Robust Principal Component Analysis on Graphs

Sofiane Ezzehi
École Normale Supérieure Paris-Saclay
École des Ponts ParisTech
sofiane.ezzehi@eleves.enpc.fr

Alexandre Lutt
École Normale Supérieure Paris-Saclay
École des Ponts ParisTech
alexandre.lutt@eleves.enpc.fr

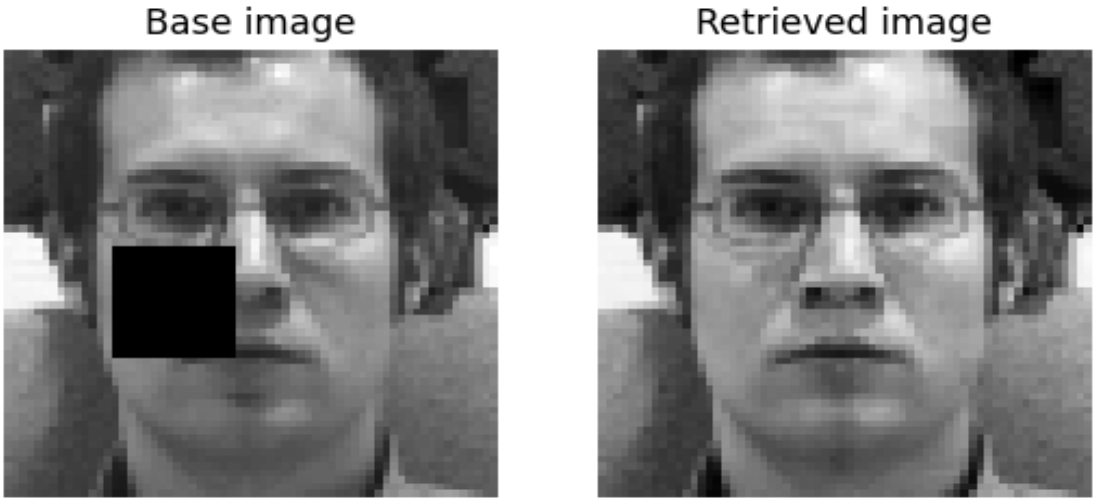

Figure 1: Low-rank reconstruction of corrupted images using the proposed method of [1].

1 ABSTRACT

Principal Component Analysis (PCA) is a very popular method for dimensionality reduction, and is used by thousands accross the world to provide 2D or 3D visualisations and insights about high-dimension data. It is used in a variety of different fields, such as image processing, finance, biology, and computer vision, to name only a few of them. In this work, we present a benchmark of different variants of PCA, which aim to tackle some of the issues of the original PCA algorithm that prevent it from being used in many real-life situations.

2 INTRODUCTION

As mentionned in the abstract, PCA is a useful tool for dimensionnality reduction. More specifically, given a matrix $X \in \mathbb{R}^{n \times p}$, it solves the following optimization problem:

$$\begin{aligned} \min_{Q,U} \quad & ||X - UQ^T||_F^2 \\ \text{subject to} \quad & Q^T Q = I \end{aligned}$$

where $||.||_F$ designates the Frobenius norm. It can be shown that this problem actually has a closed-form solution given by the principal components of X.

However, the main drawback of PCA is that it is very sensitive to noise, corrupted data and outliers and thus cannot be used in many real-world applications. The sensitivity to noise has been partially solved by the introduction of a robust variant of PCA, RPCA [3]. However, this algorithm is very slow on large datasets, which makes it impractical for many applications. This motivated the introduction of another PCA variant called GLPCA [2] which uses graph Laplacian regularization to improve the robustness of the algorithm while keeping the computational cost low. Finally, a third variant of PCA similar to the previous ones has been proposed by the authors of [1] to improve the robustness of the algorithm while keeping the computational cost low.

This project aims to provide a simple and efficient implementation of those main variants of the PCA algorithm, as well as a benchmark of those methods on different tasks (clustering and low-rank recovery for corrupted data on real-life and artificial datsets).

3 ALGORITHMS

In this section, we are going to present the algorithms we implemented and compared for the benchmark. We can divide them into two categories: factorized and non-factorized models. The first category regroups the original PCA algorithm and its variants which aim to learn two matrix factors U and Q such that $X \approx UQ$. Those models often have the constraint $Q^T Q = I$, which, in the case of the original PCA, can be understood as the orthonormality condition of the eigenvectors basis of $X^T X$. On the other hand, the second category regroups PCA variants which try to learn a low-rank matrix L such that $X = L + S$, where S is sparse. For example, the original PCA algorithm, as well as GLPCA and RGLPCA are factorized models, while RPCA and the new variant introduced in [1] are non-factorized models.

