AN ITERATIVE L_1 REGULARIZED LIMITED MEMORY STOCHASTIC BFGS ALGORITHM AND NUMERICAL EXPERIMENTS FOR BIG DATA APPLICATIONS

By

VANDAN PATEL

Bachelor of Engineering

University of Mumbai

Mumbai, India

2014

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2017

AN ITERATIVE L_1 REGULARIZED LIMITED

MEMORY STOCHASTIC BFGS ALGORITHM AND

NUMERICAL EXPERIMENTS FOR BIG DATA

APPLICATIONS

Thesis Approved:

Dr. Farzad Yousefian

Thesis Adviser

Dr. Sunderesh Heragu

Dr. Chaoyue Zhao

ACKNOWLEDGEMENTS

First of all, I am grateful to the god for providing courage and energy to complete my thesis. I would like to thank my mom, dad and brother for their trust along with my wife Rucha for constant motivation and encouragement without which this work was impossible to be done.

I wish to express my sincere gratitude to my advisor Dr. Yousefian for believing in me and guiding me during all the good and hard times. Sir, you are the reason I did not quit on this journey, thank you so much for your support. I am also thankful to the Oklahoma State University and the Industrial Engineering and Management department for providing an opportunity to gain research experience along with an excellent course work. In addition, thank to my committee members, Dr. Heragu and Dr. Zhao, for their time and valuable feedbacks on my work.

I would also mention special gratitude to my elementary school art teacher Mr. Khotre who instilled the quality of persistence and importance of devotion in me, which helps me to be creative in all aspects of my life. I am also thankful to my friend Malav who was the first person to inspire and motivate me to pursue my masters with thesis. Thanks to my friends Vishal, Jay and all others who directly or indirectly supported me throughout my masters.

Name: VANDAN PATEL

Date of Degree: JULY, 2017

Title of Study: AN ITERATIVE L_1 REGULARIZED LIMITED MEMORY STOCHASTIC BFGS ALGORITHM AND NUMERICAL EXPERIMENTS FOR BIG DATA APPLICATIONS

Major Field: INDUSTRIAL ENGINEERING & MANAGEMENT

Abstract: This research is motivated by challenges in addressing optimization models arising in Big Data. Such models are often formulated as large scale stochastic optimization problems. When the probability distribution of the data is unknown, the Sample Average Approximation (SAA) scheme can be employed which results in an Empirical Risk Minimization (EMR) problem. To address this class of problems deterministic solution methods, such as the Broyden, Fletcher, Goldfarb, Shanno (BFGS) method, face high computational cost per iteration and memory requirement issues due to presence of uncertainty and high dimensionality of the solution space. To cope with these challenges, stochastic methods with limited memory variants have been developed recently. However, the solutions generated by such methods might be dense requiring high memory capacity. To generate sparse solutions, in the literature, standard L_1 regularization technique is employed, where the term $\lambda_1 \|\bullet\|_1$ is added to the objective function of the problem. Here, λ_1 is called the L_1 regularization parameter and $\|\bullet\|_1$ denotes L_1 norm. Under this approach, addition of constant λ_1 changes the original problem and the solutions obtained by solving the regularized problem are approximate solutions. Moreover, limited information is available in the literature to obtain sparse solutions to the original problem. To address this gap, in this research we develop an iterative L_1 Regularized Limited memory Stochastic BFGS (iRLS-BFGS) method in which the L_1 regularization parameter and the step-size parameter are simultaneously updated at each iteration. Our goal is to find the suitable decay rates for these two sequences in our algorithm. To address this research question, we first implement the iRLS-BFGS algorithm on a Big Data text classification problem and provide a detailed numerical comparison of the performance of the developed algorithm under different choices of the update rules. Our numerical experiments imply that when both the step-size and the L_1 regularization parameter decay at the rate of the order $\frac{1}{\sqrt{k}}$, the best convergence is achieved. Later, to support our findings, we apply our method to address a large scale image deblurring problem arising in signal processing using the update rule from the previous application. As a result, we obtain much clear deblurred images compared to the classical algorithm's deblurred output images when both the step-size and the L_1 regularization parameter decay at the rate of the order $\frac{1}{\sqrt{k}}$.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Optimization algorithms are widely used to extract knowledge from the data in machine learning. The usage is increasing exponentially relative to the technological advancement in generating, storing and retrieving high volume of data. This research focuses on unconstrained optimization problems arising in machine learning which are given by,

$$\min_{w \in \mathbb{R}^n} F(w) := E[f(w,\xi)], \tag{1}$$

where $f : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ is a loss function, $w \in \mathbb{R}^n$ is the decision variable and $\xi \in \mathbb{R}^d$ is a random variable. In Big Data applications, the function f is given as a loss function $f(w,\xi) = l(h(w;x),z)$, where $\xi = (x,z)$ is a random variable, $x \in \mathbb{R}^n$ is the input vector, and $z \in \mathbb{R}$ is the true output vector. In addition, *F* in equation (1) can be seen as

$$E[f(w,\xi)] = \int f(w,\xi)P(\xi), \tag{2}$$

where P is the probability distribution of ξ. Under this setting, problem (1) is known as the 'Expected Risk Minimization problem'. The loss function, *l(h;z)* is a real-valued convex function defined to find the distance of the true output *z* from the prediction function output *h(w,x)* which is parametrized by *w* and *x* is the input data. Moreover, in machine learning, *l* could be logistic loss function, support vector machine (SVM) loss function, least square loss function, or etc., depending upon the application.

However, two major challenges arise while optimizing the objective function over large scale data: uncertainty (random variables) and high dimensionality of the decision space. First, when the dimensions of the random variable is high, $d > 5$, the evaluation of the multivariate integration to get expectation over random variable as seen in relation (2) becomes difficult to compute. Moreover, it is only possible to evaluate equation (2) when we have complete information of the probability distribution *P*, which is usually not the case. Therefore, to formulate the objective function of the problem in the absence of the distribution information, Monte Carlo simulation methods such as Sample Average Approximation (SAA) scheme have been widely used in the literature (Kleywegt et al., 2002 and Shapiro et al., 2003). This scheme results into an alternative formulation known as 'Empirical Risk Minimization' (ERM) problem which is an approximation of the expected risk minimization problem and is given by,

$$\min_{w \in \mathbb{R}^n} F(w) := \frac{1}{N} \sum_{i=1}^{N} f(w, \xi_i) , \tag{3}$$

where *N* is the sample size of the set of independent random input-output pairs and ξ_i is the *i*th sample of the random variable used to train the model. However, when *N* is large, application of deterministic solution methods becomes very challenging and inefficient.

The second challenge arise due to high dimensionality of the solution space. In machine learning applications *n* is the number of features, which is often large in Big Data applications. In such application, to compute solutions and store the computed solutions as the algorithm proceeds we require high storage memory. Insufficient memory results in the termination of the algorithm making it computationally inefficient and infeasible for Big Data applications.

1.1 Deterministic solution methods and challenges in machine learning

In this section we consider advantages and disadvantages of the deterministic methods in solving the unconstrained optimization applications in machine learning.

1.1.1 Deterministic gradient method (DG): The SAA scheme is employed in the deterministic gradient method also known as the 'batch' gradient method which minimizes *F* given by equation (3) by updating the vector $w \in \mathbb{R}^n$ as follows,

$$w_k := w_{k-1} - \alpha_k \nabla F(w_{k-1}), \qquad (4)$$

where α_k is the step size parameter, and the gradient at each iteration, *k*= 1, 2,…, *t*, (where *t* is the maximum number of iterations specified) is computed as,

$\nabla F(w_{k-1}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f(w_{k-1}, \xi_i)$, where $\xi_i = \{(x_{i,} z_i)\}_{i=1}^{N}$ is the *i*th input-output vector. Note that at each iteration, the gradient is evaluated using all the *N* values from the sample size. Although, we obtain a better update by using all the data points, the method has a high computation cost per iteration and may take an unreasonable time to converge. Especially, when the data is large scale with number of instances in billions or more, then the DG method becomes impractical.

1.1.2 Deterministic Newton method (DN): The Deterministic Newton method can be used to solve problem (3) in a similar fashion with additional use of second order information in the update rule given by,

$$w_k := w_{k-1} - \alpha_k B_{k-1}^{-1} \nabla F(w_{k-1}), \tag{5}$$

where α_k is the step-size parameter, *k= 1, 2,..., t* and $B_{k-1} = \nabla^2 F(w_{k-1})$ is the symmetric Hessian matrix of the objective function with the dimensions $n \times n$, where n is the dimensionality of the decision variable *w*. The gradient is evaluated as,

$\nabla F(w_{k-1}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f(w_{k-1}, \xi_i)$ and the Hessian by, $\nabla^2 F(w_{k-1}) = \frac{1}{N}\sum_{i=1}^{N} \nabla^2 f(w_{k-1}, \xi_i)$. The advantage of this method is that it has a faster convergence rate than that of the DG method. However, with high dimensionality of *n* in large-scale datasets, the memory requirements to store the Hessian becomes very high and makes the computation infeasible. Moreover, the calculation of the inverse Hessian at each iteration becomes inefficient due to high cost of computation when the data is large scale.

1.1.3 Deterministic Broyden, Fletcher, Goldfarb, and Shanno method (D-BFGS): In 1950, W.C. Davison, developed an update formula to solve large scale optimization problems by approximating the Hessian and the inverse Hessian instead of the actual computation. This led to creation of a new class of optimization methods known as the

‘Quasi-Newton methods’ (QN). Later, Fletcher and Powell demonstrated that the developed update rule was much faster and reliable (Nocedal and Wright, Chapter 6, 1999), and then various other quasi-Newton methods were developed since then. The D-BFGS algorithm is one of the most popular quasi-Newton algorithm in which instead of re-computing the Hessian and the inverse Hessian from scratch every time to be used in the update rule, the algorithm provides a much simpler approximation formula. To solve problem (3) the following update rule is used, $w_k := w_{k-1} - \alpha_k B_k^{-1} \nabla F(w_{k-1})$, where, α_k is the step size parameter, *k= 1, 2,..., t* , $B = \nabla^2 F(w)$, is the symmetric $n \times n$ Hessian matrix and the gradient is given by, $\nabla F(w_{k-1}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f(w_{k-1}, \xi_i)$. Note that the update rules in the DN and the D-BFGS method are the same, however, in the D-BFGS method new Hessian B_k is approximated from the previous Hessian approximate B_{k-1} using the following formula,

$$B_k = B_{k-1} - \frac{B_{k-1} s_{k-1} s_{k-1}^T B_{k-1}}{s_{k-1}^T B_{k-1} s_{k-1}} + \frac{y_{k-1} y_{k-1}^T}{y_{k-1}^T s_{k-1}}, \tag{6}$$

where $s_k = w_k - w_{k-1}$ is called the displacement factor and $y_k = \nabla F(w_k) - \nabla F(w_{k-1})$ is called the gradient mapping factor. Also, note that, $H_k = B_k^{-1}$, the inverse of the Hessian matrix is also approximated at each iteration unlike the DN method. New Hessian inverse, H_k is approximated iteratively from the previous approximation, H_{k-1} using the following formula,

$$H_k = (I - \rho_{k-1} s_{k-1} y_{k-1}^T) H_{k-1} (I - \rho_{k-1} y_{k-1} s_{k-1}^T) + \rho_{k-1} s_{k-1} s_{k-1}^T, \tag{7}$$

where $\rho_{k-1} = \dfrac{1}{y_{k-1}^T s_{k-1}}$. The D-BFGS update works well as long as the curvature condition $y_{k-1}^T s_{k-1} > 0$ is satisfied.

