Advanced Optimization

Techniques

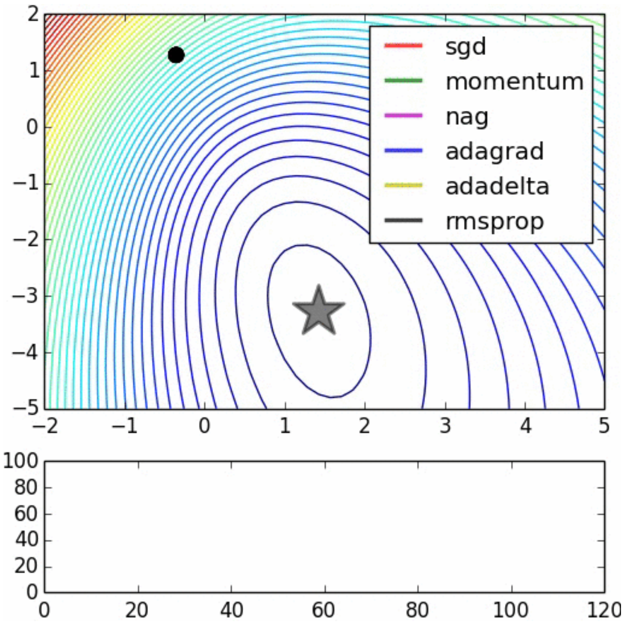
What is the need for optimization?

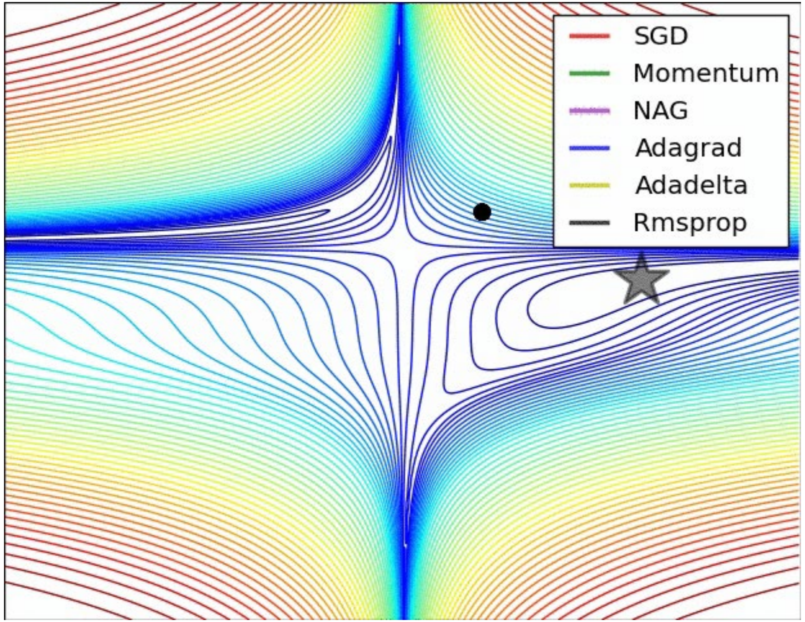
- Slow conversion rate
- Gradient gets stuck in the local optima

Gradient descent optimization algorithms

- Gradient descent without Momentum
- Nesterov accelerated gradient
- Adagrad
- RMSprop
- Adam
- Second order optimization
- Recommended practices

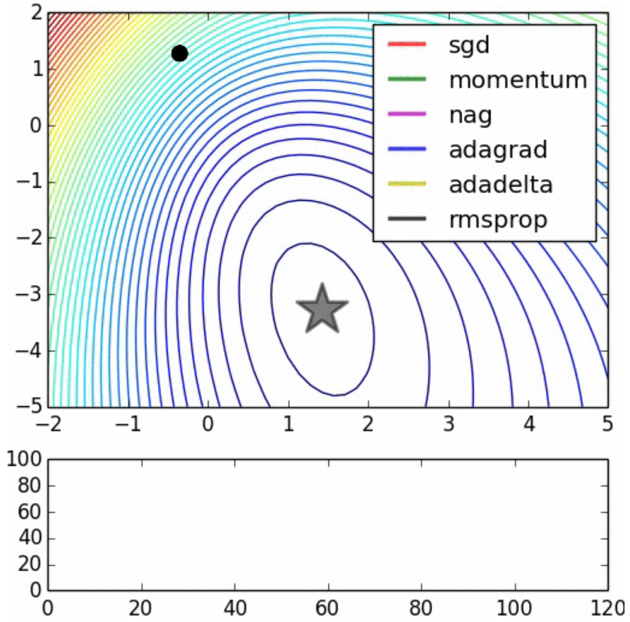
Optimizers: visualization





Source and full animations: Alec Radford

Gradient Descent without Momentum (Red)



