A Mathematical Introduction to Data Science **Mar. 13, 2024**

Homework 5. SDP Extensions of PCA and MDS

Instructor: Yuan Yao *Due: 1 week*

The problem below marked by * is optional with bonus credits.

1. *RPCA*: Construct a random rank-r matrix: let $A \in \mathbb{R}^{m\times n}$ with $a_{ij} \sim \mathcal{N}(0,1)$ whose top-r singular value/vector is λ_i, $u_i \in \mathbb{R}^m$ and $v_i \in \mathbb{R}^n$ $(i = 1, \ldots, r)$, define $L = \sum_{i=1}^r u_i v_i^T$. Construct a sparse matrix E with p percentage ($p \in [0,1]$) nonzero entries distributed uniformly. Then define
$$M = L + E.$$
   (a) Set $m = n = 20$, $r = 1$, and $p = 0.1$, use Matlab toolbox CVX or its Python version CVXPY (`https://www.cvxpy.org/install/index.html`) to formulate a semi-definite program for Robust PCA of M:
$$\begin{aligned} \min \quad & \frac{1}{2}(\mathrm{trace}(W_1) + \mathrm{trace}(W_2)) + \lambda \|S\|_1 \\ s.t. \quad & L_{ij} + S_{ij} = X_{ij}, \quad (i,j) \in E \\ & \begin{bmatrix} W_1 & L \\ L^T & W_2 \end{bmatrix} \succeq 0, \end{aligned} \tag{1}$$
   where you can use the matlab implementation in lecture notes as a reference;
   (b) Choose different parameters $p \in [0,1]$ to explore the probability of successful recover;
   (c) Increase r to explore the probability of successful recover;
   (d) * Increase m and n to values beyond 50 will make CVX difficult to solve. In this case, use the Augmented Lagrange Multiplier method, e.g. in E. J. Candes, X. Li, Y. Ma, and J. Wright (2009) "Robust Principal Component Analysis?". Journal of ACM, 58(1), 1-37. Make a code yourself (just a few lines of Matlab or Python) and test it for $m = n = 1000$. A convergence criterion often used can be $\|M - \hat{L} - \hat{S}\|_F / \|M\|_F \le \epsilon$ ($\epsilon = 10^{-6}$ for example).

2. *SPCA*: Define three hidden factors:
$$V_1 \sim \mathcal{N}(0, 290), \quad V_2 \sim \mathcal{N}(0, 300), \quad V_3 = -0.3V_1 + 0.925V_2 + \epsilon, \quad \epsilon \sim \mathcal{N}(0,1),$$
   where V_1, V_2, and ϵ are independent. Construct 10 observed variables as follows
$$X_i = V_j + \epsilon_i^j, \quad \epsilon_i^j \sim \mathcal{N}(0,1),$$
   with $j = 1$ for $i = 1, \ldots, 4$, $j = 2$ for $i = 5, \ldots, 8$, and $j = 3$ for $i = 9, 10$ and ϵ_i^j independent for $j = 1, 2, 3$, $i = 1, \ldots, 10$.

The first two principal components should be concentrated on (X_1, X_2, X_3, X_4) and (X_5, X_6, X_7, X_8), respectively. This is an example given by H. Zou, T. Hastie, and R. Tibshirani, Sparse principal component analysis, J. Comput. Graphical Statist., 15 (2006), pp. 265-286.

(a) Compute the true covariance matrix Σ (and the sample covariance matrix with n examples, say $n = 1000$);

(b) Compute the top 4 principal components of Σ using eigenvector decomposition;

(c) Use Matlab CVX toolbox or Python CVXPY to compute the first *sparse* principal component by solving the SDP problem

$$\begin{array}{ll} \max & \text{trace}(\Sigma X) - \lambda \|X\|_1 \\ s.t. & \text{trace}(X) = 1 \\ & X \succeq 0 \end{array}$$

Choose $\lambda = 0$ and other positive numbers to compare your results with normal PCA;

(d) Remove the first sparse PCA from Σ and compute the second sparse PCA with the same code;

(e) Again compute the 3rd and the 4th sparse PCA of Σ and compare them against the normal PCAs.

