Hyper-parameters/Tweaking

Yufeng Ma, Chris Dusold

Virginia Tech

November 17, 2015
Overview

1. Batch Normalization
   - Internal Covariate Shift
   - Mini-Batch Normalization
   - Key Points in Batch Normalization
   - Experiments and Results

2. Importance of Initialization and Momentum
   - Overview of first-order method
   - Momentum & Nesterov’s Accelerated Gradient (NAG)
   - Deep Autoencoders & RNN - Echo-State Networks
Challenges to be solved

Reference paper:
Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

When we are faced with training a Deep Network with saturating nonlinearities:
- Lower/smaller learning rates
- Initialize the weights from Gaussian Distributions

Figure credit: www.regentsprep.org

Yufeng Ma, Chris Dusold (Virginia Tech)  Hyper-parameters/Tweaking  November 17, 2015  3 / 40
Challenges to be solved

Reference paper: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
Challenges to be solved

Reference paper: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

When we are faced with training a Deep Network with saturating nonlinearities:

- **Lower/smaller** learning rates
- Initialize the weights from **Gaussian Distributions**
Challenges to be solved

**Reference paper:** Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

When we are faced with training a Deep Network with saturating nonlinearities:

- **Lower/smaller** learning rates
- Initialize the weights from **Gaussian Distributions**

![Normal Curve](figure.png)

figure credit: www.regentsprep.org
Challenges to be solved

Reasons behind the problem:
- Parameters change during training
- Input distributions of each layer changes
Challenges to be solved

Reasons behind the problem:

- Parameters change during training
- Input distributions of each layer changes
Challenges to be solved

Reasons behind the problem:
- Parameters change during training
- Input distributions of each layer changes

Sigmoid’s output distribution before and after parameter updates
Overview

1. **Batch Normalization**
   - Internal Covariate Shift
   - Mini-Batch Normalization
   - Key Points in Batch Normalization
   - Experiments and Results

2. **Importance of Initialization and Momentum**
   - Overview of first-order method
   - Momentum & Nesterov's Accelerated Gradient (NAG)
   - Deep Autoencoders & RNN - Echo-State Networks
Internal Covariate Shift

Covariate Shift
Change of input distributions to a Learning System

Extension to part or sub-networks

\[ \ell = F_2(F_1(u, \Theta_1), \Theta_2) \]

\[ \ell = F_2(x, \Theta_2) \]
where \( x = F_1(u, \Theta_1) \)

\[ \Theta_2 \leftarrow \Theta_2 - \alpha \sum_{i=1}^{m} \frac{\partial F_2(x_i, \Theta_2)}{\partial \Theta_2} \]

In terms of change in the distribution of \( x \), \( \Theta_2 \) will not need to readjust much.
Internal Covariate Shift

Covariate Shift

Change of input distributions to a Learning System

\[
\ell = F_2(F_1(u, \Theta_1), \Theta_2)
\]

\[
\ell = F_2(x, \Theta_2)
\]

where \( x = F_1(u, \Theta_1) \)

\[ \Theta_2 \leftarrow \Theta_2 - \alpha \sum_{i=1}^{m} \frac{\partial F_2(x_i, \Theta_2)}{\partial \Theta_2} \]

In terms of change in the distribution of \( x \), \( \Theta_2 \) will not need to readjust much.
Internal Covariate Shift

Covariate Shift

Change of input distributions to a Learning System

Extension to part or sub-networks

\[ \ell = F_2(F_1(u, \Theta_1), \Theta_2) \]
Internal Covariate Shift

Covariate Shift
Change of input distributions to a Learning System

Extension to part or sub-networks

\[ \ell = F_2(F_1(u, \Theta_1), \Theta_2) \]

\[ \ell = F_2(x, \Theta_2), \text{ where } x = F_1(u, \Theta_1) \]

\[ \Theta_2 \leftarrow \Theta_2 - \frac{\alpha}{m} \sum_{i=1}^{m} \frac{\partial F_2(x_i, \Theta_2)}{\partial \Theta_2} \]
Internal Covariate Shift

Covariate Shift
Change of input distributions to a Learning System

Extension to part or sub-networks

\[ \ell = F_2(F_1(u, \Theta_1), \Theta_2) \]

\[ \ell = F_2(x, \Theta_2), \text{ where } x = F_1(u, \Theta_1) \]

\[ \Theta_2 \leftarrow \Theta_2 - \frac{\alpha}{m} \sum_{i=1}^{m} \frac{\partial F_2(x_i, \Theta_2)}{\partial \Theta_2} \]

In terms of change in the distribution of \( x \), \( \Theta_2 \) will not need to readjust much.

Internal Covariate Shift
Change in the distributions of internal nodes of a deep network
Reducing Internal Covariate Shift

Whitening-LeCun et al., 1998b; Wiesler&Ney, 2011

The network training converges faster if its inputs are whitened—i.e., linearly transformed to have zero means and unit variances, and decorrelated.

