Deep Learning

- Optimization and Gradient Descent -

Eunbyung Park

Assistant Professor

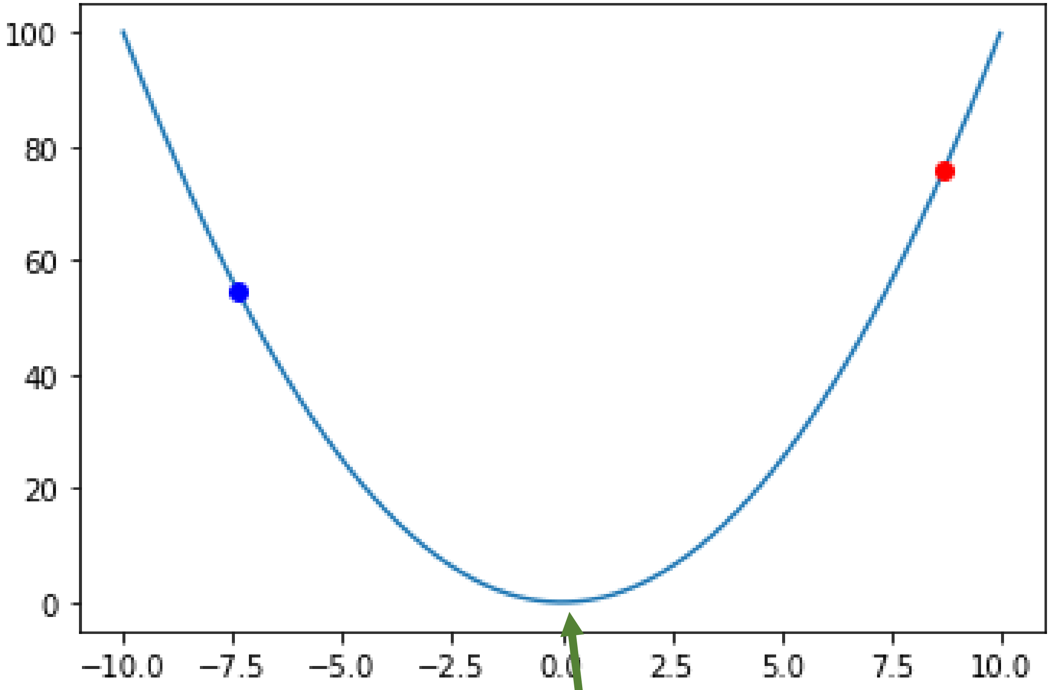
School of Electronic and Electrical Engineering

Eunbyung Park (silverbottlep.github.io)

