Chapter 5

A New Operator Splitting Algorithm and its Accelerated Variant with Convergence Guarantees and Application to Microarray Gene Analysis

5.1 Introduction

In previous chapters, we proposed various accelerated gradient methods to solve the non-smooth composite convex minimization framework. The common idea in the proposed algorithms was to design inertial-based forward-backward splitting algorithms in the real infinite dimensional space, the corresponding proximal gradient algorithms in the real finite dimensional space and apply them to solve various learning problems. As discussed in Chapter 2, forward-backward splitting techniques belong to the class of general operator splitting techniques, which include various other formulations also, such as Peaceman-Rachford splitting techniques, Douglas-Rachford splitting techniques, etc. In this chapter, we introduce a new direction in this field and propose an Extragradient-based Operator Splitting Algorithm EOSA. The accelerated variant of the algorithm named as Accelerated Extragradient-based Operator Splitting Algorithm AEOSA is also proposed. With the detailed description, the convergence guarantees for both the algorithms are given. The practical performance of the proposed algorithms is tested for microarray gene analysis, specifically for the task of cancer prediction. Experiments are performed on four publicly available benchmark microarray gene expression datasets. The performance of the proposed algorithms is compared with different the state-of-the-art operator splitting methods.

The general framework of the problem we consider, is the following,

$$\min_{x \in \mathbb{R}^d} f(x) + g(x). \tag{5.1}$$

Here both f and g functions are non-smooth, closed and convex functions. Operator splitting techniques solve the above problem and have been very popular due to their applications in various fields ranging from compressed sensing [46], statistical estimation [95] to medical imaging [126]. There are three standard operator splitting techniques, namely, Forward-backward Splitting, Peaceman-Rachford Splitting and the Douglas-Rachford splitting. Out of these, the later two consider the above framework, whereas the forward-backward splitting methods consider the case when function f is differentiable. The later two methods relax this condition of smoothness.

It is a known fact that various operator splitting techniques are designed using basic fixed-point iterative schemes. The field of fixed-point theory has been a popular area of mathematics, which is used for solving different problems in variational inequality, inclusion and convex optimization. As we will discuss in the next section, the standard operator splitting techniques, Peaceman-Rachford technique (PR) and Douglas-Rachford technique (DR) are based on very basic Picard and Mann fixed-point iterative schemes, respectively. We go beyond the usage of these standard fixed-point methods and designed a new operator splitting method corresponding to a new definition of the fixed-point scheme. We also analyze its performance for the real world problem of microarray gene analysis. In addition, the accelerated variant of the proposed algorithm is also proposed along with its convergence guarantee and application to the same problem of cancer prediction.

It should be noted that the Douglas-Rachford algorithm is the root of the popular method of alternating direction method of multipliers (ADMM) [42, 38], which solves the Fenchel dual problem of (5.1), whereas the Douglas-Rachford solves the primal problem. A variety of research work exists in the literature where the convergence guarantee and the convergence rates of the above algorithms have been analyzed such as [87, 140, 19, 97, 73, 155]. We expect that the proposed methods are also possibly extendible to dual spaces and provide interesting and efficient optimization algorithms. Various formulations and modifications of the Peaceman-Rachford algorithm are also available in the literature of operator splitting methods [69, 70].

5.1.1 Contributions

The main contributions of this chapter are as follows:

- We propose an extragradient-based operator splitting algorithm (EOSA) for the problem of minimizing the sum of two non-smooth closed convex functions. We also prove the linear convergence of the algorithm under few specific assumptions.
- We have also proposed the accelerated variant of the algorithm, namely, an accelerated extragradient-based operator splitting algorithm (AEOSA) and analyze the convergence of the proposed algorithm.
- We performed extensive experiments with four high-dimensional microarray gene expression datasets and compared our algorithms against the different standard and latest related algorithms.

5.1.2 Outline

The rest of the chapter is organized as follows. In Section 5.2, we will discuss the preliminary concepts and notations that we used in this chapter. In section 5.3, we will propose the novel **E**xtragradient-based **O**perator **S**plitting **A**lgorithm (EOSA) and discuss its convergence analysis in detail. An accelerated variant of EOSA, called as **A**ccelerated **E**xtragradient-based **O**perator **S**plitting **A**lgorithm (AEOSA) is also proposed in this section, along with its convergence analysis. The Experimental setup and result analysis with four publicly available benchmark real datasets are given in Section 5.4. We conclude by summarizing the work and the key contributions of this chapter in Section 5.5.

5.2 Related Concepts and Background

In this section, we will discuss the standard operator splitting methods and their interpretation as fixed-point iterative schemes. We also discuss the research motivation of this chapter.

5.2.1 Peaceman-Rachford Operator Splitting Algorithm

The main problem under consideration (5.1) can be interpreted as finding zero of two maximal monotone operators, as follows

$$0 \in (\partial f + \partial g), \tag{5.2}$$

where ∂f and ∂g are subgradients of functions f and g respectively, and considered as maximal monotone set-valued operators in $\mathbb{R}^d$. Let T be an operator, the reflected resolvent operator (or Cayley operator) with respect to operator T is defined for $\lambda > 0$ as follows [18],

$$R_\lambda^T = 2J_\lambda^T - Id. \tag{5.3}$$

Let us define the resolvent and reflected resolvent operators of two maximal monotone operators A and B, with respect to parameter λ as J_λ^A, J_λ^B (see appendix for the definition of resolvent operators) and R_λ^A, R_λ^B, respectively. Since the A and B are maximal monotone, it is clear that all the four operators i.e. J_λ^A, J_λ^B, R_λ^A, and R_λ^B are non-expansive operators [62]. It is also known that the combination of Cayley operators $R_\lambda^B R_\lambda^A$ is also non-expansive.

For solving the problem (5.1), the corresponding definitions of reflected resolvent operators R_λ^f and R_λ^g will be given as, $(2\,\mathrm{prox}_{\lambda f} - Id)$ and $(2\,\mathrm{prox}_{\lambda g} - Id)$, respectively. From now on, we will use $R_\lambda^f\ R_\lambda^g$ for $R_\lambda^B\ R_\lambda^A$. To solve problem (5.1), the Peaceman-Rachford splitting is an undamped iteration defined as follows, with $z_1 \in \mathbb{R}^d$,

$$\begin{cases} x_n & \leftarrow prox_{\lambda g}(z_n) \\ z_{n+1} & \leftarrow R_\lambda^f R_\lambda^g\ (z_n), \end{cases} \tag{5.4}$$

where $\lambda > 0$, and $prox_{\lambda g}$ is the proximity operator defined as follows,

$$prox_{\lambda g}(z) = \underset{v}{\operatorname{argmin}} \{\lambda g(v) + \frac{1}{2}\|z-v\|^2\}.$$

Few authors showed the convergence of this iterative technique with few assumptions. For example in [61], the author showed the strong convergence of a perturbed extension of this algorithm under the Slater condition, where as in [69], the author proposed a relaxed Peaceman-Rachford algorithm.

