SPARSE BAYESIAN LEARNING FOR COMPRESSIVE RECOVERY

CS754 ADVANCED IMAGE PROCESSING

Pradipta Bora
190050089

Harshit Gupta
190050048

May 10, 2021

1 Introduction

In this project we aim to study a method for compressive reconstruction based on bayes theorem. This technique known as Sparse Bayesian Learning, aims on learning a prior on the fly for the original image from the reconstructed image. In the following sections we first describe compressive recovery as an MAP problem, look at existing L1 norm algorithms as a special case of the MAP problem and then look at methods for learning priors for improved recovery.

2 Compressive Recovery as an MAP problem

Consider a compressed sensing setup where:

$$y = \Phi\Psi x = Ax$$

Here x is sparse vector. We have already observed y (with added noise) and need to estimate x. Under the bayesian setup we have:

$$P(x|y)P(y) = P(y|x)P(x)$$

In Bayesian Machine Learning we aim to choose an x that maximizes the posterior probability. This is the MAP estimate. The likelihood term is estimated by a gaussian, if

$$y - Ax = \eta$$

and $\eta \sim N(0, \sigma^2)$ Then:

$$P(y|x) = \exp(||y - Ax||^2/2\sigma^2)$$

Therefore the estimated x, $\hat{x}$ is:

$$\hat{x} = \arg\max_x P(y|x)P(x)$$

Minimising the negative log likelihood, we get:

$$\hat{x} = \arg\min_x ||y - Ax||^2/2\sigma^2 - \log P(x)$$

We now need to choose a prior on x. ISTA and other L1 norm based algorithms assume a laplacian prior on x, such that:

$$P(x) \propto \exp(-\lambda||x||_1)$$

This will give us:

$$\hat{x} = \arg\min_x ||y - Ax||^2/2\sigma^2 + \lambda||x||_1$$

But instead of generalising the prior ahead of time can we infer it from the observed samples? This is the essence of Sparse Bayesian Learning.

3 Sparse Bayesian Learning

Consider a generalised prior, such that each entry in x, x_i is a gaussian with mean 0 and variance γ_i. Here we are modelling each entry as a gaussian with separate variance. This variance will be learnt on the fly.

We get:

$$P(x_i; \gamma_i) = \frac{1}{\sqrt{2\pi\gamma_i}} \exp(-\frac{x_i^2}{2\gamma_i})$$

We can model x as a multivariate gaussian with covariance matrix

$$\Gamma = \text{diag}(\gamma_1, \gamma_2 \cdots)$$

Suppose x has k entries then we get:

$$P(x; \Gamma) = \frac{\exp(-\frac{1}{2}x^T\Gamma^{-1}x)}{\sqrt{(2\pi)^k|\Gamma|}}$$

The MAP estimate for this will therefore become:

$$\hat{x} = \arg\min_{x,\Gamma} ||y - Ax||^2/2\sigma^2 + \log(|\Gamma)| + \frac{x^T\Gamma^{-1}x}{2}$$

$$= \arg\min_{x,\Gamma} \frac{||y - Ax||^2}{2\sigma^2} + \sum_{i=1}^{k} \frac{x_i^2}{2\gamma_i} + \log(\gamma_i)$$

Sparse Bayesian Learning now optimises over both x and Γ above for the reconstruction.

There are two methods used for this leading to two algorithms. One is more direct but can result in accuracy issues due to matrix inversion and the other relies on optimising a upper bound on the likelihood. We will describe them and prove how they work below:

3.1 Approximate MAP (A-MAP) algorithm

This algorithm begins by assuming $\Gamma = I$ and then fixing Γ we compute the best x (minimising the cost) and then for that x update the value of Γ. This is similar to blind compressed sensing. We repeat this alternating process until the cost does not change much. At that stage, we stop and return the best x for that value of Γ.

