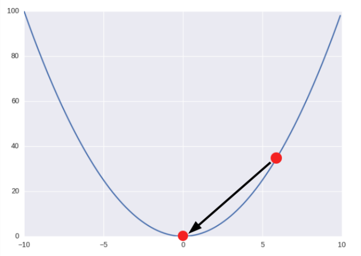
Basic algorithms

Stochastic gradient descent (SGD)

- Most used algorithm for deep learning
- Do not confuse with (deterministic) gradient descent
    - Stochastic uses minibatches
- Algorithm is similar, but there are some important modifications



Gradient descent algorithm

- Full training samples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with targets $\boldsymbol{y}^{(i)}$
- Compute gradient

$$\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \left(\sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \right) \tag{5}$$

- Apply update

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g} \tag{6}$$

where

  - ϵ is the learning rate
  - $\boldsymbol{\theta}$ are the network parameters
  - $L(\cdot)$ is the loss function

Stochastic gradient descent algorithm

- Minibatch of training samples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with targets $\boldsymbol{y}^{(i)}$
- Compute gradient

$$\hat{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \left(\sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \right) \tag{7}$$

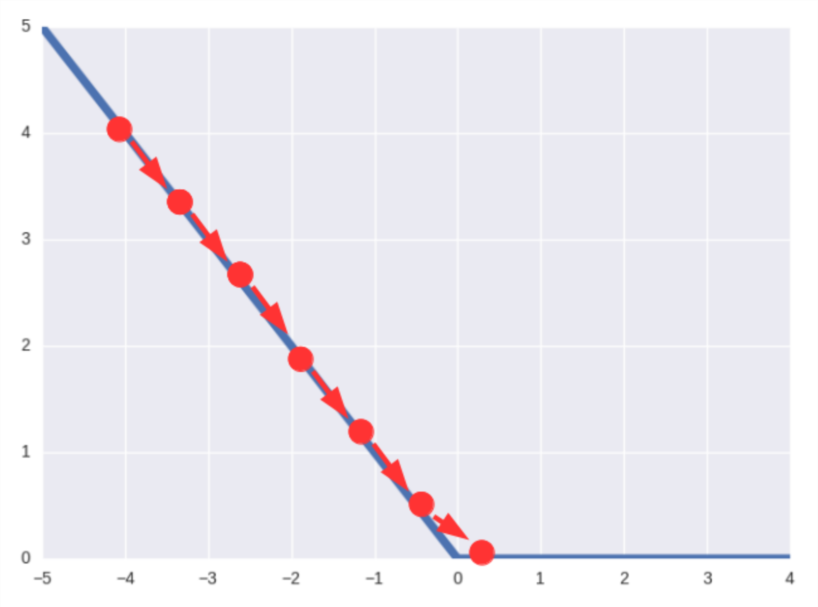
- Apply update

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \boxed{\epsilon_k}\hat{g} \tag{8}$$

Learning rate for SGD

- The main problem is how to choose ϵ_0
- Tipically:
  - Higher than the best value for the first 100 iterations
  - Monitor the initial results and use a higher value
  - Too high will cause instability

