Albert-Ludwigs-Universität
Freiburg

A Neural Network for RPCA

A project in

Stochastic Machine Learning

Eric Brunner, Aron Distelzweig, Mirko Grimm, Bastian Schnitzer, Simon Beyer

December 18, 2020

Contents

1 Introduction

Principal Component Analysis (PCA) is a method for finding the principal components of a set of data points. The principal components of a collection of points in the space $\mathbb{R}^n$ are a sequence of n direction vectors, where the i^{th} vector is the direction of a line that best fits the data while being orthogonal to the first $i-1$ vectors. A best fitting line is defined as one that minimizes the average squared distance from the points to the line. The principal components form an orthonormal basis of the space $\mathbb{R}^n$. One can think of PCA as fitting a p-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component. One of the main applications for PCA is in quantitative finance, where data of large quantities is being reduced to a minimum amount of principal components that still hold the most important information about the data. In specific we look at a modification of PCA which is the Robust Principal Component Analysis. Here our aim is to recover a low rank Matrix L from highly corrupted measurements M.

We start with a data matrix $M \in \mathbb{R}^{n\times n}$ with corrupted entries and try to find a composition into a sum $M = L + S$, where L is a matrix of low rank and S is sparse, which means that a lot of entries are zero. M can be for example a covariance matrix which is by nature symmetric and positive definite. In this project our aim is to implement the algorithm "Denise", an algorithm to compute a robust PCA for semidefinite matrices via a deep neuronal network. Due to slow computation the usual algorithms for PCA are inefficient in some applications e.g. finance, where instantaneous calculation might be required. Therefore the use of a neuronal network is seen to perform better and can once trained be used to reduce computation times for a given problem.

2 Denise

2.1 Generation of Training Data

2.2 Algorithm

We consider $\mathbb{S}_n \subseteq \mathbb{R}^{n\times n}$ to be the set of n-by-n symmetric matrices and $P_n \subseteq \mathbb{S}_n$ to be the subset of positive semi-definite matrices and $P_{k,n} \subseteq P_n$ the subset of matrices with rank at most k. As the input of our deep neuronal network we consider a Matrix $M = [M_{i,j}]_{i,j} \in P_n$. The matrix M is to be decomposed as a sum $M = L + S$ where $L = [L_{i,j}]_{i,j} \in P_{k,n}$ is of rank at most k and a sparse matrix $S = [S_{i,j}]_{i,j} \in P_n$. Considering the loss-function of the neural network we look at the l_1-matrix-norm of S since S should be sparse and therefore $\|S\|_{l_1}$ reduces along the sparsity of S. Since L is assumed to be symmetric by the Cholesky decomposition we can represent it as $L = UU^T$, where $U = [U_{i,j}]_{i,j} \in \mathbb{R}^{n\times k}$. Therefore M can be expressed as $M = UU^T + S$. We now calculate U as the output of a multi-layer feed-forward neural network where the loss-function to minimize is $\|UU^T - M\|_{l_1} = \|S\|_{l_1}$. As the matrix M is symmetric, we can reduce the input from n^2 to $n(n+1)/2$ by taking the triangular lower matrix of M. The lower matrix is then transformed into a vector using the operator h

$$h : S^n \to \mathbb{R}^{n(n+1)/2},\ M \mapsto (M_{1,1}, M_{2,1}, M_{2,2}, \ldots, M_{n,1}, \ldots, M_{n,n})^T \tag{2.1}$$

Similarly we convert the output vector of the neural network into a matrix with the operator g defined as

$$g : \mathbb{R}^{nk} \to \mathbb{R}^{n\times k},\ X \mapsto \begin{pmatrix} X_1 & \cdots & X_k \\ \vdots & & \vdots \\ X_{(n-1)k+1} & \cdots & X_{(n-1)k+k} \end{pmatrix} \tag{2.2}$$

Our multi-layer feed-forward neural network $\mathcal{N} : \mathbb{R}^{n(n+1)/2} \to \mathbb{R}^{nk}$ has 4 layers each with the same activation function $\sigma : \mathbb{R} \to \mathbb{R}$ that we use componentwise.

Using h and g the matrix U is the output of the neural network $U = g(\mathcal{N}(h(M)))$ and we get the desired matrix $L = \rho(\mathcal{N}(h(M)))$ for

$$\rho : \mathbb{R}^{rd} \to P_{r,d}, X \mapsto g(X)g(X)^T \tag{2.3}$$

3 Testing and Comparism

After the neural network has been trained with the synthetic data, as discribed in section 2.1, it can be tested on synthetic data, where the decomposition is known by construction, aswell as on real world data. To compare the decompositions of real world data, another algorithm is needed, which calculates the decomposition of an arbitrary positive semidefinite matrix M into a low rank, symmetric and positive-semidefinite matrix L_0 plus a sarse matrix S_0:

$$M = L_0 + S_0. \tag{3.1}$$

With *Principal Component Pursuit* (PCP), there exists an algorithm, which, under some suitable assumptions, calculates the decomposition exactly via singular value decomposition (SVD) [1]. The assumptions and the main ideas of this algorithm is presented in the first subsection. In the second subsection the results of the decomposition of portfolio correlation matrices via the AI algorithm Denise with the results of the Principal Component Pursuit algorithm.

