% %

**AU311, Pattern Recognition Tutorial (Fall 2020)** **Homework: 1. Linear Regression**

# 1. Linear Regression

*Lecturer: Xiaolin Huang*

**Problem 1**

We know that $Y = Xw$ with $X \in R^{m\times n}$ could be solved if the rank of $X$ is larger than the dimension of $w$. Now if $m < n$, compressive sensing is still possible to find the real solution, if it is sparse. To verify the recovery capability, you could consider the following situation:

$n = 40, n = 100$

the sparsity $K$, i.e., the number of non-zero components of $\bar{w}$ (the underlying signal), varies from 1 to 100.

Here, the elements of $X$ follow a Gaussian distribution; the non-zero components of $\bar{w}$ are randomly selected uniformly; and for those non-components, their value follows a Gaussian distribution.

- Suppose the solution of the following problem is $\hat{w}$,

$$\min_{w} \quad \|w\|_1, \quad \text{s.t. } Y = Xw,$$

  then the ratio of *successful recovery* ($\|\hat{w} - \bar{w}\| < 10^{-2}$) could be calculated.

- Furthermore, if there is noise on observations, i.e., $Y = Xw + n$ with $n$ being a Gaussian noise with zero-mean (the signal-to-noise ratio is 20), then we need the following problem, of which the solution is denoted by $\hat{w}$,

$$\min_{w} \quad \lambda\|w\|_1 + \|Y - Xw\|_2^2,$$

  and the *recovery accuracy* could be measured by $\|\hat{w} - \bar{w}\|_2 / \|\bar{w}\|_2$

You are required to report

i) Matlab code (using cvx for the first one and using iterative soft thresholding for the second).

ii) the ratio of successful recovery for noise-free case;

iii) the recovery accuracy for noise-corrupted case (you need to choose a good $\lambda$ by cross-validation).

**Answer 1**

In the problem, we are required to use cvx, a tool of Matlab. CVX is a modeling system for constructing and solving disciplined convex programs (DCPs).CVX is implemented in Matlab, effectively turning Matlab into an optimization modeling language. Model specifications are constructed using common Matlab operations and functions, and standard Matlab code can be freely mixed with these specifications.

Using CVX, we can easily solve this problem with a few codes.To verify the correctness of the algorithm, I execute the code 100 times and try to verify the result of each calculation. If the result meets the formula ($\|\hat{w} - \bar{w}\| < 10^{-2}$), I consider this calculation is accurate, and finally calculate the correct rate after 100 executions

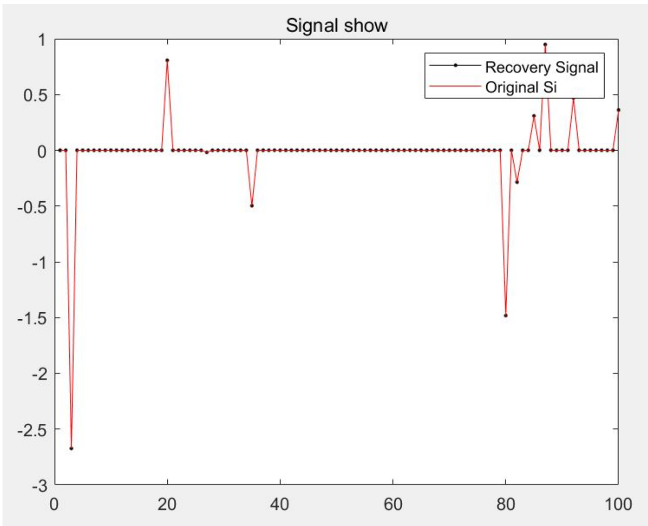


FIG. 1: K=0.9, successful recovery ratio=94%

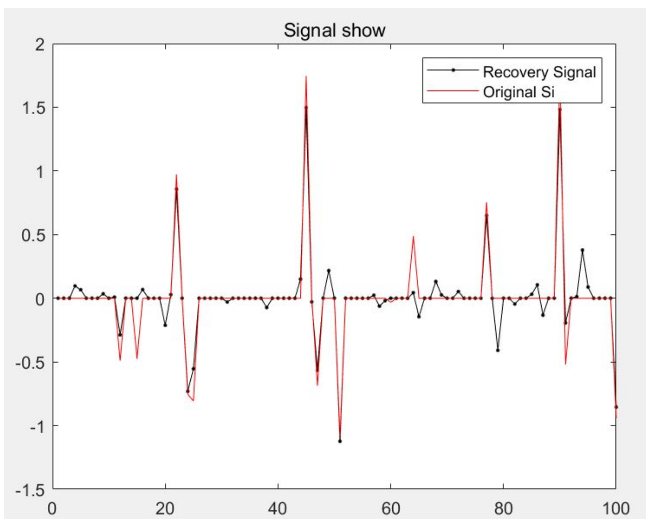


FIG. 2: K=0.86, successful recovery ratio=47%

And as shown in the table below here are some experimental results showing the correlation between sparsity and accuracy .We can see that when the sparsity is less than 0.8, the recovery success rate is approximately 0%.When the sparsity is greater than 0.8, the recovery success rate gradually increases, and when the sparsity is greater than 0.92, the recovery success rate is close to 100%.

