Online Tensor Factorization for Feature Selection in EEG

Alric Althoff

Honors Thesis, Department of Cognitive Science, University of California - San Diego

Supervised by Dr. Virginia de Sa

Abstract

Tensor decompositions are a valuable tool in data analysis, but the computational cost of standard tensor algorithms quickly becomes prohibitive, especially when considering large and time-evolving data sets such as those found in signal processing applications. In this work *multilinear PCA*, a common tensor analysis technique, will be modified to enable the processing of large scale tensorial time-evolving data, such as EEG, with much improved performance both in terms of memory and CPU time.

1 Introduction

Multilinear data analysis via tensors is a topic which has been gaining momentum in the brain-computer interface (BCI) community over the past several years. This is due to the limitations imposed by the use of matrices. In particular, standard techniques of matrix factorization, such as the Singular Value Decomposition (SVD), are not equipped to handle more than two modes, or "ways," in the data, while electroencephalogram (EEG) data consist of three or more modes—these commonly being channel, frequency, and time (Figure 1.1). Thus, while an algorithm like principle component analysis (PCA—SVD with zero-mean data) will determine the directions of greatest variance in a two mode dataset, interactions between data sources in the third mode will be lost.

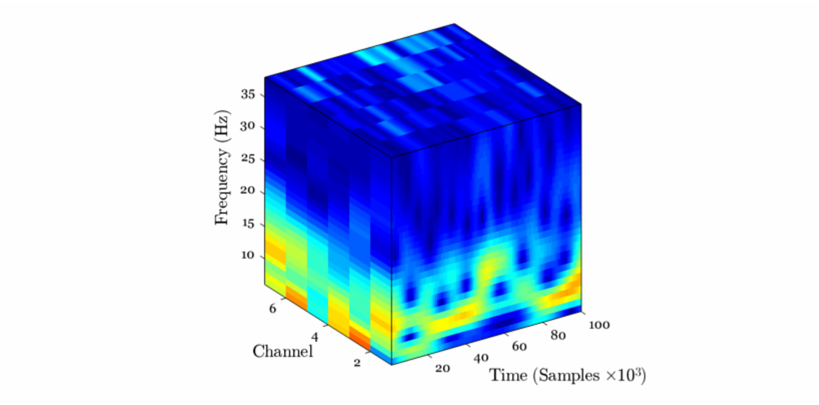

Figure 1.1: A 3-mode EEG tensor

Tensor factorizations seek to emulate the abilities of PCA in the variance-extraction sense, but are more difficult to perform and interpret. These difficulties spring both from the greater time and space complexity inherent in tensor algorithms, and the fact that notions such as rank and the value of orthogonality which apply to matrices differ in tensor applications.

The two most common tensor factorizations are the CP (CANDECOMP/PARAFAC, or recently Canonical Polyadic) and Tucker algorithms [3]. The Tucker decomposition yields projection matrices and a small core tensor (not usually orthogonal) containing *features* representing the variance of the data. While more flexible and simpler to compute, Tucker allows interactions between modes to "spill over," or more abstractly, allows for rotation.

The CP decomposition, instead of projection matrices, gives a unique set of vectors and corresponding elements of a diagonal core. CP does not have the rotation issue, simplifying analysis in many cases. However, CP is calculated using optimization strategies which may run for an indeterminate amount of time and are not guaranteed to converge for general tensors. This makes the CP decomposition a more difficult target for an online implementation.

This work is focused on speeding up a simple extension of the Tucker decomposition, *multilinear PCA*, where orthonormality of the projection matrices is enforced.

2 Background and Previous Work

The purpose of the explanations below is to introduce both the inexperienced reader and the skilled analyst with only the details most relevant to the tensor decomposition task at hand, and should not be considered exhaustive descriptions.

2.1 Singular Value Decomposition

The SVD is commonly used to find the dimension of a linear system by allowing simple determination of a statistically independent set of variables. It can be thought of as a generalization of the eigenvalue decomposition from square to rectangular matrices. The SVD decomposes any $m \times n$ matrix A into three factors, two orthonormal matrices, U and V, and a matrix Σ with decreasing real values along the main diagonal, and zeros elsewhere, such that

$$A = U\Sigma V^T \iff U^T AV = \Sigma \tag{2.1}$$

in other words, A is diagonalized by the rotators U and V.

U and V are the eigenvectors of AA^T and A^TA respectively. Because AA^T is the covariance matrix of the columns of A, and U forms an orthogonal basis for the column space of AA^T, U is a projector into the variance space of A.

