Chapter 3

NAGA: New Accelerated Gradient Algorithm with Convergence and Stability Guarantees and Applications

3.1 Introduction

Many machine learning problems are designed using a regularized convex minimization framework, which is described as the minimization of the sum of a smooth convex loss function $f(\cdot)$ with gradient having Lipschitz constant L and a non-smooth convex regularization function $g(\cdot)$, both defined over d-dimensional real space $\mathbb{R}^d$, as,

$$\min_{x \in \mathbb{R}^d} \; F(x) = f(x) + \rho \cdot g(x). \tag{3.1}$$

Few classic applications where this framework is applicable include recognition [209, 189, 132], recommender systems [5, 116], natural language processing [68, 60], computational biology [32, 223], web search ranking [51], similarity learning [50] etc. The main problem in solving (3.1) is due to the non-smooth regularization function. Previously proposed methods that solve this framework include interior point method (IPM) [196], projected subgradient method [160], blockwise coordinate descent algorithm [121], forward-looking subgradients [77] and so on. Although IPMs provide highly accurate

results in low iteration counts, the iteration cost grows super-linearly with the number of decision variables [106]. In addition, the convergence of subgradient-based methods is slow in practice. The computationally cheaper iterations and faster convergence of proximal-based methods motivated us to analyze them further.

In this work, the nonsmooth sparsity-inducing regularizers are considered, which are defined, based on non-smooth ℓ_1 norm. The main reason to employ ℓ_1 norm is to induce sparsity while learning the model, which is suitable for large dimensional problems. A squared loss function along with the ℓ_1 norm formulates the popular LASSO framework [191], which is well-utilized for the regression problems. Such sparsity-inducing regularizers consider the internal structure of the feature sets, which are efficiently learned with the help of proximal-based methods. In the settings of machine learning, various lasso extension based frameworks are available such as Group Lasso [14, 217] via the $\ell_1 - \ell_2$ block-norm [52] or the $\ell_1 - \ell_q$ block-norm [200], $\ell_2 - \ell_0$ norm [125], tree structure [219, 105], fused lasso [192], graph structure [218] etc. We followed the original lasso framework in the current work, however, a direct extension is also applicable for the extended lasso settings.

As discussed in Chapter 2, in its standard form, proximal gradient methods (PGA) belong to the general class of iterative forward-backward splitting algorithms of convex optimization theory. The main drawback of the traditional proximal gradient methods is its slow convergence, which can sometimes be overcome with the help of an additional acceleration step. Such acceleration step need not necessarily accelerate the speed of convergence. Thus, a detailed study is required for such acceleration based gradient methods. In the current chapter, we propose a novel accelerated proximal gradient algorithm, namely New Accelerated Gradient Algorithm (NAGA), which outperforms the traditional proximal gradient methods (both accelerated and non-accelerated versions). We provide the convergence and stability analysis of the algorithm. The forward-backward operator can be both non-expansive and contraction. We provide the convergence proof for both types of operators. We tested the practical performance of NAGA for three real-world applications, one of which is the high-dimensional regression/classification problem, the second one is the joint subspace learning from the cross-domain datasets, and the third is to solve the extended lasso frameworks with the overlapping group and fused penalties. We compared its performance against previously proposed proximal algorithms based on the number of iterations needed to converge, the minimum objective function value

achieved and the prediction as well as classification accuracy. Experimental results for all the tasks (regression, classification and joint subspace learning) conclude that NAGA outperforms previous proximal algorithms and thus we can claim it to be suitable for use as the optimization method in machine learning tasks.

3.1.1 Contributions

The contribution of the chapter is five-fold:

- We propose a new extragradient-based acceleration gradient algorithm, named as NAGA for solving the non-smooth convex optimization problem. We applied the algorithm to solve the Lasso framework. New definitions of proximal gradient methods are also designed using latest fixed-point iterations.
- We present the convergence and stability guarantees of the proposed algorithm. We derived the boundedness and convergence of the algorithm and proved that with the contraction mappings, the proposed fixed-point scheme is stable. The convergence of the proposed algorithm is proved under two cases. The first case considers the contraction mapping, whereas the second case considers the more general nonexpansive mappings.
- We applied all the algorithms to the task of the high-dimensional regression/classification problem, and experiments are performed on nine publicly available real datasets.
- We also applied the algorithm to the task of unified sparse representation learning, and present the experiments and result analysis with two real cross-modal datasets.
- In the end, we applied the algorithm to solve few extended-lasso frameworks, including overlapped group lasso and fused lasso penalties, and demonstrated the performance with three publicly available real datasets.

3.1.2 Outline

The organization of this chapter is as follows. Section 3.2 describes the background of the problem, few mathematical terms, and related concepts. Section 3.3 introduces our proposed algorithm to solve the LASSO problem. The mathematical derivations and analysis are discussed in section 3.4. Section 3.5 includes the experimental analysis of the algorithm, where several publicly available real-world datasets were utilized, and a detailed comparison between traditional proximal algorithms are presented. We also present the performance of the proposed NAGA algorithm on the tasks of unified sparse representation learning and to solve the extended lasso frameworks with several publicly available real datasets in this section. Finally, we conclude our work in Section 3.6.

3.2 Related Concepts

We focus to solve the following problem,

$$\min_{x\in\mathbb{R}^d} F(x) = f(x) + g(x), \tag{3.2}$$

which can be solved by traditional PGA defined by the following iterative scheme,

$$x_{n+1} \leftarrow prox_{\lambda g}(Id - \lambda\nabla f)(x_n), \tag{3.3}$$

where x_n denotes the n^{th} iteration of the algorithm. In order to accelerate the convergence of PGA, the following Acceleration Gradient Algorithm (AGA) was proposed in [23],

$$\begin{aligned} y_n &\leftarrow x_n + \alpha_n(x_n - x_{n-1}), \\ x_{n+1} &\leftarrow prox_{\lambda g}(Id - \lambda\nabla f)(y_n) \quad \forall n \in \mathbb{N}. \end{aligned} \tag{3.4}$$

Here $\alpha_n = \left(\frac{t_{n-1}-1}{t_n}\right)$ with $t_{n+1} \leftarrow \frac{1+\sqrt{1+4t_n^2}}{2}$ and $g(\cdot)$ is a $\|\cdot\|_1$ function. Other definitions for α_n are also available, for example [47, 138].

We already defined the proximal gradient methods in more general settings of the infinite dimensional Hilbert space $\mathscr{H}$ in Chapter 2. The forward-backward operator $J_\lambda^{f,g} = (I +$

$\lambda\partial g)^{-1}(I-\lambda\nabla f)$ can be denoted as T and $J_{\lambda_n}^{f,g}=(I+\lambda_n\partial g)^{-1}(I-\lambda_n\nabla f)$ can be denoted as T_n for the sake of simplicity in the proofs.

In order to guarantee the convergence, the value of $\lambda_n \in (0,2/L)$ [23]. The iterative procedure in (3.3) can be rewritten as,

$$x_{n+1} \leftarrow T_n(x_n), \tag{3.5}$$

which is called the Picard fixed-point iterations. Considering the definition of T_n as a forward-backward operator, it is also called the Forward-Backward Algorithm (FBA). In the field of fixed point theory, this scheme plays an important role because of the *Banach Contraction Principle*, which states that for contraction type mappings, this scheme converges to a fixed point in Banach space. Many iterative schemes are proposed thereafter for different mappings and spaces in the field of fixed point theory. In the similar spirit Mann in [128] proposed another fixed point iteration as follows:

$$x_{n+1} \leftarrow \gamma_n x_n + (1-\gamma_n)T_n x_n, \ \text{ for } \ n \geq 0, \tag{3.6}$$

for an arbitrary $x_1 \in \mathscr{H}$. In other words, it is the successive average iteration, that considers the following conditions on the sequence $\{x_n\}$,
(A_1) $0 \leq \gamma_n < 1$,
(A_2) $\sum_{n=1}^{\infty}\gamma_n = \infty$.

If we consider the operator T_n as a forward-backward operator, we will call (3.6) as the Mann-based Forward-Backward Algorithm (MFBA). In [1], authors successfully designed a new iterative scheme which converges with a rate similar to Picard iterations, but faster than Mann iteration for contraction mappings. The authors called this iteration as the S-iteration process defined for $x_1 \in \mathscr{H}$ as follows,

$$\begin{aligned} y_n &\leftarrow (1-\beta_n)x_n + \beta_n T_n x_n, \\ x_{n+1} &\leftarrow (1-\gamma_n)T_n x_n + \gamma_n T_n y_n, \end{aligned} \tag{3.7}$$

where $\{\gamma_n\}$ and $\{\beta_n\}$ are two real sequences in the interval (0,1] with the following condition,

$(B_1)\sum_{n=1}^{\infty}\gamma_n\beta_n(1-\beta_n)=\infty$.

Since the S-iteration process cannot reduce to the Mann iterative scheme, so this algorithm is free from the Mann algorithm. In the field of fixed point theory, S-iteration and similar schemes are known as extra-gradient fixed-point methods. A brief discussion about the extragradient methods is presented in Chapter 2. With respect to the forward-backward operator T_n, we name (3.7) as the S-iteration-based Forward-Backward Algorithm (SFBA). In [173], author presented normal S-iteration algorithm as follows,

$$x_{n+1} \leftarrow T_n((1-\beta_n)x_n + \beta_n T_n x_n), \tag{3.8}$$

where $\{\beta_n\} \in (0,1]$. This scheme is also referred as hybrid Picard-Mann iteration in the literature of fixed point theory. The author also claimed that the proposed fixed-point iterative scheme converges faster than the Picard iterative scheme. We name (3.8) with respect to the forward-backward operator T_n as, the Normal S-iteration-based Forward-Backward Algorithm (NSFBA). It should be noted that corresponding to these fixed point iterative schemes, different definitions of proximal gradient algorithms can be given.

In this work, it is the first time that we applied these methods to the problem of convex minimization for the task of machine learning and compare the performance of these algorithms on the real datasets. In addition, we combine the Normal S-iteration-based Forward-Backward Algorithm with an inertial step to design its accelerated version. We investigate the behavior of the proposed algorithm for the task of prediction. Note that in the similar work, [127] included the inertial term with the Mann iteration process.

3.3 NAGA

In this section, the New Inertial-based Forward-Backward Algorithm (NIFBA) is proposed. The algorithm is a modification of classic inertial-based forward-backward splitting algorithm (3.4). Consider $A : \mathscr{H} \to 2^{\mathscr{H}}$ and $B : C \to \mathscr{H}$ are two maximal monotone operators, and B has the co-coercive property with respect to the solution set. Let $\{\alpha_n\}$ and $\{\beta_n\}$ are two sequences in $(0,1]$ and $\{\lambda_n\}$ is a regularization sequence in $(0,2/L)$. We define $T = J^A_\lambda(I-\lambda B)$ and $T_n = J^A_{\lambda_n}(I-\lambda_n B)$, where $\lim_{k\to\infty} \lambda_n = \lambda$. For any x_0 and $x_1 \in \mathscr{H}$,

the NIFBA is defined as follows:

$$\begin{aligned} y_n &\leftarrow x_n + \alpha_n(x_n - x_{n-1}) \\ z_n &\leftarrow (1-\beta_n)\, y_n + \beta_n\, T_n y_n \\ x_{n+1} &\leftarrow T_n z_n \end{aligned} \tag{3.9}$$

The term y_n introduces an inertial extrapolate step that produces acceleration with proper parameter settings. In our experiments, the sequence α_n is generated in a way similar to [47]. This algorithm is applied to solve the Lasso problem given as follows.

Consider a learning framework with the training dataset with m instances denoted as $\mathscr{D} = \{(x_i, y_i),\ x_i \in \mathbb{R}^d$, and $y_i \in \mathbb{R}$ for $i = 1, \cdots, m\}$. Here, each pair (x_i, y_i) represents i^{th} input-output pair. We consider function $f(\cdot)$ of (3.1) as the squared loss function and function $g(\cdot)$ as non-smooth l_1 norm, which reduces problem (3.1) into the popular LASSO framework [191] as follows,

$$\min_{w \in \mathbb{R}^d} \ \frac{1}{2}\|Y - Xw\|_2^2 + \rho\|w\|_1. \tag{3.10}$$

Note that $X = \{x_1, x_2, \cdots, x_i\}$ where $i = 1, 2, \cdots, m$, and each $x_i \in \mathbb{R}^d$ for $d-$dimensions and Y is a set of m real values (outcomes) for regression or the distinct class labels for classification, i.e. $Y = \{y_1, y_2, \cdots y_i\}$ for $i = 1, 2, \cdots, m$. The parameter $w \in \mathbb{R}^d$ is the weight parameter, which sets weights to each dimension subject to the minimum loss. The ℓ_1 norm on parameter w 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. For any $\lambda_n \in (0, 2/L]$, the corresponding forward backward operator $J^{f,g}_{\rho\lambda_n}$ is defined as follows,

$$J^{f,g}_{\rho\lambda_n}(w) = \text{prox}_{\rho\lambda_n g}(w - \rho\lambda_n f(w)),$$

The proximity operator for l_1-norm is the shrinkage operator, defined as follows [151],

$$\text{prox}_{\rho\lambda_n\|\cdot\|_1}(w) = (|w_i| - \rho\lambda_n)_+ \ \text{sgn}(w_i),$$

where sgn$(\cdot)$ is signum function. We define the new proximal gradient algorithm with respect to the NIFBA (3.9) as New Accelerated Gradient Algorithm (NAGA). The pseudo-code of NAGA is given in algorithm 2.

Algorithm 2: NAGA

Data: Training/Testing Data, ρ, tol
Result: w_{n+1}
begin
  $w_0 = w_1 \in \mathscr{H}, \lambda_1 = 1, \alpha_1 = 0, n = 0;$
  repeat
    $n \leftarrow n+1;$
    Find λ_n using backtracking step-size rule, and compute α_n and β_n;
    $T_n \leftarrow w_n + \alpha_n(w_n - w_{n-1});$
    $u_n \leftarrow \text{prox}_{\rho\lambda_n\|\cdot\|_1}(T_n - \lambda_n(X^TXT_n - X^TY));$
    $v_n \leftarrow (1-\beta_n)T_n + \beta_n u_n;$
    $w_{n+1} \leftarrow \text{prox}_{\rho\lambda_n\|\cdot\|_1}(v_n - \lambda_n(X^TXv_n - X^TY));$
  until *converge*;

The condition *converge* is considered to be achieved when the difference between the function value at the 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-5. We have considered that the value of L is not known in advance and thus we adopt the backtracking line search in each iteration, as considered in [23] and almost all the proximal gradient techniques proposed thereafter. Also, to establish the convergence, we ensure that value of λ_n belongs to set $(0, 2/L]$. It should be noted that the computational cost of each iteration is slightly higher than the classic accelerated method. However, the reduction in the number of iterations due to the proposed arrangement compensates for this overhead. In the next section, we will discuss the mathematical properties of the algorithm.

