CS 6220, Fall 2017 **October 31, 2017**

Lecture 21

Lecturer: Anil Damle *Scribes: Louise Lee (zl245), Qinru Shi (qs63), Scott Wu (ssw74)*

1 Introduction

In the previous lectures, we learnt about sparse recovery and compressive sensing. Today, we will prove via demo that sparse recovery and compressive sensing work in section 2, and then talk about working with noisy data in section 3.

2 MATLAB Demos

The demos will use ASP, a package that implements BPDN/LASSO. Other packages that implement BPDN/LASSO include SPGL1 and L1 Homotopy. CVX is another package that can be easily used to implement BPDN/LASSO.

The code used for the in class demo may be found here.

2.1 Demo 1: 1D Sparse Recovery without Noise

In order to demonstrate the effectiveness of sparse recovery, we will try to solve the following problem: Assume we have a sparse vector x and a signal b generated from x, can we recover x from b? For example, let F be the DFT matrix, and let

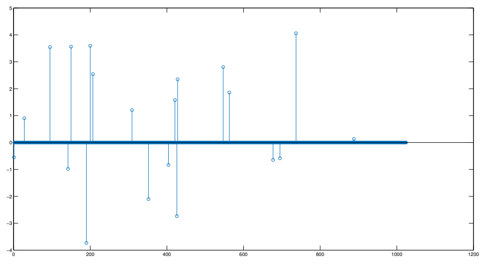
$$A = \begin{bmatrix} I & F \end{bmatrix} \text{ and } b = Ax.$$

In this framework, we can solve the following system:

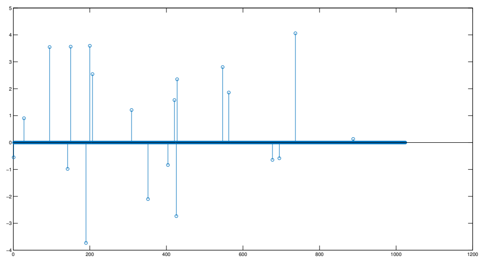
$$min_x \left\| x \right\|_1, \ s.t. \ Ax = b.$$

We denote the calculated solution as $\hat{x}$. Figure 1 shows the comparison between the true solution x and calculate solution $\hat{x}$ in one experiment. We can see that x and $\hat{x}$ are very close. The error only depends on the precision of optimization.

```
>> norm(x-xhat)

ans =

   6.9804e-08
```



(a) The true solution x



(b) The calculated solution $\hat{x}$

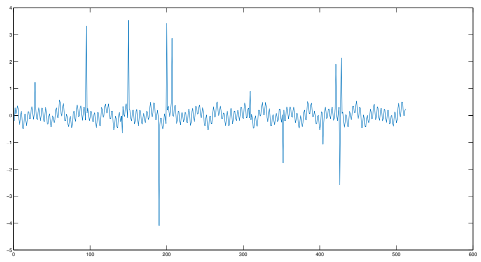
Figure 1: Comparison of the true solution x and result $\hat{x}$

Furthermore, we can verify that the calculated solution has the same sparsity pattern as the true solution:

```
>> setdiff(find(x~=0), find(xhat~=0))

ans =

   Empty matrix: 0-by-1
```

Note that we want to keep all entries of b as real numbers. This will require a careful construction of x. To simplify matters in our demo, we replaced DFT with DCT (Discrete Cosine Transform) so that b will always be real.

Another intuitive idea for solving sparse recovery problems is to approximate x with $\alpha = D^{-1}b$, where D is the DCT matrix. Figure 2 shows that the idea does not work well in practice, since the inverse DCT of x is not sparse.


(a) The original signal b

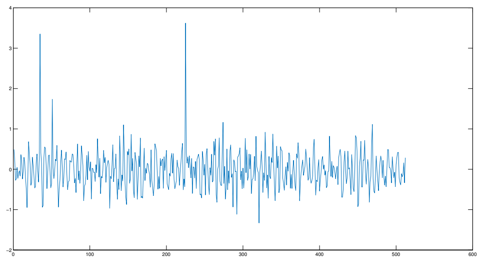

(b) $\alpha = D^{-1}b$

Figure 2: The original signal b and $\alpha = D^{-1}b$

2.2 Demo 2: 1D Sparse Recovery with Noise

For the second demo, we will consider the same problem as in demo 1, but this time with noise. Assume

