Convex Optimization 10-725, Lecture 13: Adaptive Algorithms

Yuanzhi Li

Assistant Professor, Carnegie Mellon University

Today

Last lecture

- We have learnt the pre-conditioned gradient descent.
- Algorithm that looks at the "geometry" of the function and scale the gradient accordingly.

$$x_{t+1} = x_t - M^{-1} \nabla f(x_t)$$

- Question: How do we find such "scaling"?

This lecture

- We are going to learn adaptive optimization algorithm, in particular, the Adagrad algorithm.

Motivation

- We have learnt the pre-conditioned gradient descent.

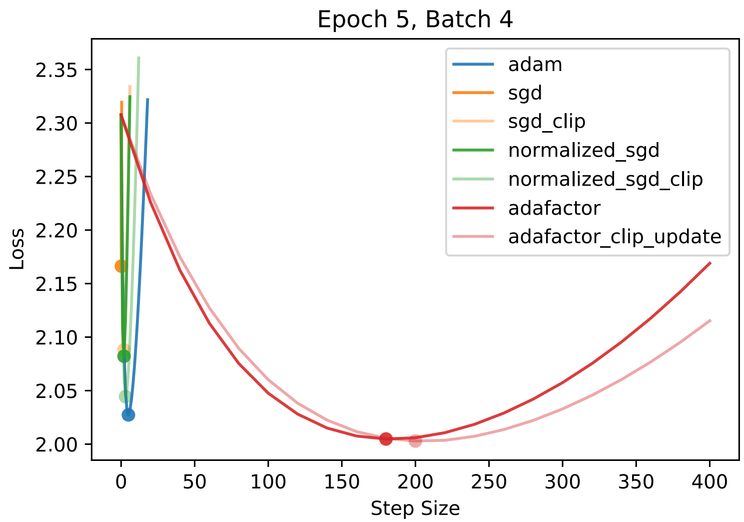
$$x_{t+1} = x_t - M^{-1} \nabla f(x_t)$$

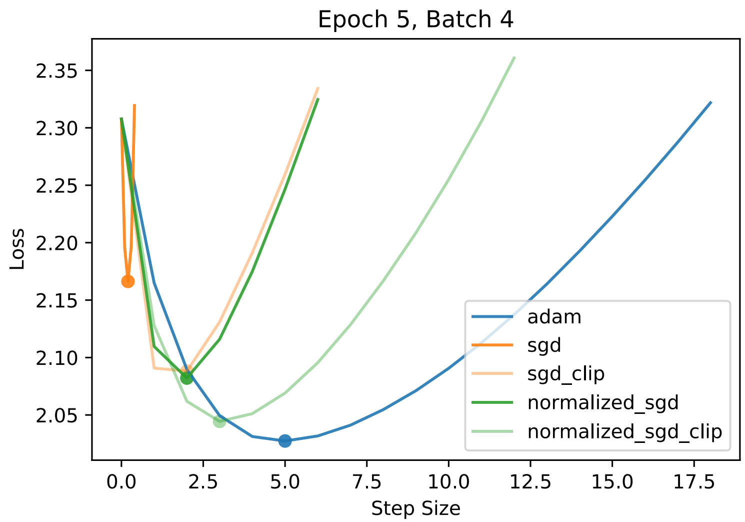
- In some cases, pre-conditioned gradient descent can be much better than gradient descent in terms of convergence rate.
- Local second-order approximation:

$$f(y) \le f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2}(y - x)^\top \nabla^2 f(x)(y - x) + O(\|y - x\|_2^3)$$

- Gradient descent $y = x - \eta \nabla f(x)$ minimizes $\langle \nabla f(x), y - x \rangle$, but the gradient direction might not be good for $(y - x)^\top \nabla^2 f(x)(y - x)$.

