Deep Learning

Momentum-based Gradient Descent

Syed Irtaza Muzaffar

So far . . .

- Neural Networks are universal approximators.
- Backpropagation allows computation of derivatives in hidden layers.
- Gradient descent finds weights corresponding to local minimum of loss surface.
- 1st- and 2nd-order variants of gradient descent can be faster and better.
- In this lecture:
  - Momentum-based first-order methods
    - Momentum
    - Nesterov Accelerated Gradient
    - RMSprop
    - ADAM

Momentum Updates

Basic idea

- Keep track of oscillating directions.
- Increase learning rate in directions that converge smoothly.
- Decrease learning rate in directions that overshoot and oscillate.

Steps

1. Compute gradient step $-\eta \left.\nabla_{\mathbf{w}} L\right|_{\mathbf{w}^\tau}$ at the current location $\mathbf{w}^\tau$.
2. Add the scaled previous step $\beta \Delta \mathbf{w}^\tau$ to obtain a running average of the step

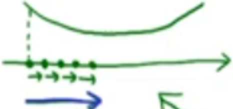
$$\Delta \mathbf{w}^{\tau+1} = \beta \Delta \mathbf{w}^\tau - \eta \left.\nabla_{\mathbf{w}} L\right|_{\mathbf{w}^\tau}$$

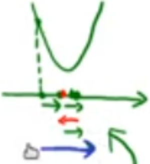
   Typically $\beta = 0.9$.

3. Update parameters by the running average of the step

$$\mathbf{w}^{\tau+1} = \mathbf{w}^\tau + \Delta \mathbf{w}^{\tau+1}$$

Momentum Updates





Basic idea

- Keep track of oscillating directions.
- Increase learning rate in directions that converge smoothly.
- Decrease learning rate in directions that overshoot and oscillate.

Steps

1. Compute gradient step $\overbrace{-\eta \, \nabla_{\mathbf{w}} L|_{\mathbf{w}^\tau}}^{\Delta \mathbf{w}^{\tau+1}}$ at the current location $\mathbf{w}^\tau$.
2. Add the scaled previous step $\beta \Delta \mathbf{w}^\tau$ to obtain a running average of the step

$$\Delta \mathbf{w}^{\tau+1} = \overset{\text{scaling}}{\beta} \underbrace{\Delta \mathbf{w}^{\tau}}_{\text{last jump}} \underbrace{- \eta \, \nabla_{\mathbf{w}} L|_{\mathbf{w}^\tau}}$$

← running average of steps.

Typically $\beta = 0.9$.

3. Update parameters by the running average of the step

$$\underbrace{\mathbf{w}^{\tau+1}}_{\text{new}} = \underbrace{\mathbf{w}^{\tau}}_{\text{old}} + \underbrace{\Delta \mathbf{w}^{\tau+1}}_{\text{step via running avg.}}$$

Why does momentum work?

- Directions that oscillate will cancel each other out in the running average.
  - So the running average will be small in magnitude.
  - So the steps for oscillating directions will be smaller.
- Directions that are consistently converging will be reinforced.
  - So the running average will be large in magnitude.
  - So those directions will gain *momentum* by having larger and larger steps.

Nesterov Accelerated Gradient

Same idea as momentum updates *but with steps 1 and 2 swapped.*

1. Extend the previous scaled step.

$$\hat{\mathbf{w}} = \mathbf{w}^{\tau} + \beta \Delta \mathbf{w}^{\tau}$$

2. Compute gradient step at resultant location $\hat{\mathbf{w}}$.

$$-\eta \left. \nabla_{\mathbf{w}} L \right|_{\hat{\mathbf{w}}}$$

3. Add previous scaled step and new gradient step to obtain the running average of the step

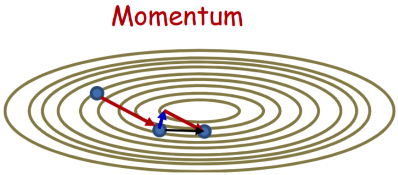
$$\Delta \mathbf{w}^{\tau+1} = \beta \Delta \mathbf{w}^{\tau} - \eta \left. \nabla_{\mathbf{w}} L \right|_{\hat{\mathbf{w}}}$$

4. Update parameters by the running average of the step

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} + \Delta \mathbf{w}^{\tau+1}$$

Nesterov's method has been shown to converge faster than momentum updates.

