Summer Research Fellowship 2011

8 week report

Name of the candidate : Hirak Jyoti Kashyap

Application Registration no. : ENGS2758

Date of Joining : June 1, 2011

Date of Completion : July 28, 2011

Name of Guide : Dr Onkar Jayant Dabeer

Guide's Institution : School of Technology and Computer Science, TIFR, Mumbai

Project Title : Distributed Linear Regression Under Privacy Constraints

Address with pin code to which the certificate could be sent:

Hirak Jyoti Kashyap
B.Tech(7th sem)
Department of Computer Science and Engineering, Tezpur University
Tezpur(Assam)
Pin: 784028

(For Office Use)

Candidate Name:	Fellowship Amount:
Student: Teacher:	Deduction:
Guide's Name:	Amount to be paid:
KVPY Fellow: INSPIRE Fellow:	A/C holder name:
Others:	

Distributed Linear Regression Under Privacy Constraints

Abstract

Regression analysis is used to estimate the unknown effect of changing one variable over another. While running a linear regression, it is assumed that there is a linear relationship between regressand and regressor variables and this relationship is additive. Various regression algorithms are used for modeling a data set using linear functions as well as estimating the unknown model parameters from data. The Recursive Least Square(RLS) algorithm is one of the most popular regression analysis method with extremely fast convergence rate. But a regression analysis method does not perform well if the analysis is performed based on a set of observations representing only a small portion of independent input variables of the system. If a set of agents collaboratively read the observations of a system, then agents can also perform regression analysis collaboratively, i.e. by sharing all their readings and then forming an estimate. But in some systems, the agents are unwilling to share their observations due to privacy reasons, but do not mind sharing estimates with others for getting similar estimates in return. To this end, a RLS based distributed regression algorithm is developed for wireless sensor networks (WSNs) whereby sensors exchange local estimates with one-hop neighbors to consent on the network-wide estimates adaptively and thereby converging to the estimates formed by a regression analysis based on all observations. The proposed algorithm has been obtained through recasting of the weighted linear least-squares cost into a separable form. Numerical simulations demonstrate that for higher number of 1 hop neighbors, this RLS based distributed regression algorithm implementation by an agent performs similar to the RLS algorithm based on full information.

I. Introduction:

Linear regression is an approach to modeling the relationship between a scalar variable y and one or more variables denoted X. In linear regression, data are modeled using linear functions, and unknown model parameters are estimated from data. Such models are called linear models. In other words, linear regression refers to a model in which the conditional mean of y given the value of X is an affine function of X. If the goal is prediction, or forecasting, linear regression can be used to fit a predictive model to an observed data set of y and X values. After developing such a model, an input vector $X = [x_1, ..., x_p]^T \in \mathbb{R}^p$ is given and the goal is to predict the real-valued scalar response y. A linear approximation to the regression function is adopted for this, namely $f(X) = X^T\beta + \varepsilon$, where $\beta = [\beta_1, ..., \beta_p] \in \mathbb{R}^p$ is the vector of model coefficients and ε, the intercept is also known as "disturbance term". Moreover linear regression analysis can also be applied to quantify the strength of relationship between y and any x_j from the set $\{x_1, x_2, ..., x_p\}$, to assess which x_j may have no relationship with y at all, and to identify which subsets of the $\{x_1, x_2, ..., x_p\}$ contain redundant information about y, thus once one of them is known, the others are no longer informative.

Methods like Least Mean Square(LMS) and Recursive Least Squares(RLS) are popularly used for estimating the model parameters $\{\beta, \varepsilon\}$ from a given training dataset $\{y_k, X_k\}_{k=1}^{K} = \{Y, \mathbf{X}\}$. Basically, the LMS algorithm is a stochastic gradient algorithm, which means that the gradient of the error performance surface with respect to the free parameter vector changes randomly from one iteration to the next. The LMS algorithm attains simplicity of implementation by using instantaneous estimates of (1) the autocorrelation matrix of the input signal vector, and (2) the cross-correlation vector between the input vector and the desired response. But, the LMS algorithm has two major drawbacks: slow rate of convergence and sensitivity to the eigenvalue spread (i.e., the ratio of the largest eigenvalue to the smallest eigenvalue) of the correlation matrix of the input signal vector.

However, the Recursive Least Squares (RLS) algorithm utilizes continuously updated estimates of autocorrelation matrix as well as cross-correlation vector and as a result; the RLS algorithm exhibits the following properties:

1. Rate of convergence is of a magnitude faster than the LMS algorithm.
2. Rate of convergence is invariant to the eigenvalue spread of the correlation matrix of the input vector.

These desirable properties are however attained at the cost of increased computational complexity. Given the set of input samples $\{X_1, X_2, ..., X_k\}$

and the set of desired response $\{y_1, y_2, ..., y_k\}$, where output is estimated according to:

$$\hat{y_k} = (\hat{\beta_k})^T X_k \qquad (1)$$

The RLS algorithm finds recursively in time the parameters $\hat{\beta}_k = [\hat{\beta}_1, \hat{\beta}_2, ..., \hat{\beta}_p]_k \in \mathbb{R}^p$ such as to minimize the sum of the error squares

$$\varepsilon_n = \sum_{i=1ton} \lambda^{n-i}[e_i^2] \qquad (2)$$

Where $e_i = y_i - \hat{y}_i$ is the error signal and λ^{n-i} is the forgetting factor or weighing factor that reduces the influence of old data.

