Hindawi Publishing Corporation
EURASIP Journal on Applied Signal Processing
Volume 2006, Article ID 71817, Pages 1–8
DOI 10.1155/ASP/2006/71817

Fast Nonnegative Matrix Factorization and Its Application for Protein Fold Recognition

Oleg Okun and Helen Priisalu

Machine Vision Group, Infotech Oulu and Department of Electrical and Information Engineering, University of Oulu, P.O. Box 4500, 90014, Finland

Received 27 April 2005; Revised 29 September 2005; Accepted 8 December 2005

Linear and unsupervised dimensionality reduction via matrix factorization with nonnegativity constraints is studied. Because of these constraints, it stands apart from other linear dimensionality reduction methods. Here we explore nonnegative matrix factorization in combination with three nearest-neighbor classifiers for protein fold recognition. Since typically matrix factorization is iteratively done, convergence, can be slow. To speed up convergence, we perform feature scaling (normalization) prior to the beginning of iterations. This results in a significantly (more than 11 times) faster algorithm. Justification of why it happens is provided. Another modification of the standard nonnegative matrix factorization algorithm is concerned with combining two known techniques for mapping unseen data. This operation is typically necessary before classifying the data in low-dimensional space. Combining two mapping techniques can yield better accuracy than using either technique alone. The gains, however, depend on the state of the random number generator used for initialization of iterations, a classifier, and its parameters. In particular, when employing the best out of three classifiers and reducing the original dimensionality by around 30%, these gains can reach more than 4%, compared to the classification in the original, high-dimensional space.

1. INTRODUCTION

It is not uncommon that for certain data sets, their dimensionality n is higher than the number of attributes or features m (here and further it is assumed that the data are accumulated in an $n \times m$ matrix accomodating m n-dimensional feature vectors). In such cases, the effect, referred to as curse of dimensionality, occurs that negatively influences the clustering and classification of a given data set. Dimensionality reduction is typically used to cure or at least to mitigate this effect and can be done by means of feature extraction (FE) or feature selection (FS). FS selects a subset of the original features based on a certain criterion of feature importance or relevance whereas FE produces a set of transformed (i.e., new) features from the original ones. Features chosen by FS are easy to interpret while those found by FE may be not. In addition, FS often assumes knowledge of class membership information, that is, it is often supervised, in contrast to FE that is usually unsupervised. Thus, FS looks naturally more attractive than FE from the classification viewpoint, that is, when FS is followed by classification of a data set. However, there can be the cases where all or almost all original features turn out, to be important (relevant) so that FS becomes inadequate. If this happens, the alternative is FE, and such a case is considered in this paper.

The simplest way to reduce dimensionality is to linearly transform the original data. Given the original, high-dimensional data gathered in an $n \times m$ matrix $\mathbf{V}$, a transformed or reduced matrix $\mathbf{H}$, composed of m r-dimensional vectors ($r < n$ and often $r \ll n$), is obtained from $\mathbf{V}$ according to the following linear transformation: $\mathbf{W}$: $\mathbf{V} \approx \mathbf{WH}$ (symbol $\approx$ indicates that an exact reconstruction of the original data is unlikely to happen in general), where $\mathbf{W}$ is an $n \times r$ (basis) matrix. It is said that $\mathbf{W}$ and $\mathbf{H}$ are the factorized matrices and $\mathbf{WH}$ is a factorization of $\mathbf{V}$. Principal component analysis (PCA) [1] and independent component analysis (ICA) [2] are well-known techniques performing this operation.

Nonnegative matrix factorization (NMF) also belongs to this class of methods. Unlike the others, it is based on nonnegativity constraints on all matrices involved. Thanks to this fact, it can generate a part-based representation since no subtractions are allowed. Lee and Seung [3] proposed a simple iterative algorithm for NMF and proved its convergence. The factorized matrices are initialized with positive random numbers before starting matrix updates.

It is well known that initialization is of importance for any iterative algorithm: properly initialized, an algorithm converges faster. However, this issue was not yet investigated

in case of NMF. In order to speed up convergence, we propose to perform feature scaling (normalization) before iterations begin so as to bring values of all three matrices involved in factorization within the same range (in our case, between 0 and 1). Justification of why this change leads to faster convergence is provided.

Since dimensionality reduction is typically followed by classification in low-dimensional space, it is important to know when the error rate in this space is lower than that in the original space. Regarding classification, we propose to combine two known techniques for mapping unseen data prior to classification in low-dimensional space. For certain values of the state of the random number generator used to initialize matrices **W** and **H**, this combination results in higher accuracy in the low-dimensional space than in the original space. The gains depend not only on the state of the random number generator, but also on a classifier and its parameters.

