Proc. BIOC 218
Vol. 1, pp. 1-20, March 2005
Genetics

# Critical assessment of microarray analyses for gene identification

KENNETH E. HILD II* **

*Department of Radiology, University of California at San Francisco,
*Contributed by Kenneth Hild, March 15, 2005*

**ABSTRACT** **Clustering of microarray data has been and continues to be an effective tool for identifying the function of novel genes. The size of microarray experiments is currently limited to 10-30 arrays but researchers are likely to migrate towards larger array sizes. Increasing the number of arrays improves the feasibility of several more advanced statistical signal processing methods, which can be used to denoise the data prior to clustering or as a replacement for clustering. Several of these methods, *i.e.*, PCA, SVD, FA, BSP, and ICA, are analyzed for their ability and potential to improve gene identification.**

Microarray data represents the expression of thousands, or even tens of thousands, of different genes as a function of experimental conditions (environments), tissue types (samples), or time. Typically microarray data is processed by clustering the genes into distinct subsets that are mutually exclusive and collectively exhaustive. Assuming that genes that cluster together perform similar or related functions, clustering can be used to help identify the function of unidentified genes. Indeed, much success has already been made along these lines (1).

Current experiments? involve on the order of 10 to 30 arrays. The number of arrays used is likely to increase since it is only a matter of time before future technological advances reduces the cost and effort of data collection. As the array sizes increase then there are a number of statistical signal processing methods that become viable for microarray analyses, *e.g.*, Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Factor Analysis (FA), Beamspace Projections (BSP), and Independent Component Analysis (ICA). Any of these five methods may be used to denoise the microarray data prior to clustering and the last method, ICA, can also be used to separate the environmental and experimental influences in a manner that makes it an appropriate alternative to clustering. Both of these approaches, *i.e.*, denoising/clustering and blind separation, are discussed herein for the purpose of gene identification.

Denoising is a preprocessing step that is applied in hopes of removing extraneous signals and thereby improving the performance of the subsequent clustering. For the methods considered here, the denoising consists of either finding a linear transformation of the microarray data that is optimal in a particular sense or finding/selecting a set of bases which explains well the portion of the observations that are due to the signals of interest.

Separation can be obtained using ICA under certain conditions (9). Whereas the items of interest for denoising are the estimates of the cleaned signals, the item of interest for blind separation is the estimate of the system. The system defines the relationship between each of the individual gene expression levels and the underlying biological and environmental factors that are responsible for changes in expression levels. Since the system describes how much each of these factors contributes to the overall gene expression, the system estimate can be used in place of clustering either the columns or the rows of the gene expression matrix.

## GENERATIVE MODEL OF GENE EXPRESSION

The observations are given by $\boldsymbol{x}(n)$, which is a ($M_x$ $x$ 1) vector that contains the gene expression for each of the $M_x$ genes in the $n^{th}$ array (henceforth the independent variable is commonly assumed to be time without loss of generality). The ($M_x$ $x$ $N$) data matrix is given by $\boldsymbol{X} = [\boldsymbol{x}(0)\ \boldsymbol{x}(1)\ \ldots\ \boldsymbol{x}(N\text{-}1)]$, the constituents of which are the vectors of the $N$ arrays/time points. The gene expression data is found using one of two methods. The first method, which is in common use, is to extract RNA from both experimental and reference samples. The experimental samples are labeled with a red fluorescent dye during reverse

transcription, whereas the reference samples are labeled with a green fluorescent dye (1), (8). The samples are mixed and then placed on a, *e.g.,* spotted DNA microarray that contains the vast majority of the ORF's for the organism in question. The final observation, $\boldsymbol{x}(n)$, represents the log of the ratio of the quantity of biological material labeled with red dye with respect to that of the green dye. The data is trimmed so that the minimum value is -3 and the maximum value is +3. The second method does not use a reference sample nor is clipping applied. Hence the final observation represents the amount of biological material labeled with only the red dye. This method of data collection is required for three of the denoising methods listed in the following section, FA/ICA, BSP, and ICA, since they assume that there is a linear relationship between the biological sources and the observations.

Clustering is usually applied to ratio data, which is not a problem since linearity is not assumed. However, using ratio data for clustering does have one important drawback. The observability of a particular gene depends heavily on the mean expression values of the sample and the reference. Genes that have a small mean expression (relative to the random fluctuations about the mean) are much more likely to produce a noticeable change in the observed gene expression data, whereas genes that have a large mean expression for both the sample and the reference will have log ratios that are nearly constant over all arrays. Hence these latter genes are ignored in the subsequent clustering. This problem is avoided by the denoising methods that are based on the linear model assumption since their performance is invariant to the mean value.

**Model.** Several of the methods to be described are based on knowing a generative model for the observations. Even though the other methods do not require a model, it is convenient to provide one since it provides a mechanism by which the different methods can be unambiguously compared (using simulated data) and it makes explicit several otherwise non-conspicuous assumptions of the different gene identification methods. The inferences that follow are based on this model and, therefore, are only as valid as the model on which they are based. The generative model used here is given by,

$$\boldsymbol{x}(n) = \boldsymbol{A}\boldsymbol{s}(n) + \boldsymbol{B}\boldsymbol{u}(n) + \boldsymbol{v}(n) \qquad (1)$$

where $\boldsymbol{s}(n)$ is the ($M_s$ *x* 1) vector of zero-mean biological factors of interest, $\boldsymbol{u}(n)$ is the ($M_u$ *x* 1) vector of biological factors of no interest, and $\boldsymbol{v}(n)$ is the ($M_v$ *x* 1) vector of zero-mean Gaussian-distributed random noise. Likewise the ($M_x$ *x* $M_s$) matrix $\boldsymbol{A}$ and the ($M_x$ *x* $M_u$) matrix $\boldsymbol{B}$ contain the strength of the relationship between the factors and the gene expressions. The signals $\boldsymbol{s}(n)$, $\boldsymbol{u}(n)$, and $\boldsymbol{v}(n)$ may be expressed in matrix notation as $\boldsymbol{S}$, $\boldsymbol{U}$, and $\boldsymbol{V}$, respectively, in the same manner used for defining $\boldsymbol{X}$. The problem of finding $\boldsymbol{A}$ or $\boldsymbol{s}(n)$ in Equation (1) is identical to that addressed in the Blind Source Separation (BSS) literature, where the factors are commonly referred to as independent components, $\boldsymbol{A}$ is referred to as the mixing matrix, and the remainder is either lumped together as noise or ignored altogether. Notice that only $\boldsymbol{u}(n)$ is allowed to have a mean differing from zero. This is done only as a matter of convenience. Also, $\boldsymbol{s}(n)$ is referred to hereafter as the set of sources, $\boldsymbol{u}(n)$ as the interference or interfering sources, and $\boldsymbol{v}(n)$ as the noise.

**Example.** The signals for this model can be explained using a trivial example. Suppose that the temperature and serum levels are the experimental conditions of interest and that they are either measured or are externally controlled. The $M_s = 2$ sources are then the two time courses of the gene expressions due to these two experimental conditions. The $\boldsymbol{A}$ matrix contains two column vectors, one for each of these factors, whose $ij^{th}$ element is zero if the $j^{th}$ factor has no effect on the $i^{th}$ gene and is positive/negative if the $j^{th}$ factor up-regulates/down-regulates the $i^{th}$ gene, respectively. The vector $\boldsymbol{u}(n)$ and matrix $\boldsymbol{B}$ are similarly defined except that they account for all other (interfering) cell activities. The interference is assumed to be active both before and during the experiment as opposed to $\boldsymbol{s}(n)$, which is assumed to be active only during the experiment. Consequently the baseline gene expression is simply $\boldsymbol{B}\boldsymbol{u}(n)+\boldsymbol{v}(n)$.

This model allows for the possibility that several different functions affect the same gene or set of genes. Since multiple functions can affect an overlapping set of genes it is appropriate to think of the gene expression levels as representing a mixture of multiple functions. It is also trivial in

this model to incorporate the possibility that multiple genes are co-regulated. This is accomplished by making similar the rows of $\boldsymbol{A}$ that pertain to the genes in question.

## DENOISING

The ideal solution for denoising, according to the model given in Equation (1), is $\boldsymbol{As}(n)$. There are numerous possible approaches that can be used for denoising. The following non-comprehensive list includes several approaches that appear in the microarray literature and several methods found in the literature of other fields.