This training data set $\{y_k, X_k\}_{k=1}^K$ is typically assumed to be centrally available, so that it can be jointly processed to obtain the model parameters. However, collecting all data in a central location may be prohibitive in certain applications. Distributed linear regression problems commonly arise with wireless sensor networks (WSNs) with sensor nodes deployed to perform collaborative estimation tasks, where data are inherently scattered across a large geographical area. This may be due to power constraints imposed on the individual nodes as they are battery operated and transferring all collected data to central coordinator(possibly located far away) consumes lot of energy. In lieu of the central coordinator, the agents themselves process their locally available training sets. The so-termed in-network processing introduces additional algorithmic challenges, as the information is not centrally available and can not be flooded for massive dissemination. Absence of hierarchy and the purely decentralized nature of in-network processing dictate that the collection of local (per agent) estimates should consent to the global solution, namely, the parameter estimates that would be obtained if the entire data were centrally available.

Again in some real life applications such as the Internet or collaborative inter-laboratory studies imposes data privacy constraints. In such cases, agents providing private data for the purpose of fitting a linear model may not be willing to share their training data and they may like to share their learning results only. Decentralized linear regression is based on successive refinements of local model parameter estimates maintained at individual agents. Each iteration of this fully distributed algorithm comprises of i) a communication step where agents exchange messages with neighbors and ii) an update step where each agent uses this information to refine its estimate. If privacy constraint is imposed on the nodes such that they can not share their training data, then in first step of each iteration, the agents can exchange their learning results and use them for updating their estimates. Here in this report, a reformulation of the RLS algorithm has been done for the purpose of fitting a linear model where data are scattered among different agents and agents can not share their collected data due to privacy constraints. It is assumed that the agents are empowered with signal processing tools that enable low-cost estimation of stationary signals as well as reduced-complexity tracking of non-stationary processes.

Outline of the report is as follows. In Section II, the basic objective of the work is discussed in view of distributed implementation of the new algorithm. In section III, a stepwise approach has been adopted for formulating the RLS based distributed regression algorithm while maintaining the privacy constraints imposed on the agents. In section III.A, we develop the RLS based semi-distributed regression method with privacy constraints, where a central coordinator forms the estimate using the local estimates of all the agents, while maintaining their data privacy. In III.B, we reformulate it into a fully separable form without any coordinator, where the agents form estimates of their own. Numerical tests with simulated data sets and their results are presented in section IV. In section V, we discuss various factors that influences the performance of the proposed distributed regression algorithm with the help of some experimental results and concluding remarks are provided in section VI.

II. Problem Statement:

Consider a network of N agents that are capable of performing some local computations, as well as exchanging messages among neighbors. We are not concerned about the details of an agent and we will consider it as an abstract entity only, possibly representing a sensor node in a WSN, a router monitoring Internet traffic, a hospital or laboratory involved in e.g., a medical study or a sensing CR from a next-generation mobile communications technology. The network is modeled as an undirected graph $G(\eta, E)$, where the set of vertices $\eta = \{1, 2, ..., N\}$ corresponds to the agents, and the edges E represent pairs of agents that can communicate. Let us consider $\aleph_n$ denotes the neighboring set of agent n , where $\aleph_n \subset \{1, 2, ..., N\}$ and an agent n can communicate only with its neighbors $n' \in \aleph_n$. The symmetric adjacency matrix $E \in \mathbb{R}^{NXN}$ with entries $E_{ij} = 1$ if $j \in \aleph_i$ and $E_{ij} = 0$ otherwise stores the global connectivity information.

Let agent n observes the entries of $\mathbf{X}$ from a set $I_n \subset \{1, 2, ..., p\}$. Let X_k^n denotes the predictor variables observed by agent n at any step k. So agent n can run a RLS based on observations $\{y_k, X_k^n\}_{\forall k \leq K}$, i.e. for K iterations. If RLS estimate of agent n at step k is $\hat{\beta_k^n}$, then it's prediction of y_k will be:

$$\hat{y}_k^n = (\hat{\beta_k^n})^T X_k^n$$

But $\hat{y}_k^n$ may significantly differ from actual response y_k when size of I_n, i.e. number of observed independent regressor variables is very small in comparison to p. In most of the cases it fails to converge to the global estimate computed from all the observations. In order to improve their estimates without sharing observations, the agents will try to exchange their local predictions with their neighbours in absence of a central agent. Then using these information from the neighbors, the agents will again try to predict the y_k values such that the new estimates converges to the global solution.

The objective of this work is to develop and analyze in terms of convergence, a distributed linear regression method using the RLS algorithm based on in-network processing of the locally available training data. The algorithms should exhibit following three characteristics:

i) Convergence to the global solution in (1)

ii) Per agent processing should be simple and efficient

iii) Communications among agents should be confined to the single-hop neighbors and avoids exchange of elements from the different training sets.

