Hyper Parameter Optimization in near real time for Blind Source Separation

Vishodu Shaozae, Tarsh Sahu, Shubham Dodeja,

Karan Singh

Mentored By: Dr.Rishav Singh

Abstract

EGHR-β smoothly interpolates between performing PCA and ICA as parameter β that controls the weight of PCA varies. This algorithm can achieve dimensionality reduction and ICA simultaneously. The only issue being that the value of β has been calculated with Hit and trial approach. Our goal is to find an optimized value of the parameter and increase the processing speed of Blind Source Separation algorithm using parallel processing.

Introduction

A fixed-point BSS algorithm inherits the advantages of the well-known FastICA algorithm of ICA, which converges fast and does not need to choose any learning step sizes. Its higher separation accuracy is verified by numerical experiments. Meanwhile, we also give the consistency analysis and prove convergence properties of the algorithm, which has a (locally) consistent estimator and at least quadratic convergence. A biologically plausible independent component analysis (ICA) algorithm [4] called the error-gated Hebbian rule (EGHR). The EGHR modulates the magnitude of Hebbian plasticity by a global (error) factor. Hebbian Rule assumes that if two neighbor neurons be activated and deactivated at the same time, then the weight connecting these neurons should increase. For neurons operating in the opposite phase, the weight between them should decrease and if there is no signal correlation, the weight should remain unchanged[1][2]. Blind source separation (BSS) and related methods, e.g., independent component analysis (ICA), employ a wide class of unsupervised learning algorithms and have found important applications across several areas from engineering to neuroscience. [3]

Blind source separation (BSS) has been of great interest in many areas such as wireless communications and biomedical imaging. Since many of the existing BSS methods are based on a successive cancellation procedure for extracting all the unknown sources, they suffer not only from the error propagation accumulated at each stage but also from a long processing latency.

Efficient parallel and concurrent algorithms and implementation techniques are needed to meet the scalability and performance requirements entailed by scientific data analyses such as the optimization of the parameter β in EGHR-β (BSS) due to higher processing time involved in conventional ICA and PCA algorithms with the linear sequential processing.

Parallelism is the process of performing tasks concurrently, that is, more than one task per unit time[5]. A parallel computer is a computer that has the ability to exploit parallelism incorporated in its architecture. It meant the control of two or more operations which are executed virtually simultaneously, and each of which entails following a series of instructions.

Parallel processing in regards to BSS is, to be able to process the best practical value of β in the given parameter in order to achieve a better optimization for BSS.[6][7]

Related Work

For many years, a combination of principal component analysis (PCA) and independent component analysis (ICA) has been used as a blind source separation (BSS) technique to separate hidden sources of natural data. However, it is unclear why these linear methods work well because most real-world data involve nonlinear mixtures of sources. In the given paper [Isomura, Takuya, and Taro Toyoizumi. "On the achievability of blind source separation for high-dimensional nonlinear source mixtures." arXiv preprint arXiv:1808.00668 (2018).] it is showed that a cascade of PCA and ICA can solve this nonlinear BSS problem accurately as the variety of input signals increases. Specifically, in the paper there are two theorems that guarantee asymptotically zero-error BSS when sources are mixed by a feedforward network with two processing layers. The first theorem analytically quantifies the performance of an optimal linear encoder that reconstructs independent sources. Zero-error is asymptotically reached when the number of sources is large and the numbers of inputs and nonlinear bases are large relative to the number of sources. The next question involves finding an optimal linear encoder without observing the underlying sources. The second theorem guarantees that PCA can reliably extract all the subspace represented by the optimal linear encoder, so that a subsequent application of ICA can separate all sources. Thereby, for almost all nonlinear generative processes with sufficient variety, the cascade of PCA and ICA performs asymptotically zero-error BSS in an unsupervised manner.

Though the theorems provided in the given paper provides a really efficient accuracy the parameter β in EGHR-β (BSS) due is high due to processing time involved in conventional ICA and PCA algorithms with the linear sequential processing whereas in our model Parallel processing in regards to BSS is used which is able to process the best practical value of β in the given parameter in order to achieve a better optimization for BSS.

The adaptive rule, which constitutes an independence test using non-linear functions, is the main original point of this blind identification procedure. ICA vectors correspond to the axes of the data i.e. find the one in which each vector is an independent component, (independent of each other), whereas the PCA vectors try to find directions where variance is maximized i.e. data is the one that best explains the variability of the data, therefore the model should smoothly interpolate between performing PCA and ICA as parameter β that controls the weight of PCA varies. This algorithm can achieve dimensionality reduction and ICA simultaneously. Based on some biological observations, an adaptive algorithm is proposed to separate simultaneously all the unknown independent sources. [Jutten, Christian, and Jeanny Herault. "Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture."]