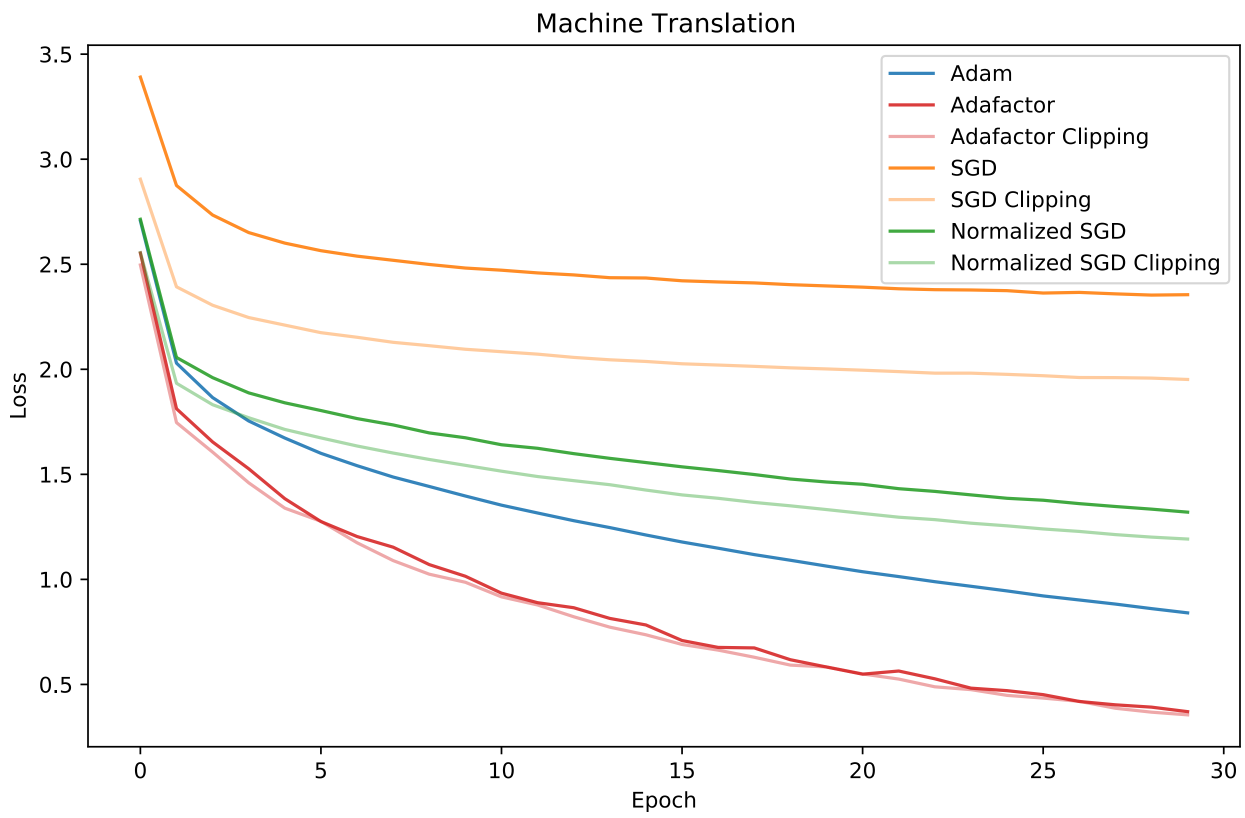
- Gradient descent $y = x - \eta \nabla f(x)$ minimizes $\langle \nabla f(x), y - x \rangle$, but the gradient direction might not be good for $(y - x)^\top \nabla^2 f(x)(y - x)$.
- Evidence in practice: Compare different update directions (normalize to unit vector): The function value of $f(x - \eta v)$ for different η and different v. Transformer on Machine Translation Task.





Motivation

- Local decrement is positively correlated with the algorithm's final performance:



Motivation

- We have learnt the pre-conditioned gradient descent.

$$x_{t+1} = x_t - M^{-1}\nabla f(x_t)$$

- In some cases, pre-conditioned gradient descent can be much better than gradient descent in terms of convergence rate.
- Key question I: How do we compute M^{-1} efficiently?
- Key question II: How do we find the best pre-condition matrix?
- Solving question I is straight forward: Instead of using a full matrix M, we only consider M that is a diagonal matrix.
- For efficiency, we only considering performing diagonal pre-conditioning, to adapt to the "diagonal geometry".
- This lecture: We will see an algorithm that can find such a diagonal matrix automatically.

