Non-negative Matrix Factorization as a Feature Selection Tool for Maximum Margin Classifiers

Mithun Das Gupta
GE Global Research,
Bangalore, India.
mithun.gupta@ge.com *

Jing Xiao
Epson Research and Development Inc.
San Jose, CA.
xiaoj@erd.epson.com

Abstract

Non-negative matrix factorization (NMF) has previously been shown to be a useful decomposition tool for multivariate data. Non-negative bases allow strictly additive combinations which have been shown to be part-based as well as relatively sparse. We pursue a discriminative decomposition by coupling NMF objective with a maximum margin classifier, specifically a support vector machine (SVM). Conversely, we propose an NMF based regularizer for SVM. We formulate the joint update equations and propose a new method which identifies the decomposition as well as the classification parameters. We present classification results on synthetic as well as real datasets.

1. Introduction

Non-negative matrix factorization (NMF) has been shown to be useful for many applications in pattern recognition, multimedia, text mining, and DNA gene expressions [6, 17, 23, 31]. The work of Lee and Seung [15, 16] brought much attention to NMF in machine learning and data mining fields. Various extensions and variations of NMF have been proposed recently [18, 8]. NMF, in its most general form, can be described by the following factorization

$$X^{d\times N} = W^{d\times r} H^{r\times N} + E \quad (1)$$

where d is the dimension of the data, N is the number of data points (usually more than d) and $r < d$, E is the error and X, W, H are non-negative. Generally, this factorization has been compared with data decomposition techniques. In this sense W is the set of r domain specific basis functions and H is data specific reconstruction weights. It has been claimed by numerous researchers that such a decomposition has some favorable properties over other similar decompositions, such as PCA etc. One of the most useful properties of NMF is that it usually produces a sparse representation of the data. Such a representation encodes much of the data using few active components, which makes the encoding easy to interpret.

One of the most important shortcomings of NMF, as identified by numerous authors, is the fact that NMF doesn't consider inherent domain knowledge embedded in the data itself. For many real world applications, such as expression recognition etc. the knowledge of prior labels present in the data can be used to generate better discriminative decompositions. This knowledge is completely ignored by generic NMF and as such many new variants such as non-negative graph embedding (NGE) [32], multiplicative NGE [29], Fisher NMF [30] etc. have been proposed in the recent past.

On the other side of the spectrum, consider the case when we have the data, belonging to two classes, residing in a space which is linearly separable by a maximum margin classifier like support vector machine (SVM). This data is then projected onto a similar dimensional space which skews the data. Such an example is shown in Fig. 1. As a direct consequence of the projection the classification accuracy of SVM's declines drastically. NMF decomposition has the potential to revert the projection step, by identifying the additive basis as well as the features. Hence, a decomposition followed by classification promises to alleviate some of the problems faced by the classifier in the original space. It can be argued in the defense of SVM's that non-linear projections techniques, such as kernels, can be used to compensate for the non-separability of the data. Also, more recent variants of SVM, like the relative margin machine (RMM) [27], propose additional constraints on the SVM cost function to answer some related problems. But the above mentioned dilemma points to a problem non the less.

In this work, we propose a model which tries to answer both the above mentioned problems in a joint framework. The focus of this work is to identify a common update scheme which appreciates both the decomposition as well as the classification task. Traditionally, these two aspects of

*This work was conducted while the author was working for Epson Research and Development Inc.

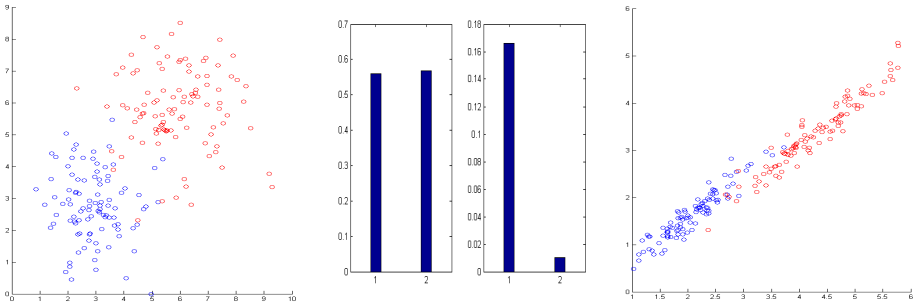

Figure 1. Left: Original features in 2D, center: projection basis, right: projected features in 2D.

data handling have been dealt separately. Generative models have been used to formulate data generation process, while discriminative models are widely preferred when data separation or clustering is the need. We propose a joint development of both the schemes into a coherent optimization framework. The framework presented in this paper extends the idea of joint learning of generative/discriminative models [1, 12, 13, 25].

Motivated by the recent array of work in the field of SVM like classifiers in the primal domain [4, 7], we propose to combine the cost function of the NMF decomposition, as well as the primal formulation for the maximum margin (MM) classifier. Our development is similar in spirit to Bradley *et al*. [2], but their work does not implicitly learn the classifier parameters with the coding parameters. We develop combined multiplicative updates (MU) for the optimization problem. Mørup *et al*. [22] have recently argued that projected gradient (PG) techniques converge faster than MU, but also agree to the fact that if MU is used as the starting point for PG techniques then the solution obtained is very close to optimal. Sha *et al*. [26] argue in favor of MU by noting the fact that the updates are free of learning rate parameter which needs to be carefully selected for most PG techniques. Also for sparse non-negative updates, MU seems to need only a non-negative starting point, whereas PG techniques require a non-linear projection step which may not be naturally enforced. Also for PG, the relative order of performing non-negative projection and unit norm projection for the basis vectors, for every iteration, is not properly defined in the literature. It seems only fair to assume that any development in the field of MU can always further the research envelope in this field.