I. PCA
II. SVD
III. FA/ICA
IV. BSP
V. ICA

**I.** PCA computes a data-dependent linear transform that is based on the ($M_x$ *x* $M_x$) symmetric gene correlation matrix, $\mathrm{E}[\boldsymbol{x}(n)\boldsymbol{x}(n)^{\mathrm{T}}]$, or the (*N x N*) symmetric array correlation matrix, $\mathrm{E}[\boldsymbol{x}(n)^{\mathrm{T}}\boldsymbol{x}(n)]$ (11). The expression for the correlation matrix for the former of these two options is given by,

$$\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{T} \tag{2}$$

where $\mathrm{E}[\boldsymbol{x}(n)\boldsymbol{x}(n)^{\mathrm{T}}]$ is approximated well by $\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$ for wide-sense stationary, correlation ergodic data of sufficient length, $\boldsymbol{Q}$ is the ($M_x$ *x* $M_x$) matrix of eigenvectors, and $\boldsymbol{\Lambda}$ is the ($M_x$ *x* $M_x$) diagonal matrix of eigenvalues. The linear transformation for PCA is based on the ($M_x$ *x L*) matrix $\boldsymbol{Q'}$, which consists of the *L* eigenvectors that are associated with the *L* largest eigenvalues. The estimate of the cleaned observations is given by,

$$\bar{\boldsymbol{x}}(n) = \boldsymbol{Q'}\boldsymbol{Q'^{T}}\,\boldsymbol{x}(n) \tag{3}$$

It is well-known that PCA, which does not use a specific model for the noise, is optimal in terms of minimizing the squared error of the reduced-rank representation. It is also well-known that this approach works only if the *L* largest eigenvectors span the space of the sources, which basically requires that the power of $\boldsymbol{As}(n)$ is much larger than the power of $\boldsymbol{Bu}(n)+\boldsymbol{v}(n)$. The problem is that PCA is not appropriate when it is needed most, *i.e.*, for cases of low signal-to-noise ratio.

**II.** SVD can be applied directly to the data matrix, $\boldsymbol{X}$, in order to find a linear transform (11). This is tantamount to using PCA where the matrix of left singular vectors, $\boldsymbol{Q}_{\mathrm{L}}$, corresponds to $\boldsymbol{Q}$, the eigenvector matrix of $\mathrm{E}[\boldsymbol{x}(n)\boldsymbol{x}(n)^{\mathrm{T}}]$, and the matrix of right singular vectors, $\boldsymbol{Q}_{\mathrm{R}}$, corresponds to the eigenvector matrix of $\mathrm{E}[\boldsymbol{x}(n)^{\mathrm{T}}\boldsymbol{x}(n)]$ where,

$$\boldsymbol{X} = \boldsymbol{Q}_L\boldsymbol{\Sigma}\,\boldsymbol{Q}_R^{T} \tag{4}$$

and $\boldsymbol{\Sigma}$ is the matrix of singular values. However, this is not the only manner in which SVD can be applied. Alter *et al*. use SVD to generate a set of basis vectors (4), which are then used in place of the gene expression data. This is an interesting approach, although it raises several questions. SVD produces left and right eigenmatrices that differ in dimensionality. The choice of associating the eigengenes with one of the eigenmatrices and the eigenarray with the other seems quite arbitrary. This point is relevant since swapping these two definitions will produce a different result. This approach, which imposes a biorthogonality constraint on the data, is one that has been called into question many times in the BSS community (16). Additionally, it is not surprising that a periodic process can be fit well with two orthogonal sinusoids having the same periodicity as the process. This choice of basis functions is tantamount to approximating a periodic signal with the first non-dc term of a Fourier series expansion. Nevertheless, this paper introduces an intriguing idea and has proven to be quite thought provoking.

**III.** FA/ICA, unlike PCA, includes a specific model for the noise (10). This is a novel approach to solving the BSS problem that has only recently been submitted for publication (12). The FA/ICA approach for denoising is accomplished using three steps. The first step is to find the maximum likelihood estimate of $\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}+\boldsymbol{\Gamma}$ from the baseline data, $\boldsymbol{Bu}(n)+\boldsymbol{v}(n)$, where the likelihood is found by assuming $\boldsymbol{u}(n)$ and $\boldsymbol{v}(n)$ are multi-variate Gaussian-distributed, the correlation matrix of $\boldsymbol{v}(n)$, denoted by $\boldsymbol{\Gamma}$, is equal to $\gamma\boldsymbol{I}$, the correlation matrix of $\boldsymbol{u}(n)$ is $\boldsymbol{I}$, $\boldsymbol{I}$ is the identity matrix, and $\gamma$ is an unknown parameter to be estimated. Due to an inherent identifiability limitation, $\boldsymbol{B}$ can not be

found in this manner since any rotation of $\boldsymbol{B}$ also maximizes the likelihood for this particular model. Hence the estimate of $\boldsymbol{B}$ used in the first step can be expressed as $\boldsymbol{B}'=\boldsymbol{BR}$, where $\boldsymbol{R}$ is an unknown rotation. This is not a problem since (1) the goal is to find $\boldsymbol{BB}^{\mathrm{T}}$ and (2) the estimate of $\boldsymbol{BB}^{\mathrm{T}}$ is given by $\boldsymbol{B}'\boldsymbol{B}'^{\mathrm{T}}=\boldsymbol{BRR}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}$, which by the definition of a rotation matrix equals $\boldsymbol{BB}^{\mathrm{T}}$. The second step is similar to the first except that it is applied to the data where the sources are active. In this step $\boldsymbol{Bu}(n)+\boldsymbol{v}(n)$ is treated as the noise and the goal is to find the maximum likelihood estimate of $\boldsymbol{A}$, where the likelihood is found by assuming $\boldsymbol{s}(n)$ is a multi-variate Gaussian-distributed variable having a correlation matrix equal to $\boldsymbol{I}$ and the noise has a correlation matrix found from the first step, namely, $\boldsymbol{BB}^{\mathrm{T}}+\boldsymbol{\Gamma}$. The second step has the same identifiability issue as the first so that the estimate of $\boldsymbol{A}$ is correct up to an unknown rotation matrix, $\boldsymbol{R}$. The third step represents $\boldsymbol{s}(n)$ as $\boldsymbol{Ry}(n)$, where each element of $\boldsymbol{y}(n)$ is assumed to be a mutually statistically independent variable having a non-Gaussian distribution and $\boldsymbol{R}$ is a rotation matrix that is found using ICA, this removing the ambiguity. Using this approach the cleaned observations are given by,

$$\bar{\boldsymbol{x}}(n) = \boldsymbol{A}'\boldsymbol{R}'\boldsymbol{y}'(n) \qquad (5)$$

where $\boldsymbol{A}'$ , $\boldsymbol{R}'$, and $\boldsymbol{y}'(n)$ are the estimates of $\boldsymbol{A}$, $\boldsymbol{R}$, and $\boldsymbol{y}(n)$, respectively.

The first step of FA/ICA essentially estimates the correlation matrix of the baseline data. This could be done in the usual fashion, to wit, using $(\boldsymbol{BU}+\boldsymbol{V})(\boldsymbol{BU}+\boldsymbol{V})^{\mathrm{T}}$. The advantage of using FA is that the number of free parameters required to estimate this correlation matrix is reduced from $\mathrm{O}(M_{\mathrm{x}}^2)$ to $\mathrm{O}(M_{\mathrm{x}}M_{\mathrm{s}})$, where $M_{\mathrm{s}}$ is assumed to be much less than $M_{\mathrm{x}}$ in agreement with a publication by Holter *et al.* (5). This is critical for good estimation of the unknown parameters when the number of arrays/time points is small relative to the number of genes. Another nice feature of this approach is that it eliminates the need for ICA to separate the sources from the interference since these are eliminated in the second step. A straightforward ICA approach attempts to separate all the elements of both $\boldsymbol{s}(n)$ and $\boldsymbol{u}(n)$, which is more difficult especially when $M_{\mathrm{u}}$ is large. The validity of FA/ICA is based on the assumption that all signals are i.i.d., $\boldsymbol{u}(n)$ and $\boldsymbol{v}(n)$ are Gaussian, all of the eigenvalues of $\boldsymbol{\Gamma}$ are approximately equal, and also that all of the required conditions of the ICA method are met. These latter items are addressed when ICA is discussed below.

**IV.** BSP is based on a beamforming approach that has been used recently to localize neural sources in magnetoencephalography and electroencephalography data (2). Unlike FA/ICA, BSP does not assume a specific noise model. FA/ICA estimates the statistics of the interfering signals by using full-rank baseline data. This requires data to be collected prior to the point where the experimental conditions are varied. BSP, on the other hand, uses a reduced-rank estimate of the sources. Hence the noise statistics can be approximated as the statistics of the signals that lie in the null-space of the transformation that is used to explain the sources. In this respect BSP is similar to PCA. The difference is that BSP describes the sources using a set of both spatial and temporal basis functions that are optimal for the sources under consideration. Consequently BSP does not require that the power of $\boldsymbol{As}(n)$ is large relative to $\boldsymbol{Bu}(n)+\boldsymbol{v}(n)$. It does, however, require that the basis functions are known or that good approximations can be obtained using prior knowledge of the temporal and spatial span of the sources. For example, if it is known that the energy of the sources is focused in a specific set of frequencies then the temporal basis vectors can be chosen based on the maximum eigenvectors of the autocorrelation matrix of the sources represented in the frequency domain and integrated over the specified frequency range. This makes the inherent assumption that the autocorrelation matrix of the sources, which is unknown, is approximated well by bandpass white noise.