3.1 The Minimization Problem solved by PCP

Let M be a given element of $\mathbb{R}^{n_1 \times n_2}$. $\|\cdot\|_*$ denotes the nuclear norm, i.e. the sum over the singular values of a matrix $\|M\|_* := \sum_i \sigma_i(M)$. $\|\cdot\|_1$ is the well known ℓ_1 norm $\|M\|_1 = \sum_{ij}|M_{ij}|$. The PCP algorithm solves the convex optimazation problem

$$\text{minimize} \quad \|L\|_* + \|S\|_1, \quad \text{where } L + S = M \tag{3.2}$$

exactly, if the the low-rank component L_0 fulfills a "incoherence" condition, and that the sparse component is "reasonably sparse". The meaning of this "incoherence" condition for L_0 and the "reasonable" sparsity of S_0 is explained in [1, section 1.3]. We summatize the main points real for quadratic matrices:

(i) Let $U\Sigma V^\top$ the singular singular value decomposition of $L_0 \in R^{n\times n}$ with rank $k \geq n$, i.e.

$$L_0 = U\Sigma V^\top = \sum_{i=1}^{k} \sigma_i u_i v_i^\top, \tag{3.3}$$

where $U = (u_1, \ldots, u_k), V = (v_1, \ldots, v_k) \in \mathrm{O}(n)$, $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0) \in \mathbb{R}^{n\times n}$. $\sigma_1, \ldots, \sigma_k$ are the singular values and u_i and v_i, $i = 1, \ldots, k$, are the left-singular and right-singular vectors for σ_i, respectively. Then the matrix L_0 is called incoherent, with parameter μ, if

$$\max_i \|Ue_i\|^2 \geq \frac{\mu k}{n^2}, \quad \max_i \|Ve_i\|^2 \geq \frac{\mu k}{n^2}, \quad \left\|UV^\top\right\|_\infty \geq \frac{\sqrt{\mu k}}{n}. \tag{3.4}$$

e_i are the canonical basis vectors of $\mathbb{R}^n$.

(ii) The positions of the nonzero elements of the sparsity matrix are selected uniformly random.

If (i) is fullfilled, the matrix L_0 is considered as not sparse. With (ii) we try to prevent, that the nonzero elements are only in one, or few columns of the sparsity matrix. For example if the entries of S_0 except the first column are all zero, and the first column of S_0 is the negative of the first column of L_0, then it is impossible to recover the low rank component and sparse component exactly. To avoid, such variety of possibilities for the decomposition (ii) is a reasonable assumption.

3.2 PCP

3.3 Decomposition of Portfolio Correlations from DAX 30 with PCP and Denise

For test purposes we applied the PCP algorithm to the empirical covariance matrix based on the prices of five share certificates of the companies Allianz, BASF Bayer, Beiersdorf and BMW of the last six months. The following covariance matrix is obtained

$$(\mathrm{Cov}(x_i, x_j))_{i,j} = \begin{pmatrix} 142.67041515 & 28.06338926 & 30.56147946 & 2.52487121 & 31.18558268 \\ 28.06338926 & 14.54671933 & -10.75409144 & -2.97700557 & 20.0735403 \\ 30.56147946 & -10.75409144 & 65.84451884 & 11.30075662 & -28.04105723 \\ 2.52487121 & -2.97700557 & 11.30075662 & 10.38498412 & -5.84017695 \\ 31.18558268 & 20.0735403 & -28.04105723 & -5.84017695 & 33.39091805 \end{pmatrix}, \tag{3.5}$$

where $x = (x_1, \ldots, x_5) = (\text{Allianz, BASF Bayer, Beiersdorf, BMW})$. The PCP algorithm returns the decompostion

$$L_0 = \begin{pmatrix} 33.15288862 & 19.03429426 & -2.38353047 & 2.5249416 & 24.04138395 \\ 19.03429426 & 14.54671744 & -10.7540876 & -2.97707117 & 20.07354269 \\ -2.38353047 & -10.7540876 & 24.51612531 & 11.30069593 & -17.99311351 \\ 2.5249416 & -2.97707117 & 11.30069593 & 5.60790277 & -5.84023382 \\ 24.04138395 & 20.07354269 & -17.99311351 & -5.84023382 & 28.30038404 \end{pmatrix}, \tag{3.6}$$

$$S_0 = \begin{pmatrix} 109.51752654 & 9.029095 & 32.94500994 & 0. & 7.14419873 \\ 9.029095 & 0. & -0. & -0. & -0. \\ 32.94500994 & -0. & 41.32839353 & 0. & -10.04794373 \\ 0. & -0. & 0. & 4.77708135 & 0. \\ 7.14419873 & -0. & -10.04794373 & 0. & 5.09053401 \end{pmatrix}. \tag{3.7}$$

The rank of L_0 is 2. This is obviously a exact decomposition $(\mathrm{Cov}(x_i, x_j))_{i,j} = L_0 + S_0$.

4 Outlook

4.1 Convolutional Layer

4.2 Matrix Completion

4.3 Data from Neuroscience

4.4 Examples for Data with "Missing Entries"

References

[1] Emmanuel J. Candes et al. *Robust Principal Component Analysis?* 2009. arXiv: 0912.3599 [cs.IT].