Selecting the stepsizes in Polyak's Momentum algorithm by quadratic planar search

Luigi Grippo, Matteo Lapucci, Sefano Lucidi, Marco Sciandrone

DINFO, Università degli Studi di Firenze,
DIAG, Sapienza Università di Roma

1 Momentum with Inexact Plane Search: A Summary

The classical gradient descent (GD) algorithm is defined by the iteration

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k),$$

where α_k is a suitable stepsize usually computed performing a (inexact) line search (e.g., Armijo or Wolfe methods [1]). More general first-order optimizers carry out iterations of the form

$$x^{k+1} = x^k - \alpha_k H_k \nabla f(x^k), \tag{1}$$

where H_k is a positive definite matrix; of course selecting $H_k = I$ leads back to gradient descent, whereas $H_k \approx (\nabla^2 f(x^k))^{-1}$ gives rise to Quasi-Newton methods [1]. [Add longer discussion on first-order method].

The idea of introducing H_k is to exploit curvature information to improve the quality of the search direction.

A different strategy to speed up the computation was firstly proposed by Polyak [5]; the idea of Polyak consists of exploiting information about past iterates to speed up the process; in particular, the iterations of Polyak Heavy-ball or Momentum method have the form:

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k) + \beta(x^k - x^{k-1}). \tag{2}$$

The idea of the above update is that the direction of last iteration will arguably be a good descent direction even at the current one. Partly repeating the previous step helps with controlling oscillation and providing acceleration in low curvature regions.

The main computational advantage of introducing the *momentum* term lies in the fact that it only exploits already computed information: no additional evaluations of the objective function or, even worse, the gradient, are required.

In the case of strictly convex quadratic functions, the "optimal" values [argomentare meglio su cosa si intende per ottimali] can be determined, a priori, as follows [6]:

$$\alpha^\star = \frac{4}{(\sqrt{\lambda_M} + \sqrt{\lambda_m})^2}, \qquad \beta^\star = \left(\frac{\sqrt{\lambda_M} - \sqrt{\lambda_m})}{\sqrt{\lambda_M} + \sqrt{\lambda_m}}\right)^2$$

where λ_M and λ_m are the largest and smallest eigenvalues of the Hessian matrix, respectively. The choice of this optimal parameters allows to obtain a local accelerated linear convergence rate with iteration error $\mathcal{O}((\frac{\sqrt{\lambda_M}-\sqrt{\lambda_m}}{\sqrt{\lambda_M}+\sqrt{\lambda_m}})^k)$. [Inserire qui commento su equivalenza con gradiente coniugato; io non sono riuscito a trovare riferimenti]

The (local) linear convergence rate (with better factor than both GD and Nesterov's method [parlare brevemente di nesterov da qualche parte]), can be generalized under hypotheses of twice continuous differentiability, Lipschitz-continuous gradient and strong convexity [5].

Sufficient conditions for the heavy-ball method to converge to critical points in the nonconvex setting have been proven by [8]. However, counterexamples have been recently provided [4] to show that the heavy-ball method does not necessarily converge a) in the convex case under twice differentiability assumptions and b) in the strongly convex (but not twice differentiable) case, even if one chooses Polyak's optimal step-size parameters. As of today, no evidence has been found that Polyak's method outperforms Nesterov's method, or even the basic GD, when the objective is not twice continuously differentiable.

Under convexity and smoothness assumptions, the ergodic $\mathcal{O}(1/k)$, i.e., $f(\sum_{i=1}^{k} \frac{x^i}{k}) - \min f = \mathcal{O}(1/k)$, has been proved [3]. In the same work, the algorithm is proved to be globally convergent with linear speed in the strongly convex case (even though for β in a possibly small range of values).

Due to the inertial term $\beta(x^k - x^{k-1})$, momentum, in contrast to gradient descent, does not satisfy Fejér monotonicity [2]. Hence, it is difficult to prove its non-ergodic convergence rates. To this end, a novel Lyapunov analysis was designed and better convergence results were derived [7], like the non-ergodic sublinear convergence rate $f(x^k) - \min f = \mathcal{O}(1/k)$ and larger step size for linear convergence.

