Convex Analysis : Three-operators splitting

Attali Raphaël

May 2019

1 Table of contents

2 Description of the problem

Real life problems may be complex. With the appearance of Machine Learning, and in fact every large-scale application, needs to tackle this curse of dimensionality, if we may say. One way to solve this kind of complex problem is to efficiently split it into a series of simpler sub-problems that are easier to solve regarding computation, therefore leveraging parallel computation, or at least achieving complexity reduction. This is exactly the purpose of the operator splitting scheme. A k-operator splitting is supposed to split the problem as a sum of k operators. Every operator-splitting scheme is based on a 2-operator splitting one, one of the following : the forward- backward-forward splitting, the forward-backward splitting (FBS) or the Douglas-Rachford splitting (DRS).

That is surprising that we only use 2-operator splitting. Indeed, splitting the problem in $k > 2$ subproblems seems to be much more effective. One great idea would be to find a way to achieve a 3-operator splitting to enhance the performances. It turns out that, under some conditions on these so-called operators, we can achieve to compute 3-operator splitting.

We all know the strategy "divide and conquer", then how do we split our problem ?

The state of the art regarding operators splitting has been focusing on 2-operator splitting, as we earlier discussed. The original problem is to find $x \in$

H *such that* $0 \in Ax + Bx$, H is a Hilbert space, A maximal monotone, and B is cocoercive. We will first, quickly remind the principle of the Alternating Direction Method of Multipliers (ADMM).

The ADMM is an algorithm that solves convex optimization problems by breaking them into smaller pieces, each of which are then easier to handle. In our case, it namely allows you to solve the following problem : finding $min_{x,z} f(x) + g(z)$ s.t. $Ax + Bz = b$.

Actually, ADMM is closely linked to the 2-operators splitting. In fact, the ADMM is simply a Douglas-Rachford splitting method - one of the three so-called 2-operators splitting - applied to the dual problem (which is maximizing $-b^T z - f_1^*(A_1^T z) - f_2^*(A_2^T)z$, meaning that its opposite can be minimized by Douglas-Rachford). In the following we will write the dual under the form $d = d_1 + d_2$ to simplify the equations.

As the Douglas-Rachford iteration is :

$$w^{k+1} = (1-\alpha)w^k + \alpha Refl_{\gamma d_2} Refl_{\gamma d_1} w^k \tag{1}$$

$$y_{k+1} = y_k + Prox_{\gamma d_1} w^{k+1} \tag{2}$$

With the reflection operator defined as

$$Refl_l(v) = 2\frac{v \cdot l}{l \cdot l} l - v = 2Proj_l(v) - v,$$

Then after a few calculations, we can deduce when fixing $\alpha = \frac{1}{2}$ the iteration of ADMM :

$$z^{k+1} = argmin_z g(z) + < y^k + Bz > + \frac{\gamma}{2}||Ax^k + Bz - b||^2 \tag{3}$$

$$z^{k+!} = argmin_x f(x) + < y^k + Ax > + \frac{\gamma}{2}||Ax + Bz^{k+1} - b||^2 \tag{4}$$

$$y_{k+1} = y_k + \gamma(Ax^{k+1} + Bz^{k+1} - b) \tag{5}$$

3 Three-operator splitting

Three-operator splitting can be summarized as the following :

$$find\ x \in H\ such\ that\ 0 \in Ax + Bx + Cx \tag{6}$$

for three maximal monotone operators A,B,C defined on a Hilbert space H. We only have one constraint on the operator C, it has to be co-coercive. This formula is very general, we can imagine many operators that would fit. The most easy one is A,B,C as functions, and C is Lipschitz differentiable (hence co-coercive) as we will show in a further detailed example.

Ideally, we would like to find an algorithm that may solve the above problem. Thus we propose in the following to describe the problem using well-known operators, namely the resolvent operator and the identity, in order to land on our feet and find a known situation. If we note J_S the resolvent of the operator S, and I the identity in H, we define the new operator T as followed :

$$T = I - J_{\gamma B} + J_{\gamma A} \circ (2J_{\gamma B} - I - \gamma C \circ J_{\gamma B}) \tag{7}$$

One can prove that the solution of problem (1) are the fixed points of that new operator T. As a remark, we can see that the expression is quite close to the structure of the iteration of the ADMM.

