The Non-Negative Matrix Factorization Toolbox for Biological Data Mining

Yifeng Li*[1] and Alioune Ngom[1]

[1]School of Computer Science, University of Windsor, Windsor, Ontario, Canada

Email: Yifeng Li*- li11112c@uwindsor.ca; Alioune Ngom - angom@cs.uwindsor.ca;

*Corresponding author

Abstract

Background: Non-negative matrix factorization (NMF) has been being an important approach to analyze biological data. Though there exists some packages implemented in R and other programming languages, they either only provide some optimization algorithms, or focus on a specific application field. There is no complete package for the bioinformatics community to perform various data mining on biological data.
Results: We provide a powerful but convenient MATLAB toolbox including both the implementations of various NMFs and a variety of NMF-based data mining approaches for analyzing biological data. Through this toolbox, data mining approaches such as clustering, biclustering, feature extraction, feature selection, classification, overcoming missing values, visualization, and statistical comparison can be easily done.
Conclusions: A series of analysis such as molecular pattern discovery, biological process identification, dimension reduction, disease prediction, visualization, and statistical comparison can be conducted via this toolbox.
Keywords: non-negative matrix factorization; clustering, biclustering, feature extraction, feature selection, classification, missing values.

Background

Non-negative matrix factorization (NMF) is a matrix decomposition approach that factorizes a non-negative matrix into two low-rank non-negative matrices [1]. It has made great success in biological data mining. The following is some well-known examples. [2] and [3] used NMF as clustering method to discover metagenes and molecular patterns. [4] applied *non-smooth NMF* (nsNMF) for biclustering of gene expression data. *Least-squares NMF* (LS-NMF) was proposed to take into account of the uncertainty information in gene expression data [5]. [6] proposed kernel NMF for the dimension reduction of gene expression data. Many authors indeed provide their implementation along with their publications. It is convenient for the interested readers to verify and apply them to their specific fields. However, there exists at least three issues that prevent NMF from a better application in biological and medical studies. First, their codes are implemented in diverse programming languages, such as R, MATLAB, C++, and Java, and usually only one optimization algorithm is provided in a paper. It is inconvenient for some researchers who want to choose a suitable NMF for their data among many different implementations which are realized in different languages and termination criteria. Second, some papers only provide optimization algorithms at a basic level, rather than a data mining implementation at a higher level. If a biologist wants to do clustering, biclustering, and then visualizing the results, it might be hard for him or her to investigate these three aspect of data mining and visualization. Third, the existing implementations often stick to a specific application. There is no systematic package for NMF-based biological data mining.

There are some existing NMF toolboxes which are introduced in the following. However, neither address the above three issues altogether. *NMFLAB* [7] is MATLAB toolbox for signal and image processing. NMFLAB provides a user-friendly interface to load data, processing signals, and save result. It includes a variety of algorithms such as multiplicative rules, exponentiated gradient, projected gradient, conjugate gradient, and Quasi-Newton. It also can visualize the signals and components. But it does not provide any data mining functionality, which is the second issue mentioned above. It is also lack of some important NMF concepts such as semi-NMF and kernel NMF. *NMF:DTU Toolbox* [8] is also a MATLAB toolbox including five NMF optimization algorithms such as multiplicative rules, projected gradient, probabilistic NMF, alternating least squares, and alternating least squares with optimal brain surgeon. It does not address the second and third issues. *NMFN: Nonnegative Matrix Factorization* [9] is a R package that implemented several algorithms. It has the same issue as the DTU toolbox. *NMF: Algorithms and framework for Nonnegative Matrix Factorization* [10] is a R package. It implemented several algorithms and their main interface allows parallel computations. It does not solve the second issue. *Text to Matrix Generator (TMG)* is a MATLAB toolbox for text mining. It does not solve the third issue. [11] provided a NMF plug-in for BRB-ArrayTools. This plug-in only implemented the standard NMF and semi-NMF for clustering gene expression profiles. Therefore it does not address the second and third issues well.

In order to address the three issues, we propose a NMF toolbox in MATLAB in this paper. The toolbox is implemented in two levels. The basic level is composed of the algorithms of NMF variants, and the advanced level consists of diverse applications of NMF in biological data mining. The contributions of our toolbox are enumerated in the following:

1. The NMF algorithms are relatively complete, and all of them are implemented in MATLAB. This is to address the first issue. It is impossible and unnecessary to implement all NMF algorithms. We focus on the well-known NMF representatives. This repository of NMFs allows users to select the most suitable one in specific scenarios.

2. It includes comprehensive functionalities of data mining, such as clustering, biclustering, feature extraction, feature selection, and classification. This solves the second and third issues to a great extend.

3. It provides functions of biological data visualization. For instance, heat maps of a NMF can be plotted via our toolbox. It is pretty helpful to interpret result. It also provides statistical methods to compare the performance of multiple methods.

The rest of this paper is organized as below. The implementations of of the basis level are given in Section . Examples of applying the advanced level in biological data mining are demonstrated in Section . The conclusion and future works are given finally.

Implementation

As mentioned above, this toolbox is implemented at two levels. The fundamental level is composed of various NMF algorithms. And the advanced level includes many data mining approaches based on the fundamental level. The critical issues of implementing these NMF variants are addressed in this section. Table 1 summarized all the NMF algorithms implemented in our toolbox. Potential users need to use the command `help nmfrule` in the command line, for example, to learn how to set the parameters of a function in our toolbox. The advanced level and some examples are given in the next section.

