CPEN 455: Deep Learning

Lecture 4: Backpropagation II

Renjie Liao

University of British Columbia

Winter, Term 2, 2024

Outline

- Learning Algorithm for Feedforward Neural Networks:
  - Backpropagation
  - Weight Initialization
  - Learning Rate & Momentum & Adam
  - Weight Decay & Early Stopping

Outline

- Learning Algorithm for Feedforward Neural Networks:
  - Backpropagation
  - Weight Initialization
  - **Learning Rate & Momentum & Adam**
  - Weight Decay & Early Stopping

Backpropagation (BP)

Now we know how to compute gradients, let us see how to perform gradient-based learning (e.g., SGD)

Algorithm 1 SGD

Input: step T, learning Rate η, initial W^0, batch size B
For $t = 1, \cdots, T$
    Get mini-batch data (X^t, Y^t)
    $L(W^{t-1}, X^t, Y^t) = \frac{1}{B} \sum_{i=1}^{B} \ell(f(W^{t-1}, X^t[i]), Y^t[i])$
    $W^t = W^{t-1} - \eta \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$
Return W^T

Backpropagation (BP)

Now we know how to compute gradients, let us see how to perform gradient-based learning (e.g., SGD)

Algorithm 1 SGD

1: **Input**: step T, learning Rate η, initial W^0, batch size B
2: **For** $t = 1, \cdots, T$
3: Get mini-batch data (X^t, Y^t)
4: $L(W^{t-1}, X^t, Y^t) = \frac{1}{B}\sum_{i=1}^{B} \ell(f(W^{t-1}, X^t[i]), Y^t[i])$
5: $W^t = W^{t-1} - \eta \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$
6: **Return** W^T

Forward pass

Backpropagation (BP)

Now we know how to compute gradients, let us see how to perform gradient-based learning (e.g., SGD)

Algorithm 1 SGD

1: **Input**: step T, learning Rate η, initial W^0, batch size B
2: **For** $t = 1, \cdots, T$
3: Get mini-batch data (X^t, Y^t)
4: $L(W^{t-1}, X^t, Y^t) = \frac{1}{B}\sum_{i=1}^{B} \ell(f(W^{t-1}, X^t[i]), Y^t[i])$ — Forward pass: $f(W^{t-1}, X^t[i])$
5: $W^t = W^{t-1} - \eta \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$ — Backward pass: $\frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$
6: **Return** W^T

Backpropagation (BP)

SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another [1, 2].

Backpropagation (BP)

SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another [1, 2].

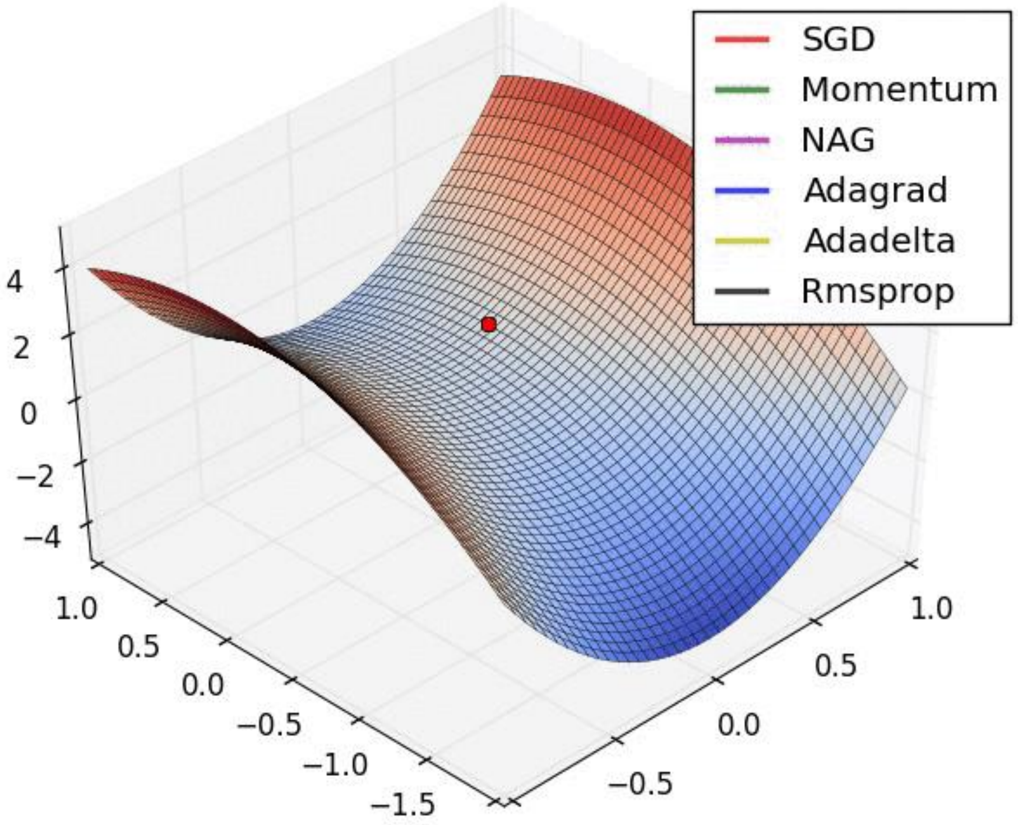


Image Credit: [1]

BP + Momentum

SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another [1, 2].

[3]: Gradient descent is a man walking down a hill. He follows the steepest path downwards; his progress is slow, but steady. Momentum is a heavy ball rolling down the same hill. The added inertia acts both as a smoother and an accelerator, dampening oscillations and causing us to barrel through narrow valleys, small humps and local minima.

Image Credit: [1]

BP + Momentum

SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another [1, 2].

[3]: Gradient descent is a man walking down a hill. He follows the steepest path downwards; his progress is slow, but steady. Momentum is a heavy ball rolling down the same hill. The added inertia acts both as a smoother and an accelerator, dampening oscillations and causing us to barrel through narrow valleys, small humps and local minima.