Note that these formulae only use most recent observed information of the objective function and the curvature steps (s_{k-1}, y_{k-1}) to approximate the new Hessian (B_k **)** or the inverse Hessian (H_k). The algorithm is robust and has a super-linear rate of convergence which is faster than the Deterministic Gradient method but slower than the Newton method. Also, it is known that the D-BFGS has much better self-correcting property than its competitor Davison-Fletcher-Powell (DFP) update under certain conditions, that is if the matrix H_k estimated incorrect curvature in the objective function which slows down the convergence, then the approximation formula corrects itself with in a few iterations. However, when the data to be learnt is large scale, the D-BFGS method requires lot of memory as it stores all the computed Hessian or the inverse Hessian ($n \times n$) information and makes the algorithm infeasible to use when enough computational memory is not available (Liu & Nocedal, 1989). Moreover, the computation of the deterministic gradient incurs high computational cost and makes the algorithm infeasible to use when number of samples (*N)* is high. This challenge is addressed by the following method.

1.1.4 Limited memory Deterministic BFGS update (LD-BFGS): The D-BFGS method approximates the new inverse Hessian H_k from the previous inverse Hessian H_{k-1} and curvature pairs (s_{k-1}, y_{k-1}). Throughout the execution the LD-BFGS method stores all the obtained inverse Hessians and the curvature information which is generally dense and incurs high cost of storing the matrices each iteration. This prohibits the use of it when the dimensions of decision variable, *n* is large. The high memory requirement drawback of the D-BFGS method is addressed by the limited memory variant of the D-BFGS update developed by Nocedal in 1990 called LD-BFGS.

To address the memory issues in solving problem (3), the same update rule is used which is given by $w_k := w_{k-1} - \alpha_k H_k \nabla F(w_{k-1})$, where, $k= 1, 2, ..., t,$ α_k is the step length parameter, and the deterministic gradient is given by, $\nabla F(w_{k-1}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f(w_{k-1}, \xi_i)$. However, instead of obtaining the new inverse Hessian H_k from the previous inverse Hessian H_{k-1} and multiplying by $\nabla F(w_{k-1})$ for the update, the LD-BFGS directly approximates the product $H_k \nabla F(w_{k-1})$ by only using latest user specified number of curvature vector pairs.

The advantage of the LD-BFGS method is that instead of storing all the inverse Hessian matrices and the curvature pair information, only the latest certain user specified number of vectors of curvature pairs $\{(s_i, y_i)\}$ where $i = 1, 2, ..., m$, are stored and used in the approximation formula. The user specified number of the vectors *m* is called the batch size where usually, 3<*m*<*20*. At each iteration, the product $H_k \nabla F(w_{k-1})$ is evaluated using the 'L-BFGS two-loop recursion' scheme in that a sequence of inner products and vector summation of $\nabla F(w_{k-1})$ and *m* recent $\{s_i, y_i\}$ pairs are used as mentioned in the Chapter 4 Algorithm 2.

The oldest vector pair is less likely to be relevant in describing the actual behavior of the current Hessian therefore, it is discarded to save storage space. That is after the computation of new iterate, the oldest vector pair in the set $\{(s_i, y_i)\}$ where $i = 1, 2, ..., m$ is replaced by the new curvature pair (s_k, y_k). This method requires much less memory and gets comparative convergence results effectively. However, when the uncertainty in the data, i.e., the number of samples (*N*) is high then the computation of gradient at each iteration over the entire data set incurs high cost of computation per iteration and prohibits the use of LD-BFGS in that case of machine learning applications.

1.2 Stochastic solution methods and challenges in machine learning

In this section, we consider advantages and disadvantages of stochastic methods in unconstrained optimization applications in machine learning.

1.2.1 Stochastic approximation (SA) method: Robbins & Monro in 1951 studied a root finding problem in regression analysis and devised SA update rule to find the roots of the equation $E[g(w,\xi)] = 0$, where g is parameterized by $w \in \mathbb{R}^n$ and ξ is the random variable. The proposed update rules is as follows,

$$w_k := w_{k-1} - \alpha_k g(w_{k-1}, \xi_k), \tag{8}$$

where k=1, 2,…,t, and α_k is the step size parameter. When $g(w_{k-1}, \xi_k) = \nabla F(w_{k-1})$ in equation (8) then the method is known as the 'Stochastic Gradient Descent (SGD) method'. Note that the update rule is similar to the DG's update rule, however, the only difference is in the computation of the gradient.

Here, the gradient is evaluated as $\nabla F(w_{k-1}) = \nabla f(w_{k-1}, \xi_k)$, where $\xi_k = (x_{k,} z_k)$ is the *k*th input-output vector pair. The advantage of the stochastic approach is that, at each step *k*=1, 2,...,*t*, the update involves computation of the gradient ∇F_{k-1} using only a single sample pair ξ_k. This reduces computational efforts by not evaluating over redundant data points present in large data. This also results in to less computationally expensive iterative steps and is comparatively memory efficient. However, the performance of the SA method is highly influenced by the choice of the step size parameter. As incorrect step size parameter may result in to poor convergence and noisy iterations. Also when the dimensionality of the data increases, the computation of actual gradient becomes costly and infeasible.

1.2.2 Stochastic BFGS method (S-BFGS): As mentioned in details in Section 1.1.3, the D-BFGS method resolved the issues associated with computation of the Hessian and the inverse Hessian by providing an approximation formula for both as mentioned by equations (6 and 7). However, in the D-BFGS, gradients at each iteration are obtained by taking summation of the gradients over all the data points such as,

$\nabla F(w_{k-1}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f(w_{k-1}, \xi_i)$), which makes the Deterministic BFGS method impractical to use when the number of data points (*N*) is large. On the other hand, the S-BFGS method replaces the summation of the gradients obtained over entire data points by a noisy gradient computed using only a single data point, that is $\nabla F(w_k) = \nabla f(w_{k-1}, \xi_k)$, where $\xi_k = (x_{k,} z_k)$ is the *k*th input-output vector pair. However, the S-BFGS method has high memory requirements to store the approximated inverse Hessians, which makes the method infeasible to be used on large scale data.

1.2.3 Limited memory Stochastic BFGS method (LS-BFGS): The LS-BFGS (Byrd et.al, 2015) method computes the gradients stochastically as mentioned in Section 1.2.2 which helps the algorithm to be less expensive computationally. Moreover, the product of inverse Hessian and the gradient is approximated using the Algorithm 2 (see Chapter 4) and the old less influencing curvature information s_{k-1} and y_{k-1} is overridden by new curvature information s_k and y_k as mentioned in the Section 1.1.4 in details. By discarding the old inverse Hessian and old curvature pair vectors, the algorithm requires much less memory. The aforementioned reasons make LS-BFGS suitable and an algorithm of choice for large scale unconstrained optimization applications in machine learning.

1.3 Preliminaries

Before we explore key concepts of large scale stochastic optimization, we define the following terms:

Definition 1 (Large scale stochastic optimization problem) Problem (3) is considered to be large scale when it deals with high dimensional data (i.e. *n,* number of variables is very large) and is considered to be stochastic when it deals with high volume of data (i.e. *N*, number of instances is very large).

Definition 2 (Norm) A p-norm is a function that assigns a positive length or size to each vector in a vector space given as

$$\|w\|_p = \left(\sum_{i=1}^{n} |w_i|^p \right)^{1/p}. \tag{9}$$

By inserting different values of p in equation (9), a variety of norms can be realized as follows,

(i) $L_1 norm$: When $p=1$, we get $\|w\|_1 = \sum_{i=1}^{n} |w_i|$.

(ii) $L_2 norm$: Also known as Euclidean norm, when $p=2$, we get, $\|w\|_2 = \sqrt{\sum_{i=1}^{n} w_i^2}$.

(iii) $L_\infty norm$: Also known as Max-norm, when $p = \infty$, we get $\|w\|_\infty = \max |w_i|$ for i=1, 2,…, n.

Definition 3 (Hessian) The Hessian $\nabla^2 f(x)$ is a symmetric $n \times n$ matrix whose entries are second order partial derivatives of f at x:$[\nabla^2 f(x)]_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$, for $i, j = 1, \dots, n$.

Definition 4 (Positive definite matrix) A matrix A is positive definite if it is symmetric and all its eigenvalues are positive or if $x^T A x > 0$ for all vectors $x \in \mathbb{R}^n$, where $x \neq 0$.

Definition 5 (Semi positive definite matrix) A matrix A is semi-positive definite if it is symmetric and $x^T A x \geq 0$ for all vectors $x \in \mathbb{R}^n$, where $x \neq 0$.

Definition 6 (Convex sets) A set in $X \in \mathbb{R}^n$ is convex if the line segment between any two points in X lies in X, i.e., if for any $x_1, x_2 \in X$ and for any θ with $0 \leq \theta \leq 1$, we have, $\theta x_1 + (1-\theta) x_2 \in X$.

Definition 7 (Convex functions) A function $f : X \in \mathbb{R}^n \to \mathbb{R}$ is convex if X is convex set in $\mathbb{R}^n$ and if for all $x_1, x_2 \in$ X, and $\theta \in (0,1)$, when it satisfies either case:

(i) In the absence of differentiability,

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta\, f(x_1) + (1-\theta)\, f(x_2).$$

(ii) If the function is differentiable,

$$\left(\nabla f(x_1) - \nabla f(x_2)\right)^T (x_1 - x_2) \geq 0\,.$$

(iii) If the function is twice differentiable,

$$\nabla^2 f(x_1) \succcurlyeq 0\,.$$

Definition 8 (Strictly convex functions) A function $f : X \in \mathbb{R}^n \to \mathbb{R}$ is strictly convex if X is a strictly convex set in $\mathbb{R}^n$ and if for all $x_1, x_2 \in$ X, and $\theta \in (0,1)$, when it satisfies either case:

(i) In the absence of differentiability,

$$f(\theta x_1 + (1-\theta)x_2) < \theta\, f(x_1) + (1-\theta)\, f(x_2).$$

(ii) If the function is differentiable,

$$\left(\nabla f(x_1) - \nabla f(x_2)\right)^T (x_1 - x_2) > 0\,.$$

(iii) If the function is twice differentiable,

$$\nabla^2 f(x_1) \succ 0\,.$$

Definition 9 (Strongly convex functions) A function $f : X \in \mathbb{R}^n \to \mathbb{R}$ is strongly convex with parameter $\lambda > 0$, when X is convex for all $x_1, x_2 \in$ X,

(i) If the function is differentiable,

$$\left(\nabla f(x_1) - \nabla f(x_2)\right)^T (x_1 - x_2) \geq \lambda \left\| x_1 - x_2 \right\|_2^2.$$

(ii) If the function is twice differentiable,

$$\nabla^2 f(x_1) \succcurlyeq \lambda I.$$

Definition 10 (Convex optimization problem) A convex optimization problem is one of the form, min $f_0(x)$, subject to $f_i(x) \leq b_i, \quad i = 1, \dots, m,$ where, the functions $f_0, \dots, f_m : \mathbb{R}^n \to \mathbb{R}$ are convex that is they satisfy, $f_i(\alpha x + \beta y) \leq \alpha\, f_i(x) + \beta\, f_i(y)$ for all $x, y \in \mathbb{R}$ and all $\alpha, \beta \in R$ with $\alpha + \beta = 1,\ \alpha, \beta \geq 0.$

1.4 Loss function equations

The loss function in (3) can represent variety of problems arising in machine learning such as:

(i) *Logistic loss function:* A logistic regression loss function used for binary classification problems is given as $l(h(w; x_i), z_i) = \log(1 + e^{-z_i x_i^T w})$ where $\log(1 + e^{-z_i x_i^T w})$ is the logit loss function, $x_i \in \mathbb{R}^n$ is the input vector, $z_i \in \{-1,1\}$ is actual binary output vector and function is parameterized by $w \in \mathbb{R}^n$. The goal is to learn a classifier w which can classify the input vectors in to binary true or false with minimum error possible. See Section 5.1.b for more detailed information.

(ii) *Least square loss function:* It is when in equation (3) we have $l(h(w; x_i), z_i) = \frac{1}{2}||w^T x_i - z_i||^2$ to find the squared error between the predicted output $h(w; x_i)$ and the actual output z_i. Such loss function are used in image deblurring applications (see Beck & Teboulle, 2009) where x_i is denoted as blur operator, w^T as true image vectors and z_i as the captured image vectors to get clear images. See Section 5.2 for more detailed information.