Standard-NMF

The *standard-NMF* decomposes a non-negative matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ into two non-negative factors $\boldsymbol{A} \in \mathbb{R}^{m\times k}$ and $\boldsymbol{Y} \in \mathbb{R}^{k\times n}$ (where $k < \min\{m, n\}$), that is

$$\boldsymbol{X}_+ = \boldsymbol{A}_+\boldsymbol{Y}_+ + \boldsymbol{E}, \tag{1}$$

where $\boldsymbol{E}$ is error (or residual). Its optimization in the Euclidean distance is formulated in the following equation

$$\min_{\boldsymbol{A},\boldsymbol{Y}} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{Y}\|_{\text{F}}^2, \text{ s.t. }, \boldsymbol{A}, \boldsymbol{Y} \geq 0. \tag{2}$$

Statistically speaking, this formulation is obtained from the log-likelihood function under the assumption of Gaussian error. If multivariate data points are arranged in the columns of $\boldsymbol{X}$, then $\boldsymbol{A}$ is called *basis matrix*, each column of $\boldsymbol{A}$ is thus called a *basis vector*, and $\boldsymbol{Y}$ is called *coefficient matrix*. The interpretation is that each data point is a (sparse) non-negative linear combination of the basis vectors. It is well-known that the objective is non-convex. Block-coordinate descent is therefore the main prescription of such problem. Multiplicative update rules were provided in [12] to optimize Equation 2. Though simple to implement, this algorithm is even not guaranteed to converge to a stationary point [13]. Essentially the optimizations above with respect to $\boldsymbol{A}$ and $\boldsymbol{Y}$, respectively, are *non-negative least squares* (NNLS). Therefore we implemented the alternating NNLS algorithm proposed in [13]. It can be proved that this algorithm converges to a stationary point. In our toolbox, functions `nmfrule` and `nmfnnls` are the implementations of the two algorithms above.

Semi-NMF

The standard NMF only works for non-negative data, which limits its applications. [14] extended it to *semi-NMF* which removes the non-negative constraints on the data $\boldsymbol{X}$ and basis matrix $\boldsymbol{A}$. It can be expressed in the following equation:

$$\min_{\boldsymbol{A},\boldsymbol{Y}} \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{A}\boldsymbol{Y}\|_{\mathrm{F}}^2, \text{ s.t. } \boldsymbol{Y}\geq 0. \tag{3}$$

Semi-NMF can be applied to the matrix of mixed signs, therefore it expands NMF to many fields. However, the gradient-descent-based update rule proposed in [14] is slow to converge (implemented in function `seminmfrule` in our toolbox). Keeping $\boldsymbol{Y}$ fixed, updating $\boldsymbol{A}$ is a least squares problem which has an analytical solution

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathrm{T}}(\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}})^{-1} = \boldsymbol{X}\boldsymbol{Y}^{\dagger}. \tag{4}$$

Updating $\boldsymbol{Y}$ while fixing $\boldsymbol{A}$ is a NNLS problem essentially as above. Therefore we implemented the fast NNLS based algorithm to optimize semi-NMF in function `seminmfnnls`.

Sparse-NMF

The standard NMF and semi-NMF have the issues of scale-invariance and non-unique solution, which imply that the non-negativity constrained on the least squares is insufficient in some cases. Sparsity is a popular regularization principle in statistical modeling [15]. It has been applied to reduce non-uniqueness and enhance interpretation of NMF. The *sparse-NMF* proposed in [3] is expressed in the following equation

$$\begin{aligned}&\min_{\boldsymbol{A},\boldsymbol{Y}}\frac{1}{2}\|\boldsymbol{X}-\boldsymbol{A}\boldsymbol{Y}\|_{\mathrm{F}}^2 + \frac{\eta}{2}\|\boldsymbol{A}\|_F^2 + \frac{\lambda}{2}\sum_{i=1}^{n}\|\boldsymbol{y}_i\|_1^2 \qquad (5)\\ &\text{s.t. } \boldsymbol{A},\boldsymbol{Y}\geq 0,\end{aligned}$$

where $\boldsymbol{y}_i$ is the i-th column of $\boldsymbol{Y}$. From the Bayesian viewpoint, This formulation is obtained from the log-posterior probability under the assumptions that Gaussian error, Gaussian distributed basis vector, and Laplace distributed coefficient. Keeping one matrix fixed, updating another matrix can be formulated into a NNLS problem. In order to improve the interpretation of basis vectors and speed up the algorithm, we implemented the following model instead:

$$\begin{aligned}&\min_{\boldsymbol{A},\boldsymbol{Y}}\frac{1}{2}\|\boldsymbol{X}-\boldsymbol{A}\boldsymbol{Y}\|_{\mathrm{F}}^2 + \lambda\sum_{i=1}^{n}\|\boldsymbol{y}_i\|_1 \qquad (6)\\ &\text{s.t. } \boldsymbol{A},\boldsymbol{Y}\geq 0,\\ &\quad \|\boldsymbol{a}_i\|_2^2 = 1, \quad i=1,\cdots,k.\end{aligned}$$

We optimize this using three alternating steps in each iteration. First of all, the following task is optimized:

$$\begin{aligned}&\min_{\boldsymbol{Y}}\frac{1}{2}\|\boldsymbol{X}-\boldsymbol{A}\boldsymbol{Y}\|_{\mathrm{F}}^2 + \lambda\sum_{i=1}^{n}\|\boldsymbol{y}_i\|_1 \qquad (7)\\ &\text{s.t. } \boldsymbol{Y}\geq 0.\end{aligned}$$

Second, $\boldsymbol{A}$ is updated in the following equation:

$$\begin{aligned}&\min_{\boldsymbol{A}}\frac{1}{2}\|\boldsymbol{X}-\boldsymbol{A}\boldsymbol{Y}\|_{\mathrm{F}}^2 \qquad (8)\\ &\text{s.t. } \boldsymbol{A}\geq 0.\end{aligned}$$