(f) * Construct an example with 200 observed variables which is hard to deal with by CVX. In this case, try the Augmented Lagrange Multiplier method by Allen Yang et al. (UC Berkeley) whose Matlab codes can be found at `http://www.eecs.berkeley.edu/~yang/software/SPCA/SPCA_ALM.zip`, or Python `scikit-learn` Sparse PCA package.

3. **Protein Folding:* Consider the 3D structure reconstruction based on incomplete MDS with uncertainty. Data file:

`http://yao-lab.github.io/data/protein3D.zip`

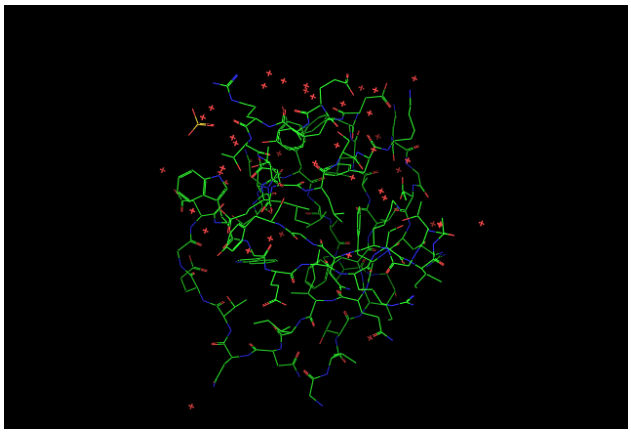

Figure 1: 3D graphs of file PF00018_2HDA.pdf (YES_HUMAN/97-144, PDB 2HDA)

In the file, you will find 3D coordinates for the following three protein families:

PF00013 (PCBP1_HUMAN/281-343, PDB 1WVN),

PF00018 (YES_HUMAN/97-144, PDB 2HDA), and

PF00254 (O45418_CAEEL/24-118, PDB 1R9H).

For example, the file `PF00018_2HDA.pdb` contains the 3D coordinates of alpha-carbons for a particular amino acid sequence in the family, YES_HUMAN/97-144, read as

`VALYDYEARTTEDLSFKKGERFQIINNTEGDWWEARSIATGKNGYIPS`

where the first line in the file is

97 V 0.967 18.470 4.342

Here

- '97': start position 97 in the sequence
- 'V': first character in the sequence
- $[x, y, z]$: 3D coordinates in unit $\mathring{A}$.

Figure 1 gives a 3D representation of its structure.

Given the 3D coordinates of the amino acids in the sequence, one can computer pairwise distance between amino acids, $[d_{ij}]^{l\times l}$ where l is the sequence length. A *contact map* is defined to be a graph $G_\theta = (V, E)$ consisting l vertices for amino acids such that and edge $(i, j) \in E$ if $d_{ij} \leq \theta$, where the threshold is typically $\theta = 5\mathring{A}$ or $8\mathring{A}$ here.

Can you recover the 3D structure of such proteins, up to an Euclidean transformation (rotation and translation), given noisy pairwise distances restricted on the contact map graph G_θ, i.e. given noisy pairwise distances between vertex pairs whose true distances are no more than θ? Design a noise model (e.g. Gaussian or uniformly bounded) for your experiments.

When $\theta = \infty$ without noise, classical MDS will work; but for a finite θ with noisy measurements, SDP approach can be useful. You may try the matlab package SNLSDP by Kim-Chuan Toh, Pratik Biswas, and Yinyu Ye, or the facial reduction speed up by Nathan Krislock and Henry Wolkowicz. For python users, you may try the Python version of CVX (CVXPY): `https://www.cvxpy.org/install/index.html`.