(19) **United States**

(12) **Patent Application Publication** (10) Pub. No.: **US 2016/0275416 A1**

Min et al. (43) **Pub. Date:** **Sep. 22, 2016**

(54) **FAST DISTRIBUTED NONNEGATIVE MATRIX FACTORIZATION AND COMPLETION FOR BIG DATA ANALYTICS**

(71) Applicant: **NEC Laboratories America, Inc.**, Princeton, NJ (US)

(72) Inventors: **Renqiang Min**, Princeton, NJ (US); **Dongjin Song**, San Diego, CA (US)

Publication Classification

(57) **ABSTRACT**

Systems and methods are disclosed for operating a machine, by receiving training data from one or more sensors; training a machine learning module with the training data by: partitioning a data matrix into smaller submatrices to process in parallel and optimized for each processing node; for each submatrix, performing a greedy search for rank-one solutions; using alternating direction method of multipliers (ADMM) to ensure consistency over different data blocks; and controlling one or more actuators using live data and the learned module during operation.
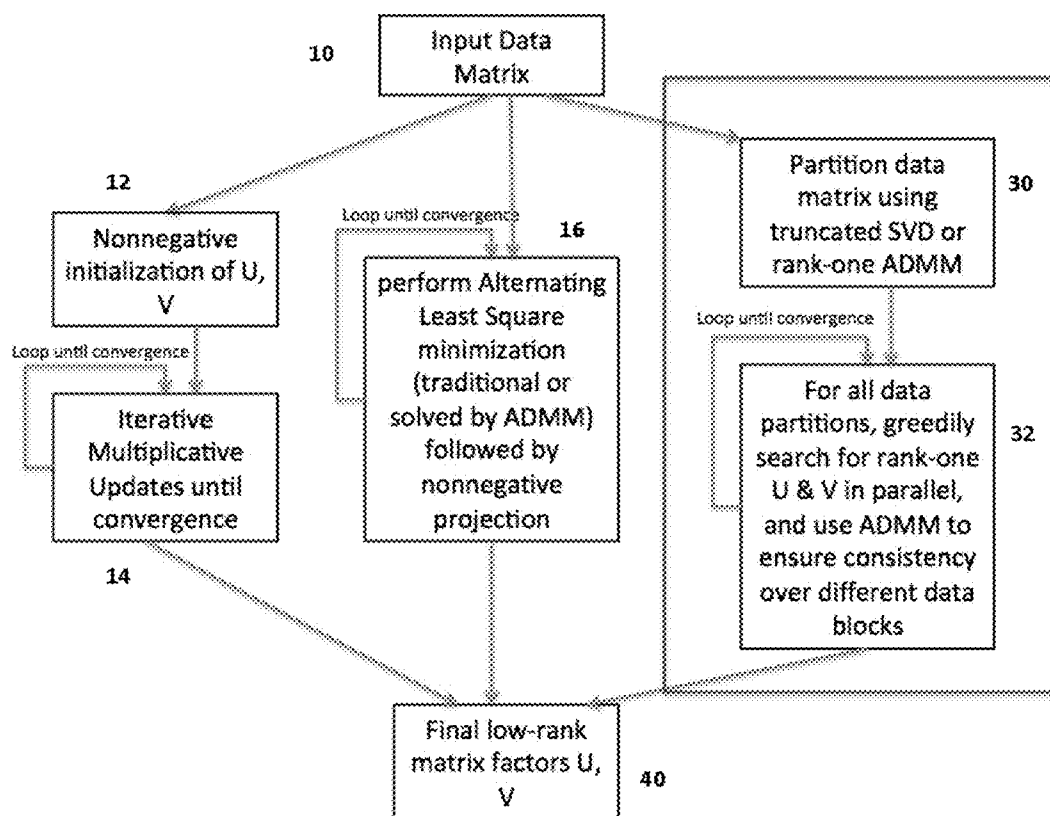
Input Data
Matrix

10

12

Nonnegative
initialization of U,
V

Loop until convergence   16

perform Alternating
Least Square
minimization
(traditional or
solved by ADMM)
followed by
nonnegative
projection

Loop until convergence

Iterative
Multiplicative
Updates until
convergence

14

Partition data
matrix using
truncated SVD or
rank-one ADMM

30

Loop until convergence

For all data
partitions, greedily
search for rank-one
U & V in parallel,
and use ADMM to
ensure consistency
over different data
blocks

