Max-Margin Non-Negative Matrix Factorization

B. G. Vijay Kumar, Irene Kotsia, and Ioannis Patras

School of Electronic Engineering and Computer Science
Queen Mary, University of London
London, E1 4NS

Abstract

In this paper we introduce a supervised, maximum margin framework for linear and non-linear Non-negative Matrix Factorization. By contrast to existing methods in which the matrix factorization phase (i.e. the feature extraction phase) and the classification phase are separated, we incorporate the maximum margin classification constraints within the NMF formulation. This results to a non-convex constrained optimization problem with respect to the bases and the separating hyperplane, which we solve following a block coordinate descent iterative optimization procedure. At each iteration a set of convex (constrained quadratic or Support Vector Machine-type) sub-problems are solved with respect to subsets of the unknown variables. By doing so, we obtain a bases matrix that maximizes the margin of the classifier in the low dimensional space (in the linear case) or in the high dimensional feature space (in the nonlinear case). The proposed algorithms are evaluated on several computer vision problems such as pedestrian detection, image retrieval, facial expression recognition and action recognition where they are shown to consistently outperform schemes that extract features using bases that are learned using semi-NMF and classify them using an SVM classifier.

Keywords: Non negative matrix factorization, Supervised feature extraction, Semi-NMF, Max-Margin classifier

1. Introduction

The Non-Negative Matrix Factorization (NMF) algorithm is one of the most popular Machine Learning techniques for dimensionality reduction. It has been widely used in several computer vision applications such as image
retrieval, face and gesture recognition, object detection and action recognition. NMF decomposes the data matrix into non-subtractive combinations of non-negative bases [1]. Its ability to produce parts-based representations has been theoretically justified and experimentally demonstrated in [2]. By contrast, other dimensionality reduction methods, such as the Principal Component Analysis (PCA) [3] result in bases and projection coefficients that can take either positive or negative values.

NMF was initially proposed in [2, 4]. In both approaches, the bases and coefficient matrices were obtained by minimizing the reconstruction error, that is the discrepancy between the approximation obtained by the matrix factorization algorithm and the original data. The reconstruction error was quantified either using the Kullback-Leibler divergence [2] or the least squares error [5]. In [6] the authors proposed an efficient implementation that uses a set of multiplicative update rules that were derived from the optimization of an upper bound of the cost function. In [7] the authors showed that the minimization of the upper bound indeed reduces the cost function but does not guarantee the convergence of the algorithm to the stationary point of the original optimization problem. In [8] the authors proposed two projected gradient-based methods for NMF that exhibited strong optimization properties. Motivated by the fact that the multiplicative update rules for computing the factor matrices converge slowly and aiming at reducing expensive NMF update steps, a few matrix initialization techniques that ensured rapid error reduction rate and faster convergence were proposed in [9].

Although NMF usually results in a part-based representation, its various parts are not always well localized. In order to obtain a better localized (sparse) representation, local constraints were imposed along with the non-negativity constraints [1, 10]. Several other algorithms aiming to achieve sparsity with tunable parameters were also developed [11, 12, 13]. In [11, 12], sparseness constraints were imposed on the elements of the coefficient matrix and a parameter was used to control the trade-off between the sparseness and the accuracy of the reconstruction. Such methods have an implicit control over the degree of sparseness. The approaches presented in [13] impose explicit sparseness constraints on both the base and coefficient matrices, allowing in that way an explicit control on the degree of sparseness.

The fact that NMF leads to a low rank approximation of the data makes it suitable for subspace learning, that is for embedding high dimensional data into a low dimensional subspace. In this context, it has been extensively used for facial analysis including detection [10], recognition [1], verification
[14] and expression recognition [15, 16]. Several other applications of NMF in Computer Vision include pose estimation [17], action recognition [18, 19], object recognition [20], subspace learning [21] and clustering [22].

In [17], NMF bases and coefficients were learned using a set of features extracted from clutter-free images containing objects. In [19], the NMF coefficients were extracted using appearance features and motion vectors. These coefficients were subsequently used to train a cascaded Linear Discriminant Analysis (LDA)-based classifier. The method presented in [18] followed an approach similar to [17] for the detection of humans in image sequences, where NMF was employed to learn a set of pose primitives. In [20], two approaches were followed in order to improve the recognition rate using features extracted by NMF. The first used a Riemannian metric for the learned feature vectors instead of the classic Euclidean distance, while the second orthonormalized the NMF bases and then used the features projected onto these bases. The authors in [21] introduced the Graph Regularized NMF (GNMF) that modeled the data subspace as a sub-manifold embedded in an ambient space. By learning NMF on such a manifold, GNMF showed better discriminative ability when compared to NMF that only considers the Euclidean space. Semi-NMF was introduced in [22] for clustering by relaxing the non-negativity constraints on the bases matrix. This lead to a bases matrix that contained the cluster centers and non-negative coefficients that can be interpreted as cluster indicators. A non linear extension to NMF, the so-called Kernel NMF (KNMF), was presented in [23].

Only few NMF-based works obtain the matrices in a supervised manner, that is, by utilizing the label information of the samples. In [14] the authors introduced discriminative constraints in order to extract bases that correspond to discriminative facial regions for the problem of face recognition. The proposed Discriminant NMF (DNMF) [14] results in bases corresponding to salient facial features such as eyes and mouth, that are useful for discrimination. The authors in [15] proposed the Projected Gradients DNMF (PG-DNMF) algorithm for facial expression recognition which extends DNMF in two major ways. First, projected gradients were used instead of multiplicative update rules in order to guarantee the convergence of the algorithm to a limit point that is also a stationary point of the original optimization problem. Second, discriminant analysis was employed on the classification features and not on the reconstructed data. In both of the above mentioned approaches the discriminant constraints were introduced in the cost function, yielding in this way discriminative bases. However, the introduced
constraints were tailored for a rather simplistic LDA-based classifier. Here, we propose a method in which the acquired projections are chosen so that they maximize the discriminative ability of a Support Vector Machine (SVM) classifier, a fact that results in higher classification performance as will be demonstrated in the section of experimental results.

In this paper, we first introduce soft max-margin constraints to the objective function of NMF in order to obtain a bases matrix that will enable us to extract features that maximize the classification margin. More precisely, in the proposed scheme we optimize a weighted combination of the reconstruction error term, that is used in the typical NMF formulations, and the cost function, that is used in typical SVM formulations, under SVM-type linear inequality constraints. The optimization is performed with respect to the unknown bases, the projection coefficients and the parameters of the separating hyperplane and is solved in an iterative manner, where at each iteration we solve only for a subset of the unknown parameters while keeping the others fixed. The resulting sub-optimization problems are either instances of Quadratic programming with linear inequality constraints or classical SVM-type problems. We proceed with extending the above framework to include the nonlinear version of NMF (KNMF) [23]. In that way we are able to obtain a bases matrix that maximizes the classification margin of the classifier in the reconstructed high dimensional feature space. The proposed method is applied to publicly available databases (the INRIA pedestrian, the BU-3DFE, the KTH action and the Mediamill datasets) where we demonstrate that it consistently outperforms SVM classification schemes that use features extracted using Semi-NMF [22], KNMF [23] and DNMF [14].

Summarizing, the main contributions of this paper are

- A max-margin framework for Semi Non-negative Matrix Factorization (MNMF) and Kernel NMF (KMNMF). The proposed framework incorporates the maximum margin constraints within the Semi-NMF formulation, in order to jointly find both the factorization matrices and the separating SVM hyperplane with respect to the projections along the bases that are obtained by the factorization.