Supervised Dictionary Learning: In classical dictionary learning tasks, one considers a signal $x \in \mathbb{R}^d$ and a fixed dictionary $\mathbf{D} = [d_1, \ldots, d_k] \in \mathbb{R}^{d,k}$ (allowing $k > d$, makes the dictionary over complete). In this setting dictionary learning leads to $\min_{\mathbf{D},\alpha\in\mathbb{R}^k} \|x - \mathbf{D}\alpha\|_2^2 + \lambda R(\alpha)$ The regularization term is usually an l_1 norm penalty, that leads to a sparse representation. Mairal *et al*. [21], took this formulation to the scenario, when the signal $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ is a collection of samples assumed to belong to one of p different classes. Their problem setup was to learn *jointly* a single dictionary D, adapted to the classification task and a function f which is essentially a classifier, separating the data into the p different classes.

$$\min_{\mathbf{D},\boldsymbol{\theta},\boldsymbol{\alpha}\in\mathbb{R}^{k,n}} \sum_{i=1}^{n} C(y_i f(x_i, \alpha_i, \boldsymbol{\theta})) + \lambda_0 \|\mathbf{X} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda_1 R(\boldsymbol{\alpha}) + \lambda_2 \|\boldsymbol{\theta}\|_2^2 \tag{2}$$

There is a subtle difference between dictionary learning and matrix decompositions. Matrix decomposition can be thought of as a special case of dictionary learning, where the size of the dictionary is constrained to be less than or equal to the observed data dimension. The work described in this paper explores the problem of matrix factorization with some proposed extensions, which can also be explored in a generic dictionary learning setup. Our work builds on the work by Mairal *et al*. [21], introducing the following new features to the problem description: (a) all the components of the decomposition, basis as well as weights should be non-negative, (b) the classification function should belong to the family of maximum margin classifiers, (c) introduction of the concept of kernels into the joint development, and (d) development of combined multiplicative update routines for all the constituents of the model.

2. NMF decomposition with Maximum Margin Classifier

Our problem takes a data matrix $\mathbf{X} \in R_+^{d,n}$, which consists of n observations which are d dimensional column vectors, and a label vector $\mathbf{Y} \in \{-1, 1\}^n$. Our goal is to find a non-negative decomposition for $\mathbf{X}$ as well learn a classifier in the decomposition space. Writing the weighted combination of NMF and maximum margin classifier, the combined

cost function which we want to minimize is

$$\min_{\mathbf{F},\mathbf{G},\mathbf{w}} \quad \gamma\|\mathbf{X}-\mathbf{FG}\|^2 + \left(\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} L(y_i, \mathbf{w}^T\mathbf{g}_i + \beta_0)\right) \tag{3}$$

$$s.\,t. \quad \mathbf{F} \in \mathbb{R}_+^{d,r},\ \mathbf{G} \in \mathbb{R}_+^{r,n},\ \mathbf{w} \in \mathbb{R}^{r,1},\ y_i \in \{-1,1\}\ \forall i$$

where $\mathbf{g}_i \in \mathbb{R}^{r,1}$ is the i^{th} column of $\mathbf{G}$, β_0 is an unknown bias, C is some constant and $L(y,t)$ is the loss function for the classifier. We interpret the decomposition such that the columns of $\mathbf{F}$ behave as the basis, and the columns of $\mathbf{G}$ are the corresponding features, which generate the points in the observation $\mathbf{X}$. By learning the classifier in the sense defined above, we are essentially transforming the classification task from the domain $\{\mathbf{X},\mathbf{Y}\}$ to $\{\mathbf{G},\mathbf{Y}\}$, while also finding the basis $\mathbf{F}$. The loss function is chosen to be of the form $L(y,t) = \max(0, 1-yt)^p$ and γ is some constant which distributes the relative importance of the two terms in the optimization. This can be identified as the relative weighting between the generative component and the discriminative component of the cost function.

The cost function is not jointly convex for all the unknowns. For any one unknown, with all the other unknowns held constant, the cost function is a convex quadratic function (for $p = 1, 2$). We would like to point out that the optimization for $\mathbf{F}$ is exactly similar to simple NMF and hence we keep the multiplicative update by Lee and Seung.

$$\mathbf{F}^{n+1} = \mathbf{F}^n \odot \frac{\mathbf{X}\mathbf{G}^T}{(\mathbf{F}\mathbf{G})\mathbf{G}^T} \tag{4}$$

Subsequently, we need to find update equations for $\mathbf{G}$ as well as the weight vector $\mathbf{w} \in \mathbb{R}^d$ together with the bias β_0.