SGD

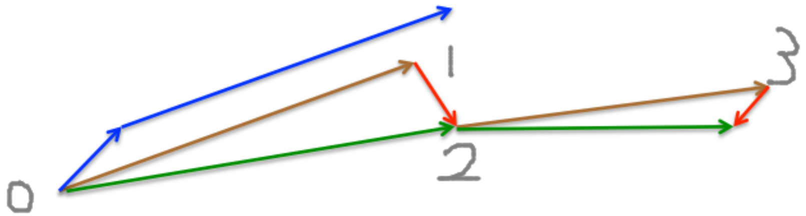
Source and full animations: Alec Radford

(Regular) Momentum Update

Nesterov Momentum Update

A picture of the Nesterov method

- First make a big jump in the direction of the previous accumulated gradient.
- Then measure the gradient where you end up and make a correction.



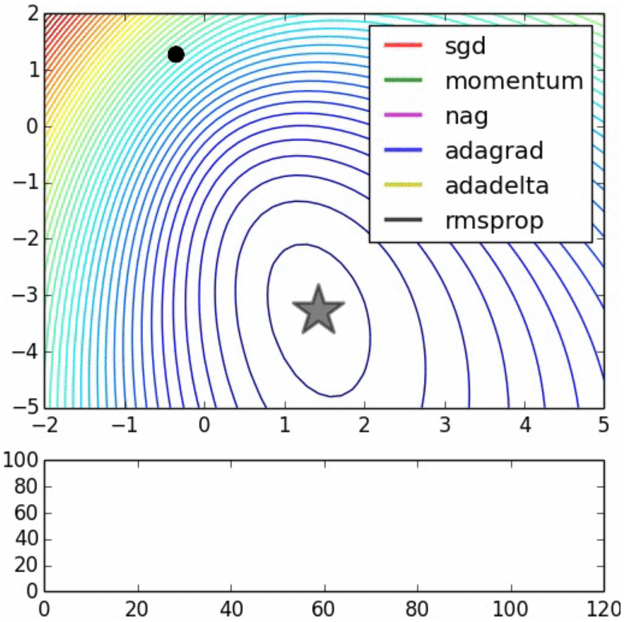
brown vector = jump, red vector = correction, green vector = accumulated gradient

blue vectors = standard momentum

$$m_{t+1} = \alpha m_t - \lambda \frac{\partial L}{\partial (W_t + \alpha m_t)}$$

$$W_{t+1} = W_t + m_{t+1}$$

Gradient Descent with Nesterov Momentum (Pink)



Nesterov accelerated gradient (NAG)

AdaGrad Update

[Duchi et al., 2011]

$$\kappa_t = \kappa_{t-1} + \left(\frac{\partial L}{\partial W_t}\right)^2$$

$$W_t = W_t + \frac{-\lambda \frac{\partial L}{\partial W_t}}{\sqrt{\kappa_t + 10^{-7}}}$$

Elementwise

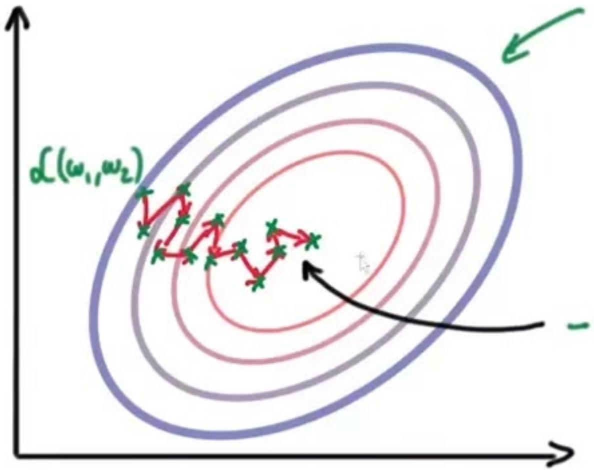
```
# Assume the gradient dw and parameter vector w
cache += dw**2
w += - learning_rate * dw / (np.sqrt(cache) + eps)
```