In particular, the sparsity here refers to the ratio of zero.

TABLE I: Accuracy and Sparsity

| *Sparsity* | 0.79 | 0.80 | 0.81 | 0.82 | 0.83 | 0.84 | 0.85 | 0.86 | 0.87 | 0.88 | 0.89 | 0.90 | 0.91 | 0.92 | 0.93 | 0.94 | 0.95 | 0.96 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Accuracy/%* | 0 | 2 | 6 | 7 | 10 | 15 | 45 | 47 | 59 | 75 | 88 | 94 | 96 | 99 | 100 | 100 | 100 | 100 |

In the way, we can solve the problem1,and the main code is placed in the appendix.

**Answer 2**

In the second problem, we are required to use iterative soft thresholding which has a wide range of applications when solving sparse matrices.

Here,I use the wthresh function to solve the problem.Y = wthresh(X,sorh,lamda) returns the soft or hard thresholding, indicated by sorh, of the vector or matrix X,and lamda is the threshold value.

In the soft thresholding problem, lamda is a quiet important parameter which may have a greater impact on the results.After consulting some papers, I choose the dynamic threshold method mentioned in some papers and the final lamda varies according to the test matrix.

$$\lambda = 0.1 * \max \ (abs(X^T * Y))$$

In this way, $\lambda$ can be adjusted dynamically instead of a static parameter.To verify experiment results I use the *recovery accuracy*$(\|\hat{w} - \bar{w}\|_2/\|\bar{w}\|_2)$

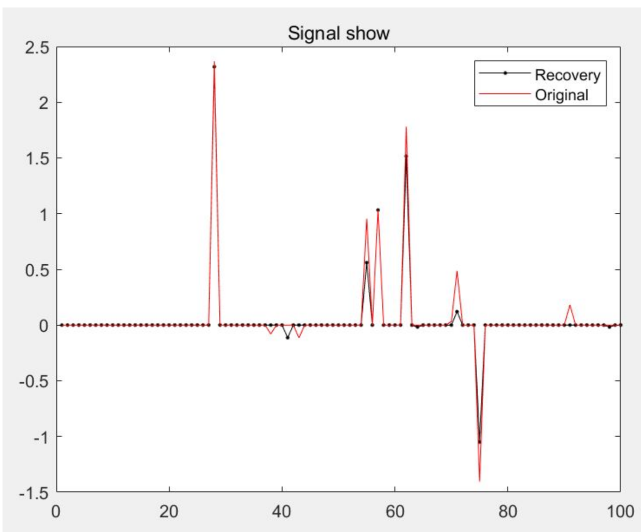


FIG. 3: K=0.9 and recovery accuracy=0.38

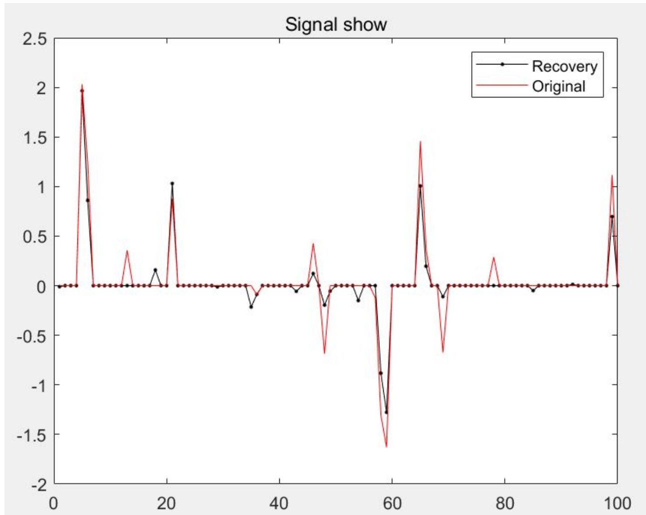


FIG. 4: K=0.85 and recovery accuracy=0.47

And as shown in the table below here are some experimental results showing the correlation between sparsity and accuracy .We can see that when the sparsity is less than 0.5, the recovery accuracy is approximately 0.8%.When the sparsity is greater than 0.8, the recovery accuracy gradually decreases, and get the best value around 0.9, the recovery accuracy is close to 0.38%.

