EE4530 Applied Convex Optimization - Compressed Sensing Project 5

Kunlei Yu (5519764) & Xuan Gao (5232481)

1. Optimization problem formulation

The compressed sensing requires sparsity in the signal. This project uses a frequency domain sampling which is similar to the MRI scanners. For this project, the signal is given in both time domain and frequency domain. There are five variables in cs.mat: x, F_{us}, X_{us}, sampling mask, and n. The x is the original signal and X_{us} is the original signal in frequency domain. These two variables are related by $X_{us} = F_{us}x$**,** where the F_{us} is the Fourier transform matrix.

1.1 Preparation

The preparation of this project is the generation of the compressed signal by random frequency domain sampling. As it assumed that the x is the result of random sampling in frequency domain and the sampling mask is known, we created an incomplete Fourier matrix A to filter out the zero entries in the original sparse signal. The sampling mask scales down the dimension of the signal by 4 times. As a result, the compressed signal is generated as $b = Ax$, where b is compressed signal, A is incomplete Fourier matrix and x is the original spares signal.

1.2 Signal reconstruction algorithm design

Once the generation of compressed signal is done, reconstruction of original signal $\boldsymbol{x}$ starts. As the compression utilized the sparsity of the original signal, the problem can be interpreted as finding a sparse solution with the constraint of $b = Ax$. There are three candidates for the objective function in this optimization problem: L_2 norm, L_0 norm and L_1 norm. Although the L_2 norm has a most straightforward meaning (as its graph is a circle in 2-D and a ball in 3-D), the L_2 norm barely provides a sparse solution, which is inconsistent with the sparsity of the original signal. L_0 and L_1 norm both provide sparse solution as their intersection with a hyperplane is usually sparse. As the solution of L_0 norm is hard to find, the L_1 norm is the best choice for this project.

Therefore, the optimization problem is formed as:

$$\text{minimize } \|x\|_1$$

$$\text{subject to } Ax = b$$

which is a linear programming can be solved by many off-the shelf solvers (e.g., CVX) and algorithms (e.g., projected subgradient methods, interior-point methods).

2. Convex optimization problem implementation in MATLAB

We implement the proposed convex optimization problem in the off-the-shelf solver CVX. It successfully solves the problem, and the result is shown in figure 1. The number of iterations, total CPU time, and the algorithm CVX used are shown in table 1.

In figure 1, the red circles represent the original signal x. It is a 5-sparse signal and in this figure, we can see five non-zero points. The blue stars represent the recovery signal x' after the implementation in CVX solver. Almost for every point the red circle overlaps the blue star, which means the signal x can be successfully recovered by solving least $l1$-norm in the CVX solver.

Table 1 parameters of CVX solver implementation

Proposed convex optimization problem implemented in CVX solver	
Number of iterations	12
Total CPU time in seconds	0.28
Algorithm	Infeasible path-following algorithms

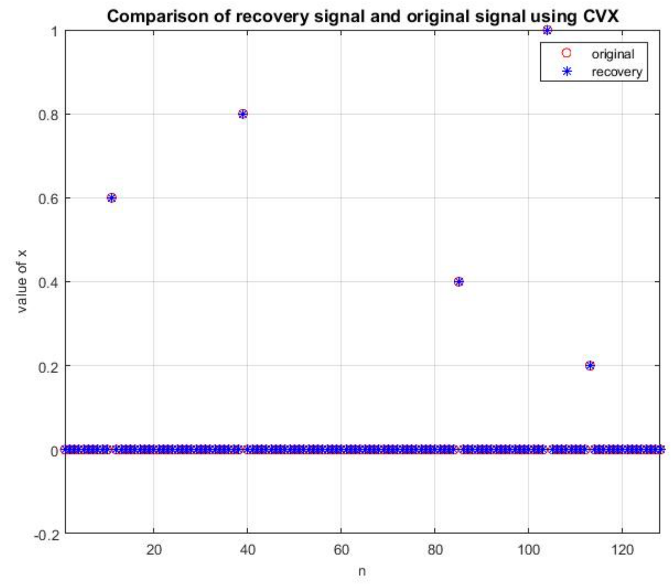

Figure 1 Comparison of recovery signal and original signal using an off-the-shelf solver CVX

3. Low-complexity algorithm implementation

3.1 Projected subgradient method

Subgradient methods are first-order methods for minimizing a nondifferentiable convex function. Projected subgradient method is adopted in our project to solve the convex optimization problem. C is the convex set.

$$\text{minimize } f(x) = \|x\|_1$$

$$\text{subject to } x \in C, C = \{x|Ax = b\}$$

This method is given by:

$$x^{k+1} = \Pi(x^k - \alpha_k g^k)$$

Where g^k is the subgradient of f at x^k. Π is Euclidean projection on C.

