Math 170A Lecture Notes (Professor: Lei Huang)

Isabelle Mills

March 20, 2024

Week 1 Notes (1/8 - 1/12/2024)

For this class, we shall define the number of flops an algorithm takes as the number of individual $+$, $-$, $\times$, $/$, and $\sqrt{\ }$ operations on real numbers used in the algorithm.

> For example: taking the inner product of two vectors $\vec{u} = (u_1, u_2, \ldots, u_n)$ and $\vec{v} = (v_1, v_2, \ldots, v_n)$ requires n multiplications and $(n-1)$ additions. So we'll say it has a flops count of $2n - 1$.

Technically, the word "flop" stands for floating (point) operation. Based on that knowledge, hopefully it is easier to guess what is and is not a flop. For instance, observe the code written below for taking an inner product of two n-vectors.

```
1   P = 0
2   for i = 1:n
3      P = P + v(i) * w(i)
4   end
```

Neither incrementing `i` nor initializing any other variables are counted towards the flop number. Because the code does `n` additions and multiplications between floating point numbers, we say this function has $2n$ flops.

Here is how we formally define Big-$\mathcal{O}$ Notation:

For a sequence a_n, we define $a_n = \mathcal{O}(b_n)$ if there exists real constants $C, N \geq 0$ such that for $n \geq N$, $a_n \leq Cb_n$.

Example problem:

```
1       function x = lowertriangsolve(L, b)
2          N = size(L);
3          n=size(b,1);
4          x=b;
5
6          for i=1:N(1):
7             for j=1:i-1
8                x(i) = x(i) - L(i,j)*x(j);
9             end
10
11            if L(i, j) == 0
12               error('matrix is singular')
13            end
14            x(i) = x(i)/L(i,i);
15         end
16      end
```

Inside the main for loop (lines 6-15):
Line 8 has $2(\mathtt{i}-1)$ flops.
Line 14 has an additional flop.

Thus, the total number of flops is:

$$\sum_{\mathtt{i}=1}^{n}(2\mathtt{i}-1) = 2\left(\sum_{\mathtt{i}=1}^{n}\mathtt{i}\right) - n$$

This then simplifies to:

$$2\left(\frac{n(n+1)}{2}\right) - n = 1n^2$$

So `x` has $\mathcal{O}(n^2)$ flops as `n` goes to infinity.

Lecture 4: 1/17/2024

Row operations used in Gaussian elemination can be represented via matrix multiplication as demonstrated below.

Action	Matrix representation of the operation
$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} \xrightarrow{R_2 \leftrightarrow R_3} \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} \xrightarrow{\frac{2}{3}R_2 \to R_2} \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ \frac{2}{3}a_{2,1} & \frac{2}{3}a_{2,2} & \frac{2}{3}a_{2,3} & \frac{2}{3}a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{2}{3} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} \downarrow R_3 + \frac{2}{3}R_2 \to R_3 \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1}+\frac{2}{3}a_{2,1} & a_{3,2}+\frac{2}{3}a_{2,2} & a_{3,3}+\frac{2}{3}a_{2,3} & a_{3,4}+\frac{2}{3}a_{2,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \frac{2}{3} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

The importance of representing elementary row operations as matrices is that we can multiply these representations together to compose rows operations. Thus, these representations are central to many matrix decompositions.

For example, we can represent turning a matrix into row echelon form as follows:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{2}{9} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & 5 \\ 6 & 3 & 4 \\ 4 & 6 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 5 \\ 0 & -9 & -11 \\ 0 & 0 & \frac{-50}{9} \end{bmatrix}$$

Here are some more observations about row operation matrices:

- Row scaling operations are represented by diagonal matrices and thus are also both lower and upper triangular.
- For $i < j$, adding a multiple of the ith row to the jth row is represented by a lower triangular matrix.
- For $i > j$, adding a multiple of the ith row to the jth row is represented by an upper triangular matrix.
- Row swaps are not represented as triangular matrices. However, they are permutation matrices.

Now note that in the normal algorithm for Gaussian elimination, assuming we never need to swap rows, all row operations will be such that their matrix representation is lower triangular.

Thus, given an invertible square matrix $\mathbf{A}$, we can represent doing Gaussian elimination on $\mathbf{A}$ by the equation: $\mathbf{L}_n\mathbf{L}_{n-1}\cdots\mathbf{L}_2\mathbf{L}_1\mathbf{A} = \mathbf{U}$ where $\mathbf{L}_i$ is a lower triangular matrix and $\mathbf{U}$ is an upper triangular matrix. Then, because the product of two lower triangular matrices is also lower triangular, we can multiply all the lower triangular matries together to get an equation of the form $\mathbf{LA} = \mathbf{U}$. Finally, we multiply both sides of the equation on the left by $\mathbf{L}^{-1}$ to get that $\mathbf{A} = \mathbf{L}^{-1}\mathbf{U}$. And as the inverse of a lower triangular matrix is also lower triangular, we know that we have decomposed $\mathbf{A}$ into the product of a lower triangular matrix and an upper triangular matrix. This algorithm is called LU decomposition.

As for why we would want to use LU decomposition, we can look at the number of flops different matrix operations require.

Assume we are given an invertible $n \times n$ matrix $\mathbf{A}$ and two vectors $\vec{x}, \vec{b} \in \mathbb{R}^n$.

Firstly note that row reduction takes approximately $\frac{2}{3}n^3$ flops while the back substitution algorithm takes approximately n^2 flops. Thus, for larger values of n, solving the matrix equation $\mathbf{A}\vec{x} = \vec{b}$ takes around $\frac{2}{3}n^3$ flops.

Now lets compare this to the number of flops the best algorithm we can make to do LU decomposition takes.

Assuming we don't need to do any row swaps, than the only elementary row operations we need to do are adding scaled rows to other rows. This is where our first optimization comes into play. The inverse of a row addition is merely a row subtraction. For example:

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

Thus, we can fairly straightforwardly get an expression of the form $\mathbf{A} = \mathbf{L}_1\mathbf{L}_2\cdots\mathbf{L}_k\mathbf{U}$ by multiplying the inverse of each elementary row operation matrix to both sides of the equation representing the actions we took to reduce $\mathbf{A}$ to $\mathbf{U}$.

Now, we apply another two shortcuts to speed up our calculations.

Firstly, observe that: $\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&a&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&b&0&1\end{bmatrix} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&b&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&a&1&0\\0&0&0&1\end{bmatrix} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&a&1&0\\0&b&0&1\end{bmatrix}$

There's nothing special about that column or that particular matrix size. In fact, this should make sense when you consider that those two elementary row operations don't effect each other.

Next, observe that: $\begin{bmatrix}1&0&0&0\\a&1&0&0\\b&0&1&0\\c&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&d&1&0\\0&e&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&f&1\end{bmatrix} = \begin{bmatrix}1&0&0&0\\a&1&0&0\\b&d&1&0\\c&e&f&1\end{bmatrix}$

To intuit why this works, think about how the matrix product has every row affect the rows underneath it before it itself is affected by any rows above it. So, it is specifically the initial state of every row that is effecting every other row in the matrix product.

The end result of these observations is that the (i,j)th element of $\mathbf{L}$ where $i > j$ is just going to be the negative of whatever coefficient one multiplied a copy of row i by before then adding that to row j as part of doing the row reduction algorithm. But now note that means that every element in $\mathbf{L}$ can be directly extracted from the calculations done to find $\mathbf{U}$. So finding $\mathbf{U}$ takes approximately $\frac{2}{3}n^3$ flops and finding $\mathbf{L}$ takes no additional flops.

Once, we've found $\mathbf{LU}$, we can then use back substitution twice to solve any matrix vector equation $\mathbf{A}\vec{x} = \mathbf{LU}\vec{x} = \vec{b}$. This will take $2n^2$ flops.

Technically, this means that if you are trying to solve a single matrix vector equation $\mathbf{A}\vec{x} = \vec{b}$ by first decomposing $\mathbf{A}$, then it will take longer than if you just solved it directly. However, if you have many matrix vector equations involving $\mathbf{A}$, then you can work way faster by decomposing $\mathbf{A}$. This is because once you decompose $\mathbf{A}$ once, you can just store $\mathbf{L}$ and $\mathbf{U}$ for later use. Thus, every subsequent matrix vector equation involving $\mathbf{A}$ can be done in approximately $2n^2$ flops instead of taking approximately $\frac{2}{3}n^3$ flops.

Lecture 5: 1/19/2024

A crucial assumption we made in the last lecture was that we never would have to do row swaps when doing row reduction. In this lecture, we'll now allow ourselves to do row swaps.

Firstly, here are two general reasons to want to do row swaps.

- Firstly, sometimes we have no choice.

$$\begin{bmatrix} 0 & 4 & 1 \\ 1 & 3 & 4 \\ 2 & 2 & 5 \end{bmatrix}$$

For this matrix, if we were to divide the second and third row by $a_{1,1}$, we'd be dividing them by 0. Thus, we clearly can't do that.

- Secondly, dividing by small numbers causes more roundoff erros. So, we can slow the accumulation of roundoff erros by swapping rows in order to prioritize dividing by larger elements.

So here's an algorithm called parial pivoting for taking a PLU decomposition of an invertible $n \times n$ matrix $\mathbf{A}$.

When doing Guassian elimination, for every new pivot of $\mathbf{A}$, first perform a row swap with the top non-reduced row and the non-reduced row with the largest would-be pivot element. Only, after that do you then add a scaled version of the pivot row to the other rows like you were doing before.

Now note that performing the same row swap twice in a row is equivalent to doing nothing. Thus, every matrix representing a row swap is its own inverse.
Additionally, as we already covered, the inverse of a row addition operation is just a row subtraction operation. Because of this, we can easily get an equation for A as the product of many elementary triangular matrices and permutation matrices.

Let's denote $\mathbf{L}_i$ to be the lower triangular matrix representing all row additions of row i to the rest of the matrix. In other words, every element below the main diagonal in $\mathbf{L}_i$ will be nonzero except for the elements in column i. Additionally, let's denote $\mathbf{P}_{i,j}$ to be the permutation matrix swapping row i and row j. Then, we can say that:

$$\mathbf{A} = \mathbf{P}_{1,k_1}\mathbf{L}_1\mathbf{P}_{2,k_2}\mathbf{L}_2 \cdots \mathbf{P}_{n-1,k_{n-1}}\mathbf{L}_{n-1}\mathbf{U}$$

(Note that $i < k_i \leq n$ since it doesn't make sense to swap a row that has already been dealt with.)

We can rewrite this as $\mathbf{P}_{1,k_1}\mathbf{A} = \mathbf{L}_1\mathbf{P}_{2,k_2}\mathbf{L}_2 \cdots \mathbf{P}_{n-1,k_{n-1}}\mathbf{L}_{n-1}\mathbf{U}$. And now here is where the magic starts. If we multiply both sides of that equation on the left by $\mathbf{P}_{2,k_2}$, then we can say that $(\mathbf{P}_{2,k_2}\mathbf{L}_1\mathbf{P}_{2,k_2}) = \mathbf{L}_1'$ where $\mathbf{L}_1'$ is the matrix which would have resulted if we had only applied the permutation $\mathbf{P}_{2,k_2}$ to the elements off the main diagonal.

Now we can do the same process again to $(\mathbf{L}_1'\mathbf{L}_2)\mathbf{P}_{3,k_3}$, multiplying both sides of our equation by $\mathbf{P}_{3,k_3}$ and then saying that $\mathbf{L}_2' = (\mathbf{P}_{3,k_3}\mathbf{L}_1'\mathbf{L}_2\mathbf{P}_{3,k_3})$ is a lower triangular matrix whose only nonzero elements below the diagonal are in columns 1 and 2. Doing this for all remaining permutation matrices on the right side of the equation, we will eventually get an equation of the form:

$$\mathbf{P}_{n-1,k_{n-1}} \cdots \mathbf{P}_{2,k_2}\mathbf{P}_{1,k_1}\mathbf{A} = \mathbf{LU}$$

Finally, by multiplying those permutation matrices together, we get an equation:

$$\mathbf{PA} = \mathbf{LU}$$

Here is why the product $\mathbf{P}_{i+1,k_{i+1}}\mathbf{L}_{i-1}'\mathbf{L}_i\mathbf{P}_{i+1,k_{i+1}}$ was equivalent to only applying the row permutation underneath the diagonal of $\mathbf{L}_{i-1}'\mathbf{L}_i$.

