NIC-TR-2004-019

NIC Signal Cleaning Framework User Manual

Purpose

This document will describe the NIC Signal Cleaning Framework (NICSCF), how to use and update it and how it is expected to fit into the larger NIC Brain Modeling Grid Framework.

Framework Structure

The framework implements a general signal process consisting of data reader, an optional preprocessing stage, a processing stage and an output stage see Figure 1. In the preprocessing stage, we have implemented a data whitening process, however, as the framework progresses we will add options such as data transformations, the identification and removal of outliers and data filtering. The processing stage is responsible for decomposing using factor analysis techniques (e.g., ICA and PCA), spectral techniques and other techniques as appropriate. The output stage presents the results and other data related to the decomposition as screen or file output. The signal cleaning process is controlled by a set of text-based forms, which the user edits for his or her specific needs.

Figure 1. Signal processing data flow.

Data Reading

The first step in the signal cleaning process is reading the data. We assume the data is stored in a binary file in which each column is a single observation and each row is the time series for a single channel. This format is called “channels x observations.” The user is not obligated to store his or her data in this format as the NICSCF provides data readers for “observations x channels” and he or she can store the data as a binary or ASCII file. The user should be aware that he NICSCF will store a copy of the data file in a binary, “channels x observation” format for future use by the user.

Preprocessing

Currently, NICSCF preprocessing procedures include data whitening only. Data whitening is a statistical procedure, which requires finding covariance matrix and channel average vector of the desired multi-channel data. The covariance matrix is decomposed into its eigenvalues and eigenvectors, which are used to generate a sphering matrix and the data are centered by subtracting the channel average vector from each column of the

data. After centering the source signals, the sphering matrix is multiplied by the centered signals to produce whitened data. The data-whitening flow is shown in figure 2. For details regarding the purpose and procedures of data whitening, please refer to NIC Technical Report NIC-TR-2004-001. Please note that sphering matrix is not unique. It depends on the decomposition algorithm so the sphering matrix produced by NICSCF may not be identical to the sphering matrix produced by another set of code.

Figure 2 shows the data flow of the data whitening.

Processing

Currently, the processing procedures include sequential and parallel versions of the FastICA algorithm and a sequential version of the Infomax algorithm. As we can convert the whitening process a PCA process, so we will incorporate a PCA implementation when time permits.

FastICA

The details of the FastICA algorithm are described in NIC-TR-2004-016 and the procedure is schematically outlined in Figure 3. After whitening the data, the FastICA algorithm attempts to find the rotation of multi-channel signal data such that rotated data is maximally non-gaussian. The rotation matrix, which we refer to as the weight matrix, is found by iteratively estimating each column of the weight matrix. The procedure makes an initial guess about the column of the weight matrix, which in orthogonalizes relative to all previously found weights. To refine the estimate of a column matrix, each observation is passed through a contrast function that maximizes the non-gaussianity of estimation and the new estimate is compared to the old estimate. If the two values are sufficiently close, the procedure terminates and the output is produced. If, after a specified number of iterations, the procedure does not produce an output, the process terminates with an error message.

The NICSCF allows the user to configure some aspects of the FastICA procedure. Currently, the user can specify one of three contrast functions: cubic, Gaussian or hyperbolic tangent (tanh). For an explanation of these choices please see [1]. The user can also specify the maximum number of iterations, the convergence tolerance and he or she can specify the method for making initial guess about the weight matrix. These choices for these methods are the identity matrix (i.e., the guess for the i^{th} estimate is the i^{th} column of the identity matrix) and random.

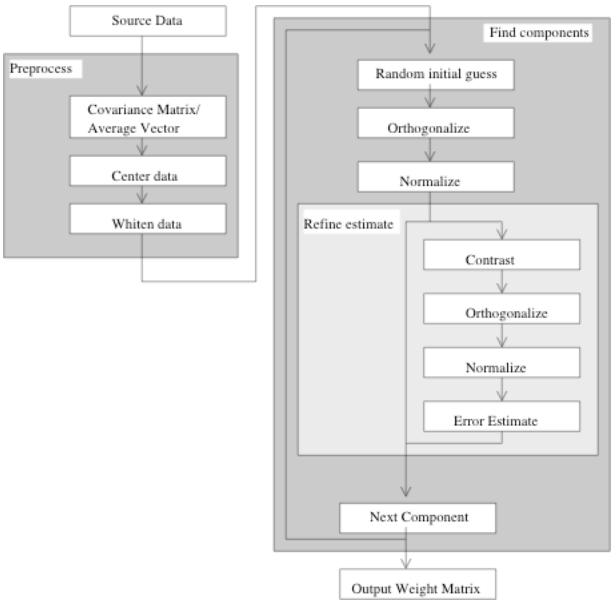

Figure 3 shows the FastICA data flow. The user can customize the process by selecting the contrast function, the number of iteration, the convergence tolerance and the method for making initial guesses.

