Consider a cost function
\[ E(\theta) = \sum_{i=1}^{n} E_i(\vec{x}_i, \phi) \]

fitting params
input
datapoint

Gradient descent (GD):
\[ \Theta_{t+1} = \Theta_t - \eta_t \nabla_\theta E(\Theta_t) \]
where
\( t \) is the step # & \( \eta_t \) is the learning rate (may be changing with \( t \)).

Finally,
\[ (\nabla_\theta)_i = \frac{\partial}{\partial \theta_i} \]

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

Compare GD with Newton-Raphson (NR):
\[ \Theta_{t+1} = \Theta_t - H^{-1}(\Theta_t) \nabla_\theta E(\Theta_t) \]
where
\( H \) is the Hessian (in practice, \( H \) is often regularized: \( H^{-1} = (H + \epsilon I)^{-1} \), where \( \epsilon \) 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) $\eta_t$.

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 $\nabla_{\theta} E(\theta_t)$ 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 $s.t.$

   $$\frac{N}{M} \approx 10^1 - 10^2.$$

   # datapoints in a minibatch

   at each step, replace $\nabla_{\theta} E(\theta_t)$ by

   $$\frac{1}{M} \sum_{i=1}^{M} \nabla_{\theta} \ell_i(\theta_t, \bar{x}_i, \bar{\theta}_t).$$
\[ \nabla_{\tilde{\theta}} E^{MB}(\tilde{\theta}_t) = \sum_{i \in \mathcal{B}_k} \nabla_{\tilde{\theta}} l_i(\tilde{x}_i, \tilde{\theta}_t), \text{ where} \]

\[ \mathcal{B}_k \text{ is a set of all datapoints in minibatch } k \ (k=1, \ldots, M). \]

Then
\[
\tilde{\theta}_{t+1} = \tilde{\theta}_t - \eta_t \nabla_{\tilde{\theta}} E^{MB}(\tilde{\theta}_t) \tag{3}
\]

One full cycle over all M batches is called an epoch.

2. **SGD with momentum (SGDM)**

\[
\begin{cases}
    \tilde{v}_t = \gamma \tilde{v}_{t-1} + \eta_t \nabla_{\tilde{\theta}} E(\tilde{\theta}_t), \\
    \tilde{\theta}_{t+1} = \tilde{\theta}_t - \tilde{v}_t,
\end{cases} \tag{4}
\]

where

\[ 0 < \gamma \leq 1 \text{ is a momentum factor}; \text{ for } \gamma = 0, \text{ SGDM} \rightarrow \text{SGD}. \]

Eqs. (4) can be rewritten as

\[
\Delta \tilde{\theta}_{t+1} = -\gamma \Delta \tilde{\theta}_{t-1} - \eta_t \nabla_{\tilde{\theta}} E(\tilde{\theta}_t), \text{ or}
\]

\[ \Delta \tilde{\theta}_{t+1} = \gamma \Delta \tilde{\theta}_t - \eta_t \nabla_{\tilde{\theta}} E^{MB}(\tilde{\theta}_t) \tag{4'} \]
Clearly, \( t = 0 \):

\[
\begin{align*}
\Delta \bar{\theta}_1 &= -\eta_0 \nabla_\bar{\theta} \mathcal{E}^{MB}(\bar{\theta}_0), \quad \Leftarrow \Delta \bar{\theta}_0 = 0 \\
\Delta \bar{\theta}_2 &= \gamma \Delta \bar{\theta}_1 - \eta_1 \nabla_\bar{\theta} \mathcal{E}^{MB}(\bar{\theta}_1) = \\
&= -\gamma \eta_0 \nabla_\bar{\theta} \mathcal{E}^{MB}(\bar{\theta}_0) - \eta_1 \nabla_\bar{\theta} \mathcal{E}^{MB}(\bar{\theta}_1), \\
\Delta \bar{\theta}_3 &= -\gamma^2 \eta_0 \nabla_\bar{\theta} \mathcal{E}^{MB}(\bar{\theta}_0) - \gamma \eta_1 \nabla_\bar{\theta} \mathcal{E}^{MB}(\bar{\theta}_1) - \\
&\quad - \gamma^3 \eta_2 \nabla_\bar{\theta} \mathcal{E}^{MB}(\bar{\theta}_2),
\end{align*}
\]

Thus \( \Delta \bar{\theta}_t \) is a running average over previous gradients weighted by \( \gamma \).