Observe the following diagram of where the permutation matrices send the elements in row $i+1$, row k_{i+1}, column $i+1$, and column k_{i+1} of the matrix $\mathbf{L}_{i-1}'\mathbf{L}_i$:

$$\begin{bmatrix} & & 0 & & 0 & & \\ & & 0 & & 0 & & \\ a & b & 1 & 0 & 0 & 0 & 0 \\ & & 0 & & 0 & & \\ & & 0 & & 0 & & \\ c & d & 0 & 0 & 1 & 0 & 0 \\ & & 0 & & 0 & & \\ & & 0 & & 0 & & \end{bmatrix}$$

$$\downarrow$$

$$\begin{bmatrix} & & 0 & & 0 & & \\ & & 0 & & 0 & & \\ c & d & 1 & 0 & 0 & 0 & 0 \\ & & 0 & & 0 & & \\ & & 0 & & 0 & & \\ a & b & 0 & 0 & 1 & 0 & 0 \\ & & 0 & & 0 & & \\ & & 0 & & 0 & & \end{bmatrix}$$

Theorem: Every invertible matrix has a PLU-decomposition.

Lecture 6: 1/22/2024

A matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is said to be positive definite if:

1. $\mathbf{A}^T = \mathbf{A}$ ($\mathbf{A}$ is symmetric)
2. $\vec{x}^T \mathbf{A} \vec{x} = \langle \vec{x}, \mathbf{A}\vec{x} \rangle > 0$ for all vectors $\vec{x} \in \mathbb{R}^n$ with $\vec{x} \neq 0$.

$$\langle \vec{x}, \mathbf{A}\vec{x} \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i a_{i,j} x_j$$

Lemma: If $\mathbf{A}$ is positive definite, then $\mathbf{A}$ is invertible.

Proof: (we proceed towards a contradiction)

Assume there exists a vector $\vec{y} \in \mathbb{R}^n$ not equal to 0 such that $\mathbf{A}\vec{y} = 0$. Then $\langle \vec{y}, \mathbf{A}\vec{x} \rangle = \langle \vec{y}, 0 \rangle = 0$. So, $\mathbf{A}$ cannot be positive definite.

Theorem: Let $\mathbf{M} \in \mathbb{R}^{n \times n}$ be invertible. Then $\mathbf{A} = \mathbf{M}^T\mathbf{M}$ is positive definite.

Proof:

1. $\mathbf{A}^T = \left(\mathbf{M}^T\mathbf{M}\right)^T = \mathbf{M}^T \left(\mathbf{M}^T\right)^T = \mathbf{M}^T\mathbf{M} = \mathbf{A}$. So $\mathbf{A}$ is symmetric.
2. Note that $\vec{x}^T\mathbf{A}\vec{x} = \vec{x}^T\mathbf{M}^T\mathbf{M}\vec{x} = (\mathbf{M}\vec{x})^T\mathbf{M}\vec{x} = \langle \mathbf{M}\vec{x}, \mathbf{M}\vec{x} \rangle = |\mathbf{M}\vec{x}|$. Now, $|\mathbf{M}\vec{x}| > 0$ for all $\mathbf{M}\vec{x} \neq 0$. And because $\mathbf{M}$ is invertible, we know the only $\vec{x}$ such that $\mathbf{M}\vec{x} = 0$ is $\vec{x} = 0$. So, $\vec{x}^T\mathbf{A}\vec{x} > 0$ for all $\vec{x} \neq 0$.

   If $\mathbf{M}$ is not invertible, then $\mathbf{A}$ is positive semidefnite as $|\mathbf{M}\vec{x}|$ cannot be less than 0 but we can find a nonzero $\vec{x}$ such that $|\mathbf{M}\vec{x}| = 0$.

Theorem (Cholesky decomposition):

Let $\mathbf{A}$ be positive definite. Then there exists an upper triangular matrix $\mathbf{R}$ such that $\mathbf{A} = \mathbf{R}^T\mathbf{R}$. We call $\mathbf{R}$ the Cholesky factor and can calculate it as follows:

If such an $\mathbf{R}$ exists, then we can arrive at the following formula for each element of $\mathbf{A} = \mathbf{R}^T\mathbf{R}$:

$$a_{i,j} = \sum_{k=1}^{\min(i,j)} r_{k,i} r_{k,j}$$

Now as $\mathbf{A}$ must be symmetric based on our formula, we can safely ignore the elements of $\mathbf{A}$ below the main diagonal. So, assuming that $j \geq i$, we can rearrange terms to get the following equation including the element $a_{i,j}$:

$$r_{i,i} r_{i,j} = a_{i,j} - \sum_{k=1}^{i-1} r_{k,i} r_{k,j}$$

And with that, we now have a way of expressing any element $r_{i,j}$ in terms of elements of $\mathbf{R}$ which are in previous rows or previous columns of $\mathbf{R}$. So, we can inductively solve for the coefficients of $\mathbf{R}$ as follows:

for $i = 1, \ldots, n$:

$$r_{i,i} = \sqrt{a_{i,i} - \sum_{k=1}^{i-1} r_{k,i}^2}$$

for $j = (i+1), \ldots, n$:

$$r_{i,j} = \frac{1}{r_{i,i}} \left(a_{i,j} - \sum_{k=1}^{i-1} r_{k,i} r_{k,j} \right)$$

end

end

Now it's worth noting what can go wrong in the above algorithm.

Firstly, if $a_{i,i} - \sum_{k=1}^{i-1} r_{k,i}^2 < 0$, then $r_{i,i} \notin \mathbb{R}$. So, $\mathbf{R}$ does not exist

Secondly, if $r_{i,i} = 0$, then you can't divide by it to isolate $r_{i,j}$. Thus, one of two possibilities will arise:

- If $\left(a_{i,j} - \sum_{k=1}^{i-1} r_{k,i} r_{k,j} \right) = 0$, then you can set $r_{i,j}$ to anything and maintain equality. Thus, if a Cholesky factorization exists, it is not unique.

- If $\left(a_{i,j} - \sum_{k=1}^{i-1} r_{k,i} r_{k,j} \right) \neq 0$, then there is nothing you can set $r_{i,j}$ to. So, no Cholesky factor of $\mathbf{A}$ exists.

Now unfortunately, this class is not interested in proving when and when not each of the above problems will arise. So, for now just know that:

1. $\mathbf{A}$ is positive definite if and only if $\mathbf{R}$ exists and is unique.
2. $\mathbf{A}$ is positive semidefinite if and only if $\mathbf{R}$ exists but is not invertible and thus also not unique.
3. $\mathbf{A}$ is not positive definite or semidefinite if and only if $\mathbf{R}$ does not exist.

The second theorem covered in lecture today let's us easily prove one direction in each of the above implications. So, the challenging task would be to prove the other direction, and to do that you would need to show that for any positive definite or semidefinite matrix $\mathbf{A}$, you can always use the above algorithm to find a matrix $\mathbf{R}$.

It takes approximately $\frac{1}{3}n^3$ flops to do Cholesky decomposition as n gets larger. This is notably half of what LU decomposition takes.

Lecture 7: 1/24/2024

A norm of a vector $\vec{x} \in \mathbb{R}^n$ is a real number $\|\vec{x}\|$ that is assigned to $\vec{x}$ satisfying:

- $\vec{x} \neq 0 \implies \|\vec{x}\| > 0$ whereas $\|\vec{0}\| = 0$.
- $\|c\vec{x}\| = |c|\|\vec{x}\|$ for all $c \in \mathbb{R}$.
- $\|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\|$ for all $\vec{x}, \vec{y} \in \mathbb{R}^n$

Some important vector norms:

1. Vector p-norm: For an integer $p \geq 1$, we define $\|\vec{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$
2. Infinity Norm: $\|\vec{x}\|_\infty = \max\{|x_i| \mid 1 \leq i \leq n\}$

A matrix norm assigns a real value $\|\mathbf{A}\|$ to a matrix $\mathbf{A}$ satisfying:

- $\mathbf{A} \neq \mathbf{0} \implies \|\mathbf{A}\| > 0$ whereas $\|\mathbf{0}\| = 0$.
- $\|c\mathbf{A}\| = |c|\|\mathbf{A}\|$ for all $c \in \mathbb{R}$.
- $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$ for all $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$

Some important matrix norms:

1. Frobenius Norm: $\|\mathbf{A}\| = \left(\sum_{i,j=1}^{n} a_{i,j}^2\right)^{\frac{1}{2}}$

   Assuming $\mathbf{A}$ is an $m \times n$ matrix, this norm is equivalent to stringing out $\mathbf{A}$'s rows or columns to form an mn element vector and then taking the 2-norm of the resulting vector.

   $$\left\| \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \right\| = \|(a, b, c, d, e, f, g, h, i)\|$$

2. Induced Norm: Let $\|\cdot\|$ be a vector norm on $\mathbb{R}^m$ and $\mathbb{R}^n$. Then the matrix norm induced by $\|\cdot\|$ is:

   $$\|\mathbf{A}\| = \max_{\vec{x} \neq 0} \frac{\|\mathbf{A}\vec{x}\|}{\|\vec{x}\|}$$

The induced norm can be thought of as measuring the maximum stretch which a linear function $L(\vec{x}) = \mathbf{A}\vec{x}$ applies to $\vec{x}$ relative to $\vec{x}$'s starting length.

Importantly by the properties of a norm:

$$\frac{\|\mathbf{A}\vec{x}\|}{\|\vec{x}\|} = \left\| \frac{1}{\|\vec{x}\|}\mathbf{A}\vec{x} \right\| = \left\| \mathbf{A}\frac{\vec{x}}{\|\vec{x}\|} \right\| = \|\mathbf{A}\hat{x}\|$$

Thus, it's also common to see induced norms defined as: $\|\mathbf{A}\| = \max_{\|\vec{x}\|=1} \|\mathbf{A}\vec{x}\|$

Some important formulas: (let $\mathbf{A}$ be an $m \times n$ matrix)

- $\|\mathbf{A}\|_1 = \max_{1\le j\le n} \sum_{k=1}^{m} |a_{k,j}|$

  To prove this, first note that:

  $$\|\mathbf{A}\vec{x}\|_1 = \sum_{i=1}^{m} \left| \sum_{j=1}^{n} a_{i,j}x_j \right| \le \sum_{i=1}^{m}\sum_{j=1}^{n} |a_{i,j}|\,|x_j| = \sum_{j=1}^{n} \left(|x_j| \sum_{i=1}^{m} |a_{i,j}| \right)$$

  Now lets replace each $a_{i,j}$ with the max element $a_{i,k}$ in the ith row. Thus:

  $$\sum_{j=1}^{n} \left(|x_j| \sum_{i=1}^{m} |a_{i,j}| \right) \le \sum_{j=1}^{n} \left(|x_j| \max_{1\le k\le m} \sum_{i=1}^{m} |a_{i,k}| \right)$$

  And now we can seperate the summands to get that:

  $$\|\mathbf{A}\vec{x}\|_1 \le \|\vec{x}\|_1 \max_{1\le k\le m} \left(\sum_{i=1}^{m} |a_{i,k}| \right)$$

  Now $\|\mathbf{A}\|_1$ is the max value of $\|\mathbf{A}\vec{x}\|_1$ when $\|\vec{x}\|_1 = 1$. So clearly:

  $$\|\mathbf{A}\|_1 \le \max_{1\le k\le m} \left(\sum_{i=1}^{m} |a_{i,k}| \right)$$

  Finally, to show equality note that the kth. standard unit basis vector e_k satisfies both that $\|e_k\|_1 = 1$ and that:

  $$\|\mathbf{A}e_k\| = \max_{1\le k\le m} \left(\sum_{i=1}^{m} |a_{i,k}| \right)$$

  .