Motivation

- Example when gradient descent needs pre-conditon: $x \in \mathbb{R}^d$,

$$f(x) = x^\top \begin{pmatrix} 100 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \dots & & & & \\ 0 & 0 & \dots. & 1 & 0 \\ 0 & 0 & \dots. & 0 & 1 \end{pmatrix} x = x^\top A x$$

- Gradient Descent $x_{t+1} = x_t - \eta(200[x_t]_1, 2[x_t]_2, 2[x_t]_3, \cdots, 2[x_t]_d)$: Can not use learning rate $\eta \geq \frac{1}{100}$.
- How do we find a good "pre-conditioner"?

Motivation

- Example when gradient descent needs pre-conditon: $x \in \mathbb{R}^d$,

$$f(x) = x^\top \begin{pmatrix} 100 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \dots & & & & \\ 0 & 0 & \dots. & 1 & 0 \\ 0 & 0 & \dots. & 0 & 1 \end{pmatrix} x = x^\top A x$$

- Gradient Descent $x_{t+1} = x_t - \eta(200[x_t]_1, 2[x_t]_2, 2[x_t]_3, \cdots, 2[x_t]_d)$.
- Key idea: always try to use large learning rate .
- If some coordinate of the function has bad smoothness, then gradient will be large at that coordinate (since its zig-zagging).
- When a coordinate of $\nabla f(x_t)$ is too large, scale it down.

Motivation

- If some coordinate of the function has bad smoothness, then gradient will be large at that coordinate (since its zig-zagging).
- When a coordinate of $\nabla f(x_t)$ is too large, scale it down.
- Evidence in practice (Transformer, Machine Translation): We measure $v^\top \nabla^2 f(x) v / \|v\|_2^2$ for different direction v.

Algorithm	**Sharpness**
Adam	0.16190993
SGD	31.04433435
SGD Clipping top 0.1% coordinate	1.77876506
Normalized SGD	0.77112307
Normalized SGD Clipping	0.38075357
Adafactor	3.1928×10^{-6}
Adafactor Clipping	2.5258×10^{-6}

Motivation

- Example when gradient descent needs pre-conditon: $x \in \mathbb{R}^d$,

$$f(x) = x^\top \begin{pmatrix} 100 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \ldots & & & & \\ 0 & 0 & \ldots. & 1 & 0 \\ 0 & 0 & \ldots. & 0 & 1 \end{pmatrix} x = x^\top A x$$

- Gradient Descent $x_{t+1} = x_t - \eta(200[x_t]_1, 2[x_t]_2, 2[x_t]_3, \cdots, 2[x_t]_d)$.
- When we use learning rate 0.1, from $x_0 = (1, 1, \cdots, 1)$:
- $\nabla f(x_0) = (200, 2, 2, \cdots, 2)$. The first coordinate is too large.
- If we somehow "scale down" the first coordinate of $\nabla f(x_0)$ by a factor of 100, then the update will be nice.
- But how do we know "100" is the correct value?

Motivation

- Gradient Descent $x_{t+1} = x_t - \eta(200[x_t]_1, 2[x_t]_2, 2[x_t]_3, \cdots, 2[x_t]_d)$.
- When we use learning rate 0.1, from $x_0 = (1, 1, \cdots, 1)$:
- $\nabla f(x_0) = (200, 2, 2, \cdots, 2)$. The first coordinate is too large!
- If we somehow "scale down" the first coordinate of $\nabla f(x_0)$ by a factor of 100, then everything will be nice.
- Idea: We can just scale down the gradient so each coordinate of the gradient has absolute value one.
- Update $x_1 = x_0 - M_0^{-1}\nabla f(x_0)$ where

$$M_0 = \mathrm{diag}(|[\nabla f(x_0)]_1|, |[\nabla f(x_0)]_2|, \cdots, |[\nabla f(x_0)]_d|)$$

- In this case, we are updating $x_1 = x_0 - \eta(1, 1, \cdots, 1)$.

