Image and Vision Computing 30 (2012) 915–921

Contents lists available at SciVerse ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Low-rank matrix decomposition in L_1-norm by dynamic systems [☆]

Yiguang Liu [a,b,*], Bingbing Liu [d], Yifei Pu [a], Xiaohui Chen [b], Hong Cheng [c]

[a] Vision and Image Processing Lab, College of Computer, Sichuan University, Chengdu 610065, PR China
[b] College of Computer and Information Technology China Three Gorges University Yichang, 443002, PR China
[c] Pattern Recognition and Machine Intelligence Lab., University of Electronic Science and Technology of China, Chengdu 611731, PR China
[d] Institute for Infocomm Research, A*STAR, 1 Fusionopolis Way, Singapore 138632

ARTICLE INFO

Article history:
Received 4 July 2011
Received in revised form 20 June 2012
Accepted 28 June 2012

Keywords:
Low-rank matrix approximation
Dynamic system
L_1 norm
Computational efficiency

ABSTRACT

Low-rank matrix approximation is used in many applications of computer vision, and is frequently implemented by singular value decomposition under L_2-norm sense. To resist outliers and handle matrix with missing entries, a few methods have been proposed for low-rank matrix approximation in L_1 norm. However, the methods suffer from computational efficiency or optimization capability. Thus, in this paper we propose a solution using dynamic system to perform low-rank approximation under L_1-norm sense. From the state vector of the system, two low-rank matrices are distilled, and the product of the two low-rank matrices approximates to the given measurement matrix with missing entries, in L_1 norm. With the evolution of the system, the approximation accuracy improves step by step. The system involves a parameter, whose influences on the computational time and the final optimized two low-rank matrices are theoretically studied and experimentally valuated. The efficiency and approximation accuracy of the proposed algorithm are demonstrated by a large number of numerical tests on synthetic data and by two real datasets. Compared with state-of-the-art algorithms, the newly proposed one is competitive.

1. Introduction

Many applications in the field of computer vision, as well as several others in pattern recognition [1] and bioinformatics, require finding an appropriate low-rank matrix that approximates the measurement matrix [2]. This requirement becomes especially challenging when the measurement matrix has noisy elements or missing entries. For example, while recovering shape and motion from image streams, some tracked feature points may be occluded in some frames and the measurement noise is also unavoidable [3]. For the measurement matrix without missing entries, low-rank matrix approximation in L_2 norm can be solved by singular value decomposition. However, this method is not practical to the measurement matrix with missing entries. To perform low-rank approximations for the matrices with missing entries, a lot of techniques were proposed by many researchers, such as Morita and Kanade [4], Hartley and Schaffalitzky [5], Wang and Wu [6], Buchanan and Fitzgibbon [7,8], Ye [9], Liu et al. [10], Cai et al. [11], and Chen [12]. These techniques include the sequential factorization method, the singular value thresholding algorithm and the power factorization method, etc. It was first introduced by T. Okatani etc. [13] to solve the problem using the Wiberg algorithm, and it was shown that on many problems, the method (L_2-Wiberg for short) outperforms many other methods.

In real applications of low-rank matrix approximations, L_2-norm approximation is justified only when noise is negligible [14], otherwise the obtained low-rank approximating matrix gives no guarantee of statistical optimality, and may be highly biased as well [3,4]. On the contrary, L_1-norm approximation is much more robust to handle outliers than L_2-norm approximation, as reported in [15]. Thus, the low-rank approximation of matrices in L_1 norm has been attracting more and more attention in recent years [16,15,17]. Wright et al. [16] introduced the robust principal component analysis (where L_1 norm is used to recover corrupted low-rank matrices), and the same idea was also used for Robust Video Restoration [18]. Ke and Kanade [15] presented an alternative convex programming method to solve the L_1-norm minimization problem, which is formulated from low-rank matrix factorization. Eriksson and Hengel [17] experimentally revealed that the alternated convex programming approach frequently converges to a point that is not a local minimum (typically, the evolution of the L_1-norm cost function stops after a small number of iterations), and thus they introduce the state-of-the-art L_1-Wiberg approach.

