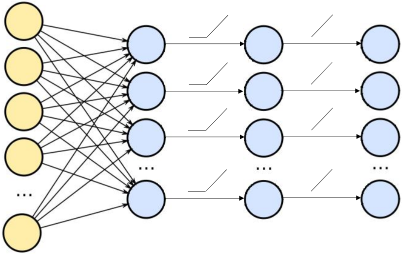
$$\text{Initialize: } \gamma_i = \frac{1}{\sqrt{\text{var}_{\mathcal{B}}(\mathbf{W}_i^T \mathbf{x})/\|\mathbf{W}_i\|_2}} \quad \beta_i = -\frac{\mathbb{E}_{\mathcal{B}}[\mathbf{W}_i^T \mathbf{x}/\|\mathbf{W}_i\|_2]}{\sqrt{\text{var}_{\mathcal{B}}(\mathbf{W}_i^T \mathbf{x})/\|\mathbf{W}_i\|_2}}$$

- Equivalently pre-activations get normalized using 1 mini-batch initially
- Optimization doesn't have explicit mean and variance normalization

Weight Normalization (Salimans and Kingma, 2016)

- A traditional vs. Weight Normalized ReLU layer:

Traditional: $\mathbf{x}$ $a_i = \mathbf{W}_i^T\mathbf{x} + \beta_i$ $\mathbf{h} = \mathrm{ReLU}(\mathbf{a})$ $\mathbf{o} = \mathbf{h}$

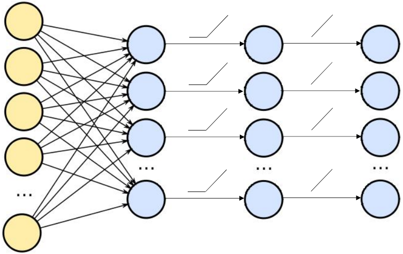


WeightNorm: $\mathbf{x}$ $\hat{a}_i = \frac{\gamma_i(\mathbf{W}_i^T\mathbf{x})}{\|\mathbf{W}_i\|_2} + \beta_i$ $\mathbf{h} = \mathrm{ReLU}(\hat{\mathbf{a}})$ $\mathbf{o} = \mathbf{h}$

Weight Normalization (Salimans and Kingma, 2016)

- A traditional vs. Weight Normalized ReLU layer:

Traditional: $\mathbf{x}$ $a_i = \mathbf{w}_i^T \mathbf{x} + \beta_i$ $\mathbf{h} = \text{ReLU}(\mathbf{a})$ $\mathbf{o} = \mathbf{h}$



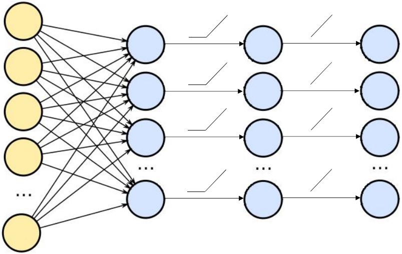
WeightNorm: $\mathbf{x}$ $\hat{a}_i = \frac{\gamma_i (\mathbf{w}_i^T \mathbf{x})}{\|\mathbf{w}_i\|_2} + \beta_i$ $\mathbf{h} = \text{ReLU}(\hat{\mathbf{a}})$ $\mathbf{o} = \mathbf{h}$

- γ_i and β_i are learnable scale and bias parameters

Weight Normalization (Salimans and Kingma, 2016)

- A traditional vs. Weight Normalized ReLU layer:

Traditional: $\mathbf{x}$ $a_i = \mathbf{W}_i^T \mathbf{x} + \beta_i$ $\mathbf{h} = \text{ReLU}(\mathbf{a})$ $\mathbf{o} = \mathbf{h}$



