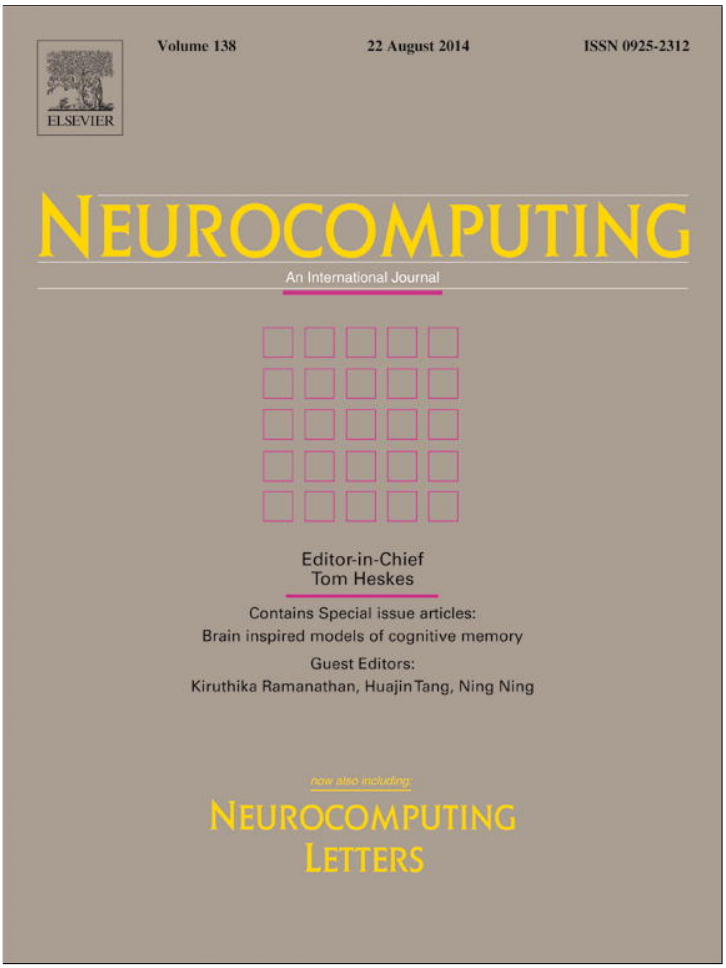


Neurocomputing 138 (2014) 238–247

Contents lists available at ScienceDirect

Neurocomputing

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

Recurrent neural network for approximate nonnegative matrix factorization

Giovanni Costantini [a], Renzo Perfetti [b,*], Massimiliano Todisco [a]

[a] Department of Electronic Engineering, University of Rome 'Tor Vergata', Italy
[b] Department of Electronic and Information Engineering, University of Perugia, Italy

ARTICLE INFO

Article history:
Received 11 July 2013
Received in revised form
30 October 2013
Accepted 2 February 2014
Communicated by L.C. Jain
Available online 17 February 2014

Keywords:
Recurrent neural networks
Lagrangian networks
Nonnegative matrix factorization
Features extraction
Clustering

ABSTRACT

A recurrent neural network solving the approximate nonnegative matrix factorization (NMF) problem is presented in this paper. The proposed network is based on the Lagrangian approach, and exploits a partial dual method in order to limit the number of dual variables. Sparsity constraints on basis or activation matrices are included by adding a weighted sum of constraint functions to the least squares reconstruction error. However, the corresponding Lagrange multipliers are computed by the network dynamics itself, avoiding empirical tuning or a validation process. It is proved that local solutions of the NMF optimization problem correspond to as many stable steady-state points of the network dynamics. The validity of the proposed approach is verified through several simulation examples concerning both synthetic and real-world datasets for feature extraction and clustering applications.

1. Introduction

The idea of using analogue circuits to solve mathematical programming problems can be traced back to the works of Pyne [1] and Dennis [2]. A canonical nonlinear programming circuit was proposed by Chua and Lin [3], later extended by Wilson [4]. Kennedy and Chua [5] recast the canonical circuit in a neural network framework and proved the stability. All the networks in [3–5] are based on the penalty function method, which gives exact solutions only if the penalty parameter tends to infinity, a condition impossible to meet in practice. To avoid the penalty functions, Zhang and Constantinides [6] proposed a Lagrangian approach to solve quadratic programming (QP) problems with equality constraints. The method can be extended to problems including both equality and inequality constraints converting inequalities into equalities by introducing slack variables. In addition, bound constraints on the variables, often arising in practical problems, can be treated in the same way at the expense of a huge number of variables. In the last decades several Lagrange neural networks have been proposed to solve specific optimization problems, handling both equality and inequality constraints as well as bounds on the variables [7–23].

Among the optimization problems of main interest in the context of machine learning and data, analysis there is nonnegative matrix factorization (NMF) [24]. The problem consists in finding reduced rank nonnegative factors to approximate a given nonnegative data matrix. This factorization can be interpreted as a representation of data using nonnegative basis vectors and nonnegative activation vectors. Like PCA, it can be used to accomplish the goal of reducing the number of variables required for data representation, with the additional constraint of non-negativity to enforce an additive, not subtractive, combination of parts. The idea of NMF can be traced back to Paatero and Tapper [25]. However, they were the seminal papers of Lee and Seung [26,27] which attracted the interest of many researchers. Applications of NMF have been proposed in diverse fields, e.g. text mining [28], document clustering [29,30], image reconstruction [31], human action recognition [32], discovering muscle synergies [33], EEG classification [34] and music transcription [35,36]. The relation between NMF and some clustering techniques has been proven [37,38], and several extensions and variants have been proposed in the literature [39–42].

Different algorithms can be used to solve the NMF problem. In particular, the most known are the multiplicative rules [26,27,42], and projected alternating least squares (ALS) algorithms [39]. With respect to other dimensionality reduction methods, probably the most intriguing feature of NMF is the capacity of finding the underlying parts-based structure of complex data. However, there is no explicit guarantee in the method to support this property,

* Corresponding author. Tel.: +39 0755853631.
E-mail addresses: costanti@uniroma2.it (G. Costantini), renzo.perfetti@unipg.it (R. Perfetti), massimiliano.todisco@uniroma2.it (M. Todisco).

http://dx.doi.org/10.1016/j.neucom.2014.02.007
0925-2312

which can be enforced introducing sparseness constraints as proposed by Hoyer [43] and Pascual-Montano et al. [44]. Due to the nonnegativity constraints, sparsity is strictly related to orthogonality among the basis vectors. Vice versa, imposing sparsity on the activation vectors, we can enforce an holistic representation of the data.

In the present paper we propose a neural network solver for the approximate NMF problem. It is a Lagrange programming neural network, using a projection operator to implement the nonnegativity constraints. A similar network has been proposed by the authors to solve convex optimization problems [14,17]. In this paper it is shown how this approach can properly work in a non-convex problem as the approximate NMF.

The rest of this paper is organized as follows. In Section 2, the NMF optimization problem is formulated. In Section 3, the proposed neural network is introduced and illustrated. Section 4 we investigate the network's dynamic behaviour. Section 5 presents the simulation results. Finally, some comments conclude the paper.

2. NMF optimization problem

Let $\Re_+$ denote the set of nonnegative real numbers. Given a nonnegative matrix $\mathbf{V}\in\Re_+^{m\times n}$ and an integer $p<\min(m,n)$, the NMF problem consists in computing a reduced rank approximation of $\mathbf{V}$ given by the product $\mathbf{WH}$ of nonnegative matrices $\mathbf{W}\in\Re_+^{m\times p}$ and $\mathbf{H}\in\Re_+^{p\times n}$. This problem can be formulated as the minimization of the objective function $J(\mathbf{W},\mathbf{H})=||\mathbf{WH}-\mathbf{V}||^2$ with non-negativity constraints on $\mathbf{W}$ and $\mathbf{H}$. The NMF optimization problem is not convex, so it admits multiple local minima and the solution found by iterative algorithms depends on initialization. Moreover the problem is characterized by an intrinsic invariance, since the product $\mathbf{WH}$ is unchanged by replacing matrices $\mathbf{W}$ and $\mathbf{H}$ by the nonnegative matrices $\mathbf{WD}$ and $\mathbf{D}^{-1}\mathbf{H}$, where $\mathbf{D}$ is any invertible nonnegative matrix; this implies the non-existence of isolated local minima of the objective function.