3.4 Analysis of NAGA

In this section, the convergence of the algorithm for the contraction as well as non-expansive mappings are presented. In order to analyze the method, we will utilize the following lemmas and theorems.

Lemma 3.1. *[127] Assume* $\phi_n \in [0,\infty)$ *and* $\delta_n \in [0,\infty)$ *satisfy:*
(1) $\phi_{n+1} - \phi_n \leq \theta_n(\phi_n - \phi_{n-1}) + \delta_n$,
(2) $\sum_{n=1}^{+\infty} \delta_n < \infty$,

(3) $\{\theta_n\} \subset [0,\theta]$, *where* $\theta \in [0,1)$.
Then, the sequence $\{\phi_n\}$ *is convergent with* $\sum_{n=1}^{+\infty}[\phi_{n+1}-\phi_n] < \infty$, *where* $[t]_+ = max\{t,0\}$ *for any* $t \in \mathbb{R}$.

From the property of projection operator we have,

$$\forall x^* \in C, \quad \text{we have } \langle x - J^A_{\lambda_n}(x), x^* - J^A_{\lambda_n}(x)\rangle \leq 0 \tag{3.11}$$

Lemma 3.2. *[149] Let* $\mathscr{H}$ *be a Hilbert space and* $\{x_n\}$ *a sequence such that there exists a nonempty set* $S \subset \mathscr{H}$ *verifying:*
(i) For every $\bar{x} \in S$, $lim_{n\to\infty}|x_n - \bar{x}|$ *exists.*
(ii) If x_v *weakly converges to* $x \in \mathscr{H}$ *for a subsequence* $v \to \infty$, *then* $x \in S$.
Then, there exists $x^* \in S$ *such that* $\{x_n\}$ *weakly converges to* x^* *in* $\mathscr{H}$.

Lemma 3.3. *[27] Let* δ *be a real number satisfying* $0 \leq \delta < 1$ *and* $\{\varepsilon_n\}_{n=0}^{\infty}$ *be a sequence of positive numbers such that* $\lim_{n\to\infty}\varepsilon_n = 0$. *Then, for any sequence of positive numbers* $\{u_n\}_{n=0}^{\infty}$ *satisfying* $u_{n+1} \leq \delta u_n + \varepsilon_n$, *for* $n = 0,1,\cdots$, *we have* $\lim_{n\to\infty} u_n = 0$.

We start the mathematical analysis with the boundedness proof of the sequence generated by the proposed algorithm.

3.4.1 Boundedness of $\{x_n\}$ from NAGA

Let x^* is the solution of problem (3.1), then can write, $x^* = J^A_{\lambda_n}(x^*)$ and $B(x^*) = 0$. For convenience, we define the following terms,

$$\phi_n = \frac{\|x_n - x^*\|^2}{2},\ \delta_n = \frac{\|x_n - x_{n-1}\|^2}{2},\ \Gamma_n = \frac{\|x_n - y_n\|^2}{2}$$

We will use the following identity in our proofs,

$$\langle a-b, a-c\rangle = \frac{1}{2}\|a-b\|^2 + \frac{1}{2}\|a-c\|^2 - \frac{1}{2}\|b-c\|^2 \tag{3.12}$$

From the definition of T_n, non-expansivity of the resolvent operator $J^A_{\lambda_n}$ and coercivity of B, we get

$$\begin{aligned}
\phi_{n+1} &= \frac{1}{2}\|x_{n+1}-x^*\|^2 = \frac{1}{2}\|J^A_{\lambda_n}(I-\lambda_n B)z_n - x^*\|^2 \\
&\leq \frac{1}{2}\|(z_n-x^*)-\lambda_n B(z_n)\|^2 \\
&= \frac{1}{2}[\|z_n-x^*\|^2 - 2\langle z_n-x^*, B(z_n)-B(x^*)\rangle + \lambda_n^2\|B(z_n)\|^2] \\
&\leq \frac{1}{2}[\|z_n-x^*\|^2 + \lambda_n^2\|B(z_n)\|^2 - \frac{2\lambda_n}{L}\|B(z_n)\|^2] \\
&= \frac{1}{2}\|z_n-x^*\|^2 - \left(\frac{\lambda_n}{L}-\frac{\lambda_n^2}{2}\right)\|B(z_n)\|^2. \qquad (3.13)
\end{aligned}$$

Note that, from the property of non-expansivity of operator T_n and (3.9), we have,

$$\begin{aligned}
\frac{1}{2}\|z_n-x^*\|^2 &= \frac{1}{2}\|(1-\beta_n)y_n + \beta_n T_n y_n - x^*\|^2 \\
&= \frac{1}{2}[(1-\beta_n)^2\|y_n-x^*\|^2 + \beta_n^2\|T_n y_n - x^*\|^2 + 2(1-\beta_n)\beta_n\langle y_n-x^*, T_n y_n - x^*\rangle] \\
&\leq \frac{1}{2}\|y_n-x^*\|^2.
\end{aligned}$$

Substituting back this value in (3.13) we get,

$$\phi_{n+1} = \frac{1}{2}\|y_n-x^*\|^2 - \left(\frac{\lambda_n}{L}-\frac{\lambda_n^2}{2}\right)\|B(z_n)\|^2. \qquad (3.14)$$

From the definition of y_n in (3.9), we get,

$$\begin{aligned}
\frac{1}{2}\|y_n-x^*\|^2 &= \frac{1}{2}\|x_n+\alpha_n(x_n-x_{n-1})-x^*\|^2 \\
&= \frac{1}{2}[\|x_n-x^*\|^2 + \alpha_n^2\|x_n-x_{n-1}\|^2 + 2\alpha_n\langle x_n-x^*, x_n-x_{n-1}\rangle] \\
&= \phi_n + \alpha_n^2\delta_n + \alpha_n\langle x_n-x^*, x_n-x_{n-1}\rangle
\end{aligned}$$

From identity (3.12), we get,

$$\frac{1}{2}\|y_n-x^*\|^2 = \phi_n + \frac{\alpha_n^2+\alpha_n}{2}\delta_n + \alpha_n(\phi_n-\phi_{n-1}). \qquad (3.15)$$

Substituting back in (3.14), we get,

$$\phi_{n+1} \leq \phi_n + \frac{\alpha_n^2 + \alpha_n}{2}\delta_n + \alpha_n(\phi_n - \phi_{n-1}) - \left(\frac{\lambda_n}{L} - \frac{\lambda_n^2}{2}\right)\|B(z_n)\|^2. \tag{3.16}$$

Since $0 < \lambda_n < \frac{2}{L}$, we have $(\frac{\lambda_n}{L} - \frac{\lambda_n^2}{2}) > 0$. Also from the conditions on α_n, we have $\frac{\alpha_n^2+\alpha_n}{2} \leq \alpha_n$. Thus, we derive,

$$\phi_{n+1} \leq \phi_n + \alpha_n\delta_n + \alpha_n(\phi_n - \phi_{n-1}). \tag{3.17}$$

Following the same line of derivation of [123], we deduce that $\{\phi_n\}$ is convergent. Thus, the sequence $\{x_n\}$ is bounded. Notice that at this point we get the first requirement of Opial's theorem.

3.4.2 Convergence with Contraction Mappings

Let $\{x_n\}$ be a sequence in $\mathscr{H}$, and $T : \mathscr{H} \to \mathscr{H}$ be a contraction mapping. Let p be a fixed point of T. In [35] authors proved that every contraction mapping that has a fixed point, satisfies the following inequality, for $0 \leq \delta < 1$,

$$\|p - Tx\| \leq \delta\|p - x\| \quad \text{for } x \in \mathscr{H}. \tag{3.18}$$

The property of non-expansivity of the operator T_n holds if $0 \leq \lambda_n \leq 2/L$. However, to prove the convergence and stability guarantee with the contraction property, we assume the contraction property of the operators, which holds for $0 < \lambda_n < 2/L$ [6].

Theorem 3.4. *Let $\mathscr{H}$ be a Hilbert space and $T : \mathscr{H} \to \mathscr{H}$ be a contraction mapping that satisfies the condition (3.18) for $0 \leq \delta < 1$ and fixed point p. For the initial points x_0 and x_1, let $\{x_n\}_{n=0}^{\infty}$ is a sequence generated by (3.9), where $\{\alpha_n\}$ and $\{\beta_n\} \in (0,1]$. Let the sequence $\{x_n\}$ satisfies the following condition,*

$$\|x_n - x_{n-1}\|/\alpha_n \to 0 \ \ as \ \ n \to \infty. \tag{3.19}$$

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

Proof. From (3.18) and (3.9), we have

$$\begin{aligned}
\|x_{n+1}-p\| &= \|T_n[(1-\beta_n)y_n+\beta_n T_n y_n]-p\| \\
&\leq \delta\ \|((1-\beta_n)y_n+\beta_n T_n y_n)-p\| \\
&\leq \delta\ ((1-\beta_n)\ \|y_n-p\|+\beta_n\ \|T_n y_n-p\|) \\
&\leq \delta\ ((1-\beta_n(1-\delta))\ \|y_n-p\|) \\
&\leq \delta\ \left[(1-\beta_n(1-\delta))\ \|x_n-p\|+\alpha_n(1-\beta_n(1-\delta))\ \|x_n-x_{n-1}\|\ \right] \\
&\leq \delta\ \left[(1-\beta_n(1-\delta))\|x_n-p\|+(1-\beta_n(1-\delta))\ \frac{\|x_n-x_{n-1}\|}{\alpha_n}\ \right]
\end{aligned}$$

Since $\{\alpha_n\}$ and $\{\beta_n\} \in (0,1]$, $0 \leq \delta < 1$, from condition (3.19) and lemma (3.3), we get $lim_{n\to\infty}\|x_{n+1}-p\| = 0$. This ends the proof.

□

3.4.3 Convergence with Non-expansive Mappings

Theorem 3.5. *Let $\mathscr{H}$ be a Hilbert space and $T : \mathscr{H} \to \mathscr{H}$ be a non-expansive mapping. Let x^* is the solution of problem 3.1. For the initial points x_0 and x_1, let $\{x_n\}_{n=0}^{\infty}$ is a sequence generated by (3.9), where α_n and $\beta_n \in (0,1]$. Let the sequence $\{x_n\}$ satisfies the following condition:*

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

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

Proof. Since the sequence $\{\phi_n\}$ is convergent (proved in section 3.4.1), from (3.17) and Lemma (3.1), we have $\sum_{n=1}^{+\infty}[\|x_n-x^*\|^2 - \|x_{n-1}-x^*\|^2] < \infty$. Thus, from (3.16) we can write,

$$\left(\frac{\lambda_n}{L}-\frac{\lambda_n^2}{2}\right)\|B(z_n)\|^2 \leq \phi_n - \phi_{n+1} + \alpha_n\delta_n + \alpha_n(\phi_n-\phi_{n-1}) \tag{3.20}$$

Thus,

$$\sum_{n=1}^{+\infty}\left(\frac{\lambda_n}{L}-\frac{\lambda_n^2}{2}\right)\|B(z_n)\|^2 < \infty$$

Since $0 < \lambda_n < \frac{2}{L}$, we have $B(z_n) \to 0$. Also, we can write,

$$\begin{aligned}\|x_{n+1} - x^*\|^2 &= \|(x_{n+1} - y_n) + (y_n - x^*)\|^2 \\ &= \|x_{n+1} - y_n\|^2 + \|y_n - x^*\|^2 + 2\langle x_{n+1} - y_n, y_n - x_{n+1}\rangle \\ &\quad + 2\langle x_{n+1} - y_n, x_{n+1} - x^*\rangle\end{aligned}$$

Rearranging the terms, we get,

$$\|x_{n+1} - y_n\|^2 = \|y_n - x^*\|^2 - \|x_{n+1} - x^*\|^2 + 2\langle x_{n+1} - y_n, x_{n+1} - x^*\rangle$$

Substituting value of $\|y_n - x^*\|^2$ from (3.15) we get,

$$\begin{aligned}\|x_{n+1} - y_n\|^2 &= 2\phi_n + (\alpha_n^2 + \alpha_n)\delta_n + 2\alpha_n(\phi_n - \phi_{n-1}) \\ &\quad - \|x_{n+1} - x^*\|^2 + 2\langle x_{n+1} - y_n, x_{n+1} - x^*\rangle \\ &= (\|x_n - x^*\|^2 - \|x_{n+1} - x^*\|^2) + \alpha_n(\|x_n - x^*\|^2 - \|x_{n-1} - x^*\|^2) \\ &\quad + 2\alpha_n\delta_n + 2\langle x_{n+1} - y_n, x_{n+1} - x^*\rangle \qquad (3.21)\end{aligned}$$

Now, for projection operator $J^A_{\lambda_n}$, we put $x = z_n - \lambda_n B(z_n)$ in (3.11) to get the following inequality,

$$\begin{aligned}0 &\geq \langle z_n - \lambda_n B(z_n) - x_{n+1}, x^* - x_{n+1}\rangle \\ \lambda_n\langle B(z_n), x^* - x_{n+1}\rangle &\geq \langle x_{n+1} - z_n, x_{n+1} - x^*\rangle \\ \lambda_n\langle B(z_n), x^* - x_{n+1}\rangle &\geq \langle x_{n+1} - y_n, x_{n+1} - x^*\rangle + \beta_n\langle y_n - T_n y_n, x_{n+1} - x^*\rangle. \qquad (3.22)\end{aligned}$$

From (3.9), and the nonexpansivity of operator T_n we can write,

$$\begin{aligned}\|y_n - T_n y_n\| &= \|y_n - x_{n+1}\| + \|x_{n+1} - T_n y_n\| \\ &\leq \|x_n - x_{n+1}\| + \alpha_n\|x_n - x_{n-1}\| + \|z_n - y_n\| \\ &= \|x_{n+1} - x_n\| + \alpha_n\|x_n - x_{n-1}\| + \beta_n\|y_n - T_n y_n\| \\ &\leq \frac{1}{(1-\beta_n)}\left[\alpha_n\|x_n - x_{n-1}\| + \|x_n - x_{n+1}\|\right] \to 0 \quad \text{as } n \to \infty\end{aligned}$$