Except for the quadratic case, the selection of parameters α and β is not trivial; α might be chosen by a line search, whereas β is usually set to a constant chosen somewhat heuristically; some adaptive strategies have also been proposed in recent years [approfondire un po' questa cosa, serve per motivare in modo più robusto la nostra idea]

The proposal of this paper consists in a novel approach for the selection of the two parameters α and β in (2). Similarly as what is commonly done for the selection of the stepsize α in algorithms of the form (1), here we try to (approximately) solve the optimization problem in two variables

$$\min_{\alpha,\beta} \phi(\alpha,\beta) = f(x^k - \alpha\nabla f(x^k) + \beta(x^k - x^{k-1})),$$

or, more generally,

$$\min_{\alpha,\beta} \phi(\alpha,\beta) = f(x^k + \alpha d_1 + \beta d_2).$$

Computing the objective function and the gradient for this subproblem unfortunately requires to compute the corresponding function value and gradients of the original function of n variables at points we have no information about:

$$\phi_k(\alpha,\beta) = f(x^k + \alpha d_1 + \beta d_2)$$

$$\nabla\phi_k(\alpha,\beta) = (\nabla f(x^k + \alpha d_1 + \beta d_2)^T d_1, \nabla f(x^k + \alpha d_1 + \beta d_2)^T d_2)^T = D\nabla f(x^k + \alpha d_1 + \beta d_2),$$

where $D = (d_1, d_2)^T$. For the sake of completeness, let us remark that

$$\nabla^2\phi_k(\alpha,\beta) = D\nabla^2 f(x^k + \alpha d_1 + \beta d_2)D^T.$$

Selecting the step sizes solving, even inexactly, the above subproblem, we can reasonably expect to reduce the number of iterations of algorithm REF [this claim is supported by computational experiments]; however, at each iteration of the solver employed to tackle the subproblem, f and ∇f of the original problem will be computed several times, leading to an overall increase in the computational cost of the algorithm [this claim is also supported by numerical evidence].

Ideally, we would like the subproblem to be an actual two-variables problem, for which computing the function value or the gradient is cheap. Our idea is to exploit Taylor's approximation to construct a surrogate function to optimize in the subproblem. [dire che l'approssimazione al primo ordine dava risultati schifosi...]

The second order expansion of $\phi_k(\alpha,\beta)$ around $(0,0)$ is given by:

$$\phi_k(\alpha,\beta) \approx \phi_k(0,0) + \nabla\phi_k(0,0)^T(\alpha,\beta) + \frac{1}{2}(\alpha,\beta)^T\nabla^2\phi_k(0,0)(\alpha,\beta),$$

i.e.,

$$\phi_k(\alpha,\beta) \approx f(x^k) + (D\nabla f(x^k))^T(\alpha,\beta) + \frac{1}{2}(\alpha,\beta)^T D\nabla^2 f(x^k)D^T(\alpha,\beta).$$

Now, at iteration k we clearly compute $f(x^k)$ and $\nabla f(x^k)$, so the constant and the linear terms in the above approximation are basically known; on the other hand, we obviously do not want to compute the Hessian matrix at x^k, thus we do not have access to the exact values of $\nabla^2\phi_k(0,0)$. However, we can estimate the three degrees of freedom for the (symmetric) quadratic part of the approximation by interpolation [strategia alla Powell?]. Of course, interpolation at $(\alpha,\beta)=(0,0)$ does not provide information, as we only get the trivial equality $f(x^k)=f(x^k)$. We can exploit the information from iteration $k-1$, as we clearly know $f(x^{k-1})$ and $\nabla f(x^{k-1})$: x^{k-1} indeed lies on the plane generated by $d_1=-\nabla f(x^k)$ and $d_2=(x^k-x^{k-1})$ passing through x^k, specifically obtainable for $(\alpha,\beta)=(0,-1)$. Thus, in order to define the quadratic subproblem in two variables, we only need to compute the objective value at two additional points, which is a reasonable cost, comparable to that of a common line-search.

Two reasonable candidate points to evaluate the function at might be, for example, $(\alpha_{k-1},\beta_{k-1})$ and $(\alpha_{k-1},0)$. The interpolation system to obtain the quadratic matrix B,

$$\frac{1}{2}\begin{pmatrix} 0 & 0 & 1\\ \alpha_{k-1}^2 & 0 & 0\\ \alpha_{k-1}^2 & 2\alpha_{k-1}\beta_{k-1} & \beta_{k-1}^2 \end{pmatrix}\begin{pmatrix} B_{11}\\ B_{12}\\ B_{22}\end{pmatrix}=\begin{pmatrix} f(x^{k-1})-f(x^k)+\nabla_\beta\phi_k(0,0)\\ f(x^k+\alpha_{k-1}d_1)-f(x^k)-\alpha_{k-1}\nabla_\alpha\phi_k(0,0)\\ f(x^k+\alpha_{k-1}d_1+\beta_{k-1}d_2)-f(x^k)-\alpha_{k-1}\nabla_\alpha\phi_k(0,0)-\beta_{k-1}\nabla_\beta\phi_k(0,0)\end{pmatrix}, \tag{3}$$

can be easily solved in closed form:

$$B_{11}=\frac{2}{\alpha_{k-1}^2}(f(x^k+\alpha_{k-1}d_1)-\alpha_{k-1}\nabla_\alpha\phi_k(0,0)-f(x^k))$$
$$B_{22}=2(\nabla_\beta\phi_k(0,0)+f(x^{k-1})-f(x^k))$$
$$B_{12}=B_{21}=\frac{f(x^k+\alpha_{k-1}d_1+\beta_{k-1}d_2)-f(x^k)-\alpha_{k-1}\nabla_\alpha\phi_k(0,0)-\beta_{k-1}\nabla_\beta\phi_k(0,0)-0.5\alpha_{k-1}^2B_{11}-0.5\beta_{k-1}^2B_{22}}{\alpha_{k-1}\beta_{k-1}}$$

