Source Code for Biology and Medicine

The non-negative matrix factorization toolbox for biological data mining

Source Code for Biology and Medicine 2013, **8**:10 doi:10.1186/1751-0473-8-10

Yifeng Li (li11112c@uwindsor.ca)
Alioune Ngom (angom@uwindsor.ca)

ISSN	1751-0473
Article type	Methodology
Submission date	30 November 2012
Acceptance date	10 April 2013
Publication date	16 April 2013
Article URL	http://www.scfbm.org/content/8/1/10

The non-negative matrix factorization toolbox for biological data mining

Yifeng Li[1]*
*Corresponding author
Email: li11112c@uwindsor.ca

Alioune Ngom[1]
Email: angom@uwindsor.ca

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

Abstract

Background

Non-negative matrix factorization (NMF) has been introduced as an important method for mining biological data. Though there currently exists packages implemented in R and other programming languages, they either provide only a few optimization algorithms or focus on a specific application field. There does not exist a complete NMF package for the bioinformatics community, and in order to perform various data mining tasks on biological data.

Results

We provide a convenient MATLAB toolbox containing both the implementations of various NMF techniques and a variety of NMF-based data mining approaches for analyzing biological data. Data mining approaches implemented within the toolbox include data clustering and bi-clustering, feature extraction and selection, sample classification, missing values imputation, data visualization, and statistical comparison.

Conclusions

A series of analysis such as molecular pattern discovery, biological process identification, dimension reduction, disease prediction, visualization, and statistical comparison can be performed using this toolbox.

Keywords

Non-negative matrix factorization, Clustering, Bi-clustering, Feature extraction, Feature selection, Classification, Missing values

Background

Non-negative matrix factorization (NMF) is a matrix decomposition approach which decomposes a non-negative matrix into two low-rank non-negative matrices [1]. It has been successfully applied in the mining of biological data.

For example, Ref. [2,3] used NMF as a clustering method in order to discover the metagenes (i.e., groups of similarly behaving genes) and interesting molecular patterns. Ref. [4] applied *non-smooth NMF* (NS-

NMF) for the biclustering of gene expression data. *Least-squares NMF* (LS-NMF) was proposed to take into account the uncertainty of the information present in gene expression data [5]. Ref. [6] proposed kernel NMF for reducing dimensions of gene expression data.

Many authors indeed provide their respective NMF implementations along with their publications so that the interested community can use them to perform the same data mining tasks respectively discussed in those publications. However, there exists at least three issues that prevent NMF methods from being used by the much larger community of researchers and practitioners in the data mining, biological, health, medical, and bioinformatics areas. First, these NMF softwares are implemented in diverse programming languages, such as R, MATLAB, C++, and Java, and usually only one optimization algorithm is provided in their implementations. It is inconvenient for many researchers who want to choose a suitable NMF method or mining task for their data, among the many different implementations, which are realized in different languages with different mining tasks, control parameters, or criteria. Second, some papers only provide NMF optimization algorithms at a basic level rather than a data mining implementation at a higher level. For instance, it becomes hard for a biologist to fully investigate and understand his/her data when performing clustering or bi-clustering of his data and then visualize the results; because it should not be necessary for him/her to implement these three data mining methods based on a basic NMF. Third, the existing NMF implementations are application-specific, and thus, there exists no systematic NMF package for performing data mining tasks on biological data.

There currently exists NMF toolboxes (which we discuss in this paragraph), however, none of them addresses the above three issues altogether.

NMFLAB [7] is MATLAB toolbox for signal and image processing which provides a user-friendly interface to load and process input data, and then save the results. It includes a variety of optimization algorithms such as multiplicative rules, exponentiated gradient, projected gradient, conjugate gradient, and quasi-Newton methods. It also provide methods for visualizing the data signals and their components, but does not provide any data mining functionality. Other NMF approaches such as semi-NMF and kernel NMF are not implemented within this package.

NMF:DTU Toolbox [8] is a MATLAB toolbox with no data mining functionalities. It includes only five NMF optimization algorithms, such as multiplicative rules, projected gradient, probabilistic NMF, alternating least squares, and alternating least squares with *optimal brain surgery* (OBS) method.

NMFN: Non-negative Matrix Factorization [9] is an R package similar to *NMF:DTU* but with few more algorithms.

NMF: Algorithms and framework for Nonnegative Matrix Factorization [10] is another R package which implements several algorithms and allows parallel computations but no data mining functionalities.

Text to Matrix Generator (TMG) is a MATLAB toolbox for text mining only.

Ref. [11] provides a NMF plug-in for BRB-ArrayTools. This plug-in only implements the standard NMF and semi-NMF and for clustering gene expression profiles only.

Coordinated Gene Activity in Pattern Sets (CoGAPS) [12] is a new package implemented in C++ with R interface. In this package, the *Bayesian decomposition* (BD) algorithm is implemented and used in place of the NMF method for factorizing a matrix. Statistical methods are also provided for the inference of biological processes. CoGAPS can give more precise results than NMF methods [13]. However, CoGAPS uses a Markov chain Monte Carlo (MCMC) scheme for estimating the BD model parameters, which is slower than the NMFs optimization algorithms implemented with the block-coordinate gradient descent scheme.

In order to address the lack of data mining functionalities and generality of current NMF toolboxes, we propose a general NMF toolbox in MATLAB which is implemented in two levels. The basic level is composed of the different variants of NMF, and the top level consists of the diverse data mining methods for biological data. The contributions of our toolbox are enumerated in the following:

1. The NMF algorithms are relatively complete and implemented in MATLAB. Since it is impossible and unnecessary to implement all NMF algorithms, we focus only on well-known NMF representatives. This repository of NMFs allows users to select the most suitable one in specific scenarios.
2. Our NMF toolbox includes many functionalities for mining biological data, such as clustering, bi-clustering, feature extraction, feature selection, and classification.
3. The toolbox also provides additional functions for biological data visualization, such as heat-maps and other visualization tools. They are pretty helpful for interpreting some results. Statistical methods are also included for comparing the performances of multiple methods.

The rest of this paper is organized as below. The implementations of the basis level are first discussed in the next section. After that, examples of implemented data mining tasks at a high level are described. Finally, we conclude this paper and give possible avenues for future research directions.

Implementation

As mentioned above, this toolbox is implemented at two levels. The fundamental level is composed of several NMF variants and the advanced level includes many data mining approaches based on the fundamental level. The critical issues in implementing these NMF variants are addressed in this section. Table 1 summarizes all the NMF algorithms implemented in our toolbox. Users (researchers, students, and practitioners) should use the command `help nmfrule`, for example, in the command line, for help on how to select a given funtion and set its parameters.

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 the error (or residual) and $\boldsymbol{M}_+$ indicates the matrix $\boldsymbol{M}$ is non-negative. Its optimization in the Euclidean space is formulated as

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

Statistically speaking, this formulation is obtained from the log-likelihood function under the assumption of a Gaussian error. If multivariate data points are arranged in the columns of $\boldsymbol{X}$, then $\boldsymbol{A}$ is called the *basis matrix* and $\boldsymbol{Y}$ is called the *coefficient matrix*; each column of $\boldsymbol{A}$ is thus a *basis vector*. The interpretation is that each data point is a (sparse) non-negative linear combination of the basis vectors. It is well-known that the optimization objective is a non-convex optimization problem, and thus, *block-coordinate descent* is the main prescribed optimization technique for such problem. Multiplicative update rules were introduced in [14] for solving Equation (2). Though simple to implement, this algorithm is not guaranteed to converge to a stationary point [15]. Essentially the optimizations above,

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 rules.
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.
sparsenmf2rule	Sparse-NMF on both factors optimized by multiplicative rules.
sparsenmf2nnqp	Sparse-NMF on both factors optimized by NNQP.
vsmf	Versatile sparse matrix factorization optimized by NNQP and l_1QP.
nmf	The omnibus of the above algorithms.
computeKernelMatrix	Compute the kernel matrix k(A,B) given a kernel function.

with respect to $\boldsymbol{A}$ and $\boldsymbol{Y}$, are *non-negative least squares* (NNLS). Therefore we implemented the alternating NNLS algorithm proposed in [15]. It can be proven 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. Ref. [16] extended it to s*emi-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:

$$\min_{\boldsymbol{A},\boldsymbol{Y}} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{Y}\|_{\mathrm{F}}^2, \text{subject to } \boldsymbol{Y} \geq 0. \quad (3)$$

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 [16] 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