- $\|\mathbf{A}\|_\infty = \max_{1\le i\le m} \sum_{k=1}^{n} |a_{i,k}|$

  This one is more obvious. The $\vec{x}$ which maximizes $\|\mathbf{A}\vec{x}\|_\infty$ while $\|\vec{x}\|_\infty = 1$ is the vector whose elemnts are 1 and -1 such that the kth element of $\vec{x}$ has the same sign as the kth element of the row of $\mathbf{A}$ whose elements have the largest sum of absolute values.

- $\|\mathbf{A}\|_2 = \sqrt{\lambda_{\max}(\mathbf{A}^T\mathbf{A})}$

  ($\lambda_{\max}(\mathbf{M})$ refers to the largest eigenvalue of $\mathbf{M}$.)

  Neither the lecture notes or the author of our textbook Börgers at this point gives a proof of this. However, at the very least we know from the proposition on page 8 that $\mathbf{A}^T\mathbf{A}$ is positive definite or positive semidefinite. So, all of its eigenvalues are greater than or equal to zero, meaning that our distance formula will always give a real value.

Lecture 8: 1/26/2024

Here is proof that induced norms are matrix norms.

1. Positivity:

   If $\|\mathbf{A}\| = 0$, then $\|\mathbf{A}\vec{x}\| = 0$ for all nonzero $\vec{x}$. This means that $\mathbf{A}\vec{x} = \vec{0}$ for all nonzero $\vec{x}$. However, note that $\mathbf{A}e_k$ equals the kth column of $\mathbf{A}$. So, all columns of $\mathbf{A}$ must be zero. Hence, $\|\mathbf{A}\| = 0 \implies \mathbf{A} = \mathbf{0}$

2. Multiplication by a constant:

$$\|c\mathbf{A}\| = \max_{\vec{x}\neq 0} \frac{\|c\mathbf{A}\vec{x}\|}{\|\vec{x}\|} = \max_{\vec{x}\neq 0} \frac{|c|\,\|\mathbf{A}\vec{x}\|}{\|\vec{x}\|} = |c| \left(\max_{\vec{x}\neq 0} \frac{\|\mathbf{A}\vec{x}\|}{\|\vec{x}\|}\right) = |c|\,\|\mathbf{A}\|$$

3. Triangle inequality:

$$\|\mathbf{A}+\mathbf{B}\| = \max_{\vec{x}\neq 0} \frac{\|(\mathbf{A}+\mathbf{B})\vec{x}\|}{\|\vec{x}\|} = \max_{\vec{x}\neq 0} \frac{\|\mathbf{A}\vec{x}+\mathbf{B}\vec{x}\|}{\|\vec{x}\|}$$

   Now, by the triangle inequality of the vector norm, we have that:

$$\frac{\|\mathbf{A}\vec{x}+\mathbf{B}\vec{x}\|}{\|\vec{x}\|} \leq \frac{\|\mathbf{A}\vec{x}\|+\|\mathbf{B}\vec{x}\|}{\|\vec{x}\|} = \frac{\|\mathbf{A}\vec{x}\|}{\|\vec{x}\|} + \frac{\|\mathbf{B}\vec{x}\|}{\|\vec{x}\|}$$

   So, $\|\mathbf{A}+\mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$.

Here are two more properties of induced norms:

4. $\|\mathbf{AB}\| \leq \|\mathbf{A}\|\|\mathbf{B}\|$ This is called submultiplicativity.

$$\|\mathbf{AB}\| = \max_{\vec{x}\neq 0} \frac{\|(\mathbf{AB})\vec{x}\|}{\|\vec{x}\|} = \max_{\vec{x}\neq 0} \left(\frac{\|\mathbf{AB}\vec{x}\|}{\|\mathbf{B}\vec{x}\|}\frac{\|\mathbf{B}\vec{x}\|}{\|\vec{x}\|}\right) \text{ if } \mathbf{B}\vec{x} \neq 0.$$

   Now this is obviously less than or equal to $\max_{\mathbf{B}\vec{x}\neq 0} \left(\frac{\|\mathbf{AB}\vec{x}\|}{\|\mathbf{B}\vec{x}\|}\right) \max_{\vec{x}\neq 0} \left(\frac{\|\mathbf{B}\vec{x}\|}{\|\vec{x}\|}\right)$

   Finally defining $\vec{y} = \mathbf{B}\vec{x}$, we get that:

$$\|\mathbf{AB}\| \leq \max_{\vec{y}\neq 0} \left(\frac{\|\mathbf{A}\vec{y}\|}{\|\vec{y}\|}\right) \max_{\vec{x}\neq 0} \left(\frac{\|\mathbf{B}\vec{x}\|}{\|\vec{x}\|}\right) = \|\mathbf{A}\|\|\mathbf{B}\|$$

5. $\|\mathbf{A}\vec{x}\| \leq \|\mathbf{A}\|\|\vec{x}\|$ for all $\vec{x} \in \mathbb{R}^n$.

To explain this, remember that $\|\mathbf{A}\|$ is the maximum value that $\frac{\|\mathbf{A}\vec{x}\|}{\|\vec{x}\|}$ could be.

Sensitivity of $\mathbf{A}\vec{x} = \vec{b}$ with respect to perturbation

Due to rounding errors and limited precision, we know that $\vec{b}$ and $\mathbf{A}$ will typically not be perfectly accurate when doing calculations. We refer to introducing this inaccuracy as perturbing $\vec{b}$ and $\mathbf{A}$. Now as seen in the demonstration below, this will effect the accuracy of our calculations. So, it would be nice to be able to predict and quantify that loss in accuracy.

Demonstration of perturbing $\vec{b}$:

$\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} = \begin{bmatrix}2\\ 0\end{bmatrix}$ gives a solution of $\vec{x} = \begin{bmatrix}2\\ 0\end{bmatrix}$. Meanwhile,

$\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} = \begin{bmatrix}2\\ 0.001\end{bmatrix}$ gives a solution of $\vec{x}' = \begin{bmatrix}1.999\\ 0.001\end{bmatrix}$.

The error in $\vec{b}$ is $\|\vec{b} - \vec{b}'\|_2 = 0.001$, whereas the corresponding error in $\vec{x}$ is $\|\vec{x} - \vec{x}'\|_2 = \sqrt{2}(0.001) \approx 0.0014$

For now, let's address perturbing $\vec{b}$...

Assume $\mathbf{A}$ is an invertible matrix.

Formally, we shall write:

Original equation: $\mathbf{A}\vec{x} = \vec{b}$

New equation: $\mathbf{A}\vec{x}' = \vec{b}'$

where $\vec{b}' = \vec{b} + \vec{\delta b}$ and $\vec{x}' = \vec{x} + \vec{\delta x}$

Then for any vector norm and its induced matrix norm: $\frac{\|\vec{\delta x}\|}{\|\vec{x}\|} \leq \|\mathbf{A}\|\|\mathbf{A}^{-1}\|\frac{\|\vec{\delta b}\|}{\|\vec{b}\|}$.

Proof:

Firstly, $\mathbf{A}\vec{x}' = \vec{b}' \implies \mathbf{A}\vec{x} + \mathbf{A}\vec{\delta x} = \vec{b} + \vec{\delta b}$. Now since $\mathbf{A}\vec{x} = \vec{b}$, it must be the case that $\mathbf{A}\vec{\delta x} = \vec{\delta b}$. So, $\vec{\delta x} = \mathbf{A}^{-1}\vec{\delta b}$. From this, we now get that for any vector norm and corresponding induced matrix norm, we have that $\|\vec{\delta x}\| = \|\mathbf{A}^{-1}\vec{\delta b}\| \leq \|\mathbf{A}^{-1}\|\|\vec{\delta b}\|$.

Secondly, using the same norms as before, we have that: $\vec{b} = \mathbf{A}\vec{x} \implies \|\vec{b}\| = \|\mathbf{A}\vec{x}\| \leq \|\mathbf{A}\|\|\vec{x}\|$. Thus dividing both sides by $\|\vec{b}\|\|\vec{x}\|$, we get that:

$$\frac{1}{\|\vec{x}\|} \leq \|\mathbf{A}\|\frac{1}{\|\vec{b}\|}$$

Thus in conclusion: $\frac{\|\vec{\delta x}\|}{\|\vec{x}\|} \leq \|\mathbf{A}\|\|\mathbf{A^{-1}}\|\frac{\|\vec{\delta b}\|}{\|\vec{b}\|}$

We call $K(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A^{-1}}\|$ the condition number of $\mathbf{A}$ under some induced norm (indicated as a subscript to K). Clearly, it is useful as it gives us an upper bound to the relative error of our solution $\vec{x}$ given the relative error in $\vec{b}$.

- If $K(\mathbf{A})$ is small (close to 1), then $\mathbf{A}$ is called well conditioned.
- If $K(\mathbf{A})$ is large, then $\mathbf{A}$ is called ill conditioned.

Lecture 9: 1/29/2024

Now let's consider perturbing $\mathbf{A}$. As before, we shall assume that $\mathbf{A}$ is invertible.

Formally, we shall write:

Original equation: $\mathbf{A}\vec{x} = \vec{b}$

New equation: $\mathbf{A}'\vec{x}' = \vec{b}$

where $\mathbf{A}' = \mathbf{A} + \delta\mathbf{A}$ and $\vec{x}' = \vec{x} + \vec{\delta x}$

Then for any vector norm and its induced matrix norm: $\frac{\|\vec{\delta x}\|}{\|\vec{x}'\|} \leq K(A)\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|}$.

Proof:

We can rewrite $\mathbf{A}'\vec{x}' = \vec{b}$ as $\mathbf{A}\vec{x} + \mathbf{A}\vec{\delta x} + \delta\mathbf{A}(\vec{x} + \vec{\delta x}) = \vec{b}$. Then, by canceling out $\vec{b}$ and $\mathbf{A}\vec{x}$, we get the formula: $\mathbf{A}\vec{\delta x} + \delta\mathbf{A}\vec{x}' = 0$. And, further rearranging of this yields:

$$\vec{\delta x} = -\mathbf{A}^{-1}(\delta\mathbf{A})\vec{x}'$$

So by the submultiplicativity property of matrix norms:

$$\|\vec{\delta x}\| = \|\mathbf{A}^{-1}(\delta\mathbf{A})\vec{x}'\| \leq \|\mathbf{A}^{-1}\|\|(\delta\mathbf{A})\vec{x}'\| \leq \|\mathbf{A}^{-1}\|\|(\delta\mathbf{A})\|\|\vec{x}'\|$$

Dividing both sides by our above ineqality by $\|\vec{x}'\|$ and multiplying the greater side by $1 = \frac{\|\mathbf{A}\|}{\|\mathbf{A}\|}$, we finally get that:

$$\frac{\|\vec{\delta x}\|}{\|\vec{x}'\|} \leq \|\mathbf{A}^{-1}\|\|\mathbf{A}\|\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} = K(A)\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|}$$

Finally, let's consider perturbing $\mathbf{A}$ and $\vec{b}$. Once again, we will assume $\mathbf{A}^{-1}$ exists.

Formally, we shall write:

Original equation: $\mathbf{A}\vec{x} = \vec{b}$ where $\mathbf{A}' = \mathbf{A} + \delta\mathbf{A}$, $\vec{b}' = \vec{b} + \vec{\delta b}$,

New equation: $\mathbf{A}'\vec{x}' = \vec{b}'$ and $\vec{x}' = \vec{x} + \vec{\delta x}$

Then for any vector norm and its induced matrix norm:

$$\frac{\|\vec{\delta x}\|}{\|\vec{x}'\|} \leq K(A)\left(\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} + \frac{\|\vec{\delta b}\|}{\|\vec{b}'\|} + \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|}\frac{\|\vec{\delta b}\|}{\|\vec{b}'\|}\right)$$

.