In this framework the goal is to approximate the sources in the space of the observations (as given by $\boldsymbol{AS}$, as opposed to estimating the sources in the space of the sources, which is given by $\boldsymbol{S}$) by,

$$\boldsymbol{AS} \cong \boldsymbol{H\Theta D}^{T} \qquad (6)$$

where $\boldsymbol{H}$ is a $(M_{\mathrm{x}} \times L_1)$ matrix whose $L_1$ columns consist of the spatial basis vectors, $\boldsymbol{D}$ is a $(N \times L_2)$ matrix whose $L_2$ rows consist of the temporal

basis vectors, and $\boldsymbol{\Theta}$ is the ($L_1$ x $L_2$) reduced-rank approximation of $\boldsymbol{AS}$ that is found using the maximum likelihood technique. Once these three matrices are obtained the estimate of the cleaned observations is given by,

$$\bar{\boldsymbol{x}}(n) = \boldsymbol{H\Theta}\,\boldsymbol{d}(n) \qquad (7)$$

where $\boldsymbol{d}(n)$ is the column of $\boldsymbol{D}^{\mathrm{T}}$ corresponding to time $n$.

For BSP the signals of interest are described by the portion of the observations that lie in the space spanned by the columns of $\boldsymbol{H}$ *and* the columns of $\boldsymbol{D}$. Likewise, the interference, by definition, is the portion of the observations that lie outside the space spanned by the columns of $\boldsymbol{H}$ *or* the columns of $\boldsymbol{D}$. Since the same data is used to estimate both the source and interference statistics, BSP is more robust to non-stationary data than the FA/ICA approach. Additionally, since BSP does not require baseline data it may require a fewer number of arrays for a given level of performance.

**V.** ICA can be applied to a given dataset in two fundamentally different ways. The model given by Equation (1) is appropriate when ICA is applied to the rows of $\boldsymbol{x}(n)$. This approach seeks to find the unknown sources, $\boldsymbol{s}(n)$, and the unknown mixing matrix, $\boldsymbol{A}$, given only the observations, $\boldsymbol{x}(n)$. In the second approach ICA is applied to the columns of the data matrix. In this case the model should be changed to,

$$\boldsymbol{x}^{T}(m) = \widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{s}}(m) + \widetilde{\boldsymbol{B}}\widetilde{\boldsymbol{u}}(m) + \boldsymbol{v}^{T}(m) \qquad (8)$$

where $\boldsymbol{x}^{\mathrm{T}}(m)$ and $\boldsymbol{v}^{\mathrm{T}}(m)$ are the $m^{\mathrm{th}}$ row of $\boldsymbol{X}$ and $\boldsymbol{V}$, respectively, and all other quantities are different from their counterpart in Equation (1) since they have different dimensionalities and represent different signals/systems.

There are several trade-offs between the row-wise and column-wise approaches. ICA performs better if the data length is longer than the dimensionality. The column-wise approach is therefore well suited for microarray data since the dimensionality in this case equals the number of arrays and the data length is given by the number of genes. This is, in fact, the reason given by Liebermeister for chosing the column-wise approach (6).

The downside of the column-wise approach is that that "sources" do not correspond to biological phenomena. In the row-wise case it may be surmised that each of the sources is related to a single response/function. If this is the case and if there is an unambiguous match between an externally controlled and/or measured condition and a source estimate then the associated column of $\boldsymbol{A}$ directly indicates which genes are used for that function/response. Whether or not there is a close match between the experimental conditions and the source estimates, a threshold can be applied to the columns of the $\boldsymbol{A}$ matrix to generate a set of gene clusters or to the rows of $\boldsymbol{A}$ to generate clusters of functions/responses. The difference from this approach and performing traditional clustering on the observation matrix is that the former can place a single gene into multiple clusters (this can also be accomplished with clustering using fuzzy memberships). Another difference is that, supposing the ICA step is successful, thresholding the rows/columns of $\boldsymbol{A}$ should be less sensitive to the mixing that occurs when genes are regulated by multiple processes. Row-wise ICA can be used in experiments where multiple conditions are varied simultaneously, which is somewhat reminiscent of the advantage cited for using the two-color fluorescent labeling system. Another nice feature of the row-wise ICA approach is that, if one is willing to assume that the source estimates from a given experiment represent true sources, these source estimates may be stored in a database and used in subsequent experiments. Having the source estimates *a priori* reduces the problem of finding $\boldsymbol{A}$ from an ICA problem to the much simpler multi-variate regression problem. In addition it allows researches to compare cell responses from different tissues or organisms with previously published work in a manner that is robust to the mixing that naturally occurs in microarray data. This is not possible for the column-wise ICA approach, which is valid only for the dataset on which it is originally applied.

Denoising using a standard ICA method involves three steps. First, the sources and the interferences must be separated from themselves and from each other. Second, the outputs of the ICA method must be categorized into those that are sources and those that are interference. Third, the interference components are removed and the

ICA outputs are inverse transformed back into the space of the observations. This is represented mathematically by,

$$\bar{\boldsymbol{x}}(n) = \boldsymbol{W}^{-1}\boldsymbol{s}'(\boldsymbol{n}) \qquad (9)$$

where $\boldsymbol{W}$ is the transformation that separates the observations and $\boldsymbol{s}'(n)$ is the source estimate vector with the elements that correspond to interfering sources replaced with a value of 0. Also, it is common to use PCA as a preprocessor to reduce the dimensionality prior to demixing. Notice that ICA can be used for denoising with or without FA preprocessing, although the reverse is not true.

ICA has a certain minimum set of requirements, none of which are addressed in either of the papers by Liebermeister (6) or Saidi *et al*. (7). For example, the number of significantly energetic sources, upper bounded by $M_s+M_u$, must be less than or equal to the number of observations, $M_x$. This requirement is reduced to $M_s \leq M_x$ for FA/ICA. The standard ICA approach also requires that the sources and interference (sources only for FA/ICA) are either not Gaussian-distributed or have unique temporal autocorrelation functions. More importantly for microarray data are the requirements for linearity and memory-less mixing. If a linear model provides a good approximation of the underlying biological processes then, to preserve the linearity, one must *not* use microarray data generated using a ratio of expression levels. While it is always possible to apply ICA to any data, including the case when ratios are used, the resulting basis functions lose all biological significance. It is also crucial to avoid saturation, which can occur both in the cell and during the data collection process.

Another concern is whether or not instantaneous (memory-less) mixtures are appropriate. This is implicitly assumed both in this paper and in the papers by Liebermeister and Saidi *et al*. Actual cell processes do not occur instantaneously and can only be approximated to be instantaneous if the sampling period is chosen sufficiently large (large relative to the rate at which the impulse response of the mixing decays to 0, which in turn depends on the depth of the associated cellular networks). However, the sample rate must also be at least twice the highest frequency of any component having non-negligible energy in order to prevent aliasing in accordance with the Nyquist criterion (13). It is possible that these conditions cannot be simultaneously met and is an item that deserves further research. The trade-off in sample rate selection can be avoided by using Convolutive ICA methods (14), but their level of performance is significantly lower than that of instantaneous ICA methods.

**Model Selection and Overfitting.** All of the denoising methods listed above share the problem of model order selection and the potential problem of over-fitting the data. Optimal model order selection is an ill-posed problem that is usually addressed by human expert intervention. The potential for overfitting depends on the number of adaptable parameters relative to the data length. A general rule of thumb is that there should be at least 5-10 times as much data as adjustable parameters.

The amount of data in all cases considered here is $M_xN$. The number of adaptable parameters is $M_x^2$ for PCA (since it requires estimation of the autocorrelation matrix), ??? for SVD (using Alter's approach), $M_x(M_s+M_u)+1$ for FA/ICA, $L_1L_2$ for BSP (assuming $\boldsymbol{H}$ and $\boldsymbol{D}$ are known), $M_x(M_x-1)$ for row-wise ICA (without PCA preprocessing), and $N(N-1)$ for column-wise ICA (without PCA preprocessing). As long as $M_x >> N$, the most appropriate methods above with regards to overfitting are column-wise ICA, FA/ICA, and possibly BSP.