Goal:
Whitening the inputs of each layer to have fixed distributions in order to Reduce the ill effects of Internal Covariate Shift.
Reducing Internal Covariate Shift

Whitening-LeCun et al., 1998b; Wiesler&Ney, 2011

The network training converges faster if its inputs are whitened-i.e., linearly transformed to have zero means and unit variances, and decorrelated.
Reducing Internal Covariate Shift

Whitening-LeCun et al., 1998b; Wiesler & Ney, 2011

The network training converges faster if its inputs are whitened—i.e., linearly transformed to have zero means and unit variances, and decorrelated.

**Goal:** Whitening the inputs of each layer to have fixed distributions in order to Reduce the ill effects of Internal Covariate Shift.
Reducing Internal Covariate Shift

Interspersal lead to reduced gradient descent

\[ b' \leftarrow b + \Delta b, \text{ where } \Delta b \propto -\frac{\partial \ell}{\partial \hat{x}} \]

\[ \hat{x} = x - E[x] = u + (b + \Delta b) - E[u + b] = u + b - E[u + b] \]

Normalizations are NOT taken into account in Gradient Descent Optimization.
Interspersal lead to reduced gradient descent

\[ b \leftarrow b + \Delta b, \quad \text{where} \quad \Delta b \propto -\frac{\partial \ell}{\partial \hat{x}} \frac{\partial \hat{x}}{\partial b} \]

\[ \hat{x} = x - E[x] = u + (b + \Delta b) - E[u + (b + \Delta b)] = u + b - E[u + b] \]
Reducing Internal Covariate Shift

- Interspersal lead to reduced gradient descent

\[ b \leftarrow b + \Delta b, \quad \text{where} \quad \Delta b \propto -\frac{\partial \ell}{\partial \hat{x}} \frac{\partial \hat{x}}{\partial b} \]

\[ \hat{x} = x - E[x] = u + (b + \Delta b) - E[u + (b + \Delta b)] = u + b - E[u + b] \]

- Normalizations are NOT taken into account in Gradient Descent Optimization.
Reducing Internal Covariate Shift

Introducing Normalization

\[ \hat{x} = \text{Norm}(x, X) \]

and Jacobians in backpropagation

\[ \frac{\partial \text{Norm}(x, X)}{\partial x} \] and \[ \frac{\partial \text{Norm}(x, X)}{\partial X} \]

New challenges: expensive to compute covariance matrix and its inverse.

Covariance matrix

\[ \text{Cov}[x] = \mathbb{E}_{x \in X}[xx^T] - \mathbb{E}[x] \mathbb{E}[x]^T \]

Whitening

\[ \text{Cov}[x] - \frac{1}{2}(x - \mathbb{E}[x]) \frac{1}{2}(x - \mathbb{E}[x])^T \]

Yufeng Ma, Chris Dusold (Virginia Tech)
Introducing Normalization

\[ \hat{x} = \text{Norm}(x, \mathcal{X}) \]

and Jacobians in backpropagation

\[ \frac{\partial \text{Norm}(x, \mathcal{X})}{\partial x} \quad \text{and} \quad \frac{\partial \text{Norm}(x, \mathcal{X})}{\partial \mathcal{X}} \]
Reducing Internal Covariate Shift

Introducing Normalization

\[ \hat{x} = \text{Norm}(x, \mathcal{X}) \]

and Jacobians in backpropagation

\[ \frac{\partial \text{Norm}(x, \mathcal{X})}{\partial x} \text{ and } \frac{\partial \text{Norm}(x, \mathcal{X})}{\partial \mathcal{X}} \]

**New challenges:** expensive to compute covariance matrix and its inverse square root.

**Covariance matrix**

\[ \text{Cov}[x] = \mathbb{E}_{x \in \mathcal{X}}[xx^T] - \mathbb{E}[x] \mathbb{E}[x]^T \]

**Whitening**

\[ \text{Cov}[x]^{-1/2}(x - \mathbb{E}[x]) \]
Overview

1. Batch Normalization
   - Internal Covariate Shift
   - Mini-Batch Normalization
   - Key Points in Batch Normalization
   - Experiments and Results

2. Importance of Initialization and Momentum
   - Overview of first-order method
   - Momentum & Nesterov’s Accelerated Gradient (NAG)
   - Deep Autoencoders & RNN – Echo-State Networks
Mini-Batch Normalization

Two simplifications and Identity Transform

Normalize each scalar feature independently

Use mini-batch to estimate the mean and variance instead of whole population

Ensure Identity Transform can be represented

\[ y(k) = \gamma(k) \hat{x}(k) + \beta(k) \]

Two new parameters for each activation are introduced for learning.

Batch Normalization Transform, see reference paper for details

Yufeng Ma, Chris Dusold (Virginia Tech)

Hyper-parameters/Tweaking

November 17, 2015 11 / 40
Two simplifications and Identity Transform

- Normalize each scalar feature independently
- Use mini-batch to estimate the mean and variance instead of whole population
- Ensure Identity Transform can be represented

$$y(k) = \gamma(k) \hat{x}(k) + \beta(k)$$

Two new parameters for each activation are introduced for learning.