32

Final low-rank
matrix factors U,
V

40

FIG. 1

FIG. 2

200

Physical System(s)
202

Component 1
204

Component 2
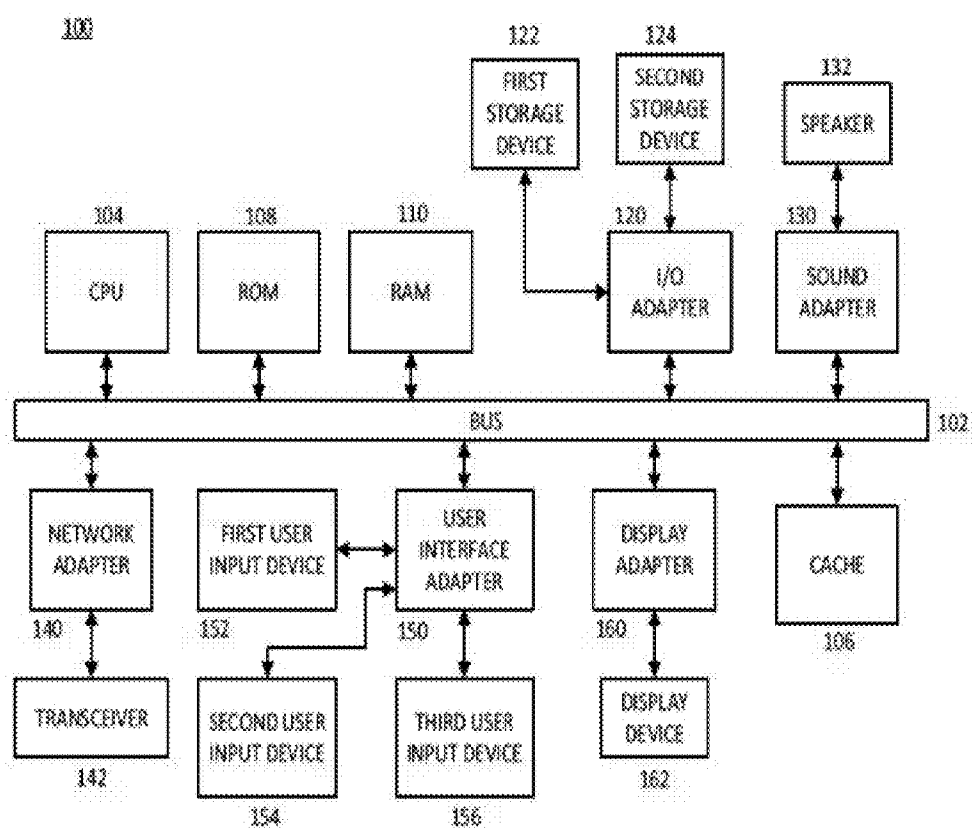206

Component 3
208

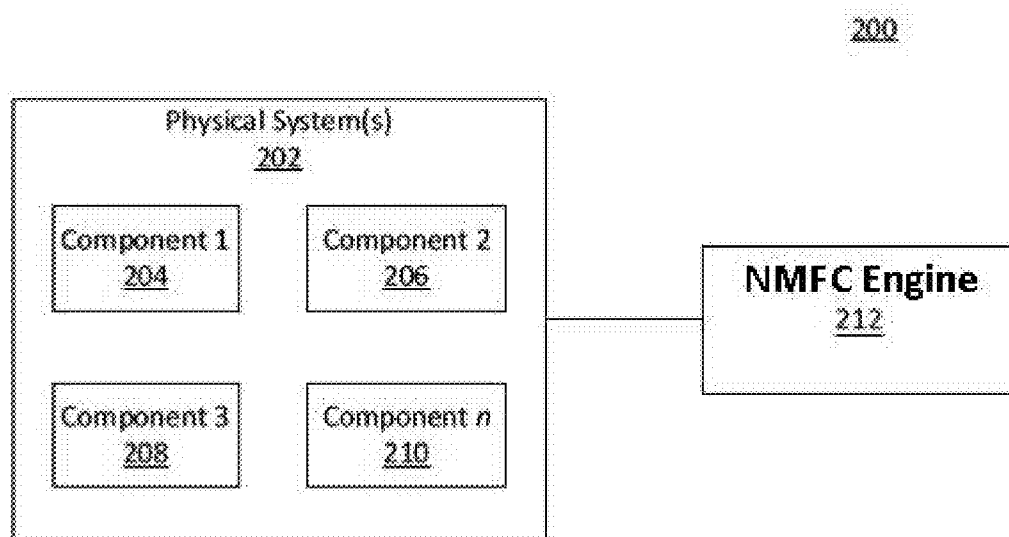Component n
210
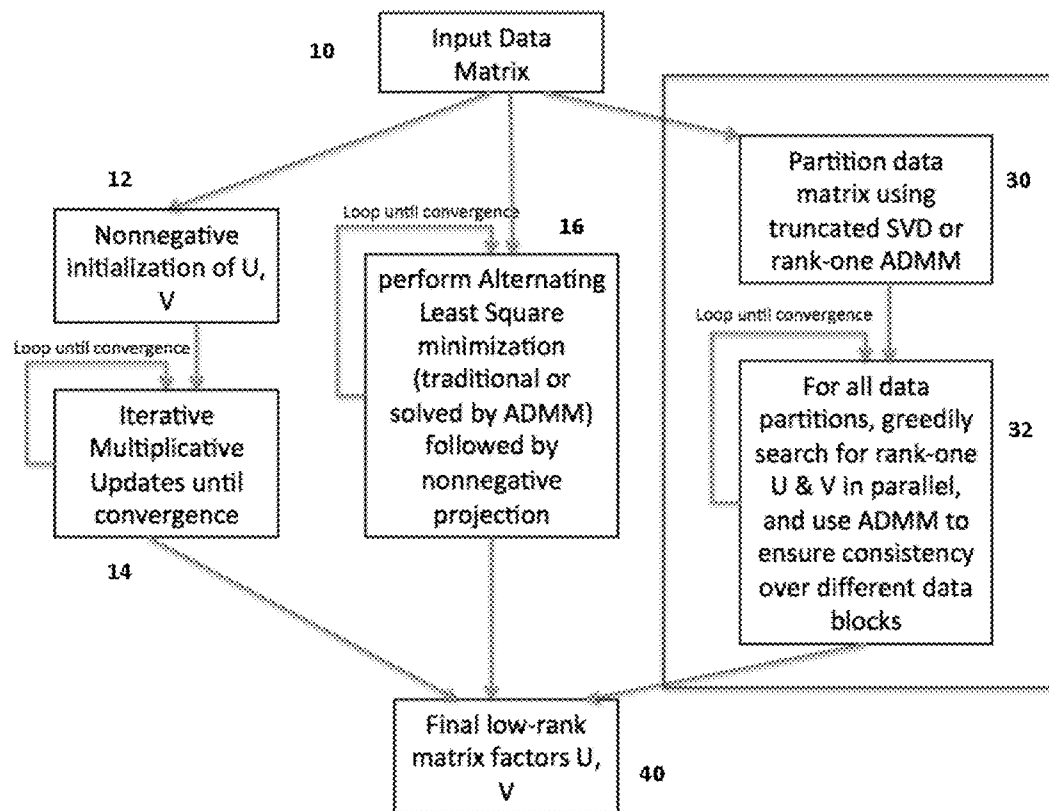
NMFC Engine
212