Traditional solution for SVM classifiers is generally obtained in the dual domain. One problem with writing the dual for our formulation is the inherent coupling of the weight vector $\mathbf{w}$ and the features $\mathbf{G}$. Since $\mathbf{G}$ is no longer a constant in our formulation, hence the dual formulation becomes complicated and involved. Borrowing from the development mentioned by Chapelle [4], we replace $\mathbf{w}$, with a functional form $f(\mathbf{x}) = \sum_{i=1}^{n} \beta_i k(\mathbf{x}_i, \mathbf{x})$ where $k(\mathbf{x},\mathbf{y})$ is a kernel as given by the the representer theorem [11]. Introducing this formulation into our cost function, with a temporary suppression of the bias term β_0, yields the modified cost function

$$\min_{\mathbf{F},\mathbf{G}\geq 0,\boldsymbol{\beta}} \quad \gamma\|\mathbf{X}-\mathbf{FG}\|^2 + \lambda\sum_{i,j=1}^{n}\beta_i\beta_j k(\mathbf{g}_i,\mathbf{g}_j) + \sum_{i=1}^{n} L(y_i, \sum_{j=1}^{n} k(\mathbf{g}_j,\mathbf{g}_i)\beta_j) \tag{5}$$

where $\lambda = 1/C$ is the relative weighting between the loss function and the margin and γ is the relative weighting between the generative and the discriminative component. It can be easily observed that the second and third terms in the above equation are very similar in form to the dual expression solved for SVM.

Writing the kernel matrix $\mathbf{K}$, such that $K_{ij} = k(\mathbf{g}_i, \mathbf{g}_j)$, and $\mathbf{k}_i$ the i^{th} column of $\mathbf{K}$, we get

$$\underbrace{\min_{\mathbf{F},\mathbf{G}\geq 0,\boldsymbol{\beta}} \gamma\|\mathbf{X}-\mathbf{FG}\|^2 + \lambda\boldsymbol{\beta}^T\mathbf{K}\boldsymbol{\beta} + \sum_{i=1}^{n} L(y_i, \mathbf{k}_i^T\boldsymbol{\beta})}_{F(\mathbf{F},\mathbf{G},\boldsymbol{\beta})} \tag{6}$$

Writing the first order gradient with respect to $\boldsymbol{\beta}$, we get $\nabla_{\boldsymbol{\beta}} = (2\lambda\mathbf{K}\boldsymbol{\beta} + \sum_{i=1}^{n}\mathbf{k}_i\frac{\partial L}{\partial t}|_{t=\mathbf{k}_i^T\boldsymbol{\beta}})$. Similar to Chapelle [4], we call a point $\mathbf{g}_i$ a support vector when $y_i f(\mathbf{g}_i) < 1$, i.e. a non-zero loss for this point is encountered. WLOG, after re-ordering the training points such that the first n_{sv} points are the support vectors, the gradient and the Hessian, with respect to $\boldsymbol{\beta}$ can be written as

$$\nabla_{\boldsymbol{\beta}} = 2(\lambda\mathbf{K}\boldsymbol{\beta} + \mathbf{K}\mathbf{I}^0(\mathbf{K}\boldsymbol{\beta} - Y)), \tag{7}$$

$$H_{\boldsymbol{\beta}} = 2(\lambda\mathbf{K} + \mathbf{K}\mathbf{I}^0\mathbf{K}) \tag{8}$$

where $\mathbf{I}^0 = \begin{bmatrix} \mathbf{I}_{sv} & 0 \\ 0 & 0 \end{bmatrix}_{n\times n}$. The Newton step for $\boldsymbol{\beta}$ can now be written as $\boldsymbol{\beta} \Leftarrow \boldsymbol{\beta} - \eta H_{\boldsymbol{\beta}}^{-1}\nabla_{\boldsymbol{\beta}}$, where η is the Newton step size, which we keep fixed at 1. For the update of the vector $\boldsymbol{\beta}$, we note that from Eq. (7), we can write the update vector as

$$\boldsymbol{\beta} = (\lambda\mathbf{K} + \mathbf{K}\mathbf{I}^0\mathbf{K})^{-1}\mathbf{K}\mathbf{I}^0\mathbf{Y} \tag{9}$$

$$= (\lambda\mathbf{I}_n + \mathbf{I}^0\mathbf{K})^{-1}\mathbf{I}^0\mathbf{Y} = \begin{pmatrix} (\lambda\mathbf{I}_{n_{sv}} + \mathbf{K}_{sv})^{-1}\mathbf{Y}_{sv} \\ 0 \end{pmatrix}$$

where $\mathbf{I}_{n_{sv}}$ is the identity matrix of size n_{sv}, and $\mathbf{K}_{sv}, \mathbf{Y}_{sv}$ contain only the indices pertaining to the support vectors. To incorporate the bias term β_0, we solve the following system of linear equations

$$\begin{pmatrix} (\lambda\mathbf{I}_{n_{sv}} + \mathbf{K}_{sv}) & c \\ c & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{\beta} \\ \beta_0 \end{pmatrix} = \begin{pmatrix} \mathbf{Y}_{sv} \\ 0 \end{pmatrix} \tag{10}$$

where c is some constant which is of the order of some kernel statistic. The only limiting assumption in this formulation is that the kernel matrix is invertible. For the inner product kernel this assumption is not a problem, but for other kernels, it is advisable to add a small ridge to the kernel matrix.

