Deep Learning

Lecture 2

Recap

- Multi-layer perceptron
  - Activations
  - Properties

- Gradient calculation
  - How to calculate gradient?

Neural network optimization

Optimization problem

F some loss function: l2 loss, cross-entropy, etc.

$$F = \frac{1}{n}\sum_{i=1}^{n} f_i(x) \to \min_x, n \gg 1$$

Gradient descent:

Function	Calculation cost
$f_i(x)$	$\mathcal{O}(q)$
$\nabla f_i(x)$	$\mathcal{O}(q)$
$F(x)$	$\mathcal{O}(nq)$
$\nabla F(x)$	$\mathcal{O}(nq)$

Stochastic gradient descent

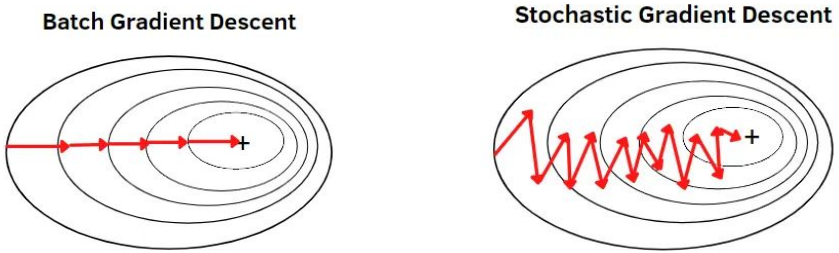
$$i_k \sim Unif(1, 2, \ldots, n)$$

$$g_k = \nabla f_{ik}(x_k)$$

$$x_{k+1} = x_k - \alpha_k g_k$$

$$\mathbb{E} g_k = \nabla F(x_k)$$

The problem is high variance of gradient



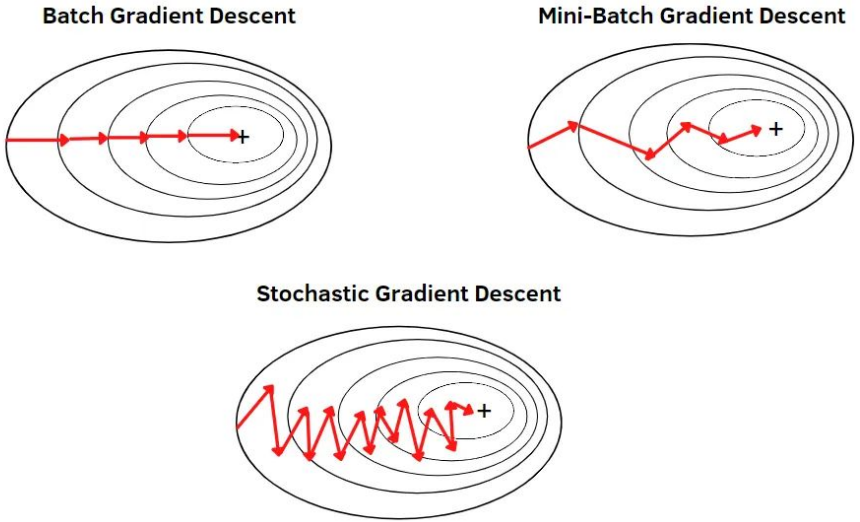
Batch SGD

Instead of one sample we can take several

$$I_k \subset Unif(1, 2, \ldots, n)$$

$$g_k = \frac{1}{|I_k|} \sum_{i \in I_k} \nabla f_{ik}(x_k)$$

$$x_{k+1} = x_k - \alpha_k g_k$$



How stochastic optimization works?

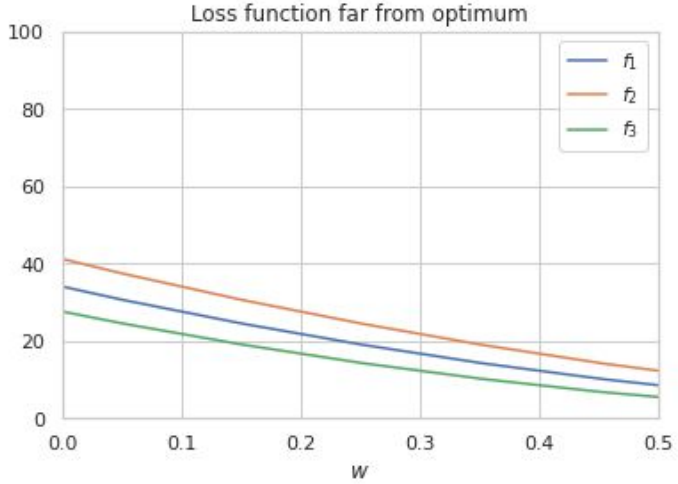
$$F(w) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(y_i - wx_i)^2$$

