Recover Sparse Signals from Under-Sampled Observations.

Date. 1.

1. sparse signal: a signal contains only a small number of non-zero elements compared to its dimension,

for sparse signal, ① it has good performance in comparession,

② kernals can be used for efficient data analysis

③ general signals can be transformed into sparse signals through simple operations.

2. For a sparse signal, its supple set is:

$supp(\underline{x})$ = index of non-zero entries.

$$S = \{ i, \ x_i \neq 0 \}.$$

the sparsity of $\underline{x}$ is the size of S.

For $\underline{1} = supp(\underline{x})$, $\underline{A}_{\underline{1}}$ is just the compact vector constructed by $\underline{1}$, then.

$$\underline{y} = \underline{A}\,\underline{x} = \underline{A}_{\underline{1}}\,\underline{x}_{\underline{1}}$$

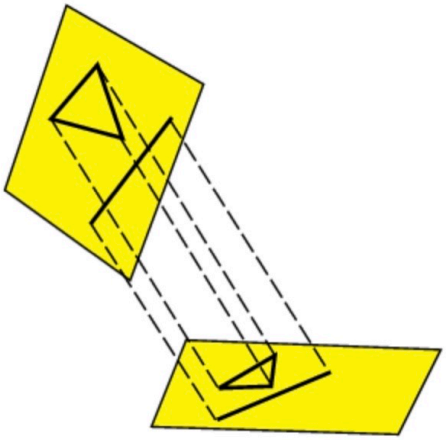
Hard thresholding function $H_s(x)$ $(x \in R^n, \ s < n)$

Remain the S largest entries (in magnitude) and set all the other entries to 0.

2. Projection:

A projection is the transformation of points and line in one plane onto another plane by connecting corresponding points on the two

planes with parallel lines.



Date 2.

Exercise 1.

according to the question.

$A \in R^{m \times n}$ is a random Gaussian matrix, and need to column normalization.

编辑器 - E:\Desktop\Lab\WD\wd1.m

wd1.m

```
1 —  close all;
2 —  clc;
3 —  m=128;
4 —  n=256;
5 —  A=normc(rand(m,n));
6 —  x=randn(n,1);
7 —  y=A*x;
8 —  x1=pinv(A)*y;
9 —  x2=A\y;
```

we can get the result:

	$\underline{x}$	$\hat{x}_1$	$\hat{x}_2$
	1	2	3
1	1.3170	0.8990	0
2	-0.3726	-0.0948	0
3	1.7504	1.3314	-1.8612
4	1.2969	0.8569	4.8958
5	1.3646	0.3297	-2.3347
6	0.6011	0.7239	-2.7327
7	-1.4182	-1.1342	-1.3405
8	-1.1523	-0.7381	0
9	-0.7873	-0.2945	-0.2907
10	0.1265	-0.2987	0.0011
11	-0.7378	-0.1526	0
12	-0.4158	-0.6667	-4.6947
13	0.4423	0.0923	0
14	0.4003	0.3706	0
15	0.3899	0.3606	0
16	0.1400	-0.1092	0
17	0.9496	0.6087	4.0248
18	0.8588	-0.1983	-0.8858
19	-0.4334	0.3871	3.4714
20	-1.5787	-0.5323	0
21	0.8568	0.3985	0
22	0.7340	1.0407	1.0612
23	0.2676	0.9329	1.9010
24	0.1300	0.4593	0
25	1.4696	0.6356	0
26	0.1994	0.2926	-1.5145
27	-0.5492	-0.5539	0.2523
28	1.6112	1.3741	0
29	-1.1482	-0.2856	0
30	1.0402	0.4745	0
31	-1.2368	-0.0825	2.0668

we can see that $\underline{x}$, $\hat{x}_1$, $\hat{x}_2$, these three vectors are distinct.

for $\underline{y} = \underline{A}\,\underline{x}$, where $\underline{y} \in R^m$, $\hat{x} \in R^n$ and $m < n$,

there are many different solutions.

Date. 3.

1. Greedy Algorithm.

a greedy Algorithm is an algorithm paradigm that follows the problem solving heuristic of making the locally optimal choice at each stage with the intent of finding a global optimum

2. Orthogonal Matching Pursuit (omp) Algorithm:

select one column that is most correlated to the current residue from the dictionary matrix $\underline{A}$ at each step, then add it to current selected column set. the current residue is obtained by projecting the observation $\underline{y}$. when the stop condition satisfies, the $\underline{y}$ can be generated.

3. the Subspace Pursuit (sp) Algorithm.

SP improves the omp. in SP, the selected column in each step may be abandoned in next iteration. However, the selected column in OMP will be kept. therefore, the SP has backtracing, it has low computation complexity, especially for reconstruction of relatively sparse sigoal And it has similar reconstruction accuracy to the linear LP algorithm.

4. The Iterative Hardthresholding (IHT)

for IHT. we first substantiate that there exists a threshold. it can find the optimal sparse signal x by continuously iterating with the hardthresholding equations. It learns from the residue and optimize the result in each iteration

Date. 4. 5.

