Efficient coordinate-descent for orthogonal matrices through Givens rotations

Uri Shalit
ICNC-ELSC & Computer Science Department
Hebrew University of Jerusalem
91904 Jerusalem, Israel
uri.shalit@mail.huji.ac.il

Gal Chechik
The Gonda Brain Research Center
Bar Ilan University
52900 Ramat-Gan, Israel
gal.chechik@biu.ac.il

Abstract

Optimizing over the set of orthogonal matrices is a central component in problems like sparse-PCA or tensor decomposition. Unfortunately, such optimization is hard since simple operations on orthogonal matrices easily break orthogonality, and correcting orthogonality usually costs a large amount of computation. Here we propose a framework for optimizing orthogonal matrices, that is the parallel of coordinate-descent in Euclidean spaces. It is based on *Givens-rotations*, a fast-to-compute operation that affects a small number of entries in the learned matrix, and preserves orthogonality. We show two applications of this approach: an algorithm for tensor decomposition that is used in learning mixture models, and an algorithm for sparse-PCA. We study the parameter regime where a Givens rotation approach converges faster and achieves a superior model on a genome-wide brain-wide mRNA expression dataset.

1 Introduction

Optimization over orthogonal matrices – matrices whose rows and columns form an orthonormal basis of $\mathbb{R}^d$ – is central to many machine learning optimization problems. Prominent examples include *Principal Component Analysis* (PCA), *Sparse PCA*, and *Independent Component Analysis (ICA)*. In addition, many new applications of tensor orthogonal decompositions were introduced recently, including Gaussian Mixture Models, Multi-view Models and Latent Dirichlet Allocation (e.g., [1, 2]).

A major challenge when optimizing over the set of orthogonal matrices is that simple updates such as matrix addition usually break orthonormality. Correcting by orthonormalizing a matrix $V \in \mathbb{R}^{d\times d}$ is typically a costly procedure: even a change to a single element of the matrix, may require $O(d^3)$ operations in the general case for re-orthogonalization. In this paper, we present a new approach for optimization over the manifold of orthogonal matrices, that is based on a series of sparse and efficient-to-compute updates that operate **within the set of orthonormal matrices**, thus saving the need for costly orthonormalization. The approach can be seen as the equivalent of coordinate descent in the manifold of orthonormal matrices. Coordinate descent methods are particularly relevant for problems that are too big to fit in memory, for problems where one might be satisfied with a partial answer, or in problems where not all the data is available at one time [3].

We start by showing that the orthogonal-matrix equivalent of a single coordinate update is applying a single *Givens rotation* to the matrix. In section 3 we prove that for a differentiable objective the procedure converges to a local optimum under minimal conditions, and prove an $O(1/T)$ convergence rate for the norm of the gradient. Sections 4.1 and 4.2 describe two applications: (1) sparse PCA, including a variant for streaming data; (2) a new method for orthogonal tensor decomposition. We study how the performance of the method depends on the problems hyperparameters using synthetic data, and demonstrate that it achieves superior accuracy on an application of sparse-PCA for analyzing gene expression data.

2 Coordinate descent on the orthogonal matrix manifold

Coordinate descent (CD) is an efficient alternative to gradient descent when the cost of computing and applying a gradient step at a single coordinate is small relative to computing the full gradient. In these cases, convergence can be achieved with a smaller number of computing operations, although using a larger number of (faster) steps. Applying coordinate descent to optimize a function involves choosing a coordinate basis, usually the standard basis. Then calculating a directional derivative in the direction of one of the coordinates. And finally, updating the iterate in the direction of the chosen coordinate.

To generalize CD to operate over the set of orthogonal matrices, we need to generalize these ideas of directional derivatives and updating the orthogonal matrix in a "straight direction". In the remaining of this section, we introduce the set of orthogonal matrices, $\mathcal{O}_d$, as a Riemannian manifold. We then show that applying coordinate descent to the Riemannian gradient amounts to multiplying by Givens rotations. Throughout this section and the next, the objective function is assumed to be a differentiable function $f : \mathcal{O}_d \to \mathbb{R}$.