The problem formulation is often extended to include auxiliary constraints on $\mathbf{W}$ and/or $\mathbf{H}$, in order to avoid the invariance problem, limit the number of local minima and enforce some desired characteristics of the solution. Sparsity of $\mathbf{W}$ is sometimes required to enforce a parts-based decomposition [24,39,43,44]; sparsity of $\mathbf{H}$ is required to improve the performance in clustering applications. It has been shown that imposing L_1 normalization on rows or columns is a straightforward way to enforce sparsity; L_1 normalization of nonnegative vectors simply requires a constraint on the sum of elements. In this paper we take into account NMF with the following additional constraints: L_1 normalization of columns of $\mathbf{W}$; L_1 normalization of rows of $\mathbf{H}$.

The NMF optimization problem, with L_1 normalization of $\mathbf{W}$ columns, can be stated as follows:

minimize

$$J(\mathbf{W},\mathbf{H})=||\mathbf{WH}-\mathbf{V}||^2 \tag{1a}$$

such that

$$\mathbf{W}\geq 0 \tag{1b}$$

$$\mathbf{H}\geq 0 \tag{1c}$$

$$||\mathbf{w}_j||_1=\sum_{i=1}^{m} w_{ij}=1,\quad j=1,\ldots,p \tag{1d}$$

where $\mathbf{w}_j\in\Re_+^m$ denotes the jth column of $\mathbf{W}$.

The Lagrangian function corresponding to problem (1) is [45]:

$$\mathcal{L}=J(\mathbf{W},\mathbf{H})+\sum_{j=1}^{p}\alpha_j\left(\sum_{i=1}^{m}w_{ij}-1\right)-\sum_{i=1}^{m}\sum_{j=1}^{p}\lambda_{ij}w_{ij}-\sum_{j=1}^{p}\sum_{k=1}^{n}\mu_{jk}h_{jk} \tag{2}$$

where λ_{ij} and μ_{jk} are the Lagrange multipliers corresponding to inequality constraints (1b) and (1c), respectively; α_j is the Lagrange multiplier of the jth equality constraint (1d).

The Karush–Khun–Tucker (KKT) first order conditions for the existence of a local minimizer of problem (1) are the following [45]:

$$\frac{\partial\mathcal{L}}{\partial w_{ij}}=\frac{\partial J}{\partial w_{ij}}+\alpha_j-\lambda_{ij}=0 \tag{3a}$$

$$\frac{\partial\mathcal{L}}{\partial h_{jk}}=\frac{\partial J}{\partial h_{jk}}-\mu_{jk}=0 \tag{3b}$$

$$\lambda_{ij}\geq 0 \tag{3c}$$

$$\mu_{jk}\geq 0 \tag{3d}$$

$$\lambda_{ij}w_{ij}=0 \tag{3e}$$

$$\mu_{jk}h_{jk}=0 \tag{3f}$$

$$w_{ij}\geq 0 \tag{3g}$$

$$h_{jk}\geq 0 \tag{3h}$$

$$\sum_{i=1}^{m}w_{ij}-1=0 \tag{3i}$$

In relations (3) we assume $i=1,\ldots,m$, $j=1,\ldots,p$, and $k=1,\ldots,n$.

Since the objective function (1a) is non-convex, KKT conditions (3) are only necessary [45].

3. Neural network model

For a convex constrained optimization problem, Lagrangian duality can be used to obtain the global solution [6,45]. The basic idea is to find the saddle point of the Lagrangian function, which is maximized with respect to the Lagrange multipliers (dual variables) and minimized with respect to the primal variables. Here, we propose the same strategy to find a (local) solution of non-convex problem (1). To limit the number of variables, we adopt a *partial dual* approach introducing the following *reduced* Lagrangian function:

$$L(\mathbf{W},\mathbf{H},\boldsymbol{\alpha})=J(\mathbf{W},\mathbf{H})+\sum_{j=1}^{p}\alpha_j\left(\sum_{i=1}^{m}w_{ij}-1\right) \tag{4}$$

where $\boldsymbol{\alpha}=[\alpha_1\ \ldots\ \alpha_p]^{\mathrm{T}}$ is the vector of Lagrange multipliers (dual variables) corresponding to the equality constraints (1d). Constraints (1b) and (1c) are not included in (4), avoiding $p(m+n)$ additional dual variables. To fulfill constraints (1b) and (1c), avoiding the drawbacks of the penalty function approach, we introduce the auxiliary variables $\omega_{ij},\eta_{ij}\in\Re$, being $w_{ij}=P(\omega_{ij})$, $h_{jk}=P(\eta_{jk})$ and $P(.)$ is the piecewise linear function defined as follows (Fig. 1):

$$P(x)=\begin{cases}0 & \text{if}\quad x<0\\ x & \text{if}\quad x\geq 0\end{cases} \tag{5}$$

Function (5) is a projection operator: the auxiliary variables can vary in $\Re$ according to the gradient of the Lagrangian function (4) while the true variables w_{ij}, h_{jk} are confined in $\Re_+$.

To find a saddle point of the Lagrangian function (4) a dynamical system can be used such that, along a trajectory, function L is

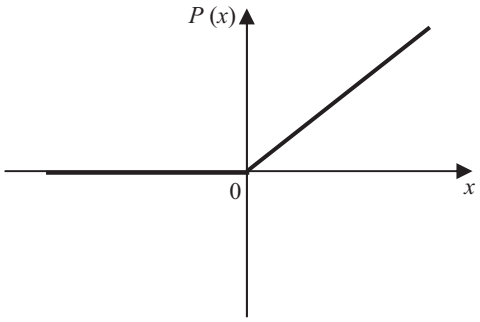


Fig. 1. Piecewise linear function P.

decreasing with each w_{ij} and h_{jk} and increasing with each α_j. Taking into account the definition (5), such system can be obtained by equating the time derivative of each variable to the corresponding component of the negative or positive gradient of L, i.e.

$$\tau\dot{\omega}_{ij} = -\frac{\partial L}{\partial w_{ij}} = -\frac{\partial J}{\partial w_{ij}} - \alpha_j, \quad i = 1,\ldots,m;\ j = 1,\ldots,p \tag{6a}$$

$$\tau\dot{\eta}_{jk} = -\frac{\partial L}{\partial h_{jk}} = -\frac{\partial J}{\partial h_{jk}}, \quad j = 1,\ldots,p;\ k = 1,\ldots,n \tag{6b}$$

$$\tau\dot{\alpha}_j = \frac{\partial L}{\partial \alpha_j} = \sum_{i=1}^{m} w_{ij} - 1, \quad j = 1,\ldots,p \tag{6c}$$

$\tau > 0$ is a time scaling factor.

On the boundary of the feasible region where $w_{ij}=0$, according to KKT condition (3a) we can have

$$\frac{\partial J}{\partial w_{ij}} + \alpha_j = \lambda_{ij} > 0$$

As a consequence, the time derivative in Eq. (6a) would be a negative constant, pushing ω_{ij} to $-\infty$. To guarantee the existence of finite equilibrium points, a corrective term is added to the right member of (6a):

$$\tau\dot{\omega}_{ij} = -\frac{\partial J}{\partial w_{ij}} - \alpha_j + w_{ij} - \omega_{ij}$$

If $w_{ij} > 0$, the corrective term is ineffective, being $w_{ij}=\omega_{ij}$. When $w_{ij}=0$, we obtain the following (finite) value at equilibrium:

$$\omega_{ij} = -\left(\frac{\partial J}{\partial w_{ij}} + \alpha_j\right) < 0$$

The same conclusion can be drawn for Eq. (6b), taking into account condition (3b). In conclusion, the state equations of the proposed neural network are as follows:

$$\tau\dot{\omega}_{ij} = -\frac{\partial J}{\partial w_{ij}} - \alpha_j + w_{ij} - \omega_{ij}, \quad i = 1,\ldots,m;\ j = 1,\ldots,p \tag{7a}$$

$$\tau\dot{\eta}_{jk} = -\frac{\partial J}{\partial h_{jk}} + h_{jk} - \eta_{jk}, \quad j = 1,\ldots,p;\ k = 1,\ldots,n \tag{7b}$$

$$\tau\dot{\alpha}_j = \sum_{i=1}^{m} w_{ij} - 1, \quad j = 1,\ldots,p \tag{7c}$$

As it will be shown in the following section, locally optimal solutions of problem (1) correspond to as many stable equilibrium points of system (7).

