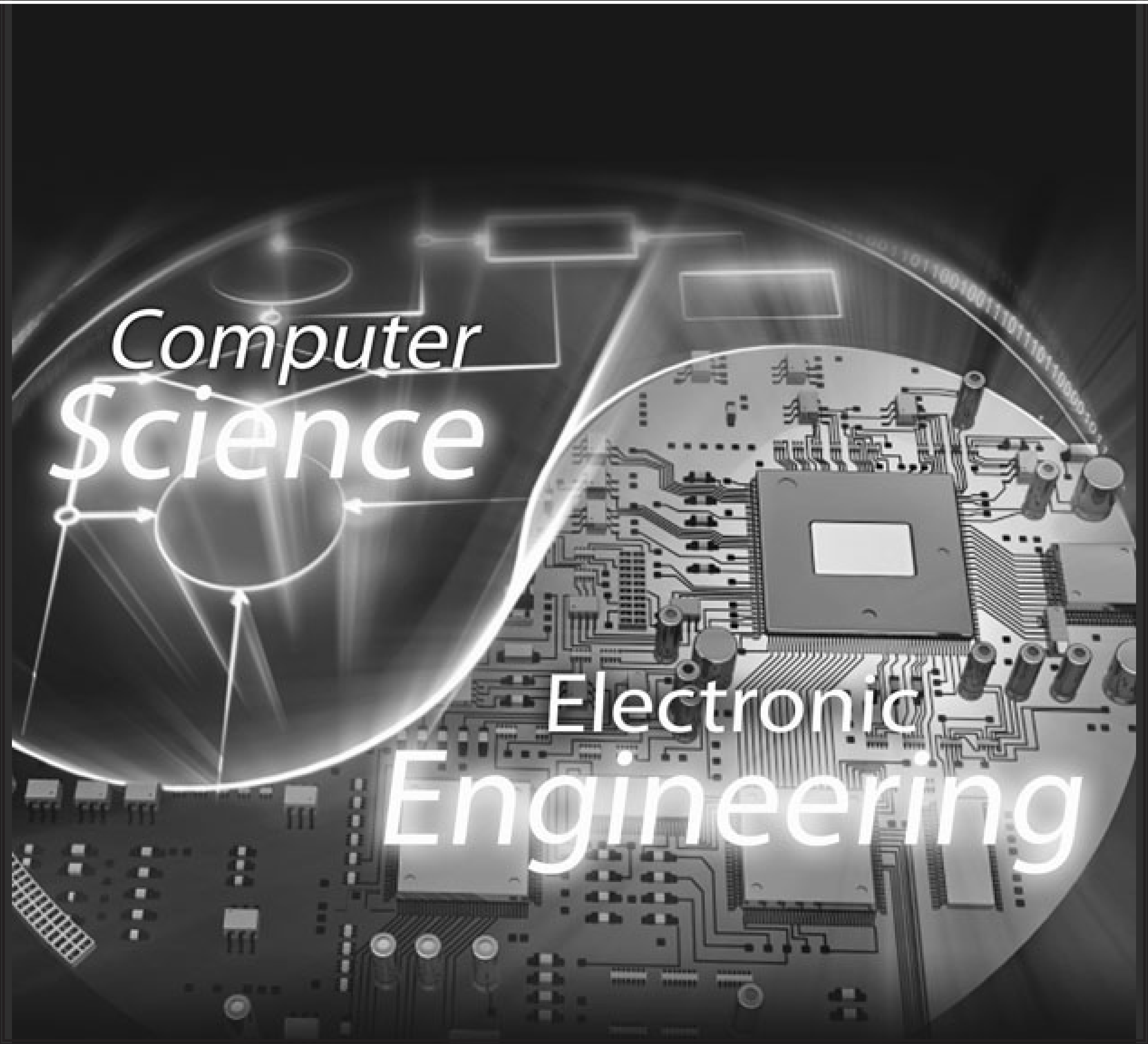
ISSN 2043-0167

Non Negative (Kernel) Max-Margin Matrix Factorization

Vijay Kumar B G and Dr. Ioannis Patras

EECSRR-11-02 May 2011

Queen Mary University of London
School of Electronic Engineering and Computer Science



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

Technical Report

Non Negative (Kernel) Max-Margin Matrix Factorization.

Authors
Vijay Kumar B G
Dr. Ioannis Patras

Contents

List of Figures

Abstract

In this work, we introduce linear and non-linear maximum margin framework for classification using Non-negative Matrix Factorization. By contrast to the traditional setting in which the classification and the matrix factorization stages are separated we incorporate the maximum margin 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 propose to solve in an iterative way, where at each iteration we solve a set of convex (constrained quadratic or SVM-type) sub-problems with respect to subsets of the unknown variables. By doing so, we obtain a bases matrix by which we extract features that maximize the margin of the resulting classifier in the low dimensional feature space. We also extend this framework to Kernel NMF where we maximize the margin of the classifier in the reconstructed feature space. The performance of the proposed algorithms are evaluated on several publicly available datasets where it is shown to consistently outperform SVM classifiers that use features that are extracted by Kernel NMF [28], semi-NMF [7] or Discriminative NMF [9].

Chapter 1

Introduction

The Non-Negative Matrix Factorization (NMF) algorithm is one of the most popular Machine Learning techniques for data dimensionality reduction. NMF decomposes the data matrix into non-subtractive combinations of non-negative bases [12]. Its ability to produce parts-based representations is theoretically justified and experimentally demonstrated in [10]. By contrast, other dimensionality reduction methods, such as the Principal Component Analysis (PCA) [23] result in bases and projection coefficients that can take either positive or negative values.

The first formulations for NMF were proposed in Lee *et al.* [10] and Paatero *et al.* [17]. In both approaches, the bases and coefficient matrices are 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 is quantified either using the Kullback-Leibler divergence [10] or the least squares error [18] . In an efficient implementation, Lee *et al.* [11] defined a set of multiplicative update rules that are derived from the optimization of an auxiliary function that bounds the cost function from above. Lin *et al.* [14] showed that the minimization of the auxiliary function indeed reduces the cost function but does not warranty the convergence of the algorithm to the stationary point of the original optimization problem. Two projected gradient-based methods for NMF that exhibited strong optimization properties were proposed in [13]. 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 were proposed in [3] that ensure rapid error reduction rate and faster convergence.

Although NMF usually results in a part-based representation, the 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 [5,12]. Several other algorithms [8,16,19] aim to achieve sparsity with tunable parameters. In [8,16], 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. By contrast, the approaches in [19] 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 in a low dimensional subspace. In this context, it has been extensively used for facial analysis including detection [5], recognition [12], verification [27] and expression recognition [9,26]. Several other applications of NMF in Computer Vision include pose estimation [1], action recognition [25,21], object recognition [15], subspace learning [4] and clustering [7].

In [1], NMF bases and coefficients are learned using a set of features extracted from clutter-free images containing objects. In [21], the NMF coefficients are extracted using appearance features and motion vectors. These coefficients are subsequently used to train a cascaded LDA-based classifier. The technique in [25] follows an approach similar to [1] for detection of humans in image sequences, where NMF is employed to learn a set of pose primitives. In [15], two approaches are followed in order to improve the recognition rate using features extracted by NMF. The first approach uses a Riemannian metric for the learned feature vectors instead of the classic Euclidean distance, while the second one orthonormalizes the NMF bases and then uses the features projected onto these bases. Cai *et al.* [4] proposed Graph Regularized NMF (GNMF) that models the data subspace as a submanifold embedded in an ambient space. By learning NMF on such a manifold, GNMF has more discriminative power when compared to NMF which only considers the Euclidean space. Ding *et al.* [7] proposed Semi-NMF for clustering in which the non-negativity constraints on the basis matrix are relaxed. This leads to a basis matrix that contains cluster centers and non-negative coefficients that can be interpreted as cluster indicators.

Only few works exist that use constraints that aim at increasing the discriminative power of the extracted features. Zafeiriou *et al.* [27] introduced discriminative constraints in order to extract bases that correspond to discriminative facial regions for the problem of face recognition. Indeed, DNMF [27] results in bases corresponding to salient facial features such as eyes, mouth etc, that are vital for discrimination. Kotsia *et al.* [9]

proposed projected gradients DNMF (PGDNMF) for facial expression recognition which differed from DNMF in two main ways. First, projected gradients were used instead of multiplicative update rules as the former guarantee the convergence of the algorithm to a limit point that is also a stationary point of the original optimization problem. Second, the discriminant analysis is 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 to yield more discriminative bases.

However, the introduced constraints were taylored for a rather simplistic LDA-based classifier. In the proposed framework, the choice of the projections acquired is performed in such a way that it maximizes the discriminative ability of an SVM classifier, a fact that results in higher classification performance as will be demonstrated in the section of the experimental results. In chapter 2 we propose a max-margin framework for Semi-NMF and we extend this framework for kernel NMF in chapter 3

Chapter 2

Max-Margin Linear Semi NMF[1]

2.1 Introduction

In this paper we introduce soft max-margin constraints to the objective function of NMF to obtain a bases matrix that maximizes the classification margin using the features that are extracted using those bases. 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 that is used in typical SVM formulations, under SVM-type linear inequality constraints. The optimization is 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 one of them 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. The proposed method is applied to publicly available databases (the INRIA pedestrian, the KTH action and the mushroom1 datasets) where we demonstrate that it consistently outperforms SVM classification schemes that use features that are extracted using Semi-NMF [7], or DNMF [27].

Summarizing, the main contributions of this part are

- We introduce a max-margin framework for Semi non Negative Matrix Factorization (MNMF).
- We propose an optimization scheme that solves for a max-margin classifier simultaneously with the decomposition matrices and bases.

[1]The text except of Section 2.4.1 appear as submitted to CVPR2011, manuscript id:1831

The rest of the paper is organized as follows. In Section 2.2, we briefly describe the NMF and the Semi-NMF schemes. In Section 2.3, we formulate the proposed max-margin framework for semi-NMF as an optimization problem and describe an algorithm to solve it. We demonstrate the performance of the proposed algorithm in Section 2.4 and, we draw conclusions in Section 2.5.

2.2 Semi Non-negative Matrix Factorization

In this section, we present a brief overview of the semi-NMF technique for matrix decomposition. Let $\mathbf{X} \in \mathbb{R}^{m\times n}$ represent a non-negative matrix having n examples in its columns. The NMF algorithm [10] decomposes $\mathbf{X}$ into two non-negative matrices, the bases matrix $\mathbf{G} \in \mathbb{R}^{m\times k}$ and the coefficients matrix $\mathbf{H} \in \mathbb{R}^{k\times n}$ such that $\mathbf{X} \approx \mathbf{GH}$. k is typically chosen to be small ($< \min(m, n)$) in order to accomplish dimensionality reduction. The columns of $\mathbf{G}$ can be regarded as the bases vectors and thus each example can be represented as the linear combination of those bases vectors as $\mathbf{x}_i = \mathbf{Gh}_i$. Here $\mathbf{x}_i$ and $\mathbf{h}_i$ are the i^{th} columns of $\mathbf{X}$ and $\mathbf{H}$, respectively.