## CLUSTERING

The rows of the observation matrix, $\boldsymbol{x}(n)$, can be clustered with or without the aid of any of the denoising methods listed above. Besides denoising, another option to improve the accuracy of gene identification is to improve the clustering method. Simple hierarchical clustering methods are commonly used, although there is no lack of more sophisticated methods. One such method, developed by Jenssen *et al.* (3), is considered

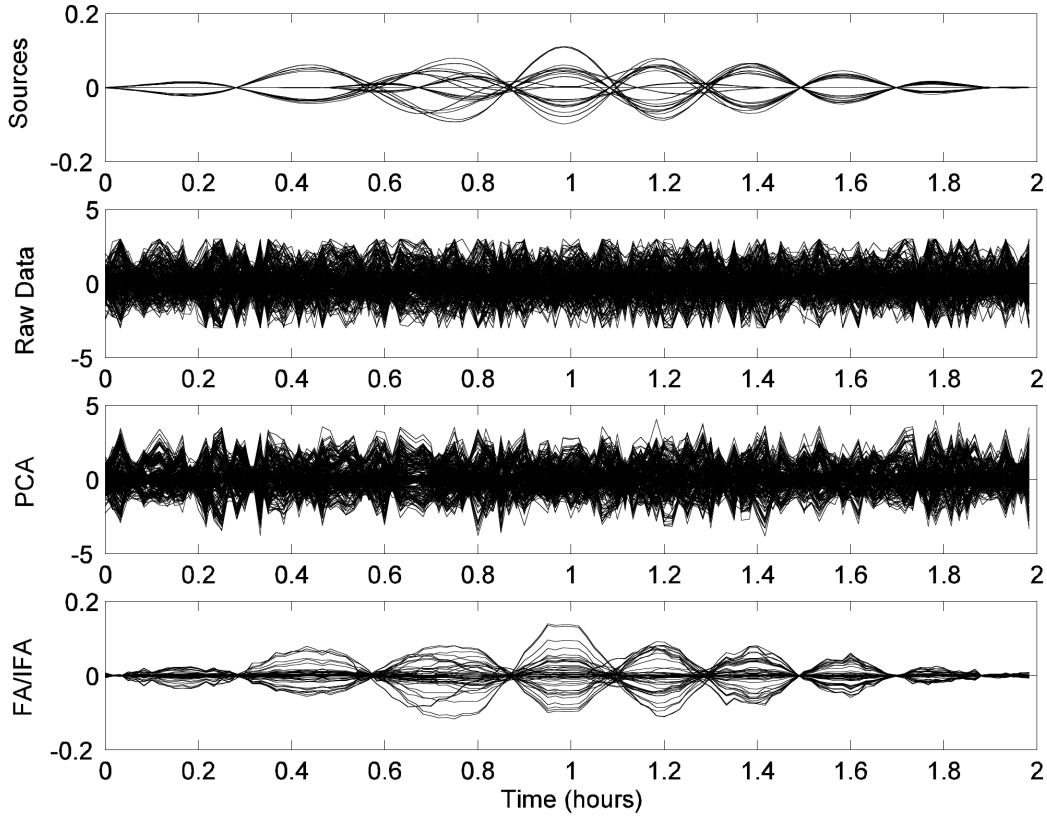


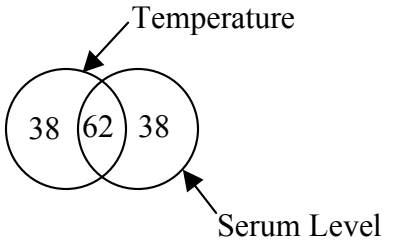

Figure 1. Top: Time courses of the unobserved sources (in sensor space); Second from top: observed time courses (one for each gene); Third from top: cleaned signal using PCA ($L$=10); Bottom: cleaned signals using FA/ICA.

here. This kernel-method approach maximizes the Cauchy-Schwarz divergence (15) between clusters (the divergence is an information-theoretic quantity that is analogous to a distance measure between probability distributions). The first step of this method involves mapping the data to a high-dimensional kernel feature space. In this space the cosine of the angle between the resulting cluster vector means is minimized, which is equivalent to maximizing the Cauchy-Schwarz divergence when the probability density functions are estimated using Parzen Window density estimation. This method has been shown to perform very well when highly non-linear decision boundaries are required. However the performance is fairly sensitive to the user-defined variable referred to as the kernel size.

## MATERIALS AND METHODS

The model given in Equation (1) is used to create $N$=200 arrays/time points of data, the first 80 of which precede the two externally controlled/ monitored conditions, *i.e.*, temperature and serum level. The time courses of these $M_s$=2 sources, $\boldsymbol{s}(n)$, are damped sinusoids, which are shown in the top of Figure 1. Each observation vector consists of the instantaneous gene expression levels of $M_x$=250 genes. As shown in Figure 2 each source regulates a total of 100 different genes, 62 of which are shared. The topmost subplot in Figure 3 shows the (unobserved) time courses of the gene expression levels due only to the sources, $\boldsymbol{As}(n)$, and the subplot directly inferior shows the observation time courses, $\boldsymbol{x}(n)$, which include the $M_s$=2 sources, the $M_u$=5 interfering sources, and the additive noise. The source distributions, owing to the damping, are super-Gaussian and the interfering sources and the noise both have Gaussian distributions.

Clustering is performed on the PCA and FA/ICA denoised data as well as on the original observations. The results for the latter are denoted




F
igure 2. Number of genes regulated by each of the two sources both individually and collectively.

as the Raw Data results. Two results for PCA are included. The first uses $L$=2 maximum eigenvectors and the second uses $L$=10. Likewise, FA/ICA assumes there are $M_s$=2 sources. Clustering is performed using both a hierarchical clustering method known as Unweighted Pair-Group Method Using Arithmetic Averages (UPGMA) and the kernel-based information-theoretic clustering method briefly described in the previous section, which is referred to as CS Divergence. The CS Divergence clustering method uses the affinity matrix (as opposed to the Laplacian matrix) and the UPGMA method uses the linear correlation coefficient for the distance metric. The results for UPGMA are obtained from the on-line Gene Expression Analysis Suite V1.1 (GEPAS) located at http://gepas.bioinfo.cnio.es/cgi-bin/cluster.

Gene identification accuracy is measured by assigning all the elements of each generated cluster to one of three classes, *i.e.*, Temperature, Serum, or Other. Since both clustering methods generate non-overlapping clusters, the 62 shared genes are assigned to a single source by associating each with the source that has the larger magnitude regulation factor. Using this approach yields 66 and 72 genes associated with the Temperature and Serum Level experimental conditions, respectively. The classification of each cluster is determined as the class that has the largest weighted cardinality, where the weighting equals the inverse of the probability of a gene belonging to the class in question. For example, if a particular cluster contains three genes, one gene from each class, then the cluster is assigned to the class that has the fewest exemplars. This selection essentially chooses the class that maximizes the *a posteriori* probability given the usual simplifying assumptions that each gene is drawn independently from a uniform distribution and the prior probability equals the number of exemplars divided by $M_x$. Since knowledge of the true class of each gene is required, this approach is not feasible to use in practice and it represents an upper bound to performance given the specific cluster labels.

The blind separation approach is also used for gene identification. The columns of the estimated $\boldsymbol{A}$ matrix generated by FA/ICA are used without a clustering method to directly form two overlapping clusters (this could also be performed on the rows of $\boldsymbol{A}$ to group genes according to gene expression modes). Since overlapping clusters are allowed with this linear model approach, the performance metric may be based on two 2-class assignments. In the first assignment all the genes are classified as either regulated by Temperature or not regulated by Temperature. The second assignment is identical to the first except it concerns the Serum Level experimental condition. Hence there is no restriction that prevents clusters associated with Temperature from overlapping those associated with Serum Level. Using this information two ROC curves are generated.

## RESULTS

Figure 4 shows the original set of gene expression data and Figures 8, 9, and 10 (in the Appendix) show the cleaned expression data that result from the PCA ($L$=2), PCA ($L$=10), and FA/ICA methods, respectively. The hierarchical UPGMA clustering results are shown in each figure to the left of the associated gene expression data. The time courses of the cleaned gene expression data (source estimates in the space of the sensors) are shown in Figure 3 and Figure 1 shows the source estimates (in source space) for FA/ICA. This latter plot does not apply for PCA or for the Raw Data. In this example FA/ICA is able to recover blindly good approximations of the true sources even though they are not visible in the original data.

Figure 5 is a plot of the accuracy of the clustering results as a function of the number of clusters when UPGMA is used. Using this clustering algorithm PCA ($L$=10) performed slightly better than the other three. The tendency for all methods is for the accuracy to improve as the number of clusters increases. This is a consequence of the fact that the class of each

cluster is optimally chosen and, as the number of clusters increases, the number of genes in each cluster approaches one. It is trivial to show that the accuracy, under these conditions, approaches 100%. Figure 6 is identical to Figure 5 except that it pertains to the CS Divergence clustering method. In this case FA/ICA performs best. The results for this clustering algorithm are 15-20% better than that produced by UPGMA.

Figure 6 shows the two ROC curves for FA/ICA that result from applying a threshold to each column of the estimate of the $\boldsymbol{A}$ matrix. Recall that each column corresponds to one of the experimental conditions and the $M_x$ elements of each column represent the degree to which the cell response to a given experimental condition regulates each gene. The resulting ROC curve for the Temperature response cannot be seen since the true positive rate equals 100% before the first false positive is found, which represents the best performance achievable. Likewise, the ROC curve for the Serum Level is nearly perfect.

