COMPUTATIONAL ISSUES IN PRINCIPAL COMPONENTS ANALYSIS

Daniel Povey

Microsoft Research,
Microsoft, One Microsoft Way, Redmond, WA
dpovey@microsoft.com

Ariya Rastrow

Center for Language and Speech Processing,
Johns Hopkins University, MD
ariya@jhu.edu

ABSTRACT

Many applications require a Principal Components Analysis (PCA) type of computation, in which we need to compute the largest eigenvalues and the corresponding eigenvectors, of a scatter matrix. When the dimension is large and only a small number of eigenvectors are required, techniques are available to make this computation fast. We discuss SVD on inner products of data points, and a Lanczos technique with full orthogonalization that is efficient for non-sparse matrices and is relatively easy to implement. The value of this paper is in presenting a simple, accessible recipe for this problem, since much of the literature on Lanczos techniques is geared towards applications on sparse matrices and is much more complicated than the recipe we describe here.

Index Terms— Principal Components Analysis, Singular value decomposition, Matrix decomposition, Lanczos methods, Fast PCA

1. INTRODUCTION

It is common in signal processing to encounter a PCA type of problem, and when the dimension is very large the computation required to solve it can be significant. In this paper we summarize the PCA problem and the basic SVD based approaches to solve it, and we also describe a simple and effective Lanczos procedure which can quickly solve the PCA problem when the retained dimension is much smaller than the data dimension. Lanczos methods have been extensively studied but to our knowledge a simple Lanczos recipe to solve this problem has not been published before since the literature on Lanczos methods focuses on more advanced techniques that are geared towards sparse problems.

Section 2 introduces the Singular Value Decomposition (SVD). Section 3 introduces the PCA problem. Section 4 shows why SVD solves the PCA problem. Sections 5 and 6 show how to solve the PCA problem via SVD on a scatter matrix and matrix of inner products respectively. Section 7 shows how to solve the problem via a Lanczos method, Section 8 describes our experiments mainly relating to the convergence of the Lanczos recipe, Section 9 describes some related previous work, and Section 10 summarizes our recommendations.

This work was partially conducted at the Johns Hopkins University Summer Workshop which was supported by National Science Foundation Grant Number IIS-0833652. Thanks to Ondrej Glembek and Lukas Burget of Brno University of Technology for matrix software used in experimental work, and to John Platt and Lin Xiao of Microsoft for useful discussions.

2. SINGULAR VALUE DECOMPOSITION

The Singular Value Decomposition (SVD), in the real case, is a matrix decomposition of $\mathbf{M} \in \Re^{m \times n}$ of the form

$$\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^T, \tag{1}$$

with $\mathbf{U}$ and $\mathbf{V}$ square and orthogonal (i.e. $\mathbf{U}\mathbf{U}^T = \mathbf{I}$) and $\mathbf{D}$ diagonal with nonnegative diagonal elements (but not necessarily square). See [1, Theorem 2.5.2] for a proof that the SVD exists for any $\mathbf{M}$. If $\mathbf{M}$ is real, $\mathbf{U}$, $\mathbf{D}$ and $\mathbf{V}$ will be real. A common form of the SVD is the "thin" SVD in which $\mathbf{D}$ is square and the "extra" columns of $\mathbf{U}$ or $\mathbf{V}$ are removed (so either $\mathbf{U}$ or $\mathbf{V}$ may not be square). Some formulations [1] and implementations (e.g. LAPACK) assume that $m \geq n$, and here $\mathbf{V}$ is always square. We mention in passing the "compact" SVD (discard all the zero singular values) and the "truncated" SVD (keep only a specified number of singular values).

Note that although some formulations of the SVD, e.g. [1], assume that the diagonal elements of $\mathbf{D}$ are sorted from greatest to least, and although most implementations will ensure this ordering *most of the time*, not all of them guarantee it. We refer to the sorted form as the *sorted SVD*.