Recall that we are optimising the cost function:

$$C = ||y - Ax||^2/2\sigma^2 + \log(|\Gamma)| + \frac{x^T\Gamma^{-1}x}{2}$$

The updates are then as follows:

1. Update x keeping Γ fixed.
   For this update we differentiate with respect to x.

$$\frac{\partial C}{\partial x} = \frac{1}{2\sigma^2}\frac{\partial(yy^T - 2y^TAx + x^TA^TAx)}{\partial x} + \frac{x^T((\Gamma^{-1})^T + \Gamma^{-1})}{2} = 0$$

$$\implies \frac{-2y^TA + 2x^TA^TA}{2\sigma^2} + x^T(\Gamma^{-1}) = 0$$

   Simplifying the above we get:

$$x^T(A^TA + \sigma^2\Gamma^{-1}) = y^TA$$

   Taking transpose and inverting gives:

$$x = (A^TA + \sigma^2\Gamma^{-1})^{-1}A^Ty$$

   This is the update step keeping Γ fixed.

2. Update Γ for the previous value of x
   This is easy to do if we update γ_i one by one. Observe that we also have:
   $$C = \frac{||y - Ax||^2}{2\sigma^2} + \sum_{i=1}^{k} \frac{x_i^2}{2\gamma_i} + \log(\gamma_i)$$
   Then we get:
   $$\frac{\partial C}{\partial \gamma_i} = -\frac{x_i^2}{2\gamma_i^2} + \frac{1}{2\gamma_i} = 0$$
   Solving we get:
   $$\gamma_i = x_i^2$$
   Thus we update Γ by replacing it with diag$(x_1^2, x_2^2 \cdots x_k^2)$
   We do this alternating update until convergence. This is the A-MAP algorithm.

Unfortunately the above algorithm requires inverting a matrix which may be computationally intractable. To prevent this we look at another algorithm which uses Expectation Maximisation algorithm for optimising the cost. This is called the standard SBL algorithm in the literature.

3.2 SBL Algorithm

Before we get into the SBL algorithm let us rephrase the problem. We essentially want to find the best prior that fits the data. The best set of parameters Γ would maximise the overall log likelihood

$$l(\Gamma) = \log P(y; \Gamma)$$

One algorithm that finds this set of parameters is called the EM algorithm. Here we will describe the EM algorithm before seeing its implementation in SBL.

3.2.1 Expectation Maximisation Algorithm

Let the parameter set that we are estimating be called θ. Suppose the observation is y and there is some hidden information (latent variable) termed as x (in CS this is the original sample).

Then we have:

$$l(\theta) = \log p(y|\theta) = \log \sum_x p(y, x|\theta)$$
$$= \log \sum_x q(x|y,\theta) \frac{p(y, x|\theta)}{q(x|y,\theta)}$$

Here q is an arbitrary distribution over x. We can therefore write this as an expected value:

$$= \log \mathbf{E}_{q(x|y,\theta)} \frac{p(y, x|\theta)}{q(x|y,\theta)} \geq \mathbf{E}_{q(x|y,\theta)} \log \frac{p(y, x|\theta)}{q(x|y,\theta)}$$

Here we have used Jensen's inequality as $\log$ is concave. Thus we have a lower bound on the likelihood. What EM algorithm does is that it optimises the lower bound alternatingly. First we find a q that maximises this lower bound. Then we find the value of θ maximising this lower bound. That is if we had an estimate θ^t then:

$$q^{t+1} = \arg\max_q \mathbf{E}_{q(x|y,\theta^t)} \log \frac{p(y, x|\theta^t)}{q(x|y,\theta^t)} \text{ (E Step)}$$

And then we get:

$$\theta^{t+1} = \arg\max_\theta \mathbf{E}_{q^{t+1}(x|y,\theta^t)} \log \frac{p(y, x|\theta)}{q^{t+1}(x|y,\theta^t)} = \arg\max_\theta \mathbf{E}_{q^{t+1}(x|y,\theta^t)} \log p(y, x|\theta) \text{ (M Step)}$$

The power of this algorithm comes from the fact that the first step is rather easy, it turns out that the q maximising this is:

$$q^{t+1}(x|y,\theta^t) = p(x|y,\theta^t)$$

The reason behind this is that to maximise the expectation at θ^t, we would have to ensure that the lower bound equals the upper bound $l(\theta^t)$. This is true when the expectation is over a constant value.