2.1 The orthogonal manifold and Riemannian gradient

The orthogonal matrix manifold $\mathcal{O}_d$ is the set of $d \times d$ matrices U such that $UU^T = U^TU = I_d$. It is a $\frac{d(d-1)}{2}$ dimensional smooth manifold, and is an embedded submanifold of the Euclidean space $R^{d\times d}$ [4]. Each point $U \in \mathcal{O}_d$ has a tangent space associated with it, a $\frac{d(d-1)}{2}$ dimensional vector space, that we will use below in order to capture the notion of "direction" on the manifold. The tangent space is denoted $T_U\mathcal{O}_d$, and defined by $T_U\mathcal{O}_d = \{Z \in \mathbb{R}^{d\times d}, Z = U\Omega : \Omega = -\Omega^T\} = USkew(d)$, where $Skew(d)$ is the set of skew-symmetric $d \times d$ matrices.

The natural generalization of straight lines to manifolds are *geodesic curves*. A geodesic curve is the shortest curve between two points on the manifold, or equivalently, a curve with no acceleration tangent to the manifold [4]. For a point $U \in \mathcal{O}_d$ and a "direction" $U\Omega \in T_U\mathcal{O}_d$ there exists a single geodesic line that passes through U in direction Ω. Fortunately, while computing a geodesic curve in the general case might be hard, computing it for the orthogonal matrix manifold has a closed form expression: $\gamma : (-1, 1) \to \mathcal{O}_d$, $\gamma(\theta) = U\text{Expm}(\theta\Omega)$, where $\gamma(\theta)$ with $\theta \in (-1, 1)$ is the parameterization of the curve, and Expm is the matrix exponential function. In the special case where the operator $Expm(\Omega)$ is applied to a skew-symmetric matrix Ω, it maps Ω into an orthogonal matrix [1]. As a result, $\gamma(\theta) = U\text{Expm}(\theta\Omega)$ is also an orthogonal matrix for all $-1 < \theta < 1$. This provides a useful parametrization for orthogonal matrices.

In analogy to the Euclidean case, the Riemannian directional derivative of f in the direction of a vector $U\Omega \in T_U\mathcal{O}_d$ is defined as the derivative of a single variable function which involves looking at f along a curve [4]:

$$\nabla_\Omega f(U) \equiv \frac{\mathrm{d}}{\mathrm{d}\theta} f(\gamma(\theta))\Big|_{\theta=0} = \frac{\mathrm{d}}{\mathrm{d}\theta} f(U\text{Expm}(\theta\Omega))\Big|_{\theta=0}. \tag{1}$$

Note that $\nabla_\Omega f(U)$ is a scalar. The definition means that the directional derivative is the limit of f along the geodesic curve going through U in the direction $U\Omega$. Since the Riemannian equivalent of walking in a straight line is walking along the geodesic curve, taking a step of size $\eta > 0$ from a point $U \in \mathcal{O}_d$ in direction $U\Omega \in T_U\mathcal{O}_d$ amounts to:

$$U_{next} = U\text{Expm}(\eta\Omega). \tag{2}$$

We also have to define the orthogonal basis for $Skew(d)$. Here we use $\{e_ie_j^T - e_je_i^T : 1 \le i < j \le d\}$. We denote each basis vector as $H_{ij} = e_ie_j^T - e_je_i^T$, $1 \le i < j \le d$.

2.2 Givens rotations as coordinate descent

Coordinate descent is a popular method of optimization in Euclidean spaces. It can to be more efficient than computing full gradient steps when it is possible to (1) compute efficiently the coordinate directional derivative, and (2) apply the update efficiently. We will now show that in the case of the orthogonal manifold, applying the update (step 2) can be achieved efficiently. The cost of computing the coordinate derivative (step 1) depends on the specific nature of the objective function f, and we we show below several cases where that can be achieved efficiently.

