Lecture 8

Banded, LU, Cholesky, SVD

T. Gambill

Department of Computer Science
University of Illinois at Urbana-Champaign

February 16, 2010

More Algorithms for Special Systems

- tridiagonal systems
- banded systems
- LU decomposition
- Cholesky factorization

Tridiagonal

A tridiagonal matrix A

$$\begin{bmatrix} d_1 & c_1 & & & & & & \\ a_1 & d_2 & c_2 & & & & & \\ & a_2 & d_3 & c_3 & & & & \\ & & \cdots & \cdots & \cdots & & & \\ & & & a_{i-1} & d_i & c_i & & \\ & & & & \cdots & \cdots & \cdots & \\ & & & & \cdots & \cdots & \cdots & \\ & & & & & & a_{n-1} & d_n \end{bmatrix}$$

- storage is saved by not saving zeros
- only $n + 2(n-1) = 3n - 2$ places are needed to store the matrix versus n^2 for the whole system
- can operations be saved? yes!

Tridiagonal

$$\begin{bmatrix} d_1 & c_1 & & & & & & \\ a_1 & d_2 & c_2 & & & & & \\ & a_2 & d_3 & c_3 & & & & \\ & & \dots & \dots & \dots & & & \\ & & & a_{i-1} & d_i & c_i & & \\ & & & & \dots & \dots & \dots & \\ & & & & \dots & \dots & \dots & \\ & & & & & & a_{n-1} & d_n \end{bmatrix}$$

Start forward elimination (without any special pivoting)

1. subtract a_1/d_1 times row 1 from row 2
2. this eliminates a_1, changes d_2 and does not touch c_2
3. continuing:

$$\tilde{d}_i = d_i - \left(\frac{a_{i-1}}{\tilde{d}_{i-1}} c_{i-1}\right) \quad \tilde{b}_i = b_i - \left(\frac{a_{i-1}}{\tilde{d}_{i-1}} \tilde{b}_{i-1}\right)$$

for $i = 2 \dots n$

Tridiagonal

$$\begin{bmatrix} \tilde{d}_1 & c_1 & & & & & & \\ & \tilde{d}_2 & c_2 & & & & & \\ & & \tilde{d}_3 & c_3 & & & & \\ & & & \dots & \dots & & & \\ & & & & \tilde{d}_i & c_i & & \\ & & & & & \dots & \dots & \\ & & & & & & \dots & \dots \\ & & & & & & & \tilde{d}_n \end{bmatrix}$$

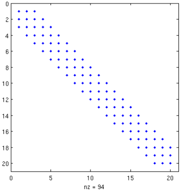
This leaves an upper triangular (2-band). With back substitution:

1. $x_n = \tilde{b}_n / \tilde{d}_n$
2. $x_{n-1} = (1/\tilde{d}_{n-1})(\tilde{b}_{n-1} - c_{n-1} x_n)$
3. $x_i = (1/\tilde{d}_i)(\tilde{b}_i - c_i x_{i+1})$

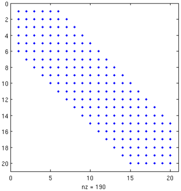
Tridiagonal Algorithm

```
input: n, a, d, c, b
for i = 2 to n
    xmult = a_{i-1} / d_{i-1}
    d_i = d_i - xmult · c_{i-1}
    b_i = b_i - xmult · b_{i-1}
end
x_n = b_n / d_n
for i = n-1 down to 1
    x_i = (b_i - c_i x_{i+1}) / d_i
end
```

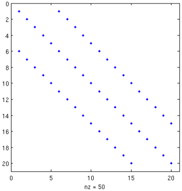
m-band



$m = 5$



$m = 11$



$m = 11$

- the m correspond to the total width of the non-zeros
- after a few passes of GE *fill-in* with occur within the band
- so an empty band costs (about) the same as a non-empty band
- one fix: reordering (e.g. Cuthill-McKee)
- generally GE will cost $\mathcal{O}\left(\left(\frac{m-1}{2}\right)^2 n\right)$ for m-band systems

Motivation: Symmetric Matrix

- A is symmetric, if $A = A^T$
- If $A = LU$ and A is symmetric, then could $L = U^T$?
- If so, this could save 50% of the computation of LU by only calculating L
- Save 50% of the FLOPS!
- This is achievable: LDL^T and Cholesky (LL^T) factorization

Factorization Methods

Factorizations are the common approach to solving $Ax = b$: simply organized Gaussian elimination.