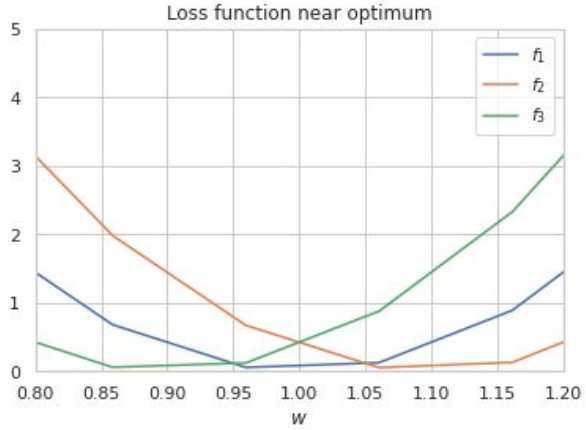
Consider simple example:

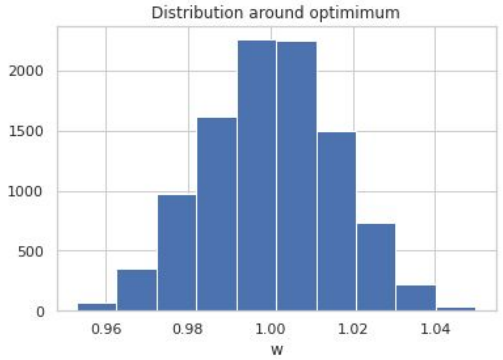
$$y_1(x) = x \qquad y_2(x) = 0.9x$$

$$y_3(x) = 1.1x$$

Convergence







Theorem

The convergence of convex function under some constraints can be explained by the following formula:

$$\mathbb{E}F(\bar{x}_k) - F_{opt} \leqslant \frac{\|x_o - x_{opt}\|^2 + \sum_{i=1}^{k} \alpha_i^2 \mathbb{E}\|g_i\|^2}{2(\sum_{i=1}^{k} \alpha_i)} \qquad \bar{x}_k = \frac{\sum_i \alpha_i x_i}{\sum_i \alpha_i}$$

Proof

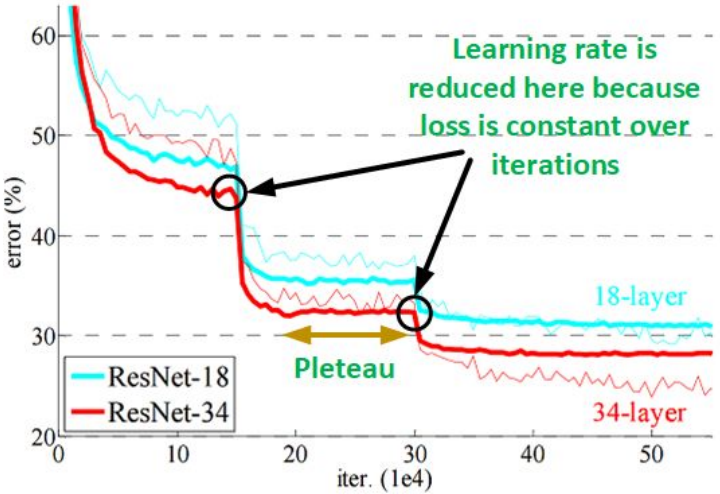
Tradeoff of learning rate

$$\mathbb{E}F(\bar{x}_k) - F_{opt} \leqslant \frac{\|x_o - x_{opt}\|^2 + \sum_{i=1}^{k} \alpha_i^2 \mathbb{E}\|g_i\|^2}{2(\sum_{i=1}^{k} \alpha_i)}$$

Let $R = \|x_0 - x_{opt}\|^2$

$G = \mathbb{E}\|g_i\|^2$

$\alpha_i = h$



Then: $\frac{R^2 + G^2(k+1)h^2}{2(k+1)h} = \frac{R^2}{2(k+1)h} + \frac{G^2 h}{2}$

Learning rate constraints

$$\mathbb{E}F(\bar{x}_k) - F_{opt} \leqslant \frac{\|x_o - x_{opt}\|^2 + \sum_{i=1}^{k} \alpha_i^2 \mathbb{E}\|g_i\|^2}{2(\sum_{i=1}^{k} \alpha_i)}$$