That is:

$$\frac{p(y,x|\theta^t)}{q^{t+1}(x|y,\theta^t)} = c$$

for a constant c independent over the variable over which we are taking the expectation (x).

It is easy to find c due to normalisation. We get:

$$\sum_x p(y,x|\theta^t) = c\sum_x q^{t+1}(x|y,\theta^t) = c$$

Thus:

$$c = \sum_x p(y,x|\theta^t) = p(y|\theta^t)$$

This gives:

$$q^{t+1}(x|y,\theta^t) = \frac{p(y,x|\theta^t)}{p(y|\theta^t)} = p(x|y,\theta^t)$$

Since this is always true, we therefore have simplified the steps to:

1. E step:
   Compute the distributions:
   $$Q(\theta|\theta^t) = \mathbf{E}_{p(x|y,\theta^t)} \log p(y,x|\theta)$$
2. M step:
   Compute the estimate of θ, θ^{t+1} as:
   $$\theta^{t+1} = \arg\max_{\theta} Q(\theta|\theta^t)$$

Now we will see how to apply this for Sparse Bayesian Learning. The idea is simple, we will compute the value of Γ maximising the likelihood (the best prior) using EM and then solve for the corresponding x like we did in AMAP.

3.3 EM in SBL

The process remains the same, however since here we have concrete values of the various terms we can compute close form expressions.

First we have the E step. Here we have:

$$Q(\Gamma|\Gamma^t) = \mathbf{E}_{p(x|y,\Gamma^t)} \log p(y,x|\Gamma)$$

Let us find first the distribution of x here:

$$p(x|y,\Gamma^t) = c\exp(-\frac{||y-Ax||^2}{2\sigma^2} - \frac{x^T(\Gamma^t)^{-1}x}{2})$$

Here c is a normalising constant. Clearly this is a multivariate gaussian say $N(\mu,\Sigma)$. We will find the mean and covariance matrix of this multivariate gaussian.

To find mean we can simply find the x at which the above density is maximised. This is similar to the update of x keeping Γ fixed in A-MAP. Thus the same expression will be obtained, which will be the value of μ

So we get:

$$\mu = (A^TA + \sigma^2(\Gamma^t)^{-1})^{-1}A^Ty$$

From this we can obtain the covariance matrix Σ as the terms inside the $\exp$ will be equal and so we get:

$$-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2} = -\frac{||y-Ax||^2}{2\sigma^2} - \frac{x^T(\Gamma^t)^{-1}x}{2}$$

Solving this we will get:

$$\Sigma = (\sigma^{-2}A^TA + (\Gamma^t)^{-1})^{-1}$$

Now for the M step we get:

$$\Gamma^{t+1} = \arg\max_{\Gamma} Q(\Gamma|\Gamma^t)$$

$$= \arg\max_{\Gamma} \int_x p(x|y,\Gamma^t)\log p(y,x;\Gamma)$$

$$= \arg\min_{\Gamma} \int_x p(x|y,\Gamma^t)(\frac{||y-Ax||^2}{2\sigma^2} + \frac{x^T(\Gamma^t)^{-1}x}{2} + +\frac{\log|\Gamma|}{2})$$

Observe that we know $p(x|y,\Gamma^t)$ and hence we can differentiate the above with respect to Γ to obtain the value of Γ^{t+1}. We skip the lengthy algebra here and the result comes out to be:

$$\Gamma^{t+1} = \text{diag}(\mu_i^2 + \Sigma_{ii})$$

This is very good news because we an obtain Γ^{t+1} directly from μ, Σ.

Thus the overall algorithm comes out to be:

1. Initialise Γ to a base value say I
2. In the E step compute
$$\mu = (A^TA + \sigma^2(\Gamma^t)^{-1})^{-1}A^Ty$$
$$\Sigma = (\sigma^{-2}A^TA + (\Gamma^t)^{-1})^{-1}$$
3. In the M step update
$$\Gamma^{t+1} = \text{diag}(\mu_i^2 + \Sigma_{ii})$$
Repeat the two steps alternatingly until convergence of Γ.
4. Finally return the value of x from Γ as:
$$\hat{x} = (A^TA + \sigma^2\Gamma^{-1})^{-1}A^Ty$$

This finishes our theoretical analysis. Now we will look at some results:

4 Experimental Analysis

In the first set of experiments, we applied OMP, ISTA, AMAP, SBL for compressive recovery in synthetically generated data from the Barbara Image. A random matrix with samples drawn from a gaussian was taken as the measurement matrix (such a matrix obeys RIP bounds with high probability) and the number of samples taken was varied. The time taken for the reconstruction was also observed.

