Discover acceleration of gradient descent

Seminar

Optimization for ML. Faculty of Computer Science. HSE University

GD. Convergence rates

$$\min_{x \in \mathbb{R}^n} f(x) \qquad x_{k+1} = x_k - \alpha_k \nabla f(x_k) \qquad \kappa = \frac{L}{\mu}$$

	smooth & convex	smooth & strongly convex (or PL)
Upper bound	$f(x_k) - f^* \approx \mathcal{O}\left(\frac{1}{k}\right)$	$\|x_k - x^*\|^2 \approx \mathcal{O}\left(\left(\frac{\kappa - 1}{\kappa + 1}\right)^k\right)$
Lower bound	$f(x_k) - f^* \approx \Omega\left(\frac{1}{k^2}\right)$	$\|x_k - x^*\|^2 \approx \Omega\left(\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k\right)$

Three update schemes

- **Normal gradient**

$$
\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)
$$

  Move the point $\boldsymbol{x}_k$ in the direction $-\nabla f(\boldsymbol{x}_k)$ for $\alpha_k \|\nabla f(\boldsymbol{x}_k)\|$ amount.

Three update schemes

- **Normal gradient**

$$\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

  Move the point $\boldsymbol{x}_k$ in the direction $-\nabla f(\boldsymbol{x}_k)$ for $\alpha_k \|\nabla f(\boldsymbol{x}_k)\|$ amount.

- **Polyak's Heavy Ball Method**

$$\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k) + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$$

  Perform a GD, move the updated-$\boldsymbol{x}$ in the direction of the previous step for $\beta_k \|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|$ amount.

Three update schemes

- **Normal gradient**

$$\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

Move the point $\boldsymbol{x}_k$ in the direction $-\nabla f(\boldsymbol{x}_k)$ for $\alpha_k \|\nabla f(\boldsymbol{x}_k)\|$ amount.

- **Polyak's Heavy Ball Method**

$$\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k) + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$$

Perform a GD, move the updated-$\boldsymbol{x}$ in the direction of the previous step for $\beta_k \|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|$ amount.

- **Nesterov's acceleration**

$$\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})) + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$$

Move the not-yet-updated-$\boldsymbol{x}$ in the direction of the previous step for $\beta_k \|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|$ amount, perform a GD on the shifted-$\boldsymbol{x}$, then move the updated-$\boldsymbol{x}$ in the direction of the previous step for $\beta_k \|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|$.

HBM for a quadratic problem

Question

Which step size strategy is used for GD?

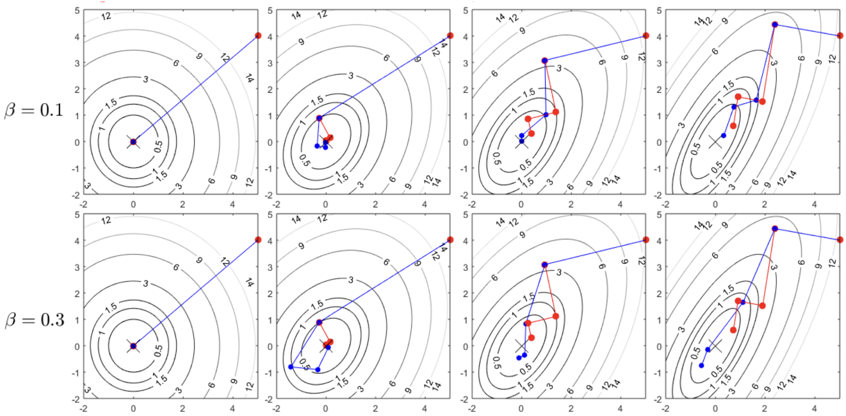

Figure 1: GD vs. HBM with fixed β.

Observation: for nice f (with spherical level sets), GD is already good enough and HBM adds a little effect. However, for bad f (with elliptic level sets), HBM is better in some cases.

HBM for a quadratic problem

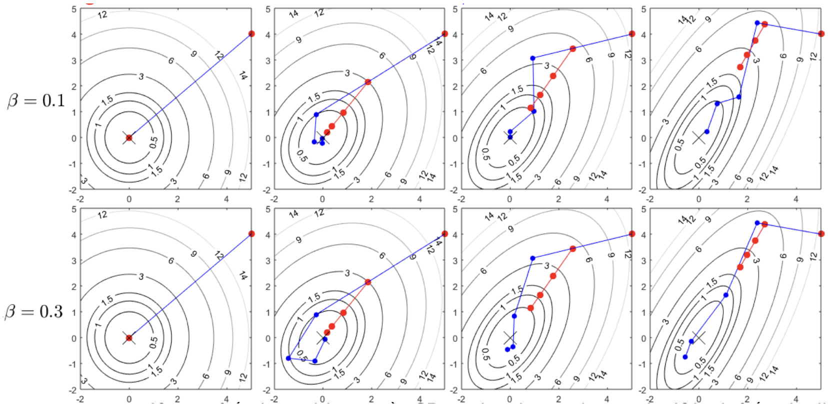

Figure 2: GD with $\alpha = \dfrac{1}{L}$ vs. HBM with fixed β.