$$A = \begin{bmatrix} I & D \end{bmatrix} \text{ and } b = Ax + \sigma z,$$

where σz denotes Gaussian random noise scaled by a constant. Now, instead of solving

$$min_x \left\|x\right\|_1, \; s.t. \; Ax = b,$$

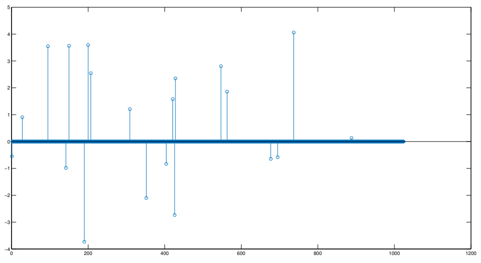
we change our constraint and solve

$$min_x \left\|x\right\|_1, \; s.t. \; \|Ax - b\|_2 \leq \epsilon,$$

We again denote the calculated solution as $\hat{x}$. Figure 2 shows the comparison between the true solution x and calculate solution $\hat{x}$ in one experiment. Note that we used the same x as in demo 1. This time, the error is larger due to the noise, but $\hat{x}$ is still close enough to the true solution.

```
>> norm(x-xhat)

ans =

    0.2416
```

We can still verify that $\hat{x}$ has the same sparsity pattern as x.


(a) The true solution x

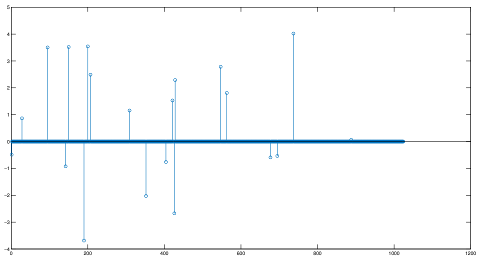

(b) The calculated solution $\hat{x}$

Figure 3: Comparison of the true solution x and result $\hat{x}$

```
>> setdiff(find(x~=0), find(xhat~=0))

ans =

   Empty matrix: 0-by-1
```

The code solves the problem

$$\|Ax-b\|_2 + \lambda \|x\|_1$$

where λ needs to depend on the noise level in order to retrieve the sparse solution. This is discussed in more detail in section 3.1.

2.3 Demo 3: 1D Compressed Sensing with Redundancy

Another question that can be asked is the following: Given a signal f that is sparse in some basis, for example

$$f = \begin{bmatrix} I & D \end{bmatrix} x$$

does one need to know all of f to solve for x?

So far, we have been solving

$$min_x \|x\|_1 \, s.t. Ax = b$$

Can we recover f with a small number of linear measurements of the form $y_i^T f$ for $i = 1, 2, \ldots m$? The answer is usually "no". However, if x is sparse, we can choose some small number of y_i to recover f.

For this demo, we will use random mixings of sparse b. Despite not having all parts of b, can we still recover x? We mix b by multiplying it with a 63-by-512 matrix G with normal random entries. G can therefore be seen as a downsampling and randomising operator. Notably, G is not invertible.

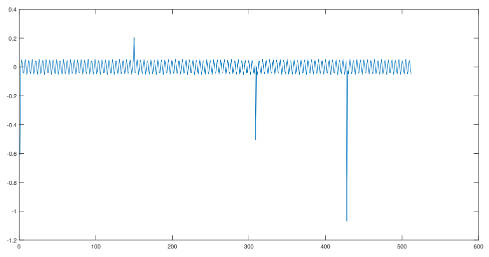
The following shows that the structure of Gb is very dissimilar to that of b, and has far fewer samples compared to b.

We then run BPDN on GA and Gb. The following stem plot shows that the solution is very close to the true solution:

We can verify that the solution recovers all the zeros of the true solution:

```
>> setdiff(find(x~=0), find(xhat~=0))

ans =

   Empty matrix: 0-by-1
```

We can also check that the coefficients of $\hat{x}$ are very close to x:


(a) The original signal b

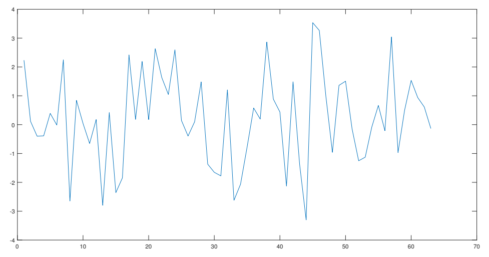
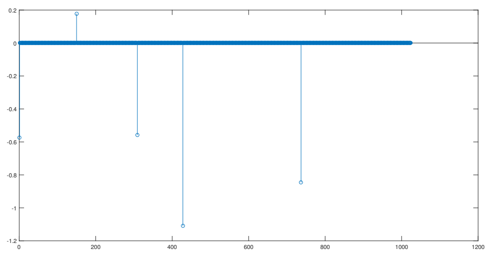

(b) The mixed signal Gb

Figure 4: Comparison of b and the mixed signal Gb


(a) The true solution x

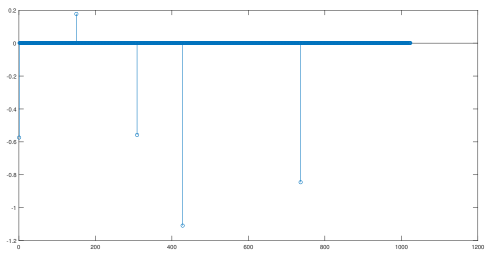

(b) The calculated solution $\hat(x)$

Figure 5: Comparison of the true solution x and result $\hat{x}$

```
>> norm(x-xhat)

ans =

   5.6493e-10
```

This demo shows the core idea behind compressive sensing: one does not need as many samples as one thinks one needs to recover the signal. This holds when the signal has a sparse representation that we can leverage. For example, in this demo the signal has a sparse representation in time space and DCT-space.

2.4 Demo 4: 1D Compressed Sensing with Incoherent Matrix

The Nyquist-Shannon sampling theorem [1] tells us that we need to sample in time space at every $1/(2b)$ seconds for a signal consisting of frequencies that are at most b Hz. However, if b is sparse, we do not need that many samples.

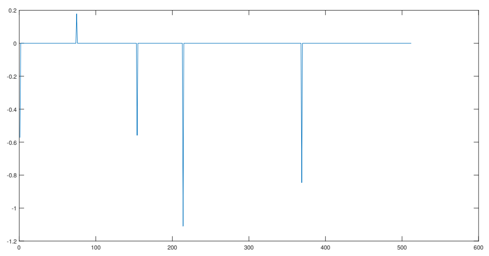
In this demo, $f = \left[D\right] x$. We pick 32 random entries of b and rearrange them randomly using the matrix DS. We then take $b2 = DS * b$.

We then run BPDN on $DS * A$ and $b2$. The following stem plot shows that the solution is very close to the true solution:

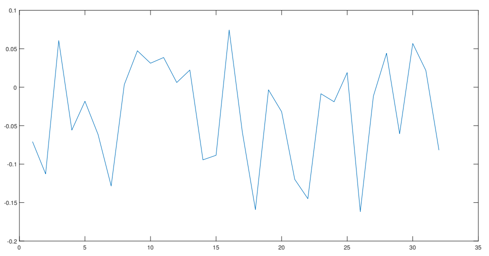
We can verify that the solution recovers all the zeros of the true solution:

```
>> setdiff(find(x~=0), find(xhat~=0))

ans =

   Empty matrix: 0-by-1
```

We can also check that the coefficients of $\hat{x}$ are very close to x:

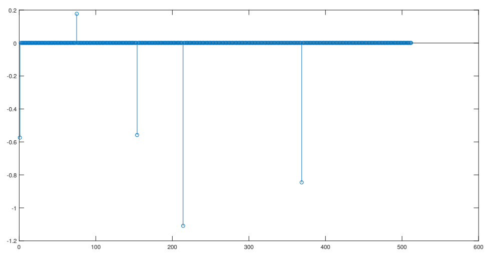


(a) The original signal b

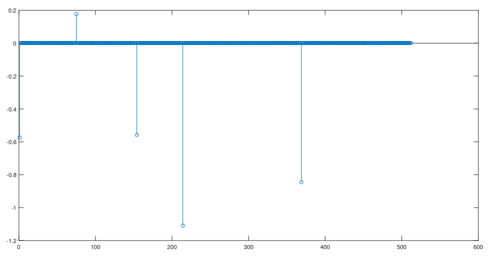


(b) The randomly sampled signal $b2$

Figure 6: Comparison of b and randomly sampled signal $b2$



(a) The true solution x



(b) The calculated solution $\hat(x)$

Figure 7: Comparison of the true solution x and result $\hat{x}$

```
>> norm(x-xhat)

ans =

   1.2413e-16
```

Even given just 32 samples from b, we can exactly recover all the coefficients. This is because the matrix that b is multiplied with is incoherent with the DCT matrix.