Infomax

The details of the Infomax algorithm are described in NIC-TR-2004-017 and the procedure is schematically outlined in Figure 4. As with the FastICA algorithm, the Infomax algorithm whitens the data before it searches for the weight matrix. Unlike FastICA, Infomax uses a learning algorithm to refine its guesses. The learning algorithm randomizes the input data then partitions it into blocks of equal size. Each block is passed through a learning algorithm that estimates changes to the weight matrix based on a gradient search algorithm. After the entire data set is processed, the algorithm estimates the convergence of the weight matrix relative to the previous estimate. If the two matrices are sufficiently close, the algorithm terminates otherwise it adjusts the learning rate, which can alter the speed of convergence.

The NICSCF allows the user to configure some aspects of the Infomax procedure. Currently, the user can specify the maximum number of iterations, the convergence tolerance, the number of observations in a training block and the initial learning rate.

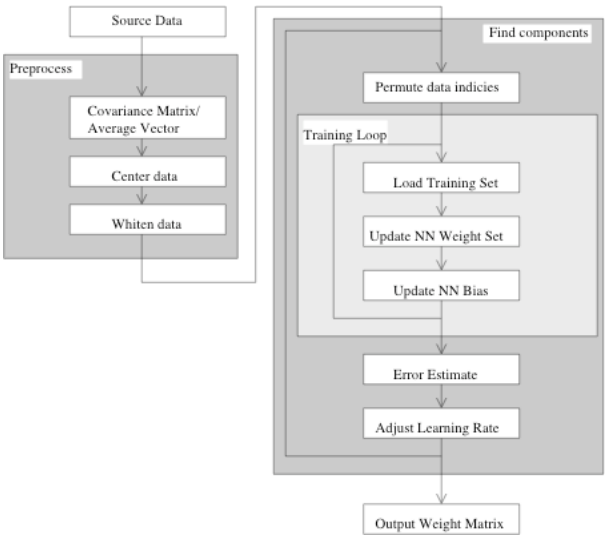

Figure 4 shows the Infomax process. The user can customize the process by selecting the the number of iteration, the convergence tolerance and the learning rate.

Output Results

The framework allows the user to output the primary results of the ICA process (i.e., the mixing and the unmixing matrices). These matrices can be printed as ASCII text or binary files and they can be printed to screen. In addition to the primary results, the user can select to print various intermediate results including the sphering matrix, the eigenvalue and eigenvector matrixes and the whitened data.

Using the Framework

The framework relies on a set of text-based forms that the user is required to fill out. All form files are held in a forms directory whose path, relative to the NICSCF home directory is *toolbox/models/signal_cleaner/forms*. Both ICA procedures use the cleaning form (cleaning.form) and the output form (output.form) and each procedure has its own configuration form (fastica.form and infomax.form).

An example of the cleaning form is shown in listing 1. This form will cause the NICSCF to use the FastICA algorithm on a data set called "b_mat1_6_4.ov.dat," which is located in directory *../data*. The data is in an "observations x channels" layout and the file is binary with 4 data channels and 7 observations. The user can change the processing procedure by replacing FastICA with Infomax (note that the names are case-sensitive). The data file can be changed by specifying the absolute data path to the desired data or using a relative path to the data (the relative path begins in *toolbox/bin*). The user must make certain that the data information is correct as the signal cleaner algorithm does minimal error checking.

Algorithm	**: FastICA**
Data Source File	**: "../data/b_mat1_7_4_ov.dat"**
Data Layout	**: observations x channels**
Data Type	**: binary**
Total Channels	**: 4**
Total Samples	**: 7**

Listing 1 is cleaning.form located in toolbox/models/signal_cleaning/forms/. The user can edit the file to setup data reading and to select the processing procedure.

The output form is shown in listing 2. The form allows the user to specify how the NICSCF will output data. For screen output, the user can type 'n' or 'y' to indicate whether a particular data object is to be displayed and he or she and type 'a,' 'bi' or 'bo' to store a data object in an ASCII or binary file or both. If no file storage is wanted, the user types 'n.' If the user wants a file output, he or she must specify a file name and the NICSCF will append an extension to the file name indicting the contents of the file.