The following results were observed on the Barbara image:

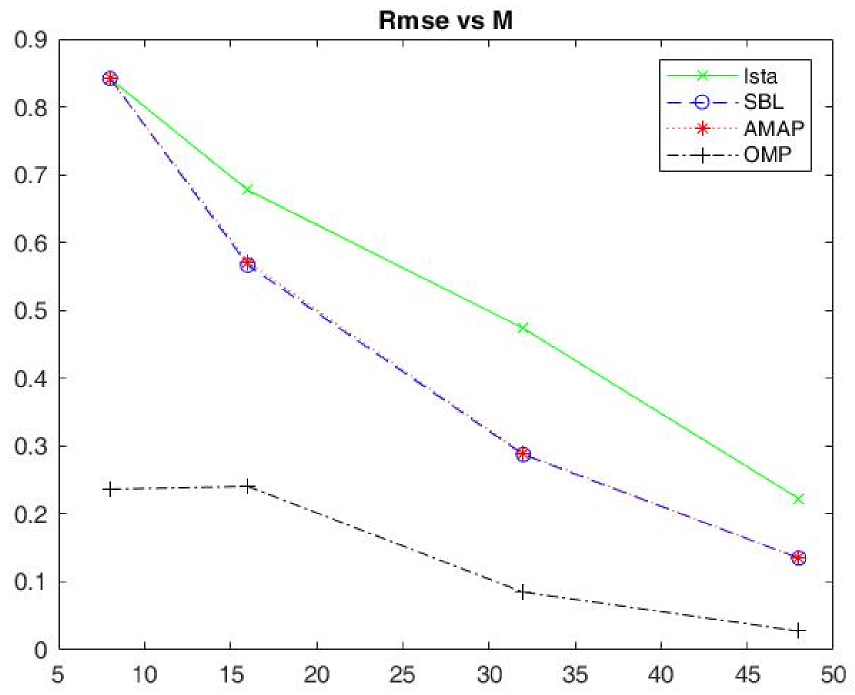

Figure 1: RMSE vs M (Number of Samples)