Note that Eq. (7c) can be explicitly solved as follows:

$$\alpha_j(t) = \frac{1}{\tau}\int_0^t \left(\sum_{k=1}^{m} w_{kj}(t') - 1\right) dt' + \alpha_j(0) \tag{7d}$$

while the gradients of J can be computed using the formulas [15]:

$$\frac{\partial J}{\partial \mathbf{W}} = 2(\mathbf{WH} - \mathbf{V})\mathbf{H}^T \tag{8a}$$

$$\frac{\partial J}{\partial \mathbf{H}} = 2\mathbf{W}^T(\mathbf{WH} - \mathbf{V}) \tag{8b}$$

The ijth element of (8a) and (8b) equals $\partial J/\partial w_{ij}$ and $\partial J/\partial h_{jk}$ respectively.

Lagrange multipliers α_j can be interpreted as *regularization parameters* trading off reconstruction error and sparsity of basis vectors. Usually, regularization parameters are empirically selected or obtained through a time-consuming cross validation process. In the proposed approach such procedures are avoided, since the regularization parameters, along with $\mathbf{W}$ and $\mathbf{H}$, are computed by the network during the dynamic evolution. Only the number p of basis vectors must be selected in advance.

The L_1 normalization of rows of $\mathbf{H}$ can be imposed replacing (1d) with the following:

$$||\mathbf{h}_j||_1 = \sum_{k=1}^{n} h_{jk} = 1, \quad j = 1,\ldots,p$$

where $\mathbf{h}_j \in \Re_+^n$ denotes the jth row of $\mathbf{H}$. Normalization of $\mathbf{H}$ rows can be obtained using the reduced Lagrangian function

$$L(\mathbf{W},\mathbf{H},\boldsymbol{\beta}) = J(\mathbf{W},\mathbf{H}) + \sum_{j=1}^{p} \beta_j \left(\sum_{k=1}^{n} h_{jk} - 1\right) \tag{9}$$

corresponding to the following state equations:

$$\tau\dot{\omega}_{ij} = -\frac{\partial J}{\partial w_{ij}} + w_{ij} - \omega_{ij}, \quad i = 1,\ldots,m;\ j = 1,\ldots,p \tag{10a}$$

$$\tau\dot{\eta}_{jk} = -\frac{\partial J}{\partial h_{jk}} - \beta_j + h_{jk} - \eta_{jk}, \quad j = 1,\ldots,p;\ k = 1,\ldots,n \tag{10b}$$

$$\tau\dot{\beta}_j = \frac{\partial L}{\partial \beta_j} = \sum_{k=1}^{n} h_{jk} - 1, \quad j = 1,\ldots,p \tag{10c}$$

Eqs. (7) and (10) describe the dynamic behavior of the recurrent neural network shown in Fig. 2. It is composed of $p(m+n)$ nonlinear 'neurons' whose outputs correspond to the elements of the unknown matrices. The inputs of each neuron are the negative derivative of the objective function (1a) and the negative Lagrange multipliers (for $\mathbf{W}$ or $\mathbf{H}$). The neuron consists of an integrator followed by a limiting nonlinearity, realizing function (5); feedback loops realize the terms $w_{ij}-\omega_{ij}$ and $h_{jk}-\eta_{jk}$. Moreover, there are p linear integrators computing the Lagrange multipliers α_j or β_j. The fully recurrent network in Fig. 2 can be used also in alternating mode, by letting the output of $\mathbf{H}$ ($\mathbf{W}$) integrators fixed, so that the network evolution concerns the $\mathbf{W}$ ($\mathbf{H}$) variables only. In this way the proposed network can be used to naturally implement the well known alternating least squares (ALS) approach [39], based on the property that the NMF problem is convex with respect to $\mathbf{W}$ or $\mathbf{H}$.

4. Stability of equilibrium points

In this section we present some results on the stability of equilibrium points of the proposed neural network. To this end, in the following we briefly review some definitions and theorems on dynamical systems.

An autonomous dynamical system is described by the differential equation:

$$\dot{\mathbf{z}} = f(\mathbf{z}) \tag{11}$$

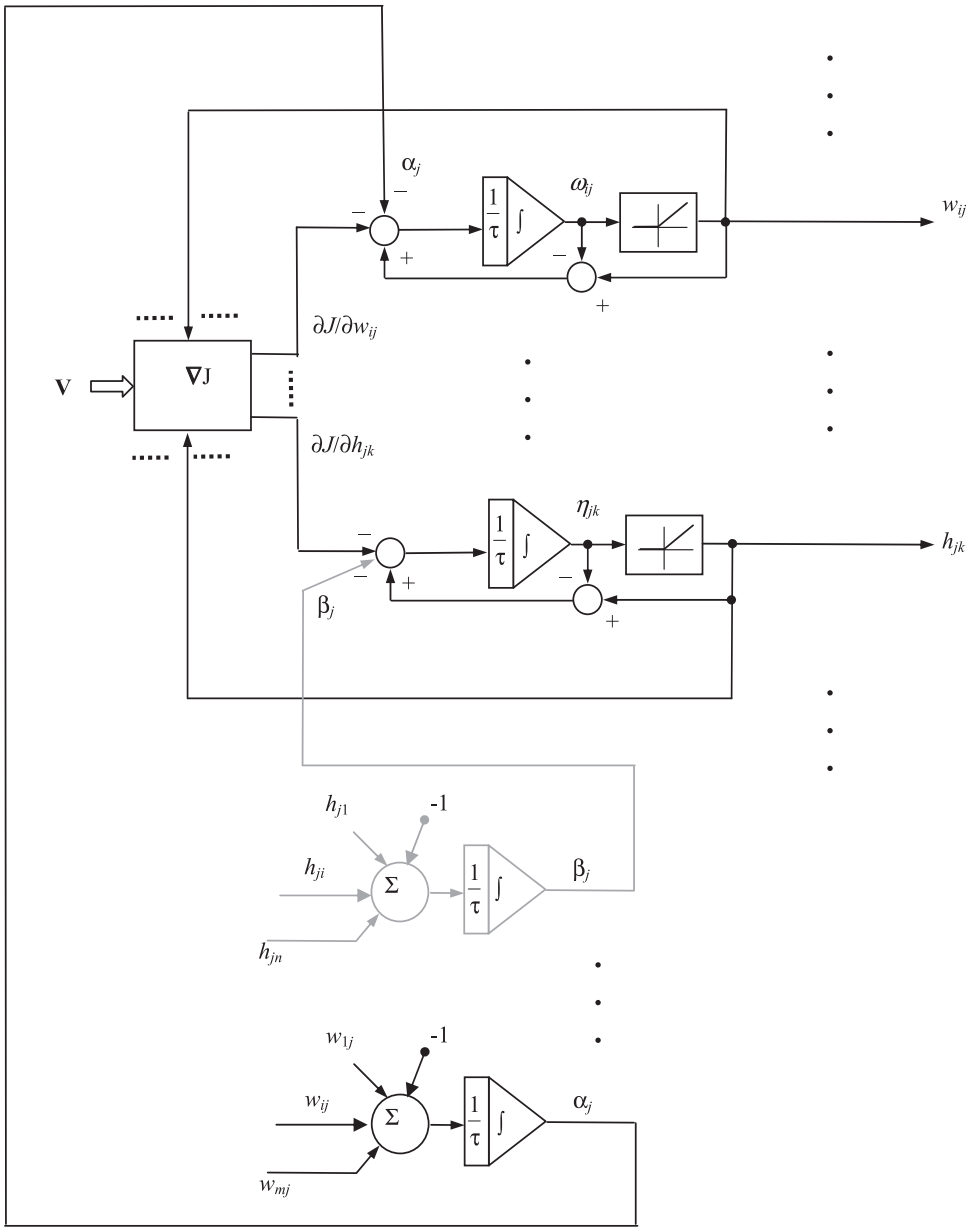


Fig. 2. Block diagram of proposed neural network.

where **z** is the state vector. If f is a C^1 function, the solution of (11) exists for $t > t_0$ and is unique for some given initial condition $\mathbf{z}(t_0)$ [46]. Let us introduce some useful definitions.

Definition 1. If $f(\mathbf{z}^*)=0$, $\mathbf{z}^*$ is called an *equilibrium point* or fixed point of system (11). If $\mathbf{z}(t_0)=\mathbf{z}^*$ then $\mathbf{z}(t)=\mathbf{z}^*$ for $t > t_0$.

Definition 2. An equilibrium point $\mathbf{z}^*$ of (11) is said *stable* if for every neighborhood U of $\mathbf{z}^*$ there is a neighborhood $U_1 \subset U$ such that every solution $\mathbf{x}(t)$ with $\mathbf{z}(t_0) \in U_1$ lies in U for every $t > t_0$.