3.1 Classical PCA

We used Scikit-Learn implementation of PCA, which simply computes the SVD of X and creates a low-rank approximation by using the first k singular vectors of X.

3.2 Robust PCA

We implemented (see algorithm 1) the algorithm described in [3] using the ADMM [4] (Alternating Direction Method of Multipliers) and following as closely as possible the pseudo-code provided in the paper. This algorithm solves the following convex optimization problem:

$$\begin{aligned} \min_{L,S} \quad & ||L||_* + \lambda||S||_1 \\ \text{subject to} \quad & L + S = X \end{aligned}$$

where $||.||_*$ is the nuclear norm and $||.||_1$ is the ℓ_1 norm. This problem is solved iteratively by alterning between phases of optimization over L and over S.

3.3 GLPCA

We also implemented (see algorithm 2) the algorithm described in [2] using the closed-form solution proved in the original paper.

This algorithm solves the following optimization problem:

$$\begin{aligned} \min_{Q,U} \quad & ||X - UQ^T||_F^2 + \alpha Tr(Q^T(D-W)Q) \\ \text{subject to} \quad & Q^TQ = I \end{aligned}$$

Where W is the adjacency matrix of the graph $\mathcal{G}$, D is the degree matrix of $\mathcal{G}$, and α is a hyperparameter of the algorithm. In pratice, we followed the pseudo-code provided in paper [2] and used the closed-form solution obtained when reformulating the problem using $\alpha = \dfrac{\beta}{1-\beta}\dfrac{\lambda_n}{\xi_n}$ (where λ_n and ξ_n are respectively the maximum eigenvalues of X^TX and $D - W$) and differentiating the objective function with respect to Q and U.

3.4 RGLPCA

We also implemented (see algorithm 3) the robust version of the previous algorithm also mentionned in [2]. This algorithm solves the following optimization problem:

$$\begin{aligned} \min_{Q,U} \quad & ||X - UQ^T||_{2,1} + \alpha Tr(Q^T(D-W)Q) \\ \text{subject to} \quad & Q^TQ = I \end{aligned}$$

where $||.||_{2,1}$ is the $\ell_{2,1}$ norm defined by $||A||_{2,1} = \sum_{j=1}^{n}\sqrt{\sum_{i=1}^{p} A_{ij}^2}$.

This problem is solved using the ADMM [4] algorithm, following the pseudo-code provided in the paper.

3.5 RPCA on graphs

Finally, we implemented (see algorithm 4) the RPCA on graphs algorithm described in [1] which claims to be more robust than the previous ones while keeping the computational cost low. This algorithm solves the following convex optimization problem:

$$\begin{aligned} \min_{L,S} \quad & ||L||_* + \lambda||S||_1 + \gamma Tr(L\phi L^T) \\ \text{subject to} \quad & L + S = X \end{aligned}$$

Where ϕ is the graph Laplacian of $\mathcal{G}$ and γ is a hyperparameter of the algorithm. As the previous methods, it uses the ADMM [4] algorithm to solve this problem by alterning phases of optimization over L and over S.

4 PSEUDO-CODE

In order to make notations more concise, we will define the following shrinkage and singular value tresholding operators (defined on the space of square matrices):

$$\begin{cases} \mathcal{S}_\tau(\mathbf{Z}) = \text{sign}(\mathbf{Z}) \times \max(\mathbf{0}, |\mathbf{Z}| - \tau\mathbf{I}) \\ \mathcal{D}_\tau(\mathbf{Z}) = \mathbf{U}\mathcal{S}_\tau(\Sigma)\mathbf{V}^T \text{ with } \mathbf{Z} = \mathbf{U}\Sigma\mathbf{V}^T \end{cases}$$

Algorithm 1 RPCA algorithm

Require: $\mathbf{X} \in \mathbb{R}^{p\times n}$, ε, n_{iter}
  $\lambda \leftarrow \frac{1}{\sqrt{\max(m,n)}}$
  $\mu \leftarrow \frac{\max(m,n)}{4\times||\mathbf{X}||_1}$
  $\mathbf{S} \leftarrow \mathbf{0}$
  $\mathbf{Y} \leftarrow \mathbf{0}$
  for $i = 1$ **to** n_{iter} **do**
    $\mathbf{L} \leftarrow \mathcal{D}_\mu(\mathbf{X} - \mathbf{S} + \frac{1}{\mu}\mathbf{Y})$
    $\mathbf{S} \leftarrow \mathcal{S}_{\frac{\lambda}{\mu}}(\mathbf{X} - \mathbf{L} + \frac{1}{\mu}\mathbf{Y})$
    $\mathbf{Y} \leftarrow \mathbf{Y} + \mu(\mathbf{X} - \mathbf{L} - \mathbf{S})$
    if $||\mathbf{X} - \mathbf{L} - \mathbf{S}||_F \leqslant \varepsilon$ **then**
      break
    end if
  end for
  return L, S