5.2.2 Douglas Rachford Operator Splitting Algorithm

The Douglas-Rachford splitting algorithm is a damped iteration with respect to $\tau \in (0,1)$ defined as follows, with $z_1 \in \mathbb{R}^d$,

$$\begin{cases} x_n & \leftarrow prox_{\lambda g}(z_n) \\ z_{n+1} & \leftarrow ((1-\tau)z_n + \tau\, R_\lambda^f R_\lambda^g)\ (z_n). \end{cases} \tag{5.5}$$

With the general choice of $\tau = \frac{1}{2}$, another way to write the above scheme is as follows,

$$\begin{cases} x_n \leftarrow \mathrm{prox}_{\lambda g}\, z_n \\ y_n \leftarrow \mathrm{prox}_{\lambda f}\, (2x_n - z_n) \\ z_{n+1} \leftarrow z_n + y_n - x_n \end{cases} \tag{5.6}$$

It has been shown in [62] that this damped iteration always converges to a point $x \in \mathscr{H}$, and $\mathrm{prox}_{\lambda f}(x) \in zer(\partial f + \partial g)$ (here $zer(T)$ denotes the set of zeros of operator T). The convergence proof given in [62] is for the inexact version of the Douglas-Rachford splitting method.

The iterative process given in (5.4) can be interpreted as the Picard's iterative scheme with respect to the nonexpansive operator $R_\lambda^f\ R_\lambda^g$. Similarly, the averaged mapping of the nonexpansive operator $R_\lambda^f\ R_\lambda^g$ given in (5.5) can be interpreted as the fixed-point iterative scheme from Mann [128]. Both of these schemes are very basic and proposed a long time back. In the literature of fixed-point theory, there exists a variety of advanced fixed-point schemes that work with the nonexpansive operators and converge to a solution point, assuming the solution point exists. In this chapter, we go beyond the standard operator splitting algorithms and propose a new splitting algorithm which is based upon an advanced fixed-point iteration scheme. We prove the linear convergence of the algorithm under few specific assumptions and also derive the accelerated variant of the algorithm. The algorithm is applied to the bioinformatics problem of cancer prediction with real benchmark microarray gene expression datasets, where we solve the popular lasso framework used for the sparsity regularized risk minimization problem.

5.3 Proposed Operator Splitting Algorithms

As described earlier, the two standard operator splitting methods Peaceman-Rachford splitting and the Douglas-Rachford splitting is based on the Picard's and Mann's iteration respectively. Although these iterative schemes are simple and effective in solving the problem (5.1), we are interested in analyzing the effect of latest advanced fixed point iterations based operator splitting techniques.

5.3.1 EOSA

In this work, we will analyze a very recent fixed point iteration based operator splitting technique, which we also used in Chapter 3. The fixed point technique we utilize in this work was introduced in [173], which considers the concept of extra-gradient. This method is also considered to be a hybrid Picard-Mann fixed point iteration scheme. We designed a generalized operator-splitting algorithm and applied the method for the task of classification. Following the same notations used in previous sections, we propose the **E**xtragradient-based **O**perator **S**plitting **A**lgorithm (EOSA) to solve problem (5.1), as follows, with $z_1 \in \mathbb{R}^d$, sequence $\beta_n \in (0,1)$ and $\lambda > 0$,

$$\begin{cases} x_n \leftarrow \operatorname{prox}_{\lambda g} z_n, \\ z_{n+1} \leftarrow R_\lambda^f \, R_\lambda^g \, ((1-\beta_n)Id + \beta_n \, R_\lambda^f R_\lambda^g) \, z_n, \end{cases} \tag{5.7}$$

Algorithm 6: Extragradient-based Operator Splitting Algorithm

```
Data: Training/Testing Data, $tol$
Result: $x_n$
function EOSA
    initialize $z_1 \in \mathbb{R}^d$, $\lambda \in (0,+\infty)$, $\beta_1 \in (0,1)$, $n = 1$;
    repeat
        Compute $\beta_n$ such that $\beta_n \in (0,1)$;
        $x_n \leftarrow prox_{\lambda g}(z_n)$;
        $v_n \leftarrow R_\lambda^f R_\lambda^g\ z_n$;
        $y_n \leftarrow (1-\beta_n)\ z_n + \beta_n\ v_n$;
        $z_{n+1} \leftarrow R_\lambda^f R_\lambda^g\ (y_n)$;
        $n \leftarrow n+1$;
    until $converge$;
```

with $(R_\lambda^f = 2\ \text{prox}_{\lambda f} - Id)$ and $(R_\lambda^g = 2\ \text{prox}_{\lambda g} - Id)$, respectively. For the general choice of $\beta_n = 1/2$, another way to write this iteration is as follows, with $z_1 \in \mathbb{R}^d$ and $\lambda > 0$,

$$\begin{cases} x_n \leftarrow \text{prox}_{\lambda g}\ (z_n) \\ y_n \leftarrow \text{prox}_{\lambda f}\ (2x_n - z_n) \\ u_n \leftarrow z_n + y_n - x_n \\ w_n \leftarrow 2\ \text{prox}_{\lambda g}(u_n) - u_n \\ z_{n+1} \leftarrow 2\ \text{prox}_{\lambda f}(w_n) - w_n \end{cases}$$

For the sake of simplicity, we will use the notation T for the nonexpansive operator $R_\lambda^f\ R_\lambda^g$. As described in previous chapters also, the scheme is defined for the operator T and $\beta_n \in (0,1)$ as follows, with $z_1 \in \mathbb{R}^d$,

$$z_{n+1} \leftarrow T((1-\beta_n)z_n + \beta_n T(z_n)), \quad x \in \mathscr{H}. \tag{5.8}$$

The pseudo code of the algorithm is given in 6. The condition *converge* is considered to be achieved when the difference between the function value at previous step and the function value at the current step becomes lesser than a previously defined tolerance value tol. In our experiments, we set the value of tol as 10e-4. In the next subsection, we will analyse the convergence of EOSA.