- An optimization scheme that solves simultaneously for the separating hyperplane of the max-margin classifier and the factorization matrices. The constrained optimization problem of the proposed framework is a non-convex one with respect to the unknown parameters. We propose an iterative procedure where at each iteration we solve a set of convex
(quadratic or SVM-type) subproblems. Each of those convex subproblems results from the original one when we fix some of the unknown parameters.

The rest of the paper is organized as follows. In Section 2.1, we briefly describe the NMF and Semi-NMF algorithms. In Section 2.2, we formulate the proposed max-margin framework for semi-NMF and present an algorithm that solves the corresponding optimization problem. In Section 2.4 we discuss convergence issues of the proposed MNMF algorithm. In Section 3, we introduce the Kernel NMF algorithm. In Section 4 we present experimental results on several Computer Vision and Multimedia problems using publicly available datasets. Finally, in Section 5 we draw some conclusions.

2. Max-Margin Semi-NMF (MNMF)

In this section, we give a brief overview of the semi-NMF algorithm for matrix decomposition and proceed with formulating the proposed maximum margin NMF framework.

2.1. Semi Non-negative Matrix Factorization

Let \( X \in \mathbb{R}^{m \times n} \) represent a non-negative matrix having \( n \) examples in its columns. The NMF algorithm [2] decomposes \( X \) into two non-negative matrices, the bases matrix \( G \in \mathbb{R}^{m \times k} \) and the coefficients matrix \( H \in \mathbb{R}^{k \times n} \) such that \( X \approx GH \). For dimensionality reduction \( k \) is typically chosen to be small \((< \min(m, n))\). The columns of \( G \) can be regarded as the bases vectors and thus each example can be represented as the linear combination of those bases vectors as \( x_i = Gh_i \). Here \( x_i \) and \( h_i \) are the \( i^{th} \) columns of \( X \) and \( H \), respectively.

Semi-NMF [22] relaxes the non-negativity constraints on \( G \) and on the data matrix \( X \). The unknown matrices \( G \) and \( H \) are estimated by minimizing the reconstruction error \( \|X - GH\|_F^2 \) or the Kullback-Leibler divergence \( D(X||GH) \) w.r.t. \( G \) and \( H \)

\[
\text{argmin}_{G,H \geq 0} \|X - GH\|_F^2, \quad \text{or} \\
\text{argmin}_{G,H \geq 0} D(X||GH) 
\]
where \( \| \cdot \| \) denotes the Frobenius norm. From now onwards we will use the notation \( \mathbf{H} \succeq 0 \) to express that the elements of the matrix \( \mathbf{H} \) are non-negative. The above minimization problems are iteratively solved with respect to the matrices \( \mathbf{G} \) and \( \mathbf{H} \) using a set of update rules \([22]\):

Step 1: Update \( \mathbf{G} \) by keeping \( \mathbf{H} \) fixed

\[
\mathbf{G} = \mathbf{XH}^T(\mathbf{HH}^T)^{-1}
\]

Step 2: Update \( \mathbf{H} \) by keeping \( \mathbf{G} \) fixed,

\[
\mathbf{H} = \mathbf{H} \odot \sqrt{\frac{[\mathbf{G}^T\mathbf{X}]^+ + [\mathbf{G}^T\mathbf{G}]^+ \mathbf{H} + [\mathbf{G}^T\mathbf{X}]^-}{[\mathbf{G}^T\mathbf{G}]^+ \mathbf{H} + [\mathbf{G}^T\mathbf{X}]^-}}
\]

where \( \mathbf{M}^+ \) and \( \mathbf{M}^- \) correspond to a positive and a negative part of the matrix \( \mathbf{M} \), respectively, given by

\[
\mathbf{M}_{ik}^+ = \frac{1}{2}(|\mathbf{M}_{ik}| + \mathbf{M}_{ik}), \quad \mathbf{M}_{ik}^- = \frac{1}{2}(|\mathbf{M}_{ik}| - \mathbf{M}_{ik}).
\]

### 2.2. Max-Margin Semi-NMF (MNMF)

As stated above, the NMF algorithm \([2]\) minimizes the cost function defined either as in Eq. 1 or as in Eq. 2, imposing at the same time non-negativity constraints on both \( \mathbf{G} \) and \( \mathbf{H} \). These non-negativity constraints result in a part-based representation of the data. Several NMF variants incorporating discriminant constraints were proposed in \([14, 15, 22]\). The variations were obtained by introducing application specific discriminant constraints to the cost function. Inspired by this, we aim at creating a framework that allows us to find a set of basis vectors that maximizes the margin of an SVM classifier.

Let \( \{\mathbf{x}_i, y_i\}_{i=1}^L \) denote a set of data vectors and their corresponding labels, where \( \mathbf{x}_i \in \mathbb{R}^m, y_i \in \{-1, 1\} \). Our aim is to determine a bases matrix \( \mathbf{G} \) that can be used to extract features that are optimal under a max-margin classification criterion. This is accomplished by imposing constraints on the feature vectors derived from \( \mathbf{G} \). In this work, similarly to \([14, 15, 24]\), the features that are extracted from a data example \( \mathbf{x} \) are given by \( \mathbf{G}^T\mathbf{x} \). That is, they are the projections of the data example \( \mathbf{x} \) on the bases vectors stored
in $G$. Then, the optimization problem is given by

$$\text{argmin}_{G,H,w,b,\xi} \lambda \|X - GH\|_F^2 + \frac{1}{2} w^T w + C \sum_{i=1}^{L} \xi_i$$

s.t. $y_i(w^T G^T x_i + b) \geq 1 - \xi_i$

$$\xi_i \geq 0, \quad 1 \leq i \leq L, \quad H \succeq 0,$$

where $\xi = \{\xi_1, \ldots, \xi_i, \ldots, \xi_L\}$ is the slack variable vector, $\lambda$ is a scalar that controls the relative importance for the NMF cost and $C$ a scalar that controls the relative importance of the penalty imposed for the training examples that are either too close to the separating hyperplane or misclassified.

The first term of the above optimization problem ($\lambda \|X - GH\|_F^2$) is a classical NMF-type reconstruction error, while the second ($\frac{1}{2} w^T w$) and the third term ($C \sum_{i=1}^{L} \xi_i$) is an SVM-type cost. Notice that the slack variables $\xi_i$ in third term control the misclassification errors. Notice also that the inequality constraints $y_i(w^T G^T x_i + b) \geq 1 - \xi_i$ that involve the slack variables, depend on both the parameters $w$ of the classifier and on the data projection matrix $G^T$ that is used to extract the features $G^T x$. In this way, we jointly optimize with respect to both the Semi-NMF data projections and the maximum margin classifier.

Notice, that classical NMF-based algorithms use $G^\dagger = (G^T G)^{-1} G^T$ as the projection matrix, that is the features that are extracted for a data example $x$ are given by $\hat{x} = G^\dagger x$. By contrast we follow [14, 15, 24] and use $G^T$ as the projection matrix. Note, that both NMF and our MNMF find a bases matrix $G$ and express an arbitrary $x$ as a (non negative) linear combination (with coefficients $h$) of the column vectors of $G$. Using $G^\dagger$ as the projection matrix results in features that are the (non-negative) coefficients $h$ of the linear combination that minimizes the MSE. The projection $G^T x$ that we propose, uses as features the projection of the vector $x$ on the bases vectors. Both choices are equally valid. Ours is easier to work with and results to a formulation that is quadratic with respect to $G$.