Algorithm 1 SGD with Momentum

1: **Input**: step T, learning Rate η, initial W^0, batch size B, momentum β, initial M^0
2: **For** $t = 1, \cdots, T$
3: Get mini-batch data (X^t, Y^t)
4: $L(W^{t-1}, X^t, Y^t) = \frac{1}{B}\sum_{i=1}^{B} \ell(f(W^{t-1}, X^t[i]), Y^t[i])$
5: $M^t = \beta M^{t-1} + \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$
6: $W^t = W^{t-1} - \eta M^t$
7: **Return** W^T

BP + Momentum

SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another [1, 2].

[3]: Gradient descent is a man walking down a hill. He follows the steepest path downwards; his progress is slow, but steady. Momentum is a heavy ball rolling down the same hill. The added inertia acts both as a smoother and an accelerator, dampening oscillations and causing us to barrel through narrow valleys, small humps and local minima.

Algorithm 1 SGD with Momentum

1: **Input**: step T, learning Rate η, initial W^0, batch size B, momentum β, initial M^0
2: **For** $t = 1, \cdots, T$
3: Get mini-batch data (X^t, Y^t)
4: $L(W^{t-1}, X^t, Y^t) = \frac{1}{B}\sum_{i=1}^{B} \ell(f(W^{t-1}, X^t[i]), Y^t[i])$
5: $M^t = \beta M^{t-1} + \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$
6: $W^t = W^{t-1} - \eta M^t$
7: **Return** W^T

$$M^t = \beta M^{t-1} + \eta \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$$
$$W^t = W^{t-1} - M^t$$

Alternative form (so-called Nesterov momentum) in the literature, see, e.g., [4]

BP + Momentum

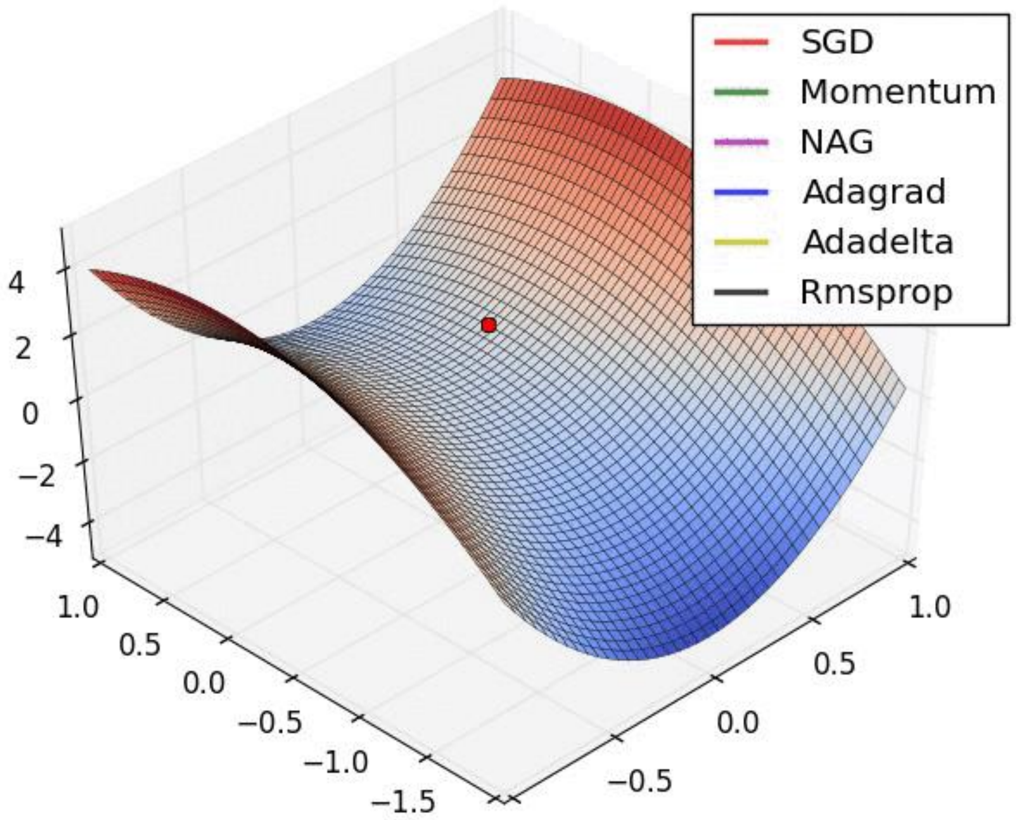


Image Credit: [1]