## DISCUSSION

A simple generative model is introduced in order to compare denoising methods, clustering algorithms, and column thresholding for the purpose of gene identification. For the simpler clustering algorithm, PCA ($L$=10) performs well. Even though there are only two sources PCA ($L$=2) performs worse than PCA ($L$=10). This is not surprising since the signal-to-noise ratio is very low in this example, so that the maximum eigenvectors are not guaranteed to be associated with the sources of interest. It is also not surprising that the performance of the CS Divergence clustering algorithm is noticeably better than the simple UPGMA method. What may be surprising is how well the linear model approach to gene identification performs, although any excitement must be tempered by the fact that the FA/ICA model is identical to the generative model by which the data is produced. It remains to be seen whether, or how good, this generative model applies to real microarray data. Initial results were collected using $N$=1000 arrays (not shown). It was originally believed that the array size would need to be this large in order for these statistical signal processing techniques to show a noticeable improvement over existing cluster-only methods. The results of using $N$=1000 were exemplary so the value of $N$ was reduced to 200 for all results shown here. However, this value is still prohibitively large by today's standards.

1. M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proc. Natl. Acad. Sci.*, Vol. 95, pp. 14863-14868, Dec. 1998.
2. B.V. Baryshnikov, B.D. Van Veen, and R.T. Wakai, "Maximum likelihood estimation of low rank signals for multiepoch MEG/EEG analysis," *IEEE. Trans. Biomed. Engr.*, Vol. 51, No. 11, pp. 1981-1993, Nov. 2004.
3. R. Jenssen, D. Erdogmus, J.C. Principe, and T. Eltoft, "Spectral clustering based on information theory and Parzen windowing," preprint.
4. O. Alter, P.O. Brown, and D. Botstein, "Singular value decomposition for genome-wide expression data processing and modeling," *Proc. Natl. Acad. Sci.*, Vol. 97, No. 18, pp. 10101-10106, Aug. 2000.
5. N.S. Holter, A. Maritan, M. Cieplak, N.V. Federoff, and J.R. Banavar, "Dynamic modeling of gene expression data," *Proc. Natl. Acad. Sci.*, Vol. 98, No. 4, pp. 1693-1698, Feb. 2001.
6. W. Liebermeister, "Linear modes of gene expression determined by independent component analysis," *Bioinfomatics*, Vol. 18, No. 1, pp. 51-60, 2002.
7. S.A. Saidi, C.M. Holland, D.P. Kreil, D.JC MacKay, D.S. Charnock-Jones, C.G. Print, and S.K. Smith, "Independent component analysis of microarray data in the study of endometrial cancer," *Oncogene*, Vol. 23, pp. 6677-6683, 2004.
8. A.C. Pease, D. Solas, E.J. Sullivan, M.T. Cronin, C.P. Holmes, and S.P.A. Fodor, "Light-generated oligonucleotide arrays for rapid DNA sequence analysis," *Proc. Natl. Acad. Sci.*, Vol. 91, pp. 5022-5026, May 1994.
9. J.F. Cardoso, "Blind signal separation: Statistical principles," *Proc. IEEE*, Vol. 86, No. 10, pp. 2009-2025, Oct. 1998.
10. H. Attias, "Independent factor analysis," *Neural Computation*, Vol. 11, pp. 803-851, 1999.
11. S. Haykin, *Adaptive Filter Theory*, 4th ed., Prentice-Hall, Englewood Cliffs, NJ, 2001.
12. K. Hild, H. Attius, K. Sekihara, and S.S. Nagarajin, "A Graphical Model for Estimating Stimulus-Evoked Brain Responses in Noisy MEG data with Large Background Brain Activity," Submitted to *Intl. Conf. on Bioelectromagnetism* (BEM&NFSI '05), May 2005.
13. J.G. Proakis, *Digital Communications*, 3rd ed., McGraw-Hill, Inc., Boston, MA, 1995.
14. K.E. Hild II, D. Pinto, D. Erdogmus, and J.C. Principe, "Convolutive blind source separation by minimizing mutual information," Submitted to *IEEE Trans. Circuits and Systems I*, June 2004.
15. T.M. Cover and J.A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc., New York, NY, 1991.
16. S. Makeig, T. Ping, A. Bell, D. Ghahremani, and T.J. Sejnowski, "Blind separation of auditory event-related brain responses into independent components," *Proc. Natl. Acad. Sci*., Vol. 94, pp. 10979-10984, Sept. 1997.

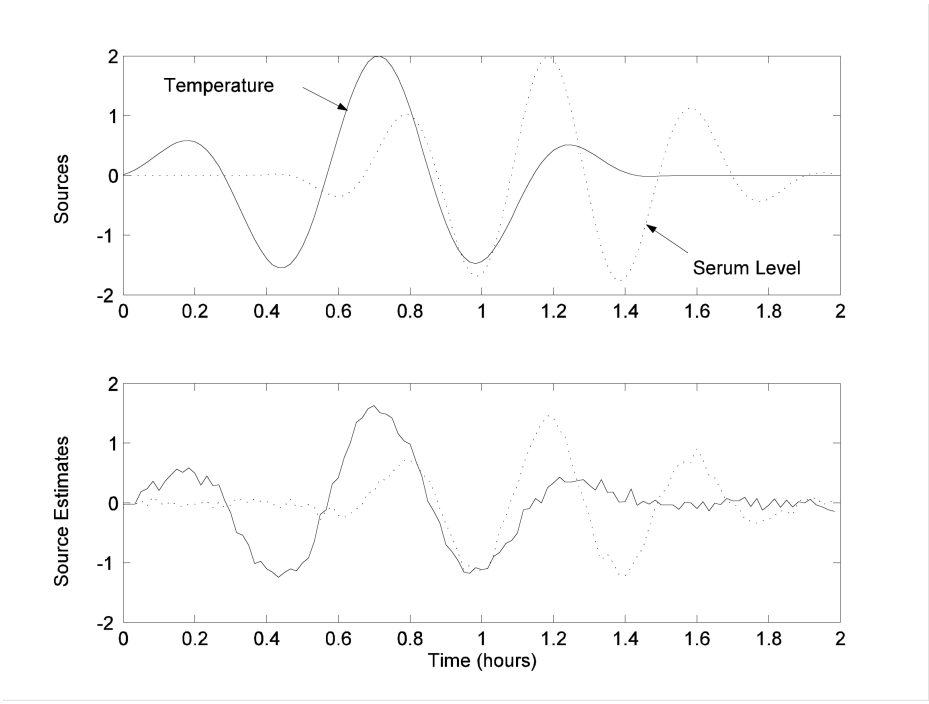



Figure 3. Top: Unobserved sources (in source space); Bottom: source estimates using FA/ICA.

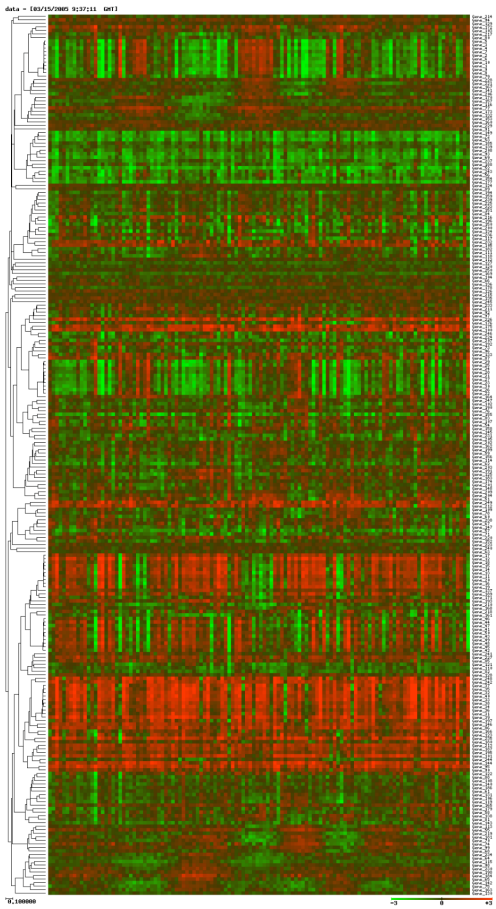


Figure 4. Gene expression matrix of the unprocessed data. The UPGMA clustering is shown on the left.