If we want to reconstruct b, we can take $\hat{b} = A * \hat{x}$. A quick check shows that $\hat{b}$ is very close to b:

```
>> norm(A*xhat - b)

ans =

   1.8127e-16
```

Therefore, the important thing is to not sample based on the size of the representation, but on the information content of the matrices. Because of this, one needs to know how the subsampling is done, for example what G is in demo 3 and DS is in demo 4.

Another question that might arise in practical usage is how does one know how sparse x is going to be in order to tweak the algorithm. In practice, compressed sensing finds application in areas such as imaging which have signals of finite energy. Therefore, one can assert that one has a sparse representation in an appropriate basis, such as a wavelet basis.

3 Working with Noisy Data

3.1 Quadratic Programming

In the previous lecture, we were able to write the solution to the sparse optimization problem $Ax = b$ as a linear program. We were trying to solve

$$\min \|x\|_1, \text{ s.t. } Ax - b = 0$$

represented this as the following linear program,

$$\min_{\begin{bmatrix} u & v \end{bmatrix}^T \in \mathbb{R}^{2n}} 1^T \begin{bmatrix} u \\ v \end{bmatrix}, \text{s.t. } \begin{bmatrix} A & -A \end{bmatrix} \begin{bmatrix} u \\ V \end{bmatrix} = b, \text{where } u, v \geq 0$$

and solved it using simplex or interior point methods.

If our output b is noisy, our problem can no longer be solved by a linear program.

$$\min \|x\|_1, \text{ s.t. } Ax + \sigma z - y = 0,$$
$$\text{where } \sigma z \text{ denotes Gaussian random noise magnified by a constant}$$

If we relax the constraint to

$$\min \frac{1}{2} \|Ax - b\|_2^2 + \lambda \|x\|_1$$

then we can rewrite this as a quadratic program

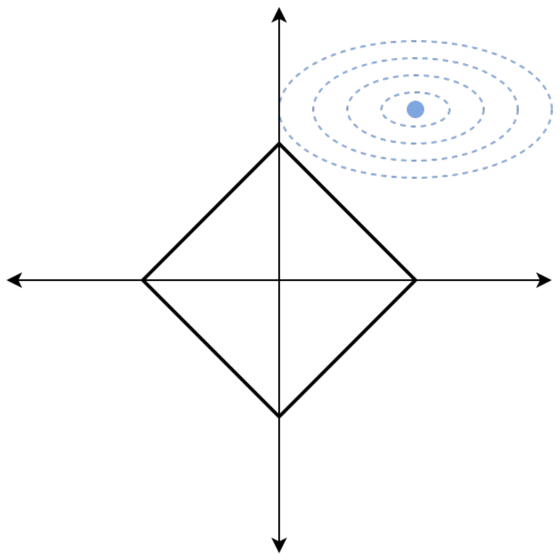
$$\min_{u,v,r} \lambda 1^T (u + v) + \frac{1}{2} r^T r,$$
$$\text{such that } \begin{bmatrix} A & -A \end{bmatrix} \begin{bmatrix} u \\ V \end{bmatrix} + r = b,$$
$$\text{where } u, v \geq 0, \text{residual } r = \|Ax - b\|_2^2$$

The variable λ is used to adjust the objective function between sparsity and accuracy. A smaller value of λ will decrease the influence of $\|x\|_1$, thus generating a more accurate solution. A larger value will generate a more sparse solution.

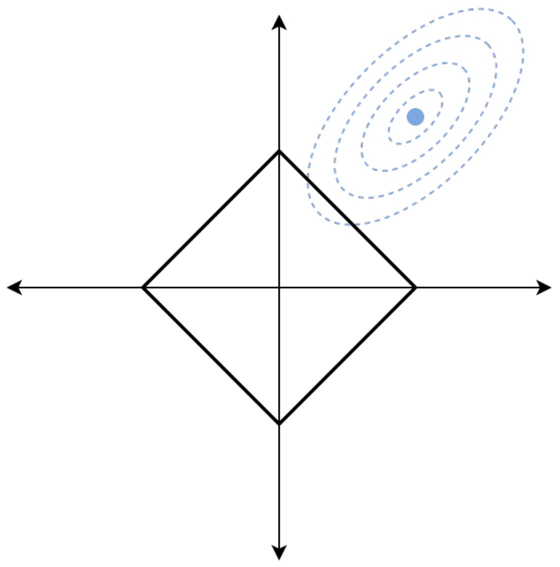
Problems in this form are called Basis Pursuit Denoising (BPDN), and are a class of quadratic programs. As with linear programs, there exist solvers for BPDN and quadratic programs in general using interior point and conjugate gradient methods.