2.2 Principal Component Analysis

The vectors of the matrix U corresponding to the k largest singular values can be removed to form an orthogonal projector U_k into a subspace of reduced dimensionality. Thus $U_k^T A$ is in a subspace which preserves majority of the variance of the data in A.

Because the majority contributor to the variance of A could simply be its offset from the origin, it is common to center the columns of A by subtracting their individual means[1].

[1] It is also clear from the above discussion that U_k could be formed from the eigenvectors of AA^T using an eigenvalue decomposition.

PCA is thus a technique for dimensionality reduction which preserves the basic pattern of variance in the original data. A suitable metaphor for its action might be the reduction of a piece of paper from three dimensions to two. While paper lying flat on a tabletop exists in three dimensions, one of these dimensions is much smaller than the rest, and in certain contexts, such as viewing anything written on the paper, the third dimension need not be taken into account. PCA handles this sort of transformation automatically, without the analyst needing to manually determine which of the dimensions (of perhaps hundreds) to pay attention to.

2.3 Relevant Notions in Multilinear Analysis

In many applications the real format of the data is much less like a matrix than we would prefer, given the wide variety of matrix techniques that have been developed. In these cases structuring the data so that the relationships between all the variables can be represented may reveal hidden patterns. Multilinear techniques allow for the maintenance of this natural structure through the use of tensors.

If $\mathcal{X}$ is a three-mode tensor (a rectangular "box" of values) with dimensionalities N_1, N_2 and N_3, then a *first-mode-unfolding* of $\mathcal{X}$ is written $\mathbf{X_{(1)}}$, and is a matrix of dimensionality $(N_2 \cdot N_3) \times N_1$ (Figure 2.1)[2], likewise for the other modes.

These unfoldings, or *matricized tensors*, are matrices where one whole mode of the tensor forms the column vectors. Thus matrix operations on the unfolding result in a partial operation on the whole tensor, and repeated unfoldings can be used with matrix procedures to develop tensor extensions of those algorithms. A *refolding* reverses this operation.

N-mode multiplication is one such extension, where ordinary matrix-matrix multiplications are paired with repeated unfoldings and refoldings to project one tensor to another with modified dimensionality. There are several notations in use for this operation, one of these is $\mathcal{C} = \mathcal{X} \prod_{i=1}^{M} \times_i U$ as used by [6], where $\mathcal{C}$ (a tensor) is the M mode

[2]Figure from [6].

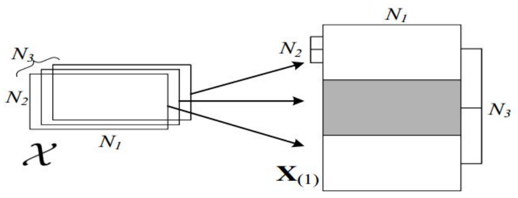

Figure 2.1: Tensor unfolding

multilinear projection of $\mathcal{X}$ over the M matrices $U_{1,2,\dots,M}$. This is generalization of matrix projection to the tensor-matrix case.

2.4 Multilinear PCA

It is a small step from here to the tensor extension of PCA, first formulated by De Lathauwer and Vandewalle in [4].

Algorithm 1 Multilinear PCA

```
Input:
An $M$-mode tensor $\mathcal{X}$
Output:
An $M$-mode tensor $\mathcal{C}$ and $M$ matrices $U_1, U_2, \dots, U_M$
Algorithm:
for $i = 1, 2, \dots, M$:
  Unfold $\mathcal{X} \to \mathbf{X_{(i)}}$
  $\mathbf{X_{(i)}} \to U_i \Sigma_i V_i^T$ via SVD
  Form projection $U_i^T \mathbf{X_{(i)}} \to \tilde{\mathbf{X}}_{\mathbf{(i)}}$
  Refold $\tilde{\mathbf{X}}_{\mathbf{(i)}} \to \tilde{\mathcal{X}}$
  $\tilde{\mathcal{X}} \to \mathcal{X}$
end
$\mathcal{X} \to \mathcal{C}$
```

The result of this procedure is that the variance and interactions between the modes of $\mathcal{X}$ are compressed into the elements of $\mathcal{C}$, while the bases for the projections are stored in $U_{1,\dots,M}$.

The downside of this algorithm is that in many cases the dimensionality of the unfolding is relatively massive, in which case computing each of the several SVDs could require large amounts of memory and CPU time. To overcome this common infeasibility, an alternative iterative optimization procedure, alternating least squares (ALS), is generally used. While this may make the impossible possible, it is more time consuming. This makes the technique hardly suitable for online applications unless the tensors are small, or memory and computational power are plentiful.