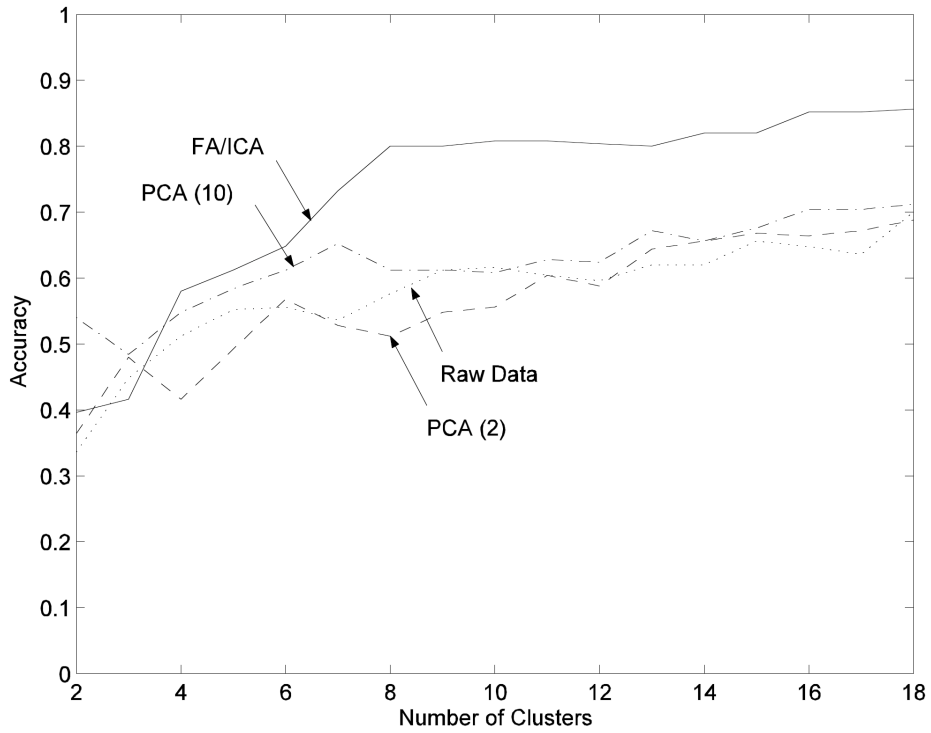



Figure 5. Accuracy as a function of the number of clusters for the UPGMA clustering method.

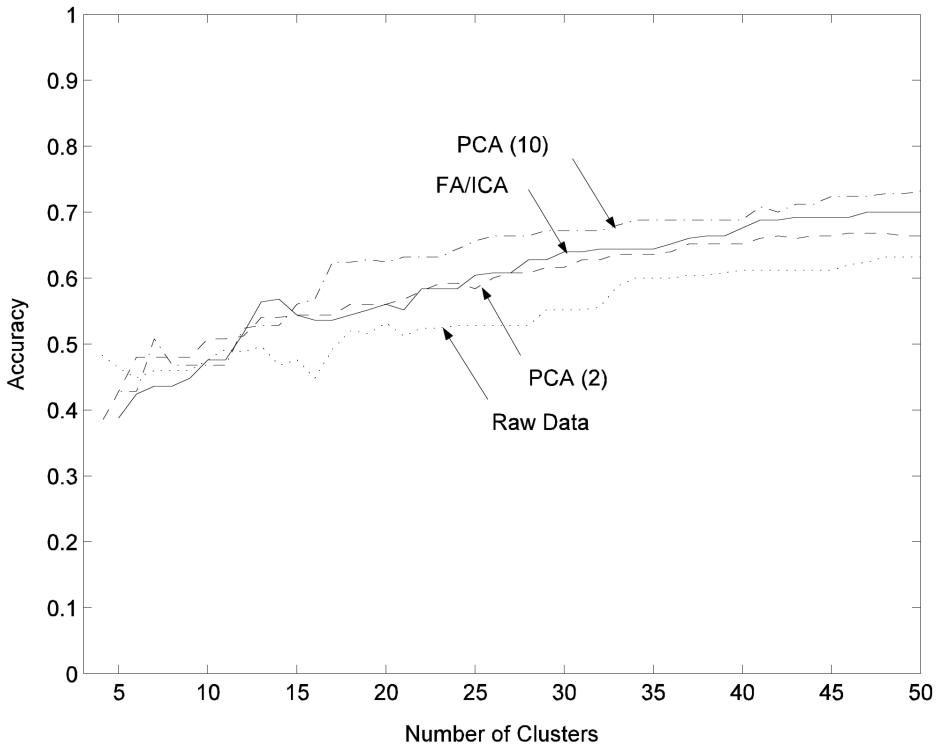



Figure 6. Accuracy as a function of the number of clusters for the CS Divergence clustering method.

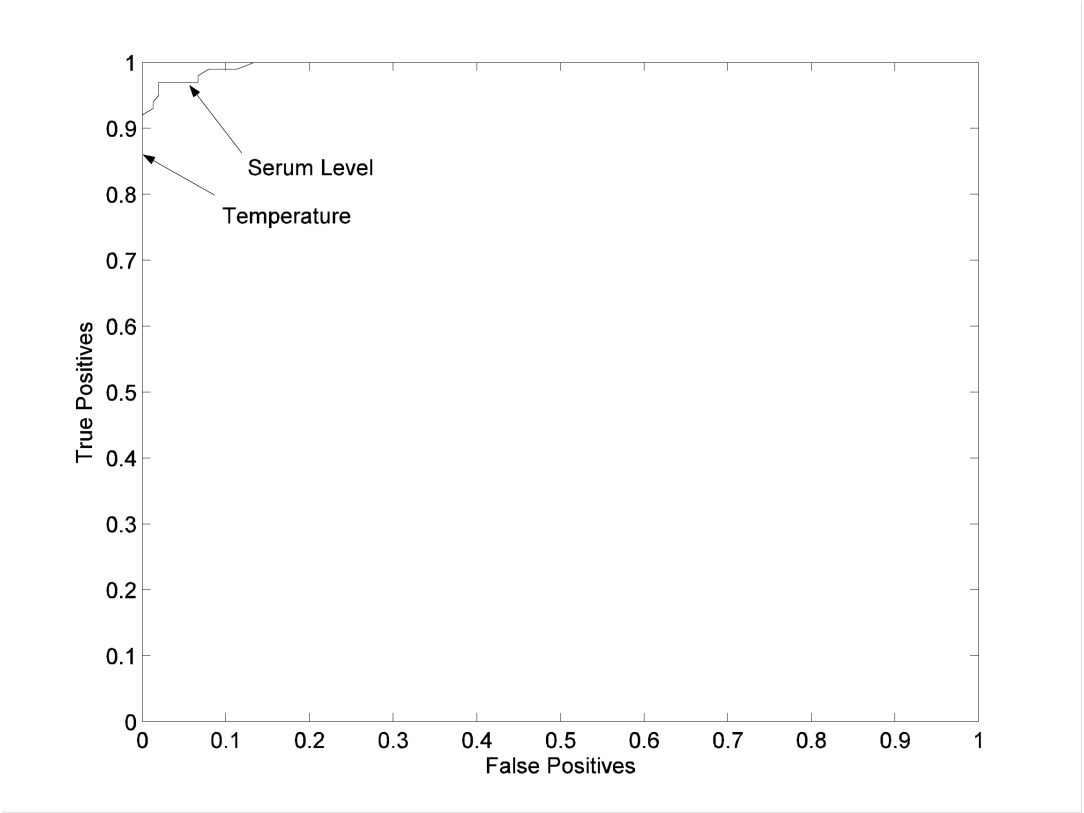



Figure 7. ROC curves for the Temperature and Serum Level experimental conditions using the blind separation method of thresholding the columns of the estimate of $\boldsymbol{A}$.

## APPENDIX

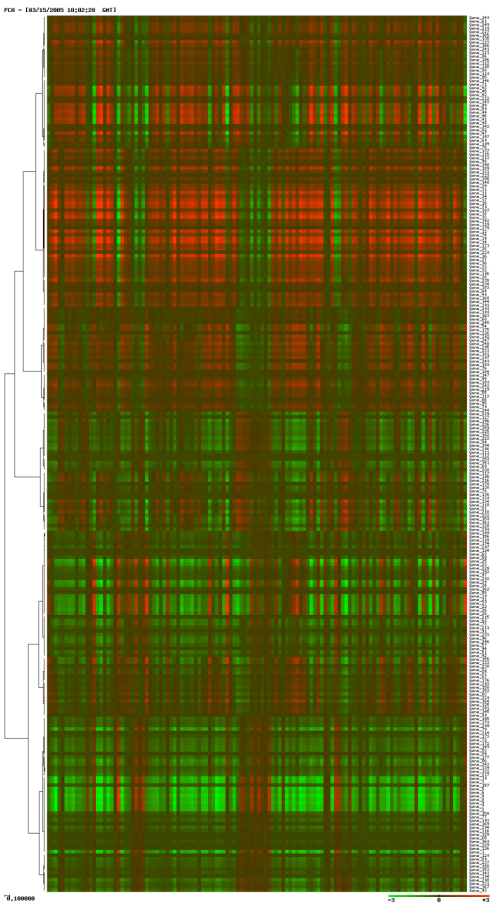


Figure 8. Gene expression data of cleaned data using PCA ($L$=2).