Let $x'^{(k+1)} = x^k - \alpha_k g^k$, which is a standard subgradient update before the projection on the convex set. Then, we have:

$$\begin{aligned}\left\|x'^{(k+1)} - x^*\right\|_2^2 = \left\|x^k - \alpha_k g^k - x^*\right\|_2^2 &= \left\|x^k - x^*\right\|_2^2 - 2\alpha_k g^{(k)T}\left(x^k - x^*\right) + \alpha_k^2\left\|g^k\right\|_2^2 \\ &\le \left\|x^k - x^*\right\|_2^2 - 2\alpha_k\left(f(x^k) - f^*\right) + \alpha_k^2\left\|g^k\right\|_2^2\end{aligned}$$

Since the point moves closer to the optimal point when we project a point onto C, then:

$$\begin{aligned}\left\|x^{k+1} - x^*\right\|_2^2 = \left\|\Pi\left(x'^{(k+1)}\right) - x^*\right\|_2^2 &\le \left\|x'^{(k+1)} - x^*\right\|_2^2 \\ &\le \left\|x^k - x^*\right\|_2^2 - 2\alpha_k\left(f(x^k) - f^*\right) + \alpha_k^2\left\|g^k\right\|_2^2\end{aligned}$$

Since $C = \{x|Ax = b\}$ is affine and A is fat, the projection operator is affine and shown below:

$$\Pi(x') = x' - A^T(AA^T)^{-1}(Ax' - b)$$

Then the subgradient update can be simplified as:

$$x^{k+1} = x^k - \alpha_k(I - A^T(AA^T)^{-1}A)g^k$$

As explained in 1.2, the optimization problem for compressed sensing can be formed as least l_1 norm:

$$\text{minimize } \|x\|_1$$

$$\text{subject to } Ax = b$$

Since $\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$ is non-differentiable. For finding the subgradient, $\|x\|_1$ can be represented as:

$$\|x\|_1 = max\{g^T x | g_i \in \{-1,1\}\}$$

$$g_i = \begin{cases} +1 & x_i > 0 \\ -1 & x_i < 0 = sign(x) \\ +1 \ or -1 & x_i = 0 \end{cases}$$

Then the subgradient rules can be applied. For choosing the s, we can choose $s = +1$ when $x > 0$ and $s = -1$ when $x < 0$. If $x = 0$, then s can be either $+1$ or -1. $g^T x$ now is differentiable and g is the subgradient.

The projected subgradient update is then:

$$x^{k+1} = x^k - \alpha_k (I - A^T (AA^T)^{-1} A) sign(x^k)$$

For the starting point, we can choose the least-norm solution: $x^1 = A^T (AA^T)^{-1} b$.

3.2 MATLAB implementation results

We implemented the proposed convex optimization problem in projected subgradient method. It successfully solved the problem, and the result is shown in figure 2 and figure 3. The number of iterations, total CPU time, and the algorithm used are shown in table 2. Compared to the results from CVX solver, much more iterations using projected subgradient method are needed, the total CPU time is much longer, and the accurate of the signal recovery is lower.

Table 2 parameters of projected subgradient method implementation

Proposed convex optimization problem implemented in a first-order method	
Number of iterations	10000
Total CPU time in seconds	6.1343
Algorithm	projected subgradient method

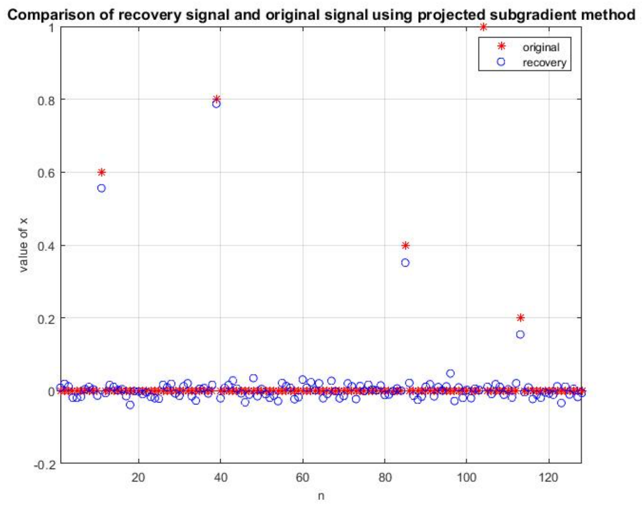

Figure 2 Comparison of recovery signal and original signal using projected subgradient method with iteration 3000

Firstly, we set the iteration number as 3000. The result is shown in figure 2. For every point, the red star appears near the blue circle. However, the red stars cannot fully overlap the blue circles, which means the recovery is not very accurate. Then, we increase the number of iterations to 10000, and the result is shown in figure 3. The recovery performance in 10000 iterations is better than that in 3000 iterations. For every point, the blue circle gets closer to the red start with more iterations. As the projected subgradient method is a simple first order low-complexity algorithm, it has restrictions on signal recovery and not exactly accurate.