Ding *et al.* [7] introduced Semi-NMF that relaxes the non-negativity constraints on $\mathbf{G}$ and hence on the data matrix $\mathbf{X}$. Their motivation was based on the case of clustering with $\mathbf{G}$ representing the cluster centers and $\mathbf{H}$ denoting the cluster indicators. The matrices are determined by minimizing the reconstruction error $||\mathbf{X} - \mathbf{GH}||_F^2$ or the Kullback-Leibler divergence $\mathrm{D}(\mathbf{X}||\mathbf{GH})$ w.r.t. $\mathbf{G}$ and $\mathbf{H}$

$$\underset{\mathbf{H}\geq 0}{\operatorname{argmin}} ||\mathbf{X} - \mathbf{GH}||_F^2 \tag{2.1}$$

or

$$\underset{\mathbf{H}\geq 0}{\operatorname{argmin}}\, \mathrm{D}(\mathbf{X}||\mathbf{GH}) \tag{2.2}$$

where $||.||$ corresponds to the Frobenious norm. From now onwards we will use the notation $\mathbf{H} \geq 0$ to specify 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 [7]:
Step 1: Update $\mathbf{G}$ by keeping $\mathbf{H}$ fixed

$$\mathbf{G} = \mathbf{XH}^T(\mathbf{HH}^T)^{-1} \tag{2.3}$$

Step2: Update $\mathbf{H}$ by keeping $\mathbf{G}$ fixed at the value computed in the above step,

$$\mathbf{H} = \mathbf{H} \odot \sqrt{\frac{[\mathbf{G}^T\mathbf{X}]^+ + [\mathbf{G}^T\mathbf{G}]^-\mathbf{H}}{[\mathbf{G}^T\mathbf{G}]^+\mathbf{H} + [\mathbf{G}^T\mathbf{X}]^-}} \tag{2.4}$$

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

$$\mathbf{M}^+{}_{ik} = \frac{1}{2}(|\mathbf{M}_{ik}| + \mathbf{M}_{ik}), \qquad \mathbf{M}^-{}_{ik} = \frac{1}{2}(|\mathbf{M}_{ik}| - \mathbf{M}_{ik}).$$

2.3 Max-Margin Semi-NMF

The NMF algorithm described in [10] minimizes either the cost function defined in Eq. 2.1 or the one in Eq. 2.2, imposing at the same time non-negativity constraints on $\mathbf{G}$ and $\mathbf{H}$. These non-negativity constraints result in a part-based representation of the data. Several variants of NMF with discriminant constraints imposed were proposed in [7,9,27]. The variations were obtained by introducing application specific discriminant constraints to the cost function. Inspired by this, we aim at finding a set of basis vectors that maximizes the margin of a SVM classifier.

2.3.1 Cost Function

Let $\{\mathbf{x}_i, y_i\}_{i=1}^{L}$ denote a set of data vectors and their corresponding labels, where $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \{-1, 1\}$. The objective is to determine a set of basis vectors 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 $\mathbf{G}$. Let us assume that the projection vector for a data example $\mathbf{x}_j$ is given by $\acute{\mathbf{x}}_j = \mathbf{G}^{\dagger}\mathbf{x}$ where $\mathbf{G}^{\dagger}$ corresponds to the pseudo-inverse of $\mathbf{G}$ and is defined as $\mathbf{G}^{\dagger} = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T$. In practice, $\mathbf{G}^{\dagger}$ may suffer from numerical stability problems and is hard to work with since its calculation requires a matrix inversion. In order to overcome this, we use $\mathbf{G}^T\mathbf{x}$ as the features for the classifier [9,27]. Then, the optimization problem for the proposed criterion is given by

$$\begin{aligned} &\underset{\mathbf{G},\mathbf{H},\mathbf{w},b,\xi_i}{\operatorname{argmin}} \lambda\|\mathbf{X}-\mathbf{G}\mathbf{H}\|_F^2 + \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{L}\xi_i \\ &\text{s.t. } y_i(\mathbf{w}^T\mathbf{G}^T\mathbf{x}_i + b) \geqslant 1-\xi_i \\ &\xi_i > 0, \forall\ 1 \le i \le L,\ \mathbf{H} \ge 0 \end{aligned} \tag{2.5}$$

where $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{L}$, λ and C are positive constants and λ is the weight factor for the NMF cost. The first term in the above optimization problem corresponds to the NMF reconstruction error while the remaining terms correspond to the maximum margin classifier. The above formulation aims at maximizing the margin of the support vectors while at the same time minimizing the reconstruction and misclassification error. The classifier is trained on the projected data points $\mathbf{G}^T\mathbf{x}$, obtaining in this way the hyperplane parameter $\mathbf{w} \in \mathbb{R}^k$, typically $k \ll m$. We iteratively solve for one of the terms $\mathbf{G}$, $\mathbf{H}$ and $\mathbf{w}, b, \xi_i$ by keeping the remaining parameters fixed as described below.

The steps followed in the proposed max-margin Semi-NMF framework are summarized in Algorithm 1.

Algorithm 1: Algorithm for Max-Margin Semi-NMF

```
input  : $\mathbf{X}$, $\mathbf{G}init$, $\mathbf{H}init$, $MAXITER$, $\lambda$, $C$
output: $\mathbf{G}$, $\mathbf{H}$, $\mathbf{w}$, $b$
begin
    $\mathbf{G} = \mathbf{G}init$;
    $\mathbf{H} = \mathbf{H}init$;
    repeat
        $S1$ : Solve for $\boldsymbol{\alpha}$ in Eq. 2.10
        $S2$ : Compute $\mathbf{G}$ using Eq. 2.8
        $S3$ : Find the classifier parameters, $\mathbf{w}$, $b$ for the updated $\mathbf{G}$
        $S4$ : foreach column of $\mathbf{H}$ do
            Calculate $\boldsymbol{\gamma}$ using Eq. 2.18
            Compute $\mathbf{h}$ using Eq. 2.17
        end
    until $iter \le MAXITER$  or  $convergence$;
end
```

Solve for G by keeping H, w and b fixed: Since $\mathbf{w}$ is fixed, the optimization problem in Eq. 2.5 is simplified as

$$\begin{aligned} \underset{\mathbf{G},\xi_i}{\operatorname{argmin}} \quad & \lambda\|\mathbf{X}-\mathbf{GH}\|_F^2 + C\sum_{i=1}^{L}\xi_i \\ & \text{s.t. } y_i(\mathbf{w}^T\mathbf{G}^T\mathbf{x}_i + b) \geqslant 1-\xi_i \\ & \xi_i > 0, \forall \;\; 1 \leq i \leq L \end{aligned} \tag{2.6}$$

The above formulation 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 (2nd term). Hence we want to find a set of bases $\mathbf{G}$ that simultaneously reduce the reconstruction error and the misclassification. Note, that we arrived at a cost function that is quadratic or linear with respect to the unknowns and at linear inequality constraints. We propose solving the above problem using its dual formulation. The Lagrangian of Eq. 2.6 is given by

$$\begin{aligned} L(\mathbf{G},\xi_i,\alpha_i,\beta_i) &= \lambda Tr\Big((\mathbf{X}-\mathbf{GH})(\mathbf{X}-\mathbf{GH})^T\Big) + \\ C\sum_{i=1}^{L}\xi_i - &\sum_{i=1}^{L}\alpha_i\left[y_i(\mathbf{w}^T\mathbf{G}^T\mathbf{x}_i+b)-1+\xi_i\right] - \sum_{i=1}^{L}\beta_i\xi_i \\ &\alpha_i,\beta_i > 0 \;\; \forall \;\; 1\leq i\leq L \end{aligned} \tag{2.7}$$

where α_i, β_i are the Lagrangian multipliers. Taking the derivative w.r. to the primal variables and equating to 0, we have

$$\frac{\partial L}{\partial \mathbf{G}} = -2\mathbf{XH}^T + 2\mathbf{GHH}^T - \sum_{i=1}^{L}\alpha_i y_i \mathbf{x}_i\mathbf{w}^T = 0$$

$$\Rightarrow \quad \mathbf{G} = \left(2\mathbf{XH}^T + \sum_{i=1}^{L}\alpha_i y_i\mathbf{x}_i\mathbf{w}^T\right)(2\mathbf{HH}^T)^{-1} \tag{2.8}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \quad \Rightarrow \quad 0 \leq \alpha_i \leq \theta, \;\; \forall \;\; 1 \leq i \leq L \tag{2.9}$$

where $\theta = C/\lambda$. Substituting the value of $\mathbf{G}$ in Eq. 2.7 and simplifying, we get the dual problem

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{argmax}} \quad & \boldsymbol{\alpha}^T(\mathbf{T}_1 - \mathbf{T}_2)\boldsymbol{\alpha} + (\mathbf{t}_3 - \mathbf{t}_4 - \mathbf{t}_5 - \mathbf{t}_6 + \mathbf{t}_7)\boldsymbol{\alpha} \\ & \text{s.t.} \;\; 0 \leq \alpha_i \leq \theta \end{aligned} \tag{2.10}$$

where

$$\begin{aligned} & \boldsymbol{\alpha} \in \mathbb{R}^L, \;\; \mathbf{T}_1, \mathbf{T}_2 \in \mathbb{R}^{L\times L} \;,\; \mathbf{t}_3, \mathbf{t}_4, \mathbf{t}_5, \mathbf{t}_6, \mathbf{t}_7 \in \mathbb{R}^{1\times L}, \\ & \mathbf{T}_1 = \left[\sum_{k=1}^{L} y_i y_j \mathbf{h}_k^T \mathbf{B} \mathbf{M}_i^T \mathbf{M}_j \mathbf{B} \mathbf{h}_k\right]_{ij} \\ & \mathbf{T}_2 = \left[y_1 y_2 \mathbf{w}^T \mathbf{B} \mathbf{M}_j^T \mathbf{x}_i\right]_{ij} \\ & \mathbf{t}_3 = \left[4 \sum_{k=1}^{L} y_i \mathbf{h}_k^T \mathbf{B} \mathbf{H} \mathbf{X}^T \mathbf{M}_i \mathbf{B} \mathbf{h}_k\right]_{1i} \\ & \mathbf{t}_4 = \left[2 \sum_{k=1}^{L} y_i \mathbf{h}_k^T \mathbf{B} \mathbf{w} \mathbf{x}_i^T \mathbf{x}_k\right]_{1i} \\ & \mathbf{t}_5 = \left[2 y_i \mathbf{w}^T \mathbf{B} \mathbf{H} \mathbf{X}^T \mathbf{x}_i\right]_{1i}, \;\; \mathbf{t}_6 = b\,[y_i]_{1i} \\ & \mathbf{t}_7 = [111\cdots 1]_{1\times L}, \;\; \mathbf{B} = (2\mathbf{H}\mathbf{H}^T)^{-1}, \;\; \mathbf{M}_i = \mathbf{x}_i \mathbf{w}^T, \end{aligned} \tag{2.11}$$

and $\mathbf{h}_k$ is the k^{th} column of the matrix $\mathbf{H}$.

