MDS5210: Machine Learning

Final project—Robust Matrix Factorization

The goal of this project is to implement a robust version of the matrix factorization/PCA and build a machine learning system for removing the shadow of face images and extracting backgrounds form video frames.

Project description. We have studied the vanilla PCA/matrix factorization in class, which aims to factorize the data matrix $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ into $\boldsymbol{U}\boldsymbol{V}^\top$. The underlying idea for applying low-rank matrix factorization is that the data matrix $\boldsymbol{X}$ possesses intrinsically low-rank structures. We have studied the following very naive learning problem for matrix factorization

$$\underset{\boldsymbol{U}\in\mathbb{R}^{d\times r},\boldsymbol{V}\in\mathbb{R}^{n\times r}}{\text{minimize}} \quad \|\boldsymbol{X}-\boldsymbol{U}\boldsymbol{V}^\top\|_F^2. \tag{1}$$

As what we have learned, the above formulation is reasonable in the principle of maximum likelihood estimation when the residual error $\boldsymbol{X}-\boldsymbol{U}\boldsymbol{V}^\top$ follows i.i.d. Gaussian distribution. However, this is usually not the case in practice. Instead, we usually have many additional structural assumptions on the residual error $\boldsymbol{X}-\boldsymbol{U}\boldsymbol{V}^\top$. One of these assumptions is *sparsity*.

Suppose that we know the residual error should be sparse, one possible learning problem formulation is

$$\underset{\boldsymbol{U}\in\mathbb{R}^{d\times r},\boldsymbol{V}\in\mathbb{R}^{n\times r}}{\text{minimize}} \quad \|\boldsymbol{X}-\boldsymbol{U}\boldsymbol{V}^\top\|_0, \tag{2}$$

where the ℓ_0-quasinorm $\|\boldsymbol{a}\|_0$ counts the number of nonzero elements of vector $\boldsymbol{a}$ and is not a continuous function. The intuition for the formulation (2) is quite straightforward. Let us clarify it. Assume additionally

$$\boldsymbol{X} = \boldsymbol{L}^\star + \boldsymbol{S}^\star \tag{3}$$

where $\boldsymbol{L}^\star \in \mathbb{R}^{d\times n}$ is a low-rank matrix with rank $r \ll \min(d,n)$ (the underlying low-dimensional approximation of $\boldsymbol{X}$) and $\boldsymbol{S}^\star \in \mathbb{R}^{d\times n}$ is a sparse matrix (due to the assumption that the residual error is sparse). Using the fact that any rank-r matrix $\boldsymbol{L}^\star$ can be factorized as

$$\boldsymbol{L}^\star = \boldsymbol{U}^\star\boldsymbol{V}^{\star\top} \tag{4}$$

where $\boldsymbol{U}^\star \in \mathbb{R}^{d\times r}$ and $\boldsymbol{V}^\star \in \mathbb{R}^{n\times r}$. Thus, our goal is equivalent to find the underlying $\boldsymbol{U}^\star$ and $\boldsymbol{V}^\star$ given $\boldsymbol{X}$. Plugging $\boldsymbol{U}^\star$ and $\boldsymbol{V}^\star$ into (2) gives

$$\|\boldsymbol{S}^\star\|_0,$$

where we have used (3)–(4). The fact that $\boldsymbol{S}^\star$ is sparse, together with the sparsity promoting nature of ℓ_0-quasinorm, suggests that $(\boldsymbol{U}^\star,\boldsymbol{V}^\star)$ is likely to be one optimal solution of (2). Until now, we have justified the formulation (2) under the sparsity assumption of the residual error in the sense that solving it to the globally optimal solution is likely to provide us the underlying $(\boldsymbol{U}^\star,\boldsymbol{V}^\star)$ and hence $\boldsymbol{L}^\star = \boldsymbol{U}^\star\boldsymbol{V}^{\star\top}$.

However, the ℓ_0-quasinorm makes the learning problem (2) intractable. In practice, we often use the ℓ_1-norm to replace the ℓ_0-quasinorm. Finally, we can formulate the following learning problem:

$$\underset{\boldsymbol{U}\in\mathbb{R}^{d\times r},\boldsymbol{V}\in\mathbb{R}^{n\times r}}{\text{minimize}} \quad \|\boldsymbol{X}-\boldsymbol{U}\boldsymbol{V}^\top\|_1. \tag{5}$$

It is known that the formulation (5) is much more robust than the naive one (1) and it is called *robust matrix factorization*.