Proof:

Step 1:
$\mathbf{A}'\vec{x}' = \vec{b}'$ can be rewritten as: $\mathbf{A}\vec{x} + \mathbf{A}\vec{\delta x} + \delta\mathbf{A}(\vec{x} + \vec{\delta x}) = \vec{b} + \vec{\delta b}$. Then as $\mathbf{A}\vec{x} = \vec{b}$, we cancel those two terms to ge that $\mathbf{A}\vec{\delta x} + \delta\mathbf{A}(\vec{x}') = \vec{\delta b}$. And so, a little rearranging yields:

$$\vec{\delta x} = \mathbf{A}^{-1}\left(\vec{\delta b} - \delta\mathbf{A}\,\vec{x}'\right)$$

Next, using properties of the induced matrix norm, we get that:

$$\begin{aligned}\|\vec{\delta x}\| = \|\mathbf{A}^{-1}\left(\vec{\delta b} - \delta\mathbf{A}\,\vec{x}'\right)\| &\leq \|\mathbf{A}^{-1}\|\|\vec{\delta b} - \delta\mathbf{A}\,\vec{x}'\| \\ &\leq \|\mathbf{A}^{-1}\|\left(\|\vec{\delta b}\| + \|\delta\mathbf{A}\,\vec{x}'\|\right) \\ &\leq \|\mathbf{A}^{-1}\|\left(\|\vec{\delta b}\| + \|\delta\mathbf{A}\|\|\vec{x}'\|\right)\end{aligned}$$

Therefore, by dividing both sides of our inequality by $\|\vec{x}'\|$ and multiplying the greater side by $1 = \frac{\|\mathbf{A}\|}{\|\mathbf{A}\|}$, we finally get that:

$$\frac{\|\vec{\delta x}\|}{\|\vec{x}'\|} \leq K(A)\left(\frac{\|\vec{\delta b}\|}{\|\mathbf{A}\|\|\vec{x}'\|} + \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|}\right)$$

Step 2:
$\vec{b}' = \mathbf{A}'\vec{x}' \Longrightarrow \|\vec{b}'\| = \|(\mathbf{A}')\vec{x}'\| \leq \|\mathbf{A} + \delta\mathbf{A}\|\|\vec{x}'\|$
So dividing by $\|\vec{b}'\|\|\vec{x}'\|$, we get that:

$$\frac{1}{\|\vec{x}'\|} \leq \frac{\|\mathbf{A} + \delta\mathbf{A}\|}{\|\vec{b}'\|} \leq \frac{\|\mathbf{A}\| + \|\delta\mathbf{A}\|}{\|\vec{b}'\|}$$

Step 3:
By combining the inequalities in steps 1 and 2, we know:

$$\frac{\|\vec{\delta x}\|}{\|\vec{x}'\|} \leq K(A)\left(\frac{\|\vec{\delta b}\|}{\|\mathbf{A}\|}\left(\frac{\|\mathbf{A}\|+\|\delta\mathbf{A}\|}{\|\vec{b}'\|}\right)+\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|}\right)$$

We can then manipulate that into the inequality claimed above.

If you are wondering why the upper bounds we derived in this lecture are for $\frac{\|\vec{\delta x}\|}{\|\vec{x}'\|}$ instead of $\frac{\|\vec{\delta x}\|}{\|\vec{x}\|}$, know that I am too. I asked the professor and he said its because the upper bounds for $\frac{\|\vec{\delta x}\|}{\|\vec{x}\|}$ are more difficult. That said, our textbook does derive upper bounds for $\frac{\|\vec{\delta x}\|}{\|\vec{x}\|}$ when perturbing $\mathbf{A}$.

Proposition: Let $\|\cdot\|$ be an induced matrix norm. Then:

1. $\|\mathbf{I}\| = 1$
2. $K(\mathbf{A}) \geq 1$.

Proof:

1. $\|\mathbf{I}\| = \max_{\vec{x}\neq 0}\frac{\|\mathbf{I}\vec{x}\|}{\|\vec{x}\|} = \max_{\vec{x}\neq 0}\frac{\|\vec{x}\|}{\|\vec{x}\|} = 1$
2. $\mathbf{I} = \mathbf{A}\mathbf{A}^{-1} \implies 1 = \|\mathbf{I}\| = \|\mathbf{A}\mathbf{A}^{-1}\| \leq \|\mathbf{A}\|\|\mathbf{A}^{-1}\| = K(\mathbf{A})$

Lecture 10: 2/5/2024

A matrix $\mathbf{Q}$ is orthogonal if $\mathbf{Q}^{-1} = \mathbf{Q}^T$

If $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is orthogonal, then:

1. $\langle \mathbf{Q}\vec{x}, \mathbf{Q}\vec{y}\rangle = \vec{x}^T\mathbf{Q}^T\mathbf{Q}\vec{y} = \vec{x}^T\vec{y} = \langle \vec{x}, \vec{y}\rangle$
2. $\|\mathbf{Q}\vec{x}\|_2 = \langle \mathbf{Q}\vec{x}, \mathbf{Q}\vec{x}\rangle = \langle \vec{x}, \vec{x}\rangle = \|\vec{x}\|_2$

Theorem (QR decomposition / full QR decomposition):
Suppose $\mathbf{A} \in \mathbb{R}^{n\times m}$ where $n \geq m$. Then there exists an orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and a matrix $\mathbf{R} = \begin{bmatrix}\hat{\mathbf{R}}\\ \mathbf{0}\end{bmatrix} \in \mathbb{R}^{n\times m}$ where $\hat{\mathbf{R}}$ is an $m \times m$ upper triangular matrix such that $\mathbf{A} = \mathbf{Q}\mathbf{R}$.

Some uses of QR decompositions:

- Solving matrix-vector equations:

  Using the QR decomposition of a matrix, you can rewrite an equation $\mathbf{A}\vec{x} = \vec{b}$ as follows:

  $$\mathbf{A}\vec{x} = \mathbf{QR}\vec{x} = \vec{b} \Rightarrow \mathbf{R}\vec{x} = \mathbf{Q}^{-1}\vec{b}$$

  And now you can find $\vec{x}$ using back substitution and matrix vector multiplication.

- The Least Square Problem (LSE): Suppose $\mathbf{A} \in \mathbb{R}^{n\times m}$ where $n \geq m$ and $b \in \mathbb{R}^n$. Then the least square problem is of the form: $\min_{\vec{x}\in\mathbb{R}^m} (\|\vec{b} - \mathbf{A}\vec{x}\|_2)^2$.

  For $n \geq m$, if the matrix-vector equation $\mathbf{A}\vec{x} = \vec{b}$ has a solution, then the solution of the least square problem is also the solution of that matrix vector equation.

  However, if there is no solution to the matrix-vector equation, then the least square problem is asking us to find a vector $\vec{x}$ that minimizes the distance between $\mathbf{A}\vec{x}$ and $\vec{b}$.

  Step 1. Find a QR-decomposition such that $\mathbf{A} = \mathbf{QR}$

  Step 2. $$(\|\vec{b} - \mathbf{A}\vec{x}\|_2)^2 = (\|\vec{b} - \mathbf{QR}\vec{x}\|_2)^2 = (\|\mathbf{Q}^T(\vec{b} - \mathbf{QR}\vec{x})\|_2)^2$$
  $$= (\|\mathbf{Q}^T\vec{b} - \mathbf{Q}^T\mathbf{QR}\vec{x}\|_2)^2 = (\|\mathbf{Q}^T\vec{b} - \mathbf{R}\vec{x}\|_2)^2$$

  Now denote: $\mathbf{Q}^T\vec{b} = \begin{bmatrix}\vec{c}\\ \vec{d}\end{bmatrix}$ and $\mathbf{R}\vec{x} = \begin{bmatrix}\hat{\mathbf{R}}\vec{x}\\ \vec{0}\end{bmatrix}$

  ...where $\vec{c}, \hat{\mathbf{R}}\vec{x} \in \mathbb{R}^m$ and $\vec{d} \in \mathbb{R}^{n-m}$

  $$(\|\mathbf{Q}^T\vec{b} - \mathbf{R}\vec{x}\|_2)^2 = \left(\left\|\begin{bmatrix}\vec{c}\\ \vec{d}\end{bmatrix} - \begin{bmatrix}\hat{\mathbf{R}}\vec{x}\\ \vec{0}\end{bmatrix}\right\|_2\right)^2 = \left\|\begin{bmatrix}\vec{c} - \hat{\mathbf{R}}\vec{x}\\ \vec{d}\end{bmatrix}\right\|_2^2$$
  $$= (\|\vec{c} - \hat{\mathbf{R}}\vec{x}\|_2)^2 + (\|\vec{d}\|_2)^2$$

  Step 3. Now $$\min_{\vec{x}\in\mathbb{R}^m} (\|\vec{b} - \mathbf{A}\vec{x}\|_2)^2 = \left(\min_{\vec{x}\in\mathbb{R}^m} (\|\vec{c} - \hat{\mathbf{R}}\vec{x}\|_2)^2\right) + (\|\vec{d}\|_2)^2$$

  Step 4. If $\mathbf{A}$ has full-rank, then $\hat{\mathbf{R}} \in \mathbb{R}^{m\times m}$ is invertible. So: $\hat{\mathbf{R}}\vec{x} = \vec{c}$ has a unique solution, meaning that $\min_{\vec{x}\in\mathbb{R}^m} (\|\vec{b} - \mathbf{A}\vec{x}\|_2)^2 = (\|\vec{d}\|_2)^2$. So we get the theorem on the next page:

Theorem: (For if $\mathbf{A}$ has full rank...)

- The solution to the least squares problem is the solution to the linear system $\hat{\mathbf{R}}\,\vec{x} = \vec{c}$ where $\vec{c}$ is the first m values of $\mathbf{Q}^T\,\vec{b}$.
- Meanwhile, the optimal value is $\min_{\vec{x}\in\mathbb{R}^m} (\|\,\vec{b} - \mathbf{A}\,\vec{x}\|_2)^2 = (\|\,\vec{d}\,\|_2)^2$ where $\vec{d}$ is the remaining elements of $\mathbf{Q}^T\,\vec{b}$.

Lecture 11: 2/7/2024

A QR decomposition of $\begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix}$ is: $\mathbf{QR} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 0 & 2 & 8 \\ 0 & 0 & 4 \\ 0 & 0 & 0 \end{bmatrix}$

Meanwhile, a reduced QR decomposition of that same matrix would be:

$$\hat{\mathbf{Q}}\hat{\mathbf{R}} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 0 & 2 & 8 \\ 0 & 0 & 4 \end{bmatrix}$$

...where $\hat{\mathbf{R}}$ is the $m \times m$ upper triangular inside $\mathbf{R}$ while $\hat{\mathbf{Q}}$ is the matrix of the first m columns in $\mathbf{Q}$.

Two notes:

- $\mathbf{QR} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$.
- $\hat{\mathbf{Q}}$ is not orthogonal but it does have the property that $\hat{\mathbf{Q}}^T\hat{\mathbf{Q}} = I_m$. In other words, the columns of $\hat{\mathbf{Q}}$ are an orthonormal set (a.k.a it's column-orthogonal).

Using the Gram-Schmit algorithm, for any $\mathbf{A} \in \mathbb{R}^{n\times m}$ with full rank, we can calculate a column-orthogonal matrix $\hat{\mathbf{Q}}$ and upper triangular matrix $\hat{\mathbf{R}}$ such that $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$.

Classsic Gram-Schmit Algorithm:

We'll denote $\mathbf{A} = \begin{bmatrix} \vec{a}_1 & \cdots & \vec{a}_m \end{bmatrix}$ and $\hat{\mathbf{Q}} = \begin{bmatrix} \vec{q}_1 & \cdots & \vec{q}_m \end{bmatrix}$

For $i = 1, \cdots, m$:

  For $k = 1, \cdots (i-1)$:

    $r_{k,i} = \langle \vec{a}_i, \vec{q}_k \rangle$

  end

  $r_{i,i} = \left\| \vec{a}_i - \sum_{k=1}^{i-1} (r_{k,i}\, \vec{q}_k) \right\|_2$ and $\vec{q}_i = \dfrac{\vec{a}_i - \sum_{k=1}^{i-1} (r_{k,i}\, \vec{q}_k)}{r_{i,i}}$

end

To remember this algorithm, note that:

- $\hat{\mathbf{Q}}\hat{\mathbf{R}} = \mathbf{A} \implies \hat{\mathbf{R}} = \hat{\mathbf{Q}}^T\mathbf{A}$. So, $r_{i,j}$ equals the inner product of the ith column of $\hat{\mathbf{Q}}^T$ and the jth column of $\mathbf{A}$.
- If V_i is the space spanned by the first i columns of $\hat{\mathbf{Q}}$, then $\vec{q}_{i+1}$ is equal to the normalized result of $\vec{a}_{i+1} - \text{Proj}_{V_i}(\vec{a}_{i+1})$. However, each $\vec{q}_i$ is a unit vector. So, the projection formula is simplified.