Definition 3. If in addition to the property of Definition 2, U_1 can be chosen such that $\lim_{t\to\infty}\mathbf{z}(t)=\mathbf{z}^*$ for every $\mathbf{z}(t_0) \in U_1$, then $\mathbf{z}^*$ is said *asymptotically stable*.

Often, the stability of equilibrium points can be ascertained finding a positive definite function, $V(\mathbf{x})$, which decreases along solution curves of the dynamical system. Such function is called a *Liapunov function*, from the name of the Russian mathematician Liapunov for his work of 1892.

Liapunov direct method [46,47]. Let $\mathbf{z}^*$ be an equilibrium point of system (11). Let $V(\mathbf{z})$ be a continuous scalar function defined in a neighborhood U of $\mathbf{z}^*$, differentiable in $U-\mathbf{z}^*$ and such that:

1. $V(\mathbf{z}*)=0$ and $V(\mathbf{z}) > 0$ for $\mathbf{z} \neq \mathbf{z}v$.
2. $\dot{V}(\mathbf{z}) \leq 0$ for $\mathbf{z} \in U-\mathbf{z}^*$
   then $\mathbf{z}^*$ is stable.
   Furthermore, if
3. $\dot{V}(\mathbf{z}) < 0$ for $\mathbf{z} \in U-\mathbf{z}^*$

then $\mathbf{z}^*$ is asymptotically stable.

Now let us consider the dynamical system described by Eqs. (7). For ease of notation, we introduce the vector

$$\mathbf{x} = [w_{11}w_{12}\ldots w_{mp}h_{11}h_{12}\ldots h_{pn}]^T$$

with $Q=p(m+n)$ components given by the elements of both **W** and **H**, and vector **z** such that $\mathbf{x}=P(\mathbf{z})$. The state Eqs. (7a)–(7c)

become:

$$\tau\dot{z}_i = -\frac{\partial J}{\partial x_i} - \alpha_i + x_i - z_i, \quad i = 1, \ldots, Q \tag{12a}$$

$$\tau\dot{\alpha}_j = \sum_{r=1}^{m} w_{rj} - 1, \quad j = 1, \ldots, p \tag{12b}$$

With an abuse of notation, in Eq. (12a) it is assumed that $\alpha_i = \alpha_j$ if z_i corresponds to w_{rj}, $\forall r$; $\alpha_i = 0$ if z_i corresponds to an element of **H**. We can prove the following results.

Theorem 1. *Every local solution of the NMF optimization problem (1) corresponds to an equilibrium point of dynamical system (12).*

Proof. Taking into account the notation introduced above, it is easy to verify that KKT conditions (3) imply the following constraints for every i and j:

$$x_i > 0, \quad \frac{\partial J}{\partial x_i} + \alpha_i = 0 \tag{13a}$$

$$x_i = 0, \quad \frac{\partial J}{\partial x_i} + \alpha_i \geq 0 \tag{13b}$$

$$\sum_{r=1}^{m} w_{rj} - 1 = 0 \tag{14}$$

Condition (14) implies $\dot{\alpha}_j = 0$ for $\forall j$. Condition (13a) gives $\dot{z}_i = 0$ when $z_i = x_i$ $(x_i > 0)$. Condition (13b) implies $\dot{z}_i = 0$ when $z_i = -((\partial J/\partial x_i) + \alpha_i) \leq 0$. ▯

Theorem 2. *Every local solution of the NMF optimization problem (1) corresponds to a stable equilibrium point of dynamical system (12).*

Proof. Let us consider a local solution $\mathbf{x}^*$ of the underlying problem (1). According to Theorem 1 there exists an equilibrium point $(\mathbf{z}^*, \alpha^*)$ of system (12) corresponding to $\mathbf{x}^*$. Let

$$x'_i = x_i - x_i^* = P(z_i) - P(z_i^*) \tag{15a}$$

$$\alpha'_j = \alpha_j - \alpha_j^* \tag{15b}$$

$$z'_i = z_i - z_i^* \tag{15c}$$

From the definition of $P(.)$ it follows:

$$0 \leq \frac{x_i - x_i^*}{z_i - z_i^*} \leq 1, \quad i = 1, \ldots, Q \tag{16}$$

In a neighborhood U of the local minimum $\mathbf{x}^*$, we can approximate $J(\mathbf{x})$ using the following Taylor expansion:

$$J(\mathbf{x}) \cong J(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^{\mathrm{T}} \nabla J(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^{\mathrm{T}} \mathbf{A} (\mathbf{x} - \mathbf{x}^*)$$

where $\mathbf{A} = \nabla^2 J(\mathbf{x}^*)$ is the Hessian matrix of J at $\mathbf{x}^*$. A necessary condition for $\mathbf{x}^*$ to be a local minimum of problem (1) is that the Hessian $\nabla^2 L(\mathbf{x}^*)$ is positive semidefinite [45]; in the present case of affine constraints, also the Hessian $\nabla^2 J(\mathbf{x}^*)$ must be positive semidefinite. Hence, for $\mathbf{x} \in U$,

$$\nabla J(\mathbf{x}) = \nabla J(\mathbf{x}^*) + \mathbf{A}\mathbf{x}' \Rightarrow \frac{\partial J}{\partial x_i} = \left(\frac{\partial J}{\partial x_i}\right)_{\mathbf{x}^*} + \sum_{j=1}^{Q} a_{ij} x'_j \tag{17}$$

where $\mathbf{A} = [a_{ij}]$ is a positive semidefinite matrix. Using (15) and (17) we can write (we assume $\tau = 1$ without loss of generality):

$$\begin{aligned} \dot{z}'_i = \dot{z}_i &= -\tfrac{\partial J}{\partial x_i} - \alpha_i + x_i - z_i \\ &= -\left(\frac{\partial J}{\partial x_i}\right)_{\mathbf{x}^*} - \sum_{j=1}^{Q} a_{ij} x'_j - \alpha'_i - \alpha_i^* + x'_i - z'_i + x_i^* - z_i^* \end{aligned} \tag{18a}$$

$$\dot{\alpha}'_j = \dot{\alpha}_j = \sum_{r=1}^{m} w'_{rj} + \sum_{r=1}^{m} w^*_{rj} - 1, \quad j = 1, \ldots, p \tag{18b}$$

From equilibrium conditions (13) it follows:

$$-\left(\frac{\partial J}{\partial x_i}\right)_{x^*} - \alpha_i^* + x_i^* - z_i^* = 0$$

$$\sum_{r=1}^{m} w^*_{rj} - 1 = 0$$

Hence

$$\dot{z}'_i = -\sum_{j=1}^{Q} a_{ij} x'_j - \alpha'_i + x'_i - z'_i, \quad i = 1, \ldots, Q \tag{19a}$$

$$\dot{\alpha}'_j = \sum_{r=1}^{m} w'_{rj}, \quad j = 1, \ldots, p \tag{19b}$$

Consider the following candidate Liapunov function:

$$V(\mathbf{z}, \boldsymbol{\alpha}) = \sum_{i=1}^{Q} \int_0^{z'_i} x'_i(\xi) d\xi + \frac{1}{2} \sum_{j=1}^{p} \alpha'^2_j \tag{20}$$

This kind of function has already been used in the stability analysis of some neural network models for optimization [8,17]. V is nonnegative as a consequence of (16) and $V(\mathbf{z}^*, \alpha^*) = 0$. Taking the time derivative and using (19) we have:

$$\begin{aligned} \dot{V}(\mathbf{z}, \boldsymbol{\alpha}) &= \sum_{i=1}^{Q} \frac{\partial V}{\partial z'_i} \dot{z}'_i + \sum_{j=1}^{p} \frac{\partial V}{\partial \alpha'_j} \dot{\alpha}'_j \\ &= \sum_{i=1}^{Q} x'_i \left[-\sum_{j=1}^{Q} a_{ij} x'_j - \alpha'_i + x'_i - z'_i \right] \\ &\quad + \sum_{j=1}^{p} \alpha'_j \left(\sum_{r=1}^{m} w'_{rj} \right) \end{aligned} \tag{21}$$

Since $\alpha_i = 0$ if x_i corresponds to an element of **H**, it is:

$$\sum_{i=1}^{Q} \alpha'_i x'_i = \sum_{j=1}^{p} \sum_{r=1}^{m} \alpha'_j w'_{rj}$$

Hence

$$\dot{V}(\mathbf{z}, \boldsymbol{\alpha}) = -\sum_{i=1}^{Q} \sum_{j=1}^{Q} a_{ij} x'_i x'_j + \sum_{i=1}^{Q} x'_i (x'_i - z'_i) \tag{22}$$

Taking into account (16) and the positive semidefinite character of **A**, it follows $\dot{V} \leq 0$ for $\forall \mathbf{x} \in U$. Thus the equilibrium point $(\mathbf{z}^*, \alpha^*)$ is stable. ▯

Corollary. *If $\nabla^2 J(\mathbf{x}^*)$ is positive definite, the equilibrium point $(\mathbf{z}^*, \alpha^*)$ is asymptotically stable.*

Proof. If the Hessian $\nabla^2 J(\mathbf{x}^*)$ is positive definite the Hessian $\nabla^2 L(\mathbf{x}^*)$ is positive definite as well, then $\mathbf{x}^*$ is a strict local solution [45]; from (22) it follows $\dot{V} = 0$ if and only if, $\forall i$, $x'_i = 0$. Thus the equilibrium point $(\mathbf{z}^*, \alpha^*)$ is asymptotically stable. ▯

According to Theorem 2 and its corollary, we can say that the neural network can find every local solution of the NMF problem (1), provided that the initial state is inside the corresponding domain of attraction. The presented results do not allow to exclude that the network dynamics could be non-convergent or could stop in a point which is not a local minimum. However, simulation results presented in Section 5, concerning feature extraction and data clustering tasks, indicate a convergent behaviour to near optimal solutions in all cases.