Batch Normalization Transform, see reference paper for details
Two simplifications and Identity Transform

- Normalize each scalar feature independently
- Use mini-batch to estimate the mean and variance instead of whole population
- Ensure Identity Transform can be represented

\[ y^{(k)} = \gamma^{(k)} \hat{x}^{(k)} + \beta^{(k)} \]

Two new parameters for each activation are introduced for learning.
Two simplifications and Identity Transform

- Normalize each scalar feature independently
- Use mini-batch to estimate the mean and variance instead of whole population
- Ensure Identity Transform can be represented

\[ y^{(k)} = \gamma^{(k)} \hat{x}^{(k)} + \beta^{(k)} \]

Two new parameters for each activation are introduced for learning.

**Batch Normalization Transform**, see reference paper for details
Overview

1. **Batch Normalization**
   - Internal Covariate Shift
   - Mini-Batch Normalization
   - Key Points in Batch Normalization
   - Experiments and Results

2. **Importance of Initialization and Momentum**
   - Overview of first-order method
   - Momentum & Nesterov’s Accelerated Gradient (NAG)
   - Deep Autoencoders & RNN - Echo-State Networks
Key Points in Batch Normalization

Original parameters and newly introduced $\gamma$ and $\beta$ will be trained. When in inference, the whole population of training data is used for mean and variance statistics instead of the estimate.

$E(x) \leftarrow E_B[\mu_B]$

$\text{Var}(x) \leftarrow m - 1 E_B[\sigma^2_B]$

In Convolutional layers, different locations of a feature map should be normalized in the same way.

$m' = |B| = m \cdot pq$, and $\gamma(k), \beta(k)$ per feature map.
Key Points in Batch Normalization

- Original parameters and newly introduced $\gamma$ and $\beta$ will be trained.
Key Points in Batch Normalization

- Original parameters and newly introduced $\gamma$ and $\beta$ will be trained.
- When in inference, the **whole population** of training data is used for mean and variance statistics instead of the estimate.

$$E(x) \leftarrow E_B[\mu_B]$$

$$Var[x] \leftarrow \frac{m}{m-1}E_B[\sigma^2_B]$$
Key Points in Batch Normalization

- Original parameters and newly introduced $\gamma$ and $\beta$ will be trained.
- When in inference, the whole population of training data is used for mean and variance statistics instead of the estimate.

\[ E(x) \leftarrow E_B[\mu_B] \]
\[ \text{Var}[x] \leftarrow \frac{m}{m-1} E_B[\sigma_B^2] \]

- In Convolutional layers, different locations of a feature map should be normalized in the same way.

\[ m' = |B| = m \cdot pq, \text{ and } \gamma^{(k)}, \beta^{(k)} \text{ per feature map} \]
Key Points in Batch Normalization

Batch Normalization will regularize the model with less overfitting.
Key Points in Batch Normalization

- Higher learning rates are allowed

Batch Normalization will regularize the model with less overfitting.

Yufeng Ma, Chris Dusold (Virginia Tech)
Key Points in Batch Normalization

- Higher learning rates are allowed

\[ BN(Wu) = BN((aW)u) \]
Key Points in Batch Normalization

- Higher learning rates are allowed

\[ \frac{\partial BN(Wu)}{\partial u} = \frac{\partial BN((aW)u)}{\partial u}, \quad \frac{\partial BN(Wu)}{\partial aW} = \frac{1}{a} \cdot \frac{\partial BN((aW)u)}{\partial W} \]
Key Points in Batch Normalization

- Higher learning rates are allowed

\[ \frac{\partial BN(Wu)}{\partial u} = \frac{\partial BN((aW)u)}{\partial u} \quad \frac{\partial BN(Wu)}{\partial aW} = \frac{1}{a} \frac{\partial BN((aW)u)}{\partial W} \]

- Batch Normalization will regularize the model with less overfitting.
Overview

1. **Batch Normalization**
   - Internal Covariate Shift
   - Mini-Batch Normalization
   - Key Points in Batch Normalization
   - Experiments and Results

2. **Importance of Initialization and Momentum**
   - Overview of first-order method
   - Momentum & Nesterov’s Accelerated Gradient (NAG)
   - Deep Autoencoders & RNN - Echo-State Networks
Activations over time

Batch Normalization helps train faster and achieve higher accuracy.

Figure credit: reference paper

Yufeng Ma, Chris Dusold (Virginia Tech)
Activations over time

Batch Normalization helps train faster and achieve higher accuracy.
Activations over time

Batch Normalization helps train faster and achieve higher accuracy.

figure credit: reference paper
Activations over time

Batch Normalization makes input distribution more stable.

figure credit: reference paper

Yufeng Ma, Chris Dusold (Virginia Tech)
Batch Normalization makes input distribution more stable.
Batch Normalization makes input distribution more stable.