2.1. Singular value decomposition in the symmetric positive semi-definite case

A special case of Singular Value Decomposition (SVD) that is applicable to the PCA application is the symmetric positive semi-definite case. If $\mathbf{A} \in \Re^{n \times n}$ is symmetric positive semi-definite and we do the singular value decomposition

$$\mathbf{A} = \mathbf{U}\mathbf{L}\mathbf{V}^T \quad \text{then} \tag{2}$$

$$\mathbf{A} = \mathbf{U}\mathbf{L}\mathbf{U}^T. \tag{3}$$

This is so because the left (or right) singular vectors of a symmetric positive semi-definite matrix are also its eigenvectors and the singular values are also its eigenvalues. It is possible to prove this fact directly using a similar approach to [1, Theorem 2.5.2]. We make use of this identity to apply SVD where it might be more appropriate to solve a symmetric eigenvalue problem; we do this because SVD code is more widely available and generally more robust, although it can be a factor of 2-3 slower than doing a symmetric eigenvalue decomposition [1]. Anywhere we use the symmetric SVD, a symmetric eigenvalue decomposition would also be appropriate (e.g. $\mathbf{A} = \mathbf{P}\mathbf{L}\mathbf{P}^{-1} = \mathbf{P}\mathbf{L}\mathbf{P}^T$, with $\mathbf{P}$ in the place of $\mathbf{U}$), but it is important to use code specific to the symmetric case (or complex eigenvalues can arise, even for symmetric inputs), to floor to zero any negative eigenvalues if we know the input is positive semi-definite, and to sort the eigenvalues from greatest to least if required.

[1] Personal communication, John Platt

3. THE PCA PROBLEM

Principal Components Analysis (PCA) is a technique used to compute the principal directions in which a set of vector quantities vary. The general scenario we are considering is as follows: suppose we have a number of vector quantities $\mathbf{c}_1 \ldots \mathbf{c}_p$, with $\mathbf{c}_i \in \Re^n$, and first compute:

$$\mathbf{M} = \sum_{i=1}^{p} \mathbf{c}_i \mathbf{c}_i^T. \quad (4)$$

$\mathbf{M} \in \Re^{n \times n}$ is the scatter of the vectors and it will be symmetric positive semi-definite. If we want to compute a weighted sum of vectors with nonnegative weights we can do it in this framework by absorbing the square root of the weight into the vectors $\mathbf{c}_i$. If the problem involves weights that can be negative, it is beyond the scope of this paper. In some applications we will already have removed the mean of the $\mathbf{c}_i$ before computing the scatter in Equation 4.

The PCA problem is, for some dimension k such that $1 \leq k < n$, to find a matrix $\mathbf{T} \in \Re^{n \times k}$ such that the columns of $\mathbf{T}$ are orthonormal, and the quantity $\mathrm{tr}\,(\mathbf{P})$ is maximized, where

$$\mathbf{P} = \mathbf{T}^T \mathbf{M} \mathbf{T}. \quad (5)$$

4. PCA AND THE SVD

In this section we show why the Singular Value Decomposition gives a solution to the PCA problem. This is a very standard result but we derive it briefly here; the reader may skip to Section 5 to avoid the proof. Firstly, consider the matrix $\mathbf{S} \in \Re^{n \times n}$ which is $\mathbf{T}$ extended to be a square orthogonal matrix, i.e. $[\mathbf{s}_1 \ldots \mathbf{s}_k] = \mathbf{T}$ and $\mathbf{S}\mathbf{S}^T = \mathbf{I}$. Consider the matrix

$$\mathbf{Q} = \mathbf{S}^T \mathbf{M} \mathbf{S}, \quad (6)$$

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{12}^T & \mathbf{Q}_{22} \end{bmatrix} \quad (7)$$