Modified Gram-Schmit Algorithm: (more stable for computers)

```
function [Q, R] = modifiedGS(A)
   % Storing the number of columns in A
   m = size(A, 2);

   % Copy Q into A (Matlab doesn't seem to alias the matrices)
   Q = A;

   for i=1:m
      R(i,i) = norm(Q(:,i));
      Q(:,I) = Q(:,i) / R(i, i) %Q(:,i) * (1 / R(i, i))

      for j = (i+1):m
         R(i,j) = Q(:,i)' * Q(i,j); % A' is the transpose of A
         Q(:,j) = Q(:,j) - (R(i,j) * Q(:, i));
      end
   end
end
```

The way that this modified algorithm differs from the original one is that instead of building a column of $\hat{\mathbf{Q}}$ every iteration, it instead builds a row of $\hat{\mathbf{R}}$ every iteration. This is much more numerically stable.

Lecture 12: 2/9/2024

A matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$ is called a projector if $\mathbf{P}^2 = \mathbf{P}$.

Given $\vec{u} \in \mathbb{R}^n$ with $\|\vec{u}\|_2 = 1$, then $\mathbf{P} = \vec{u}\,\vec{u}^T$ is an orthogonal projector.

Proof:

- $\mathbf{P}^2 = (\vec{u}\,\vec{u}^T)(\vec{u}\,\vec{u}^T) = \vec{u}\langle \vec{u}, \vec{u}\rangle \vec{u}^T = 1 \cdot \vec{u}\,\vec{u}^T = \mathbf{P}$
- $\mathbf{P}^T = \left(\vec{u}\,\vec{u}^T\right)^T = \left(\vec{u}^T\right)^T \vec{u}^T = \vec{u}\,\vec{u}^T = \mathbf{P}$

Properties of $\mathbf{P} = \vec{u}\,\vec{u}^T$:

- $\mathbf{P}\vec{u} = \vec{u}$.
- $\langle \vec{u}, \vec{v}\rangle = 0 \Longrightarrow \mathbf{P}\vec{v} = 0$

Let $\vec{u} \in \mathbb{R}^n$ with $\|u\|_2 = 1$. Then the matrix $\mathbf{Q} = \mathbf{I} - 2\vec{u}\,\vec{u}^T$ is called a (Householder) reflector.

If $\|\vec{u}\|_2 \neq 1$, you an always normalize it and construct a reflector: $\mathbf{Q} = \mathbf{I} - \frac{2}{(\|\vec{u}\|_2)^2}\vec{u}\,\vec{u}^T$

Properties of $\mathbf{Q} = \mathbf{I} - 2\vec{u}\,\vec{u}^T$ where $\|\vec{u}\|_2 = 1$:

- $\mathbf{Q}\vec{u} = -\vec{u}$.
  Proof: $\mathbf{Q}\vec{u} = (\mathbf{I} - 2\vec{u}\,\vec{u}^T)\vec{u} = \vec{u} - 2\vec{u} = -\vec{u}$
- $\langle \vec{u}, \vec{v}\rangle = 0 \Longrightarrow \mathbf{Q}\vec{v} = \vec{v}$
  Proof: $\mathbf{Q}\vec{v} = (\mathbf{I} - 2\vec{u}\,\vec{u}^T)\vec{v} = \vec{v} - 2\vec{u}(0) = \vec{v}$
- $\mathbf{Q}^T = \mathbf{Q}$
  Proof: $\mathbf{Q}^T = (\mathbf{I} - 2\vec{u}\,\vec{u}^T)^T = \mathbf{I}^T - 2(\vec{u}\,\vec{u}^T)^T = \mathbf{I} - 2\vec{u}\,\vec{u}^T = \mathbf{Q}$
- $\mathbf{Q}^2 = \mathbf{Q}$
  Proof: $\mathbf{Q}^2 = (\mathbf{I} - 2\vec{u}\,\vec{u}^T)^2 = \mathbf{I} - 4\vec{u}\,\vec{u}^T + 4(\vec{u}\,\vec{u}^T)^2 = \mathbf{I}$

Basically, $\mathbf{Q}$ reflects any vector $\vec{x}$ about the hyperplane perpendicular to $\vec{u}$.

Suppose $\vec{x} \in \mathbb{R}^n$ and $\vec{y} = \begin{bmatrix} \|\vec{x}\|_2 & 0 & \cdots & 0 \end{bmatrix}^T = \|\vec{x}\|_2 e_1$. (Thus $\|\vec{y}\| = \|\vec{x}\|$)
Let $\vec{v} = \vec{x} - \vec{y}$, and $\vec{u} = \frac{\vec{v}}{\|\vec{v}\|_2}$. Then $\mathbf{Q} = \mathbf{I} - 2\vec{u}\,\vec{u}^T$ satisfies: $\mathbf{Q}\vec{x} = \vec{y}$.

Proof:

$$\begin{aligned}
\mathbf{Q}\vec{x} = (\mathbf{I} - 2\vec{u}\,\vec{u}^T)\vec{x} &= \vec{x} - 2\langle \vec{u}, \vec{x} \rangle \vec{u} \\
&= \vec{x} - \frac{2}{(\|\vec{x} - \vec{y}\|_2)^2}((\vec{x} - \vec{y})^T \vec{x})(\vec{x} - \vec{y}) \\
&= \vec{x} - \frac{1}{(\|\vec{x} - \vec{y}\|_2)^2}(2(\|\vec{x}\|_2)^2 - 2\vec{y}^T\vec{x})(\vec{x} - \vec{y}) \\
&= \vec{x} - \frac{1}{(\|\vec{x} - \vec{y}\|_2)^2}((\|\vec{x}\|_2)^2 + (\|\vec{y}\|_2)^2 - 2\vec{y}^T\vec{x})(\vec{x} - \vec{y}) \\
&= \vec{x} - \frac{1}{(\|\vec{x} - \vec{y}\|_2)^2}(\|\vec{x} - \vec{y}\|_2)^2(\vec{x} - \vec{y}) \\
&= \vec{x} - \vec{x} + \vec{y} = \vec{y}
\end{aligned}$$

Now let's consider taking a full QR decomposition:

Consider $\mathbf{A} = \begin{bmatrix} \vec{a}_1 & \vec{a}_2 & \cdots & \vec{a}_m \end{bmatrix} \in \mathbb{R}^{n \times m}$.

Step 1. Compute a reflector $\mathbf{Q}_1$ such that $\mathbf{Q}_1 \vec{a}_1 = \|\vec{a}_1\|_2 e_1$. Thus:

$$\mathbf{Q}_1\mathbf{A} = \begin{bmatrix} \mathbf{Q}_1\vec{a}_1 & \mathbf{Q}_1\vec{a}_2 & \cdots & \mathbf{Q}_1\vec{a}_m \end{bmatrix} = \begin{bmatrix} \|\vec{a}_1\|_2 & \vec{b}^T \\ \vec{0} & \tilde{\mathbf{A}}_2 \end{bmatrix}$$

So, now consider $\tilde{\mathbf{A}}_2 = \begin{bmatrix} \tilde{a}_2 & \cdots & \tilde{a}_m \end{bmatrix} \in \mathbb{R}^{(n-1)\times(m-1)}$

Step 2. Now consider a reflector $\tilde{\mathbf{Q}}_2 \in \mathbb{R}^{(n-1),(m-1)}$ such that $\tilde{\mathbf{Q}}_2\tilde{a}_2 = \|\tilde{a}_2\|_2 e_1$.
Then define: $\mathbf{Q}_2 = \begin{bmatrix} \mathbf{I}_1 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Q}}_2 \end{bmatrix}$

Therefore, we have that $\mathbf{Q}_2\mathbf{Q}_1\mathbf{A} = \begin{bmatrix} \hat{\mathbf{R}} & \mathbf{B} \\ \mathbf{0} & \tilde{\mathbf{A}}_3 \end{bmatrix}$, where $\tilde{\mathbf{A}}_3 \in \mathbb{R}^{(n-2),(m-2)}$, $\hat{\mathbf{R}}$ is 2×2 upper triangular, and $\mathbf{B}$ is some rectangular matrix.

Note that only the lowest row in $\mathbf{B}$ is effected by this step.

Step 3. Repeat step 2. over and over again on $\tilde{\mathbf{Q}}_k$ until you can't any more. At the end, you will get an equation of the form:

$$\mathbf{Q}_m \cdots \mathbf{Q}_2\mathbf{Q}_1\mathbf{A} = \begin{bmatrix} \hat{\mathbf{R}} \\ \mathbf{0} \end{bmatrix}$$

Step 4. Each $\mathbf{Q}_i$ is orthogonal by the properties of a reflector. And since the product of orthogonal matrices is orthogonal, we can thus coallesce the $\mathbf{Q}$ in order to get that $\mathbf{A} = \mathbf{QR}$.

Lecture 13: 2/12/2024

Let $\mathbf{A} \in \mathbb{R}^{n\times m}$ with $\text{rank}(\mathbf{A}) = r$. Then there exists orthogonal matrices $\mathbf{U} \in \mathbb{R}^{n\times n}$ and $\mathbf{V} \in \mathbb{R}^{m\times m}$, as well as positive numbers $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ such that letting:

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_r & & \\ & & & \ddots & \\ & & & & 0 \end{bmatrix} \in \mathbb{R}^{n\times m},$$

we have that $\mathbf{A} = \mathbf{U\Sigma V}^T$.

$\mathbf{U\Sigma V}^T$ is called the SVD of $\mathbf{A}$. Also, each σ_i is called a singular value of $\mathbf{A}$.

Specifically, SVD stands for "Singular Value Decomposition".

Geometric Explanation:
There exists an orthonormal basis $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_m$ for $\mathbb{R}^m$, an orthonormal basis $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_n$ for $\mathbb{R}^n$, and real numbers $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ such that:

$$\mathbf{A}\vec{v}_i = \begin{cases} \sigma_i \vec{u}_i & \text{if } i \in \{1, \ldots, r\} \\ \vec{0} & \text{if } i \in \{r+1, \ldots, m\} \end{cases} \qquad \mathbf{A}^T\vec{u}_i = \begin{cases} \sigma_i \vec{v}_i & \text{if } i \in \{1, \ldots, r\} \\ \vec{0} & \text{if } i \in \{r+1, \ldots, n\} \end{cases}$$

To see this, let $\mathbf{U} = \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_n \end{bmatrix}$, and $\mathbf{V} = \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \cdots & \vec{v}_m \end{bmatrix}$. Also let $\mathbf{\Sigma}$ be a matrix defined as seen above.

