ECE 510: Mathematical Foundations of Machine Learning **Spring 2019**

Demo: Robust Principal Component Analysis

Instructor Name: John Lipor

Introduction

PCA as Nuclear Norm Minimization

In this demo, you will implement the *Principal Component Pursuit* (PCP) algorithm for robust principal component analysis using the alternating direction method of multipliers (ADMM) framework. Recall that PCA performs a low-rank approximation to the data matrix $X \in \mathbb{R}^{D \times N}$, which we formulated as

$$\begin{aligned} \hat{L} &= \arg\min_{L \in \mathbb{R}^{D \times N}} \|X - L\|_F^2 \\ \text{subject to} & \quad \operatorname{rank}(L) \leq r. \end{aligned}$$

The above formulation assumes we know an upper bound on the desired rank. If this value is unknown, we could solve the alternative formulation

$$\begin{aligned} \hat{L} &= \arg\min_{L \in \mathbb{R}^{D \times N}} \operatorname{rank}(L) \\ \text{subject to} & \quad \|X - L\|_F^2 \leq \varepsilon \end{aligned}$$

for some small value of ε that determines our reconstruction error. Unfortunately, as with minimizing the number of nonzero elements in a vector, minimizing the rank of a matrix is a non-convex problem that is NP-hard to solve. The solution is to again use a convex relaxation! Recall that the relaxation of $\|\cdot\|_0$ in the Lasso is $\|\cdot\|_1$, i.e., we went from minimizing the *total* number of nonzero elements to minimizing the *sum* of absolute values. In the case of matrices, we can view the rank as the total number of directions in which there is a nonzero component, which can be expressed as

$$\operatorname{rank}(X) = \left\| \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_{\min(D,N)} \end{bmatrix} \right\|_0 .$$

In words, since the rank of a matrix is the number of nonzero singular values, it is equivalent to the ℓ_0-"norm" of the vector of all singular values. With this in mind, the natural convex relaxation of rank is the ℓ_1-norm of the singular values, which is known as the *nuclear norm*

$$\|X\|_* = \sum_{i=1}^{\min(D,N)} \sigma_i .$$

Note that since $\sigma_i \geq 0$, this is exactly the ℓ_1-norm of the vector of singular values. With this definition, we can reformulate PCA as

$$\begin{aligned} \hat{L} &= \arg\min_{L \in \mathbb{R}^{D \times N}} \|L\|_* \\ \text{subject to} & \quad \|X - L\|_F^2 \leq \varepsilon, \end{aligned}$$

which is a convex problem that we do have a hope of solving.

Sparse Plus Low-Rank Models

Suppose the matrix X is corrupted by *sparse* outliers stored in a matrix $S \in \mathbb{R}^{D\times N}$ and we observe $Y = X+S$. In this case, we may wish to formulate an optimization problem that solves for the sparse and low-rank components separately

$$\begin{aligned} \hat{L}, \hat{S} &= \underset{L,S\in\mathbb{R}^{D\times N}}{\arg\min}\ \operatorname{rank}(L) + \lambda \|S\|_0 \\ \text{subject to} &\quad Y = L + S. \end{aligned}$$

This formulation has both the rank and ℓ_0-"norm" issues, so we again solve the convex relaxation of the problem, which is written as

$$\begin{aligned} \hat{L}, \hat{S} &= \underset{L,S\in\mathbb{R}^{D\times N}}{\arg\min}\ \|L\|_* + \lambda \|S\|_1 \\ \text{subject to} &\quad Y = L + S. \end{aligned} \tag{1}$$

This problem is known as *principal component pursuit* and is probably the most widely-considered formulation for robust PCA. In fact, nearly all robust PCA algorithms are variants of (1). In this demo, we will code the solution to the PCP problem using ADMM.

ADMM Iterations

The ADMM iterations for solving (1) are given below. In a homework, you may be asked to derive these, but for now simply note that the update on L involves another soft-thresholding operator that is performed only on the singular values of the appropriate matrix. This should reinforce the similarity between nulcear norm minimization and ℓ_1-norm minimization.

The augmented Lagrangian for this problem is

$$\mathcal{L}(L,S,Z) = \|L\|_* + \lambda \|S\|_1 + \langle Z, Y - L - S\rangle + \frac{\rho}{2}\|Y - L - S\|_F^2,$$

where the matrix inner product above is defined as $\langle A, B\rangle = \operatorname{tr}(A^T B)$ and Z is the matrix of Lagrange multipliers.

The update for the low-rank component L is

$$L_{k+1} = \mathcal{D}_{1/\rho}(Y - S_k + \frac{1}{\rho}Z_k) \tag{2}$$

which is the singular value thresholding operator defined as

$$\mathcal{D}_\tau(X) = U\mathcal{S}_\tau(\operatorname{diag}(\Sigma))V^T$$

where $X = U\Sigma V^T$ and $\mathcal{S}_\tau(x)$ is the soft-thresholding operator.

The update for the sparse component S is the soft-thresholding operator applied to the entire matrix

$$S_{k+1} = \mathcal{S}_{\lambda/\rho}(Y - L_{k+1} + \frac{1}{\rho}Z_k). \tag{3}$$

Finally, the update for the Lagrange multipliers is the usual gradient-descent type update

$$Z_{k+1} = Z_k + \rho(Y - L_{k+1} - S_{k+1}). \tag{4}$$

Task 1: Implement PCP

Your first task is to implement the above ADMM iterations to complete the `pcp.m` function. Follow the steps below in order.

- Complete the `st.m` (soft thresholding) and `svt.m` (singular value thresholding) files and test them using the script `threshTest.m`. **Hint:** Your singular value thresholding function can and should call your soft thresholding function.
- Raise your hand when you have successfully completed the thresholding functions.
- Integrate your `st` and `svt` functions to complete `pcp.m`. Test this on the script `syntheticTest.m`
- Raise your hand when you have successfully completed the PCP function.

Task 2: Test PCP on Benchmark Data

Your final task is to run your `pcp.m` algorithm on the included `lobby.mat` dataset using the script `lobbyTest.m`. This dataset includes a video of an office lobby in which a person walks through near the end. The stationary/background portion of the video is modeled as the low-rank component, while the person walking through is the sparse component. You should see that the second and third plots show the background only and person only. This is an example of using robust PCA for foreground-background separation.