

Discover acceleration of gradient descent

Seminar

Optimization for ML. Faculty of Computer Science. HSE University

GD. Convergence rates

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

	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**

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

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

Three update schemes

- **Normal gradient**

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

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

- **Polyak's Heavy Ball Method**

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

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

Three update schemes

- Normal gradient

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

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

- Polyak's Heavy Ball Method

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

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

- Nesterov's acceleration

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

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

Black box iteration

The iteration of gradient descent:

$$\begin{aligned}x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\&\vdots \\&= x^0 - \sum_{i=0}^k \alpha^{k-i} \nabla f(x^{k-i})\end{aligned}$$

Black box iteration

The iteration of gradient descent:

$$\begin{aligned}x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\&\vdots \\&= x^0 - \sum_{i=0}^k \alpha^{k-i} \nabla f(x^{k-i})\end{aligned}$$

Consider a family of first-order methods, where

$$\begin{aligned}x^{k+1} &\in x^0 + \text{span} \{ \nabla f(x^0), \nabla f(x^1), \dots, \nabla f(x^k) \} && f - \text{smooth} \\x^{k+1} &\in x^0 + \text{span} \{ g_0, g_1, \dots, g_k \}, \text{ where } g_i \in \partial f(x^i) && f - \text{non-smooth}\end{aligned} \tag{1}$$

Black box iteration

The iteration of gradient descent:

$$\begin{aligned}x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\&\vdots \\&= x^0 - \sum_{i=0}^k \alpha^{k-i} \nabla f(x^{k-i})\end{aligned}$$

Consider a family of first-order methods, where

$$\begin{aligned}x^{k+1} &\in x^0 + \text{span} \{ \nabla f(x^0), \nabla f(x^1), \dots, \nabla f(x^k) \} && f - \text{smooth} \\x^{k+1} &\in x^0 + \text{span} \{ g_0, g_1, \dots, g_k \}, \text{ where } g_i \in \partial f(x^i) && f - \text{non-smooth}\end{aligned} \tag{1}$$

In order to construct a lower bound, we need to find a function f from corresponding class such that any method from the family 1 will work at least as slow as the lower bound.

Smooth case

Theorem

There exists a function f that is L -smooth and convex such that any method 1 satisfies for any $k : 1 \leq k \leq \frac{n-1}{2}$:

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

Smooth case

i Theorem

There exists a function f that is L -smooth and convex such that any method 1 satisfies for any $k : 1 \leq k \leq \frac{n-1}{2}$:

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

- No matter what gradient method you provide, there is always a function f that, when you apply your gradient method on minimizing such f , the convergence rate is lower bounded as $\mathcal{O}\left(\frac{1}{k^2}\right)$.
- The key to the proof is to explicitly build a special function f .
- Note, that this bound $\mathcal{O}\left(\frac{1}{k^2}\right)$ does not match the rate of gradient descent $\mathcal{O}\left(\frac{1}{k}\right)$. Two options possible:
 - a. The lower bound is not tight.
 - b. The gradient method is not optimal for this problem.

Smooth case

i Theorem

There exists a function f that is L -smooth and convex such that any method 1 satisfies for any $k : 1 \leq k \leq \frac{n-1}{2}$:

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

- No matter what gradient method you provide, there is always a function f that, when you apply your gradient method on minimizing such f , the convergence rate is lower bounded as $\mathcal{O}\left(\frac{1}{k^2}\right)$.
- The key to the proof is to explicitly build a special function f .
- Note, that this bound $\mathcal{O}\left(\frac{1}{k^2}\right)$ does not match the rate of gradient descent $\mathcal{O}\left(\frac{1}{k}\right)$. Two options possible:
 - a. The lower bound is not tight.
 - b. The gradient method is not optimal for this problem.