III. Distributed Linear Regression using RLS:

In this section, we will try to construct the Distributed Linear Regression method and the aim will be to recast the Recursive Least Square algorithm into a separable form that facilitates distributed implementation. Moreover the distributed regression algorithm should also obey the privacy constraint, i.e. observed data from different training data sets must not be exchanged among the agents. We will follow a stepwise approach to formulate the distributed linear regression algorithm. In first step, we will discuss the privacy constraint, but not the fully distributed implementation. In this step we will have a central coordinator and the agents will share their learning results with the central coordinator for computation of the global estimate by the central coordinator. In next step, we will introduce the fully distributed case, where there will be no central coordinator. Here the concept of neighborhood will arise as the agents will share their learning results only with their 1 hop neighbors. The agents compute their estimates by linear combination of their local observations as well as the estimates obtained from their neighbors.

A. The Case of Central Coordinator

Let us consider, several agents partially observe the entries of $\mathbf{X}$ and are reluctant to share their data for privacy reasons. These agents fit a model of their own by applying the RLS algorithm on the partial training data set and then send their local estimates to the central coordinator. The central coordinator upon receiving local estimates by all the agents, runs the RLS algorithm again on these local estimates and calculates the global estimate.

Suppose $\{y_k, X_k^n\}$ represents the set of observations available to agent n at time step k and the co-efficient estimate of agent n at step k is $\hat{\beta_k^n}$ and the same after step K is $\hat{\beta_K^n}$. Now assuming that $\hat{\beta_K^n}$ is more precise than $\{\hat{\beta_k^n}\}_{\forall k<K}$ values, using $\hat{\beta_K^n}$ we can make a better prediction of the y_k's for

all recent data:

$$\hat{y}_k^n = (\hat{\beta_K^n})^T X_k^n \tag{3}$$

Now for improving the estimates without sharing observations, the agents will transfer their local estimates to the central agent. The central agent upon receiving local estimates from all the agents, again executes the RLS algorithm using those local estimates and computes the global estimate.

$$\hat{y}_k = \sum_{n=1 to N} \hat{\omega}_k^n \hat{y}_k^n \tag{4}$$

where the weights $\hat{\omega}_k^n$ are obtained by running another RLS algorithm based on observations $\{y_k, \hat{y}_k^1, \hat{y}_k^2, ..., \hat{y}_k^N\}_{k=1}^K$.

ALGORITHM

```
Given observations $\{X_1, X_2, ..., X_K\}$ and $\{y_1, y_2, ..., y_K\}$ and let $X_k^n$ denotes
the regressor variables observed by agent $n$ at step $k$.
I. For each agent $n = 1, 2, ..., N$
  1. Initialize all elements of $\beta_0^n$ vector as 0 and $P = \delta I$
  2. For each time instant $k = 1, 2, ..., K$ compute
    a. Compute $Z = X_k^n \times P$
    b. Compute $\pi = Z \div (\lambda + (X_k^n \times Z))$
    c. Compute $\hat{y}_k^n = \beta_{k-1}^n \times X_k^n$ and calculate the error $\varepsilon_k = y_k - \hat{y}_k^n$
    d. Calculate $\beta_k^n = \beta_{k-1}^n + \varepsilon_k * \pi$
    e. Compute $P = (P - (\pi \times Z)) \div \lambda$
  3. Now using $\beta_K^n$, recompute $\hat{y}_k^n$ values $\forall k \in \{1, .., K\}$ for betterment of
     its estimate
  4. Send $\hat{y}_k^n$ values $\forall k \in \{1, .., K\}$ to the central coordinator

II. The central agent computes the global estimate by executing the RLS
algorithm based on $\{\hat{y}_k^1, \hat{y}_k^2, ..., \hat{y}_k^N\}_{k=1}^K$ and $\{y_1, y_2, ..., y_K\}$.
  1. Initialize all elements of $\beta_0$ vector as 0 and $P = \delta I$
  2. Repeat step I.(2) using $\{\hat{y}_k^n\}_{n=1}^N$ in place of $X_k^n$ and compute the $\hat{y}_k$
     values $\forall k \in \{1, .., K\}$
```

B. A RLS Based Algorithm for Distributed Regression

In this step, we consider that we do not have a central coordinator. But instead we have a graph with scattered agents as the nodes. During each time step, an agent computes a tentative estimate by running the RLS algorithm based on its observations and sends it to its neighbors in the graph. The final estimate at any agent is formed by linear combination of its own observations and the tentative estimates obtained from its neighbors. The coefficients in the linear combination are adapted using the RLS algorithm.