The above problem is quadratic in $\boldsymbol{\alpha}$. Therefore the conventional quadratic programming tools can be used to solve for $\boldsymbol{\alpha}$. The $\boldsymbol{\alpha}_i$ obtained is then used to compute $\mathbf{G}$ using Eq. 2.8. The constant term θ in Eq. 2.10 is used as a tuning parameter. Large values of λ (compared to C), result in low values of θ which in turn reduces α_i. This causes the second term in Eq. 2.8 to vanish making the update rule of $\mathbf{G}$ to be the one used in semi-NMF, as given in Eq. 2.3. Hence for large values of λ, the update rule for $\mathbf{G}$ tends to approach the update rule of semi-NMF, something that is also evident in Eq. 2.6.

Solve for $\mathbf{w}, b, \boldsymbol{\xi}$ by keeping $\mathbf{G}$ and $\mathbf{w}$ fixed: In Eq. 2.8, we computed the updated basis $\mathbf{G}$ using quadratic programming. We now keep the basis $\mathbf{G}$ and weight matrix $\mathbf{H}$ fixed and determine a hyperplane that maximizes the margin of the classifier. The features are obtained by projecting the data points onto the updated basis matrix. Since $\mathbf{G}$ and $\mathbf{H}$ are fixed, the optimization problem in Eq. 2.5 is simplified to that of a classical SVM:

$$\begin{aligned}&\underset{\mathbf{w},b,\xi_i}{\text{argmin}}\,\frac{1}{2}\mathbf{w}^T\mathbf{w}+C\sum_{i=1}^{L}\xi_i\\&\text{s.t. }\; y_i(\mathbf{w}^T\mathbf{G}^T\mathbf{x}_i+b)\geqslant 1-\xi_i\\&\xi_i>0,\forall\;\; 1\leq i\leq L.\end{aligned} \tag{2.12}$$

The above optimization problem intends to maximize the margin of the classifier while reducing the misclassification error. The hyperplane parameters $\mathbf{w}$, and b are obtained using a off-the-shelf SVM classifier.

Solve for H by keeping G, w, and *b* fixed: We now solve for matrix $\mathbf{H}$ by keeping all the remaining variables fixed. Since only the reconstruction error term of the optimization problem (Eq. 2.5) depends on $\mathbf{H}$, the objective function is simplified as

$$\begin{aligned}&\underset{\mathbf{H}}{\text{argmin}}\|\mathbf{X}-\mathbf{GH}\|_F^2,\\&\text{s.t. }\;\mathbf{H}\geq 0\end{aligned} \tag{2.13}$$

In order to find an $\mathbf{H}$ that is consistent we solve for $\mathbf{H}$ using quadratic programming. The i^{th} column of $\mathbf{H}$, $\mathbf{h}_i$ contributes only to the i^{th} data point $\mathbf{x}_i$ and hence the columns of $\mathbf{H}$ can be solved independently of each other. The above optimization problem can be solved using the update equation Eq. 2.4. Here we adopt an alternative optimization method. In particular, the objective function in Eq. 2.13 can be rewritten as

$$\sum_{i=1}^{L}\|\mathbf{X}_i-\mathbf{G}\mathbf{h}_i\|_F^2=\sum_{i=1}^{L}(\mathbf{X}_i-\mathbf{G}\mathbf{h}_i)^T(\mathbf{X}_i-\mathbf{G}\mathbf{h}_i) \tag{2.14}$$

where the i^{th} column of the matrix $\mathbf{H}$, denoted as $\mathbf{h}_i$ is given by

$$\begin{aligned}&\underset{\mathbf{h}_i}{\text{argmin}}\quad(\mathbf{x}_i-\mathbf{G}\mathbf{h}_i)^T(\mathbf{x}_i-\mathbf{G}\mathbf{h}_i),\\&\text{s.t. }\;\mathbf{h}_i\geq 0\end{aligned} \tag{2.15}$$

The Lagrangian of the above cost function is

$$L(\mathbf{h}_i)=(\mathbf{x}_i-\mathbf{G}\mathbf{h}_i)^T(\mathbf{x}_i-\mathbf{G}\mathbf{h}_i)-\boldsymbol{\gamma}^T\mathbf{h}_i,\quad \gamma>0 \tag{2.16}$$

where $\boldsymbol{\gamma} \in \mathbb{R}^k$ is a vector of positive Lagrangian multiplier. Differentiating the above equation w.r.t. $\mathbf{h}_i$ and equating to zero, we get

$$\mathbf{h}_i = (2\mathbf{G}^T\mathbf{G})^{-1}(2\mathbf{G}^T\mathbf{x}_i + \boldsymbol{\gamma}) \tag{2.17}$$

The dual formulation for Eq. 2.16 is given by

$$\begin{gathered}\underset{\boldsymbol{\gamma}>0}{\text{argmax}} \frac{1}{2}\boldsymbol{\gamma}^T\mathbf{B}\boldsymbol{\gamma} + 2\boldsymbol{\gamma}^T\mathbf{B}\mathbf{G}^T\mathbf{x}_i \\ \text{where } \ \mathbf{B} = (2\mathbf{G}^T\mathbf{G})^{-1}\end{gathered} \tag{2.18}$$

The above problem is quadratic in $\boldsymbol{\gamma}$. We use a Quadratic Programming solver to solve Eq. 2.18. The weight vector $\mathbf{h}_i$ is obtained by substituting the computed value of $\boldsymbol{\gamma}$ in Eq. 2.17. This procedure is repeated for all columns of $\mathbf{H}$.

During testing, the input test vector $\mathbf{x}_{test}$ is projected onto the basis matrix to obtain the feature vector, $\mathbf{f}_{test} = \mathbf{G}^T\mathbf{x}_{test}$. The feature vector thus obtained is applied to the max-margin classifier which predicts the class $\hat{\mathbf{y}}_{test} = sign(\mathbf{w}^T\mathbf{f}_{test} + b)$ where $\mathbf{w}, b, \mathbf{G}$ are computed during training.

2.4 Experimental Results

In this section we demonstrate the performance of the proposed framework using both artificial and real, publicly available datasets. More specifically, apart from an artificial toy datasets we use the INRIA pedestrian dataset [6], a subset of the KTH actions dataset [22] and the mushrooms1 dataset from the Proben 1 dataset [20]. To allow comparisons with previously reported methods, we also train SVM classifiers on the features that are exracted using Semi-NMF [7] and DNMF [27] algorithms. We show that the classification performance of the proposed scheme that jointly learns the classifier and performs matric factorization is consistently higher, especially when when only few dimensions are retained.

In order to give an insight to the workings of the proposed algorithm, we first apply it on a toy dataset consisting of two classes each of which contains 100 points that are sampled from 50-dimensional Gaussian distributions. In order to better visualize our results, we restrict the number of bases taken under consideration to be equal to two ($k = 2$). The bases matrix $\mathbf{G}$ and the weight matrix $\mathbf{H}$ are computed using the Semi-NMF

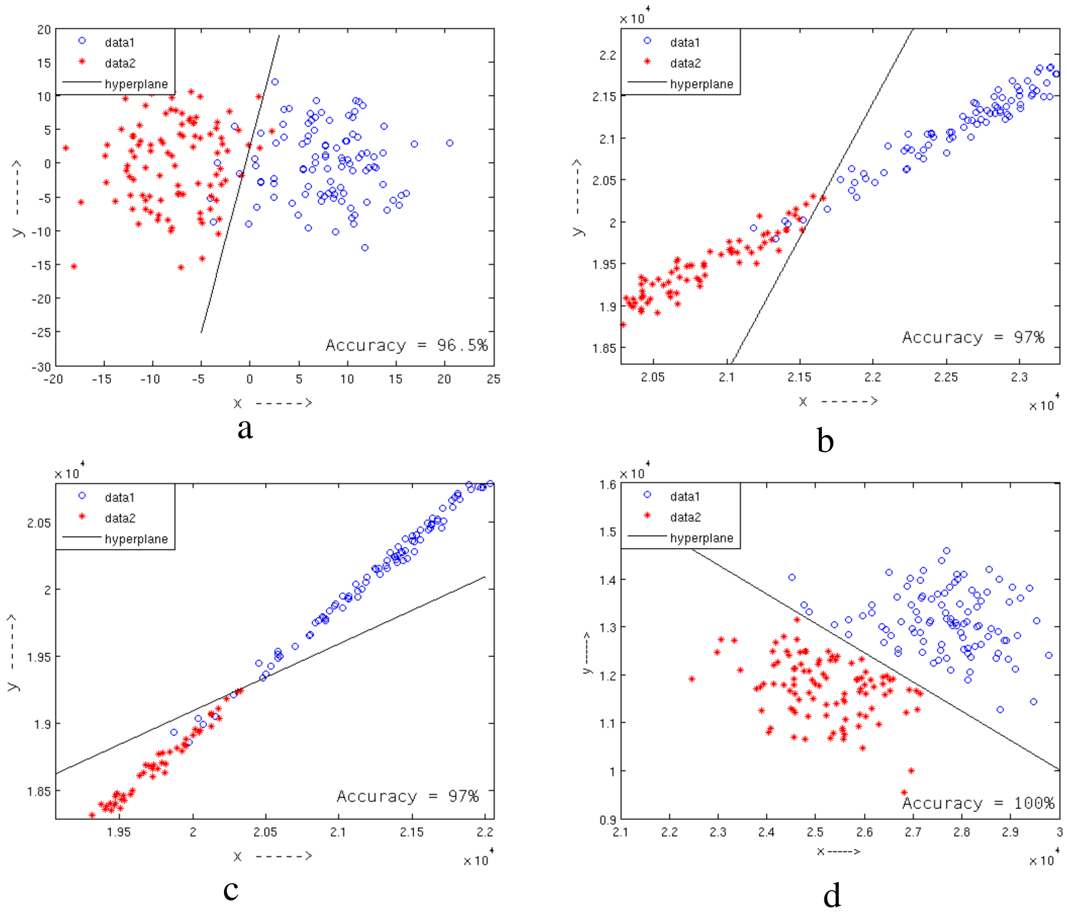


Figure 2.1: The projections and the SVM separating hyperplane using (*a*) PCA (*b*) Semi-NMF bases. (*c*) Max-margin NMF bases (1st iteration) and (*d*) Max-margin NMF bases (6th iteration) respectively.

algorithm and the input datapoints are projected onto the lower dimensional subspace using the acquired **G**. In Fig. 2.1(a) we show the projections of the points after applying a common dimensionality reduction technique, Principal Component Analysis (PCA). Fig. 2.1(b) depicts the projections of the input datapoints using the bases extracted using Semi-NMF. Fig. 2.1(c) and Fig. 2.1(d) show the projections after the first and the sixth iterations respectively of the proposed MNMF algorithm. It is clear that in the case of the proposed MNMF the projections are such that the different classes become 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.