Assuming all $\mathbf{g}_k$'s other than $\mathbf{g}_i$ are held constant and we want to minimize $F(\mathbf{g}_i)$, we write the second order Taylor expansion around a point $\mathbf{g}_i'$ as

$$F(\mathbf{g}_i) = F(\mathbf{g}_i') + (\mathbf{g}_i - \mathbf{g}_i')^T\nabla_{\mathbf{g}_i'} + (\mathbf{g}_i - \mathbf{g}_i')^T\mathbf{H}_{\mathbf{g}_i'}(\mathbf{g}_i - \mathbf{g}_i') \tag{11}$$

Next, we identify an auxiliary function such that the minimization of the auxiliary function leads to a guaranteed

minimization of the original cost function. This property is guaranteed by the construction of the auxiliary function $G(v, v')$, which has to fulfil two crucial properties that $F(v') \leq G(v, v')$ and $F(v) = G(v, v)$ for all non-negative v. Minimizing the auxiliary function G leads to a guaranteed minimization of the objective function F. Having identified such properties of auxiliary functions, the basic idea for handling quadratic costs similar to $F(\mathbf{g}_i)$ in Eq. (11), is to identify a matrix $\mathbf{H}'$, such that the difference between the second order terms $\mathbf{H}' - \mathbf{H} \succcurlyeq 0$ (semi-positive definite).

3. Generic Non-linear kernels

One of the main implications of combining the SVM cost function with NMF is the fact that the features themselves are evolving with the iterations and so is the kernel. In this section we look into the development of generic non-linear kernels of the form $K(\phi(\mathbf{x}), \phi(\mathbf{y}))$ where ϕ is some polynomial higher order mapping. For this work we choose the L_2 penalization for the loss, namely $L(y, f(\mathbf{x}_i)) = \max(0, 1 - yf(\mathbf{x}_i))^2$. Different loss functions such as the KL divergence loss [2] can be incorporated seamlessly into our work.

As mentioned earlier in Eq. (11), we consider a second order approximation of the cost function, hence the choice of the kernels should be such that the third and higher order derivatives are negligible in magnitude. Though this is a drawback theoretically, it can be easily accounted for by choosing proper kernel weighting parameters. Also note that the squared loss term in general is not differentiable, but we incorporate only the terms (support vectors) for which a loss is incurred, which is then differentiable.

3.1. Inner Product Kernel

The inner product kernel is the simplest to analyze, since the second order approximation in Eq. (11) is exact. We present the subsequent analysis for the inner product kernel, specifically $k(\mathbf{g}_i, \mathbf{g}_j) = \mathbf{g}_i^T \mathbf{g}_j$. For this kernel the gradient and the Hessian can be written as

$$\nabla_{\mathbf{g}_i} = \overbrace{-2\gamma \mathbf{F}^T \mathbf{x}_i + 2\gamma(\mathbf{F}^T\mathbf{F})\mathbf{g}_i}^{\triangle(\mathbf{F}, \mathbf{x}_i, \mathbf{g}_i)} + \lambda 2\beta_i \sum_{j=1}^{n} \beta_j \mathbf{g}_j \quad (12)$$
$$+2\left(\sum_{j=1}^{n_{sv}} l_j \beta_j \mathbf{g}_j [i \in n_{sv}] + \beta_i \sum_{j=1}^{n_{sv}} l_j \mathbf{g}_j\right)$$
$$\mathbf{H}_{\mathbf{g}_i} = 2\gamma(\mathbf{F}^T\mathbf{F}) + (2\lambda\beta_i^2 + 4l_i\beta_i[i \in n_{sv}])\mathbf{I}_n \quad (13)$$

where $\mathbf{I}_n$ is the identity matrix of size n and $[i \in n_{sv}]$ is an indicator function indicating that the term is present in the gradient only when the index i belongs to the set of support vectors. Note that $\triangle(\mathbf{F}, \mathbf{x}_i, \mathbf{g}_i)$ denotes the gradient obtained by simple NMF. Consequently, we need to find an upper bound for the Hessian, noting that the last term $4l_i\beta_i$ is unbounded. Also note that for an index i which is *not* a support vector, the Hessian is already positive definite. Using the triangle inequality we can bound $4l_i\beta_i \leq (l_i + \beta_i)^2 \leq 2(l_i^2 + \beta_i^2)$. Using this we can write the auxiliary function as

$$G(\mathbf{g}_i, \mathbf{g}_i') = F(\mathbf{g}_i') + (\mathbf{g}_i - \mathbf{g}_i')^T \nabla_{\mathbf{g}_i'} \quad (14)$$
$$+(\mathbf{g}_i - \mathbf{g}_i')^T \mathbf{D}_{\mathbf{g}_i'} (\mathbf{g}_i - \mathbf{g}_i')$$
$$\mathbf{D}_{\mathbf{g}_i'} = \text{diag}\left(\frac{\mathbf{A}_{\mathbf{g}_i'}\mathbf{g}_i'}{\mathbf{g}_i'}\right), \quad (15)$$
$$A_{\mathbf{g}_i'} = 2\gamma(\mathbf{F}^T\mathbf{F}) + 2\left(\lambda\beta_i^2 + (\beta_i^2 + l_i^2)[i \in n_{sv}]\right)\mathbf{I}_n \quad (16)$$

which brings us to the following lemmas:

Lemma 1 *Let* $\mathbf{Q}$ *be a symmetric non-negative matrix and* $\mathbf{v}$ *be a positive vector, then the matrix* $\hat{\mathbf{Q}} = diag\left(\frac{\mathbf{Q}\mathbf{v}}{\mathbf{v}}\right) - \mathbf{Q} \succcurlyeq 0$ *(Proved in [16]).*