5.3.2 Convergence Analysis of EOSA

In order to prove the linear convergence of the algorithm, we first introduce the notion of the NE-operator (New Extragradient operator) and analyze the property of the operator. Let C be a nonempty convex subset of a Hilbert space $\mathscr{H}$ and $T : C \to C$ is an operator. For $\beta_n \in (0,1)$ the NE-operator is defined as follows,

$$T_{NE} = T[(1-\beta_n)Id + \beta_n T]. \tag{5.9}$$

Following [87], the proposed iterative scheme can be re-written in terms of T and $\beta \in (0,1)$, as follows,

$$z_{n+1} \leftarrow T[(1-\beta_n)Id + \beta_n T]z_n \tag{5.10}$$

Although it is discussed in [173], we give a formal lemma for the property of operator T_{NE}.

Lemma 5.1. *Let C be a nonempty convex subset of a Hilbert space $\mathscr{H}$ and $T : C \to C$ is an operator. Let T_{NE} be an operator defined as in (5.9). Then T_{NE} will be a non-expansive operator, if T is non-expansive and it will be $k(1-\beta_n(1-k))$-contractive if T is k-contractive.*

Let us denote the fixed-point of an operator by $fix(\cdot)$. The fixed-point of operator T_{NE} is formalized for a Hilbert space as follows,

Proposition 5.2. *[173] Let C be a nonempty closed convex subset of $\mathscr{H}$ and $T : C \to C$ a contraction operator. Assume that $\beta_n \in (0,1)$. Then $fix(T_{NE}) = fix(T)$.*

The convergence of the proposed algorithm for the nonexpansive definition of operator T can be directly implied from theorem 6.7.4 from [174].

Theorem 5.3. *Let $\mathscr{H}$ be a real Hilbert space. Let C be a nonempty closed convex subset of and $T : C \to C$ a nonexpansive mapping with $fix(T) \neq \phi$. Let $\{z_n\}$ be the sequence defined by (5.10). Then z_n converges weakly to a fixed point of T.*

Since $x_n \leftarrow J_\lambda^g z_n$, and J_λ^g is again a nonexpansive operator, following statements from [87] and [18], we can conclude that the proposed algorithm converges to a fixed point solution x^*. It is well-known fact that for $T = R_\lambda^f R_\lambda^g$, the Douglas-Rachford algorithm follows the Krasnosel'skii-Mann(KM) algorithm, for which the linear convergence has been already proved. Next, we prove the linear convergence of iterative sequence (5.10).

Theorem 5.4. *Let $z_0 \in \mathscr{H}$ and $\beta_n \in (0,1)$. Let C be a closed convex subset of a Hilbert space $\mathscr{H}$ and $T : C \to C$ be a nonexpansive operator. Then the algorithm given in (5.10) satisfies the following:*

$$\|Tz_n - z_n\|^2 \leq \frac{\|z_0 - z^*\|_2^2}{\sum_{i=0}^n \beta_n(1-\beta_n)}. \tag{5.11}$$

Proof: To prove the linear convergence of the algorithm (5.10), we first show that $\|z_n - Tz_n\|_2$ is non-increasing.

$$\begin{aligned}
\|z_{n+1} - Tz_{n+1}\|_2 &= \|T[(1-\beta_n)z_n + \beta_n Tz_n] - Tz_{n+1}\|_2 \\
&\leq \|(1-\beta_n)z_n + \beta_n Tz_n - z_{n+1}\|_2 \qquad \text{(nonexpansivity of } T) \\
&= \|(1-\beta_n)(z_n - Tz_n) + Tz_n - z_{n+1}\|_2 \qquad \text{(by adding and subtracting } Tz_n) \\
&\leq (1-\beta_n)\|z_n - Tz_n\|_2 + \|Tz_n - z_{n+1}\|_2 \qquad \text{(from triangular inequality)} \\
&= (1-\beta_n)\|z_n - Tz_n\|_2 + \|Tz_n - T[(1-\beta_n)z_n + \beta_n Tz_n]\|_2 \qquad \text{(expanding } z_{n+1}) \\
&\leq (1-\beta_n)\|z_n - Tz_n\|_2 + \|z_n - (1-\beta_n)z_n - \beta_n Tz_n\|_2 \qquad \text{(nonexpansivity of } T) \\
&= (1-\beta_n)\|z_n - Tz_n\|_2 + \beta_n\|z_n - Tz_n\|_2 \\
&= \|z_n - Tz_n\|_2
\end{aligned}$$

This shows that $\|z_n - Tz_n\|_2$ is non-increasing.

Let z^* be the solution of the fixed-point problem (5.10). Then we have,

$$\begin{aligned}
\|z_{n+1} - z^*\|_2^2 &= \|T[(1-\beta_n)z_n + \beta_n Tz_n] - z^*\|_2^2 \\
&= \|T[(1-\beta_n)z_n + \beta_n Tz_n] - Tz^*\|_2^2 \qquad \text{(since } z^* = Tz^*) \\
&\leq \|(1-\beta_n)z_n + \beta_n Tz_n - z^*\|_2^2 \qquad \text{(nonexpansivity of } T) \\
&= \|(1-\beta_n)(z_n - z^*) + \beta_n(Tz_n - Tz^*)\|_2^2 \qquad \text{(by adding and subtracting } \beta_n z^*)
\end{aligned} \tag{5.12}$$

To proceed further, we will use the following relation from [65], for any $\varepsilon \in [0,1]$ and $u, v \in \mathscr{H}$,

$$\|(1-\beta_n)u - \beta_n v\|_2^2 = (1-\beta_n)\|u\|_2^2 + \beta_n\|v\|_2^2 - \beta_n(1-\beta_n)\|u-v\|_2^2$$

Using the above relation in (5.12), we get,

$$\|z_{n+1} - z^*\|_2^2 \leq \|z_n - z^*\|_2^2 - \beta_n(1-\beta_n)\|z_n - Tz_n\|_2^2$$

Taking sum over n leads to,

$$\left(\sum_{i=0}^{n} \beta_n(1-\beta_n)\right)\|z_n - Tz_n\|_2^2 \leq \sum_{i=0}^{n} \beta_n(1-\beta_n)\|z_i - Tz_i\|_2^2 \leq \|z_0 - z^*\|_2^2.$$

Thus, we have,

$$\|Tz_n - z_n\|^2 \leq \frac{\|z_0 - z^*\|_2^2}{\sum_{i=0}^{n}\beta_n(1-\beta_n)}.$$

It should be noted that, for $\beta_n = \beta \in (0,1)$, we have

$$\|Tz_n - z_n\|^2 \leq \frac{\|z_0 - z^*\|_2^2}{\beta(1-\beta)(n+1)},$$

Since $x_n \leftarrow J_\lambda^g\, z_n$, and J_λ^g is again a nonexpansive operator, following statements from [87] and [18], we can conclude that the EOSA algorithm achieves an overall $O(1/n)$ rate of convergence for solving the fixed point problem (5.1). We propose an accelerated variant of EOSA in the next subsection.

