

ML and Optimization Lecture 14

- Convergence rates for SGD when optimizing nonconvex objectives
- Adaptive Gradient Descent Methods

Convergence rate of SGD (for nonconvex optim probs)

Due to nonconvexity, we measure convergence with

$$\min_{t=0, \dots, T-1} \mathbb{E} \|\nabla f(x_t)\|_2^2 \leq \epsilon$$

called an ϵ -stationary point

Assumptions

- f bounded below
- f is β -smooth
- assume $\mathbb{E} g_t = \nabla f(x_t)$
- $\mathbb{E} \|g_t\|_2^2 \leq \sigma^2$ (could relax to $\mathbb{E} \|\nabla f(x_t) - g_t\|_2^2 \leq \sigma^2$)

Idea: bound

$$\sum_{t=0}^{T-1} \alpha_t \mathbb{E} \|\nabla f(x_t)\|_2^2 \leq B$$

which implies

$$\min_{t=0, \dots, T-1} \mathbb{E} \|\nabla f(x_t)\|_2^2 \sum_{t=0}^{T-1} \alpha_t \leq B, \text{ so}$$

$$\min_{t=0, \dots, T-1} \mathbb{E} \|\nabla f(x_t)\|_2^2 \leq \frac{B}{\sum_{t=0}^{T-1} \alpha_t}$$

Prf

Recall that b/c f is β -smooth,

$$\begin{aligned} f(y) &\leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\beta}{2} \|x - y\|_2^2 \\ \Rightarrow f(x_{t+1}) &\leq f(x_t) + \langle \nabla f(x_t), -\alpha_t g_t \rangle + \frac{\beta}{2} \|\alpha_t g_t\|_2^2 \\ &= f(x_t) - \alpha_t \langle \nabla f(x_t), g_t \rangle + \frac{\alpha_t^2 \beta}{2} \|g_t\|_2^2 \end{aligned}$$

Take expectations (b/c x_t, g_t are random vectors)

$$\mathbb{E} f(x_{t+1}) \leq \mathbb{E} f(x_t) - \alpha_t \mathbb{E} \|\nabla f(x_t)\|_2^2 + \frac{\alpha_t^2 \beta \sigma^2}{2}$$

this implies

$$\alpha_t \mathbb{E} \|\nabla f(x_t)\|_2^2 \leq \mathbb{E} [f(x_t) - f(x_{t+1})] + \frac{\alpha_t^2 \beta \sigma^2}{2}$$

By summing these inequalities from $t=0$ to $t=T-1$, we see

$$\sum_{t=0}^{T-1} \alpha_t \mathbb{E} \|\nabla f(x_t)\|_2^2 \leq \underbrace{\sum_{t=0}^{T-1} \mathbb{E} [f(x_t) - f(x_{t+1})]}_{\text{is a telescoping sum}} + \sum_{t=0}^{T-1} \frac{\alpha_t^2 \beta \sigma^2}{2}$$

$$\begin{aligned} &= f(x_0) - \mathbb{E} f(x_T) + \frac{\beta \sigma^2}{2} \sum_{t=0}^{T-1} \alpha_t^2 \\ &\leq f(x_0) - f(x_*) + \frac{\beta \sigma^2}{2} \sum_{t=0}^{T-1} \alpha_t^2 \end{aligned}$$

so finally

$$\underbrace{\min_{t=0, \dots, T-1} \mathbb{E} \|\nabla f(x_t)\|_2^2}_{:= \text{gap}(T)} \leq \frac{f(x_0) - f(x_*)}{\sum_{t=0}^{T-1} \alpha_t} + \frac{\frac{\beta \sigma^2}{2} \sum_{t=0}^{T-1} \alpha_t^2}{\sum_{t=0}^{T-1} \alpha_t}$$