Lemma 2 *The choice of the function* $G(\mathbf{g}_i, \mathbf{g}_i')$ *in Eq. (14) is a valid auxiliary function for* $F(\mathbf{g}_i)$ *in Eq. (11).*

Proof: The first condition $G(\mathbf{g}_i', \mathbf{g}_i') = F(\mathbf{g}_i')$ is obvious by simple substitution. The second condition can be obtained by proving that $\mathbf{D}_{\mathbf{g}_i'} - \mathbf{H}_{\mathbf{g}_i'} \succcurlyeq 0$.

$$\begin{aligned}
&\mathbf{D}_{\mathbf{g}_i'} - \mathbf{H}_{\mathbf{g}_i'} = \mathbf{D}_{\mathbf{g}_i'} - \mathbf{A}_{\mathbf{g}_i'} + \mathbf{A}_{\mathbf{g}_i'} - \mathbf{H}_{\mathbf{g}_i'} \\
&= \mathbf{D}_{\mathbf{g}_i'} - \mathbf{A}_{\mathbf{g}_i'} + \gamma(2\beta_i^2 + 2l_i^2 - 4l_i\beta_i)[i \in n_{sv}]\mathbf{I}_n \\
&= \underbrace{\mathbf{D}_{\mathbf{g}_i'} - \mathbf{A}_{\mathbf{g}_i'}}_{\mathbf{P}} + \underbrace{2\gamma(\beta_i - l_i)^2[i \in n_{sv}]\mathbf{I}_n}_{\mathbf{S}} \succcurlyeq 0
\end{aligned}$$

The last condition above is satisfied since the matrix $\mathbf{P} \succcurlyeq 0$, from Lemma. 1, and the second matrix $\mathbf{S}$ is a non-negative diagonal matrix which is added to $\mathbf{P}$. □

Finally the update for $\mathbf{g}_i$ can be found by evaluating $\frac{\partial G(\mathbf{g}_i, \mathbf{g}_i')}{\partial \mathbf{g}_i} = 0$, which gives

$$\begin{aligned}
\mathbf{g}_i^{k+1} &= \mathbf{g}_i^k - \mathbf{D}_{\mathbf{g}_i'}^{-1} \nabla_{\mathbf{g}_i} \quad (17) \\
&= \mathbf{g}_i^k \odot \left(\frac{\mathbf{A}_{\mathbf{g}_i^k}\mathbf{g}_i^k - \nabla_{\mathbf{g}_i^k}}{\mathbf{A}_{\mathbf{g}_i^k}\mathbf{g}_i^k}\right)
\end{aligned}$$

The above technique can be used for many different kernels. Once the matrix $A_{\mathbf{g}_i}$ in Eq. 16 is identified, the update for $\mathbf{g}_i$ follows from Eq. 17.

3.2. Implementation Details

The weighting parameter γ is one of the free parameters in our experiments. Since the features $\mathbf{G}$ at the start of the iterations are random, hence the discrimination afforded by them is very limited. As a heuristic for our joint

Table 1. Error percentages for tenfold cross-validation on UCI datasets. M = classes and D = dimension of data. The methods compared to are SVM and geometric level sets (GLS). NMFSVM is our method.

Data Set (M, D)	SVM	GLS	NMFSVM
Pima (2,8)	22.66	25.94	23.04
WDBC (2,30)	2.28	4.04	2.20
Liver (2,6)	41.72	37.61	33.09
Ionos. (2,34)	11.40	13.67	10.22

optimization scheme, similar to the heuristic in [20, 12], we start with large values of the weighting term γ. After every few steps this value is decreased by a small amount, either automatically, or by checking whether the new choice is at least as good as the previous value. Empirically we have found that decreasing γ value constantly with iterations $\gamma = \frac{\gamma_0}{(1+\epsilon)^{iterations}}$ generates robust classification accuracy across many restarts. During testing, we perform a non-negative decomposition of the test data, with constant $\mathbf{F}$ obtained from the training phase to generate $\mathbf{G}_{test}$. This update is similar to the self-taught transfer learning technique proposed in [24]. Once $\mathbf{G}_{test}$ is obtained we generate the kernel matrix $\mathbf{K}_{test} = \mathbf{G}^T\mathbf{G}_{test}$. The classification of the test data points can now be obtained by the function $sign(\mathbf{K}_{test}^T\boldsymbol{\beta} + \beta_0)$.

4. Experiments

Before presenting comparative experimental details against well established factorization methods, we diverge into the initial motivation that our technique can perform better than simple svm based classifier, at least for certain scenarios. A deeper look into our formulation will reveal that our technique is somewhat similar to the geometric level set (GLS) based regularization introduced by Varshney and Willsky [28]. They allude to the fact that squared distance is not a signed function and hence cannot be used as a level set. But they counter this drawback by normalizing the l_1 norm after every iteration. For NMF this step is already taken care of by the positivity constraint, hence NMF cost can be thought of as a pseudo level set. In essence, we extend their level set method by replacing it with the NMF factorization term and provide multiplicative updates for the model variables. Comparison based on basic classification performance for binary datasets from the UCI Machine Learning Repository[1] are presented in Table. 1. Our method (NMFSVM) beats the geometric level set method on all the binary tasks and performs comparative to normal SVM beating it in three out of four tests. In defence of the GLS technique it must be realized that this technique can be seamlessly used for multiclass problems as well, where as our technique can only perform binary classification. In defence of SVM it must be said that the vast space of kernels can be further explored to enhance its performance. But the simplistic experiment still guarantees a fair comparison for our method against the published techniques.