In order to optimize the cost function in Eq. 5, we follow an iterative optimization procedure. More precisely, at each iteration we solve for subsets of the unknown parameters $G$, $H$ and $w, b, \xi_i$ by keeping the remaining parameters fixed. The optimization procedure is described below, and the steps followed in the proposed max-margin Semi-NMF framework are summarized in Algorithm 1.
Algorithm 1: Algorithm for MNMF

**input**: \( X, G_{\text{init}}, H_{\text{init}}, \text{MAXITER}, \lambda, C \)

**output**: \( G, H, w, b \)

begin
  \[ G = G_{\text{init}}; \]
  \[ H = H_{\text{init}}; \]
  repeat
    \[ S1 : \text{Solve for } \alpha \text{ in Eq. 10} \]
    \[ S2 : \text{Compute } G \text{ using Eq. 8} \]
    \[ S3 : \text{Find the classifier parameters, } w, b, \xi_i \text{ for the updated } G \]
    \[ S4 : \text{foreach column } h_i \text{ of } H \text{ do} \]
      \[ \text{Calculate } \gamma \text{ using Eq. 19} \]
      \[ \text{Compute } h_i \text{ using Eq. 23} \]
    end
  until \( \text{iter} \leq \text{MAXITER} \) or convergence;
end

**Solve for \( G, \xi \) by keeping \( H, w \) and \( b \) fixed**: Since \( w \) is fixed, the optimization problem in Eq. 5 is simplified as

\[
\arg\min_{G, \xi_i} \lambda \|X - GH\|_F^2 + C \sum_{i=1}^{L} \xi_i \\
\text{s.t. } y_i(w^T G^T x_i + b) \geq 1 - \xi_i \\
\xi_i \geq 0, \ 1 \leq i \leq L.
\]

The above formulation can be derived from Eq. 5 if the second term is omitted. It is a weighted combination of the reconstruction error (1st term) and soft constraints/penalizations for the examples that do not maintain the appropriate distance (margin) from the separating hyperplane (3rd term). In this step we solve for a projection matrix that projects the input examples to a lower dimensional feature space. Since it is necessary to reduce the misclassification error at each iteration in order to get an optimal projection matrix \( G \), we penalize the misclassified examples by also solving for \( \xi \). This results in a set of bases \( G \) that simultaneously reduce the reconstruction error while ensuring a low misclassification error. We should note that the cost function in Eq. 6 is either a quadratic or linear function that imposes linear inequality.
constraints on the set of unknowns. We proceed to solve it using its dual formulation. The Lagrangian of Eq. 6 is given by
\[ L(G, \xi_i, \alpha_i, \beta_i) = \lambda^T \left[ (X - GH)(X - GH)^T \right] + C \sum_{i=1}^{L} \xi_i - \sum_{i=1}^{L} \alpha_i \left[ y_i (w^T G^T x_i + b) - 1 + \xi_i \right] - \sum_{i=1}^{L} \beta_i \xi_i \] (7)
\[ \alpha_i, \beta_i \geq 0, \quad 1 \leq i \leq L \]
where \( \alpha_i, \beta_i \) are the Lagrangian multipliers. Taking the derivative w.r.t. the primal variables and equating to 0, we get
\[ G = \left( 2XH^T + \sum_{i=1}^{L} \alpha_i y_i x_i w^T \right)(2HH^T)^{-1} \] (8)
\[ \frac{\partial L}{\partial \xi_i} = 0 \quad \Rightarrow \quad 0 \leq \alpha_i \leq \theta, \quad 1 \leq i \leq L \] (9)
where \( \theta = C/\lambda \). Substituting the value of \( G \) in Eq. 7 and simplifying, we get the dual problem
\[ \arg\max_{\alpha} \quad \alpha^T (T_1 - T_2)\alpha + (t_3 - t_4 - t_5 - t_6 + t_7)\alpha \]
\[ \text{s.t.} \quad 0 \leq \alpha_i \leq \theta \] (10)
where

\[ \alpha \in \mathbb{R}^L, \ T_1, T_2 \in \mathbb{R}^{L \times L}, \ t_3, t_4, t_5, t_6, t_7 \in \mathbb{R}^{1 \times L}, \]

\[ T_1 = \left[ \sum_{k=1}^{L} y_i y_j h^T_k B M^T_i M_j B h_k \right]_{ij} \]

\[ T_2 = [y_1 y_2 w^T B M^T_j x_i]_{ij} \]

\[ t_3 = \left[ 4 \sum_{k=1}^{L} y_i h^T_k B H x^T_i M_i B h_k \right]_{1i} \]

\[ t_4 = \left[ 2 \sum_{k=1}^{L} y_i h^T_k B w x^T_i x_k \right]_{1i} \]

\[ t_5 = 2 y_i w^T B H x^T_i x_{1i}, \ t_6 = b [y_i]_{1i} \]

\[ t_7 = [111 \cdots 1]_{1 \times L}, \ B = (2HH^T)^{-1}, \ M_i = x_i w^T, \]

(11)

and \( h_k \) is the \( k^{th} \) column of the matrix \( H \).

The above problem is quadratic in \( \alpha \), thus can be solved by using conventional quadratic programming tools. The estimated \( \alpha \) is then used to compute \( G \) using Eq. 8. The constant term \( \theta \) in Eq. 10 is used as a tuning parameter. Large values of \( \lambda \) (when compared to \( C \)), result in low values of \( \theta \) something that leads to small \( \alpha_i \). This in turn causes the second term in Eq. 8 to disappear making the update rule of \( G \) to be the one used in semi-NMF, as given in Eq. 3. Hence for large values of \( \lambda \), the update rule for \( G \) tends to approach the update rule of semi-NMF, something that is also evident in Eq. 6.

**Solve for \( w, b, \xi \) by keeping \( G \) and \( H \) fixed:** In the previous step we computed the updated basis \( G \). We now proceed in solving for the hyperplane that maximizes the margin of the classifier keeping the bases matrix \( G \) and weights matrix \( H \) fixed. The features are obtained by projecting the data points onto the updated basis matrix \( G \) as calculated in the previous step. Since \( G \) and \( H \) are fixed, the optimization problem in Eq. 5 is simplified to
a form that strongly resembles that of a classical SVM:

\[
\begin{align*}
\text{argmin}_{w,b,\xi} & \frac{1}{2}w^T w + C \sum_{i=1}^{L} \xi_i \\
\text{s.t.} & \quad y_i(w^T G^T x_i + b) \geq 1 - \xi_i \\
& \quad \xi_i \geq 0, \quad 1 \leq i \leq L.
\end{align*}
\]  

(12)

The hyperplane parameters \(w\), and \(b\) and the slack variable vector \(\xi\) are obtained using an off-the-shelf SVM classifier.

**Solve for \(H\) by keeping \(G\), \(w\), \(\xi\) and \(b\) fixed:** Having already acquired the values for \(G, w, b\) and \(\xi\) from the previous steps, we proceed with solving for the weights matrix \(H\) by keeping all the other variables fixed. Since only the reconstruction error term of the optimization problem (Eq. 5) depends on \(H\), the objective function is simplified as

\[
\begin{align*}
\text{argmin}_{H} & \|X - GH\|_F^2, \\
\text{s.t.} & \quad H \succeq 0.
\end{align*}
\]  

(13)

We solve for \(H\) using quadratic programming. The \(i^{th}\) column of \(H\), \(h_i\) contributes only to the \(i^{th}\) data point \(x_i\) and hence the columns of \(H\) can be solved independently of each other. The above optimization problem can be solved using the update equation Eq. 4. Here we adopt an alternative optimization method. In particular, the objective function in Eq. 13 can be rewritten as

\[
\begin{align*}
\text{argmin}_{h_i} & \quad (x_i - Gh_i)^T (x_i - Gh_i), \\
\text{s.t.} & \quad h_i \succeq 0 \quad \forall i.
\end{align*}
\]  

(14)

The Lagrangian of the above cost function is

\[
L(h_i) = (x_i - Gh_i)^T (x_i - Gh_i) - \gamma^T h_i, \quad \gamma > 0
\]  

(15)

where \(\gamma \in \mathbb{R}^k\) is a vector of positive Lagrangian multiplier.