Algorithm 2 GLPCA algorithm

Require: $\mathbf{X} \in \mathbb{R}^{p\times n}$, $\mathcal{G}$, β
  $\mathbf{W} \leftarrow adj(\mathcal{G})$
  $\mathbf{D} \leftarrow diag(\{d_1, d_2, ..., d_{n_{\text{nodes}}}\})$
  $\mathbf{L} \leftarrow \mathbf{D} - \mathbf{W}$
  $\lambda_n \leftarrow \Re(\max(eigenval(\mathbf{X}^T\mathbf{X})))$
  $\xi_n \leftarrow \Re(\max(eigenval(\mathbf{L})))$
  $\mathbf{G}_\beta \leftarrow (1-\beta)(\mathbf{I} - \frac{1}{\lambda_n}\mathbf{X}^T\mathbf{X}) + \frac{\beta}{\xi_n}\mathbf{L}$
  $\mathbf{Q} \leftarrow \Re(eigenvect(\mathbf{G}_\beta))$
  $\mathbf{U} \leftarrow \mathbf{X}\mathbf{Q}$
  return Q, U

Algorithm 3 RGLPCA algorithm

Require: $\mathbf{X} \in \mathbb{R}^{p \times n}$, $\mathcal{G}$, β, k, ρ, n_{iter}
  $\mathbf{X}_0 \leftarrow \mathbf{X}$
  $\beta_0 \leftarrow \beta$
  $\mathbf{E} \leftarrow \mathbf{1}$
  $\mathbf{C} \leftarrow \mathbf{1}$
  $\mu \leftarrow 1$
  $\mathbf{W} \leftarrow adj(\mathcal{G})$
  $\mathbf{D} \leftarrow diag(\{d_1, d_2, ..., d_{n_{\text{nodes}}}\})$
  $\mathbf{L} \leftarrow \mathbf{D} - \mathbf{W}$
  $\xi \leftarrow \Re(max(eigenval(\mathbf{X}_0^T \mathbf{X}_0)))$
  for $i = 1$ **to** n_{iter} **do**
    $\mathbf{X}_i \leftarrow \mathbf{X} - \mathbf{E} - \frac{1}{\mu}\mathbf{C}$
    $\lambda \leftarrow \Re(\max(eigenval(\mathbf{X}_i^T \mathbf{X}_i)))$
    $\alpha \leftarrow \frac{2\beta_0 \lambda}{\mu(1-\beta_0)\xi}$
    $\beta \leftarrow \frac{\alpha\xi}{\lambda + \alpha\xi}$
    $\mathbf{Q}, \mathbf{U} \leftarrow GLPCA(\mathbf{X}, \mathcal{G}, \beta, k)$
    $\mathbf{A} \leftarrow \mathbf{X}_k - \mathbf{U}\mathbf{Q}^T - \frac{1}{\mu}\mathbf{C}$
    $\mathbf{a} \leftarrow (||\mathbf{A}_1||_2, ||\mathbf{A}_2||_2, ..., ||\mathbf{A}_m||_2)$
    $\mathbf{E} \leftarrow \mathcal{S}_{\frac{1}{\mu}}(\mathbf{a})$
    $\mathbf{C} \leftarrow \mathbf{C} + \mu(\mathbf{E} - \mathbf{X} - \mathbf{U}\mathbf{Q}^T)$
    $\mu \leftarrow \rho\mu$
  end for
  return $\mathbf{Q}$, $\mathbf{U}$, $\mathbf{E}$

