Lecture 18

Gradient descent and its generalizations

Consider a cost function

$$E(\vec{\theta}) = \sum_{i=1}^{n} \ell_i(\vec{x}_i, \vec{\theta})$$

(n ← total # datapoints; $\vec{x}_i$ ↑ input datapoint; $\vec{\theta}$ ↖ fitting prms)

Gradient descent (GD): $\vec{v}_t$

(1) $$\vec{\theta}_{t+1} = \vec{\theta}_t - \eta_t \vec{\nabla}_{\vec{\theta}} E(\vec{\theta}_t)$$, where

t is the step # & η_t is the learning rate (may be changing with t).

Finally, $$(\vec{\nabla}_{\vec{\theta}})_i = \frac{\partial}{\partial \theta_i}$$.

Main drawback: η_t is a hyperprm whose value (or "schedule" if we wish to change it with t) is difficult to obtain a priori.

Compare GD with Newton-Raphson (NR):

(2) $$\vec{\theta}_{t+1} = \vec{\theta}_t - \underbrace{H^{-1}(\vec{\theta}_t) \vec{\nabla}_{\vec{\theta}} E(\vec{\theta}_t)}_{\vec{v}_t}$$, where

H is the Hessian (in practice, H is often regularized: $H^{-1} \Rightarrow (H + \varepsilon \mathbb{I})^{-1}$, where ε is a small offset).

NR requires H (which is extremely expensive), but

its learning rate is set by H^{-1} instead of an ad hoc function (or constant) η_t.

In other words, NR automatically takes large steps in flat directions and smaller steps in directions with large curvature, whereas GD is not as adaptive.

Besides, computing $\vec{\nabla}_{\vec{\theta}} E(\vec{\theta}_t)$ at each step is expensive for large n.

[Various improvements]

(1.) Stochastic gradient descent (SGD) with mini-batches

Idea: incorporate stochasticity into GD while improving performance by computing the gradient on a subset of data called a mini-batch.

Specifically, divide n datapoints randomly into M minibatches s.t.

$\frac{n}{M} \simeq 10^1 - 10^2$.

↳ # datapoints in a minibatch

at each step, replace $\vec{\nabla}_{\vec{\theta}} E(\vec{\theta}_t)$ by

$$\sum_{i=1}^{n} \vec{\nabla}_{\vec{\theta}} \ell_i(\vec{x}_i, \vec{\theta}_t)$$

$$\vec{\nabla}_{\vec{\theta}} E^{MB}(\vec{\theta}_t) = \sum_{i \in B_k} \vec{\nabla}_{\vec{\theta}} \ell_i(\vec{x}_i, \vec{\theta}_t) \quad , \text{ where}$$

B_k is a set of all datapoints in minibatch k ($k = 1, \ldots, M$).

Then $$\vec{\theta}_{t+1} = \vec{\theta}_t - \overbrace{\eta_t \vec{\nabla}_{\vec{\theta}} E^{MB}(\vec{\theta})}^{\vec{v}_t} \qquad (3)$$

One full cycle over all M batches is called an <u>epoch</u>.

(2.) <u>SGD with momentum (SGDM)</u>

$$\begin{cases} \vec{v}_t = \gamma \vec{v}_{t-1} + \eta_t \vec{\nabla}^{MB_2}_{\vec{\theta}} E(\vec{\theta}_t), & (4) \\ \vec{\theta}_{t+1} = \vec{\theta}_t - \vec{v}_t \quad , \text{ where} \end{cases}$$

$0 \leq \gamma \leq 1$ is a momentum prm; for $\gamma = 0$, SGDM $\to$ SGD.