Taking under consideration the KKT conditions [25], we get:

\[
\nabla L(h_i) = 0,
\]  

(16)

\[
\gamma_j h_{ij} = 0 \quad \text{and}
\]  

(17)

\[
\gamma_j \geq 0,
\]  

(18)
where $h_{ij}$ is the $j$-th element of $h$. From Eq. 16 we get:
\[
\nabla L(h_i) = 0 \Rightarrow \nabla F(h_i) - \gamma = 0 \Rightarrow [\nabla F(h_i)]_j = \gamma_j
\]
(19)
and since $\gamma_j h_{ij} = 0 \Rightarrow [\nabla F(h_i)]_j h_{ij} = 0$. Therefore
\[
\nabla F(h_i) = -2G^T x_i + 2G^T G h_i
\]
(20)
\[
= - ([2G^T x_i]^+ - [2G^T x_i]^-) + ([2G^T G h_i]^+ - [2G^T G h_i]^-)
\]
(21)
\[
= - ([2G^T x_i]^+ + [2G^T G h_i]^-) + ([2G^T G h_i]^+ + [2G^T x_i]^-).
\]
(22)
Hence
\[
h_i = h_i \odot \frac{([G^T x_i]^+ + [G^T G h_i]^-) - ([G^T G h_i]^+ + [G^T x_i]^-)}{([G^T G h_i]^+ + [G^T x_i]^-) - ([G^T G h_i]^+ + [G^T x_i]^-)}. 
\]
(23)
The procedure is repeated for all columns of $H$.

During testing, the input test vector $x_{test}$ is projected onto the basis matrix to obtain the feature vector, $f_{test} = G^T x_{test}$. This feature vector is then given as input to the max-margin classifier which predicts the class
\[
\hat{y}_{test} = \text{sign}(w^T f_{test} + b)
\]
where $w, b, G$ are computed during training.

2.3. Experiments on a Synthetic Toy Dataset

In order to provide an insight to the way the proposed MNMF algorithm works, we first conduct experiments on a toy dataset consisting of two classes each of which contains 100 points that are sampled from 50-dimensional Gaussian distributions. For visualization purposes, we restrict the number of bases taken under consideration to be equal to two ($k = 2$). The bases matrix $G$ and the weights matrix $H$ are computed using the Semi-NMF algorithm and the input data points are projected onto the lower dimensional subspace using the acquired $G$. In Fig. 1a we show the projections of the points after applying a common dimensionality reduction technique, namely Principal Component Analysis (PCA) [3]. Fig. 1b depicts the projections of the input datapoints using the bases extracted using Semi-NMF. Fig. 1c and Fig. 1d show the projections of the proposed MNMF algorithm after the first and the sixth iterations, respectively. It is clear that the projections acquired from the proposed MNMF algorithm make the classes more separable.

In order to examine the discriminative power of the features extracted by each of the above mentioned methods we trained an SVM classifier on the acquired projections and report the obtained classification accuracies. When
PCA, Semi-NMF (at convergence, i.e. after 2000 iterations) and the proposed MNMF algorithm (at convergence, i.e. after only 6 iterations) were applied, the accuracies obtained were equal to 96.5%, 97% and 100%, respectively. This verifies the fact that the proposed algorithm updates the bases in such a way that the margin of the classifier in the projected space is maximized, thus achieving lower misclassification error.
2.4. Convergence Issues

In this section, we discuss the convergence of the proposed MNMF framework. The objective function proposed in Eq. 5 is a weighted combination of the NMF cost and the classifier (SVM) cost. We optimize this objective function using a block coordinate descent where at each step we solve a set of convex sub-problems each of which is guaranteed to converge. Therefore, the whole procedure converges to a (local) minimum. A proof of the convergence is provided in Appendix A.

We verified, experimentally, that the proposed optimization procedure does reduce at each step the objective function in Eq. 5 and that the Frobenius norm of the differences (between subsequent iterations) in $G$, $H$ and hyperplane parameter $w$ converge to zero. In order to demonstrate this we use a subset of the KTH dataset consisting of the action classes Run and Walk. The plot in Fig. 2a shows that the objective function decreases and converges after few iterations. The convergence of the parameters $G$, $H$ and $w$ is also evident from Fig 2b, 3a, and 3b, respectively.

![Figure 2: (a) Objective function in Eq. 5 vs Iterations, (b) $\|G_{i+1} - G_i\|^2_F$ vs Iterations](image)

Note that in the proposed framework we solve for $\xi$ in two places, *i.e.* in Eq. 6 and Eq. 12. Alternatively we could for example solve for $\xi$ only in Eq. 6 and substitute the obtained values in Eq. 12. As optimization strategies, both are valid block-coordinate descent optimisations, which at each step reduce the objective function and therefore lead to local minima. However, we notice
that a change in either the projection matrix $G$ or the hyperplane parameters can violate the inequality $\mathbf{w}^T \mathbf{G}^T \mathbf{x}_i + b \geq 1 - \xi_i$. Fixing the slack variables when solving for say $\mathbf{w}$ in Eq. 12 would make the inequality constraints hard, therefore lead to worse solutions of $\mathbf{w}$ (in terms of the objective function). By contrast, if we solve for the penalty term $\xi_i$ in both Eq. 6 and Eq. 12 the constraints are soft in both cases. Fig. 4 shows the plot of $\sum_i \|\xi_i - \xi'_i\|^2$ where $\xi_i$ is obtained by solving for $\xi_i$ in Eq. 6 and $\xi'_i$ is obtained by solving for $\xi_i$ in Eq. 12. As we can see from Fig. 4 the sum of the differences converges to zero, that is, $\xi_i$ and $\xi'_i$ converge to the same values.

3. Max-Margin Kernel NMF (KM-NMF)

In the previous Section we presented our proposed framework assuming that a linear Semi-NMF was used. However, linear NMF algorithms cannot properly capture the non-linear structure that the data may follow. To tackle this problem the authors in [23] proposed the Kernel extension of NMF (KNMF) and showed that it significantly improves the performance over NMF in classification applications. In the following sections, we extend the max-margin framework described in the previous section to include KNMF.
3.1. Overview of KNMF

Let $\Phi$ denote a non-linear transformation that maps data $x \in \mathbb{R}^m$ in the input space to a higher dimensional feature space, i.e. $\Phi : x \in \mathbb{R}^m \rightarrow \Phi(x) \in \mathbb{R}^f$, typically $f \gg m$. Let $\Phi(X) = [\Phi(x_1), \Phi(x_2) \cdots \Phi(x_n)]$ denote the data matrix where each example $\Phi(x_i) \in \mathbb{R}^f$. KNMF decomposes the data matrix as

$$\Phi(X) \approx G_{\Phi}H$$

where the base matrix $G_{\Phi} \in \mathbb{R}^{f \times k}$ contains the basis vectors in the feature space and the coefficients matrix $H \in \mathbb{R}^{k \times n}$ indicates the contribution of each basis vector in the reconstruction of the example. In practice, the computation of $\Phi(X)$ and $G_{\Phi}$ is impractical and thus the kernel trick [26] is employed to efficiently compute the similarities in the feature space,

$$K \approx YH,$$

where $K = \Phi^T(X)\Phi(X)$ is the kernel matrix and $Y = \Phi^T(X)G_{\Phi}$. The coefficient vector $h_{test}$ for a test example $x_{test}$ is given by,

$$h_{test} = Y^\dagger K_{test},$$

where $K_{test} = \Phi^T(X)\Phi(x_{test})$ and $\dagger$ denotes the pseudo-inverse.
3.2. Max-Margin Kernel NMF (KMNMF)

In this section, we formulate our proposed framework by imposing max-margin constraints within KNMF. We aim at finding a set of basis vectors in the feature space, derived using KNMF, that maximizes the margin of an SVM classifier in the reconstructed feature space.