Goals for today:

- LU factorization
- Cholesky factorization
- Use of the backslash operator

LU Factorization

Find L and U such that

$$A = LU$$

and L is lower triangular, and U is upper triangular.

$$L = \begin{bmatrix} 1 & 0 & \cdots & & 0 \\ \ell_{2,1} & 1 & 0 & & 0 \\ \ell_{3,1} & \ell_{3,2} & 1 & & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ \ell_{n,1} & \ell_{n,2} & \cdots & \ell_{n-1,n} & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \cdots & u_{1,n} \\ 0 & u_{2,2} & u_{2,3} & \cdots & u_{2,n} \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & & & u_{n-1,n} \\ 0 & 0 & & & u_{n,n} \end{bmatrix}$$

Since L and U are triangular, it is easy to apply their inverses.

Why?

- Since L and U are triangular, it is easy, $\mathcal{O}(n^2)$, to apply their inverses
- Decompose once, solve k right-hand sides quickly:
  - $\mathcal{O}(kn^3)$ with GE
  - $\mathcal{O}(n^3 + kn^2)$ with LU
- Given $A = LU$ you can compute A^{-1}, $det(A)$, $rank(A)$, $ker(A)$, etc...

LU Factorization

Since L and U are triangular, it is easy to apply their inverses.
Consider the solution to $Ax = b$.

$$A = LU \Longrightarrow (LU)x = b$$

Regroup since matrix multiplication is associative

$$L(Ux) = b$$

Let $Ux = y$, then

$$Ly = b$$

Since L is triangular it is easy (without Gaussian elimination) to compute

$$y = L^{-1}b$$

This expression should be interpreted as "Solve $Ly = b$ with forward substitution."

LU Factorization

Now, since y is known, solve for x

$$x = U^{-1}y$$

which is interpreted as "Solve $Ux = y$ with backward substitution."

LU Factorization

Listing 1: LU Solve

```
Factor A into L and U
Solve Ly = b for y            use forward substitution
Solve Ux = y for x            use backward substitution
```

LU Factorization

- If we have $Ax = b$ and perform GE we end up with

$$A = \begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix} \Rightarrow \begin{bmatrix} x' & x' & x' & x' \\ 0 & x' & x' & x' \\ 0 & 0 & x' & x' \\ 0 & 0 & 0 & x' \end{bmatrix}$$

- Remember from Lecture 6, that naive Gaussian Elimination can be done by matrix multiplication

$$MAx = Mb$$

$$Ux = Mb$$

- MA is upper triangular and called U
- M is the elimination matrix

LU Factorization

As an example take one column step of GE, A becomes

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} \to \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{bmatrix}$$

using the elimination matrix

$$M_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -\frac{1}{2} & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

So we have performed

$$M_1 A x = M_1 b$$

LU Factorization

From Lecture 6

- Inverting M_i is easy: just flip the sign of the lower triangular entries

$$M_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -\frac{1}{2} & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad \Rightarrow \quad M_1^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

- M_i^{-1} is just the multipliers used in Gaussian Elimination!
- $M_i^{-1}M_j^{-1}$ is still lower triangular, for $i < j$, and is the union of the columns
- $M_1^{-1}M_2^{-1}\ldots M_j^{-1}$ is lower triangular, with the lower triangle the multipliers from Gaussian Elimination

LU Factorization

- Zeroing each column yields another elimination matrix operation:

$$M_3 M_2 M_1 A x = M_3 M_2 M_1 b$$

- $M = M_3 M_2 M_1$. Thus
- $L = M_1^{-1} M_2^{-1} M_3^{-1}$ is lower triangular

$$\begin{aligned} MA &= U \\ M_3 M_2 M_1 A &= U \\ A &= M_1^{-1} M_2^{-1} M_3^{-1} U \\ A &= LU \end{aligned}$$