Eqs. (4) can be rewritten as

$$\underbrace{\vec{\theta}_{t+1} - \vec{\theta}_t}_{\Delta\vec{\theta}_{t+1}} = -\gamma \underbrace{\vec{v}_{t-1}}_{\vec{\theta}_{t-1} - \vec{\theta}_t = -\Delta\vec{\theta}_t} - \eta_t \vec{\nabla}^{MB}_{\vec{\theta}} E(\vec{\theta}_t) \quad , \text{ or}$$

$$\Delta\vec{\theta}_{t+1} = \gamma \Delta\vec{\theta}_t - \eta_t \vec{\nabla}_{\vec{\theta}} E^{MB}(\vec{\theta}_t) \qquad (4')$$

Clearly, $t=0$:

$$\begin{cases} \Delta\vec{\theta}_1 = -\eta_0 \vec{\nabla}_{\vec{\theta}} E^{MB}(\vec{\theta}_0), & \Leftarrow \Delta\vec{\theta}_0 = 0 \\ \Delta\vec{\theta}_2 = \gamma \Delta\vec{\theta}_1 - \eta_1 \vec{\nabla}_{\vec{\theta}} E^{MB}(\vec{\theta}_{\bullet 1}) = \\ \quad = -\gamma \eta_0 \vec{\nabla}_{\vec{\theta}} E^{MB}(\vec{\theta}_0) - \eta_1 \vec{\nabla}_{\vec{\theta}} E^{MB}(\vec{\theta}_1), \\ \Delta\vec{\theta}_3 = -\gamma^2 \eta_0 \vec{\nabla}_{\vec{\theta}} E^{MB}(\vec{\theta}_0) - \gamma \eta_1 \vec{\nabla}_{\vec{\theta}} E^{MB}(\vec{\theta}_1) - \\ \quad - \gamma^0 \eta_2 \vec{\nabla}_{\vec{\theta}} E^{MB}(\vec{\theta}_2), \\ \ldots \end{cases}$$

Thus $\Delta\vec{\theta}_t$ is a running average over previous gradients weighted by γ.

Empirically, the "inertia" term $\gamma \Delta\vec{\theta}_t$ which contributes to $\Delta\vec{\theta}_{t+1}$ along with the new gradient value (cf. (4')) helps "gain speed" in directions with persistent gradients, without jumping around too much.

a slight modification of SGDM:

$$\begin{cases} \vec{v}_t = \gamma \vec{v}_{t-1} + \eta_t \vec{\nabla}_{\vec{\theta}} E^{MB}(\vec{\theta}_t - \gamma \vec{v}_{t-1}), \\ \vec{\theta}_{t+1} = \vec{\theta}_t - \vec{v}_t. \end{cases} \quad (5)$$

$\vec{\nabla}_{\vec{\theta}} E^{MB}$ is evaluated at

$\vec{\theta}_t - \underbrace{\gamma \vec{v}_{t-1}}_{\vec{v}_t \text{ w/out the gradient term in (5)}}$ instead of $\vec{\theta}_t$.

$\vec{\theta}_{t+1}$ w/out the gradient term in (5)

(3.) Methods that use the 2nd moment of the gradient (RMSprop, ADAM)

SGD & SGDM still need η_t - a major drawback. Ideally, we would need an exact or approximate H (Hessian), but this is too expensive.

RMSprop, ADAM and other algorithms try to adapt step sizes to the landscape without computing H.

RMSprop:

$$\begin{cases} \vec{g}_t = \vec{\nabla}_{\vec{\theta}} E^{MB}(\vec{\theta}_t), \\ \vec{s}_{t,j} = \beta \vec{s}_{t-1,j} + (1-\beta)\, \vec{g}^{\,2}_{t,j}, \\ \vec{\theta}_{t+1,j} = \vec{\theta}_{t,j} - \eta_t \dfrac{\vec{g}_{t,j}}{\sqrt{s_{t,j} + \underbrace{\varepsilon}_{\sim 10^{-8}\ \text{regularizer}}}} \end{cases} \qquad (6)$$

Here, $0 \le \beta \le 1$ sets the scale of the running average for all elements of the $\vec{s}_t$ vector.

Clearly, w/out $\dfrac{1}{\sqrt{s_{t,j} + \varepsilon}}$ in the update rule we recover basic SGD.

If $s_{t,j}$ is large (meaning that $g^2_{t',j}$ was large for several previous steps), $\dfrac{\eta_t}{\sqrt{s_{t,j} + \varepsilon}}$ will be reduced and the algorithm will use smaller step sizes.