Added element-wise scaling of the gradient based on the historical sum of squares in each dimension

AdaGrad Update

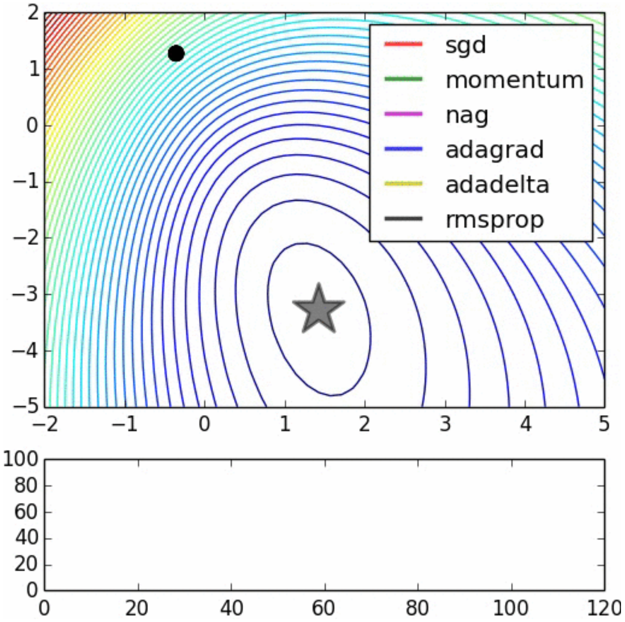
$$W_t = W_t + \frac{-\lambda \frac{\partial L}{\partial W_t}}{\sqrt{\kappa_t + 10^{-7}}} \qquad \kappa_t = \kappa_{t-1} + \left(\frac{\partial L}{\partial W_t}\right)^2$$

Elementwise



Q2: What happens to the step size over long time?

Gradient Descent with AdaGrad (Blue)



Adagrad

Source and full animations: Alec Radford

RMSProp update

greatlearning
Learning for Life

$$\boldsymbol{W}_t = \boldsymbol{W}_t + \frac{-\lambda \dfrac{\partial L}{\partial \boldsymbol{W}_t}}{\sqrt{\boldsymbol{\kappa}_t + 10^{-7}}}$$

$$\boldsymbol{\kappa}_t = \boldsymbol{\kappa}_{t-1} + \left(\frac{\partial L}{\partial \boldsymbol{W}_t}\right)^2$$

Adagrad

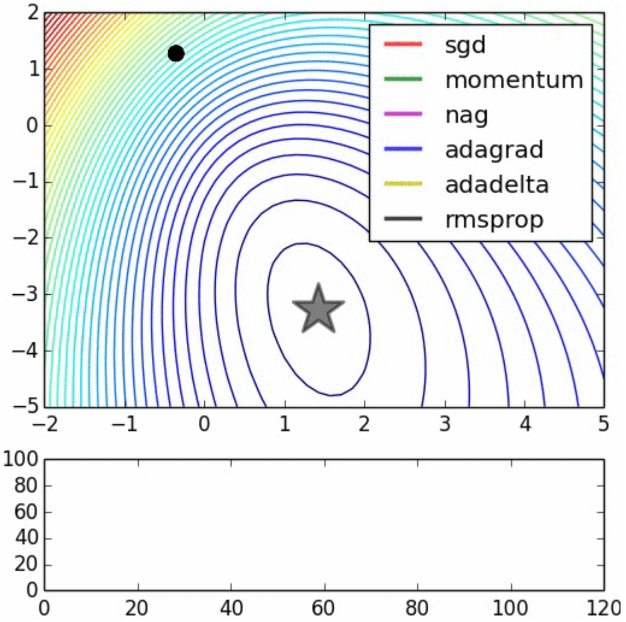
$$\boldsymbol{W}_t = \boldsymbol{W}_t + \frac{-\lambda \dfrac{\partial L}{\partial \boldsymbol{W}_t}}{\sqrt{\boldsymbol{\kappa}_t + 10^{-7}}}$$

$$\boldsymbol{\kappa}_t = \zeta \boldsymbol{\kappa}_{t-1} + (1 - \zeta) \left(\frac{\partial L}{\partial \boldsymbol{W}_t}\right)^2$$