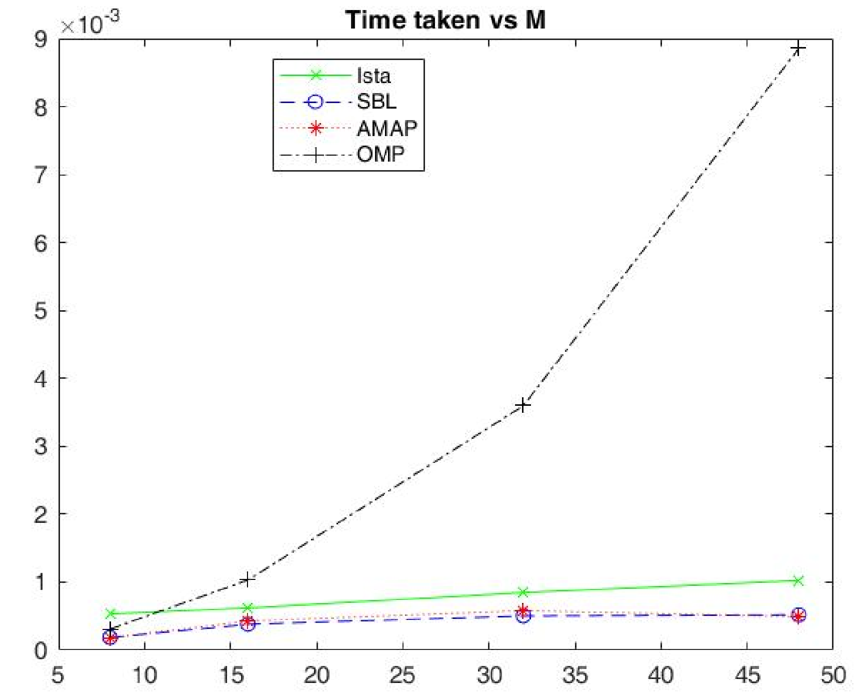

Figure 2: Time Taken vs M

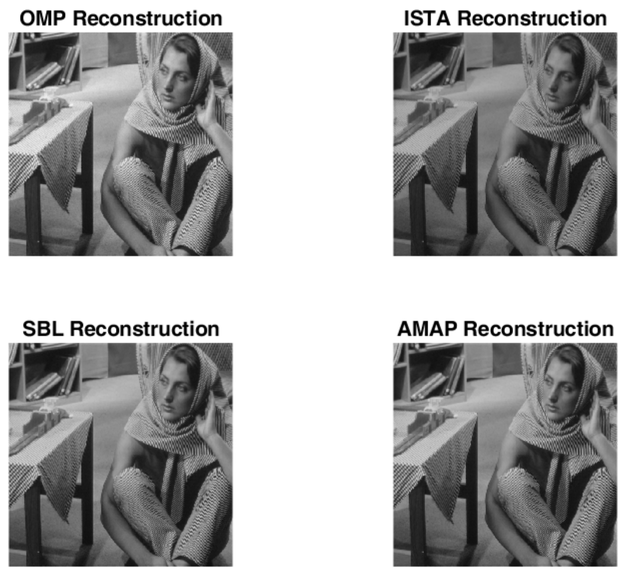

Figure 3: Reconstruction for 48 samples

The above results shows that although OMP gives the best results, it takes far more time then the other methods (which is expected since it is based upon basis pursuit which is computationally slower). Among the cost function optimisation based techniques, we see that SBL/AMAP give far better reconstruction than ISTA and take less time to do so.

Between SBL and AMAP, we were unable to get AMAP working for 64 samples as the matrix turned out to be non invertible. SBL did not face this problem and hence SBL works better with higher number of samples. Other than this the difference in RMSE between SBL and AMAP is negligble which is expected since they both optimise the same cost function. Similar trends were observed on the Lena image.