where $\mathbf{Q}_{11}$ is a k by k matrix and $\mathbf{Q}_{22}$ is $(n-k)$ by $(n-k)$, and $\mathbf{P}$ is equivalent to $\mathbf{Q}_{11}$ whose trace we are maximizing. We first show that for any orthogonal $\mathbf{T}$ which solves the PCA problem (maximizes $\mathrm{tr}\,(\mathbf{Q}_{11})$), and any orthogonal square $\mathbf{S}$ extended from it, we will have $\mathbf{Q}_{12} = \mathbf{0}$. We do this by assuming that q_{ij} is nonzero for some $i \leq k$ and $j > k$ and deriving a contradiction. Consider the Givens rotation $\mathbf{G} = G(i, j, \theta)$, which is a unit matrix except for $g_{ii} = g_{jj} = c$, $g_{ij} = s$, $g_{ji} = -s$, where $c = \cos(\theta)$, $s = \sin(\theta)$. A Givens rotation is orthogonal for any value of θ, so $\mathbf{G}\mathbf{S}^T$ is also orthogonal. Let $\tilde{\mathbf{Q}} = \mathbf{G}\mathbf{Q}\mathbf{G}^T$. We can show that around $\theta = 0$, $\partial / \partial\theta (\mathrm{tr}\,\tilde{\mathbf{Q}}_{11})$ equals $q_{ij} + q_{ji} = 2q_{ij}$, which is nonzero since we stated that $q_{ij} \neq 0$, and clearly $\mathrm{tr}\,(\tilde{\mathbf{Q}}_{11})$ is a smooth function of θ, so by varying θ we can increase the value of $\mathrm{tr}\,(\tilde{\mathbf{Q}}_{11})$ which means we were not at an optimal solution.

The next two steps of the proof are to show that for any proposed solution we can find a solution with the same objective function such that $\mathbf{Q}_{11}$ and $\mathbf{Q}_{22}$ are diagonalized. We restrict ourselves to diagonal solutions, and then to show that no diagonal element of $\mathbf{Q}_{22}$ can be greater than a diagonal element of $\mathbf{Q}_{11}$. At this point we can show that the first k columns of $\mathbf{U}$ in a sorted SVD $\mathbf{M} = \mathbf{U}\mathbf{L}\mathbf{V}$ give an optimal value of $\mathbf{T}$.

5. PCA VIA SVD ON SCATTER

Here we summarize the simplest way to compute the PCA projection. We construct $\mathbf{M} = \sum_{i=1}^{p} \mathbf{c}_i \mathbf{c}_i^T$, do the sorted singular decomposition $\mathbf{M} = \mathbf{U}\mathbf{L}\mathbf{V}^T$ (which implies $\mathbf{M} = \mathbf{U}\mathbf{L}\mathbf{U}^T$), and the answer is the matrix $\mathbf{T}$ consisting of the first k columns of $\mathbf{U}$. The corresponding eigenvalues (if needed for diagnostics, for example) are the first k diagonal elements of $\mathbf{L}$. This algorithm will take approximately pn^2 flops to compute $\mathbf{M}$ and $12n^3$ flops to compute the SVD [1, Section 5.4.4].

6. PCA VIA SVD ON INNER PRODUCTS

If the number of data points p is smaller than the dimension n of the vectors, it is efficient to do the SVD on a matrix of data inner products. Let $\mathbf{C}$ be the n by p matrix, each of whose columns is a data point $\mathbf{c}_i$, so $\mathbf{M} = \mathbf{C}\mathbf{C}^T$. We can compute PCA via a smaller SVD on the p by p matrix

$$\mathbf{B} = \mathbf{C}^T \mathbf{C}. \quad (8)$$

We should exploit symmetry when computing $\mathbf{B}$. Consider the SVD $\mathbf{B} = \mathbf{U}\mathbf{L}\mathbf{V}^T$, implying

$$\mathbf{B} = \mathbf{U}\mathbf{L}\mathbf{U}^T \quad (9)$$

by positive semi-definiteness of $\mathbf{B}$. Initially assuming that $\mathbf{L}$ has no zero diagonal entries, we can do the following steps:

$$\mathbf{M} = \mathbf{C}\mathbf{C}^T \quad (10)$$

$$= \mathbf{C}(\mathbf{U}\mathbf{L}^{-0.5}\mathbf{L}\mathbf{L}^{-0.5}\mathbf{U}^T)\mathbf{C}^T \quad (11)$$

$$= (\mathbf{C}\mathbf{U}\mathbf{L}^{-0.5})\mathbf{L}(\mathbf{L}^{-0.5}\mathbf{U}^T\mathbf{C}^T) \quad (12)$$

$$= \mathbf{W}\mathbf{L}\mathbf{W}^T, \text{ where} \quad (13)$$

$$\mathbf{W} = \mathbf{C}\mathbf{U}\mathbf{L}^{-0.5}, \quad (14)$$

and $\mathbf{W}$ is orthogonal because $\mathbf{W}^T\mathbf{W} = \mathbf{L}^{-0.5}\mathbf{U}^T\mathbf{C}^T\mathbf{C}\mathbf{U}\mathbf{L}^{-0.5} = \mathbf{L}^{-0.5}\mathbf{U}^T\mathbf{B}\mathbf{U}\mathbf{L}^{-0.5} = \mathbf{I}$. The answer $\mathbf{T}$ will be the first k columns of $\mathbf{W}$, and the associated singular values are the first k diagonal elements of $\mathbf{L}$. In this case (13) is a "compact" SVD of $\mathbf{M}$. We can cover the case where $\mathbf{L}$ has zero diagonal elements by truncating the dimension of $\mathbf{L}$ and $\mathbf{U}$ (and hence of $\mathbf{W}$). The algorithm below takes special care to handle the case where the ratio of singular values is large which can cause numerical problems, or where the number of points p is less than k, by orthonormalizing the columns of $\mathbf{T}$. Naturally it makes no sense to have $k > p$, but from a software engineering point of view it is advisable to handle these conditions gracefully. The threshold of 1000 is the amount of relative precision in the orthogonality of $\mathbf{T}$ we are willing to lose in order to avoid reorthogonalizing.

Algorithm *PCA on inner products*($\mathbf{C}$)

1. $\mathbf{B} \leftarrow \mathbf{C}^T\mathbf{C}$
2. Do the sorted SVD $\mathbf{B} = \mathbf{U}\mathbf{L}\mathbf{V}^T$.
3. **if** $k \leq p$
4. $\quad \mathbf{T} \leftarrow \mathbf{C}\,[\mathbf{u}_1 \ldots \mathbf{u}_k]$, i.e. $\mathbf{C}$ times the first k columns of $\mathbf{U}$.
5. **else**
6. $\quad \mathbf{R} \leftarrow \mathbf{C}\,[\mathbf{u}_1 \ldots \mathbf{u}_p]$
7. $\quad \mathbf{S} \in \Re^{n \times (p-k)} \leftarrow$ matrix with random elements
8. $\quad \mathbf{T} \leftarrow [\mathbf{R}\,\mathbf{S}]$
9. **if** $k \leq p \wedge l_{kk} > l_{11}/1000$
10. $\quad$ **for** $i \leftarrow 1$ **to** k,
11. $\quad\quad \mathbf{t}_i \leftarrow l_{ii}^{-0.5}\mathbf{t}_i$, i.e. normalize columns of $\mathbf{t}$.

12. **else**
13. **for** $i \leftarrow 1$ **to** k,
14. $x \leftarrow \|t_i\|$
15. **for** $j \leftarrow 1$ **to** $i-1$,
16. $\mathbf{t}_i \leftarrow \mathbf{t}_i - (\mathbf{t}_i \cdot \mathbf{t}_j)\mathbf{t}_j$
17. **if** $\|\mathbf{t}_i\| < x/2$,
18. **if** $\|\mathbf{t}_i\| = 0$,
19. $\mathbf{t}_i \leftarrow$random vector
20. **goto** 14.
21. **else**
22. $\mathbf{t}_i \leftarrow \|\mathbf{t}_i\|^{-1}\mathbf{t}_i$
23. The singular values are $l_{ii} \ldots l_{kk}$ (if $k \leq p$) and $l_i \ldots l_{kk}\, 0\, 0 \ldots$ otherwise.
24. Return $\mathbf{T}$, and the singular values if needed.