Let us consider a graph where $\aleph_n$ denotes the neighbouring set of node n, where $\aleph_n \subset \{1, 2, ..., N\}$. Similar to previous step, agent n partially observes the entries of $\mathbf{X}$ from a set $I_n \subset \{1, 2, ..., p\}$ and $\{y_k, X_k^n\}$ denotes the set of variables observed by agent n at time step k. Agent n now executes the RLS algorithm based on observations $\{y_k, X_k^n\}_{\forall k \leq K}$. Let us consider the co-efficient estimate of agent n after K iterations is $\breve{\beta}_K^n$. Now assuming that $\breve{\beta}_K^n$ is more precise than $\{\breve{\beta}_k^n\}_{\forall k<K}$ values, using $\breve{\beta}_K^n$ we can tentatively make a better prediction of the y_k's for all recent data:

$$\breve{y}_k^n = (\breve{\beta}_K^n)^T X_k^n \qquad , \forall k \leq K \tag{5}$$

When estimations based on training data block is complete, instantaneous estimates of y_k's can also be sent to the neighbors. As in the previous scenario, $\breve{y}_k^n$ may significantly differ from actual response, y_k when size of I_n, i.e. number of observed predictor variables is very small in comparison to p. In most of the cases it fails to converge to the global estimate computed from all the observations. In order to improve the estimates without sharing the observations, the agents will transfer their tentative estimates to their 1 hop neighbors in the graph. The final estimate at any step at any agent is formed by a linear combination of its observations and the estimates obtained from it's neighbors. The coefficients in the linear combination are adapted using the RLS algorithm based on $\{y_k, X_k^n\}_{\forall k \leq K}$ as well as the tentative estimates received from it's neighbors, $\{\breve{y}_1^{n'}, \breve{y}_2^{n'}, ..., \breve{y}_K^{n'}\}_{\forall n' \in \aleph_n}$. So at any time step k, agent n forms an estimate

$$\hat{y}_k^n = (\hat{\beta}_k^n)^T X_k^n + \textstyle\sum_{\forall n' \in \aleph_n} \hat{\omega}_k^{n'} \breve{y}_k^{n'} \tag{6}$$

where $\hat{\beta}_k^n$ and $\hat{\omega}_k^{n'}$ are obtained by running the RLS algorithm based on observations $\{y_k, X_k^n, \{\breve{y}_{k-1}^{n'}\}_{\forall n' \in \aleph_n}\}_{\forall k \leq K}$.

ALGORITHM

Given observations $\{X_1, X_2, ..., X_K\}$ and $\{y_1, y_2, ..., y_K\}$ and let X_k^n denotes the regressor variables observed by agent n at step k.

```
I. For each agent $n = 1, 2, ..., N$
   1. Initialize all elements of $\beta_0^n$ vector as 0 and $P = \delta I$
   2. For each time instant $k = 1, 2, ..., K$ compute
      a. Compute $Z = X_k^n \times P$
      b. Compute $\pi = Z \div (\lambda + (X_k^n \times Z))$
```

```
        c. Compute $\breve{y}_k^n = \beta_{k-1}^n \times X_k^n$ and calculate the error $\varepsilon_k = y_k - \breve{y}_k^n$
        d. Calculate $\beta_k^n = \beta_{k-1}^n + \varepsilon_k * \pi$
        e. Compute $P = (P - (\pi \times Z)) \div \lambda$
    3. Now using $\beta_K^n$, recompute $\breve{y}_k^n$ values $\forall k \in \{1,..,K\}$ for betterment of
       its estimate
    4. Send $\breve{y}_k^n$ values $\forall k \in \{1,..,K\}$ to all 1 hop neighbors

II. An agent computes the global estimate after receiving estimates from
all its neighbors by running the RLS algorithm based on observations
$\{y_k, X_k^n, \{\breve{y}_{k-1}^{n'}\}_{\forall n' \in \aleph_n}\}_{\forall k \leq K}$.
    1. Initialize all elements of $\beta_0$ vector as 0 and $P = \delta I$
    2. Repeat step I.(2) using $\{X_k^n, \{\breve{y}_{k-1}^{n'}\}_{\forall n' \in \aleph_n}\}$ in place of $X_k^n$ and
       compute $\hat{y}_k^n$ values $\forall k \in \{1,..,K\}$
```

IV. Simulation Results

Here we test performance of the proposed RLS based Distributed Regression method (discussed in III.B) with the set up described below, conducting performance comparison with the central coordinator case discussed in III.A as well as with the basic RLS algorithm based on full information. Consider in a linear regression model where X_k and y_k evolve as per following state space equation:

$$X_{k+1} = AX_k + U_k$$
$$y_k = CX_k + v_k$$

Where $\{U_k\}_{k\geq 0}$ and $\{v_k\}_{k\geq 0}$ are i.i.d. gaussian variables across time with zero mean and variance 0.5. C is a vector representing the actual co-efficients of the predictor variables of the system. A is any constant between 0 and 1 times the identity matrix of rank equal to length of vector X_k.

