Sparse Recovery (Non-convex optimization for Machine Learning)

Banda Naveen - 22104061

Sanikommu Srikanth Reddy – 22104092

Chandrima Upadhyay – 22104401

Department of Electrical Engineering, Indian Institute of Technology Kanpur

*Abstract*—In practical applications like signal processing, image processing, machine learning and many more, it is observed that the data follows a certain pattern where, the measurement vector is obtained by projections on design matrix and the rows of this design matrix look like noise-like waveforms, the corresponding original parameters vector (sometimes called regressor vector) is sparse when encoded. Therefore, capable algorithms are required to recover these sparse vectors. This paper discusses three such algorithms, one from each category (convexity type). Those algorithms are BP (convex relaxation), IRLS (non-convex) and IHT (greedy). This paper also shows the simulations of their performance characteristics such as NMSE, recovery time and number of floating-point operations for two different design matrices, one is iid Gaussian and other is iid Bernoulli, in presence of three different noise variances and also their comparisons.

Index Terms— Basis Pursuit (BP), Iteratively Reweighted Least Squares (IRLS), Iterative Hard Thresholding (IHT), Normalized Mean Square Error (NMSE), Independent Identically Distributed (iid), Restricted Isometry Property (RIP).

# Introduction

In many practical situations such as image processing, signal processing in wireless communications and machine learning, hidden sparsity exists. Consider an example of image processing, 1.536G bits (~192Mb) are required to store a 64MP image, but size of typical compressed image is approximately 3-4 Mb. In conventional approach large amount of information is dumped during compression by many sensors. It can be improved if we were able to compress data while sensing itself. This leads to considerable reduction in number of sensors. If the sensed image is converted into another domain where it consists of large number of zeros (sparse), we need to store only the non-zeros values and their locations that leads to drastic reduction in memory size. Therefore, research has been carried out in this regard to identify one-to-one transformation techniques and came up with techniques like wavelet transform, DCT, jpeg2000, etc. To recover the original image from the sparse image inverse transformation can be applied since the one-to-one transformation is known to us.

Now consider an example of signal processing in wireless communications, i.e., mmWave MIMO channel estimation. mmWave MIMO channel has unique properties and the channel model is given as H = RHbTH. This is known as beamspace channel representation. Where R and T are dictionary of array response vectors at Receiver and Transmitter respectively and Hb is the beamspace channel matrix where the property tells us that this matrix has large number of zeros because of existence of only few multipath components while the rest of the components are zeros because of low scattering and higher obstruction of buildings. Implies it is sparse in nature. Using pilot transmission on vectorizing the observation matrix, resultant model becomes sparse vector recovery model.

Now consider an example of machine learning in mMTC system. There are M single antenna devices and N antennas at Base Station (BS), K out of M devices (K<N<<M) transmit data which BS needs to process. But BS does not know which devices are active. A model is designed where BS assumes all devices as active. Then in this model the transmitting signal vector consists of only K number of non-zero elements out of M. Hence it is a sparce vector. In such scenarios Machine Learning techniques such as Sparse Bayesian Learning, Expectation Maximation algorithms exploit this concept of sparsity. There are many such scenarios, as stated already, in real world hence sparse recovery has been studied, implemented many algorithms and research is still being carried out. This paper gives a view on three such algorithms utilizing sparisity.

The rest of this paper is organized as follows. In Section II, the model for sparse recovery system is presented, stating the properties of design matrix alongside. The sparse recovery techniques, their optimization problem formulation and the respective algorithms are proposed in Section III. Simulation results are presented in Section IV. Finally, conclusions are drawn in Section V.

# System model and design matrix properties

## System Model

Let be the sparse vector of dimension Nx1, be the design matrix of dimension MxN, be the observation vector of dimension Mx1 and be the vector of dimension Mx1 consisting of zero-mean iid Gaussian samples. Then the system model is given as

A picture containing text, screenshot, diagram, line

Description automatically generated

Fig. 1. System Model

When , system becomes underdetermined. Implies has no unique solution in conventional approach of linear algebra.

## Design Matrix and RIP Property

For the system model described above, if the design matrix has certain structure and if  is a sparse vector then it is possible to reconstruct uniquely if number of observations is at least twice the number of non-zero elements (k) in **.** Design matrix having certain structure implies that it must satisfy RIP property. has RIP of order k if

for all k-sparse vectors.

RIP of order 2k implies that for all k sparse vector , the following property is satisfied by

If , then following are the results for recovery of sparse vector .

Tractable recovery: All k-sparse vectors are exactly recovered via norm minimization.

Robust recovery: For any vector

and

Where indicates the best k term approximation to vector .