Gradient Descent

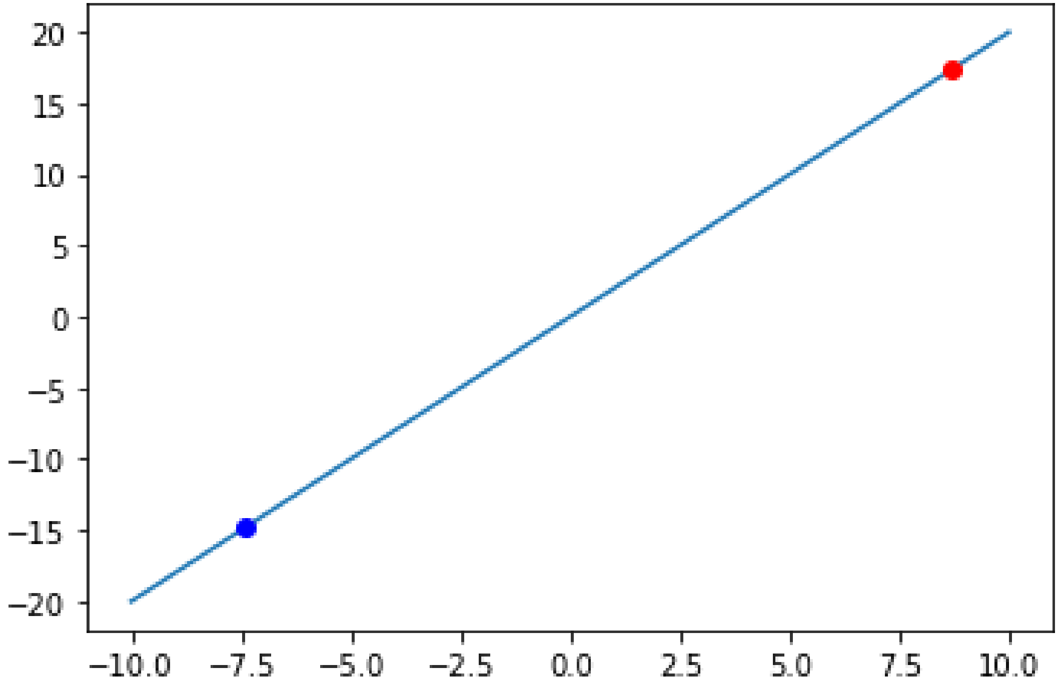
1D Gradient Descent

$$f(x) = x^2$$



$$x^* = 0 = \arg\min_x f(x)$$
$$f(x^*) = 0$$

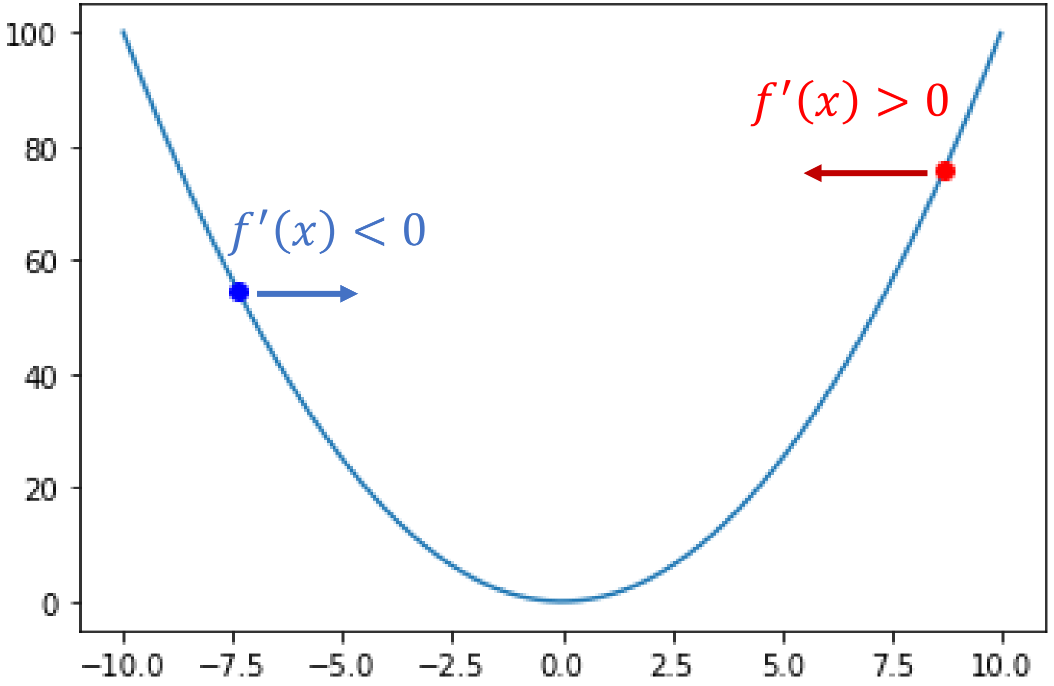
$$f'(x) = 2x$$

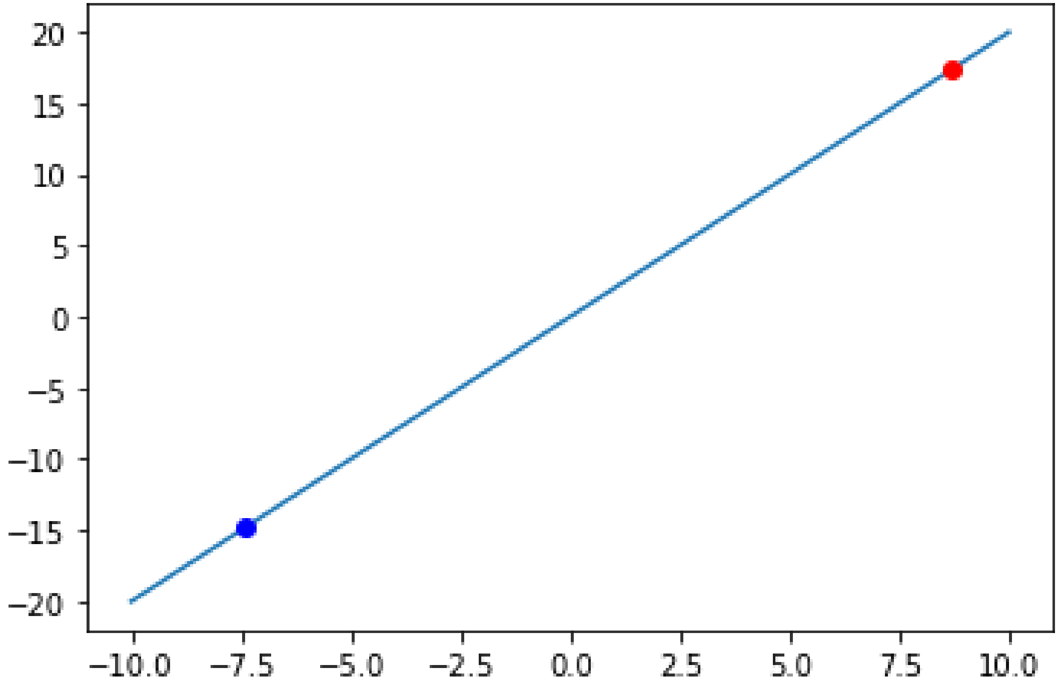


1D Gradient Descent

$$f(x) = x^2$$

$$f'(x) = 2x$$





$$x \leftarrow x - \alpha f'(x)$$

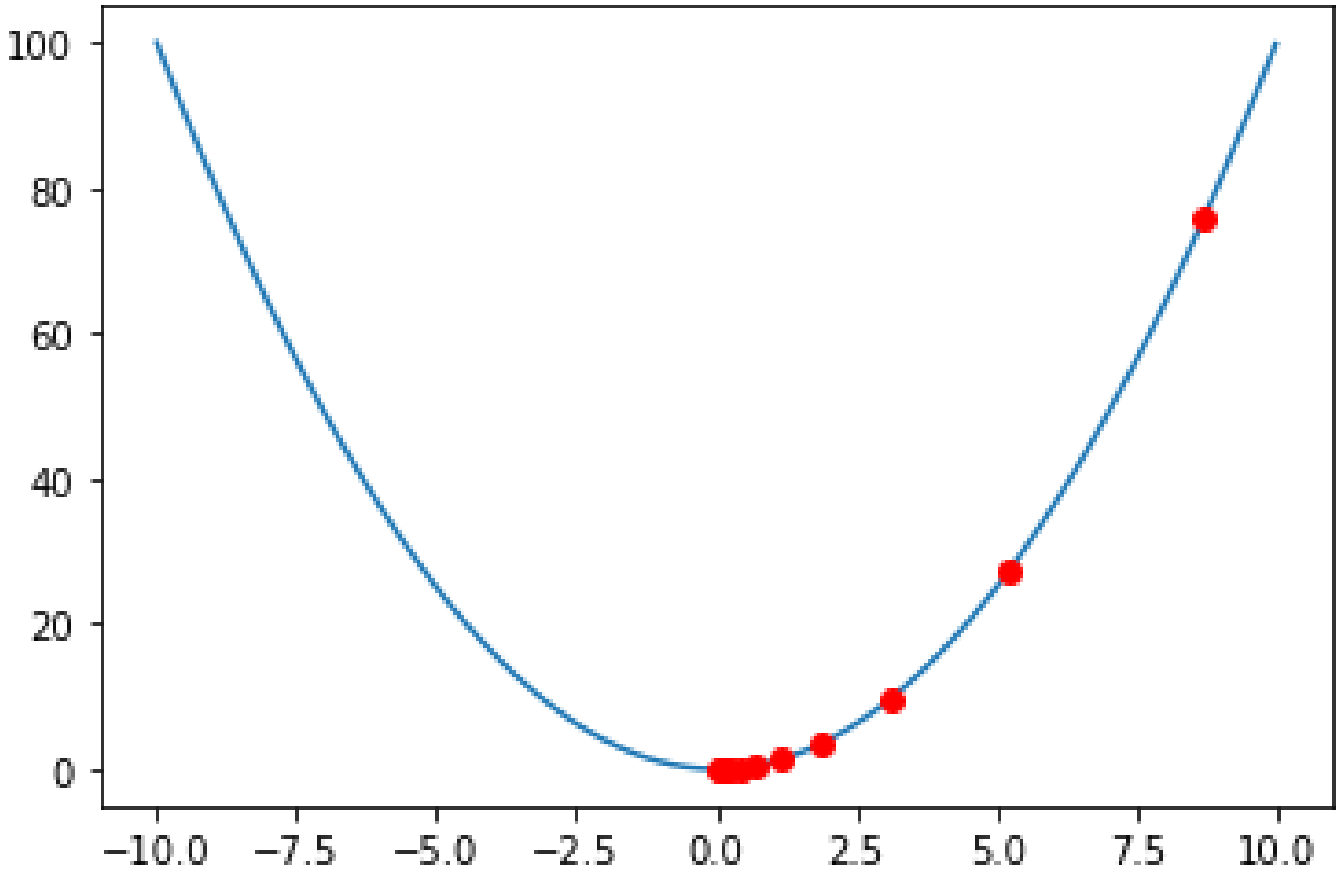
1D Gradient Descent

$$f(x) = x^2$$

$$f'(x) = 2x$$

$$x_0 = 8.7, \alpha = 0.2$$

$$x \leftarrow x - \alpha f'(x)$$



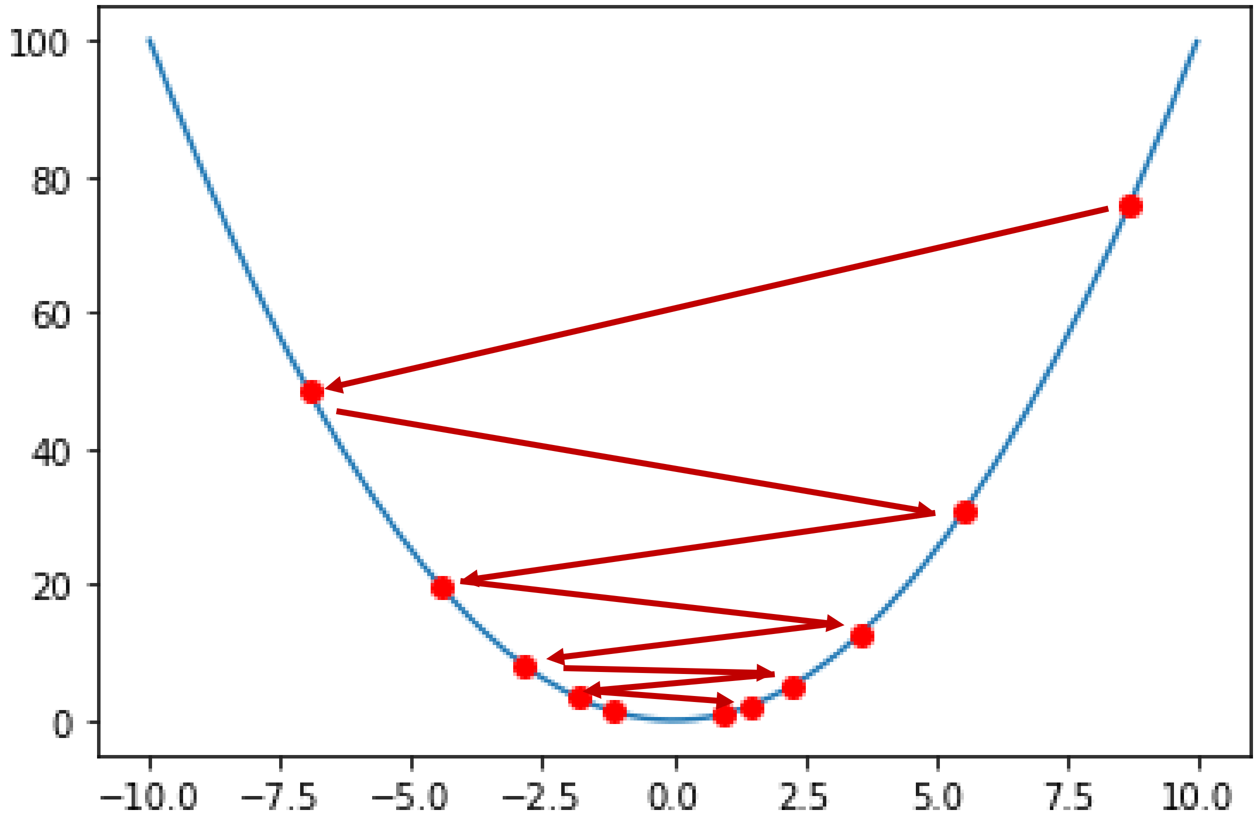
1D Gradient Descent

$$f(x) = x^2$$

$$f'(x) = 2x$$

$$x_0 = 8.7, \alpha = 0.9$$

$$x \leftarrow x - \alpha f'(x)$$



