Some Examples, Questions, and Answers: Lectures 3,4,5 – COMPSCI 369

Georgy Gimel'farb

Solving Systems of Linear Equations

Example 3-4-5.1: Computing SVD of a given 4×3 matrix $\mathbf{A} = \begin{bmatrix} 2. & 3. & 0. \\ 2. & 2. & 0. \\ 3. & 0. & 1. \\ 3. & 2. & 2. \end{bmatrix}$:

$$\mathbf{A} = \underbrace{\begin{bmatrix} -0.512 & -0.132 & -0.612 \\ -0.423 & -0.384 & 0.299 \\ -0.407 & -0.554 & -0.687 \\ -0.627 & 0.727 & -0.255 \end{bmatrix}}_{\mathbf{U}} \underbrace{\begin{bmatrix} 6.380 & 0.000 & 0.000 \\ 0.000 & 1.076 & 0.000 \\ 0.000 & 0.000 & 2.477 \end{bmatrix}}_{\mathbf{D}} \underbrace{\begin{bmatrix} -0.779 & -0.570 & -0.260 \\ 0.477 & 0.271 & 0.836 \\ -0.406 & 0.776 & -0.483 \end{bmatrix}}_{\mathbf{V}^{\mathsf{T}}}$$

Use the C-subroutine `svd.c` in the file `sources.c` – see e.g. the following simple example:

```
int main(){
      double **A,      // the m x n matrix to be factored into SVD U diag{W} V^T
             *W,       // the n-array of singular values
             **U,      // the m x n column-orthonormal matrix of eigen-vectors of AA^T
             **V,      // the n x n  orthonormal matrix of eigen-vectors of A^TA
             *rv1;     // the working space
      double ini1[4][3] = { { 2., 3., 0.}, {2., 2., 0.}, {3., 0., 1.}, {3., 2., 2.} };

      int    i, j, index;
      const double error = 1.e-6, // parameters of svd.c
                   tolerance = 1.e-6;
      const int m = 4, n = 3, withu = 1, withv = 1;

      A = calloc( m, sizeof( double* ) );
      for( i = 0; i < m; i++ ) {
           A[i] = calloc( n, sizeof( double ) );
           for( j = 0; j < n; j++ ) {
                A[ i ][ j ] = ini1[ i ][ j ];
           }
      }
```

```
      U = calloc( m, sizeof( double* ) );
      for( i = 0; i < m; i++ )
           U[ i ] = calloc( n, sizeof( double ) );
      V = calloc( n, sizeof( double* ) );
      for( i = 0; i < n; i++ )
            V[ i ] = calloc( n, sizeof( double ) );
      W = calloc( n, sizeof( double ) );
      rv1 = calloc( n, sizeof( double ) );

      index = svd( m, n, withu, withv, error, tolerance, A, W, U, V );
      printf( "\nOutput index = %d", index );

      if ( index == 0 ) {
          printf( "\n%d x %d matrix A", m, n );
          for( i = 0; i < m; i++ ) {
                printf("\n  Row %2d: ", i );
                for( j = 0; j < n; j++ ) printf( " %6.3lf", A[i][j] );
          }
          printf( "\n%d x %d column-orthonormal matrix U", m, n );
          for( i = 0; i < m; i++ ) {
                printf("\n  Row %2d: ", i );
                for( j = 0; j < n; j++ ) printf( " %6.3lf", U[i][j] );
           }
           printf( "\n%d x %d diag matrix D of singular values\n  Diag values: ", n, n );
           for( i = 0; i < n; i++ ) printf( " %9.7lf", W[i] );
           printf( "\n%d x %d orthonormal matrix V: ", n, n );
           for( i = 0; i < n; i++ ) {
                 printf("\n  Row %2d: ", i );
                 for( j = 0; j < n; j++ ) printf( " %6.3lf", V[i][j] );
           }
     }
}
```