[1] http://archive.ics.uci.edu/ml/

4.1. Simulated Experiments

The primary aim of this section is to illustrate the underlying properties of the penalized decomposition mentioned in the previous sections. We demonstrate the efficacy of the method on a synthetic dataset which is generated as follows. We generate 3 (noisy) orthogonal basis of 7 dimensions to form the basis matrix $\mathbf{F} \in \mathbb{R}^{7\times3}$. The feature matrix $\mathbf{G} \in \mathbb{R}^{3\times400}$, is drawn from two Gaussian distributions such that

$$G = \begin{bmatrix} [\mathcal{N}(0,1)]_{3\times200} & [\mathcal{N}(5,5)]_{3\times200} \end{bmatrix}$$

where $\mathcal{N}(\mu, \sigma)$ is a Gaussian kernel with mean μ and variance σ. The labels for the points are now assigned as $Y = [-1_{200} \;\; 1_{200}]$. Finally, the data matrix is generated by $\mathbf{X} = \mathbf{FG} + \alpha$, where α is some white Gaussian noise. We project the data to the positive orthant to guarantee the non-negativity of the training data. We partition half the

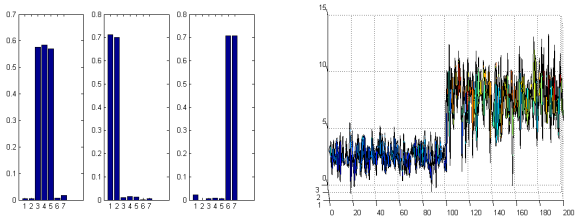

Figure 2. Left: original basis $\mathbf{F}$, right: original features $\mathbf{G}$.

points from each class for training and the other half for testing. The original basis as well as the features are shown in Fig. 2. The outputs of the training phase are the estimates for the basis $\mathbf{F}$, the features $\mathbf{G}$ (Fig. 3), and the vector $\boldsymbol{\beta}$ which has only 5 non-zero elements pointing to the support vectors. The means of the two Gaussian are shifted due to the minimum subtraction, but the variance can still point towards correct estimation of the features. After training, the variances for the features belonging to the two classes were around 1.2, and 5.8. The misclassification error for testing is around 2%.

In the next set of experiments, we work with real waveforms, similar to the experiments proposed by Cichocki *et al.* [5]. The feature matrix $\mathbf{G}$ used to generate the observations is similar to the first experiment. The constituent bases as well as the bases found by our decomposition scheme are shown in Fig. 4. The classification accuracy for training is around 98%, and for testing is around 96.5%. The variances estimated for the two Gaussian kernels, used to generate the feature matrix $\mathbf{G}$, are 1.07 and 5.09 for training and 1.17 and 5.84 for testing (true values being 1 and 5).

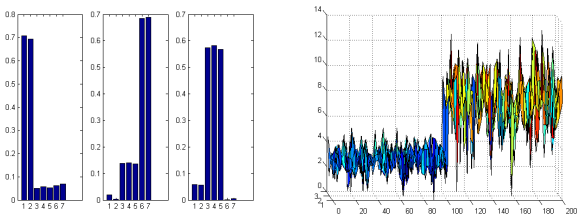

Figure 3. Left: learned basis $\mathbf{F}$, right: learned features $\mathbf{G}$.

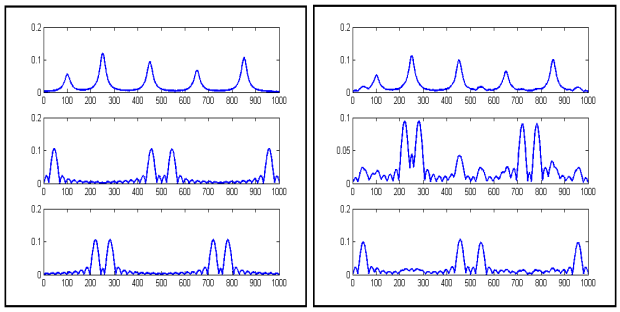

Figure 4. Left: original basis, right: learned basis $\mathbf{F}$.

Table 2. Error rates on the MNIST and USPS datasets.

Algorithm	MNIST	USPS
REC-L	2.83	3.76
SDL-G	3.56	6.67
SDL-D	1.05	3.54
k-NN l_2	5.0	5.2
NMFSVM	**1.02**	**3.01**