The flop count of this algorithm is p^2n to compute $\mathbf{B}$, $12p^3$ to compute the SVD, $2npk$ to compute $\mathbf{T}$ and nk^2 to orthogonalize, which overall is $O(p^2n)$ assuming $k \leq p \leq n$.

7. COMPUTING PCA VIA LANCZOS ITERATIONS

The Lanczos method for finding the largest magnitude eigenvalues and corresponding eigenvectors (or equivalently, the largest singular values and singular vectors) of a symmetric matrix, consists of starting with a random vector and repeatedly multiplying by the matrix, keeping the resulting vectors and orthogonalizing as we go. This is related to the power method, but it leads to more exact results as we work with the entire sequence of vectors rather than just keeping the last one.

We first explain the technique assuming we have already computed the scatter matrix $\mathbf{M}$; we will later generalize to an operation on the original vectors $\mathbf{v}_i$. Assuming we run the Lanczos method for q iterations, it generates an orthonormal sequence of vectors $\mathbf{v}_i$, $1 \leq i \leq q$, and a corresponding sequence of vectors $\mathbf{w}_i$ such that $\mathbf{w}_i = \mathbf{M}\mathbf{v}_i$. Each unit-length vector $\mathbf{v}_{i+1}$ is set to the direction in $\mathbf{w}_i$ that is orthogonal to all previous $\mathbf{v}_i$. If $\mathbf{V}$ is a matrix whose columns are all the $\mathbf{v}_i$ and $\mathbf{W}$'s columns are $\mathbf{w}_i$, then $\mathbf{W} = \mathbf{M}\mathbf{V}$. Let $\mathbf{P} = \mathbf{V}^T\mathbf{W} = \mathbf{V}^T\mathbf{M}\mathbf{V}$. Because each $\mathbf{w}_i$ lies in the space spanned by $\mathbf{v}_1 \ldots \mathbf{v}_{i+1}$, and all the $\mathbf{v}_i$'s are mutually orthogonal, each $\mathbf{v}_i$ is orthogonal to $\mathbf{w}_1 \ldots \mathbf{w}_{i-2}$. This implies that $p_{ij} = 0$ for $j < i-1$. But $\mathbf{P}$ is symmetric so it is tridiagonal. $\mathbf{P}$ is what we get by projecting $\mathbf{M}$ into the subspace defined by the columns of $\mathbf{V}$. The algorithm works by doing the SVD on $\mathbf{P}$, selecting the top eigenvectors and eigenvalues, and using the columns of $\mathbf{V}$ to get the representation in the original space. Following Kaniel-Page theory (see [1]), after enough iterations the result will tend to approach the result obtained using conventional SVD.

Note that $\mathbf{P}$ is tridiagonal and in principle we should take advantage of this somehow, but tridiagonal SVD routines are less widely available than general SVD routines and this part of the calculation does not dominate the overall process, so we recommend to use normal SVD for convenience. On each iteration we orthogonalize $\mathbf{v}_{i+1}$ against all the previous $\mathbf{v}_j$ (not just $\mathbf{v}_i$ and $\mathbf{v}_{i-1}$), even though it is mathematically orthogonal for $j < i-1$. This procedure is known as Lanczos method "with complete orthogonalization" (c.f. [1, Section 9.2.3]), and it guards against numerical instabilities. Much of the Lanczos literature focuses on ways to avoid complete orthogonalization.