If however $s_{t,j}$ is small the step sizes will be large.

ADAM:

$$
(7)\quad
\begin{cases}
\vec{g}_t = \vec{\nabla}_{\vec{\theta}} E^{MB}(\vec{\theta}_t), \\
\vec{m}_t = \beta_1 \vec{m}_{t-1} + (1-\beta_1)\vec{g}_t, \\
S_{t,j} = \beta_2 S_{t-1,j} + (1-\beta_2) g^2_{t,j}, \\
\hat{\vec{m}}_t = \dfrac{\vec{m}_t}{1-\beta_1^t}, \\
\hat{\vec{S}}_t = \dfrac{\vec{S}_t}{1-\beta_2^t}, \\
\vec{\theta}_{t+1} = \vec{\theta}_t - \eta_t \dfrac{\hat{m}_t}{\sqrt{\hat{S}_t} + \underbrace{\xi}_{\text{regularizer}}}
\end{cases}
$$

Here, $\vec{m}_t$ & $\vec{S}_t$ are the usual running averages with the scales β_1 & β_2, respectively. $\hat{\vec{m}}_t$ & $\hat{\vec{S}}_t$ turn running averages into "true" (i.e., unbiased) averages.

For simplicity, consider a single prm θ.

Then $\underbrace{\sigma^2_t}_{\text{variance}} = \hat{S}_t - \underbrace{\hat{m}^2_t}_{\substack{\text{mean} \\ \text{squared}}}$, and

$$\Delta\theta_{t+1} = -\eta_t \frac{\hat{m}_t}{\sqrt{\sigma^2_t + \hat{m}^2_t} + \xi}$$

If $\underbrace{\sigma_t^2 \ll \hat{m}_t^2}_{\text{persistent gradient}}$ & $\hat{m}_t \gg \varepsilon$, we obtain

$\underbrace{\Delta\theta_{t+1} \to -\eta_t}_{\text{prm changes according to the maximum allowed step size, } \eta_t}$.

(i.e., the max allowed step is regularized)

If $\sigma_t^2 \gg \hat{m}_t^2$, $\Delta\theta_{t+1} \to -\eta_t \underbrace{\frac{\hat{m}_t}{\sigma_t}}_{\text{signal-to-noise ratio}}$

$\sigma_t^2 \gg \varepsilon$

[Here, σ_t serves as the natural adaptive scale]

[Unbiased (rescaled) averages:]

Consider a single prm for simplicity, s.t.

$m_0 = 0$,

$m_1 = \beta_1 m_0 + (1-\beta_1) g_1 = (1-\beta_1) g_1$,

$m_2 = \beta_1 m_1 + (1-\beta_1) g_2 = \beta_1 (1-\beta_1) g_1 + (1-\beta_1) g_2 =$

$= (1-\beta_1)[g_2 + \beta_1 g_1]$,

$m_3 = (1-\beta_1)[g_3 + \beta_1 g_2 + \beta_1^2 g_1]$, etc.

In general,

$$m_t = (1-\beta_1)\left[g_t + \beta_1 g_{t-1} + \beta_1^2 g_{t-2} + \ldots + \beta_1^{t-1} g_1 \right].$$

If $t=\infty$, the total weight of all the terms is

$$(1-\beta_1)\left[1 + \beta_1 + \beta_1^2 + \ldots \right] = (1-\beta_1)\frac{1}{1-\beta_1} = 1,$$

as expected.

With finite t, the weight is

$$(1-\beta_1)\left[1 + \beta_1 + \ldots + \beta_1^{t-1} \right] = (1-\beta_1)\frac{1-\beta_1^t}{1-\beta_1} =$$

$$= 1-\beta_1^t < 1.$$

To remove this bias, we rescale the series for m_t by $\frac{1}{1-\beta_1^t}$:

$$m_t \to \frac{m_t}{1-\beta_1^t} = \hat{m}_t.$$

Same argument holds for S_t and for multiple fitting prms.