4.2. Digits Recognition

In this section, we present experiments on the popular MNIST [14] and USPS handwritten digit datasets. MNIST is composed of 70,000, 28×28 images, 60,000 for training, 10,000 for testing, each of them containing one handwritten digit. USPS is composed of 7291 training images and 2007 test images of size 16×16. As is often done in classification, we have chosen to learn pairwise binary classifiers, one for each pair of digits. Five-fold cross-validation is performed to find the average error rate. For a given image $\mathbf{x}$, the test procedure consists of selecting the class which receives the most votes from the pairwise classifiers. We compare against the work of Mairal *et al*. [21], since their work is closest in principle to the work presented in this paper. They present results for linear as well as bilinear model. We compare against the linear model only, since it has lower reported error rates. For the results presented in Table 2, **REC-L** stands for learning a reconstructive dictionary D (also similar to [24]) and then learning the parameters of the classifier a posteriori, **k-NN** l_2 stands for k-nearest neighbor with Euclidean distance metric, **SDL-G** stands for supervised dictionary learning with generative training, and **SDL-D** stands for supervised dictionary learning with discriminative training, with explicit inclusion of all label states [21]. Our method, denoted as **NMFSVM**, outperforms all other techniques. Note that our method is competitive since the best error rates published on these datasets are 0.60% [14] for MNIST and 2.4% [9] for USPS, using methods tailored to these tasks, whereas our method is generic and has not been tuned for the handwritten digit classification domain. Fig. 5 shows the discriminative bases for digit 9 vs all others. We compare the dictionary learned by our technique to the supervised dictionary learning with discriminative training (SDL-D) [21]. The bases have been scaled for proper display. The structural similarity to the digits is more striking in the bases obtained by our technique.

Table 3. Error rates for the texture recognition task.

Algorithm	m=500	m=1500
REC-L	42.18	38.82
SDL-G	47.34	46.30
SDL-D	44.84	42.00
NMFSVM	**32.02**	**28.01**

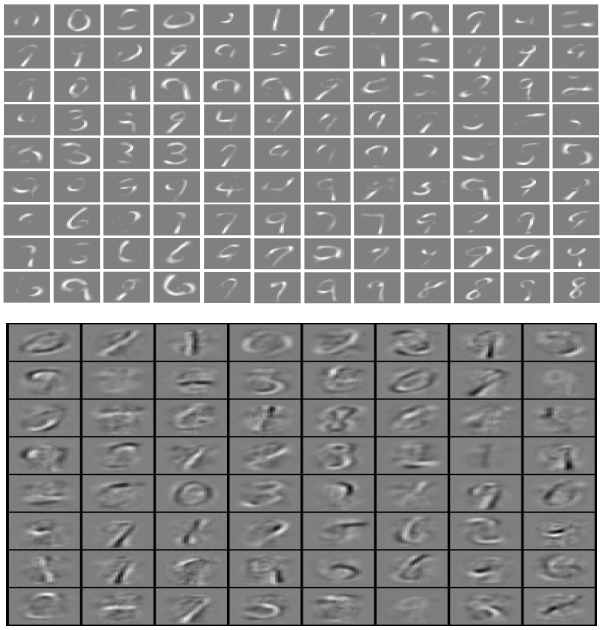

Figure 5. Discriminative bases for digit 9 vs all for MNIST dataset. Top: our technique, bottom: SDL-D [21].

4.3. Texture classification

Formulating a similar experiment to Mairal *et al*. [21], we choose two texture images from the Brodatz data-set, presented in Fig. 6, and build two classes, composed of 12×12 patches taken from these two textures. We compare the classification performance of all the methods mentioned

in the previous section. The training set was composed of patches from the left half of each texture and the test sets of patches from the right half, so that there is no overlap between them in the training and test set. Error rates are reported in Table. 3 for varying sizes of the training set (m).

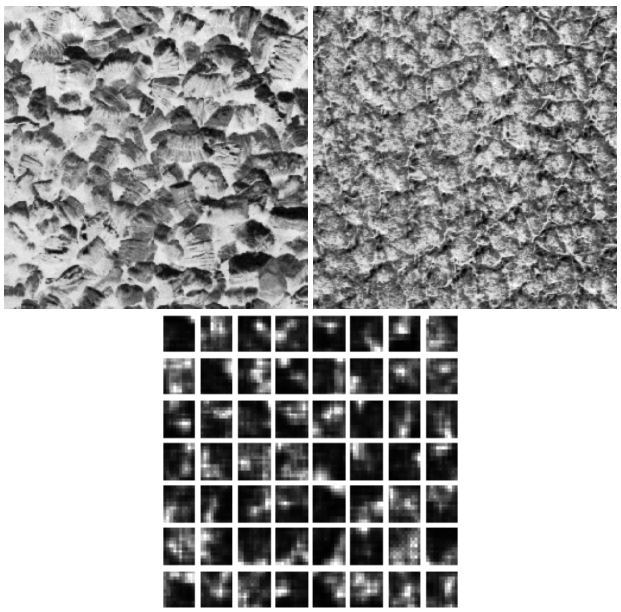

Figure 6. Top: two texture examples, bottom: the learned basis.

4.4. Expression recognition

In our experiments, we use Japanese female facial expression (JAFFE) database [19]. JAFFE database contains 213 images of Japanese female facial expressions. Each one of the 10 subjects produced 3 or 4 samples for each of 7 basic facial expressions: happiness, surprise, fear, sadness, disgust, anger, and neutral pose. After face region is cropped, each image is downsampled to an image size of 40×30 pixels. 150 images are randomly selected, such that each expression class has at least 20 images in the training set, the rest are used as test images. We perform the experiments 5 times each and report the average performance. Comparative results against well known techniques are shown in Table. 4. Also note that for all the other techniques, we need to train a separate svm classifier, but for our technique, the classifier is learned along with the factorization. The basis images obtained by our method, **NMFSVM**, are shown in Fig. 7.