Because of its straightforward implementation, NMF has been applied to solve many tasks: information retrieval [4, 5], object classification (faces, handwritten digits, documents) [6–15], sparse coding [16–18], speech and audio analysis and recognition [19–22], mining web logs [23], estimation of network distances between arbitrary Internet hosts [24], video summarization [25], image rendering [26], and independent component analysis [27]. Here we extend the application of NMF to bioinformatics: NMF coupled with three nearest-neighbor classifiers is applied for protein fold recognition. Experiments demonstrate that it is possible to achieve higher accuracy in the low-dimensional space of NMF, compared to the classification in the original space. For instance, when employing the best out of three classifiers and reducing the original dimensionality by around 30%, the accuracy rate can grow by more than 4%.

2. METHODS

2.1. Original nonnegative matrix factorization

Given the nonnegative matrices **V**, **W**, and **H** whose sizes are $n \times m$, $n \times r$, and $r \times m$, respectively, we aim at such factorization that $\mathbf{V} \approx \mathbf{WH}$. The value of r is selected according to the rule $r < (nm)/(n+m)$ in order to obtain data compression.[1] Each column of **W** is a basis vector while each column of **H** is a reduced representation of the corresponding column of **V**. In other words, **W** can be seen as a basis that is optimized for linear approximation of the data in **V**.

NMF provides the following simple learning rule guaranteeing monotonical convergence to a local maximum without the need for setting any adjustable parameters [3]:

$$W_{ia} \longleftarrow W_{ia} \sum_{\mu} \frac{V_{i\mu}}{(WH)_{i\mu}} H_{a\mu}, \tag{1}$$

$$W_{ia} \longleftarrow \frac{W_{ia}}{\sum_j W_{ja}}, \tag{2}$$

$$H_{a\mu} \longleftarrow H_{a\mu} \sum_{i} W_{ia} \frac{V_{i\mu}}{(WH)_{i\mu}}. \tag{3}$$

The matrices **W** and **H** are initialized with positive random values. Equations (1)–(3) iterate until convergence to a local maximum of the following objective function:[2]

$$F = \sum_{i=1}^{n} \sum_{\mu=1}^{m} \left(V_{i\mu} \log(WH)_{i\mu} - (WH)_{i\mu} \right). \tag{4}$$

In its original form, NMF can be slow to converge to a local maximum for large matrices and/or high data dimensionality. On the other hand, stopping after a predefined number of iterations, as sometimes is done, can be too premature to get a good approximation. Introducing a parameter tol ($0 < \text{tol} \ll 1$) to decide when to stop iterations significantly speeds up the convergence without negatively affecting the mean-square error (MSE), measuring the approximation quality[3]. That is, iterations stop when $F_{\text{new}} - F_{\text{old}} < \text{tol}$.

After learning the NMF basis functions, that is, the matrix **W**, new (previously unseen) data in the matrix $\mathbf{V}_{\text{new}}$ are mapped to r-dimensional space by fixing **W** and using one of the following techniques:

(1) randomly initializing **H** as described above and iterating (3) until convergence [9];
(2) as a least-squares solution of $\mathbf{V}_{\text{new}} = \mathbf{WH}_{\text{new}}$, that is, $(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{V}_{\text{new}}$ [8].

Further we will call the first technique *iterative* while the second *direct*, because the latter provides a straightforward noniterative solution. The direct technique can produce negative entries of $\mathbf{H}_{\text{new}}$, thus violating nonnegativity constraints. There are two possible remedies of this problem: (1) enforcing nonnegativity by setting negative values to zero and (2) using nonnegative least squares. Each solution has its own pros and cons. For instance, setting negative values to zero is much more computationally simpler than solving least squares with nonnegativity constraints, but some information is lost after zeroing. On the other hand, the nonnegative least-squares solution has no negative components, but it is known that it may not fit as well as the least-squares solution without nonnegativity constraints. Since our goal is to accelerate convergence, we prefer the first (zeroing) solution when employing the direct technique.

2.2. Modified nonnegative matrix factorization

We propose two modifications of the original iterative NMF algorithm.

The first modification is concerned with feature scaling (normalization) linked to the initialization of the factorized matrices. Typically, these matrices are initialized with positive random numbers, say uniformly distributed between

[1] For dimensionality reduction, it is, however, sufficient if $r < n$.

[2] See the appendix for its derivation.

[3] It was observed in numerous experiments that MSE quickly decreases after not very many iterations and after that, its rate of decrease dramatically slows down.