FIG. 3

FAST DISTRIBUTED NONNEGATIVE MATRIX FACTORIZATION AND COMPLETION FOR BIG DATA ANALYTICS

[0001]   The present application claims priority to Provisional Application 62135751 filed Mar. 20, 2015, the content of which is incorporated by reference.

BACKGROUND

[0002]   The present system is related to learning systems and big data analytic engines.

[0003]   Nonnegative matrix factorization and completion (NMFC), which aims to approximate a (partially) observed data matrix with two nonnegative low rank matrix factors, has been successfully applied in a wide range of machine learning applications, such as dimensionality reduction, collaborative filtering, compressed sensing. Nevertheless, due to the non-convex formulation and the underlying inverse problem, many existing solutions are not accurate and not scalable to handle large problems.

SUMMARY

[0004]   In one aspect, systems and methods are disclosed for operating a machine, by receiving training data from one or more sensors; training a machine learning module with the training data by: partitioning a data matrix into smaller submatrices to process in parallel and optimized for each processing node; for each submatrix, performing a greedy search for rank-one solutions; using alternating direction method of multipliers (ADMM) to ensure consistency over different data blocks; and controlling one or more actuators using live data and the learned module during operation.

[0005]   In another aspect, an efficient greedy and distributed process for nonnegative matrix factorization and completion (GD-NMFC) is disclosed. The system first partitions a large-scale data matrix into smaller submatrices in appropriate manners (e.g., random, rank-one SVD, and rank-one ADMM). Next, for each submatrix, the system searches for rank-one solutions of NMFC based upon ADMM in a greedy manner and concatenate them to form the low rank matrix factors. Finally, the solutions of each subproblem are concatenated to form the final solution and we show both the convergence of the algorithm and its error bound.

[0006]   Advantages of the system may include one or more of the following. The matrix inverse problem is eliminated and the engine load is only linearly proportional to the matrix rank, and the system is much faster than other methods for NMFC. The system can use a warm start technique for dividing large-scale datasets and perform distributed optimization in parallel. Therefore, it is scalable for big data analytics.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007]   FIG. 1 shows an efficient greedy and distributed engine for nonnegative matrix factorization and completion (GD-NMFC).

[0008]   FIG. 2 shows an exemplary processing system to which the present principles may be applied, in accordance with an embodiment of the present principles.

[0009]   FIG. 3 shows a high level diagram of an exemplary physical system including an aging profiling engine, in accordance with an embodiment of the present principles.

DESCRIPTION

[0010]   An efficient greedy and distributed engine for nonnegative matrix factorization and completion (GD-NMFC) is detailed.

[0011]   In FIG. 1, an input data matrix is received (10). A non-negative initialization of matrices U and V is done (12). The process performs iterative multiplicative updates until convergence is achieved (14). Alternatively, from 10, the process performs least square minimization followed by a non-negative projection (16). In another alternative from 10, the process partitions the data matrix using truncated SVD or rank-one ADMM (30). Next, for all data partitions, the process greedily searches for rank-one U&V in parallel, and uses ADMM to ensure consistency over different data blocks (32) and then loops until convergence is achieved. From 14, 16, or 32, the process selects the final low-rank matrix factors U& V as the result (40).