Screen Output ----------		
Weight matrix screen	: y	
Mixing matrix screen	: n	
Unmixing matrix screen	: n	
Q matrix screen	: n	
Lambda ^(-1/2) matrix screen	: n	
Average vector screen	: n	
Covariance matrix screen	: n	
Sphering matrix screen	: n	
File Output----------		
File name	: output	
Sphering matrix file	: n	.sph
Weight matrix file	: a	.wgt
Unmixing matrix file	: bi	.umx
Average vector file	: bo	.avg
Covariance matrix file	: n	.cov

Listing 2 is the output form. This form will print the weight matrix to the screen, and print the weight matrix to an ASCII file called Aoutput.wgt, the unmixing matrix to a binary file called Boutput.umx and the average vector to an ASCII file called Aoutput.avg and a binary file called Boutput.avg.

The FastICA form is shown in listing 3. The form allows the user to select the error tolerance, the number of iterations, the contrast function and the method for initializing the weight guesses.

FastICA form

Error tolerance : 0.00001
Number of iterations : 1000

contrast function choices are:
cubic
gaussian
tanh

Contrast function : cubic

Initial W : identity(random, identity)

Listing 3 shows the FastICA configuration form. In this case, the user has selected an error tolerance of 0.00001, 1000 iterations, a cubic contrast function and the identity matrix as the initial guess for the weight matrix.

The Infomax form allows the user to select the error tolerance and the number of iterations for the convergence loop and it allows the user to select the size of a training block and the learing rate for the learning algorithm.

Infomax form

Error tolerance : 1.0e-6
Number of iterations : 1000

Learning algorithm configuration
Learning rate : 9.3775e-04
Block size : 34

Listing 4 shows the Infomax configuration form. In this case, the user as selected an error rate of 10-6, a maximum of 1000 iterations, a learning rate of 9.3775 x 10^{-4} and a block size of 34.

After configuring the file and ensuring the data file is in the location described in the form, the user can execute *signal_cleaner* from the *toolbox/bin/* directory.

Modifying the Framework

The framework is used to manage a data flow process for signal cleaning. The process, shown in Figure 1, is decomposed into four steps, each of which can be configured by the user via forms. The forms allow the user to modify the existing code, however, the point of the framework is to support the addition of other code. Developers can modify the data flow process by producing alternate procedures or modifying existing ones. To make changes to the framework, the developer must be familiar with structure of the framework.

Framework Design

To understand the modification procedures, it is necessary to first understand the design of the framework. The NICSCF consists of three main components: the model component, the form readers and mathematical/statistical tools. The model package includes specific statistical models (e.g., ICA and PCA) and data structures that contain model specific data such as the covariance and sphering matrices. The form reader package consists of a generic reader class and model specific subclasses. The reader classes act as intermediaries between the forms and the model classes. The mathematical/statistical package contains general computational classes and functions, e.g., eigenvector/eigenvalue decompositions and covariance matrix computation. The framework structure is shown in figure 5.

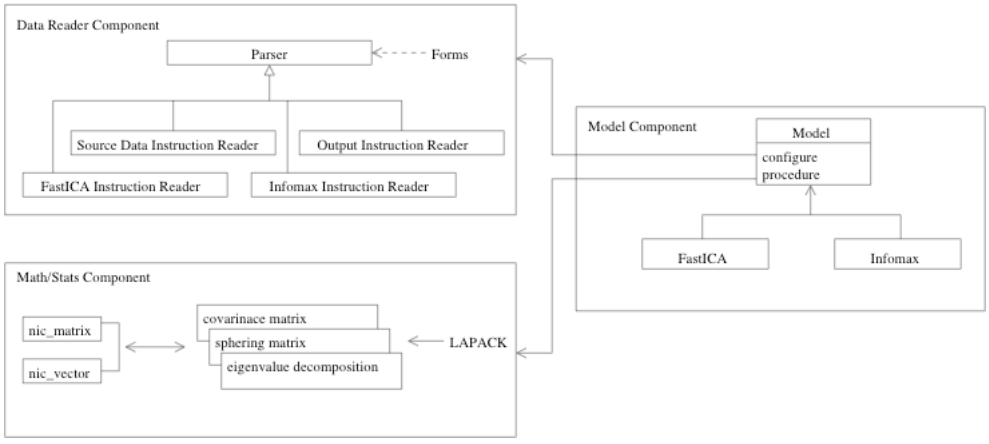

Figure 5 shows the structure of the NICSCF.

Model Component

THE REMAINDER OF THIS SECTION IS UNDER CONSTRUCTION