We do have found an operator that uses already known operators, but it seems to be something more complex that the first formula. Don't be mistaken, the operator T is precious. Indeed, we can prove that **T is a averaged operator**.

We remind that, given an averaged operator, we have a simple algorithm that converges to one of its fixed points, if there exists at least one. According to Krasnosel'skii theorem, if T is an averaged operator with a fixed point, the iteration

$$x_{k+1} \leftarrow Tx_k \tag{8}$$

makes x_k converges weakly to a fixed point of T.

We would rather use another algorithm that tends to converge more efficiently to the fixed point, whose iteration is described as followed :

$$x_{k+1} \leftarrow (1 - \lambda_k)x_k + \lambda_k T x_k \tag{9}$$

with a condition on the λ_k :

$$\sum_{k=1}^{\infty} \lambda_k (1 - \lambda_k) = \infty$$

(That condition is simple thus we can easily choose λ_k that validate it)

This leads us to a simple algorithm :

Algorithm 1 - finding a fixed point of T :

$x_B^k \leftarrow J_{\gamma B}(z^k)$
$x_A^k \leftarrow J_{\gamma A}(2x_B^k - z^k - \gamma C x_B^k)$
$z_{k+1} \leftarrow \lambda_k (x_A^k - x_B^k)$

This algorithm is enough to compute the three-operator splitting, because $Tx = x$ is equivalent to $0 \in Ax + Bx + Cx$. The only constraint is to be able to compute $J_{\gamma A}$ and $J_{\gamma B}$, that is often not so difficult.

We can easily see that the 2-operator splitting problem is a special case of that first algorithm, and moreover it will be the basis of every algorithm that solve our problem.

We solved the three-operator splitting problem, but what about the performances of the algorithm ? To compare with a known algorithm, it has be shown that Douglas-Rachford is likely to converge with linear rate, except in some cases [1] [2]. Our algorithm has to be better than that, however it is not efficient under this simple form. Indeed, most of the time we obtain a convergence rate of $O(\frac{1}{\sqrt{k+1}})$. That is the reason we have to improve it.

4 Modifications / enhancements

4.1 Averaging

Averaging is one of the most used method in operator-splitting. It consists on averaging the variables x_B^k and x_A^k in the algorithm 1. Theorically, that is a great technic, as we reach a convergence $O(\frac{1}{k+1})$ instead of $O(\frac{1}{\sqrt{k+1}})$. However, when averaging, we lose some structures that have been formed, and it may lead to even worst practical results.

In order not to lose these structures, we propose to give more weight to the latest iterates, instead of having a constant λ_k as a weight. Then, each iteration is more important than the older, and thanks to that technic, we can equal the performance of Douglas-Rachford algorithm.

4.2 Exceeding the performances of D-R

Whenever B or C is strongly monotone, a smart change and a varying sequence of stepsizes in the algorithm makes the rate way better : $O(\frac{1}{(k+1)^2})$. The proof is long and it would be pointless to show it here, but the idea is to use in a relevant way the properties on the operators, because the strongly monotone property is strong.
Finally we found a way to exceed the convergence of the Douglas-Rachford algorithm on the basis of the Algorithm 1, but only on a very special configuration.

4.3 Other improvements

The algorithm 1 has much more improvements, namely using the Line Search whose acceleration boost is not proved yet, but found experimentally. There are plenty of them, it will not be very meaningful to describe them all as they are extremely well detailed in the article.

5 ADMM

As we have seen before, ADMM is basically the state-of-art regarding operator-splitting. The traditionnal ADMM turns out to be a hidden 2-operators splitting because it is obtained by applying Douglas-Rachford to the dual problem (please see the quick explanation above). Considering the improvements that we treated, we have naturally the idea to apply and adapt our three-operators splitting algorithm to create a new ADMM method. That new ADMM, well-named 3-blocks ADMM, is precisely one of the main applications of the 3-operators splitting scheme.