$\mathbf{A} = \mathbf{U\Sigma V}^T \implies \mathbf{AV} = \mathbf{U\Sigma}$. Therefore, we have that:
$$\begin{bmatrix} \mathbf{A}\vec{v}_1 & \mathbf{A}\vec{v}_2 & \cdots & \mathbf{A}\vec{v}_m \end{bmatrix} = \begin{bmatrix} \sigma_1\vec{u}_1 & \sigma_2\vec{u}_2 & \cdots & \sigma_r\vec{u}_r & \vec{0} & \cdots & \vec{0} \end{bmatrix}$$

Meanwhile $\mathbf{A}^T = \mathbf{V\Sigma}^T\mathbf{U}^T$. Thus, $\mathbf{A}^T\mathbf{U} = \mathbf{V\Sigma}^T$, which means that:
$$\begin{bmatrix} \mathbf{A}^T\vec{u}_1 & \mathbf{A}^T\vec{u}_2 & \cdots & \mathbf{A}^T\vec{u}_m \end{bmatrix} = \begin{bmatrix} \sigma_1\vec{v}_1 & \sigma_2\vec{v}_2 & \cdots & \sigma_r\vec{v}_r & \vec{0} & \cdots & \vec{0} \end{bmatrix}$$

Important fact:

For any $\vec{w} \in \mathbb{R}^m$, we have that $\vec{w} = \sum_{i=1}^{m} c_i\vec{v}_i$. Therefore:

$$\mathbf{A}\vec{w} = \sum_{i=1}^{m} c_i(\mathbf{A}\vec{v}_i) = \sum_{i=1}^{r} c_i\sigma_i\vec{u}_i$$

Proposition:

(Note: In this class, the professor is writing $R(\mathbf{A})$ to refer to the "range" [a.k.a column space] of $\mathbf{A}$. I don't personally like it but I'm also not going to rewrite $R(\mathbf{A})$ every time he says it.)

$$R(\mathbf{A}) = \text{span}\{\vec{u}_1, \ldots, \vec{u}_r\} \qquad N(\mathbf{A}) = \text{span}\{\vec{v}_{r+1}, \ldots, \vec{v}_m\}$$
$$R(\mathbf{A}^T) = \text{span}\{\vec{v}_1, \ldots, \vec{v}_r\} \qquad N(\mathbf{A}^T) = \text{span}\{\vec{u}_{r+1}, \ldots, \vec{u}_n\}$$
$$R(\mathbf{A}^T) = N(\mathbf{A})^\perp \qquad R(\mathbf{A}) = N(\mathbf{A}^T)^\perp$$

Similarly to QR decompositions, we can also find a condensed / reduced SVD of a matrix $\mathbf{A}$...

Let $\mathbf{A} \in \mathbb{R}^{n\times m}$ with $\text{rank}(\mathbf{A}) = r$. Then there exists $\hat{\mathbf{U}} \in \mathbb{R}^{n\times r}$, $\hat{\mathbf{\Sigma}} \in \mathbb{R}^{r\times r}$, and $\hat{\mathbf{V}} \in \mathbb{R}^{m\times r}$ such that:

- $\hat{\mathbf{U}}^T\hat{\mathbf{U}} = \mathbf{I}_r$
- $\hat{\mathbf{V}}^T\hat{\mathbf{V}} = \mathbf{I}_r$
- $\hat{\mathbf{\Sigma}}$ is a diagonal square matrix with decreasing entries as one goes down the diagonal.
- $\mathbf{A} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^T$

One last note for this lecture:

We won't be learning how to calculate SVDs of matrices yet...

That said, know that the singular values of $\mathbf{A}$ are uniquely determined by $\mathbf{A}$.

Lecture 14: 2/14/2024

Theorem: For $\mathbf{A} \in \mathbb{R}^{n\times m}$, we have that $\|\mathbf{A}\|_2$ equals the largest singular value of $\mathbf{A}$.

Proof:

Let $\mathbf{U\Sigma V}^T$ be the SVD of $\mathbf{A}$ where $\mathbf{U} = \begin{bmatrix}\vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_n\end{bmatrix}$, $\mathbf{V} = \begin{bmatrix}\vec{v}_1 & \vec{v}_2 & \cdots & \vec{v}_m\end{bmatrix}$, and the singular values in $\mathbf{\Sigma}$ are $\sigma_1, \ldots, \sigma_r$.

From last lecture, we know that $\mathbf{A}\vec{v}_1 = \sigma_1\vec{u}_1$. So:

$$\|\mathbf{A}\|_2 = \max_{\vec{x}\neq 0} \frac{\|\mathbf{A}\vec{x}\|_2}{\|\vec{x}\|_2} \geq \frac{\|\mathbf{A}\vec{v}_1\|_2}{\|\vec{v}_1\|_2} = \frac{\sigma_1\|\vec{u}_1\|}{\|\vec{v}_1} = \frac{\sigma_1(1)}{(1)} = \sigma_1$$

Meanwhile, note that $\vec{x} \in \mathbb{R}^m$ can be expressed as a linear combination: $c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_m\vec{v}_m$ because the columns of $\mathbf{V}$ form a basis for $\mathbb{R}^m$. And, since $\vec{v}_i \cdot \vec{v}_j = 0$ for all $i \neq j$, we know that:

$$(\|\vec{x}\|_2)^2 = \vec{x}\cdot\vec{x} = \sum_{i=1}^{n} c_i^2(\vec{v}_i \cdot \vec{v}_i)$$

Also, because each $\|\vec{v}_i\|_2 = 1$, this further simplifies to: $(\|\vec{x}\|_2)^2 = \sum_{i=1}^{n} c_i^2$

At the same time:
$\mathbf{A}\vec{x} = \mathbf{A}(c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_m\vec{v}_m) = c_1\sigma_1\vec{u}_1 + c_2\sigma_2\vec{u}_2 + \ldots + c_r\sigma_r\vec{u}_r$.
And as each u_i is a unit vector and $u_i \cdot u_j = 0$ for all $i \neq j$, we thus have that:

$$(\|\mathbf{A}\vec{x}\|_2)^2 = \sum_{i=1}^{r} c_i^2\sigma_i^2$$

Then because σ_1 is the largest singular value, we have that:

$$\sum_{i=1}^{r} c_i^2\sigma_i^2 \leq \sum_{i=1}^{r} c_i^2\sigma_1^2 = \sigma_1^2\sum_{i=1}^{r} c_i^2 = \sigma_1^2(\|\vec{x}\|_2)^2$$

Thus, $(\|\mathbf{A}\vec{x}\|_2)^2 \leq \sigma_1^2(\|\vec{x}\|_2)^2$. And since σ_1, $\|\mathbf{A}\vec{x}\|$, $\|\vec{x}\|$ are all positive, we have that $\|\mathbf{A}\vec{x}\|_2 \leq \sigma_1\|\vec{x}\|_2$. Therefore:

$$\|\mathbf{A}\|_2 = \max_{\vec{x}\neq 0} \frac{\|\mathbf{A}\vec{x}\|_2}{\|\vec{x}\|_2} \leq \max_{\vec{x}\neq 0} \frac{\sigma_1\|\vec{x}\|_2}{\|\vec{x}\|_2} = \sigma_1$$

Two notes:
- On page 12, it was mentioned that $\|\mathbf{A}\|_2 = \sqrt{\lambda_{\max}(\mathbf{A}^T\mathbf{A})}$. So, consider that a hint about how SVDs are calculated.
- $\mathbf{A}^T$ and $\mathbf{A}$ have the same singular values. So $\|\mathbf{A}\|_2 = \|\mathbf{A}^T\|_2$.

Theorem: Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be an invertible matrix with singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n > 0$. Then:

$$K_2(\mathbf{A}) = \|\mathbf{A}\|_2\|\mathbf{A}^{-1}\|_2 = \frac{\sigma_1}{\sigma_n}$$

Proof:
Let $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ be the SVD of $\mathbf{A}$. Note that as $\mathbf{A}$ is invertible (and square), we know that $\mathbf{\Sigma}$ is a diagonal square matrix which constains n strictly positive singular values. So, $\mathbf{\Sigma}$ is invertible.

Now note: $\mathbf{A}^{-1} = (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^{-1} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T$. Thus, we see that an SVD of $\mathbf{A}^{-1}$ is $\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T$. Additionally, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n > 0$ implies that $\frac{1}{\sigma_n} \geq \ldots \geq \frac{1}{\sigma_2} \geq \frac{1}{\sigma_1} > 0$. So the largest singular value in $\mathbf{\Sigma}^{-1}$ is $\frac{1}{\sigma_n}$.
We therefore conclude that $\|\mathbf{A}^{-1}\|_2 = \frac{1}{\sigma_n}$.

Hence, $\|\mathbf{A}\|_2\|\mathbf{A}^{-1}\|_2 = \sigma_1\frac{1}{\sigma_n}$.

Problem: For $\mathbf{A} \in \mathbb{R}^{n \times n}$ and an integer $k > 0$, find the matrix X with rank k that minimizes $\|\mathbf{A} - \mathbf{X}\|_2$.

To do this, first consider the following lemma:
Let $\mathbf{A} \in \mathbb{R}^{n \times m}$ be a matrix with $\text{rank}(\mathbf{A}) = r$. Consider the singular values of $\mathbf{A}$: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, with their respective right and left singular vectors: $\vec{v}_1, \ldots, \vec{v}_r$ and $\vec{u}_1, \ldots, \vec{u}_r$. Then, we can write that:

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \, \vec{u}_i \, \vec{v}_i^{\,T}$$

Proof:

$$\mathbf{A} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^T = \begin{bmatrix} \vec{u}_1 & \ldots & \vec{u}_r \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix} \begin{bmatrix} \vec{v}_1^{\,T} \\ \vdots \\ \vec{v}_r^{\,T} \end{bmatrix}$$

$$= \begin{bmatrix} \sigma_1 \vec{u}_1 & \ldots & \sigma_r \vec{u}_r \end{bmatrix} \begin{bmatrix} \vec{v}_1^{\,T} \\ \vdots \\ \vec{v}_r^{\,T} \end{bmatrix}$$

$$= \sigma_1 \vec{u}_1 \vec{v}_1^{\,T} + \ldots + \sigma_r \vec{u}_r \vec{v}_r^{\,T}$$

In other words, $\mathbf{A}$ is a linear combination of r many rank 1 matrices.

If $\mathbf{A}$ has the SVD: $\mathbf{A} = \sum_{i=1}^{r} \sigma_i \, \vec{u}_i \, \vec{v}_i^{\,T}$, then the optimal matrix $\mathbf{X}$ is $\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \, \vec{u}_i \, \vec{v}_i^{\,T}$.

Specifically, $\mathbf{A}_k$ is the rank-k approximation of $\mathbf{A}$.

Proposition: $\|\mathbf{A} - \mathbf{A}_k\|_2 = \sigma_{k+1}$

Proof: $\mathbf{A} - \mathbf{A}_k = \sum_{i=k+1}^{r} \sigma_i \, \vec{u}_i \, \vec{v}_i^{\,T}$.

Thus, the largest singular value of $\mathbf{A} - \mathbf{A}_k$ is σ_{k+1}.

Lecture 15: 2/16/2024

The least square problem revisited:
We previously covered a method using the QR decomposition of $\mathbf{A}$. But that required $\mathbf{A}$ to be full rank. Now, here is a method using an SVD of $\mathbf{A}$ which works even if $\mathbf{A}$ is not full rank.

Let $\mathbf{A} \in \mathbb{R}^{n \times m}$ where $n \geq m$.

Suppose $\mathbf{A}$ has an SVD: $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. Then:
$(\| \vec{b} - \mathbf{A}\vec{x}\|_2)^2 = (\| \vec{b} - \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\vec{x}\|_2)^2 = (\|\mathbf{U}^T\vec{b} - \mathbf{\Sigma}\mathbf{V}^T\vec{x}\|_2)^2$

As a reminder, $\mathbf{U}$ is an orthogonal matrix and so $\mathbf{U}\vec{x}\cdot\mathbf{U}\vec{y} = \vec{x}\cdot\vec{y}$.

Thus, we get that: $(\| \vec{b} - \mathbf{A}\vec{x}\|_2)^2 = \left(\left\| \begin{bmatrix} \vec{c} \\ \vec{d} \end{bmatrix} - \begin{bmatrix} \hat{\mathbf{\Sigma}}\vec{y} \\ \vec{0} \end{bmatrix} \right\|_2\right)^2$

...where $\vec{y}$ is the vector containing the first r elements of $\mathbf{V}^T\vec{x}$ with $r = \text{rank}(\mathbf{A})$ and $\mathbf{U}^T\vec{b} = \begin{bmatrix} \vec{c} \\ \vec{d} \end{bmatrix}$ where $\vec{c} \in \mathbb{R}^r$. Also, $\mathbf{\Sigma} = \begin{bmatrix} \hat{\mathbf{\Sigma}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$ such that $\hat{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$.