Stable recovery:

Measure , with

For any vector ,

There are a set of matrices which satisfy the above properties.

For example, iid Gaussian entries, iid Bernoulli entries, iid subgaussian entries, random Fourier ensemble and random subset of incoherent dictionary.

# Sparse recovery techniques and their algorithms

## Basis Pursuit

Number of non-zeros elements in any vector can be expressed as its l0 norm. Minimizing norm leads to sparse solution. Therefore, optimization problem for reconstructing of sparse vector for the proposed model

It can also be formulated as

Where k represents the number of non-zero elements of .

However, norm is highly non-convex and NP hard problem to solve. Replacing norm by norm (convex) produces similar sparse solution. This relaxation of non-convex problem into convex problem formulates as Basis Pursuit.

Equivalently written as

**Algorithm 1** Basis Pursuit (BP)

**Input:** Data , Ф,

**Output:** A sparse vector

1: M ← dimension of signal vector

2: while not ()

3: min ()

4: end while

5: return

It’s up to designer discretion to set the value of to a certain threshold.

## Iteratively Reweighted Least Squares

It is one of the most popular non-convex problem-solving techniques used for sparse vector recovery and becomes tedious task when the size of the observations is very large because it involves weighted pseudo inverse operation. Using IRLS sparse recovery problem can be formulated as

Where .

In presence of noise, problem can be formulated as

First, we need to solve the least squares problem.

Solution is obtained as

Now this is used to calculate the weights as follows.

To solve truly non-convex problem, p is considered 0. Then weight vector becomes . If is 0 in any iteration, then weight vector solution is infeasible. To overcome this problem a small value (>0) is added to . Then resultant weight vector is given by.

Using this weighted vector, the least square problem can be formulated as Weighted Least Squares problem as shown below.

Then solution is obtained as

Where P is a diagonalised matrix of the weighted vector

As it is an iterative process, it requires a stopping criterion.

Based on the stopping criterion, the final leads to a sparse vector solution.

**Algorithm 2** Iteratively Reweighted Least Squares (IRLS)

**Input:** Data , Ф, threshold, step size s

**Output:** A sparse vector

1: th ← 1, i ← 0

2: ←

3: while ()

4:

5:

6: i++

7: ←

8: if

9: th ← th\*s

10: endif

11: ←

12: end while

13: return

## Iterative Hard Thresholding

It is a greedy algorithm which implies it takes lesser time for sparse vector solution even for large number of observations unlike the non-convex and convex-relaxation algorithms. But here we need to provide the sparsity level of the signal vector as input. General optimization problem of sparse vector recovery is given by.

Iterative Hard Thresholding (IHT) algorithm solves the above problem iteratively leading to a sparse solution based on sparsity level k. Initially is set to 0. Then, calculate projection of columns of on . As the sparsity level k is given, choose indices of k largest projections. These indices are used to update step length.

In any iteration, is given as projection of columns of on residue. It is given as

After calculating step length , the updated estimate is given by.

And then apply hard thresholding operator on . This operator takes k indices corresponding to largest absolute values of .

Performing the above steps iteratively gives sparse vector solution.

**Algorithm 3** Iterative Hard Thresholding (IHT)

**Input:** Data , Ф, k

**Output:** A sparse vector

1: i ← 0, 0,

2: ← 0

3: while ()

4:

5: update

6:

7: indices ← k-max (

8: temp←0

9:

10:

11: i++

12:

13: end while

14: return

# Simulation results

Considering the system model described in section II we constructed two design matrices satisfying RIP properties. One with iid Gaussian entries and another with iid Bernoulli entries. We also generated sparse vector and obtained observations in presence of three different noises. Based on the recovery of this sparse vector we evaluated performance parameters such as NMSE, recovery time and number of floating-point operations (FLOPS).

We constructed ten different design matrices for both Gaussian entries and Bernoulli entries, for each noise variance respectively. Numerical values are shown in below table I.

TABLE I. Simulation Parameters

|  |  |
| --- | --- |
| **Parameters** | **Value/Range/Size** |
| Observation vector | 40x1 – 400x1 |
| Design matrix | 40x80 – 400x800 |
| Sparsity level (K) | 14 – 140 |
| Noise variances (e) | -20dB, -10dB, 0dB |
| Threshold (IRLS) | 1e-08 |
| Step size (s) (IRLS) | 0.1 |
| Regularizing parameter (BP) | 1 |

Considering 10 data sets by incrementing observation vector by +40, design matrix by (+40)x(+80) and sparsity level by +14.

## NMSE simulation

## Comparing NMSE for BP in presence of 3 different noise variances.

Here simulations are carried out for both Gaussian and Bernoulli design matrices for increasing number of observations vs corresponding NMSE of recovered sparse vector. As the observations increase, we can see that NMSE keeps decreasing. This is because of law of large numbers i.e., as the size of observations increase there is improvement in sparse vector estimation.