After that, the columns of $\boldsymbol{A}$ are normalized to have unit l_2 norm. The first and second step can be solved by *non-negative quadratic programming* (NNQP), whose general formulation is

$$\begin{aligned}&\min_{\boldsymbol{Z}}\sum_{i=1}^{n}\frac{1}{2}\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{H}\boldsymbol{z}_i + \boldsymbol{g}_i^{\mathrm{T}}\boldsymbol{z}_i + c_i \qquad (9)\\ &\text{s.t. } \boldsymbol{Z}\geq 0,\end{aligned}$$

where $\boldsymbol{z}_i$ is the i-th column of matrix variable $\boldsymbol{Z}$. It is easy to prove that NNLS is a specific problem of NNQP. For example, Equation 7 can be rewritten as

$$\begin{aligned}&\min_{\boldsymbol{Y}}\sum_{i=1}^{n}\frac{1}{2}\boldsymbol{y}_i^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})\boldsymbol{y}_i + (\lambda - \boldsymbol{A}^{\mathrm{T}}\boldsymbol{x}_i)^{\mathrm{T}}\boldsymbol{y}_i + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i \\ &\qquad (10)\\ &\text{s.t. } \boldsymbol{Y}\geq 0.\end{aligned}$$

The implementations of the method in [3] and our method are given in functions `sparsenmfnnls` and `sparseNMFNNQP`, respectively. We also implemented the sparse semi-NMF in function `sparseseminmfnnls`.

Kernel-NMF

Two features of a kernel approach are that i) it can represent complex patterns, and ii) the optimization of the model is dimension-free. We now show that NMF can also be kernelized. The difficulty is that the basis matrix is dimension-related, and therefore it is impossible to represent it in a very high (even infinite) dimensional fashion. We notice that the NNLS optimization of updating $\boldsymbol{Y}$ in Equation 10 only needs inner products $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$, $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X}$, and $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$. From Equation 4, we obtain that $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = (\boldsymbol{Y}^{\dagger})^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{Y}^{\dagger}$, $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X} = (\boldsymbol{Y}^{\dagger})^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$. Therefore, we can see that only inner product $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ is needed in the optimization of NMF! It is hence very convenient to obtain *kernel-NMF* by replacing the inner product $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ by a kernel matrix $K(\boldsymbol{X}, \boldsymbol{X})$. For further detail, interested readers are refereed to our recent paper [6]. Based on the above derivation, we implemented the kernel semi-NMF using multiplicative update rule (in `kernelseminmfrule`) and using NNLS (in `kernelseminmfnnls`). The sparse kernel semi-NMFs are implemented in functions `kernelsparseseminmfnnls` and `kernelSparseNMFNNQP` which are equivalent to each other. The kernel method of decomposing kernel matrix proposed in [16] is implemented in `kernelnmfdecom`.

Other Variants

[14] proposed *Convex-NMF*, where the columns of $\boldsymbol{A}$ are constrained to be the convex combinations of data points in $\boldsymbol{X}$. It is thus formulated as $\boldsymbol{X}_{\pm} = \boldsymbol{X}_{\pm}\boldsymbol{W}_{+}\boldsymbol{Y}_{+} + \boldsymbol{E}$, where $\boldsymbol{X}\boldsymbol{W} = \boldsymbol{A}$ and each column of $\boldsymbol{W}$ contains the convex coefficients of all the data points to get the corresponding column of $\boldsymbol{A}$. It has been verified that columns of $\boldsymbol{A}$ obtained by convex-NMF are close to the real cluster centroids. Convex-NMF can be kernelized as well [14]. We implemented convex-NMF and its kernel version in `convexnmfrule` and `kernelconvexnmf`, respectively.

The basis vectors obtained by all the above NMF are non-orthogonal. Alternatively, *orthogonal NMF* (ortho-NMF) imposes orthogonality to enhance sparsity [17]. Its formulation is

$$\begin{aligned} &\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}\boldsymbol{Y} + \boldsymbol{E} \\ &\text{s.t. } \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \boldsymbol{I}, \quad \boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} = \boldsymbol{I}, \quad \boldsymbol{A}, \boldsymbol{S}, \boldsymbol{Y} \geq 0, \end{aligned} \tag{11}$$

where the input $\boldsymbol{X}$ is non-negative, $\boldsymbol{S}$ absorbs the magnitude due to the normalization of $\boldsymbol{A}$ and $\boldsymbol{Y}$. Function `orthnmfrule` is its implementation in our toolbox.

If we apply NMF on data subject to missing values, there are two fast ways. First, the missing values can be estimated prior to running NMF. Alternatively, *weighted-NMF* [18] can be directly applied to decompose it. Weighted-NMF puts a zero weight on the missing elements and hence only the present data contribute to the final result. An expectation-maximization (EM) based missing value estimation during the running of NMF might not a wise choice due to computational concern. The weighted-NMF is given in our toolbox in function `wnmfrule`.

Results and discussion

Based on the various NMFs implemented, a series of data mining can be conducted via our toolbox. Table 2 lists the functions in this level. These applications are described along with examples as below.