Smooth case

i Theorem

There exists a function f that is L -smooth and convex such that any method 1 satisfies for any $k : 1 \leq k \leq \frac{n-1}{2}$:

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

- No matter what gradient method you provide, there is always a function f that, when you apply your gradient method on minimizing such f , the convergence rate is lower bounded as $\mathcal{O}\left(\frac{1}{k^2}\right)$.
- The key to the proof is to explicitly build a special function f .
- Note, that this bound $\mathcal{O}\left(\frac{1}{k^2}\right)$ does not match the rate of gradient descent $\mathcal{O}\left(\frac{1}{k}\right)$. Two options possible:
 - a. The lower bound is not tight.
 - b. The gradient method is not optimal for this problem.

Smooth case

Theorem

There exists a function f that is L -smooth and convex such that any method 1 satisfies for any $k : 1 \leq k \leq \frac{n-1}{2}$:

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

- No matter what gradient method you provide, there is always a function f that, when you apply your gradient method on minimizing such f , the convergence rate is lower bounded as $\mathcal{O}\left(\frac{1}{k^2}\right)$.
- The key to the proof is to explicitly build a special function f .
- Note, that this bound $\mathcal{O}\left(\frac{1}{k^2}\right)$ does not match the rate of gradient descent $\mathcal{O}\left(\frac{1}{k}\right)$. Two options possible:
 - a. The lower bound is not tight.
 - b. The gradient method is not optimal for this problem.

Smooth case

i Theorem

There exists a function f that is L -smooth and convex such that any method 1 satisfies for any $k : 1 \leq k \leq \frac{n-1}{2}$:

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

- No matter what gradient method you provide, there is always a function f that, when you apply your gradient method on minimizing such f , the convergence rate is lower bounded as $\mathcal{O}\left(\frac{1}{k^2}\right)$.
- The key to the proof is to explicitly build a special function f .
- Note, that this bound $\mathcal{O}\left(\frac{1}{k^2}\right)$ does not match the rate of gradient descent $\mathcal{O}\left(\frac{1}{k}\right)$. Two options possible:
 - a. The lower bound is not tight.
 - b. The gradient method is not optimal for this problem.

Smooth case

i Theorem

There exists a function f that is L -smooth and convex such that any method 1 satisfies for any $k : 1 \leq k \leq \frac{n-1}{2}$:

$$f(x^k) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(k+1)^2}$$

- No matter what gradient method you provide, there is always a function f that, when you apply your gradient method on minimizing such f , the convergence rate is lower bounded as $\mathcal{O}\left(\frac{1}{k^2}\right)$.
- The key to the proof is to explicitly build a special function f .
- Note, that this bound $\mathcal{O}\left(\frac{1}{k^2}\right)$ does not match the rate of gradient descent $\mathcal{O}\left(\frac{1}{k}\right)$. Two options possible:
 - a. The lower bound is not tight.
 - b. **The gradient method is not optimal for this problem.**

HBM for a quadratic problem

Question

Which step size strategy is used for **GD**?

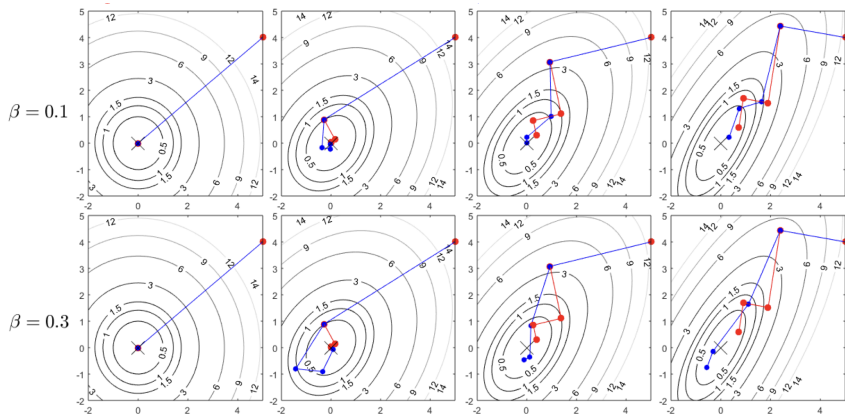


Figure 1: **GD** vs. **HBM** with fixed β .

