06-20416 and 06-12412
(Intro to) Neural Computation

07 – Optimisation Algorithms

Per Kristian Lehre

Last lecture

- A **softmax** output layer allows output nodes to be interpreted as probabilities
- The probabilities indicate the likelihood of a class, given the input and the network
- A naive implementation of the softmax function can be numerically unstable

Outline

- Learning rate in stochastic gradient descent
- Alternatives to standard SGD
  - SGD with momentum
  - SGD with Nesterov momentum
  - AdaGrad
  - Adam

Repetition: Gradient Descent

Input: cost function $J : \mathbb{R}^m \to \mathbb{R}$

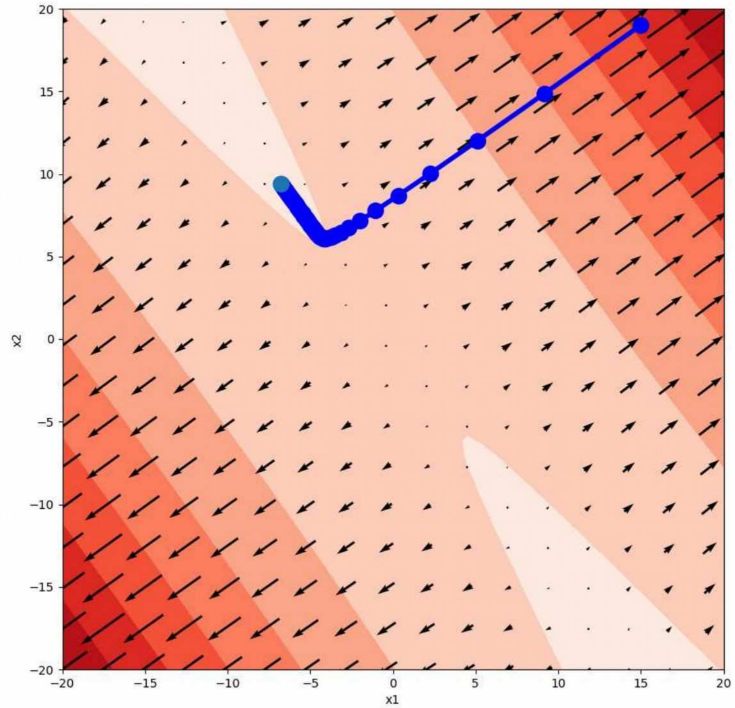
learning rate $\varepsilon \in \mathbb{R}$, $\varepsilon > 0$

$x \leftarrow$ some initial point in $\mathbb{R}^m$

while termination condition not met {

$$x \leftarrow x - \varepsilon \cdot \nabla J(x)$$

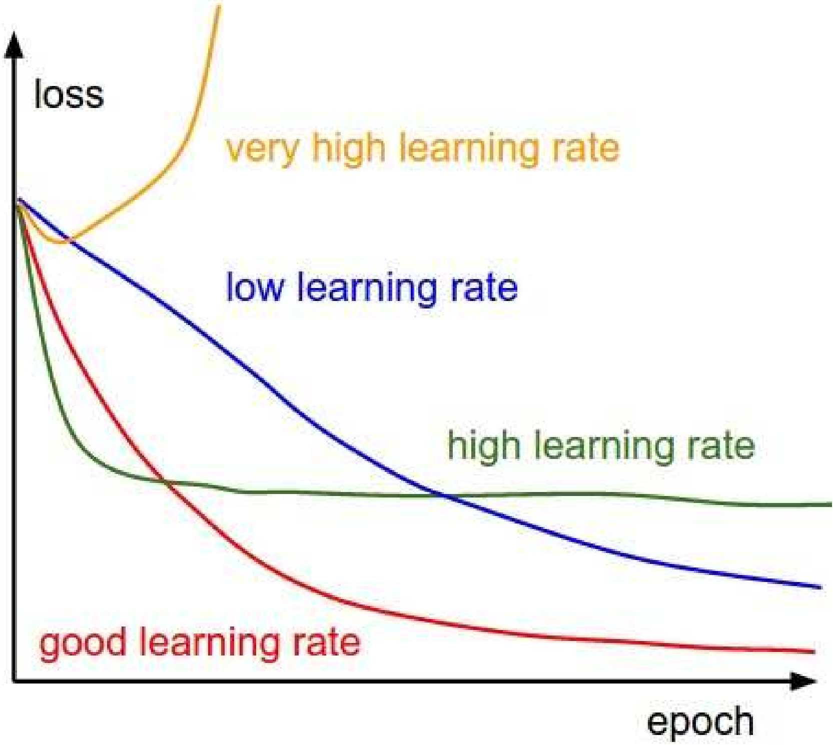
}



Impact of learning rate on SGD

- The learning rate in SGD often strongly impacts the optimisation time
- Often necessary to adjust the learning rate according to the specific setting.