Clustering, biclustering, and biological process discovery

NMF has been applied for clustering. Given data $\boldsymbol{X}$ with multivariate data points in the columns, the idea is that, after applying NMF on $\boldsymbol{X}$, a multivariate data point, say $\boldsymbol{x}_i$ is a non-negative linear combination of the columns of $\boldsymbol{A}$, that is $\boldsymbol{x}_i \approx \boldsymbol{A}\boldsymbol{y}_i = y_{1i}\boldsymbol{a}_1 + \cdots + y_{ki}\boldsymbol{a}_k$. The greatest coefficient in the ith column of $\boldsymbol{Y}$ indicates the cluster this data point belongs to. The reason is that if the data points are mainly composed by the same basis vector, they should be in the same group. A basis vector is usually viewed as a cluster centroid or prototype. It has been applied for clustering microarray data for the discovery of tumor subtypes in [2]. We implemented function `NMFCluster`, through which various NMF algorithms can be chosen. An example is provided in `exampleCluster` file in the folder of our toolbox.

Furthermore, we can obtain more interpretation via the basis vectors. When applying NMF on microarray data, the basis vectors are interpreted as potential biological processes [3] [19]. Figure 1 shows the heat map of NMF clustering of a time-series data of yeast metabolic cycle in [20], and Figure 2 illustrates the three basis vectors. We used `nmfnnls` function to decompose the data and `NMFHeatMap` to plot the heat map. The detailed script is given in the `exampleBioProcess` file in the toolbox. We can clearly see that the three periodical biological processes corresponds exactly to the Ox (oxidative), R/B (reductive, building), and R/C (reductive, charging) processes discovered in [20].

If we interpret both of the basis matrix and coefficient matrix, NMF can be applied as biclustering approach Interested readers are referred to [21] for a survey of biclustering. We implemented a biclustering approach based on NMF in `biCluster` function. The biclusters can be visualized via function `NMFBicHeatMap`. We applied NMF to simultaneously grouping the genes and samples of a leukemia dataset [2] which includes tumor samples of three subtypes. The goal is to find strongly correlated genes over a subset of samples. A subset of such genes and a subset of such samples form a bicluster. The heat map is shown in Figure 3. Readers can find the script in `exampleBiCluster` file of our toolbox.

Feature Extraction

Microarray data and mass spectrometry data have tens of thousands of features but only tens or hundreds of samples. This leads to the notorious curses of dimensionality. For example, it is impossible to estimate the parameters of some statistical models as the number of their parameters grow exponentially as the dimension increases. Another issue is that biological data is normally subject to much noise, which crucially affects the analysis. In cancer study, a common hypothesis is that only several biological factors (for example oncogenes) play a crucial role in the development of a cancer. When we generate data from control and unhealthy patients, the huge amount of data (due to high dimensions) therefore contain lots of irrelevant and redundant information. Orthogonal factors obtained by *principal component analysis* (PCA) or *independent component analysis* (ICA) are not appropriate in many cases. Since NMF generates non-orthogonal (non-negative) factors, therefore it is much plausible to extract new features from such data using NMF. As mentioned above, training data $\boldsymbol{X}_{m\times n}$, with m features and n samples, can be decomposed into k metasamples $\boldsymbol{A}_{m\times k}$ and $\boldsymbol{Y}_{k\times n}$, that is

$$\boldsymbol{X} \approx \boldsymbol{A}\boldsymbol{Y}_{\text{tr}}, \text{ s.t. } \boldsymbol{A}, \boldsymbol{Y}_{\text{tr}} \geq 0. \tag{12}$$

The k columns of $\boldsymbol{A}$ span the k-dimensional *feature space* and each column of $\boldsymbol{Y}_{\text{tr}}$ is the representation of the corresponding original training sample in the feature space. In order to project the p future unknown samples $\boldsymbol{S}_{m\times p}$ into this feature space, we have to solve the following non-negative least squares problem:

$$\boldsymbol{S} \approx \boldsymbol{A}\boldsymbol{Y}_{\text{uk}}, \text{ s.t. } \boldsymbol{Y}_{\text{uk}} \geq 0. \tag{13}$$

After obtaining $\boldsymbol{Y}_{\text{tr}}$ and $\boldsymbol{Y}_{\text{uk}}$, the learning and prediction can be quickly done in the k-dimensional feature space instead of the m-dimensional original space. A classifier can learn over $\boldsymbol{Y}_{\text{tr}}$, and then predict the class labels of the representations of unknown samples, that is $\boldsymbol{Y}_{\text{uk}}$. From the aspect of interpretation, the advantage of NMF over PCA and ICA is that the metasamples are meaningful to understand the underlining physical processes, as mentioned above. We implemented a pair of functions `featureExtractionTrain` and `featureExtractionTest` including many linear and kernel NMF algorithms. The basis matrix (or inner product of basis matrix in the kernel case) is learned from training data via the former function, and unknown samples can be projected into the feature space via the later function. We give the examples of how to use these functions in files `exampleFeatureExtraction` and `exampleFeatureExtractionKernel`. Figure 4 shows the classification performance of no dimension reduction, linear NMF, kernel NMF using *radial basis function* (rbf) kernel, and PCA on two datasets, SRBCT [22] and Breast [23]. Since ICA is computationally very slow, we did not compare with it. The bars represent the averaged 4-fold cross-validation accuracies using *support vector machine* (SVM) as classifier over 20 runs. We can see that NMF is comparable with PCA on SRBCT, and is slightly better than PCA on Breast data. Also, with only few factors, the performance after dimension reduction using NMF is similar with, even better than, that without any dimension reduction. As future work, supervised NMF will be investigated and implemented in order to extract discriminative features.