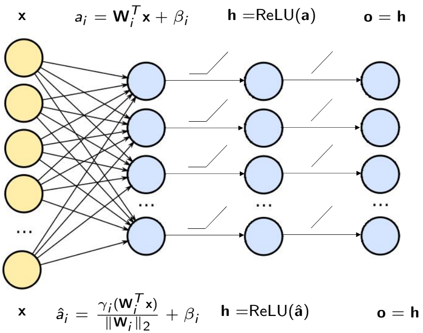
WeightNorm: $\mathbf{x}$ $\hat{a}_i = \frac{\gamma_i(\mathbf{W}_i^T \mathbf{x})}{\|\mathbf{W}_i\|_2} + \beta_i$ $\mathbf{h} = \text{ReLU}(\hat{\mathbf{a}})$ $\mathbf{o} = \mathbf{h}$

- γ_i and β_i are learnable scale and bias parameters
- train and test normalizations identical

Weight Normalization (Salimans and Kingma, 2016)

- A traditional vs. Weight Normalized ReLU layer:

Traditional:


$\mathbf{x}$
$a_i = \mathbf{W}_i^T \mathbf{x} + \beta_i$
$\mathbf{h} = \text{ReLU}(\mathbf{a})$
$\mathbf{o} = \mathbf{h}$
...
$\mathbf{x}$
$\hat{a}_i = \frac{\gamma_i (\mathbf{W}_i^T \mathbf{x})}{\|\mathbf{W}_i\|_2} + \beta_i$
$\mathbf{h} = \text{ReLU}(\hat{\mathbf{a}})$
$\mathbf{o} = \mathbf{h}$

WeightNorm:

- γ_i and β_i are learnable scale and bias parameters
- train and test normalizations identical
- Applicable with batch-size 1

Weight Normalization (Salimans and Kingma, 2016)

Effect of backpropagating through normalization:

- Let $\theta = \frac{\gamma(\mathbf{w})}{\|\mathbf{w}\|_2}$, then using SGD:

$$\begin{aligned}\Delta \mathbf{w} &= -\eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}} \\ &= -\eta \frac{\gamma}{\|\mathbf{w}^t\|_2} \left(\mathbf{I} - \frac{\theta^t \theta^{tT}}{\|\theta^t\|^2} \right) \frac{\partial \mathcal{L}}{\partial \theta}\end{aligned}$$

- Thus, $\mathbf{w} \perp \Delta \mathbf{w}$

Weight Normalization (Salimans and Kingma, 2016)

Effect of backpropagating through normalization:

- Let $\theta = \frac{\gamma(\mathbf{w})}{\|\mathbf{w}\|_2}$, then using SGD:

$$\begin{aligned}\Delta\mathbf{w} &= -\eta\frac{\partial\mathcal{L}}{\partial\mathbf{w}} \\ &= -\eta\frac{\gamma}{\|\mathbf{w}^t\|_2}\left(\mathbf{I} - \frac{\theta^t\theta^{tT}}{\|\theta^t\|^2}\right)\frac{\partial\mathcal{L}}{\partial\theta}\end{aligned}$$

- Thus, $\mathbf{w} \perp \Delta\mathbf{w}$
- Let $c = \|\Delta\mathbf{w}\|/\|\mathbf{w}\|$, notice: $\|\mathbf{w}^{t+1}\| = \sqrt{(1+c^2)}\|\mathbf{w}^t\|$

Weight Normalization (Salimans and Kingma, 2016)

Effect of backpropagating through normalization:

- Let $\theta = \frac{\gamma(\mathbf{w})}{\|\mathbf{w}\|_2}$, then using SGD:

$$\begin{aligned}\Delta\mathbf{w} &= -\eta\frac{\partial\mathcal{L}}{\partial\mathbf{w}}\\ &= -\eta\frac{\gamma}{\|\mathbf{w}^t\|_2}\left(\mathbf{I} - \frac{\theta^t\theta^{tT}}{\|\theta^t\|^2}\right)\frac{\partial\mathcal{L}}{\partial\theta}\end{aligned}$$

- Thus, $\mathbf{w} \perp \Delta\mathbf{w}$
- Let $c = \|\Delta\mathbf{w}\|/\|\mathbf{w}\|$, notice: $\|\mathbf{w}^{t+1}\| = \sqrt{(1+c^2)}\|\mathbf{w}^t\|$
- Large $c \implies$ large $\|\mathbf{w}^{t+1}\| \implies$ small gradient $\Delta\mathbf{w}$