In particular, the sparsity here refers to the ratio of 0

TABLE II: Accuracy and Sparsity

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Sparsity* | 0.4 | 0.42 | 0.44 | 0.46 | 0.48 | 0.50 | 0.52 | 0.54 | 0.56 | 0.58 | 0.60 | 0.62 | 0.64 | 0.66 | 0.68 |
| *Accuracy*/% | 0.83 | 0.833 | 0.81 | 0.80 | 0.81 | 0.80 | 0.79 | 0.78 | 0.76 | 0.76 | 0.75 | 0.75 | 0.74 | 0.69 | 0.68 |
| *Sparsity* | 0.70 | 0.72 | 0.74 | 0.76 | 0.78 | 0.80 | 0.82 | 0.84 | 0.86 | 0.88 | 0.90 | 0.92 | 0.94 | 0.96 | 0.98 |
| *Accuracy*/% | 0.64 | 0.67 | 0.61 | 0.61 | 0.55 | 0.56 | 0.52 | 0.49 | 0.45 | 0.40 | 0.38 | 0.41 | 0.44 | 0.53 | 0.98 |

In the way, we can solve the problem2,and the main code is placed in the appendix.

**Conclusion**

I complete the code of the lab and improved on the basis. From the test in my PC, I found my code could get good accuracy easily using the two methods.

Now is the ending time, first I want to thank for the help from teaching assistants and our professor Xiaolin Huang. Through this homework, I have attained lots of skills and experiences. The heuristic function give us great inspiration. I gain a lot.

I have added an adjunct of source codes with this test report, Thanks again!

## X. EXPERIMENT CODE

This section contains my code block of this homework.

Answer1:

```
% @Author C_zihao
% @Student_id 518021911187
% @Address chaizihao@sjtu.edu.cn
% @Data  2020.11.26

clear all; close all; clc;
%% Setting parameters
m = 40; n = 100;
succ_count = 0;
K = 0.92; test_count = 100;

%% Execute m calculation accuracy
for i = 1:test_count
    X = randn(m, n);
    %Generate a matrix with a sparsity of k according to the requirements of Q1
    zeros_mat = sparse(K * n, 1);
    gauss_mat = randn(n - K * n, 1);
    spilce_mat = cat(1, zeros_mat, gauss_mat);
    rowrank = randperm(size(spilce_mat, 1));
    w = spilce_mat(rowrank, :);
    %Start solve the problem
    cvx_begin quiet
    variable w_t(n, 1);
    Y = X * w;
    minimize(norm(w_t, 1));
    subject to
    Y == X * w_t;
    cvx_end

    if (norm(w_t - w, 2) < 0.01)
        succ_count = succ_count + 1;
    end
end

%% draw
figure;
plot(w_t, 'k.-');
hold on;
plot(w, 'r');
legend('Recovery Signal', 'Original Si')
title('Signal show');
Accuracy = vpa(double(succ_count / test_count) * 100, 4);
disp(['Successful Recovery Accuracy', '  =  ', char(Accuracy), '%'])
```

Answer2:

```matlab
% @Author C_zihao
% @Student_id 518021911187
% @Address chaizihao@sjtu.edu.cn
% @Data  2020.11.27

clear all; close all; clc;
%% Setting parameters
m = 40; n = 100;
K = 0.9; test_count = 100;
Accuracy = 0;

for i = 1:test_count
    %Generate a matrix with a sparsity of k according to the requirements of Q1
    X = randn(m, n); X = orth(X')';
    zeros_mat = sparse(K * n, 1);
    gauss_mat = randn(n - K * n, 1);
    spilce_mat = cat(1, zeros_mat, gauss_mat);
    rowrank = randperm(size(spilce_mat, 1));
    w = spilce_mat(rowrank, :);

    x = zeros(m, 1);
    Gauss_noisy = awgn(x, 20);
    Y = X * w + Gauss_noisy;
    %% recover signal
    lamda = 0.1 * max(abs(X' * Y));
    x_r = zeros(100, 1);
    tmp_loop = 0;
    while 1
        x_pre = x_r;
        x_r = wthresh(x_r + X' * (Y - X * x_r), 's', lamda);
        if norm(Y - X * x_r) < 1e-10 || norm(x_r - x_pre) < 1e-10 || tmp_loop>100000
            break;
        end
        tmp_loop=tmp_loop+1;
    end
    Accuracy = norm(x_r - w, 2) / norm(w, 2) + Accuracy;
end
Accuracy = vpa(Accuracy / test_count, 5);
disp(['Recovery Accuracy', '  =  ', char(Accuracy)])
%% draw
figure;
plot(x_r, 'k.-');
hold on;
plot(w, 'r');
legend('Recovery', 'Original');
title('Signal show');
```