This can be rewritten as $(\| \vec{b} - \mathbf{A}\vec{x}\|_2)^2 = (\| \vec{c} - \hat{\mathbf{\Sigma}}\vec{y}\|_2)^2 + (\| \vec{d}\|_2)^2$. $\vec{d}$ is independent from $\vec{x}$. So we need only worry about minimizing $(\| \vec{c} - \hat{\mathbf{\Sigma}}\vec{y}\|_2)^2$ in order to minimize $(\| \vec{b} - \mathbf{A}\vec{x}\|_2)^2$. Luckily, $\hat{\mathbf{\Sigma}}$ is invertible since it is a diagonal matrix with only positive entries. So, we can solve for $\vec{y}$ such that $\vec{c} - \hat{\mathbf{\Sigma}}\vec{y} = \vec{0}$

Therefore, any solution of the least squares problem of $\mathbf{A}$ and $\vec{b}$ satisfies that: $\mathbf{V}^T\vec{x} = \begin{bmatrix} \vec{y} \\ \vec{z} \end{bmatrix}$ where $\vec{z} \in \mathbb{R}^{m-r}$ is any vector.

Because V^T is orthogonal, we can thus say that $\vec{x} = \mathbf{V}\begin{bmatrix} \vec{y} \\ \vec{z} \end{bmatrix}$. So the set of solutions to the least square problem is:

$$\left\{ \mathbf{V}\begin{bmatrix} \vec{y} \\ \vec{z} \end{bmatrix} \quad \middle| \quad \vec{z} \in \mathbb{R}^{m-r} \right\}$$

Note: One important solution is the solution that minimizes $\| \vec{x}\|_2$. To get this, note that $\|\mathbf{V}\vec{v}\|_2 = \| \vec{v}\|_2$ because $\mathbf{V}$ is orthogonal. So, the solution to the least square problem that minimizes $\| \vec{x}\|_2$ is $\vec{x} = \mathbf{V}\begin{bmatrix} \vec{y} \\ \vec{0} \end{bmatrix}$.

If $\mathbf{A}$ is not a square matrix, there is no inverse. A pseudo inverse is a generalization that works for all matrices.

Note that if $\mathbf{A}$ has the SVD: $\mathbf{U\Sigma V}^T$, and is invertible, then $\mathbf{A}^{-1} = \mathbf{V\Sigma}^{-1}\mathbf{U}^T$ where:

$$\mathbf{\Sigma}^{-1} = \begin{bmatrix} \frac{1}{\sigma_1} & & \\ & \ddots & \\ & & \frac{1}{\sigma_n} \end{bmatrix}$$

Importantly, no matter how $\mathbf{A}$ isn't invertible (whether that be because it is rank deficient or non-square), the expression $\mathbf{U\Sigma V}^T$ is mostly well defined. The only problem that arrises if $\mathbf{A}$ is not invertible is that $\mathbf{\Sigma}$ is not invertible anymore either.

Given $\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \\ & & \end{bmatrix} \in \mathbb{R}^{n\times m}$, define $\mathbf{\Sigma}^+ = \begin{bmatrix} \frac{1}{\sigma_1} & & & \\ & \ddots & & \\ & & \frac{1}{\sigma_r} & \end{bmatrix} \in \mathbb{R}^{m\times n}$

Then, define $\mathbf{A}^+ = \mathbf{V\Sigma}^+\mathbf{U}^T \in \mathbb{R}^{m\times n}$.

Note that if $\mathbf{A}$ is invertible, then $\mathbf{A}^+ = \mathbf{A}^{-1}$.

Theorem: If $\mathbf{A} \in \mathbb{R}^{n\times m}$ where $n \geq m$, then the solution $\vec{x}$ to the least square problem of $\mathbf{A}$ and $\vec{b}$ such that $\|\vec{x}\|_2$ is minimized is given by $\vec{x} = \mathbf{A}^+\vec{b}$.

Why: (recall how $\vec{c}$, $\vec{d}$, and $\vec{y}$ were defined on the previous page...)

- $\mathbf{U}^T\vec{b} = \begin{bmatrix} \vec{c} \\ \vec{d} \end{bmatrix}$
- $\mathbf{\Sigma}^+ \begin{bmatrix} \vec{c} \\ \vec{d} \end{bmatrix} = \begin{bmatrix} \vec{y} \\ \vec{0} \end{bmatrix}$

Lecture 17: 2/23/2024

Suppose $\mathbf{A} \in \mathbb{R}^{n\times n}$. Then $\mathbf{A}$ is called semisimple if $\mathbf{A}$ has n linearly independent eigenvectors. Otherwise, $\mathbf{A}$ is called defective.

Power method:

Assume $\mathbf{A} \in \mathbb{R}^{n\times n}$ is semisimple. That way, there are n linearly independent eigenvectors: $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$, which form a basis for $\mathbb{R}^n$. Associate $\vec{v}_i$ with the eigenvalue λ_i and order the eigenvalues such that $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$.

Also assume $|\lambda_1| > |\lambda_2|$. Otherwise, this algorithm will not converge.

Pre iteration steps:

Start with a random $\vec{q}_0 \in \mathbb{R}^n$. Let s_0 equal the first entry of $\vec{q}_0$ whose absolute value equals $\|\vec{q}_0\|_\infty$. Then define $\vec{q}_0' = \frac{1}{s_0}\vec{q}_0$.

Iterative steps:

For $j \geq 0$, define:

- $\vec{q}_{j+1} = \mathbf{A}\,\vec{q}_j'$
- s_{j+1} equals the first entry of $\vec{q}_{j+1}$ whose absolute value equals $\|\vec{q}_{j+1}\|_\infty$
- $\vec{q}_{j+1}' = \frac{1}{s_{j+1}}\vec{q}_{j+1}$.

Note:
Dividing by s_{j+1} makes all entries of $\vec{q}_{j+1}'$ have an absolute value of at most 1. Also, to be clear what the algorithm is saying s_{j+1} equals:

$$\vec{q}_{j+1} = \begin{bmatrix} -1 \\ 0 \\ 1/2 \\ 1 \\ 0 \end{bmatrix} \Longrightarrow s_{j+1} = -1 \text{ while } \vec{q}_{j+1} = \begin{bmatrix} 1 \\ 0 \\ 1/2 \\ -1 \\ 0 \end{bmatrix} \Longrightarrow s_{j+1} = 1$$

.

Theorem: If the conditions assumed on the previous page about $\mathbf{A}$ are true, then:

$s_j \longrightarrow$ the dominant eigenvalue: λ_1
$\vec{q}_j \longrightarrow$ a dominant eigenvector: $\vec{v}_1$

Intuition why:
As the algorithm converges, we will get that $\vec{q}_{j+1} = \frac{1}{s_j}\mathbf{A}\,\vec{q}_j \approx \vec{q}_j$. Or in other words, $\mathbf{A}\,\vec{q}_j \approx s_j\,\vec{q}_j$. This indicates that s_j is approaching an eigenvalue of $\mathbf{A}$ and $\vec{q}_j$ is approaching an eigenvector of $\mathbf{A}$.

As for why $\vec{q}_j$ and s_j are specifically approaching the dominant eigenvector and eigenvalue respectively, note the following:

We know that there exists constants $c_1, c_2, \ldots c_n$ such that $\vec{q} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_n\vec{v}_n$. Thus, we then have that:

$$\begin{aligned} \mathbf{A}^j\vec{q} &= c_1\lambda_1^j\vec{v}_1 + c_2\lambda_2^j\vec{v}_2 + \ldots + c_n\lambda_n^j\vec{v}_n \\ &= \lambda_1^j\left(c_1\vec{v}_1 + c_2\left(\tfrac{\lambda_2}{\lambda_1}\right)^j\vec{v}_2 + \ldots + c_n\left(\tfrac{\lambda_n}{\lambda_1}\right)^j\vec{v}_n\right) \end{aligned}$$

Now because we assumed $\lambda_1 > \lambda_i$ for all $i > 1$, we can assume that each $\left(\frac{\lambda_i}{\lambda_1}\right)^j$ will approach 0 as j gets bigger. Therefore, for large j we have that:

$$\mathbf{A}^j\vec{q} \approx \lambda_1^j c_1\vec{v}_1$$

Now $\vec{q}_j$ is just $\mathbf{A}^j\vec{q}$ scaled by some constant.

The reason why we rescale $\vec{q}_j$ in every iteration of the algorithm is because if we didn't, then $\|\vec{q}_j\|$ would either grow without bound or approach 0. Hence, our algorithm would either not converge or would converge to a trivial eigenvector: $\vec{0}$.

So for large j, we have that $\vec{q}_j \approx d\,\vec{v}_1$ where d is some constant. Therefore, we've shown that $\vec{q}_j$ will approach being the dominant eigenvector.

It follows that since s_j approaches the eigenvalue associated with the dominant eigenvector, s_j approaches being the dominant eigenvalue.

Three more notes about the power method:

If $\left|\frac{\lambda_2}{\lambda_1}\right|$ is close to 0, than the power method will converge quickly.

If $\left|\frac{\lambda_2}{\lambda_1}\right|$ is cloes to 1, than the power method will converge slowly.

The power method takes $\mathcal{O}(n^2 N)$ flops where $\mathbf{A} \in \mathbb{R}^{n\times n}$ and N is the number of iterations we do.

Lecture 18: 2/26/2024

Let λ be an eigenvalue of a matrix $\mathbf{A}$.

- Its geometric multiplicity is $\dim\left(\ker\left(\lambda\mathbf{I}-\mathbf{A}\right)\right)$.

  $\ker(\mathbf{A})$ is a common way of referring to the null space of $\mathbf{A}$.

- Its algebraic multiplicity is its multiplicity as a root of the equation: $\det(\lambda\mathbf{I}-\mathbf{A})=0$

Suppose $\mathbf{A},\mathbf{B}\in\mathbb{R}^{n\times n}$. They are similar if there exists an invertible matrix $\mathbf{S}\in\mathbb{R}^{n\times n}$ such that $\mathbf{AS}=\mathbf{SB}$ (or in other words: $\mathbf{A}=\mathbf{SBS}^{-1}$).

Furthermore, $\mathbf{A}$ and $\mathbf{B}$ are orthogonally similar if $\mathbf{S}$ is orthogonal.

Theorem: If $\mathbf{A}$, $\mathbf{B}$ are similar, then they have the same eigenvalues.

Proof:

$$\mathbf{A}\,\vec{v}=\lambda_A\,\vec{v}\implies\mathbf{SBS}^{-1}\,\vec{v}=\lambda_A\,\vec{v}\implies\mathbf{B}\left(\mathbf{S}^{-1}\,\vec{v}\right)=\lambda_A\left(\mathbf{S}^{-1}\,\vec{v}\right)$$
$$\mathbf{B}\,\vec{v}=\lambda_B\,\vec{v}\implies\mathbf{S}^{-1}\mathbf{AS}\,\vec{v}=\lambda_B\,\vec{v}\implies\mathbf{A}\left(\mathbf{S}\,\vec{v}\right)=\lambda_B\left(\mathbf{S}\,\vec{v}\right)$$

Observation: If $\mathbf{B}=\mathbf{S}^{-1}\mathbf{AS}$ and $\vec{v}$ is an eigenvector of $\mathbf{A}$ with respect to the eigenvalue λ, then $\mathbf{S}^{-1}\,\vec{v}$ is an eigenvector of $\mathbf{B}$ with respect to λ.

Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ be a complex square matrix.

- $\overline{\mathbf{A}}$ refers to the matrix whose elements are the complex conjugates of the elements of $\mathbf{A}$.
- $\mathbf{A}^* = \overline{\mathbf{A}}^T$ is the conjugate transpose of $\mathbf{A}$.

This is a complex generalizations of transposes.

$\mathbf{U} \in \mathbb{C}^{n\times n}$ is called unitary if $\mathbf{U}\mathbf{U}^* = \mathbf{U}^*\mathbf{U} = \mathbf{I}$.