In order to test for convergence we measure a quantity r which is the summed length of the residual vectors $\mathbf{r}_i = \mathbf{M}\mathbf{t}_i - l_{ii}\mathbf{t}_i$, where $\mathbf{t}_i$ is the i'th approximated singular vector and l_{ii} is the corresponding singular value. $\mathbf{r}_i$ is the error in the eigenvalue equation. We compute

$$r = \sum_{i=1}^{k} \|\mathbf{r}_i\| \tag{15}$$

where $\|\mathbf{r}\|$ is the 2-norm $\sqrt{\mathbf{r} \cdot \mathbf{r}}$, and compare r to $\epsilon \sum_{i=1}^{k} l_{ii}$ (e.g. for $\epsilon = 10^{-5}$); if it is larger, we do more iterations. We increase the number of iterations in substantial chunks as this comparison requires the SVD. Space does not permit a derivation of our formula for $\|\mathbf{r}_i\|$ in the algorithm.

The algorithm is below. The vectors $\mathbf{d}$ and $\mathbf{e}$ contain the diagonal and sub-diagonal elements of $\mathbf{P}$. Lines 13 to 20 relate to orthogonalization (necessary because of finite precision arithmetic) and restarting (Line 16).

Algorithm *Lanczos*($\mathbf{M} \in \Re^{n \times n}, 0 < k < n, k < q \leq n, \epsilon$)

1. $q \leftarrow \min(2k+20, n)$ // This is a heuristic, only require $k \leq q \leq n$
2. Allocate $\mathbf{V} \in \Re^{n \times q}$, $\mathbf{d} \in \Re^q$, $\mathbf{e} \in \Re^q$ and $\mathbf{u} \in \Re^n$.
3. Set $\mathbf{u}$ to a random, unit-length vector // Throughout, $\mathbf{u} \equiv \mathbf{v}_{i+1}$
4. $i \leftarrow 1$
5. **while true**:
6. $\mathbf{v}_i \leftarrow \mathbf{u}$
7. $\mathbf{u} \leftarrow \mathbf{M}\mathbf{v}_i$
8. $d_i \leftarrow \mathbf{u} \cdot \mathbf{v}_i$
9. $\mathbf{u} \leftarrow \mathbf{u} - d_i\mathbf{v}_i$.
10. **if** $i > 1$
11. $\mathbf{u} \leftarrow \mathbf{u} - e_{i-1}\mathbf{v}_{i-1}$
12. $e_i \leftarrow \|\mathbf{u}\|$
13. **if** $i < n$
14. $x \leftarrow \|\mathbf{u}\|$
15. **if** $x = 0$
16. $\mathbf{u} \leftarrow$ random nonzero vector, $x \leftarrow \|\mathbf{u}\|$.
17. **for** $j \leftarrow 1$ **to** i,
18. $\mathbf{u} \leftarrow \mathbf{u} - \mathbf{v}_j(\mathbf{u} \cdot \mathbf{v}_j)$.
19. **if** $\|u\| \leq \frac{1}{2}x$
20. **goto** 14
21. $\mathbf{u} \leftarrow \|\mathbf{u}\|^{-1}\mathbf{u}$
22. **if** $i = q$
23. Construct symmetric tridiagonal matrix $\mathbf{P}$ whose diagonal is $\mathbf{d}$ and whose sub-diagonal and super-diagonal are equal to $\mathbf{e}_{1:q-1}$.
24. Do sorted SVD $\mathbf{P} = \mathbf{X}\mathbf{L}\mathbf{Y}^T$ (discard $\mathbf{Y}$).
25. $r \leftarrow |e_q| \sum_{j=1}^{k} |x_{qj}|$.
26. **if** $q < n \wedge r > \epsilon \sum_{j=1}^{k} l_{jj}$,
27. $q \leftarrow \min(n, \lceil 1.5q \rceil)$ // This is a heuristic, must increase q and need $q \leq n$
28. Extend arrays $\mathbf{V}$, $\mathbf{d}$, $\mathbf{e}$ while copying contents
29. **else** $\mathbf{T} \leftarrow \mathbf{V}[\mathbf{x}_1 \ldots \mathbf{x}_k]$
30. **return** $\mathbf{T}$ and the eigenvalues $l_{11} \ldots l_{kk}$.
31. $i \leftarrow i+1$

Assuming we construct $\mathbf{M}$ explicitly and we do not have to redo the SVD, the flop count for this algorithm would be pn^2 to compute $\mathbf{M}$, $\sim 2qn^2$ for the Lanczos iterations, q^2n for the reorthogonalizatoin, $12q^3$ for the SVD on $\mathbf{P}$ (which could be done faster), and $2kqn$ for the final multiply. This will always be dominated by the time to construct $\mathbf{M}$ itself, assuming $q \ll n$ and $q \ll p$.