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

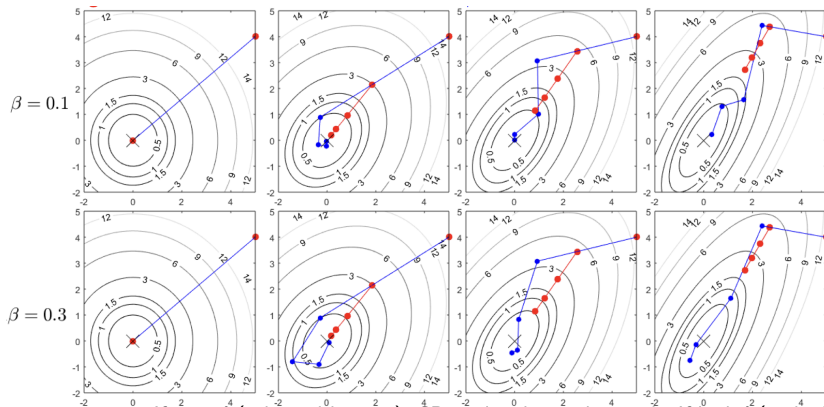


Figure 2: GD with $\alpha = \frac{1}{L}$ vs. HBM with fixed β .

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

$$\text{Gradient update } \mathbf{x}_k = \mathbf{y}_k - \alpha_k \nabla f(\mathbf{y}_k) \quad (2)$$

$$\text{Extrapolation weight } a_{k+1} = \frac{1 + \sqrt{1 + 4a_k^2}}{2} \quad (3)$$

$$\text{Extrapolation } \mathbf{y}_{k+1} = \mathbf{x}_k + \frac{a_k - 1}{a_{k+1}} (\mathbf{x}_k - \mathbf{x}_{k-1}) \quad (4)$$

Note that here fix step-size is used: $\alpha_k = \frac{1}{L} \forall k$.

- Theorem.** If f is L -smooth and convex, the sequence $\{f(\mathbf{x}_k)\}_k$ produced by NAG converges to the optimal value f^* as the rate $\mathcal{O}(\frac{1}{k^2})$ as

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

- 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

$$\mathbf{v}_k \equiv \mathbf{x}_k - \mathbf{x}_{k-1} \quad (5)$$

$$\beta_k \equiv \frac{a_k - 1}{a_{k+1}} \quad (6)$$

then the combination of Equation 4 and Equation 6 implies:

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

which can be used to rewrite Equation 2 as follows using $\alpha_k = \alpha_{k-1}$:

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

$$\mathbf{v}_k = \beta_{k-1} \mathbf{v}_{k-1} - \alpha_{k-1} \nabla f(\mathbf{x}_{k-1} + \beta_{k-1} \mathbf{v}_{k-1}) \quad (8)$$

where Equation 8 is a consequence of Equation 5. Alternatively:

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

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

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

task	$0_{(\text{SGD})}$	0.9N	0.99N	0.995N	0.999N	0.9M	0.99M	0.995M	0.999M	SGD _C
Curves	0.48	0.16	0.096	0.091	0.074	0.15	0.10	0.10	0.10	0.16
Mnist	2.1	1.0	0.73	0.75	0.80	1.0	0.77	0.84	0.90	0.9
Faces	36.4	14.2	8.5	7.8	7.7	15.3	8.7	8.3	9.3	NA

Figure 3: The table reports the squared errors on the problems for each combination of β_{max} and a momentum type (NAG=N, HB=M). When β_{max} is 0 the choice of NAG vs HB 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.