(iii) *Support vector machine (SVM):* The SVM problem is in which we have a large number of training data of inputs, $x_i \in \mathbb{R}^n$ and output, $z_i \in \mathbb{R}$ pairs where $i = 1, 2, \dots, N$ and $v_i \in \{-1,1\}$ is the categorical output. The goal is to learn a hyperplane $h(w; x_i)$

parametrized by vector w. Then a function which measures the distance of the observed output z_i from the classifier function h can be modeled as a convex loss function given as $l(h(w; x_i), z_i)$. See Section 5.1.c for more detailed information.

2.5 Regularization

To find an optimal solution to the problem (3) using any quasi-Newton method such as the BFGS algorithm the objective function needs to be strongly convex. This implies an assumption of the Hessian to be positive definite to achieve proper rate of convergence. However, most of the machine learning optimization problem are convex but not strongly convex which results in undesired rate of convergence when quasi-Newton algorithms are applied. Moreover, the solutions generated by solving even the strongly convex function using such methods result in non-sparse solutions that have high memory requirements and high cost of computation per iteration. In order to achieve the desired missing properties such as strong convexity and sparsity, the regularization technique is used. A regularized convex optimization objective function can be given as,

$$F_\lambda(w) = F(w) + \lambda R(w), \tag{10}$$

where $R: \mathbb{R}^n \to \mathbb{R}$ is a proper convex regularizing function with respective regularization parameter, $\lambda > 0$ being a scalar.

To convert a convex function into a strongly convex function L_2 regularization is used i.e. when $R(w) = \frac{1}{2}\|w\|_2^2$ and $\lambda = \lambda_2$ in equation (10) where λ_2 is called the L_2 regularization parameter. Therefore, when the objective function in the problem (3) is added with the term $\frac{1}{2}\lambda_2\|w\|_2^2$ and the approximate problem of the form,

$$\min_{w \in \mathbb{R}^n} F(w) + \frac{1}{2}\lambda_2 \|w\|_2^2, \tag{11}$$

where $\|w\|_2$ denotes L_2 or Euclidean vector norm, is solved to establish convergence properties equipped with rate results. This type of regularization is popularly known as Tikhonov regularization named after Andrey Tikhonov in machine learning community.

To induce sparsity in the solutions, L_1 regularization is used i.e. when $R(w) = \|w\|_1$ and $\lambda = \lambda_1$ in equation (10) where λ_1 is called the L_1 regularization parameter. Therefore, when the objective function in the problem (3) is added with the term $\lambda_1 \|w\|_1$ and the approximate problem of the form,

$$\min_{w \in \mathbb{R}^n} F(w) + \lambda_1 \|w\|_1, \tag{12}$$

where $\|w\|_1$ denotes L_1 norm, is solved, sparse solutions are obtained. This type of regularization is popularly used in LASSO introduced by Tibshirani (1996) for linear least square functions.

Therefore, to obtain sparse solutions to non-strongly convex objective functions using methods like the BFGS methods both $L_1$and L_2 regularization are used and the regularized problem becomes,

$$\min_{w \in \mathbb{R}^n} F(w) + \lambda_1 \|w\|_1 + \frac{1}{2}\lambda_2 \|w\|_2^2. \tag{13}$$

CHAPTER II

LITERATURE REVIEW

Since past century, we have witnessed technological advancements in powerful computing platforms. Today data is massively generated and stored which is readily available to extract information like never before. To tap the information from such large data, various data mining models are employed, but the models generally have prediction errors. These errors can be written in the form of a loss function, $f(w,\xi_i):\mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ in the expected minimization problem such as, $\min_{w\in\mathbb{R}^n} F(w) := E[f(w,\xi_i)] = \int f(w,\xi_i)P(\xi_i)$ where $w \in \mathbb{R}^n$ is the decision variable and $\xi_i \in \mathbb{R}^d$ is the random variable. The random variable $\xi_i = \{(x_{i,} z_i)\}_{i=1}^N$ has $x_i \in \mathbb{R}^n$ as the ith input vector, $z_i \in \mathbb{R}$ as the ith true output vector, N as the total number of input-output pairs and n as the dimensionality of w and x vectors. Such problems are minimized using various convex optimization methods and algorithms which started developing since 19th century.

However, with high dimensionality and uncertainty in the data, the evaluation of the multi-dimensional integrals in the expected objective function as mentioned above becomes infeasible in the absence of the probability distribution *P*. To formulate the objective function of the problem in the absence of the distribution information, Monte Carlo simulation method such as Sample Average Approximation (SAA) schemes has been widely used in the literature Kleywegt et al. (2002) and Shapiro et al. (2003). As per the SAA scheme, the minimization problem could be approximated as,

$\min_{w \in \mathbb{R}^n} F(w) = \frac{1}{N}\sum_{i=1}^{N} f(w, \xi_i)$, which is also known as ‘Empirical Risk Minimization (EMR)’ problem in machine learning.

Traditionally, to solve EMR problems deterministic schemes such as the Deterministic Gradient method (DG) (see Section 1.1.1) and the Deterministic Newton method (DN) (see Section 1.1.2) were used. The deterministic methods evaluate the gradients at each iteration using all the data points such as, $\nabla F(w_{k-1}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f(w_{k-1}, \xi_i)$ for k= 1, 2,…, t which results in high cost of computation at each iteration and might take infeasible amount of time to converge when the number of data points N is large. Therefore, the deterministic scheme becomes inefficient (see Nemirovski et al., 2009), expensive (see Bottou, 2010) and ill-suited for learning continuous stream of big data (see Schraudolph et al., 2007).

In 1951, Robbins and Monro developed first Stochastic Approximation (SA) (see Section 1.2.1) method (Robbins & Monro, 1951) which resolved the above mentioned issues by considering small random subsamples or a single data point at a time from the training data to compute gradients reducing the computational requirement as studied in Bottou (2010). In the stochastic methods, the gradients at each iteration is evaluated as $\nabla F(w_k) = \nabla f(w_k, \xi_k)$, where $\xi_k = (x_{k,} z_k)$ is the k^{th} input-output vector pair for k= 1, 2,…, t. The advantage of the stochastic approach is that, at each step k=1, 2,...,t, the update involves computation of the gradient ∇F_k using only a single random pairs ξ_k which results in less computationally expensive iterative steps and is comparatively memory efficient. Although, the SA is comparatively efficient (Nemirovski et al. 2009), its performance is highly sensitive to the choice of step-size sequence as mentioned in Yousefian et al. (2012). Also the convergence results become poorer with increase in the dimensionality of the data (see Mokhtari & Ribeiro, 2014).

On the other hand, often it is either expensive or impossible to compute actual gradients, in such cases, gradients are approximated using finite-difference methods as first mentioned by Kiefer and Wolfowitz in 1952 and later developed more by Spall in 1992. Such methods are known as quasi-Newton (QN) methods. The Deterministic QN methods exhibit reliable performance while handling high dimensional problems (Schraudolph et al., 2007). Moreover, in some schemes the curvature information of objective function is incorporated to approximate the gradients and the Hessians by increasing robustness. One of the popular update rule in such regime is the Deterministic BFGS method (see Section 1.1.3), named after Broyden, Fletcher, Goldfarb, and Shanno (Fletcher, 1987) which incorporates displacement factor s_k and gradient mapping y_k (see Chapter 4, Algorithm 2).

However, when the data is large scale, the D-BFGS method and other deterministic QN method requires lot of memory and have slow convergence when enough computational memory is not available (Liu & Nocedal, 1989). Therefore, limited memory deterministic version such as LD-BFGS (see Byrd et al., 2015) (see Section 1.1.4), is employed to handle such issues arising in large scale optimization. The main idea in the LD-BFGS is that instead of storing entire $n \times n$ matrix at each iteration, only a fix number of vectors (R^n) are stored and used to compute approximate Hessian inverse using two loop recursive scheme given Nocedal & Wright (1999). However, the computation of the gradients and the Hessians deterministically incurs high computational cost per iteration and makes the deterministic methods impractical for large scale data optimization.

Moreover, in the developed full memory and limited memory QN methods, it is assumed that the objective function is strongly convex (Byrd et al., 2015) and violation of that assumption hampers the rate of convergence as mentioned by Yousefian et al. (2016). As, the objective functions in most of the applications are convex but not strongly convex, they are regularized to make them strongly convex as seen in Regularized version of the BFGS also known as RES (see Mokhtari & Ribeiro, 2014). However, the solutions generated with regularization are of approximate problem and limited information is available to find the solutions to the original problem.

To overcome the mentioned challenges such as high computational-memory requirements, slow convergence and high cost of gradient computation faced by classical deterministic quasi-Newton algorithms such as LD-BFGS method while handling Big Data, new limited memory stochastic variants of classical methods have been developed such as the Limited memory Stochastic BFGS (LS-BFGS) method as mentioned in Byrd et. al., 2015 (see Section 1.2.3). However, the solutions generated through the LS-BFGS method are dense which could result in potential memory outage issues and a limited literature is available to obtain sparse solutions iteratively to non-strongly convex objective functions using LS-BFGS method. To address this gap, we present an 'Iterative L_1 Regularized LS-BFGS (iRLS-BFGS) method' for big data applications. The proposed iRLS-BFGS algorithm uses L_2 regularization to make the objective function strongly convex and an iterative L_1 regularization term to induce sparsity. We identify the best update rules through the application of the iRLS-BFGS method for text classification using logistic regression which generates sparse solutions and demonstrates faster rate of convergence on the non-strongly convex optimization problems arising in machine learning. Later, we test the best found update rate on the Image deblurring application in signal processing and compare the convergence results.

CHAPTER III

MOTIVATION AND OBJECTIVES

The focus of the thesis is to develop and implement an iterative Limited memory Stochastic Broyden, Fletcher, Goldfarb, and Shanno method (LS-BFGS) to solve large scale unconstrained optimization problems arising in machine learning of the form (3). All the quasi-Newton methods discussed in Section 1.1 and 1.2 to solve the problem (3) require the objective function to be strongly convex to assure convergence, however, in reality most applications have convex but not strongly convex objectives. To make the objective function strongly convex, we use L_2 regularization where we add the term $\frac{1}{2}\lambda_2 \|w\|_2^2$ to the objective function where, $\lambda_2 > 0$ is a constant L_2 regularization parameter (Nocedal et.al, pg. 9, 2015) and $\|w\|_2$ is the L_2 norm as defined in Section 1.3 *Definition 2 (ii)*.

Moreover, the solutions generated by solving L_2 regularized objective function may be dense which require high storage memory, therefore, we employ L_1 regularization term $\lambda_1 \|w\|_1$ to induce sparsity in the solutions where, λ_1 is the L_1 regularization parameter and $\|w\|_1$ is the L_1 norm as defined in Section 1.3 *Definition 2 (ii)*. Therefore the problem (3) is written as,

$$\min_{w\in\mathbb{R}^n} F(w) = \frac{1}{N}\sum_{i=1}^{N} f(w,\xi_i) + \frac{1}{2}\lambda_2 \|w\|_2^2 + \lambda_1 \|w\|_1, \tag{12}$$

where $\xi_i = \{(x_{i,} z_i)\}_{i=1}^{N}$ is the ith input-output pair vector. However, addition of constant L_1 regularization term to get sparse solutions changes the original problem (3) and the obtained solutions of regularized problem (12) are not exact, instead they are solutions to an approximate problem.

Therefore, there is a trade-off in selection of L_1 regularization parameter λ_1 in which higher value of it leads to sparse solutions but the solutions obtained are not exact while lower value of λ_1 might generate exact optimal solutions but with low sparsity. Moreover, limited information is available in the literature for the selection of appropriate L_1 regularization term to obtain exact sparse solutions to the original problem. Motivated by this gap our goal is to find exact sparse solutions to problem (12) by decaying the L_1 regularization parameter, λ_1 using the LS-BFGS algorithm. We employ iterative L_1regularization scheme in which the L_1regularization parameter decays at the rate of $\frac{\lambda_1}{k^{\beta}}$, that is the value of the initial L_1regularization parameter λ_1 decreases during the implementation of the algorithm as the iteration number increases from $k = 1, 2, ..., t$. We test the algorithm on test application for different values of simulation parameters, λ_1 and β to find the best update rule.