3.2.1. Cost Function for KMNMF

Let \( \{ \Phi(x_i), y_i \}_{i=1}^L \) denote a set of data vectors in the feature space and their corresponding labels, where \( \Phi(x_i) \in \mathbb{R}^f \), \( y_i \in \{-1, 1\} \). The objective is to determine a set of basis vectors acquired using KNMF that can be used to reconstruct the data in the feature space in such a way that they are optimal under a max-margin classification criterion. This is accomplished by imposing constraints on the reconstructed data computed using the bases matrix \( G_\Phi \). Let the reconstructed vector for a data example \( \Phi(x_j) \) be given as \( \Phi(\tilde{x}_j) \approx G_\Phi h_j \) where \( h_j \) is the coefficient vector.

The optimization problem for the proposed criterion is given by

\[
\begin{align*}
\arg\min_{G_\Phi, H, w_\Phi, b, \xi_i} & \quad \lambda \| \Phi(X) - G_\Phi H \|_F^2 + \frac{1}{2} w_\Phi^T w_\Phi + C \sum_{i=1}^L \xi_i \\
\text{s.t.} & \quad y_i (w_\Phi^T G_\Phi h_i + b) \geq 1 - \xi_i \\
& \quad \xi_i \geq 0, \quad 1 \leq i \leq L, \quad H \succeq 0
\end{align*}
\]

where \( \Phi(X) = \{ \Phi(x_i) \}_{i=1}^L \) and \( \lambda \) is the weight factor for the KNMF cost. The first term \( (\lambda \| \Phi(X) - G_\Phi H \|_F^2) \) corresponds to the KNMF reconstruction error, the second term \( (\frac{1}{2} w_\Phi^T w_\Phi) \) to the maximum margin classifier in the reconstructed space and the third term \( (C \sum_{i=1}^L \xi_i) \) is the common term shared by the two previously mentioned terms, used to penalize the misclassified examples with respect to the input projections acquired from KNMF.

The above formulation aims at maximizing the margin of the support vectors while at the same time minimizing the reconstruction and misclassification errors. The classifier is trained on the reconstructed data points \( G_\Phi h \), obtaining in this way the hyperplane parameter \( w_\Phi \in \mathbb{R}^f \). We follow the same procedure described in the previous section and iteratively solve for one of the terms \( G_\Phi, H \) and \( w_\Phi, b, \xi \) while keeping the remaining parameters fixed.

We should note here that since the columns of the bases matrix \( G_\Phi \), the data matrix \( \Phi(X) \) and the SVM hyperplane parameter \( w_\Phi \) lie in the feature space, their explicit computation is not necessary.
Instead we solve explicitly for the parameters of the dual formulations of their corresponding constrained optimization problems and use them in order to calculate quantities in the form of dot products in the feature space. More specifically, when we solve for $G_\phi$, we calculate $G_\phi^T G_\phi$ and $G_\phi^T \Phi(X)$, while when we solve for the max-margin hyperplane $w_\phi$ we calculate $w_\phi^T w_\phi$ and $w_\phi^T \Phi(X)$. For the data kernel matrix $\Phi(X)^T \Phi(X)$ we use the Gaussian kernel [26],

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{\sigma^2}\right). \tag{26}$$

The steps followed in the proposed max-margin KNMF framework are summarized in Algorithm 2.

**Algorithm 2: Algorithm for KNMF**

**input**: $X$, $H_{init}$, $MAXITER$, $\lambda$, $C$, $\sigma$

**output**: $G_\phi^T G_\phi$, $H$, $w_\phi$, $b$

**begin**

$H = H_{init}$;

**repeat**

$S1$: Solve for $\alpha$ in Eq. 31

$S2$: Compute $G_\phi^T G_\phi$ using the values of $\alpha$ as in Eq. 33

$S3$: Compute the kernel matrix $H^T G_\phi^T G_\phi H$

$S4$: Use the computed kernel matrix to find the classifier parameters, $w_\phi$, $b$.

$S5$: **foreach** column $h_i$ of $H$ **do**

 Calculate $\gamma$ using Eq. 38

 Compute $h_i$ using Eq. 39

**end**

**until** $\text{iter} \leq MAXITER$ or convergence;

**end**

Solve for $G_\phi$ and $\xi$ by keeping $H$, $w_\phi$ and $b$ fixed: Since $w_\phi$ remains
fixed, the optimization problem in Eq. 25 is simplified as
\[
\arg\min_{G,\xi_i} \lambda \|\phi(X) - \sum_{i=1}^{L} \xi_i \|_F^2 + C \sum_{i=1}^{L} \xi_i
\]
s.t. \( y_i (w_\phi^T G\phi h_i + b) \geq 1 - \xi_i \)
\( \xi_i \geq 0, \quad 1 \leq i \leq L. \)  

The above formulation is derived from Eq. 25 if the second term is omitted. It is a weighted combination of the reconstruction error caused by the KNMF (1st term) and the soft constraints/penalizations for the examples that do not maintain the appropriate distance (margin) from the separating hyperplane (3rd term). Our aim is therefore to find a set of bases \( G\phi \) that simultaneously reduce the reconstruction and misclassification errors. We should note that the cost function in Eq. 27 is either a quadratic or linear function that imposes linear inequality constraints on the set of unknowns. We proceed with solving it using its dual formulation. The Lagrangian of Eq. 27 is given by
\[
L(G\phi, \xi_i, \alpha_i, \beta_i) =
\lambda Tr \left( (\phi(X) - G\phi H) (\phi(X) - G\phi H)^T \right) + 
C \sum_{i=1}^{L} \xi_i - \sum_{i=1}^{L} \alpha_i [y_i (w_\phi^T G\phi h_i) + b] - 1 + \xi_i - \sum_{i=1}^{L} \beta_i \xi_i 
\]
where \( \alpha_i, \beta_i \) are the Lagrangian multipliers. Taking the derivative w.r. to the primal variables and equating to 0, we have
\[
G\phi = \left( 2\phi(X)H^T + \sum_{i=1}^{L} \alpha_i y_i w_\phi^T h_i^T \right) (2HH^T)^{-1}
\]
\[
\frac{\partial L}{\partial \xi_i} = 0 \quad \Rightarrow \quad 0 \leq \alpha_i \leq \theta, \quad 1 \leq i \leq L
\]
where \( \theta = C/\lambda \). Substituting the value of \( G\phi \) in Eq. 28 and simplifying, we get the dual problem
\[
\arg\max_{\alpha} \quad \alpha^T (T_1 - T_2) \alpha + (t_3 - t_4 - t_5 - t_6 + t_7) \alpha 
\]
s.t. \( 0 \leq \alpha_i \leq \theta \)
where

\[
\alpha \in \mathbb{R}^L, \quad T_1, T_2 \in \mathbb{R}^{L \times L}, \quad t_3, t_4, t_5, t_6, t_7 \in \mathbb{R}^{1 \times L},
\]

\[
T_1 = \left[ \sum_{k=1}^{L} y_i y_j h_k^T B h_i w^T \phi h_j B h_k \right]_{ij}
\]

\[
T_2 = \left[ y_i y_j w^T \phi h_i^T B h_j \right]_{ij}
\]

\[
t_3 = \left[ 4 \sum_{k=1}^{L} y_i h_k^T B \phi(X)^T w \phi h_j B h_k \right]_{li}
\]

\[
t_4 = \left[ 2 \sum_{k=1}^{L} y_i h_k^T B h_i w^T \phi(X) \right]_{li}
\]

\[
t_5 = \left[ 2 y_i w^T \phi(X) H^T B h_i \right]_{li}, \quad t_6 = b \left[ y_i \right]_{li}
\]

\[
t_7 = [111 \cdots 1]_{1 \times L}, \quad B = (2HH^T)^{-1}
\]