Choices of step-sizes:

$$\begin{aligned} - \alpha_t = \alpha \text{ then } \text{gap}(T) &\leq O\left(\frac{f(x_0) - f(x^*)}{T\alpha} + \frac{\beta\sigma^2\alpha}{2}\right) \\ &= O\left(\frac{1}{T\alpha} + \beta\sigma^2\alpha\right) \end{aligned}$$

$$- \alpha_t = \frac{\alpha}{\sqrt{t+1}} \text{ then}$$

$$\sum_{t=0}^{T-1} \alpha_t = \sum_{t=0}^{T-1} \frac{\alpha}{\sqrt{t+1}} = O(\sqrt{T})$$

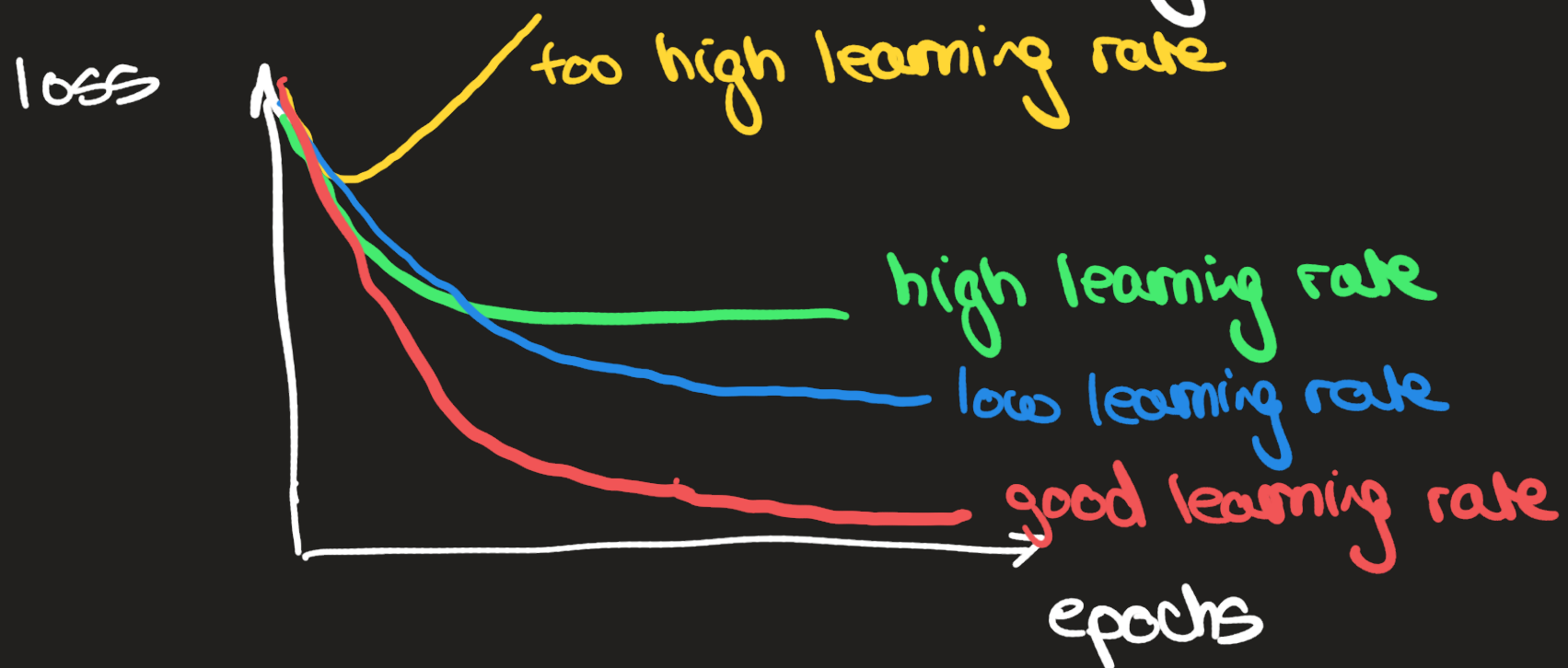
$$\Rightarrow \text{gap}(T) \leq O\left(\frac{f(x_0) - f(x^*)}{\sqrt{T}} + \frac{\beta\sigma^2 \ln T}{\sqrt{T}}\right)$$

$$\text{and } \sum_{t=0}^{T-1} \alpha_t^2 = \sum_{t=0}^{T-1} \frac{\alpha^2}{t+1} = O(\ln T)$$

$$= O\left(\frac{\ln T}{\sqrt{T}}\right)$$

Tips & Tricks for SGD (Bottou 2012)