5.3.3 AEOSA

A recent trend in the field of first-order proximal methods is to accelerate the speed of convergence. With this aim, in [152], authors proposed an accelerated Douglas-Rachford algorithm and showed the convergence rate of the proposed algorithm. With the help of simulated results, they also showed that their proposed accelerated Douglas-Rachford algorithm (ADR) converges in lesser number of iterations than that of the traditional Douglas-Rachford algorithm.

With the same motivation, here we propose a novel accelerated operator splitting algorithm, named as **A**ccelerated **E**xtragradient-based **O**perator **S**plitting **A**lgorithm (AEOSA). For two convex functions f and g, with the initial values $z_0 = z_1 \in \mathscr{H}$ and $\beta \in (0,1)$, the AEOSA algorithm is defined as follows,

$$\begin{aligned} & x_n \leftarrow prox_{\lambda g}\, z_n \\ & y_n \leftarrow z_n + \alpha_n \,(z_n - z_{n-1}) \\ & z_{n+1} \leftarrow R^f_\lambda R^g_\lambda \,((1-\beta_n)\, y_n + \beta_n \, R^f_\lambda R^g_\lambda (y_n)) \quad \forall n = 1,2,\cdots \mathbb{N}, \end{aligned}$$

where sequences $\{\alpha_n\}_{n\in\mathbb{N}}$ and $\{\beta_n\}_{n\in\mathbb{N}} \in (0,1)$ and $\lambda > 0$. The pseudo code of the algorithm is given in 7. Another way to write this algorithm is as follows:

$$\begin{cases} x_n \leftarrow \text{prox}_{\lambda g}\,(v_n) \\ y_n \leftarrow \text{prox}_{\lambda f}\,(2x_n - v_n) \\ u_n \leftarrow z_n + y_n - x_n \\ w_n \leftarrow 2\ \text{prox}_{\lambda g}(u_n) - u_n \\ z_{n+1} \leftarrow 2\ \text{prox}_{\lambda f}(w_n) - w_n \\ v_{n+1} \leftarrow z_{n+1} + \alpha\,(z_{n+1} - z_n) \end{cases} \tag{5.13}$$

The condition *converge* is considered to be achieved when the difference between the function value at previous step and the function value at the current step becomes lesser than a previously defined tolerance value tol. In our experiments, we set the value of tol as 10e-4. The term $(x_n - x_{n-1})$ in algorithm 7 introduces an inertial step that produces acceleration with proper parameter settings and conditions on α_n. It should be noted that the term α_n is a generalized term, that was defined by the expression $\left(\frac{t_{n-1}-1}{t_n}\right)$ in [23] and $\frac{n-1}{n+3}$ in [47].

5.3.4 Convergence Analysis of AEOSA

To prove the convergence of the algorithm 7, we will give the following theorem based on theorem 3.5 of Chapter 3.

Theorem 5.5. *Let $\mathscr{H}$ be a Hilbert space and $T : \mathscr{H} \to \mathscr{H}$ is a non-expansive mapping. Let x^* be the solution of problem 5.1. For the initial points z_0 and $z_1 \in \mathscr{H}$, let $\{x_n\}_{n=0}^{\infty}$*

Algorithm 7: Accelerated Extragradient-based Operator Splitting Algorithm

```
Data: Training/Testing Data, $tol$
Result: $x_n$
function AEOSA
    initialize $z_0,\ z_1 \in \mathbb{R}^d,\ \lambda \in (0,+\infty),\ n = 0$;
    repeat
        $n \leftarrow n+1$;
        compute $\alpha_n$ and $\beta_n$ such that both $\alpha_n,\ \beta_n \in (0,1)$;
        $x_n \leftarrow prox_{\lambda g} z_n$;
        $u_n \leftarrow z_n + \alpha_n\,(z_n - z_{n-1})$;
        $v_n \leftarrow R_\lambda^f R_\lambda^g u_n$;
        $y_n \leftarrow (1-\beta_n)u_n + \beta_n v_n$;
        $z_{n+1} \leftarrow R_\lambda^f R_\lambda^g (y_n)$;
    until converge;
```

be a sequence generated by (5.13), where α_n and $\beta_n \in (0,1]$. Let $\lambda \in (0,2/L)$ and the sequence $\{z_n\}$ satisfies the following condition:

$$\sum_{n=1}^{\infty} \alpha_n \|z_n - z_{n-1}\|^2 < \infty.$$

Then $\{x_n\}_{n=0}^{\infty}$ converges to x^*.

Proof: Since $x_n \leftarrow J_\lambda^g\ z_n$, and J_λ^g is again a nonexpansive operator, following statements from [87] and [18], we can conclude that the theorem follows from (3.5).

5.4 Experiments and Result analysis

In this section, we present the lasso framework and the practical performance of the proposed algorithms for the task of binary classification for microarray gene analysis. We demonstrate the results on four benchmark real gene-expression datasets, which are used in the domain of bio-informatics.

Consider a learning framework with the training dataset with m instances denoted as $\mathscr{D} = \{(a_i, y_i),\ a_i \in \mathbb{R}^d$, and $y_i \in \mathbb{R}$ for $i = 1, \cdots, m\}$. Here, each pair (a_i, y_i) represents i^{th} input-output pair. We consider function $f(\cdot)$ of (5.1) as the squared loss function and function $g(\cdot)$ as non-smooth l_1 norm, which reduces problem (5.1) into the popular LASSO framework [191] as follows,

$$\min_{x \in \mathbb{R}^d} \frac{1}{2}\|Ax - b\|_2^2 + \rho\|x\|_1. \tag{5.14}$$

Note that $A = \{a_1, a_2, \cdots, a_i\}$ where $i = 1, 2, \cdots, m$, and each $a_i \in \mathbb{R}^d$ for $d-$dimensions and Y is a set of m real values (outcomes) for regression or the distinct class labels for classification (2 in this chapter, for binary classification), i.e. $Y = \{y_1, y_2, \cdots, y_i\}$ for $i = 1, 2, \cdots, m$. The parameter $x \in \mathbb{R}^d$ is the weight parameter, which sets weights to each dimension subject to the minimum loss. The ℓ_1 norm on parameter x shows that the resulting weights are required to be sparse. The parameter ρ is the sparsity controlling parameter and $\|\cdot\|_2$ is the Euclidean norm. The proximity operators with respect to the above equation (5.14) are computed using the procedure described in[151].