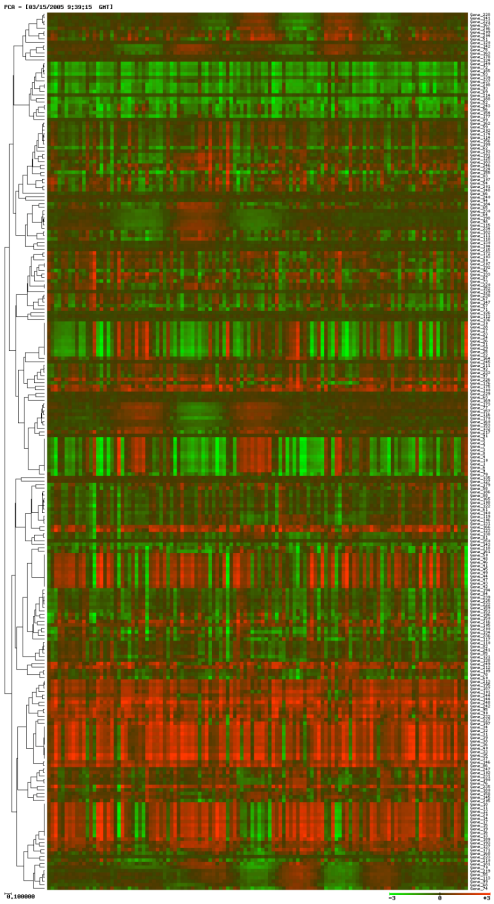


Figure 9. Gene expression data of cleaned data using PCA ($L$=10).

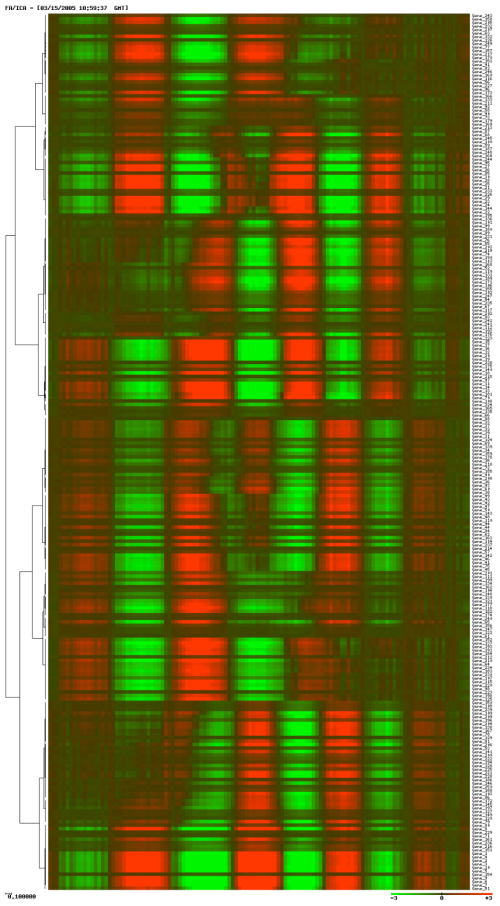


Figure 10. Gene expression data of cleaned data using FA/ICA.

