ML & Optimization Lecture 13

- Tips & Tricks for SGD

- Adaptive Gradient Descent Methods
  - faster convergence for first-order methods, e.g. $O(\frac{1}{T})$ for β-smooth using SGD to $O(\frac{1}{T^2})$ (optimal) using adaptive methods

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 & β_1 should be optimized
  - use a small subset of data to test the convergence behavior (look at 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
- **stochastic easily** very easy to replace gradient w/ random estimates to further reduce 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 function (slower convergence than 2nd order methods)

Question: can we design first-order methods which are 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

Today:

1964 - SGD w/ momentum (Polyak's heavy ball method)

1983 - Nesterov's accelerated gradient descent (NAG)

2014 - AdaGrad

2012 - RMSProp

2012 - AdaDelta

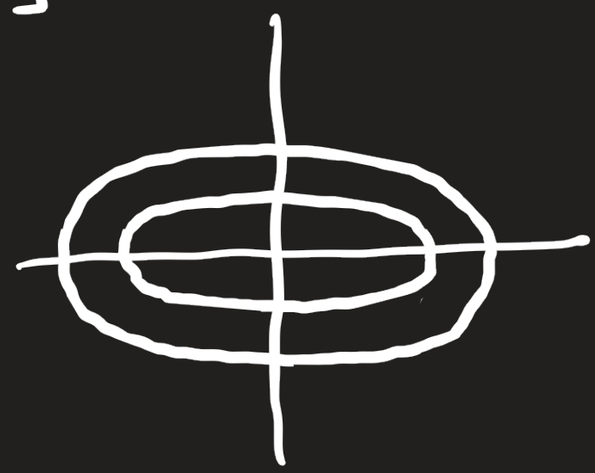
2015 - ADAM

Ex

$$f(x) = \frac{1}{2}\|Dx\|_2^2 \qquad D = \begin{bmatrix} 1 & \\ & 10 \end{bmatrix}$$

$$= \frac{1}{2}x_1^2 + 5x_2^2$$

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



with GD:

$$x_{t+1} = x_t - \alpha D^2 x_t = (I - \alpha D^2)x_t = \dots = (I - \alpha D^2)^{t+1} x_0$$

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

The "optimal" learning rate is $\alpha = \frac{1}{100}$, i.e. controlled by coordinate with highest curvature

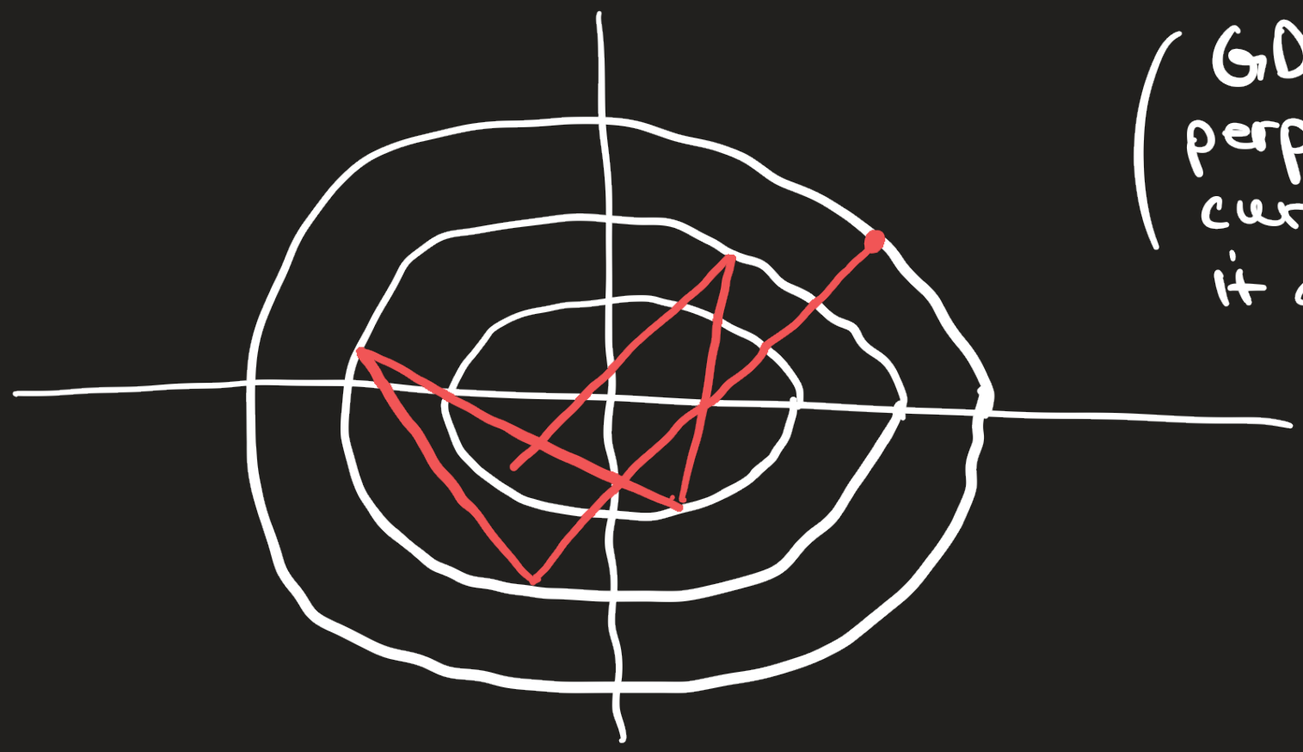
$$\Rightarrow x_t = \begin{bmatrix} (99/100)^t & \\ & 0 \end{bmatrix} \begin{bmatrix} x_{0,1} \\ x_{0,2} \end{bmatrix}$$