Adaptive Blind Signal and Image Processing delivers an unprecedented collection of useful techniques for adaptive blind signal/image separation, extraction, decomposition and filtering of multi-variable signals and data.[A. Cichocki and S. Amari. Adaptive Blind Signal and Image Processing. John Wiley & Sons Ltd, New York, 2003.] This volume unifies and extends the theories of adaptive blind signal and image processing and provides practical and efficient algorithms for blind source separation: Independent, Principal, Minor Component Analysis, and Multichannel Blind Deconvolution (MBD) and Equalization.

The approach enables us to recover the blurs and the original image from channels severely corrupted by noise. We observe that the exact knowledge of the blur size is not necessary and we prove that translation misregistration up to a certain extent can be automatically removed in the restoration process, whereas our model aims to find the best value of parameter β thus reducing the time involved in hit and trial and with the help of parallel processing the process of finding the parameter β is significantly reduced.

The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent. In the paper [Bugli, Céline, and Philippe Lambert. "Comparison between principal component analysis and independent component analysis in electroencephalograms modelling."] We shall see that ICA provides a more powerful data representation than PCA in the EEG analysis context. Hence creates a minus point in our research. EEG activity is present in a spontaneous way. It is affected by external stimuli (e.g. tone or light flash). The alteration of the ongoing EEG due to stimuli is named an event related potential (ERP). ERP can be visualised during a short period following the stimulation time, with a response pattern which is predictable under similar condition. These are some functions of EEG that gives the slighter advantage to this paper. The separation of the sources by ICA has great potential in applications such as the separation of sound signals (like voices mixed in simultaneous multiple records, for example), in telecommunication or in the treatment of medical signals. The lack of a priori knowledge about the mixture is compensated by a statistically strong but often physically plausible assumption of independence between source signals and of linearity. On the other hand, the direction of the axes in ICA are not only computed from the second order statistics like in PCA but also from higher orders statistics.

Other than this the main objective of our paper to reduce the time required for calculating the parameter in EGHR-β i.e. β through introduction of parallel processing in our algorithm. In [“A Review of parallel processing for statistical computation” by Adams, Niall M., et al.] we see that Parallel computers differ from conventional serial computers in that they can, in a variety of ways, perform more than one operation at a time. Parallel processing, the application of parallel computers, has been successfully utilized in many fields of science and technology. The purpose of this paper is to review efforts to use parallel processing for statistical computing. In the paper the author presents some technical background, followed by a review of the literature that relates parallel computing to statistics.

The review material demonstrates that parallel processing has been investigated for, and applied to, a wide range of statistical methods indicative of the common reason for its usage

i.e. processing speed. The use of parallel computing is also recognized in data analysis situations where the magnitude of computation makes interactive analysis impossible. As such, the optimization of the parameter β in EGHR-β (BSS) along with matrix computation are directed towards this approach while also acknowledging the unlikeliness of published parallel algorithms to be readily ported to systems with different architectures and thus, its lack of portability.

[S. Gill; Parallel Programming, The Computer Journal, Volume 1, Issue 1, 1 January 1958, Pages 2–10] The volume, variety, and velocity properties of big data and the valuable information it contains have motivated the investigation of many new parallel data processing systems in addition to the approaches using traditional database management systems (DBMSs). MapReduce pioneered this paradigm change and rapidly became the primary big data processing system for its simplicity, scalability, and fine-grain fault tolerance. And From a programming point of view, by minimizing the time lost in the operation of peripheral equipment, the adoption of time-sharing will encourage and enable more economical use to be made of a system in which the speeds of the auxiliary system vary greatly and are slow compared with the speed of the computer itself. This approach will enable the computer to make short calculations from time to time in order to apply short-term corrections, whilst intervening periods are utilized in long-term calculation which may also relate to the same job itself or may be a different one altogether.

Methodology

Principal Component Analysis

Principal Components Analysis (PCA) is a practical and standard statistical tool in modern data analysis that has found application in different areas such as face recognition, image compression and neuroscience. It has been called one of the most precious results from applied linear algebra. PCA is a straightforward, non-parametric method for extracting pertinent information from confusing data sets. Principle Component Analysis can be considered as a projection method which projects observations from a p-dimensional space with p variables to a k-dimensional space (where $k < p$) so as to conserve the maximum amount of information (information is measured here through the total variance of the scatter plots) from the initial dimensions. If the information associated with the first 2 or 3 axes represents a sufficient percentage of the total variability of the scatter plot, the observations will be able to be represented on a 2- 3- dimensional chart, thus making interpretation much easier.