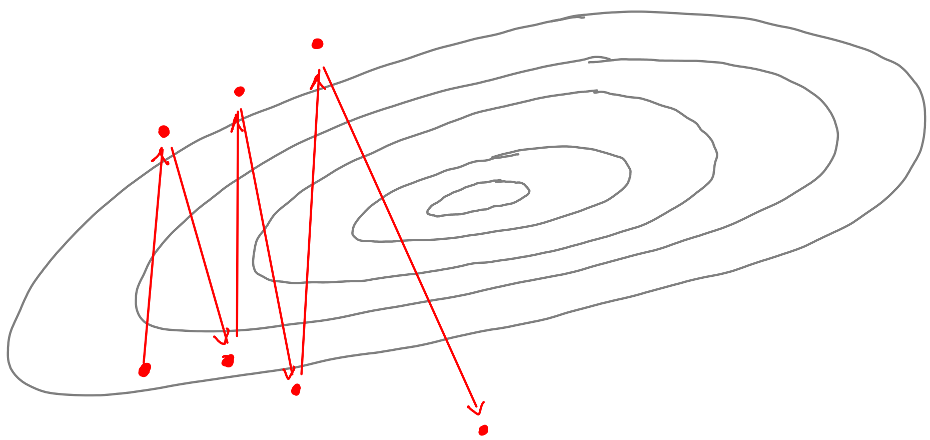
“Cartoon Picture”



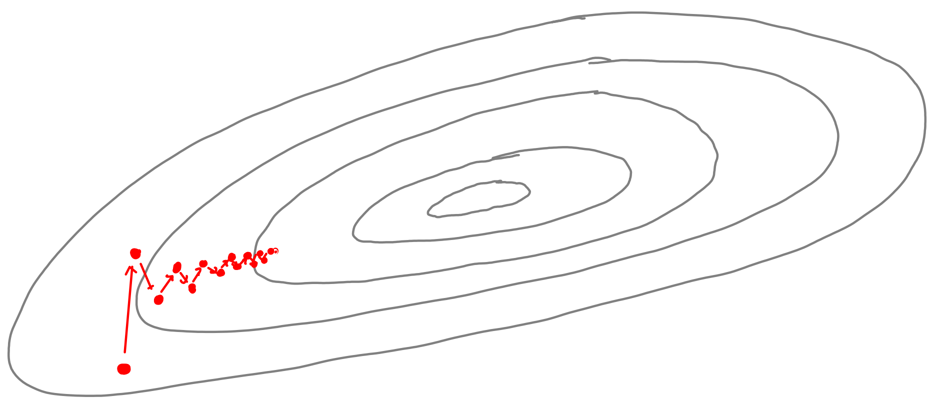
Source: Karpathy, CS231n

Typical Behavior of Standard SGD

Case 1: Too high learning rate ε



Case 2: Too low learning rate ε



Gradient Descent with Momentum

choose an initial parameter θ

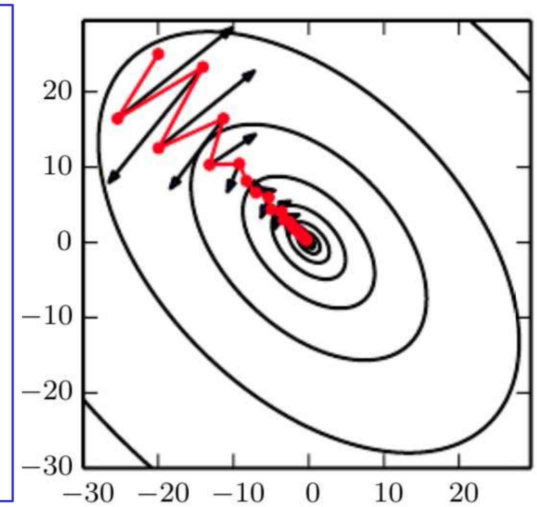
$v = 0$

while termination condition not satisfied {

$$v = \alpha v - \varepsilon \nabla_\theta C(\theta)$$

$$\theta = \theta + v$$

}



Physical interpretation:

A ball with position θ and velocity v influenced by two forces, one which pushes the ball opposite of the current gradient, and a viscuos drag determined by paramete $\alpha < 1$

The momentum "smoothes out" update steps.
The size of updates depends on how aligned the previous gradients are.