2.5 Randomized Matrix Rank Reduction

In general, the accuracy of a measurement is only guaranteed up to a few significant digits. It is therefore acceptable to use numerical techniques which, while imperfect, are precise enough for the data under analysis, and in some cases much faster than their more precise counterparts.

After the work of Halko et al. in [2], a basic intuition behind the use of randomness to speed up matrix decompositions is this: Suppose have an $m \times n$ matrix A and we seek an $n \times k$ matrix Q where $k < n$ such that

$$\|A - QQ^T A\| \tag{2.2}$$

is minimized[3]. This minimizing Q should be an orthogonal matrix that spans as much of the range of A as possible. It is convenient that a random set of vectors is likely to be linearly independent, with none in the null-space of A. This means that for some random $n \times k$ matrix Ω, $Y = A\Omega$ is also linearly independent. Once Y has been formed all that is required to obtain the desired basis for the range of A is to orthonormalize[4] Y. As is often the case in real life, just because something is likely doesn't mean that it will actually occur, and in practice it is necessary to augment Ω to $n \times (k + q)$ where $q \in [5, 10]$ to obtain a decent spanning set.

Because we are concerned with the eigenvectors of the covariance matrix, we can take $AA^T\Omega$ as our desired approximation. Accuracy can be improved by reorthonormaliz-

[3]This is equivalent to finding the eigenvectors of A.
[4]Using a QR decomposition or Gram-Schmidt process.

ing the resulting matrix between alternating A and A^T passes.

As an additional note, the ordinary $O(n^3)$ complexity of the several matrix multiplications can be reduced. Due to our ability to choose *any* random $n \times (k+q)$ linearly independent Ω, we are free to choose a circulant[5] Ω and thereby obtain the excellent matrix multiplication time complexity of $O(n^2 \log n)$, due to the following principle:

$$\Omega a = \mathcal{F}^{-1}(\mathcal{F}(\omega^{(1)}) \circledast \mathcal{F}(a))$$

where Ω is circulant and $\omega^{(1)}$ is its first column, a is a vector, and $\mathcal{F}$ is the Fourier transform operator. This result is due to the convolution theorem and detailed in [1]. An additional benefit is the ability to dispense with the need to store all of the values of Ω, as only the first column is ever used. This form of matrix multiplication was not considered in [2], and allows for additional reduction in memory use and time complexity.

2.6 Dynamic Tensor Analysis

Work by Sun et al. in [6] introduces Dynamic Tensor Analysis (DTA), an incremental algorithm for multilinear PCA which seeks to remedy the of computational complexity issue and enable online analysis. DTA xis quite similar to ordinary multilinear PCA, but adds a memory component[6] to allow for the tracking of fundamental changes in the tensor space. In practice, the addition of this memory component is costly[7] and for the purpose of feature selection, unnecessary and perhaps even detrimental. The key idea taken from this work was the notion that the additional iterations usually preformed in the ALS phase of multilinear PCA are not required to obtain sufficient accuracy for online applications.

[5]In a circulant matrix each column is a cyclic shift by one element of the preceding column.

[6]This is supplied by the covariance matrix of the last iteration.

[7]Sun presents an elegant remedy for this in the form of streaming tensor analysis (STA), which is also developed in the same paper, but is outside the scope of the current work.

2.7 Higher Order Discriminant Analysis

Phan and Cichocki in [5] present a supervised[8] method, higher order discriminant analysis (HODA), for extracting discriminative features from EEG tensors. This method, though very effective, was not designed with online decomposition in mind, and is too slow for such applications.

3 Goal of the Current Work

The fundamental goal of this research is to develop an online algorithm with discriminatory power competitive with HODA on EEG tensors. The step in that direction addressed here is the development of a fast unsupervised dimensionality reduction technique for general tensors. Implementation of discriminatory extensions will be explored later.

4 Algorithm

Given a basic understanding of the background, the principles of the algorithm of the current work are simple. The expensive SVD step in standard multilinear PCA is replaced by the randomized method of finding a reduced rank basis, yielding the form in Algorithm 2. The results are approximately the same as multilinear PCA, with a greatly reduced costs in both time and memory.

[8]Incorporating the actual class labels of training data.