Motivation

- Idea: We can just scale down the gradient so each coordinate of the gradient has absolute value one.
- Update $x_{t+1} = x_t - M_t^{-1}\nabla f(x_t)$ where

$$M_t = \mathrm{diag}(|[\nabla f(x_t)]_1|, |[\nabla f(x_t)]_2|, \cdots, |[\nabla f(x_t)]_d|)$$

- In fact, we are simply doing gradient sign method:

$$x_{t+1} = x_t - \eta \mathrm{sign}(\nabla f(x_t))$$

- This is an analog of the "Rprop" Algorithm, and today we are going to study a more stable version of it called the Adagrad.

Adaptive algorithm

- Adagrad: Adaptive algorithm that can find the "best scaling" of each coordinate automatically.
- Using the "sum of the past gradients" to define the scaling matrix, instead of just the gradient from the last iteration.
- We have learnt in Momentum: "weighted" sum of the past gradients makes the update more stable.

Adagrad

- Adagrad to minimize a function f: At every iteration t, update

$$x_{t+1} = x_t - \eta M_t^{-1} \nabla f(x_t)$$

- Where

$$M_t = \mathrm{diag}\left(\left\{\sqrt{\sum_{s \le t} [\nabla f(x_s)]_j^2}\right\}_{j=1}^{d}\right)$$

- Adagrad to minimize a function f with stochastic gradient $\tilde{\nabla} f$:

$$x_{t+1} = x_t - \eta M_t^{-1} \tilde{\nabla} f(x_t)$$

- Where

$$M_t = \mathrm{diag}\left(\left\{\sqrt{\sum_{s \le t} [\tilde{\nabla} f(x_s)]_j^2}\right\}_{j=1}^{d}\right)$$

Adagrad

- Instead of using $M_t = \mathrm{diag}(|[\nabla f(x_t)]_1|, |[\nabla f(x_t)]_2|, \cdots, |[\nabla f(x_t)]_d|)$ for one iteration, Adagrad looks at the history:

$$M_t = \mathrm{diag}\left(\left\{\sqrt{\sum_{s \le t} [\nabla f(x_s)]_j^2}\right\}_{j=1}^{d}\right)$$

- Spirit: We should always learn from history: If one coordinate of the gradient has been large for a while, scale it down. If it has been small for a while, scale it up.
- This is more stable than looking at the last iteration along, especially in stochastic gradient case.

Convergence Rate of Adagrad

- Adagrad: At every iteration t, update

$$x_{t+1} = x_t - \eta M_t^{-1} \nabla f(x_t)$$

- Where

$$M_t = \mathrm{diag}\left(\left\{\sqrt{\sum_{s \le t} [\nabla f(x_s)]_j^2}\right\}_{j=1}^{d}\right)$$

- Convergence: For a convex function $f : \mathbb{R}^d \to \mathbb{R}$, suppose $\max_t \|x_t - x^*\|_\infty \le D$, we have that with $\eta = D/\sqrt{2}$,

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \le f(x^*)$$

$$+\frac{\sqrt{2d}D}{T}\sqrt{\inf_{Q \in \mathbb{R}^d, Q \ge 0, \|Q\|_1 \le d} \sum_{t=0}^{T-1} \nabla f(x_t)^\top \mathrm{diag}(Q)^{-1} \nabla f(x_t)}$$

Convergence Rate of Adagrad (Stochastic)

- Adagrad: At every iteration t, update

$$x_{t+1} = x_t - \eta M_t^{-1} \tilde{\nabla} f(x_t)$$

- Where

$$M_t = \operatorname{diag}\left(\left\{\sqrt{\sum_{s \le t} [\tilde{\nabla} f(x_s)]_j^2}\right\}_{j=1}^{d}\right)$$

- Convergence: For a convex function $f : \mathbb{R}^d \to \mathbb{R}$, suppose $\max_t \|x_t - x^*\|_\infty \le D$, we have that with $\eta = D/\sqrt{2}$,

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[f(x_t)] \le f(x^*)$$

$$+\frac{\sqrt{2d}D}{T}\sqrt{\inf_{Q \in \mathbb{R}^d, Q \ge 0, \|Q\|_1 \le d} \sum_{t=0}^{T-1} \mathbb{E}\left[\tilde{\nabla} f(x_t)^\top \operatorname{diag}(Q)^{-1} \tilde{\nabla} f(x_t)\right]}$$