Weight Normalization (Salimans and Kingma, 2016)

Effect of backpropagating through normalization:

- Let $\theta = \frac{\gamma(\mathbf{w})}{\|\mathbf{w}\|_2}$, then using SGD:

$$\begin{aligned}\Delta\mathbf{w} &= -\eta\frac{\partial\mathcal{L}}{\partial\mathbf{w}} \\ &= -\eta\frac{\gamma}{\|\mathbf{w}^t\|_2}\left(\mathbf{I} - \frac{\theta^t\theta^{tT}}{\|\theta^t\|^2}\right)\frac{\partial\mathcal{L}}{\partial\theta}\end{aligned}$$

- Thus, $\mathbf{w} \perp \Delta\mathbf{w}$
- Let $c = \|\Delta\mathbf{w}\|/\|\mathbf{w}\|$, notice: $\|\mathbf{w}^{t+1}\| = \sqrt{(1+c^2)}\|\mathbf{w}^t\|$
- Large $c \implies$ large $\|\mathbf{w}^{t+1}\| \implies$ small gradient $\Delta\mathbf{w}$
- Small $c \implies \|\mathbf{w}^{t+1}\| \approx \|\mathbf{w}^t\| \implies$ gradient scale $\Delta\mathbf{w}$ unchanged

Weight Normalization (Salimans and Kingma, 2016)

Effect of backpropagating through normalization:

- Let $\theta = \frac{\gamma(\mathbf{w})}{\|\mathbf{w}\|_2}$, then using SGD:

$$\begin{aligned}\Delta\mathbf{w} &= -\eta\frac{\partial\mathcal{L}}{\partial\mathbf{w}} \\ &= -\eta\frac{\gamma}{\|\mathbf{w}^t\|_2}\left(\mathbf{I} - \frac{\theta^t\theta^{tT}}{\|\theta^t\|^2}\right)\frac{\partial\mathcal{L}}{\partial\theta}\end{aligned}$$

- Thus, $\mathbf{w} \perp \Delta\mathbf{w}$
- Let $c = \|\Delta\mathbf{w}\|/\|\mathbf{w}\|$, notice: $\|\mathbf{w}^{t+1}\| = \sqrt{(1+c^2)}\|\mathbf{w}^t\|$
- Large $c \implies$ large $\|\mathbf{w}^{t+1}\| \implies$ small gradient $\Delta\mathbf{w}$
- Small $c \implies \|\mathbf{w}^{t+1}\| \approx \|\mathbf{w}^t\| \implies$ gradient scale $\Delta\mathbf{w}$ unchanged

  The above analysis applies to NormProp as well!

Weight Normalization (Salimans and Kingma, 2016)

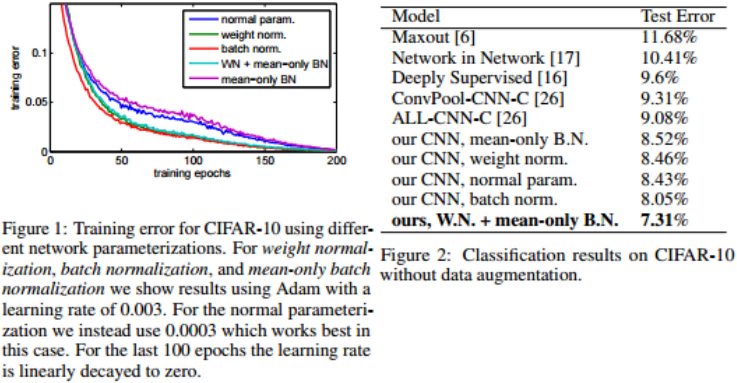


Figure 1: Training error for CIFAR-10 using different network parameterizations. For *weight normalization*, *batch normalization*, and *mean-only batch normalization* we show results using Adam with a learning rate of 0.003. For the normal parameterization we instead use 0.0003 which works best in this case. For the last 100 epochs the learning rate is linearly decayed to zero.