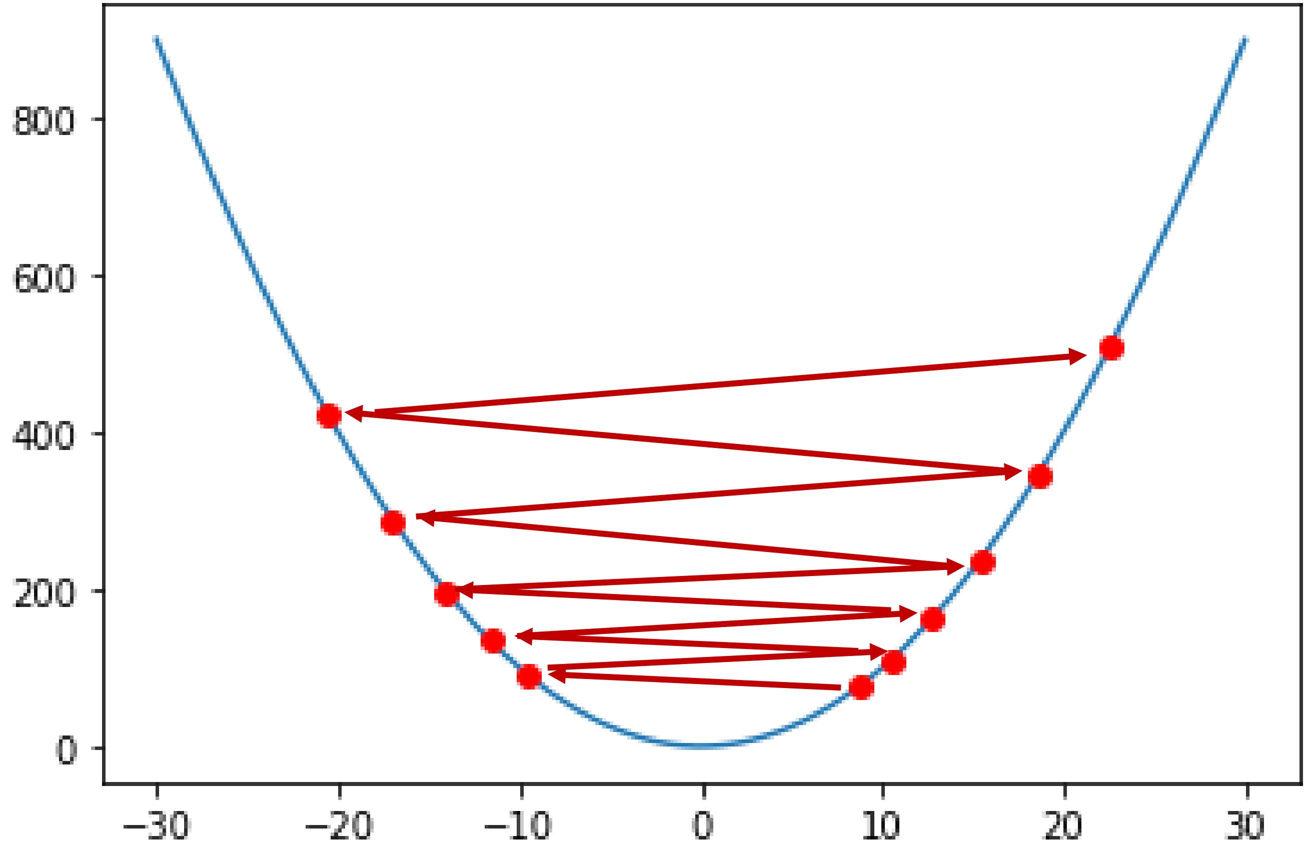
1D Gradient Descent

$$f(x) = x^2$$

$$f'(x) = 2x$$

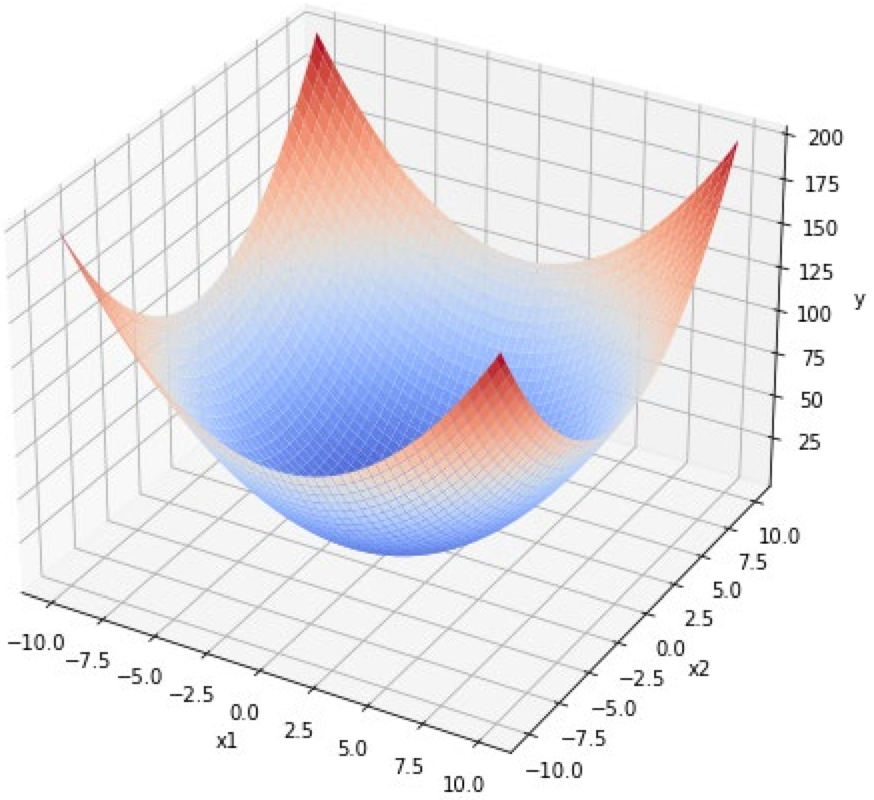
$$x_0 = 8.7, \alpha = 1.05$$

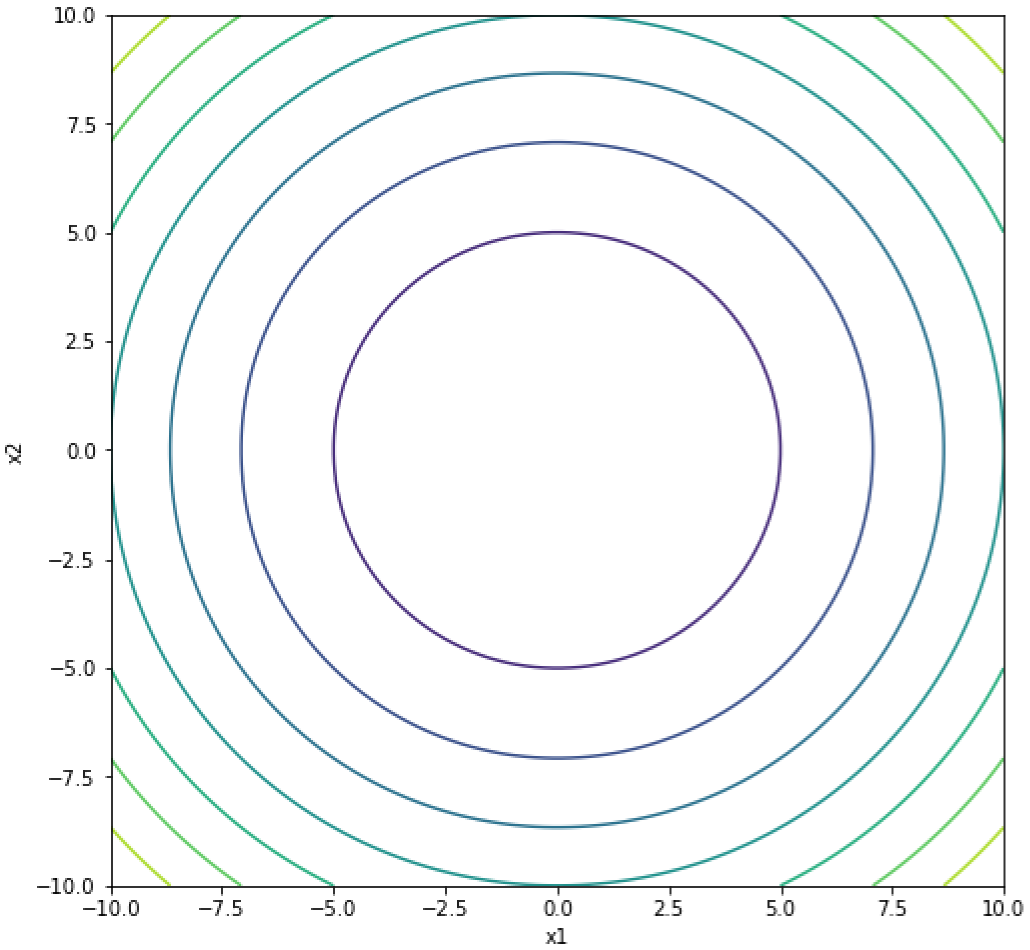
$$x \leftarrow x - \alpha f'(x)$$



2D Gradient Descent

$$f(x_1, x_2) = x_1^2 + x_2^2$$



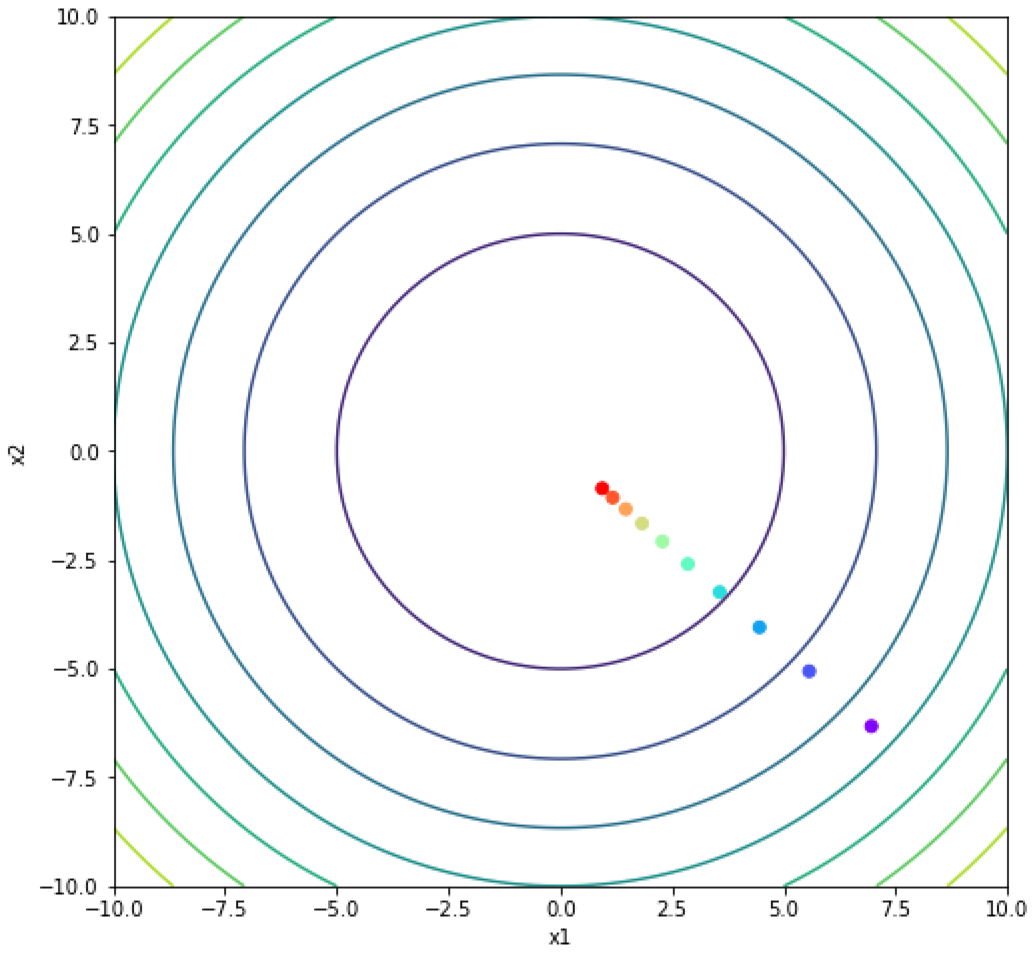


2D Gradient Descent

$$f(x_1, x_2) = x_1^2 + x_2^2$$

$$x_0 = [8.7, -7.9], \alpha = 0.1$$

$$x \leftarrow x - \alpha \nabla f(x)$$

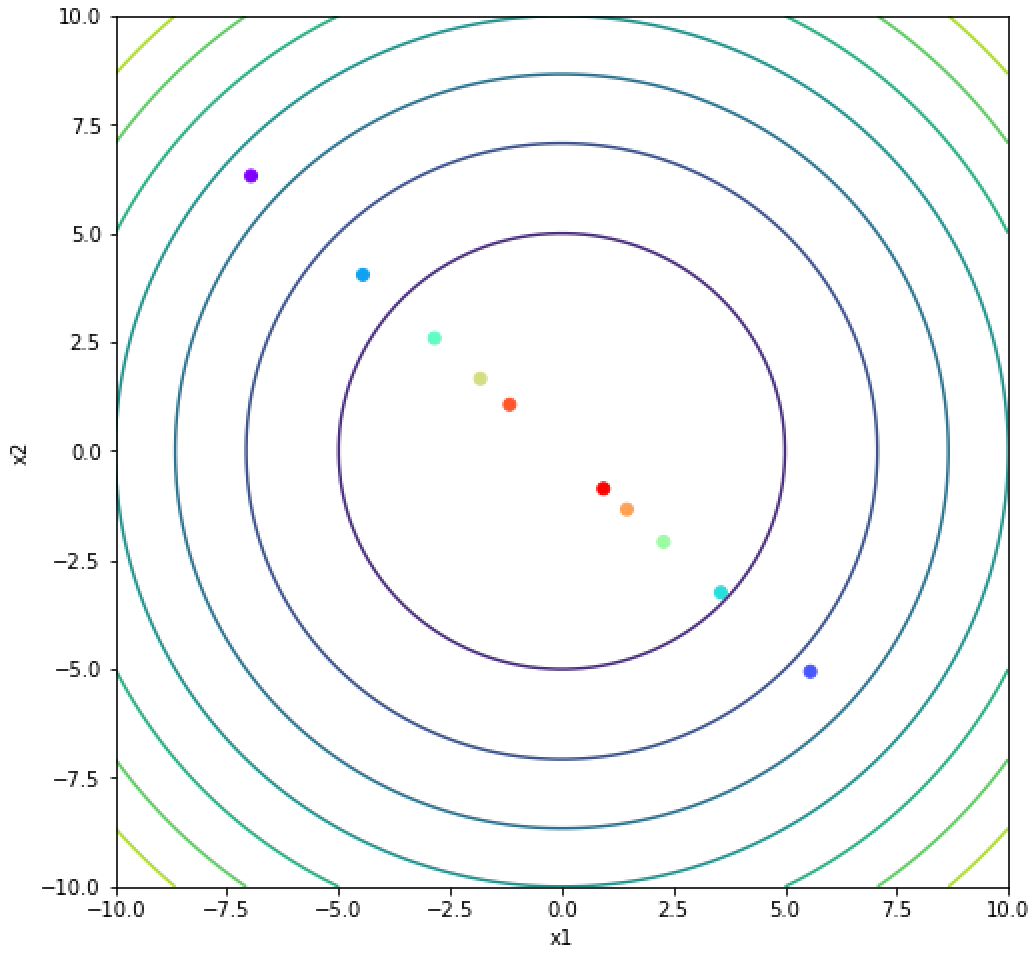


2D Gradient Descent

$$f(x_1, x_2) = x_1^2 + x_2^2$$

$$x_0 = [8.7, -7.9], \alpha = 0.9$$

$$x \leftarrow x - \alpha \nabla f(x)$$