Convergence Rate of Adagrad

- Convergence of adagrad: For a convex function $f : \mathbb{R}^d \to \mathbb{R}$, suppose $\max_t \|x_t - x^*\|_\infty \le D$, we have that with $\eta = D/\sqrt{2}$,

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \le f(x^*)$$

$$+\frac{\sqrt{2d}D}{T}\sqrt{\inf_{Q\in\mathbb{R}^d, Q\ge 0, \|Q\|_1\le d} \sum_{t=0}^{T-1} \nabla f(x_t)^\top \mathrm{diag}(Q)^{-1}\nabla f(x_t)}$$

- Convergence rate using a fixed diagonal precondition matrix $M \ge 0$ ($\mathrm{Tr}(M) \le d$): $x_{t+1} = x_t - \eta M^{-1}\nabla f(x_t)$: something of shape:

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \le f(x^*) + \frac{dD^2}{\eta T} + \frac{\eta}{T}\sum_{t=0}^{T-1}\nabla f(x_t)^\top M^{-1}\nabla f(x_t)$$

Convergence Rate of Adagrad

- Convergence of adagrad: For a convex function $f : \mathbb{R}^d \to \mathbb{R}$, suppose $\max_t \|x_t - x^*\|_\infty \le D$, we have that with $\eta = D/\sqrt{2}$,

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \le f(x^*)$$

$$+\frac{\sqrt{2d}D}{T}\sqrt{\inf_{Q\in\mathbb{R}^d, Q\ge 0, \|Q\|_1\le d} \sum_{t=0}^{T-1} \nabla f(x_t)^\top \mathrm{diag}(Q)^{-1} \nabla f(x_t)}$$

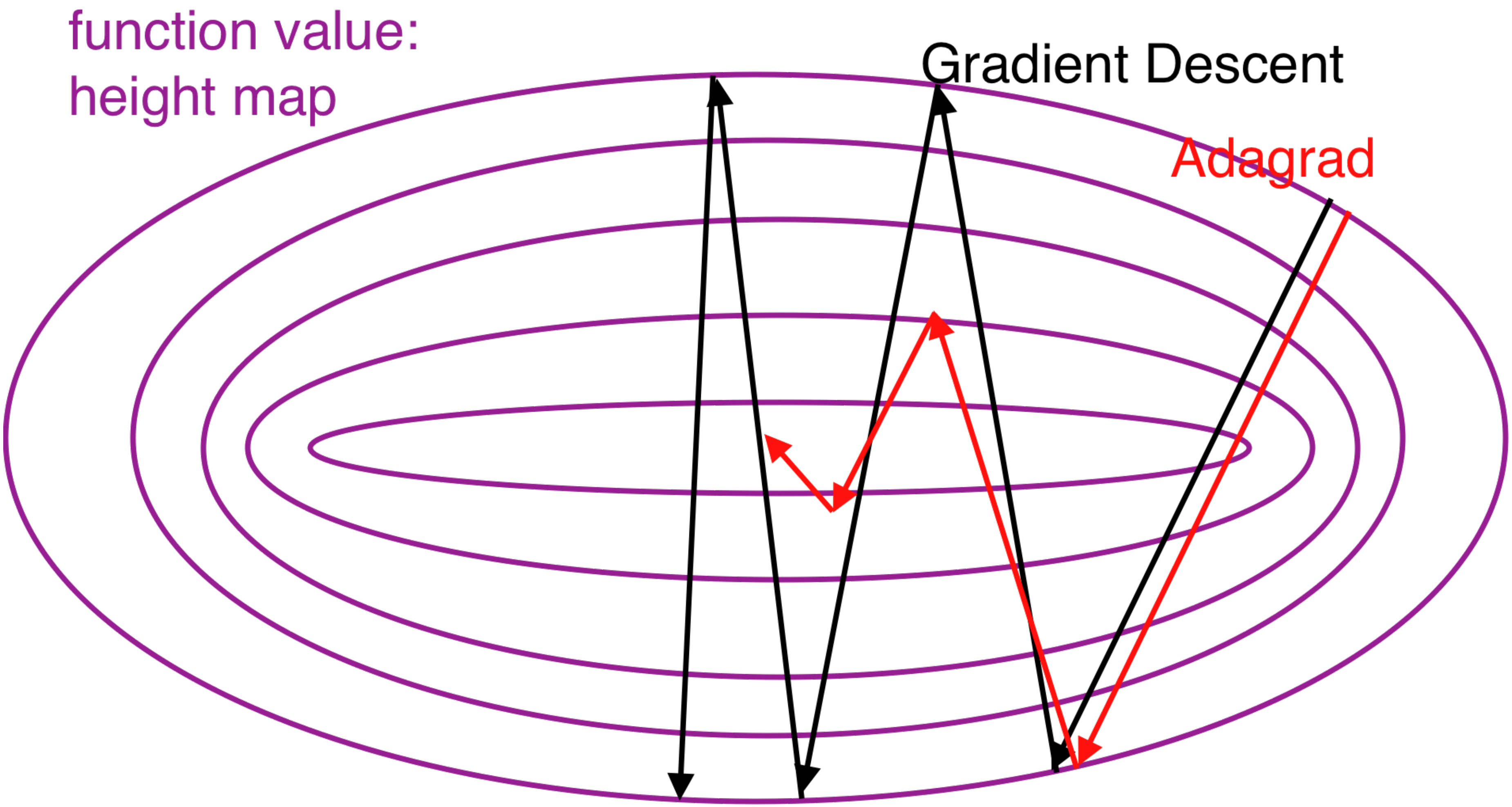
- Convergence rate using a fixed diagonal precondition matrix $M \ge 0$ ($\mathrm{Tr}(M) \le d$): $x_{t+1} = x_t - \eta M^{-1}\nabla f(x_t)$: something of shape:

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \le f(x^*) + \frac{dD^2}{\eta T} + \frac{\eta}{T}\sum_{t=0}^{T-1} \nabla f(x_t)^\top M^{-1} \nabla f(x_t)$$

- Precondition gradient descent: need to tune η carefully and choose the pre-conditioning matrix carefully.
- Adagrad does not need to tune the learning rate η or choose the pre-conditioning matrix!

Adagrad: What would happen?

- Adagrad: Scale down the gradient along a coordinate if it's too large, scale up if it's too small.



-

Adagrad: proof

- Recall:

$$M_t = \mathrm{diag}\left(\left\{\sqrt{\sum_{s\le t}[\nabla f(x_s)]_j^2}\right\}_{j=1}^{d}\right)$$

- Key observation: Let $\Phi_t(x) = \frac{1}{2}x^\top M_t x$, then $\nabla \Phi_t(x) = M_t x$.
- Therefore, Adagrad is actually doing a Mirror Descent update:

$$\nabla \Phi_t(x_{t+1}) = \nabla \Phi_t(x_t) - \eta \nabla f(x_t)$$

- Recall the Mirror Descent Lemma: for every y,

$$f(x_t) \le f(y) + \frac{1}{\eta}\left(D_{\Phi_t}(y, x_t) - D_{\Phi_t}(y, x_{t+1}) + D_{\Phi_t}(x_t, x_{t+1})\right)$$

Adagrad: proof

- Now we have for every y,

$$f(x_t) \leq f(y) + \frac{1}{\eta}\left(D_{\Phi_t}(y, x_t) - D_{\Phi_t}(y, x_{t+1}) + D_{\Phi_t}(x_t, x_{t+1})\right)$$

- Where the Bregman Divergence is given as

$$\begin{aligned} D_{\Phi_t}(x, y) &= \Phi_t(x) - \Phi_t(y) - \langle \nabla \Phi_t(y), x - y \rangle \\ &= \frac{1}{2} x^\top M_t x - \frac{1}{2} y^\top M_t y - \langle M_t y, x - y \rangle \\ &= \frac{1}{2}(x - y)^\top M_t (x - y) \end{aligned}$$

Adagrad: proof

- Now we have for every y,

$$f(x_t) \le f(y) + \frac{1}{\eta}\left(D_{\Phi_t}(y, x_t) - D_{\Phi_t}(y, x_{t+1}) + D_{\Phi_t}(x_t, x_{t+1})\right)$$

- Average the above inequality from $t = 0, 1$ up to $T - 1$, we have:

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \le f(y) + \frac{1}{\eta T} D_{\Phi_0}(y, x_0)$$

$$+\frac{1}{\eta T}\sum_{t=0}^{T-2}\left(D_{\Phi_{t+1}}(y, x_{t+1}) - D_{\Phi_t}(y, x_{t+1})\right)$$

$$+\frac{1}{\eta T}\sum_{t=0}^{T-1} D_{\Phi_t}(x_t, x_{t+1})$$

- We first look at the red term, then we look at the blue term.