Model	Test Error
Maxout [6]	11.68%
Network in Network [17]	10.41%
Deeply Supervised [16]	9.6%
ConvPool-CNN-C [26]	9.31%
ALL-CNN-C [26]	9.08%
our CNN, mean-only B.N.	8.52%
our CNN, weight norm.	8.46%
our CNN, normal param.	8.43%
our CNN, batch norm.	8.05%
ours, W.N. + mean-only B.N.	**7.31%**

Figure 2: Classification results on CIFAR-10 without data augmentation.

Figures taken from Salimans, Tim, and Diederik P. Kingma. ”Weight normalization: A simple reparameterization to accelerate training of deep neural networks.” Advances in Neural Information Processing Systems. 2016.

Table of Contents

1. Introduction
2. Motivation
3. Batch Normalization
4. Normalization Propagation
5. Weight Normalization
6. Layer Normalization
7. Conclusion

Layer Normalization (Ba et al, 2016)

Key idea:

- Compute statistics for each sample separately

Layer Normalization (Ba et al, 2016)

Key idea:

- Compute statistics for each sample separately
- Trick: Compute mean and standard deviation across units instead of mini-batch

Layer Normalization (Ba et al, 2016)

Key idea:

- Compute statistics for each sample separately
- Trick: Compute mean and standard deviation across units instead of mini-batch
- Aimed towards application to recurrent neural nets:
  - RNNs have variable number of temporal layers
  - There can be more number of layers at test time (BN is not applicable)

Layer Normalization (Ba et al, 2016)

- Consider a temporal hidden layer's pre-activation $\mathbf{a}^t = \mathbf{W}_{hh}\mathbf{h}^{t-1} + \mathbf{W}_{xh}\mathbf{x}^t$
- Then layer normalized pre-activation $\mathbf{a}^t$ is given by:

$$\mathrm{LN}(\mathbf{a}^t) = \frac{\gamma}{\sigma^t} \odot (\mathbf{a}_t - \mu^t) + \beta$$

$$\mu^t = \frac{1}{H}\sum_{i=1}^{H} a_i^t \quad \sigma^t = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^t - \mu^t)^2}$$

Layer Normalization (Ba et al, 2016)

$$\mathrm{LN}(\mathbf{a}^t) = \frac{\gamma}{\sigma^t} \odot (\mathbf{a}_t - \mu^t) + \beta$$

$$\mu^t = \frac{1}{H}\sum_{i=1}^{H} a_i^t \quad \sigma^t = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^t - \mu^t)^2}$$

Effects of layer normalization:

- Invariance to weight scaling and translation
- Data rescaling and translation
- Norm of weight controls effective learning rate

MILA

Layer Normalization (Ba et al, 2016)

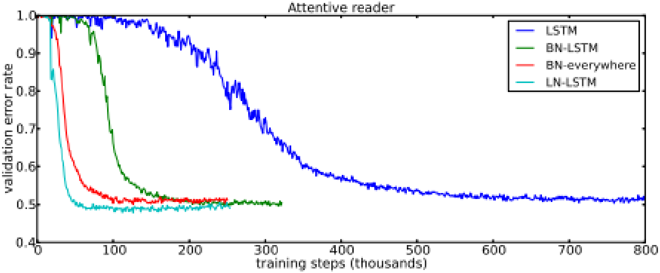


Figure 2: Validation curves for the attentive reader model. BN results are taken from [Cooijmans et al., 2016].

Figures taken from Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization." arXiv preprint arXiv:1607.06450 (2016).

Table of Contents

1. Introduction
2. Motivation
3. Batch Normalization
4. Normalization Propagation
5. Weight Normalization
6. Layer Normalization
7. Conclusion

Conclusion

- Removing internal covariate shift in DNNs leads to faster convergence
- Generally, even good initialization plays an important role

  (Mishkin and Matas, 2016)
- Making representations (scale, shift) invariant seems to boost convergence
- SGD + better optimization = better generalization?
  – what about bad local minima?

  (Zhang et al, 2016)