BP + Momentum

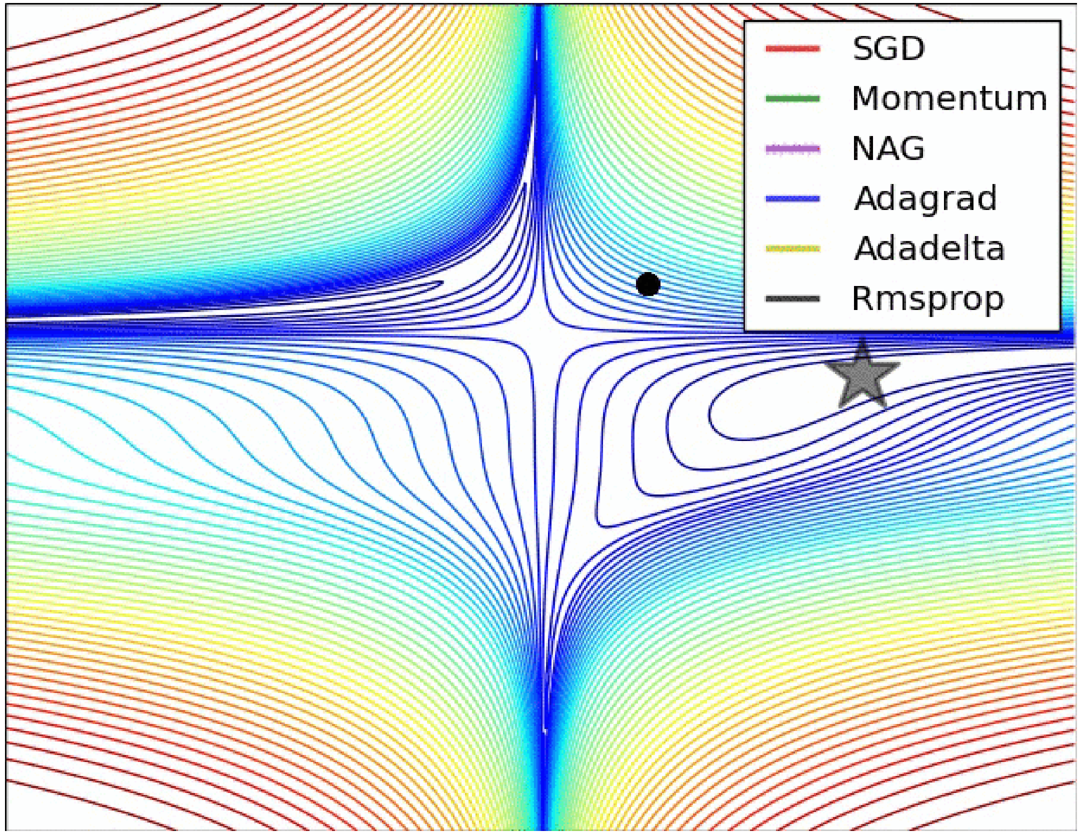


Image Credit: [1]

BP + Momentum

SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another [1, 2].

[3]: Gradient descent is a man walking down a hill. He follows the steepest path downwards; his progress is slow, but steady. Momentum is a heavy ball rolling down the same hill. The added inertia acts both as a smoother and an accelerator, dampening oscillations and causing us to barrel through narrow valleys, small humps and local minima.

Algorithm 1 SGD with Momentum

1: **Input**: step T, learning Rate η, initial W^0, batch size B, momentum β, initial M^0
2: **For** $t = 1, \cdots, T$
3: Get mini-batch data (X^t, Y^t)
4: $L(W^{t-1}, X^t, Y^t) = \frac{1}{B} \sum_{i=1}^{B} \ell(f(W^{t-1}, X^t[i]), Y^t[i])$
5: $M^t = \beta M^{t-1} + \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$
6: $W^t = W^{t-1} - \eta M^t$
7: **Return** W^T

Demo: https://distill.pub/2017/momentum/

BP + Adaptive Learning Rate

Denoting $g^t = \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$, recall in GD/SGD, we have the following update rule:

$$W^t = W^{t-1} - \eta g^t$$

How can we tune the learning rate?

BP + Adaptive Learning Rate

Denoting $g^t = \dfrac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$, recall in GD/SGD, we have the following update rule:

$$W^t = W^{t-1} - \eta g^t$$

How can we tune the learning rate?

AdaGrad (Adaptive Gradient) [5] proposes to assign larger learning rates to infrequently updated weights and smaller ones to frequently updated weights:

$$G^t = \sum_{\tau=1}^{t} g^\tau {g^\tau}^\top$$

$$W^t = W^{t-1} - \eta \text{diag}\left(\epsilon I + G^t\right)^{-1/2} g^t$$

BP + Adaptive Learning Rate

Denoting $g^t = \dfrac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$, recall in GD/SGD, we have the following update rule:

$$W^t = W^{t-1} - \eta g^t$$

How can we tune the learning rate?

AdaGrad (Adaptive Gradient) [5] proposes to assign larger learning rates to infrequently updated weights and smaller ones to frequently updated weights:

$$G^t = \sum_{\tau=1}^{t} g^\tau {g^\tau}^\top$$
$$W^t = W^{t-1} - \eta \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} g^t$$

Let us look at the element-wise update rule of AdaGrad:

$$W^t[i] = W^{t-1}[i] - \frac{\eta}{\sqrt{\epsilon + G^t[i,i]}} g^t[i]$$

BP + Adaptive Learning Rate

Denoting $g^t = \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$, recall in GD/SGD, we have the following update rule:

$$W^t = W^{t-1} - \eta g^t$$

How can we tune the learning rate?

AdaGrad (Adaptive Gradient) [5] proposes to assign larger learning rates to infrequently updated weights and smaller ones to frequently updated weights:

$$\boxed{G^t = \sum_{\tau=1}^{t} g^\tau {g^\tau}^\top}$$

$$W^t = W^{t-1} - \eta \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} g^t$$

In practice, we only need to compute diagonal terms $g^t \odot g^t$

Let us look at the element-wise update rule of AdaGrad:

$$W^t[i] = W^{t-1}[i] - \frac{\eta}{\sqrt{\epsilon + G^t[i,i]}} g^t[i]$$

BP + Adaptive Learning Rate

AdaGrad (Adaptive Gradient) [5] proposes to assign larger learning rates to infrequently updated weights and smaller ones to frequently updated weights:

$$G^t = \sum_{\tau=1}^{t} g^\tau {g^\tau}^\top$$

$$W^t = W^{t-1} - \eta \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} g^t$$

Since we accumulate (squared) gradients from the beginning, our learning rate is always decaying and could vanish before we converge.

BP + Adaptive Learning Rate

AdaGrad (Adaptive Gradient) [5] proposes to assign larger learning rates to infrequently updated weights and smaller ones to frequently updated weights:

$$G^t = \sum_{\tau=1}^{t} g^\tau {g^\tau}^\top$$

$$W^t = W^{t-1} - \eta \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} g^t$$

Since we accumulate (squared) gradients from the beginning, our learning rate is always decaying and could vanish before we converge. To deal with this, RMSProp [6] proposes to use exponential moving average (EMA) of past squared gradients:

$$G^t = \rho G^{t-1} + (1-\rho) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \eta \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} g^t$$

BP + Adaptive Learning Rate

AdaGrad (Adaptive Gradient) [5] proposes to assign larger learning rates to infrequently updated weights and smaller ones to frequently updated weights:

$$G^t = \sum_{\tau=1}^{t} g^\tau {g^\tau}^\top$$

$$W^t = W^{t-1} - \eta \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} g^t$$

Since we accumulate (squared) gradients from the beginning, our learning rate is always decaying and could vanish before we converge. To deal with this, RMSProp [6] proposes to use exponential moving average (EMA) of past squared gradients:

$$G^t = \rho G^{t-1} + (1-\rho) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \eta \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} g^t$$

Adadelta [7] leverages the same EMA trick (denominator) and additional weighting (numerator) based on parameter updates.