¹Link

Heavy Ball Global Convergence²

i Theorem

Assume that f is smooth and convex and that

$$\beta \in [0, 1), \quad \alpha \in \left(0, \frac{2(1-\beta)}{L}\right).$$

Then, the sequence $\{x_k\}$ generated by Heavy-ball iteration satisfies

$$f(\bar{x}_T) - f^* \leq \begin{cases} \frac{\|x_0 - x^*\|^2}{2(T+1)} \left(\frac{L\beta}{1-\beta} + \frac{1-\beta}{\alpha} \right), & \text{if } \alpha \in \left(0, \frac{1-\beta}{L}\right], \\ \frac{\|x_0 - x^*\|^2}{2(T+1)(2(1-\beta) - \alpha L)} \left(L\beta + \frac{(1-\beta)^2}{\alpha} \right), & \text{if } \alpha \in \left[\frac{1-\beta}{L}, \frac{2(1-\beta)}{L}\right), \end{cases}$$

where \bar{x}_T is the Cesaro average of the iterates, i.e.,

$$\bar{x}_T = \frac{1}{T+1} \sum_{k=0}^T x_k.$$

²Global convergence of the Heavy-ball method for convex optimization, Euhanna Ghadimi et.al.

Heavy Ball Global Convergence³

i Theorem

Assume that f is smooth and strongly convex and that

$$\alpha \in (0, \frac{2}{L}), \quad 0 \leq \beta < \frac{1}{2} \left(\frac{\mu\alpha}{2} + \sqrt{\frac{\mu^2\alpha^2}{4} + 4(1 - \frac{\alpha L}{2})} \right).$$

Then, the sequence $\{x_k\}$ generated by Heavy-ball iteration converges linearly to a unique optimizer x^* . In particular,

$$f(x_k) - f^* \leq q^k (f(x_0) - f^*),$$

where $q \in [0, 1)$.

³Global convergence of the Heavy-ball method for convex optimization, Euhanna Ghadimi et.al.

NAG Global Convergence

i Theorem

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex and L -smooth. The Nesterov Accelerated Gradient Descent (NAG) algorithm is designed to solve the minimization problem starting with an initial point $x_0 = y_0 \in \mathbb{R}^n$ and $\lambda_0 = 0$. The algorithm iterates the following steps:

Gradient update: $y_{k+1} = x_k - \frac{1}{L} \nabla f(x_k)$

Extrapolation: $x_{k+1} = (1 - \gamma_k)y_{k+1} + \gamma_k y_k$

Extrapolation weight: $\lambda_{k+1} = \frac{1 + \sqrt{1 + 4\lambda_k^2}}{2}$

Extrapolation weight: $\gamma_k = \frac{1 - \lambda_k}{\lambda_{k+1}}$

The sequences $\{f(y_k)\}_{k \in \mathbb{N}}$ produced by the algorithm will converge to the optimal value f^* at the rate of $\mathcal{O}\left(\frac{1}{k^2}\right)$, specifically:

$$f(y_k) - f^* \leq \frac{2L\|x_0 - x^*\|^2}{k^2}$$

NAG Global Convergence

i Theorem

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is μ -strongly convex and L -smooth. The Nesterov Accelerated Gradient Descent (NAG) algorithm is designed to solve the minimization problem starting with an initial point $x_0 = y_0 \in \mathbb{R}^n$ and $\lambda_0 = 0$. The algorithm iterates the following steps:

Gradient update: $y_{k+1} = x_k - \frac{1}{L} \nabla f(x_k)$


Extrapolation: $x_{k+1} = (1 + \gamma_k)y_{k+1} - \gamma_k y_k$

Extrapolation weight: $\gamma_k = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$


The sequences $\{f(y_k)\}_{k \in \mathbb{N}}$ produced by the algorithm will converge to the optimal value f^* linearly:

$$f(y_k) - f^* \leq \frac{\mu + L}{2} \|x_0 - x^*\|_2^2 \exp\left(-\frac{k}{\sqrt{\kappa}}\right)$$



Hobbits

Let's code!  Colab

Logistic Regression

Let's code!  Colab

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](#).

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

$$A = Q \text{diag}(\lambda_1, \dots, \lambda_n) Q^\top = Q \Lambda Q^\top, \quad Q = [q_1, \dots, q_n].$$

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 :

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

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 .