We begin with an adjusted data matrix, X, which consists of n observations (rows) on p variables (columns). The adjustment is made by subtracting the variable's mean from each value. That is, the mean of each variable is subtracted from all of that variable's values. This adjustment is made since PCA deals with the covariances among the original variables, so the means are irrelevant.[13]

The matrix of scores will be referred to as the matrix Y. The basic equation of PCA is, in matrix notation, given by:

$$Y = W'X$$

where W is a matrix of coefficients that is determined by PCA. These equations are also written as:

$y_{ij} = w_{1i} x_{1j} + w_{2i} x_{2j} + \ldots\ldots\ldots + w_{pi} x_{pj}$

Principal Component Analysis Method [13]

Step 1: Get some data Let us consider a simple arbitrary three dimensional data set.

Step 2: Subtract the mean For PCA analysis, you have to subtract the mean from each of the data dimensions. The mean subtracted is the average across each dimension. So, all the x values have (the mean of the values x of all the data points) subtracted, and all the y values have subtracted from them and all the z values have subtracted from them. This produces a dataset whose mean is zero.

Step 3: Calculate the covariance matrix Since the data set is 3 three dimensional, so the covariance matrix will be 3×3.

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix Since the covariance matrix is square, we can calculate the eigenvectors and eigenvalues for this matrix. These are rather important, as they tell us useful information about our data.

Step 5: Choosing components and forming a feature vector. Here is where the notion of data compression and reduced dimensionality comes into it.

Step 6: Deriving the new data set. In order to find out new final data set, we have chosen the components (eigenvectors) that we wish to keep in our data and formed a feature vector, we simply take the transpose of the vector and multiply it on the left of the original data set, transposed.

Step 7: Getting the old data back This step is most important to recover the original data set. If we took all the eigenvectors in our transformation will we get exactly the original data back

Independent component analysis

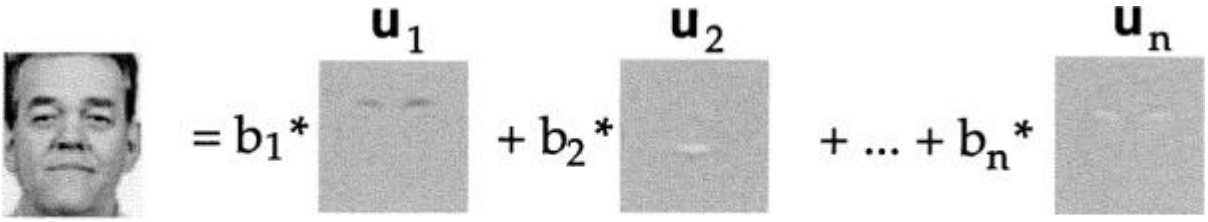
There are a number of algorithms for performing ICA[10]. We chose the infomax algorithm proposed by Bell and Sejnowsk[12]i ,which was derived from the principle of optimal information transfer in neurons with sigmoidal transfer functions.

The goal in this approach is to find a set of statistically independent basis images. We organize the data matrix X so that the images are in rows and the pixels are in columns. ICA finds a matrix W such that the rows of U = WX are as statistically independent as possible. The source images estimated by the rows of U are then used as basis images to represent faces. Face image representations consist of the coordinates of these images with respect to the image basis defined by the rows of U, as shown in Fig. 1. These coordinates are contained in the mixing matrix $A=\Delta W-1I10]$

Assume that we observe n linear mixtures x1,...,xn of n independent components

$x_j = a_{j1}s_1 + a_{j2}s_2 + . . . + a_{jn}s_n$, for all j

Fig 1: The independent basis image representation consisted of the coefficients, b, for the linear combination of independent basis images, u, that comprised each face image x.



$$\text{ICA representation} = (b_1, b_2, \ldots , b_n)$$

The number of ICs found by the ICA algorithm corresponds to the dimensionality of the input. In order to have control over the number of ICs extracted by the algorithm, instead of performing ICA on the nr original images, we performed ICA on a set of m linear combinations of those images, where m < nr.[10]