(32)

and \( h_k \) is the \( k^{th} \) column of the matrix \( H \). The above problem is quadratic in \( \alpha \), thus enabling us to use conventional quadratic programming tools to solve it. Once \( \alpha \) is estimated we can compute

\[
G_{\phi}^T G_{\phi} = B^T \left( 4 \phi(X)^T \phi(X) H^T + 4 \sum_{i=1}^{L} \alpha_i y_i \phi(X)^T \right)
\]

\[
+ w^T \phi h_i^T + w^T \phi \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j h_j^T \right) B
\]

(33)

and

\[
G_{\phi}^T \phi(X) = B \left( \phi(X)^T \phi(X) + \sum_{i=1}^{L} \alpha_i y_i h_i w^T \phi(X) \right)
\]

(34)

that are used in the subsequent optimization problems (e.g. Eq. 38 and Eq. 39).

The constant term \( \theta \) in Eq. 31 is used as a tuning parameter. Large values of \( \lambda \) (when compared to \( C \)), result in low values of \( \theta \) which cause \( \alpha_i \) to decrease and the second term of Eq. 29 to disappear. Hence for large values of \( \lambda \), the KMNMF cost function resembles that of KNMF.

20
Solve for $w_\phi, b, \xi$ by keeping $G_\phi$ and $H$ fixed: Having computed the updated bases matrix $G_\phi$ from Eq. 29 and keeping the weights matrix $H$ fixed, we proceed with calculating the maximum margin of the classifier. The features are obtained by reconstructing the data points in the feature space using the updated bases matrix. The optimization problem in Eq. 25 in that case strongly resembles that of a classical SVM:

$$\arg\min_{w_\phi, b, \xi} \frac{1}{2} w_\phi^T w_\phi + C \sum_{i=1}^{L} \xi_i$$

s.t. $y_i(w_\phi^T G_\phi h_i + b) \geq 1 - \xi_i$

$\xi_i \geq 0, \; 1 \leq i \leq L$.  

The above optimization problem intends to maximize the margin of the classifier in the feature space while reducing the misclassification error appearing when using the projections acquired from KNMF. The hyperplane parameters $w_\phi$, and $b$ are obtained using an off-the-shelf SVM classifier. The later takes as input the kernel matrix in the feature space, that is $H^T G_\phi^T G_\phi H$. This can be calculated using $H$ and the $G_\phi^T G_\phi$ that is explicitly computed in Eq. 33. After obtaining the support vectors and the solution of the dual formulation of the problem, we can explicitly compute $w_\phi^T w_\phi$ and $w_\phi^T \Phi(X)$.

Solve for $H$ by keeping $G_\phi, w_\phi, b, \text{ and } \xi$ fixed: We proceed with solving for the weights matrix $H$ by keeping all the remaining variables fixed ($G_\phi, w_\phi, b, \text{ and } \xi$). Since only the reconstruction error term of the optimization problem (Eq. 25) depends on $H$, the objective function in Eq. 25 is simplified as

$$\arg\min_{H} \| \Phi(X) - G_\phi H \|^2_F,$$

s.t. $H \succeq 0$.  

(36)

We solve for the weights matrix $H$ using quadratic programming. The $i^{th}$ column of $H$, $h_i$, contributes only to the $i^{th}$ data point $\Phi(x_i)$ and hence the columns of $H$ can be solved independently of each other. We adopt an alternative optimization procedure and rewrite the objective function in Eq. 36 as

$$\arg\min_{h_i} (\Phi(x_i) - G_\phi h_i)^T (\Phi(x_i) - G_\phi h_i),$$

s.t. $h_i \succeq 0 \; \forall i$.  

(37)
If $\gamma$ is the Lagrangian parameter, then the dual formulation for Eq. 37 is given by

$$\arg\max_{\gamma > 0} \frac{1}{2} \gamma^T M \gamma + 2 \gamma^T M G_{\phi}^T \Phi(x_i)$$

where $M = (2G_{\phi}^T G_{\phi})^{-1}$.

Following a procedure similar to the one used in MNMF, we get

$$h_i = h_i \odot \left( \frac{[G_{\phi}^T \Phi(x_i)]^+ + [G_{\phi}^T G_{\phi} h_i]^-}{[G_{\phi}^T G_{\phi} h_i]^+ + [G_{\phi}^T \Phi(x_i)]^-} \right).$$

This procedure is repeated for all columns of $H$.

During testing, the weight vector $h_t$ for the test data $x_t$ is computed as

$$h_t = (G_{\phi}^T G_{\phi})^{-1} (G_{\phi}^T \Phi(x_t))$$

where $G_{\phi}^T G_{\phi}$ is computed in Eq. 33 and $G_{\phi}^T \Phi(x_t)$ by substituting $X$ with $x_t$ in Eq. 34. The kernel matrix between the training and test samples is computed as $H^T G_{\phi}^T G_{\phi} h_t$ and is used as input to the SVM classifier that classifies the given test sample.

4. Experimental Results

In this section we demonstrate the performance of the proposed framework using real, publicly available datasets. More specifically, we use the INRIA-pedestrian dataset [27], the BU-3DFE [28] facial expression dataset, the Mediamill [29] dataset and the KTH actions dataset [30], which we will describe in detail below. To allow comparisons with previously reported methods, we report results obtained by SVM classifiers trained with the features extracted using the Semi-NMF [22] and the KNMF [23] algorithms. We also report results with the DNMF algorithm [14] followed by a K Nearest Neighbors (KNN) classifier. We show that the classification performance of the proposed algorithms that jointly learn the classifier parameters and the matrix factorization is consistently higher, especially when only few dimensions are retained.

4.1. INRIA Dataset

First, we tested our algorithm on the INRIA pedestrian dataset [27] that contains mostly front and back views. The dataset includes several variations caused by partial occlusions and scale, pose, clothing and illumination
changes. For our experiments we created a set of positive examples (containing pedestrians) and another one of negative examples (by sampling the background). The extracted bounding boxes were cropped to a size of $51 \times 100$ pixels. In order to handle possible illumination changes we used as features Histogram of Oriented Gradients (HOGs) [27]. The HOGs were extracted by creating a non-overlapping spatial grid size of $8 \times 8$, using 9 orientation bins per histogram. For each histogram four different normalizations were computed using adjacent histograms. This procedure resulted in a vector of length equal to 36 per region. For color images, the gradient was separately computed for each channel and the one with maximum magnitude was chosen, resulting in feature vectors of size 1440. In total we used 3548 positive examples and 3795 negative examples. For training, we randomly chose 200 positive and 200 negative images from the positive and negative image sets and used the remaining images for testing. This procedure was repeated several times and we averaged the acquired accuracies to calculate the accuracy of the classifier. In Fig. 5 a set of positive (pedestrian) and negative (background) examples extracted from the INRIA dataset are depicted.