Thus, from (3.22), we can write,

$$\langle x_{n+1} - y_n, x_{n+1} - x^*\rangle \leq \lambda_n\langle B(z_n), x^* - x_{n+1}\rangle \to 0 \quad \text{as } n \to \infty$$

Hence, from (3.21), we get,

$$\|x_{n+1} - y_n\| \to 0$$

In last step, we will show that the solution x^* is the solution of the problem (3.1). In the previous subsection we proved that the sequence $\{x_n\}$ generated by (3.9) is bounded. Thus, we have a convergent subsequence x_s such that $x_s \rightharpoonup x^*$. From the definition of y_n in (3.9), we can write $y_s \rightharpoonup x^*$. Also, since $\|y_n - T_n y_n\| \to 0$, we get,

$$\|z_n - x_n\| \leq \|y_n - x_n\| + \beta_n \|y_n - T_n y_n\|,$$

which gives $z_s \rightharpoonup x^*$. Thus, from (3.9), we get the following:

$$\begin{aligned} x^* &= (1-\beta_n)x^* + \beta_n (I + \lambda_n A)^{-1}(I - \lambda_n B)x^* \text{ and} \\ x^* &= (I + \lambda_n A)^{-1}(I - \lambda_n B)x^*, \end{aligned}$$

which is equivalent to $-B(x^*) \in A(x^*)$, which proves that x^* is a solution. Thus, from Lemma 3.2, we conclude the proof. □

3.4.4 Stability Analysis of NAGA

The concept of T-stability of an iterative process is defined as follows. Let $T : \mathscr{H} \to \mathscr{H}$ be an operator. Let $\{x_n\}_{n\in\mathbb{N}} \in \mathscr{H}$ be the sequence generated by an iterative procedure involving T which is defined by,

$$x_{n+1} = h(T, x_n) \text{ for } n \in \mathbb{N}, \tag{3.23}$$

where h is some function, that defines the iteration scheme. Suppose $\{x_n\}_{n\in\mathbb{N}} \in \mathscr{H}$ converges to a fixed point p of T. Let $\{y_n\}_{n\in\mathbb{N}} \in \mathscr{H}$ and set $\varepsilon_n := d(y_{n+1}, h(T, y_n))$ for $n \in \mathbb{N}$, where $d(\cdot)$ is a distance function ($\|\cdot\|$ in real normed linear space). Then, the iteration process (3.23) is said to be T-stable or stable with respect to T if $\lim\limits_{n\to\infty} \varepsilon_n = 0$ implies $\lim\limits_{n\to\infty} y_n = p$.

Theorem 3.6. *Let $\mathscr{H}$ be a Hilbert space and $T : \mathscr{H} \to \mathscr{H}$ be a contraction mapping that satisfies the condition (3.18) for $0 \leq \delta < 1$ and fixed point p. For the initial points*

x_0 and x_1, let $\{x_n\}_{n=0}^{\infty}$ is a sequence generated by (3.9), where α_n and $\beta_n \in (0,1]$. Let the sequence satisfies condition (3.19). Then the iterative scheme in (3.9) is T-stable.

Proof. Let $\{p_n\}_{n=0}^{\infty}$ is a real sequence in $\mathscr{H}$. According to (3.23) and (3.9), We define $\varepsilon_n = \|p_{n+1} - T_n[(1-\beta_n)q_n + \beta_n T_n q_n]\|$, where $q_n = p_n + \alpha_n(p_n - p_{n-1})$. Let $\lim_{n\to\infty} \varepsilon_n = 0$. Then we shall prove that $\lim_{n\to\infty} p_n = p$.

$$\begin{aligned}
\|p_{n+1} - p\| &= \|p_{n+1} - T_n[(1-\beta_n)q_n + \beta_n T_n q_n] + T_n[(1-\beta_n)q_n + \beta_n T_n q_n] - p\| \\
&\leq \varepsilon_n + \delta \ \|(1-\beta_n)\ q_n + \beta_n T_n q_n - p\| \\
&\leq \varepsilon_n + \delta \ [(1-(1-\delta)\ \beta_n)\ \|q_n - p\| \\
&\leq \varepsilon_n + \delta \ \Big[(1-(1-\delta)\ \beta_n)\ \|p_n - p\| + \alpha_n(1-(1-\delta)\ \beta_n)\ \|p_n - p_{n-1}\|\Big] \\
&\leq \varepsilon_n + \delta \ \Big[(1-(1-\delta)\ \beta_n)\ \|p_n - p\| + (1-(1-\delta)\ \beta_n)\frac{\|p_n - p_{n-1}\|}{\alpha_n}\Big]
\end{aligned}$$

Since $\{\alpha_n\}$ and $\{\beta_n\} \in (0,1]$, $0 \leq \delta < 1$, from condition (3.19) and lemma (3.3), we have $\lim_{n\to\infty} p_n = p$.

Conversely, let $\lim_{n\to\infty} p_n = p$, we shall show that $\lim_{n\to\infty} \varepsilon_n = 0$. We have,

$$\begin{aligned}
\varepsilon_n &= \|p_{n+1} - T_n[(1-\beta_n)q_n + \beta_n T_n q_n]\| \\
&\leq \|p_{n+1} - p\| + \delta \ \|(1-\beta_n)q_n + \beta_n T_n q_n - p\| \\
&\leq \|p_{n+1} - p\| + \delta \ \Big[(1-(1-\delta)\beta_n)\ \|q_n - p\|\Big] \\
&\leq \|p_{n+1} - p\| + \delta \ \Big[(1-(1-\delta)\beta_n)\ \|p_n - p\| + (1-(1-\delta)\ \beta_n)\ \frac{\|p_n - p_{n-1}\|}{\alpha_n}\Big]
\end{aligned}$$

Since $\{\alpha_n\}$ and $\{\beta_n\} \in (0,1]$, $0 \leq \delta < 1$, from condition (3.19) and our assumption, we have $\lim_{n\to\infty} \varepsilon_n = 0$. This ends the proof. □

3.5 Applications, Experiments and Result Analysis

We first present the regression and classification performance of the algorithms on nine benchmark high-dimensional datasets. In the next subsection, we will discuss the task of joint subspace learning and present the experimental results with two real cross-modal datasets. The performance of the algorithms to solve the extended lasso problem with

the overlapping group and fused lasso penalties will be discussed next. All the tests have been performed on an Intel Core i7 processor with 10 GB RAM, under the MATLAB computing environment.

3.5.1 Machine Learning with high-dimensional datasets

In this section, we give the details of the high-dimensional real datasets we used in our experiments, the experimental setup, and the result analysis for the task of high-dimensional regression and classification. We used the following publicly available high dimensional datasets:

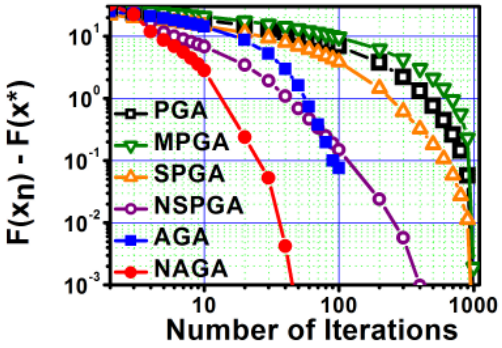
- **Colon-cancer Dataset** [1]**:** The dataset contains expression level of 2000 genes with highest minimal intensity in descending order from 62 patients. Among them, 40 tumor biopsies are from tumors, and 22 normal biopsies are from healthy parts of the colons of the same patients. We used 37 samples 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.

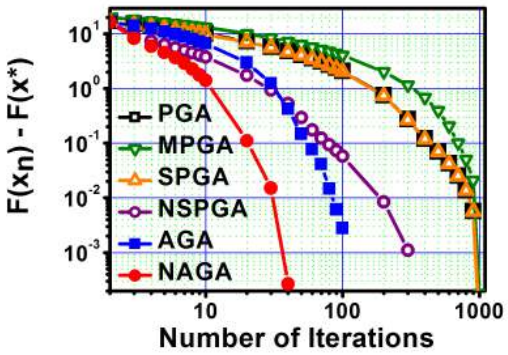
- **Gisette Dataset**[1]**:** A handwritten digit recognition dataset. The problem is to separate the highly confusable digits '4' and '9'. The high order features are constructed as a product of randomly sampled pixels from the middle top part of the images of digits to plunge the problem in a higher dimensional feature space. We used 250 samples as training and 250 samples as testing samples.

- **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, which are all acute leukemia patients, either acute lymphoblastic leukemia (ALL)

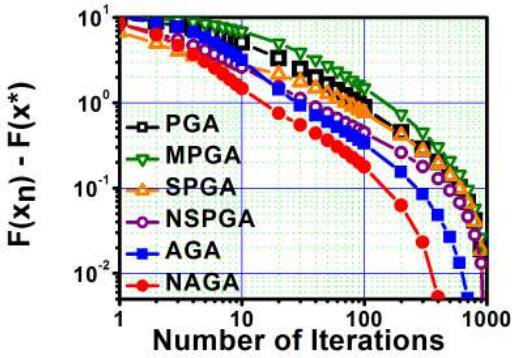
[1] http://featureselection.asu.edu/datasets.php
[2] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/



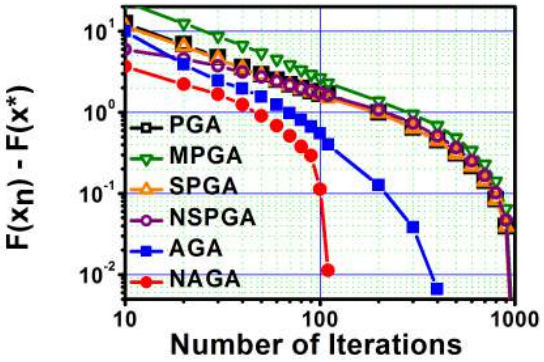
(a) Lymph Dataset



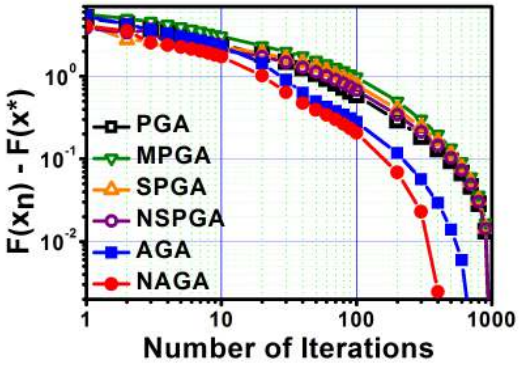
(b) Nci9 Dataset



(c) Prostate Dataset

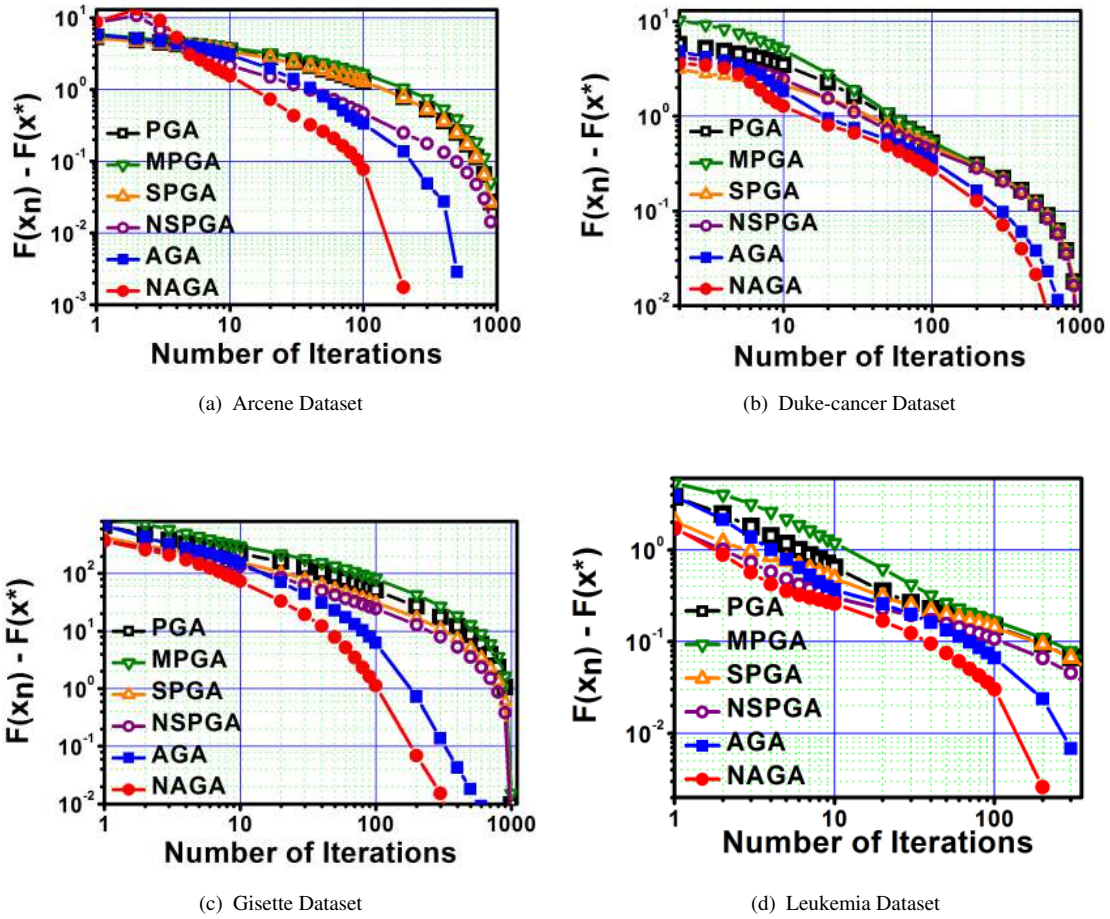


(d) Lung Dataset



(e) Colon-cancer Dataset

FIGURE 3.1: Performance of algorithms on the basis of $F(x_n) - F(x^*)$ on datasets. $F(x_n)$ is the function value achieved after n^{th} iteration and $F(x^*)$ is the function value at the optimal point x^*. Shown graphs are log-log plots. We consider the convergence is reached if $F(x_n) - F(x_{n-1})$ becomes 10e-5. Maximum number of iterations is set as 10e3. The value of parameter θ is set 0.05 in all the experiments.