Comparison Again

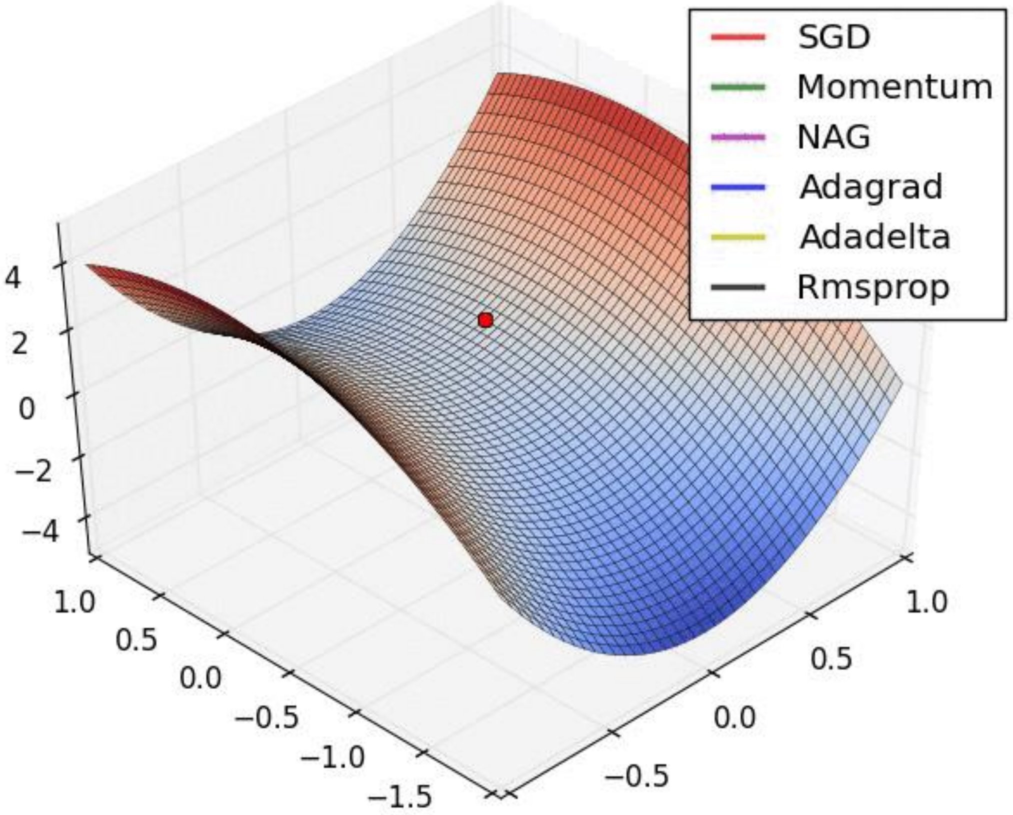


Image Credit: [1]

Comparison Again

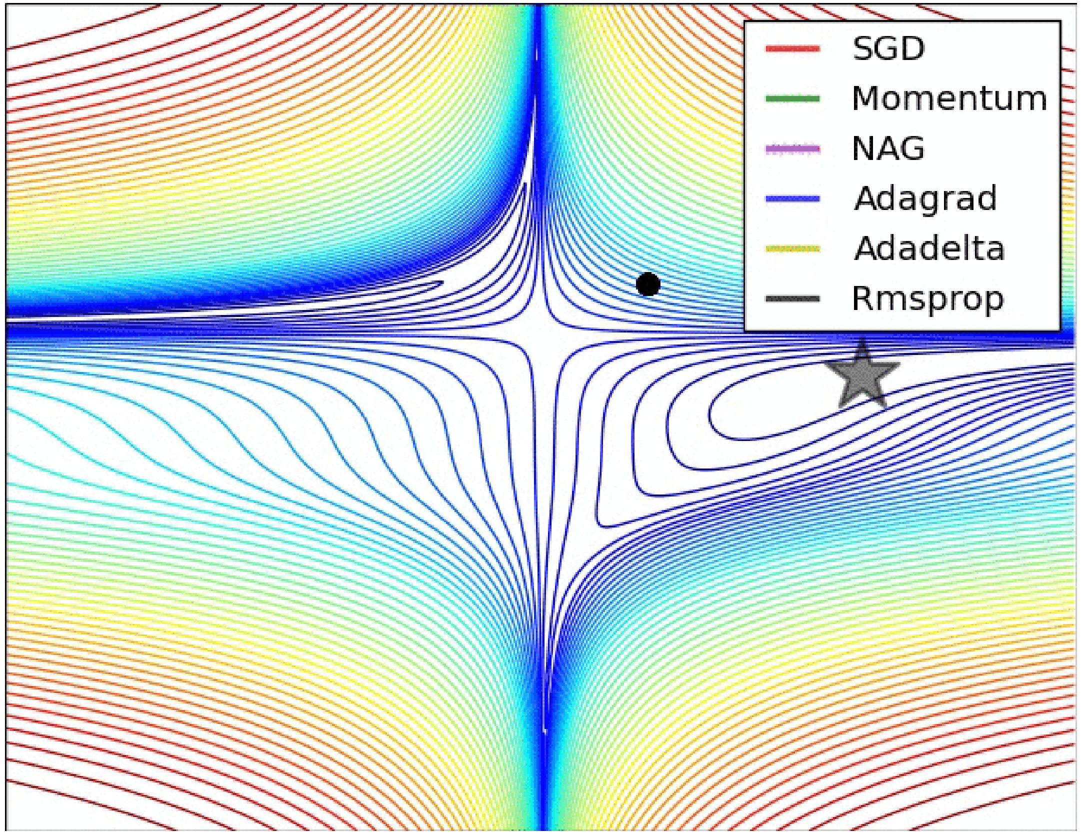


Image Credit: [1]