The reformulated algorithm described in section III.A where a central coordinator forms the global estimate from the local estimates received from all agents, performs similar to the RLS algorithm implementation based on full information in terms of MSE. This is because of the fact that the final estimation by the central coordinator is based on local estimates by all the agents and hence it is based on all observations collected collaboratively by all the agents. The performance of this algorithm in terms of MSE after 100 runs is compared with that of the RLS algorithm implementation based on full information is shown in the figure below:

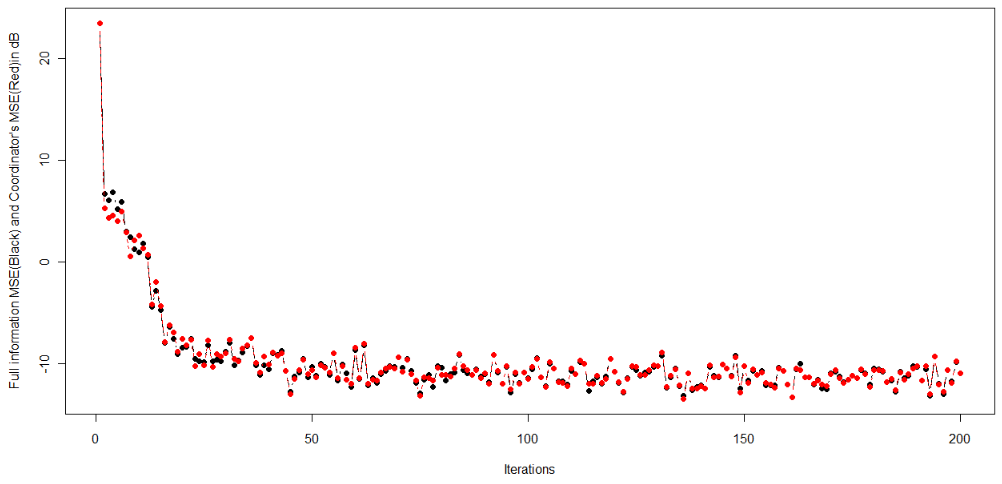

Figure 1. Comparison of MSE of the RLS implementation based on full information (in black) and MSE of the central coordinator case discussed in section III.A (in red)

Now let us look at the performance of the RLS based distributed regression method discussed in section III.B and compare it to the performance of the RLS algorithm based on full information. The set up used for the simulation of the RLS based distributed regression algorithm is described below.

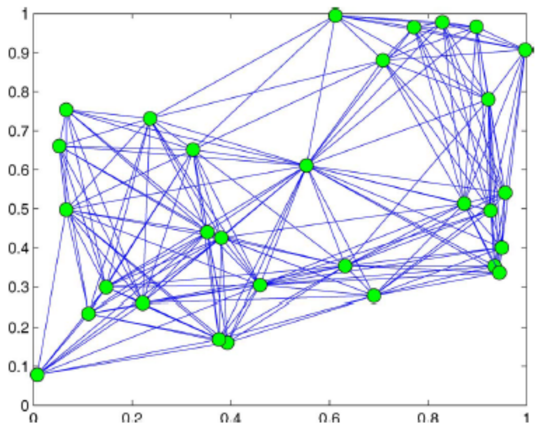

Figure 2. Random graph of 30 nodes with $P = 0.6$

Now we do not have a central coordinator. But instead we have a graph with agents as the nodes. Let us take a random graph of 30 nodes and each node is connected to every other node with some probability P. We will experiment with different values of P ranging from 0 to 1. The performance of an individual node depends on number of nodes it is connected to in the graph. The nodes with higher number of neighbors are expected to perform well, i.e. close to the performance of full information case due to more available information.