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

where $\boldsymbol{Y}^{\dagger} = \boldsymbol{Y}^{\mathrm{T}}(\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}})^{-1}$ is Moore-Penrose pseudoinverse. 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-variance and non-unique solutions, 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 [17], and has already been used in order to reduce the non-uniqueness of solutions and also and enhance interpretability of the NMF results. The *sparse-NMF* proposed in [3] is expressed in the following equation

$$\min_{\boldsymbol{A},\boldsymbol{Y}} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{Y}\|_{\mathrm{F}}^2 + \frac{\eta}{2}\|\boldsymbol{A}\|_F^2 + \frac{\lambda}{2}\sum_{i=1}^{n}\|\boldsymbol{y}_i\|_1^2 \quad (5)$$
$$\text{subject to } \boldsymbol{A}, \boldsymbol{Y} \geq 0,$$

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

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

We optimize this using three alternating steps in each iteration. First, we optimize the following task:

$$\min_{\boldsymbol{Y}} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{Y}\|_{\mathrm{F}}^2 + \lambda\sum_{i=1}^{n}\|\boldsymbol{y}_i\|_1 \quad (7)$$
$$\text{subject to } \boldsymbol{Y} \geq 0.$$

then, $\boldsymbol{A}$ is updated as follows:

$$\min_{\boldsymbol{A}} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{Y}\|_{\mathrm{F}}^2 \quad (8)$$
$$\text{subject to } \boldsymbol{A} \geq 0.$$

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

$$\min_{\boldsymbol{Z}} \sum_{i=1}^{n} \frac{1}{2}\boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{H}\boldsymbol{z}_i + \boldsymbol{g}_i^{\mathrm{T}}\boldsymbol{z}_i + c_i \quad (9)$$
$$\text{subject to } \boldsymbol{Z} \geq 0,$$

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

$$\min_{\boldsymbol{Y}} \sum_{i=1}^{n} \frac{1}{2}\boldsymbol{y}_i^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})\boldsymbol{y}_i + (\lambda - \boldsymbol{A}^{\mathrm{T}}\boldsymbol{x}_i)^{\mathrm{T}}\boldsymbol{y}_i + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i \quad (10)$$
$$\text{subject to } \boldsymbol{Y} \geq 0.$$

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 functionl `sparseseminmfnnls`.

Versatile sparse matrix factorization

When the training data $\boldsymbol{X}$ is of mixed signs, the basis matrix $\boldsymbol{A}$ is not necessarily constrained to be non-negative; this depends on the application or the intentions of the users. However, without non-negativity, $\boldsymbol{A}$ is not sparse any more. In order to obtain sparse basis matrix $\boldsymbol{A}$ for some analysis, we may use l_1-norm on $\boldsymbol{A}$ to induce sparsity. The drawback of l_1-norm is that correlated variables may not be simultaneously non-zero in the l_1-induced sparse result. This is because l_1-norm is able to produce sparse but non-smooth results. It is known that l_2-norm is able to obtain smooth but non-sparse results. When both norms are used together, then correlated variables can be selected or removed simultaneously [18]. When smoothness is required on $\boldsymbol{Y}$, we may also use l_2-norm on it in some scenarios. We thus generalize the aforementioned NMF models into a versatile form as expressed below

$$\begin{aligned}\min_{\boldsymbol{A},\boldsymbol{Y}} f(\boldsymbol{A},\boldsymbol{Y}) = \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{A}\boldsymbol{Y}\|_F^2 + \sum_{i=1}^{k}(\frac{\alpha_2}{2}\|\boldsymbol{a}_i\|_2^2 \\ + \alpha_1\|\boldsymbol{a}_i\|_1) + \sum_{i=1}^{n}(\frac{\lambda_2}{2}\|\boldsymbol{y}_i\|_2^2 + \lambda_1\|\boldsymbol{y}_i\|_1) \qquad (11)\\ \text{subject to } \begin{cases}\boldsymbol{A}\geq 0 & \text{i.e., if } t_1 = 1\\ \boldsymbol{Y}\geq 0 & \text{i.e., if } t_2 = 1\end{cases},\end{aligned}$$

where, parameters: $\alpha_1 \geq 0$ controls the sparsity of the basis vectors; $\alpha_2 \geq 0$ controls the smoothness and the scale of the basis vectors; $\lambda_1 \geq 0$ controls the sparsity of the coefficient vectors; $\lambda_2 \geq 0$ controls the smoothness of the coefficient vectors; and, parameters t_1 and t_2 are boolean variables (0: false, 1: true) which indicate if non-negativity needs to be enforced on $\boldsymbol{A}$ or $\boldsymbol{Y}$, respectively. We can call this model *versatile sparse matrix factorization* (VSMF). It can be easily seen that the standard NMF, semi-NMF, and the sparse-NMFs are special cases of VSMF.

We devise the following multiplicative update rules for the VSMF model in the case of $t_1 = t_2 = 1$ (implemented in function `sparsenmf2rule`):

$$\begin{cases}\boldsymbol{A} = \boldsymbol{A} * \frac{\boldsymbol{X}\boldsymbol{Y}^{\mathrm{T}}}{\boldsymbol{A}\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}+\alpha_2\boldsymbol{A}+\alpha_1}\\ \boldsymbol{Y} = \boldsymbol{Y} * \frac{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X}}{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Y}+\lambda_2\boldsymbol{Y}+\lambda_1}\end{cases}, \qquad (12)$$

where, $\boldsymbol{A} * \boldsymbol{B}$ and $\frac{\boldsymbol{A}}{\boldsymbol{B}}$ are the element-wise multiplication and division operators of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively. Alternatively, we also devise an active-set algorithm for VSMF (implemented in function `vsmf`). When $t_1(\text{or } t_2) = 1$, $\boldsymbol{A}$ (or $\boldsymbol{Y}$) can be updated by NNQP (this case is also implemented in `sparsenmf2nnqp`). When $t_1(\text{or } t_2) = 0$, $\boldsymbol{A}$ (or $\boldsymbol{Y}$) can be updated using 1_1QP.

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 basis matrix is dependent on the dimension of the data, and it is difficult to represent it in a very high (even infinite) dimensional space. We notice that in the NNLS optimization, updating $\boldsymbol{Y}$ in Equation (10) needs only the 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

$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = (\boldsymbol{Y}^{\dagger})^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{Y}^{\dagger}$, $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X} = (\boldsymbol{Y}^{\dagger})^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$. Therefore, we can see that only the inner product $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ is needed in the optimization of NMF. Hence, we can obtain the kernel version, *kernel-NMF*, by replacing the inner product $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ with a kernel matrix $K(\boldsymbol{X}, \boldsymbol{X})$. Interested readers can refer to our recent paper [6] for further details. Based on the above derivations, we implemented the kernel semi-NMF using multiplicative update rule (in `kernelseminmfrule`) and 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 a kernel matrix proposed in [19] is implemented in `kernelnmfdecom`.

Other variants

Ref. [16] proposed the *Convex-NMF*, in which the columns of $\boldsymbol{A}$ are constrained to be the convex combinations of data points in $\boldsymbol{X}$. It is formulated as $\boldsymbol{X}_{\pm} = \boldsymbol{X}_{\pm}\boldsymbol{W}_{+}\boldsymbol{Y}_{+} + \boldsymbol{E}$, where $\boldsymbol{M}_{\pm}$ indicates that matrix $\boldsymbol{M}$ is of mixed signs. $\boldsymbol{X}\boldsymbol{W} = \boldsymbol{A}$ 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 demonstrated that the columns of $\boldsymbol{A}$ obtained with the convex-NMF are close to the real centroids of clusters. Convex-NMF can be kernelized as well [16]. We implemented the convex-NMF and its kernel version in `convexnmfrule` and `kernelconvexnmf`, respectively.

The basis vectors obtained with the above NMFs are non-orthogonal. Alternatively, *orthogonal NMF* (ortho-NMF) imposes the orthogonality constraint in order to enhance sparsity [20]. Its formulation is

$$\begin{aligned} &\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}\boldsymbol{Y} + \boldsymbol{E} \\ &\text{subject to } \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \boldsymbol{I}, \quad \boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} = \boldsymbol{I}, \quad \boldsymbol{A}, \boldsymbol{S}, \boldsymbol{Y} \geq 0, \end{aligned} \tag{13}$$

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. Ortho-NMF is very similar with the *non-negative sparse PCA* (NSPCA) proposed in [21]. The disjoint property on ortho-NMF may be too restrictive for many applications, therefore this property is relaxed in NSPCA. Ortho-NMF does not guarantee the maximum-variance property which is also relaxed in NSPCA. However NSPCA only enforces non-negativity on the basis vectors, even when the training data have negative values. We plan to devise a model in which the disjoint property, the maximum-variance property, the non-negativity and sparsity constraints can be controlled on both basis vectors and coefficient vectors.

There are two efficient ways of applying NMF on data containing missing values. First, the missing values can be estimated prior to running NMF. Alternatively, *weighted-NMF* [22] can be directly applied to decompose the data. Weighted-NMF puts a zero weight on the missing elements and hence only the non-missing data contributes to the final result. An expectation-maximization (EM) based missing value estimation during the execution of NMF may not be efficient. The weighted-NMF is given in our toolbox in function `wnmfrule`.