Feature Selection

The columns in $\boldsymbol{A}$ for the gene expression data is called *metasample* in [2]. They can be interpreted as biological processes, because its values imply the activation and silence of all the genes. Gene selection aims to find marker genes to aid the disease prediction and understand pathways. Rather than selecting genes on the original data, the novel idea is to conduct gene selection on the metasamples. The reason is that the discovered biological process via NMF are biologically meaningful for the class discrimination, and the genes, expressing differently across these processes, should contribute to the classification. In Figure 3, for example, three biological processes are discovered and only selected genes are shown. We have implemented the information-entropy-based gene selection approach proposed in [3] in function `featureFilterNMF`. We give an example on how to call this function in file `exampleFeatureSelection`. It has been reported that it can select meaningful genes, which has been verified by gene ontology analysis. Feature selection based on supervised NMF will also be implemented.

Classification

If we make a reasonable assumption that every unknown sample is a sparse non-negative linear combination of the training samples, then we can directly derive a classifier from NMF. Indeed, this is a specific case of NMF where the training samples are the basis vectors. Since the optimization is actually NNLS problem, we dub this approach as *NNLS classifier*. A NNLS problem is essentially quadratic programming problem as formulated in Equation 9, therefore, only inner products are needed for the optimization. We hence can naturally extend the NNLS classifier to kernel version. Two glaring features of this approach are that i) the sparsity regularization can avoid overfitting the model; and ii) kernelization allows dimension-free optimization and linearizes complex patterns. The implementation of NNLS classifier is in file `nnlsClassifier`. Also, we provide many other classification approaches, for example SVM, in our toolbox. Please see file `exampleClassification` for demonstration. In our experiment of 4-fold cross-validation, accuracies of 0.7865 and 0.7804 are, respectively, obtained by linear and kernel (rbf kernel) NNLS classifier on Breast dataset. They achieved accuracies of 0.9762 and 0.9785, respectively, over SRBCT data.

Biological data are usually very noisy and sometimes suffer from the severe issue of missing values. Two key strengths of NNLS classifier are that it is very robust to noise and missing values, which make NNLS classifier very suitable for classifying biological data. In order to show its robustness to noise, we added Gaussian noise of mean 0 and variance from 0 to 4 by step 0.5 on SRBCT. Figure 5 illustrates the results of NNLS, SVM, and *1-nearest neighbor* classifiers using the noisy data. It can be seen that as the noise increases, NNLS outperforms SVM and 1-NN significantly. Biological data sometimes suffer from missing values. To deal with this problem, three strategies, removal, imputation, and disregard, are usually used. The first one will remove much other useful information, particularly when there is a large percent of missing values. The second one has a high risk of introducing false data. The third one is to avoid using missing values during analysis, which is the best way. Our idea in the following belongs to the last case. We base our idea of coping with missing values on the fact that NNLS optimization only needs inner products of pairs of samples. When computing the inner product of two samples, say $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, we normalize them to have unit l_2 norm using only the features present in both samples, and then take the inner product. In order to impress the readers, we removed 10% to 70% of the data in STBCT. Using such data, we compared our method with 0-imputation method (the more sophisticated $k - NN$ method [24] would fail in high missing rate because no complete vector exists) in Figure 6. We can see that the NNLS classifier using our idea outperforms the imputation method in the case of large missing rate.

Statistical Comparison

In order to evaluate the performance of a method statistically, we need to conduct statistical tests. In this toolbox, we provide two methods for statistical evaluation. The first is a two-stage method proposed in [25]. The importance of this method is that it can estimate the data-size requirement for significant accuracy and extrapolate the performance based on current available data. Generating biological data is usually very expensive. This method helps researchers to evaluate the necessity of producing more data. At the first stage, the minimum data size required for significant accuracy is estimated. This is implemented in function `significantAcc`. The second stage is to fit the learning curve using error rates of large data sizes. It is implemented in function `learnCurve`. In our experiment, we found that the NNLS classifier usually requires few number of samples for significant accuracy. For example on SRBCT data, NNLS requires only 4 training samples, while SVM needs 19 training samples for significant accuracy. The fitted learning curves of NNLS and SVM classifiers are shown in Figure . We provide an example of how to plot this figure in file `exampleFitLearnCurve`.

The second method is the nonparametric Friedman test coupled with post-hoc Nemenyi test to compare multiple classifiers over multiple datasets [26]. It is difficult to draw overall conclusion, if we compare multiple approaches in a pairwise fashion. Friedman test is recommended in [26] because it is simple, safe, and robust, compared with parametric tests. It is implemented in function `FriedmanTest`. The result can be presented graphically by CD diagram as implemented in function `plotNemenyiTest`. Figure is an example of the result Nemenyi test which was conducted to compare 8 classifiers over 13 high dimensional biological datasets. This example can be found in file `exampleFriedmanTest`. In this figure, CD stands for crucial difference. If the distance of two methods is greater than CD, then we conclude that they differ significantly.

Conclusions

In order to address the issues of existing NMF implementations, we propose the powerful NMF MATLAB Toolbox which includes basic algorithmic level and advanced data ming level. It enable users to analyze biological data via NMF-based data mining approaches, such as clustering, biclustering, feature extraction, feature selection, and classification. The following are the future works to improve the toolbox. First, we will include more NMF algorithms such as nsNMF, LS-NMF, and supervised NMF. Second, we are very interested in implementing and speeding up Bayesian decomposition which is actually a probabilistic NMF invented independently in the same period as the standard NMF. Third, we will implement more statistical comparison methods. Furthermore, we will investigate the performance of NMF on denoising and data compression.

Availability and requirements

Project name: The NMF Toolbox in MATLAB
Project home page: http://cs.uwindsor/~li11112c/nmf
Operating system(s): Platform independent
Programming language: MATLAB
Other requirements: MATLAB 7.11 or higher
License: GNU GPL Version 3
Any restrictions to use by non-academics: Licence needed

Acknowledgements

This research has been partially supported by IEEE CIS Walter Karplus Summer Research Grant 2010, Ontario Graduate Scholarship 2011-2013, and Canadian NSERC Grants #RGPIN228117-2011.