(b) Without BN  (c) With BN

figure credit: reference paper
Accelerating Batch Normalization Networks

Tricks to follow

- Increasing learning rate
- Remove or Reduce Dropout
- Reduce $\ell_2$ weight regularization
- Accelerate the learning rate decay
- Remove Local Response Normalization
- Shuffle training examples more thoroughly
- Reduce the photometric distortions

Yufeng Ma, Chris Dusold (Virginia Tech)
Tricks to follow

- Increasing learning rate
- Remove or Reduce Dropout
- Reduce $\ell_2$ weight regularization
- Accelerate the learning rate decay
- Remove Local Response Normalization
- Shuffle training examples more thoroughly
- Reduce the photometric distortions
Accelerating Batch Normalization Networks

Tricks to follow

- Increasing learning rate
- Remove or Reduce Dropout
- Reduce $\ell_2$ weight regularization
- Accelerate the learning rate decay
- Remove Local Response Normalization
- Shuffle training examples more thoroughly
- Reduce the photometric distortions
Network Comparisons

- Inception, BN-Baseline, BN-x5, BN-x30, BN-x5-Sigmoid

Figure credit: reference paper

Yufeng Ma, Chris Dusold (Virginia Tech)
Network Comparisons

Inception, BN-Baseline, BN-x5, BN-x30, BN-x5-Sigmoid
Network Comparisons

Inception, BN-Baseline, BN-x5, BN-x30, BN-x5-Sigmoid
Network Comparisons

Inception, BN-Baseline, BN-x5, BN-x30, BN-x5-Sigmoid

<table>
<thead>
<tr>
<th>Model</th>
<th>Steps to 72.2%</th>
<th>Max accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>$3.1 \cdot 10^6$</td>
<td>72.2%</td>
</tr>
<tr>
<td>BN-Baseline</td>
<td>$1.3 \cdot 10^6$</td>
<td>72.7%</td>
</tr>
<tr>
<td>BN-x5</td>
<td>$2.1 \cdot 10^6$</td>
<td>73.0%</td>
</tr>
<tr>
<td>BN-x30</td>
<td>$2.7 \cdot 10^6$</td>
<td>74.8%</td>
</tr>
<tr>
<td>BN-x5-Sigmoid</td>
<td></td>
<td>69.8%</td>
</tr>
</tbody>
</table>

figure credit: reference paper
Ensemble Classification

Top-5 validation error of 4.9% and test error of 4.82%, exceeds the estimated accuracy of human raters.

Figure credit: reference paper

Yufeng Ma, Chris Dusold (Virginia Tech)
Top-5 validation error of 4.9% and test error of 4.82%, exceeds the estimated accuracy of human raters.
Ensemble Classification

Top-5 validation error of 4.9% and test error of 4.82%, exceeds the estimated accuracy of human raters.

<table>
<thead>
<tr>
<th>Model</th>
<th>Resolution</th>
<th>Crops</th>
<th>Models</th>
<th>Top-1 error</th>
<th>Top-5 error</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoogLeNet ensemble</td>
<td>224</td>
<td>144</td>
<td>7</td>
<td>-</td>
<td>6.67%</td>
</tr>
<tr>
<td>Deep Image low-res</td>
<td>256</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>7.96%</td>
</tr>
<tr>
<td>Deep Image high-res</td>
<td>512</td>
<td>-</td>
<td>1</td>
<td>24.88</td>
<td>7.42%</td>
</tr>
<tr>
<td>Deep Image ensemble</td>
<td>variable</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5.98%</td>
</tr>
<tr>
<td>BN-Inception single crop</td>
<td>224</td>
<td>1</td>
<td>1</td>
<td>25.2%</td>
<td>7.82%</td>
</tr>
<tr>
<td>BN-Inception multicrop</td>
<td>224</td>
<td>144</td>
<td>1</td>
<td>21.99%</td>
<td>5.82%</td>
</tr>
<tr>
<td>BN-Inception ensemble</td>
<td>224</td>
<td>144</td>
<td>6</td>
<td>20.1%</td>
<td><strong>4.9%</strong></td>
</tr>
</tbody>
</table>

figure credit: reference paper
Overview

1. **Batch Normalization**
   - Internal Covariate Shift
   - Mini-Batch Normalization
   - Key Points in Batch Normalization
   - Experiments and Results

2. **Importance of Initialization and Momentum**
   - Overview of first-order method
   - Momentum & Nesterov’s Accelerated Gradient (NAG)
   - Deep Autoencoders & RNN - Echo-State Networks
Challenges to be solved

Reference paper: On the importance of initialization and momentum in deep learning

Difficult to use first-order method to reach performance previously only achievable by second-order method like Hessian-Free.

Well-designed random initialization

Slowly increasing schedule for momentum parameter

No need for sophisticated second-order methods.
Challenges to be solved

**Reference paper**: On the importance of initialization and momentum in deep learning
Challenges to be solved

Reference paper: On the importance of initialization and momentum in deep learning

- Difficult to use first-order method to reach performance previously only achievable by second-order method like Hessian-Free.
Challenges to be solved

**Reference paper:** On the importance of initialization and momentum in deep learning

- Difficult to use first-order method to reach performance previously only achievable by second-order method like Hessian-Free.

