Regret: and SGD

- at each step, we see a new objective function
- online optimization

at time step t, alg sees $f_t : K \to R$ and proposes x_t

- how to score this?

$$\text{regret} = \sum_t f_t(x_t) - \min_{x^* \in K} \sum_t f_t(x^*)$$

time average of cost functions ~~at~~ chosen points.

you see all ~~alg~~ fn's in advance - find smallest average.

The reason this is helpful is we can prove bounds, which lead to new alg.s (2)

- take g.d. and substitute

$$x_{t+1} \longleftarrow x_t - \eta \nabla f_t(x_t)$$

use current function

from proof of G-2, we have

$$f_t(x_t) + \Phi_{t+1} - \Phi_t \leq f_t(x^*) + \frac{1}{2}\eta G^2$$

(cause true for any convex).

Sum this

$$\sum_{t=1}^{T} \left(f_t(x_t) - f_t(x^*) \right) \leq \sum_{t=1}^{T} \left(\Phi_t - \Phi_{t+1} \right) + \frac{1}{2}\eta G^2$$

$$\leq \Phi_1 + \frac{1}{2}\eta T G^2$$

now plug in $T \geq \frac{\|x_1 - x^*\|^2 G^2}{\varepsilon^2}$

$\eta = \frac{\|x_1 - x^*\|}{G\sqrt{T}}$, do algebra. to

get.

$$\frac{1}{T} \sum_{t=0}^{T} \left(f_t(x_t) - f_t(x^*) \right) \leq \frac{\|x_1 - x^*\| G}{\sqrt{T}} \leq \varepsilon$$

↑
regret.

<u>But</u> pure sgd is not particularly well adapted to some cases

- for regret bound, we assumed <u>nothing</u> about $f_t(x)$. (except convexity)

- Now imagine we <u>know</u> something.

- e.g.

$$f_t(x) = \frac{1}{2}\left[a_t^T x\right]^2$$

where a is very sparse

i'th component of a_t has prob of non-zero P_i, IID

and (aren't we lucky!) p is sorted

$$P_1 \geqslant P_2 \cdots \geqslant P_d .$$

think about the sequence of gradients

$1 \qquad\qquad t \qquad\qquad T \longrightarrow$

$0, \ldots \xi \ldots 0 \ldots \xi \ldots .$ ← few zeros

$0, 0, 0, \xi \ldots 0, 0, 0, \xi, 0\ 0$ ← about $(1-p_i)T$ zeros

$0, 0, 0, \ldots 0, \xi, 0, 0, 0$ ← many zeros

$d \downarrow$

In this case, we face a serious problem:

for some components, we take a large step very occasionally

- correcting this step would be hard (slow) cause gradient would be 0 at most t
- we would like to reduce the step using some estimate of sparsity.

Example

$$\underline{x} = \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ x_d \end{bmatrix}$$

$$f_t(\underline{x}) = \sum_i \delta_{it} \frac{x_i^2}{2} \qquad \text{where } \delta_{i,t} = \begin{cases} 1 & \text{with prob } p_i \\ 0 & (1-p_i) \end{cases}$$

and IID over t

Now think about component i?

$$x_i^{(\overset{\downarrow \text{iteration}}{w}+1)} = x_i^{(w)} - \eta g_i^{(w)} \qquad g_i^{(w)} = \begin{cases} x_i^{w} & \text{with prob } p_i \\ 0 & \text{otherwise} \end{cases}$$

so $x^{(w+1)} \approx \underbrace{(1-\eta)^{p(w+1)}}_{} x_0$

essentially, you see the gradient about $p(w+1)$ times but this converges rather slowly

IDEA :

$$x_i^{(\omega+1)} = (\text{shrink})\left[x_i^{(\omega)} - \frac{\eta}{\text{shrink}}\, g_i^{(\omega)}\right]$$

where shrink tises the frequency with which $g_i^{(\omega)} \neq 0$

~~ADAM~~ ADAGRAD :

Define $G_\omega = \sum_{t=1}^{\omega} g^{(t)} g^{(t)T}$

outer product matrix of past gradients

- Consider $\operatorname{diag} G_\omega = \Delta_\omega$
- i'th diagonal component is approx $E[g_i^2]$

update

$$\underline{x}^{(w+1)} = \left[\Delta^{(w)}\right]^{1/2}\left[\underline{x}^{(w)} - \eta\left[\Delta^{(w)}\right]^{(-1/2)}\underline{g}^{(w)}\right]$$

notice you could have a divide by zero issue here — actually use

$$\left[\Delta^{(w)} + \varepsilon I\right]^{(-1/2)}$$

Fact: (Duchi et al, p2125)

let $x^{(w)}$ be generated as above

and assume

$$\max_t \| x^* - x_t \|_\infty \leq D_\infty$$

use $\eta = \frac{D_\infty}{\sqrt{2}}$. then for x^* we

get.

$$R(T) \leqslant \sqrt{2}\, D_\infty \sum_{i=1}^{d} \|g_{1:T,i}\|_2$$

regret at T

length of i'th component of gradient sequence (NOTE: NOT squared length)

this is better than previous bound when

- D_∞ is reasonable (eg, we're in a box
- gradients have nice sparsity properties

<u>More examples</u>: Duchi et al 2125....

Alternative view

- $f_t(x)$ is a "noisy" function rather than an arbitrary seq of convex functions (avoid defining, as do authors!)

- consider steps.
  - in each component, we would like the step to be bounded by <u>about</u> the same amount

  - consider

$$\underbrace{\alpha\left[\frac{\text{mean(gradient)}}{\sqrt{\text{mean(gradient}^2)}}\right]}$$

↳ this will tend to be about 1 or −1

· this "filters" the gradient

ISSUE

· estimate means ?

· these estimates should be

mean(gradient$_t$)
over all possible
f_t

· But we can't compute that.

· instead, compute moving average

$$v_0 = 0$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\, g_t^2$$

element wise

· this incorporates past gradients into est of mean(gradient^2)

BUT as an est of mean(g_t^2)
it is biased (~~Welling et al~~ Kingma + Welling, p3)

use

$$\frac{v_t}{(1-\beta_2^t)}$$

(same arg for mean(g_t))

We now have ADAM

Experience shows
- fast convergence to slightly worse solns
- SGD with good steplengths gets better solns but much slower

WHY ?

Algorithm 2: *AdaMax*, a variant of Adam based on the infinity norm. See section 7.1 for details. Good default settings for the tested machine learning problems are $\alpha = 0.002$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. With β_1^t we denote β_1 to the power t. Here, $(\alpha/(1-\beta_1^t))$ is the learning rate with the bias-correction term for the first moment. All operations on vectors are element-wise.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $u_0 \leftarrow 0$ (Initialize the exponentially weighted infinity norm)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $u_t \leftarrow \max(\beta_2 \cdot u_{t-1}, |g_t|)$ (Update the exponentially weighted infinity norm)
    $\theta_t \leftarrow \theta_{t-1} - (\alpha/(1 - \beta_1^t)) \cdot m_t/u_t$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

regret.