![Figure 5: A sample of the positive and negative image examples used from the INRIA dataset.](image)

A set of bases images obtained using the proposed MNMF are shown in Fig. 6. For comparison reasons, we also depict the equivalent bases images acquired when using classic NMF. As we can see, the proposed algorithm results in bases images that have good localization characteristics.

In order to compare the performance of our algorithm with semi-NMF and KNMF we report the classification performance that is obtained when the bases that are obtained with each of these methods are used to train an SVM-classifier. We also report the classification performance of DNMF combined with a KNN classifier. The classification performance for different number of bases considered, that is for various values of $k$, is summarized in Fig. 7. We note that for each $k$ we repeat the experiments with different
training sets sampled from the main dataset and report the average accuracy over all runs. For simplicity, for each value of $k$, we used the value of $C$ that provided the best results for the semi-NMF+SVM algorithm as input for all the rest of the methods tested (including ours). It can be seen that the proposed method clearly outperforms all other methods in terms of the recognition accuracy for all values of $k$.

In Table 1 we report the performance of the proposed MNMF and KM-NMF algorithms as well as that of the baseline techniques in terms of recognition accuracy. As we can see, both the proposed MNMF and KM-NMF outperform the baseline techniques.

Figure 6: An example of the bases acquired for (a) NMF and (b) proposed MNMF algorithms.

Figure 7: The accuracy obtained for the INRIA dataset.
Table 1: Comparison of MNMF and KMNMF with baseline techniques (INRIA dataset).

<table>
<thead>
<tr>
<th>Method</th>
<th>SNMF + SVM</th>
<th>DNMF + KNN</th>
<th>KNMF + SVM</th>
<th>MNMF</th>
<th>KMNMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>92.29%</td>
<td>90.47%</td>
<td>92.51%</td>
<td>93.27%</td>
<td>94.96%</td>
</tr>
</tbody>
</table>

4.2. BU-3DFE Dataset

We then applied the proposed algorithms on the BU-3DFE [28] facial expression dataset that consists of 100 subjects (about 60% female and 40% male). The dataset contains six different facial expressions (anger, disgust, fear, happiness, sadness, surprise) performed by each subject at four intensities plus an image of the neutral state, captured at 5 yaw angles. In our experiments we only used the 2D images of the frontal view. The original images were cropped and down sampled to a size of $24 \times 40$ pixels. Fig. 8 shows a set of sample images used for the experiments.

![Figure 8: A sample of the images used from the BU-3DFE dataset.](image)

To extract features, we calculated the difference image for each subject, by subtracting the image corresponding to the neutral state from the equivalent image corresponding to the fully formed expression (highest intensity). A Gabor filter [31] of 2 scales and 4 orientations was applied on these difference images to yield feature vectors of size $7680 \times 1$.

For testing, we adopted a five-fold cross validation protocol. The dataset of 100 subjects was divided into 5 non overlapping groups of 20 subjects each. We used images from one group to form the test set and the images corresponding to the remaining 4 groups to create the training set. This procedure was repeated 5 times so that each group would be used for testing and the average classification accuracy was considered for the classifier. The number of bases ($k$) was set to 100. In order to perform multi-class classification, we
Table 2: Comparison of MNMF and KMNMF with baseline techniques (BU-3DFE dataset).

<table>
<thead>
<tr>
<th>Method</th>
<th>SNMF + SVM</th>
<th>DNMF + KNN</th>
<th>KNMF + SVM</th>
<th>MNMF</th>
<th>KMNMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>66.5%</td>
<td>65.00%</td>
<td>70.00%</td>
<td>69.00%</td>
<td>71.67%</td>
</tr>
</tbody>
</table>

employed a All-versus-All approach, *i.e.* a binary classifier was built for each pair of classes. During testing, the class that received the maximum number of votes won.

In Table 2 we report the classification accuracy of the proposed MNMF and KMNMF, as well as that of some baseline techniques (Semi-NMF and KNMF followed by SVM and DNMF followed by KNN). As we can see, the proposed MNMF and KMNMF both outperform the baseline techniques SNMF + SVM, DNMF + KNN and KNMF + SVM, with accuracy rates equal to 69.00%, 71.67%, 66.5%, 65.00% and 70.00%, respectively. Therefore, the introduction of the maximum margin constraints within the factorization procedure efficiently leads to better performance in terms of recognition accuracy.

4.3. Mediamill Dataset

We next studied the performance of the proposed algorithms on object classification using the Mediamill [29] dataset. The dataset consists of 43907 sub-shots of 101 classes. Each image was represented using a 120-dimensional feature vector. We randomly chose two object categories from the available 101 ones and performed a binary classification task. For training, we randomly chose 200 images from each class and used the remaining images for testing. This procedure was repeated several times and the average classification accuracy was regarded as the classification accuracy of the classifier.

The nonlinear kernel parameter σ in Eq. 26 was chosen to be same for both the proposed KMNMF and KNMF [23] algorithms, to ensure a fair comparison. As baseline techniques we again used the Semi-NMF and KNMF algorithms followed by a SVM classifier and the DNMF algorithm followed by a KNN classifier. The parameter σ was set equal to the standard deviation of the data, that is \( σ^2 = \frac{1}{N} \sum_{i=1}^{N} \|x_i - \bar{x}\|^2 \) [23]. We set the parameter values \( C = 100 \) and \( \lambda = 100 \) for MNMF and \( C = 100 \) and \( \lambda = 10^5 \) for KMNMF. We should note here that for each \( k \) we repeat the experiments with different
Figure 9: Comparison of the performance of the MNMF and KMNMF algorithms with DNMF +KNN [14], Semi-NMF [22] + SVM, KNMF [23] + SVM versus the number of bases $k$ for different categories of Mediamill dataset.
training sets sampled from the main dataset and report the average accuracy over all the runs. For all experiments, for each value of $k$, we used the value of $C$ (100) that provided the best results for the KNMF+SVM algorithms.

The classification performance for different number of bases ($k$) is summarized in Fig. 9. It can be seen that the proposed method clearly outperforms all other methods in terms of the classification error for all values of $k$. In particular, we notice that the proposed method significantly outperforms other methods when the number of basis vectors is small.

4.4. KTH Action Dataset

Subsequently, we studied the performance of the proposed algorithm on the KTH dataset consisting of 25 subjects in four scenarios performing six actions (box, handclap, handwave, jog, run and walk) under various scale and illumination scenarios. An example of each action is shown in Fig. 10. For each action we considered a period of 9 ‘naively’ chosen frames (not time-scaled). In order to have a better alignment for the training data we extracted bounding boxes of size $60 \times 80$ around the objects. Possible illumination changes were handled using as features HOGs. The HOGs were extracted by creating a non-overlapping spatial grid size of $8 \times 8$ and using 9 orientation bins per histogram. For each histogram four different normalizations were computed using adjacent histograms. This procedure resulted in a vector of length equal to 36 per region. The concatenation of these vectors resulted in a feature vector of size 1440 per image. The number of bases $k$ was set 135. In order to perform multi-class classification we employed the same

![Figure 10: Examples of action images from the KTH dataset.](image-url)
Table 3: Comparison of MNMF and KMNMF with baseline techniques (KTH dataset).