The 3-blocks ADMM aspire to solve the following problem :

$$minimize_{x_1,x_2,x_3}\ f_1(x_1) + f_2(x_2) + f_3(x_3)$$

$$subject\ to\ L_1x_1 + L_2x_2 + L_3x_3 = b$$

with all f_i that are convex proper functions, L_i are linear mappings, and f_1 is μ-strongly convex. Like we found the traditional ADMM by applying D-R algorithm to the dual, we apply our method to the dual of the expression above, that is :

$$D = f_1^*(L_1^*w) + f_2^*(L_2^*w) + f_3^*(L_3^*w) - < w, b > . = d_1(w) + d_2(w) + d_3(w) \quad (10)$$

After a few calculations, we can deduce an algorithm that solves the original problem and that is called 3-blocks ADMM. we will define the operator s : $s_\gamma(x_1, x_2, x_3, w) = L_1x_1 + L_2x_2 + L_3x_3 - b - \frac{1}{\gamma}w$, that will allow us to write properly the algorithm :

$$get\ x_1^{k+1} = argmin_{x_1} f_1(x_1) + < w^k, L_1, x_1 > \quad (11)$$

(*one must pay attention about the* 2^{nd} *and* 3^{rd} *step, we use* $\in$ *instead of* $=$ *as argmin is not necessarily unique*)

$$get\ x_2^{k+1} \in argmin_{x_2} f_2(x_2) + \frac{\gamma}{2}||s(x_1^(k+1), x_2, x_3^k)||^2 \quad (12)$$

$$get\ x_3^{k+1} \in argmin_{x_3} f_3(x_3) + \frac{\gamma}{2}||s(x_1^(k+1), x_2^{k+1}, x_3)||^2 \quad (13)$$

$$w^{k+1} = w^k - \gamma(L_1x_1^(k+1) + L_2x_2^{k+1} + L_3x_3^{k+1} - b) \quad (14)$$

Note that, unless the traditionnal ADMM, the first step of our algorithm does not have a quadratic penalty. That is a huge advantage. Indeed, we created a 3-blocks ADMM, but if f_1 turns out to be separable, we could split again that first step. Then, we can extend our algorithm to the case where we have (m-2) strongly convex operator, and compute the first step in parallel for each of them : we just created the m-blocks ADMM.

Using traditionnal ADMM directly on three blocks does not converge [3], then the specific 3-blocks ADMM is already better than the classic one. Another great thing, our 3-blocks ADMM works under the simplest and works under the weakest assumption compared to the recent work. Indeed, the algorithm we depicted works if only f_1 is strictly convex, with no other assumptions about all the other operators. Moreover, the 3-blocks algorithm experience a linear convergence rate, where the other current ADMM extensions need additionnal assumption without necessarily converging faster.

6 Conclusion

It is easy to show on first sight that 3-blocks is an expansion of the 2-blocks scheme. However, as shown, what we qualify as 'standard' ADMM, does not extend directly onto a 3-blocks scheme. More than finding high similarity between ADMM and our 3-operator splitting, we indeed showed that the standard ADMM is not only a special case, but also that our scheme is the simplest and working with the weakest assumptions and better convergence results, allowing us to qualify it as a natural, yet efficient 3-block ADMM extended scheme.

References

[1] Heinz H.Bauschkea, J.Y.Bello CruzbTran, T.A.NghiaaHung, M.Phana Xianfu Wanga
The rate of linear convergence of the Douglas–Rachford algorithm for subspaces is the cosine of the Friedrichs angle. https://www.sciencedirect.com/science/article/pii/S0021904514001166!

[2] Mingyi Hong and Zhi-Quan Luo
On the Linear Convergence of the Alternating Direction Method of Multipliers. https://arxiv.org/pdf/1208.3922.pdf

[3] Chen, C., He, B., Ye, Y., Yuan, X.: *The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. 2014* http://dx.doi.org/10.1007/s10107- 014- 0826- 5