In mathematics, a dynamic system consists of a state vector and an evolution rule, which describes how a future state vector will be obtained from the current one. Dynamic systems have been used in computer vision field for a long time [19], and have also been applied successfully to solve eigen-problems of matrices [20,21]. The L_1-norm cost function is not smooth all the time and non-convex, which causes many traditional optimization techniques suitable for

[☆] This paper has been recommended for acceptance by Yiannis Aloimonos.
[*] Corresponding author at: Vision and Image Processing Lab, College of Computer, Sichuan University, Chengdu 610065, PR China. Tel.: +86 2885419688.
E-mail address: liuyg@scu.edu.cn (Y. Liu).

0262-8856/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2012.06.012

L_2-norm cost function not applicable. In this paper, in order to perform low-rank matrix approximation in L_1 norm, we intend to design a dynamic system to tackle the following problem:

$$\min_{\mathbf{R},\mathbf{S}} \|\mathbf{W}\odot(\mathbf{Y}-\mathbf{RS})\|_1. \tag{1}$$

Compared to the known techniques specially designed for low-rank matrix approximation in L_1 norm [15,17], we will demonstrate that the proposed dynamic system provides competitive capability on optimizing Eq. (1) with a lot of experimental results.

The rest of this paper is organized as follows. In Section 2, the dynamic system is built, and the analysis of its properties is given, which reveals that the dynamic system is able to get the optimal low-rank matrix approximation. Subsequently, in Section 3, we demonstrate the performance of the computational model using experimental results. Finally, concluding remarks are drawn in Section 4.

2. The computation model

Let H_{ij} denote an $(m+n)r\times(m+n)r$ matrix with its sub-matrix from rows $ir+1$ to $(i+1)r$ and from columns $mr+jr+1$ to $mr+(j+1)r$ being an identity matrix. The function vec $(\mathbf{A})$ denotes the vectorization of an $m\times n$ matrix $\mathbf{A}$, i.e., vec $(\mathbf{A})$ is the $mn\times 1$ column vector obtained by stacking the columns of the matrix $\mathbf{A}$ on top of one another. For the convenience of presentation, we equivalently reformulate Eq. (1) as follows:

$$\begin{aligned} &\min_x \mathrm{V}(\mathbf{x}), V(\mathbf{x})\triangleq\sum_{i=1}^{m}\sum_{j=1}^{n}\left|f_{\mathrm{ij}}(\mathbf{x})\right|,\\ &f_{\mathrm{ij}}(\mathbf{x})\triangleq w_{\mathrm{ij}}\left(y_{\mathrm{ij}}-\mathbf{x}^T\mathbf{H}_{\mathrm{ij}}\mathbf{x}\right),\\ &\mathbf{x}\triangleq\left[\mathrm{vec}^T\left(\mathbf{R}^T\right)\mathrm{vec}^T(\mathbf{S})\right]^T. \end{aligned} \tag{2}$$

In practice, the term $\mathbf{x}^T\mathbf{H}_{\mathrm{ij}}\mathbf{x}$ in Eq. (2) corresponds to the (i,j) entry of the product matrix $\mathbf{RS}$ in Eq. (1). In Eq. (2), it can be seen that the cost function V(x) is not convex and not always smooth with respect to $f_{\mathrm{ij}}(\mathbf{x})$. Thus, the techniques based on gradient [22] are not directly applicable. It is also impossible to transform Eq. (2) into a standard linear or quadratic mathematical programming problem, and thus the methods applicable to mathematical programming problems are still not suitable for solving Eq. (2).