<table>
<thead>
<tr>
<th>Method</th>
<th>SNMF + SVM</th>
<th>DNMF + KNN</th>
<th>KNMF + SVM</th>
<th>MNMF</th>
<th>KMNMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>87.50%</td>
<td>84.00%</td>
<td>91.30%</td>
<td>90.17%</td>
<td>91.83%</td>
</tr>
</tbody>
</table>

all-versus-all strategy that we adopted in the experiments conducted for the BU-3DFE database. The leave-one-person-out cross validation approach was used to test the performance of the algorithms.

In Table 3 we report the performance of the proposed MNMF and KMNMF algorithms as well as that of the baseline techniques (SNMF + SVM, KNMF + SVM and DNMF + KNN) in terms of recognition accuracy. As we can see, both the proposed MNMF and KMNMF outperform the baseline techniques. KMNMF also introduces a 1.66% increase in recognition accuracy over MNMF, something that implies that the classes included in the KTH dataset are better separated when non-linear techniques are used.

4.5. Effect of Parameter $\lambda$

Having already shown that the proposed MNMF and KMNMF algorithms outperform schemes that employ factorization techniques followed by a SVM classifier, we further study the effect of the parameter $\lambda$ on the cost function Eq. 5 and on $\xi_i$. As we have defined in Eq. 9, the weight for the NMF cost $\lambda$ is inversely proportional to the tuning parameter $\theta$, i.e. $\theta = C/\lambda$. High values of $\lambda$ result in low values of $\alpha_i$, since $\alpha_i$ is upper bounded by $\theta$. Thus for high values of $\lambda$ the second term in Eq. 8 approaches zero and the update rules for $G$ are similar to the update rules of Semi-NMF [22] while for lower values of $\lambda$ the second term in Eq. 8 is introduced formulating in that way our framework. In Fig. 11 we plot the classification accuracy achieved for the train and test sets of the KTH dataset versus the value log($\lambda$). As we can see the proposed MNMF achieves a higher training and testing classification accuracy when a smaller value for the parameter $\lambda$ is considered. For larger values of $\lambda$ the classification accuracy of MNMF converges to the same value with that of SNMF followed by SVM.
5. Conclusions

In this paper we propose a maximum margin framework for the linear and non-linear Non-negative Matrix Factorization algorithm. Our aim is to impose soft max-margin constraints on the cost function of NMF in order to calculate the decomposition matrices that will enable us to perform classification while reducing simultaneously both the reconstruction and misclassification errors. In that way, we obtain the maximum margin of the classifier in the low or high dimensional space (for the case of linear and non linear NMF, respectively) while ensuring that it achieves the highest classification accuracy. To achieve that, we formulate a novel cost function that combines the reconstruction error term introduced by the matricization algorithm and the misclassification error introduced by the maximum margin classifier, bound together under SVM-type linear inequality constraints. The introduced cost function formulates a non-convex constrained optimization problem with respect to the bases and the separating hyperplane, which we solve following an iterative optimization procedure. At each iteration we solve for a set of convex (constrained quadratic or Support Vector Machine-type) sub-problems employing typical quadratic programming tools. We demonstrate the performance of the proposed algorithms on several computer vision problems such as pedestrian detection, image retrieval, facial expression recognition and action recognition using not only toy datasets but also publicly avail-
Appendix A. Proof of the convergence of the iterative optimization procedure

Here, we provide a proof of convergence for the proposed algorithm. More precisely, the iterative optimization method used, also known as alternative projections, never increases the value of Eq. 5 between two successive iterations, as it can be regarded to be a monotonic function (see also Fig. 1a). We define a continuous function of the form:

\[
D : \{G, H, w, \xi_i, b\} \times \mathbb{R} \rightarrow \mathbb{R}
\]  

where \( G \in \mathbb{G} \subset \mathbb{R}^{m \times k} \), \( H \in \mathbb{H} \subset \mathbb{R}^{k \times n}^+ \) and \( w \in \mathbb{W} \subset \mathbb{R}^m \). We define the following three functions

\[
D_1 : G \times \mathbb{R} \rightarrow \mathbb{R}
\]

\[
D_2 : W \times \mathbb{R} \rightarrow \mathbb{R}
\]

\[
D_3 : H \rightarrow \mathbb{R}
\]

(A.2)

defined as \( D_1(G, b) = D(G, \xi_i, b; H, w) \) (i.e., acquired fixing \( H \) and \( w \)) and \( D_2(w, b) = D(w, \xi_i, b; G, H) \), (i.e., acquired fixing \( G \) and \( H \)) and \( D_3(H) = D(H; G, w, \xi_i, b) \), (i.e, acquired fixing \( G, w, \xi_i \) and \( b \)) . By definition the function \( D \) has 3 mappings:

\[
g_1(G^*, b^*) \triangleq \arg\min_{G, b} D_1(G, b)
\]  

(A.3)

\[
g_2(w^*, b^*) \triangleq \arg\min_{w, b} D_2(w, b)
\]  

(A.4)

\[
g_3(H^*) \triangleq \arg\min_{H} D_3(H)
\]  

(A.5)

and * denotes optimality.

The sequence of produced solutions are characterized by the following relationships:

\[
D_1(G^*, b^*) \geq D_1(G, b)
\]

\[
D_2(w^*, b^*) \geq D_2(w, b)
\]

\[
D_3(H^*) \geq D_3(H).
\]  

(A.6)
Given an initial estimate \{G_0, H_0, w_0, b_0\}, the proposed algorithm generates a sequence of solutions \{G(t), H(t), w(t), b(t)\} via

\begin{align}
g_1(G^*_t, b^*_t) & \triangleq \arg\min_{G,b} D_1(G(t), b(t)) \\
g_2(w^*_t, b^*_t) & \triangleq \arg\min_{w,b} D_2(w(t), b(t)) \\
g_3(H^*_t) & \triangleq \arg\min_{H} D_3(H(t)).
\end{align}

The sequence of produced solutions are characterized by the following relationships:

\begin{align}
a_1 &= g_1(G^*_{{(1)}}, b^*_{{(1,1)}}) \\
& \geq g_2(w^*_{{(1)}}, b^*_{{(1,2)}}) \\
& \geq g_3(H^*_{{(1)}}) \\
& \geq \cdots \geq \\
& \geq g_1(G^*_{{(t)}}, b^*_{{(t,1)}}) \\
& \geq g_2(w^*_{{(t)}}, b^*_{{(t,2)}}) \\
& \geq g_3(H^*_{{(t)}}) = a_2
\end{align}

where \( t \to \infty \) and \( a_1, a_2 \) are limit values in \( \mathbb{R} \) and \( b^*_{{(1,1)}} \) is the value for \( b \) acquired when solving for \( G \) at time 1, while \( b^*_{{(1,2)}} \) is the value for \( b \) acquired when solving for \( w \) at time 1. Therefore we can regard the alternating optimization procedure to be a composition of 3 subalgorithms defined as:

\begin{align}
\Omega^1 : (G, b) & \to \mathbb{R}^{m \times k} \times \mathbb{R} \\
\Omega^2 : (w, b) & \to \mathbb{R}^m \times \mathbb{R} \\
\Omega^3 : (H) & \to \mathbb{R}^{k \times n} \times \mathbb{R}
\end{align}

producing \( G, H, w \) and \( b \). Then \( \Omega = \Omega_1 \circ \Omega_2 \circ \Omega_3 = \circ_{d=1}^3 \Omega_d \) is closed when all \( G, H, W \) are compact. We should emphasize here that since all subalgorithms decrease the value of \( D, \Omega \) is monotonic with respect to \( D \). Consequently, we can say that the alternating projection method converges.

References