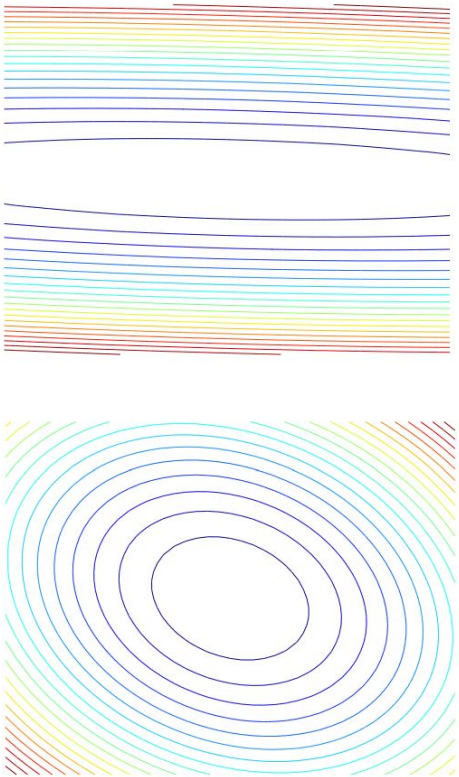
Learning rate constraints

$$\sum_{i=1} \alpha_i = \infty$$

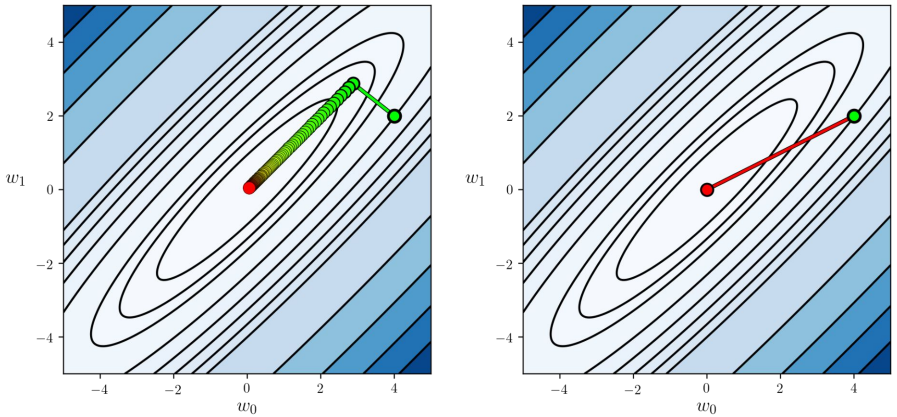
$$\sum_{i=1} \alpha_i^2 < \infty$$

Adaptive methods motivation

$$f(x) = \frac{1}{2}x^\top Ax - x^\top b \to \min_x; A = A^\top$$

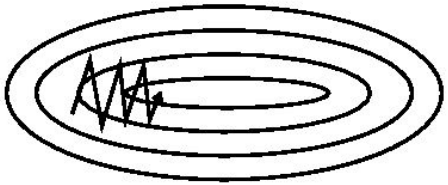


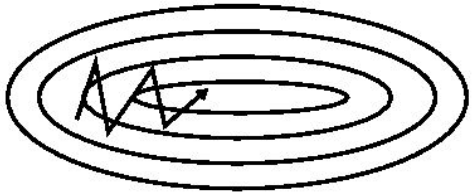
Perfect solution



$$x_{k+1} = x_k - \alpha_k (\nabla^2 f(x_k))^{-1} \nabla f(x_k)$$

Momentum





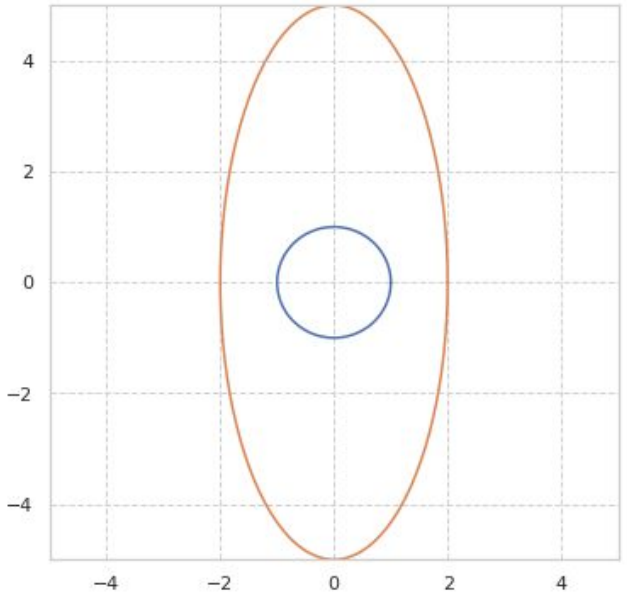
$$x_{k+1} = x_k - \alpha_k g_k + \beta_k (x_k - x_{k-1})$$

Why does momentum work?