Two hyperparameters

- ε learning rate
- α factor which determines the influence of past gradients on the current update of the paramete (often 0.5, 0.9, or 0.99)

Nesterov Momentum

choose an initial parameter θ

$v = 0$

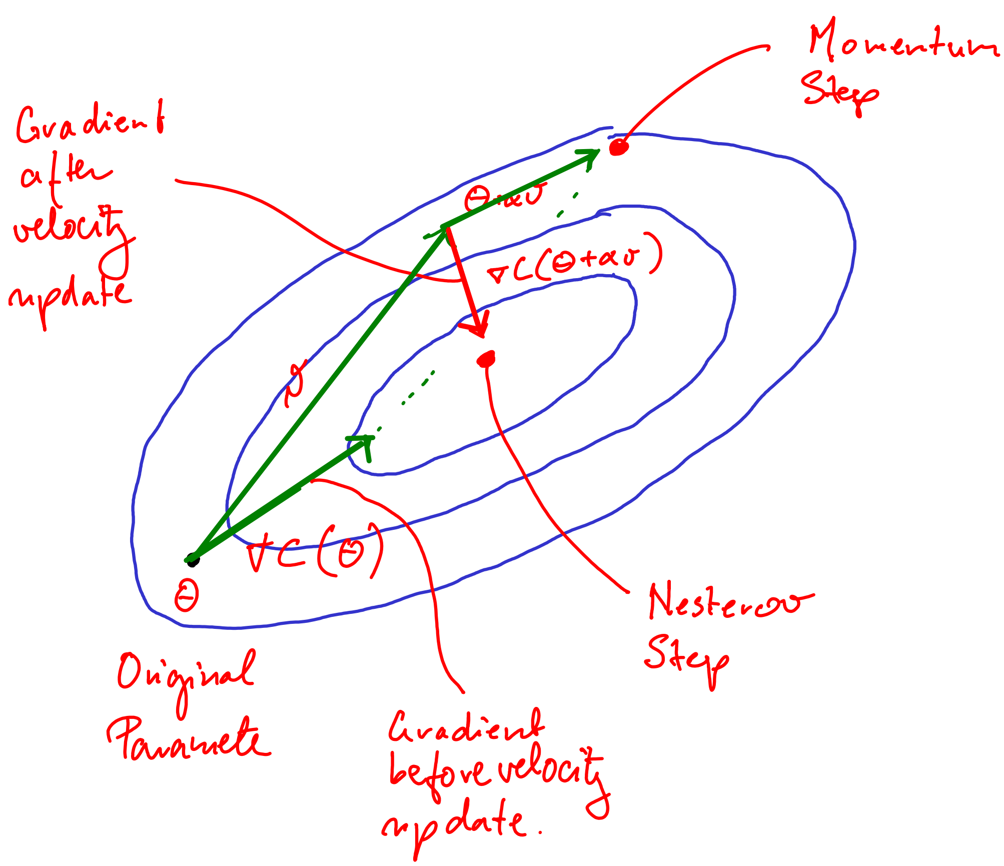
while termination condition not satisfied {

$$v = \alpha v - \varepsilon \nabla_{\theta} C(\theta + \alpha v)$$

$$\theta = \theta + v$$

}

Nesterov momentum is a variant of standard momentum where the gradient is computed after the velocity is applied.



Adagrad (Duchi et al, 2011)

choose an initial parameter Θ

$r = 0$

while termination condition not satisfied {

$$g = \nabla_\Theta C(\Theta)$$

$$r = r + g \odot g$$

squared gradient

$$v = -\frac{\varepsilon}{\delta + \sqrt{r}} \odot g$$

(division and square root applied componentwise)

$$\Theta = \Theta + v$$

}

δ is a hyperparameter, typically $\delta = 10^{-6}$.

AdaGrad adapts a (possibly different) "learning rate" $\frac{\varepsilon}{\delta + \sqrt{r}}$ for each dimension according to accumulated square gradient r

- large r implies small $\frac{\varepsilon}{\delta + \sqrt{r}}$
- small r implies large $\frac{\varepsilon}{\delta + \sqrt{r}}$

AdaGrad works well for problems with sparse gradients.

All gradients (new and old) weighted equally by r.

Adam (Kingma and Ba, 2014)

choose an initial parameter θ

$r = 0$, $s = 0$, $t = 0$

while termination condition not satisfied {

$$t = t + 1$$

$$g = \nabla_\theta C(\theta)$$

$$s = \rho_1 s + (1 - \rho_1) g$$

$$r = \rho_2 r + (1 - \rho_2) g \odot g$$

$$\hat{s} = \frac{s}{1 - \rho_1^t}, \quad \hat{r} = \frac{r}{1 - \rho_2^t}$$

$$v = -\varepsilon \cdot \frac{\hat{s}}{\sqrt{\hat{r}} + \delta}$$

$$\theta = \theta + v$$

}

$$\frac{\text{avg. gradient}}{\sqrt{\text{avg. squared gradient}}}$$

Hyperparameters typically chosen as

$\varepsilon = 0.001$

$\rho_1 = 0.9$

$\rho_2 = 0.999$

$\delta = 10^{-8}$

Widely used method in deep learning.

Summary

- Learning rate has strong impact on SGD
- Alternatives to SGD
  - SGD with momentum
  - SGD with Nesterov momentum
  - AdaGrad
  - Adam
- Open research problem how to choose appropriate algorithms