Results and discussion

Based on the various implemented NMFs, a number of data mining tasks can be performed via our toolbox. Table 2 lists the data mining functionalities we provide in this level. These mining tasks are also described along with appropriate examples.

Table 2 NMF-based data mining approaches

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

Clustering and bi-clustering

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 largest coefficient in the i-th column of $\boldsymbol{Y}$ indicates the cluster this data point belongs to. The reason is that if the data points are mainly composed with the same basis vectors, they should therefore be in the same group. A basis vector is usually viewed as a cluster centroid or prototype. This approach has been used in [2] for clustering microarray data and in order to discover tumor subtypes. We implemented function `NMFCluster` through which various NMF algorithms can be selected. An example is provided in `exampleCluster` file in the folder of our toolbox.

The task of interpreting both the basis matrix and the coefficient is equivalent to simultaneously clustering the rows and columns of matrix $\boldsymbol{X}$. This is bi-clustering and the interested readers can refer to [23] for an excellent survey on bi-clustering algorithms and to [4] for a bi-clustering method based on NMF. We implemented a bi-clustering approach based on NMF in `biCluster` function. The bi-clusters can be visualized via the function `NMFBicHeatMap`. We applied NMF to simultaneously grouping the genes and samples of a leukemia data set [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 bi-cluster. The heat-map is shown in Figure 1. Readers can find the script in `exampleBiCluster` file of our toolbox.

Figure 1 Heat map of NMF biclustering result. Left: the gene expression data where each column corresponds to a sample. Center: the basis matrix. Right: the coefficient matrix.

Basis vector analysis for biological process discovery

We can obtain interesting and detailed interpretations via an appropriate analysis of the basis vectors. When applying NMF on a microarray data, the basis vectors are interpreted as potential biological processes [3,13,24]. In the following, we give one example for finding biological factors on gene-sample data, and two examples on time-series data. Please note they only serve as simple examples. Fine tuning of the parameters of NMF is necessary for accurate results.

First example

We ran our VSMF on the ALLAML gene-sample data of [2] with the settings $k = 3$, $\alpha_1 = 0.01$, $\alpha_2 = 0.01$, $\lambda_1 = 0$, $\lambda_2 = 0.01$, $t_1 = 1$, and $t_2 = 1$. Next, we obtain 81, 37, and 448 genes for the three factors, respectively. As in [3], we then performed gene set enrichment analysis (GSEA) by applying Onto-Express [25] on each of these sets of genes. Part of the result is shown in Table 3. We can see that the factor-specific genes selected by NMF correspond to some biological processes significantly. Please see file `exampleBioProcessGS` in the toolbox for details. GSEA can also be done using other tools, such as MIPS [26], GOTermFinder [27], and DAVID [28,29].

Table 3 Gene set enrichment analysis using Onto-Express for the factor specific genes identified by NMF

Factor 1		Factor 2		Factor 3	
biological process	**p-value**	**biological process**	**p-value**	**biological process**	**p-value**
reproduction (5)	0	response to stimulus (15)	0.035	regulation of bio. proc. (226)	0.009
metabolic process (41)	0	biological regulation(14)	0.048	multi-organism proc. (39)	0.005
cellular process (58)	0			biological regulation (237)	0.026
death (5)	0				
developmental process (19)	0				
regulation of biological process (19)	0				

Second example

We used NMF to cluster a time-series data of yeast metabolic cycle in [30]. Figure 2 shows the heat-map of NMF clustering, and Figure 3 shows 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 `exampleBioProcessTSYeast` 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 [30].

Figure 2 Heat map of NMF clustering result on yeast metabolic cycle time-series data. Left: the gene expression data where each column corresponds to a sample. Center: the basis matrix. Right: the coefficient matrix.

Figure 3 Biological processes discovered by NMF on yeast metabolic cycle time-series data.

Third example

We used NMF to factorize a breast cancer time-series data set, which includes wild type MYCN cell lines and mutant MYCN cell lines [31]. The purpose of this example is to show that NMF is a potential tool to finding cancer drivers. One basic methodology is in the following. First, basis vectors are produced applying NMF on a time-series data. Then factor-specific genes are identified by computational or statistical methods. Finally, the regulators of these factor-specific genes are identified from any prior biological knowledge. This data set has 8 time points (0, 2, 4, 8, 12, 24, 36, 48 hr.). The zero time point is untreated and samples were collected at the subsequent time points after treatment with 4-hydroxytamoxifen (4-OHT). In our computational experiment, we use our VSMF implementation (function `vsmf`). we set $k = 2$. Because this data set has negative values we set $t_1 = 0$ and $t_2 = 1$. We set $\alpha_1 = 0.01$, $\alpha_2 = 0$, $\lambda_1 = 0$, and $\lambda_2 = 0.01$. The basis vectors of both wild-type and mutant data are compared in Figure 4. From the wild-type time-series data, we can successfully identify two patterns. The rising pattern corresponds to the induced signature and the falling pattern corresponding to the repressed signature in [31]. It is reported in [31] that the MYC target genes contributes to both patterns. From the mutant time-series, we can obtain two flat processes, which are reasonable. The source code of this example can be found in `exampleBioProcessMYC`. We also recommend the readers to see the methods based on matrix decompositions which are proposed in [13,32] and devised for identifying signaling pathways.

Figure 4 Biological processes discovered by NMF on breast cancer time-series data.

Basis vector analysis for gene selection

The columns of $\boldsymbol{A}$ for a gene expression data set are called *metasamples* in [2]. They can be interpreted as biological processes, because their values imply the activation or inhibition of some the genes. Gene selection aims to find marker genes for disease prediction and to understand the pathways they contribute to. 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 processes via NMF are biologically meaningful for class discrimination in disease prediction, and the genes expressed differentially across these processes contribute to better classification performance in terms of accuracy. In Figure 1 for example, three biological processes are discovered and only the 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 with gene ontology analysis. Feature selection based on supervised NMF will also be implemented.

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 issues of *curse of dimensionality*. For example, it is impossible to estimate the parameters of some statistical models since the number of their parameters grow exponentially as the dimension increases. Another issue is that biological data are usually noisy; which crucially affects the performances of classifiers applied on the data. In cancer study, a common hypothesis is that only a few biological factors (such as the oncogenes) play a crucial role in the development of a given cancer. When we generate data from control and sick patients, the high-dimensional data will contain a large number of irrelevant or redundant information. Orthogonal factors obtained with *principal component* *analysis* (PCA) or *independent component analysis* (ICA) are not appropriate in most cases. Since NMF generates non-orthogonal (and non-negative) factors, therefore it is much reasonable to extract important and interesting 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

$$\boldsymbol{X} \approx \boldsymbol{A}\boldsymbol{Y}_{\text{tr}}, \text{subject to } \boldsymbol{A}, \boldsymbol{Y}_{\text{tr}} \geq 0, \tag{14}$$

where, $\boldsymbol{Y}_{\text{tr}}$ means that $\boldsymbol{Y}$ is obtained from the training data. 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 unknown samples $\boldsymbol{S}_{m \times p}$ into this feature space, we have to solve the following non-negative least squares problem:

$$\boldsymbol{S} \approx \boldsymbol{A}\boldsymbol{Y}_{\text{uk}}, \text{subject to } \boldsymbol{Y}_{\text{uk}} \geq 0, \tag{15}$$

where, $\boldsymbol{Y}_{\text{uk}}$ means the $\boldsymbol{Y}$ is obtained from the unknown samples. After obtaining $\boldsymbol{Y}_{\text{tr}}$ and $\boldsymbol{Y}_{\text{uk}}$, the learning and prediction steps can be done quickly in the k-dimensional feature space instead of the m-dimensional original space. A classifier can learn over $\boldsymbol{Y}_{\text{tr}}$, and then predicts 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 very useful in the understanding of the underlying biological processes, as mentioned above.

We have implemented a pair of functions `featureExtractionTrain` and `featureExtractionTest` including many linear and kernel NMF algorithms. The basis matrix (or, the inner product of basis matrices in the kernel case) is learned from the training data via the function `featureExtractionTrain`, and the unknown samples can be projected onto the feature space via the function `featureExtractionTest`. We give examples of how to use these functions in files `exampleFeatureExtraction` and `exampleFeatureExtractionKernel`.

Figure 5 shows the classification performance of SVM *without* dimension reduction and SVM *with* dimension reduction using linear NMF, kernel NMF with *radial basis function* (RBF) kernel, and PCA on two data sets, SRBCT [33] and Breast [34]. Since ICA is computationally costly, we did not include it in the comparisons. 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 to 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 at least comparable to that without using any dimension reduction. As future work, supervised NMF will be investigated and implemented in order to extract discriminative features.