(a) Arcene Dataset

(b) Duke-cancer Dataset

(c) Gisette Dataset

(d) Leukemia Dataset

FIGURE 3.2: Performance of algorithms on the basis of $F(x_n) - F(x^*)$ on datasets. $F(x_n)$ is the function value achieved after n^{th} iteration and $F(x^*)$ is the function value at the optimal point x^*. Shown graphs are log-log plots. We consider the convergence is reached if $F(x_n) - F(x_{n-1})$ becomes 10e-5. Maximum number of iterations is set as 10e3. The value of parameter θ is set 0.05 in all the experiments.

or acute myelogenous leukemia (AML). We used 38 samples as training samples and 34 samples as testing samples.

- **Arcene Dataset[1]:** It is mass-spectrometric data that is useful in distinguishing between the cancer patients versus normal patients. This is a two-class classification problem with continuous input variables. Dataset consists of 200 samples (88 cancer patients and rest healthy or control patients) with 10,000 number of features. 1200 training samples and 800 testing samples are used.

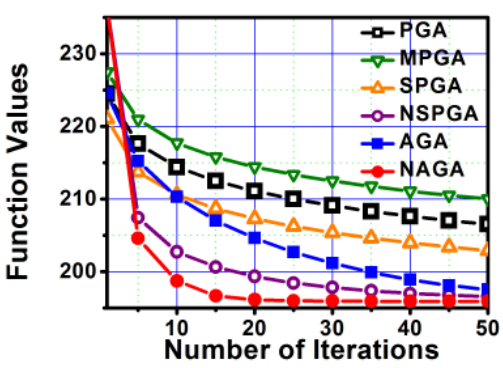
- **Lung Dataset[1]:** A gene expression dataset with 325 genes, 73 samples, and 7 classes. We used 44 training samples and 29 testing samples in our experiments.

- **Lymphoma Dataset**[1]**:** A gene expression dataset with 4026 genes, 96 samples, and 9 classes. We used 58 training samples and 38 testing samples in our experiments.
- **Nci9 Dataset**[1]**:** A gene expression dataset with 9712 genes, 60 samples, and 9 classes. We used 36 training samples and 24 testing samples in our experiments.
- **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.

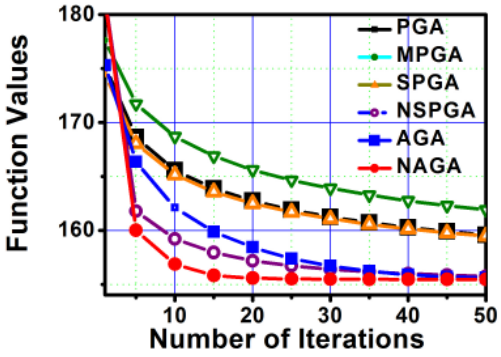
We compared our proposed New Accelerated Gradient Algorithm (3.9) (**NAGA**) with the classic Proximal Gradient Algorithm (3.3) (**PGA**) (with Picard iterations), the Proximal Gradient Descent Algorithm with Mann iteration scheme (3.6) (**MPGA**) as proposed in [128], the Proximal Gradient Algorithm with the S-iteration scheme (3.7) (**SPGA**) as proposed in [1], the Proximal gradient Algorithm with Normal S-iteration scheme (3.8) (**NSPGA**) proposed in [173] and the Accelerated Gradient Algorithm (3.4) (AGA) from [47]. In all the above stated equations, we replace the forward-backward operator T_n as $\text{prox}_{\rho c_n \|\cdot\|_1}(I - c_n \nabla f)$ in order to give the proximal-gradient-based definitions. It should be noted that it is the first time that these fixed-point iteration schemes are compared based on their performances for the task of learning.

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 $\{1, 0.0001\}$ with an increment of 0.001. We performed experiments with 60% - 40% training and testing samples split for most of the datasets. 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-5, which also notifies the convergence. The maximum number of iteration is set to 10e3. All the vectors are initialized with a zero-valued vector. Value of λ_n is initialized with 1, and γ_n and β_n are set to $\frac{1}{(n+1)}$.

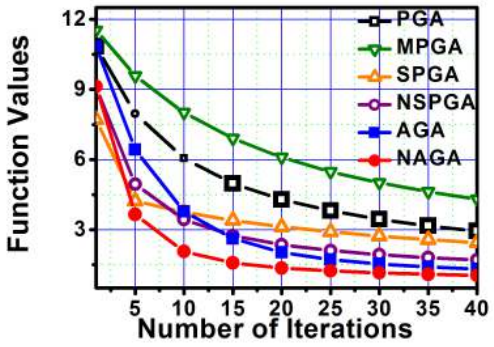
Our first experiment is the comparison of first order proximal gradient algorithms based on their convergence speed. Results are shown in figures 3.1 and 3.2 as log-log plot, in which the y-axis shows the term $F(x_n) - F(x^*)$ and x-axis is the number of iterations. It



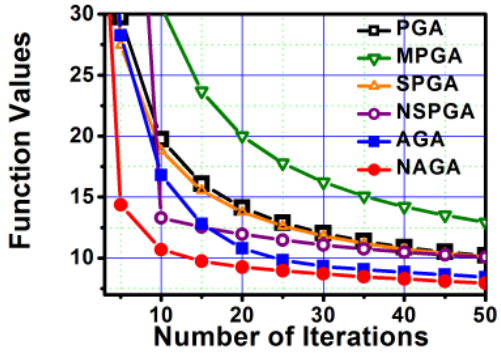
(a) Lymph Dataset



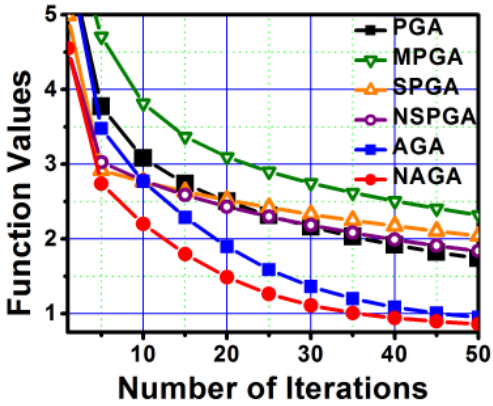
(b) Nci9 Dataset



(c) Prostate Dataset

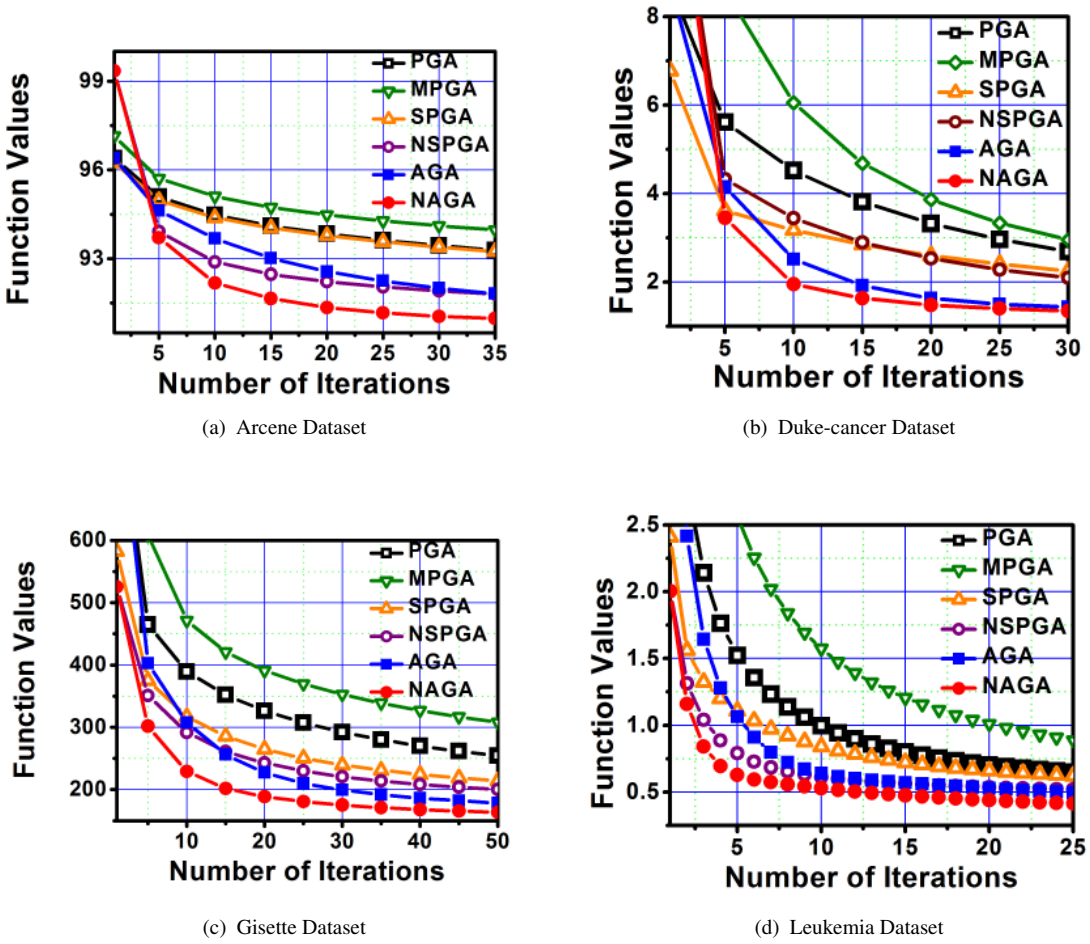


(d) Lung Dataset



(e) Colon-cancer Dataset

FIGURE 3.3: Performance of NAGA on the basis of Reduction in Function Values in each iteration for the datasets. We consider the convergence is reached if $F(x_n) - F(x_{n-1})$ becomes 10e-5. Maximum number of iterations is set as 10e3. The value of parameter θ is set 0.05 in all the experiments.



(a) Arcene Dataset

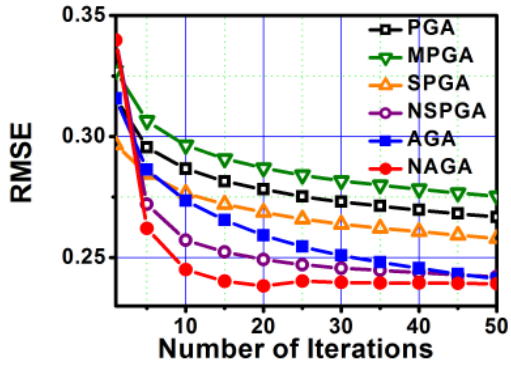
(b) Duke-cancer Dataset

(c) Gisette Dataset

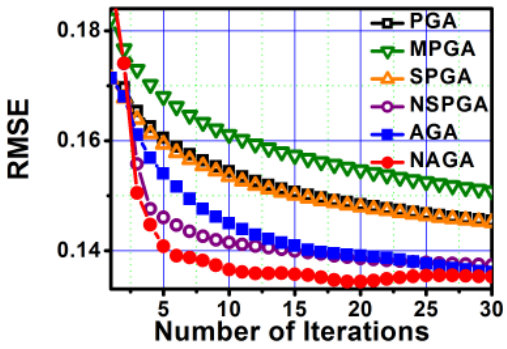
(d) Leukemia Dataset

FIGURE 3.4: Performance of NAGA on the basis of Reduction in Function Values in each iteration for the datasets. We consider the convergence is reached if $F(x_n) - F(x_{n-1})$ becomes 10e-5. Maximum number of iterations is set as 10e3. The value of parameter θ is set 0.05 in all the experiments.

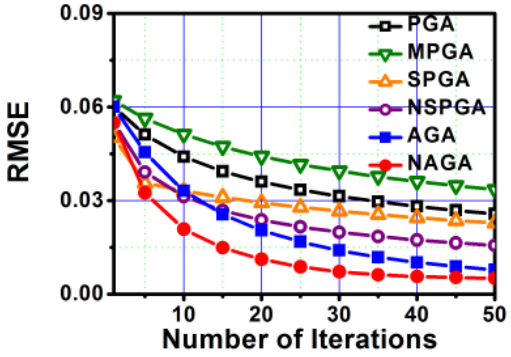
can be observed that NAGA algorithm converges faster than all the previous state-of-the-art algorithms on all the datasets. It can be noted that the convergence speed of MPGA is worst in all cases. It is interesting to note that for *Lymph* and *Nci9* datasets, the NSPGA algorithm converges faster than AGA for first few iterations. This observation points out the behavior of an extragradient-based method (NSPGA) in comparison to the inertial based method (AGA).