All the experiments including extensive parameter tuning are performed under the MATLAB computing environment. The configuration of the servers where the experiments are performed is as follows: Dell Power Edge R-930: Populated with a 4x18 core of Intel Xeon E7-8870 v3 @2.10 GHz processor with 45MB L3 Cache, 4U Form Factor, 256 GB DDR4 RAM, 8 x 1.2 TB 15K hot-plug SAS. We used the following publicly available high-dimensional gene expression datasets:

- **Colon-cancer Dataset [1]:** The dataset contains expression level of 2000 genes with highest minimal intensity in descending order from 62 patients. Among them, 40 tumor biopsies are from tumors, and 22 normal biopsies are from healthy parts of the colons of the same patients. We used 37 samples as training and 25 samples as testing selected at random. The number of dimensions is 2000.
- **Duke-cancer Dataset [2]:** This data set details microarray experiment for 44 breast cancer patients, out of which we use 26 records as training and 18 records as testing. The number of dimensions is 7,129. The binary variable *Status* is used to classify the patients into estrogen receptor-positive (*Status* = 0) and estrogen receptor-negative (*Status* = 1). The other variables contain the expression level of the considered genes.
- **Leukemia Dataset[1]:** Leukemias are primary disorders of bone marrow. The total number of genes to be tested is 7129, and the number of samples to be tested is 72,

[1]http://featureselection.asu.edu/datasets.php

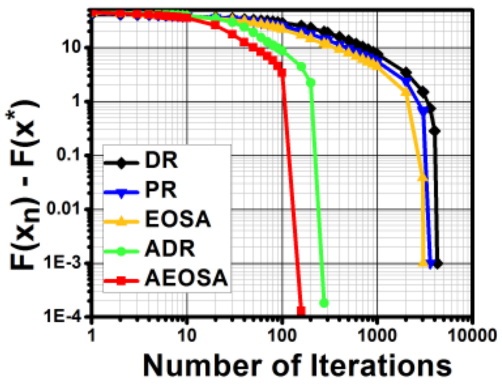
[2]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

which are all acute leukemia patients, either acute lymphoblastic leukemia (ALL) or acute myelogenous leukemia (AML). We used 38 samples as training samples and 34 samples as testing samples.

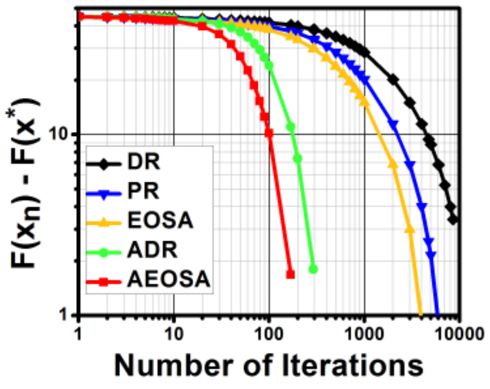
- **Prostate Dataset**[1]**:** The Prostate dataset contains 102 samples out of which 52 samples are samples of the person suffering from a prostate tumor, and 50 samples are of healthy persons. The total number of genes is 5966. We used 61 training samples and 41 testing samples in our experiments.

We compared our proposed algorithms, i.e. New Extragradient-based Operator splitting Algorithm (5.7) (**EOSA**) and its accelerated variant, Accelerated New Extragradient-based Operator splitting Algorithm (5.13) (**AEOSA**) with the classic Peaceman-Rachford Splitting Algorithm (5.4) (**PR**), the classic Douglas-Rachford splitting algorithm (5.5) (**DR**), a recent accelerated Douglas-Rachford splitting algorithm [152] (**ADR**). The value of sparsity controlling parameter ρ is set as $\{\theta \times \rho_{max}\}$, where ρ_{max} is set as $\|X^T Y\|_\infty$. The parameter θ is tuned in the range $\{$0-100$\}$ with an increment of 1. The parameter θ is tuned by five-fold cross-validation for all methods. As a pre-processing step, z-score is performed on X to normalize, and a bias column is added to the data. For the stopping criteria, the tolerance value (the difference between two consecutive function values) is set to 10e-4, which also notifies the convergence. The maximum number of iteration is set to 10e4. All the vectors are initialized with a zero-valued vector. Value of λ is initialized with 1, and β_n is set to $\frac{1}{(n+1)}$.

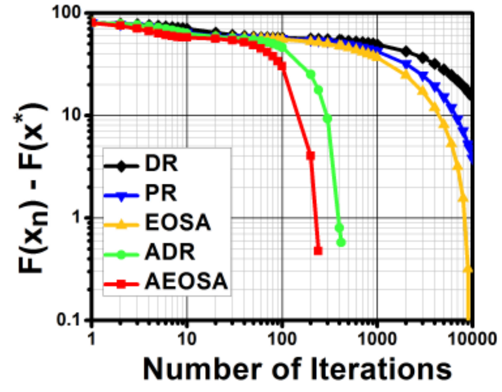
Our first experiment is the comparison of the first order operator splitting algorithms based on their convergence speeds. Results are shown in figure 5.1 as log-log plots, in which the y-axis show the term $F(x_n) - F(x^*)$ and x-axis are the number of iterations. It can be observed that among the non-accelerated algorithms EOSA algorithm converges faster on all the datasets. In the accelerated version, AEOSA is faster than the ADR algorithm. We also tried to implement the accelerated variant of PR algorithm, however, for the parameter set, we considered, the algorithm is not converging. Observing the figures, we can also say that among the non-accelerated traditional operator splitting algorithms, EOSA converges comparably faster than the PR and DR algorithms. With the *Prostate* dataset, few fluctuations can be observed with graph of ADR, which is a normal tendency with such inertial algorithms. Various restarting-based algorithms [148] can be adapted to handle such cases.