As the noise variance increases, estimating the sparse vector solution is poor. Consequently, NMSE also increases. These results are shown in below figures Fig. 2. And Fig. 3.

A picture containing text, line, plot, diagram

Description automatically generated

Fig. 2. NMSE of BP for different noise (Gaussian)A picture containing text, plot, line, diagram

Description automatically generated

Fig. 3. NMSE of BP for different noise (Bernoulli)

## Comparing NMSE for IRLS in presence of 3 different noise variances.

Simulations in IRLS are carried out as described in BP. Here as well we observe similar results as shown in below figures Fig. 4 and Fig. 5

A picture containing text, line, plot, diagram

Description automatically generated

A picture containing text, plot, diagram, line

Description automatically generatedFig. 4. NMSE of IRLS for different noise (Gaussian)

Fig. 5. NMSE of IRLS for different noise (Bernoulli)

## Comparing NMSE for IHT in presence of 3 different noise variances.

Simulations in IHT are also done in similar fashion. Here we observe similar results in Gaussian but slightly different in Bernoulli i.e., the NMSE varies randomly as the data size increases shown in below figures Fig. 6 and Fig. 7

A picture containing plot, line, text, diagram

Description automatically generated

Fig. 6. NMSE of IHT for different noise (Gaussian)

A picture containing text, diagram, plot, line

Description automatically generated

Fig. 7. NMSE of IHT for different noise (Bernoulli)

## Comparing NMSE for BP, IRLS and IHT in presence of particular noise variance.

In presence of low noise variance, all three algorithms gave better results for Gaussian design matrix compared to that of Bernoulli.

Among the three algorithms IRLS has lower NMSE for different noise variance as the number of observations increases.

IHT behaves random for different scenarios and gives higher NMSE among three algorithms as the number of observations increase. BP seems to give moderate NMSE for any noise variance.

A picture containing text, diagram, line, plot

Description automatically generated These simulations are shown in the following figures Fig. 8., Fig. 9., Fig. 10.

Fig. 8. NMSE among 3 algorithms (e1)

A picture containing text, diagram, plot, line

Description automatically generated

Fig. 9. NMSE among 3 algorithms (e2)

A picture containing text, diagram, plot, line

Description automatically generated

Fig. 10. NMSE among 3 algorithms (e3)

## Recovery Time simulation

Here the simulations are carried out between increasing number of observations and time taken for recovering its corresponding sparse vector. Among the three algorithms we notice that IHT takes very less time, IRLS takes significantly large amount of time and BP takes moderate amount of time for finding sparse solution for any number of observations. This is because IHT is a greedy algorithm, IRLS is a non-convex algorithm and BP is a convex relaxation algorithm. And any algorithm takes more time as the number of observations increases.

Across design matrices, all the algorithms take less time for Gaussian when compared to Bernoulli. We also see that as noise variances increase all the algorithms run slightly longer. Simulations results are shown in below figures Fig. 11., Fig. 12., Fig. 13.

A picture containing text, plot, line, diagram

Description automatically generated

Fig. 11. Recovery Time among 3 algorithms (e1)

## FLOPS simulation

Here the simulations are carried out between increasing number of observations and log of number of floating-point operations for recovering its corresponding sparse vector.

A picture containing text, diagram, line, plot

Description automatically generated

Fig. 12. Recovery Time among 3 algorithms (e2)

A picture containing text, diagram, plot, line

Description automatically generated

Fig. 13. Recovery Time among 3 algorithms (e3)

We witnessed that for a particular observation, BP takes same number of FLOPS irrespective of design matrix and noise variance. IRLS also works in exact same way as that of BP by a very small increment in FLOPS. When it comes to IHT, it takes large number of FLOPS among all three algorithms because it involves calculation of projections multiple times.

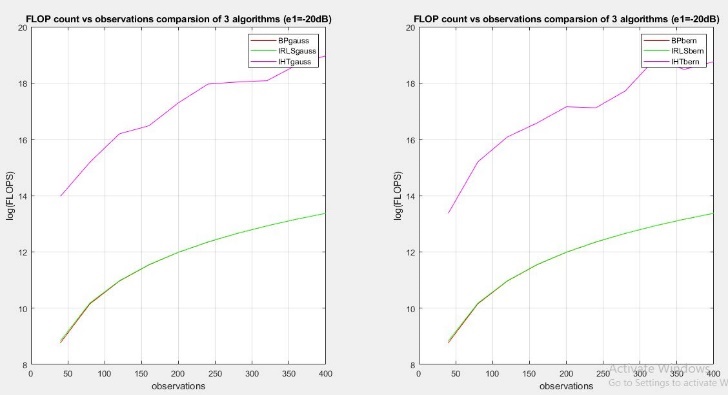


Fig. 14. FLOPS among 3 algorithms (e1)

A picture containing text, plot, line, diagram

Description automatically generated

Fig. 15. FLOPS among 3 algorithms (e2)

As the number of observations increase FLOPS also increase for all the algorithms. These results are shown in figures Fig. 14., Fig. 15., Fig. 16.

