NIC-TR-2004-017

NIC Infomax Implementation

Purpose

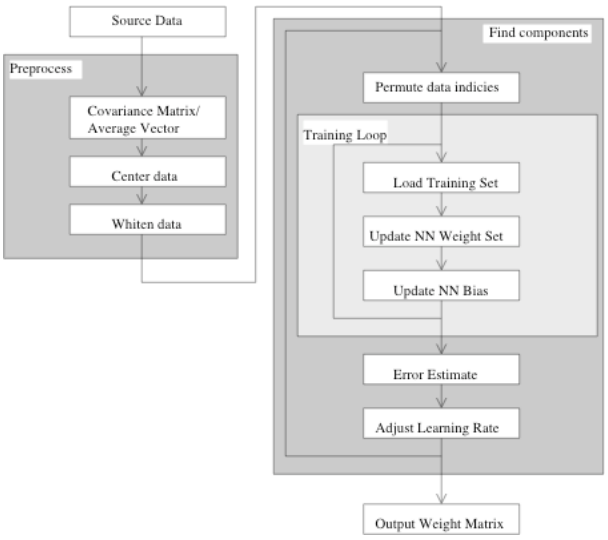
This document will describe the NIC Infomax implementation. The Infomax ICA algorithm was developed by Bell and Sejnowski [1] and implemented in the EEGLab package [2]. EEGLab is a set of Matlab procedures designed to analyze EEGs. As with the FastICA algorithm (see NIC-TR-2004-016), the EEGLab Infomax algorithm is not fast enough for the problems required by the NIC and it cannot handle the problem sets required by the NIC. To overcome these issues, we have re-implemented the algorithm using the C and C++ programming languages, the LAPACK linear algebra package and the OpenMP library. ~~{The results of our re-implementation, including numerical validation using the EEGLab version of the Infomax code and the performance of the parallel implementations are included in this document.}~~

For an explanation of the ICA process, see NIC-TR-2004-001.

The Infomax Algorithm

The Infomax algorithm attempts to find a set of independent components using a neural network-based, mutual information minimization algorithm [1]. There are variations of this algorithm [3], but we have chosen the default version supplied by the developers of EEGLab as our comparison standard. See Figure 1 for a flow representation of the algorithm.

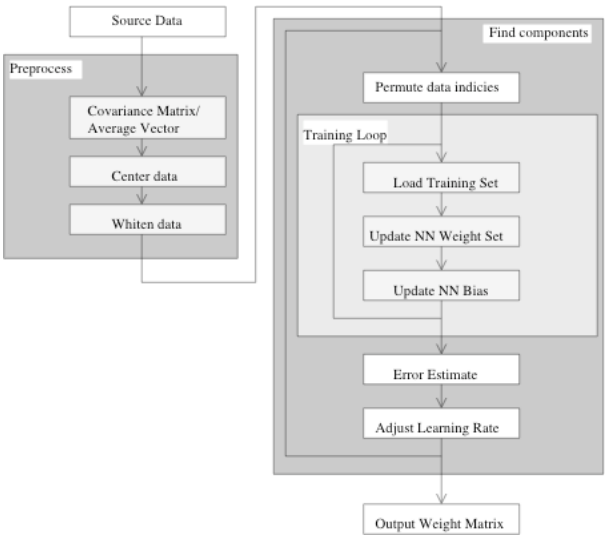
As with the FastICA algorithm (NIC-TR-2004-016), the Infomax algorithm whitens the source data before generating the weight matrix **W** associated with the independent components. The weight matrix is initialized to the identity matrix, which the algorithm iteratively refines using a learning algorithm. The learning algorithm random selects, without replacement, a subset of the whitened data **x** from which it computes an estimate of the unmixed signals $\mathbf{u} = \mathbf{W}\mathbf{x} + \mathbf{b}$, where **b** is a bias vector used by the neural network. The algorithm searches for **W** by estimating the likelihood that we can obtain the distribution of the observations from our estimate of **W**. From the likelihood function, we can estimate a ΔW that maximizes the likelihood that **W** produces a correct estimate. The formulation derived in [3] for this estimate is $\Delta\mathbf{W} = [\mathbf{I} - 2\tanh(\mathbf{u})\mathbf{u}^T]\mathbf{W}$. We update **W** from $\Delta\mathbf{W}$ and we update **b** using the current learning rate, then repeat the process using the next training set. After the last training set is processed, we estimate the difference between the current estimate of **W** and the previous estimate. If the difference exceeds a predefined tolerance the learning process starts again.



Parallelizing Infomax

The Infomax algorithm presents some difficulties for parallelization. Each iteration of the convergence loop depends on the current estimate of the weight matrix at the beginning of the loop. This value is not accurately known until the end of the previous iteration of the convergence loop, we cannot readily unroll the convergence loop. Each iteration of the training loop uses a subset of the source data, which, if the subsets are sufficiently large, may form the basis of a distributed computation. Unfortunately, the recommended size of the subsets is too small (e.g., EEGLab uses a default training set of 20 observations which is far too small).

Rather than pursuing a distributed computing implementation, we are pursuing a shared memory implementation with OpenMP. OpenMP provides a set of “pragmas” that support automatic loop unrolling and other shared-memory parallelization techniques. The loop unrolling “pragmas” were added to the NIC Infomax code as show in figure 2.



Validation of Infomax

At this point the NIC Infomax algorithm has been tested on a simple data set only. The results match those of the EEGLab Infomax implementation on the same data set, however, this is not a sufficient test.

Sequential Validation

~~{We compared the results NIC Infomax implementation to those of the EEGLAB Infomax implementation, which we assume to be correct. To ensure consistency between tests, we replaced the permutation of data indices with an ordered list of indices. The whitening algorithm used in the EEGLAB implementation differs from the algorithm used in the NIC implementation, so, for testing purposes, we skip the whitening step and use the whitened data produced by the EEGLAB version.~~

~~To validate the code, we used the data set from experiment #5, described in NIC TR-2004-002, to generate a weight matrix using both implementations.}~~

Parallel Validation

~~{ To validate the parallel versions of the Infomax algorithm, we followed the same testing procedures and validation criteria as in the sequential validation process. We ran the procedures for 1 – 32 processors on the NIC's Dell Cluster (*neuronic*) and on 1 – 8 processors on the p655 shared memory cluster. (A description of the clusters will be presented in a pending technical report). }~~

Performance of Parallel NIC Infomax

This is in progress as of the writing of this draft.

Bibliography

[1] Bell and Sejnowski

[2] Delorme, Makeig. EEGLAB: An Open Source Toolbox for Analysis of Single-Trial EEG Dynamics Including Independent Component Analysis. Journal of Neuroscience Methods