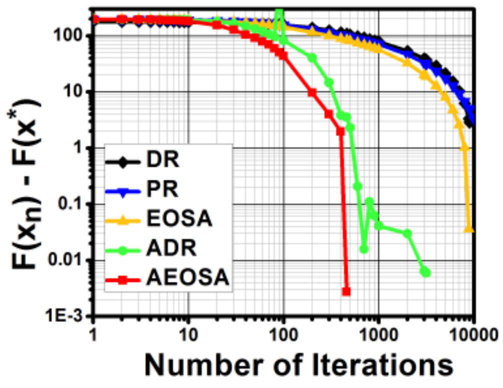
(a) Colon



(b) Duke



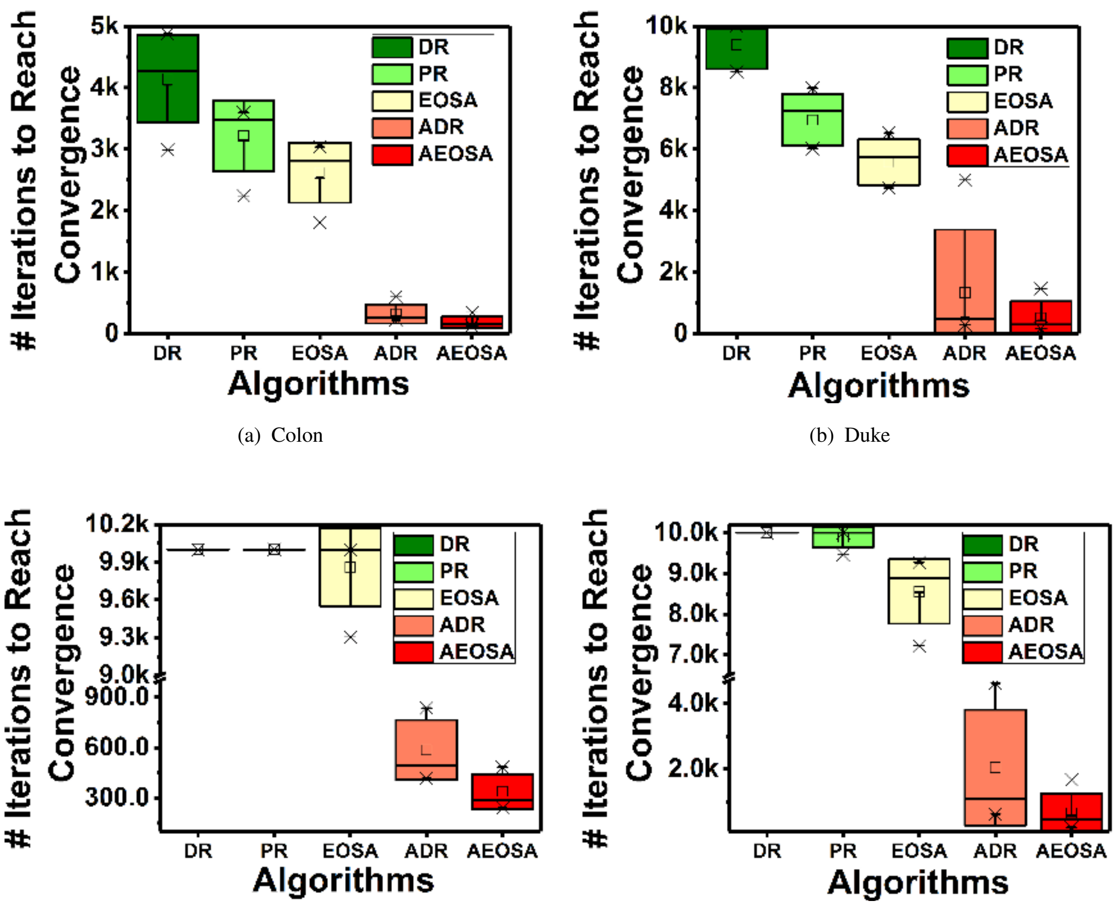
(c) Leukemia



(d) Prostate

FIGURE 5.1: Performance of VAGA on the basis of $F(x_n) - F(x^*)$ on the four Datasets, Colon, Duke, Leukemia and Prostate, respectively. $F(x_n)$ is the function value achieved after n^{th} iteration and $F(x^*)$ is the optimal function value. Shown graphs are log-log plots. We consider the convergence is reached if $F(x_n) - F(x_{n-1})$ becomes 10e-4. Maximum iterations is considered as 10e4.

In figure 5.2, the box-plot representations of the required number of iterations to reach the convergence point for all the algorithms are shown. As we discussed earlier also, the convergence is said to be achieved if the difference between the function values at consecutive points becomes lesser than a value tol, which we set as 10e-4. It can be observed that for most of the datasets, the DR algorithm consumes a larger number of iterations. The least number of iterations is consumed by AEOSA. For the *Leukemia* dataset, in all the experiments, PR and DR are reaching upto maximum number of iterations to converge. For this dataset, in most of the experiments, EOSA is also taking similar number of iterations to converge.



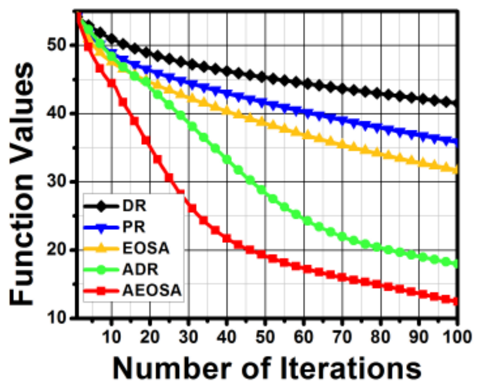
(a) Colon (b) Duke

(c) Leukemia (d) Prostate

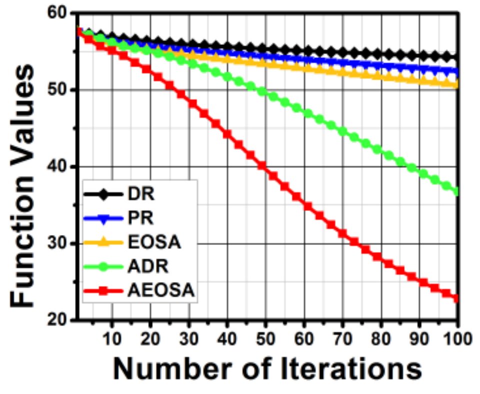
FIGURE 5.2: Performance of the newly proposed operator splitting algorithm along with the accelerated variant on the basis of number of iterations required by each algorithm for the four Datasets, Colon, Duke, Leukemia and Prostate, respectively.

We show the graphs between the objective function values in each iteration for the four datasets in figure 5.3. The rapid reductions in values of objective functions demonstrate the efficiency of the proposed algorithms. Here again, the AEOSA algorithm outperforms all the other algorithms. These values of objective functions are for $\theta = 10$ with initial 100 iterations.