References

1. Lee DD, Seung S: **Learning the Parts of Objects by Non-Negative Matrix Factorization**. *Nature* 1999, **401**:788–791.

2. Brunet J, Tamayo P, Golub T, Mesirov J: **Metagenes and Molecular Pattern Discovery Using Matrix Factorization**. *PNAS* 2004, **101**(12):4164–4169.

3. Kim H, Park H: **Sparse Non-Negatice Matrix Factorization via Alternating Non-Negativity-Constrained Least Aquares for Microarray Data Analysis**. *SIAM J. Matrix Analysis and Applications* 2007, **23**(12):1495–1502.

4. Carmona-Saez P, Pascual-Marqui RD, Tirado F, Carazo JM, Pascual-Montano A: **Biclustering of Gene Expression Data by Non-Smooth Non-Negative Matrix Factorization**. *BMC Bioinformatics* 2006, **7**:78.

5. Wang G, Kossenkov A, Ochs M: **LS-NMF: A Modified Non-Negative Matrix Factorization Algorithm Utilizing Uncertainty Estimates**. *BMC Bioinformatics* 2006, **7**:175.

6. Li Y, Ngom A: **A New Kernel Non-Negative Matrix Factorization and Its Application in Microarray Data Analysis**. In *CIBCB*, IEEE CIS Society, Piscataway: IEEE Press 2012:371–378.

7. Cichocki A, Zdunek R: **NMFLAB - MATLAB Toolbox for Non-Negative Matrix Factorization**. Tech. rep. 2006, [http://www.bsp.brain.riken.jp/ICALAB/nmflab.html].

8. **The NMF: DTU Toolbox**. Tech. rep., Technical University of Denmark[http://www.bsp.brain.riken.jp/ICALAB/nmflab.html].

9. Liu S: **NMFN: Non-negative Matrix Factorization**. Tech. rep., Duke University 2011, [http://cran.r-project.org/web/packages/NMFN].

10. Gaujoux1 R, Seoighe C: **A Flexible R Package for Nonnegative Matrix Factorization**. *BMC Bioinformatics* 2010, **11**:367, [http://cran.r-project.org/web/packages/NMF].

11. Qi Q, Zhao Y, Li M, Simon R: **Non-Negative Matrix Factorization of Gene Expression Profiles: A Plug-in for BRB-ArrayTools**. *Bioinformatics* 2009, **25**(4):545–547.

12. Lee D, Seung S: **Algorithms for Non-Negative Matrix Factorization**. In *Advances in Neural Information Processing Systems*, MIT Press 2001:556–562.

13. Kim H, Park H: **Nonnegative Matrix Factorization based on Alternating Nonnegativity Constrained Least Squares and Active Set Method**. *SIAM J. Matrix Analysis and Applications* 2008, **30**(2):713–730.

14. Ding C, Li T, Jordan MI: **Convex and Semi-Nonnegative Matrix Factorizations**. *TPAMI* 2010, **32**:45–55.

15. Tibshirani R: **Regression Shrinkage and Selection via The Lasso**. *Journal of the Royal Statistical Society. Series B (Methodological)* 1996, **58**:267–288.

16. Zhang D, Zhou Z, Chen S: **Non-Negative Matrix Factorization on Kernels**. *LNCS* 2006, **4099**:404–412.

17. Ding C, Li T, Peng W, Park H: **Orthogonal Nonnegative Matrix Tri-Factorizations for Clustering**. In *KDD*, ACM, New York: ACM 2006:126–135.

18. *PhD thesis*.

19. Ochs M, Fertig E: **Matrix Factorization for Transcriptional Regulatory Network Inference**. In *CIBCB*, IEEE CIS Society, Piscataway: IEEE Press 2012:387–396.

20. Tu B, Kudlicki A, Rowicka M, McKnight S: **Logic of the Yeast Metabolic Cycle: Temporal Compartmentalization of Cellular Processes**. *Science* 2005, **310**:1152–1158.

21. Madeira S, Oliveira A: **Biclustering Algorithms for Biological Data Analysis: A Survey**. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2004, **1**:24–45.

22. Khan J: **Classification and Diagnostic Prediction of Cancers Using Gene Expression Profiling and Artificial Neural Networks**. *Nature Medicine* 2001, **7**(6):673–679.

23. Hu Z: **The Molecular Portraits of Breast Tumors are Conserved Across Microarray Platforms**. *BMC Genomics* 2006, **7**:96.

24. Troyanskaya O, Cantor M, Sherlock G, Brown P, Hastie T, Tibshirani R, Botstein D, Altman R: **Missing Value Estimation Methods for DNA Microarrays**. *Bioinformatics* 2001, **17**(6):520–525.

25. Mukherjee S, Tamayo P, Rogers S, Rifkin R, Engle A, Campbell C, Golub T, Mesirov J: **Estimating dataset size requirements for classifying DNA microarray data**. *Journal of Computational Biology* 2003, **10**(2):119–142.

26. Demsar J: **Statistical Comparisons of Classifiers over Multiple Data Sets**. *Journal of Machine Learning Research* 2006, **7**:1–30.

Figures

Figure 1 - Heat map of NMF clustering on a yeast metabolic cycle time-series data

The left is the gene expression time-series data where each column corresponds to a gene, the middle is the basis matrix, and the right is the coefficient matrix.