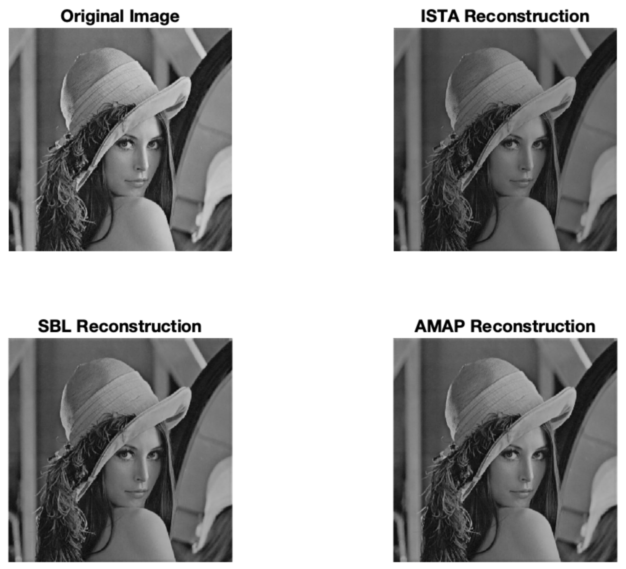

Figure 4: Reconstruction for 48 samples

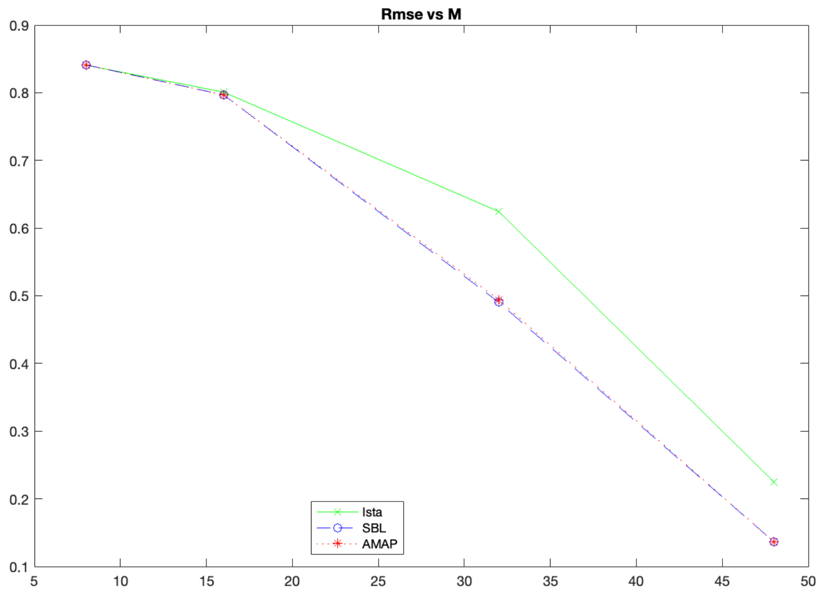

Figure 5: RMSE vs M for Lena

In the second set of experiments we varied the sparsity level of the sample. We took a random vector x of size 100 with varying sparsity level. The measurement matrix was taken to be a random 50×100 matrix and so the compression rate was 2. Gaussian noise was added to the measurements(zero mean and variance 0.1) and the reconstruction for various algorithms were compared.