**Matlab code used to generate and analyze the gene expression data:**

```
clear all
N = 250; % total genes
L = 200;

randn('state',5);
rand('state',5);

% generate s, sc
[jnk,jnk0,jnk1,jnk2,jnk3,jnk4,jnk5,jnk6,jnk7,jnk8,jnk9,jnk10,jnk11,s] =
create_lead_field(275,2,0,0,10,[],'x=0',L,9,7);
s(3:7,1:L)=randn(5,L);
s(1:2,:)=s(1:2,:)*2;

% t, pre, post
t=(-80:119)/60; % time in hours
pre=find(t<0);post=find(t>=0);

% geneset
K = 50; % multiple of 10
for a=1:2,r=randperm(N-50);geneset{a}=[1:K sort(r(1:50))+K];end % temp_tot = geneset{1}, serum_tot
= genest{2}
for a=3:7,r=randperm(N-50);geneset{a}=[1:K sort(r(1:50))+K];end

% A, Ac
A=zeros(N,7); for a=1:10:K, tmp=sign(randn(1,7)).*(1+rand(1,7)); A(a:a+9,:)=repmat(tmp,10,1); end
for a=1:7, lng=length(geneset{a})-K; tmp=sign(randn(lng,1)).*(1+rand(lng,1));
A(geneset{a}(K+1:end),a)=tmp; end
A=A/50;

% temp/serum genes
f=zeros(1,N);f(geneset{1})=1; f2=zeros(1,N);f2(geneset{2})=1; f3=find(f.*f2==1);
f(f3)=0; temp_uni=sort([find(f==1) find(abs(A(f3,1))>=abs(A(f3,2)))']);
f2(f3)=0; serum_uni=sort([find(f2==1) find(abs(A(f3,1))<abs(A(f3,2)))']);
temp_ndx=zeros(1,N);temp_ndx(temp_uni)=1;
serum_ndx=zeros(1,N);serum_ndx(serum_uni)=1;

% mux, muxc, noise, noisec
mux=0.2*rand(1,N)+0.4;mux=repmat(mux,L,1)';
noise=randn(N,L)*0.02;

% s, x
x=A*s+noise;
xs=x+mux; f=find(xs<0);xs(f)=1e-3;f=find(xs>1);xs(f)=1;

% yproj, yprojc
[yproj,sbar,w,jnk,jnk1,jnk2,jnk3,b0,lam0,alp0,g0]=sefaica(x(:,pre),x(:,post),[],0,2,'peaky');

% clusters
z2=A(:,1:2)*s(1:2,post);f=find(sum(abs(z2'))==0);z2(f,1)=1e-3;
```

```
z3=w(:,1:2)*sbar;
h=zeros(1,250);for
a=1:N,h(a)=x(1,post)*x(a,post)'/600/sqrt(cov(x(1,post))*cov(x(a,post)));end,figure(7),clf,plot(h),hold
on,axis([0 100 -0.2 0.2])
h2=zeros(1,250);for a=1:N,h2(a)=z2(1,:)*z2(a,:)'/600/sqrt(cov(z2(1,:))*cov(z2(a,:)));end,plot(h2,'k')
h3=zeros(1,250);for a=1:N,h3(a)=z3(1,:)*z3(a,:)'/600/sqrt(cov(z3(1,:))*cov(z3(a,:)));end,plot(h3,'r')

% saturation image
q=5*log2(abs((xs(:,post))./repmat(xs(:,pre(end)),1,length(post))));f=find(q>3);q(f)=3;f=find(q<-3);q(f)=-
3;
figure(1), clf,colormap gray,imagesc(q)
dim=2; [z,w,w2]=white(q,dim); q=w2*w'*q; % PCA
q=1*squeeze(yproj(:,:,1))+1*squeeze(yproj(:,:,2));q=q*8/max(max(abs(q))); % FA/ICA

% create data.txt file
y=double(['#NAMES']);
for a=post, y=[y 9 double(num2str(a))]; end, y=[y 10];
tmp=zeros(1,10000);
kk=[];for a=1:size(q,1), y=[y double('Gene_') double(num2str(a))];
  aaa=1; for aa=1:size(q,2), tmp2=[9 double(sprintf('%10.20f',q(a,aa)))]; tmp(aaa:aaa+length(tmp2)-
1)=tmp2; aaa=aaa+length(tmp2); end, y=[y tmp(1:aaa-1) 10]; end
fid=fopen('/home/hild/data.txt','w');
fwrite(fid,y);fclose(fid);

% read data.nw
fid=fopen('/home/hild/data.faica.nw');
y=fread(fid);fclose(fid); y=y'; y=[y 32*ones(1,10)];
mx_val=0; tmp=0; for a=1:length(y), if y(a)==40, tmp=tmp+1; if tmp>mx_val, mx_val=tmp; end, elseif
y(a)==41, tmp=tmp-1; end, end
cor_mat=NaN*ones(N,mx_val);
for a = 1:N
  f = findstr(char(y),['Gene_' int2str(a) ':']);
  f2 = find(y(f+5:end)==58) + f+5-1;
  f3 = find(y(f+5:end)==44 | y(f+5:end)==41) + f+5-1;
  f4 = length(find(y(f3(1)-1:end)==41)) - length(find(y(f3(1)-1:end)==40));
  cor_mat(a,mx_val+1-f4) = str2num(char(y(f2(1)+1:f3(1)-1)));
  if y(f3(1))==41, f3(1)=f3(1)-1; end % offset when f3(1) == ")"

  done = 0;
  pcnt = 0;
  while ~done
    f3 = find(y(f3(1)+1:end)==41 | y(f3(1)+1:end)==40) + f3(1)+1-1;
    if isempty(f3), done=1;
    elseif y(f3(1))==40, pcnt=pcnt+1;
    elseif y(f3(1))==41 & pcnt>0, pcnt=pcnt-1;
    elseif y(f3(1))==41 & pcnt<=0
      f2 = find(y(f3(1)+1:end)==44 | y(f3(1)+1:end)==41) + f3(1)+1-1;
      if isempty(f2), done=1;
      else f3 = find(y(f3(1)+1:end)==58) + f3(1)+1-1;
        f4 = length(find(y(f2(1)-1:end)==41)) - length(find(y(f2(1)-1:end)==40));
        cor_mat(a,mx_val+1-f4)=str2num(char(y(f3(1)+1:f2(1)-1))); f3=f2(1);
```

```
        if y(f3(1))==41, f3(1)=f3(1)-1; end % offset when f3(1) == ")"
      end
    end
  end
end
f=find(isnan(cor_mat)==1); cor_mat(f)=0;
cor_mat=cumsum(cor_mat')'; cor_mat(f)=NaN;

% create clustermap
corr_ndx=sort(DEL(cor_mat(:)))'; f=find(isnan(corr_ndx)==1); corr_ndx(f)=[];
cluster=cell(length(corr_ndx)-1,N);
mx_ndx=0;
for a=1:length(corr_ndx)-1
  ndx=1; gene_ndx=ones(1,N);
  f=find(abs(cor_mat-corr_ndx(a)) < 1e3*eps);
  if length(DEL(ceil(f/N))) > 1, disp(' '); disp('Duplicate values found in different columns'), disp(' '); end
  tmp = sort(DEL(rem(f-1,N)+1))';
  row=rem(f(1)-1,N)+1; col=ceil(f(1)/N);
  f2 = find(cor_mat(row,col+1:end) < corr_ndx(a)) + col+1-1;
  for aa=1:length(f2), tmp=DEL([tmp'; find(abs(cor_mat(:,f2(aa))-cor_mat(row,f2(aa))) < 1e3*eps)])';
end
  cluster{a,ndx}=tmp; ndx=ndx+1; gene_ndx(tmp)=0;

  done = 0;
  while ~done
    f=find(gene_ndx==1);
    if isempty(f), done=1;
    else f2=find(cor_mat(f(1),:) < corr_ndx(a));
      if isempty(f2), cluster{a,ndx}=f(1); ndx=ndx+1; gene_ndx(f(1))=0;
        if ndx > mx_ndx, mx_ndx=ndx; end
      else
        tmp=f(1);
        for aa=2:length(f)
          f3=find(abs(cor_mat(f(aa),:)-cor_mat(f(1),f2(end))) < 1e3*eps);
          if length(f3)>0, tmp=[tmp f(aa)]; end
        end
        cluster{a,ndx}=tmp; ndx=ndx+1; gene_ndx(tmp)=0;
        if ndx > mx_ndx, mx_ndx=ndx; end
      end
    end
  end
end
cluster(:,mx_ndx+1:end)=[];

s(1,post)=center(s(1,post));
s(2,post)=center(s(2,post));
Lpost=1/length(post);
thresh=0.5;

% determine percent correctly classified
zz=zeros(1,size(cluster,1));
```

```
for a=1:size(cluster,1), aa=1; while ~isempty(cluster{a,aa}), aa=aa+1; end, zz(a)=aa-1; end
av=1:size(cluster,1);
truepos=zeros(1,length(av)); k=1;
truepos2=zeros(1,length(av)); k=1;
zscore=1./[sum(serum_ndx) sum(temp_ndx) N-sum(serum_ndx)-sum(temp_ndx)];
for a=av
  aa=1;
  while ~isempty(cluster{a,aa})
    LL=length(cluster{a,aa});
    z=[sum(serum_ndx(cluster{a,aa})) sum(temp_ndx(cluster{a,aa}))];z=[z LL-
sum(z)];[val,pos]=max(z.*zscore);
    rho1=sum(abs(diag(center(q(cluster{a,aa},:))*repmat(s(1,post),LL,1)')))*Lpost;
    rho2=sum(abs(diag(center(q(cluster{a,aa},:))*repmat(s(2,post),LL,1)')))*Lpost;
    if max([rho1 rho2]) > thresh*LL
      if rho1>rho2, truepos2(k)=truepos2(k)+z(2);
      else truepos2(k)=truepos2(k)+z(1);
      end
    else truepos2(k)=truepos2(k)+z(3);
    end
    truepos(k)=truepos(k)+z(pos);
    aa=aa+1;
  end
  k=k+1;
end
truepos=truepos/N;
truepos2=truepos2/N;

f=find(zz<=3*15);plot(zz,truepos,zz(f(1)),truepos(f(1)),'*')
figure(1),plot(zz,truepos,'k'), hold on
round(max(truepos(f(1):end))*1000)/10

% information cut and Kmeans clusters
av2=2:19;
k=1;
truepos_ic=zeros(1,length(av2));
truepos2_ic=zeros(1,length(av2));
for a=av2
  [labels_ic]=jenssen_InformationCut(q','trad',0,-1,'ic',a,a,1);
  for aa=1:a
    f=find(labels_ic==aa); LL=length(f);
    z=[sum(serum_ndx(f)) sum(temp_ndx(f))];z=[z LL-sum(z)];[val,pos]=max(z.*zscore);
    rho1=sum(abs(diag(center(q(f,:))*repmat(s(1,post),LL,1)')))*Lpost;
    rho2=sum(abs(diag(center(q(f,:))*repmat(s(2,post),LL,1)')))*Lpost;
    if max([rho1 rho2]) > thresh*LL
      if rho1>rho2, truepos2_ic(k)=truepos2_ic(k)+z(2);
      else truepos2_ic(k)=truepos2_ic(k)+z(1);
      end
    else truepos2_ic(k)=truepos2_ic(k)+z(3);
    end
    truepos_ic(k)=truepos_ic(k)+z(pos);
  end
```

```
  k=k+1;
end
truepos_ic=truepos_ic/N;
truepos2_ic=truepos2_ic/N;

plot(av2,truepos_ic,'k')
[round(max(truepos_ic)*1000)/10 round(max(truepos_km)*1000)/10]

%%%%%%%%%%%%%%%%%%%%

tv = 0:0.01:1;
temp_tp = zeros(1,length(tv));
temp_fp = zeros(1,length(tv));
serum_tp = zeros(1,length(tv));
serum_fp = zeros(1,length(tv));

www=w*inv(g0);
if 1==1
  tmp=www(:,1);
  www(:,1)=www(:,2);
  www(:,2)=tmp;
end

k=1;
mx1 = max(www(:,1));
mx2 = max(www(:,2));
for thresh = tv
  f = find(abs(www(:,1)) > thresh*mx1);
  temp_tp(k) = length([geneset{1}';f]) - length(DEL([geneset{1}';f]));
  temp_fp(k) = (length(f) - temp_tp(k))/150;
  temp_tp(k) = temp_tp(k)*0.01;

  f = find(abs(www(:,2)) > thresh*mx2);
  serum_tp(k) = length([geneset{2}';f]) - length(DEL([geneset{2}';f]));
  serum_fp(k) = (length(f) - serum_tp(k))/150;
  serum_tp(k) = serum_tp(k)*0.01;

  k = k + 1;
end

figure(1),plot(temp_fp,temp_tp,'k',serum_fp,serum_tp,'k')
xlabel('False Positives','fontsize',12),ylabel('True Positives','fontsize',12)

%%%%%%%%%%%%%%%%%%%%

function [A,x] = fa(seednum,noise_pow)

% factor analysis
% y = Ax + v

L=5;
```

```
M=275; N=1000;
randn('seed',seednum);
Atrue=randn(M,L);
y=Atrue*randn(L,N) + randn(M,N)*sqrt(noise_pow);

% definitions
max_iter = 1000; [M,N] = size(y);

% initial conditions
[V,D,flg] = eigs(y*y'/N,L);
if flg ~= 0, disp(' '); error('Eigs did not converge'), end
A = V*D.^(0.5); Anew = A;
Lambda = 2*diag(1./diag(y*y'/N));
wmtrx = [A(:) zeros(M*L,max_iter+1)];
invpowAAtrue = 1/sum(sum((Atrue*Atrue').^2));
err = [sqrt(sum(sum((Atrue*Atrue'-A*A').^2))*invpowAAtrue) zeros(1,max_iter+1)];

done = 0;
iter = 0;
while ~done
  % E-step
  Gamma = A'*Lambda*A + eye(L);
  invGamma = inv(Gamma);
  x = invGamma*A'*Lambda*y;

  % Sufficient statistics
  Rxy = x*y';
  Rxx = x*x' + N*invGamma;
  Ryy = y*y';

  % M-step
  Anew = Rxy'*inv(Rxx);
  Lambda = N*diag(1./diag(Ryy-A*Rxy));

  powAAnew = sum(sum((Anew*Anew').^2));
  rel_dif = sqrt(sum(sum((Anew*Anew'-A*A').^2))/powAAnew);
  A = Anew;
  iter = iter + 1;
  wmtrx(:,iter+1) = A(:);
  err(iter+1) = sqrt(sum(sum((Atrue*Atrue'-A*A').^2))*invpowAAtrue);

  if rel_dif < 1e-3 | iter >= max_iter
    done = 1;
    if iter >= max_iter, disp(' '), disp('Did not finish converging'); end
  end
end
wmtrx(:,iter+2:end) = [];
err(:,iter+2:end) = [];
err(end)

figure,subplot(2,1,1),plot(wmtrx'),subplot(2,1,2),plot(err)return
```