Algorithm 4 RPCA on graphs algorithm

Require: $\mathbf{X} \in \mathbb{R}^{p \times n}$, $\mathcal{G}$, n_{iter}, γ, ε
  $\mathbf{A} \leftarrow adj(\mathcal{G})$
  $\mathbf{D} \leftarrow diag(\{d_1, d_2, ..., d_{n_{\text{nodes}}}\})$
  $\boldsymbol{\phi} \leftarrow D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
  $\lambda \leftarrow \frac{1}{\sqrt{\max(p,n)}}$
  $\mathbf{L} \leftarrow random(p, n)$
  $\mathbf{W} \leftarrow random(p, n)$
  $\mathbf{S} \leftarrow random(p, n)$
  $r_1 \leftarrow 1$
  $r_2 \leftarrow 1$
  $\mathbf{Z}_1 \leftarrow \mathbf{X} - \mathbf{L} - \mathbf{S}$
  $\mathbf{Z}_2 \leftarrow \mathbf{W} - \mathbf{L}$
  for $i = 1$ **to** n_{iter} **do**
    $\mathbf{H}_1 \leftarrow \mathbf{X} - \mathbf{S} + \frac{1}{r_1}\mathbf{Z}_1$
    $\mathbf{H}_2 \leftarrow \mathbf{W} + \frac{1}{r_2}\mathbf{Z}_2$
    $\mathbf{H} \leftarrow \frac{1}{r_1 + r_2}(r_1\mathbf{H}_1 + r_2\mathbf{H}_2)$
    $\mathbf{L} \leftarrow \mathcal{D}_{\frac{2}{r_1+r_2}}(\mathbf{H})$
    $\mathbf{S} \leftarrow \mathcal{S}_{\frac{\lambda}{r_1}}(\mathbf{X} - \mathbf{L} + \frac{1}{r_1}\mathbf{Z}_1)$
    $\mathbf{W} \leftarrow r_2(\gamma\boldsymbol{\phi} + r_2\mathbf{I})^{-1}(L - \frac{1}{r_2}Z_2)$
    $\mathbf{Z}_1 \leftarrow \mathbf{Z}_1 + r_1(\mathbf{X} - \mathbf{L} - \mathbf{S})$
    $\mathbf{Z}_2 \leftarrow \mathbf{Z}_2 + r_2(\mathbf{W} - \mathbf{L})$
    if $||\mathbf{X} - \mathbf{L} - \mathbf{S}||_F \leqslant \varepsilon$ **then**
      break
    end if
  end for
  return $\mathbf{L}$, $\mathbf{S}$

5 EXPERIMENTAL SETUP

For the benchmark, we compared these algorithms on two different tasks: low-rank recovery for corrupted data and clustering. For the low-rank recovery task, we used the CMU PIE dataset which is a dataset of images of faces with different angles and lighting. For the clustering task, we used the AT&T dataset which is another dataset of images of faces.

5.1 Low-rank recovery for corrupted data

For this task, we used the PIE dataset, and more specifically 20 64×64 images of the same 40 people under different lighting with a constant camera angle, and corrupted them by adding a single random black block of 25% of the image dimensions to each sample. We then used the different algorithms to recover the concatenation of the original images from the concatenation of the corrupted ones. The main hypothesis here is that this long concatenation matrix will be well approximated by a low-rank matrix, since the difference in lighting and corruption is supposed to be sparse. We evaluated the different methods mostly by visual inspection of the results, but also by computing the mean squared error between the original and recovered images.

5.2 Clustering

For this task, we used the AT&T dataset, and more specifically 400 112×92 images aggregated in 40 equally sampled classes (each class

corresponding to one individual). We then compute the low-rank approximation of the concatenation of the images, and used the resulting matrix as a feature matrix for clustering. We then evaluated the different methods by computing the labels produced by the KMeans algorithm trained on this feature matrix, and comparing them with the original labels of the dataset. We compared the different methods by computing the error between, as well as the Adjusted Rand Index (ARI) as well as the Normalized Mutual Information (NMI). These metrics can be easily computed given the confusion matrix and the raw labels:

$$ARI = 2\frac{TP \times TN - FN \times FP}{(TP + FN) \times (FN + TN) + (TP + FP) \times (FP + FN)}$$

$$NMI = \frac{I(y_{\text{true}}, y_{\text{pred}})}{\sqrt{H(y_{\text{true}}) \times H(y_{\text{pred}})}}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, FN is the number of false negatives, $I(y_{\text{true}}, y_{\text{pred}})$ is the mutual information between y_{true} and y_{pred}, and $H(y)$ is the entropy of y.

6 RESULTS

7 LIMITATIONS

8 CONCLUSION

REFERENCES

[1] Patricia S. Abril and Robert Plant. 2015. Robust Principal Component Analysis on Graphs. (April 2015). http://arxiv.org/abs/1504.06151

[2] Bin Luo Bo Jiang, Chris Ding and Jin Tang. 2013. Graph-Laplacian PCA: Closed-form Solution and Robustness. (January 2013).

[3] Yi Ma Emmanuel J. Candès, Xiaodong Li and John Wright. 2009. Robust Principal Component Analysis? (December 2009). http://arxiv.org/abs/0912.3599

[4] Eric Chu Stephen Boyd, Neal Parikhl, Borja Peleato, and Jonathan Eckstein. 2010. Distributed optimization and statistical learning via the alternating direction method of multipliers. (January 2010).