LU (forward elimination) Algorithm

Listing 2: LU

```
given A

for k = 1...n-1
  for i = k+1...n
    xmult = a_ik/a_kk
    a_ik = xmult
    for j = k+1...n
      a_ij = a_ij - (xmult)a_kj
    end
  end
end
```

- U is stored in the upper triangular portion of A
- L (without the diagonal) is stored in the lower triangular

Doolittle Factorization (LU)

Listing 3: Doolittle

```
given A
output L, U

for k = 1...n
   ℓ_kk = 1
   for j = k...n
         u_kj = a_kj − sum_{i=1}^{k−1} ℓ_ki u_ij
   end
   for j = k+1...n
         ℓ_jk = (a_jk − sum_{i=1}^{k−1} ℓ_ji u_ik) / u_kk
   end
end
```

- Mathematically the same as previous LU
- Difference is we now explicitly form L and U

What About Pivoting?

- Pivoting (that is row exchanges) can be expressed in terms of matrix multiplication
- Do pivoting during elimination, but track row exchanges in order to express pivoting with matrix P
- Let P be all zeros
  - Place a 1 in column j of row 1 to exchange row 1 and row j
  - If no row exchanged needed, place a 1 in column 1 of row 1
  - *Repeat for all rows of* P
- P is a permutation matrix
- Now using pivoting,

$$LU = PA$$

Python LU

Like GE, LU needs pivoting. With pivoting the LU factorization always exists, even if A is singular. With pivoting, we get

$$LU = PA$$

```
>>> import numpy as np
>>> import scipy.linalg
>>> A = scipy.rand(4,4)
>>> b = scipy.rand(4,1)
>>> A
array([[ 0.50742833,  0.29832637,  0.87906078,  0.11219151],
       [ 0.58297164,  0.31504083,  0.33923234,  0.294866  ],
       [ 0.45099647,  0.34853809,  0.55473901,  0.52446345],
       [ 0.07995563,  0.31020355,  0.88319642,  0.9922531 ]])
>>> b
array([[ 0.04539488],
       [ 0.25711279],
       [ 0.55651992],
       [ 0.24906525]])
>>> LU = scipy.linalg.lu_factor(A)
>>> LU
>>> x = scipy.linalg.lu_solve(LU,b)
(array([[ 0.87906078,  0.57723918,  0.12762657,  0.33936945],
       [ 0.33923234,  0.38715344,  0.64979646,  0.51637339],
       [ 0.55473901,  0.13077937,  0.36868404,  0.25155858],
       [ 0.88319642, -0.42985995,  1.15885524, -0.05907806]]),
           array([2, 2, 3, 3], dtype=int32))
>>> x
array([[ -5.75628116],
       [ 15.83236907],
       [ -1.64503985],
       [ -2.77051444]])
```

LU Tutorial Module

http://www.cse.illinois.edu/iem/linear_equations/gaussian_elimination/

Use SYMMETRY ! YRTEMMYS esU

- Suppose

$$A = LU, \text{ and } A = A^T$$

- Then

$$LU = A = A^T = (LU)^T = U^T L^T$$

- Thus

$$U = L^{-1} U^T L^T$$

and

$$U (L^T)^{-1} = L^{-1} U^T = D$$

- We can conclude that

$$U = D L^T$$

and

$$A = LU = LDL^T$$

Symmetric Doolittle Factorization (LDL^T)

Listing 4: Symm Doolittle

```
given A
output L, D

for k = 1...n
   ℓ_kk = 1

   d_k = a_kk − Σ_{ν=1}^{k−1} d_ν ℓ_kν^2

   for j = k+1...n
      ℓ_kj = 0
      ℓ_jk = ( a_jk − Σ_{ν=1}^{k−1} ℓ_jν d_ν ℓ_kν ) / d_k
   end
end
```

- Special form of the Doolittle factorization

LL^T: Cholesky Factorization

- A must be symmetric and positive definite (SPD)
- A is Positive Definite (PD) if for all $x \neq 0$ the following holds

$$x^T A x > 0$$

- Positive definite gives us an all positive D in $A = LDL^T$
  - Let $x = (L^T)^{-1} e_i$, where e_i is the i-th column of I
- L becomes $LD^{1/2}$ (**NOT** necessarily unit lower triangular as in LU factorization!
- $A = LL^T$, i.e. $L = U^T$
  - Half as many flops as LU!
  - Only calculate L not U