- Well-designed random initialization

- Slowly increasing schedule for momentum parameter
Challenges to be solved

Reference paper: On the importance of initialization and momentum in deep learning

- Difficult to use first-order method to reach performance previously only achievable by second-order method like Hessian-Free.
- Well-designed random initialization
- Slowly increasing schedule for momentum parameter
- No need for sophisticated second-order methods.
Overview of first-order method

First-order Methods

- Vanilla Stochastic Gradient Descent
- SGD + Momentum
- Nesterov’s Accelerated Gradient (NAG)
- AdaGrad
- Adam
- Rprop
- RMSProp
- AdaDelta

slide credit: Ishan Misra
Overview

1. Batch Normalization
   - Internal Covariate Shift
   - Mini-Batch Normalization
   - Key Points in Batch Normalization
   - Experiments and Results

2. Importance of Initialization and Momentum
   - Overview of first-order method
   - Momentum & Nesterov’s Accelerated Gradient (NAG)
   - Deep Autoencoders & RNN - Echo-State Networks
Several First-order Methods

- **θ**: Parameters of network
- **f**: Objective function
- **ϵ**: Learning rate
- **\(\nabla f\)**: Gradient of \(f\)
- **v**: Velocity vector
- **µ**: Momentum coefficient

**Vanilla SGD**

\[
v_{t+1} = \epsilon \nabla f(\theta_t)
\]

\[
\theta_{t+1} = \theta_t - v_{t+1}
\]
Notation:
- $\theta$ - Parameters of network, $f$ - Objective function, $\epsilon$ - Learning rate
- $\nabla f$ - Gradient of $f$, $v$ - Velocity vector, $\mu$ - Momentum coefficient
Several First-order Methods

Notation:

- $\theta$ - Parameters of network, $f$ - Objective function, $\epsilon$ - Learning rate
- $\nabla f$ - Gradient of $f$, $v$ - Velocity vector, $\mu$ - Momentum coefficient

Vanilla SGD

$$v_{t+1} = \epsilon \nabla f(\theta_t)$$
$$\theta_{t+1} = \theta_t - v_{t+1}$$

slide credit: Ishan Misra
Several First-order Methods

Rprop Update

\[
\begin{align*}
\text{if } \nabla f_t \cdot \nabla f_{t-1} > 0 & \quad v_t = \eta + v_{t-1} \\
\text{else if } \nabla f_t \cdot \nabla f_{t-1} < 0 & \quad v_t = \eta - v_{t-1} \\
\text{else} & \quad v_t = v_{t-1}
\end{align*}
\]

\[
\theta_{t+1} = \theta_t - v_t
\]

where \(0 < \eta < 1\)

slide credit: Ishan Misra

Yufeng Ma, Chris Dusold (Virginia Tech)

Hyper-parameters/Tweaking

November 17, 2015

26 / 40
Several First-order Methods

Rprop Update

\[
\text{if } \nabla f_t \nabla f_{t-1} > 0 \\
\quad \nu_t = \eta^+ \nu_{t-1} \\
\text{else if } \nabla f_t \nabla f_{t-1} < 0 \\
\quad \nu_t = \eta^- \nu_{t-1} \\
\text{else} \\
\quad \nu_t = \nu_{t-1}
\]

\[\theta_{t+1} = \theta_t - \nu_t\]

where \(0 < \eta^- < 1 < \eta^+\)

slide credit: Ishan Misra
Several First-order Methods

\[ r_t = \theta_{t-1} + \frac{\alpha}{\sqrt{r_t}} \nabla f(\theta_t) \]

\[ \theta_{t+1} = \theta_t - v_{t+1} \]

RMSProp = Rprop + SGD

\[ r_t = (1 - \gamma) \theta_{t-1} + \gamma r_{t-1} \]

\[ v_{t+1} = \alpha \sqrt{r_t} \nabla f(\theta_t) \]

\[ \theta_{t+1} = \theta_t - v_{t+1} \]
Several First-order Methods

AdaGrad

\[ r_t = \theta_t^2 + r_{t-1} \]
\[ v_{t+1} = \frac{\alpha}{\sqrt{r_t}} \nabla f(\theta_t) \]
\[ \theta_{t+1} = \theta_t - v_{t+1} \]
Several First-order Methods

**AdaGrad**

\[ r_t = \theta_t^2 + r_{t-1} \]

\[ v_{t+1} = \frac{\alpha}{\sqrt{r_t}} \nabla f(\theta_t) \]

\[ \theta_{t+1} = \theta_t - v_{t+1} \]

**RMSProp = Rprop + SGD**

\[ r_t = (1 - \gamma)\theta_t^2 + \gamma r_{t-1} \]

\[ v_{t+1} = \frac{\alpha}{\sqrt{r_t}} \nabla f(\theta_t) \]

\[ \theta_{t+1} = \theta_t - v_{t+1} \]

slide credit: Ishan Misra
Several First-order Methods

AdaDelta

$$v_{t+1} = H_{t-1} \nabla f,$$

$$\propto f' f'' \propto \frac{1}{\text{units of } \theta} \left( \frac{1}{\text{units of } \theta} \right)^2 \propto \text{units of } \theta$$