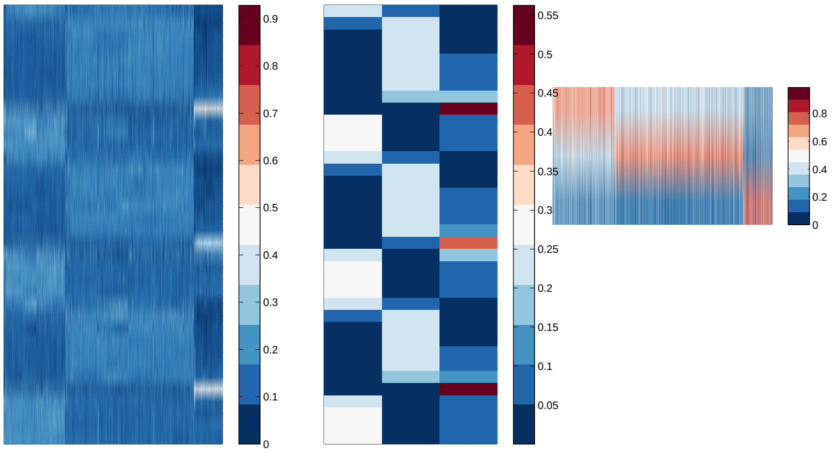


Figure 2 - Biological processes discovered by NMF on a yeast metabolic cycle time-series data

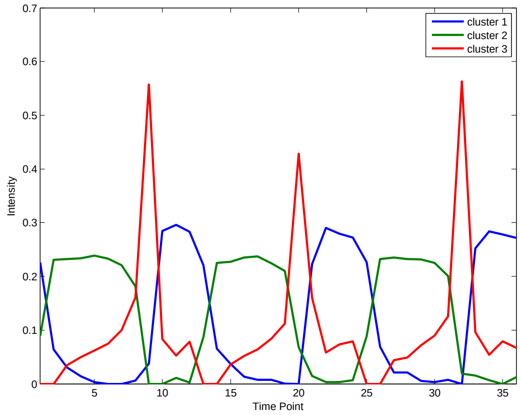


Figure 3 - heat map of NMF biclustering

The left is the gene expression data where each column corresponds to a sample, the middle is the basis matrix, and the right is the coefficient matrix.

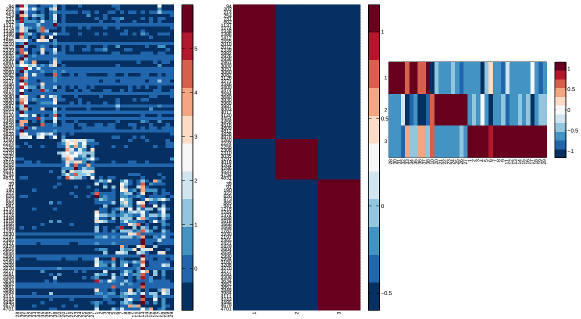


Figure 4 - The mean accuracy and standard deviation of NMF-based feature extraction on SRBCT data

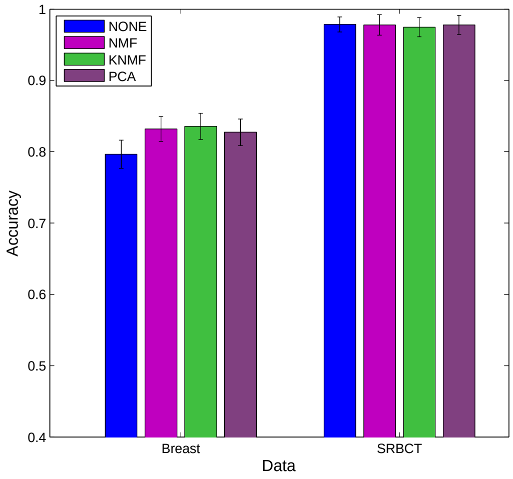


Figure 5 - The mean accuracy of NNLS classifier for different missing rate on SRBCT data

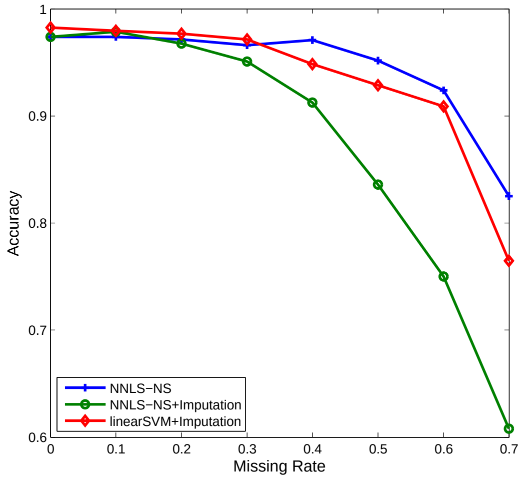


Figure 6 - The mean accuracy of NNLS classifier for different amount of noise on SRBCT data