Cholesky 2x2 example

$$\begin{bmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} * \begin{bmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{bmatrix}$$

implies that

$$l_{11} = \sqrt{a_{11}}, \;\; l_{21} = a_{21}/l_{11}, \;\; l_{22} = \sqrt{a_{22} - l_{21}^2}$$

Cholesky Factorization

Listing 5: Cholesky

given A
output L

for $k = 1 \ldots n$
$$\ell_{kk} = \left(a_{kk} - \sum_{i=1}^{k-1} \ell_{ki}^2\right)^{1/2}$$

  for $j = k+1 \ldots n$
$$\ell_{jk} = \left(a_{jk} - \sum_{i=1}^{k-1} \ell_{ji}\ell_{ki}\right) / \ell_{kk}$$
  end
end

Why SPD?

In general, SPD gives us

- non singular
  - If $x^T Ax > 0$, for all nonzero x
  - Then $Ax \neq 0$ for all nonzero x
  - Hence, the columns of A are linearly independent
- No pivoting
  - From algorithm, can derive that $|l_{kj}| \leqslant \sqrt{a_{kk}}$
  - Elements of L do not grow with respect to A
  - *For short proof see book*
- Cholesky faster than LU
  - No pivoting
  - Only calculate L, not U

Why SPD?

A matrix is Positive Definite (PD) if for all $x \neq 0$ the following holds

$$x^T A x > 0$$

- For SPD matrices, use the Cholesky factorization, $A = LL^T$
- Cholesky Factorization
  - Requires no pivoting
  - Requires one half as many flops as LU factorization, that is only calculate L not L and U.
  - Cholesky will be more than *twice* as fast as LU because no pivoting means no data movement
- Use Python linalg.cholesky(A) or MATLAB's built-in `chol(A)` function for routine work.
- If A is positive definite then so is A^{-1} and A^n for $n = 2, 3, 4, \ldots$.
- If A and B are positive definite then so is $A + B$.

Motivation Revisited

Multiple right hand sides

- Solve $Ax = b$ for k different b vectors
- Using LU factorization, the cost is $\mathcal{O}(n^3) + \mathcal{O}(kn^2)$
- Using Gaussian Elimination, the cost is $\mathcal{O}(kn^3)$

If A is symmetric

- Save 50% of the flops with LDL^T factorization
- Save 50% of the flops and many memory operations with Cholesky (LL^T) factorization

SVD: motivation

SVD uses in practice:

1. Search Technology: find closely related documents or images in a database
2. Clustering: aggregate documents or images into similar groups
3. Compression: efficient image storage
4. Principal axis: find the main axis of a solid (engineering/graphics)
5. Summaries: Given a textual document, ascertain the most representative tags
6. Graphs: partition graphs into subgraphs (graphics, analysis)

A geometric view of $y = Ax$

Example

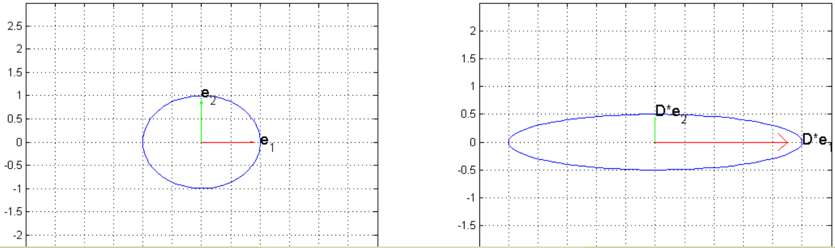
Given the matrix

$$D = \begin{bmatrix} 10 & 0 \\ 0 & 0.5 \end{bmatrix}$$

then

$$y = Dx$$

maps the unit circle onto an ellipse.



A geometric view of $y = Ax$

Example

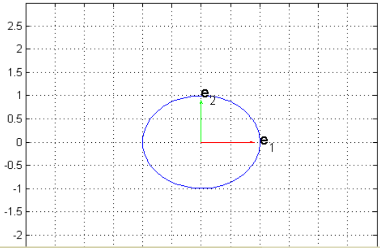
Given the matrix

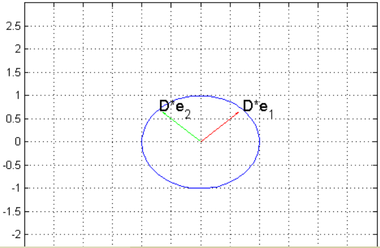
$$Q = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

then

$$y = Qx$$

maps the unit circle onto the unit circle.