Moreover, as we know from Section 1.2 that in stochastic optimization algorithms, the choice step-size also plays a vital role to get convergence. In order to obtain feasible update rules for step-size, we employ iterative step-size parameter with drop rate of $\frac{\theta}{k^{\alpha}}$ as the algorithm proceeds with $k = 1, 2, ..., t$ for different values of simulation parameters, θ and α .

The objective of the thesis is to implement an iterative L_1 Regularized Limited memory Stochastic BFGS (iRLS-BFGS) algorithm to find out the best update rule for the iterative L_1 regularization term and the step-size parameter by considering the simulation parameters, θ, α , λ_1 and β that gives best convergence results. For that, the iRLS-BFGS algorithm is coded in MATLAB v9.0.0.341360 (R2016a) and is tested on two large scale (see Section 1.3 *Definition 6*) applications. First, text classification using logistic regression and Support Vector Machine (SVM) over RCV1 dataset (see Lewis et al., 2004) to classify the document type (Section 5.1). Second, image deblurring problem where the blur error is realized in the form of the least square loss function and is minimized using the developed algorithm (Section 5.2) to get a clear image. In this study, we run simulation scenarios with different combinations of θ, α , λ_1 and β on the Cowboy (Oklahoma State University's super computer) and compare the convergence results for each application to determine the best update rules.

CHAPTER IV

ALGORITHMS

In this section, we present the notations used in the developed 'iterative *L_1* Regularized Limited memory Stochastic BFGS algorithm' (iRLS-BFGS), the outline of the iRLS-BFGS algorithm, and the 'Two loop scheme' developed by Nocedal. The developed algorithm is characterized by the iterative step-size parameter given by $\alpha_k = \frac{\theta}{k^{\alpha}}$ and the iterative L_1 regularization parameter given by $\lambda_{1k} = \frac{\lambda_1}{k^{\beta}}$, where we find the best update rules by simulating different possible parameter settings for λ_1, β, θ and α. The novelty of the iRLS-BFGS algorithm over the classical LS-BFGS algorithm is that, in the iRLS-BFGS algorithm the step-size parameter and the L_1 regularization parameter decay during the algorithm to generate exact sparse solutions. In this research, based on the earlier work on the iterative regularization for stochastic approximation methods as mentioned in Yousefian et al., 2017, we assume the simulation decay rate parameter values such that $\alpha + \beta = 1$ where $\alpha > \beta$ for all applications.

4.1 Algorithm outline:

4.1 Algorithm outline: Before we present our algorithm, we define following notations,

N: the number of training data points used to train the model

w_k: the decision vector generated at k^{th} iteration, $w_1 = 0_{n\times 1}$

n: the number of variables associated with each data point

m: the batch size which is user specified number of vectors stored per iteration, $m = 20$

x_i: the i^{th} input vector ($\mathbb{R}^n$) from *X,* the input $n \times N$ matrix. (*i=1,2,…,N)*

z_i: the i^{th} ouput ($\mathbb{R}$) from *Z,* the output $1\times n$ vector. (*i=1, 2,..., N)*

λ_{1k}: the iterative L_1regularization parameter, $\lambda_{1k} = \frac{\lambda_1}{k^{\beta}}$, where $k = 1, 2, ..., t$

λ_2: the fixed L_2 regularization parameter, (An arbitrary small value)

$F(w, x, z, \lambda_{1k}, \lambda_2)$: the objective function as per application where

$F(w, x, z, \lambda_{1k}, \lambda_2) = \frac{1}{N}\sum_{i=1}^{N} f(w, x_i, z_i) + \frac{1}{2}\lambda_2 \|w\|_2^2 + \lambda_{1k}\|w\|_1$ in which $f(w, x_i, z_i)$ is the loss function as mentioned in Section 1.4

$\nabla F(w_k; x_k, z_k, \lambda_{1k}, \lambda_2)$: the gradient of the objective function $F(w, x, z, \lambda_{1k}, \lambda_2)$ respectively

s_k: the k^{th} displacement vector given as, $s_k = w_{k+1} - w_k$

S: the $n\times m$ matrix which gets updated at each iteration by only storing *m* latest s vectors

y : the gradient difference vector given as, $y_k = \nabla F(w_{k+1}; x_k, z_k) - \nabla F(w_{k;}; x_k, z_k)$

Y: the $n\times m$ matrix which is updated at each iteration by only storing *m* latest y vectors

e: the number of epochs, checkpoints at which objective function is evaluated

q: the counter, 1,2,…, *e*

epoch: {*qN/ e* | *q=1,2,..., e* }

O_q: the objective function value at each epoch

α_k : the iterative step length parameter, $\alpha_k = \frac{\theta}{k^{\alpha}}$, where $k = 1, 2, ... t$ and θ, α

We implemented the following developed algorithm in our research:

Algorithm 1: An Iterative L_1 Regularized Limited memory Stochastic BFGS (iRLS-BFGS)

```
Initialization
Set $\lambda_2$, $m$, $e$, $w_1$, $\lambda_1$, $\beta$, $\theta$ and $\alpha$
for $k = 1$ to $m$ do
$\alpha_k = \frac{\theta}{k^{\alpha}}$
$\lambda_{1k} = \frac{\lambda_1}{k^{\beta}}$
$w_{k+1} := w_k - \alpha_k \nabla F(w_k; x_k, z_k, \lambda_{1k}, \lambda_2)$
$s_k = w_{k+1} - w_k$
$y_k = \nabla F(w_{k+1}; x_k, z_k, 0, \lambda_2) - \nabla F(w_{k;}; x_k, z_k, 0, \lambda_2)$
        if $k = epoch$
        $O_q = F(w_{k+1}, X, Z, \lambda_{1k}, \lambda_2)$
        $q = q + 1$
        end if
end for
for $k = m$ to $N$ do
$w_{k+1} := w_k - \alpha_k H_k \nabla F(w_k; x_k, z_k, \lambda_{1k}, \lambda_2)$,        ⊳ Evaluate $H_k \nabla F_k$ as per Algorithm 2
$s_k = w_{k+1} - w_k$
$y_k = \nabla F(w_{k+1}; x_k, z_k, 0, \lambda_2) - \nabla F(w_{k;}; x_k, z_k, 0, \lambda_2)$
        if $k = epoch$
        $O_q = F(w_{k+1}, X, Z, \lambda_{1k}, \lambda_2)$
        $q = q + 1$
        end if
update $S$ and $Y$
end for
```

4.2 Two loop recursion scheme:

Two loop recursion scheme: This scheme was developed by Nocedal in 1999 to evaluate the product of the Hessian and the gradient using only the recent curvature pair information. The advantage this scheme is that it evaluates the product of the Hessian and the gradient without calculating or storing any matrix.

The two-loop scheme is given as:

Algorithm 2: L-BFGS two-loop recursion

```
$q \leftarrow \nabla F_k$;
**for** $i = k-1, k-2, \ldots, k-m$ **do**
    $\alpha_i \leftarrow \rho_i s_i^T q$;                ⊳ Where, $\rho_i = \frac{1}{y_i^T s_i}$
    $q \leftarrow q - \alpha_i y_i$;
**end for**
$r \leftarrow H_k^0 q$;                             ⊳ Initialize $H_k^0 = \gamma_k I$, where, $\gamma_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}$,
**for** $i = k-m, k-m+1, \ldots, k-1$ **do**
    $\beta \leftarrow \rho_i y_i^T r$;
    $r \leftarrow r + s_i(\alpha_i - \beta)$;
**end for**

Stop with result $H_k \nabla F_k = r$
```

CHAPTER V

NUMERICAL EXPERIMENTS

In this section we compare the convergence properties of the developed iterative *L1* Regularized Limited memory Stochastic BFGS (iRLS-BFGS) algorithm with classical *L1* Regularized Limited memory Stochastic BFGS (cRLS-BFGS) algorithm. For that, we implemented the iRLS-BFGS algorithm (Chapter 4) on various problems arising in machine learning of the form,

$$\min_{w\in\mathbb{R}^n} F(w) = \frac{1}{N}\sum_{i=1}^{N} f(w,\xi_i) + \frac{1}{2}\lambda_2 \|w\|_2^2, \tag{13}$$

where $\frac{1}{2}\lambda_2 \|w\|_2^2$ is the L_2 regularization term in which $\lambda_2 > 0$ is the L_2 regularization parameter. The classical approach uses constant *L1* regularization to solve the problem 13, however, in the developed iRLS-BFGS algorithm the *L1* regularization is performed iteratively in the algorithm, i.e., by addition of iterative *L1* regularization term to the gradient function during the optimization to generate exact sparse solutions.

Also, the update rule of the iRLS-BFGS is given by, $w_k := w_{k-1} - \alpha_k H_k \nabla F(w_k)$, where $\alpha_k = \frac{\theta}{k^{\alpha}}$ is the iterative step-size parameter in which θ is the initial step-size parameter and α is the step-size decay rate parameter. Therefore, to obtain the best update rule of the iterative L_1 regularization parameter and the iterative step-size parameter, we implemented the iRLS-BFGS algorithm with different combinations of parameter setting $\theta, \alpha, \lambda_1$ and β in two different big data applications as mentioned later in this chapter.

5.1 Binary text classification

We consider a binary classification problem using logistic regression and Support Vector Machine (SVM) in our research to train a classification model and have the minimum classification error.

5.1.a RCV1 Dataset: The dataset in consideration is the RCV1dataset (see Lewis et al., 2004) which consists of newswire articles produced in the Reuters magazine from 1996-1997. The dataset consists of words present in the articles that are characterized into four categories namely, Corporate/Industrial, Economics, Government/Social, and Markets. Our goal is to classify the documents into 'Markets' and 'non-Markets'. The raw data file consisted tokenized words present in the articles, which were then organized into a term-document matrix using Python. A term-document matrix is a matrix, which has rows representing the documents, i.e., individual articles, and columns representing the tokens. The presence of a word in an article is indicated by a binary parameter in the respective article's row and column. As a result we obtain a feature input vector in $\mathbb{R}^n$ where, $n = 13892$ is the total number of unique words for each of $N = 199328$ documents. Each input vector $x_i \in [0,1]^n$ is extremely sparse and output labels $z_i \in \{-1,1\}$ where, 1 indicates the article of 'Markets' and -1 of other categories. The iRLS-BFGS algorithm is implemented to minimize the objective function for number of iterations, $t = 10000$.

5.1.b Logistic Regression: Here we train a binary logistic classification model using the training dataset which has *N* random input-output pairs, $\xi_i = \{(x_{i,} z_i)\}_{i=1}^N$ in which $x_i \in \mathbb{R}^n$ is the ith binary input vector and $z_i \in \{-1,1\}$ is the ith true output for all $i = 1, 2, ..., N$.

i. Objective function: At each iteration of the algorithm, we evaluate the regularized objective function in the problem (3) which is given by,

$$F(w; x, z, \lambda_{1k}, \lambda_2) = \frac{1}{N}\sum_{i=1}^{N} f(w, x_i, z_i) + \frac{1}{2}\lambda_2 \|w\|_2^2 + \lambda_{1k}\|w\|_1, \tag{14}$$

where $f(w, x_i, z_i) = \log(1 + e^{-z_i x_i^T w}) : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ is the logistic binary classification loss function, $w \in \mathbb{R}^n$ is the decision variable vector, $x_i \in \mathbb{R}^n$ is the ith binary input vector, $z_i \in \mathbb{R}$, $z_i = \{-1,1\}$ is the ith true output for all $i = 1,2, \dots, N$ and *n* is the dimensionality of parameter *w* and input *x* vectors. Also, λ_2 is the constant L_2 regularization parameter, here λ_2 =0.1 and $\lambda_{1k} = \frac{\lambda_1}{k^\beta}$ is the iterative L_1 regularization parameter which decreases the influence of initial L_1 regularization parameter λ_1 on the objective function as the algorithm proceeds ($k = 1, 2, ..., t$). The corresponding regularized stochastic logistic gradient at the kth iteration is evaluated as,

$$\nabla F(w; x_i, z_i, \lambda_{1k}, \lambda_2) = -\frac{z_i}{1 + e^{z_i x_i^T w}} x_i + \lambda_{1k} sign(w) + \frac{\lambda_2}{2} w, \tag{15}$$

where *sign(.)* is the sign operator.

ii. **Concluding remarks:** To find the best update rules given by $\lambda_{1k} = \frac{\lambda_1}{k^{\beta}}$ and $\alpha_k = \frac{\theta}{k^{\alpha}}$ we find the best parameter set of the parameters λ_1, β, θ and α by comparing the convergence properties of the iterative L_1 Regularized Limited memory Stochastic BFGS (iRLS-BFGS) algorithm with the classical L_1 Regularized Limited memory Stochastic BFGS algorithm using scenario approach. **Table 5.1.b** shows all the scenario codes with combination of algorithm parameters.