The ICA algorithm produced a matrix $W_I = WW_Z$ such that

$$W_I P_{Tm} P_{Tm} = U = W_{-1I} U.$$

Therefore

$$X\hat{}X\hat{} = R_m P_{Tm} = R_m W_{-1I} U.$$

where W_Z was the sphering matrix defined in . Hence, the rows of $R_m W_{-1I}$ contained the coefficients for the linear combination of statistically independent sources U that comprised $X\hat{}$, where $X\hat{}$ was a minimum squared error approximation of X, just as in PCA. The IC representation of the face images based on the set of m statistically independent feature images, U was, therefore, given by the rows of the matrix

$$B = R_m W_{-1I}.$$

A representation for test images was obtained by using the PC representation based on the training images to obtain $R_{test} = X_{test} P_m$, and then computing[10]

$$B_{test} = R_{test} W_{-1I}.$$

Parallel Processing

Our work is based on NVIDIA's Compute Unified Device Architecture CUDA, a hardware and software architecture for general purpose computing on graphics processing units (GPGPU). The modern GPU is a highly data-parallel processor, optimized to provide very high floating point arithmetic throughput for problems suitable to solve with a single program multiple data SPMDmodel. As all current CUDA-enabled devices are optimized for 32-bit floating point precision, this was the precision used in our code the new NVIDIA GTX-1060 architecture does, however, support double, 64-bit, floating point precision.[11]

EGHR- β

We start by investigating how the PCA and ICA terms of the EGHR-β are related to previously proposed non-local learning rules: Oja's subspace rule for PCA11 and Bell-Sejnowski's ICA, respectively.[12]

However the value of β in the given paper[Error-Gated Hebbian Rule: A Local Learning Rule for Principal and Independent Component Analysis] is considered by hit and trial method and hence we take a look at Stochastic Gradient Descent Algorithms and Gradient Descent Algorithms.

Linear prediction methods, such as least squares for regression, logistic regression and support vector machines for classification, have been extensively used in statistics and machine learning. We study stochastic gradient descent (SGD) algorithms on regularized forms of linear prediction methods. This class of methods, related to online algorithms such as perceptron, are both efficient and very simple to implement. We obtain numerical rate of convergence for such algorithms, and discuss its implications. Experiments on text data will be provided to demonstrate numerical and statistical consequences of our theoretical.[8]

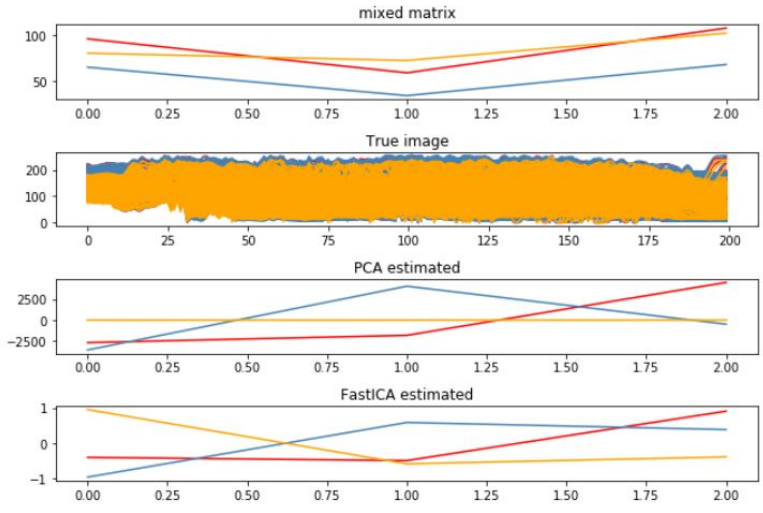
Comparison with other work

Parallel Processing

One of the most important, yet least understood, aspects of parallel processing is the effect of interconnections of registers and processing units on computational speed. Particular parallel algorithms are well suited for particular interconnection patterns, but these "good"

```
c:\users\tarsh\appdata\local\programs\python\python35\lib\site-packages\sklearn\decomposition\fastica_.py:118: UserWarning: Fa
stICA did not converge. Consider increasing tolerance or the maximum number of iterations.
  warnings.warn('FastICA did not converge. Consider increasing '

[[-0.40204012 -0.96012843  0.94677162]
 [-0.49287891  0.58193016 -0.585495  ]
 [ 0.90259743  0.38230371 -0.38867378]]
####
[[-2.66056867e+03 -3.53320063e+03  1.11896811e-11]
 [-1.80333827e+03  4.01646990e+03  1.11896811e-11]
 [ 4.46390694e+03 -4.83269269e+02  1.11896811e-11]]
#######
[[ -7.86961149  61.53718729   6.91330846]
 [-32.31226495 167.14501187  15.6896631 ]
 [-17.51085578  -8.80049455  -6.25048685]
 ...
 [ -7.35951667 -14.41695497  -4.218105  ]
 [-25.53941796 -13.12355147  -9.15904666]
 [-19.09184669 -33.48292644 -10.36098757]]
```