Momentum vs. Nesterov Momentum



Nesterov – Sometimes it is better to make a correction after making an error.

Source: Bhiksha Raj

Momentum vs. Nesterov Momentum

Nesterov – Sometimes it is better make a correction after making an error.

Source: Bhiksha Raj

RMSprop

- Decouple each direction.
- We can compute the running average of the *squared* 1st-derivative in direction i as

$$\bar{v}_i^{\tau} = \gamma \bar{v}_i^{\tau-1} + (1-\gamma)\left(\frac{\partial L}{\partial w_i}\right)^2$$

  with initialization $\bar{v}_i^0 = 0$.
- Root-mean-squared (RMS) value $\sqrt{\bar{v}_i^{\tau}} + \epsilon$ represents average magnitude of 1st-derivative for direction i.
  - High value indicates oscillating derivatives. So reduce learning rate.
  - Low value indicates flat region. So increase learning rate.
- So divide learning rate by this average before performing gradient descent.

$$w_i^{\tau} = w_i^{\tau-1} - \frac{\eta}{\sqrt{\bar{v}_i^{\tau}} + \epsilon}\frac{\partial L}{\partial w_i}$$

Rprop vs RMSprop

Rprop

Multiplicatively increase learning rate when derivative retains its sign.

$$\eta \leftarrow \alpha\eta$$

Multiplicatively decrease learning rate when derivative oscillates.

$$\eta \leftarrow \beta\eta$$

RMSprop

Multiplicatively increase/decrease learning rate when average derivative magnitude decreases/increases.

$$\eta \leftarrow \frac{\eta_0}{\sqrt{\bar{v}} + \epsilon}$$

Fixed multiplicative factors α and β in Rprop are replaced by *adaptive* factor $\frac{1}{\sqrt{\bar{v}}+\epsilon}$ in RMSprop.

ADAM

RMSprop+Momentum

- RMSprop uses the current derivative.
- ADAM[1] replaces current derivative by its running average.

$$\bar{m}_i^{\tau} = \delta \bar{m}_i^{\tau-1} + (1-\delta)\frac{\partial L}{\partial w_i}$$

- *Currently the most popular flavor of gradient descent.*
- Statistics terminology:
  - Average of random variable x is also called its 1st statistical moment $E[x]$.
  - Average of the square of a random variable is also called its 2nd *uncentered* statistical moment $E[x^2]$.
  - Average of the square of a centered random variable is also called its 2nd statistical moment $E[(x-\mu)^2]$ or variance.

[1] Kingma and Ba, 'ADAM: A Method for Stochastic Optimization'.

ADAM
RMSprop+Momentum

- Initialize moments $\bar{m}_i^0 = 0$ and $\bar{v}_i^0 = 0$.
- Compute 1st moment and 2nd uncentered moment of derivative

$$\bar{m}_i^\tau = \delta \bar{m}_i^{\tau-1} + (1-\delta)\frac{\partial L}{\partial w_i}$$

$$\bar{v}_i^\tau = \gamma \bar{v}_i^{\tau-1} + (1-\gamma)\left(\frac{\partial L}{\partial w_i}\right)^2$$

- Correct for bias of initial moments $(=0)$ by scaling up in early iterations.

$$\bar{m}_i^\tau = \frac{\bar{m}_i^\tau}{1-\delta^\tau} \text{ and } \bar{v}_i^\tau = \frac{\bar{v}_i^\tau}{1-\gamma^\tau}$$

- Perform update

$$w_i^\tau = w_i^{\tau-1} - \frac{\eta}{\sqrt{\bar{v}_i^\tau} + \epsilon}\bar{m}_i^\tau$$

- Proposed hyperparameter values: $\eta = 10^{-3}, \delta = 0.9, \gamma = 0.999, \epsilon = 10^{-8}$.

Summary

- For complex and non-convex loss functions of deep networks, vanilla gradient descent can get stuck in poor local minima and saddle points.
- It can also converge very slowly.
- Different directions require different learning rates.
- Adaptive learning rates are very important.
- Most useful technique is to adapt learning rate based on *recent trend* of 1st-derivative.