Figure 5 Mean accuracy and standard deviation results of NMF-based feature extraction on SRBCT data.

Classification

If we make the 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 in which the training samples are the basis vectors. Since the optimization process within NMF is a NNLS problem, we call this classification approach the *NNLS classifier* [35]. A NNLS problem is essentially a quadratic programming problem as formulated in Equation (9), therefore, only inner products are needed for the optimization. We thus can naturally extend the NNLS classifier to kernel version. Two features of this approach are that: i) the sparsity regularization help avoid overfitting the model; and ii) the kernelization allows a dimension-free optimization and also linearizes the non-linear complex patterns within the data. The implementation of the NNLS classifier is in file `nnlsClassifier`. Our toolbox also provides many other classification approaches including SVM classifier. Please see file `exampleClassification` for demonstration. In our experiment of 4-fold cross-validation, accuracies of 0.7865 and 0.7804 are respectively obtained with linear and kernel (RBF) NNLS classifier on Breast data set. They achieved accuracies of 0.9762 and 0.9785, respectively, over SRBCT data.

Biological data are usually noisy and sometimes contains missing values. A strength of the NNLS classifier are that it is robust to noise and to missing values, making NNLS classifiers quite suitable for classifying biological data [35].

In order to show its robustness to noise, we added a Gaussian noise of mean 0 and variance from 0 to 4 with increment 0.5 on SRBCT. Figure 6 illustrates the results of NNLS, SVM, and *1-nearest neighbor* (1-NN) classifiers using this noisy data. It can be seen that as the noise increases, NNLS outperforms SVM and 1-NN significantly.

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

To deal with the missing value problem, three strategies are usually used: incomplete sample or feature removal, missing value imputation (i.e., estimation), and ignoring missing values. Removal methods may delete important or useful information for classification and particularly when there is a large percentage of missing values in the data. Imputation methods may create false data depending on the magnitude of the true estimation errors. The third method ignores using the missing values during classification. Our approach in dealing with the missing value problem is also to ignore them. The NNLS optimization needs only the inner products of pairs of samples. Thus, 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 we take their inner product. As an example, we randomly removed between 10% to 70% of data values in STBCT data. Using such incomplete data, we compared our method with the zero-imputation method (that is, estimating all missing values as 0). In Figure 7, we can see that the NNLS classifier using our missing value approach outperforms the zero-imputation method in the case of large missing rate. Also, the more sophisticated k-nearest neighbor imputation (KNNimpute) method [36] will fail on data with in high percentage of missing values.

Figure 7 The mean accuracy results of NNLS classifier for different missing value rates on SRBCT data.

Statistical comparison

The toolbox provides two methods for statistical comparisons and evaluations of different methods. The first is a two-stage method proposed in [37]. The importance of this method is that it can estimate the data-size requirement for attaining a significant accuracy and extrapolate the performance based on the current available data. Generating biological data is usually very expensive and thus this method can help researchers to evaluate the necessity of producing more data. At the first stage, the minimum data size required for obtaining a significant accuracy is estimated. This is implemented in function `significantAcc`. The second stage is to fit the learning curve using the error rates of large data sizes. It is implemented in function `learnCurve`. In our experiments, we found that the NNLS classifier usually requires fewer number of samples for obtaining a significant accuracy. For example on SRBCT data, NNLS requires only 4 training samples while SVM needs 19 training samples. The fitted learning curves of NNLS and SVM classifiers are shown in Figure 8. We provide an example of how to plot this figure in file `exampleFitLearnCurve`.

Figure 8 The fitted learning curves of NNLS and SVM classifiers on SRBCT data.

The second method is the nonparametric Friedman test coupled with post-hoc Nemenyi test to compare multiple classifiers over multiple data sets [38]. It is difficult to draw an overall conclusion if we compare multiple approaches in a pairwise fashion. Friedman test has been recommended in [38] because it is simple, safe and robust, compared with parametric tests. It is implemented in function `FriedmanTest`. The result can be presented graphically using the *crucial difference* (CD) diagram as implemented in function `plotNemenyiTest`. CD is determined by significant level α. Figure 9 is an example of the result of the Nemenyi test for comparing 8 classifiers over 13 high dimensional biological data sets. This example can be found in file `exampleFriedmanTest`. If the distance of two methods is greater than the CD then we conclude that they differ significantly.

Figure 9 Nemenyi test comparing 8 classifiers over 13 high dimensional biological data ($\alpha = 0.05$).

Conclusions

In order to address the issues of the existing NMF implementations, we propose a NMF Toolbox written in MATLAB, which includes a basic NMF optimization level and an advanced data mining level. It enable users to analyze biological data via NMF-based data mining approaches, such as clustering, bi-clustering, feature extraction, feature selection, and classification.

The following are the future works planned in order to improve and augment 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 the Bayesian decomposition method which is actually a probabilistic NMF introduced independently in the same period as the standard NMF. Third, we will include more statistical comparison and evaluation methods. Furthermore, we will investigate the performance of NMF for denoising and for data compression.

Availability and requirements

Project name: The NMF Toolbox in MATLAB
Project home page: https://sites.google.com/site/nmftool and http://cs.uwindsor.ca/~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

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

YL did comprehensive survey on NMF, implemented the toolbox, and drafted this manuscript. AN supervised the whole project, provided constructive suggestions on this toolbox, and wrote the final manuscript. All authors read and approved the final manuscript.

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 Anal Appl* 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. Gaujoux 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. Fertig E, Ding J, Favorov A, Parmigiani G, Ochs M: **CoGAPS: an R/C++ package to identify patterns and biological process activity in transcriptomic data.** *Bioinformatics* 2010, **26**(21):2792–2793.

13. Ochs M, Fertig E: **Matrix factorization for transcriptional regulatory network inference.** In *CIBCB*, IEEE CIS Society. Piscataway: IEEE Press; 2012:387–396.

14. Lee D, Seung S: **Algorithms for non-negative matrix factorization.** In *NIPS*. Cambridge: MIT Press; 2001:556–562.

15. Kim H, Park H: **Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method.** *SIAM J Matrix Anal Appl* 2008, **30**(2):713–730.

16. Ding C, Li T, Jordan MI: **Convex and semi-nonnegative matrix factorizations.** *TPAMI* 2010, **32:**45–55.

17. Tibshirani R: **Regression shrinkage and selection via the lasso.** *J R Stat Soc Ser B (Methodol)* 1996, **58:**267–288.

18. Zou H, Hastie T: **Regularization and variable selection via the elastic Net.** *J R Stat Soc- Ser B: Stat Methodol* 2005, **67**(2):301–320.

19. Zhang D, Zhou Z, Chen S: **Non-negative matrix factorization on kernels.** *LNCS* 2006, **4099:**404–412.

20. Ding C, Li T, Peng W, Park H: **Orthogonal nonnegative matrix tri-factorizations for clustering.** In *KDD*. New York: ACM; 2006:126–135.

21. Zass R, Shashua A: **Non-negative sparse PCA.** In *NIPS*. Cambridge: MIT Press; 2006.

22. Ho N: **Nonnegative matrix factorization algorithms and applications.** *PhD thesis*, Louvain-la-Neuve: Belgium; 2008.

23. Madeira S, Oliveira A: **Biclustering algorithms for biological data analysis: A survey.** *IEEE/ACM Trans Comput Biol Bioinformatics* 2004, **1:**24–45.

24. Kim P, Tidor B: **Subsystem identification through dimensionality reduction of large-scale gene expression data.** *Genome Res* 2003, **13:**1706–1718.

25. Draghici S, Khatri P, Bhavsar P, Shah A, Krawetz S, Tainsky M: **Onto-tools, the toolkit of the modern biologist: onto-express, onto-compare, onto-design and onto-translate.** *Nucleic Acids Res* 2003, **31**(13):3775–3781.

26. Mewes H, Frishman D, Gruber C, Geier B, Haase D, Kaps A, Lemcke K, Mannhaupt G, Pfeiffer F, Schuller C, Stocker S, Weil B: **MIPS: A database for genomes and protein sequences.** *Nucleic Acids Res* 2000, **28:**37–40.

27. Boyle E, Weng S, Gollub J, Jin H, Botstein D, Cherry J, Sherlock G: **GO::TermFinder – open source software for accessing gene ontology information and finding significantly enriched gene ontology terms associated with a list of genes.** *Bioinformatics* 2004, **20:**3710–3715.

28. Huang D, Sherman B, Lempicki R: **Systematic and integrative analysis of large gene lists using DAVID Bioinformatics resources.** *Nature Protoc* 2009, **4:**44–57.

29. Huang D, Sherman B, Lempicki R: **Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists.** *Nucleic Acids Res* 2009, **37:**1–13.