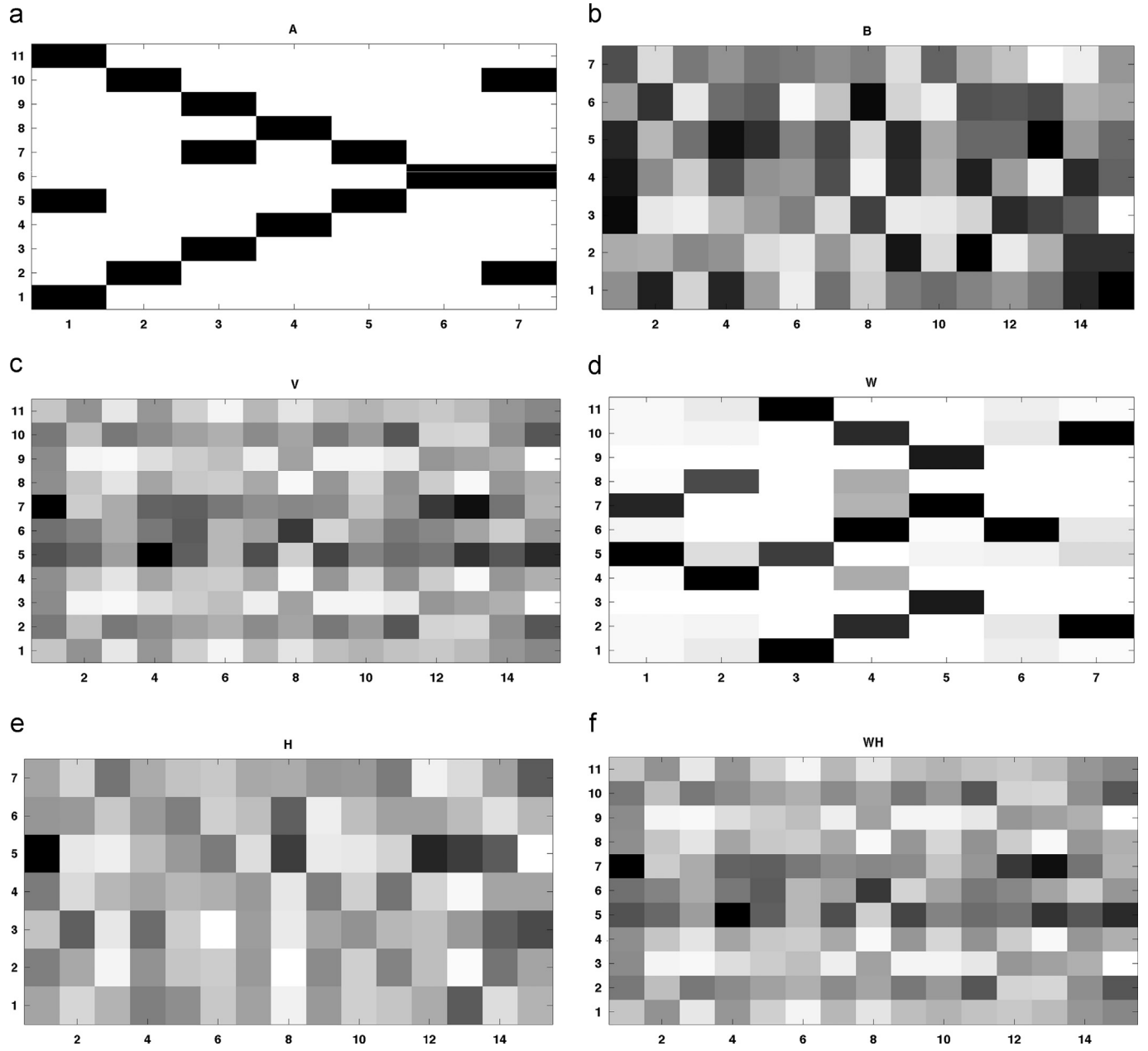


Fig. 3. Synthetic example of feature extraction: (a) matrix **A**, (b) matrix **B**, (c) product **V**=**AB**, (d) and (e) matrices **W** and **H** computed by the network, and (f) product **WH** approximating **V**.

Finally, let us consider the alternating mode of operation, when either **W** or **H** integrators outputs are held fixed. Since the objective functions $J(\mathbf{W})$ and $J(\mathbf{H})$ are quadratic and positive semidefinite, the network in Fig. 2 becomes a particular case of the network for quadratic optmization proposed in [17], where global convergence to an optimal solution has been proved.

5. Experimental results

We have simulated the proposed neural network using ODE23 numerical integration algorithm of Simulink (MathWorks, Inc.), and tested it on various synthetic and real-world datasets to show its effectiveness. As concerns feature extraction, we imposed L_1 normalization on **W** columns and we considered three tests: a synthetic example, the Swimmer dataset and the FERET dataset.

Fig. 4. Four samples from the Swimmer dataset.

First, we generated a data matrix **V**=**AB**, where **A** (shown in Fig. 3a) has seven binary columns and **B** is a random matrix with entries between zero and one (Fig. 3b). In Fig. 3d and e we can see the basis matrix **W** and the activation matrix **H** corresponding to the network steady state. A visual comparison of matrix product **WH** (Fig. 3f) with matrix **V** shows a good agreement. We repeated

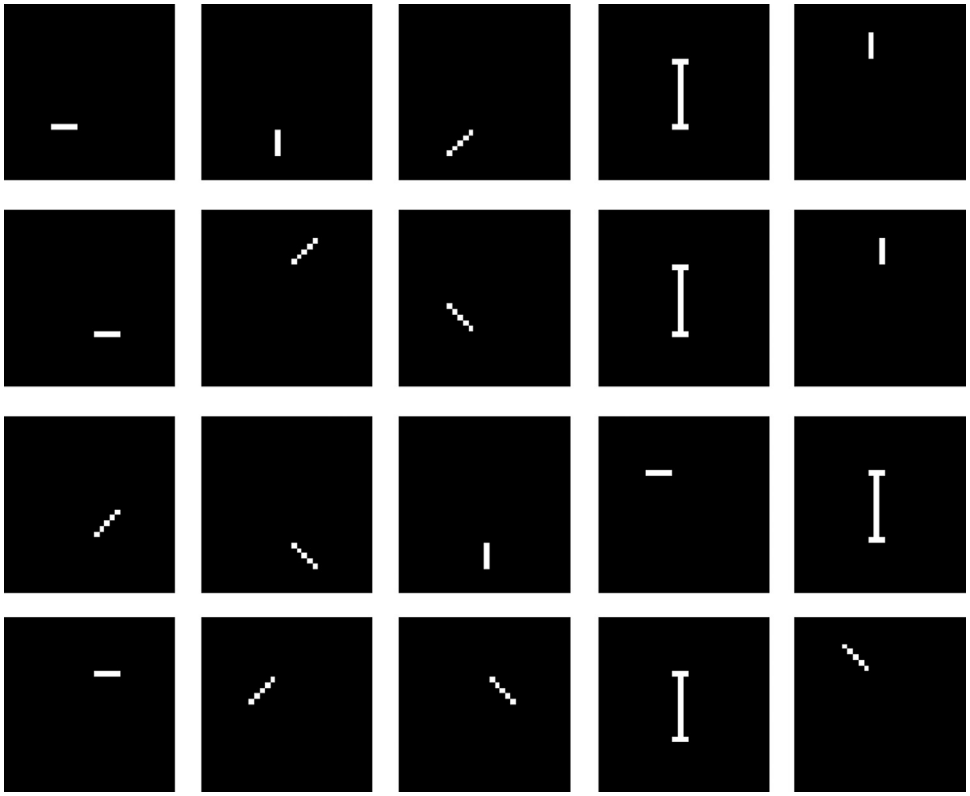


Fig. 5. Basis vectors found by the neural network for the Swimmer dataset.