7.1. Lanczos on data directly

It will always be faster to do the Lanczos algorithm on the vectors $\mathbf{c}_i$ directly if they are available rather than constructing $\mathbf{M}$. In this case we can just replace the statement $\mathbf{u} \leftarrow \mathbf{M}\mathbf{v}_i$ in the above algorithm with: $\mathbf{u} \leftarrow \sum_{j=1}^{p}(\mathbf{c}_j \cdot \mathbf{v}_i)\mathbf{c}_j$. In this case the flop count

becomes $\sim 2npq$ for the Lanczos iterations, q^2n for reorthogonalization, $12q^3$ for the SVD and $2kqn$ for this final matrix multiply. The only reasons not to do this relate to convenience, memory and parallelizability.

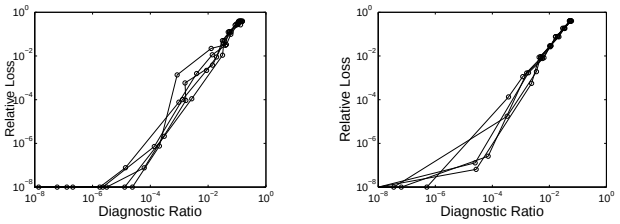
8. EXPERIMENTAL RESULTS

We have tested four different algorithms for PCA on random data to verify that they give the same results. The four different algorithms were: SVD on the scatter $\mathbf{M}$, SVD on the inner-product matrix $\mathbf{B}$, Lanczos on the scatter $\mathbf{M}$, and Lanczos on the data. The main reason for this exercise was to verify that the algorithms described above work and are are robust to various pathologies in the data. The different scenarios we tested included different combinations of $k = n$ vs $k < n$, $p < n$ vs $p \geq n$, and $k \geq p$ vs $k < p$. We also tested with data points that were generated to be linearly dependent in a dimension smaller than k, and with zero data. The feature dimension n that we used for these tests varied from 10 to 100.

We first checked the orthogonality of the PCA solution matrix $\mathbf{T}$ by measuring $\|\mathbf{T}^T\mathbf{T} - \mathbf{I}\|_F^2$, i.e. the sum of squared elements of the difference of $\mathbf{T}^T\mathbf{T}$ from the unit matrix. This was similar for all algorithms, with an error of around 10^{-31} in double precision arithmetic for all methods used, and around 10^{-13} in single precision arithmetic.

We then checked the value of the objective function $\mathrm{tr}\,(\mathbf{T}^T\mathbf{M}\mathbf{T})$. All four techniques give approximately the same answer, although as expected the Lanczos-based techniques tended to have a slightly smaller objective function, e.g. by a factor of $1 - 10^{-6}$.

In order to investigate how small the constant ϵ must be set in order to guarantee sufficiently accurate results, we did the following experiment. For a specified n and k we generated random (Gaussian) data and did PCA on its scatter using the Lanczos technique with various specified numbers of iterations (i.e. we did not use the heuristics described in the algorithm above). Each time, after computing the SVD on the last iteration of the Lanczos method, we computed the "diagnostic ratio" $R = r / \sum_{i=1}^{k} l_{ii}$, with r as defined in Equation 15 and $\sum_{i=1}^{k} l_{ii}$ being the sum of singular values computed in the Lanczos method. This ratio relates to our stopping criterion: if $R > \epsilon$, we will do more iterations. We also computed the "relative loss" $0 \leq l \leq 1$ which is the difference between $\mathrm{tr}\,(\mathbf{T}^T\mathbf{M}\mathbf{T})$ where $\mathbf{T}$ is computed using this Lanczos approach, versus the exact value as computed by SVD; and expressed it as a ratio versus the exact value.