Algorithm 2 Randomized Multilinear PCA

```
Input:
An $M$-mode tensor $\mathcal{X}$
Output:
An $M$-mode tensor $\mathcal{C}$ and $M$ matrices $U_1, U_2, \ldots, U_M$
Algorithm:
for $i = 1, 2, \ldots, M$:
  Unfold $\mathcal{X} \to \mathbf{X_{(i)}}$
  Let $\Omega$ be an $n \times (k+q)$ matrix with random Gaussian entries, $n$ is compatible with $\mathbf{X_{(i)}}$.
  $\mathbf{X_{(i)}\Omega} \to \mathbf{Y_0}$
  $\mathbf{Y_0} \to \mathbf{Q_0R_0}$
  for $j = 1, 2, 3$
    $\mathbf{X_{(i)}^T Q_{j-1}} \to \mathbf{\tilde{Y}_j}$
    $\mathbf{\tilde{Y}_j} \to \mathbf{\tilde{Q}_j\tilde{R}_j}$
    $\mathbf{X_{(i)}\tilde{Q}_j} \to \mathbf{Y_j}$
    $\mathbf{Y_j} \to \mathbf{Q_jR_j}$
  end
  $\mathbf{Q_k} \to \mathbf{U_{(i)}}$
end
$\mathcal{X} \prod_{i=1}^{M} \times_i \mathbf{U} \to \mathcal{C}$
```

5 Results

Prototype implementations of the classical and randomized algorithms indicate substantial performance gains by the randomized approach. Of particular note is the transition from fast (cache) memory to slower (RAM) memory as the size of the tensor increases. While initially the algorithms differ by what appears to be a constant factor, the randomized approach accesses memory less frequently, and therefore when the size of the tensor requires more memory, the speed doesn't decrease as dramatically. Figure 5.1 displays these results. Tests were also conducted with tensors of size 33×100 × 10000[9] and decompositions via the randomized algorithm completed in 8 seconds on average, while both multilinear PCA and DTA failed to terminate after 10 minutes and the process was manually exited (RAM was not exhausted at any time)[10]. The error in the approximation

[9]To put this in perspective, this equivalent to taking the SVD of a 3300×10000 matrix, a 330000×100 matrix, and a 10^6×33 matrix. Tensors of this size are not uncommon in signal processing applications, such as EEG.

[10]All analysis was carried out on laptop with a dual core AMD Brazos 1.6 GHz processor, and 4 GB of RAM

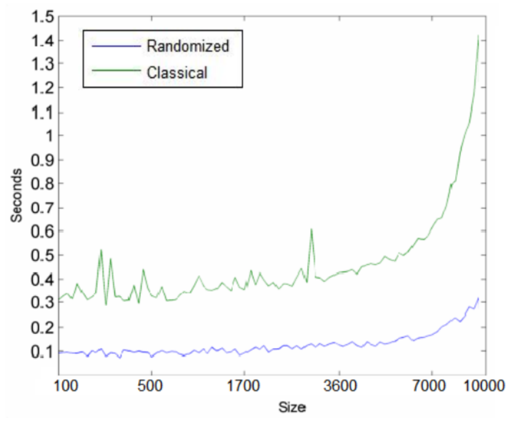

Figure 5.1: Relative performance of $33 \times 7 \times [100, 10000]$ tensors with classical multilinear PCA vs. its randomized counterpart

follows the bounds given in [2] exactly.

6 Conclusion and Future Work

As can be seen from the above results, the randomized algorithm presented here has significant speed and memory advantages over previous approaches. In the case of larger tensors, randomization took the tensor approach from a task not possible on commodity hardware to a feasible technique. In addition, general principles which can greatly improve the speed of matrix algorithms were developed and applied to the tensor case.

The next step in this work will be exploration of methods for improving the relevance of the features extracted into the core tensor to classification. As it stands, there is no way to know whether or not the variance representations in the core are indicative of changes within the data that indicate class differences. Preliminary work in this regard has been promising.

References

[1] C.M. Fiduccia. Fast matrix multiplication. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 45–49. ACM, 1971.

[2] N. Halko, P.G. Martinsson, and J.A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *Arxiv preprint arXiv:0909.4061*, 2009.

[3] T.G. Kolda and B.W. Bader. Tensor decompositions and applications. *SIAM review*, 51 (3), 2009.

[4] Lieven De Lathauwer and Joos Vandewalle. Dimensionality reduction in higher-order signal processing and rank-$(r_1, r_2, \ldots, r_n)$ reduction in multilinear algebra. *Linear Algebra and its Applications*, 391(0):31 – 55, 2004.

[5] A.H. Phan and A. Cichocki. Analysis of interactions among hidden components for tucker model. In *Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference*, pages 154–158. Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference, International Organizing Committee, 2009.

[6] J. Sun, D. Tao, S. Papadimitriou, P.S. Yu, and C. Faloutsos. Incremental tensor analysis: Theory and applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(3):11, 2008.