RMS Prop

~0.99 to make it leaky

Gradient Descent with RMSProp (Black)



RMSProp

Adam Update (Incomplete but close)

Looks a bit like RMSProp with momentum

$$\boldsymbol{m_t} = \propto_1 \boldsymbol{m_{t-1}} + (1-\propto_1)\frac{\partial L}{\partial \boldsymbol{W}_t}$$

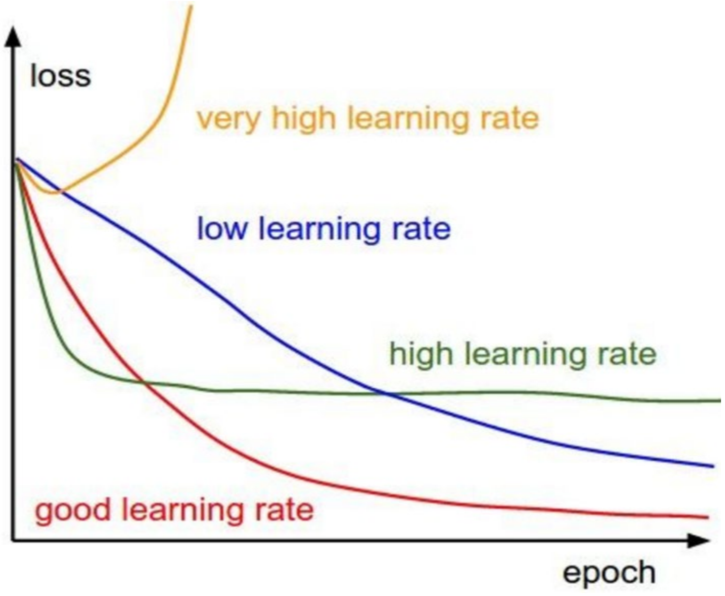
Momentum like

$$\boldsymbol{\kappa_t} = \zeta \boldsymbol{\kappa_{t-1}} + (1-\zeta)\left(\frac{\partial L}{\partial \boldsymbol{W}_t}\right)^2$$

RMSProp like

$$\boldsymbol{W}_t = \boldsymbol{W}_t + \frac{-\lambda \boldsymbol{m_t}}{\sqrt{\boldsymbol{\kappa_t}} + 10^{-7}}$$

SGD, SGD+Momentum, Adagrad, RMSProp, Adam all have learning rate as a hyperparameter.



=> Learning rate decay over time!

step decay:
e.g. decay learning rate by half every few epochs.

exponential decay:

$$\alpha = \alpha_0 e^{-kt}$$

1/t decay:

$$\alpha = \alpha_0 / (1 + kt)$$

Second order optimization methods

Second order Taylor expansion:

$$J(\boldsymbol{\theta}) \approx J(\boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$$

Solving for the critical point we obtain the Newton parameter update:

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 - \boldsymbol{H}^{-1} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0)$$

Q: what is nice about this update?

notice: no hyperparameters! (e.g. learning rate), fast convergence

Second order optimization methods

Second order Taylor expansion:

$$J(\boldsymbol{\theta}) \approx J(\boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$$

Solving for the critical point we obtain the Newton parameter update:

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 - \boldsymbol{H}^{-1} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0)$$

Q: why is this impractical for training Deep Neural Nets?

If 100 million params, H = 100x100 million. Then we need to invert it!

In practice:

- Begin with SGD with vanilla momentum
- Using Adam is a good choice in many cases

Thank you!

Happy Learning :)

Adam

- Adam can be defined as the merger of RMSprop and SGD with momentum.
- Like RMSprop, it utilizes the squared gradients to scale the learning rate and similarly like SGD with momentum, it uses the moving average of the gradient as an alternative of the gradient itself.
- Adam utilizes the estimations of first and second moments of gradient to adapt the learning rate for each weight in the neural network.
- The first moment is the mean and the second moment is not centred variance,ie. During calculation of variance, we don’t subtract the mean.

Adam

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

m_t and v_t are estimates of first and second moment respectively

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{V_t}{1 - \beta_2^t}$$

$\hat{m}_t$ and $\hat{v}_t$ are bias corrected estimates of first and second moment respectively

$$\theta_{t+1} = \theta_t - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}}$$