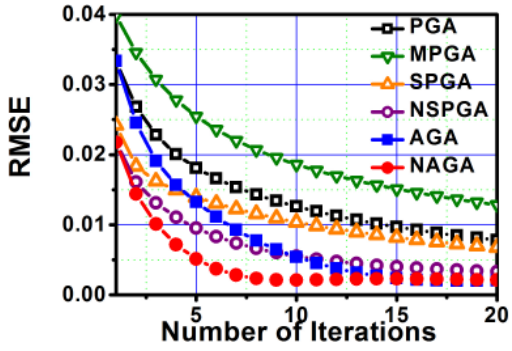
(a) Lymph Dataset



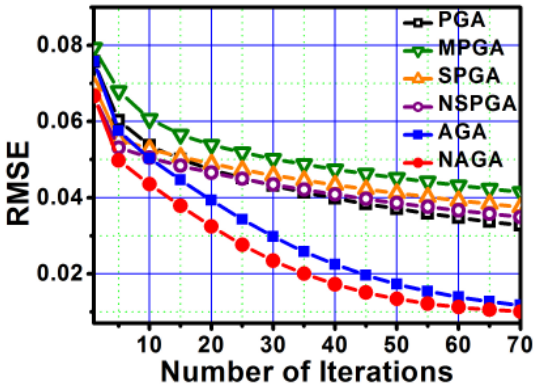
(b) Nci9 Dataset



(c) Prostate Dataset

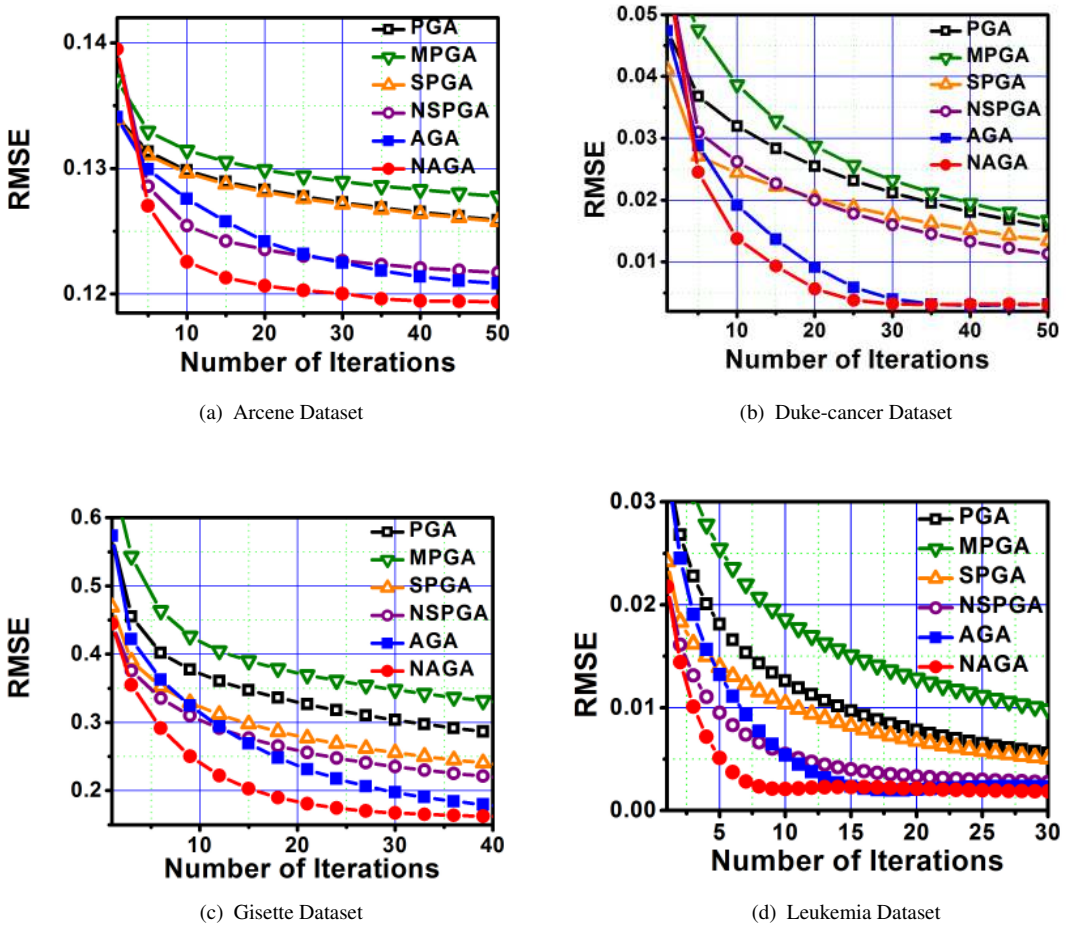


(d) Lung Dataset



(e) Colon-cancer Dataset

FIGURE 3.5: Performance of NAGA on the basis of root Mean Square Error rate on the datasets. We consider the convergence is reached if $F(x_n) - F(x_{n-1})$ becomes 10e-5. Maximum number of iterations is set as 10e3. The value of parameter θ is set 0.05 in all the experiments.



(a) Arcene Dataset

(b) Duke-cancer Dataset

(c) Gisette Dataset

(d) Leukemia Dataset

FIGURE 3.6: Performance of NAGA on the basis of root Mean Square Error rate on the datasets. We consider the convergence is reached if $F(x_n) - F(x_{n-1})$ becomes 10e-5. Maximum number of iterations is set as 10e3. The value of parameter θ is set 0.05 in all the experiments.

In figures 3.3 and 3.4, we show the graphs between the objective function values at each iteration for the nine datasets. The rapid reductions in values of objective functions demonstrate the efficiency of the proposed algorithm. With SPGA and NSPGA, objective function values reduce very rapidly in first few iterations, however, with increasing iterations these reductions become lower than that of with PGA. These lowering in the reductions in objective function values, however, are not much significant with NAGA, and thus overall, the final function value is significantly lesser than the one obtained with AGA. The similar phenomenon has been observed with *Duke-cancer*, *Leukemia* and *Gisette* datasets, however, here the performances of SPGA and NSPGA are better than PGA algorithm.

The regression accuracy at each iteration for all the algorithms on all the datasets are shown in figures 3.5 and 3.6. We consider the standard root Mean Square Error (rMSE) to demonstrate the prediction performance of the proposed algorithm. It can be concluded that NAGA gives better error reduction rate, i.e., the reduction in rMSE per iteration on all the datasets. The detailed results are given in table 3.1.

Table 3.1 shows the detailed result for all the 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, (ii) the rMSE values, (iii) the minimum objective value obtained and (iv) the per iteration CPU time. Here, the convergence is 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 NAGA takes significantly less number of iterations on all the datasets. The optimal objective function value achieved by NAGA is also the least among the majority of datasets. As far as rMSE values are concerned, with nine out of nine datasets NAGA beats all the other algorithms. For *Duke*, *Leukemia*, *Gisette* and *Lung* datasets AGA achieves the least rMSE values. However, the number of iterations it takes to reach this value is much higher than that of NAGA. It is evident from the results that the per iteration CPU time of the proposed NAGA algorithm is slightly greater than the traditional AGA, which is clear from the basic design of the iterative process. However, the significantly lesser number of iterations overcomes this problem. It should be noted in general that CPU time increment is not very significant and can be further reduced by exploiting multi-core programming techniques, which we will investigate in future.

To further analyze the algorithms, we also conducted the experiments for the classification task. We selected the *Colon, Duke, Leukemia, Gisette, Arcene* and *Prostate* datasets for the binary classification problem. Again, we split the datasets into 60% - 40% training and testing samples and the parameter θ is tuned by five-fold cross-validation for all methods. Results are shown in table 3.2 in terms of the classification accuracy (**Acc**), precision (**Pre**), recall (**Rec**) and specificity (**Spec**) values. On four out of six datasets, the NAGA algorithm outperforms other methods in terms of classification accuracy. For *Gisette* dataset, the MPGA algorithm beats other methods in all the four measures of performance. It should be noted that the shown results are the average of ten experiments.

We finally conclude this section with the statistical comparison of the above-mentioned algorithms PGA, MPGA, SPGA, NSPGA, AGA and NAGA based on their rMSE values.

TABLE 3.1: Detailed Results for all the datasets for the task of Regression. Shown CPU time (in seconds) is for one iteration.

		PGA	MPGA	SPGA	NSPGA	AGA	NAGA
Colon	**# Iter**	1000	1000	1000	886	291	**147**
	optFV	2.4286	2.2897	2.4267	2.1679	2.1254	**2.1105**
	rMSE	0.0312	0.0285	0.0311	0.0259	0.0243	**0.0241**
	CPU time	0.00271	**0.00266**	0.00497	0.00485	0.00278	0.00520
Duke	**# Iter**	1000	1000	1000	1000	391	**185**
	optFV	7.6053	7.6056	7.4364	7.3786	7.2746	**7.2639**
	rMSE	0.0199	0.0199	0.0195	0.0192	0.0186	**0.0183**
	CPU time	0.0311	**0.0309**	0.0610	0.0609	0.0317	0.0646
Leukemia	**# Iter**	1000	1000	883	807	265	**129**
	optFV	2.5215	2.6466	2.4501	2.4234	**2.3630**	2.3686
	rMSE	0.0113	0.0120	0.0109	0.0108	0.0108	**0.0106**
	CPU time	0.0312	**0.0310**	0.0617	0.0618	0.0326	0.0674
Gisette	**# Iter**	1000	1000	1000	1000	683	**400**
	optFV	2.4169	2.6083	2.4159	2.1318	1.8416	**1.813**
	rMSE	0.0035	0.0036	0.0035	0.0031	0.0029	**0.0028**
	CPU time	0.0213	**0.0211**	0.0364	0.0360	0.0213	0.0368
Arcene	**# Iter**	1000	1000	1000	1000	788	**164**
	optFV	54.7590	56.9161	54.7389	51.6142	**50.7922**	50.9181
	rMSE	0.0636	0.0692	0.0635	0.0568	0.0552	**0.0550**
	CPU time	**0.0645**	**0.0645**	0.1267	0.1244	0.0653	0.1349
Lung	**# Iter**	764	1000	761	301	111	**64**
	optFV	58.7491	58.8276	58.7491	58.7420	58.7351	**58.7344**
	rMSE	0.3407	0.3412	0.3407	0.3406	0.3408	**0.3402**
	CPU time	0.000195	**0.000169**	0.000232	0.000223	0.000187	0.000297
Lymphoma	**# Iter**	1000	1000	1000	1000	410	**206**
	optFV	68.2763	71.6653	68.2616	63.3859	62.9609	**62.9456**
	rMSE	0.0794	0.0840	0.0794	0.0755	0.0750	**0.0742**
	CPU time	**0.01006**	0.01007	0.01940	0.01990	0.01073	0.02062
Nci9	**# Iter**	1000	1000	1000	1000	495	243
	optFV	35.7367	38.2826	38.2332	32.9622	31.2347	31.09237
	rMSE	0.02532	0.02767	0.02763	0.02283	0.02088	0.02052
	CPU time	0.0601	0.0618	0.1216	0.1209	0.0716	0.1331
Prostate	**# Iter**	1000	1000	1000	1000	260	**127**
	optFV	6.2932	6.6072	6.2885	5.7212	5.6583	**5.6275**
	rMSE	0.0296	0.0312	0.0296	0.0276	0.0273	**0.0270**
	CPU time	0.02198	**0.02192**	0.04346	0.04306	0.02328	0.04661

TABLE 3.2: Comparison of Algorithms based on their Classification Performances. Shown CPU time (in seconds) is for one iteration.

		PGA	MPGA	SPGA	NSPGA	AGA	NAGA
Colon	**Acc**	**0.92**	**0.92**	0.88	0.88	0.833	0.833
	Pre	**1.000**	**1.000**	**1.000**	**1.000**	0.9333	0.9333
	Rec	**0.8889**	**0.8889**	0.8333	0.8333	0.7778	0.7778
	Spec	**1.000**	**1.000**	**1.000**	**1.000**	0.8571	0.8571
Duke	**Acc**	0.8333	0.8333	0.8333	0.8333	**0.8889**	**0.8889**
	Pre	0.8182	0.8182	0.8182	0.8182	**0.9000**	**0.9000**
	Rec	**0.9000**	**0.9000**	**0.9000**	**0.9000**	**0.9000**	**0.9000**
	Spec	0.7500	0.7500	0.7500	0.7500	**0.8750**	**0.8750**
Leukemia	**Acc**	0.9655	0.9655	0.9310	0.9655	0.9655	**1.000**
	Pre	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	Rec	0.95	0.95	0.9	0.95	0.95	**1.000**
	Spec	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Gisette	**Acc**	0.9683	**0.9717**	0.9683	0.9642	0.9608	0.9692
	Pre	0.9666	**0.9699**	0.9666	0.9586	0.9629	0.9667
	Rec	0.9698	**0.9732**	0.9698	0.9698	0.9581	0.9715
	Spec	0.9669	**0.9702**	0.9669	0.9586	0.9635	0.9669
Arcene	**Acc**	0.7284	0.7284	0.7284	0.7161	**0.7531**	**0.7531**
	Pre	0.6545	0.6545	0.6545	0.6379	**0.6727**	**0.6727**
	Rec	0.9231	0.9231	0.9231	**0.9487**	**0.9487**	**0.9487**
	Spec	0.5476	0.5476	0.5476	0.500	**0.5714**	**0.5714**
Prostate	**Acc**	0.9268	0.9268	0.9512	**0.9756**	0.9512	**0.9756**
	Pre	0.9583	0.9583	0.96	**0.9615**	0.96	**0.9615**
	Rec	0.92	0.92	0.96	**1.000**	0.96	**1.000**
	Spec	**0.9375**	**0.9375**	**0.9375**	**0.9375**	**0.9375**	**0.9375**

We conducted the Wilcoxon signed-ranks test [203], which shows the pair-wise comparison of the six algorithms based on the rMSE values. The null hypothesis is set to check whether there is a significant reduction in the rMSE values returned by the NAGA algorithm with respect to other algorithms. The test results are listed in the table (3.3) in terms of the p-values and the corresponding z-values. It can be concluded from the table that the reduction in rMSE values with the NAGA algorithm is significant in comparison to the other algorithms. Also, it can be seen that there are no significant differences in rMSE values between PGA, MPGA and SPGA algorithms.

TABLE 3.3: The result of Wilcoxon signed-ranks test for Comparison of the rMSE values obtained by the algorithms. Shown are the z-scores (above diagonal) and p-values (below diagonal).

Methods	PGA	MPGA	SPGA	NSPGA	AGA	NAGA
PGA	–	1.43676	-1.64317	-2.45049	-2.31046	-2.45049
MPGA	0.94531	–	-1.61032	-2.45049	-2.45049	-2.45049
SPGA	0.0625	0.05469	–	-2.45049	-2.31046	-2.45049
NSPGA	0.00781	0.00781	0.00391	–	-2.11289	-2.45049
AGA	0.00781	0.00391	0.00781	0.01563	–	-2.4535
NAGA	0.00391	0.00391	0.00391	0.00391	0.00391	–

3.5.2 Unified Sparse Representation Learning

Given representations of an object in different modalities (e.g., image, text, audio, etc.), to learn a unified representation of the object, has been a popular problem in the literature of multimedia retrieval. In this subsection, we present the performance of the proposed algorithm NAGA for the task of unified sparse representation learning task. Our work concerns the iterative methods for the multi-view learning, for which few popular contributions include [213, 204]. Following the similar trend, recently in [202] authors proposed a joint dictionary learning method for cross-modal retrieval task, namely Multi-modal Unified Representation Learning (MURL). They have used the ℓ_1-norm to impose the sparsity in the learned model, as well as utilized the discriminant information of objects using *cannot-link* (i.e., the pair of objects belonging to different classes) and *must-link* (i.e., pair of objects belonging to the same class) pairs among data objects representations in different domains. They have utilized accelerated gradient method [23] to find out optimal multimodal unified sparse representations.