[1]Because $\text{Expm}(\Omega)\text{Expm}(\Omega)^T = \text{Expm}(\Omega)\text{Expm}(\Omega^T) = \text{Expm}(\Omega)\text{Expm}(-\Omega) = I$

Let H_{ij} be a coordinate direction, let $\nabla_{H_{ij}} f(U)$ be the corresponding directional derivative, and choose step size $\eta > 0$. A straightforward calculation based on Eq. 2 shows that the update $U_{next} = U\text{Expm}(-\eta H_{ij})$ obeys

$$\text{Expm}(-\eta H_{ij}) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & cos(\eta) & \cdots & -sin(\eta) & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & sin(\eta) & \cdots & cos(\eta) & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

This matrix is known as a *Givens rotation* [5] and is denoted $G(i, j, -\eta)$. It has $cos(\eta)$ at the (i, i) and (j, j) entries, and $\pm sin(\eta)$ at the (j, i) and (i, j) entries. It is a simple and sparse orthogonal matrix, with a fast implementation in low-level linear algebra subroutines like BLAS. For a dense matrix $A \in \mathbb{R}^{d\times d}$, the linear operation $A \mapsto AG(i, j, \eta)$ rotates the i^{th} and j^{th} columns of A by an angle η in plane they span. Computing this operation costs $6d$ multiplications and additions. As a result, computing Givens rotations successively for all $\frac{d(d-1)}{2}$ coordinates H_{ij} takes $O(d^3)$ operations, the same order as ordinary matrix multiplication. Therefore the relation between the cost of a single Givens relative to a full gradient update is the same as the relation between the cost of a single coordinate update and a full update is in Euclidean space. We note that any determinant-1 orthogonal matrix can be decomposed into at most $\frac{d(d-1)}{2}$ Givens rotations.

2.3 Coordinate minimization algorithm with Givens rotations

Based on the definition of Givens rotation, a natural algorithm for optimizing over orthogonal matrices is to perform a sequence of rotations, where each rotation is equivalent to a coordinate-step in CD. To fully specify the algorithm we need two more ingredients: (1) Selecting a schedule for going over the coordinates and (2) Selecting a step size. For scheduling, we chose here to use a random order of coordinates, following many recent coordinate descent papers [3, 6, 7]. For choosing the step size η we use exact minimization, since we found that for the problems we attempted to solve, using exact minimization was usually the same order of complexity as performing approximate minimization (like using an Armijo step rule [8, 4]). Based on these two decisions, Algorithm (1) is a random coordinate minimization technique.

Algorithm 1 Riemannian coordinate minimization on $\mathcal{O}_d$

```
Input: Differentiable objective function $f$, initial matrix $U_0 \in \mathcal{O}_d$
  $t = 0$
  while not converged do
    1. Sample uniformly at random a pair $(i(t), j(t))$ such that $1 \leq i(t) < j(t) \leq d$.
    2. $\theta_{t+1} = \underset{\theta}{\text{argmin}}\, f\left(U_t \cdot G(i, j, \theta)\right)$.
    3. $U_{t+1} = U_t \cdot G(i, j, \theta_{t+1})$.
    4. $t = t + 1$.
  end while
Output: $U_{final}$.
```

3 Convergence rate for Givens coordinate minimization

Assume that the objective function f is differentiable.We show that Algorithm 1 converges to critical point of the function f, and the only stable convergence points are local minima. We further show that the expectation w.r.t. the random choice of coordinates of the squared l_2-norm of the Riemannian gradient converges to 0 with a rate of $O(\frac{1}{T})$ where T is the number of iterations. The proofs, including some auxiliary lemmas, are provided in the supplemental material. Overall we provide the same convergence guarantees as provided in standard non-convex optimization (e.g., [9, 8]). As in the Euclidean case, stronger assumptions, such as manifold-convexity of f, could yield stronger results. However, we have so far not found an interesting case of an objective function which is manifold-convex on the orthogonal manifold.

Theorem 1. *Convergence to local optimum*
(1) The sequence of iterates U_t of Algorithm (1) satisfies: $\lim_{t\to\infty} ||\nabla f(U_t)|| = 0$. *This means that the accumulation points of the sequence $\{U_t\}_{t=1}^{\infty}$ are critical points of f.*
(2) Assume the critical points of f are isolated. Let U_* be a critical point of f. Then U_* is a local minimum of f if and only if it is asymptotically stable with regard to the sequence generated by Algorithm (1).