AdaGrad

$$x_{k+1,i} = x_{k,i} - \alpha_k \frac{g_{k,i}}{\sqrt{v_{k,i} + \varepsilon}}$$

$$v_{k,i} = \sum_{j=0}^{k} g_{k,i}^2$$



RMSProp

$$x_{k+1,i} = x_{k,i} - \alpha_k \frac{g_{k,i}}{\sqrt{v_{k,i} + \varepsilon}}$$

$$v_{k,i} = \beta v_{k-1,i} + (1 - \beta) g_{k,i}^2$$

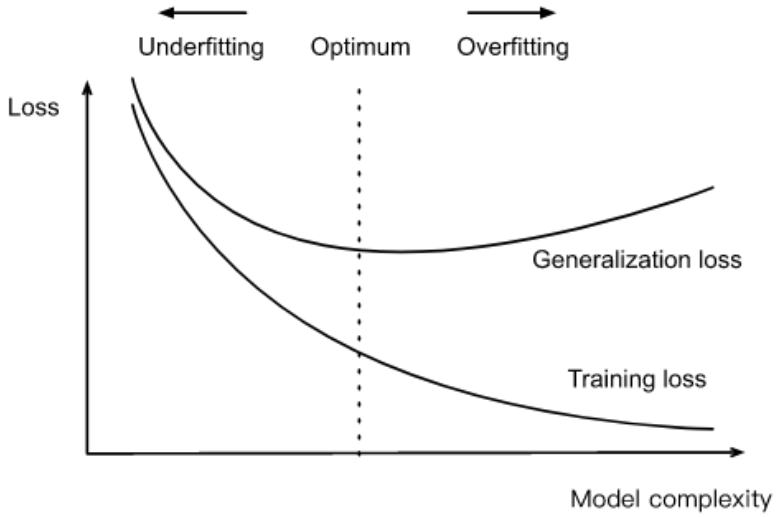
Adam

$$x_{k+1,i} = x_{k,i} - \alpha_k \frac{\mu_{k,i}}{\sqrt{v_{k,i} + \varepsilon}}$$

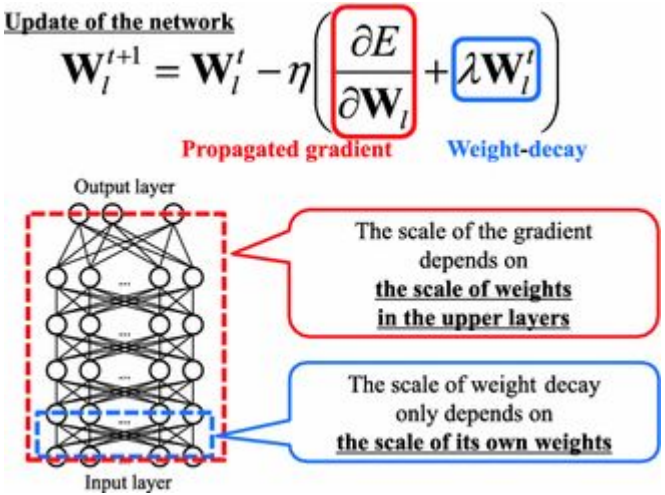
$$v_{k,i} = \beta v_{k-1,i} + (1 - \beta) g_{k,i}^2$$

$$\mu_{k,i} = \beta_2 \mu_{k-1,i} + (1 - \beta_2) g_{k,i}$$

Model regularization



Weight decay

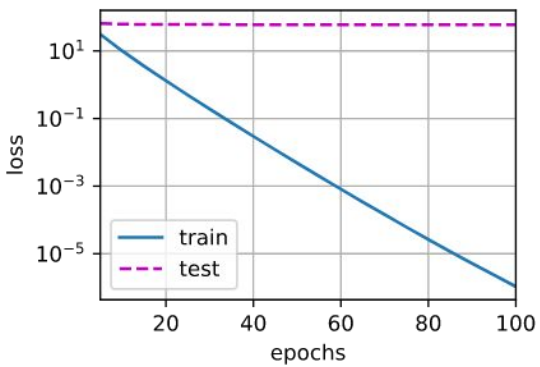


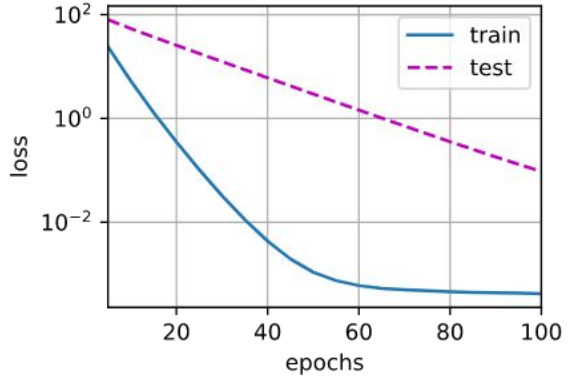
$$L(\mathbf{w}, b) + \frac{\lambda}{2}\|\mathbf{w}\|^2,$$