Then, α_k and β_k can be obtained solving the simple linear system in two variables

$$B(\alpha,\beta)=-\nabla\phi(0,0).$$

The overall resulting algorithm is reported here below

Algorithm 1: `Momentum with Plane Search`

```
1  Input: $x^0, x^{-1}$, $\alpha_{-1}$, $\beta_{-1}$.
2  Set $k=0$
3  while $\|\nabla f(x^k)\| \leq \epsilon$ do
4      Set $d_1^k=-\nabla f(x^k)$, $d_2^k=(x^k-x^{k-1})$, $D_k=(d_1^k,d_2^k)^T$
5      Compute $\nabla\phi_k(0,0)=D_k\nabla f(x^k)$
6      Compute $B_k$ solving system (3)
7      Compute $\hat\alpha_k,\hat\beta_k$ solving $B(\alpha,\beta)=-\nabla\phi_k(0,0)$
8      Set $\hat x^{k+1}=x^k+\hat\alpha_k d_1+\hat\beta_k d_2$
9      if $f(\hat x^{k+1})<f(x^k)$ then
10         Set $\alpha_k,\beta_k=\hat\alpha_k,\hat\beta_k$
11         Set $x^{k+1}=\hat x^{k+1}$
12     else
13         Set $\beta_k=0$
14         Compute $\alpha_k$ by an Armijo line-search (with initial stepsize defined according to
            Barzilai-Borwein)
15         Set $x^{k+1}=x^k+\alpha_k d_1$
16     Set $k=k+1$
17 return $x^k$
```

Additional elements of the algorithm to be described:

- After unsuccessful iterations where the plane search does not lead to an objective decrease, a number of consecutive steps of Barziali-Borwein should be performed to fully exploit the power of this version of gradient descent;
- If either α_{k-1} or β_{k-1} is equal to zero, the linear system for the computation of B is not full-rank; in this cases, use $\alpha_\delta=\delta>0$ or $\beta_\delta=\delta>0$ to define the two interpolation points;

- Moreover, if either $|\alpha_{k-1}| < \varepsilon$ or $|\beta_{k-1}| < \varepsilon$, use $\alpha_\delta = \delta \text{sign}(\alpha_{k-1})$ or $\beta_\delta = \delta \text{sign}(\beta_{k-1})$ to define the two interpolation points, otherwise the quadratic parameters of the approximation might result in being "too local";
- the linear system $B(\alpha, \beta) = -\nabla f(0, 0)$ might be not full-rank; in that case, solve it as by regularized least-squares;
- If the obtained values of α_k and β_k do not provide a decrease in the objective function, they can be used to recompute the quadratic approximation matrix B with higher accuracy; since we have computed $f(x^k + \alpha_k d_1 + \beta_k d2)$, we can add a row in the linear system, which can then be solved again by least squares; this strategy could be used repeatedly up to a predefined number of times (e.g., 3).

2 Experiments

2.1 A

Momentum constant stepsize vs. derivative-free plane search vs. plane search vs. plane search multistart vs. quadratic plane search

2.2 B

momentum-QPS vs. QPS with random second direction

2.3 C

momentum-qps vs. scipy-cg vs. scipy-lbfgs

References

[1] D. P. Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[2] P. L. Combettes. *Fejér monotonicity in convex optimizationFejér Monotonicity in Convex Optimization*, pages 1016–1024. Springer US, Boston, MA, 2009.

[3] E. Ghadimi, H. R. Feyzmahdavian, and M. Johansson. Global convergence of the heavy-ball method for convex optimization. In *2015 European control conference (ECC)*, pages 310–315. IEEE, 2015.

[4] L. Lessard, B. Recht, and A. Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

[5] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

[6] B. T. Polyak. Introduction to optimization. optimization software. *Inc., Publications Division, New York*, 1:32, 1987.

[7] T. Sun, P. Yin, D. Li, C. Huang, L. Guan, and H. Jiang. Non-ergodic convergence analysis of heavy-ball algorithms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5033–5040, 2019.

[8] S. Zavriev and F. Kostyuk. Heavy-ball method in nonconvex optimization problems. *Computational Mathematics and Modeling*, 4(4):336–341, 1993.