Adam

$$r_t = (1 - \gamma_1) \nabla f(\theta_t) + \gamma_1 r_{t-1}$$

$$p_t = (1 - \gamma_2) \nabla f(\theta_t) + \gamma_2 p_{t-1}$$

$$\hat{r}_t = r_t \left(1 - (1 - \gamma_1) t\right)$$

$$\hat{p}_t = p_t \left(1 - (1 - \gamma_2) t\right)$$

$$v_t = \alpha \hat{r}_t \sqrt{\hat{p}_t}$$

$$\theta_{t+1} = \theta_t - v_t$$

---

slide credit: Ishan Misra
Several First-order Methods

AdaDelta

\[ v_{t+1} = H^{-1} \nabla f, \]

\[ \propto \frac{f'}{f''} \]

\[ \propto \frac{1/\text{units of } \theta}{(1/\text{units of } \theta)^2} \]

\[ \propto \text{units of } \theta \]

slide credit: Ishan Misra

Yufeng Ma, Chris Dusold (Virginia Tech)
**Several First-order Methods**

### AdaDelta

\[ v_{t+1} = H^{-1} \nabla f, \]
\[ \alpha \frac{f'}{f''} \]
\[ \alpha \frac{1/\text{units of } \theta}{(1/\text{units of } \theta)^2} \]
\[ \alpha \text{ units of } \theta \]

### Adam

\[ r_t = (1 - \gamma_1) \nabla f(\theta_t) + \gamma_1 r_{t-1} \]
\[ p_t = (1 - \gamma_2) \nabla f(\theta_t)^2 + \gamma_2 p_{t-1} \]
\[ \hat{r}_t = \frac{r_t}{(1-(1-\gamma_1)^t)} \]
\[ \hat{p}_t = \frac{p_t}{(1-(1-\gamma_2)^t)} \]
\[ v_t = \alpha \frac{\hat{r}_t}{\sqrt{\hat{p}_t}} \]
\[ \theta_{t+1} = \theta_t - v_t \]

*slide credit: Ishan Misra*
Overview

1. **Batch Normalization**
   - Internal Covariate Shift
   - Mini-Batch Normalization
   - Key Points in Batch Normalization
   - Experiments and Results

2. **Importance of Initialization and Momentum**
   - Overview of first-order method
   - **Momentum & Nesterov’s Accelerated Gradient (NAG)**
   - Deep Autoencoders & RNN - Echo-State Networks
Momentum and NAG

Notation:

θ - Parameters of network,
f - Objective function,
ϵ - Learning rate

\( \nabla f \) - Gradient of f,
v - Velocity vector,
µ - Momentum coefficient

Classical Momentum

\[ v_{t+1} = \mu v_t - \epsilon \nabla f(\theta_t) \]
\[ \theta_{t+1} = \theta_t + v_{t+1} \]

Nesterov's Accelerated Gradient

\[ v_{t+1} = \mu v_t - \epsilon \nabla f(\theta_t + \mu v_t) \]
\[ \theta_{t+1} = \theta_t + v_{t+1} \]
Momentum and NAG

Notation:
- $\theta$ - Parameters of network, $f$ - Objective function, $\epsilon$ - Learning rate
- $\nabla f$ - Gradient of $f$, $v$ - Velocity vector, $\mu$ - Momentum coefficient
Momentum and NAG

Notation:
- $\theta$ - Parameters of network, $f$ - Objective function, $\epsilon$ - Learning rate
- $\nabla f$ - Gradient of $f$, $v$ - Velocity vector, $\mu$ - Momentum coefficient

Classical Momentum

\[
\begin{align*}
\nu_{t+1} &= \mu \nu_t - \epsilon \nabla f(\theta_t) \\
\theta_{t+1} &= \theta_t + \nu_{t+1}
\end{align*}
\]
Momentum and NAG

Notation:
- $\theta$ - Parameters of network, $f$ - Objective function, $\epsilon$ - Learning rate
- $\nabla f$ - Gradient of $f$, $v$ - Velocity vector, $\mu$ - Momentum coefficient

Classical Momentum

$$v_{t+1} = \mu v_t - \epsilon \nabla f(\theta_t)$$
$$\theta_{t+1} = \theta_t + v_{t+1}$$

Nesterov’s Accelerated Gradient

$$v_{t+1} = \mu v_t - \epsilon \nabla f(\theta_t + \mu v_t)$$
$$\theta_{t+1} = \theta_t + v_{t+1}$$
Relationship between CM and NAG

NAG uses \[ \theta_t + \mu \nu_t \] but MISSING the yet unknown correction. Thus when the addition of \( \mu \nu_t \) results in an immediate undesirable increase in the objective \( f \).