A geometric view of $y = Ax$

Diagonal Matrices

The matrix D in a previous example is called a **diagonal** matrix.

Orthogonal Matrices

The matrix Q in a previous example is called a **orthogonal** matrix since $QQ^T = I$.

Orthogonality, Orthonormality

Definition

If u and v are $n \times 1$ vectors then $< u, v >= u^T * v$ is called the standard inner product of u with v. We have $< u, u >= ||u||_2^2$.

From calculus, we know that the angle θ between two vectors can be computed from the following,

Angle between vectors

$$< u, v >= ||u||_2 ||v||_2 \cos(\theta)$$

Definition

Vectors u and v are said to be orthogonal (perpendicular) if $< u, v >= 0$.

Definition

Vectors u and v are said to be orthonormal if $< u, v >= 0$ and $||u||_2 = ||v||_2 = 1$.

Properties of A^T

Theorem

If A is an $n \times n$ real valued matrix then for any $n \times 1$ vectors u and v,

$$< Au, v > = < u, A^T v >$$

This follows from the definition of the standard inner product, since

$$< Au, v > = (Au)^T v = (u^T A^T) v = u^T (A^T v) = < u, A^T v >$$

Properties of Orthogonal Matrices

Definition

An $n \times n$ matrix Q is called orthogonal if $QQ^T = Q^TQ = I$.

- $Q^{-1} = Q^T$
- the columns of Q are orthonormal
- the rows of Q are orthonormal
- $||Qx||_2 = ||x||_2$ for all x

Another example of an orthogonal matrix

Example

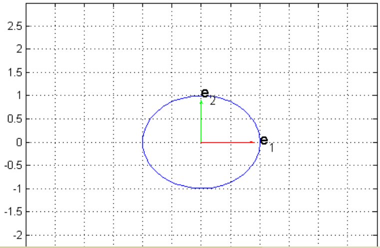
Given the matrix

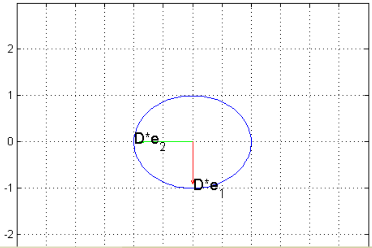
$$Q = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

then

$$y = Qx$$

maps the unit circle onto the unit circle.





An arbitrary matrix

Example

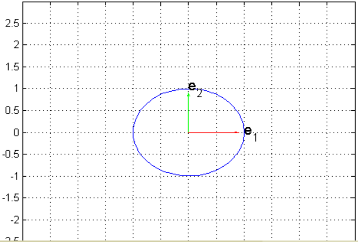
Given the matrix

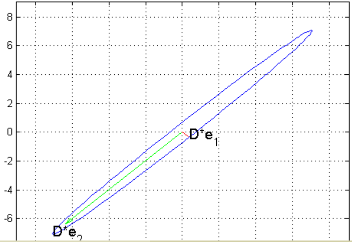
$$A = \begin{bmatrix} \sqrt(2)/4 & -5\sqrt(2) \\ -\sqrt(2)/4 & -5\sqrt(2) \end{bmatrix}$$

then

$$y = Ax$$

maps the unit circle onto an ellipse. Is this a coincidence?