0 and 1, in order to satisfy the nonnegativity constraints. Hence, elements of **V** (matrix of the original data) also need to be within the same range. Given that V_j is an n-dimensional-feature vector, where $j = 1, \ldots, m$, its components V_{ij} are normalized as follows: V_{ij}/V_{kj}, where $k = \arg\max_l V_{lj}$. In other words, components of each feature vector are divided by the maximal value among them. As a result, feature vectors are composed of components whose nonnegative values do not exceed 1. Since all three matrices (**V**, **W**, **H**) have now entries between 0 and 1, it takes much less time to perform matrix factorization $\mathbf{V} \approx \mathbf{WH}$ (values of the entries in the factorized matrices do not have to grow much in order to satisfy the stopping criterion for the objection function F in (4)) than if **V** had the original (unnormalized) values. As additional benefit, MSE becomes much smaller too.

Though this modification is simple, it brings significant speed of convergence. The following theorem helps to understand why it happens.

Theorem 1. *Assume that F^{direct} and F^{iter} are values of the objective function in (4) when mapping the data with the direct and iterative techniques, respectively. Then $F^{\text{direct}} - F^{\text{iter}} \geq 0$ always holds at the start of iterations.*

Proof. By definition,

$$F^{\text{iter}} = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(V_{ij}\log(WH)_{ij} - (WH)_{ij}\right),$$
$$F^{\text{direct}} = \sum_{i=1}^{n}\sum_{j=1}^{m}(V_{ij}\log V_{ij} - V_{ij}). \tag{5}$$

The difference $F^{\text{direct}} - F^{\text{iter}}$ is equal to

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{m}\left(V_{ij}\log V_{ij} - V_{ij} - V_{ij}\log(WH)_{ij} + (WH)_{ij}\right)\\
&\quad= \sum_{i=1}^{n}\sum_{j=1}^{m}\left(V_{ij}\left(\log\frac{V_{ij}}{(WH)_{ij}} - 1\right) + (WH)_{ij}\right)\\
&\quad= \sum_{i=1}^{n}\sum_{j=1}^{m}(WH)_{ij}\left(\frac{V_{ij}}{(WH)_{ij}}\left(\log\frac{V_{ij}}{(WH)_{ij}} - 1\right) + 1\right).
\end{aligned} \tag{6}$$

Let us introduce a new variable, $x : x = V_{ij}/(WH)_{ij}$. Since $(WH)_{ij}$ is always nonnegative, the following condition must hold: $x(\log x - 1) + 1 \geq 0$. The plot of $x(\log x - 1) + 1$ versus x is shown in Figure 1. It can be seen that this function is always nonnegative. □

The higher x, that is, the bigger the ratio of V_{ij} to WH_{ij} (the case of unnormalized **V**), the larger the difference between F^{direct} and F^{iter}. In other words, if no normalization of **V** occurs, the direct mapping technique moves the beginning of iterations far away from the point where the conventional iterative technique starts since the objective function in (4) is increasing [3]. This, in turn, implies that normalization can significantly speed up convergence.

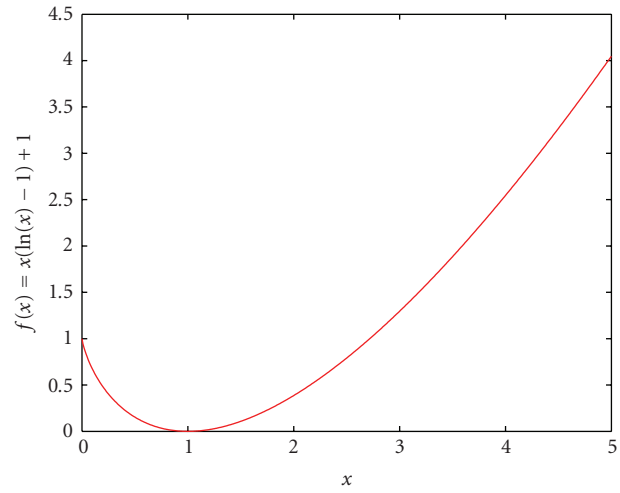

FIGURE 1: Function $x(\log x - 1) + 1$ for several values of x.

On the other hand, as follows from Figure 1, the only minimum occurs at $x = 1$, which means $\mathbf{V} = \mathbf{WH}$ and $F^{\text{direct}} = F^{\text{iter}}$. In practice, the strict equalities do not hold because of zeroing some entries in **H**. This means that both direct and iterative techniques start from approximately the same point if **V** is normalized as described above. To remedy the effect of zeroing, we propose to add a small random number, uniformly distributed between 0 and 1, to each entry of H_{new} obtained after applying the direct technique. After that, the iterative technique is used for mapping unseen data. In this way, we combine both mapping techniques. This is our second modification and the proposed technique is called *iterative2*.

2.3. Summary of our algorithm

Suppose that the whole data set is divided into training and test (unseen) sets. Our algorithm is summarized as follows.