Theorem 2. *Rate of convergence*
Let U_t be the sequence generated by Algorithm (1), and L be a universal Lipschitz constant for f, which always exists by compactness of $\mathcal{O}_d$ and differentiability of f. For the sequence of Riemannian gradients $\nabla f(U_t) \in T_{U_t}\mathcal{O}_d$ we have:

$$\max_{0\leq t\leq T} E\left[||\nabla f(U_t)||_2^2\right] \leq \frac{L \cdot d^2\left(f(U_0) - f_{min}\right)}{T+1} \quad . \tag{3}$$

The proof is a Riemannian version of the proof for the rate of convergence of Euclidean random coordinate descent for non-convex functions [7] and is provided as supplemental material.

4 Results

We applied Givens coordinate minimization to two problems: Sparse PCA (SPCA) and orthogonal tensor decomposition, and compare its performance with state of the art methods - the generalized power method for SPCA [10] and the tensor power method [1] for tensor decomposition. For SPCA we find our method finds sparser solutions faster than the competition. For tensor decomposition we find that our method finds better solutions than the competition, though not necessarily faster.

4.1 Sparse PCA

Principal component analysis (PCA) is a basic dimensionality reducing technique used throughout the sciences. One drawback of ordinary PCA is lack of interpretability. In the data matrix $A \in \mathbb{R}^{d\times n}$, each dimension usually has an understandable meaning, such as the level of expression of a certain gene. The dimensions of the PCA transformed matrix however are typically linear combinations of all gene expression levels, and as such are much more difficult to interpret. A common approach to finding *interpretable* principal components is Sparse PCA [11, 10]. SPCA aims to find m loading vectors $z_i \in \mathbb{R}^d$ as in PCA, which are also sparse. In the gene-expression example, the non-zero components of z_i might correspond to a few genes that explain well the structure of the data A.

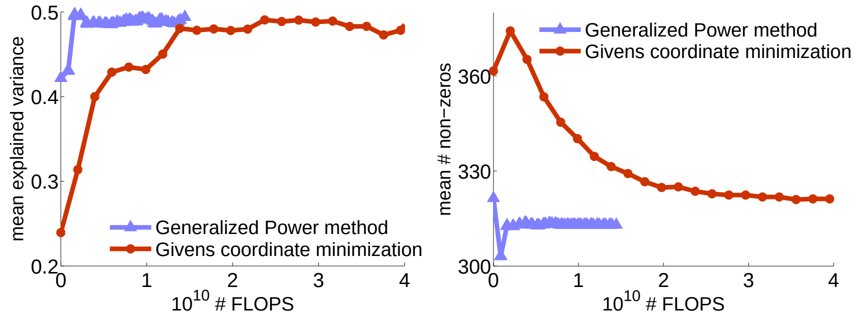
One of the most popular approaches for solving the problem of finding sparse principal components is the work by Journée et al. [10]. They show how to solve the SPCA problem by transforming it to a problem of finding a certain matrix with orthogonal columns:
$\text{argmax} \sum_{j=1}^{m}\sum_{i=1}^{d}[|(A\cdot U)_{ij}| - \gamma]_+^2$, s.t. $U \in \mathbb{R}^{n\times m}$, $U^TU = I_m$, where n is the number of samples, d is the input dimensionality and m is the number of PCA components computed. This objective is once-differentiable and the objective matrix U grows with the number of samples n.

If we set number of components m to be equal to the number of samples n we have that the objective matrix U is an orthogonal matrix, and can apply Algorithm ((1)) directly to solve the optimization problem above. At each round t, for choice of coordinates (i, j) and a matrix $U_t \in \mathcal{O}_d$, the resulting coordinate minimization problem is:

$$\begin{aligned}\underset{\theta}{\text{argmin}} - \sum_{k=1}^{d} &[|cos(\theta)(AU_t)_{ki} + sin(\theta)(AU_t)_{kj}| - \gamma]_+^2 \\ &+ [|-sin(\theta)(AU_t)_{ki} + cos(\theta)(AU_t)_{kj}| - \gamma]_+^2.\end{aligned} \tag{4}$$

In practice, there is no need to store the matrices U_t in memory, and one can work directly with the matrix AU_t. Evaluating the above expression for a given θ requires $O(d)$ operations. We found in practice that optimizing required an order of 5-10 evaluations.