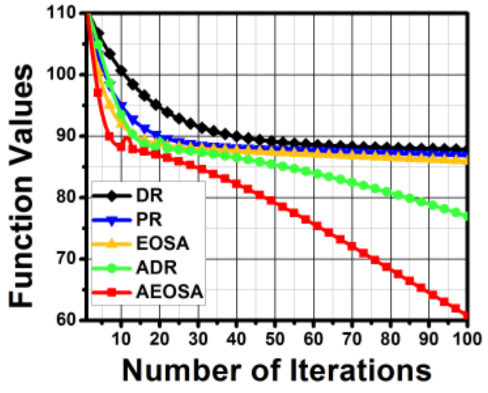
The classification accuracies for all the algorithms on all the datasets are shown in figure 5.4. The performance of both the accelerated algorithms ADR and AEOSA are almost the same, and better than the rest of the algorithms in most of the cases. For the *Duke* dataset, DR performs the best with respect to classification accuracy.



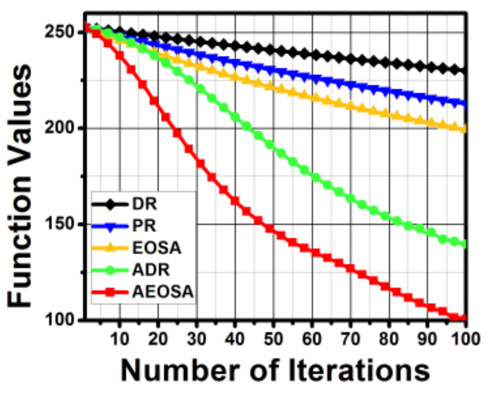
(a) Colon



(b) Duke



(c) Leukemia

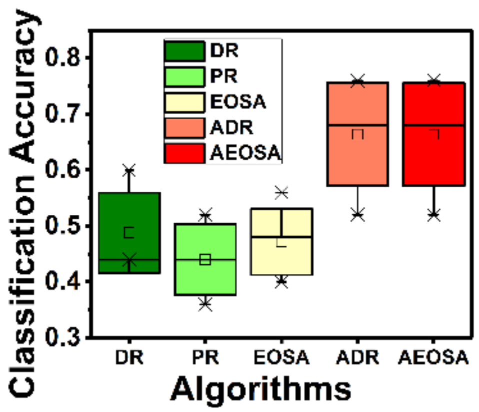


(d) Prostate

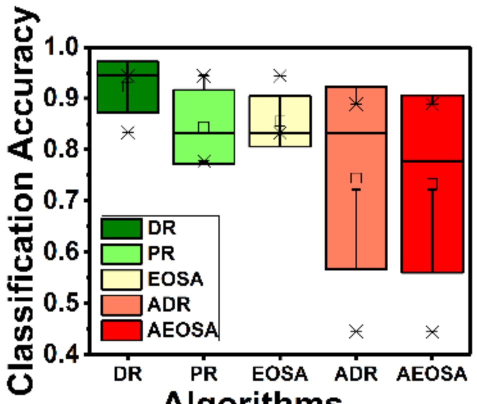
FIGURE 5.3: Performance of the newly proposed operator splitting algorithm along with the accelerated variant on the basis of Reduction in Function Value in each Iteration for the four Datasets, Colon, Duke, Leukemia and Prostate, respectively.

The one of the most important metric to check the performance of an optimization algorithm is the time consumption by that algorithm. In our experiments, we show the comparison between the CPU time consumed by all the algorithms in figure 5.5. The figures show that the CPU time consumed by the ADR and AEOSA algorithms is least and more and less equal. The time consumed by the EOSA algorithm is the largest, which can be directly implied by the designing of the iterative process.

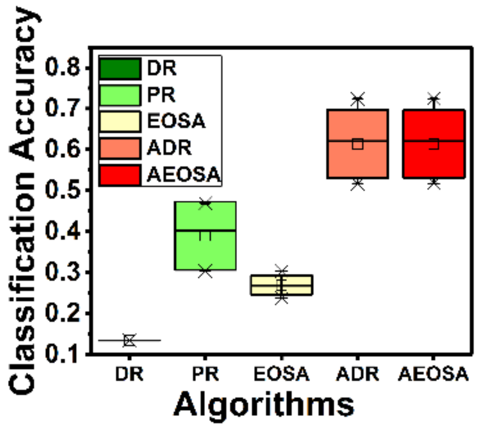
Table 5.1 shows the detailed result of all the four datasets, where the best values are shown in bold letters. The shown results are in terms of (i) the number of iterations to reach the convergence (**#Iter**), (ii) the minimum objective function value achieved (**optFV**), (iii) the classification accuracy (**Acc**) and (iv) the **CPU time** in seconds. Here, the convergence is



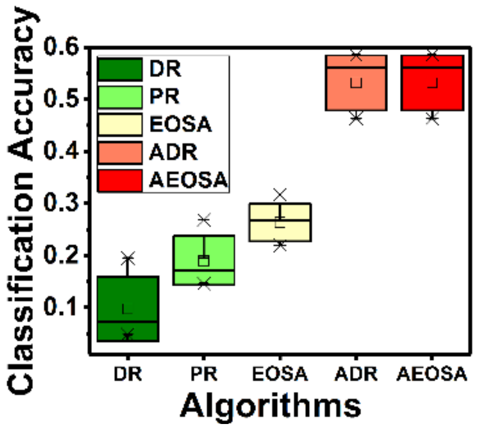
(a) Colon



(b) Duke



(c) Leukemia

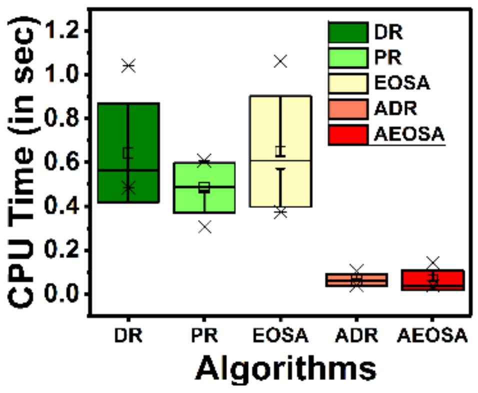


(d) Prostate

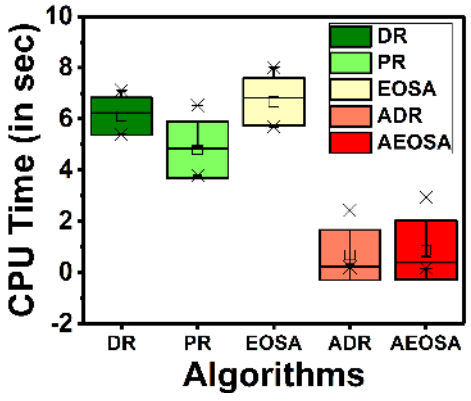
FIGURE 5.4: Performance of the newly proposed operator splitting algorithm along with the accelerated variant on the basis of classification accuracy by each algorithm for the four Datasets, Colon, Duke, Leukemia and Prostate, respectively.

considered to be achieved when the difference between two consecutive objective function values becomes less than the tolerance value (which is set as 10e-5). It is evident from the table that AEOSA takes significantly less number of iterations on all the datasets. The optimal objective function value achieved by AEOSA is also the least in all of datasets.

