ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

IF: Iterative Fractional Optimization

Sarthak Chatterjee[1], Subhro Das[2], Sérgio Pequito[3]

1- Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute, Troy NY, USA

2- MIT-IBM Watson AI Lab, IBM Research, Cambridge MA, USA

3- Delft Center for Systems and Control
Delft University of Technology, The Netherlands

Abstract. Most optimization problems lack closed-form solutions of the argument that minimizes a given function, and even if these were available it might be prohibitive to compute it. As such, we rely on iterative numerical algorithms to find an approximate solution. In this paper, we propose to leverage fractional calculus in the context of time series analysis methods to devise a new iterative algorithm. Specifically, we propose to leverage autoregressive fractional-order integrative moving average time series, whose coefficients encode a proxy for local spatial information. We provide evidence that our algorithm is efficient and particularly suitable for cases where the Hessian is ill-conditioned.

1 Introduction

Many problems in today's world can be modeled as optimization problems where we seek to find the minimum (or maximum) of an objective function $f : \mathbb{R}^n \to \mathbb{R}$ [1]. For instance, in a learning problem, we aim to minimize a loss index that measures the performance of a neural network on a data set.

Notwithstanding, most optimization problems do not possess numerically viable closed-form solutions [2]. Furthermore, due to the ever increasing dimensionality of data used to test a variety of optimization problems, iterative algorithms need to be employed to attain an approximate solution. At the core of the iterative algorithms, we can commonly find three key ingredients [3]: (i) a descent direction $d \in \mathbb{R}^n$, (ii) a learning rate (or step) $\alpha \in \mathbb{R}$; and (iii) local spatial information across different variables (i.e., $w \in \mathbb{R}^n$). In a nutshell, the iterative algorithms can be written as

$$w_{k+1} = w_k + \alpha_k d_k, \qquad k = 0, 1, \ldots, \tag{1}$$

where the descent direction, $d_k \in \mathbb{R}^n$ and the step, $\alpha_k \in \mathbb{R}$, might change over time k, with gradient descent and Newton's method being particular instances of the same [4]. We can also consider memory in the process, which is an alternative to speeding up the convergence while not resorting to second-order methods, such as in the Heavy-Ball method [5] or Nesterov's Accelerated Gradient method [6], where we have the general form of the update step

$$w_{k+1} = w_k + \alpha_k d_k + m(w_{k-1}, \ldots, w_{k-T}), \qquad k = 0, 1, \ldots, \tag{2}$$

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

where $m : \mathbb{R}^{n \times T} \to \mathbb{R}^n$ is a function that accounts for the previous iterations, or memory in short. Since second-order methods capture (local) spatial properties of the function through its cross-derivatives, they are generally faster than first-order iterative methods. However, first-order iterative algorithms require less computational power and memory storage.

In order to motivate the use of time series processes in optimization, we start with the update step in (2), where the last term could be seen as the time series model. Furthermore, there is no learning step and the process is implicitly cast by the data obtained, sampled from the function.

In this paper, we seek to leverage *fractional-order calculus* [7], in order to propose a novel iterative algorithm termed as `IF` (Iterative Fractional). Specifically, we leverage *autoregressive fractional-order integrative moving average* (ARFIMA) processes that allow the modeling of long-term memory and interlaced couplings among spatial and temporal time-scales [8]. We show that the fractional-order differencing parameter can effectively be perceived as a substitute for local curvature. We demonstrate the efficacy of `IF` on a variety of problems and show that our method solves unconstrained optimization problems without the explicit need for a step size, gradient, or Hessian information. The major advantages of `IF` lie in cases where the Hessian is ill-conditioned, which is particularly important in the context of neural networks [9].

2 `IF` : The Iterative Fractional Algorithm

A class of stationary long-term processes z_t modeled as *autoregressive fractionally integrated moving average* (ARFIMA) processes are described by

$$\phi(B)(1-B)^d z_t = \theta(B) a_t, \tag{3}$$

for $d \in \mathbb{R}$, d being the *fractional differencing parameter*. Here, a_t is a white noise sequence having zero mean and bounded variance σ_a^2, the polynomial equations $\phi(B) = 1 - \sum_{i=1}^{p} \phi_i B^i = 0$ and $\theta(B) = 1 + \sum_{i=1}^{q} \theta_i B^i = 0$ have roots that are greater than unity in absolute value, and B is the *backward shift operator* with the property $B^m z_t = z_{t-m}$. The general form of the processes that can be represented using (3) are called ARFIMA(p, d, q) processes.