(1) Scale both training and test data and randomly initialize the factorized matrices as described in Section 2.1. Set parameters tol and r.
(2) Iterate (1)–(3) until convergence to obtain the NMF basis matrix **W** and to map training data to the NMF (reduced) space.
(3) Given **W**, map test data by using the direct technique. Set negative values in the resulting matrix $\mathbf{H}_{\text{new}}^{\text{direct}}$ to zero.
(4) Fix the basis matrix and iterate (3) until convergence by using our initialization in Section 2.2. The resulting matrix $\mathbf{H}_{\text{new}}^{\text{iterative2}}$ provides reduced representations of the test data in the NMF space.

3. APPLICATION

3.1. Task

As a challenging task, we selected protein fold recognition from bioinformatics. Protein is an amino acid sequence. In

Table 1: Error rates when classifying protein folds in the original space with different methods.

Source	Classifier	Error rate (%)
[28]	DIMLP	61.8
[29]	MLP	51.2
[29]	GRNN	55.8
[29]	RBFN	50.6
[29]	SVM	48.6
[30]	RBFN	48.8
[31]	SVM	46.1
[32]	HKNN	42.7

bioinformatics, one of the current trends is to understand evolutionary relationships in terms of protein function. Two common approaches to identify protein function are sequence analysis and structure analysis. Sequence analysis is based on a comparison between unknown sequences and those whose function is already known. However, some closely related sequences may not share the same function. On the other hand, proteins may have low sequence identity but their structure and, in many cases, function suggest a common evolutionary origin.[4]

Protein fold recognition is structure analysis without relying on sequence similarity. Proteins are said to have a common fold if they have the same major secondary structure[5] in the same arrangement and with the same topology, whether or not they have a common evolutionary origin. The structural similarities of proteins in the same fold arise from the physical and chemical properties favoring certain arrangements and topologies, meaning that various physicochemical features such as fold compactness or hydrophobicity are utilized for recognition. As the gap widens between the number of known sequences and the number of experimentally determined protein structures (the ratio is more than 100 to 1, and sequence databases are doubling in size every year), the demand for automated fold recognition techniques rapidly grows.

3.2. Data set

A challenging data set derived from the SCOP (structural classification of proteins) database [33] was used in experiments described below. It is available on line[6] and its detailed description can be found in [31]. The data set contains the 27 most populated folds represented by seven or more proteins. Ding and Dubchak already split it into the training and test sets, which we will use as other authors did.

Six-feature sets compose the data set: amino acids composition, predicted secondary structure, hydrophobicity, normalized van der Waals volume, polarity, and polarizability. A feature vector combining six features has 125 dimensions. The training set consists of 313 protein folds having (for each two proteins) no more than 35% of the sequence identity for aligned subsequences longer than 80 residues. The test set of 385 folds is composed of protein sequences of less than 40% identity with each other and less than 35% identity with the proteins of the first set. In fact, 90% of the proteins of the test set have less than 25% sequence identity with the proteins of the training set. This, as well as multiple classes many of which are sparsely represented in the training set, render the task extremely difficult.

3.3. Previous work

All approaches, briefly mentioned below, use the data set described in the previous section. Unless otherwise stated, a 125-dimensional feature vector is assumed for each protein fold. In order to provide fair comparison, we concentrate on single classifiers rather than ensembles of classifiers. All but one papers below do not utilize dimensionality reduction prior to classification.

Ding and Dubchak [31] employed support vector machines (SVMs) (one-versus-all, unique one-versus-all, and one-versus-one methods for building multiclass SVMs). Bologna and Appel [28] used a 131-dimensional feature vector (protein sequence length was added to other features) and a four-layer discretized interpretable multi layer perceptron (DIMLP). Chung et al.[29] selected different models of neural networks (NNs) with a single hidden layer (MLP, radial basis function network (RBFN), and general regression neural network (GRNN)) and SVMs as basic building blocks for classification. Huang et al.[34] exploited a similar approach by utilizing gated NNs (MLPs and RBFNs). Gating is used for on-line feature selection in order to reduce the number of features fed to a classifier. Gates are open for useful features and they are close for bad ones. First, the original data are used to train the gating network. At the end of the training, the gate-function values for each feature indicate whether a particular feature is relevant or not by comparing these values against a threshold. Only the relevant features are then used to train a classifier. Pal and Chakraborty [30] trained MLPs and RBFNs with new features (400 in number) based on the hydrophobicity of the amino acids. In some cases, the 400-dimensional feature vectors led to a higher accuracy than when using the traditional (125-dimensional) ones. Okun [32] applied a variant of the nearest-neighbor classifier (HKNN). Table 1 summarizes the best results achieved with the above mentioned methods.

As one can observe, the error rates when employing a single classifier are high due to the discussed challenges. Ensembles of classifiers can sometimes reduce the error rate to about 39% as demonstrated in [28], but their consideration is beyond this work. For HKNN, the *normalized* features were used since feature normalization to zero mean and unit variance prior to HKNN dramatically increases classification accuracy (on average, by 6% [35]). However, this normalization can produce negative features and, thus, it is not appropriate for NMF requiring nonnegativity constraints to hold. This is the reason why we prefer another feature scaling in Section 2.2.