Printout of the above program:

```
Output index = 0
4 x 3 matrix A
  Row  0:   2.000  3.000  0.000
  Row  1:   2.000  2.000  0.000
  Row  2:   3.000  0.000  1.000
  Row  3:   3.000  2.000  2.000
4 x 3 column-orthonormal matrix U
  Row  0:  -0.512 -0.132  0.612
  Row  1:  -0.423 -0.384  0.299
  Row  2:  -0.407 -0.554 -0.687
  Row  3:  -0.627  0.727 -0.255
3 x 3 diag matrix D of singular values
  Diag values:  6.3801470 1.0755882 2.4772635
3 x 3 orthonormal matrix V:
  Row  0:  -0.779 -0.477 -0.406
  Row  1:  -0.570  0.271  0.776
  Row  2:  -0.260  0.836 -0.483
```

Question 3-4-5.1: Compute SVD of a given 6×2 matrix $\mathbf{A} = \begin{bmatrix} 1. & 2. \\ 2. & 3. \\ 3. & 4. \\ 4. & 5. \\ 5. & 6. \\ 6. & 7. \end{bmatrix}$

Example 3-4-5.2: Approximating the matrix $\mathbf{A}$ in Example 3-4-5.1 with k; $1 \leq k \leq n$, top-rank singular values:

$$\widehat{\mathbf{A}}_k = \sum_{\kappa=1}^{k} \sigma_\kappa \mathbf{u}_\kappa \mathbf{v}_\kappa^\mathsf{T}$$

where σ_κ is the κ-th ordered singular value; $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$, and $\mathbf{u}_\kappa$ and $\mathbf{v}_\kappa$ denote the relevant columns in $\mathbf{U}$ and $\mathbf{V}$, respectively.

The singular values along the main diagonal of $\mathbf{D}$ have to be in the decreasing order and the relevant columns of $\mathbf{U}$ and $\mathbf{V}$ have to be permuted accordingly. The SVD algorithm in the subroutine `svd.c` does not provide exactly this order, although returns typically almost ordered arrays W; $\mathbf{D} = \mathrm{diag}\{W\} \equiv \mathrm{diag}\{\sigma_1, \ldots, \sigma_n\}$, of singular values.

A naïve straightforward ordering with permutation of the matrix columns is exemplified below (with the same identifiers as in Example 3-4-5.1; a few other identifiers have obvious meaning). Due to quadratic complexity, such ordering could be useful only for relatively small values of n or for already almost ordered arrays W:

```
for ( i = 0; i < n - 1; i++) {
       Wmax   = W[ i ];
       imax      = i;
       for ( k = i + 1; k < n; k++) {
             if ( W[ k ] > Wmax ) {
                 Wmax   = W[ k ];
                 imax     = k;
             }
       }
       if ( imax > i ) {
           for ( j = 0; j < m; j++ ) {
                  temp  = U[ j ][ imax ];
                  U[ j ][ imax ] = U[ j ][ i ];
                  U[ j ][ i ] = temp;
           }
           temp = W[ imax ];
           W[ imax ] = W[ i ];
           W[ i ]  = temp;
           for ( j = 0; j < n; j++) {
                  temp = V[ j ][ imax ];
                  V[ j ][ imax ] = V[ j ][ i ];
                  V[ j ][ i ] = temp;
           }
     }
}
```

Given the ordered SVD, the approximation with the k singular values and vectors is obtained as follows (here, the identifier `Ak` indicates the $m \times n$ approximate matrix):

```
Ak = calloc( m, sizeof( double* ) );
for( i = 0; i < m; i++ ) {
     Ak[ i ] = calloc( n, sizeof( double ) );
     for ( j = 0; j < n; j++ ) {
  Ak[ i ][ j ] = 0.;
     }

}
for ( i = 0; i < m; i++ ) {
      for( j = 0; j < n; j++ ) {
            for( t = 0; t < k; t++ ) {
                    Ak[ i ][ j ] += W[ k ]  * U[ i ] [ k ]  * V[ j ][ k ];
            }
      }
}
```