SVD: Singular Value Decomposition

SVD takes any $m \times n$ matrix A and factors it:

$$A = USV^T$$

where U $(m \times m)$ and V $(n \times n)$ are orthogonal and S $(m \times n)$ is diagonal.
S is made up of "singular values":

$$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_r \geqslant \sigma_{r+1} = \cdots = \sigma_p = 0$$

Here, $r = rank(A)$ and $p = min(m, n)$. For $m > n$ the factorization appears as,

$$A = \begin{bmatrix} \vdots & \vdots & \vdots \\ u_1 & \dots & u_m \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \\ 0 & & & \dots & & 0 \\ 0 & & & \dots & & 0 \end{bmatrix} \begin{bmatrix} \dots & v_1^T & \dots \\ \dots & \vdots & \dots \\ \dots & v_n^T & \dots \end{bmatrix}$$

SVD in Python

From our previous example,

$$A = \begin{bmatrix} \sqrt(2)/4 & -5\sqrt(2) \\ -\sqrt(2)/4 & -5\sqrt(2) \end{bmatrix}$$

We use the Python "svd" function,

```
>>> import numpy.linalg
>>> import numpy as np
>>> A = np.array([[np.sqrt(2.)/4., -5.*np.sqrt(2.)],[-np.sqrt
    (2.)/4.,-5*np.sqrt(2)]])
>>> A
array([[ 0.35355339, -7.07106781],
       [-0.35355339, -7.07106781]])
>>> U, S, V = numpy.linalg.svd(A)
>>> U
array([[ 0.70710678, -0.70710678],
       [ 0.70710678,  0.70710678]])
>>> S
array([ 10. ,   0.5])
>>> V
array([[-0., -1.],
       [-1., -0.]])
```

SVD Application: Spheres map to Ellipsoids

If A is a non-singular $n \times n$ matrix then A maps circles(spheres) into ellipses(ellipsoids) and further,

$$A = USV^T$$

$$AV = US$$

$$A \left[v_1 | v_2 | \dots | v_n \right] = \left[u_1 | u_2 | \dots | u_n \right] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix}$$

$$\begin{aligned} Av_1 &= \sigma_1 u_1 \\ Av_2 &= \sigma_2 u_2 \\ &\vdots \\ Av_n &= \sigma_n u_n \end{aligned}$$

and the singular values σ_i and left singular vectors u_i are the length and directions respectively of the principal axes of the ellipsoid.

SVD Application: computing $||A||_2$, $\kappa_2(A)$

The singular values can be used to calculate the 2-norm of a matrix and the matrix condition number $\kappa(A)$.

$$||A||_2 = \sigma_{max} = \sigma_1$$

$$||A||_2 \; ||A^{-1}||_2 = \kappa(A) = \frac{\sigma_{max}}{\sigma_{min}} = \frac{\sigma_1}{\sigma_n}$$

How is SVD performed?

We want to factorize A into U, S, and V^T. First step: find V. Consider

$$A = USV^T$$

and multiply by A^T

$$A^T A = (USV^T)^T (USV^T) = VS^T U^T USV^T$$

Since U is orthogonal

$$A^T A = VS^2_{n \times n} V^T$$

This is called a similarity transformation.

Definition

Matrices A and B are similar if there is an invertible matrix Q such that

$$Q^{-1} AQ = B$$

Theorem

Similar matrices have the same eigenvalues.

Proof

Eigenvalues

Remember that a number λ (which may be a complex number) is an eigenvalue of a matrix A if there is a non-zero vector v such that,

$$(A - \lambda I)v = 0$$

$$\begin{aligned} Bv &= \lambda v \\ Q^{-1}AQv &= \lambda v \\ AQv &= \lambda Qv \\ Aw &= \lambda w. \end{aligned}$$

Further, if v is an eigenvector of B, Qv is an eigenvector of A.

So far...

Need $A = USV^T$

Look for V such that $A^TA = VS^2_{n\times n}V^T$. Here S^2 is diagonal.

If A^TA and S^2 are similar, then they have the same eigenvalues. So the diagonal matrix S^2 is just the eigenvalues of A^TA and V is the matrix of eigenvectors. To see the latter, note that since S^2 is diagonal, the eigenvectors

are e_i, and so we can write,

$$S^2 e_i = \sigma_i^2 e_i$$

and since,

$$V^T v_i = e_i$$

thus

$$VS^2V^T v_i = VS^2 e_i = V\sigma_i^2 e_i = \sigma_i^2 v_i$$

Similarly...

Now consider

$$A = USV^T$$

and multiply by A^T from the right

$$AA^T = (USV^T)(USV^T)^T = USV^TVS^TU^T$$

Since V is orthogonal

$$AA^T = US^2_{m\times m}U^T$$

Now U is the matrix of eigenvectors of AA^T.