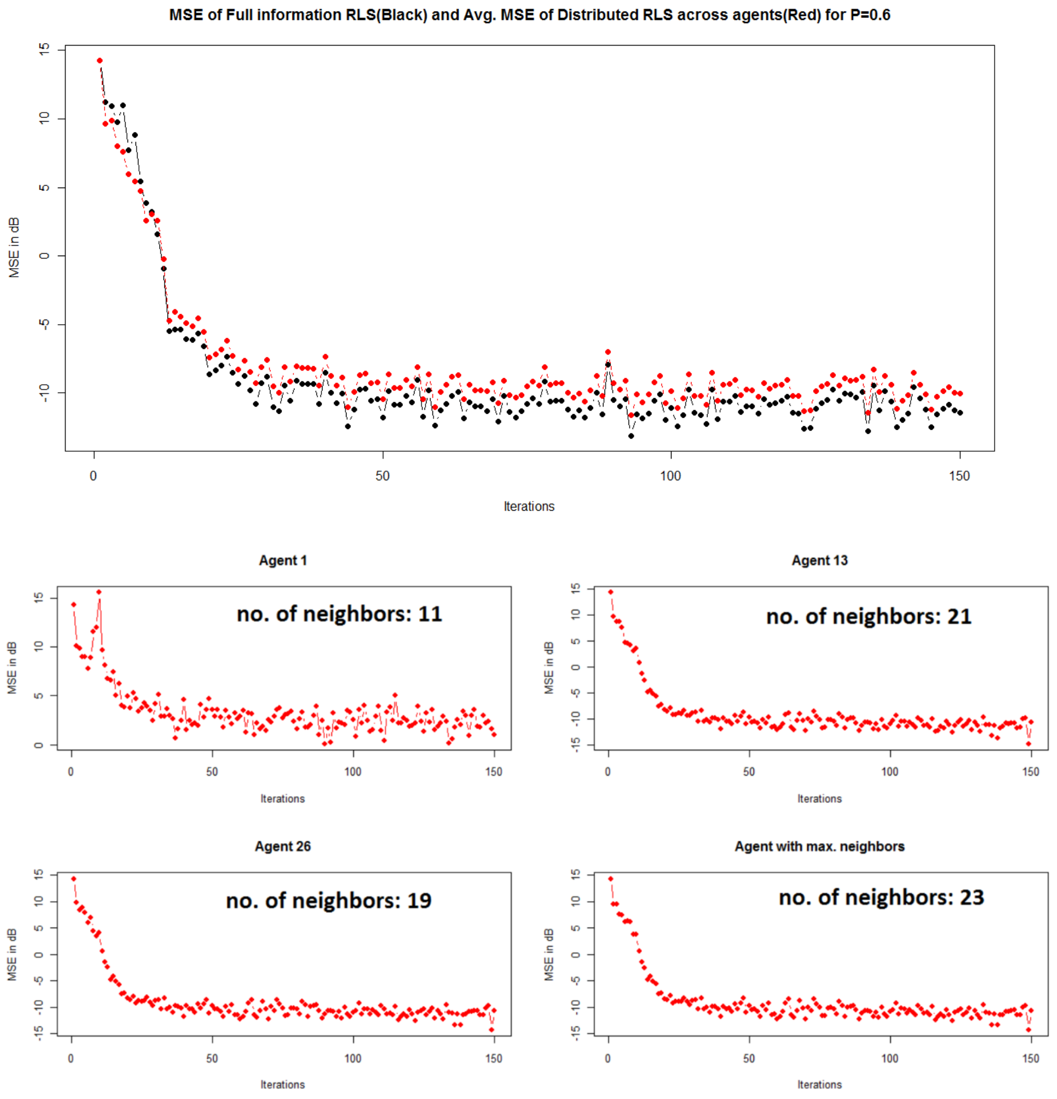

Figure 3. For P=0.6, avg. of MSE across all agents implementing the distributed regression algorithm vs. MSE of the RLS implementation on full information (top). Performance of a few agents(bottom).

The variables X_k and y_k of the linear regression model evolve as per the state space equation cited above. After running the implementation of the algorithm for 100 times, the MSE values of all the agents have been calculated. Above figures plot the average of MSE across all the agents implementing the distributed regression algorithm as well as individual performances of a few agents. MSE of the RLS implementation based on full information has also been plot alongside for the purpose of performance comparison.

It is seen that the MSE of full information case is approximately -10dB. For P equal to 0.6, the average of MSE across all the agents implementing the distributed algorithm differs from the MSE of full information case by a small margin of 1-2dB. Since it is simply an average of MSE values of all the agents, so it is highly influenced by individual performance of each of the agents. In second part of figure 3, it is seen that the performance of an agent is directly proportional to the number of its 1 hop neighbors. This is because number of readings that propagate through the implementation of an agent increases with increase in number of immediate neighbors of the agent. The node with maximum number of neighbors(23) performs almost similar to the full information case with a MSE value of approximately -10dB. Agents having 2nd highest(21) and 3rd highest(19) number of immediate neighbors also performs close to the full information case. But performance of the agent having 11 neighbors immensely differs from that of the RLS implementation on full information. Its MSE values are fluctuating between 0-5dB. The situation could be even worse with the agents having even lesser number of neighbors. This is the main reason behind the gap between performance the RLS implementation on full information and average of performances across all agents implementing the RLS based distributed regression algorithm.

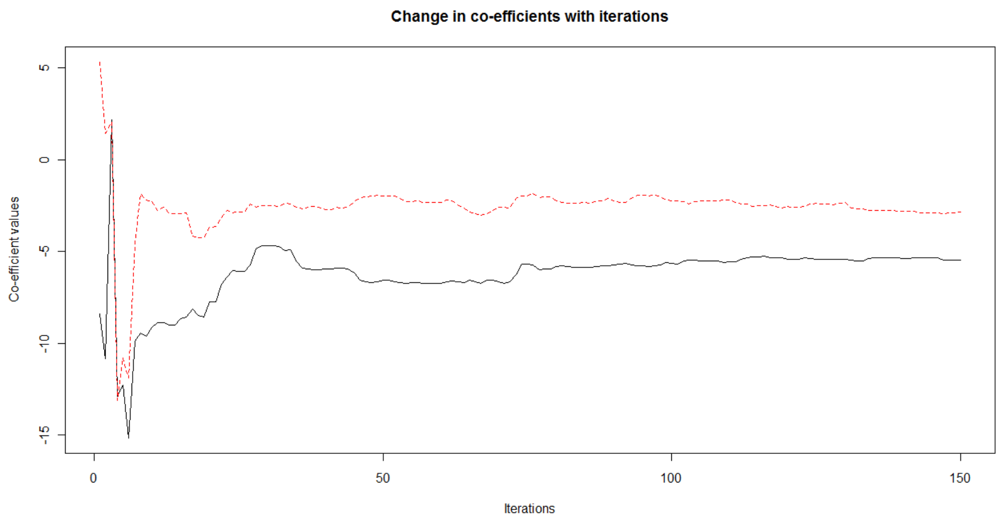

Figure 4. Change in estimates of 2 Co-efficients of an agent with iterations

V. Discussion

In the simulation results presented above, it is observed that the performance of an agent is proportional to the amount of information that propagates through the implementation of the agent. Now since the agents observe only a small portion of information, so it has to depend on the estimates provided by its neighbors. Since the neighbors form their estimates from different sets of observations, taking neighbor's estimates into consideration increases the amount of information used by the agent for forming its estimate. In our discussion, we have taken a variable P which gives the probability of two nodes being 1 hop neighbor. As P increases, the number of neighbors of the agents also increases. Hence increase in P should lead to increase in average performance across all the agents. The following plot depicts the average MSE across all the agents for different P values ranging from 0 to 1. P can also be considered as the communication range of an agent, i.e. the distance upto which a node can directly communicate with an another node. In that case also, increase in P will increase number of neighbors of the agents. Hence, in either interpretation of P, its effect on performance of the distributed regression algorithm will be same.