patterns vary widely from algorithm to algorithm.Inspired by this work, and noting an apparent architectural convergence of CPUs and GPUs, our goal in this paper is to examine the effectiveness of CUDA as a tool to express parallel computation with differing sets of performance characteristics— problems from different dwarves—on GPUs s. All of our applications show satisfactory speedups, but the main contribution of our work is a discussion of the advantages and disadvantages of the CUDA programming model in the context of a more general multicore world. CUDA is an extension to C based on a few easily-learned abstractions for parallel programming, coprocessor offload, and a few corresponding additions to C syntax. CUDA represents the coprocessor as a device that can run a large number of threads. The threads are managed by representing parallel tasks as kernels (the sequence of work to be done in each thread) mapped over a domain (the set of threads to be invoked). Kernels are scalar and represent the work to be done at a single point in the domain. The kernel is then invoked as a thread at every point in the domain. The parallel threads share memory and synchronize using barriers. Data is prepared for processing on the GPU by copying it to the graphics board's memory. Data transfer is performed using DMA and can take place concurrently with kernel processing. Once written, data on the GPU is persistent unless it is deallocated or overwritten, remaining available for subsequent kernels

---

Conclusion

As per our initial approach we work upon the finding of Error Gated Hebbian Rule in which the value of β was being calculated through hit and trial method using C code for PCA and ICA algorithms. Due to which there was a delay in finding out the optimal value of β. We take a look at NumPy algorithms to create a matrix for PCA and ICA values and use Gradient Descent and stochastic gradient descent to optimize those value through which we find out the optimal value of β. Since the calculation of matrix for PCA and ICA was being done Individually as well as the calculation for eigen vectors was separate the improvement in execution time was not much hence we implement Parallel processing through NVIDIA's CUDA technology gradually decreasing our execution time.

Using the algorithms created by us in python we get a matrix of possible Eigen Vectors for PCA equation and for ICA equation after which we plugin our values of β calculated through the loss function finally implementing that equation on the image vector to separate out the noise from the Image and get a clear resolution image.

Future Work

As of now the value/parameter or the equation created by us is only used for 2 parameter namely PCA and ICA . In future another parameter LDA can be implemented to increase the proficiency of the algorithm.

The Parallel processing implemented can still be worked upon to increase the efficiency and the time taken by the algorithm

---

References

1. Isomura, Takuya, and Taro Toyoizumi. "On the achievability of blind source separation for high-dimensional nonlinear source mixtures." arXiv preprint arXiv:1808.00668 (2018).
2. Jutten, Christian, and Jeanny Herault. "Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture." Signal processing 24.1 (1991): 1-10.
3. A. Cichocki and S. Amari. Adaptive Blind Signal and Image Processing. John Wiley & Sons Ltd, New York, 2003.
4. Bugli, Céline, and Philippe Lambert. "Comparison between principal component analysis and independent component analysis in electroencephalograms modelling." Biometrical Journal: Journal of Mathematical Methods in Biosciences 49.2 (2007): 312-327.
5. Adams, Niall M., et al. "A review of parallel processing for statistical computation." Statistics and Computing 6.1 (1996): 37-49.
6. S. Gill; Parallel Programming, The Computer Journal, Volume 1, Issue 1, 1 January 1958, Pages 2–10, https://doi.org/10.1093/comjnl/1.1.2
7. Zhang, Yunquan, et al. "Parallel processing systems for big data: a survey." Proceedings of the IEEE 104.11 (2016): 2114-2136.
8. Zhang, T., 2004, July. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In Proceedings of the twenty-first international conference on Machine learning (p. 116). ACM.
9. Tseng, P. and Yun, S., 2009. A coordinate gradient descent method for nonsmooth separable minimization. Mathematical Programming, 117(1-2), pp.387-423.
10. Hyvärinen, A. and Oja, E., 2000. Independent component analysis: algorithms and applications. Neural networks, 13(4-5), pp.411-430.
11. Hintze, J., 1998. NCSS statistical software. NCSS, Kaysville, UT.
12. Isomura, T. and Toyoizumi, T., 2018. Error-Gated Hebbian Rule: A Local Learning Rule for Principal and Independent Component Analysis. Scientific reports, 8(1), p.1835.
13. Paul, L.C., Suman, A.A. and Sultan, N., 2013. Methodological analysis of principal component analysis (PCA) method. International Journal of Computational Engineering & Management, 16(2), pp.32-38.