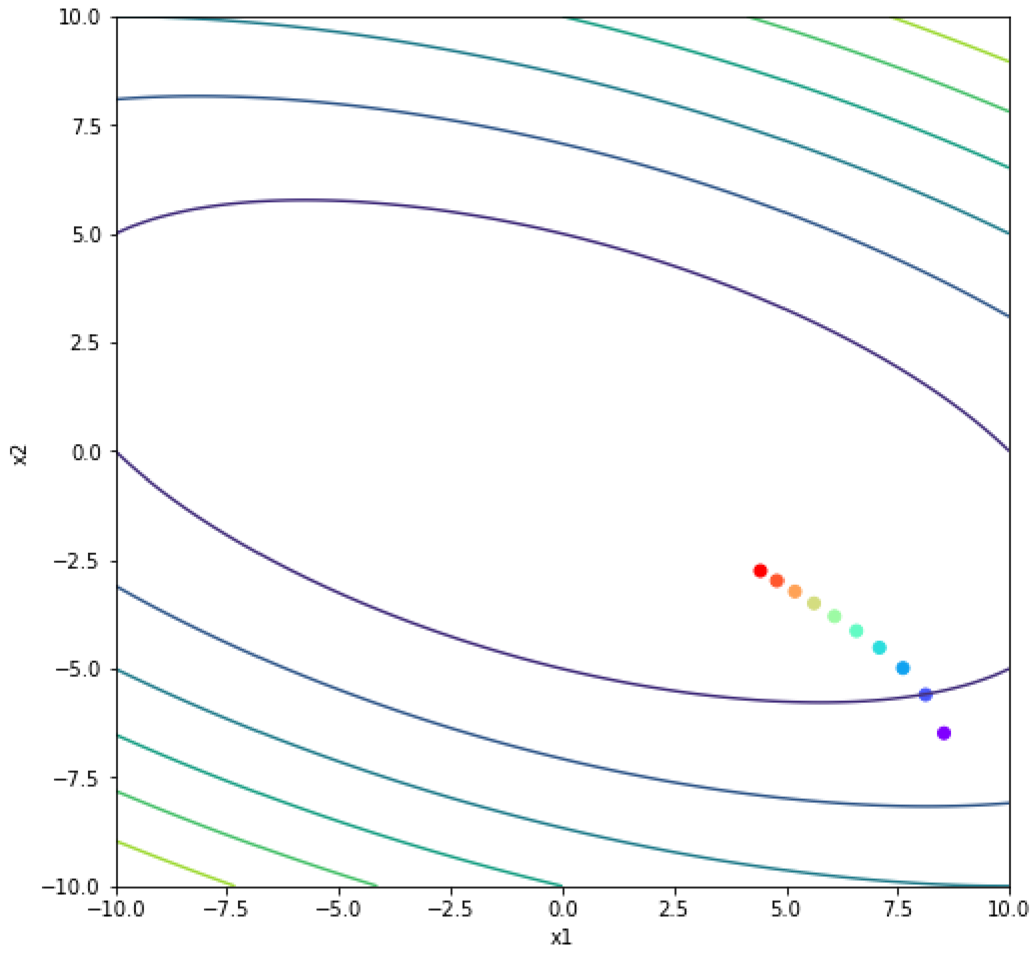


2D Gradient Descent

$$f(x_1, x_2) = 0.5x_1^2 + 2x_2^2 + x_1 x_2$$

$$x_0 = [8.7, -7.9], \alpha = 0.2$$

$$x \leftarrow x - \alpha \nabla f(x)$$



Steepest Descent

- *‘The negative gradient is the direction of steepest descent’*

Directional Derivatives

- The gradient vector is a vector of partial derivatives
- It represents the instantaneous rates of change of the function f w.r.t one of its variables
  - $\frac{\partial f}{\partial x_i}$: how much f changes as x_i change while fixing other components at any given point
- Directional derivative is about how much f changes as all components change together at any given point