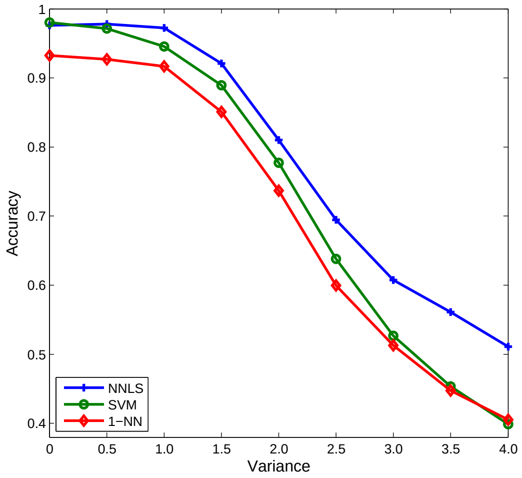


Figure 7 - The fitted Learning curves of NNLS and SVM classifiers on SRBCT data

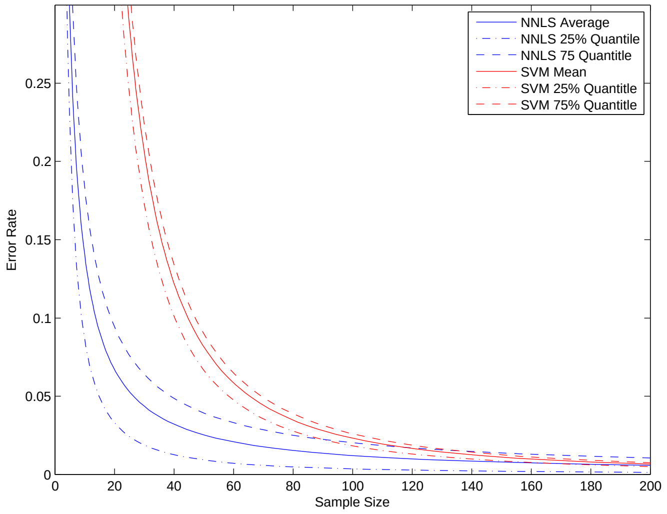
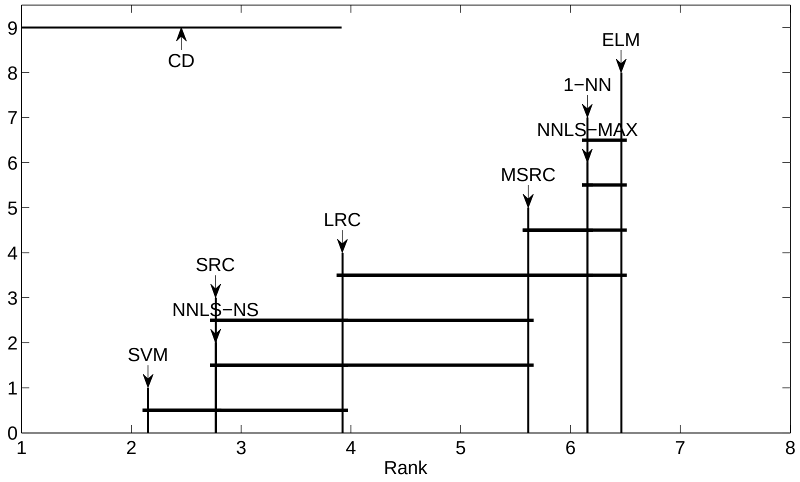


Figure 8 - Graphical representation of Nemenyi test for comparing 8 classifiers over 13 high dimensional biological data



Tables

Table 1 - Algorithms of NMF variants

Function	Description
nmfrule	The standard NMF optimized by gradient-descent-based multiplicative rules.
nmfnnls	The standard NMF optimized by NNLS active-set algorithm.
seminmfrule	Semi-NMF optimized by multiplicative rule.
seminmfnnls	Semi-NMF optimized by NNLS.
sparsenmfnnls	Sparse-NMF optimized by NNLS.
sparsenmfNNQP	Sparse-NMF optimized by NNQP.
sparseseminmfnnls	Sparse semi-NMF optimized by NNLS.
kernelnmfdecom	Kernel NMF through decomposing the kernel matrix of input data.
kernelseminmfrule	Kernel semi-NMF optimized by multiplicative rule.
kernelseminmfnnls	Kernel semi-NMF optimized by NNLS.
kernelsparseseminmfnnls	Kernel sparse semi-NMF optimized by NNLS.
kernelSparseNMFNNQP	Kernel sparse semi-NMF optimized by NNQP.
convexnmfrule	Convex-NMF optimized by multiplicative rules.
kernelconvexnmf	Kernel convex-NMF optimized by multiplicative rules.
orthnmfrule	Orth-NMF optimized by multiplicative rules.
wnmfrule	Weighted-NMF optimized by multiplicative rules.
nmf	The omnibus of the above algorithms.
computeKernelMatrix	Compute the kernel matrix k(A,B) given a kernel function.

Table 2 - NMF-based data mining approaches

Function	Description
NMFCluster chooseBestk biCluster	Take the coefficient matrix produced by a NMF algorithm, and output the clustering result. Search the best number of clusters based on dispersion Coefficients. The biclustering method using one of the NMF algorithms.
featureExtractionTrain featureExtractionTest	General interface. Using training data, generate the bases of the NMF feature space. General interface. Map the test/unknown data into the feature space.
featureFilterNMF featSel	On training data, select features by various NMFs. feature selection methods.
nnlsClassifier perform changeClassLabels01 gridSearchUniverse classificationTrain classificationPredict multiClassifiers cvExperiment	The NNLS classifier. Evaluate the classifier performance. Change the class labels to be in $\{0, 1, 2, \cdots, C-1\}$ for C-class problem. a framework to do line or grid search. Train a classifier, many classifiers are included. Predict the class labels of unknown samples via the model learned by classificationTrain. Run multiple classifiers on the same training data. Conduct experiment of k-fold cross-validation on a dataset.
significantAcc learnCurve FriedmanTest plotNemenyiTest	Check if the given data size can obtain significant accuracy. Fit the learning curve. Friedman test with post-hoc Nemenyi test to compare multiple classifiers on multiple datasets. Plot the CD diagram of Nemenyi test.
NMFHeatMap NMFBicHeatMap plotBarError	Draw and save the heat maps of NMF clustering. Draw and save the heat maps of NMF biclustering. Plot Bars with STD.
normmean0std1 sparsity MAT2DAT	Normalization to have mean 0 and STD 1. Calculate the sparsity of a matrix. Write a dataset from MATLAB into .dat format in order to be readable by other languages.