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1-\beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \eta_t \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t$$

$$\eta_t = \eta \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$$

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1 - \beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1 - \beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \eta_t \text{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t$$

$$\eta_t = \eta \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t}$$

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1-\beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \eta_t \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t$$

$$\eta_t = \eta \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$$

In practice, we only need to compute diagonal terms $g^t \odot g^t$

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1 - \beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1 - \beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \eta_t \text{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t$$

$$\eta_t = \eta \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t}$$

In practice, we only need to compute diagonal terms $g^t \odot g^t$

These are powers of scalars (abuse of superscript)

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1-\beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \eta_t \text{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t$$

$$\eta_t = \eta \boxed{\frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}}$$

When $\beta_1 = 0$, Adam is almost the same as RMSProp besides the correction term (red box)

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1 - \beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1 - \beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \eta_t \text{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t$$

$$\eta_t = \eta \boxed{\frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t}}$$

When $\beta_1 = 0$, Adam is almost the same as RMSProp besides the correction term (red box):

$$G^t = \rho G^{t-1} + (1 - \rho) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \eta \text{diag}\left(\epsilon I + G^t\right)^{-1/2} g^t$$

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1 - \beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1 - \beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \eta_t \text{diag} \left(\epsilon I + G^t \right)^{-1/2} \tilde{g}^t$$

$$\eta_t = \eta \boxed{\frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t}}$$

Where does this bias-correction term (red box) come from?

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1-\beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \eta_t \text{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t$$

$$\eta_t = \eta \boxed{\frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}}$$

Where does this bias-correction term (red box) come from?

$$\begin{aligned}\mathbb{E}\left[\tilde{g}^t\right] &= \mathbb{E}\left[\beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t\right] = \mathbb{E}\left[\beta_1\left(\beta_1 \tilde{g}^{t-2} + (1-\beta_1) g^{t-1}\right) + (1-\beta_1) g^t\right] \\ &= \mathbb{E}\left[\beta_1^2 \tilde{g}^{t-2} + \beta_1 (1-\beta_1) g^{t-1} + (1-\beta_1) g^t\right] \\ &= \mathbb{E}\left[\sum_{i=1}^{t} \beta_1^{t-i} (1-\beta_1) g^i + \beta_1^t \tilde{g}^0\right]\end{aligned}$$

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$
\begin{aligned}
\tilde{g}^t &= \beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t \\
G^t &= \beta_2 G^{t-1} + (1-\beta_2) g^t {g^t}^\top \\
W^t &= W^{t-1} - \eta_t \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t \\
\eta_t &= \eta \boxed{\frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}}
\end{aligned}
$$

Where does this bias-correction term (red box) come from?

$$
\begin{aligned}
\mathbb{E}\left[\tilde{g}^t\right] &= \mathbb{E}\left[\beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t\right] = \mathbb{E}\left[\beta_1 \left(\beta_1 \tilde{g}^{t-2} + (1-\beta_1) g^{t-1}\right) + (1-\beta_1) g^t\right] \\
&= \mathbb{E}\left[\beta_1^2 \tilde{g}^{t-2} + \beta_1 (1-\beta_1) g^{t-1} + (1-\beta_1) g^t\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{t} \beta_1^{t-i} (1-\beta_1) g^i + \boxed{\beta_1^t \tilde{g}^0}\right]
\end{aligned}
$$

We ignore this term due to the zero initialization and assume gradients at each time step are stationary!

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1-\beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \eta_t \text{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t$$

$$\eta_t = \eta \boxed{\frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}}$$

Where does this bias-correction term (red box) come from?

$$\begin{aligned}
\mathbb{E}\left[\tilde{g}^t\right] &= \mathbb{E}\left[\beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t\right] = \mathbb{E}\left[\beta_1\left(\beta_1 \tilde{g}^{t-2} + (1-\beta_1) g^{t-1}\right) + (1-\beta_1) g^t\right] \\
&= \mathbb{E}\left[\beta_1^2 \tilde{g}^{t-2} + \beta_1 (1-\beta_1) g^{t-1} + (1-\beta_1) g^t\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{t} \beta_1^{t-i} (1-\beta_1) g^i + \boxed{\beta_1^t \tilde{g}^0}\right] \\
&\approx \sum_{i=1}^{t} \beta_1^{t-i} (1-\beta_1) \mathbb{E}\left[g^i\right] \\
&= \left(\sum_{i=0}^{t-1} \beta_1^i\right) (1-\beta_1) \mathbb{E}\left[g^i\right] = (1-\beta_1^t) \mathbb{E}\left[g^i\right]
\end{aligned}$$

We ignore this term due to the zero initialization and assume gradients at each time step are stationary!

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1 - \beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1 - \beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \eta_t \text{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t$$

$$\eta_t = \eta \boxed{\frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t}}$$

Where does this bias-correction term (red box) come from?

Similar reasoning applies to G^t. We have $\mathbb{E}\left[G^t\right] \approx (1 - \beta_2^t) \mathbb{E}\left[g^i {g^i}^\top\right]$ $\mathbb{E}\left[g^t\right] \approx (1 - \beta_1^t) \mathbb{E}\left[g^i\right]$

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1-\beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \boxed{\eta_t \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t} \qquad \text{denoted as } \Delta W^t$$

$$\eta_t = \eta \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$$

Where does this bias-correction term (red box) come from?

Similar reasoning applies to G^t. We have $\mathbb{E}\left[G^t\right] \approx (1-\beta_2^t)\mathbb{E}\left[g^i {g^i}^\top\right]$ $\quad \mathbb{E}\left[g^t\right] \approx (1-\beta_1^t)\mathbb{E}\left[g^i\right]$

Check the update (blue box) element-wise:

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1-\beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \boxed{\eta_t \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t}$$

denoted as ΔW^t

$$\eta_t = \eta \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$$

Where does this bias-correction term (red box) come from?