[0012]   Nonnegative matrix factorization and completion (NMFC) has been used to address scalability issues in machine learning by looking for two nonnegative low rank matrix factors, $U \in R^{n \times d}$ and $V \in R^{r \times d}$ to recover a partially (fully) observed data matrix $X \in R^{n \times d}$ of n samples and d features, i.e.,

$$\min_{U,V} \frac{1}{2} \|P_\Omega(X - UV)\|_F^2 \tag{1}$$

$$\text{s.t.} \quad U \geq 0, V \geq 0$$

where r denotes rank, $\|\cdot\|_F$ is the Frobenius norm, the inequalities are element-wise, and P_Ω is a projection operator such that the (i,j)-th entry of $P_\Omega(X)$ is $X_{i,j}$ if $(i,j) \in \Omega$ and zero otherwise.

[0013]   GD-NMFC partitions a large-scale data matrix into smaller submatrices with appropriate techniques (e.g., random, rank-one SVD, and rank-one ADMM) such that each submatrix is parallelly optimized over a single node. Then, for each submatrix, we conduct greedy search for rank-one solutions based upon ADMM. The matrix inverse problem is eliminated and the optimization of NMFC can be dramatically accelerated, especially when r is large.

[0014]   GD-NMFC avoids the possible failures caused by biased rank estimation and is SVD-free since it relies on optimizing over U and V, iteratively. Meanwhile, greedy searching for the optimal directions based upon rank-one approximation from 1 to r is more efficient than updating r directions in all iterates. In addition, the lower rank solution before each rank increment can be seen as a "warm start" of the next higher rank minimization and thus speed up convergence.

[0015]   Our empirical studies over several real-world large scale datasets verify that GD-NMFC is more efficient than state-of-the-art methods while achieving similar or better performance.

[0016]   Given a matrix $X \in R^{n \times d}$ of n rows and d columns, we use $x_j \in R^n$ to denote its j-th column, use $x^i \in R^{1 \times d}$ to denote its i-th row, and use X_{ij} to denote the entry in i-th row and j-th column of X. The Frobenius norm of X is defined as $\|X\|_F = \sqrt{Tr(X^T X)}$ where Tr(•) denote the trace of a square matrix. P_Ω is a projection operator such that the (i,j)-th entry

of $P_\Omega(X)$ is $X_{i,j}$ if $(i,j) \in \Omega$ and zero otherwise. The inner product of two matrices X and Y is defined as $\langle X,Y \rangle = \text{Tr}(X^T Y)$.

[0017] NMF aims to reconstruct a fully observed data matrix with two nonnegative low rank matrix factors and has been widely applied for face recognition, object classification, and clustering. To optimize NMF, multiplicative update rule (MUR) can be used. MUR is the most popular method for NMF and it can be derived based upon Karush-Kuhn-Tucker (KKT) optimality condition, i.e.,

$$U_{ik} \leftarrow U_{ik} \frac{(XV^T)_{ik}}{(UVV^T)_{ik}} \tag{2}$$

and

$$V_{kj} \leftarrow V_{kj} \frac{(U^T X)_{kj}}{(U^T UV)_{kj}}. \tag{3}$$

[0018] With nonnegative initializations of U and V, both U and V will remain nonnegative in each iteration. Since MUR converges slowly in many practical applications, ALS and projected gradient method can be utilized as the substitutions. ALS minimizes the least square cost function $\|X - UV\|_F^2$ with respect to U and V, iteratively. The procedure is based upon two closed form solutions, iteratively, i.e.,

$$U = P_+(XV^T(VV^T)^*) \tag{4}$$

and

$$V = P_+((U^T U)^* U^T X), \tag{5}$$

where * denotes pseudo-inverse and P_+ projects the matrix onto a nonnegative set, i.e., $P_+(U) = \max(U,0)$. Instead of calculating pseudo-inverse in ALS, projected gradient method first derive the gradient of $PX - UVP_F^2$ with respect to U (or V) and then update U (or V) based upon projected gradient update. Fast HALS [9] is a coordinate descent approach which optimize the i-th row of U and i-th column of V when i iterates from 1 to r until convergence is achieved.

[0019] An alternative to ALS is called alternating direction method of multiplier (ADMM). ADMM introduces two auxiliary variables A and B and considers the following optimization problem:

$$\min_{U,V,A,B} \frac{1}{2}\|X - UV\|_F^2 \tag{6}$$

s.t. $U - A = 0, V - B = 0, A \geq 0, B \geq 0.$

[0020] To solve this problem, the augmented Lagrangian function of 6 can be derived as:

$$L(U, V, A, B, \Lambda, \Pi) = \tag{7}$$

$$\frac{1}{2}\|X - UV\|_F^2 + \langle \Lambda, U - A \rangle + \langle \Pi, V - B \rangle + \frac{\rho}{2}\|U - A\|_F^2 + \frac{\rho}{2}\|V - B\|_F^2$$

where $A \subset R^{n \times r}$ and $\Pi \subset R^{r \times d}$ are Lagrange multipliers, $\langle \bullet \rangle$ is the matrix inner product, and $\rho > 0$ is the penalty parameter

for the constraints. By minimizing L with respect to U, V, A, B, Λ, and Π, we can update each parameter iteratively until convergence.