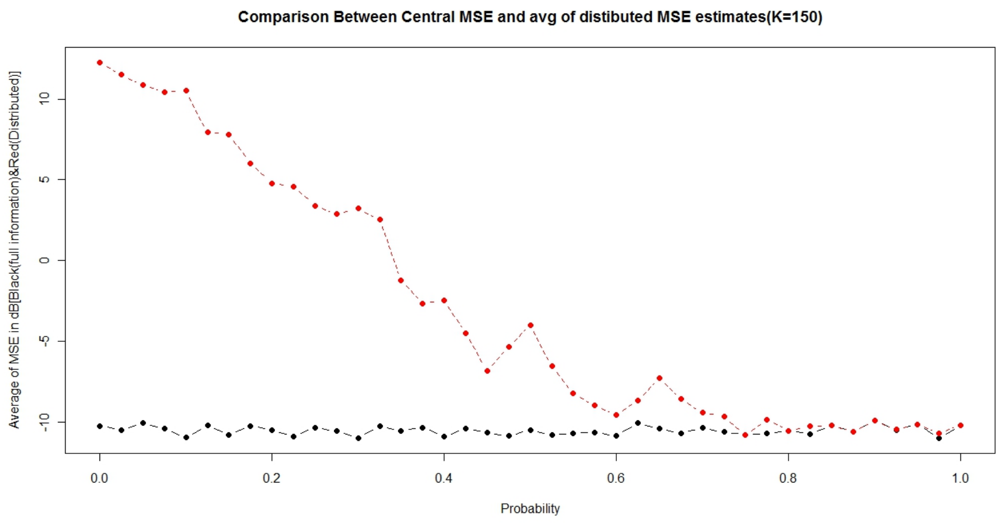

Figure 5. Performance of the RLS based distributed regression algorithm(Red) for different values of P. Performance of the RLS execution based on Full information (Black) is also plotted alongside for performance comparison at higher P values.

Now let us take a process of the form

$$X_{k+1} = AX_k + (\sqrt{1-a^2})U_k$$

Where X_0 is Gaussian with zero mean and unit variance and a is the diagonal entry of A. Now, as a comes close to 1, we approach a very smooth signal (DC in limit) and as it approaches 0, we have i.i.d. data. Performance of the distributed regression algorithm is plotted in the figure below. It is observed for higher values of a towards 1, the algorithm performs well. The performance of the distributed regression algorithm as well as the RLS implementation on full information becomes maximum when a becomes 1, i.e. when the process becomes static. But as it approaches zero, some fluctuations in performance have been observed, which is not there in full information RLS performance. Moreover, these fluctuations increases with decrease in value of a. Therefore, we can infer that predictions based on smaller amount of information are more affected by robust data than the prediction based on full information.

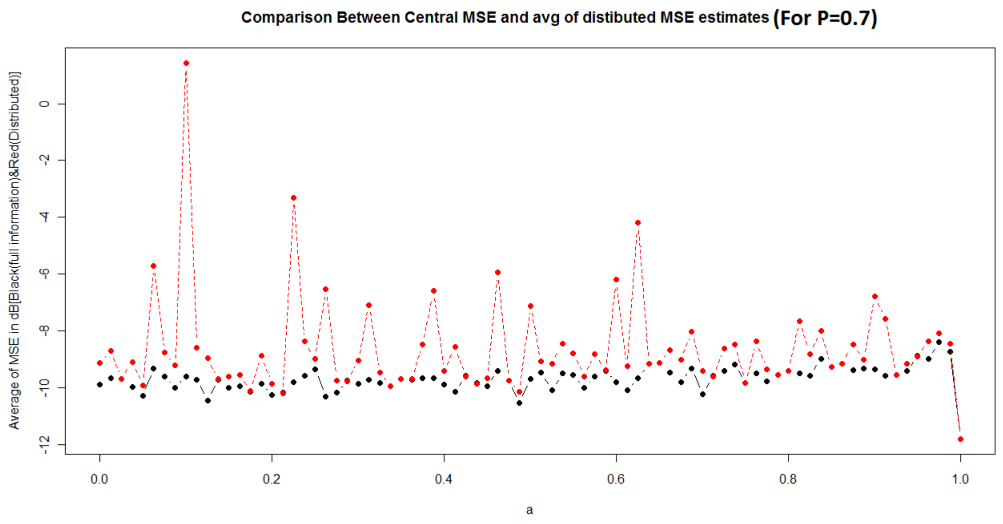

Figure 7. Performance of the RLS based distributed regression algorithm(Red) for different values of a