In the end...

We get (for $m > n$)

$$A = \begin{bmatrix} \vdots & \vdots & \vdots \\ u_1 & \dots & u_m \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \\ 0 & & & \dots & & 0 \\ 0 & & & \dots & & 0 \end{bmatrix} \begin{bmatrix} \dots & v_1^T & \dots \\ \dots & \vdots & \dots \\ \dots & v_n^T & \dots \end{bmatrix}$$

Example

Decompose

$$A = \begin{bmatrix} 2 & -2 \\ 1 & 1 \end{bmatrix}$$

First construct $A^T A$:

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

Eigenvalues: $\lambda_1 = 8$ and $\lambda_2 = 2$. So

$$S^2 = \begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix} \quad \Rightarrow \quad S = \begin{bmatrix} 2\sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$$

Example

Now find V^T and U. The columns of V are the eigenvectors of A^TA.

- $\lambda_1 = 8$: $(A^TA - \lambda_1 I)v_1 = 0$

$$\Rightarrow \begin{bmatrix} -3 & -3 \\ -3 & -3 \end{bmatrix} v_1 = 0 \quad \Rightarrow \quad \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} v_1 = 0 \quad \Rightarrow \quad v_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

normalized,

$$v_1 = \begin{bmatrix} -\sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix}$$

- $\lambda_2 = 2$: $(A^TA - \lambda_2 I)v_2 = 0$

$$\Rightarrow \begin{bmatrix} 3 & -3 \\ -3 & 3 \end{bmatrix} v_2 = 0 \quad \Rightarrow \quad \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix} v_2 = 0 \quad \Rightarrow \quad v_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

normalized,

$$v_2 = \begin{bmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix}$$

- Finally:

$$V = \begin{bmatrix} -\sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$$

Example

Now find U. The columns of U are the eigenvectors of AA^T.

- $\lambda_1 = 8$: $(AA^T - \lambda_1 I)u_1 = 0$

$$\Rightarrow \begin{bmatrix} 0 & 0 \\ 0 & -6 \end{bmatrix} u_1 = 0 \quad \Rightarrow \quad \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} u_1 = 0 \quad \Rightarrow \quad u_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

- $\lambda_2 = 2$: $(AA^T - \lambda_2 I)u_2 = 0$

$$\Rightarrow \begin{bmatrix} 6 & 0 \\ 0 & 0 \end{bmatrix} u_2 = 0 \quad \Rightarrow \quad \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} u_2 = 0 \quad \Rightarrow \quad u_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- Finally:

$$U = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Together:

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2\sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} -\sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$$

SVD is Not Unique!

The normalized eigenvectors of A^TA and AA^T are not unique. We have the following valid combinations:

$$\pm v_1 = \pm \begin{bmatrix} -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}, \pm v_2 = \pm \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}, \pm u_1 = \pm \begin{bmatrix} -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}, \pm u_2 = \pm \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

However the only combinations of $\pm u_1, \pm v_1$ and $\pm u_2, \pm v_2$ that are valid are those that satisfy,

$$Av_n = \sigma_n u_n$$

which are for this specific problem,

$$(+v_1, +u_1), (-v_1, -u_1), (+v_2, +u_2), (-v_2, -u_2)$$

and this gives rise to a variety of value U and V matrix pairs,

$$\begin{aligned} U &= [+u_1 \mid +u_2], \quad V = [+v_1 \mid +v_2] \\ U &= [+u_1 \mid -u_2], \quad V = [+v_1 \mid -v_2] \\ U &= [-u_1 \mid +u_2], \quad V = [-v_1 \mid +v_2] \\ U &= [-u_1 \mid -u_2], \quad V = [-v_1 \mid -v_2] \end{aligned}$$

SVD Application: Data Compression

How can we actually *use* $A = USV^T$? We can use this to represent A with far fewer entries...

Notice what $A = USV^T$ looks like:

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T + 0 u_{r+1} v_{r+1}^T + \cdots + 0 u_p v_p^T$$

This is easily truncated to

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T$$

or even more terms can be truncated for small σ_i (see MP3).
What are the savings?

- A takes $m \times n$ storage
- using k terms of U and V takes $k(1 + m + n)$ storage