Adagrad: proof

- For the red term:

$$\frac{1}{\eta T}\sum_{t=0}^{T-2}\left(D_{\Phi_{t+1}}(y, x_{t+1}) - D_{\Phi_t}(y, x_{t+1})\right)$$

- For each t, since $D_{\Phi_t}(x, y) = \frac{1}{2}(x-y)^\top M_t(x-y)$, we conclude that:

$$D_{\Phi_{t+1}}(y, x_{t+1}) - D_{\Phi_t}(y, x_{t+1}) = \frac{1}{2}(x_{t+1}-y)^\top(M_{t+1}-M_t)(x_{t+1}-y)$$

- Let $s_t = \left(\sqrt{\sum_{s\le t}[\nabla f(x_s)]_j^2}\right)_{j=1}^{d} \in \mathbb{R}^d$, we have that $M_t = \mathrm{diag}(s_t)$ and

$$\frac{1}{2}(x_{t+1}-y)^\top(M_{t+1}-M_t)(x_{t+1}-y) \le \|x_{t+1}-y\|_\infty^2\|s_{t+1}-s_t\|_1 \le D^2\|s_{t+1}-s_t\|_1$$

Adagrad: proof

- For the red term:

$$\frac{1}{\eta T} \sum_{t=0}^{T-2} \left(D_{\Phi_{t+1}}(y, x_{t+1}) - D_{\Phi_t}(y, x_{t+1}) \right)$$

- Now we have with $s_t = \left(\sqrt{\sum_{s \le t} [\nabla f(x_s)]_j^2} \right)_{j=1}^d \in \mathbb{R}^d$,

$$D_{\Phi_{t+1}}(y, x_{t+1}) - D_{\Phi_t}(y, x_{t+1}) \le D^2 \| s_{t+1} - s_t \|_1$$

- Notice that $s_{t+1} \ge s_t$, so $\| s_{t+1} - s_t \|_1 = \langle s_{t+1} - s_t, u \rangle$ where u is the all one vector.
- Together, we have that

$$\frac{1}{\eta T} \sum_{t=0}^{T-2} \left(D_{\Phi_{t+1}}(y, x_{t+1}) - D_{\Phi_t}(y, x_{t+1}) \right) \le \frac{D^2}{\eta T} \langle s_{T-1} - s_0, u \rangle$$

Adagrad: proof

- Now we have

$$\frac{1}{\eta T}\sum_{t=0}^{T-2}\left(D_{\Phi_{t+1}}(y, x_{t+1}) - D_{\Phi_t}(y, x_{t+1})\right) \le \frac{D^2}{\eta T}\langle s_{T-1} - s_0, u\rangle$$

- Key observation: By Cauchy-Swartz inequality: for every vector $Q \ge 0$

$$\langle s_{T-1}, u\rangle^2 = \left(\sum_{j\in[d]}\sqrt{\sum_{s\le T-1}[\nabla f(x_s)]_j^2}\right)^2$$

$$\le \left(\sum_{j\in[D]} Q_j\right)\left(\sum_{j\in[D]} Q_j^{-1}\sum_{s\le T-1}[\nabla f(x_s)]_j^2\right)$$

$$= \left(\sum_{j\in[D]} Q_j\right)\left(\sum_{t=0}^{T-1}\nabla f(x_t)^\top \mathrm{diag}(Q)^{-1}\nabla f(x_t)\right)$$

