Anurag MIGLANI[1]*, Thomas ZAMOJSKI[2]

Abstract
The project includes the implementation in Matlab of 3 recovery algorithms: matching pursuit (MP), orthogonal matching pursuit (OMP) and iterative hard thresholding (IHT).

Keywords
MP/OMP — IHT — Sparse

***Corresponding author**: miglanianurag@protonmail.com

Contents

1. Algorithms

The project includes the implementation in Matlab of 3 recovery algorithms: matching pursuit (MP), orthogonal matching pursuit (OMP) and iterative hard thresholding (IHT). The unit tests were performed as shown in the sample code using random gaussian matrices. Unfortunately, even OMP could not recover 3-sparse vectors. Actually, OMP could sometimes recover the support, but still failed to have the same solutions. The design of the matrix is therefore not appropriate as there is no exact recovery guarantee for 3-sparse vectors.

We found that IHT tends to diverge. We therefore included a decreasing learning rate $\mu_k = \mu/k^{0.2}$to try to make the algorithm converge better, but even that was sometimes not sufficient.

2. Image Restoration

Modeling

Here our image v is represented in a dictionary D: v = Dx. We will make x a sparse vector. Some pixels gets dropped. This is a linear condition, and therefore can be represented as applying a measurement matrix M .
Finally, small noise ε is added to the pixture. The model becomes:

$$y = MDx + \varepsilon$$

The image impainting consists in recovering the missing pixels. This is modeled by the problem:

$$argmin_z ||z||_0 \text{ subject to } MDx = MD_z.$$

Image denoising and inpainting of a noisy image are both modeled with the following sparse recovery problem:

$$argmin_z ||z||_0 \text{ subject to } ||y - MDz||_2 \leq \eta.$$

where η is a small error.

Data Generation

A random dictionary is generated with the provided code in createDic- tionary.m. It is 5 times redundant. The image is also generated with the provided code in createImage.m. For the missing pixels, the code was also provided. One has to note only that removing pixels reduces the dimension of the data. Therefore, we keep another copy with zeros added at the right position to display the image with the missing pixels. In the recovery, we have no information on the location of the pixels.
We have to make sure that $A = MD$ has no zero columns. If it did, then at that index we could put anything in z, making even the recovery algorithm *ill-posed*. Another problem is that the algorithms would always pick that index to start with, and the pseudo inverse would fail.

We include now the visualisation of a few images with different choices of parameters in Figure 1 and and Figure 2.

We test the 3 algorithms on an image with 70% pixels lost. The recovery is shown along side the original image in Figure 3. OMP performs very well, IHT as mentioned has divergence issues, shown by the black part of the image. Matching pursuit is doing better than we expected. If we add noise however, MP is now doing less good, OMP gets a little confused but still performs relatively well. Figure 4 shows the performances. Note also that OMP is very fast, but that could be because we know a priori the sparsity level, so that it does not have to loop so much. We ran MP with 1000 iterations, and still is fast.

As for the peak signal-to-noise-ratio, the values gave 64.56 for MP and 71.92 for OMP. The supports they recovered though were not the same as the groundtruth: only the first index was correct, after MP and OMP got confused. It does not necessarily impact the recovery as we suspect there are many sparse solutions to the recovery problem.

Finally, we have also included in the code the running of the above experiments tweaking with the parameters such as percentage of pixels dropped.

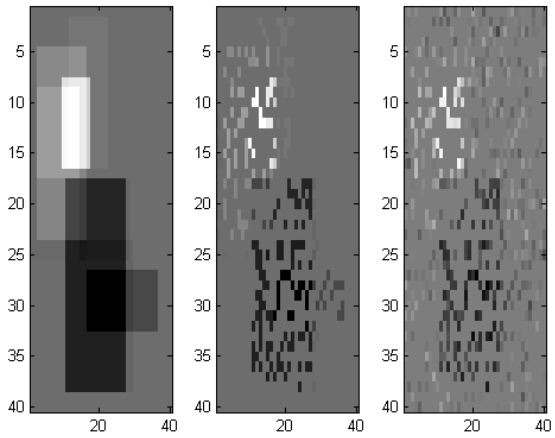

Figure 1. Image with 10 sparse, 70% missing pixels, 0.05 variance gaussian noise.

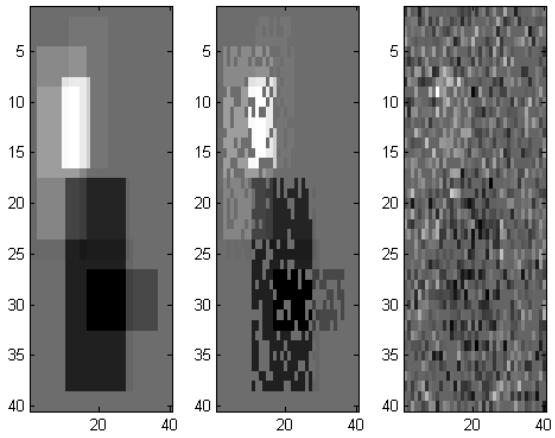

Figure 2. Image with 10 sparse, 30% missing pixels, 0.25 variance gaussian noise.

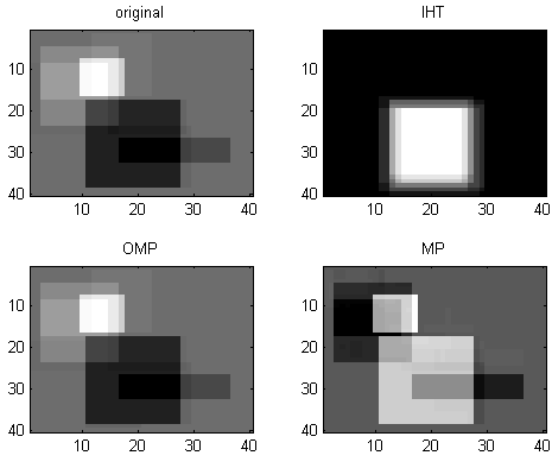

Figure 3. Recovery image with 70% missing pixels.

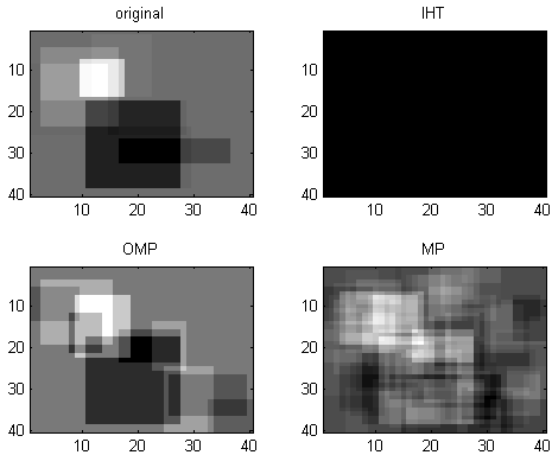

Figure 4. Recovery image with 70% missing pixels.