- How to choose stepsizes, in practice:
 - use a stepsize schedule of the form $\alpha_t = \frac{\beta_0}{1 + \beta_1 t}$ where β_0 and β_1 should be optimized
 - use a small subset of your training data to test the convergence behavior (look at the model accuracy, not the training loss)



Adaptive (Sub)gradient methods

Recall advantages of first-order methods:

- **scalable** much cheaper per iteration than 2nd order methods, don't require solving linear systems, use less memory
- **easily randomizable** very easy to replace gradient w/ a randomized estimate to further reduce the cost per iteration
- **generally applicable** does not require differentiability or strong convexity

Disadvantages:

- **sensitive to hyperparameter choices** (stepsize in particular)
- **do not model curvature of the objective** (slower convergence than 2nd order methods)

Question: can we design first-order methods which are more insensitive to the hyperparameter choices and better model curvature?

Tools:

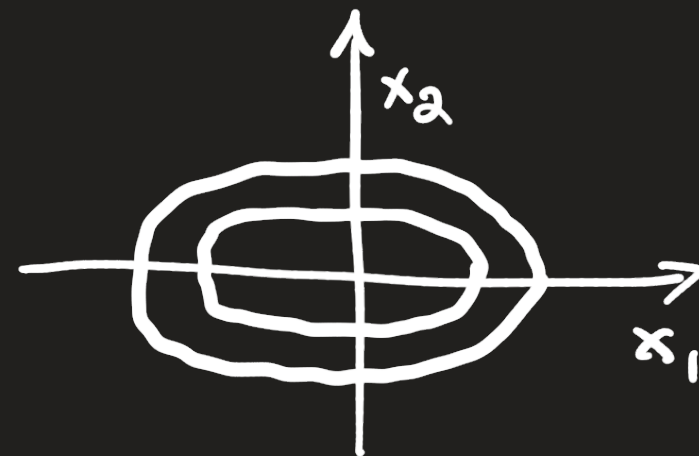
- **momentum**: keep moving in directions that historically made progress
- **exponential averaging**: forget history judiciously to balance between the current local model and the global information from history
- **preconditioning**: cheaply modeling curvature using just first order information

Adaptive GD methods

- 1964 – SGD w/ momentum (Polyak's heavy ball method)
- 1983 – Nesterov's accelerated gradient descent (NAG)
- 2014 – AdaGrad
- 2012 – RMSProp
- 2012 – AdaDelta
- 2015 – ADAM

Ex

$$f(x) = \frac{1}{2} \|Dx\|_2^2 \quad D = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}$$
$$= \frac{1}{2} x_1^2 + 5x_2^2$$



$$x_* = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \nabla f(x) = D^2 x = \begin{bmatrix} x_1 \\ 100x_2 \end{bmatrix}$$

with GD:

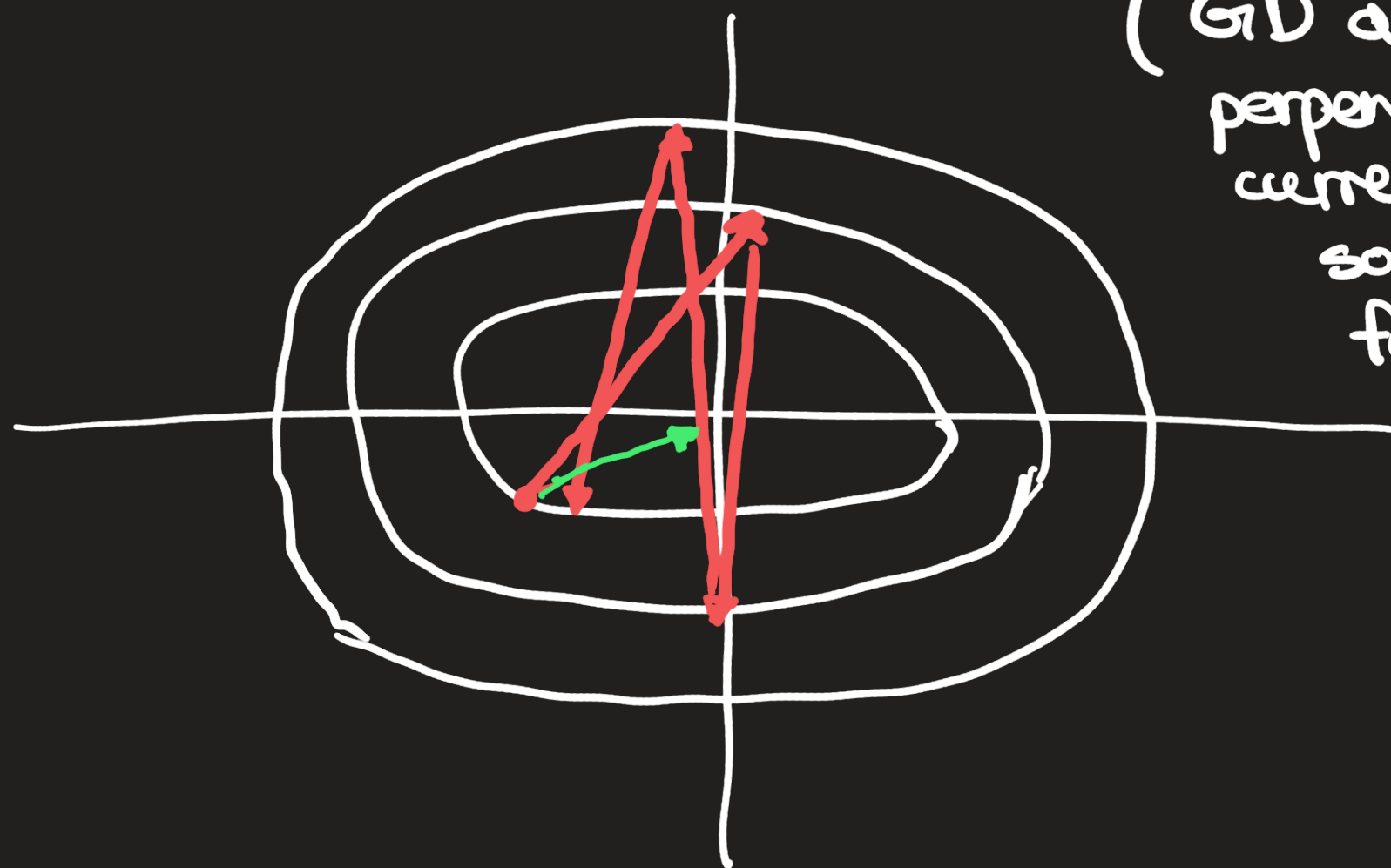
$$x_{t+1} = x_t - \alpha D^2 x_t = (I - \alpha D^2) x_t = \dots = (I - \alpha D^2)^{t+1} x_0$$
$$= \begin{bmatrix} (1-\alpha)^{t+1} & 0 \\ 0 & (1-\alpha 100)^{t+1} \end{bmatrix} \begin{bmatrix} x_{0,1} \\ x_{0,2} \end{bmatrix}$$

The learning from theory that guarantees linear convergence is $\alpha = \frac{1}{100}$
Note this is controlled by the coordinate w/ the highest curvature

$$\Rightarrow x_t = \begin{bmatrix} (99/100)^t & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_{0,1} \\ x_{0,2} \end{bmatrix} \quad \text{so } x_t \rightarrow x^* \text{ very slowly}$$

Three lessons:

- 1) We see the usefulness of **per-coordinate step-sizes**
- 2) We see the importance of accounting for curvatures
- 3) We see the advantage of **dampening the oscillations of GD**



(GD always moves perpendicular to the current level set, so it oscillates for this problem)

