Data reconstruction based on incomplete inter- point distance

Name: Yunyun Zhang, Yiheng Chen
RIN : 661482565, 661482522

Abstract

The main topic is data reconstruction based on incomplete inter-point distance. In this project, given less than 4 percentage of the total original data which contains the distance between any two points on the surface of a 3D object, then recover the complete matrix from the given data using the low-rank matrix completion theory. To complete the goal, singular value thresholding (SVT) algorithm is used in this project. This paper is an explanation of the algorithm as well as the results.

1 Problems

Consider a set of points $\{x_1, ..., x_n\}$ sampled on a surface $\mathcal{M} \in \mathbb{R}^3$. Given small portion of the distance matrix $||x_i - x_k||_2, (i,j) \in \Omega$. Reconstruct $\{x_1, ..., x_n\}$ using the low-rank matrix completion theory.

Contributions:

We discuss and write the approaches for the problem together. Then each of us writes one half of the report. Yunyun Zhang writes section Abstract and Methodology. Yiheng Chen writes section Problem, Results, and Observation.

2 Methodology

2.1 Model

We want to find the lowest rank matrix X which matches the matrix M, for all entries in the set Ω of given entries. The mathematical formulation can be written as follows:

$$\min_x \text{rank}(X) \quad \text{s.t.} \quad X_{ij} = M_{ij} \quad \forall i,j \in \Omega \tag{1}$$

Since the problem is NP-hard, we introduce another optimization problem to solve the matrix completion. As [1] states, most of matrices M can be exactly reconstructed by solving the optimization problem

$$\min_x ||X||_* \quad \text{s.t.} \quad X_{ij} = M_{ij} \quad \forall i,j \in \Omega \tag{2}$$

where $||X||_*$ is the nuclear norm of the matrix M.

2.2 The Singular Value Thresholding Algorithm

2.2.1 The singular value shrinkage operator

Consider the Singular Value Decomposition of matrix X,

$$X = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \quad \mathbf{\Sigma} = \text{diag}(\{\sigma_i\}_{1<i<r}) \tag{3}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices.

For each $\tau \geq 0$, we define the soft-thresholding operator $\mathcal{D}_\tau$:

$$\mathcal{D}_\tau(X) := \mathbf{U}\mathcal{D}_\tau(\Sigma)V^*, \quad \mathcal{D}_\tau(\Sigma) = \text{diag}(\{\sigma_i - \tau_+\}) \tag{4}$$

where $\tau_+ = \max(0, t)$. According to the proof in [1], the above equation is equivalent to:

$$\mathcal{D}_\tau(Y) = \arg\min_X \frac{1}{2}||X - Y||_F^2 + \tau||X||_* \tag{5}$$

2.2.2 Shrinkage iterations

For a fixed $\tau > 0$, a sequence $\{\delta_k\} \geq 0$ of step sizes and $k = 1, 2, ...,$

$$\begin{cases} X^k = \mathcal{D}_\tau(Y^{k-1}), \\ Y^k = Y^{k-1} + \delta_k \mathcal{P}_\Omega(M - X^k) \end{cases}$$

Starting with Y_0 until a stopping criterion is reached.

2.2.3 Stopping criteria

As [1] proved, the sequence $\{X^k\}$ converges to the unique solution of an minimization problem:

$$\min_x \tau||X||_* + \frac{1}{2}||X||_F^2 \quad \text{s.t.} \quad X_{ij} = M_{ij} \quad \forall i, j \in \Omega \tag{6}$$

which is closely related to equation (2). The stopping criterion is motivated by the first-order optimality conditions or KKT conditions tailored to the equation (6). The solution to the equation (6) must satisfy following two condition.

$$\begin{cases} X = \mathcal{D}_\tau(Y), \\ \mathcal{P}_\Omega(X - M) = 0 \end{cases}$$

where Y is a matrix vanishing outside of Ω^c. By definition, the first equation is always be true. Then the stopping criteria equation defined as second equation is less than a predefined tolerance.

$$\frac{||\mathcal{P}_\Omega(X^k - M)||_F}{||\mathcal{P}_\Omega(M)||_F} \leq \epsilon \tag{7}$$

where ϵ is a fixed tolerance, and we set $\epsilon = 0.015$ in the project.

2.2.4 Choose value

We use step size $\delta = 1.2$ times the undersampling ratio from [1].
Parameter $\tau = 3000$ and increment $l = 5$ are chosen empirically.

2.2.5 Algorithm

Algorithm 1 Singular Value Thresholding algorithm

Step 0. Initialize δ, τ, l
Initialize $Y_0 = 0$
while not converge **do**
Step 1. Updating formula for X.
$X_k = \mathcal{D}_\tau(Y^{k-1})$
Step 2. Updating formula for Y.
$Y_k = Y_{k-1} + \delta_k \mathcal{P}_\Omega(M - X^k)$
end while

3 Results

The recovery is performed via the SVT algorithm , and we use

$$\frac{||\mathcal{P}_\Omega(X^k - M)||_F}{||\mathcal{P}_\Omega(M)||_F} \leq 0.015 \tag{8}$$

as a stopping criterion. During the process, we choose the step sizes are constant and we set $\delta = 1.2 * n * n/m$. The size of full unknown distance matrix is 401231*401231, and the given distance matrix size is 2601*2601. The relative error defined as

$$\frac{||(X^{opt} - M)||_F}{||(M)||_F} \tag{9}$$

The computational result is shown within 200 iteration, the iteration error is smaller than 0.015.