- *Scenario codes and parameters*: **Table 5.1.b** shows the 48 scenario codes and corresponding values of $\theta, \lambda_1, \alpha$ and β used in the experiment.

Table 5.1.b.i

	No.	Scenario code	α	β
	1	$S1_1$	1	0
θ =0.001	2	$S1_2$	0.5	0.5
λ_1 =0.001	3	$S1_3$	0.67	0.33
	4	$S1_4$	0.9	0.1
	5	$S2_1$	1	0
θ =0.001	6	$S2_2$	0.5	0.5
λ_1 = 0.01	7	$S2_3$	0.67	0.33
	8	$S2_4$	0.9	0.1
	9	$S3_1$	1	0
θ =0.001	10	$S3_2$	0.5	0.5
λ_1 = 0.1	11	$S3_3$	0.67	0.33
	12	$S3_4$	0.9	0.1
	13	$S4_1$	1	0
θ = 0.01	14	$S4_2$	0.5	0.5
λ_1 =0.001	15	$S4_3$	0.67	0.33
	16	$S4_4$	0.9	0.1
	17	$S5_1$	1	0
θ = 0.01	18	$S5_2$	0.5	0.5
λ_1 = 0.01	19	$S5_3$	0.67	0.33
	20	$S5_4$	0.9	0.1
	21	$S6_1$	1	0
θ = 0.01	22	$S6_2$	0.5	0.5
λ_1 = 0.1	23	$S6_3$	0.67	0.33
	24	$S6_4$	0.9	0.1

Table 5.1.b.ii

	No.	Scenario code	α	β
	25	$S7_1$	1	0
θ = 0.1	26	$S7_2$	0.5	0.5
λ_1 =0.001	27	$S7_3$	0.67	0.33
	28	$S7_4$	0.9	0.1
	29	$S8_1$	1	0
θ = 0.1	30	$S8_2$	0.5	0.5
λ_1 = 0.01	31	$S8_3$	0.67	0.33
	32	$S8_4$	0.9	0.1
	33	$S9_1$	1	0
θ = 0.1	34	$S9_2$	0.5	0.5
λ_1 = 0.1	35	$S9_3$	0.67	0.33
	36	$S9_4$	0.9	0.1
	37	$S10_1$	1	0
θ = 1	38	$S10_2$	0.5	0.5
λ_1 =0.001	39	$S10_3$	0.67	0.33
	40	$S10_4$	0.9	0.1
	41	$S11_1$	1	0
θ = 1	42	$S11_2$	0.5	0.5
λ_1 = 0.01	43	$S11_3$	0.67	0.33
	44	$S11_4$	0.9	0.1
	45	$S12_1$	1	0
θ = 1	46	$S12_2$	0.5	0.5
λ_1 = 0.1	47	$S12_3$	0.67	0.33
	48	$S12_4$	0.9	0.1

- *Output plot*: **Figure 5.1.b.1 to Figure 5.1.b.12** shows, the convergence plots of the objective function with *L_1* regularization and scenario parameters corresponding to legend's respective values in **Table 5.1.b**.