[4] It is argued that sequence analysis is good at high levels of sequence identity, but below 50% identity it becomes less reliable.
[5] Regions of local regularity within a fold.
[6] http://crd.lbl.gov/~cding/protein/

4. EXPERIMENTS

Experiments with NMF involve estimation of the error rate when performing classification in low-dimensional space as well as time until convergence on the data set described in Section 3.2. Three techniques for mapping test data to this space are used: direct, iterative, and iterative2. Regarding NMF, matrices **W** and **H** are initialized with random numbers uniformly distributed between 0 and 1 when the state of the random number generator ranged from 1 to 10. The same value of the state is used for mapping both training and test data. The value of tol was fixed to 0.01.

Though we tried numerous values of r (dimensionality of reduced space) between 50 and 100, we report the best results obtained with $r = 88$,[7] which constitutes 70.4% of the original dimensionality.

All algorithms were implemented in MATLAB running on a Pentium 4 (3 GHz CPU, 1 GB RAM).

4.1. Classifiers

We studied three classifiers: standard k-nearest neighbor (KNN) [36], k-nearest neighbor (KKNN) [37], and k-local hyperplane distance nearest neighbor (HKNN) [38]. HKNN was selected, since it demonstrated a competitive performance compared to SVM when both methods were applied to classify the above-mentioned protein data set in the original space [32, 39], that is, without dimensionality reduction. In addition, when applied to other data sets, the combination of NMF and HKNN showed very good results [40], thus rendering HKNN as a natural selection for NMF. Because of this reason, KNN and its kernel variant were selected for comparison with HKNN since all three algorithms belong to the same group of classifiers.

KNN has one parameter to be set: the number of nearest neighbor, k. Typical values for it are 1, 3, and 5.

KKNN is a modification of KNN when applying kernels. When selecting an appropriate kernel, the kernel nearest-neighbor algorithm, via a nonlinear mapping to a high-dimensional feature space, may be superior over KNN for some sample distributions. The same kernels as in case of SVM are commonly used, but as remarked in [37], only the polynomial kernel with degree $p \neq 1$ is actually useful, since the polynomial kernel with $p = 1$ and radial basis (Gaussian) kernel degenerate KKNN to KNN. The kernel approach to KNN consists of two steps: kernel computation, followed by distance computation in the feature space expressed via a kernel. After that, a nearest-neighbor rule is applied just as in the case of KNN. We tested KKNN for all combinations of $p = 0.5, 2, 3, 4, 5, 6, 7$ and $k = 1, 3, 5$ (21 combinations of parameters in total).

HKNN is another modification of KNN intended to compete with SVM when KNN fails to do so. HKNN computes distances of each test point x to L local hyperplanes, where L is the number of different classes. The ℓth hyperplane is composed of k nearest neighbors of x in the training set, belonging to the ℓth class. A test point x is associated with the class whose hyperplane is closest to x. HKNN needs to predefine two parameters, k and λ (regularization). Their values are 6 and 7 for k and 8, 10, 12, 20, 30, 40, 50 for λ (hence 14 combinations of two parameters in total). The value $k = 7$ is the largest possible value to choose since the minimum number of protein folds per class in the training set is seven.

4.2. Classification results

Table 2 summarizes the error ranges for three classifiers when doing classification in the original and NMF spaces for $r = 88$ and parameters of each classifier given in Section 4.1. In the first column, "NMF-Direct" stands for the NMF space and direct technique used to map test data to this space. "NMF-Iterative" and "NMF-Iterative2" mean the same regarding the iterative and iterative2 techniques, respectively.

First, we would like to analyze each classifier separately from the others. KNN using normalized features in the original space is clearly the best, since it yields the lowest minimum error as well as the narrowest range of errors. KKNN applied in the NMF-Iterative and NMF-Iterative2 spaces can sometimes lead to errors smaller than those in the original space, but the ranges of errors achieved for low-dimensional spaces are significantly wider than the range for the original space. This fact emphasizes sensitivity of KKNN to its parameter settings when the data dimensionality is reduced by 30%. Finally, HKNN employed in the NMF-Iterative and NMF-Iterative2 spaces demonstrated clear advantages of dimensionality reduction. Though the iterative technique had slight edge (only in 6 out 140 experiments) over the iterative2 technique in terms of minimum error achieved (this error is the lower for both iterative techniques than error in the original space!), the former has significantly larger maximum error than the latter. In addition, the iterative2 technique yields the same maximum error as that in the original space and this error is the lowest among all others. Hence, the iterative2 technique causes HKNN to have the smallest variance of error, thus making it least sensitive to different parameters. For each classifier, the direct technique for mapping test data to low-dimensional space lagged far behind either iterative technique in terms of classification accuracy. Therefore we do not recommend to apply it alone.