Observation: same. If nice f (spherical lv. sets), GD is already good enough. If bad f (with elliptic lv. sets), HBM is better in some cases.

NAG as a Momentum Method

- Start by setting $k=0, a_0=1, \boldsymbol{x}_{-1}=\boldsymbol{y}_0, \boldsymbol{y}_0$ to an arbitrary parameter setting, iterates

$$\text{Gradient update } \boldsymbol{x}_k = \boldsymbol{y}_k - \alpha_k \nabla f(\boldsymbol{y}_k) \tag{1}$$

$$\text{Extrapolation weight } a_{k+1} = \frac{1+\sqrt{1+4a_k^2}}{2} \tag{2}$$

$$\text{Extrapolation } \boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \frac{a_k - 1}{a_{k+1}}(\boldsymbol{x}_k - \boldsymbol{x}_{k+1}) \tag{3}$$

  Note that here fix step-size is used: $\alpha_k = \dfrac{1}{L}\ \forall k$.
- **Theorem.** If f is L-smooth and convex, the sequence $\{f(\boldsymbol{x}_k)\}_k$ produced by NAG convergences to the optimal value f^* as the rate $\mathcal{O}(\dfrac{1}{k^2})$ as

$$f(\boldsymbol{x}_k) - f^* \leq \frac{4L\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2}{(k+2)^2}$$

- The above representation is difficult to understand, so we will rewrite these equations in a more intuitive manner.

NAG as a Momentum Method

If we define

$$\boldsymbol{v}_k \equiv \boldsymbol{x}_k - \boldsymbol{x}_{k-1} \tag{4}$$

$$\beta_k \equiv \frac{a_k - 1}{a_{k+1}} \tag{5}$$

then the combination of Equation 3 and Equation 5 implies:

$$\boldsymbol{y}_k = \boldsymbol{x}_{k-1} + \beta_{k-1}\boldsymbol{v}_{k-1}$$

which can be used to rewrite Equation 1 as follows using $\alpha_k = \alpha_{k-1}$:

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \beta_{k-1}\boldsymbol{v}_{k-1} - \alpha_{k-1}\nabla f(\boldsymbol{x}_{k-1} + \beta_{k-1}\boldsymbol{v}_{k-1}) \tag{6}$$

$$\boldsymbol{v}_k = \beta_{k-1}\boldsymbol{v}_{k-1} - \alpha_{k-1}\nabla f(\boldsymbol{x}_{k-1} + \beta_{k-1}\boldsymbol{v}_{k-1}) \tag{7}$$

where Equation 7 is a consequence of Equation 4. Alternatively:

$$\boldsymbol{v}_{k+1} = \beta_k\boldsymbol{v}_k - \alpha_k\nabla f(\boldsymbol{x}_k + \beta_k\boldsymbol{v}_k)$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{v}_{k+1}$$

where $\alpha_k > 0$ is the **learning rate**, β_k is the **momentum coefficient**. Compare HBM with NAG.

NAG for a quadratic problem

Consider the following quadratic optimization problem:

$$\min_{x \in \mathbb{R}^d} q(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x, \text{ where } A \in \mathbb{S}^d_{++}.$$

Every symmetric matrix A has an eigenvalue decomposition

$$A = Q \text{diag}\left(\lambda_1, \ldots, \lambda_n\right) Q^T = Q \Lambda Q^T, \quad Q = [q_1, \ldots, q_n].$$

and, as per convention, we will assume that the λ_i's are sorted, from smallest λ_1 to biggest λ_n. It is clear, that λ_i correspond to the **curvature** along the associated eigenvector directions.

We can reparameterize $q(x)$ by the matrix transform Q and optimize $y = Qx$ using the objective

$$p(y) \equiv q(x) = q(Q^\top y) = y^\top Q(Q^\top \Lambda Q)Q^\top y/2 - b^\top Q^\top y = y^\top \Lambda y/2 - c^\top y,$$

where $c = Qb$.

We can further rewrite p as

$$p(y) = \sum_{i=1}^{n} [p]_i([y]_i),$$

where $[p]_i(t) = \lambda_i t^2/2 - [c]_i t$.

NAG for a quadratic problem

Theorem 2.1 from [1].