Design algorithms Towards solving (5), I would suggest you to use

- Subgradient method
- Alternating iteratively reweighed least squares (A-IRLS)

The information about iteratively reweighed least squares (IRLS) is available at `https://en.wikipedia.org/wiki/Iteratively_reweighted_least_squares`[1]. The A-IRLS algorithm is nothing else other than an alternating minimization algorithm with its subproblem *approximately* solved by IRLS. Certainly, you can design other algorithms for solving (5) and the requirement is that you need to have **two** learning algorithms.

Evaluate the learning algorithms on artificial dataset Before moving to real dataset, you will need to evaluate your algorithms on artificially generated dataset that satisfies (3) and (4). Please follow the instruction here to generate the data. You first generate the underlying low-rank matrix $\boldsymbol{L}^\star = \boldsymbol{U}^\star\boldsymbol{V}^{\star\top}$ by generating $\boldsymbol{U}^\star \in \mathbb{R}^{d\times r}$ and $\boldsymbol{V}^\star \in \mathbb{R}^{n\times r}$ with i.i.d. standard Gaussian entries. To generate $\boldsymbol{S}^\star \in \mathbb{R}^{d\times n}$, you first randomly select pdn locations. Then, you fill each of the selected locations with an i.i.d. mean 0 and variance 100 Gaussian entry, while the remaining locations are set to 0. Here, p is the ratio of the nonzero elements in $\boldsymbol{S}^\star$. According to (3), the data matrix is generated by $\boldsymbol{X} = \boldsymbol{L}^\star + \boldsymbol{S}^\star$.

Set $d = n = 50$, and $r = 5$. Initialize your algorithms with randomly generated standard Gaussian matrices $(\boldsymbol{U}_0, \boldsymbol{V}_0)$, i.e., random initialization. Conduct the following experiments:

- Set the parameter $p = 0.3$. Run your algorithms on the data $\boldsymbol{X}$ and plot the error $\|\boldsymbol{U}_k\boldsymbol{V}_k^\top - \boldsymbol{L}^\star\|_F$ for each algorithm.
- Varying $p \in \{0.1, 0.2, 0.3, \ldots, 0.8\}$. Run your algorithms on the data $\boldsymbol{X}$ with varying p and plot the final error $\left\|\widehat{\boldsymbol{U}}\widehat{\boldsymbol{V}}^\top - \boldsymbol{L}^\star\right\|_F$ for each algorithm (the x-axis is the value of p), where $\widehat{\boldsymbol{U}}$ and $\widehat{\boldsymbol{V}}$ are the final iterate returned by your algorithms.

You can also design more experiments to illustrate the performance of your algorithms.

Remove shadow and illumination form face images Please apply your algorithms to the **Yale-B02** face image dataset (included in the project package) to remove the shadow and illumination. This dataset contains 64 face images taken from 1 person and each of the images has

[1] In case you do not have access to wikipedia, I have downloaded the PDF for IRLS, which is available in the `Reference` folder of the project package.

size 192×168. The main idea is that the clean face images of the same person are similar to each other such that they can form a low-rank matrix (regard each face image as one column), while the illumination and shadow is sparse. Hence, it fits the model (3). The expected result is displayed in Figure 1.

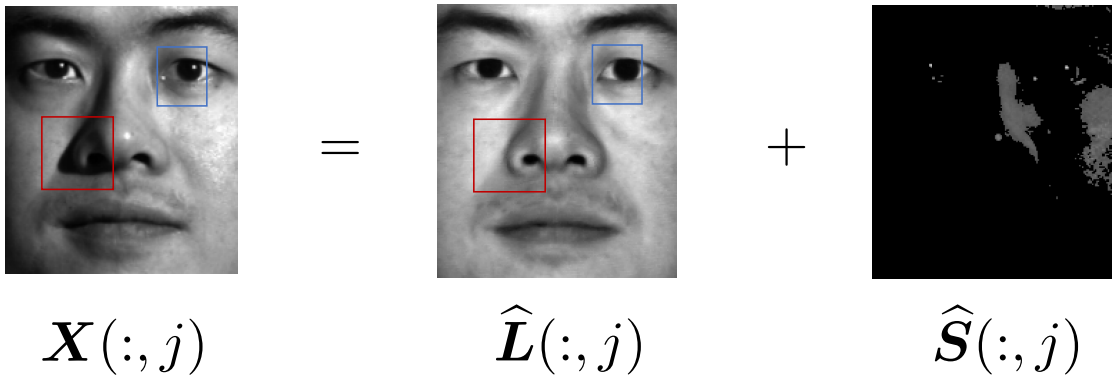

Figure 1: Expected result, where $\widehat{\boldsymbol{L}} = \widehat{\boldsymbol{U}}\widehat{\boldsymbol{V}}^\top$ and $\widehat{\boldsymbol{S}} = \boldsymbol{X} - \widehat{\boldsymbol{L}}$