[0021] Since ADMM involves matrix inverse problems which are computationally expensive when the matrix rank r is relatively large, it may not be scalable to large scale matrices as they are more likely to contain a large r. Therefore, an efficient greedy and distributed algorithm for nonnegative matrix factorization and completion (GD-NMFC) is developed.

[0022] Matrix completion aims to recover the whole data matrix X from partially observed entries or undersampled linear measurements by leveraging the interaction between different instances which can be well captured with low-rank structure. In the past few years, matrix completion has been applied various applications, such as collaborative filtering in recommender system and link recommendation in social network analysis.

[0023] Since it is intractable to minimize the matrix rank exactly in general case, the trace norm or nuclear norm is widely utilized as a convex surrogate of rank. Nevertheless, it is computationally expensive to solve the standard low rank or trace norm problem as it involves computing SVD. In recent years, many approaches have been developed to tackle this problem. Most of these methods, however, still involve the computation of SVD or truncated SVD in each iteration and thus cannot be scalable to large-scale problems. Several methods approximate the trace norm with UV and proceed by alternatively optimizing U and V. It has been shown that linear convergence rate can be established theoretically with properly designed optimization procedure and appropriate initialization (Jain et. al. 2013). The computational complexity of these approaches, however, is quadratically proportional to the rank of the estimated matrix and thus cannot be scalable for large matrices.

[0024] We sequentially detail greedy distributed NMFC with three steps. We first describe how to partition a large data matrix x into smaller submatrices. Next, we introduce rank-one ADMM for NMFC based upon a submatrix. Finally, we show how to aggregate the solutions of all submatrices.

[0025] Data Partition

[0026] When the number of samples n is extremely large in a data matrix X, solving NMFC over a single node will be infeasible. In this case, we can partition X into smaller submatrices, i.e., $X = [X(1) \in R^{r_1 \times d}; X(2) \in R^{r_2 \times d}; \ldots; X(c) \in R^{r_c \times d}]$ where $\Sigma_i r_i = r$ such that each one of them can be optimized parallelly.

[0027] In practical applications, three different partition methods could be utilized:

[0028] uniformly sample a number of samples (i.e., r_i) from X for the i-th node;

[0029] first perform truncated SVD by setting r=1, project its eigenvector over nonnegative set, and then conduct k-means clustering over the eigenvector;

[0030] first perform ADMM for NMFC by setting r=1 and then conduct k-means clustering over the eigenvector.

[0031] The second and third ways of partition can be seen as a "warm start" for the the second rank optimization and are expected to speed up convergence.

[0032] Rank-one ADMM for NMFC

[0033] Given each submatrix X(i) where i is the index of submatrices, rank-one ADMM can be used to optimize NMFC over this node. In rank-one ADMM, we greedy search for rank-one solutions based upon ADMM. In this way, the matrix inverse problem is eliminated and the

optimization of NMFC can be dramatically accelerated, especially when r is large.

[0034] In the first step (k=1) of rank-one ADMM, we set $\hat{X}(i)=X(i)$. For $k\geq2$, we set $\hat{X}(i)=\hat{X}(i)-u(i)_{k-1}{}^T v(i)^{k-1}$. Therefore, rank-one ADMM for the k-th rank can be formulated as:

$$\min_{u(i)_k, v(i)^k, a(i)_k, b(i)^k} \frac{1}{2}\left\|\hat{X}(i) - u(i)_k v(i)^k\right\|_F^2 \tag{8}$$

$$\text{s.t.} \quad u(i)_k - a(i)_k = 0, \, v(i)^k - b(i)^k = 0,$$

$$a(i)_k \geq 0, \, b(i)^k \geq 0,$$

$$v(i)^k - v(j)^k = 0, \, \forall \, j \neq i,$$

where the first three constraints are over this node and the last constraint is the communication between different nodes used to ensure that $v(i)^k$ on each node is consistent.