Let $p(y) = \sum_{i=1}^{n} [p]_i([y]_i)$ such that $[p]_i(t) = \lambda_i t^2/2 - [c]_i t$. Let α be arbitrary and fixed. Denote by $\text{HBM}_x(\beta, p, y, v)$ and $\text{HBM}_v(\beta, p, y, v)$ the parameter vector and the velocity vector respectively, obtained by applying one step of HBM (i.e., Eq. 1 and then Eq. 2) to the function p at point y, with velocity v, momentum coefficient β, and learning rate α. Define NAG_x and NAG_v analogously. Then the following holds for $z \in \{x, v\}$:

$$\text{HBM}_z(\beta, p, y, v) = \begin{bmatrix} \text{HBM}_z(\beta, [p]_1, [y]_1, [v]_1) \\ \vdots \\ \text{HBM}_z(\beta, [p]_n, [y]_n, [v]_n) \end{bmatrix}$$

$$\text{NAG}_z(\beta, p, y, v) = \begin{bmatrix} \text{HBM}_z(\beta(1 - \alpha\lambda_1), [p]_1, [y]_1, [v]_1) \\ \vdots \\ \text{HBM}_z(\beta(1 - \alpha\lambda_n), [p]_n, [y]_n, [v]_n) \end{bmatrix}$$

NAG for a quadratic problem. Proof (1/2)

Proof:

It's easy to show that if

$$x_{i+1} = \text{HBM}_x(\beta_i, [q]_i, [x]_i, [v]_i)$$
$$v_{i+1} = \text{HBM}_v(\beta_i, [q]_i, [x]_i, [v]_i)$$

then for $y_i = Qx_i, w_i = Qv_i$

$$y_{i+1} = \text{HBM}_x(\beta_i, [p]_i, [y]_i, [w]_i)$$
$$w_{i+1} = \text{HBM}_v(\beta_i, [p]_i, [y]_i, [w]_i)$$

. Then, consider one step of HBM_v:

$$\begin{aligned}
&\text{HBM}_v(\beta, p, y, v) = \beta v - \alpha \nabla p(y) \\
&= (\beta[v]_1 - \alpha \nabla_{[y]_1} p(y), \ldots, \beta[v]_n - \alpha \nabla_{[y]_n} p(y)) \\
&= (\beta[v]_1 - \alpha \nabla [p]_1([y]_1), \ldots, \beta[v]_n - \alpha \nabla [p]_n([y]_n)) \\
&= (\text{HBM}_v(\beta_1, [p]_1, [y]_1, [v]_1), \ldots, \text{HBM}_v(\beta_i, [p]_i, [y]_i, [v]_i))
\end{aligned}$$

This shows that one step of HBM_v on p is precisely equivalent to n simultaneous applications of HBM_v to the one-dimensional quadratics $[p]_i$, all with the same β and α. Similarly, for HBM_x.

NAG for a quadratic problem. Proof (2/2)

Next we show that NAG, applied to a one-dimensional quadratic with a momentum coefficient β, is equivalent to HBM applied to the same quadratic and with the same learning rate, but with a momentum coefficient $\beta(1-\alpha\lambda)$. We show this by expanding $\text{NAG}_v(\beta, [p]_i, y, v)$ (where y and v are scalars):

$$
\begin{aligned}
\text{NAG}_v(\beta, [p]_i, y, v) &= \beta v - \alpha \nabla [p]_i (y + \beta v) \\
&= \beta v - \alpha(\lambda_i (y + \beta v) - c_i) \\
&= \beta v - \alpha \lambda_i \beta v - \alpha(\lambda_i y - c_i) \\
&= \beta(1 - \alpha\lambda_i) v - \alpha \nabla [p]_i (y) \\
&= \text{HBM}_v(\beta(1-\alpha\lambda_i), [p]_i, y, v).
\end{aligned}
$$

QED.

Observations:

- HBM and NAG become **equivalent** when α is small (when $\alpha\lambda \ll 1$ for every eigenvalue λ of A), so NAG and HBM are distinct only when α is reasonably large.
- When α is relatively large, NAG uses smaller effective momentum for the high-curvature eigen-directions, which **prevents oscillations** (or divergence) and thus allows the use of a larger β than is possible with CM for a given α.

NAG for DL

task	$0_{(\text{SGD})}$	0.9N	0.99N	0.995N	0.999N	0.9M	0.99M	0.995M	0.999M	SGD_C	$\text{HF}^\dagger$	HF^*
Curves	0.48	0.16	0.096	0.091	**0.074**	0.15	0.10	0.10	0.10	0.16	0.058	0.11
Mnist	2.1	1.0	**0.73**	0.75	0.80	1.0	0.77	0.84	0.90	0.9	0.69	1.40
Faces	36.4	14.2	8.5	7.8	**7.7**	15.3	8.7	8.3	9.3	NA	7.5	12.0

Figure 3: The table reports the squared errors on the problems for each combination of β_{max} and a momentum type (NAG, CM). When β_{max} is 0 the choice of NAG vs CM is of no consequence so the training errors are presented in a single column. For each choice of β_{max}, the highest-performing learning rate is used. The column SGD_C lists the results of Chapelle & Erhan (2011) who used 1.7M SGD steps and tanh networks. The column $\text{HF}^\dagger$ lists the results of HF without L2 regularization; and the column HF^* lists the results of Martens (2010).

References and Python Examples

- Figures for HBM was taken from the presentation. Visit site for more tutorials.
- Why Momentum Really Works. Link.
- Run code in Colab. The code taken from .
- On the importance of initialization and momentum in deep learning. Link.