3.5.2.1 Problem Formulation and Proposed Approach

For the problem of cross-modal retrieval, we adapt the formulation given in [202]. Let there are N objects with K representations. An object belonging to k^{th} modality is denoted as $X^k = [x_1^k, \cdots, x_N^k] \in \mathbb{R}^{d_k \times N}$, where $k = 1, \cdots, K$ and d_k is the dimensionality of k^{th} modal. Thus, an individual object $x_i = \{x_i^1, \cdots, x_i^K\}$ has K representations. The problem of cross-domain retrieval is to retrieve p top-matched objects belonging to any modality. The objective is to learn a unified representation using information from all the modalities i.e.

joint dictionary learning. Let $Z = [z_1, \cdots, z_N] \in \mathbb{R}^{P \times N}$ be the unified representation and $D^k = [d_1^k, \cdots, d_P^k] \in \mathbb{R}^{d_k \times P}$ is the dictionary corresponding to k-th modality.

Following [202], the main objective function is to solve,

$$\begin{aligned} \min_{D^k, Z} \sum_{k=1}^{K} \|X^k - D^k Z\|_F^2 + \lambda_1 \|Z\|_1 + \lambda_2 tr(ZLZ^T) \\ \|d_p^k\|_2^2 \leq 1, \forall k = 1, \cdots K, p = 1, \cdots, P, \end{aligned} \tag{3.24}$$

where, $L = D - W$, D is a diagonal matrix with $D_{ii} = \sum_j w_{ij}$, and

$$w_{ij} = \begin{cases} 1, & (x_i, x_j) \in \mathscr{S} \\ -\alpha, & (x_i, x_j) \in \mathscr{D} \\ 0, & \text{otherwise.} \end{cases} \tag{3.25}$$

Here $\mathscr{S}$ is the set of pairs of similar objects (objects belonging to same class), $tr(\cdot)$ is the trace operator and $\mathscr{D}$ is the set of pairs where the classes of objects are dissimilar. Since the problem in equation (3.24) is non-convex with both the variables, but convex with one of the variable keeping the other fixed, we solve the problem first for Z, keeping D^k fixed and vice-versa. First, we keep the dictionary fixed and optimized the following problem:

$$\min_{Z} \sum_{k=1}^{K} \|X^k - D^k Z\|_F^2 + \lambda_1 \, \|Z\|_1 + \lambda_2 \, tr(ZLZ^T). \tag{3.26}$$

Due to the ℓ_1-norm term, the above equation is non-smooth. To effectively learn the unified representation Z, we applied NAGA as described in algorithm 3. We rewrite (3.26) in terms of a loss function and regularization function as follows:

$$\min_{Z} \underbrace{\sum_{k=1}^{K} \|X^k - D^k Z\|_F^2 + \lambda_1 \, tr(ZLZ^T)}_{f(Z)} + \lambda_2 \underbrace{\|Z\|_1}_{g(Z)}.$$

The condition *converge* is considered to be achieved when the difference between the function value at the 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-5. After getting the optimal value of Z, we will solve for corresponding dictionary D^k keeping Z fixed. For this, we used the approach of Lagrange dual as used

Algorithm 3: NAGA for multimodal unified representation learning

Data: Training dataset, $c_0,\ \beta_0,\ tol$
Result: Z_{n+1}
begin
  $Z_0, Z_1 \in \mathscr{H}, \alpha_0 = 1;$
  repeat
    $n \leftarrow n+1;$
    Find c_n using backtracking stepsize rule and compute α_n and β_n such that $\alpha_n,\ \beta_n \in (0,1)$;
    $W_n \leftarrow Z_n + \alpha_n(Z_n - Z_{n-1});$
    $V_n \leftarrow (1-\beta_n)W_n + \beta_n J_{c_n}^{f,g}(W_n);$
    $Z_{n+1} \leftarrow J_{c_n}^{f,g}(V_n)$;
  until *converge*;

in [202]. We adapted the conjugate gradient method to solve for the dictionaries D^k. The initial dictionaries are obtained using k-SVD [2]. These two steps of learning the unified representation and dictionaries are repeated until we get the convergence. After learning the dictionaries, to get the new representation of a test object x_t^k belonging to the k^{th} modality, we solve the following problem:

$$\min_{z_t} \|x_t^k - D^k z_t\|_F^2 + \lambda_1 \|z_t\|_1 \tag{3.27}$$

3.5.2.2 Experiments and Result Analysis

We consider two modalities (i.e., text and image) in our cross-modal retrieval experiments. We performed our experiments on two benchmark datasets, namely Wiki [163] and NUS-WIDE [58]. Wiki dataset consists of 2866 image-text pairs with 128-dimensional visual features and 10-dimensional textual features. Here, the image-text pairs belong to ten categories. We have taken 2173 training sample pairs and 693 testing sample pairs in our experiments. The second dataset is NUS-WIDE, that consists of the tagged images from Flickr. We choose 15 categories with 600 images in each category. There are 500 visual features and 1000 textual features. For this dataset, we choose 60%-40% training-testing sample partition.

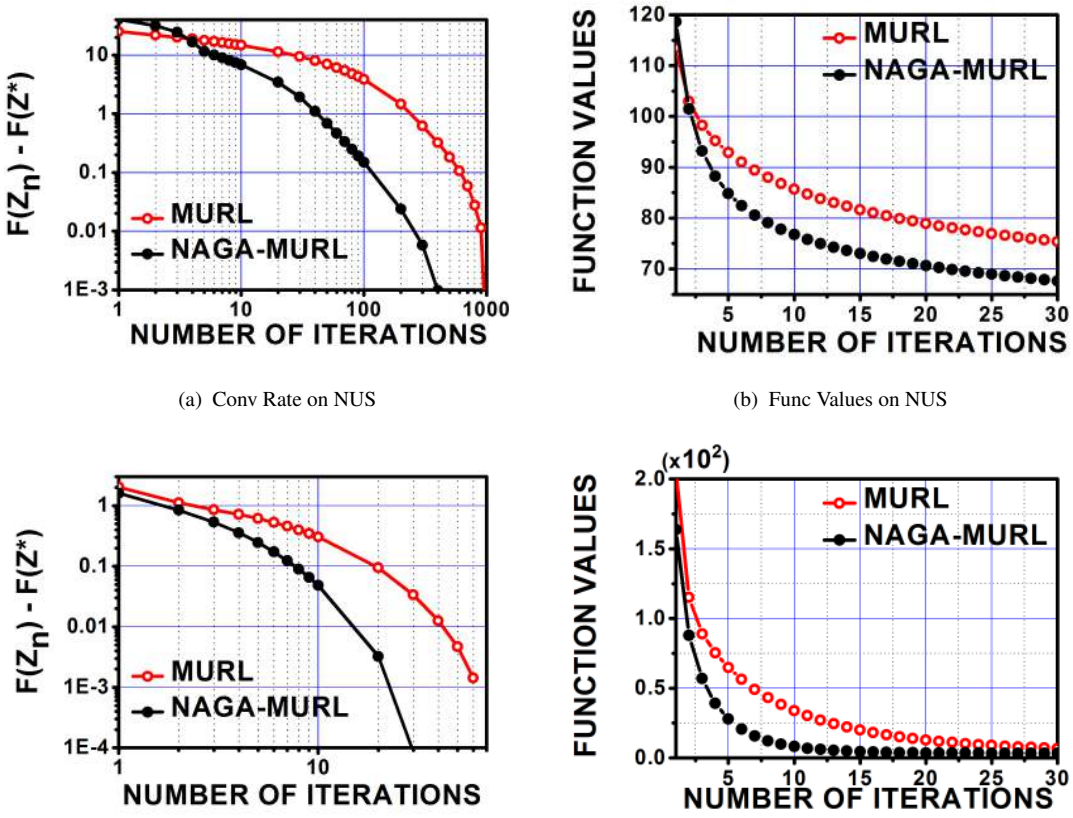
The value of sparsity controlling parameter λ_1 is set as $\{\theta \times \lambda_{max}\}$, where λ_{max} is set as $\|X^{k^T} D^k\|_\infty$ and the value of θ is chosen in the range of $\{$1-0.0001$\}$. We performed

5-fold cross-validation for tuning the parameter θ. For the stopping criteria, the tolerance value (the difference between two consecutive function values) is set to 10e-5, which also notifies the convergence. The maximum number of iterations is set to 10e4. All the vectors are initialized with a zero-valued component. Values of c_n is initialized with 1, and β_n is set to $\frac{1}{(n+1)}$. All the tests are performed on Intel Core i7 processor with 10GB RAM, under MATLAB computing environment. We have used MALSAR package [222] in our experiments.

It should be noted that it is the first time that a new proximal gradient method with extra-gradient and inertial step is applied to the task of multi-modal unified sparse representation learning. Our first result shows the main contribution of this work, which is the faster convergence of the algorithm for finding the multi-modal unified sparse representation learning. Figure 3.7 shows the results for both the datasets. In figure 3.7(a) and 3.7(c), we have shown the log-log plot between the value $F(Z_n) - F(Z^*)$ for both MURL and NAGA-MURL algorithms. Here Z_n is the value of Z at n^{th} iteration of algorithm 3 and Z^* is the optimal value. It can be easily observed that the convergence rate of NAGA-MURL is faster than that of the MURL approach. In addition, we have shown the function values $F(Z^n)$ at each iteration for both the algorithms, which directly implies that the function values decrease significantly faster with NAGA-MURL than the MURL approach.

We compared the accuracy results of our algorithm with PCA, PLS, CCA, and MURL. The standard metric to measure the performance of a cross-modal retrieval model is mean average precision (MAP), which is compared in tables 3.4 and 3.5. It can be concluded that for both the datasets the values of MAP is consistently better for the NAGA-MURL algorithm.

The comparison between the two algorithms is shown in table 3.6. It is clear from the table that the MAP values achieved by NAGA-MURL in a lesser number of iterations than that of MURL. For both the datasets, NAGA-MURL takes less than half number of iterations of MURL. The computational time is also shown in the table which shows that the NAGA-MURL approach gives better results in lesser number of iterations and less time. In the end, we demonstrate the precision-recall curves for the algorithms on both the datasets in figure 3.8. We can imply from the figure that the precision-recall curve for both the MURL and NAGA-MURL algorithms are comparable and better than that of the PCA, PLS and CCA approaches.



(a) Conv Rate on NUS

(b) Func Values on NUS

(c) Conv Rate on WIKI

(d) Func Values on WIKI

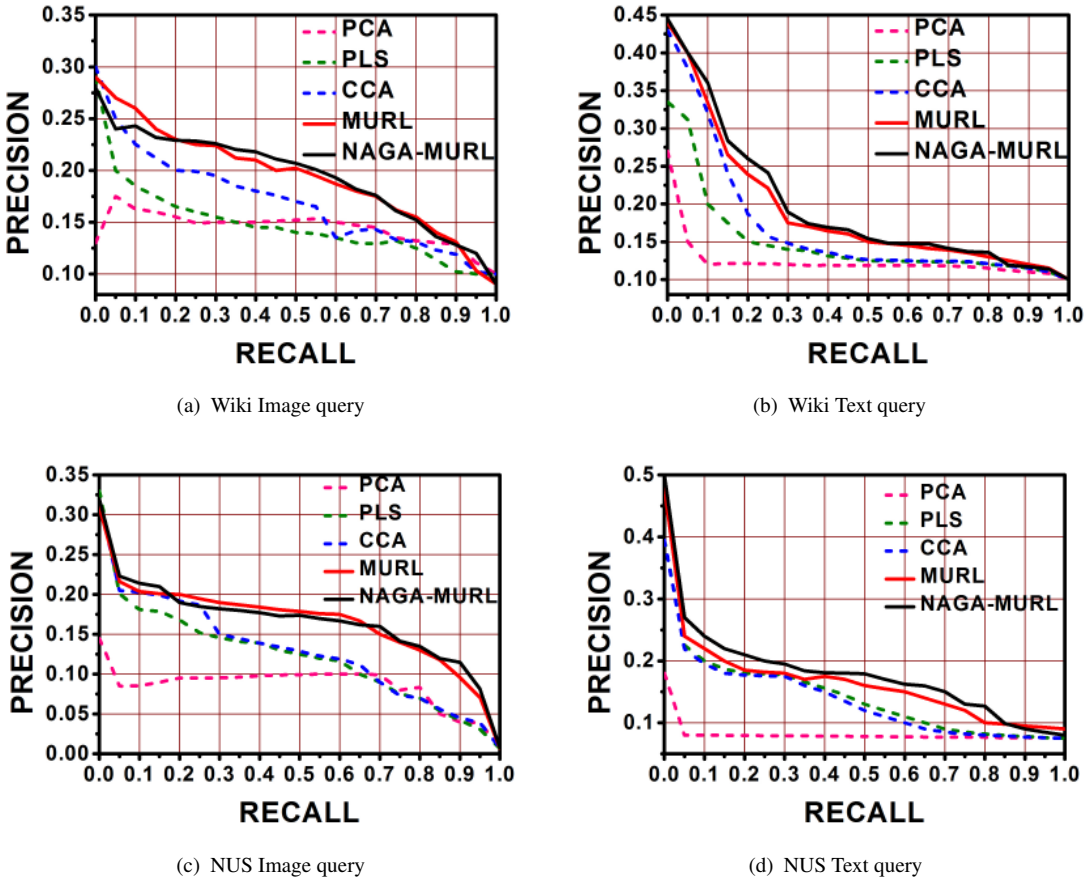
FIGURE 3.7: Performance of NAGA-MURL in comparison to MURL on both the datasets.

TABLE 3.4: Mean Avg. Precision on the Wiki Dataset

Methods	Image Sample	Text Sample	Average
PCA	0.1423	0.1103	0.1263
PLS	0.1481	0.1349	0.1415
CCA	0.1590	0.1362	0.1476
MURL	0.1894	0.1565	0.1729
NAGA-MURL	**0.1902**	**0.1596**	**0.1816**

3.5.3 Logistic Regression with Extended Lasso Frameworks

In this subsection, we will discuss the overlapped group lasso and fused lasso frameworks, the performance of the NAGA algorithm to solve such extended lasso frameworks on few real datasets, and a detailed result analysis.



(a) Wiki Image query

(b) Wiki Text query

(c) NUS Image query

(d) NUS Text query

FIGURE 3.8: Precision-recall curves on both the datasets for the two cross-modal retrieval tasks.