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

(gradient)

$$\frac{\partial f}{\partial x}(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0 + h, y_0) - f(x_0, y_0)}{h}$$

(partial derivative)

$$D_u f(x_0, y_0) = \lim_{h \to 0} \frac{f(x_0 + u_1 h, y_0 + u_2 h) - f(x_0, y_0)}{h}$$

(directional derivative)

$u = [u_1, u_2], ||u|| = 1$ (unit vector)

Directional Derivatives

$$D_u f = f_x \quad if\ u = [1,0]$$

$$D_u f = f_y \quad if\ u = [0,1]$$

$$\boxed{D_{u} f(x_0, y_0) = \nabla f(x_0, y_0) \cdot u}$$

$$x = x_0 + hu_1, \qquad y = y_0 + hu_2, \qquad g(h) = f(x_0 + hu_1, y_0 + hu_2)$$

$$g'(0) = \lim_{h\to 0} \frac{g(h) - g(0)}{h} = \lim_{h\to 0} \frac{f(x_0 + hu_1, y_0 + hu_2) - f(x_0, y_0)}{h} = D_u f(x_0, y_0)$$

(by gradient definition)

$$g'(h) = \frac{\partial f}{\partial x}\frac{\partial x}{\partial h} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial h} = f_x(x_0 + hu_1, y_0 + hu_2)u_1 + f_y(x_0 + hu_1, y_0 + hu_2)u_2$$

(multivariate chain rule)

$$g'(0) = f_x(x_0, y_0)u_1 + f_y(x_0, y_0)u_2$$

Directional Derivatives

$$a \cdot b = \|a\|\|b\|\cos(\theta)$$

$$D_u f(x_0, y_0) = \nabla f(x_0, y_0) \cdot u = \|\nabla f(x_0, y_0)\|\|u\|\cos(\theta) = \|\nabla f(x_0, y_0)\|\cos(\theta)$$

- When $\theta = 0, \cos(\theta) = 1, D_u f$ is maximized, u is the direction of steepest ascent

$$u = \frac{\nabla f(x_0, y_0)}{\|\nabla f(x_0, y_0)\|}$$

- When $\theta = \pi, \cos(\theta) = -1, D_u f$ is minimized, u is the direction of steepest descent

$$u = -\frac{\nabla f(x_0, y_0)}{\|\nabla f(x_0, y_0)\|}$$

Taylor Expansion View

- Taylor Expansion

$$f(x+\Delta x) = f(x) + \Delta x \frac{f'(x)}{1!} + \Delta x^2 \frac{f''(x)}{2!} + \Delta x^3 \frac{f'''(x)}{3!} + \cdots$$

$$f(x+\Delta x) \approx f(x) + \Delta x \frac{f'(x)}{1!} + O(\Delta x^2)$$

(First order approximation)

Δx is small

$$f(x+\Delta x) \approx f(x) + \Delta x \frac{f'(x)}{1!} + \Delta x^2 \frac{f''(x)}{2!} + O(\Delta x^3)$$

(Second order approximation)

$$f(x+\Delta x) \approx f(x) + \Delta x^{\mathsf{T}} \nabla f(x) + \Delta x^{\mathsf{T}} \nabla^2 f(x) \Delta x$$

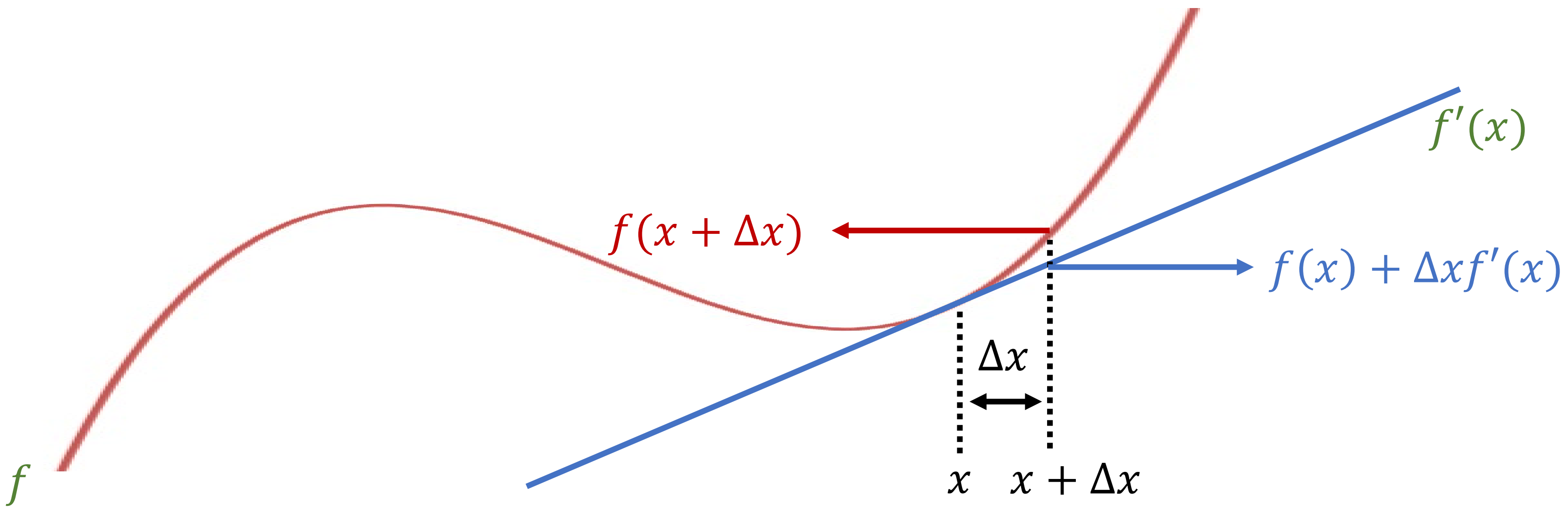
(Multivariable, second order approximation)

Taylor Expansion View

- Taylor Expansion

$$f(x+\Delta x) \approx f(x) + \Delta x \frac{f'(x)}{1!} + O(\Delta x^2)$$

(First order approximation)



Taylor Expansion View

- Taylor Expansion

$$f(x+\Delta x) \approx f(x) + \Delta x f'(x) + O(\Delta x^2)$$

(First order approximation)

One-step gradient descent

$$f(x-\alpha f'(x)) \approx f(x) - \alpha f'^2(x) + O(\alpha^2 f'^2(x))$$

$$\approx f(x) - \alpha f'^2(x) + \cancel{O(\alpha^2 f'^2(x))}$$

$$< f(x)$$

If α is small enough, we can ignore higher-order term

It's descending

Stochastic Gradient Descent

Gradient Descent in Deep Learning

- Batch gradient descent

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} L^{(i)}(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(y^{(i)}, F(x^{(i)}, \theta))$$

N is usually large

$$\nabla L(\theta) = \nabla \frac{1}{N}\sum_{i=1}^{N} L^{(i)}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla L^{(i)}(\theta)$$

θ is usually high dim, e.g. millions

F is computationally expensive

$$\theta \coloneqq \theta - \alpha \nabla L(\theta)$$

Stochastic Gradient Descent

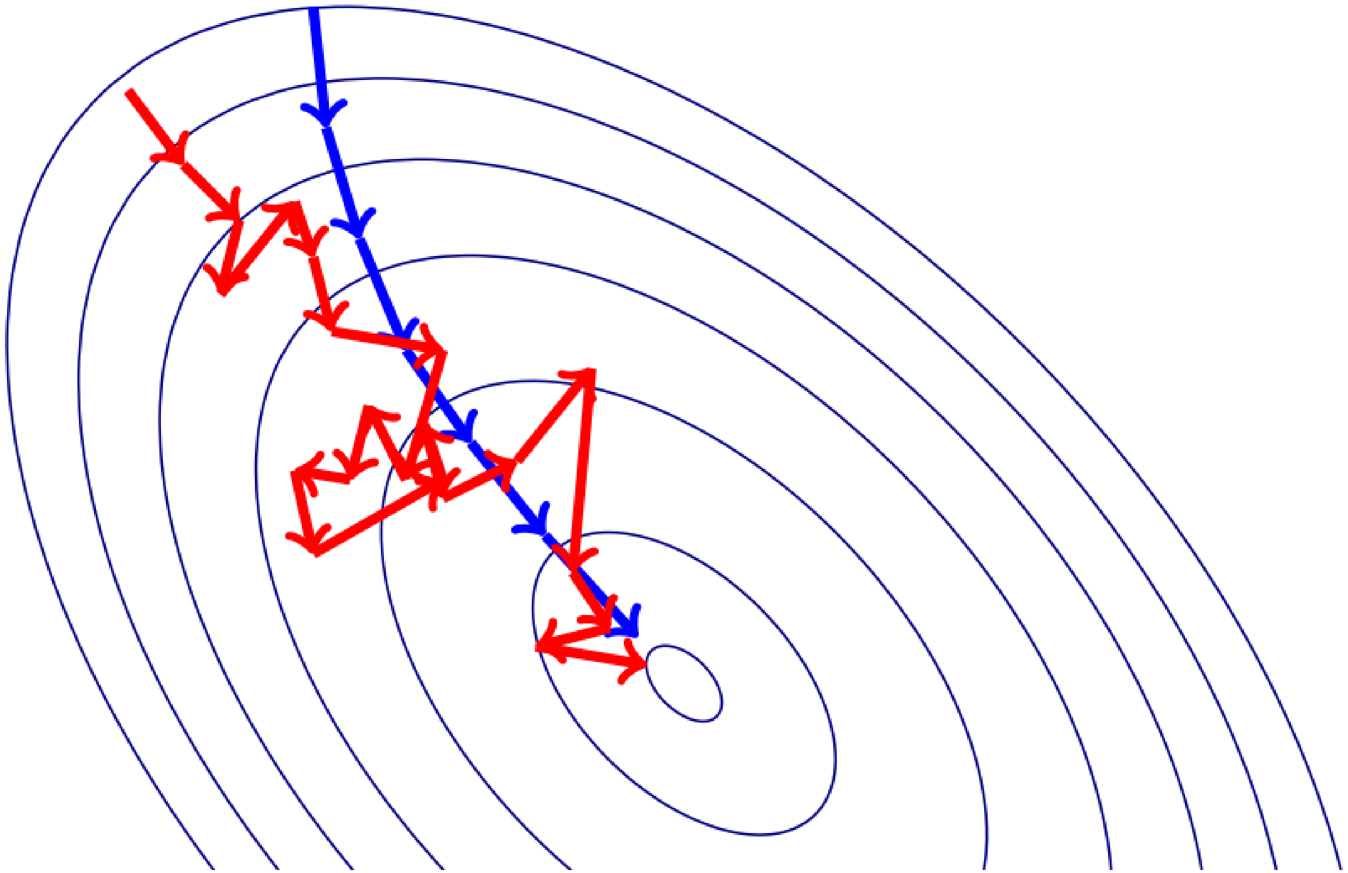
- Estimating gradient given a single training example