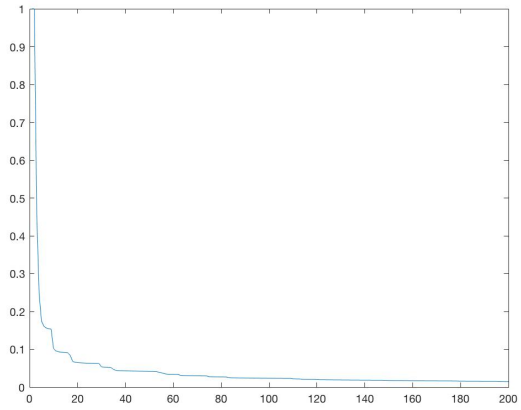

Figure 1: Relative error

The running time is approximate 5 minutes on my computer, but the running may varies, since it depends on the configuration of the laptop.

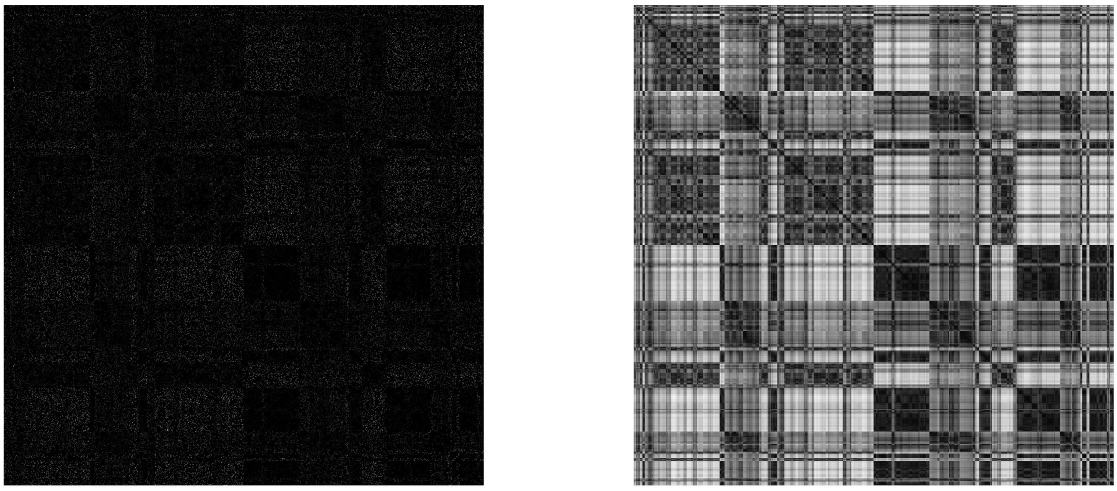

Figure 2: Left: Available distance matrix. Right: Reconstructed distance matrix

Figure 3: Two perspectives of reconstructed 3D cow

4 Observation and Conclusions

The recover result cow is relatively smooth. In this project we use singular value thresholding (SVT) algorithm to achieve data reconstruction based on incomplete inter-point distance problem. We successfully complete the matrix by using less than 4 percent of data from the whole matrix, and reconstructe the 3D cow. This paper introduce the singular value thresholding algorithm for matrix completion. This algorithm is easy to implement and surprisingly effective in terms of computational cost and storage requirement while the minimum nuclear-norm solution is also the lowest-rank solution.

References

[1] Jian-Feng Cai, Emmanuel J. Candès and Zuowei Shen. *A Singular Value Thresholding Algorithm for Matrix Completion*, Singapore, 2008.

Appendix

```
load Cow_PartialDistance
num_pt = size(Dist,1);

omega = ones(num_pt, num_pt);
for ind1 = 1:num_pt
    for ind2 = 1:num_pt
        omega(ind1, ind2) = ind1 == ind2 || Dist(ind1, ind2) ~= 0;
    end
end
omega = sparse(omega);

% Initialization
nonzero = nnz(omega);
tau = 3000; % parameter
delta = 1.2 * (num_pt * num_pt) / nonzero; % step size
Y = zeros(num_pt, num_pt); % Y0 = 0
Y = sparse(Y);
iteration = 200; % maximum iteration
l = 5; % increment
r = 0;

err = zeros(1, iteration);
err(1) = inf;


for itr = 1:iteration
    s = r + 1;
    while true
        U = sparse(num_pt, s);
        S = sparse(s, s);
        V = sparse(s, num_pt);
        [U, S, V] = svds(Y, s);

        s = s + l;
        if S(s-l, s-l) <= tau
            break
        end
    end
        r = max(find(diag(S) < tau));
        S = shrink(S, tau);
```

```
41          X = U * S * V';
42          Y = Y + delta * ((Dist - X).*omega);
43          err(itr) = norm((X - Dist).*omega,'fro')/norm(Dist.*omega,'fro');
44  end
45
46  X = (X + X.')/2; % symmetry; round off error
47  [a, ~] = size(X);
48  for i = 1:a
49      X(i,i)=0; % zeros along the diagonal
50  end
51
52  plot(err);
53  plt = mdscale(X, 3);
54  figure(2);
55  ViewMesh(plt, trg);
```