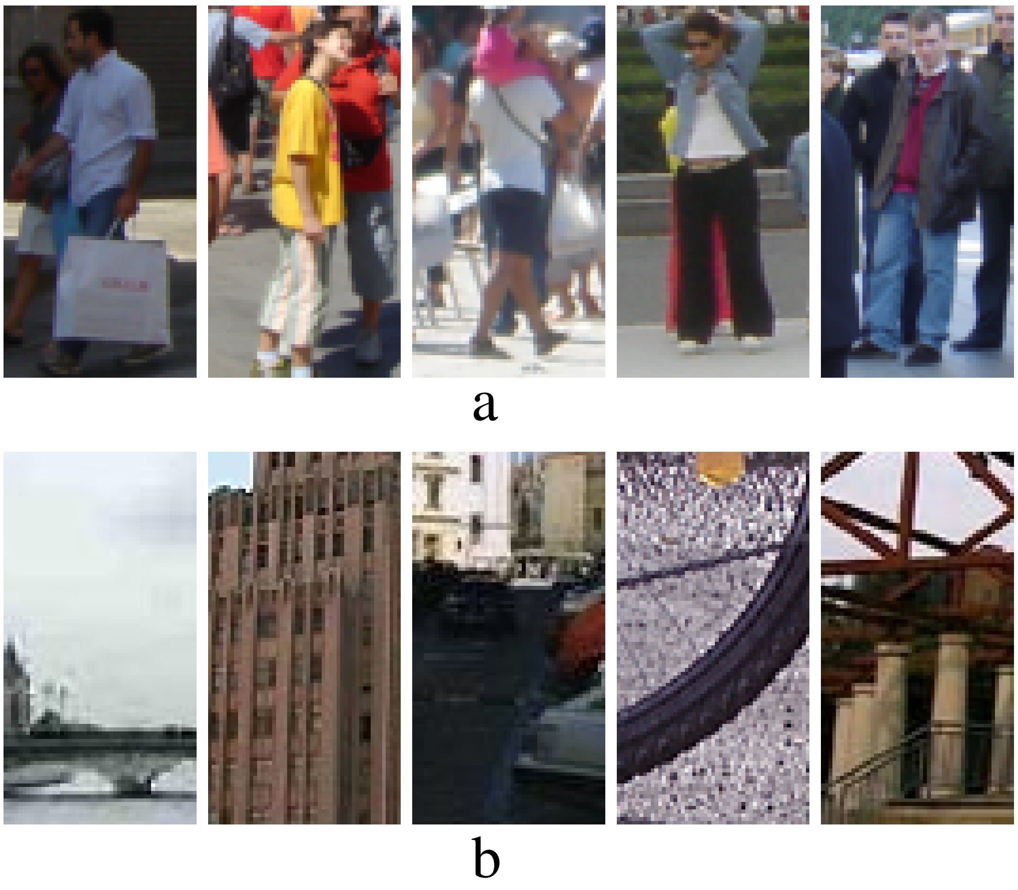


Figure 2.2: Sample images from the INRIA pedestrian dataset: a) Cropped positive training examples and b) cropped negative training examples.

Subsequently, we tested our algorithm on the INRIA-pedestrian dataset [6] that contains mostly front and back views. The dataset includes several variations caused by partial occlusions and scale, pose, clothing and illumination changes. 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 51×100. In order to handle possible illumination changes we used as features Histogram of Oriented Gradients (HOGs). The HOGs were extracted by creating a non-overlapping spatial grid size of 8×8 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. 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. The positive and negative examples were cropped to a size of 51×100. For training, we randomly chose 200 positive and 200 negative images from the positive and negative image sets and used the remaining 6943 images for testing. This procedure was repeated several times and we averaged the acquired accuracies to calculate the accuracy of the classifier. In Fig. 2.2 examples of positive (pedestrian) and negative (background) images from the INRIA dataset are depicted.

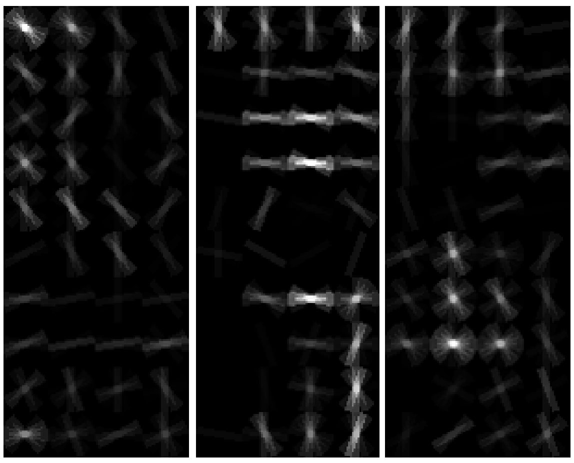


Figure 2.3: Examples of MNMF bases.

In Fig. 2.3 some examples of the bases obtained using MNMF are depicted. It is apparent that, similarly to when the bases are obtained using plain NMF, the bases that are obtained by our method have good localization characteristics. In order to compare the performance of our algorithm with semi-NMF and DNMF 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. The classification performance for different number of bases considered, that is for various values of k, is summarized in Fig. 2.4. 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 the 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 classification error for all values of k. In general, the relaxation of the non-negativity constraints seems to have a positive effect as the classification accuracy for the corresponding algorithms is higher.

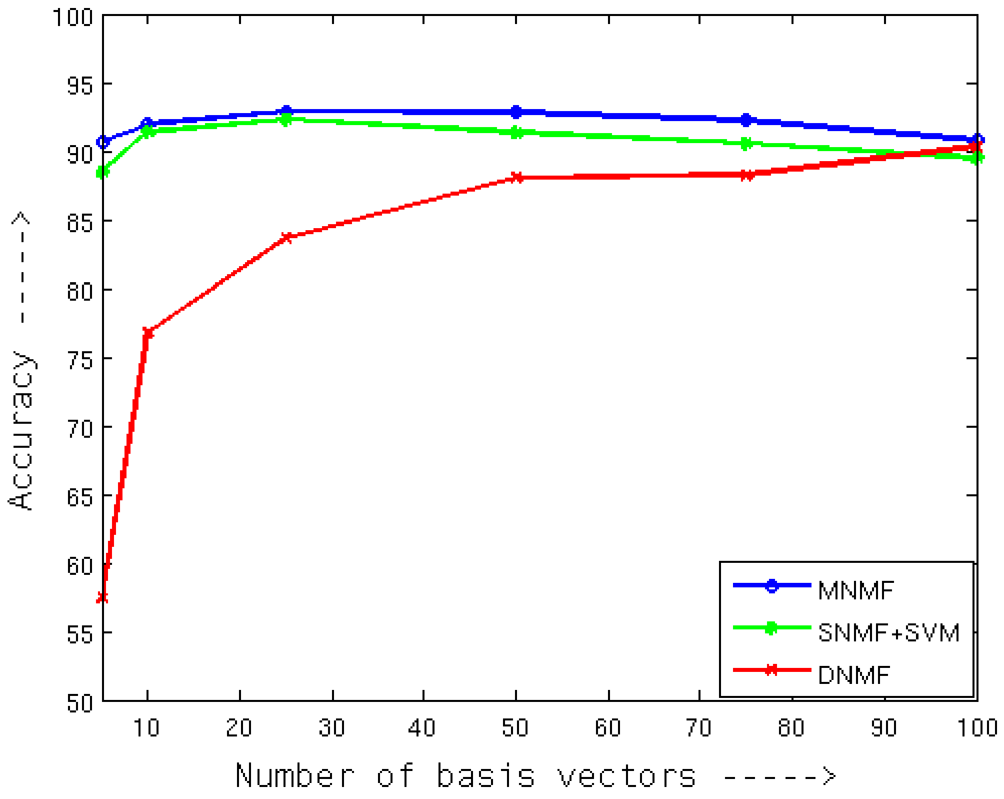


Figure 2.4: Classification accuracy vs the number of basis vectors for the INRIA dataset.

Subsequently, we test the proposed algorithm on the KTH dataset consisting of 25 subjects in four scenarios performing actions under various scale and illumination scenarios. We consider only the videos corresponding to two similar action categories namely *walking* and *running*. For the experiments we ignore the temporal information and consider only the spatial one. A period containing 9 naively chosen frames was extracted for every image sequence of every action. In order to have a better alignment for the training data we extracted bounding boxes of size 60×80 around the objects and we also calculated the HoG features from these images following the same procedure as the one used for the INRIA database. In total we extracted 900 images for each action, 200 of which were used as training data and the remaining as test samples. In Fig. 2.5 we present sample images for the actions *walking* and *running* of the KTH dataset.

We compared our algorithm with Semi-NMF followed by SVMs [7] and DNMF followed by SVMs [9] algorithms. Fig. 2.6 shows the accuracies achieved when various numbers of bases were used. For low values of k, Semi-NMF and MNMF exhibit similar performance. However, as the number of bases increases, we MNMF outperforms Semi-

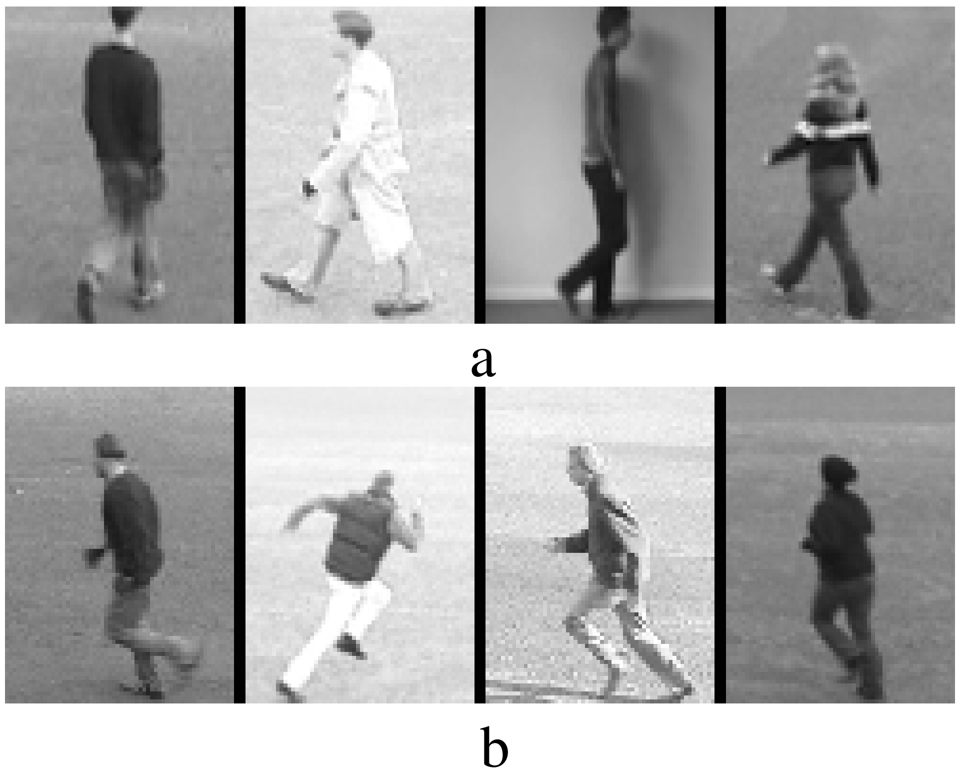


Figure 2.5: Examples of a) Walking and b) Running from the KTH dataset.

NMF. We should also note here that the proposed MNMF significantly outperforms the other discriminant method tested, namely the DNMF algorithm.