Empirically, the "inertia" term \( \gamma \Delta \bar{\theta}_t \) which contributes to \( \Delta \bar{\theta}_{t+1} \) along with the new gradient value (eq. (4')) helps "gain speed" in directions with persistent gradients, without jumping around too much.
A slight modification of SGDM:

\[
\begin{aligned}
\vec{v}_t &= \gamma \vec{v}_{t-1} + \eta_t \nabla_\theta E^{MB}(\vec{\theta}_t \circ \gamma \vec{v}_{t-1}), \\
\vec{\theta}_{t+1} &= \vec{\theta}_t - \vec{v}_t.
\end{aligned}
\]  \hspace{1cm} (5)

\[\nabla_\theta E^{MB}\] is evaluated at

\[\vec{\theta}_t \circ \gamma \vec{v}_{t-1}\] instead of \[\vec{\theta}_t\].

\[\vec{v}_t\] is used in place of the gradient term in (5)

\[\vec{\theta}_{t+1}\] is used in place of the gradient term in (5)

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

SGD & SGDM still need \[\eta_t\] - a major drawback. Ideally, we would need an exact or approximate \(\nabla^2\) (Hessian), but this is too expensive.

RMSprop, ADAM, and other algorithms try to adapt step sizes to the landscape without computing \(\nabla^2\).
RMSprop: \[
\begin{align*}
\overline{g}_t &= \nabla_{\theta_t} \mathbb{E}^{MB}(\theta_t), \\
S_{t,j} &= \beta S_{t-1,j} + (1-\beta) g_{t,j}^2 \\
\theta_{t+1,j} &= \theta_{t,j} - \eta_t \frac{g_{t,j}}{\sqrt{S_{t,j} + \epsilon}} \\
\end{align*}
\]

Here, 0 ≤ β ≤ 1 sets the scale of the running average for all elements of the \( \theta_t \) vector.

Clearly, \( w/o \frac{1}{\sqrt{S_{t,j} + \epsilon}} \) in the update rule we recover basic SGD.

If \( S_{t,j} \) is large (meaning that \( g_{t,j}^2 \) was large for several previous steps), \( \eta_t \frac{g_{t,j}}{\sqrt{S_{t,j} + \epsilon}} \) 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:

\[
\begin{aligned}
\tilde{g}_t &= \nabla_{\theta} E^M_B (\bar{\theta}_t), \\
\bar{m}_t &= \beta_1 \bar{m}_{t-1} + (1-\beta_1) \tilde{g}_t, \\
\bar{s}_{t,j} &= \beta_2 \bar{s}_{t-1,j} + (1-\beta_2) \tilde{g}^2_{t,j}, \\
\hat{m}_t &= \frac{\bar{m}_t}{1-\beta_1^t}, \\
\hat{s}_t &= \frac{\bar{s}_t}{1-\beta_2^t}, \\
\bar{\theta}_{t+1} &= \bar{\theta}_t - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{s}_t} + \varepsilon}
\end{aligned}
\]

Here, \( \bar{m}_t \) and \( \bar{s}_t \) are the usual running averages with the scales \( \beta_1 \) and \( \beta_2 \), respectively. \( \hat{m}_t \) and \( \hat{s}_t \) turn running averages into "true" (i.e., unbiased) averages.

For simplicity, consider a single parameter \( \theta \).

Then \( \sigma^2_t = \hat{s}_t - \hat{m}_t^2 \), and

\[
\Delta \theta_{t+1} = -\eta_t \frac{\hat{m}_t}{\sqrt{\sigma^2_t + \hat{m}_t^2} + \varepsilon}
\]
$g_t^2 \ll \hat{m}_t^2$ & $\hat{m}_t \gg g$, we obtain

$\Delta \theta_{t+1} \rightarrow -\eta_t$.

$\hat{m}_t$ changes according to the maximum allowed step size, $\eta_t$ (i.e., the max allowed step is regularized).

Signal-to-noise ratio

Here, $g_t$ serves as the natural adaptive scale

[Unbiased averages: ]
(rescaled)

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,

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

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 \( \frac{1}{1 - \beta_1^t} \):

\[ M_t \rightarrow \frac{M_t}{1 - \beta_1^t} = \hat{M}_t . \]

Same argument holds for \( \hat{S}_t \) and for multiple fitting prms.