A C-program implementing this approximation for $k = 1, \ldots, n$ gives:

```
Ordered singular values:  6.3801470      2.4772635       1.0755882
Approximate matrix A: k = 1
   Row 1:   2.547  1.863  0.851
   Row 2:   2.103  1.538  0.702
   Row 3:   2.025  1.481  0.676
   Row 4:   3.116  2.279  1.041
Approximate matrix A: k = 2
   Row 1:   1.932  3.038  0.118
   Row 2:   1.803  2.112  0.345
   Row 3:   2.715  0.162  1.498
   Row 4:   3.373  1.788  1.347
Approximate matrix A: k = 3
   Row 1:   2.000  3.000  0.000
   Row 2:   2.000  2.000  0.000
   Row 3:   3.000 -0.000  1.000
   Row 4:   3.000  2.000  2.000
```

Question 3-4-5.2: Approximate the matrix from Question 3-4-5.1 by using $k = 1$ and $k = 2$ singular values and the relevant singular vectors.

Example 3-4-5.3: Eigen-values λ_i and eigen-vectors $\mathbf{e}_i$ for a symmetric $n \times n$, i.e. square matrix $\mathbf{A}$ can be found easily by using an available SVD software, e.g. our C-subroutine `svd.c`.

The transposition of a symmetric matrix results in the same matrix, $\mathbf{A}^\mathsf{T} = \mathbf{A}$, e.g.

$$\begin{bmatrix} 7 & 3 & -5 \\ 3 & 4 & 2 \\ -5 & 2 & 1 \end{bmatrix}^\mathsf{T} = \begin{bmatrix} 7 & 3 & -5 \\ 3 & 4 & 2 \\ -5 & 2 & 1 \end{bmatrix}$$

so that in this case $\mathbf{A}\mathbf{A}^\mathsf{T} = \mathbf{A}^\mathsf{T}\mathbf{A} = \mathbf{A}^2$.

Therefore, the SVD $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T}$ produces two equal $n \times n$ matrices $\mathbf{U} = \mathbf{V}$ with columns $\mathbf{u}_i = \mathbf{v}_i$, being the eigen-vectors of the squared matrix $\mathbf{A}^2$, and the $n \times n$

diagonal matrix $\mathbf{D}$ of the corresponding eigen-values σ_i of the latter. Because the squared matrix $\mathbf{A}^2$ has the same eigen-vectors as the matrix $\mathbf{A}$ and its eigen-values are the squared eigen-values of $\mathbf{A}$, i.e. $\sigma_i = \lambda_i^2$, then the goal eigen-vectors $\mathbf{e}_i = \mathbf{u}_i$ and the goal eigen-values $\lambda_i = \sqrt{\sigma_i}$ for $i = 1, \ldots, n$.

Question 3-4-5.3: Show that a square $n \times n$ symmetric matrix $\mathbf{A}$ can be represented as $\mathbf{S\Lambda S}^\mathsf{T}$ where $\mathbf{S}$ is the $n \times n$ orthonormal matrix with the eigenvectors $\mathbf{e}_i$ as columns and $\mathbf{\Lambda}$ is the $n \times n$ diagonal matrix of the corresponding eigenvalues:

$$\mathbf{A} = \underbrace{[\mathbf{e}_1 \ \ldots \ \mathbf{e}_n]}_{\mathbf{S}} \underbrace{\mathrm{diag}\,\{\lambda_1, \ldots, \lambda_n\}}_{\mathbf{\Lambda}} \underbrace{\begin{bmatrix} \mathbf{e}_1^\mathsf{T} \\ \ldots \\ \mathbf{e}_n^\mathsf{T} \end{bmatrix}}_{\mathbf{S}^\mathsf{T}} = \sum_{i=1}^{n} \lambda_i \mathbf{e}_i \mathbf{e}_i^\mathsf{T}$$

Question 3-4-5.4: Show that singular values σ_i; $i = 1, \ldots, n$, for an $m \times n$ matrix $\mathbf{A}$; $m \geq n$, are square roots of the corresponding eigenvalues λ_i of the $n \times n$ matrix $\mathbf{A}^\mathsf{T}\mathbf{A}$ and $m \times m$ matrix $\mathbf{A}\mathbf{A}^\mathsf{T}$.