[0035] The augmented Lagrangian of Eq. 8 can be written as:

$$L(u(i)_k, v(i)^k, a(i)_k, b(i)^k, p, q, s) = \tag{9}$$

$$\frac{1}{2}\left\|\hat{X}(i) - u(i)_k v(i)^k\right\|_F^2 + \langle p, u(i)_k - a(i)_k \rangle +$$

$$\langle q, v(i)^k - b(i)^k \rangle + \sum_{j \neq i} \langle s, v(i)^k - v(j)^k \rangle \frac{\rho}{2}\|u(i)_k - a(i)_k\|_2^2 +$$

$$\frac{\rho}{2}\|v(i)^k - b(i)^k\|_2^2 + \sum_{j \neq i} \frac{\rho}{2}\|v(i)^k - v(j)^k\|_2^2.$$

[0036] where $p \in R^n$, $q \in R^{1 \times d}$, and $t \in R^{1 \times d}$ are Lagrangian multipliers, and $\rho > 0$ is the penalty parameter for the constraints. By minimizing L with respect to $u(i)_k$ $v(i)^k$, $a(i)_k$, $b(i)^k$, p, q, and s, iteratively, we can obtain the closed form solution of each parameter as follows:

$$\begin{cases} u(i)_k \leftarrow \dfrac{\left[\hat{X}(i)(v(i)^k)^T + \rho a(i)_k - p\right]}{[v(i)^k (v(i)^k)^T + \rho]} \\[12pt] v(i)^k \leftarrow \dfrac{\left[u(i)_k^T \hat{X}(i) + \rho b(i)^k + \sum_{j \neq i} \rho v(j)^k - q - (c-1)s\right]}{[(u(i)_k)^T u(i)_k + \rho]} \\[12pt] a(i)_k \leftarrow P_+(u(i)_k + p/\rho) \\[6pt] b(i)^k \leftarrow P_+(v(i)^k + q/\rho) \\[6pt] p \leftarrow p + \rho(u(i)_k - a(i)_k) \\[6pt] q \leftarrow q + \rho(v(i)^k - b(i)^k) \\[6pt] s \leftarrow s + \dfrac{\rho}{c-1} \sum_{j \neq i} (v(i)^k - v(j)^k) \end{cases} \tag{10}$$

[0037] In one embodiment, Technique 1 is executed by the processor 104 of FIG. 2 as follows:

Technique 1 Greedy Distributed NMFC
Input: X ∈ $R^{n \times d}$, $u(i)_k$, $v(i)^k$, $a(i)_k$, $b(i)^k$, p, q, s, ρ
Output: U and V
1:     Partition X into submatrices, i.e. X = [X (1) ∈ $R^{r1 \times d}$; X(2) ∈ $R^{r2 \times d}$; . . . ; X(c) ∈ $R^{rc \times d}$]

-continued

Technique 1 Greedy Distributed NMFC
2:    For i-th node
3:      For k = 1 to r,
4:       Repeat:
5:        Update $u(i)_k$ based upon Eq. 10;
6:        Update $v(i)^k$ based upon Eq. 10;
7:        Update $a(i)_k$ based upon Eq. 10;
8:        Update $b(i)^k$ based upon Eq. 10;
9:        Update p based upon Eq. 10;
10:      Update q based upon Eq. 10;
11:       k = k + 1
12:      Until convergence.
13:      U(i) = [U(i); $u(i)_k$], V(i) = [V(i), $v(i)^k$], A(i) = [A(i); $a(i)_k$], B(i) = [B(i), $b(i)^k$].
14:     End for.
15:   End for i-th node.
16:   U = [U; U(i)], V = V(i), S = [S; S(i)], T = T(i)

[0038] In another embodiment, Technique 2 is executed by the processor 104 of FIG. 2. In Technique 2, U and V are randomly initialized and the matrices are set to zero matrices of appropriate sizes. The stopping criterion for ADMM is met if the objective in Eq. 6 does not improve relative to a tolerance value. In each step, the main computation are U(k+1) updating and V(k+1) updating.

Technique 2 ADMM for NMF
1: Input: X ∈ $R^{n \times d}$, U ∈ $R^{n \times r}$, V ∈ $R^{r \times d}$, S ∈ $R^{n \times r}$, T ∈ $R^{r \times d}$, ρ
2: Output: U and V
3: Set k = 0 (index of iteration)
4: Repeat:
5:     U (k + 1) = (XV(k)^T + ρS(k) − Λ(k)) (V(k)V(k)^T + ρI)$^{-1}$
6:     V(k + 1) = (U(k + 1)^T U(k + 1) + ρI)$^{-1}$ (U(k + 1)^T X + ρT(k) − Π(k))
7:     S(k + 1) = P$_+$ (U(k + 1) + Λ(k)/ρ)
8:     T (k + 1) = P$_+$ (V(k + 1) + Π(k)/ρ)
9:     Λ(k + 1) = Λ(k) + ρ(U(k + 1) − S(k + 1))
10:    Π(k + 1) = Π(k) + ρ(V(k + 1) − T(k + 1))
11:    k = k + 1.
12: Until convergence.

[0039] The foregoing details a rank-one alternating direction method of multiplier (ADMM) for nonnegative matrix factorization (NMF). In each step, rank-one ADMM seeks for a rank-one solution of NMF based upon ADMM and utilizes greedy search to obtain low rank matrix factors. In this way, rank-one ADMM avoids the underlying matrix inverse problem and the computation is only linearly proportional to the rank r. Thorough empirical studies over several real-world large scale datasets demonstrate that the present system improves computer performance and is more efficient than conventional systems while achieving similar or better learning performance.