30. Tu B, Kudlicki A, Rowicka M, McKnight S: **Logic of the yeast metabolic cycle: temporal compartmentalization of cellular processes.** *Science* 2005, **310:**1152–1158.

31. Chandriani S, Frengen E, Cowling V, Pendergrass S, Perou C, Whitfield M, Cole M: **A core MYC gene expression signature is prominient in basal-like breast cancer but only partially overlaps the core serum response.** *PloS ONE* 2009, **4**(5):e6693.

32. Ochs M, Rink L, Tarn C, Mburu S, Taguchi T, Eisenberg B, Godwin A: **Detection of treatment-induced changes in signaling pathways in sastrointestinal stromal tumors using transcripttomic data.** *Cancer Res* 2009, **69**(23):9125–9132.

33. Khan J: **Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks.** *Nat Med* 2001, **7**(6):673–679.

34. Hu Z: **The molecular portraits of breast tumors are conserved across microarray platforms.** *BMC Genomics* 2006, **7:**96.

35. Li Y, Ngom A: **Classification approach based on non-negative least squares.** *Neurocomputing* 2013, in press.

36. 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.

37. 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.** *J Comput Biol* 2003, **10**(2):119–142.

38. Demsar J: **Statistical comparisons of classifiers over multiple data sets.** *J Mach Learn Res* 2006, **7:**1–30.

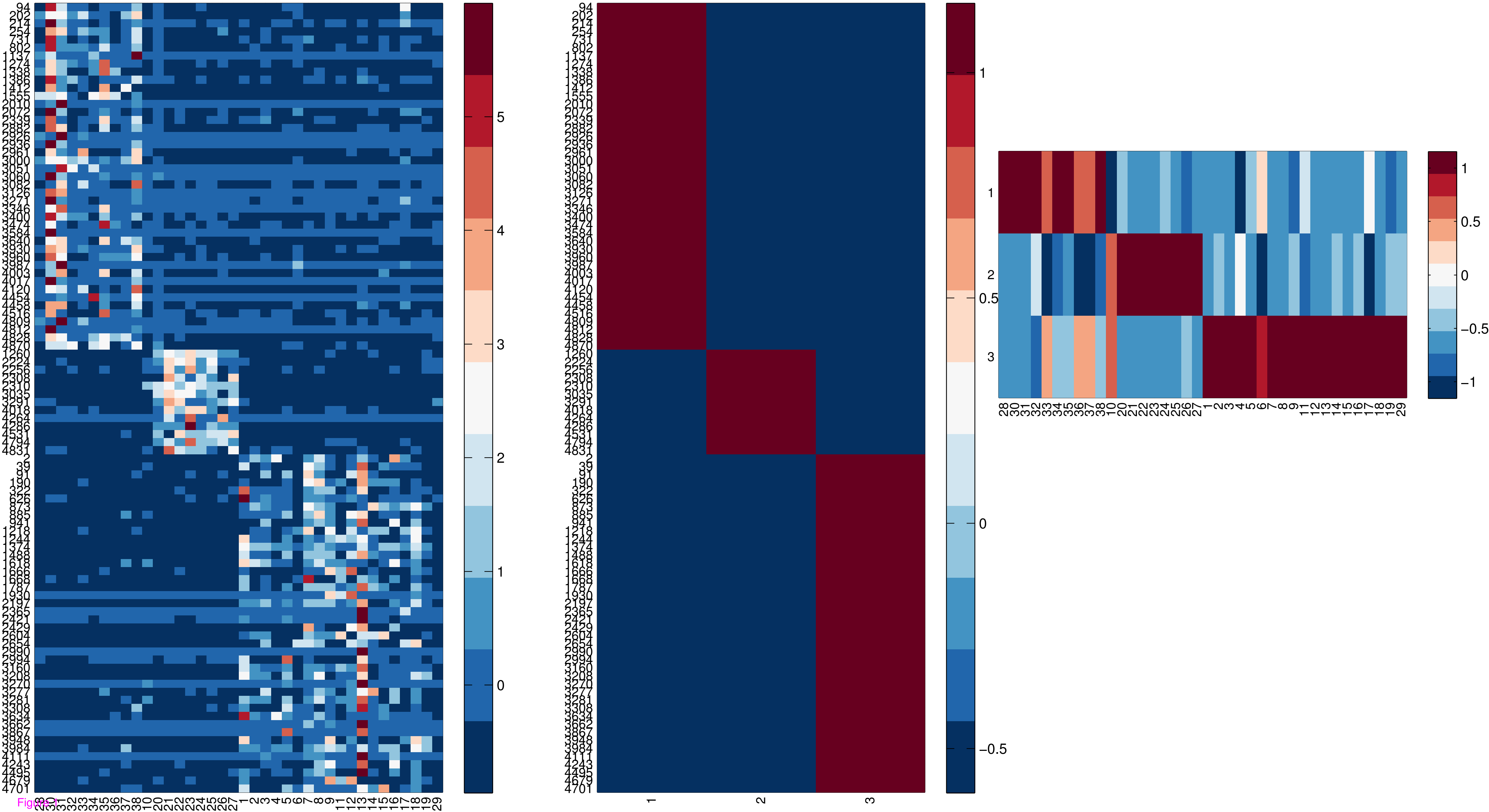


Figure 1

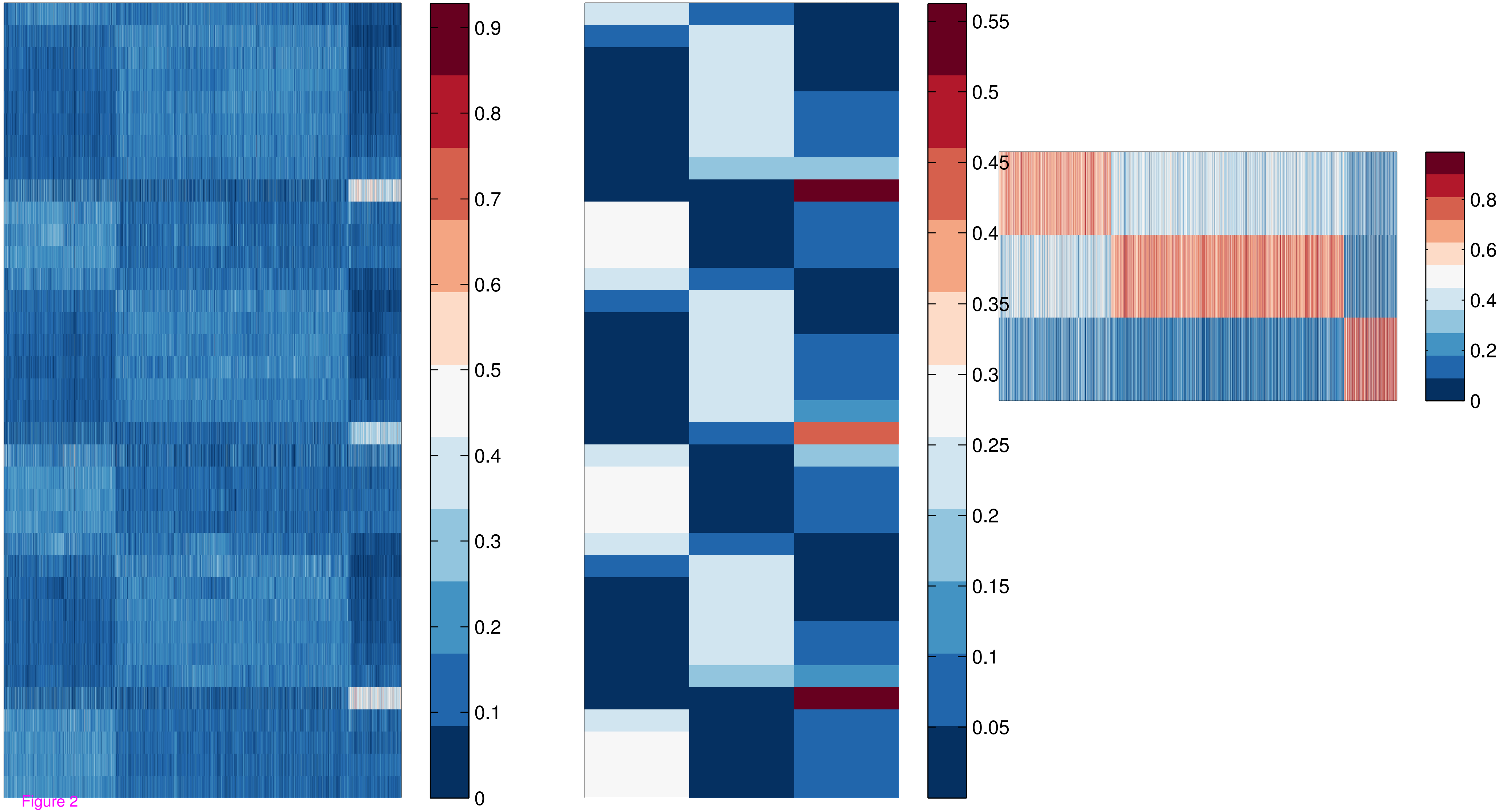
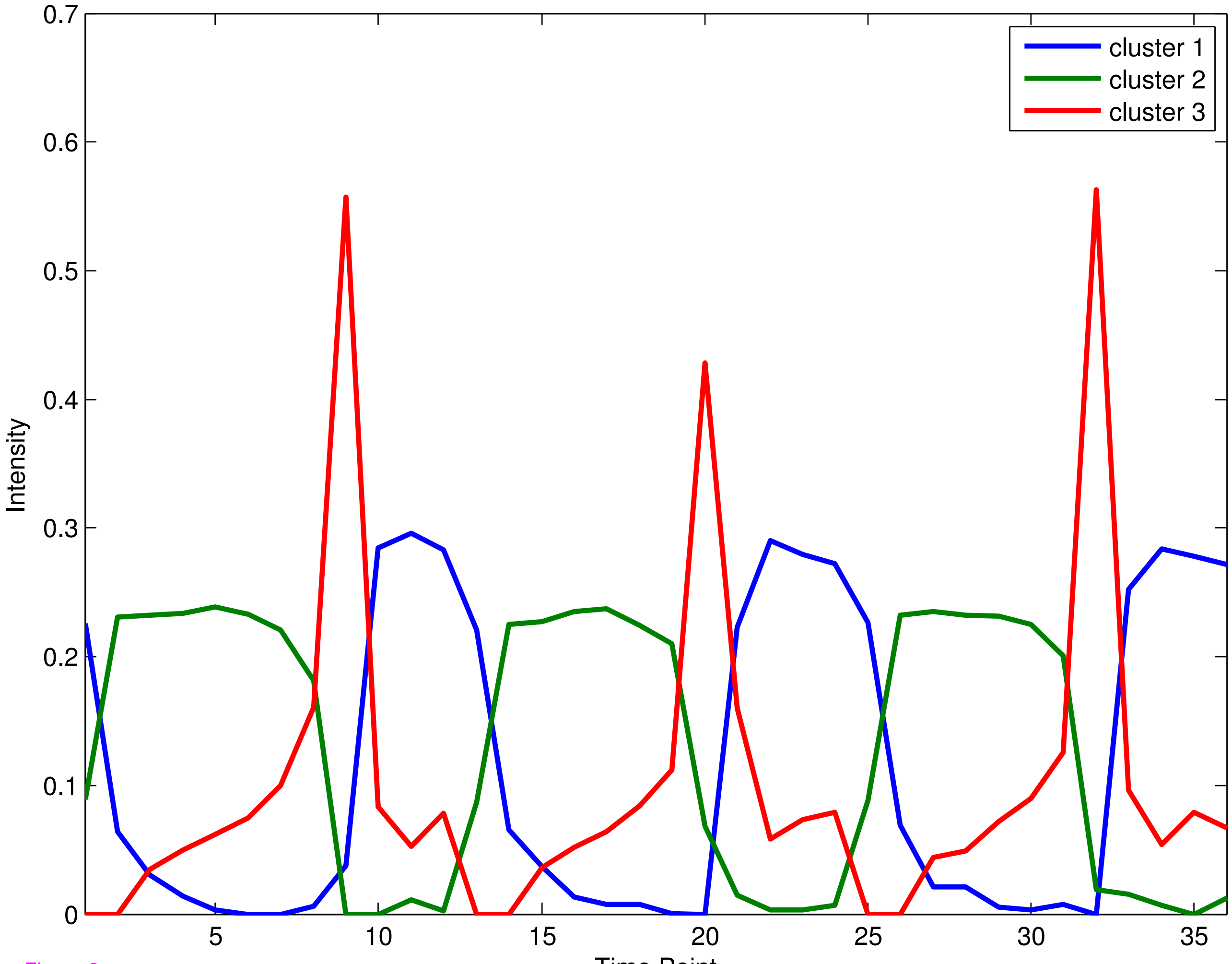

0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0.8
0.6
0.4
0.2
0

Figure 2



0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Intensity
5
10
15
20
25
30
35
Time Point
cluster 1
cluster 2
cluster 3