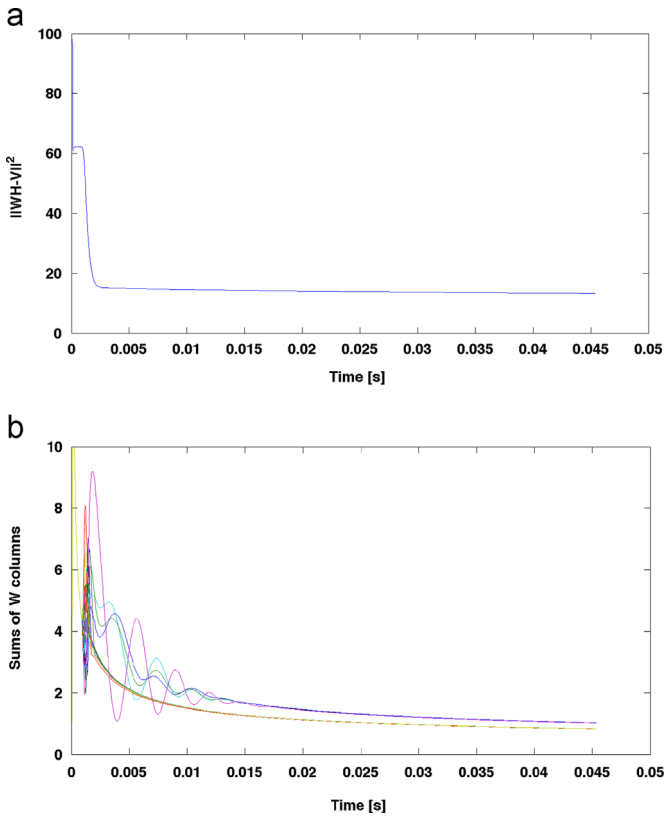


Fig. 6. Dynamic network evolution for the Swimmer dataset ($\tau=1$ ms): (a) reconstruction error and (b) sums of **W** columns.

the simulation 30 times using different initial conditions for **W** and **H**; the average value of $\|\mathbf{WH}-\mathbf{V}\|$ at steady state was 3.29×10^{-4} with standard deviation 1.75×10^{-5}.

The Swimmer dataset[1] [48] consists of 256 images with size 32×32, each of which depicts a figure with one static part (the torso) and four moving parts (the limbs). Each moving part has four different positions. Four of the 256 images are displayed in Fig. 4. The task here is to extract the 16 limb positions and one torso position. Firstly, each image was vectorized and stored in one column of the input matrix **V**. The component number was set to $p=20$. Each column of **W** has the same dimensionality as the input column vectors and thus can be displayed as base images, as shown in Fig. 5. We found that the network can correctly extract all the 17 desired features with three duplicates of the torso (Fig. 5). In Fig. 6 the dynamic network evolution is displayed assuming $\tau=1$ ms. In Fig. 6a, the reconstruction error $\|\mathbf{WH}-\mathbf{V}\|^2$ is shown while the 20 sums of column entries of **W** are shown in Fig. 6b. The reconstruction error converges to a minimum after 2τ while the sums of column entries converge to one after 50τ. The FERET face dataset[2] [49] consists of the inner part of 2409 faces with size 32×32. We normalized the images by dividing the pixel values by their maximal value 255. The component number was chosen as $p=55$. Fig. 7 shows the resulting base images, which demonstrates high sparseness in the factorizing matrix **W**, which captures nearly all facial parts. The result is similar to others reported in the literature (see e.g. [50]).

As concerns clustering, we explored two approaches: (1) we construct the input matrix **V** by using each data vector as a column; we force L_1 normalization of **H** rows; after convergence of neural network dynamics, the index of the maximal value in each column of **H** indicates the cluster membership of the corresponding input vector; (2) as before but with L_1 normalization of **W** columns. After convergence, the **H** columns represent

[1] http://www.stanford.edu/~vcs/Papers.html.
[2] http://www.nist.gov/itl/iad/ig/colorferet.cfm.

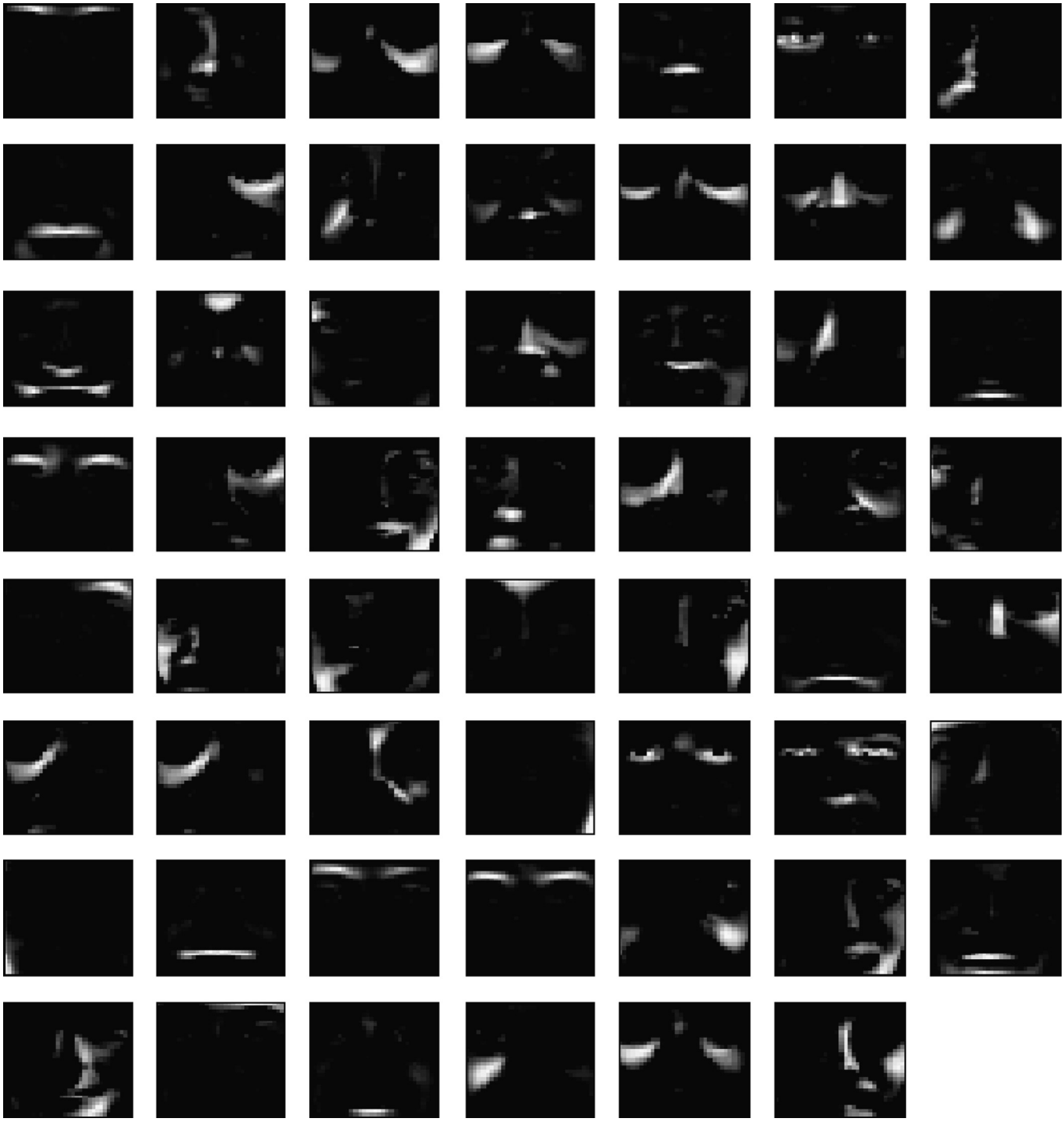


Fig. 7. Basis vectors found by the neural network for the FERET dataset.

a new coding of data vectors which are then clustered using standard *k*-means.