Similar reasoning applies to G^t. We have $\mathbb{E}\left[G^t[j,j]\right] \approx (1-\beta_2^t)\mathbb{E}\left[\left(g^i[j]\right)^2\right]$ $\qquad \mathbb{E}\left[\tilde{g}^t\right] \approx (1-\beta_1^t)\mathbb{E}\left[g^i\right]$

Check the update (blue box) element-wise:

$$\mathbb{E}\left[\Delta W^t[j]\right] = \mathbb{E}\left[\eta_t \frac{1}{\sqrt{\epsilon + G^t[j,j]}} \tilde{g}^t[j]\right] \boxed{\approx \eta_t \mathbb{E}\left[\frac{1}{\sqrt{G^t[j,j]}} \tilde{g}^t[j]\right]}$$

ignore epsilon

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1-\beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \boxed{\eta_t \text{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t}$$

denoted as ΔW^t

$$\eta_t = \eta \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$$

Where does this bias-correction term (red box) come from?

Similar reasoning applies to G^t. We have $\mathbb{E}\left[G^t[j,j]\right] \approx (1-\beta_2^t)\mathbb{E}\left[\left(g^i[j]\right)^2\right]$ $\quad \mathbb{E}\left[\tilde{g}^t\right] \approx (1-\beta_1^t)\mathbb{E}\left[g^i\right]$

Check the update (blue box) element-wise:

$$\mathbb{E}\left[\Delta W^t[j]\right] = \mathbb{E}\left[\eta_t \frac{1}{\sqrt{\epsilon + G^t[j,j]}} \tilde{g}^t[j]\right] \approx \eta_t \mathbb{E}\left[\frac{1}{\sqrt{G^t[j,j]}} \tilde{g}^t[j]\right]$$

Independent assumption of G^t and $\tilde{g}^t$

$$= \boxed{\eta_t \frac{\mathbb{E}\left[\tilde{g}^t[j]\right]}{\mathbb{E}\left[\sqrt{G^t[j,j]}\right]}}$$

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1-\beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \boxed{\eta_t \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t}$$

denoted as ΔW^t

$$\eta_t = \eta \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$$

Where does this bias-correction term (red box) come from?

Similar reasoning applies to G^t. We have $\mathbb{E}\left[G^t[j,j]\right] \approx (1-\beta_2^t)\mathbb{E}\left[\left(g^i[j]\right)^2\right]$ $\qquad$ $\mathbb{E}\left[\tilde{g}^t\right] \approx (1-\beta_1^t)\mathbb{E}\left[g^i\right]$

Check the update (blue box) element-wise:

$$\mathbb{E}\left[\Delta W^t[j]\right] = \mathbb{E}\left[\eta_t \frac{1}{\sqrt{\epsilon + G^t[j,j]}} \tilde{g}^t[j]\right] \approx \eta_t \mathbb{E}\left[\frac{1}{\sqrt{G^t[j,j]}} \tilde{g}^t[j]\right]$$

A crude assumption (Jensen's inequality is tight) on G^t

$$= \eta_t \frac{\mathbb{E}\left[\tilde{g}^t[j]\right]}{\mathbb{E}\left[\sqrt{G^t[j,j]}\right]} \approx \boxed{\eta_t \frac{\mathbb{E}\left[\tilde{g}^t[j]\right]}{\sqrt{\mathbb{E}\left[G^t[j,j]\right]}}}$$

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1-\beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \boxed{\eta_t \mathrm{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t}$$

denoted as ΔW^t

$$\eta_t = \eta \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$$

Where does this bias-correction term (red box) come from?

Similar reasoning applies to G^t. We have $\mathbb{E}\left[G^t[j,j]\right] \approx (1-\beta_2^t)\mathbb{E}\left[\left(g^i[j]\right)^2\right]$ $\quad \mathbb{E}\left[\tilde{g}^t\right] \approx (1-\beta_1^t)\mathbb{E}\left[g^i\right]$

Check the update (blue box) element-wise:

$$\mathbb{E}\left[\Delta W^t[j]\right] = \mathbb{E}\left[\eta_t \frac{1}{\sqrt{\epsilon + G^t[j,j]}} \tilde{g}^t[j]\right] \approx \eta_t \mathbb{E}\left[\frac{1}{\sqrt{G^t[j,j]}} \tilde{g}^t[j]\right]$$

Substitute with above results

$$= \eta_t \frac{\mathbb{E}\left[\tilde{g}^t[j]\right]}{\mathbb{E}\left[\sqrt{G^t[j,j]}\right]} \approx \eta_t \frac{\mathbb{E}\left[\tilde{g}^t[j]\right]}{\sqrt{\mathbb{E}\left[G^t[j,j]\right]}}$$

$$\approx \eta_t \frac{(1-\beta_1^t)\mathbb{E}\left[g^i[j]\right]}{\sqrt{1-\beta_2^t}\sqrt{\mathbb{E}\left[\left(g^i[j]\right)^2\right]}}$$

BP + Adaptive Learning Rate + Momentum (Adam)

Now that we can tune the learning rate adaptively, how can we incorporate momentum?

Based on RMSProp, Adam (Adaptive Momentum) [8] proposes to keep another EMA of past gradients:

$$\tilde{g}^t = \beta_1 \tilde{g}^{t-1} + (1-\beta_1) g^t$$

$$G^t = \beta_2 G^{t-1} + (1-\beta_2) g^t {g^t}^\top$$

$$W^t = W^{t-1} - \boxed{\eta_t \text{diag}\left(\epsilon I + G^t\right)^{-1/2} \tilde{g}^t} \quad \text{denoted as } \Delta W^t$$

$$\eta_t = \eta \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$$

Where does this bias-correction term (red box) come from?