Figure credit: reference paper

Yufeng Ma, Chris Dusold (Virginia Tech)
NAG uses $\theta_t + \mu \nu_t$ but **MISSING** the yet unknown correction. Thus when the addition of $\mu \nu_t$ results in an **immediate undesirable increase** in the objective $f$, 
NAG uses $\theta_t + \mu \nu_t$ but **MISSING** the yet unknown correction. Thus when the addition of $\mu \nu_t$ results in an **immediate undesirable increase** in the objective $f$, 

*Figure 1. (Top) Classical Momentum (Bottom) Nesterov Accelerated Gradient*
NAG uses $\theta_t + \mu \nu_t$ but **MISSING** the yet unknown correction. Thus when the addition of $\mu \nu_t$ results in an **immediate undesirable increase** in the objective $f$,
Relationship between CM and NAG

Apply CM and NAG to a positive definite quadratic objective

\[ q(x) = x^T Ax / 2 + b^T x. \]

Difference in effective momentum coefficient

Classical Momentum: \( \mu \)

NAG: \( \mu (1 - \lambda \epsilon) \), where \( \lambda \) is the eigenvalue of \( A \).

\( \epsilon \) small, CM and NAG are equivalent

\( \epsilon \) large, NAG gives smaller \( \mu (1 - \lambda \epsilon) \) to stop oscillations.
Apply CM and NAG to a positive definite quadratic objective

\[ q(x) = \frac{x^T Ax}{2} + b^T x. \]

**Difference in effective momentum coefficient**

- Classical Momentum: \( \mu \)
- NAG: \( \mu(1 - \lambda \epsilon) \), where \( \lambda \) is the eigenvalue of \( A \).
Apply CM and NAG to a positive definite quadratic objective
\[ q(x) = \frac{x^T A x}{2} + b^T x. \]

**Difference in effective momentum coefficient**

- Classical Momentum: \( \mu \)
- NAG: \( \mu (1 - \lambda \epsilon) \), where \( \lambda \) is the eigenvalue of \( A \).

- \( \epsilon \) small, CM and NAG are equivalent
- \( \epsilon \) large, NAG gives smaller \( \mu (1 - \lambda_i \epsilon) \) to stop oscillations.
Overview

1. Batch Normalization
   - Internal Covariate Shift
   - Mini-Batch Normalization
   - Key Points in Batch Normalization
   - Experiments and Results

2. Importance of Initialization and Momentum
   - Overview of first-order method
   - Momentum & Nesterov’s Accelerated Gradient (NAG)
   - Deep Autoencoders & RNN - Echo-State Networks
Structure of Deep Autoencoder
Deep Autoencoders

Structure of Deep Autoencoder

figure credit: http://deeplearning4j.org/deepautoencoder.html
Deep Autoencoders

Sparse Initialization - each random unit connected to 15 randomly chosen units in the previous layer, drawn from a unit Gaussian.

Schedule for Momentum Coefficient: \( \mu_t = \min(1 - \frac{2}{t+1}, \mu_{\text{max}}) \)

\( \mu_t = 1 - \frac{3}{t+5} \), not strongly convex - Nesterov (1983)

constant \( \mu_t \), strongly convex - Nesterov (2003)

Table credit: reference paper

Yufeng Ma, Chris Dusold (Virginia Tech)
Deep Autoencoders

- **Sparse Initialization**—each random unit connected to 15 randomly chosen units in the previous layer, drawn from a unit Gaussian

- **Schedule for Momentum Coefficient**

  \[ \mu_t = \min(1 - 2^{-1 - \log_2(\lfloor t/250 \rfloor + 1)}, \mu_{\max}) \]
Deep Autoencoders

- **Sparse Initialization** - each random unit connected to 15 randomly chosen units in the previous layer, drawn from a unit Gaussian.

- **Schedule for Momentum Coefficient**

  \[ \mu_t = \min\left(1 - 2^{-1 - \log_2(\lfloor t/250 \rfloor + 1)}, \mu_{max}\right) \]

- \[ \mu_t = 1 - 3/(t + 5), \text{ not strongly convex } - \text{ Nesterov(1983)} \]

- constant \( \mu_t \), strongly convex - Nesterov(2003)
Deep Autoencoders

- **Sparse Initialization** - each random unit connected to 15 randomly chosen units in the previous layer, drawn from a unit Gaussian

- **Schedule for Momentum Coefficient**

  \[ \mu_t = \min(1 - 2^{-1 - \log_2(\lfloor t/250 \rfloor + 1)}, \mu_{max}) \]

- \( \mu_t = 1 - 3/(t + 5) \), not strongly convex - Nesterov(1983)

- constant \( \mu_t \), strongly convex - Nesterov(2003)

<table>
<thead>
<tr>
<th>task</th>
<th>0_{(SGD)}</th>
<th>0.9N</th>
<th>0.99N</th>
<th>0.995N</th>
<th>0.999N</th>
<th>0.9M</th>
<th>0.99M</th>
<th>0.995M</th>
<th>0.999M</th>
<th>SGD_C</th>
<th>HF^†</th>
<th>HF^*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curves</td>
<td>0.48</td>
<td>0.16</td>
<td>0.096</td>
<td>0.091</td>
<td>0.074</td>
<td>0.15</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
<td>0.16</td>
<td>0.058</td>
<td>0.11</td>
</tr>
<tr>
<td>Mnist</td>
<td>2.1</td>
<td>1.0</td>
<td>0.73</td>
<td>0.75</td>
<td>0.80</td>
<td>1.0</td>
<td>0.77</td>
<td>0.84</td>
<td>0.90</td>
<td>0.9</td>
<td>0.69</td>
<td>1.40</td>
</tr>
<tr>
<td>Faces</td>
<td>36.4</td>
<td>14.2</td>
<td>8.5</td>
<td>7.8</td>
<td>7.7</td>
<td>15.3</td>
<td>8.7</td>
<td>8.3</td>
<td>9.3</td>
<td>NA</td>
<td>7.5</td>
<td>12.0</td>
</tr>
</tbody>
</table>

table credit: reference paper
RNN - Echo-State Networks

(RNNs are a family)