(a) $n = 100$, $k = 10$. Increments of 4 iterations. (b) $n = 500$, $k = 50$. Increments of 25 iterations.

Fig. 1. Relative loss vs. diagnostic ratio.

Figure 1 shows these results. Each line corresponds to a different random matrix. Relative loss numbers are floored to 10^{-8}; these floored values are usually exactly zero. Figure 1 confirms that, at least for data of this type, as long as ϵ is smaller than about 10^{-2} the relative loss will tend to be less than ϵ. That is, using our recommended value of $\epsilon = 10^{-5}$ will tend to ensure a relative loss even smaller than 10^{-5}. No bound of this sort can be proved however, since it is quite possible to pick starting directions (and re-starting directions, line 16) such that the Lanczos method will give very poor results. It is just possible to see from Figure 1 how many iterations it takes to converge to our chosen threshold. The top right points always correspond to k iterations, and then we go in increments of 4 or 25. For $n = 100$, $k = 10$, we are converging after about 30 Lanczos steps and for $n = 500$, $k = 50$ we are converging after about 200 steps. When initializing q in the algorithm we have attempted to "guess" how long it will take to converge, to avoid unnecessary SVD steps.

Scenario	Recommend	Cost (parallelizable cost) (Note, $q \simeq 2k + 10$)
$k \simeq n, p \geq n$	SVD on $\mathbf{M}$	$pn^2 + 12n^3$
$k \simeq p, p < n$	SVD, inner-prod	$p^2n + 12p^3$
$p \gg n, k \ll n$ Memory limited/ parallelized	Lanczos on $\mathbf{M}$	$pn^2 + 2qn^2 + q^2n + 12q^3 + 2kqn \simeq pn^2 + 2qn^2$
$k \ll n, k \ll p$	Lanczos on data	$2qnp + q^2n + 12q^3 + 2qnk \simeq 2qnp$

Table 1. Recommended techniques to solve PCA problem

9. OTHER APPROACHES

We note that the PCA problem can be solved by an Estimation Maximization approach [2] which when $k = 1$ is equivalent to the power method for finding the largest eigenvalue of $\mathbf{M}$, but the matrix-vector product is computed via the data points $\mathbf{c}_i$. For $k > 1$ this approach is equivalent to a block form of the power method; such a formulation is described in [3] but with a much more efficient implementation avoiding unnecessary matrix inversions. We do not discuss these types of methods further because Lanczos techniques converge much faster.

10. CONCLUSIONS

We have discussed the PCA problem and described a simple Lanczos method to solve it, and have investigated its performance. Many more advanced Lanczos methods exist; the contribution of this paper is to make available a simple Lanczos method in the form of a recipe, since the difficulty and obscurity of the literature on Lanczos methods has prevented them from being widely used. Table 1 summarizes the techniques we recommend to solve the PCA problem in various situations if time is critical. When reading Table 1, bear in mind that Lanczos techniques should be avoided if a simpler method will suffice. We distinguish the cost arising from computing the scatter matrix $\mathbf{M}$ by font color, since it is more trivially parallelizable than the other steps. Note also that PCA on very sparse vectors will tend to require more advanced Lanczos methods without complete orthogonalization, because in that case the orthogonalization step comes to dominate the time taken.

11. REFERENCES

[1] Golub and van Loan, *Matrix computations*, Johns Hopkins University Press, 3 edition, 1983.

[2] S. Roweis, "EM algorithms for PCA and SPCA.," in *NIPS*, 1997.

[3] Paliwal K. K. Sharma A., "Fast principal component analysis using fixed-point algorithm," *Pattern Recognition Letters*, vol. 28, pp. 1151–1155, 2007.