Now we will discuss the effect of block size K on the performance of the distributed algorithm. The distributed regression algorithm proposed here uses an estimate betterment approach, i.e. before sending the estimates to the neighbors, an agent performs an estimate betterment process. In this process, an agent re-calculates it's own responses for all recent data using the latest co-efficient estimates for purpose of betterment of it's predictions. Since co-efficient estimates converge to their actual values with iterations, therefore throughout the approach it has been considered that using latest co-efficient estimates we can obtain more accurate predictions for all recent data. Here we are using the co-efficient estimates after running the algorithm for block size number of iterations. Since co-efficient estimates become accurate with iterations, so with increase in block size, the performance of the algorithm should also increase. The following plot depicts the MSE values for different block size values ranging from 60 to 150.

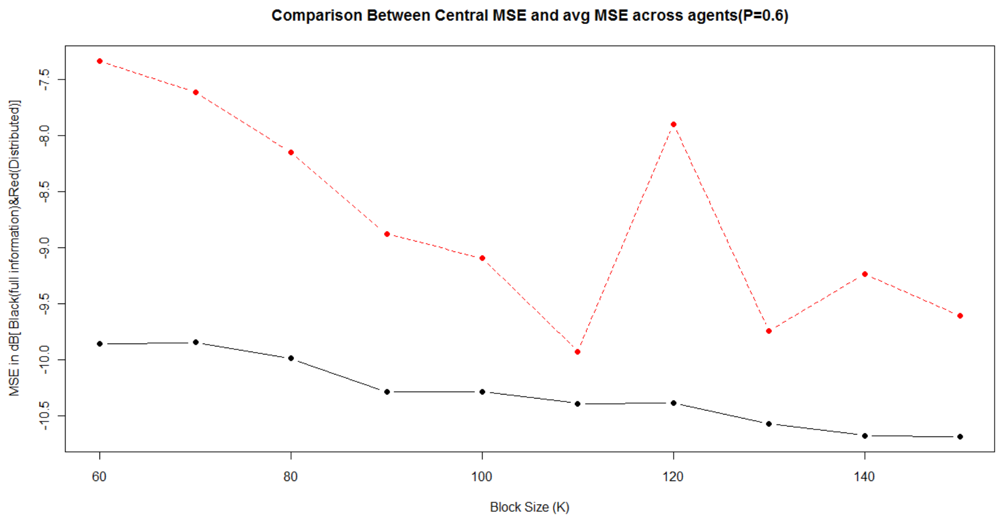

Figure 6. Performance of the RLS based distributed regression algorithm (Red) for different values of block size K

It is seen that performance of the RLS based distributed regression algorithm increases with increase in block size up to a certain level. Here average of MSE of the distributed regression algorithm implementation comes closer to MSE of full information case with increase in block size. But this increase in performance of the distributed regression algorithm stops after block size crosses a particular value. This may be due to the fact that the co-efficient estimates converges to their actual values up to a certain number of iteration only. After that the estimates almost remain unchanged (Refer to Figure 4).

VI. Conclusion

Here we developed a RLS based distributed linear regression algorithm to be implemented by individual agents to fit a linear model of its own, using a partial set of observations and the estimates received from their neighbors. We followed a stepwise approach to reformulate the basic RLS algorithm in view of privacy constraints imposed on the agents. In the first step, we recast the RLS algorithm to a semi-distributed form following privacy constraints, where a central coordinator forms the estimate using the local estimates of all the agents, while maintaining their data privacy. In second step, we reformulate it into a fully separable form without any coordinator, where the agents form estimates of their own. In the results it is observed that, convergence of an agent's estimate to the full information case is proportional to the number of neighbors of the agent. The discussion on effect of P on the estimates also proved the same. The effect of robustness of data on performance of the distributed regression algorithm has also been discussed.

Apart from in-network distributed estimation, the proposed distributed regression algorithm can also be applied in other prediction/estimation systems where distributed predictions are required to be performed while maintaining privacy of agents data. The example of such system includes Stock market estimations where shareholders are unwilling to share the stock information they have, costly laboratory experiments conducted by different organizations/agents who generally refuse to share their readings but do not mind sharing estimates for similar estimates from others in return etc. Moreover, This algorithm does not require the agents to communicate with all other agents or a coordinator. This proves beneficial in case of in-network processing as it reduces communication cost, nullifies the chances of network flooding and also whole network breakdown in case of coordinated estimation where the coordinator stands as the single point of failure.

References:

1. *Distributed Sparse Linear Regression*- Gonzalo Mateos, Juan Andrs Bazerque and Georgios B. Giannakis, Fellow, IEEE

2. *All About Linear Regression*- D.G. Simpson, Ph.D., Department of Physical Sciences and Engineering,Prince Georges Community College

3. *Formulating State Space Models in R with Focus on Longitudinal Regression Models*- Claus Dethlefsen, Aalborg University

4. *Wireless Sensor Networks*- F. L. LEWIS, Associate Director for Research, The University of Texas at Arlington

5. *Adaptive Filters* -Simon Haykin, McMaster University

6. *Statistical Digital Signal Processing*- Monson H. Hayes