[0040] We conducted empirical studies based upon two publicly available datasets, i.e., UMIST and Coil20. Our results demonstrated that rank-one ADMM is more efficient and effective than MUR, ALS, and traditional ADMM.

[0041] FIG. 2 with an exemplary processing system 100, to which the present principles may be applied, is illustratively depicted in accordance with an embodiment of the present principles, for operating a machine, by receiving training data from one or more sensors; training a machine learning module with the training data by: partitioning a data

matrix into smaller submatrices to process in parallel and optimized for each processing node; for each submatrix, performing a greedy search for rank-one solutions; using alternating direction method of multipliers (ADMM) to ensure consistency over different data blocks; and controlling one or more actuators using live data and the learned module during operation.

[0042] The processing system 100 includes at least one processor (CPU) 104 operatively coupled to other components via a system bus 102. A cache 106, a Read Only Memory (ROM) 108, a Random Access Memory (RAM) 110, an input/output (I/O) adapter 120, a sound adapter 130, a network adapter 140, a user interface adapter 150, and a display adapter 160, are operatively coupled to the system bus 102.

[0043] A first storage device 122 and a second storage device 124 are operatively coupled to system bus 102 by the I/O adapter 120. The storage devices 122 and 124 can be any of a disk storage device (e.g., a magnetic or optical disk storage device), a solid state magnetic device, and so forth. The storage devices 122 and 124 can be the same type of storage device or different types of storage devices.

[0044] A speaker 132 is operatively coupled to system bus 102 by the sound adapter 130. A transceiver 142 is operatively coupled to system bus 102 by network adapter 140. A display device 162 is operatively coupled to system bus 102 by display adapter 160.

[0045] A first user input device 152, a second user input device 154, and a third user input device 156 are operatively coupled to system bus 102 by user interface adapter 150. The user input devices 152, 154, and 156 can be any of a keyboard, a mouse, a keypad, an image capture device, a motion sensing device, a microphone, a device incorporating the functionality of at least two of the preceding devices, and so forth. Of course, other types of input devices can also be used, while maintaining the spirit of the present principles. The user input devices 152, 154, and 156 can be the same type of user input device or different types of user input devices. The user input devices 152, 154, and 156 are used to input and output information to and from system 100.

[0046] Of course, the processing system 100 may also include other elements (not shown), as readily contemplated by one of skill in the art, as well as omit certain elements. For example, various other input devices and/or output devices can be included in processing system 100, depending upon the particular implementation of the same, as readily understood by one of ordinary skill in the art. For example, various types of wireless and/or wired input and/or output devices can be used. Moreover, additional processors, controllers, memories, and so forth, in various configurations can also be utilized as readily appreciated by one of ordinary skill in the art. These and other variations of the processing system 100 are readily contemplated by one of ordinary skill in the art given the teachings of the present principles provided herein.

[0047] Referring now to FIG. 3, a high level schematic 200 of an exemplary physical system including an NMFC engine 212 is illustratively depicted in accordance with an embodiment of the present principles. In one embodiment, one or more components of physical systems 202 may be controlled and/or monitored using an engine 212 according to the present principles. The physical systems may include a plurality of components 204, 206, 208. 210 (e.g., Components 1, 2, 3, . . . n), for performing various system processes, although the components may also include data regarding, for example, financial transactions and the like according to various embodiments.

[0048] In one embodiment, components 204, 206, 208, and 210 may include any components now known or known in the future for performing operations in physical (or virtual) systems (e.g., temperature sensors, deposition devices, key performance indicator (KPI), pH sensors, financial data, etc.), and data collected from various components (or received (e.g., as time series)) may be employed as input to the engine 212 according to the present principles. The engine/controller 212 may be directly connected to the physical system or may be employed to remotely monitor and/or control the quality and/or components of the system according to various embodiments of the present principles.

[0049] While the machine-readable storage medium is shown in an exemplary embodiment to be a single medium, the term "machine-readable storage medium" should be taken to include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more sets of instructions. The term "machine-readable storage medium" shall also be taken to include any medium that is capable of storing or encoding a set of instructions for execution by the machine and that cause the machine to perform any one or more of the methodologies of the present invention. The term "machine-readable storage medium" shall accordingly be taken to include, but not be limited to, solid-state memories, and optical and magnetic media.

[0050] It is to be understood that the above description is intended to be illustrative, and not restrictive. Many other embodiments will be apparent to those of skill in the art upon reading and understanding the above description. Although the present invention has been described with reference to specific exemplary embodiments, it will be recognized that the invention is not limited to the embodiments described, but can be practiced with modification and alteration within the spirit and scope of the appended claims. Accordingly, the specification and drawings are to be regarded in an illustrative sense rather than a restrictive sense. The scope of the invention should, therefore, be determined with reference to the appended claims, along

What is claimed is:

1. A method for operating a machine, comprising:
receiving training data from one or more sensors;
training a machine learning module with the training data by:
 partitioning a large-scale data matrix into smaller sub-matrices;
 for each submatrix, greedily searching for rank-one solutions of Nonnegative matrix factorization/completion (NMFC) based upon alternating direction method of multipliers (ADMM);
 concatenating the sub-matrices to form low rank matrix factors;
 determining solutions of each subproblem and concatenated to form a final solution; and
controlling one or more actuators using live data and the learned module during operation.