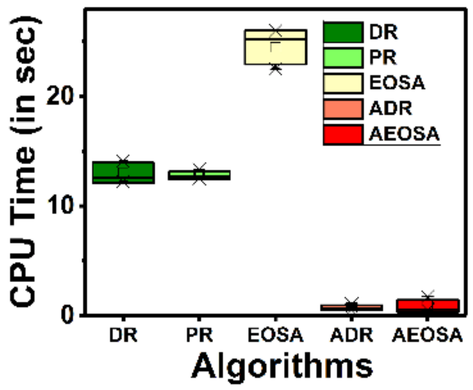
As far as the classification accuracy values are concerned, with all the datasets, the performances of algorithms ADR and AEOSA obtain same value. We have also checked for other metrics for measuring the performance of classification, such as precision and recall, but these values are also same for both the algorithms. We have found in our experiments that the per iteration CPU time of the proposed AEOSA algorithm is slightly greater than the ADR algorithm, which is clear from the basic design of the iterative process. However,



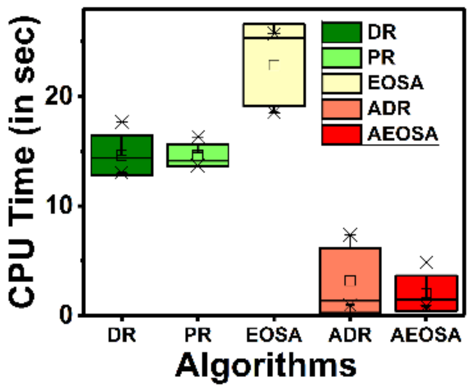
(a) Colon



(b) Duke



(c) Leukemia



(d) Prostate

FIGURE 5.5: Performance of the newly proposed operator splitting algorithm along with the accelerated variant on the basis of CPU Time required by each algorithm to reach the convergence for the four Datasets, Colon, Duke, Leukemia and Prostate, respectively. We consider the convergence is reached if $F(x_n) - F(x_{n-1})$ becomes 10e-4.

the significantly lesser number of iterations overcomes this problem. This observation reflects with *Colon*, *Duke* and *Prostate* datasets. With *Leukemia* dataset, AEOSA consumed more time than ADR.

5.5 Conclusion

In this chapter, we proposed an extragradient-based operator splitting method and an accelerated variant of the algorithm. We analyzed the convergence of both the algorithms. We applied both the algorithms to solve the classification problem with the logistic lasso

TABLE 5.1: Detailed Results for all the four datasets. Shown CPU time are in seconds. The values of objective functions are for $\theta = 10$ at 100^{th} iteration. We consider the convergence is reached if $F(x_n) - F(x_{n-1})$ becomes 10e-4.

		PR	**DR**	**EOSA**	**ADR**	**AEOSA**
Colon	**# Iter**	3213 (±573.36)	4146.4 (±714.90)	2620.4 (±489.82)	321.8 (±156.38)	**186** (±90.76)
	optFV	10.5395	12.6213	9.7823	9.1802	**9.0817**
	Acc	0.44 (±0.0632)	0.488 (±0.0715)	0.472 (±0.0593)	**0.664** (±0.0921)	**0.664** (±0.0921)
	CPU time	0.4844 (±0.1126)	0.6436 (±0.2249)	0.6492 (±0.2519)	0.0635 (±0.0273)	**0.0628** (±0.0452)
Duke	**# Iter**	6951.2 (±843.2767)	9400 (±788.0548)	5576.2 (±742.8907)	2332.8 (±4286.9963)	**514.8** (±538.6475)
	optFV	52.4176	54.2797	50.6825	36.7575	**22.8254**
	Acc	0.8444 (±0.0724)	0.9222 (±0.0496)	0.8556 (±0.0497)	0.7444 (±0.1783)	**0.7333** (±0.1730)
	CPU time	4.7981 (±1.1046)	6.1054 (±0.7289)	6.6694 (±0.9376)	1.4833 (±2.7681)	**0.8772** (±1.1620)
Leukemia	**# Iter**	10000 (±0.00)	10000 (±0.00)	9861.2 (±310.3662)	586.6 (±178.7394)	**339.4** (±103.3697)
	optFV	87.8187	86.8357	85.9028	76.9507	**60.7894**
	Acc	0.0758 (±0.0288)	0.0344 (±0.00)	0.0896 (±0.0308)	**0.6137** (±0.0823)	**0.6137** (±0.0823)
	CPU time	12.8078 (±0.3208)	13.0372 (±0.9196)	24.4879 (±1.5629)	**0.7368** (±0.2121)	0.8700 (±0.5486)
Prostate	**# Iter**	9891.4 (±242.8369)	10000 (±0.00)	8558.2 (±789.1449)	2052.4 (±1761.8692)	**670** (±580.8067)
	optFV	212.8267	230.0933	199.3723	139.6753	**100.6291**
	Acc	0.1902 (±0.0469)	0.0975 (±0.0621)	0.2634 (±0.0361)	**0.5317** (±0.0528)	**0.5317** (±0.0528)
	CPU time	14.6327 (±1.0264)	14.6291 (±1.8244)	22.8536 (±3.7409)	3.1870 (±2.9297)	**2.0190** (±1.6148)

framework along with an ℓ_1 regularizer. This framework is applied to the task of microarray gene-expression analysis. We used four benchmark publicly available microarray gene-expression datasets in our experiments. The two proposed algorithms are applied to solve this problem, and the performances are compared to the state-of-the-art operator splitting algorithms and their accelerated variants. The only limitation we found in our methods is the higher CPU-time due to the design of the iterative scheme, which in future we will try to handle by implementing the algorithms on GPUs.