In most cases researchers are interested to obtain only a small number of components, and so usually the number of components is $m << n$. We therefore developed a streaming version of our procedure. For a small given m, we treat the data as if only m samples exist at any time, giving an intermediate model $AU \in \mathbb{R}^{d\times m}$. After a few rounds of optimizing over this subset of samples, we use a heuristic to drop one of the previous samples and incorporate a new sample. This gives us a



(a) explained variance (b) number of non-zeros

Figure 1: (a) The explained variance and (b) The number of non-zeros as function of FLOPS of the coordinate minimization method and of the generalized power method be [10], on a prostate cancer gene expression dataset. The size of the sparse PCA matrix is $12,600 \times 102$.

streaming version of the algorithm because in every phase we need only m samples of the data in memory. The full details of the algorithm are given in the supplemental material.

Experiments

We first examine the case where $m = n$. We used the prostate cancer gene expression data by Singh et al. [12]. This dataset consists 102 samples over 12,600 genes. We compared the performance of our approach with that of the *Generalized Power Method* [10]. As can be seen in Figure 1, the Givens coordinate minimization method find a sparser solution with better explained variance, and does so faster than the generalized power method.

We tested the streaming version of the coordinate minimization algorithm for SPCA on a recent large gene expression data set collected from of human brains [13], measuring the expression of 20,000 genes across 3000 locations. Sparse PCA attempts to trade-off two variables: the fraction of data variance that is explained by the model's components, and the level of sparsity of the components. In our experiment, we monitor a third important parameter, the number of floating point operations (FLOPS) performed to achieve a certain solution. We again compared the performance of our approach with that of the *Generalized Power Method* [10]. We split the data into a train and test and evaluated the amount of residual variance explained by the model on the test set.

We used a range of γ values and stopping criteria for both algorithms. Each condition gives rise to a different tradeoff between sparsity, variance explained, and number of FLOPS. Figure 2 demonstrates the tradeoff between floating point operations and explained variance for sparse PCA with 3, 5 and 10 components and with sparsity levels of 5%, and 20%. Each dot represents one instance of the algorithm that was run with a certain value of γ and stopping criterion. To avoid clutter we only show instances which performed best in terms of explained variance or few FLOPS. When strong sparsity is required (5% or 10% sparsity), the Givens-rotation coordinate descent algorithm finds solutions faster (blue rectangles are more to the left in Figure 2), and these solutions are similar or better in terms of explained variance. For low-dimensional less sparse solutions (20% sparsity) we find that the generalized power method finds comparable or better solutions using the same computational cost, but only when the number of components is small, as seen in Figure 2.b,d,f.

4.2 Orthogonal tensor decomposition

Recently it has been shown that many classic machine learning problem such as Gaussian Mixture Models and Latent Dirichlet Allocation can be solved efficiently using 3rd order moments [1, 2, 14]. These methods ultimately rely on finding an orthogonal decomposition of 3-way tensors $T \in \mathbb{R}^{d\times d\times d}$, and reconstructing the solution from the decomposition. The problem of finding an orthogonal decomposition for a tensor $T \in \mathbb{R}^{d\times d\times d}$ can be naturally cast as a problem of optimization over the orthogonal matrix manifold. We apply Algorithm (1) to this problem, and compare its performance on a task of finding a Gaussian Mixture Model with a state-of-the-art tensor decomposition method, the robust Tensor Power Method [1]. We find that the Givens coordinate minimization method consistently finds better solutions when the number of mixture components is large.

Definition 1. *A symmetric tensor T is orthogonally decomposable if there exists an orthonormal set of vectors $v_1, \dots v_d \in \mathbb{R}^d$, and positive scalars $\lambda_1, \dots \lambda_d > 0$ such that $T = \sum_{i=1}^d \lambda_i (v_i \otimes v_i \otimes v_i)$.*

Unlike matrices, most symmetric tensors are not orthogonally decomposable. However, as shown by [1, 2, 15], several problems of interest, notably Gaussian Mixture Models and Latent Dirichlet