2. The method of claim 1, wherein the partitioning comprises selecting using one of random, rank-one singular value decomposition (SVD), and rank-one ADMM.

3. The method of claim **1**, wherein computation is only linearly proportional to a matrix rank.

4. The method of claim **1**, wherein for each submatrix X(i) with i as an index of submatrices, using rank-one ADMM to optimize NMFC over a node.

5. The method of claim **1**, comprising uniformly selecting a number of samples for a node.

6. The method of claim **1**, comprising performing SVD by projecting an eigenvector over the nonnegative set, and then conducting k-means clustering over the eigenvector.

7. The method of claim **1**, comprising performing ADMM for NMFC by conducting k-means clustering over an eigenvector.

8. The method of claim **1**, comprising providing a warm start for a second rank optimization.

9. The method of claim **1**, wherein for rank-one ADMM, comprising performing a greedy search for rank-one solutions based upon ADMM.

10. The method of claim **1**, comprising determining:

$$
\begin{cases}
u(i)_k \leftarrow \dfrac{\left[\hat{X}(i)(v(i)^k)^T + \rho a(i)_k - p\right]}{\left[v(i)^k(v(i)^k)^T + \rho\right]} \\[2ex]
v(i)^k \leftarrow \dfrac{\left[u(i)_k^T \hat{X}(i) + \rho b(i)^k + \sum\limits_{j \neq i} \rho v(j)^k - q - (c-1)s\right]}{\left[(u(i)_k)^T u(i)_k + \rho\right]} \\[2ex]
a(i)_k \leftarrow P_+(u(i)_k + p/\rho) \\
b(i)^k \leftarrow P_+(v(i)^k + q/\rho) \\
p \leftarrow p + \rho(u(i)_k - a(i)_k) \\
q \leftarrow q + \rho(v(i)^k - b(i)^k) \\
s \leftarrow s + \dfrac{\rho}{c-1} \sum\limits_{j \neq i}(v(i)^k - v(j)^k)
\end{cases}
$$

where X is a partially observed data matrix of n samples and d features, i is an index, c is a constant, p, q and t are Lagrangian multipliers, p is a penalty operator, and U and V are nonnegative low rank matrix factors.

11. A method for operating a machine, comprising
receiving training data from one or more sensors;
training a machine learning module with the training data by:
  partitioning a data matrix into smaller submatrices to process in parallel and optimized for each processing node;

  for each submatrix, performing a greedy search for rank-one solutions;
  using alternating direction method of multipliers (ADMM) to ensure consistency over different data blocks; and controlling one or more actuators using live data and the learned module during operation.

12. The method of claim **11**, comprising partitioning the data matrix into smaller submatrices using one of: random, rank-one SVD, and rank-one ADMM technique.

13. The method of claim **11**, wherein for each submatrix X(i) with i as an index of submatrices, using rank-one ADMM to optimize NMFC over a node.

14. The method of claim **11**, comprising uniformly selecting a number of samples for a node.

15. The method of claim **1**, comprising performing SVD by projecting an eigenvector over the nonnegative set, and then conducting k-means clustering over the eigenvector.

16. The method of claim **1**, comprising performing ADMM for NMFC by conducting k-means clustering over an eigenvector.

17. The method of claim **1**, comprising providing a warm start for a second rank optimization.

18. The method of claim **1**, wherein for rank-one ADMM, comprising performing a greedy search for rank-one solutions based upon ADMM.

19. The method of claim **1**, comprising determining:

$$
\begin{cases}
u(i)_k \leftarrow \dfrac{\left[\hat{X}(i)(v(i)^k)^T + \rho a(i)_k - p\right]}{\left[v(i)^k(v(i)^k)^T + \rho\right]} \\[2ex]
v(i)^k \leftarrow \dfrac{\left[u(i)_k^T \hat{X}(i) + \rho b(i)^k + \sum\limits_{j \neq i} \rho v(j)^k - q - (c-1)s\right]}{\left[(u(i)_k)^T u(i)_k + \rho\right]} \\[2ex]
a(i)_k \leftarrow P_+(u(i)_k + p/\rho) \\
b(i)^k \leftarrow P_+(v(i)^k + q/\rho) \\
p \leftarrow p + \rho(u(i)_k - a(i)_k) \\
q \leftarrow q + \rho(v(i)^k - b(i)^k) \\
s \leftarrow s + \dfrac{\rho}{c-1} \sum\limits_{j \neq i}(v(i)^k - v(j)^k)
\end{cases}
$$

where X is a partially observed data matrix of n samples and d features, i is an index, c is a constant, p, q and t are Lagrangian multipliers, ρ is a penalty operator, and U and V are nonnegative low rank matrix factors.

* * * * *