For $d > -1$, we can employ the binomial expansion formula to explicitly expand the operator $(1-B)^d$ as $(1-B)^d = \sum_{j=0}^{\infty} \pi_j B^j$, with $\pi_0 = 1$, and $\pi_j = \frac{\Gamma(j-d)}{\Gamma(j+1)\Gamma(-d)}$, $\quad j = 1, 2, \ldots$, with $\Gamma(\cdot)$ being the Gamma function defined by $\Gamma(x) = \int_0^{\infty} s^{x-1} e^{-s}\, \mathrm{d}s$ for all complex numbers x with $\Re(x) > 0$. We note that the weighting coefficients π_j can be defined recursively in j. Further, even though the binomial expansion of the operator $(1-B)^d$ consists of an infinite number of terms, in practice we will always consider an approximation that still preserves the dependency of parameters described.

Given a time series, we carry out the following steps to obtain the parameters in (1): *(i)* apply fractional differencing on the original time series and note the order of the fractional difference d that makes the time series (close to)

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

wide-sense stationary; *(ii)* determine the ARMA parameters $p, q, \{\phi_i\}_{i=1}^{p}$, and $\{\theta_i\}_{i=1}^{q}$ using the (fractionally) differentiated time series; *(iii)* perform a forecast for a requisite number of steps ahead in time with these ARMA terms; and *(iv)* fractionally integrate the forecasted ARMA data to obtain the forecast of the ARFIMA process. Note that fractional integration may be interpreted as fractional differentiation but with a fractional differencing parameter of $-d$.

Let us use the `IF` algorithm in order to solve the following unidimensional unconstrained optimization problem $x^\star = \text{argmin}_{x\in\mathbb{R}} f(x)$, with a given initial point x_0. For the purpose of illustration, we consider the function $f(x) = x^2$, the grid discretization step $h \in \mathbb{R}$, the number steps of memory $P \in \mathbb{R}$, and its corresponding functional values $f(x_0), f(x_0 - h), \ldots, f(x_0 - (P-1)h)$, with pre-specified values of x_0, P, and h. First, we notice that the sample autocorrelation function (sACF) obtained from the aforementioned values and depicted in Figure 1 suggests slower than exponential algebraic decay and statistically significant (for a significance level of 0.05) dependency on past lags, with a large area enclosed by the composite sACF curve and the horizontal axis. This suggests that the ARFIMA(p, d, q) processes described above can successfully predict the behavior of the functional values obtained.

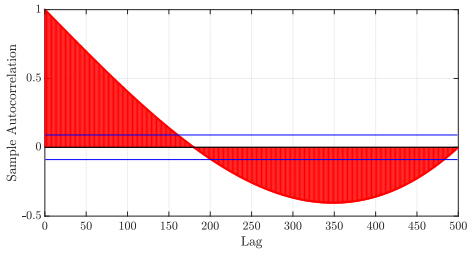

Fig. 1: sACF plot of the functional values $f(x_0), f(x_0 - h), \ldots, f(x_0 - (P-1)h)$, with $f(x) = x^2$, $x_0 = -1$, $P = 500$, and $h = 0.01$.

Next, we consider the pre-specified values of p and q along with the Whittle estimation procedure [10] to find the fractional differencing parameter d and the autoregressive and moving average coefficients $\{\phi_i\}_{i=1}^{p}$ and $\{\theta_i\}_{i=1}^{q}$ respectively. Using these estimated parameters, we perform an ARFIMA time series prediction P' steps into the future, to obtain the time series $y_1, y_2, \ldots, y_{P'}$. As depicted in the Figure 2, we find that our ARFIMA time series predictions are limited in their predictive capabilities since they can only capture information about the local behavior of the function upto a certain finite number of time steps into the future. Since many descent methods in the optimization literature require us to satisfy $f(x_{k+1}) \leq f(x_k)$ for all k, we select the largest possible value of $P'' \leq P'$, such that P'' satisfies $y_1 \geq y_2 \geq \ldots \geq y_{P''} \leq y_{P''+1}$. If, at this stage, $f(x_k + P''h) > f(x_k)$, we update the discretization step h by $h/2$ until the condition $f(x_k + P''h) \leq f(x_k)$ is satisfied. Once that is obtained, we