Three lessons:

1) We see the usefulness of per-coordinate step sizes

2) We see the importance of accounting for curvature

3) We see the advantages of dampening the oscillations of GD



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

SGD w/ momentum (aka Polyak heavy-ball method)

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

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

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

($\nabla f_t(x_t)$: gradient estimate at time t; μ_{t-1}: previous update direction; γ_t: momentum parameter (0.9))

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

Notice this is equivalent to

$$x_{t+1} = \underbrace{x_t - d_t \nabla f_t(x_t)}_{\text{update suggested by local info (SGD)}} + \underbrace{\gamma_t (x_t - x_{t-1})}_{\text{keep moving in the previous direction}}$$

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 conv rate is linear w/ rate determined by $\sqrt{\kappa}$ vs κ for GD (if we choose the optimal step size $\alpha = \frac{4}{(\sqrt{\beta} + \sqrt{\mu})^2}$ and $\gamma = \left(\frac{\sqrt{\beta} - \sqrt{\mu}}{\sqrt{\beta} + \sqrt{\mu}}\right)^2$)

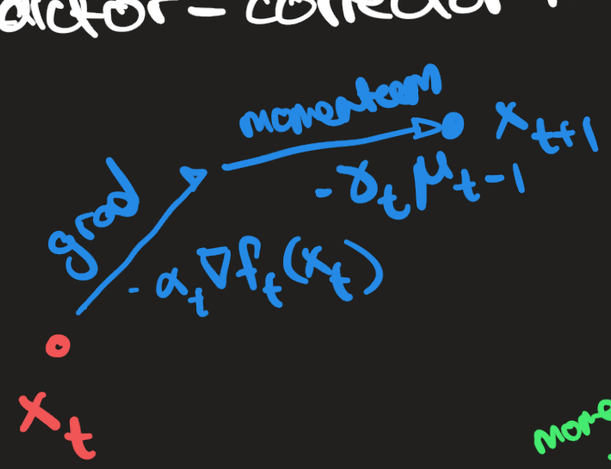
Nesterov's Accelerated Gradient Descent (NAG)