If one compares three classifiers, HKNN emerges as an undisputed winner in both high- and low-dimensional spaces. Based on the previous experience with this classifier [38, 40], this fact is not surprising. Coupled with our modifications of NMF, it demonstrated a good performance exceeding that of many neural network models and SVM employed in high-dimensional space (see Table 1). In particular, compared to the classification in the original space, the minimum error in the NMF-Iterative2 space is lower by 4% (last column in Table 2).

We also noticed that for certain values of the state of the random number generator, the iterative2 technique provides a better classification accuracy than the iterative technique in more than a half of experimental cases. For example, these

[7] Given a 125×313 training matrix, 88 is the largest possible value of r resulting in data compression.

Table 2: Ranges of the error rates (%) for three classifiers.

Space	Scaling	Classifier		
		KNN	KKNN	HKNN
Original	No	58.44 − 67.79	56.36 − 68.05	52.47 − 53.25
Original	Yes	55.58 − 67.79	55.06 − 68.57	45.45 − 47.53
NMF-direct	Yes	70.39 − 85.45	69.09 − 89.87	59.22 − 79.48
NMF-iterative	Yes	57.14 − 73.25	54.03 − 95.32	40.52 − 49.35
NMF-iterative2	Yes	57.66 − 74.03	54.29 − 96.10	41.30 − 47.53

states are 1, 5, 7, and 8 for HKNN, with states 1 and 8 shared with other two classifiers. With such states, the accuracy was better in the overwhelming number of cases. Thus, it seems that there is a link between high accuracy in the low-dimensional space of NMF and the state of the random generator, which needs further exploration.

4.3. Time

In this section, we provide the evidence that feature scaling prior to iterations significantly speeds up convergence when mapping both training and test data to low-dimensional space, compared to the case when no scaling is used. Table 3 accumulates gains resulted from feature scaling for several data dimensionalities. R_1 stands for the ratio of the average times spent on learning NMF basis and mapping training data to NMF space without and with scaling prior to NMF. R_2 means the ratio of the average times spent on mapping test data by means of the iterative technique without and with scaling prior to NMF. R_3 is the ratio of the average times spent on mapping test data by means of the iterative2 technique without and with scaling prior to NMF. Thus, the average gain from feature scaling is more than 11 times.

Table 3: Gains in time resulting from feature scaling.

	Mapping		
	Training data	Test data	
r	R_1	R_2	R_3
88	11.9	11.4	10.4
75	13.8	12.9	13.1
50	13.2	11.1	12.5
25	9.5	6.4	8.8
Average	12.1	10.4	11.2

5. CONCLUSION

The main contribution of this work is two modifications of the basic NMF algorithm [3] and its practical application to the challenging real-world task of protein fold recognition. The first modification carries out feature scaling before NMF while the second modification combines two known techniques for mapping unseen data.

When modifying NMF, we considered two aspects: (1) time until convergence since factorization is done by means of an iterative algorithm and (2) the error rate when combining NMF and a classifier. Three nearest-neighbor classifiers were tested.

We demonstrated that proper feature scaling makes the NMF algorithm 11 times faster to converge. The reason of why it happens is explained based on Theorem 1 in Section 2.2. Regarding classification in low-dimensional space, our experimental results showed that simultaneously with faster convergence, significant gains in accuracy can be achieved, too, compared to the known results in the original, high-dimensional space. However, these gains depend on the state of the random number generator, a classifier, and its parameters.

APPENDIX

Lee and Seung in [41] used a measure resembling the Kullback-Leibler divergence [42] to quantify the quality of the approximation $\mathbf{V} \approx \mathbf{WH}$. For two nonnegative matrices, $\mathbf{A}$ and $\mathbf{B}$, this measure is

$$D(\mathbf{A} \parallel \mathbf{B}) = \sum_{ij} \left(A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij} \right). \tag{A.1}$$

It is lower bounded by zero if and only if $\mathbf{A} = \mathbf{B}$. Regarding NMF, let $\mathbf{A} = \mathbf{V}$ and $\mathbf{B} = \mathbf{WH}$. In order to ensure a good approximation, one minimizes $D(\mathbf{V} \parallel \mathbf{WH})$ with respect to $\mathbf{W}$ and $\mathbf{H}$, subject to the nonnegativity constraints on $\mathbf{W}$ and $\mathbf{H}$.