Motivated by the difficulty of using conventional methods to solve Eq. (2), we intend to solve the problem by designing a dynamic system, which is proposed as:

$$\begin{cases} \dot{\mathbf{x}}(t)=\sum_{i=1}^{m}\sum_{j=1}^{n}\frac{2}{\pi}\arctan\left(\sigma f_{\mathrm{ij}}(\mathbf{x}(t))\right)\mathbf{g}_{\mathrm{ij}}(\mathbf{x}(t)),\\ \qquad \mathbf{g}_{\mathrm{ij}}(\mathbf{x}(t))\triangleq w_{\mathrm{ij}}\left(\left(\mathbf{H}_{\mathrm{ij}}+\mathbf{H}_{\mathrm{ij}}^T\right)\mathbf{x}(t)\right), t>0;\\ \mathbf{x}(t)=\mathbf{x}_0, t=0 \end{cases} \tag{3}$$

where the parameter $\sigma>0$. With the evolution of Eq. (3), the state vector $\mathbf{x}(t)$ varies with time, and so does $\mathrm{V}(\mathbf{x}(t))$. Is the evolution of Eq. (3) able to decrease $\mathrm{V}(\mathbf{x}(t))$ towards its global or local minimum? And how does the parameter σ affects the convergence of Eq. (3)? In order to answer these questions, two propositions are given. The first one is formulated as follows, and it was proven in Appendix A.

Proposition 1. *When σ approaches infinity, if the initial vector,* $\mathbf{x}(t_0)$, *satisfies* $D^+\mathrm{V}(\mathbf{x}(t_0))\neq 0$, *the objective function* $\mathrm{V}(\mathbf{x}(t))$ *will be locally minimized by the evolution of the dynamic system in Eq.* (3); *if* $\mathbf{x}(t_0)$ *exactly satisfies* $D^+\mathrm{V}(\mathbf{x}(t_0))=0$, $\mathrm{V}(\mathbf{x}(t))$ *will keep the same value as* $\mathrm{V}(\mathbf{x}(t_0))$, *whether* $\mathrm{V}(\mathbf{x}(t_0))$ *is a local maximal or minimal extremum.*

The condition that σ approaches infinity makes sure the convergence of $\mathrm{V}(\mathbf{x}(t))$ to its local minimum in terms of Proposition 1. However, it may lead to low computational efficiency for Eq. (3). It is impossible to derive the analytic solution of Eq. (3), and thus it is a common way to use numerical computation to solve Eq. (3). In numerical computation, the time step size, δt, can either be fixed or varied. Compared to the fixed-step strategy, the varied-step one is time-saving because for a given level of accuracy, it can reduce the number of steps when adjusting the time step size dynamically at necessity. Thus, varied-step strategy is usually preferred, and the time step size varies dynamically in terms of the local error. In numerically solving Eq. (3) with a variable step size, the time step size δt is usually related to the Jacobin matrix of the right hand side of Eq. (3) at time t. And a smaller value is chosen for δt if greater changes have taken place in the right hand side. Due to $\frac{\partial\arctan\left(\sigma f_{\mathrm{ij}}(\mathbf{x}(t))\right)}{\partial f_{\mathrm{ij}}(\mathbf{x}(t))}=\frac{\sigma}{\left(\sigma f_{\mathrm{ij}}(\mathbf{x}(t))\right)^2+1}$, the condition that σ goes to infinity makes $\frac{\partial\arctan\left(\sigma f_{\mathrm{ij}}(\mathbf{x}(t))\right)}{\partial f_{\mathrm{ij}}(\mathbf{x}(t))}$ positive infinite when $f_{\mathrm{ij}}(\mathbf{x}(t))$ is close to zero, and leads to great changes in the right hand side of Eq. (3) in a large probability. Thus the condition that σ approaches infinity will reduce the time step size δt, and will increase the iteration number seriously. It further allows the computation time of Eq. (3) to increase dramatically when $f_{\mathrm{ij}}(\mathbf{x}(t))$ is close to zero.