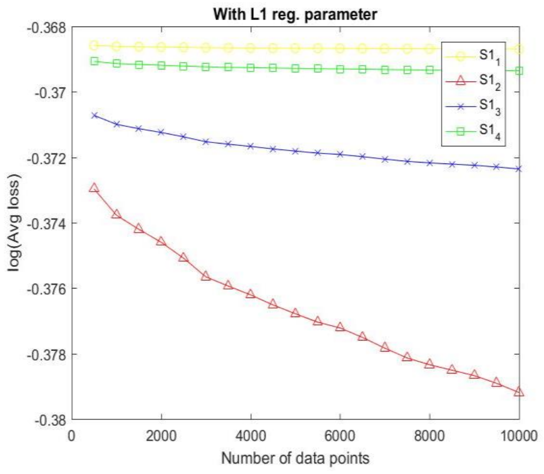


Figure 5.1.b.1

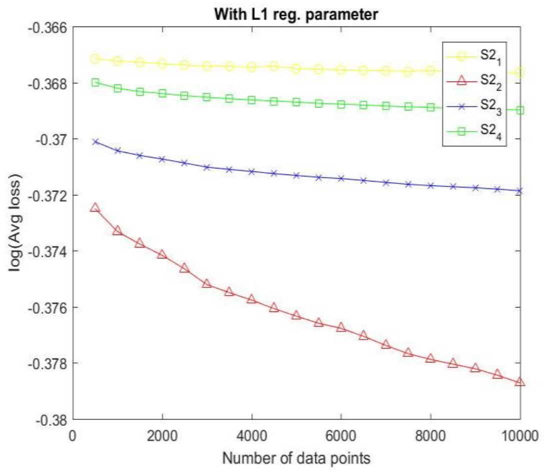


Figure 5.1.b.2

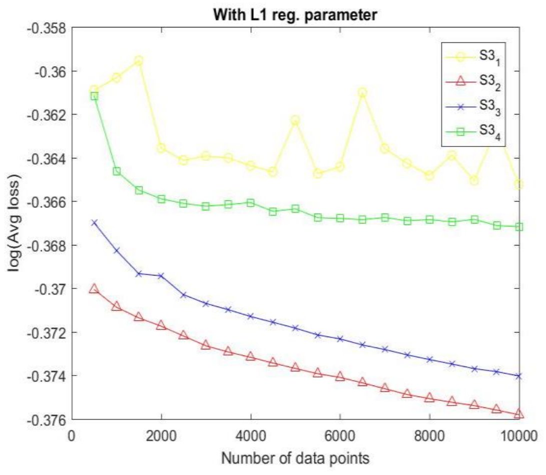


Figure 5.1.b.3

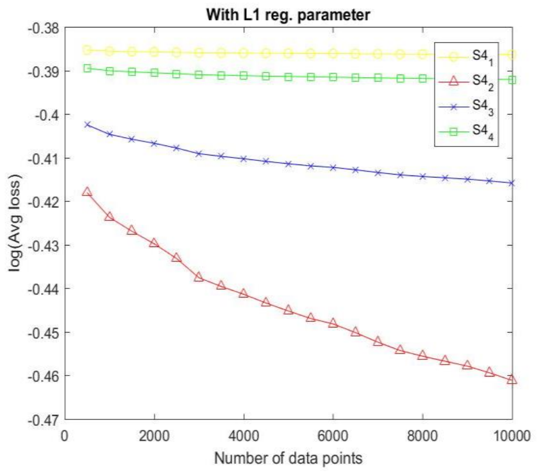


Figure 5.1.b.4

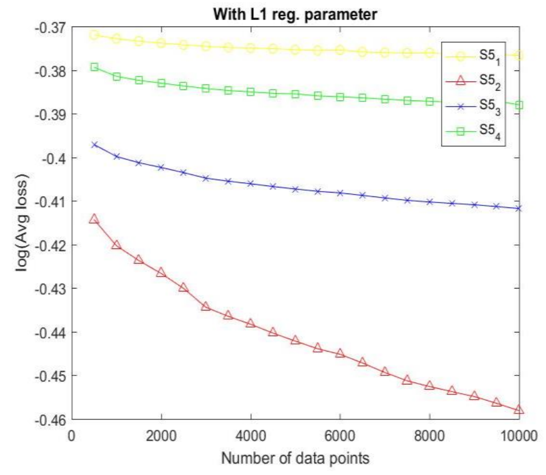


Figure 5.1.b.5

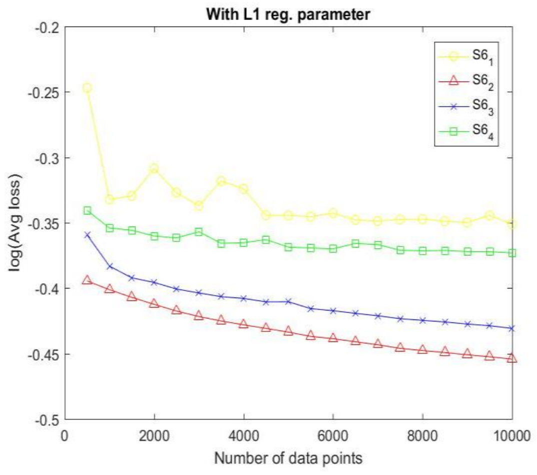


Figure 5.1.b.6

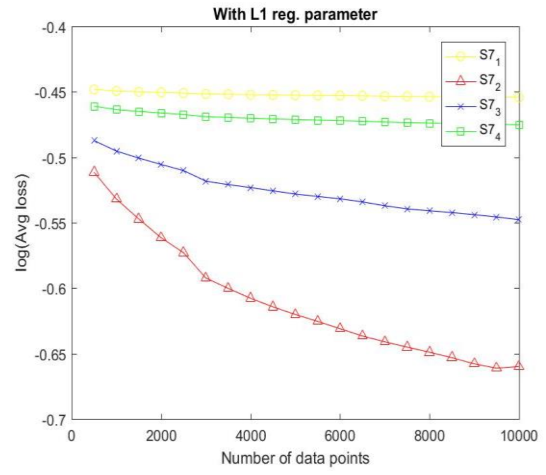


Figure 5.1.b.7

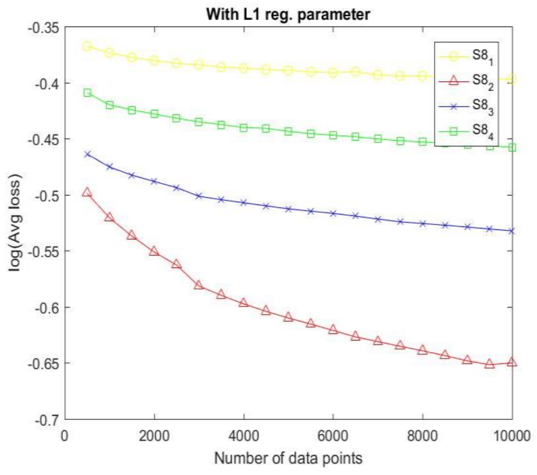


Figure 5.1.b.8

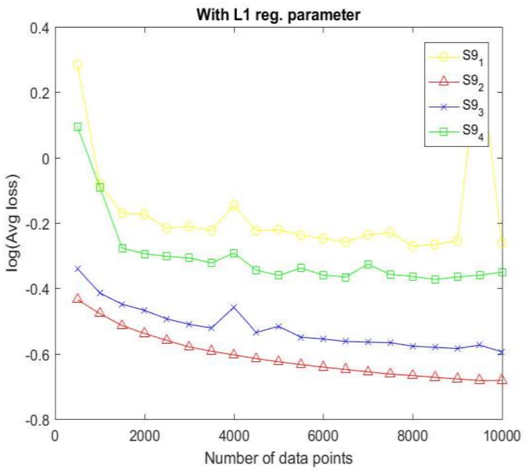


Figure 5.1.b.9

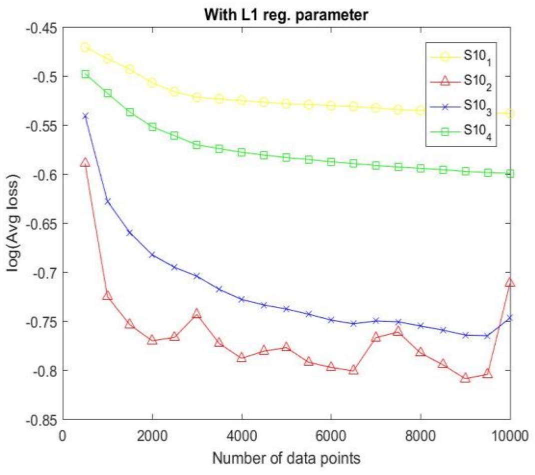


Figure 5.1.b.10

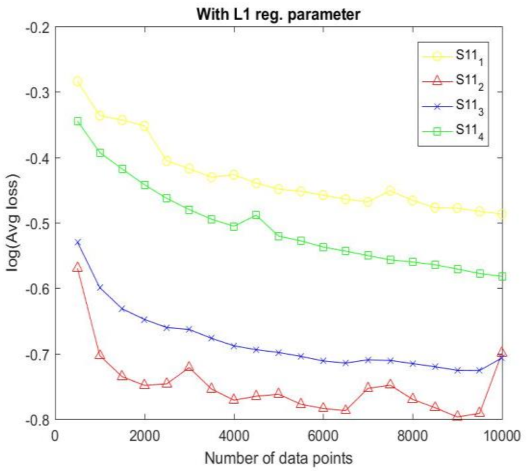


Figure 5.1.b.11

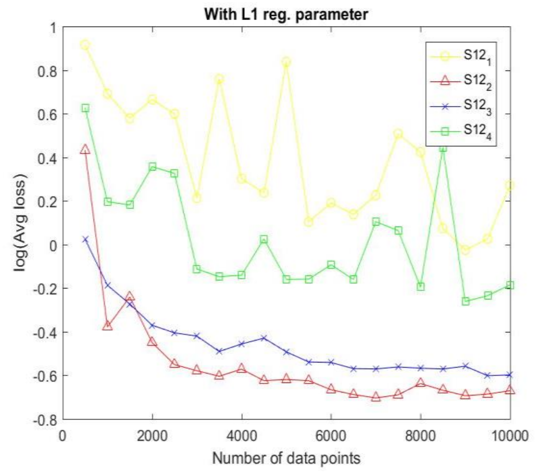


Figure 5.1.b.12

- *Interpretations:* In this experiment, we implemented the iRLS-BFGS algorithm to solve large scale unconstrained nonlinear optimization problems arising in machine learning. The iRLS-BFGS algorithm was tested on RCV1 dataset to minimize the binary logistic regression loss function used for text classification. The iterative L_1 regularization parameter in the iRLS-BFGS method is given as $\lambda_{1k} = \frac{\lambda_1}{k^{\beta}}$ and the step-size parameter is given as $\alpha_k = \frac{\theta}{k^{\alpha}}$ where, the initial value of step-size is given as θ

={0.001,0.01,0.1,1}, the initial value of L_1 regularization parameter is given as λ_1 ={0.001,0.01,0.1} and the drop rate parameters as (α , β **) =** {(1,0), (0.5,0.5), (0.67,0.33), (0.9,0.1)} for $k = 1, 2, \ldots t$. **Table 5.1.b** shows all the different combinations of the θ , λ_1 , α and β used in for the simulation. Scenario codes *S1* (θ =0.001, λ_1 =0.001), *S2* (θ =0.001, λ_1 =0.01), *S3*(θ =0.001, λ_1 =0.1), *S4*(θ =0.01, λ_1 =0.001), *S5*(θ =0.01, λ_1 =0.01), *S6*(θ =0.01, λ_1 =0.1), *S7*(θ =0.1, λ_1 =0.001), *S8*(θ =0.1, λ_1 =0.01), *S9*(θ =0.1, λ_1 =0.1), *S10*(θ =1, λ_1 =0.001), *S11*(θ =1, λ_1 =0.01) and *S12*(θ =0.1, λ_1 =0.1) have corresponding four update settings of (α , β) ={(1,0), (0.5,0.5), (0.67,0.33), (0.9,0.1)} each, which is denoted by subscript 1,2,3 and 4 respectively. For example $S2_3$ has θ =0.001, λ_1 =0.01, α =0.5 and β **=**0.5 as the scenario parameters.

In this section, we compare the convergence results, which is a plot of $\log(F(w_{epoch}))$ value at each *epoch* of the iRLS-BFGS algorithm with L_1 regularization (where, the L_1 regularization parameter λ_1 is as per **Table 5.1.b**) as seen in **Figure 5.1.b.1** to **Figure 5.1.b.12**.

Figure 5.1.b.1 to **Figure 5.1.b.12** shows the convergence plot of the objective function values at the *epochs* as the algorithm proceeds when the iRLS-BFGS algorithm (see Chapter 4 Algorithm 2) is applied to the RCV1 data set to minimize the binary logistic loss function given by equation (14). In the **Figure 5.1.b.1**, the best convergence is achieved by $S1_2$ setting of the **Table 5.1.b** in which (α , β **)** = (0.5, 0.5) as it has the fastest convergence that is steepest slope and is below other convergence lines with lowest loss function value. Similarly, in all the results as seen in **Figure 5.1.b.1** to **Figure 5.1.b.12,** the scenario parameter settings of (α , β**)** = (0.5, 0.5) in all the scenario setting of (*S1*$_2$, *S2*$_2$,..., *S12*$_2$) respectively have the best convergence. Therefore, we conclude that irrespective of the initial value of the *L₁* regularization parameter (λ_1) and the step-size parameter (θ), the best convergence is achieved with the update rule with decay parameter setting of (α , β**)** = (0.5, 0.5) which imply that when both the step-size and the L_1 regularization parameter decay at the rate of the order $\frac{1}{\sqrt{k}}$, the best convergence is achieved.

In the **Figure 5.1.b.1** to **Figure 5.1.b.12**, another interesting observation is that with the first update setting which had scenario codes Sj_1 for all *j*=1, 2, …, 12. These scenarios resulted in slowest convergence and comparatively highest loss function value. The first update setting has the update parameters (α , β) = (1, 0) in the drop rates as mentioned in Section 4.1.3. This update setting has comparatively a faster decay rate in the step-size as α =1 and no drop in the L_1 regularization parameter as β **=**0. For example, in the **Figure 5.1.b.1** the scenario code *S1*$_1$ has the worst convergence speed comparatively. Similarly in all the results as seen in **Figure 5.1.b.1** to **Figure 5.1.b.12,** the scenario parameter settings where there is no drop in the L_1 regularization parameter (*S1*$_1$, *S2*$_1$,..., *S12*$_1$) have worst convergence results. Therefore, we conclude that the performance of the iRLS-BFGS algorithm with iterative L_1 regularization parameter is generally better than the classical RLS-BFGS algorithm with constant *L1* regularization parameter.

5.1.c Support Vector Machine

Here we train a SVM hyperplane for classification using the training dataset which has *N* random input-output pairs, $\xi_i = \{(x_{i,} z_i)\}_{i=1}^N$ in which $x_i \in \mathbb{R}^n$ is the *i*th binary input vector and $z_i \in \{-1,1\}$ is the *i*th true output for all $i = 1, 2, ..., N$.

i. **Objective function:** At each iteration of the algorithm we evaluate the regularized objective function in the problem (3) which is given by,

$$F(w; x, z, \lambda_{1k}, \lambda_2) = \frac{1}{N}\sum_{i=1}^{N} f(w, x_i, z_i) + \frac{1}{2}\lambda_2 \|w\|_2^2 + \lambda_{1k}\|w\|_1, \qquad (16)$$

where $f(w, x_i, z_i) = \max\{0, 1 - z_i x_i^T w\} : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ is the hinge loss function, $w \in \mathbb{R}^n$ is the decision variable vector, $x_i \in \mathbb{R}^n$ is the *i*th binary input vector, $z_i \in \mathbb{R}$, $z_i = \{-1,1\}$ is the *i*th true output for all $i = 1, 2, ..., N$ and *n* is the dimensionality of parameter *w* and input *x* vectors. Also, λ_2 is the constant *L2* regularization parameter, here λ_2 =0.1 and $\lambda_{1k} = \frac{\lambda_1}{k^\beta}$ is the iterative *L1* regularization parameter which decreases the influence of initial *L1* regularization parameter λ_1 on the objective function as the algorithm proceeds ($k = 1, 2, ..., t$). The corresponding regularized stochastic logistic gradient at the *k*th iteration is estimated by subgradient as,

$$\nabla F(w; x, z, \lambda_{1k}, \lambda_2) \simeq -z_i \frac{1 + sign(z_i x_i^T w)}{2} x_i + \lambda_{1k} sign(w) + \frac{\lambda_2}{2} w, \qquad (17)$$

where *sign(.)* is the sign operator.

ii **Concluding remarks:** To foster the claim of the best update rules given by $\lambda_{1k} = \frac{\lambda_1}{k^\beta}$ and $\alpha_k = \frac{\theta}{k^\alpha}$ we again seek the best choices of the parameters λ_1, β, θ and α by comparing the convergence properties of the iterative L_1 Regularized Limited memory Stochastic BFGS (iRLS-BFGS) algorithm with the classical L_1 Regularized Limited memory Stochastic BFGS (cRLS-BFGS) algorithm using the scenario approach. **Table 5.1.c** shows all the scenario codes with combination of algorithm parameters.

- *Scenario codes and parameters*: **Table 5.1.c** shows the scenario codes and corresponding values of θ, λ_1, α and β used in the experiment.

Table 5.1.c

	No.	Scenario code	α	β
θ =0.001 λ_1 =0.001	1	$S1_1$	1	0
	2	$S1_2$	0.5	0.5
	3	$S1_3$	0.67	0.33
	4	$S1_4$	0.9	0.1
θ =0.001 λ_1 = 0.01	5	$S2_1$	1	0
	6	$S2_2$	0.5	0.5
	7	$S2_3$	0.67	0.33
	8	$S2_4$	0.9	0.1
θ =0.001 λ_1 = 0.1	9	$S3_1$	1	0
	10	$S3_2$	0.5	0.5
	11	$S3_3$	0.67	0.33
	12	$S3_4$	0.9	0.1
θ = 0.01 λ_1 =0.001	13	$S4_1$	1	0
	14	$S4_2$	0.5	0.5
	15	$S4_3$	0.67	0.33
	16	$S4_4$	0.9	0.1
θ = 0.01 λ_1 = 0.01	17	$S5_1$	1	0
	18	$S5_2$	0.5	0.5
	19	$S5_3$	0.67	0.33
	20	$S5_4$	0.9	0.1
θ = 0.01 λ_1 = 0.1	21	$S6_1$	1	0
	22	$S6_2$	0.5	0.5
	23	$S6_3$	0.67	0.33
	24	$S6_4$	0.9	0.1

- *Output plot*: **Figure 5.1.c.1 to Figure 5.1.c.6** shows, the convergence plots of the objective function with *L_1* regularization and scenario parameters corresponding to legend's respective values in **Table 5.1.c**.