To evaluate clustering results, we have adopted a widely used measurement, called *accuracy or purity*, which is defined as follows:

$$accuracy = \frac{1}{N}\sum_{k=1}^{p}\max_{i=1,\ldots,p} n_k^i \tag{23}$$

where p is the number of clusters (the rank of NMF decomposition), n_k^i is the number of samples in the cluster k that belongs to original class i, and N is the total number of data. Larger accuracy indicates better clustering results, and value one indicates total agreement to the ground truth.

We selected some commonly used datasets from the University of California at Irvine (UCI) repository[3] (iris, glass, ecoli, wine and digit); as concerns the optical handwritten digit database we used a subset containing "0", "2", "4" and "6". In addition, we considered the ORL database of faces taken at the AT&T laboratory.[4] It consists of 400 grey-scale images of 40 subjects taken at

[3] www.ics.uci.edu/~mlearn.

[4] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

Table 1
Clustering results (accuracy).

Dataset	Classes	L_1 norm. of H rows	L_1 norm. of W columns + k-means	*k*-Means	NMF	PNMF [40]
iris	3	**0.98 ± 0.00**	0.97 ± 0.00	0.84 ± 0.10	0.75 ± 0.05	0.97 ± 0.01
glass	6	0.80 ± 0.04	**0.89 ± 0.01**	0.87 ± 0.00	0.62 ± 0.04	–
ecoli	8	0.76 ± 0.01	**0.85 ± 0.01**	0.82 ± 0.02	0.68 ± 0.02	–
wine	3	0.66 ± 0.00	**0.70 ± 0.00**	0.69 ± 0.01	0.57 ± 0.08	–
digit	4	**0.98 ± 0.00**	**0.98 ± 0.00**	0.92 ± 0.11	0.93 ± 0.08	**0.98 ± 0.00**
ORL	40	**0.72 ± 0.02**	0.71 ± 0.01	0.69 ± 0.01	0.48 ± 0.01	**0.72 ± 0.03**

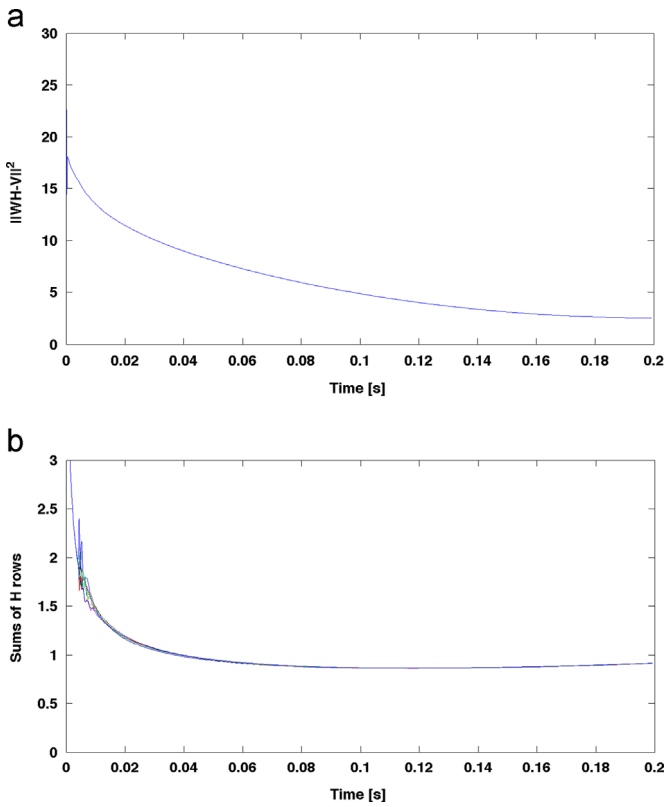


Fig. 8. Experiment with ecoli dataset (τ=1 ms): (a) recontruction error and (b) sums of **H** rows.

different times, with varying facial expressions, lighting conditions and facial details (glasses/no glasses).

For each dataset, the neural network was simulated 30 times with different random seeds for **W** initialization (**H** is initially zeroed). For comparison, we considered standard *k*-means (Lloyd's algorithm) and original NMF (multiplicative update rules [27]). Table 1 shows the mean and standard deviation of the accuracies. Overall, the network solutions correspond to good clustering performance. In particular, we obtained the best result with the iris and ORL datasets using L_1 normalization of **H** rows. As concerns three datasets (glass, ecoli and wine) the best results have been obtained through L_1 normalization of **W** columns, followed by *k*-means on **H** columns. Finally, using the digit dataset the result is the same with both approaches. For sake of comparison with different NMF algorithms, in Table 1 we have included the purity results of projective NMF (PNMF) for iris, digit and ORL datasets [40].

To better understand the network dynamics, in Fig. 8 the transient behaviour is shown for the error $\|\mathbf{WH} \leq \mathbf{V}\|^2$ and the sums of **H** rows in the experiment with ecoli dataset.

6. Conclusions

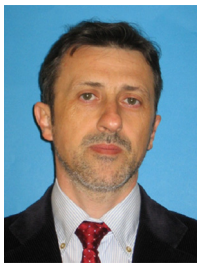
In this paper a neural network model for sparse NMF has been proposed. It is obtained by exploiting a partial dual Lagrangian approach. A main feature of the proposed network is the automatic computation of regularization parameters, trading off reconstruction accuracy and sparsity constraints. As a consequence, experimental tuning or validation are required only to set up the number of basis vectors (the rank of the decomposition). Locally optimal solutions correspond to as many stable equilibrium points of the neural network dynamics. This result has been proven using a Liapunov function method. The derivation presented in the paper proves only local stability; however, extensive simulation results confirm the robust behavior of the proposed network and the accuracy of steady state solutions. In particular, clustering results are consistent with the best state of the art algorithms. Finally, the proposed approach can be easily developed to take into account different affine constraints on basis or activation matrices, in order to enforce different characteristics of computed solution.

References

[1] I.B. Pyne, Linear programming on an electronic analogue computer, Trans. Am. Inst. Electr. Eng. 75 (1956) 139–143.
[2] J.B. Dennis, Mathematical Programming and Electrical Networks, Chapman & Hall, London, 1959.
[3] L.O. Chua, G.-N. Lin, Nlinear programming without computation, IEEE Trans. Circuits Syst. CAS-31 (1984) 182–188.
[4] G. Wilson, Quadratic programming analogs, IEEE Trans. Circuits Syst. CAS-33 (1986) 907–911.
[5] M.P. Kennedy, L.O. Chua, Neural networks for nonlinear programming, IEEE Trans. Circuits Syst. CAS-35 (1988) 554–562.
[6] S. Zhang, A.G. Constantinides, Lagrange programming neural networks, IEEE Trans. Circuits Syst. Part II 39 (7) (1992) 441–452.
[7] A. Bouzerdoum, T.R. Pattison, Neural network for quadratic optimization with bound constraints, IEEE Trans. Neural Networks 4 (2) (1993) 293–304.
[8] M. Forti, A. Tesi, New conditions for global stability of neural networks with applications to linear and quadratic programming problems, IEEE Trans. Circuits Syst.—Part I 42 (7) (1995) 354–366.
[9] Y. Xia, A new neural network for solving linear and quadratic programming problems, IEEE Trans. Neural Networks 7 (6) (1996) 1544–1547.
[10] Y.-H. Chen, S.-C. Fang, Solving convex programming problems with equality constraints by neural networks, Comput. Math. Appl. 36 (7) (1998) 41–68.
[11] E.K.P. Chong, S Hui, S.H Zak, An analysis of a class of neural networks for solving linear programming problems, IEEE. Trans. Automat. Control 44 (11) (1999).
[12] Y. Leung, K.-Z. Chen, Y.-C. Jiao, X.-B. Gao, K.S. Leung, A new gradient-based neural network for solving linear and quadratic programming problems, IEEE Trans. Neural Networks 12 (5) (2001) 1074–1083.
[13] Y. Xia, G. Feng, An improved neural network for convex quadratic optimization with application to real-time beamforming, Neurocomputing 64 (2005) 359–374.
[14] R. Perfetti, E. Ricci, Analog neural network for support vector machine learning, IEEE Trans. Neural Networks 17 (4) (2006) 1085–1091.
[15] H. Ghasabi-Oskoei, N. Mahdavi-Amiri, An efficient simplified neural network for solving linear and quadratic programming problems, Appl. Math. Comput. 175 (2006) 452–464.
[16] L. Zou, L. Zhang, A log-sigmoid lagrangian neural network for solving non-linear programming, in: Proceedings of 8th IEEE ACIS, pp. 427–431, 2007.
[17] G. Costantini, R. Perfetti, M. Todisco, Quasi-Lagrangian neural network for convex quadratic optimization, IEEE Trans. Neural Networks 19 (10) (2008) 1804–1809.

