Stochastic Gradient Descent

Monday, September 30, 2024 9:37 AM

Gradient Descent

$$w_{k+1} = w_k - \eta \nabla f(w_k)$$

Stochastic Gradient Descent

g_k "gradient-like", $\mathbb{E}[g_k] = \nabla f(w_k)$ or almost so

$w_{k+1} = w_k - \eta g_k$ a giant class of methods

Common use-case in ML

$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$ typically $f_i(w) = \ell(\hat{y}_i, y_i)$

(empirical risk) $\hat{y}_i = \text{neural\_net}_w(x_i)$

So we can think of

$$f(w) = \mathop{\mathbb{E}}_{i \sim \text{Unif}([n])} f_i(w) \quad \text{aka} \quad \underbrace{\mathop{\mathbb{E}}_{\xi} f(w;\xi)}_{\text{generic SGD notation}}$$

Then choose $g_k = \nabla f_i(w_k)$ for a sample $i \sim \text{Unif}([n])$

Theoretical aside: does $\nabla_w \mathop{\mathbb{E}}_{\xi} f(w;\xi) = \mathop{\mathbb{E}}_{\xi} \nabla_w f(w;\xi)$?

Answer:

Often but not always

Ex: $f = \frac{1}{2}(f_1 + f_2)$, $f_2 := -f_1$, f_1 something not differentiable.

Then ∇f exists, ∇f_1 and ∇f_2 do not even exist!

If things exist and are nicely bounded, then Lebesgue's DCT allow us to do the swap.

mini-batch

Instead of choosing $i \sim \text{Unif}([n])$ and set $g_k = \nabla f_i(w_k)$,
choose a batch of uniform indices $i \in B$, $|B| \le n$,
and set $g_k = \frac{1}{|B|}\sum_{i \in B} \nabla f_i(w_k)$

options: independent (w/ replacement)
not independent (i.e., shuffling)

SGD (page 2)

Monday, September 30, 2024 10:41 AM

Terminology:

$$X_{train} = \{x_1, x_2, \ldots, x_n\}$$

B_k is minibatch at step k

($B_k = X_{train}$ is "full-batch")

size $b = |B_k|$

SGD: for $k = 1, 2, \ldots$

draw B_k of size b, create g_k

$$w_{k+1} = w_k - \eta \cdot g_k$$

but we often write in an equivalent formulation:

for $\ell = 1, 2, 3, \ldots$

$w_1 = \tilde{w}_\ell$

for $j = 1, 2, \ldots, \frac{n}{b}$

draw B_j of size b

$$w_{j+1} = w_j - \eta \cdot g_j$$

as much "work" as a full batch. We call this one "epoch"

set $\tilde{w}_{\ell+1} = w_{j+1}$

often we pre-partition a shuffled dataset, so batches are without replacement and not independent

Typically not a large effect in practice if you do that or do iid w/ replacement.

SGD (page 3)

Monday, September 30, 2024 10:32 AM

SGD has been around a long time...

... but particularly effective when each f_i is similar, as is the case if $f_i(w) = \ell(h_w(x_i), y_i)$

x_i iid

Vanilla SGD uses a single stepsize η

i.e. $w_{k+1} = w_k - \eta \cdot I \cdot g_k$

Newton's method uses $\nabla^2 f^{-1}$ instead

A compromise is using a (non-constant) diagonal matrix D

$$w_{k+1} = w_k - D\, g_k$$

especially if $w = \begin{bmatrix} w_{\text{layer }1} \\ w_{\text{layer }2} \\ \vdots \end{bmatrix}$ possibly fundamentally different scales (i.e. "stiff" ODE-like)

ADAM, Adagrad, RMSProp, ... all try to address this scaling issue

Convergence ... and step-size schedulers

General SGD convergence says you converge only up to within a ball of radius $O(\eta)$ around a stationary point

- Small η = good solution, but takes long to find it since we converge slowly

  (η too large = divergence!)

Workarounds:

- $\eta = \eta_k$, eg. $\eta_k = \eta_0 \cdot \dfrac{\varepsilon}{\varepsilon + k}$ $k = 0, 1, \ldots$
- Schedulers every 100 iterations, say, decrease η