$$\mathbf{w} \leftarrow (1 - \eta\lambda)\,\mathbf{w} - \frac{\eta}{|\mathcal{B}|}\sum_{i \in \mathcal{B}} \mathbf{x}^{(i)}\left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right).$$

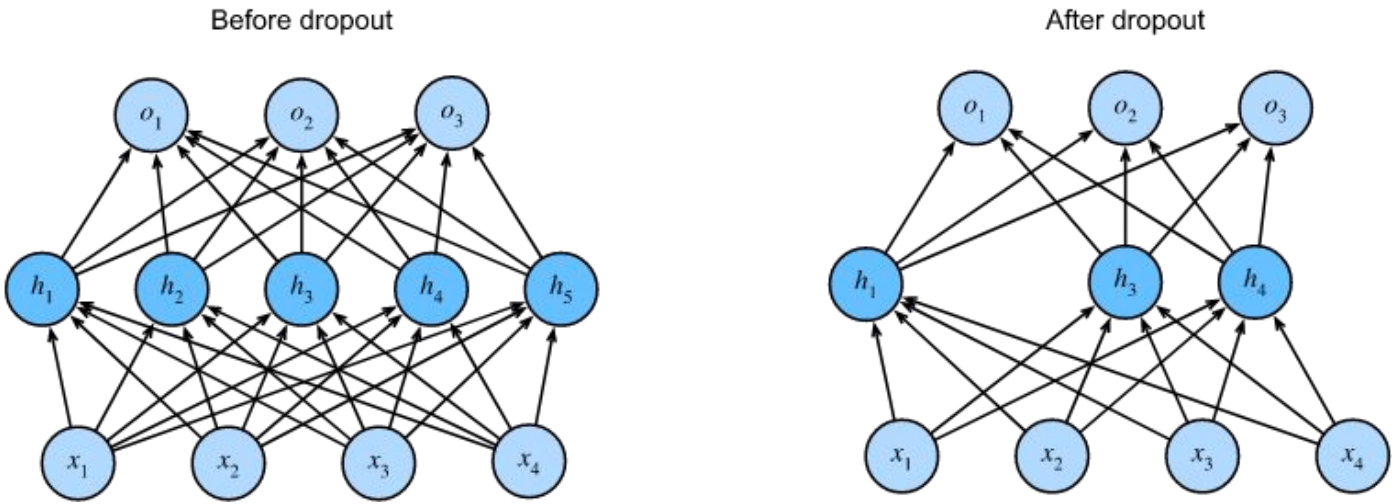
Example

$$y = 0.05 + \sum_{i=1}^{d} 0.01 x_i + \epsilon \text{ where } \epsilon \sim \mathcal{N}(0, 0.01^2).$$





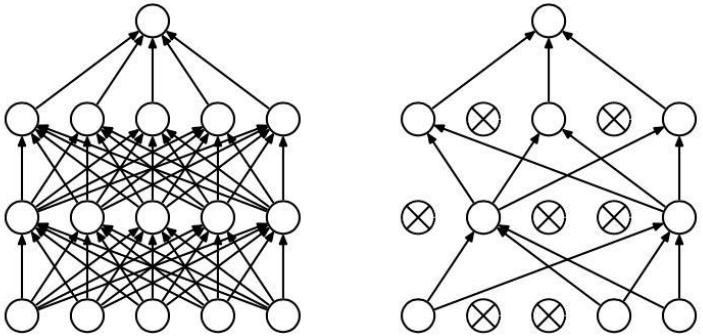
Dropout



Dropout

$$a_i^{DO} = a_i y_i \qquad y_i = \begin{cases} 0 & \text{with probability } p \\ 1 & \text{otherwise} \end{cases}$$

$$\mathbb{E}a_i^{DO} = (1-p)a_i$$



How it should work on the inference?

Recap

- Gradient descent for neural networks
  - Stochasticity
  - Momentum
  - Adaptive methods

- Weight decay
- Dropout