Figure 3

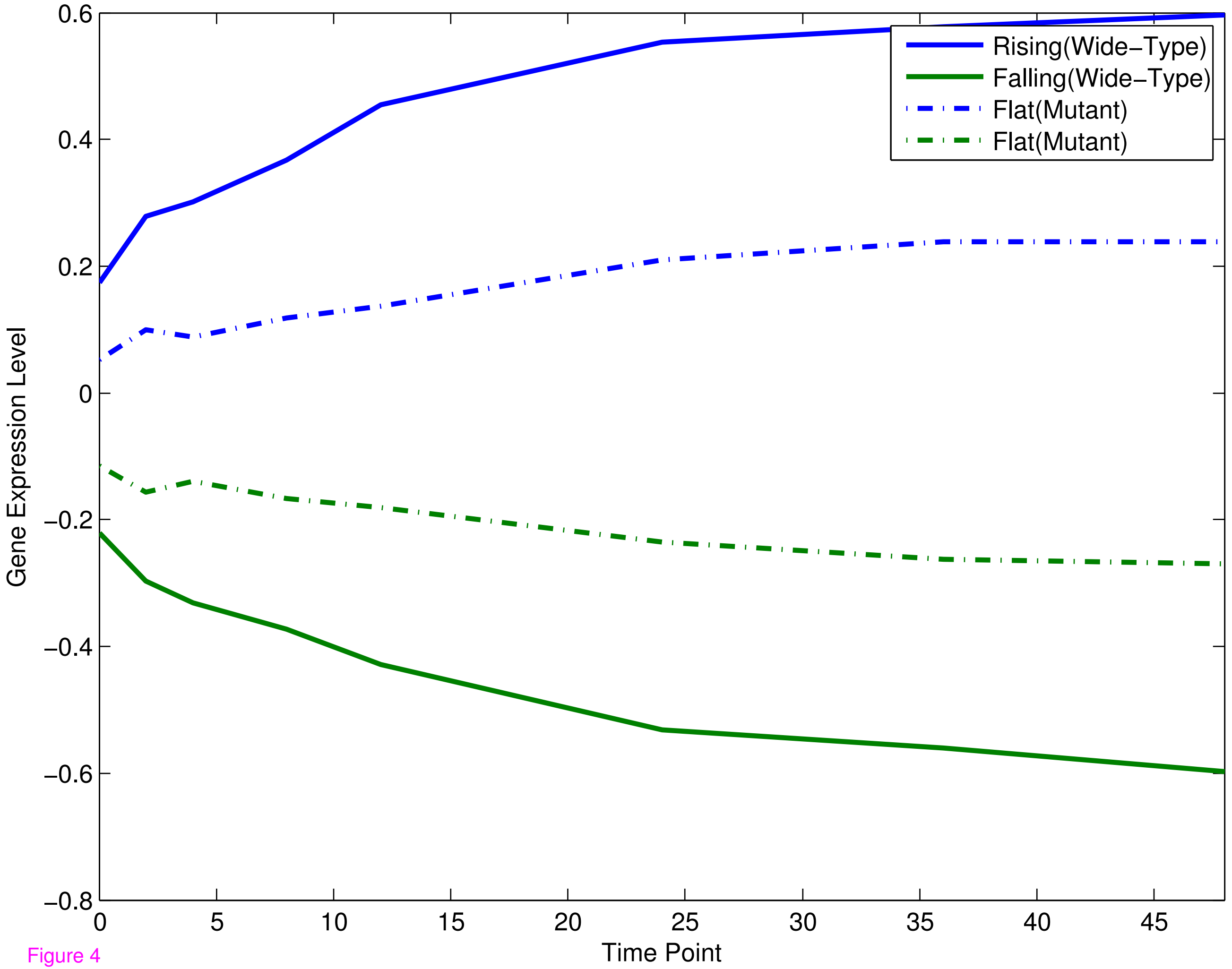
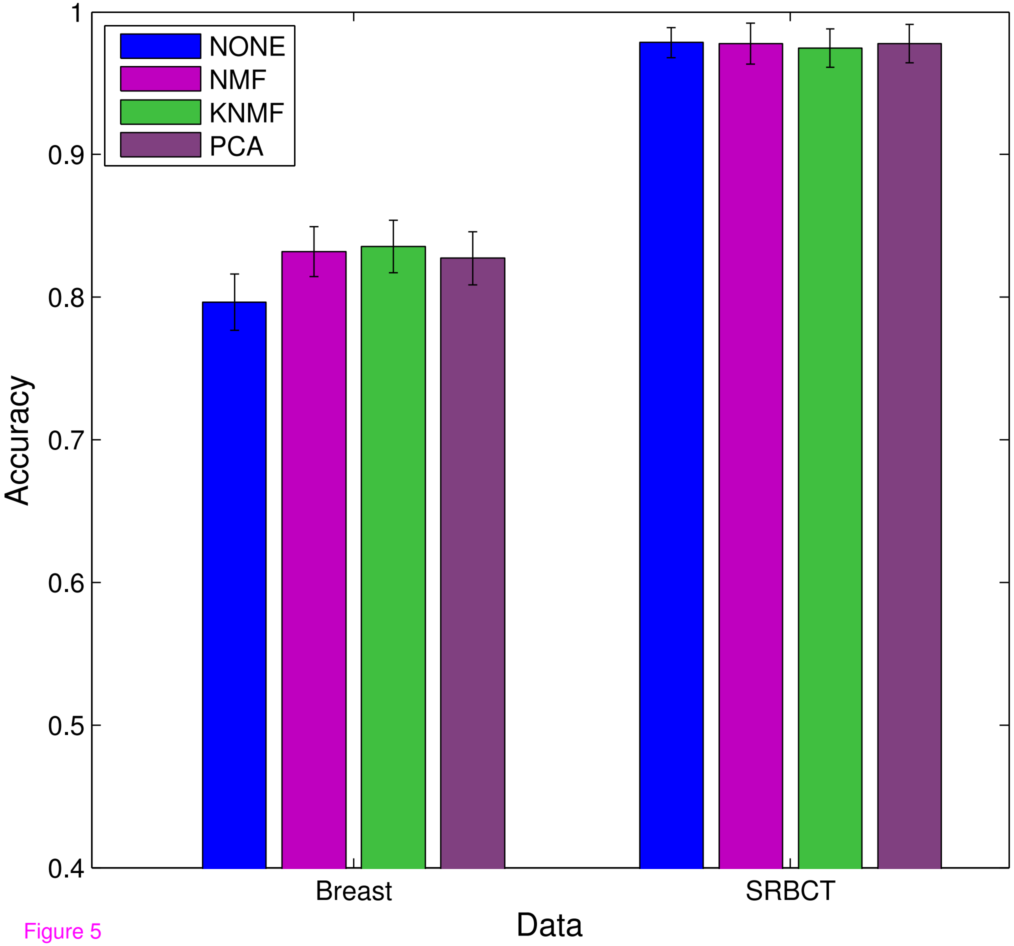
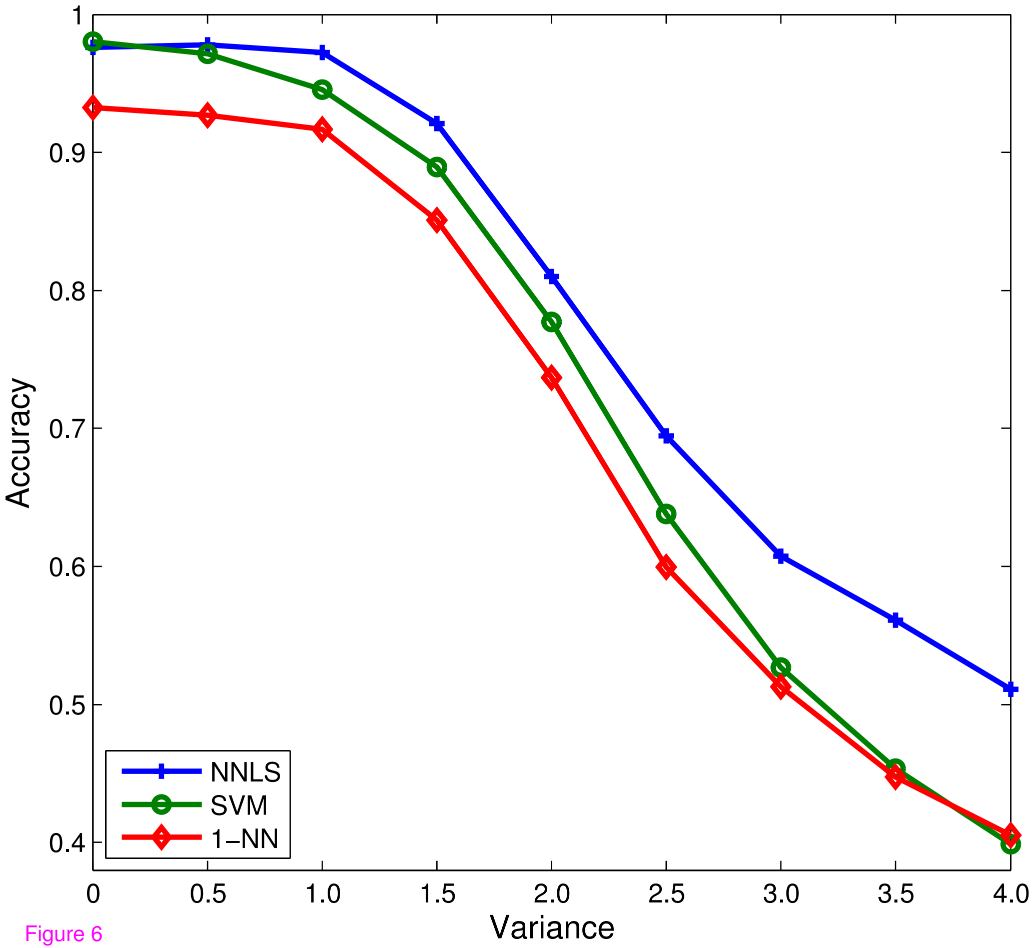

Rising(Wide−Type)
Falling(Wide−Type)
Flat(Mutant)
Flat(Mutant)
Gene Expression Level
Time Point
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
0
5
10
15
20
25
30
35
40
45

Figure 4



1
0.9
0.8
0.7
0.6
0.5
0.4
Accuracy
NONE
NMF
KNMF
PCA
Breast
SRBCT
Data

Figure 5



Accuracy
Variance
NNLS
SVM
1−NN
1
0.9
0.8
0.7
0.6
0.5
0.4
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Figure 6