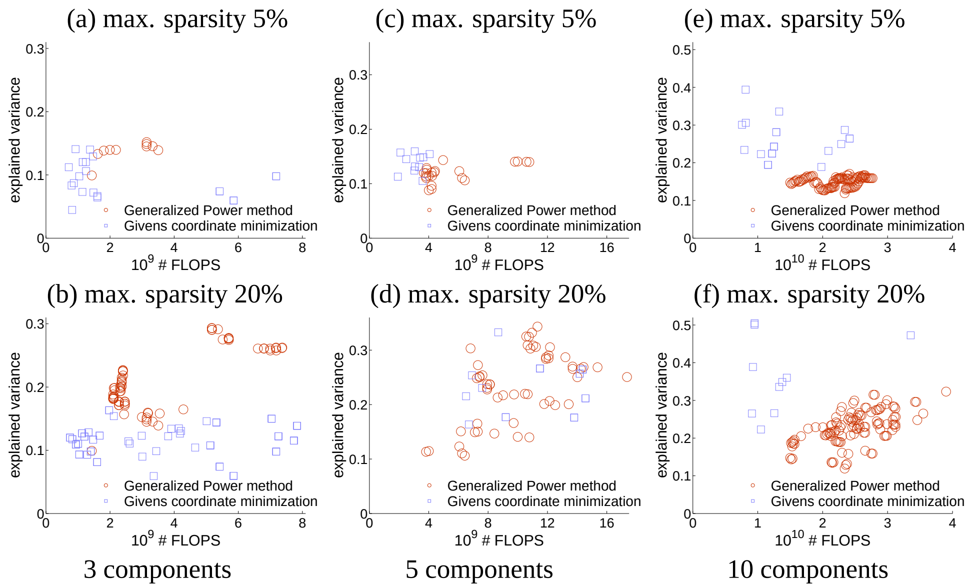


Figure 2: The tradeoff between explained variance and computational cost for 3, 5 and 10-component sparse-PCA models applied to Human gene expression data. The models are constrained for max. sparsity of 5% (top) and 20% (bottom). Red circles are instances of the Generalized Power method [10]; Blue squares represent the Givens coordinate minimization procedure.

Allocation do give rise to third-order moments which are orthogonally decomposable in the limit of infinite data. The goal of orthogonal tensor decomposition is, given an orthogonally decomposable tensor T, to find the orthogonal vector set $v_1, \dots v_d \in \mathbb{R}^d$ and the scalars $\lambda_1, \dots \lambda_d > 0$. We show that finding an orthogonal decomposition can be stated as an optimization problem over $\mathcal{O}_d$. For a vector $u \in \mathbb{R}^d$ let $T(u,u,u) = \sum_{a,b,c=1}^{d} T_{abc} u_a u_b u_c$. We have:

Theorem 3. *Let $T \in R^{d \times d \times d}$ have an orthogonal decomposition as in Definition 1, and consider the optimization problem*

$$\max_{U \in \mathcal{O}_d} f(U) = \sum_{i=1}^{d} T(u_i, u_i, u_i), \tag{5}$$

where $U = [u_1\, u_2\, \dots\, u_d]$. The stable stationary points of the problem are exactly orthogonal matrices U such that $u_i = v_{\pi(i)}$ for a permutation π on $[d]$.

The proof is given in the supplemental material. Algorithm 1 can be applied for solving the optimization problem 5, with a one-time precomputation of $O(d^4)$, and then a per-update computation of $O(d^2)$ operations. The state-of-the art method, the robust tensor power method, requires computing several tensor power iterations for finding each vector component $v_i, i = 1 \dots d$. Each such power iteration requires $O(d^3)$ operations.

Experiments

We focus on the task of fitting a Gaussian Mixture Model (GMM) with common spherical covariance by the tensor decomposition method [2, 1]. We evaluate the Givens coordinate minimization algorithm using this task and compare with the robust tensor power method, as given in [1]. We generated 20-component GMMs with dimensions ranging from 10-200, and number of samples from 10,000 - 200,000. From the samples we constructed the third order moment, decomposed it, and reconstructed the GMM following the procedure outlined in [1]. We then clustered the samples according to the reconstructed model, and measured the normalized mutual information (NMI) between the learned clustering and the true clusters.