Adagrad: proof

- Key observation: By Cauchy-Swartz inequality: for every vector $Q \geq 0$

$$\langle s_{T-1}, u\rangle^2 = \left(\sum_{j\in[d]} \sqrt{\sum_{s\leq T-1} [\nabla f(x_s)]_j^2}\right)^2$$

$$\leq \left(\sum_{j\in[D]} Q_j\right)\left(\sum_{j\in[D]} Q_j^{-1} \sum_{s\leq T-1} [\nabla f(x_s)]_j^2\right)$$

$$= \left(\sum_{j\in[D]} Q_j\right)\left(\sum_{t=0}^{T-1} \nabla f(x_t)^\top \mathrm{diag}(Q)^{-1} \nabla f(x_t)\right)$$

- Since this is true for every $Q \geq 0$, we have:

$$\langle s_T, u\rangle^2 \leq d \inf_{Q\in\mathbb{R}^d, Q\geq 0, \|Q\|_1\leq d} \sum_{t=0}^{T-1} \nabla f(x_t)^\top \mathrm{diag}(Q)^{-1} \nabla f(x_t)$$

- We finish the proof of the red term.

Adagrad: proof

- Now we check the blue term

$$\frac{1}{\eta T}\sum_{t=0}^{T-1} D_{\Phi_t}(x_t, x_{t+1})$$

- Again, recall $D_{\Phi_t}(x, y) = \frac{1}{2}(x-y)^\top M_t(x-y)$, we have that

$$2D_{\Phi_t}(x_t, x_{t+1}) = (x_t - x_{t+1})^\top M_t(x_t - x_{t+1}) = \eta^2[\nabla f(x_t)]^\top M_t^{-1}\nabla f(x_t)$$

- Recall we defined $s_t = \left(\sqrt{\sum_{s\le t}[\nabla f(x_s)]_j^2}\right)_{j=1}^{d} \in \mathbb{R}^d$ and $M_t = \text{diag}(s_t)$, $x_{t+1} = x_t - \eta M_t^{-1}\nabla f(x_t)$, so:

$$2D_{\Phi_t}(x_t, x_{t+1}) = (x_t - x_{t+1})^\top M_t(x_t - x_{t+1})$$

$$= \eta^2 \nabla f(x_t)^\top M_t^{-1}\nabla f(x_t) = \eta^2 \sum_{j\in[d]} \frac{[s_t]_j^2 - [s_{t-1}]_j^2}{[s_t]_j}$$

Adagrad: proof

- Now we reduce the blue term to

$$\frac{1}{\eta T}\sum_{t=0}^{T-1} D_{\Phi_t}(x_t, x_{t+1}) = \frac{\eta}{2T}\sum_{t=0}^{T-1}\sum_{j\in[d]}\frac{[s_t]_j^2 - [s_{t-1}]_j^2}{[s_t]_j}$$

$$\leq \frac{\eta}{T}\sum_{t=0}^{T-1}\sum_{j\in[d]}\frac{[s_t]_j^2 - [s_{t-1}]_j^2}{[s_t]_j + [s_{t-1}]_j}$$

$$= \frac{\eta}{T}\sum_{j\in[d]}\sum_{t=0}^{T-1}([s_t]_j - [s_{t-1}]_j) = \frac{\eta}{T}\langle s_{T-1}, u\rangle$$

- Which finishes the proof using

$$\langle s_{T-1}, u\rangle^2 \leq d \inf_{Q\in\mathbb{R}^d, Q\geq 0, \|Q\|_1\leq d}\sum_{t=0}^{T-1}\nabla f(x_t)^\top \mathrm{diag}(Q)^{-1}\nabla f(x_t)$$

Adam: Adagrad with Momentum

- In practice, people always use the Adam optimizer: Adagrad with Momentum.
- At each iteration, compute (Momentum) $g_{t+1} = \gamma g_t + (1-\gamma)\nabla f(x_t)$.
- Compute scaling factor $s_{t+1}^2 = \beta s_t^2 + (1-\beta)[\nabla f(x_t)]^2$ (entry wise square).
- Update using adaptive momentum:

$$x_{t+1} = x_t - \eta \mathrm{diag}(s_{t+1})^{-1} g_{t+1}$$

- Adam DOES NOT have convergence guarantee, even in convex setting (it can diverge), but it works extremely well in practice for training neural networks.