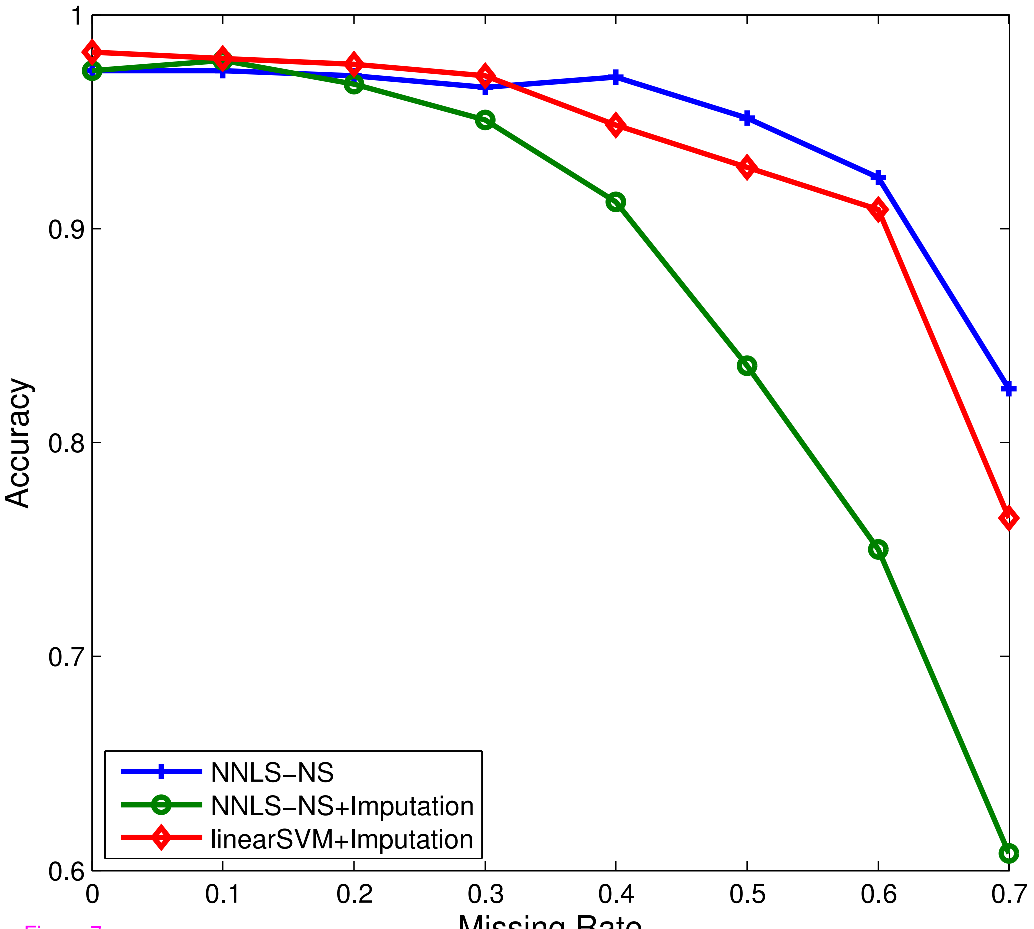
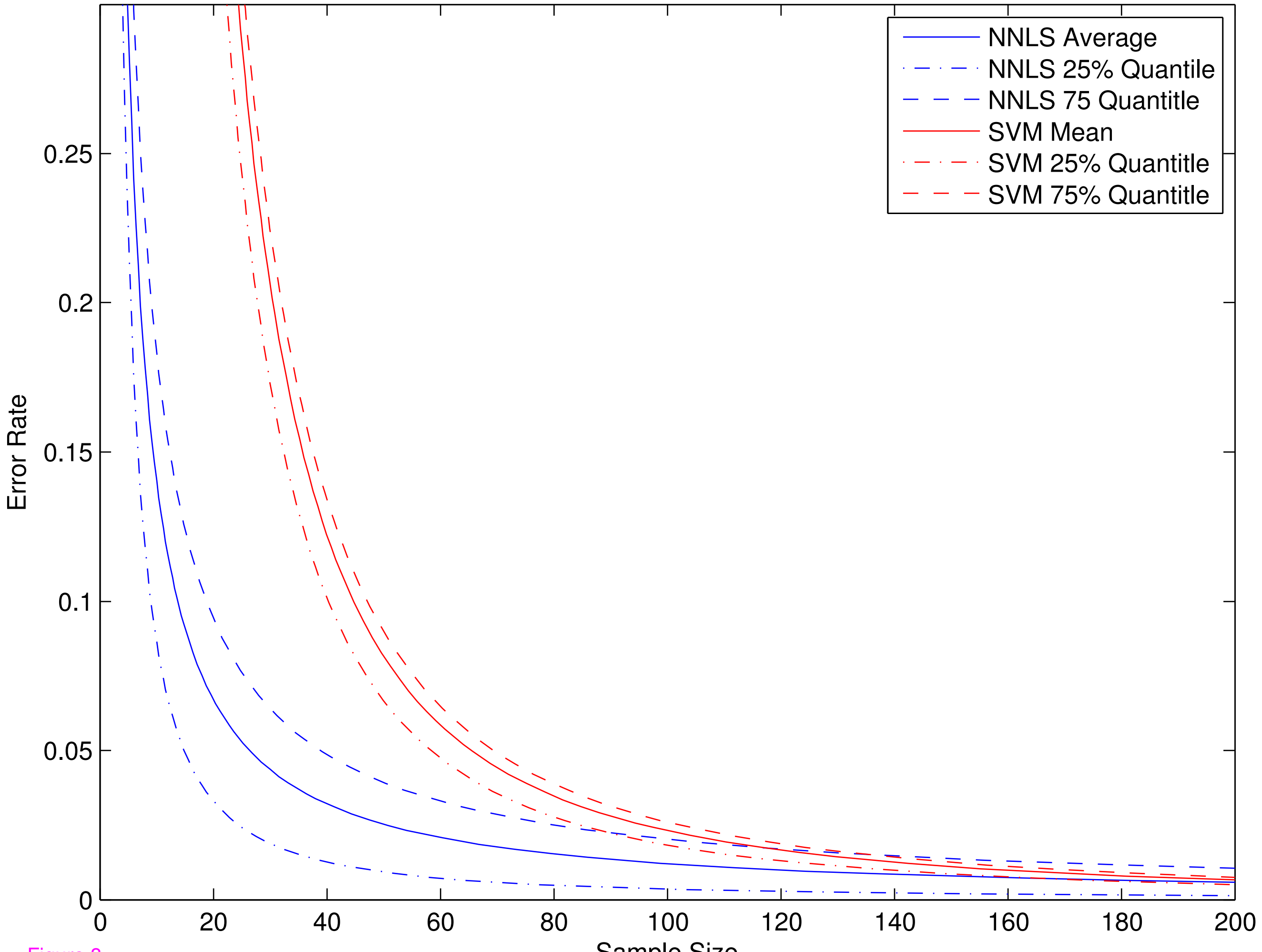

1
0.9
0.8
0.7
0.6
Accuracy
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Missing Rate
NNLS−NS
NNLS−NS+Imputation
linearSVM+Imputation

Figure 7



NNLS Average
NNLS 25% Quantitle
NNLS 75 Quantitle
SVM Mean
SVM 25% Quantitle
SVM 75% Quantitle
Error Rate
0.25
0.2
0.15
0.1
0.05
0
0
20
40
60
80
100
120
140
160
180
200
Sample Size

Figure 8

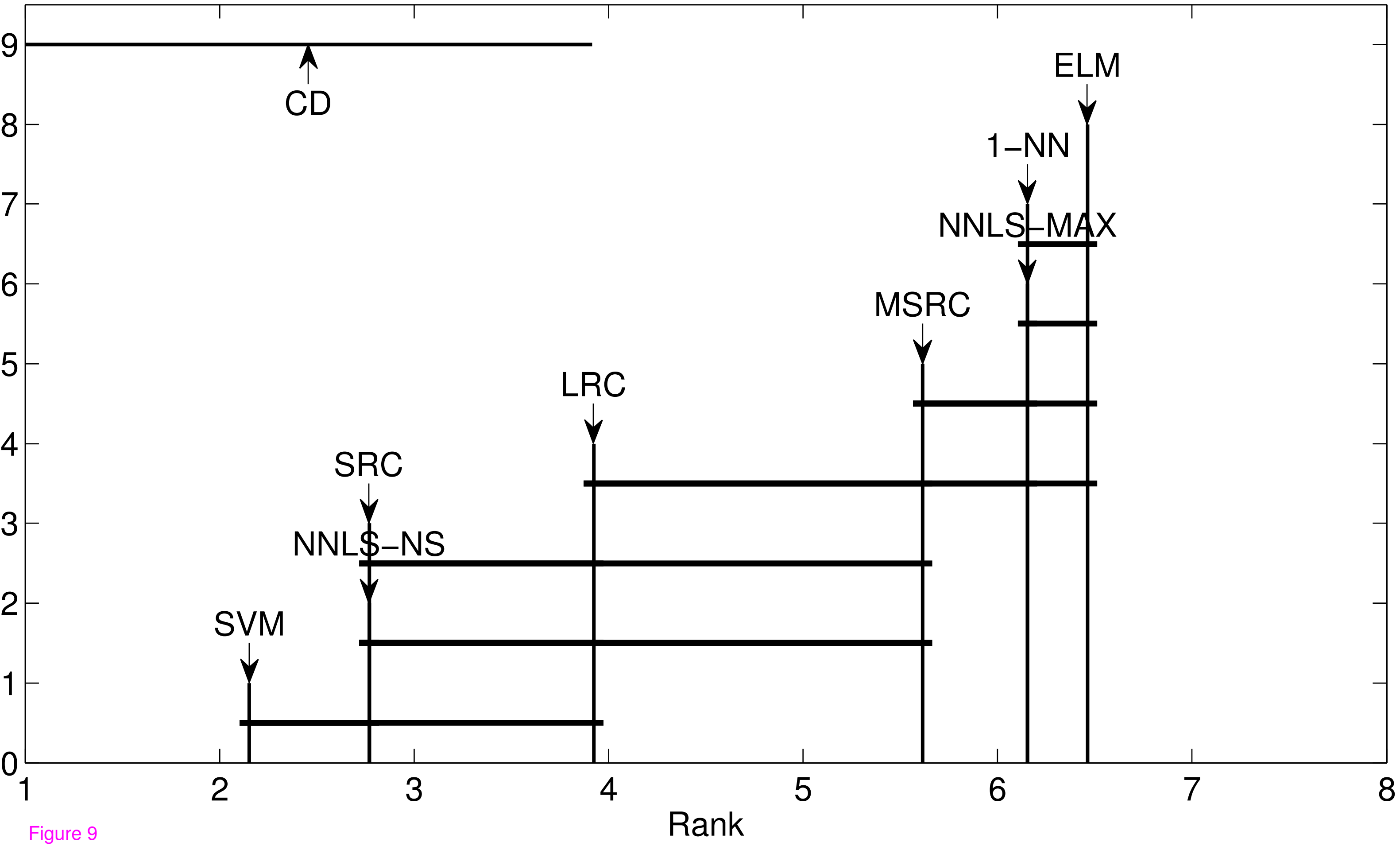

CD
ELM
1−NN
NNLS−MAX
MSRC
LRC
SRC
NNLS−NS
SVM
Rank
0
1
2
3
4
5
6
7
8
9
1
2
3
4
5
6
7
8

Figure 9