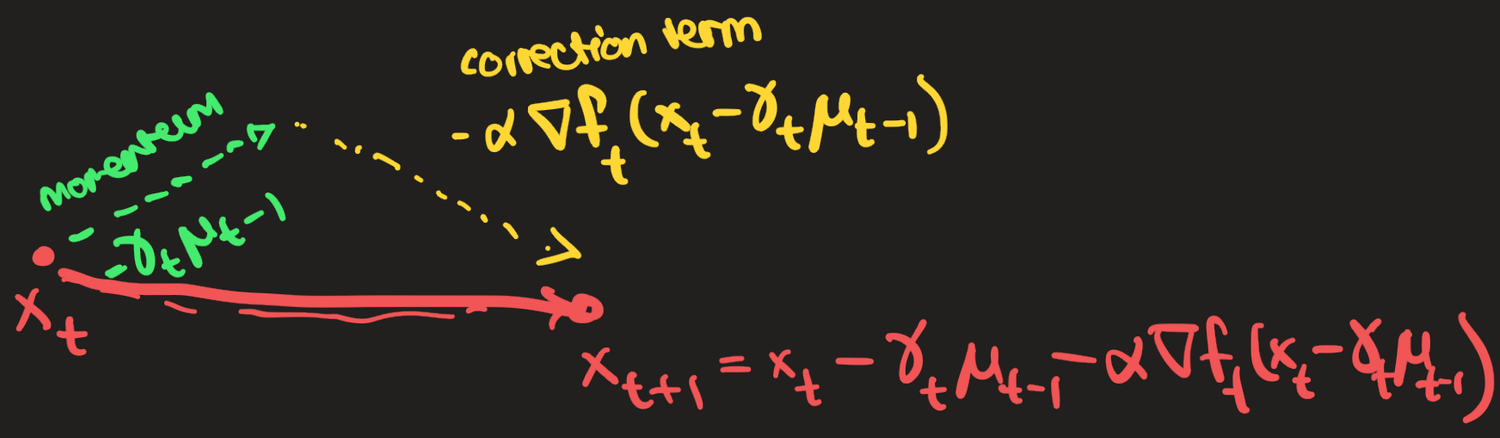
$$\mu_t = \gamma_t \mu_{t-1} + \alpha \nabla f_t (x_t - \gamma_t \mu_{t-1})$$

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

predictor-corrector method

(GD + momentum)





conv rate for strongly-cux & smooth probs depends on $\sqrt{\kappa}$

and for just smooth $O\left(\frac{1}{T^2}\right)$

Preconditioning

$$x^* = \operatorname*{argmin}_x f(x)$$

introduce invertible matrix R and consider

$$v^* = \operatorname*{argmin}_v f(Rv)$$

then $x^* = Rv^*$.

Let $g(v) = f(Rv)$, then $\nabla g(v) = R^T \nabla f(Rv)$

and $\nabla^2 g(v) = R^T \nabla^2 f(Rv) R$ } fix self reference

so we can take $R = [\nabla^2 f(Rv)]^{-1/2}$, then

$$\nabla^2 g(v) = I$$

and we have a very well-conditioned optim prob!

So we can solve

$$v^* = \operatorname{argmin}_v f(Rv)$$

very efficiently, via

$$v_{t+1} = v_t - \alpha_t R^T \nabla f_t(Rv_t)$$

Let $x_{t+1} = Rv_{t+1}$

$$\Rightarrow \quad x_{t+1} = Rv_{t+1} = R\left(v_t - \alpha_t R^T \nabla f_t(Rv_t)\right)$$

$$= Rv_t - \alpha_t RR^T \nabla f_t(Rv_t)$$

$$= x_t - \alpha_t RR^T \nabla f_t(x_t)$$

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

In practice, use <u>diagonal preconditions</u>

AdaGrad "Adaptive Gradient Descent" (diagonal version)

- learns different stepsizes for each coordinate
- useful when loss surface has different scales w/ respect to each feature

$$x_{t+1} = x_t - \alpha \left(D_t + \varepsilon I \right)^{-1/2} \nabla f(x_t) \qquad \varepsilon \approx 10^{-8}$$

where

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

notice

$$(D_t)_{ii} = \sum_{j=1}^{t} [\nabla f_j(x_j)]_i^2$$

— measures how large, historically, the component of the gradient corresp to $(x)_i$ has been.

Per coordinate update:

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

Two phenomena occur:

- all coordinates' learning rates decrease with time
- the coordinates that are seen or changed more have learning rates that go to zero faster