We next used the mushroom1 dataset from the Probe1 database [20]. The database consists of 8124 examples of 125 dimensions. For training, we randomly chose 200 positive and 200 negative images from each class and used the remaining 7724 samples for testing, following the protocol adopted for the INRIA database experiments. We compared our algorithm with Semi-NMF followed by SVMs [7] and DNMF followed by SVMs [9] algorithms.

Fig. 2.7 shows the accuracies achieved for various numbers of bases. It can be shown that MNMF consistently outperforms the other algorithms. The difference in the performance is more significant when only a few number of bases are used, a fact that is consistent with our findings in the experiments using the INRIA and KTH datasets. Indeed when only 5 bases are used MNMF achieves an accuracy of 98% while the Semi-NMF+SVM achieves an accuracy of 92.68% and DNMF+SVM an accuracy of 73%.

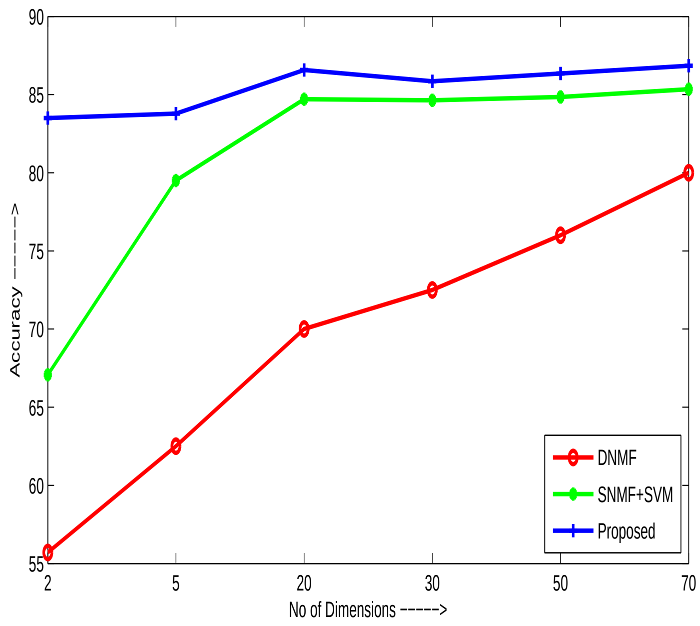


Figure 2.6: Classification accuracy vs the number of basis vectors for the KTH dataset

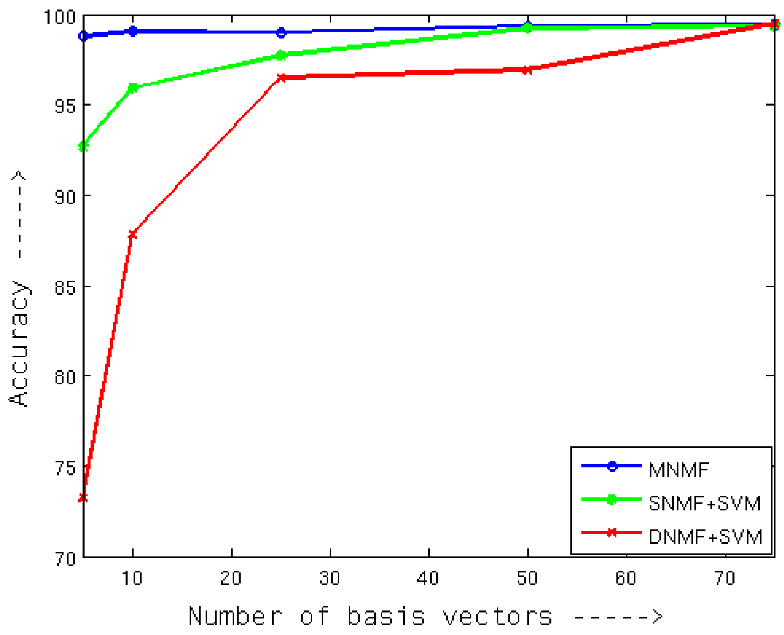


Figure 2.7: Classification accuracy vs the number of basis vectors for the mushroom1 dataset.

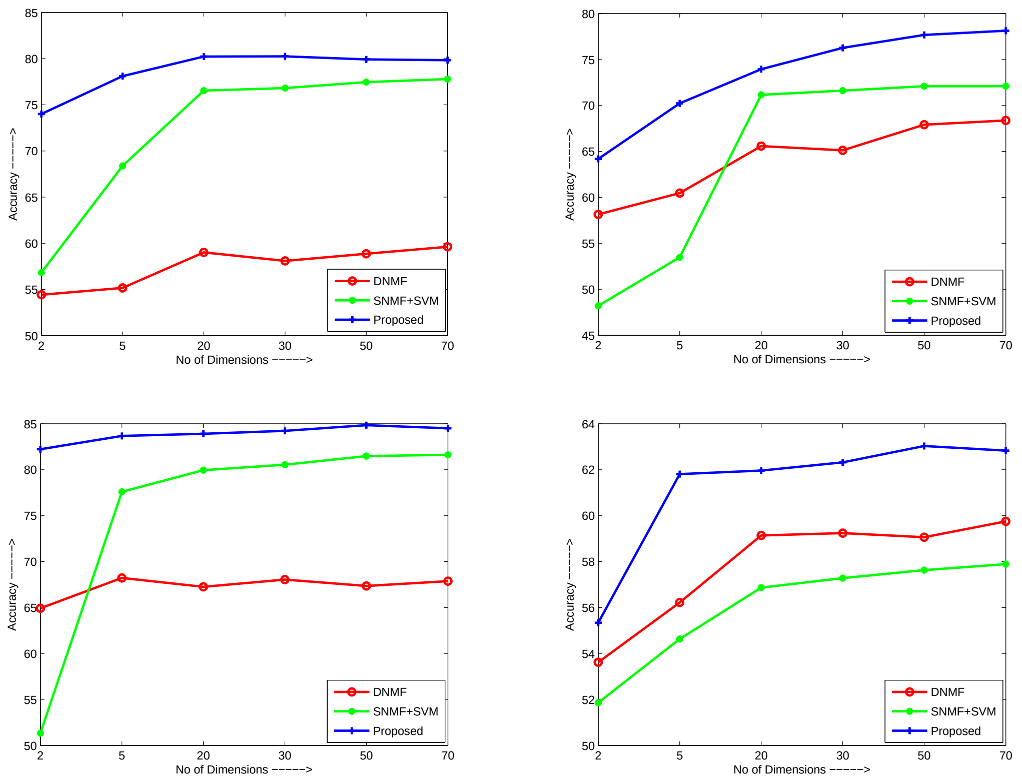


Figure 2.8: Comparison of the performance of the proposed algorithm with DNMF [27], Semi-NMF [7] + SVM on different categories of Mediamill dataset. The graph shows accuracies computed at different number of bases k.

2.4.1 Mediamill dataset

Finally we tested our algorithm on the object categories from the Mediamill [24] dataset. The dataset consists of 43907 sub-shots with 101 classes. We randomly chose two object categories from the available 101 object categories and perform a binary classification task on the chosen object categories. Fig. 2.8 shows the results obtained for four different experiments. Each image is represented using a 120-dimensional feature vector. We chose two classes at a time and used the available feature vectors of those classes as the training samples to build the classifier model. For training, we randomly chose 200 images from each classes 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.

It evident from the Fig. 2.8 that, the proposed method clearly outperforms all other methods in terms of the classification accuracy 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. In general, the relaxation of the non-negativity constraints seems to have a positive effect as the classification accuracy for the corresponding algorithms is higher.

2.5 Conclusions and Future work

In this paper we proposed a max-margin framework for Semi-NMF that introduces max-margin constraints within the NMF formulation. We simultaneously solve for the max-margin classifier and the decomposition matrices obtaining in this way a base matrix that maximizes the margin of the classifier in the projection space. We demonstrated the performance of the algorithm on a number or publicly available datasets, where it was shown that the proposed algorithm consistently outperforms the Semi-NMF and DNMF algorithms in terms of classification accuracy. Future work includes investigating towards a theoretical framework for automatically choosing the parameter λ and the extension of MNMF for non-linear cases and multiclass problems.

Chapter 3

Max-Margin Kernel Semi NMF[1]

3.1 Introduction

In this paper we introduce max-margin constraints to the objective function of non-linear version of NMF, *i.e.* Kernel NMF (KNMF) [28] to obtain a bases matrix that maximizes the classification margin of the classifier in the reconstructed feature space. In the proposed scheme we optimize a weighted combination of the reconstruction error term that is used in the typical KNMF formulations and the cost that is used in typical SVM formulations, under SVM-type linear inequality constraints. The optimization is with respect to the unknown bases and the parameters of the separating hyperplane and is solved in an iterative manner, where at each iteration we solve only for one of them 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. The proposed method is applied to publicly available databases (the KTH action and the Mediamill datasets) where we demonstrate that it consistently outperforms SVM classification schemes that use features that are extracted using KNMF [28], Semi-NMF [7], or DNMF [27].

Summarizing, the main contributions of this part are

- We introduce a max-margin framework for Kernel Semi non Negative Matrix Factorization (KSNMF).
- We propose an optimization scheme that solves for a max-margin classifier simultaneously with the decomposition matrices and bases.

[1]The text appear as submitted to ICCV2011, manuscript id:1403

The rest of the paper is organized as follows. In Section 3.2 we briefly overview the Kernel NMF. In Section 3.3, we formulate the proposed max-margin framework for semi-NMF as an optimization problem and describe an algorithm to solve it. We demonstrate the performance of the proposed algorithm in Section 3.4 and, we draw conclusions in Section 3.5.

Most of the NMF methods in the literature are linear in nature and cannot capture the non-linear structure inherent in the data. Zhang *et al.* [28] proposed the kernel extension of NMF (KNMF) that significantly improves the performance over NMF in classification applications. We briefly describe the KNMF method in the following section.

3.2 Overview of KNMF

Let Φ denote a non-linear transformation that maps data $\mathbf{x} \in \mathbb{R}^m$ in the input space to a higher dimensional feature space, *i.e.* $\Phi : \mathbf{x} \in \mathbb{R}^m \rightarrow \Phi(\mathbf{x}) \in \mathbb{R}^f$, typically $f \gg m$. Let $\Phi(\mathbf{X}) = [\Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \cdots \Phi(\mathbf{x}_n)]$ denote the data matrix where each example $\Phi(\mathbf{x}_i) \in \mathbb{R}^f$. KNMF decomposes the data matrix as

$$\Phi(\mathbf{X}) \approx \mathbf{G}_\Phi \mathbf{H} \tag{3.1}$$

where the base matrix $\mathbf{G}_\Phi \in \mathbb{R}^{f \times k}$ contains the basis vectors in the feature space and the coefficients matrix $\mathbf{H} \in \mathbb{R}^{k \times n}$ indicates the contribution of each basis vector in the reconstruction of the example. In practice, the computation of $\Phi(\mathbf{X})$ and $\mathbf{G}_\Phi$ is impractical and thus kernel trick [2] is employed to efficiently compute the similarities in the feature space,

$$\mathbf{K} \approx \mathbf{YH}$$

$\mathbf{K} = \Phi^T(\mathbf{X})\Phi(\mathbf{X})$ is the kernel matrix, $\mathbf{Y} = \Phi^T(\mathbf{X})\mathbf{G}_\Phi$. The coefficient vector $\mathbf{h}_{test}$ for a test example $\mathbf{x}_{test}$ is given by,

$$\mathbf{h}_{test} = \mathbf{Y}^\dagger \mathbf{K}_{test}$$

$\mathbf{K}_{test} = \Phi^T(\mathbf{X})\Phi(\mathbf{x}_{test})$ and $\dagger$ denotes the pseudo-inverse.