A picture containing text, diagram, line, plot

Description automatically generated

Fig. 16. FLOPS among 3 algorithms (e3)

# Conclusion

We have simulated the performance characteristics of three proposed sparse recovery algorithms. We can observe that among these 3 algorithms, IHT and BP algorithms can be used for large sized data because their recovery time is much less and having moderate error. Even though IRLS has the least error it becomes difficult for large sized dataset because it involves computing weighted pseudo inverse. We can also see that IHT has a considerably high number of FLOPS due to multiple evaluation of projections whereas IRLS has less FLOPS because there is only one step involving computation of inverse. Finally, all three algorithms perform better when the design matrix is of iid Gaussian entries compared to iid Bernoulli entries.

##### References

1. Jain, Prateek, and Purushottam Kar. "Non-convex optimization for machine learning." *arXiv preprint arXiv:1712.07897* (2017).
2. E. Crespo Marques, N. Maciel, L. Naviner, H. Cai and J. Yang, "A Review of Sparse Recovery Algorithms," in IEEE Access, vol. 7, pp. 1300-1322, 2019, doi: 10.1109/ACCESS.2018.2886471.
3. Arjoune, Youness et al. “Compressive sensing: Performance comparison of sparse recovery algorithms.” 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC) (2018): 1-7.
4. Emmanuel J. Candès. The Restricted Isometry Property and Its Implications for Compressed Sensing. Comptes Rendus Mathematique, 346(9-10):589–592, 2008.
5. Z. Qin, J. Fan, Y. Liu, Y. Gao, and G. Y. Li, ‘‘Sparse representation for wireless communications: A compressive sensing approach,’’ IEEE Signal Process. Mag., vol. 35, no. 3, pp. 40–58, May 2018.
6. G. Wunder, H. Boche, T. Strohmer, and P. Jung, ‘‘Sparse signal processing concepts for efficient 5G system design,’’ IEEE Access, vol. 3, pp. 195–208, 2015.
7. S. Foucart, ‘‘Sparse recovery algorithms: Sufficient conditions in terms of restricted isometry constants,’’ in Approximation Theory XIII: San Antonio, M. Neamtu and L. Schumaker, Eds. New York, NY, USA: Springer, 2012, pp. 65–77.
8. R. Calderbank, S. Howard, and S. Jafarpour, ‘‘Construction of a large class of deterministic sensing matrices that satisfy a statistical isometry property,’’ IEEE J. Sel. Topics Signal Process., vol. 4, no. 2, pp. 358–374, Apr. 2010.
9. D. L. Donoho and Y. Tsaig, ‘‘Fast solution of `1-norm minimization problems when the solution may be sparse,’’ IEEE Trans. Inf. Theory, vol. 54, no. 11, pp. 4789–4812, Nov. 2008.
10. Berger, Z. Wang, J. Huang, and S. Zhou, ‘‘Application of compressive sensing to sparse channel estimation,’’ IEEE Commun. Mag., vol. 48, no. 11, pp. 164–174, Nov. 2010.
11. I. F. Gorodnitsky and B. D. Rao, ‘‘Sparse signal reconstruction from limited data using FOCUSS: A re-weighted minimum norm algorithm,’’ IEEE Trans. Signal Process., vol. 45, no. 3, pp. 600–616, Mar. 1997.
12. I. Daubechies, M. Defrise, and C. De Mol, ‘‘An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,’’ Commun. Pure Appl. Math., vol. 57, no. 11, pp. 1413–1457, 2004.
13. MohammadReza Jabbari (Jabari) (2023). Basis Pursuit (https://www.mathworks.com/matlabcentral/fileexchange/73846-basis-pursuit), MATLAB Central File Exchange. Retrieved April 22, 2023.
14. MohammadReza Jabbari (Jabari) (2023). Iterative\_Hard\_Thresholding (https://www.mathworks.com/matlabcentral/fileexchange/124415-iterative\_hard\_thresholding), MATLAB Central File Exchange. Retrieved April 22, 2023.
15. aresmiki (2023). CS-Recovery-Algorithms (https://github.com/aresmiki/CS-Recovery-Algorithms), GitHub. Retrieved April 22, 2023.