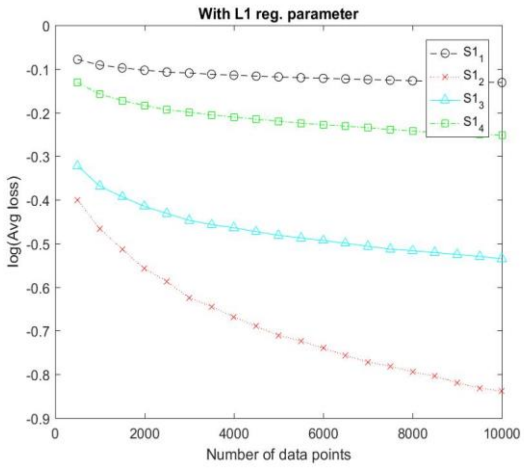


Figure 5.1.c.1

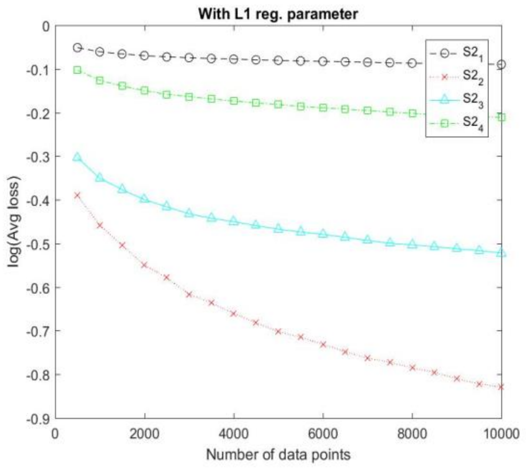


Figure 5.1.c.2

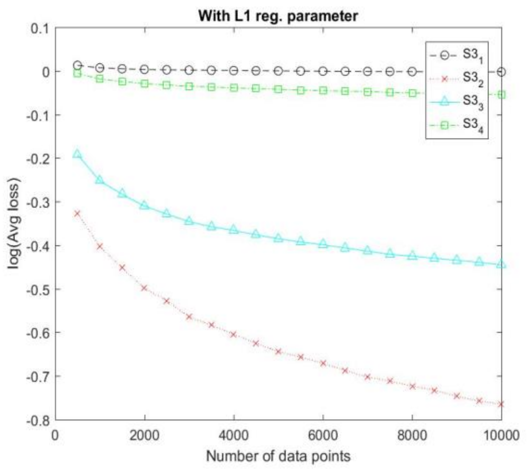


Figure 5.1.c.3

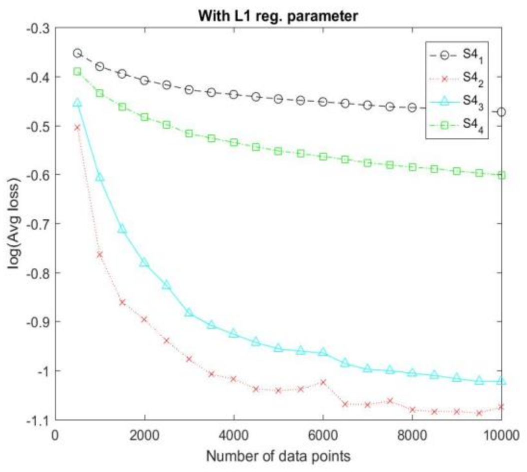


Figure 5.1.c.4

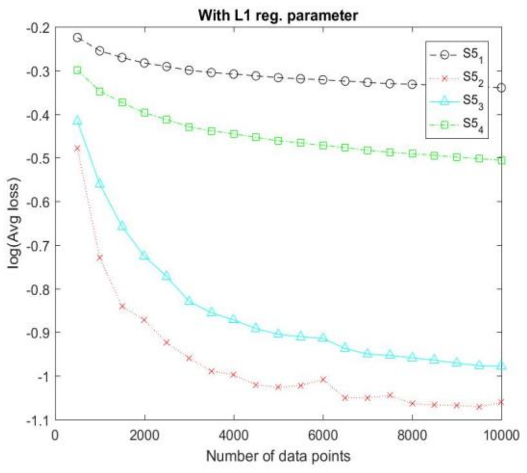


Figure 5.1.c.5

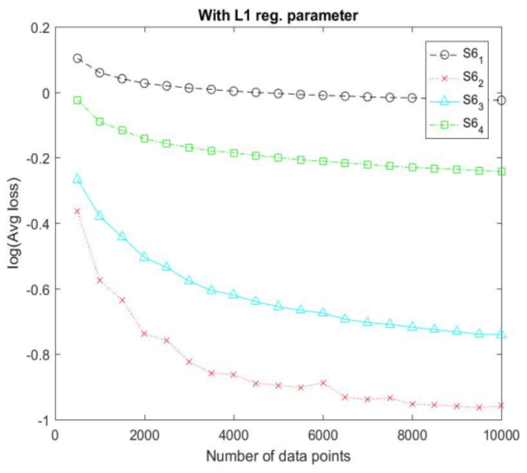


Figure 5.1.c.6

- *Interpretations*: In this experiment, the iRLS-BFGS algorithm is tested on RCV1 dataset to minimize the hinge loss function used for text classification using SVM. The iterative *L₁* regularization parameter in the iRLS-BFGS method is given as $\lambda_{1k} = \frac{\lambda_1}{k^{\beta}}$ and the step-size parameter is given as $\alpha_k = \frac{\theta}{k^{\alpha}}$ where, the initial value of step-size is given as θ={0.001,0.01}, the initial value of *L₁* regularization parameter is given as λ_1={0.001,0.01,0.1} and the drop rate parameters as (α , β **)** **=** {(1,0), (0.5,0.5), (0.67,0.33), (0.9,0.1)} for $k = 1, 2, ..., t$. **Table 5.1.c** shows all the different combinations of the θ, λ_1, α and β used in for the simulation. Scenario codes *S1* (θ =0.001, λ_1=0.001), *S2* (θ=0.001, λ_1=0.01), *S3*(θ=0.001, λ_1=0.1), *S4*(θ=0.01, λ_1 =0.001), *S5*(θ=0.01, λ_1=0.01) and *S6*(θ=0.01, λ_1=0.1) have corresponding four update settings of (α, β) ={(1,0), (0.5,0.5), (0.67,0.33), (0.9,0.1)} each, which is denoted by subscript 1,2,3 and 4 respectively. For example $S2_3$ has θ=0.001, λ_1 =0.01, α =0.5 and β **=**0.5 as the scenario parameters.

In this section, we compare the convergence results, which is a plot of $\log(F(w_{epoch}))$ value at each *epoch* of the iRLS-BFGS algorithm with L_1 regularization (where, the L_1 regularization parameter, λ_1 is as per **Table 5.1.c**) as seen in **Figure 5.1.c.1** to **Figure 5.1.c.6**.

Figure 5.1.c.1 to **Figure 5.1.c.6** shows the convergence plot of the objective function values at the *epochs* as the algorithm proceeds when the iRLS-BFGS algorithm (see Algorithm 2) is applied to the RCV1 data set to minimize the SVM loss function given by equation (16). In the **Figure 5.1.c.1**, the best convergence is achieved by $S1_2$ setting of the **Table 5.1.c** in which (α, β**)** = (0.5, 0.5) as it has the fastest convergence that is steepest slope and is below other convergence lines with lowest loss function value. Similarly, in all the results as seen in **Figure 5.1.c.1** to **Figure 5.1.c.6,** the scenario parameter settings of (α, β**)** = (0.5, 0.5) in all the scenario setting of ($S1_2$, $S2_2$,..., $S6_2$) respectively have the best convergence. Therefore, we conclude that irrespective of the initial value of the L_1 regularization parameter (λ_1) and the step-size parameter (θ), the best convergence is achieved with the update rule with decay parameter setting of (α , β) = (0.5, 0.5) which imply that when both the step-size and the L_1 regularization parameter decay at the rate of the order $\frac{1}{\sqrt{k}}$, the best convergence is achieved.

In the **Figure 5.1.c.1** to **Figure 5.1.c.6**, another interesting observation was that with the first update setting which had scenario codes Sj_1 for all j=1, 2, …, 6. These scenarios resulted in slowest convergence and comparatively highest loss function value. The first update setting has the update parameters (α , β) = (1, 0) in the drop rates as mentioned in **Table 5.1c**. This update setting has a comparatively faster decay rate in the step-size as α =1 and no drop in the L_1 regularization parameter as β =0. For example, in the **Figure 5.1.c.1** the scenario code $S1_1$ has the worst convergence comparatively. Similarly in all the results as seen in **Figure 5.1.c.1** to **Figure 5.1.c.6,** the scenario parameter settings where there is no drop in the L_1 regularization parameter ($S1_1$, $S2_1$,..., $S6_1$) have worst convergence results. Therefore, we conclude that the performance of the iRLS-BFGS algorithm with iterative L_1 regularization parameter is generally better than the classical RLS-BFGS algorithm with constant L_1 regularization parameter.

5.2 Image deblurring problem

Image deblurring process is one of the popular Big Data applications in signal processing where with the help of algorithms the noise or blur in the original image is minimized. The blur in the images occurs due to imperfections in the imaging devices (defocusing), relative motion between the camera and the object (MRI scans or CCTV footages) or atmospheric turbulence (spectral imaging). This phenomenon can be realized in the form of the linear inverse problem given by, $Xw = z + e$, where $X \in \mathbb{R}^{N\times N}$ is the blur operator ($N = n\times n$), $w \in \mathbb{R}^{N}$ is the original image vector, $z \in \mathbb{R}^{N}$ is the blurred image vector and e is the bias. The vectors w and z are formed by stacking the columns of their corresponding two dimensional images ($\mathbb{R}^{n\times n}$). The problem in which we estimate the true image w, given the blurred or noisy image z is called the image deblurring problem.

5.2.a Dataset: The image considered for this experiment is the famous black and white 'Cameraman.pgm' 8 bit image which is converted in the form of intensity matrix in Matlab 2016 using '*imread()*' command. Therefore, we obtain the true image matrix W^* of size 256×256 with pixel intensity with in the range of [7,253] in which '0' represents complete black and '256' represents complete white pixel. The columns of the matrix W^* are stacked to form the original image vector $w^* \in \mathbb{R}^{N}$ where $N = 256\times 256 = 65536$. The original image intensity matrix W^* is used to generate the blur operator matrix $X \in \mathbb{R}^{N\times N}$ where $N = 65536$ using the function *'mblur.m'* (Hansen P. C., 1997) which models a horizontal motion blur of level 10. The blurred image vector $z \in \mathbb{R}^{N}$ is obtained by the product of the blur operator (X) and the original image vector (w^*). The iRLS-BFGS algorithm is implemented to minimize the objective function for number of iterations, $t = 10000$.

5.2.b Objective function: To solve the image deblurring problem mentioned above least square approach is popularly used in the literature (Beck and Teboulle, 2009). The regularized objective function is given by,

$$F(w; x, z, \lambda_{1k}, \lambda_2) = \frac{1}{N}\sum_{i=1}^{N} f(w, x_i, z_i) + \frac{1}{2}\lambda_2 \|w\|_2^2 + \lambda_{1k}\|w\|_1, \qquad (18)$$

where $f(w, x_i, z_i) = \left\| x_i^T w - z_i \right\|^2 : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ is the least square loss function, $w \in \mathbb{R}^N$ is the estimated true image vector, $x_i \in \mathbb{R}^N$ is the *i*th row vector of the blur operator matrix $X \in \mathbb{R}^{N\times N}$, $z_i \in \mathbb{R}$ is the *i*th element of the blurred image vector $z \in \mathbb{R}^N$ for all *i=1,2,…, N*. Also, λ_2 is the constant L_2 regularization parameter, here λ_2 =0.001 and $\lambda_{1k} = \frac{\lambda_1}{k^\beta}$ is the iterative L_1 regularization parameter which decreases the influence of initial L_1 regularization parameter λ_1 on the function *F* as the algorithm proceeds ($k = 1, 2, ..., t$).

In this application we evaluate the batch gradient which is the average of the gradients evaluated using a small batch of user specified size of *M* data point rows instead of entire matrix. The corresponding regularized stochastic batch gradient at *k*th iteration is evaluated as,

$$\nabla F(w; x_i, z_i, \lambda_{1k}, \lambda_2) = \frac{1}{M} \sum_{i=1}^{M} 2(x_i^T w - b_i) x_i + \lambda_{1k} sign(w) + \frac{\lambda_2}{2} w, \qquad (19)$$

where *sign(.)* is the sign operator.

5.2.c Concluding remark: Our goal is to study the performance of the iRLS-BFGS algorithm with the update rules of the L_1 regularization parameter to be $\frac{\lambda_1}{\sqrt{k}}$ and the step size parameter to be $\frac{\theta}{\sqrt{k}}$ as suggested by our findings in the previous section. Therefore, we implement the developed iRLS-BFGS algorithm to deblur the 'cameraman' image with different values of the initial L_1 regularization parameter λ_1 and step size parameter θ. Later, we compare the performance of the iRLS-BFGS algorithm (iterative L_1

regularization parameter) with the cRLS-BFGS algorithm (constant L_1 regularization parameter) with the same initial regularization and step length values for gradient batch size *M=5000*.