Figure 3 (a) and (b) compares the performance of the two methods with the optimal NMI across dimensions. The coordinate minimization method outperforms the tensor power method for the large sample size (200K), whereas for small sample size (10K) the tensor power method performs better for the intermediate dimensions. Figure 3 (c) shows the performance of both algorithms across all sample sizes for dimension $= 100$. We see that the coordinate minimization method again performs better for larger sample sizes. We observed this phenomenon for 50 components as well.

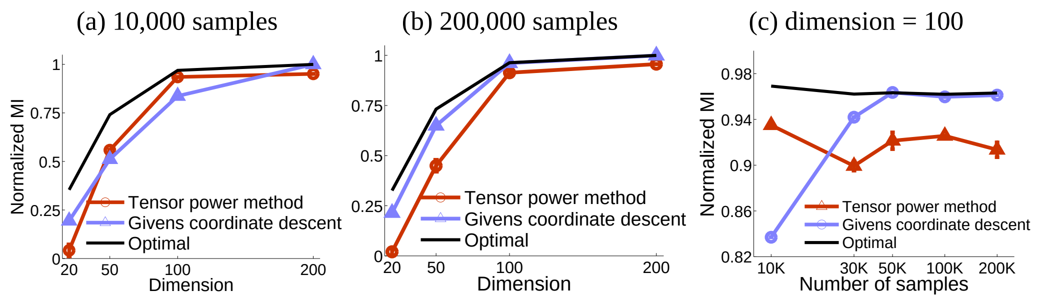


Figure 3: Clustering performance of Givens coordinate algorithm vs. the tensor power method of Anandkumar et al. [1]. Clustering by fitting a GMM from samples drawn from a 20-component GMM using 3rd order moments. The reconstruction is from (a) 10K samples and (b) 200K samples with varying dimensions, and (c) varying sample sizes with dimension $=$ 100 (note the different y-scale here). Blue line with circles marks the Givens coordinate minimization method. Red line with triangles marks the tensor power method [1], and the black line is the optimal performance if all the GMM parameters are known.

References

[1] Anima Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *arXiv preprint arXiv:1210.7559*, 2012.

[2] Daniel Hsu and Sham M Kakade. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 11–20. ACM, 2013.

[3] Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, pages 1–38, 2012.

[4] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[5] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHUP, 2012.

[6] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[7] Andrei Patrascu and Ion Necoara. Efficient random coordinate descent algorithms for large-scale structured nonconvex optimization. *arXiv preprint arXiv:1305.4027*, 2013.

[8] Dimitri P Bertsekas. *Nonlinear programming*. Athena Scientific, 1999.

[9] A Nemirovski. Optmization ii numerical methods for nonlinear continuous optimization. 1999.

[10] Michel Journée, Yurii Nesterov, Peter Richtárik, and Rodolphe Sepulchre. Generalized power method for sparse principal component analysis. *The Journal of Machine Learning Research*, 11:517–553, 2010.

[11] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.

[12] Dinesh Singh, Phillip G Febbo, Kenneth Ross, Donald G Jackson, Judith Manola, Christine Ladd, Pablo Tamayo, Andrew A Renshaw, Anthony V D'Amico, Jerome P Richie, et al. Gene expression correlates of clinical prostate cancer behavior. *Cancer cell*, 1(2):203–209, 2002.

[13] Michael J Hawrylycz, S Lein, Angela L Guillozet-Bongaarts, Elaine H Shen, Lydia Ng, Jeremy A Miller, Louie N van de Lagemaat, Kimberly A Smith, Amanda Ebbert, Zackery L Riley, et al. An anatomically comprehensive atlas of the adult human brain transcriptome. *Nature*, 489(7416):391–399, 2012.

[14] Animashree Anandkumar, Dean P Foster, Daniel Hsu, Sham M Kakade, and Yi-Kai Liu. A spectral algorithm for latent dirichlet allocation. *arXiv preprint arXiv:1204.6703*, 2012.

[15] Anima Anandkumar, Rong Ge, Daniel Hsu, and Sham M Kakade. A tensor spectral approach to learning mixed membership community models. *arXiv preprint arXiv:1302.2684*, 2013.