3.3 Max-Margin Semi-NMF

The NMF algorithm described in [10] minimizes either the cost function defined in Eq. 2.1 or the one in Eq. 2.2, imposing at the same time non-negativity constraints on **G** and **H**. These non-negativity constraints result in a part-based representation of the data. Several variants of NMF with discriminant constraints imposed were proposed in [7,9,27]. The variations were obtained by introducing application specific discriminant constraints to the cost function. Inspired by this, we impose discriminant constraints on the cost function of KNMF. We aim at finding a set of basis vectors in the feature space that maximizes the margin of a SVM classifier in the feature space.

3.3.1 Cost Function

Let $\{\Phi(\mathbf{x}_i), y_i\}_{i=1}^{L}$ denote a set of data vectors in the feature space and their corresponding labels, where $\Phi(\mathbf{x}_i) \in \mathbb{R}^f$, $y_i \in \{-1, 1\}$. The objective is to determine a set of basis vectors that can be used to reconstruct the data in the feature space that are optimal under a max-margin classification criterion. This is accomplished by imposing constraints on the reconstructed data computed using the base matrix $\mathbf{G}_\Phi$. Let the reconstructed vector for a data example $\Phi(\mathbf{x}_j)$ is given by $\Phi(\tilde{\mathbf{x}}_j) \approx \mathbf{G}_\Phi \mathbf{h}_j$ where $\mathbf{h}_j$ is the coefficient vector. We need to find bases that maximize the margin of the SVM classifier computed for the reconstructed data vectors. The optimization problem for the proposed criterion is given by

$$\begin{aligned} \underset{\mathbf{G}_\Phi, \mathbf{H}, \mathbf{w}_\Phi, b, \xi_i}{\operatorname{argmin}} \quad & \lambda \|\Phi(\mathbf{X}) - \mathbf{G}_\Phi \mathbf{H}\|_F^2 + \frac{1}{2}{\mathbf{w}_\Phi}^T \mathbf{w}_\Phi + C \sum_{i=1}^{L} \xi_i \\ \text{s.t.} \quad & y_i({\mathbf{w}_\Phi}^T \mathbf{G}_\Phi \mathbf{h}_i + b) \geqslant 1 - \xi_i \\ & \xi_i > 0, \forall \;\; 1 \le i \le L, \;\; \mathbf{H} \ge 0 \end{aligned} \tag{3.2}$$

where $\Phi(\mathbf{X}) = \{\Phi(\mathbf{x}_i)\}_{i=1}^{L}$, λ and C are positive constants and λ is the weight factor for the KNMF cost. The first term in the above optimization problem corresponds to the KNMF reconstruction error while the remaining terms correspond to the maximum margin classifier. The above formulation aims at maximizing the margin of the support vectors while at the same time minimizing the reconstruction and misclassification error. The classifier is trained on the reconstructed data points $\mathbf{G}_\Phi \mathbf{h}$, obtaining in this way the hyperplane parameter $\mathbf{w}_\Phi \in \mathbb{R}^f$. We iteratively solve for one of the terms $\mathbf{G}_\Phi$, $\mathbf{H}$ and $\mathbf{w}_\Phi, b, \xi_i$ by keeping the remaining parameters fixed.

Let us note that since the columns of the base matrix $\mathbf{G}_\Phi$, data matrix $\Phi(\mathbf{X})$ and the SVM hyperplane parameter $\mathbf{w}_\Phi$ are in the feature space, they are not explicitly computed. Instead, we solve explicitly for the parameters of the dual formulations of the constrained optimization problems to which we arrive and use them in order to calculate quantities that are dot products in the feature space. More specifically, when we implicitly solve for $\mathbf{G}_\Phi$, we explicitly calculate $\mathbf{G}_\Phi^T\mathbf{G}_\Phi$ and $\mathbf{G}_\Phi^T\Phi(\mathbf{X})$, and when we implicitly solve for the max-margin hyperplane $\mathbf{w}_\Phi$ we explicitly calculate $\mathbf{w}_\Phi^T\mathbf{w}_\Phi$ and $\mathbf{w}_\Phi^T\Phi(\mathbf{X})$. To compute the data kernel matrix $\Phi(\mathbf{X})^T\Phi(\mathbf{X})$ we use the Gaussian kernel [2],

$$k(\mathbf{x},\mathbf{y}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\right) \tag{3.3}$$

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

Algorithm 2: Algorithm for Max-Margin Semi-KNMF

```
input  : $\mathbf{X}$, $\mathbf{H}init$, $MAXITER$, $\lambda$, $C$,$\sigma$
output: $\mathbf{G}_\Phi^T\mathbf{G}_\Phi$, $\mathbf{H}$, $\mathbf{w}_\Phi$, $b$
begin
    $\mathbf{H} = \mathbf{H}init$;
    repeat
        $S1$ : Solve for $\boldsymbol{\alpha}$ in Eq. 3.8
        $S2$ : Compute $\mathbf{G}_\Phi^T\mathbf{G}_\Phi$ using the values of $\boldsymbol{\alpha}$ as        in Eq. 3.10
        $S3$ : Compute the kernel matrix $\mathbf{H}^T\mathbf{G}_\Phi^T\mathbf{G}_\Phi\mathbf{H}$ $S4$ : Use the computed kernel
        matrix to find          the classifier parameters, $\mathbf{w}_\Phi$, $b$.
        $S5$ : foreach column of $\mathbf{H}$ do
            Calculate $\boldsymbol{\gamma}$ using Eq. 3.18
            Compute $\mathbf{h}$ using Eq. 3.17
        end
    until $iter \leq MAXITER$   or   $convergence$;
end
```

Solve for $\mathbf{G}_\Phi$ by keeping $\mathbf{H}, \mathbf{w}_\Phi$ and b fixed: Since $\mathbf{w}_\Phi$ is fixed, the optimization problem in Eq. 3.2 is simplified as

$$\begin{aligned}\underset{\mathbf{G}_\Phi,\xi_i}{\operatorname{argmin}} \quad & \lambda\|\Phi(\mathbf{X}) - \mathbf{G}_\Phi\mathbf{H}\|_F^2 + C\sum_{i=1}^{L}\xi_i \\ & \text{s.t. } y_i({\mathbf{w}_\Phi}^T\mathbf{G}_\Phi\mathbf{h}_i + b) \geqslant 1 - \xi_i \\ & \xi_i > 0, \forall \ 1 \leq i \leq L\end{aligned} \tag{3.4}$$

The above formulation 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 (2nd term). Hence we want to find a set of bases $\mathbf{G}_\Phi$ that simultaneously reduce the reconstruction error and the misclassification. Note, that we arrived at a cost function that is quadratic or linear with respect to the unknowns and at linear inequality constraints. We propose solving the above problem using its dual formulation. The Lagrangian of Eq. 3.4 is given by

$$\begin{aligned}& L(\mathbf{G}_\Phi, \xi_i, \alpha_i, \beta_i) = \\ & \lambda Tr\Big(\big(\Phi(\mathbf{X}) - \mathbf{G}_\Phi\mathbf{H}\big)\big(\Phi(\mathbf{X}) - \mathbf{G}_\Phi\mathbf{H}\big)^T\Big) + \\ & C\sum_{i=1}^{L}\xi_i - \sum_{i=1}^{L}\alpha_i\left[y_i({\mathbf{w}_\Phi}^T\mathbf{G}_\Phi\mathbf{h}_i) + b) - 1 + \xi_i\right] - \sum_{i=1}^{L}\beta_i\xi_i \\ & \qquad \alpha_i, \beta_i > 0 \ \ \forall \ \ 1 \leq i \leq L\end{aligned} \tag{3.5}$$

where α_i, β_i are the Lagrangian multipliers. Taking the derivative w.r. to the primal variables and equating to 0, we have

$$\frac{\partial L}{\partial \mathbf{G}_\Phi} = -2\Phi(\mathbf{X})\mathbf{H}^T + 2\mathbf{G}_\Phi\mathbf{H}\mathbf{H}^T - \sum_{i=1}^{L}\alpha_i y_i \mathbf{w}_\Phi\mathbf{h}_i^T = 0$$

$$\Rightarrow \quad \mathbf{G}_\Phi = \left(2\Phi(\mathbf{X})\mathbf{H}^T + \sum_{i=1}^{L}\alpha_i y_i \mathbf{w}_\Phi\mathbf{h}_i^T\right)(2\mathbf{H}\mathbf{H}^T)^{-1} \tag{3.6}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \quad \Rightarrow \quad 0 \leq \alpha_i \leq \theta, \ \forall \ 1 \leq i \leq L \tag{3.7}$$

where $\theta = C/\lambda$. Substituting the value of $\mathbf{G}_\Phi$ in Eq. 3.5 and simplifying, we get the dual problem

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{argmax}} \;\; & \boldsymbol{\alpha}^T(\mathbf{T}_1 - \mathbf{T}_2)\boldsymbol{\alpha} + (\mathbf{t}_3 - \mathbf{t}_4 - \mathbf{t}_5 - \mathbf{t}_6 + \mathbf{t}_7)\boldsymbol{\alpha} \\ & \text{s.t. } 0 \le \alpha_i \le \theta \end{aligned} \tag{3.8}$$

where

$$\begin{aligned} & \boldsymbol{\alpha} \in \mathbb{R}^L, \;\; \mathbf{T}_1, \mathbf{T}_2 \in \mathbb{R}^{L\times L}\;, \;\; \mathbf{t}_3, \mathbf{t}_4, \mathbf{t}_5, \mathbf{t}_6, \mathbf{t}_7 \in \mathbb{R}^{1\times L}, \\ & \mathbf{T}_1 = \left[\sum_{k=1}^{L} y_i y_j \mathbf{h}_k^T \mathbf{B} \mathbf{h}_i \mathbf{w}_\Phi^T \mathbf{w}_\Phi \mathbf{h}_j \mathbf{B} \mathbf{h}_k\right]_{ij} \\ & \mathbf{T}_2 = \left[y_i y_j \mathbf{w}_\Phi^T \mathbf{w}_\Phi \mathbf{h}_i^T \mathbf{B} \mathbf{h}_j\right]_{ij} \\ & \mathbf{t}_3 = \left[4\sum_{k=1}^{L} y_i \mathbf{h}_k^T \mathbf{B} \mathbf{H} \Phi(\mathbf{X})^T \mathbf{w}_\Phi \mathbf{h}_i \mathbf{B} \mathbf{h}_k\right]_{1i} \\ & \mathbf{t}_4 = \left[2\sum_{k=1}^{L} y_i \mathbf{h}_k^T \mathbf{B} \mathbf{h}_i \mathbf{w}_\Phi^T \Phi(\mathbf{X})\right]_{1i} \\ & \mathbf{t}_5 = \left[2y_i \mathbf{w}^T \Phi(\mathbf{X}) \mathbf{H}^T \mathbf{B} \mathbf{h}_i\right]_{1i}, \;\; \mathbf{t}_6 = b\,[y_i]_{1i} \\ & \mathbf{t}_7 = [111\cdots1]_{1\times L}, \;\; \mathbf{B} = (2\mathbf{H}\mathbf{H}^T)^{-1} \end{aligned} \tag{3.9}$$

and $\mathbf{h}_k$ is the k^{th} column of the matrix $\mathbf{H}$.