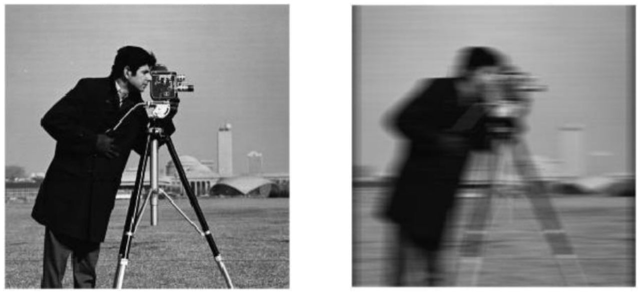
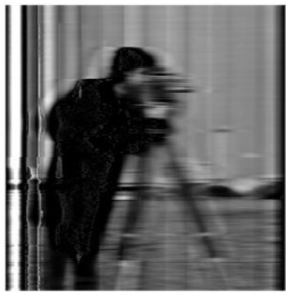
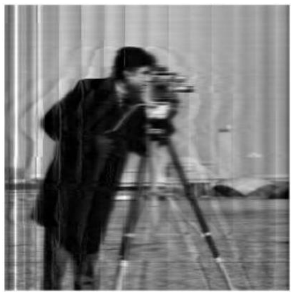
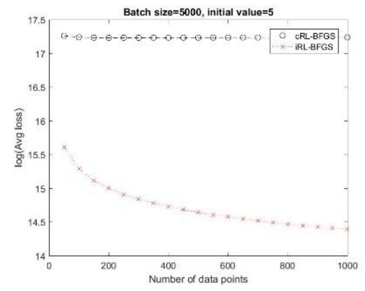


Figure 5.2.a ***Original image (left) and the blurred image (right).***

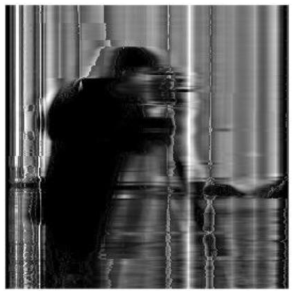
$\alpha_k = \frac{5}{k}, \lambda_1 = 5$ $\alpha_k = \frac{5}{\sqrt{k}}, \lambda_{1k} = \frac{5}{\sqrt{k}}$

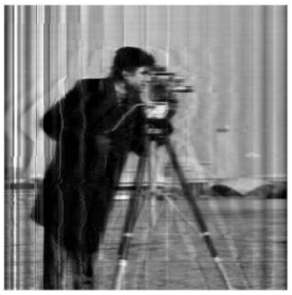


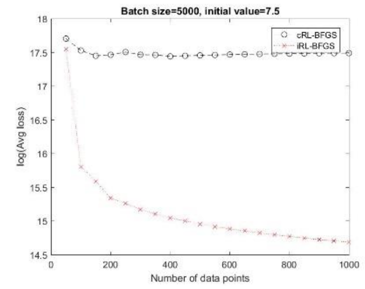




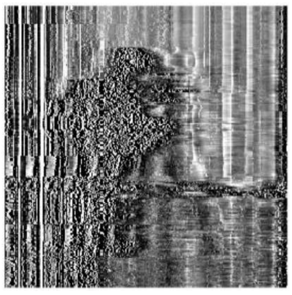
$\alpha_k = \frac{7.5}{k}, \lambda_1 = 7.5$ $\alpha_k = \frac{7.5}{\sqrt{k}}, \lambda_{1k} = \frac{7.5}{\sqrt{k}}$

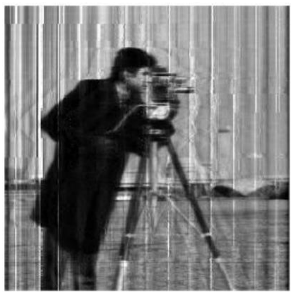






$\alpha_k = \frac{15}{k}, \lambda_1 = 15$ $\alpha_k = \frac{15}{\sqrt{k}}, \lambda_{1k} = \frac{15}{\sqrt{k}}$





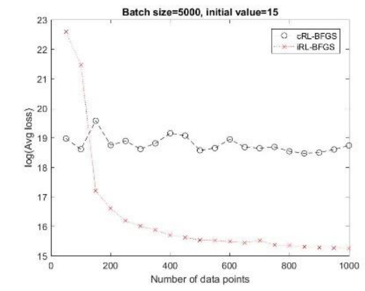


Figure 5.2.b ***Deblurred image outputs using cRLS-BFGS (left), iRLS-BFGS (middle) and convergence plots (right).***

Figure 5.2.a shows the original cameraman image and the blurred or noisy image which is to be deblurred. **Figure 5.2.b** shows the output of the deblurred images and convergence rate comparision of the cRLS-BFGS and the iRLS-BFGS for three different random initial values of $\{(\lambda_1, \theta)\}$ = {*(5, 5), (7.5, 7.5),* (*15, 15)*} where α_k is the step size and λ_{1k} is the *L₁* regularization parameter. Note that the initial *L₁* regularization parameter λ_1 remains constant in the cRLS-BFGS and the step size parameter decays at the rate of $\frac{\theta}{k}$, while, in the iRLS-BGFS the decay rate of the *L₁* regularization parameter is $\frac{\lambda_1}{\sqrt{k}}$ and the step size parameter is $\frac{\theta}{\sqrt{k}}$.

Figure 5.2.b shows that the images produced by the iterative *L₁* regularization in the iRLS-BFGS algorithm are far better than the constant *L₁* regularization as in the cRLS-BFGS algorithm. It is also observed from the convergence plots that the convergence rate of the iRLS-BFGS as indicated by red color is consistently better than the convergence rate of the cRLS-BFGS as indicated by black color. Therefore, the results support our claim that the performance of the iRLS-BFGS algorithm with iterative *L₁* regularization parameter is generally better than the cRLS-BFGS algorithm with constant *L₁* regularization parameter.

REFERENCES

Beck A. and Teboulle M. (2009), "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems", *Society for Industrial and Applied Mathematics Journal*, Imaging sciences, Vol. 2, No. 1, pp. 183–202.

Bottou L. (2010), "Large-Scale Machine Learning with Stochastic Gradient Descent", *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, 177–187, Edited by Yves Lechevallier and Gilbert Saporta, Paris, France, *Springer*.

Bottou L., Curtis F. E. and Nocedal J. (2016), "Optimization Methods for Large-Scale Machine Learning", *arXiv:1606.04838.*

Boyd S. and Vandenberghe L. (2004), "Convex Optimization*," Cambridge University Press.*

Byrd R. H., Hansen S. L., Nocedal J., and Singer Y. (2015), "A stochastic Quasi-Newton method for large-scale optimization," arXiv:1401.7020v2 [math.OC].

Fletcher R.(1987), "Practical Methods of Optimization", *Wiley, second edition*.

Friedlander M. P. & Tseng P. (2007), "Exact regularization of convex programs", *Society for Industrial and Applied Mathematics Journal*. Optim. 18, 1326-1350.

Hansen P. C. (1997), "Regularization Tools Version 4.1 for Matlab 7.3", *Numerical Algorithms*, link: http://www2.compute.dtu.dk/~pcha/Regutools/mblur.m.

Kiefer J., & Wolfowitz J. (1952), "Stochastic estimation of the maximum of a regression function", *Annals of Mathematical Statistics*, 23(3), 462–466.

Kleywegt A.J., Shapiro A., and Homem-de-Mello T., (2002), "The sample average approximation method for stochastic discrete optimization", *SIAM J. Optim.,* 12, pp. 479–502.

Lewis D. D., Yang Y., Rose T. G., and Li F. (2004), "Rcv1: A new benchmark collection for text categorization research", *The Journal of Machine Learning Research,* 5:361-397, 2004.

Lee J., Recht B., Salakhutdinov R., Srebro N. (2010), "Practical Large-Scale Optimization for Max-Norm Regularization", part of *Advances in Neural Information Processing Systems 23.*

Liu, D.C. & Nocedal J. (1989), "On the limited memory BFGS method for large scale optimization", *J. Mathematical Programming* 45: 503. doi:10.1007/BF01589116.

Lucchi A., McWilliams B. and Hofmann T. (2015) "A Variance Reduced Stochastic Newton Method", arXiv:1503.08316.

Mokhtari A. and Ribeiro A. (2014), "RES: regularized stochastic BFGS algorithm," *IEEE Transactions on Signal Processing*, vol. 62, no. 23, pp. 6089–6104.

Nemirovski A., Juditsky A., Lan G., and Shapiro A. (2009), "Robust stochastic approximation approach to stochastic programming", *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1574–1609.

Nocedal J. and Wright S. (1999), "Numerical Optimization", *Springer New York*, 2nd edition.

Robbins H. and Monro S. (1951), "A stochastic approximation method," *Ann. Math. Statistics*, vol. 22, pp. 400–407.

Schraudolph N. N., Yu J., and Günter S. (2007), "A Stochastic Quasi-Newton Method for Online Convex Optimization", *In Proc. 11th Intl. Conf. Artificial Intelligence and Statistics (AIstats)*, pp. 436–443, *Journal of Machine Learning Research,* San Juan, Puerto Rico.

Shapiro A. (2003), “Monte Carlo sampling methods,” *in Handbook in Operations Research and Management Science*. Amsterdam: Elsevier Science, vol. 10, pp. 353–426.

Shi J., Yin W., Osher S., Sajda P. (2010), “A Fast Hybrid Algorithm for Large-Scale ℓ1 Regularized Logistic Regression”, *Journal of Machine Learning Research.*

Spall, J. C. (1992), “Multivariate stochastic approximation using a simultaneous perturbation gradient approximation”, *IEEE Transactions on Automatic Control*, 37(3), 332–341.

Tibshirani, R. (1996), “Regression Shrinkage and Selection via the lasso”, *Journal of the Royal Statistical Society. Series B (methodological) 58 (1). Wiley*: 267–88.

Yousefian F., Nedich A., and Shanbhag U.V. (2012), “On Stochastic Gradient and Subgradient Methods with Adaptive Steplength sequences”, *Automatica* 48 (1) 56 -67, 2012.

Yousefian F., Nedich A., and Shanbhag U.V. (2016), “Stochastic quasi-Newton methods for non-strongly convex problems: convergence and rate analysis”, *IEEE Conference on Decision and Control.* 2016 IEEE 55th Conference, *4496-450.*

Yousefian F., Nedich A., and Shanbhag U.V. (2017), “On smoothing, regularization and averaging in stochastic approximation methods for stochastic variational inequalities”, *Mathematical Programming*

VITA

VANDAN PATEL

Candidate for the Degree of

Master of Science

Thesis: AN ITERATIVE L_1 REGULARIZED LIMITED MEMORY STOCHASTIC BFGS ALGORITHM AND NUMERICAL EXPERIMENTS FOR BIG DATA APPLICATIONS

Major Field: Industrial Engineering & Management

Biographical:
Personal Information:
Born in Nashik, Maharashtra, India, son of Mr. Rasik Patel and Mrs. Laxmi Patel, brother of Maharsh Patel and husband of Mrs. Rucha Patel.

Education:
Completed the requirements for the Master of Science in Industrial Engineering and Management at Oklahoma State University, Stillwater, Oklahoma in July 2017.

Received Bachelors of Engineering in Mechanical Engineering at Mumbai University, Mumbai, Maharashtra, India in 2014.

Experience:
Research Assistant under Dr. Farzad Yousefian at the Department of Industrial Engineering and Management at Oklahoma State University, Stillwater, OK, from May, 2016 to August, 2017.

Graduate Teaching Assistant at the Department of Industrial Engineering and Management at Oklahoma State University, Stillwater, OK, from August, 2016 to May, 2017.

Professional Memberships:
Member of Alpha Pi Mu, Honors Society Industrial Engineering and Management

Vice President of Institute of Operation Research and Management Sciences (INFORMS), Oklahoma State University Student Organization (OSUSO).

Graduate coordinator of Institute of Industrial and Systems Engineers (IISE), OSUSO.