TABLE 3.5: Mean Avg. Precision on the NUS-WIDE Dataset

Methods	Image Sample	Text Sample	Average
PCA	0.0912	0.0794	0.853
PLS	0.1352	0.1181	0.1267
CCA	0.1126	0.1065	0.1096
MURL	0.1653	0.1394	0.1524
NAGA-MURL	**0.1697**	**0.1402**	**0.1549**

3.5.3.1 Overlapping group Lasso

The regularized convex loss minimization framework is given as follows,

$$\min_{x\in\mathbb{R}^d} \; f_{log}(x) + \phi_{\rho_2}^{\rho_1}(x), \tag{3.28}$$

TABLE 3.6: Number of Iterations and computational time to converge for both Datasets

WIKI		
Methods	# of Iterations	computational time (in seconds)
MURL	99	2.7522
NAGA-MURL	**31**	**1.612**
NUS-WIDE		
Methods	# of Iterations	computational time (in seconds)
MURL	1957	127.7921
NAGA-MURL	**465**	**62.7285**

where, the second term is the regularizer for the overlapping group structure defined as,

$$\phi_{\rho_2}^{\rho_1}(x) = \rho_1 \|x\|_1 + \rho_2 \sum_{i=1}^{g} w_i \|x_{G_i}\|_2. \tag{3.29}$$

Here, $\rho_1 \geq 0$ and $\rho_2 \geq 0$ are regularization parameters, $w_i > 0$, $i = 1,2,\cdots g$, $G_i \in 1,2,\cdots,p$.

The proximal operator corresponding to the overlapping group regularizer is computed by solving the following,

$$\text{prox}_{\phi_{\rho_2}^{\rho_1}}(v) = \underset{x\in\mathbb{R}^d}{\text{argmin}}\{\frac{1}{2}\|x-v\|^2 + \phi_{\rho_2}^{\rho_1}(x)\} \tag{3.30}$$

To solve the above equation, there are several contributions available in the literature, few of them are [216, 103, 162]. We followed the work of [216] for computing the proximal operator of the overlapping group regularizer.

3.5.3.2 Fused Lasso

The regularized convex loss minimization framework is given as follows,

$$\min_{x\in\mathbb{R}^d} f_{log}(x) + \phi_{\lambda_2}^{\lambda_1}(x), \tag{3.31}$$

where, the second term is the regularizer for the ordering structure defined as,

$$\phi_{\lambda_2}^{\lambda_1}(x) = \lambda_1 \|x\|_1 + \lambda_2 \sum_{i=1}^{d-1} |x_i - x_{i+1}|. \tag{3.32}$$

Algorithm 4: NAGEL: NAGA for Extended Lasso

Data: $\rho_1,\ \rho_2$ or $\lambda_1,\ \lambda_2,\ tol$
Result: x_{n+1}
begin
  $x_0 = x_1 \in \mathbb{R}^d, L_1 = 1, \beta_1 = 1, n = 0;$
  repeat
    Find L_n using backtracking line search [23] and compute α_n and β_n such that $\alpha_n,\ \beta_n \in (0,1)$;
    $n \leftarrow n+1;$
    $y_n \leftarrow x_n + \alpha_n(x_n - x_{n-1});$
    $u_n \leftarrow \text{prox}_{\phi_{L_n}}(y_n - 1/L_n(A^TAy_n - A^Tb));$
    $z_n \leftarrow (1-\beta_n)y_n + \beta_n u_n;$
    $x_{n+1} \leftarrow \text{prox}_{\phi_{L_n}}(z_n - 1/L_n(A^TAz_n - A^Tb));$
  until *converge*;

The proximal operator corresponding to the fused penalty is computed by solving the following,

$$\text{prox}_{\phi^{\lambda_1}_{\lambda_2}}(v) = \underset{x\in\mathbb{R}^d}{\text{argmin}}\{\frac{1}{2}\|x-v\|^2 + \phi^{\lambda_1}_{\lambda_2}(x)\} \tag{3.33}$$

To compute the proximal operator corresponding to the fused lasso penalty, we adapted the sparse learning with efficient projection [122].

To solve these problems, we used the proposed NAGA algorithm. The pseudo-code of the algorithm is given in 4. Let us consider the Lipschitz constant of the gradient of the smooth loss function is L. To guarantee the convergence of the algorithm, the value of $Ł_n$ should belong to $(0, 2/L]$. For the computation of the term α_n, we follow the work of [47]. However, other definitions such as [138, 23] are also applicable. It should be noted that the term ϕ_{L_n} corresponds to the general notation for the extended lasso penalties. For the overlapping group penalty, ϕ_{L_n} will be $\phi^{\rho_1/L_n}_{\rho_2/L_n}$, whereas for fused lasso penalty, it will be equal to $\phi^{\lambda_1/L_n}_{\lambda_2/L_n}$.

The condition *converge* is considered to be achieved when the difference between the function value at the 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-5.

3.5.3.3 Experimental Results

The dataset we used for analyzing the performance of our algorithm for the logistic regression with overlapped group penalty is the benchmark breast cancer gene-expression dataset, which consists of 8,141 genes in 295 breast cancer tumors (78 metastatic and 217 non-metastatic). For groups, canonical pathways from Molecular Signatures Database (MSigDB) [186] are used, that contains 639 groups of genes. We have used 635 groups of genes in our experiments. We randomly selected 70% of data samples as the training set, and the rest of samples are considered as the test set. We compared the Proximal Gradient Algorithm PGA, Normal S-iteration based Proximal Gradient Algorithm NPGA, Accelerated Gradient Algorithm AGA, Subgradient descent SGD and the proposed NAGEL. 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 1.2TB 15K hot plug SAS.

The values of sparsity controlling parameters ρ_1, ρ_2, λ_1 and λ_2 are set as $\{\theta \times \theta_{max}\}$, where θ_{max} is set as $\|X^T Y\|_\infty$. The parameter θ is tuned in the range $\{0.0001, 1\}$ with an increment of 0.001. With X as the $m \times d$ data matrix with m number of instances with d-dimensional vectors, Y as the $m \times 1$ label vector and w as the $d \times 1$ learning parameter vector, the logistic loss is given as,

$$f(x) = w^T x,$$
$$f_{log}(w) = \frac{1}{m} \sum_{i=1}^{m} \log(1 + e^{-y_i \cdot f(x_i)}).$$

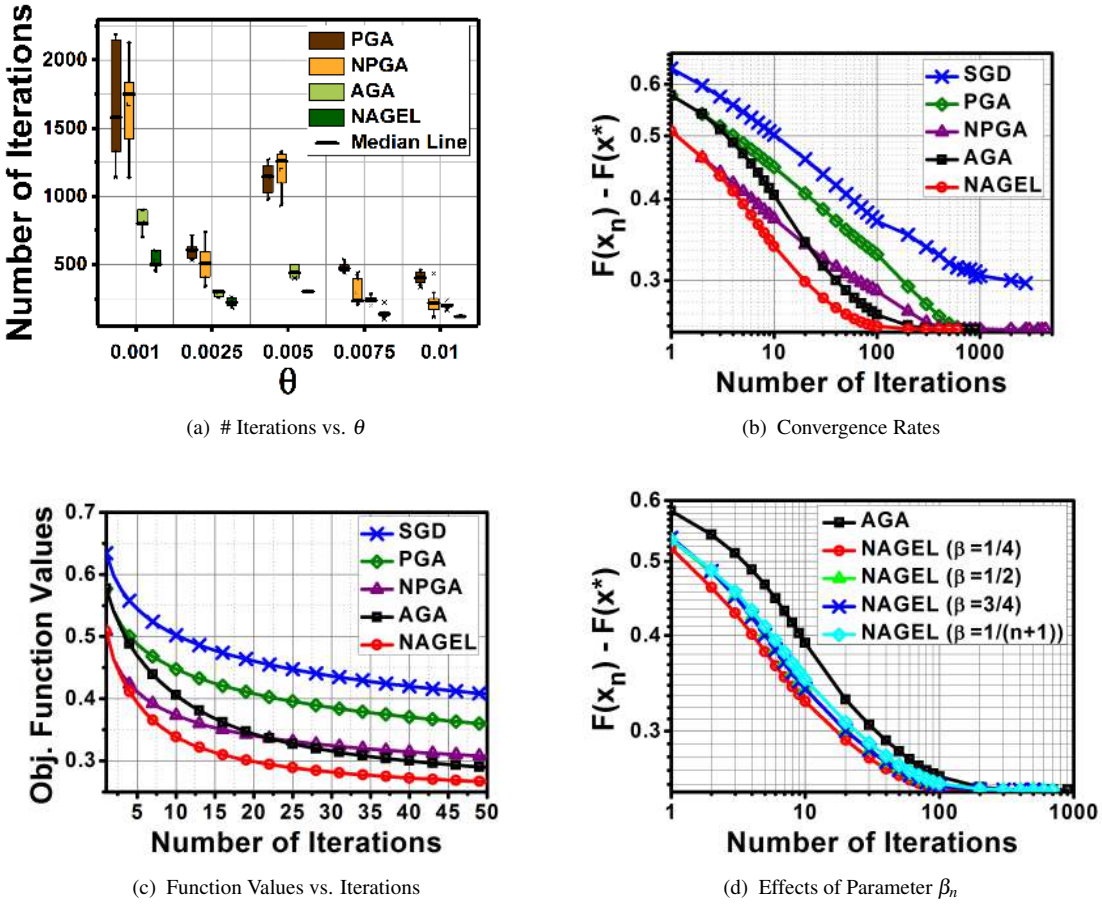
In the first experiment, we show the plot for the number of iterations with respect to changing values of θ for different training sets involved in figure 3.9(a). For a specific value of θ, we performed ten experiments with randomly selected training samples, and the plot shows the box-plots corresponding to resulting statistics. We only compare the proximal methods in this experiment. It is clear from the results that the proposed NAGEL algorithm outperforms other proximal methods in terms of the number of iterations. The variations in the number of iterations to reach the convergence for higher values of θ is larger than the lower values of θ. Also, the number of iterations in case of the NAGEL algorithm is stable in comparison to the other algorithms.

We check the convergence rate of the algorithms by plotting the log-log graphs between $F(x_n) - F(x^*)$ and the number of iterations in figure 3.9(b). From here onwards the value of θ is set to 0.005 for all the experiments of logistic regression with overlapping group lasso problem. It is evident that the convergence rate of NAGEL again outperforms other algorithms. The convergence of the subgradient descent algorithm is found to be very slow in comparison to the proximal gradient algorithms. In figure 3.9(c) we give the reduction in function values in each iteration, where it is shown that at each iteration the function value by the NAGEL algorithm is least in comparison to other algorithms.

There are few hyper-parameters on which the convergence shown in the experiments depends. The first parameter is α_n at n^{th} iteration. It is already known from [23, 47] that the definitions of α_n used in these works satisfy the condition of convergence. We set the parameter α_n as in [47]. The second parameter is β_n, belongs to set $(0,1]$. In our final experimental result figure (3.9(d)), we have shown the plot for NAGEL with both constant and variable values of β_n as (a) $1/4$, (b) $1/2$, (c) $3/4$ and (d) $1/(n+1)$, where n is the iteration number. It is clear from the figures that for all values of $\beta \in (0,1]$, we get better convergence rates than AGA.

We measured the classification performance using standard metrics that are precision, recall and the classification accuracy. The 70%-30% random training-test set partition is used in the experiments with 10 fold cross validation. The detailed results are shown in table 3.7 in terms of number of iterations (**# ITER**), accuracy (**ACC**), precision (**PRE**), recall (**REC**) and CPU time (**TIME**). It can be observed from the table that the classification performances of all the proximal algorithms are same in terms of precision, recall and accuracy, which are altogether better than that of the subgradient descent algorithm. As far as the number of iterations and the CPU time concerned, NAGEL takes the least number of iterations. It has been observed that due to its design, the per iteration time for NAGEL is a bit higher than that of state-of-the-art algorithms, however, the reduced number of iterations overcomes this problem, which can be directly concluded from the shown CPU time results from the table 3.7. Since CPU time related issues can be handled with tricky implementation specifics, an interesting open problem is to design operators that can be computed in lesser CPU time.

For logistic loss with the fused lasso penalty, we consider two cancer genome datasets, namely, Leukemia and Prostate cancer datasets. The details are as follows:



(a) # Iterations vs. θ

(b) Convergence Rates

(c) Function Values vs. Iterations

(d) Effects of Parameter β_n

FIGURE 3.9: Performance of algorithms for the overlapping group lasso problem with the Breast cancer dataset. 3.9(a) shows the variation in iterations at different values of θ for ten randomly selected training samples. For all the other experiments we set $\theta = 0.005$. 3.9(b) is the log-log plot showing the convergence rate of the algorithms, which is considered to achieve when the $F(x_n) - F(x_{n-1}) < \varepsilon$ with $\varepsilon = 10E-6$. 3.9(d) shows the convergence rates of NAGEL for different values of parameter β_n.

- **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.

- **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, which are all acute leukemia patients, either acute lymphoblastic leukemia (ALL) or acute myelogenous leukemia (AML).

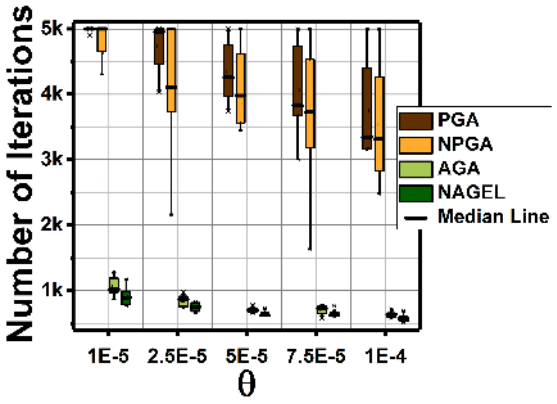
TABLE 3.7: Detailed Results for Overlapping group Lasso with Breast Cancer Dataset. All the shown Results are the average of multiple experiments with randomly partitioned training-testing datasets for $\theta = 0.01$. For the NAGEL algorithm, the subscripted values are the values set for $\beta_n = \beta$ at each iteration. Shown CPU time are in seconds.