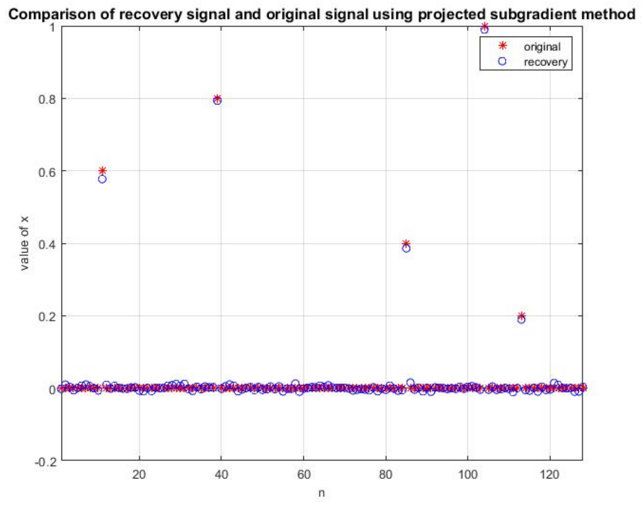

Figure 3 Comparison of recovery signal and original signal using projected subgradient method with iteration 10000

The convergence of the projected subgradient method for a L1 norm problem is shown in figure 4. Since the subgradient method is not a descent method, we choose to track the best point so far. For the step size, we use Polyak estimated step size rule $\gamma_k = \frac{100}{k}$. k is the iteration number. As figure 4 shows, $f_{best} - f^*$converges fast before 700 iterations, then it converges slowly and finally it reaches the limit of approximate 0.05. At each step, we set:

$$f_{best}^{k} = \min\{f_{best}^{k-1}, f(x^k)\}$$

Then we have:

$$f_{best}^{k} = \min\{f(x^1), \ldots, f(x^k)\}$$

$$convergence = f_{best} - f^*$$

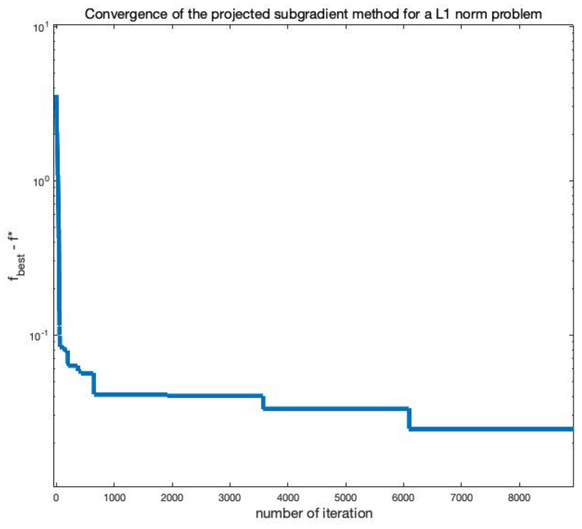

Figure 4 Convergence of the projected subgradient method, $f_{best} - f^*$ versus the number of iterations

APPENDIX – MATLAB codes

```
% EE4530 Applied Convex Optimisation Project 5

clc
clear all
load('cs.mat')

n = 128;
m = n/4;
freq = find(sampling_mask==1);
A = [real(F_us(freq,:)); imag(F_us(freq,:))];
X_us(find(X_us==0))=[];
b = [real(X_us); imag(X_us)]; % b = A*x;

% CVX implementaion
t_start1 = cputime;
cvx_begin
    variable x1(n);
    minimize(norm(x1,1));
    A*x1 == b;
cvx_end
f_1 = norm(x1,1);
t_end1 = cputime;

% least-norm solution for the starting point
```

```
x2 = A.'*pinv(A*A.')*b;
k = 10000;
stepsize = 100/k; % Polyak estimated step size rule

fbest = zeros(1,k+1);
fbest(1,1) = norm(x2,1);

t_start2 = cputime;
for i = 1:1:k
    x2 = x2 - stepsize*(eye(n)-(A.'*pinv(A*A.')*A))*sign(x2);
    fbest(1,i+1) = min(fbest(1,i),norm(x2,1));
end
t_end2 = cputime;
tot_cputime = 0.28/(t_end1-t_start1)*(t_end2-t_start2);

figure(1)
plot(1:n,x,'ro',1:n,x1,'b*'), legend('original','recovery');
title('Comparison of recovery signal and original signal using
CVX','FontSize',12)
xlabel('n','FontSize',10);
ylabel('value of x','FontSize',10);
axis([1 128 -0.2 1]);
grid on

figure(2)
plot(1:n,x,'r*',1:n,x2,'bo'), legend('original','recovery');
title('Comparison of recovery signal and original signal using projected
subgradient method','FontSize',12)
xlabel('n','FontSize',10);
ylabel('value of x','FontSize',10);
axis([1 128 -0.2 1]);
grid on

fplot = fbest(1,1:k)-fbest(1,k+1);
figure(3)
title('Convergence of the projected subgradient method for a L1 norm
problem','FontSize',15);
xlabel('number of iteration','FontSize',15);
ylabel('f_{best} - f*','FontSize',15);
axis([1 10000 10e-2 10]);
semilogy(fplot,'LineWidth',5);
```