The above problem is quadratic in $\boldsymbol{\alpha}$. Therefore conventional quadratic programming tools can be used to solve for $\boldsymbol{\alpha}$. Once $\boldsymbol{\alpha}$ is estimated we can compute

$$\begin{aligned} \mathbf{G}_\Phi^T \mathbf{G}_\Phi = B^T \Bigg(& 4\mathbf{H}\Phi(\mathbf{X})^T \Phi(\mathbf{X}) \mathbf{H}^T + 4\sum_{i=1}^{L} \boldsymbol{\alpha}_i y_i \mathbf{H} \Phi(\mathbf{X})^T \\ & \mathbf{w}_\Phi \mathbf{h}_i^T + \mathbf{w}_\Phi^T \mathbf{w}_\Phi \sum_{i=1}^{L} \sum_{j=1}^{L} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j \mathbf{h}_i \mathbf{h}_j^T \Bigg) B \end{aligned} \tag{3.10}$$

and

$$\mathbf{G}_\Phi^T \Phi(\mathbf{X}) = B\left(\mathbf{H}\Phi(\mathbf{X})^T \Phi(\mathbf{X}) + \sum_{i=1}^{L} \boldsymbol{\alpha}_i y_i \mathbf{h}_i w_\Phi^T \Phi(\mathbf{X})\right) \tag{3.11}$$

that are used in the subsequent optimization problems (e.g. Eq. 3.17 and Eq. 3.18).

The constant term θ in Eq. 3.8 is used as a tuning parameter. Large values of λ (compared to C), result in low values of θ which in turn reduces α_i. This causes the second term in Eq. 3.6 to vanish making the update rule of $\mathbf{G}_{\Phi}$ to be the one used in semi-KNMF. Hence for large values of λ, the update rule for $\mathbf{G}_{\Phi}$ tends to approach the update rule of semi-KNMF.

Solve for $\mathbf{w}_{\Phi}, b, \xi$ by keeping $\mathbf{G}_{\Phi}$ and H fixed: In Eq. 3.6, we computed the updated basis $\mathbf{G}_{\Phi}$ using quadratic programming. We now keep the basis $\mathbf{G}_{\Phi}$ and weight matrix $\mathbf{H}$ fixed and determine a hyperplane that maximizes the margin of the classifier. The features are obtained by reconstructing the data points in the feature space using updated basis matrix. Since $\mathbf{G}_{\Phi}$ and $\mathbf{H}$ are fixed, the optimization problem in Eq. 3.2 is simplified to that of a classical SVM:

$$\begin{aligned}
&\underset{\mathbf{w}_{\Phi}, b, \xi_i}{\operatorname{argmin}} \frac{1}{2}\mathbf{w}_{\Phi}^T\mathbf{w}_{\Phi} + C\sum_{i=1}^{L}\xi_i \\
&\text{s.t. } \; y_i(\mathbf{w}_{\Phi}^T\mathbf{G}_{\Phi}\mathbf{h}_i + b) \geqslant 1 - \xi_i \\
&\xi_i > 0, \forall \;\; 1 \leq i \leq L.
\end{aligned} \tag{3.12}$$

The above optimization problem intends to maximize the margin of the classifier in the feature space while reducing the misclassification error. The hyperplane parameters $\mathbf{w}_{\Phi}$, and b are obtained using a off-the-shelf SVM classifier. The later takes as input the kernel matrix in the feature space, that is $\mathbf{H}^T\mathbf{G}_{\Phi}^T\mathbf{G}_{\Phi}\mathbf{H}$. This can be calculated using the $\mathbf{G}_{\Phi}^T\mathbf{G}_{\Phi}$ that is explicitly computed in Eq. 3.10 and $\mathbf{H}$. After obtaining the support vectors and the solution of the dual formulation of the problem, we can explicitly compute $\mathbf{w}_{\Phi}^T\mathbf{w}_{\Phi}$ and $\mathbf{w}_{\Phi}^T\Phi(\mathbf{X})$.

Solve for H by keeping $\mathbf{G}_{\Phi}$, $\mathbf{w}_{\Phi}$, and b fixed: We now solve for matrix $\mathbf{H}$ by keeping all the remaining variables fixed. Since only the reconstruction error term of the optimization problem (Eq. 3.2) depends on $\mathbf{H}$, the objective function is simplified as

$$\begin{aligned}
&\underset{\mathbf{H}}{\operatorname{argmin}} \|\Phi(\mathbf{X}) - \mathbf{G}_{\Phi}\mathbf{H}\|_F^2, \\
&\text{s.t. } \; \mathbf{H} \geq 0
\end{aligned} \tag{3.13}$$

In order to find an $\mathbf{H}$ that is consistent we solve for $\mathbf{H}$ using quadratic programming. The i^{th} column of $\mathbf{H}$, $\mathbf{h}_i$ contributes only to the i^{th} data point $\Phi(\mathbf{x}_i)$ and hence the columns of $\mathbf{H}$ can be solved independently of each other. The above optimization problem can be solved using the regular update equations of NMF. Here we adopt an alternative optimization method. In particular, the objective function in Eq. 3.13 can be rewritten as

$$\sum_{i=1}^{L} \|\Phi(\mathbf{X}_i) - \mathbf{G}_\Phi \mathbf{h}_i\|_F^2 = \sum_{i=1}^{L} (\Phi(\mathbf{X}_i) - \mathbf{G}_\Phi \mathbf{h}_i)^T (\Phi(\mathbf{X}_i) - \mathbf{G}_\Phi \mathbf{h}_i) \tag{3.14}$$

where the i^{th} column of the matrix $\mathbf{H}$, denoted as $\mathbf{h}_i$ is given by

$$\underset{\mathbf{h}_i}{\operatorname{argmin}} \quad (\Phi(\mathbf{x}_i) - \mathbf{G}_\Phi \mathbf{h}_i)^T (\Phi(\mathbf{x}_i) - \mathbf{G}_\Phi \mathbf{h}_i), \quad \text{s.t. } \mathbf{h}_i \geq 0 \tag{3.15}$$

The Lagrangian of the above cost function is

$$L(\mathbf{h}_i) = (\Phi(\mathbf{x}_i) - \mathbf{G}_\Phi \mathbf{h}_i)^T (\Phi(\mathbf{x}_i) - \mathbf{G}_\Phi \mathbf{h}_i) - \boldsymbol{\gamma}^T \mathbf{h}_i, \quad \gamma > 0 \tag{3.16}$$

where $\boldsymbol{\gamma} \in \mathbb{R}^k$ is a vector of positive Lagrangian multiplier. Differentiating the above equation w.r.t. $\mathbf{h}_i$ and equating to zero, we get

$$\mathbf{h}_i = (2\mathbf{G}_\Phi^T \mathbf{G}_\Phi)^{-1} (2\mathbf{G}_\Phi^T \Phi(\mathbf{x}_i) + \boldsymbol{\gamma}) \tag{3.17}$$

The dual formulation for Eq. 3.16 is given by

$$\underset{\boldsymbol{\gamma}>0}{\operatorname{argmax}} \frac{1}{2} \boldsymbol{\gamma}^T \mathbf{M} \boldsymbol{\gamma} + 2\boldsymbol{\gamma}^T \mathbf{M} \mathbf{G}_\Phi^T \Phi(\mathbf{x}_i) \quad \text{where } \mathbf{M} = (2\mathbf{G}_\Phi^T \mathbf{G}_\Phi)^{-1} \tag{3.18}$$

The above problem is quadratic in $\boldsymbol{\gamma}$. We use a Quadratic Programming solver to solve Eq. 3.18. The weight vector $\mathbf{h}_i$ is obtained by substituting the computed value of $\boldsymbol{\gamma}$ in Eq. 3.17. This procedure is repeated for all columns of $\mathbf{H}$.

During testing, the weight vector $\mathbf{h}_t$ for the test data $\mathbf{x}_t$ is computed as $\mathbf{h}_t = (\mathbf{G}_\Phi^T \mathbf{G}_\Phi)^{-1} (\mathbf{G}_\Phi^T \Phi(\mathbf{x}_t))$ where $\mathbf{G}_\Phi^T \mathbf{G}_\Phi$ is computed in Eq. 3.10 and $\mathbf{G}_\Phi^T \Phi(\mathbf{x}_t)$ by substituting $\mathbf{X}$ with $\mathbf{x}_t$ in Eq. 3.11. Then we compute the kernel matrix between the training and test samples as $\mathbf{H}^T \mathbf{G}_\Phi^T \mathbf{G}_\Phi \mathbf{h}_t$. This matrix is input to the SVM classifier that classifies the given test data.

3.4 Experimental Results

In this section we demonstrate the performance of the proposed framework using publicly available datasets. More specifically, we use several object categories from the Mediamill dataset [24] and a subset of the KTH actions dataset [22]. To allow comparisons with previously reported methods, we use a baseline method DNMF [27], we also train SVM classifiers on the features that are exracted using Semi-NMF [7] and KNMF [28] algorithms. We show that the classification performance of the proposed scheme that jointly learns the classifier and performs matrix factorization is consistently higher, especially when only few dimensions are retained.

In our experiments, the value of σ in Eq. 3.3 was chosen to be same for both the proposed and the KNMF [28] algorithm. The parameter σ was set to be the standard deviation of the data, that is $\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i - \overline{\mathbf{x}}\|^2$ [28]. We set the parameter values $C = 100$ and $K = 10^5$ for all the experiments.