SGD w/ momentum

Idea is to dampen oscillations by continuing to move in historically useful directions

update direction: $\mu_t = \alpha_t \nabla f_t(x_t) + \delta_t \mu_{t-1}$

parameter update: $x_{t+1} = x_t - \mu_t$

historical
update
direction

momentum parameter
(0.9)

where ∇f_t is an estimate of ∇f at iteration t
(so this subsumes both GD and SGD w/ momentum)

Notice that this is equivalent to

$$x_{t+1} = x_t - \underbrace{\alpha_t \nabla f_t(x_t)}_{\text{update suggested by local model (GD or SGD)}} + \underbrace{\delta_t (x_t - x_{t-1})}_{\text{update suggested by history: keep moving in the previous direction}}$$

update suggested by
local model
(GD or SGD)

update suggested
by history: keep moving
in the previous direction

Notice that the update direction at time t is an exponentially weighted sum of the historical gradients

E.g. assume $\gamma_0 = 0$ and $\gamma_t = \gamma$ and $\alpha_t = \alpha$, then

$$x_{t+1} = x_0 - \alpha \sum_{k=0}^t \gamma^k \nabla f_{t-k}(x_{t-k})$$

Theoretical advantage of GD + momentum over GD is that its local convergence rate is linear w/ rate determined by $\sqrt{\kappa}$ vs κ for GD (if we choose the optimal step size)

$$\alpha = \frac{4}{(\sqrt{\beta} + \sqrt{\mu})^2}$$

↑
strong
convexity
parameter

momentum

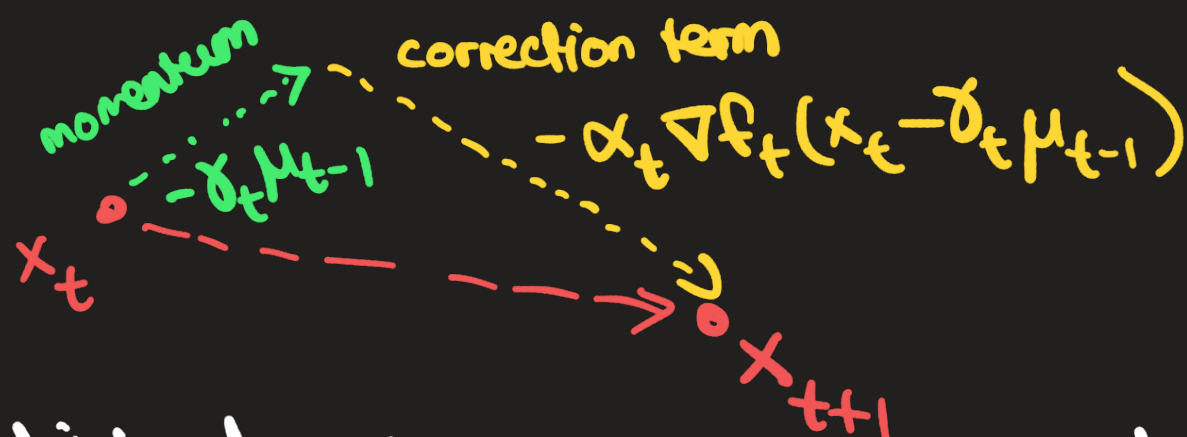
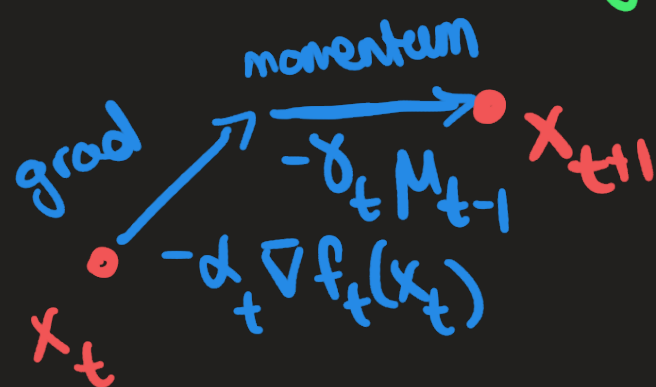
$$\gamma = \left(\frac{\sqrt{\beta} - \sqrt{\mu}}{\sqrt{\beta} + \sqrt{\mu}} \right)^2$$