Extracting the static background from video frames Please apply your algorithms to the **escalator** video dataset (included in the project package and each column is one video frame) to extract the static background. This dataset contains 200 video frames and each of the frames has size 130×160. The main idea is that the static video backgrounds are similar to each other so that they can form a low-rank matrix (regard each video frame as one column), while the moving foreground is sparse. Hence, it fits the model (3). The expected result is stored as `escalator_output.mp4` in the `Expected_results` folder in our project package. The first video is the original one (corresponding to the dataset), while the middle one is the extracted static background and the third video is the extracted moving foreground.

Bonus: Nonconvex optimization theory for solving (5). This question is a bonus. Your grade of this final project will not be affected if you do not answer this question. Nonetheless, if you can provide me a workable theory, you will automatically get a **full grade** of this course, regardless of your performance in homework and midterm.

Assume our data $\boldsymbol{X}$ satisfies the model (3)-(4). I would like to see if we can guarantee to solve problem (5) to the underlying low-rank matrix $\boldsymbol{\theta}^\star := (\boldsymbol{U}^\star, \boldsymbol{V}^\star)$ under this model. That is, our algorithm (e.g., subgradient method) is guaranteed to find $\boldsymbol{\theta}^\star$. Since problem (5) is nonconvex, we can try a two-step guarantee. We can first find an initialization strategy (I guess *spectral method* can do the job) that initializes the algorithm (e.g., subgradient method) near $\boldsymbol{\theta}^\star$. Mathematically, we wish

$$\max_j \ |\boldsymbol{\theta}_0[j] - \boldsymbol{\theta}^\star[j]| \le \gamma, \tag{6}$$

for some $\gamma > 0$. Then, we can try to show some local geometry of problem (5). What I guess can be doable is to show: for any $\boldsymbol{\theta}$ satisfying $\max_j \ |\boldsymbol{\theta}[j] - \boldsymbol{\theta}^\star[j]| \le \gamma$, we have

$$\langle \boldsymbol{D}, \boldsymbol{\theta} - \boldsymbol{\theta}^\star \rangle \ge \alpha \, \mathrm{dist}(\boldsymbol{\theta}, \boldsymbol{\theta}^\star), \quad \forall \boldsymbol{D} \in \partial \mathcal{L}(\boldsymbol{\theta}), \tag{7}$$

for some $\alpha > 0$, where $\mathcal{L}(\boldsymbol{\theta}) = \|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{V}^\top\|_1$. It is important to keep in mind that you have to show (7) within the local region $\{\boldsymbol{\theta} : \max_j \ |\boldsymbol{\theta}[j] - \boldsymbol{\theta}^\star[j]| \le \gamma\}$ and this γ should coincide with that in (6).

Combine the initialization (6) and the local geometric condition (7), we will be able to show that the subgradient method will converge at a linear rate to $\boldsymbol{\theta}^\star := (\boldsymbol{U}^\star, \boldsymbol{V}^\star)$. Thus, the remaining tasks would be:

1. find a spectral initialization method such that we can obtain $\boldsymbol{\theta}_0$ satisfying (6);
2. prove (7).

This bonus credit is for individual who can answer this question, not for his/her group. However, if two students collaborate to answer this question, both will get this bonus credit. If you have any idea, feel free to contact me and we can discuss about it.

Project Report and Presentation. A report should be written to summarize the project and to collect and present your different results. The main content of the report should take no more than **4 double-column pages**. The references and a possible additional appendix are not counted in this page limit. It should cover the following topics and items:

- What is this project about?
- What learning algorithms have you implemented? What are your main contributions?
- Display your main results and describe your observations.
- Other contents you think is meaningful to include.

Please clearly indicate the responsibilities and contributions of each group member in the footnote of the first page and mention if you have received help from other groups, the teaching assistant, or the instructor.

The deadline for the report is **20 May, 2021**. Please submit your report along with the companying codes and any other supporting materials through blackboard.

The presentation time is **22 May, 2021**.