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

update the current iterate as

$$x_{k+1} = \begin{cases} x_k + P''h, & \text{if } f(x_k + P''h) \leq f(x_k) \\ x_k - P''h, & \text{if } f(x_k - P''h) \leq f(x_k) \end{cases}. \tag{4}$$

The algorithm terminates when $|f(x_{k+1}) - f(x_k)| \leq \varepsilon$, where ε is a specified tolerance. Although we have detailed the working of the `IF` algorithm for a single dimensional problem, we can apply the `IF` algorithm to iteratively run over each individual dimension in order to solve multiple dimensional problems as well.

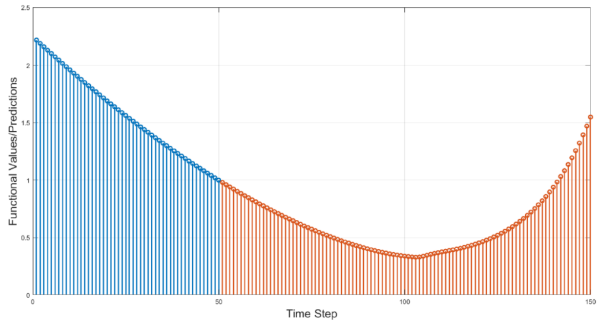

Fig. 2: ARFIMA time series predictions.

3 Illustrative Examples

In what follows, we will demonstrate the working of the `IF` algorithm on two illustrative examples and show that our approach is particularly suited to problems where the Hessian is ill-conditioned. We first consider the unconstrained optimization problem $x^\star = \text{argmin}_{x \in \mathbb{R}^2} \ (x_1^2 + 0.001x_2^2)$. This objective function has a Hessian matrix $H = \begin{bmatrix} 2 & 0 \\ 0 & 0.002 \end{bmatrix}$, with the condition number of H, $\mathsf{cond}(H) = 2/0.002 = 1000$. The starting point is chosen to be $x_0 = [\sqrt{3}/2 \quad 1/2]^\mathsf{T}$. We use $P = 100$ steps of memory, an initial grid discretization step of $h = 0.01$, and $\varepsilon = 10^{-3}$, with $P' = 100$ steps ahead ARFIMA$(4, d, 0)$ time series predictions. The convergence profile of the evolution of functional values versus the iteration number when the `IF` algorithm is used to solve this problem is shown in Figure 3. The `IF` algorithm is able to attain convergence in 43 iterations while the gradient descent algorithm with inexact backtracking line search (to tune the step size) takes 3453 iterations to converge. This suggests the significant advantages of using the `IF` algorithm in problems where the Hessian is ill-conditioned.

Next, we evaluate the performance of the `IF` algorithm on a simple feedforward neural network. The motivation behind this is the fact that feedforward

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

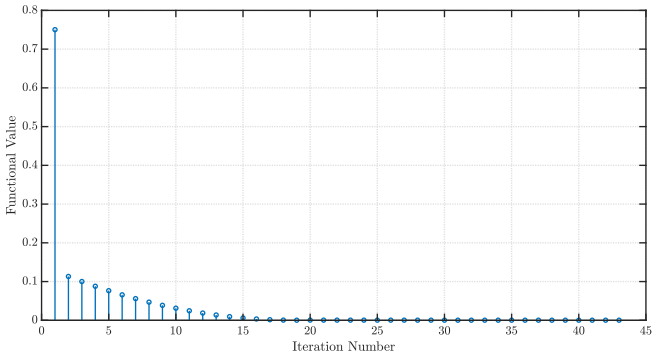

Fig. 3: Convergence profile showing the evolution of functional values with iteration number when the `IF` algorithm is used to find the minimizer of $f(x) = x_1^2 + 0.001x_2^2$.

neural networks often possess ill-conditioned Hessians, as demonstrated in [9]. Consider the simple single-layer perceptron network structure shown in Figure 4. Here, the inputs x_1 and x_2 are weighted using the weights w_1 and w_2 respectively, and the output y is a result of applying an *activation function* $\varrho(\cdot)$ on the weighted sum $\sum_{i=1}^{2} w_i x_i$.