In real applications, such as recovering shape and motion from image streams [4] and robust alignment [23], the size of the measurement matrix $\mathbf{Y}$ is usually large, which allows the dimension of $\mathbf{x}$ to be high. For example, in the Oxford dinosaur data set [13], $\mathbf{Y}$ is a 72×319 matrix as used in [17], and it makes the dimension of $\mathbf{x}$ be $(m+n)r=1564$. The low efficiency due to the condition that σ approaches infinity can leave Eq. (3) unsuitable for these large-scale problems. In order to ameliorate the calculation efficiency of Eq. (3), σ cannot be infinite. Can the dynamic system of Eq. (3) with a finite σ still have the capability to optimize $\mathrm{V}(\mathbf{x})$? In order to solve this question, we give the following proposition, the proof of which will be provided in Appendix B.

Proposition 2. *With any* $\sigma>0$, $\mathrm{V}(\mathbf{x}(t))$ *cannot keep on increasing. Moreover, if σ is finite, but large enough, the local minimal value of* $\mathrm{V}(\mathbf{x}(t))$ *can be extracted by the evolution of Eq.* (3) *if* $D^+\mathrm{V}(\mathbf{x}(t_0))\neq 0$; and Eq. (3) cannot change $\mathrm{V}(\mathbf{x}(t))$ if $D^+\mathrm{V}(\mathbf{x}(t_0))=0$ whether $\mathrm{V}(\mathbf{x}(t_0))$ is a locally maximal or minimal extremum of $\mathrm{V}(\mathbf{x})$.

Proposition 2 implies that $\mathrm{V}(\mathbf{x}(t))$ is bounded, and so is $f_{\mathrm{ij}}(\mathbf{x}(t))$. In the bounded domain of $f_{\mathrm{ij}}(\mathbf{x}(t))$, a larger $\sigma\left|f_{\mathrm{ij}}(\mathbf{x}(t))\right|$ makes $\frac{2}{\pi}$ *arctan* $\left(\sigma f_{\mathrm{ij}}(\mathbf{x}(t))\right)$ closer to $\mathrm{sgn}\left(f_{\mathrm{ij}}(\mathbf{x}(t))\right)$. Thus any aspect affecting the values of $\sigma\left|f_{\mathrm{ij}}(\mathbf{x}(t))\right|$ will influence the evolution of $\mathrm{V}(\mathbf{x}(t))$. A larger σ can enhance the decreasing speed of $\mathrm{V}(\mathbf{x}(t))$, as indicated by Propositions 1 and 2. On the other hand, $\left|f_{\mathrm{ij}}(\mathbf{x}(t))\right|$ is frequently larger for the measurement matrix $\mathbf{Y}$ with a larger dimension. Thus, if $\mathbf{Y}$ has a larger dimension, $\mathrm{V}(\mathbf{x}(t))$ usually has a better convergence speed under the same σ. Proposition 2 implies that, a finite σ may be appropriate for optimizing $\mathrm{V}(\mathbf{x}(t))$ for a given measurement matrix $\mathbf{Y}$, and it is unnecessary to assign a very large value to σ, which possibly leads to a heavy computational load. Actually for any $\sigma>0$, the probability, in which $\mathrm{V}(\mathbf{x}(t))$ increases at time t under the evolution of Eq. (3), is very low, and this will be confirmed by the experimental results in the following Section. Combining Propositions 1 and 2, it tells that Eq. (3) with a larger σ is more suitable for optimizing $\mathrm{V}(\mathbf{x})$. However, this may consume more computation time. Thus the practical value of σ is dependent on the tradeoff between the convergence speed (of $\mathrm{V}(\mathbf{x}(t))$ with respect to time (t) and the actual computation time.

3. Experimental results

In this section, experimental results are presented to evaluate the performance of the proposed approach in Eq. (3). There are two purposes of the experiments. The first is to experimentally valuate the influence of the parameter σ on both computation time and convergence