5. Conclusion and Future Work

The development presented in this paper, proposes a joint solution of the problem of feature decomposition as well as margin based classification. Inner product kernel has been shown to be useful for linearly separable data. For more complex data non-linear kernels such as radial basis functions (RBF) are needed. This can be seamlessly integrated into the approach described in this paper, as explained in the supplementary material. The update for the classifier parameter $\boldsymbol{\beta}$ is obtained by solving a Newton system. The cost function mentioned in Eq. (6), is generic enough to allow additional constraints, such as orthogonality constraints, as well as controllable sparsity of the basis vectors [10]. All the recent developments within the NMF community [3, 22], which modify the update equations, can still be applied without varying the classifier loop.

Figure 7. The bases obtained by our method, NMFSVM, for the Jaffe dataset.

References

[1] D. Blei and J. McAuliffe. Supervised topic models. In *Advances in Neural Information Processing Systems 20*, pages 121–128. 2008. 2842

[2] D. Bradley and J. A. D. Bagnell. Differentiable sparse coding. In *Proceedings of Neural Information Processing Systems 22*, December 2008. 2842, 2844

[3] D. Cai, X. He, X. Wu, and J. Han. Non-negative matrix factorization on manifold. In *ICDM '08: Proc. IEEE International Conference on Data Mining*, pages 63–72, Washington, DC, USA, 2008. 2847

[4] O. Chapelle. Training a support vector machine in the primal. *Neural Comput.*, 19(5):1155–1178, 2007. 2842, 2843

[5] A. Cichocki and R. Zdunek. Multilayer nonnegative matrix factorization using projected gradient approaches. In *International Journal of Neural Systems*, volume 17, pages 431–446, 2007. 2845

[6] M. Cooper and J. Foote. Summarizing video using nonnegative similarity matrix factorization. *Proc. IEEEWorkshop on Multimedia Signal Processing*, pages 25–28, 2002. 2841

Table 4. Expression recognition accuracy on the JAFFE dataset.

Algorithm	**PCA**	**ICA**	**NMF**	**LDA**	**NGE**	**NMFSVM**
Accuracy (mean % ± var)	63.65±3.98	66.35±3.24	64.44±3.90	72.70±4.86	68.25±4.81	**73.4±0.26**

[7] N. Cristianini, C. Campbell, and J. Shawe-Taylor. Multiplicative updatings for support vector machines. In *Proceedings of European Symposium on Artificial Neural Networks (ESANN-99)*, pages 189–194, 1999. 2842

[8] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix tri-factorizations for clustering. In *Proc SIGKDD*, 2006. 2841

[9] B. Haasdonk and D. Keysers. Tangent distance kernels for support vector machines. *ICPR*, 2, 2002. 2846

[10] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. In *J. Machine Learning Research*, volume 5, pages 1457–1469, 2004. 2847

[11] G. S. Kimeldorf and G. Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. In *Annals of Mathematical Statistics*, volume 41, page 495502, 1970. 2843

[12] H. Larochelle and Y. Bengio. Classification using discriminative restricted boltzmann machines. In *ICML*, pages 536–543, New York, NY, USA, 2008. ACM. 2842, 2845

[13] J. A. Lasserre, C. M. Bishop, and T. P. Minka. Principled hybrids of generative and discriminative models. In *CVPR*, pages 87–94, Washington, DC, USA, 2006. 2842

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998. 2846

[15] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. In *Nature*, volume 401, pages 788–791, 1999. 2841

[16] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000. 2841, 2844

[17] S. Li, X. Houa, H. Zhang, and Q. Cheng. Learning spatially localized, parts-based representation. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 207–212, 2001. 2841

[18] B. Long, Z. Zhang, and P. Yu. Co-clustering by block value decomposition. In *KDD 05*, pages 635–640, 2005. 2841

[19] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba. Coding facial expressions with gabor wavelets. In *Proc. Third IEEE Int. Conf. Automatic Face and Gesture Recognition*, pages 200–205, 1998. 2847

[20] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *CVPR*, 2008. 2845

[21] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *NIPS*, pages 1033–1040. MIT Press, 2008. 2842, 2846

[22] M. Mørup and L. K. Hansen. Tuning pruning in sparse non-negative matrix factorization., 2009. 2842, 2847

[23] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. In *Environmetrics*, volume 5, pages 111–126, 1994. 2841

[24] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, 2007. 2845, 2846

[25] R. Salakhutdinov and G. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 11, 2007. 2842

[26] F. Sha, L. Saul, and D. Lee. Multiplicative updates for nonnegative quadratic programming in support vector machines. In *Advances in Neural Information Processing Systems 15*, pages 1065–1073, 2003. 2842

[27] P. K. Shivaswamy and T. Jebara. Relative margin machines. In *NIPS*, pages 1481–1488, 2008. 2841

[28] K. R. Varshney and A. S. Willsky. Classification using geometric level sets. *Journal of Machine Learning Research*, 11:491–516, February 2010. 2845

[29] C. Wang, Z. Song, S. Yan, L. Zhang, and H.-J. Zhang. Multiplicative nonnegative graph embedding. In *CVPR*, 2009. 2841

[30] Y. Wang, Y. Jiar, C. Hu, and M. Turk. Fisher nonnegative matrix factorization for learning local features. In *ACCV*, 2004. 2841

[31] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *SIGIR03*, pages 267–273, 2003. 2841

[32] J. Yang, S. Yang, Y. Fu, X. Li, and T. Huang. Non-negative graph embedding. In *CVPR*, 2008. 2841