3.2 Visual Representation

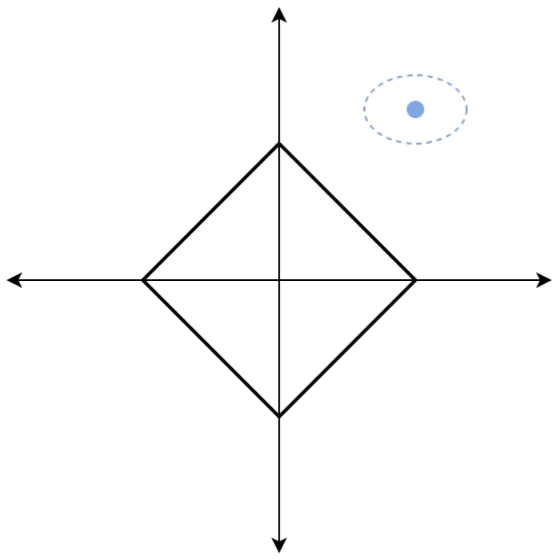
In the $Ax = b$ problem, we visualized the optimal sparse solution by growing the ℓ_1 norm until it intersected the line representing the solution. In a noisy scenario, we can instead visualize level sets of the solution at various ε. As we increase ε and expand the level sets, there might exist an x within the level set with minimum ℓ_1 norm.



However, this also means there are more cases with weaker solutions. For example, in the 2D case, if the level set happens to be oblique, then either the sparse solution is not accurate, or accurate solution is not sparse.



Finally, if we force epsilon to be too small, then there may be no good sparse solution at all.



3.3 Accuracy

Another way of representing BPDN is

$$\min \|x\|_1 \ \text{ s.t. } \ \|Ax - b\|_2 \le \varepsilon$$

If there exists a feasible solution x_0 that satisfies $\|Ax - 0 - b\|_2 \le \varepsilon$ and is sparse enough to satisfy

$$\|x_0\|_0 \le \frac{1}{4}\Big(1 + \frac{1}{\mu(A)}\Big)$$

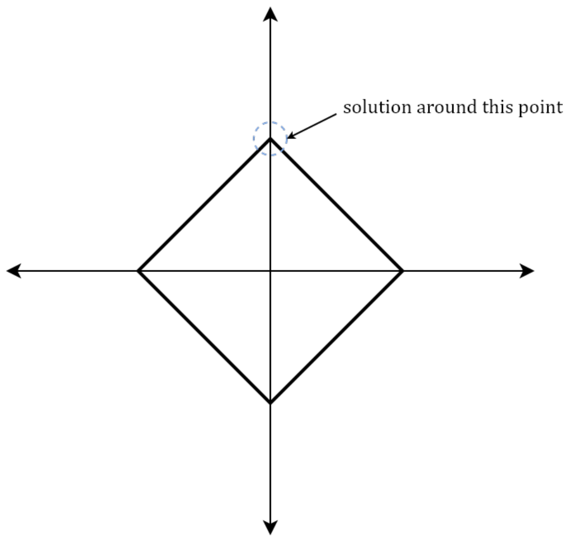
where $\mu(A)$ is the mutual coherence matrix A

$$\mu(A) = \max_{1 \le i,j \le n, i \ni j} \frac{|a_i \cdot a_j|}{\|a_i\| \, \|a_j\|}$$

then we can bound the accuracy of the recovered solutions.

$$\|\hat{x} - x_0\|_2^2 \le \frac{4\varepsilon^2}{1 - \mu(A)(4\,\|x_0\|_0 - 1)}$$

In the absence of noise, we said that if there exists a sparse enough solution to $Ax = b$, then we will be able to find it. Now if we have tighter bound on the sparsity, and it instead guarantees that the recovered solution will be within a certain distance from the true solution.



References

[1] Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.