$$\theta := \theta - \alpha \nabla L(\theta) \longrightarrow \theta := \theta - \alpha \nabla L^{(i)}(\theta)$$

- It is an unbiased gradient estimation
  - Assuming i is randomly sampled

$$\mathbb{E}_i\left[\nabla L^{(i)}(\theta)\right] = \sum_{i=1}^{N} \nabla L^{(i)}(\theta) p(i) = \sum_{i=1}^{N} \nabla L^{(i)}(\theta) \frac{1}{N} = \frac{1}{N} \sum_{i=1}^{N} \nabla L^{(i)}(\theta) = \nabla L(\theta)$$

Stochastic Gradient Descent



[Lecture 6 Optimization for Deep Neural Networks - CMSC 35246: Deep Learning (ttic.edu)]

Stochastic Gradient Descent

- SGD has higher variance, so slower convergence
- Hard to exploit parallel computations
- Group few training examples, called mini-batch

$$\theta \coloneqq \theta - \alpha \nabla L(\theta)$$

Batch gradient descent

$$\theta \coloneqq \theta - \alpha \nabla L^{(i)}(\theta)$$

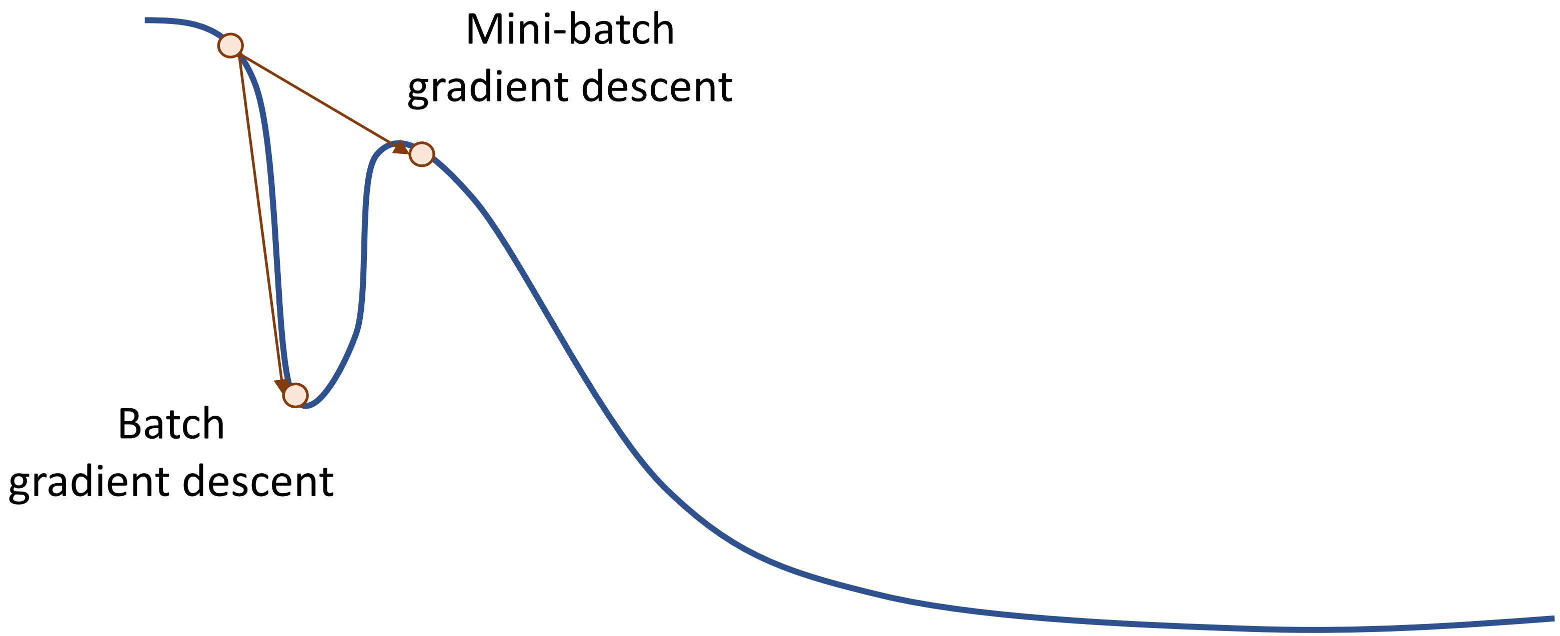
Stochastic gradient descent

$$\theta \coloneqq \theta - \alpha \frac{1}{|B|} \sum_{i \in B} \nabla L^{(i)}(\theta)$$

Mini-batch gradient descent

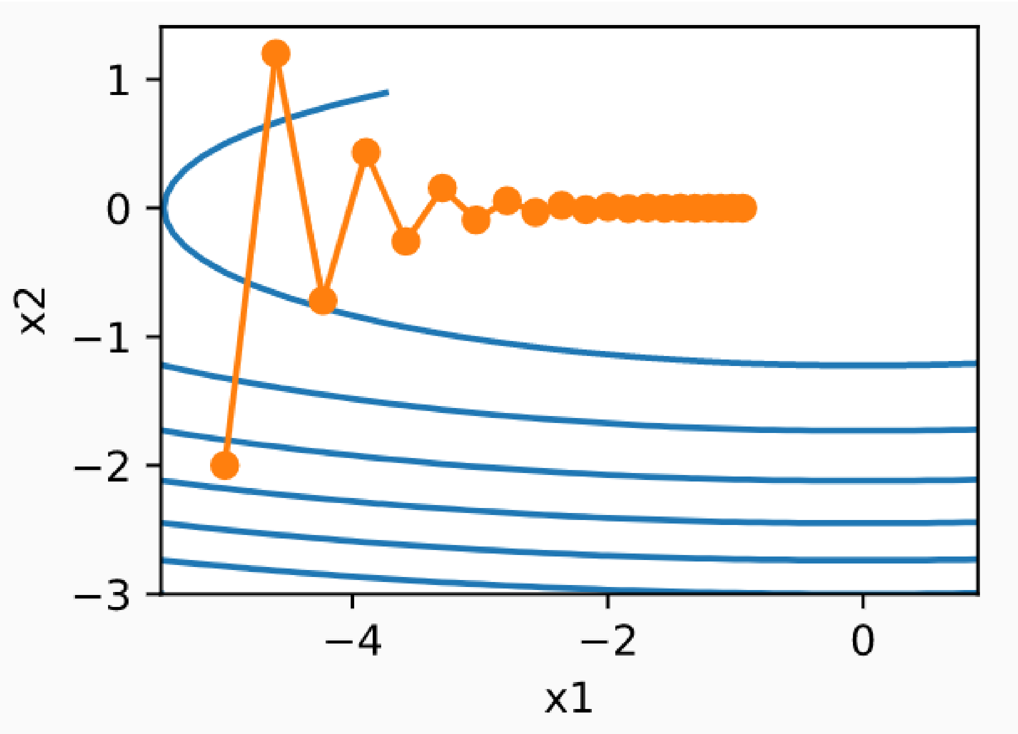
Batch Size

- Small batch has higher gradient noise, which could result in avoiding overfitting to local minima