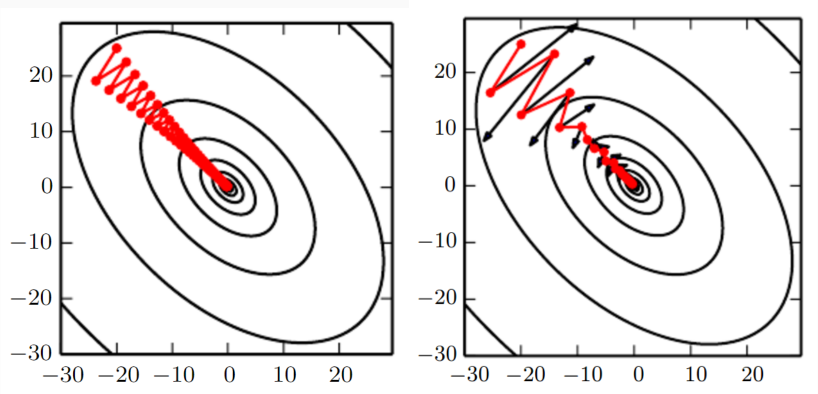
Momentum



Momentum

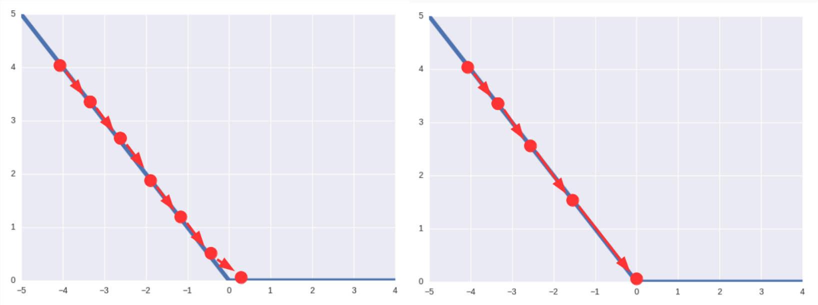
- In these cases, momentum can help
- Derived from the physics term (= mass $\times$ velocity)
- Assume unit mass, so just consider velocity

Momentum



Source: [Goodfellow et al., 2016]

Momentum



SGD with momentum

- Compute gradient

$$\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}}\left(\sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}),\boldsymbol{y}^{(i)})\right) \tag{13}$$

- Compute velocity update

$$\boldsymbol{v} \leftarrow \alpha\boldsymbol{v} - \epsilon\boldsymbol{g} \tag{14}$$

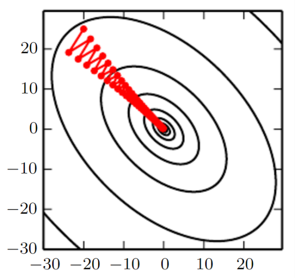
- Apply update

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v} \tag{15}$$

  where
    - α is the momentum coefficient

Adaptive learning rate

- Learning rate is one of the hyperparameter that impacts the most
- The gradient is highly sensitive to some directions
- If we assume that the sensitivity is axis-aligned, it makes sense to use separate rates for each parameter



Source: [Goodfellow et al., 2016]

Delta-bar-delta [Jacobs, 1988]

- Early heuristic approach
- Simple idea: if the partial derivative in respect to one parameter remains the same, increase the learning rate, otherwise, decrease
- Must be used in batch methods

AdaGrad [Duchi et al., 2011]

- Scale the gradient according to the historical norms
- Learning rates of parameters with high partial derivatives decrease fast
- Enforces progress in more gently sloped directions
- Nice properties for convex optimization
- But for deep learning decrease the rate in excess

AdaGrad algorithm

- Accumulate squared gradients

$$\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g} \tag{18}$$

- Element-wise update

$$\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g} \tag{19}$$

- Update parameters

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta} \tag{20}$$

where

- $\boldsymbol{g}$ is the gradient
- δ is a small constant for stabilization

RMSProp [Hinton, 2012]

- Modification of AdaGrad to perform better on non-convex problems
- AdaGrad accumulates since beginning, gradient may be too small before reaching a convex structure
- RMSProp uses an exponentially weighted moving average

RMSProp algorithm

- Accumulate squared gradients

$$\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho)\boldsymbol{g} \odot \boldsymbol{g} \tag{21}$$

- Element-wise update

$$\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g} \tag{22}$$

- Update parameters

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta} \tag{23}$$

where

  - ρ is the decay rate

Adam [Kingma and Ba, 2014]

- Adaptive Moments, variation of RMSProp + Momentum
- Momentum is incorporated directly as an estimate of the first order moment
    - In RMSProp momentum is included after rescaling the gradients
- Adam also add bias correction to the moments

Adam algorithm

- Update time step: $t \leftarrow t + 1$
- Update biased moment estimates

$$\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g} \tag{24}$$

$$\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g} \tag{25}$$

- Correct biases

$$\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1 - \rho_1^t} \tag{26}$$

$$\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1 - \rho_2^t} \tag{27}$$

- Update parameters

$$\Delta\boldsymbol{\theta} \leftarrow -\epsilon \frac{\hat{\boldsymbol{s}}}{\delta + \sqrt{\hat{\boldsymbol{r}}}} \tag{28}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta} \tag{29}$$