Similar reasoning applies to G^t. We have $\mathbb{E}\left[G^t[j,j]\right] \approx (1-\beta_2^t)\mathbb{E}\left[\left(g^i[j]\right)^2\right]$ $\quad \mathbb{E}\left[\tilde{g}^t\right] \approx (1-\beta_1^t)\mathbb{E}\left[g^i\right]$

Check the update (blue box) element-wise:

$$\mathbb{E}\left[\Delta W^t[j]\right] = \mathbb{E}\left[\eta_t \frac{1}{\sqrt{\epsilon + G^t[j,j]}} \tilde{g}^t[j]\right] \approx \eta_t \mathbb{E}\left[\frac{1}{\sqrt{G^t[j,j]}} \tilde{g}^t[j]\right]$$

$$= \eta_t \frac{\mathbb{E}\left[\tilde{g}^t[j]\right]}{\mathbb{E}\left[\sqrt{G^t[j,j]}\right]} \approx \eta_t \frac{\mathbb{E}\left[\tilde{g}^t[j]\right]}{\sqrt{\mathbb{E}\left[G^t[j,j]\right]}}$$

$$\approx \eta_t \frac{(1-\beta_1^t)\mathbb{E}\left[g^i[j]\right]}{\sqrt{1-\beta_2^t}\sqrt{\mathbb{E}\left[\left(g^i[j]\right)^2\right]}} \leq \boxed{\eta_t \frac{(1-\beta_1^t)}{\sqrt{1-\beta_2^t}}} = \eta$$

Jensen's Inequality

$$\mathbb{E}\left[g^i[j]\right] \leq \sqrt{\mathbb{E}\left[\left(g^i[j]\right)^2\right]}$$

BP + Adaptive Learning Rate + Momentum (Adam)

In summary, Adam is shown in below

Algorithm 1 Adam

Input: step T, initial learning Rate η, initial W^0, batch size B, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $m^0 = 0$, $v^0 = 0$

For $t = 1, \cdots, T$

  Get mini-batch data (X^t, Y^t)

  $L(W^{t-1}, X^t, Y^t) = \frac{1}{B}\sum_{i=1}^{B} \ell(f(W^{t-1}, X^t[i]), Y^t[i])$

  $g^t = \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$

  $m^t = \beta_1 m^{t-1} + (1 - \beta_1) g^t$

  $v^t = \beta_2 v^{t-1} + (1 - \beta_2)(g^t \odot g^t)$

  $W^t = W^{t-1} - \eta \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} \frac{m^t}{\sqrt{v^t}+\epsilon}$

Return W^T

BP + Adaptive Learning Rate + Momentum (Adam)

In summary, Adam is shown in below

Algorithm 1 Adam

1: **Input**: step T, initial learning Rate η, initial W^0, batch size B, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $m^0 = 0$, $v^0 = 0$
2: **For** $t = 1, \cdots, T$
3: Get mini-batch data (X^t, Y^t)
4: $L(W^{t-1}, X^t, Y^t) = \frac{1}{B}\sum_{i=1}^{B} \ell(f(W^{t-1}, X^t[i]), Y^t[i])$
5: $g^t = \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$
6: $m^t = \beta_1 m^{t-1} + (1-\beta_1)g^t$
7: $v^t = \beta_2 v^{t-1} + (1-\beta_2)(g^t \odot g^t)$
8: $W^t = W^{t-1} - \eta \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} \frac{m^t}{\sqrt{v^t}+\epsilon}$
9: **Return** W^T

Adam [8] has been cited by 165,000+ times since its publication in 2015! It is by far the most popular optimizer in deep learning!

Outline

- Learning Algorithm for Feedforward Neural Networks:
  - Backpropagation
  - Weight Initialization
  - Learning Rate & Momentum & Adam
  - **Weight Decay & Early Stopping**

Weight Decay

Suppose we observe overfitting and want to regularize the complexity of our neural networks to reduce it. We can penalize the weights so that they are not far from 0, thus restricting the set of models we are considering.

Weight Decay

Suppose we observe overfitting and want to regularize the complexity of our neural networks to reduce it. We can penalize the weights so that they are not far from 0, thus restricting the set of models we are considering.

$$L(W, X, Y) = \frac{1}{B} \sum_{i=1}^{B} \ell(f(W, X[i]), Y[i])$$

$$\tilde{L}(W, X, Y) = L(W, X, Y) + \frac{\lambda}{2} \|\text{Vec}(W)\|_2^2$$

We can control the lambda to trade-off model complexity and overfitting.

Weight Decay

Suppose we observe overfitting and want to regularize the complexity of our neural networks to reduce it. We can penalize the weights so that they are not far from 0, thus restricting the set of models we are considering.

$$L(W, X, Y) = \frac{1}{B}\sum_{i=1}^{B} \ell(f(W, X[i]), Y[i])$$

$$\tilde{L}(W, X, Y) = L(W, X, Y) + \frac{\lambda}{2}\|\mathrm{Vec}(W)\|_2^2$$

We can control the lambda to trade-off model complexity and overfitting.

It only adds a term to the existing gradient:

$$\frac{\partial \tilde{L}}{\partial W} = \frac{\partial L}{\partial W} + \lambda W$$

For methods like SGD, SGD + Momentum, we directly add this term to the gradient and then perform gradient update

Weight Decay

Suppose we observe overfitting and want to regularize the complexity of our neural networks to reduce it. We can penalize the weights so that they are not far from 0, thus restricting the set of models we are considering.

$$L(W, X, Y) = \frac{1}{B} \sum_{i=1}^{B} \ell(f(W, X[i]), Y[i])$$

$$\tilde{L}(W, X, Y) = L(W, X, Y) + \frac{\lambda}{2} \|\text{Vec}(W)\|_2^2$$

We can control the lambda to trade-off model complexity and overfitting.

It only adds a term to the existing gradient: $$\frac{\partial \tilde{L}}{\partial W} = \frac{\partial L}{\partial W} + \lambda W$$

For methods like SGD, SGD + Momentum, we directly add this term to the gradient and then perform gradient update

But for Adam, we have two possibilities: 1) add it to the gradient and then perform gradient update; 2) add it to the gradient update

This ordering change makes a difference in practice and 2) is called AdamW [9]!