Nesterov's Accelerated Gradient Descent (NAG)

$$\mu_t = \gamma_t \mu_{t-1} + \alpha_t \nabla f_t(x_t - \gamma_t \mu_{t-1})$$

$$x_{t+1} = x_t - \mu_t$$

$$= \underbrace{(x_t - \gamma_t \mu_{t-1})}_{\text{predict where to go using historical information}} - \underbrace{\alpha_t \nabla f_t(x_t - \gamma_t \mu_{t-1})}_{\text{correct yourself w/ the local model at that point}}$$



conv rate for well-conditioned probs is linear w/ rate determined by $\sqrt{\kappa}$
conv rate for β -smooth $O\left(\frac{1}{T^2}\right)$

Preconditioning

Imagine we want to solve $x^* = \operatorname{argmin} f(x)$

Introduce a change of variables $x = Ru$ where R is invertible

$$u^* = \operatorname{argmin} f(Ru) \Leftrightarrow x^* = Ru^*$$

We can solve in the u -space using GD via

$$u_{t+1} = u_t - \alpha_t R^T \nabla f_t(Ru_t)$$

It follows that

$$\begin{aligned} x_{t+1} &= Ru_{t+1} = R(u_t - \alpha_t R^T \nabla f_t(Ru_t)) \\ &= x_t - \alpha_t RR^T \nabla f_t(x_t) \end{aligned}$$

so if $R = [\nabla^2 f(x_t)]^{-1/2}$ then we get Newton's method.

In practice, use diagonal preconditioners.

Ada Grad "Adaptive Gradient Descent" (diagonal version)

- learns different stepsizes for each coordinate
- useful when the objective has different scales w.r.t. each feature

$$x_{t+1} = x_t - \alpha (D_t + \epsilon I)^{-1/2} \nabla f_t(x_t)$$

where

$$D_t = \text{diag} \left(\sum_{i=1}^t \nabla f_i(x_i) \nabla f_i(x_i)^T \right)$$

$$= \begin{bmatrix} \sum_{i=1}^t \left(\frac{\partial f_i(x_i)}{\partial x_1} \right)^2 & & \\ & \ddots & \\ & & \sum_{i=1}^t \left(\frac{\partial f_i(x_i)}{\partial x_d} \right)^2 \end{bmatrix}$$

← measures how large, historically, the components of the gradient corresponding to each coordinate has been

Per coordinate update:

$$(x_{t+1})_i = (x_t)_i - \frac{\alpha}{\left(\sum_{j=1}^t [\nabla f_j(x_j)]_i^2 + \epsilon\right)^{1/2}} [\nabla f_t(x_t)]_i$$

Two phenomena occur:

- all coordinates' learning rates decrease with time
- coordinates that changed more (their gradients were large) have learning rates that go to zero faster

RMSProp (2012)

attempted to fix the problem of AdaGrad's inevitable learning rate decay

$$D_t = \beta D_{t-1} + (1-\beta) \text{diag} \left(\nabla f_t(x_t) \nabla f_t(x_t)^T \right)$$

vs

$$D_t = D_{t-1} + \text{diag} \left(\nabla f_t(x_t) \nabla f_t(x_t)^T \right)$$

The update remains the same as for AdaGrad

$$x_{t+1} = x_t - \alpha (D_t + \epsilon I)^{-1/2} \nabla f_t(x_t)$$

ADAM (2014)

very popular in the ML community

- tries to fix the learning rate decay issue of AdaGrad
- also tries to estimate a better search direction using momentum

$$\mu_t = \beta_1 \mu_{t-1} + (1 - \beta_1) \nabla f_t(x_t)$$

$$D_t = \beta_2 D_{t-1} + (1 - \beta_2) \text{diag}(\nabla f_t(x_t) \nabla f_t(x_t)^T)$$

$$x_{t+1} = x_t - \alpha (D_t + \epsilon I)^{-1/2} \mu_t$$