[18] P.-M. Lam, C.S. Leung, J. Sum, A.G. Constantinides, Lagrange programming neural networks for compressive sampling, in: Proceedings of the 17th International Conference on Neural Information Processing: Models and Applications ICONIP'10, Springer-Verlag Berlin, Heidelberg, 2010, pp. 177–184.
[19] X. Hu, C. Sun, B. Zhang, Design of recurrent neural networks for solving constrained least absolute deviation problems, IEEE Trans. Neural Networks 21 (7) (2010) 1073–1086.
[20] Y. Zhang, Y. Yang, G. Ruan, Performance analysis of gradient neural network exploited for online time-varying quadratic minimization and equality-constrained quadratic programming, Neurocomputing 74 (2011) 1710–1719.
[21] M. Mohatram, P. Tewari, N. Latanath, Economic load flow using Lagrange neural network, in: Proceedings of IEEE SIECPC, April 2011, pp. 1–7.
[22] S.K. Bisoi, G. Devi, A. Rat, Neural networks for nonlinear fractional programming, Int. J. Sci. Eng. Res. 2 (12) (2011) 1–5.
[23] C.S. Leung, J. Sum, H.C. So, A.G. Constantinides, F.K.W. Chan, Lagrange programming neural networks for time-of-arrival-based source localization, Neural Comput. Appl. 24 (1) (2014) 109–116 (Springer London). http://link.springer.com/journal/521.
[24] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, R.J. Plemmons, Algorithms and applications for approximate nonnegative matrix factorization, Comput. Stat. Data Anal. 52 (2006) 155–173.
[25] P. Paatero, U. Tapper, Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values, Environmetrics 5 (1994) 111–126.
[26] D.D. Lee, H.S. Seung, Learning the parts of objects by nonnegative matrix factorization, Nature 401 (1999) 788–791.
[27] D.D. Lee, H.S. Seung, Algorithms for nonnegative matrix factorization, Adv. Neural Inf. Process. Syst. 13 (2001) 556–562.
[28] V. Pauca, F. Shahnaz, M. Berry, R. Plemmons, Text mining using non-negative matrix factorizations, in: Proceedings of the Fourth SIAM International Conference on Data Mining, April 22–24, Lake BuenaVista, FL, 2004.
[29] V. Pauca, F. Shahnaz, M. Berry, R. Plemmons., Document clustering using nonnegative matrix factorization, Inf. Process. Manag. 42 (2) (2006) 373–386.
[30] W. Xu, X. Liu, Y. Gong, Document clustering based on non-negative matrix factorization, in: Procedings of ACM Conference on Research and development in IR(SIGIR), Toronto, Canada, 2003, pp. 267–273.
[31] J.G. Nagy, Z. Strakos, Enforcing nonnegativity in image reconstruction algorithms, Math. Model. Estim. Imaging 4121 (2000) 182–190.
[32] C. Thurau, V. Hlaváč, Recognizing human actions by their pose, in: D. Cremers, et al., (Eds.), Visual Motion Analysis, Lecture Notes in Computer Science, vol. 5604, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 169–192.
[33] A. D'Avella, M.C. Tresch, Modularity in the motor system: decomposition of muscle patterns as combinations of time-varying synergies, in: Advances in Neural Information Processing Systems, vol. 14, MIT Press Cambridge MA, 2002, pp. 141–148.
[34] H. Lee, S. Choi, Group nonnegative matrix factorization for EEG classification, J. Mach. Learn. Res. Workshop Conf. Proc. 5 (2009) 320–327.
[35] P. Smaragdis, J.C. Brown, Non-negative matrix factorization for polyphonic music transcription, in: Proceedings of IEEE Workshop of Applications of Signal Processing to Audio and Acoustics, 2003, pp. 177–180.
[36] W. Wang, Y. Luo, J.A. Chambers, S. Sanei, Note onset detection via nonnegative factorization of magnitude spectrum, EURASIP J. Adv. Signal Process. (2008), Article ID 231367.
[37] C. Ding, X. He, H.D. Simon, On the equivalence of nonnegative matrix factorization and spectral clustering, in: Proceedings of the SIAM International Conference on Data Mining (SDM), 2005, pp. 606–610.
[38] C. Ding, T. Li, W. Peng, On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing, in: Computational Statistics & Data Analysis archive, vol. 52 (8), April, ISSN:0167-9473, 2008.
[39] A. Cichocki, R. Zdunek, A.H. Phan, S. Amari, Nonnegative Matrix and Tensor Factorizations, Wiley, Chichester, England, 2008.
[40] Z. Yang, E. Oja, Linear and nonlinear projective nonnegative matrix factorization, IEEE Trans. Neural Networks 21 (5) (2010) 734–749.
[41] Z. He, S. Xie, R. Zdunek, G. Zhou, A. Cichocki, Symmetric nonnegative matrix factorization: algorithms and applications to probabilistic clustering, IEEE Trans. Neural Networks 22 (12) (2011) 2117–2131.
[42] R. Badeau, N. Bertin, E. Vincent, Stability analysis of multiplicative update algorithms and application to nonnegative matrix factorization, IEEE Trans. Neural Networks 21 (12) (2010) 1869–1881.
[43] P. Hoyer, Non-negative matrix factorization with sparseness constraints, J. Mach. Learn. Res. 5 (2004) 1457–1469.
[44] A. Pascual-Montano, J.M. Carazo, K. Kochi, D. Lehmann, R.D. Pascual-Marqui, Nonsmooth nonnegative matrix factorization (nsNMF), IEEE Trans. Pattern Anal. Mach. Intell. 28 (3) (2006) 403–415.
[45] R. Fletcher, Practical Methods of Optimization, 2nd ed., Wiley, Chichester, 1987.
[46] M.W. Hirsch, S. Smale, Differential Equations, Dynamical Systems and Linear Algebra, Academic Press, San Diego, 1974.
[47] J. La Salle, The Stability of Dynamical Systems, SIAM, Philadelphia, 1976.
[48] D. Donoho, V. Stodden, When does non-negative matrix factorization give a correct decomposition into parts? 16 (2003) 1141–1148Adv. Neural Inf. Process. Syst. 16 (2003) 1141–1148.
[49] P.J. Phillips, H. Moon, S.A. Rizvi, P.J. Rauss, The FERET evaluation methodology for face recognition algorithms, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 1090–1104.
[50] Z. Yang, Z. Zhu, E. Oja, Automatic rank determination in projective nonnegative matrix factorization, in: V. Vigneron, et al., (Eds.), LVA/ICA 2010, Lecture Notes in Computer Science, vol. 6365, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 514–521.

Giovanni Costantini received the electronic engineering Laurea degree from the University of Rome "La Sapienza," Italy, the Ph.D. degree in telecommunication and microelectronics from the University of Rome "Tor Vergata," Italy, and the post-graduate Master degree in Sound Engineering in 1991, 1999 and 2006 respectively. He also graduated in piano and electronic music from the Music Conservatory, Italy. Currently, he is an Assistant Professor at the University of Rome, "Tor Vergata." His primary research interests are in the fields of neural networks, pattern recognition, algorithms and systems for audio and musical signal processing.



Renzo Perfetti received the Laurea degree with honors in electronics engineering from the University of Ancona, Italy, in 1982, and the Ph.D. degree in information and communication engineering from the University of Rome "La Sapienza" in 1992. From 1983 to 1987 he was with the radar division of Selenia, in Rome, where he was interested in radar systems design and simulation. From 1987 to 1992 he was with the radiocommunication division of Fondazione U. Bordoni in Rome. In 1992 he joined the Department of Electronic and Information Engineering of the University of Perugia, Italy, where he is currently a Full Professor of electrical engineering. His research interests include artificial neural networks, pattern recognition, machine learning, audio and biomedical signal processing.

Massimiliano Todisco received a Laurea degree in Physics from the University of Rome "La Sapienza," Italy, a post-graduate Master degree in Sound Engineering and a Ph.D. degree in Sensorial and Learning Systems Engineering from the University of Rome "Tor Vergata," Italy. Currently, he is a Research Assistant at the University of Rome, "Tor Vergata." His research interest focus on the areas of artificial intelligence, such as machine learning and pattern recognition, circuits and algorithms for signal analysis, processing, and synthesis, particularly with regard to images, biosignals and audio signals, speech and music.