For these three Algorithms:

Exercise 2.

```
%------OMP Algorithm
function[x_OMP]=OMP(S,A,y)
x_OMP=zeros(256,1);
S_set=[];
yr=y;
for l=1:S
    H1=H(1,A.'*yr);
    S_set = union(S_set,supp(H1));
    x_OMP=zeros(256,1);
    x_OMP(S_set)=A(:,S_set)\y;
    %judge stop
    yr_pre=yr;
    yr=y-A*x_OMP;
    if norm(yr,2)/norm(y,2)<1e-6||norm(yr,2)>norm(yr_pre,2)
        break;
    end
end
end
```

```
%-----IHT Algorithm
function[x_IHT]=IHT(S,A,y)
x_IHT=zeros(256,1);
yr=y;
while(1)
    x_IHT=H(S,x_IHT+A.'*(y-A*x_IHT));
    %judge stop
    yr_pre=yr;
    yr=y-A*x_IHT;
        if norm(yr,2)/norm(y,2)<1e-6||norm(yr,2)>norm(yr_pre,2)
        break;
        end
end
end
```

```
%------SP Algorithm
function[x_SP]=SP(S,A,y)
x_SP=zeros(256,1);
S_set=supp(H(S,A.'*y));
yr=resid(y,A(:,S_set));
for l=1:2*S
    S_exp=union(S_set,supp(H(S,A.'*yr)));
    b=zeros(256,1);
    b(S_exp)=A(:,S_exp)\y;
    S_set=supp(H(S,b));
    x_SP=zeros(256,1);
    x_SP(S_set)=A(:,S_set)\y;
    %judhe stop
    yr_pre=yr;
    yr=y-A*x_SP;
    if norm(yr,2)/norm(y,2)<1e-6||norm(yr,2)>norm(yr_pre,2)
        break;
    end
end
end
```

Here is the core algorithm for the exercise 2.

Result: For $\underline{x}$, $\underline{x}_1$, $\underline{x}_2$, they are totally different.

```
>> length(find(x~=0))

ans =

    12

>> length(find(x1~=0))

ans =

   256

>> length(find(x2~=0))

ans =

   128
```

for non-zero numbers.

$\underline{x}_1$ is 256, $\underline{x}_2$ is 128.

For $\underline{x}$, $\underline{x}\_omp$, $\underline{x}\_SP$, $\underline{x}\_IHT$,

```
>> x(find(x~=0))'

ans =

    0.7928    0.0257   -1.5363   -0.1471   -0.4593    0.0394    1.3952   -0.4365    1.6290   -0.0364   -0.7552    0.4716

>> x_OMP(find(x_OMP~=0))'

ans =

    0.7928    0.0257   -1.5363   -0.1471   -0.4593    0.0394    1.3952   -0.4365    1.6290   -0.0364   -0.7552    0.4716

>> x_SP(find(x_SP~=0))'

ans =

    0.7928    0.0257   -1.5363   -0.1471   -0.4593    0.0394    1.3952   -0.4365    1.6290   -0.0364   -0.7552    0.4716

>> x_IHT(find(x_IHT~=0))'

ans =

   -0.5178   -0.6283    0.6104   -1.1864   -0.7141    0.8483   -0.5145    0.8665   -0.5079    0.4998   -0.5289   -0.5047
```

we can see that

$\underline{x}\_omp$ and $\underline{x}\_SP$ are always equal to $\underline{x}$.

but $\underline{x}\_IHT$ always has error.

according to the above, we can see that under the sparsity of 12. OMP and SP can recover the orignal sparse signal.

Date 6.

according to the question,

we need to use success Rate comparison.

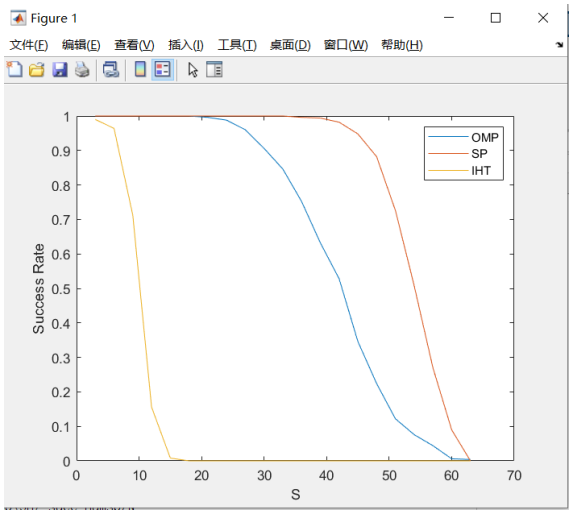
for Iteration: $S = 3, 6, 9 \cdots 63$,

@ Iteration $i = 1, 2, 3, \cdots 500$,

execute OMP. SP. and IHT Algorithm.
we just need to execute the Algorithm.
and if the result succeed, plus 1,
based on exercise 2.

```
    if ((norm(x_OMP-x,2)/norm(x))<1e-6)
        succ_numo=succ_numo+1;
    end
    if (norm(x_SP-x,2)/norm(x)<1e-6)
        succ_numsp=succ_numsp+1;
    end
    if (norm(x_IHT-x,2)/norm(x)<1e-6)
        succ_numiht=succ_numiht+1;
    end
end
succ_omp(Sn)=succ_numo/N;
succ_sp(Sn)=succ_numsp/N;
succ_iht(Sn)=succ_numiht/N;
```

and we can get the result:



we can see that SP has higher accuracy and IHT has the worst performance.