Hidden-to-output connections learned from data
Recurrence connections fixed to a random draw from a specific distribution

Yufeng Ma, Chris Dusold (Virginia Tech)
Echo-State Networks (a family RNNs)
Echo-State Networks (a family RNNs)

- Hidden-to-output connections learned from data
- Recurrent connections fixed to a random draw from a specific distribution

figure credit: Mantas Lukoevicius
RNN - Echo-State Networks

ESN-based Initialization

- Spectral radius of hidden-to-hidden matrix around 1.1
- Initial scale of Input-to-hidden connections plays an important role (Gaussian draw with a standard deviation of 0.001 achieves good balance, but is Task Dependent)

Schedule of Momentum coefficient $\mu$

- $\mu = 0.9$ for the first 1000 parameters;
- $\mu = \mu_0 \in \{0, 0.9, 0.98, 0.995\}$ afterwards;

Table credit: reference paper

Yufeng Ma, Chris Dusold (Virginia Tech)

November 17, 2015 37 / 40
ESN-based Initialization

- Spectral Radius of Hidden-to-hidden matrix around 1.1
- Initial scale of Input-to-hidden connections plays an important role (Gaussian draw with a standard deviation of 0.001 achieves good balance, but is Task Dependent)

Schedule of Momentum coefficient \( \mu \):

\[
\mu = 0.9 \text{ for the first 1000 parameters; } \\
\mu = \begin{cases} 
0, & 0 \\
0.9, & 0.98, & 0.995 \end{cases} \text{ afterwards; }
\]

Table credit: reference paper

Yufeng Ma, Chris Dusold (Virginia Tech)
ESN-based Initialization

- Spectral Radius of Hidden-to-hidden matrix around 1.1
- Initial scale of Input-to-hidden connections plays an important role (Gaussian draw with a standard deviation of 0.001 achieves good balance, but is Task Dependent)

Schedule of Momentum coefficient $\mu$

- $\mu = 0.9$ for the first 1000 parameters;
- $\mu = \mu_0 \in \{0, 0.9, 0.98, 0.995\}$ afterwards;
ESN-based Initialization

- Spectral Radius of Hidden-to-hidden matrix around 1.1
- Initial scale of Input-to-hidden connections plays an important role (Gaussian draw with a standard deviation of 0.001 achieves good balance, but is Task Dependent)

Schedule of Momentum coefficient $\mu$

- $\mu = 0.9$ for the first 1000 parameters;
- $\mu = \mu_0 \in \{0, 0.9, 0.98, 0.995\}$ afterwards;

Table:

<table>
<thead>
<tr>
<th>problem</th>
<th>biases</th>
<th>0</th>
<th>0.9N</th>
<th>0.98N</th>
<th>0.995N</th>
<th>0.9M</th>
<th>0.98M</th>
<th>0.995M</th>
</tr>
</thead>
<tbody>
<tr>
<td>add $T = 80$</td>
<td>0.82</td>
<td>0.39</td>
<td>0.02</td>
<td>0.21</td>
<td>0.00025</td>
<td>0.43</td>
<td>0.62</td>
<td>0.036</td>
</tr>
<tr>
<td>mul $T = 80$</td>
<td>0.84</td>
<td>0.48</td>
<td>0.36</td>
<td>0.22</td>
<td>0.0013</td>
<td>0.029</td>
<td>0.025</td>
<td>0.37</td>
</tr>
<tr>
<td>mem-5 $T = 200$</td>
<td>2.5</td>
<td>1.27</td>
<td>1.02</td>
<td>0.96</td>
<td>0.63</td>
<td>1.12</td>
<td>1.09</td>
<td>0.92</td>
</tr>
<tr>
<td>mem-20 $T = 80$</td>
<td>8.0</td>
<td>5.37</td>
<td>2.77</td>
<td>0.0144</td>
<td>0.00005</td>
<td>1.75</td>
<td>0.0017</td>
<td>0.053</td>
</tr>
</tbody>
</table>

Table credit: reference paper
Overview

1. Batch Normalization
   - Internal Covariate Shift
   - Mini-Batch Normalization
   - Key Points in Batch Normalization
   - Experiments and Results

2. Importance of Initialization and Momentum
   - Overview of first-order method
   - Momentum & Nesterov’s Accelerated Gradient (NAG)
   - Deep Autoencoders & RNN - Echo-State Networks
Questions?
Chris Dusold’s Part

- Variance-SGD (No More Pesky Learning Rates)
- Adam (Adam: A Method for Stochastic Optimization)
- AdaGrad (Adaptive Subgradient Methods for Online Learning and Stochastic Optimization)