Answers

To Question 3-4-5.1: Printout of the program in Example 3-4-5.1:

```
Output index = 0
6 x 2 matrix A
  Row  0:   1.000  2.000
  Row  1:   2.000  3.000
  Row  2:   3.000  4.000
  Row  3:   4.000  5.000
  Row  4:   5.000  6.000
  Row  5:   6.000  7.000
6 x 2 column-orthonormal matrix U
  Row  0:  -0.709  0.144
  Row  1:  -0.489  0.237
  Row  2:  -0.269  0.330
  Row  3:  -0.048  0.423
  Row  4:   0.172  0.515
  Row  5:   0.392  0.608
2 x 2 diag matrix D of singular values
  Diag values:  0.6763368 15.1506623
2 x 2 orthonormal matrix V:
  Row  0:   0.778  0.629
  Row  1:  -0.629  0.778
```

To Question 3-4-5.2: Printout of the C-program implementing the approximation, detailed in Example 3-4-5.2.

```
Ordered singular values:  15.1506623      0.6763368
```

```
Approximate matrix A: k = 1
   Row 1:   1.373  1.698
   Row 2:   2.257  2.792
   Row 3:   3.141  3.886
   Row 4:   4.025  4.979
   Row 5:   4.910  6.073
   Row 6:   5.794  7.167

Approximate matrix A: k = 2
   Row 1:   1.000  2.000
   Row 2:   2.000  3.000
   Row 3:   3.000  4.000
   Row 4:   4.000  5.000
   Row 5:   5.000  6.000
   Row 6:   6.000  7.000
```

To Question 3-4-5.3: By definition, $\mathbf{A}\mathbf{e}_i = \lambda_i \mathbf{e}_i$, so that $\mathbf{AS} = [\lambda_1 \mathbf{e}_1 \dots \lambda_n \mathbf{e}_n]$. Due to the orthonormality of the eigen-vectors, it holds that $\mathbf{S}^\mathsf{T}\mathbf{S} = \mathbf{S}\mathbf{S}^\mathsf{T} = \mathbf{I}_n$, i.e. the identity matrix, and

$$\mathbf{S}^\mathsf{T}\mathbf{A}\mathbf{S} = \begin{bmatrix} \mathbf{e}_1^\mathsf{T} \\ \dots \\ \mathbf{e}_n^\mathsf{T} \end{bmatrix} [\lambda_1 \mathbf{e}_1 \dots \lambda_n \mathbf{e}_n] = \underbrace{\begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}}_{\mathbf{\Lambda}}$$

Therefore, by left-muliplying and right-multiplying the above equation with $\mathbf{S}$ and $\mathbf{S}^\mathsf{T}$, respectively, we obtain that $\mathbf{A} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^\mathsf{T}$.

To Question 3-4-5.4: Orthonormality of the columns in the matrices $\mathbf{U} = [\mathbf{u}_1 \dots \mathbf{u}_n]$ and $\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_n]$ yields the basic properties of the SVD;

$$\mathbf{A}\mathbf{v}_i = \sigma_i \mathbf{u}_i \text{ and } \mathbf{A}^\mathsf{T}\mathbf{u}_i = \sigma_i \mathbf{v}_i$$

Therefore,

$$\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{v}_i = \sigma_i \mathbf{A}^\mathsf{T}\mathbf{u}_i = \sigma_i^2 \mathbf{v}_i \text{ and } \mathbf{A}\mathbf{A}^\mathsf{T}\mathbf{u}_i = \sigma_i \mathbf{A}\mathbf{v}_i = \sigma_i^2 \mathbf{u}_i$$

so that the symmetric square matrices $\mathbf{A}^\mathsf{T}\mathbf{A}$ and $\mathbf{A}\mathbf{A}^\mathsf{T}$ have the same n top eigenvalues λ_i; $i = 1, \dots, n$, which are equal to the squared singular values: $\lambda_i = \sigma_i^2$, that is, $\sigma_i = \sqrt{\lambda_i}$; $i = 1, \dots, n$.