Equation (A.1) can be rewritten as follows:

$$\begin{aligned} D(\mathbf{V} \parallel \mathbf{WH}) &= \sum_{ij} \left(V_{ij} \log \frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij} \right) \\ &= F_0 - F, \end{aligned} \tag{A.2}$$

where $F_0 = \sum_{ij}(V_{ij} \log V_{ij} - V_{ij})$ and $F = \sum_{ij}(V_{ij} \log (WH)_{ij} - (WH)_{ij})$.

F_0 does not include $(WH)_{ij}$ and therefore does not have any effect on minimization and can be omitted. As a result, minimizing $F_0 - F$ implies maximizing F, subject to the constraints.

REFERENCES

[1] I. T. Jolliffe, *Principal Component Analysis*, Springer, New York, NY, USA, 1986.

[2] P. Common, "Independent component analysis," in *Proceedings of the International Signal Processing Workshop on Higher-Order Statistics*, pp. 111–120, Chamrousse, France, July 1991.

[3] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[4] S. Tsuge, M. Shishibori, S. Kuroiwa, and K. Kita, "Dimensionality reduction using non-negative matrix factorization for information retrieval," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 2, pp. 960–965, Tucson, Ariz, USA, July–October 2001.

[5] B. Xu, J. Lu, and G. Huang, "A constrained non-negative matrix factorization in information retrieval," in *Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI '03)*, pp. 273–277, Las Vegas, Nev, USA, October 2003.

[6] I. Buciu and I. Pitas, "Application of non-negative and local non negative matrix factorization to facial expression recognition," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR '04)*, vol. 1, pp. 288–291, Cambridge, UK, 2004.

[7] X. Chen, L. Gu, S. Z. Li, and H.-J. Zhang, "Learning representative local features for face detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '01)*, vol. 1, pp. I-1126–I-1131, Kauai, Hawaii, USA, December 2001.

[8] T. Feng, S. Z. Li, H.-Y. Shum, and H.-Y. Zhang, "Local non-negative matrix factorization as a visual representation," in *Proceedings of the 2nd International Conference on Development and Learning*, pp. 178–183, Cambridge, Mass, USA, June 2002.

[9] D. Guillamet and J. Vitria, "Discriminant basis for object classification," in *Proceedings of the 11th International Conference on Image Analysis and Processing*, pp. 256–261, Palermo, Italy, September 2001.

[10] D. Guillamet and J. Vitrià, "Evaluation of distance metrics for recognition based on non-negative matrix factorization," *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1599–1605, 2003.

[11] D. Guillamet, J. Vitrià, and B. Schiele, "Introducing a weighted non-negative matrix factorization for image classification," *Pattern Recognition Letters*, vol. 24, no. 14, pp. 2447–2454, 2003.

[12] M. Rajapakse and L. Wyse, "NMF vs ICA for face recognition," in *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis (ISPA '03)*, vol. 2, pp. 605–610, Rome, Italy, September 2003.

[13] R. Ramanath, W. E. Snyder, and H. Qi, "Eigenviews for object recognition in multispectral imaging systems," in *Proceedings of the 32nd Applied Imagery Pattern Recognition Workshop*, pp. 33–38, Washington, DC, USA, October 2003.

[14] L. K. Saul and D. D. Lee, "Multiplicative updates for classification by mixture models," in *Advances in Neural and Information Processing Systems*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14, pp. 897–904, MIT Press, Cambridge, Mass, USA, 2002.

[15] Y. Wang, Y. Jia, C. Hu, and M. Turk, "Fisher non-negative matrix factorization for learning local features," in *Proceedings of the 6th Asian Conference on Computer Vision*, Jeju Island, Korea, January 2004.

[16] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of Machine Learning Research*, vol. 5, pp. 1457–1469, 2004.

[17] Y. Li and A. Cichocki, "Sparse representation of images using alternating linear programming," in *Proceedings of the 7th International Symposium on Signal Processing and Its Applications (ISSPA '03)*, vol. 1, pp. 57–60, Paris, France, July 2003.

[18] W. Liu, N. Zheng, and X. Lu, "Non-negative matrix factorization for visual coding," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, vol. 3, pp. 293–296, Hong Kong, April 2003.

[19] S. Behnke, "Discovering hierarchical speech features using convolutional non-negative matrix factorization," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 4, pp. 2758–2763, Portland, Ore, USA, July 2003.

[20] Y.-C. Cho, S. Choi, and S.-Y. Bang, "Non-negative component parts of sound for classification," in *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology (ISSPIT '03)*, pp. 633–636, Darmstadt, Germany, December 2003.

[21] M. Novak and R. Mammone, "Use of non-negative matrix factorization for language model adaptation in a lecture transcription task," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, vol. 1, pp. 541–544, Salt Lake City, Utah, USA, May 2001.

[22] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 177–180, New Paltz, NY, USA, October 2003.

[23] J. Lu, B. Xu, and H. Yang, "Matrix dimensionality reduction for mining Web logs," in *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, pp. 405–408, Halifax, NS, Canada, October 2003.