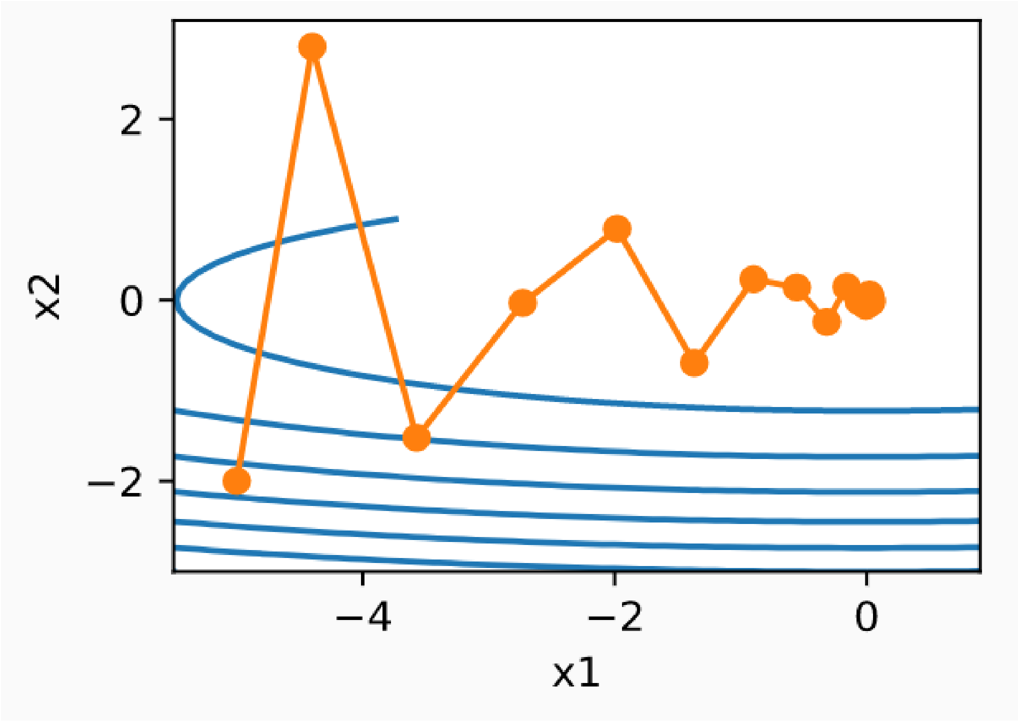


GD with Momentum

- ‘Heavy’ ball rolling down the hill
  - Smoothing the trajectory and accelerating



Gradient descent



Gradient descent w/ momentum

[11.6. Momentum — Dive into Deep Learning 0.17.0 documentation (d2l.ai)](https://d2l.ai)

Exponentially Moving Agerage (EMA)

$$v_t = \beta v_{t-1} + (1-\beta)x_t$$

$$v_{t-1} = \beta v_{t-2} + (1-\beta)x_{t-1}$$

$$v_{t-2} = \beta v_{t-3} + (1-\beta)x_{t-2}$$

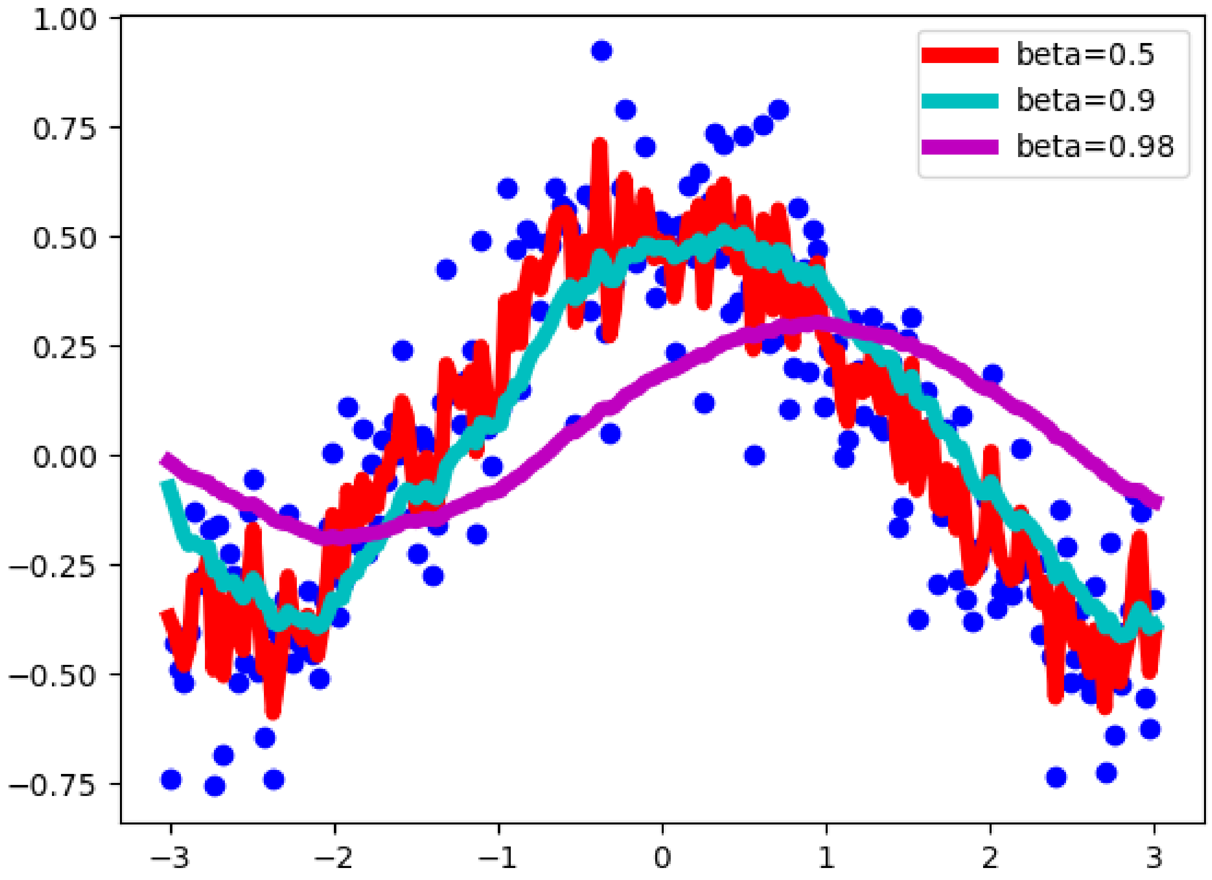


Figure from [Stochastic Gradient Descent with momentum | by Vitaly Bushaev | Towards Data Science]

GD with Momentum

Momentum parameter, higher the more history

$$v_t \leftarrow \beta v_{t-1} + \nabla L(\theta_t)$$

Exponentially moving average of gradients

$$\theta_{t+1} \leftarrow \theta_t - \alpha v_t$$

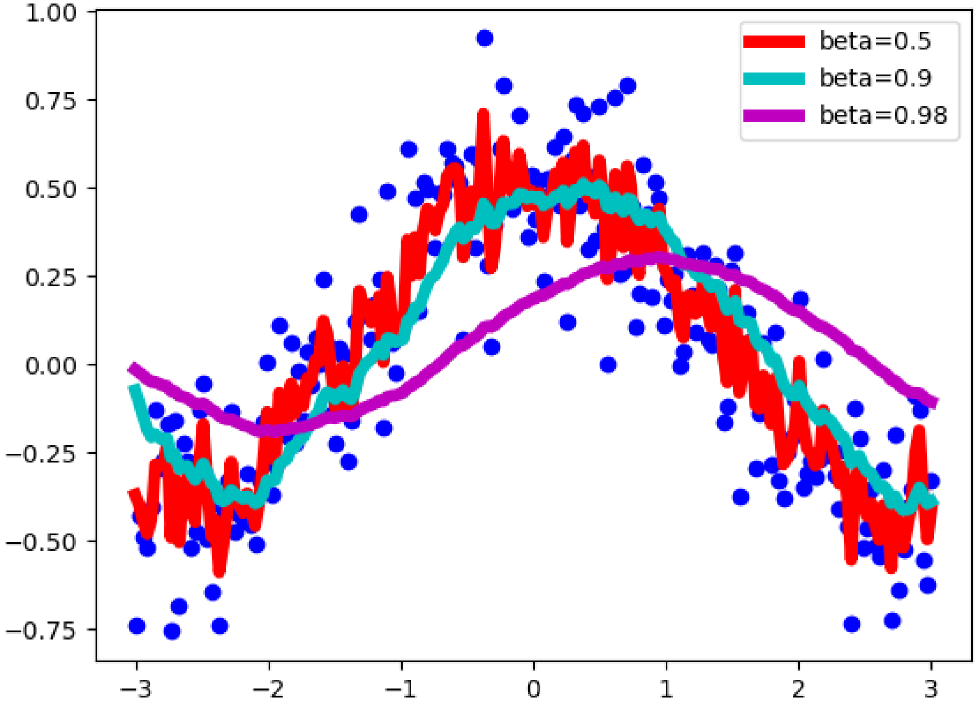


Figure from [Stochastic Gradient Descent with momentum | by Vitaly Bushaev | Towards Data Science]

Adagrad

- Accumulating the history of gradient square
  - The higher, the more gradient magnitude has been observed
  - The smaller, the less gradient magnitude has been observed
- Coordinate-wise learning rates
  - Roughly speaking, each learning rates divided by accumulated gradient
  - The higher past gradient magnitudes, the lower the learning rates

$$s_t \leftarrow s_{t-1} + \nabla L^2(\theta_t)$$

$$\theta_{t+1} \leftarrow \theta_t - \frac{\alpha}{\sqrt{s_t + \epsilon}} \odot \nabla L(\theta_t)$$

RMSprop

- Root-Mean-Square Prop
- Fixing the issue in Adagrad that s_t grows w/o bound

$$s_t \leftarrow \beta s_{t-1} + (1-\beta)\nabla L^2(\theta_t)$$

Exponentially moving average of gradients squares

$$\theta_{t+1} \leftarrow \theta_t - \frac{\alpha}{\sqrt{s_t + \epsilon}} \odot \nabla L(\theta_t)$$