	SGD	**PGA**	**NPGA**	**AGA**	**NAGEL**$_{\beta=1/2}$	**NAGEL**$_{\beta=1/4}$	**NAGEL**$_{\beta=3/4}$	**NAGEL**$_{\beta=1/(n+1)}$
# ITER	628.2 (±49.9719)	445.2 (±31.5706)	229.6 (±30.7376)	231.8 (±17.4843)	**140.6** (±12.2801)	141.8 (±09.8590)	170.2 (±20.8494)	143.0 (±12.1037)
PRE	0.4254 (±0.0293)	**0.5151** (±0.0936)	**0.5151** (±0.0936)	**0.5151** (±0.0936)	**0.5151** (±0.0936)	**0.5151** (±0.0936)	**0.5151** (±0.0936)	**0.5151** (±0.0936)
REC	0.4301 (±0.2440)	**0.6405** (±0.3409)	**0.6405** (±0.3409)	**0.6405** (±0.3409)	**0.6405** (±0.3409)	**0.6405** (±0.3409)	**0.6405** (±0.3409)	**0.6405** (±0.3409)
ACC	0.6706 (±0.0343)	**0.7129** (±0.0257)	**0.7129** (±0.0257)	**0.7129** (±0.0257)	**0.7129** (±0.0257)	**0.7129** (±0.0257)	**0.7129** (±0.0257)	**0.7129** (±0.0257)
TIME	0.9507 (±0.0250)	**0.8449** (±0.0835)	1.8302 (±0.1719)	1.4796 (±0.1789)	2.8296 (±0.2764)	0.8817 (±0.0635)	1.0531 (±0.1365)	0.8809 (±0.0377)

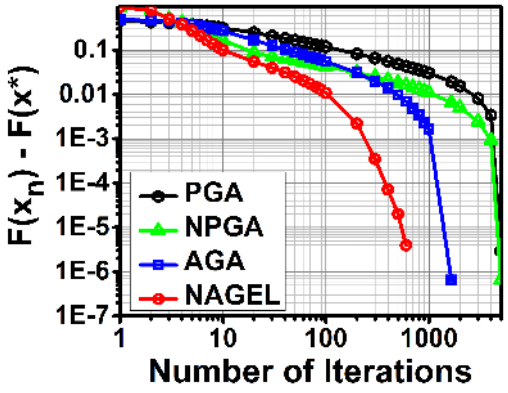
Here again we randomly sampled 70% of data samples as the training set and the rest of samples are considered as the test set. We repeat each experiment ten times and present the results in figures (3.10) and (3.11) and tables 3.8 and 3.9 for the Prostate and Leukemia datasets respectively. In our first experiment, we plot the box-plots for the number of iterations attained by each algorithm to reach the convergence for the ten randomly sampled training-test samples at different values of θs. It is found that in most of the experiments, the PGA and NPGA algorithms converge near to the maximum number of iterations. For the two accelerated algorithms, the number of iterations is much lesser than the non-accelerated counterparts. Moreover, the number of iterations for NAGEL is lesser than AGA. We are not showing the performance of the subgradient descent, since the time consumption of this algorithm is very high for the fused lasso problem.

From here onwards the value of θ is set to 5E-5 for all the experiments on the Prostate dataset and 10E-6 for Leukemia dataset for the problem of logistic regression with fused lasso penalty. The results in figure 3.10 with the Prostate dataset demonstrate the efficacy in using the NAGEL algorithm over the other proximal algorithms. With the Prostate dataset, the convergence rates of AGA and NAGEL$_{\beta=1/4}$ is almost similar. It has been seen that with NAGEL$_{\beta=1/4}$ the functional values computed at initial iterations are very large. Similar behaviour can be noticed in results with Leukemia dataset in figure 3.11.

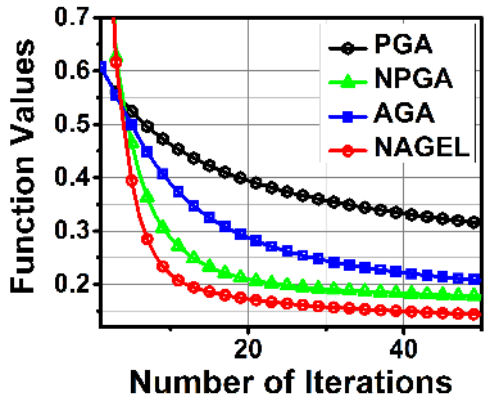
The classification performances of the algorithms with the Prostate dataset is shown in table 3.8. The best values of the precision and accuracy measures are achieved by the PGA method, however the number of iterations by this algorithm is very high. NAGEL$_{\beta=1/4}$ results in the best recall value achieved. The number of iterations needed to reached the convergence is least for the NAGEL, however, the CPU time is a bit higher than the AGA



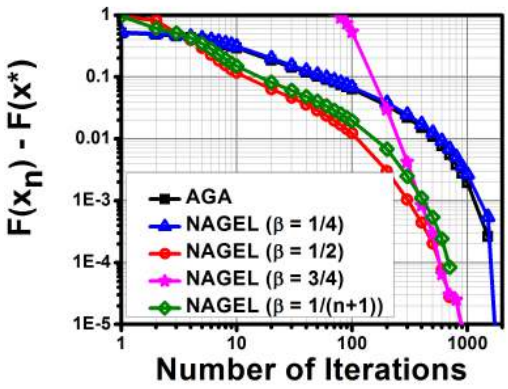
(a) # Iterations vs. θ



(b) Convergence Rates



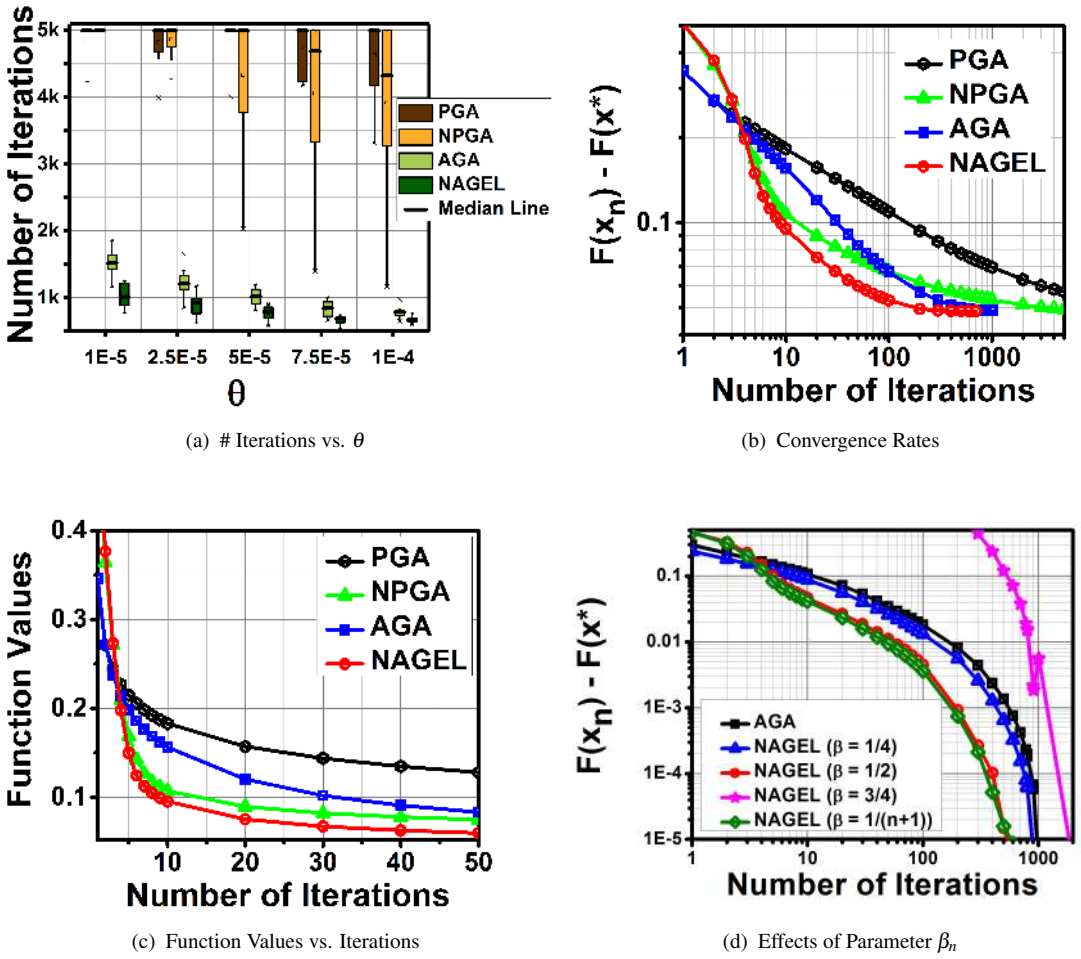
(c) Function Values vs. Iterations



(d) Effects of Parameter β_n

FIGURE 3.10: Performance of algorithms for the Prostate Dataset. 3.10(a) shows the variation in iterations at different values of θ for ten randomly selected training samples. For all the other experiments we set $\theta = 5E-5$. 3.10(b) is the log-log plot showing the convergence rate of the algorithms, which is considered to achieve when the $F(x_n) - F(x_{n-1}) < \varepsilon$ with $\varepsilon = 10E-6$. 3.10(d) shows the convergence rates of NAGEL for different values of parameter β_n.

algorithm. With the Leukemia dataset, the performance of the NAGEL is superior than the other algorithms as shown in table 3.9, again except the CPU time. This issue can be handled by designing new faster operators, which is the future research direction of our work.



(a) # Iterations vs. θ

(b) Convergence Rates

(c) Function Values vs. Iterations

(d) Effects of Parameter β_n

FIGURE 3.11: Performance of algorithms for the Leukemia Dataset. 3.11(a) shows the variation in iterations at different values of θ for ten randomly selected training samples. For all the other experiments we set $\theta = 7.5E-5$. 3.11(b) is the log-log plot showing the convergence rate of the algorithms, which is considered to achieve when the $F(x_n) - F(x_{n-1}) < \varepsilon$ with $\varepsilon = 10E-6$. 3.11(d) shows the convergence rates of NAGEL for different values of parameter β_n.

3.6 Conclusions

A new extra-gradient based accelerated gradient algorithm is proposed and analyzed in this chapter. The problem under consideration is a nonsmooth-convex minimization problem. The convergence of the proposed method is proved for two cases. Firstly, the convergence is proved for the contraction mappings, and then for the more general non-expansive mappings. In the process of proving the convergence of the algorithm, we also presented the boundedness property of the proposed algorithm under specific conditions. In the end,

TABLE 3.8: Detailed Results for Fused Lasso with Prostate Cancer Dataset. All the shown Results are the average of multiple experiments with randomly partitioned training-testing datasets for $\theta = 1E-5$. For the NAGEL algorithm, the subscripted values are the values set for β_n at each iteration. Shown CPU time are in seconds.

	AGA	**NAGEL**$_{\beta=1/2}$	**PGA**	**NPGA**	**NAGEL**$_{\beta=1/4}$	**NAGEL**$_{\beta=3/4}$	**NAGEL**$_{\beta=1/(n+1)}$
# ITER	1746.2 (±423.6829)	**919.2** (±153.0349)	5000.0 (±00.0000)	5000.0 (±00.0000)	1446.4 (±323.4691)	1140.2 (±86.5141)	1168.4 (±154.7152)
PRE	0.9300 (±0.0647)	0.9350 (±0.0602)	**0.9500** (±0.0685)	**0.9500** (±0.0685)	0.9300 (±0.0647)	0.8850 (±0.0137)	0.9346 (±0.0609)
REC	0.8964 (±0.1059)	0.9050 (±0.1037)	0.9242 (±0.0701)	0.9242 (±0.0701)	**0.9250** (±0.1118)	0.9300 (±0.0647)	0.8742 (±0.0891)
ACC	0.7727 (±0.1245)	0.7878 (±0.1016)	**0.8485** (±0.0634)	0.8182 (±0.0634)	0.7727 (±0.1245)	0.7727 (±0.0829)	0.7576 (±0.1071)
TIME	**1.9969** (±0.3266)	2.2066 (±0.1278)	4.5937 (±0.0382)	7.6840 (±0.3742)	2.9514 (±0.4413)	2.8170 (±0.4848)	2.8789 (±0.4821)

TABLE 3.9: Detailed Results for Fused Lasso with Leukemia Cancer Dataset. All the shown Results are the average of multiple experiments with randomly partitioned training-testing datasets for $\theta = 5E-5$. For the NAGEL algorithm, the subscripted values are the values set for β_n at each iteration. Shown CPU time are in seconds.

	AGA	**NAGEL**$_{\beta=1/2}$	**PGA**	**NPGA**	**NAGEL**$_{\beta=1/4}$	**NAGEL**$_{\beta=3/4}$	**NAGEL**$_{\beta=1/(n+1)}$
# ITER	1108.6 (±109.1274)	**906.0** (±84.3059)	5000.0 (±00.0000)	5000.0 (±00.0000)	1079.6 (±199.9032)	1078.6 (±130.7719)	1352.2 (±595.9381)
PRE	0.9597 (±0.0382)	**0.9764** (±0.0324)	0.9706 (±0.0656)	0.9597 (±0.0381)	0.9597 (±0.0381)	0.9763 (±0.0324)	0.9763 (±0.0324)
REC	0.9857 (±0.0319)	**0.9875** (±0.0449)	0.9473 (±0.0557)	0.9857 (±0.0319)	0.9857 (±0.0319)	0.9675 (±0.0449)	0.9675 (±0.0449)
ACC	0.8703 (±0.0687)	**0.9073** (±0.0529)	0.8518 (±0.0841)	0.8518 (±0.0841)	0.8981 (±0.0529)	0.8888 (±0.0388)	0.8888 (±0.0388)
TIME	**1.4511** (±0.2322)	2.0079 (±0.1299)	4.6714 (±0.1015)	7.9592 (±0.4328)	2.5143 (±0.5326)	2.5062 (±0.3530)	3.2674 (±1.7012)

we analyzed the stability property of the algorithm. It has been shown that under few particular conditions, the algorithm is stable, with respect to the contraction mappings. In order to show the practical performance of the algorithm, we compared it with traditional proximal and accelerated gradient algorithms. Experiments are performed with several high dimensional publicly available real datasets that belong to different domains such as image processing and bio-informatics. In our first application, we used lasso framework for single-task regression or classification problems and applied the proposed solution strategy to nine high-dimensional datasets. Secondly, we applied the algorithm to the task of joint subspace learning for cross-modal datasets. In the third application, the proposed algorithm is applied to solve extended lasso problems with overlapping group and fused penalties. From the achieved results, it can be concluded that the proposed algorithm gives better accuracy in significantly lesser number of iterations than the traditional methods.