We first tested our algorithm on the object categories from the Mediamill [24] dataset. The dataset consists of 43907 sub-shots with 101 classes. We randomly chose two object categories from the available 101 object categories and perform a binary classification task on the chosen object categories. Fig. 3.1 shows the results obtained for six different

experimentss. Each image is represented using a 120-dimensional feature vector. We chose two classes at a time and used the available feature vectors of those classes as the training samples to build the classifier model. For training, we randomly chose 200 images from each classes 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 order to compare the performance of our algorithm with semi-NMF [7], and KNMF [28] 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. The classification performance for different number of bases, that is for various values of k, is summarized in Fig. 3.1. 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 the runs. For simplicity, for each value of k, we used the value of C (100) that provided the best results for the KNMF+SVM algorithm as input for all the other methods tested (including ours). 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

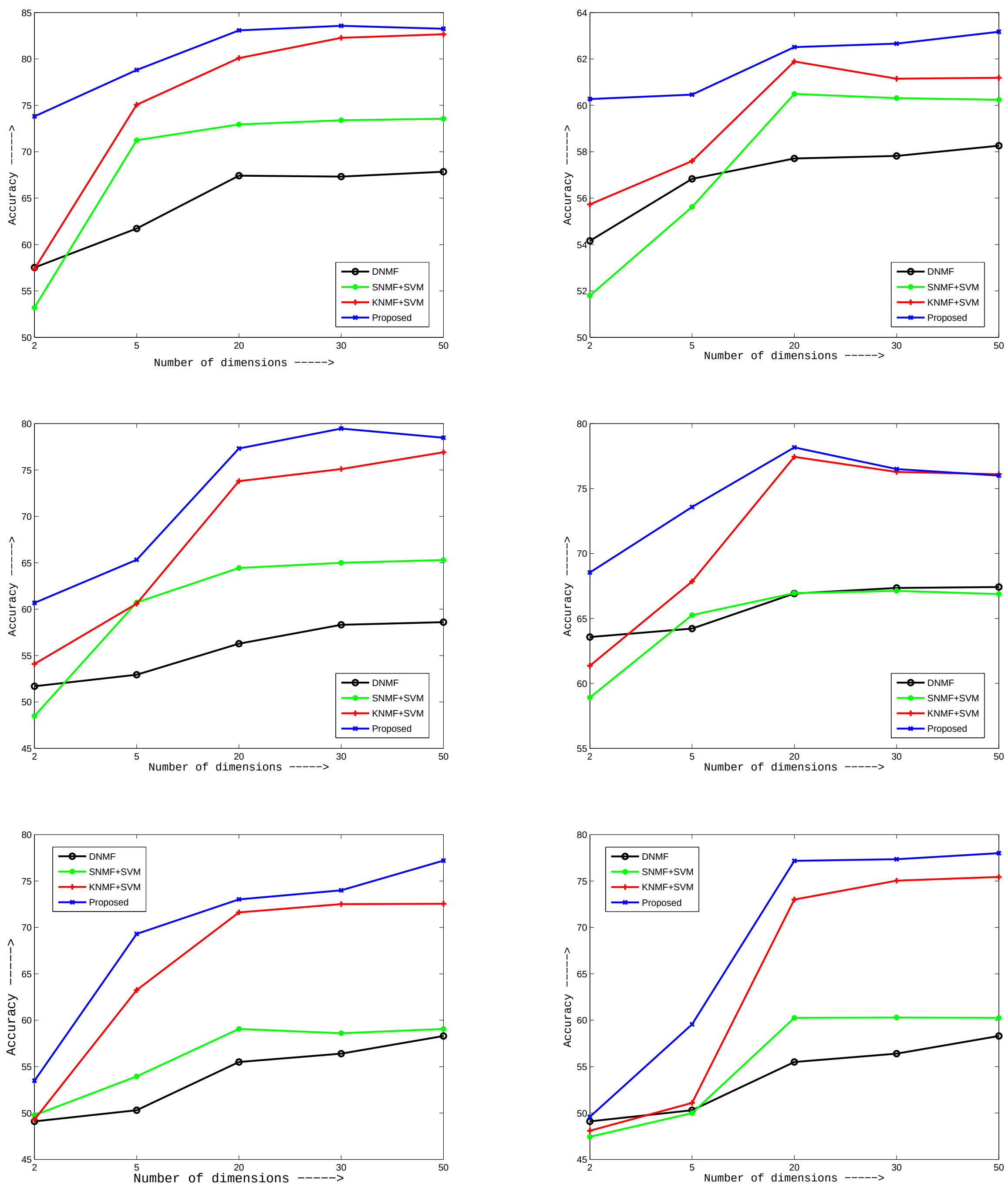


Figure 3.1: Comparison of the performance of the proposed algorithm with DNMF [27], Semi-NMF [7] + SVM, KNMF [28] + SVM on different categories of Mediamill dataset. The graph shows accuracies computed at different number of bases k. It is evident from the graph that the Proposed method significantly out-performs other methods particularly when k is low

that the proposed method significantly outperforms other methods when the number of basis vectors is small. In general, the relaxation of the non-negativity constraints seems to have a positive effect as the classification accuracy for the corresponding algorithms is higher.

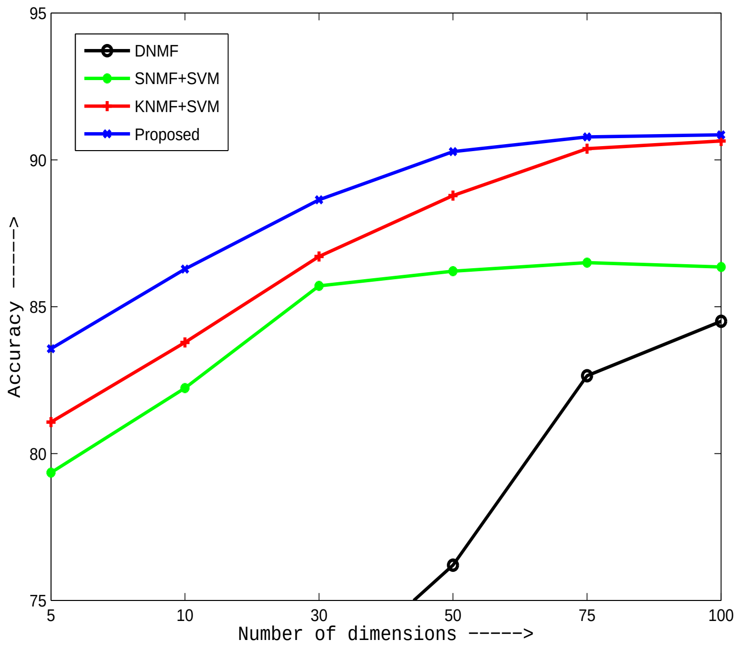


Figure 3.2: Classification accuracy vs the number of basis vectors for the KTH dataset

Subsequently, we test the proposed algorithm on the KTH dataset consisting of 25 subjects in four scenarios performing actions under various scale and illumination scenarios. We consider only the videos corresponding to two similar action categories namely *walking* and *running*. In order to handle possible illumination changes we used as features Histogram of Oriented Gradients (HOGs). The HOGs were extracted by creating a non-overlapping spatial grid size of 8×8 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. 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. For the experiments we ignore the temporal information and consider only the spatial one. A period containing 9 naively chosen frames was extracted for every image sequence of every action. In order to have a better alignment for the training data we extracted bounding boxes of size 60×80 around the objects and we also calculated the HoG features from these images following the same procedure as the one used for the INRIA database. In total we extracted 900 images for each action, 200 of which were used as training data and the remaining as test samples.

We compared our algorithm with Semi-NMF, followed by SVMs [7], KNMF Semi-NMF, followed by SVMs [28] and DNMF followed by SVMs [9] algorithms. Fig. 3.2 shows the accuracies achieved when various numbers of bases were used. For low values of k, Semi-NMF and MNMF exhibit similar performance. However, as the number of bases increases, we outperform Semi-NMF. We should also note here that the proposed algorithm significantly outperforms the other discriminant method tested, namely the DNMF algorithm.

3.5 Conclusions and Future work

In this paper we proposed a max-margin framework for Kernel Semi-NMF that introduces max-margin constraints within the KNMF formulation. We simultaneously solve for the max-margin classifier and the decomposition matrices obtaining in this way a base matrix that maximizes the margin of the classifier in the reconstructed space. We demonstrated the performance of the algorithm on a number or publicly available datasets, where it was shown that the proposed algorithm consistently outperforms the Semi-NMF and DNMF algorithms in terms of classification accuracy. Future work includes investigating towards a theoretical framework for automatically choosing the parameter λ and the extension to multiclass problems.

Bibliography

[1] A. Agarwal and B. Triggs. A local basis representation for estimating human pose from cluttered images. In *ACCV*, pages 50–59, 2006.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, 2006.

[3] C. Boutsidis and E. Gallopoulos. Svd based initialization: A head start for non-negative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008.

[4] D. Cai, X. He, X. Wu, and J. Hans. Non-negative matrix factorization on manifold. In *ICDM*, 2008.

[5] X. Chen, L. Gu, S. Z. Li, and H.-J. Zhang. Learning representative local features for face detection. In *CVPR*, volume 1, pages 1126–1131, 2001.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005.

[7] C. H. Q. Ding, T. Li, and M. I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):45–55, 2010.

[8] P. O. Hoyer. Non-negative sparse coding. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565, 2002.

[9] I. Kotsia, S. Zafeiriou, and I. Pitas. A novel discriminant non-negative matrix factorization algorithm with applications to facial image characterization problems. *IEEE Transactions on Information Forensics and Security*, 2(3):588–595, 2007.

[10] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[11] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[12] S. Z. Li, X. W. Hou, H. J. Zhang, and Q. S. Cheng. Learning spatially localized, parts-based representation. In *CVPR*, volume 1, pages 207–212, 2001.

[13] C. J. lin. Projected gradient methods for non-negative matrix factorization, 2005. Tech. Rep., Department of Computer Science, National Taiwan University.

[14] C. J. lin. On the convergence of multiplicative update algorithms for non-negative matrix factorization. pages 1589–1596, 2007. IEEE Transactions on Neural Networks.

[15] W. Liu and N. Zhen. Non-negative matrix factorization based methods for object recognition. In *Pattern Recognition Letters*, volume 25, pages 893–897, 2004.

[16] W. Liu, N. Zheng, and X. Lu. Non-negative matrix factorization for visual coding. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 293–296, 2003.

[17] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[18] P. Paatero and U. Tapper. Least squares formulation of robust non-negative factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 37:23–35, 1997.

[19] A. Pascual-Montano, J. Carazo, K. Kochi, D. Lehmann, and R. D. Pascual-Marqui. Nonsmooth nonnegative matrix factorization (nsnmf). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):403–415, 2006.

[20] L. Prechelt. Proben1 a set of neural networks benchmark problems and benchmarking rules, 1994. Technical Report 21/94, Universitat Karlsruhe, Germany.

[21] P. M. Roth, T. Mauthner, I. Khan, and H. Bischof. Efficient human action recognition by cascaded linear classification. In *IEEE Workshop on Video-Oriented Object and Event Classification*, 2009.

[22] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. *ICPR*, 2004.

[23] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, 4(3):519–524, 1987.

[24] C. G. Snoek, M. Worring, J. C. van Gemert, J.-M. Geusebroek, and A. W. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the ACM Multimedia*, pages 421–430, 2006.

[25] C. Thurau and V. Hlavac. Pose primitive based human action recognition in videos or still images. In *CVPR*, pages 1–8, 2008.

[26] Y. Wang and Y. Jia. Fisher non-negative matrix factorization for learning local features. In *ACCV*, volume 1, pages 27–30, 2004.

[27] S. Zafeiriou, A. Tefas, I. Buciu, and I. Pitas. Exploiting discriminant information in nonnegative matrix factorization with application to frontal face verification. *IEEE Transactions on Neural Networks*, 17(3):683–695, 2006.

[28] D. Zhang, Z. Zhou, and S. Chen. Non-negative matrix factorization on kernels. In *PRICAI*, pages 404–412, 2006.