AdamW

Let us look at Adam again:

Algorithm 1 Adam

1: **Input**: step T, initial learning Rate η, initial W^0, batch size B, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $m^0 = 0$, $v^0 = 0$
2: **For** $t = 1, \cdots, T$
3: Get mini-batch data (X^t, Y^t)
4: $L(W^{t-1}, X^t, Y^t) = \frac{1}{B}\sum_{i=1}^{B} \ell(f(W^{t-1}, X^t[i]), Y^t[i])$
5: $g^t = \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$
6: $m^t = \beta_1 m^{t-1} + (1-\beta_1) g^t$
7: $v^t = \beta_2 v^{t-1} + (1-\beta_2)(g^t \odot g^t)$
8: $W^t = W^{t-1} - \eta \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} \frac{m^t}{\sqrt{v^t}+\epsilon}$
9: **Return** W^T

We can it here by replacing the rex box with:

$$g^t = \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W} + \lambda W^{t-1}$$

But then weight decay also goes through EMA!

AdamW

We can add it in the final gradient update:

Algorithm 1 AdamW

1: **Input**: step T, initial learning Rate η, initial W^0, batch size B, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $m^0 = 0$, $v^0 = 0$, weight decay λ

2: **For** $t = 1, \cdots, T$

3: Get mini-batch data (X^t, Y^t)

4: $L(W^{t-1}, X^t, Y^t) = \frac{1}{B}\sum_{i=1}^{B} \ell(f(W^{t-1}, X^t[i]), Y^t[i])$

5: $g^t = \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$

6: $m^t = \beta_1 m^{t-1} + (1-\beta_1) g^t$

7: $v^t = \beta_2 v^{t-1} + (1-\beta_2)(g^t \odot g^t)$

8: $W^t = W^{t-1} - \eta \left(\frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} \frac{m^t}{\sqrt{v^t}+\epsilon} + \lambda W^{t-1} \right)$

9: **Return** W^T

This change works significantly better in some cases [9]!

Early Stopping

We do not necessarily need to run the optimization until the maximum number of iterations or until its convergence.

Algorithm 1 AdamW

Input: step T, initial learning Rate η, initial W^0, batch size B, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $m^0 = 0$, $v^0 = 0$, weight decay λ

For $t = 1, \cdots, T$

  Get mini-batch data (X^t, Y^t)

  $L(W^{t-1}, X^t, Y^t) = \frac{1}{B}\sum_{i=1}^{B} \ell(f(W^{t-1}, X^t[i]), Y^t[i])$

  $g^t = \frac{\partial L(W^{t-1}, X^t, Y^t)}{\partial W}$

  $m^t = \beta_1 m^{t-1} + (1-\beta_1) g^t$

  $v^t = \beta_2 v^{t-1} + (1-\beta_2)(g^t \odot g^t)$

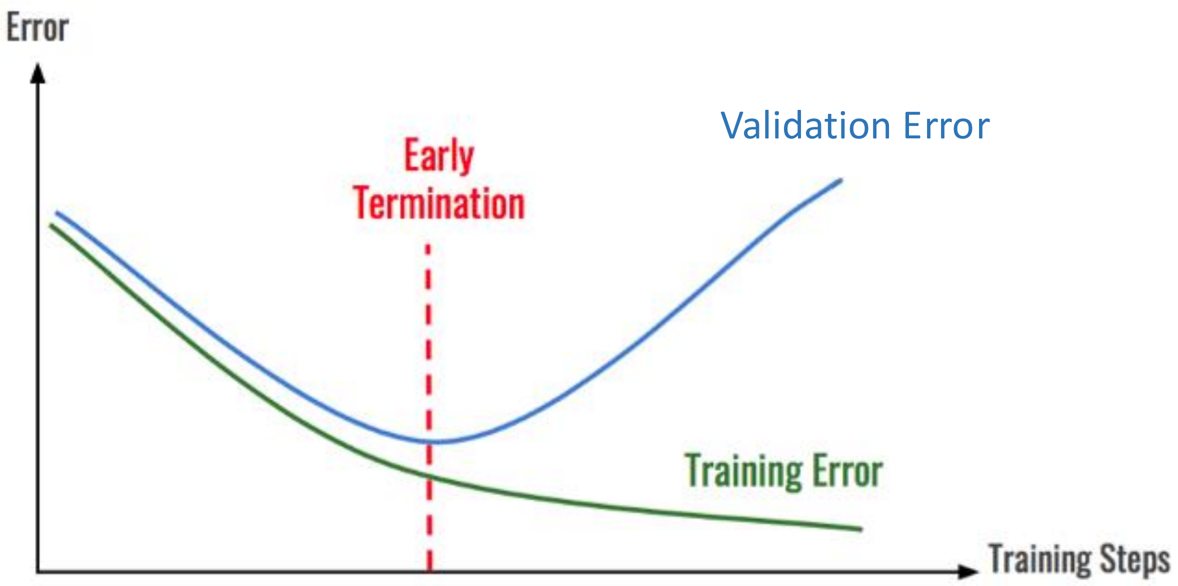
  $W^t = W^{t-1} - \eta \left(\frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} \frac{m^t}{\sqrt{v^t}+\epsilon} + \lambda W^{t-1} \right)$

Return W^T

Image Credit: [10]

Early Stopping

We do not necessarily need to run the optimization until the maximum number of iterations or until its convergence.



We can check the validation error and if it is keep increasing within a time window, then we stop the training!

If you worry that it will go down again, run until the maximum number of iterations and pick the model with the least validation error.

Image Credit: [10]

References

[1] https://www.ruder.io/optimizing-gradient-descent/

[2] Sutton, R. S. (1986). Two problems with backpropagation and other steepest-descent learning procedures for networks. In Proc. of Eightth Annual Conference of the Cognitive Science Society (pp. 823-831).

[3] https://distill.pub/2017/momentum/

[4] Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, May). On the importance of initialization and momentum in deep learning. In International conference on machine learning (pp. 1139-1147). PMLR.

[5] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 12(7).

[6] Tieleman, T. and Hinton, G. (2012). Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning. Technical report.

[7] Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.

[8] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[9] Loshchilov, I., & Hutter, F. (2017). Fixing weight decay regularization in adam.

[10] https://medium.com/analytics-vidhya/early-stopping-with-pytorch-to-restrain-your-model-from-overfitting-dce6de4081c5

Questions?