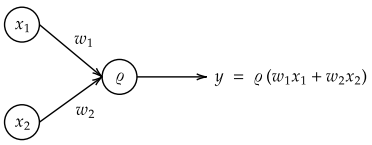

Fig. 4: Structure of the simple feedforward neural network used in our example.

Assume that the activation function $\varrho(\cdot)$ is the Gaussian Error Linear Unit (GELU) [11], given by $\varrho(s) = \frac{s}{2}\left(1 + \operatorname{erf}\left(\frac{s}{\sqrt{2}}\right)\right)$, where the *error function* $\operatorname{erf} z = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, \mathrm{d}t$ for any $z \in \mathbb{C}$. For the training process, the weights w_1 and w_2 are arbitrarily initialized such that $w_1^2 + w_2^2 = 1$. Now, consider the arrival of a single training sample $(x_1, x_2, t) = (1, 1, 0)$, where t is the true output that the neural network is to produce when it has been properly trained. In this case, $y = \varrho(w_1x_1 + w_2x_2) = \frac{w_1+w_2}{2}\left(1 + \operatorname{erf}\left(\frac{w_1+w_2}{\sqrt{2}}\right)\right)$, since $x_1 = x_2 = 1$. If we consider a regression problem using the squared error $L(t, y) = (t - y)^2$ as a *loss function*, then $L(t, y) = (0 - y)^2 = \left(\frac{w_1+w_2}{2}\left(1 + \operatorname{erf}\left(\frac{w_1+w_2}{\sqrt{2}}\right)\right)\right)^2$. To minimize the loss function $L(t, y)$ we use the `IF` algorithm. Using $P = 100$ steps of memory, an initial grid discretization step of $h = 0.01$, $\varepsilon = 10^{-3}$, with $P' = 100$ steps ahead ARFIMA$(4, d, 0)$ time series predictions, we obtain convergence in 18 iterations with the optimal values $w^\star = [-1.0144 \quad 0.9997]^\mathsf{T}$ and

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

$L^\star = 5.2488 \times 10^{-5}$, thus showing the efficacy of the `IF` algorithm on single-pass training of a feedforward neural network. In comparison, gradient descent with inexact backtracking line search takes 60 iterations to converge for the same initialization of the weights w_1 and w_2.

4 Discussion and Conclusions

In this paper, we proposed the `IF` algorithm, that uses fractional calculus-based ARFIMA time series models to determine an approximation of the argument that minimizes a given unconstrained optimization problem. Our proposed method does not require gradient or Hessian information, or explicitly tuning a step size, an issue that, in spite of receiving widespread coverage in the optimization literature, still proves to be a bottleneck in the design of such algorithms. Future work will entail the automated selection of the autoregressive and moving average orders from the functional values at every time step and the automation of determining the fractional-order coefficient, which proved to be the most computationally intensive step in our approach. Furthermore, we will validate our approach in a wider range of optimization benchmarks.

5 Acknowledgment

The authors gratefully acknowledge the support by the National Science Foundation under Grant Number CMMI 1936578.

References

[1] Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright. *Optimization for Machine Learning*. MIT Press, 2012.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Science & Business Media, 2006.

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

[5] Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[6] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Springer Science & Business Media, 2013.

[7] Keith B. Oldham and Jerome Spanier. *The Fractional Calculus: Theory and Applications of Differentiation and Integration to Arbitrary Order*. Elsevier, 1974.

[8] Andreas Klaus, Shan Yu, and Dietmar Plenz. Statistical analyses support power law distributions found in neuronal avalanches. *PloS ONE*, 6(5):e19779, 2011.

[9] Sirpa Saarinen, Randall Bramley, and George Cybenko. Ill-conditioning in neural network training problems. *SIAM Journal on Scientific Computing*, 14(3):693–714, 1993.

[10] Peter Whittle. Gaussian estimation in stationary time series. *Bulletin of the International Statistical Institute*, 39:105–129, 1961.

[11] Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.