Adam

- Combining ‘momentum’ and ‘RMSprop’ together

$$v_t \leftarrow \beta_1 v_{t-1} + (1 - \beta_1)\nabla L(\theta_t)$$

$$s_t \leftarrow \beta_2 s_{t-1} + (1 - \beta_2)\nabla L^2(\theta_t)$$

$$\theta_{t+1} \leftarrow \theta_t - \frac{\alpha}{\sqrt{s_t + \epsilon}} \odot v_t$$

$\beta_1 = 0.9, \beta_1 = 0.999$

Bias Correction

- If we initialize $v_0 = 0$, then v_t will be smaller in the early training phase

$$v_t \leftarrow \beta_1 v_{t-1} + (1-\beta_1)\nabla L(\theta_t) \qquad \widehat{v_t} = \frac{v_t}{1-\beta_1^t} \qquad \widehat{s_t} = \frac{s_t}{1-\beta_2^t}$$

$$s_t \leftarrow \beta_2 s_{t-1} + (1-\beta_2)\nabla L^2(\theta_t)$$

$$\theta_{t+1} \leftarrow \theta_t - \frac{\alpha}{\sqrt{\widehat{s_t} + \epsilon}} \odot \widehat{v_t}$$

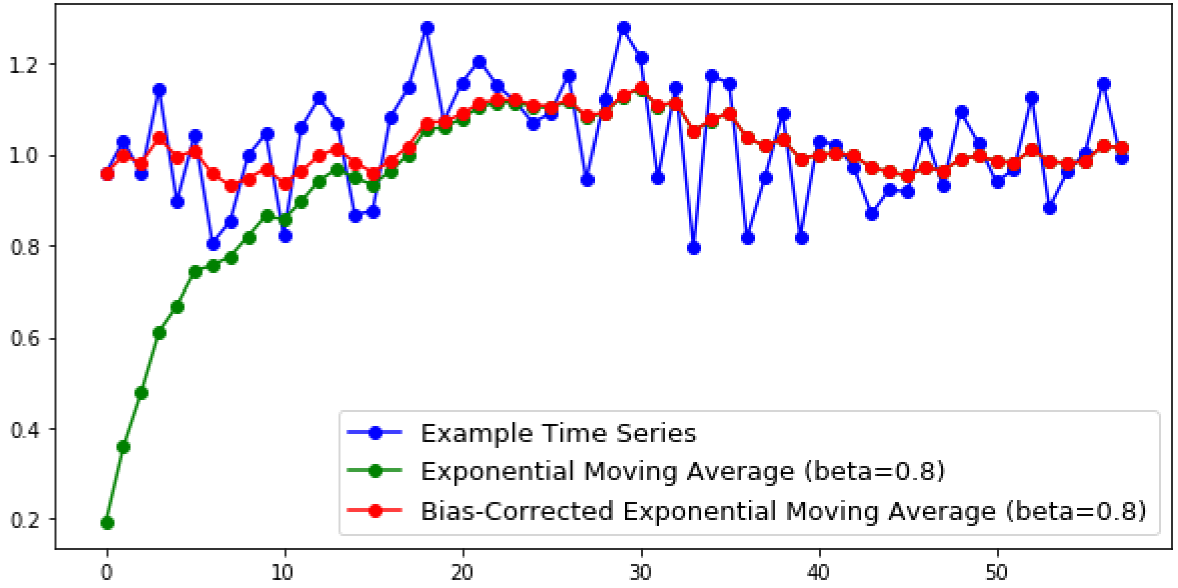
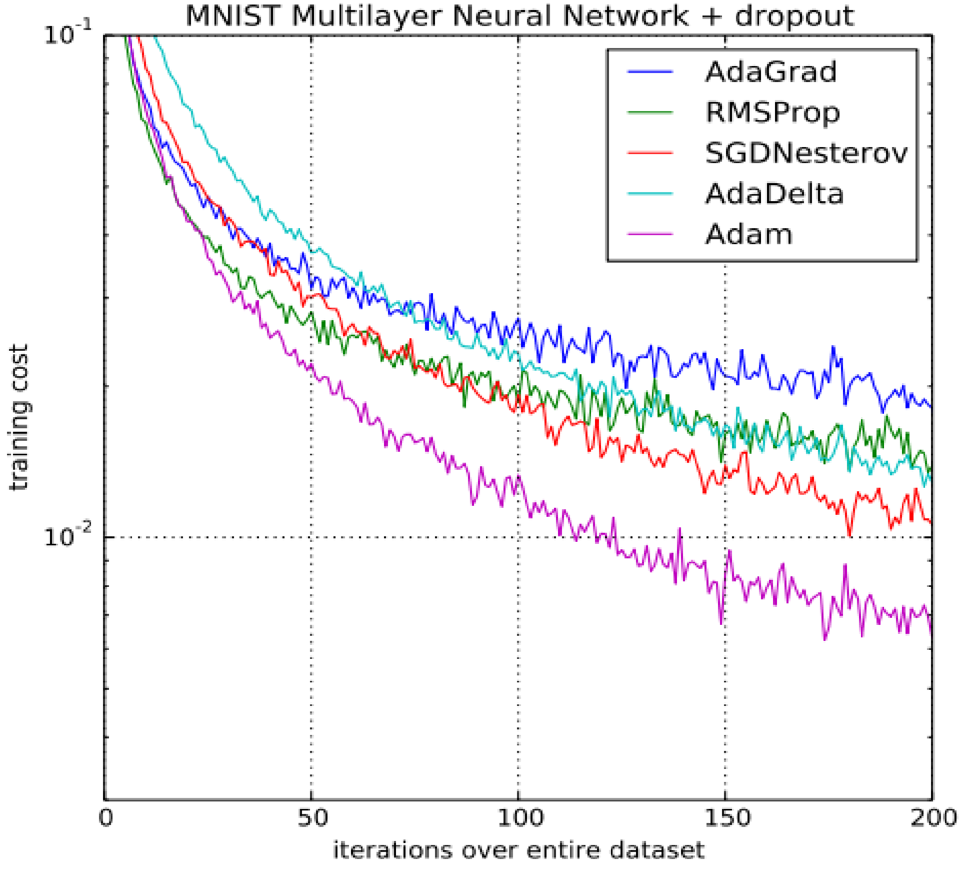
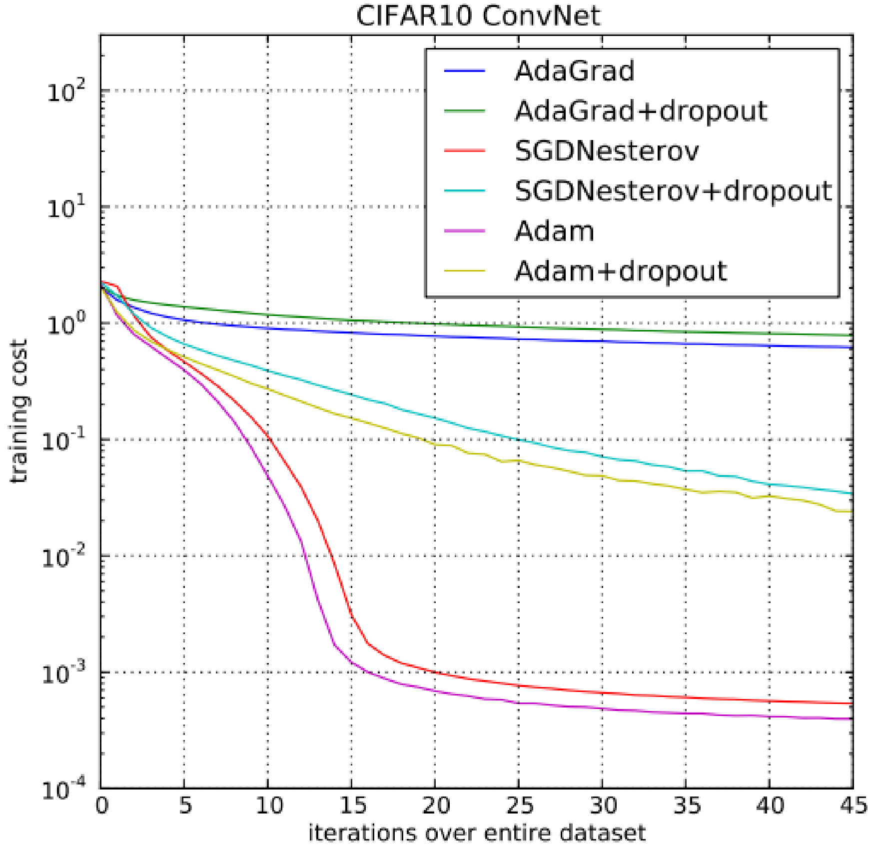


Figure from [Recap of Stochastic Optimization in Deep Learning (part 1) | by Alex Grigorevskiy | Medium]

Example



MNIST Multilayer Neural Network + dropout
AdaGrad
RMSProp
SGDNesterov
AdaDelta
Adam
training cost
10^{-1}
10^{-2}
0
50
100
150
200
iterations over entire dataset



CIFAR10 ConvNet
AdaGrad
AdaGrad+dropout
SGDNesterov
SGDNesterov+dropout
Adam
Adam+dropout
training cost
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
5
10
15
20
25
30
35
40
45
iterations over entire dataset