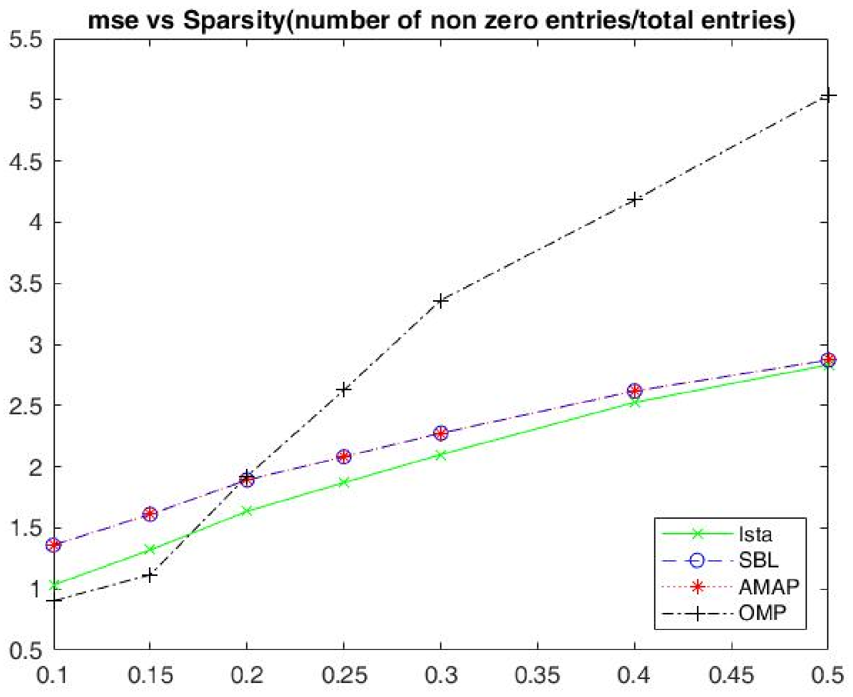

Figure 6: Mean Squared Error vs Sparsity

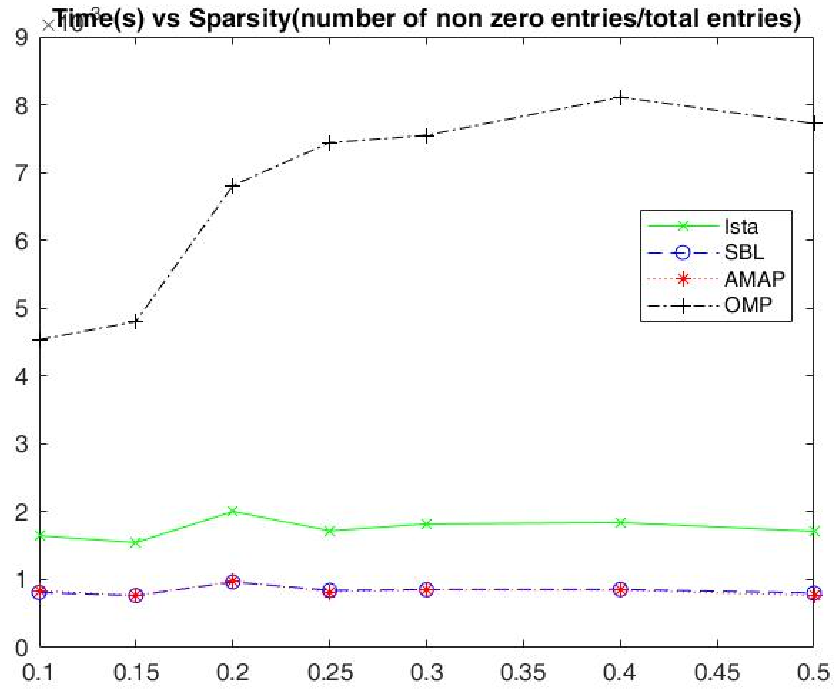

Figure 7: Time taken by different algorithms

Here we observe that OMP does worse after a certain point than the other algorithms and still takes the most time. The error rate between ISTA and SBL algorithms is again very close and SBL algorithms are way faster. We can therefore get lower reconstruction error in SBL if we change the value of ϵ in SBL so that it takes the same time as ISTA. AMAP and SBL are very similar in this case.

5 Extension to multiple sparse vectors

We consider an extension to multiple measurements as follows:
$\{x_1, x_2 \cdots x_L\}$ are sparse vectors of size n such that each of them have the same support (The exact value may differ). Further let $A_1, A_2 \cdots A_L \in R^{m\times n}$ be the measurement matrix corresponding to them and the measurements are

$$y_i = A_i x_i + \eta_i$$

where $\eta_i \sim \mathcal{N}(0, \sigma_i^2)$.
The problem is to estimate $x_i's$ given $A_i's$ and $y_i's$. Notice that there is shared information between x_i(They have same support) and hence combined reconstructions work better than the independent reconstructions.
In case of SBL-EM algorithm, it is easy to extend single vector problem to multi-vector problem. We assume that each of the x_i are generated from the same prior typically taken to be $\mathcal{N}(0, \Gamma)$ where Γ is a diagonal matrix.(so each entry of a given x_i are chosen independently)
Notice that each y_i is constructed independently if given x_i and A_i. So,

$$p(Y; \Gamma) = \int p(Y, X; \Gamma) dX = \prod_{i=1}^{L} \int p(y_i|x_i) p(x_i; \Gamma) dx_i$$

The loss function is $logp(Y;\Gamma)$ and so it is just the sum of loss for individual problems. Doing a similar analysis as in single vector problem, we get

$$p(X|Y, \Gamma^t) = \prod_{j=1}^{L} \mathcal{N}(\mu_j^{t+1}, \Sigma_j^{t+1})$$

where = $\Sigma_j^{t+1} = \Gamma^t - \Gamma^t A_j^T(\sigma_j^2 I_m + A_j \Gamma^t A_j^T)^{-1} A_j \Gamma^t$ and $\mu_j^{t+1} = \sigma_j^{-2} \Sigma_j^{t+1} A_j^T y_j$. The EM algorithm works as in single vector case with the new formula for Σ^t and μ^t. Further, since the loss function is sum of individual loss

functions, we get the update for Γ as average of individual updates.
Using the EM algorithm on this, we get the updates as:

E step: For $j = 1 \cdots L$, compute

$$\Sigma_j = \Gamma - \Gamma A_j^T(\sigma_j^2 I_m + A_j \Gamma A_j^T)^{-1} A_j \Gamma$$

$$\mu_j = \sigma_j^{-2} \Sigma_j A_j^T y_j$$

M Step:

$$\Gamma(i) = \frac{1}{L} \sum_{j=1}^{L} \left(\Sigma_j(i,i) + \mu_j(i)^2 \right)$$

This is repeated until convergence. Finally, we return the value of μ_j as value of x_j.