[24] Y. Mao and L. K. Saul, "Modeling distances in large-scale networks by matrix factorization," in *Proceedings of the ACM Internet Measurement Conference*, pp. 278–287, Sicily, Italy, October 2004.

[25] M. Cooper and J. Foote, "Summarizing video using non-negative similarity matrix factorization," in *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, pp. 25–28, St.Thomas, Virgin Islands, USA, December 2002.

[26] J. Lawrence, S. Rusinkiewicz, and R. Ramamoorthi, "Efficient BRDF importance sampling using a factored representation," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 496–505, 2004, Special issue: Proceedings of the 2004 SIGGRAPH Conference.

[27] M. D. Plumbley and E. Oja, "A "nonnegative PCA" algorithm for independent component analysis," *IEEE Transactions on Neural Networks*, vol. 15, no. 1, pp. 66–76, 2004.

[28] G. Bologna and R. D. Appel, "A comparaison study on protein fold recognition," in *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP '02)*, vol. 5, pp. 2492–2496, Singapore, November 2002.

[29] I.-F. Chung, C.-D. Huang, Y.-H. Shen, and C.-T. Lin, "Recognition of structure classification of protein folding by NN and SVM hierarchical learning architecture," in *Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP '03)*, O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, Eds., vol. 2714 of *Lecture Notes in Computer Science*, pp. 1159–1167, Istanbul, Turkey, June 2003.

[30] N. R. Pal and D. Chakraborty, "Some new features for protein fold recognition," in *Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP '03)*, O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, Eds., vol. 2714 of *Lecture Notes in Computer Science*, pp. 1176–1183, Istanbul, Turkey, June 2003.

[31] C. H. Q. Ding and I. Dubchak, "Multi-class protein fold recognition using support vector machines and neural networks," *Bioinformatics*, vol. 17, no. 4, pp. 349–358, 2001.

[32] O. Okun, "Protein fold recognition with k-local hyperplane distance nearest neighbor algorithm," in *Proceedings of the 2nd European Workshop on Data Mining and Text Mining for Bioinformatics*, pp. 47–53, Pisa, Italy, September 2004.
[33] L. Lo Conte, B. Ailey, T. J. P. Hubbard, S. E. Brenner, A. G. Murzin, and C. Chothia, "SCOP: a structural classification of proteins database," *Nucleic Acids Research*, vol. 28, no. 1, pp. 257–259, 2000.
[34] C.-D. Huang, I.-F. Chung, N. R. Pal, and C.-T. Lin, "Machine learning for multi-class protein fold classification based on neural networks with feature gating," in *Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP '03)*, O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, Eds., vol. 2714 of *Lecture Notes in Computer Science*, pp. 1168–1175, Istanbul, Turkey, June 2003.
[35] O. Okun, "Feature normalization and selection for protein fold recognition," in *Proceedings of the 11th Finnish Artificial Intelligence Conference*, pp. 207–221, Vantaa, Finland, September 2004.
[36] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
[37] K. Yu, L. Ji, and X. Zhang, "Kernel nearest-neighbor algorithm," *Neural Processing Letters*, vol. 15, no. 2, pp. 147–156, 2002.
[38] P. Vincet and Y. Bengio, "K-local hyperlane and convex distance nearest neighbor algorithms," in *Advances in Neural Information Processing Systems*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14, pp. 985–992, MIT Press, Cambridge, Mass, USA, 2002.
[39] O. Okun, "K-local hyperplane distance nearest neighbor algorithm and protein fold recognition," *Pattern Recognition and Image Analysis*, vol. 16, no. 1, pp. 19–22, 2006.
[40] O. Okun, "Non-negative matrix factorization and classifiers: experimental study," in *Proceedings of the 4th IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP '04)*, pp. 550–555, Marbella, Spain, September 2004.
[41] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural and Information Processing Systems*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., vol. 13, pp. 556–562, MIT Press, Cambridge, Mass, USA, 2001.
[42] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

Oleg Okun received his Candidate of Sciences (Ph.D.) degree from the Institute of Engineering Cybernetics, Belarusian Academy of Sciences, in 1996. Since 1998, he joined the Machine Vision Group of Infotech Oulu, Finland. Currently, he is a Senior Scientist and Docent (Senior Lecturer) in the University of Oulu, Finland. His current research focuses on image processing and recognition, artificial intelligence, machine learning, data mining, and their applications especially in bioinformatics. He has authored more than 50 papers in international journals and conference proceedings. He has also served on committees of several international conferences.

Helen Priisalu got her M.S. degree in engineering from Tallinn University of Technology, Estonia, in 2005. She is working on her Ph.D. thesis and her research involves machine learning, data mining, and their applications in bioinformatics and web log analysis.