This is a complex generalizations of orthogonality.

Schur Theorem: Let $\mathbf{A} \in \mathbb{C}^{n\times n}$. Then there exists a unitary matrix $\mathbf{U} \in \mathbb{C}^{n\times n}$ and an upper triangular matrix $\mathbf{T} \in \mathbb{C}^{n\times n}$ such that $\mathbf{A} = \mathbf{U}\mathbf{T}\mathbf{U}^*$.

Real Schur Theorem: If $\mathbf{A} \in \mathbb{R}^{n\times n}$, then there exists an orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and "almost" upper triangular matrix $\mathbf{T} \in \mathbb{R}^{n\times n}$ such that $\mathbf{A} = \mathbf{Q}\mathbf{T}\mathbf{Q}^T$.

Because I'm on a time crunch, these next lectures will be out of order. Also I'm probably never going to write down anything for lecture 18.

Lecture 21: 3/8/2024

Iterative Methods:

When solving $\mathbf{A}\vec{x} = \vec{b}$, Gaussian Elimination takes $\mathcal{O}(n^3)$ flops. When n is large, this takes too long.

Thus we instead consider iterative methods:

Step 1. Pick an inital guess $\vec{x}^{(0)}$.

Step 2. Iterate and hope that $\vec{x}^{(k)} \to \vec{x}$ where $\vec{x}$ is the solution to $\mathbf{A}\vec{X} = \vec{b}$.

Step 3. Choose $\varepsilon > 0$ and stop if $\|\vec{x}^{(k+1)} - \vec{x}^{(k)}\| \leq \varepsilon$

The advantages of iterative methods are:

- If we choose a good first guess: $\vec{x}^{(0)}$, then our iterative method might converge fast.
- Iterative methods in general take $\mathcal{O}(n^2 N)$ flops where N is the number of iterations you need to do. So when $N \ll n$, this can save a lot of time.

The Jacobi method:

Let $\mathbf{A} \in \mathbb{C}^{n \times n}$ and $\vec{b}, \vec{x} \in \mathbb{C}^n$.

Here is the motivation for the algorithm:

$$\mathbf{A}\vec{x} = \vec{b} \iff \begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n = b_2 \\ \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + n_{n,n}x_n = b_n \end{cases}$$

Equivalently, we can say that $a_{i,i}x_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{i,j}x_j = b_i$ for $i \in \{1, 2, \ldots, n\}$.

Assuming all $a_{i,i} \neq 0$, then for all $i \in \{1, 2, \ldots, n\}$ we can isolate x_i as:

$$x_i = \tfrac{1}{a_{i,i}}(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{i,j}x_j)$$

Hence, we now have the formula used in the procedure below.

The procedure:

Pick any initial guess $\vec{x}^{\,(0)} = \vec{q}$. Then inductively for $k \geq 0$, define:

$\vec{x}^{\,(k+1)} = \begin{bmatrix} x_1^{(k+1)} & x_2^{(k+1)} & \cdots & x_n^{(k+1)} \end{bmatrix}$ such that for $i \in \{1, 2, \ldots, n\}$:

$$x_i^{(k+1)} = \tfrac{1}{a_{i,i}}(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{i,j}x_j^{(k)})$$

Beware, for all i, we must have that $a_{i,i} \neq 0$ or else the formula in the algorithm isn't defined.

Now let's try to rewrite this algorithm using matrices. Firstly, let $\mathbf{D}$ be the diagonal matrix consisting of just the diagonal entries of $\mathbf{A}$. Then, we can say that:

$$\vec{x}^{\,(k+1)} = \mathbf{D}^{-1}(\vec{b} - (\mathbf{A} - \mathbf{D})\vec{x}^{\,(k)})$$

With this formula, we can then figure out when $\vec{x}^{\,(k)}$ converges to the real answer.

Let $\vec{x}$ be the solution to $\mathbf{A}\vec{x} = \vec{b}$ and denote the error: $\vec{e}^{\,(k)} = \vec{x} - \vec{x}^{\,(k)}$.

$$\begin{aligned}\vec{e}^{(k+1)} = \vec{x} - \vec{x}^{(k+1)} &= \vec{x} - \mathbf{D}^{-1}(\vec{b} - (\mathbf{A}-\mathbf{D})\vec{x}^{(k)}) \\ &= \vec{x} - \mathbf{D}^{-1}(\vec{b} - \mathbf{A}\vec{x}^{(k)} + \mathbf{D}\vec{x}^{(k)}) \\ &= \vec{x} - \mathbf{D}^{-1}((\mathbf{A}\vec{x}) - \mathbf{A}\vec{x}^{(k)}) - \vec{x}^{(k)} \\ &= \vec{e}^{(k)} - \mathbf{D}^{-1}\mathbf{A}\vec{e}^{(k)} \\ &= (\mathbf{I} - \mathbf{D}^{-1}\mathbf{A})\vec{e}^{(k)}\end{aligned}$$

As a result, we have that $\vec{e}^{(k+1)} = (\mathbf{I} - \mathbf{D}^{-1}\mathbf{A})^{k+1}\vec{e}^{(0)}$

Theorem: If all eigenvalues λ of $\mathbf{B} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$ satisfy $|\lambda| < 1$, then the Jacobi method will converge to $\vec{x}$ for every choice of $\vec{x}^{(0)}$

Lecture 22: 3/11/2024

The Gauss-Seidel method:

Still let $\mathbf{A} \in \mathbb{C}^{n\times n}$ and $\vec{b}, \vec{x} \in \mathbb{C}^n$. Also define $\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}$ where $\mathbf{D}$ consists of the diagonal entries of $\mathbf{A}$, $\mathbf{L}$ consists of the entries below the diagonal of $\mathbf{A}$, and $\mathbf{U}$ consists of the entries above the diagonal of $\mathbf{A}$.

For the Jacobi method, we defined:

$$\vec{x}^{(k+1)} = \mathbf{D}^{-1}(\vec{b} - (\mathbf{A}-\mathbf{D})\vec{x}^{(k)}) = \mathbf{D}^{-1}(\vec{b} - \mathbf{L}\vec{x}^{(k)} - \mathbf{U}\vec{x}^{(k)})$$

Now for the Gauss-Seidel method, we shall instead define:

$$\vec{x}^{(k+1)} = \mathbf{D}^{-1}(\vec{b} - \mathbf{L}\vec{x}^{(k+1)} - \mathbf{U}\vec{x}^{(k)})$$

Then $\mathbf{D}\vec{x}^{(k+1)} = \vec{b} - \mathbf{L}\vec{x}^{(k+1)} - \mathbf{U}\vec{x}^{(k)}$, which means that:

$$\vec{x}^{(k+1)} = (\mathbf{D}+\mathbf{L})^{-1}\left(\vec{b} - \mathbf{U}\vec{x}^{(k)}\right)$$

Beware, $(\mathbf{D}+\mathbf{L})^{-1}$ only exists if all $a_{i,i} \neq 0$ on the diagonal of $\mathbf{A}$.

Now to analyze the convergence of this algorithm, once again let $\vec{x}$ be the solution to $\mathbf{A}\vec{x} = \vec{b}$ and denote the error $\vec{e}^{(k)} = \vec{x} - \vec{x}^{(k)}$. Then:

$$\begin{aligned}\vec{e}^{(k+1)} = \vec{x} - \vec{x}^{(k+1)} &= \vec{x} - (\mathbf{D}+\mathbf{L})^{-1}\left(\vec{b} - \mathbf{U}\vec{x}^{(k)}\right) \\ &= \vec{x} - (\mathbf{D}+\mathbf{L})^{-1}\left(\vec{b} - (\mathbf{A}-\mathbf{L}-\mathbf{D})\vec{x}^{(k)}\right) \\ &= \vec{x} - (\mathbf{D}+\mathbf{L})^{-1}(\mathbf{A}\vec{x} - \mathbf{A}\vec{x}^{(k)}) - \vec{x}^{(k)} \\ &= \vec{e}^{(k)} - (\mathbf{D}+\mathbf{L})^{-1}\mathbf{A}\vec{e}^{(k)} \\ &= \left(\mathbf{I} - (\mathbf{D}+\mathbf{L})^{-1}\mathbf{A}\right)\vec{e}^{(k)}\end{aligned}$$

Thus $\vec{e}^{(k+1)} = (\mathbf{I} - (\mathbf{D}+\mathbf{L})^{-1}\mathbf{A})^{k+1}\vec{e}^{(0)}$

Theorem: If all eigenvalues λ of $\mathbf{B} = \mathbf{I} - (\mathbf{D} + \mathbf{L})^{-1}\mathbf{A}$ satisfy $|\lambda| < 1$, then the Gauss-Seidel method will converge to $\vec{x}$ for every choice of $\vec{x}^{\,(0)}$

Lecture 20: 3/6/2024

Suppose $\mathbf{A} \in \mathbb{R}^{n \times m}$ has the SVD $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$.

Then $\mathbf{A}\mathbf{A}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T$, which means $\mathbf{A}\mathbf{A}^T\mathbf{U} = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T$.

Now write $\mathbf{U}$ as $\begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_n \end{bmatrix}$. Then if there are $r \leq n$ singular values, we have that for $i \in \{1, \ldots, r\}$, $\mathbf{A}\mathbf{A}^T\vec{u}_i = \sigma_i^2\vec{u}_i$, whereas for $i \in \{r+1, \ldots, n\}$, $\mathbf{A}\mathbf{A}^T\vec{u}_i = 0\,\vec{u}_i$. Hence, $\vec{u}_i$ is the eigenvector of $\mathbf{A}\mathbf{A}^T$ with length 1 associated with the eigenvalue σ_i^2 or 0 of $\mathbf{A}\mathbf{A}^T$.

Meanwhile, $\mathbf{A}^T\mathbf{A} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T$, which means $\mathbf{A}^T\mathbf{A}\mathbf{V} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}$.

Similar to before, setting $\mathbf{V}$ as $\begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \cdots & \vec{v}_n \end{bmatrix}$, we have that for $i \in \{1, \ldots, r\}$, $\mathbf{A}^T\mathbf{A}\vec{v}_i = \sigma_i^2\vec{v}_i$, whereas for $i \in \{r+1, \ldots, m\}$, $\mathbf{A}^T\mathbf{A}\vec{v}_i = 0\,\vec{v}_i$. Hence, $\vec{v}_i$ is the eigenvector of $\mathbf{A}^T\mathbf{A}$ with length 1 associated with the eigenvalue σ_i^2 or 0 of $\mathbf{A}^T\mathbf{A}$.

Because $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ are symmetric, we know that the eigenvectors of both form an orthonormal basis for their respective vector spaces.

Thus, method 1 for finding an SVD of $\mathbf{A}$ is by finding the eigenvalues and eigenvectors of $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$.

- This method has a good flop count.
- $K_2(\mathbf{A}\mathbf{A}^T) = (K_2(\mathbf{A}))^2$. So $\mathbf{A}\mathbf{A}^T$ is not well conditioned.

Here is another method for finding the SVD of $\mathbf{A} \in \mathbb{R}^{n \times m}$

Let $\mathbf{X} = \begin{bmatrix} \mathbf{0}_n & \mathbf{A} \\ \mathbf{A^T} & \mathbf{0}_m \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$ where $\mathbf{0}_k$ is the $k \times k$ zero matrix.

For $1 \leq i \leq r$ where r is the number of singular values, we claim that, $\mathbf{A}$ will have the eigenpairs:

$$(\sigma_i, \begin{bmatrix} \vec{u}_i \\ \vec{v}_i \end{bmatrix}) \text{ and } (-\sigma_i, \begin{bmatrix} \vec{u}_i \\ -\vec{v}_i \end{bmatrix})$$

Proof:
For $i \leq r$, $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \implies \mathbf{U}^T\mathbf{A} = \mathbf{\Sigma}\mathbf{V}^T$. So we have that $\mathbf{A}\,\vec{u}_i = \sigma_i\,\vec{v}_i$.
Also $\mathbf{A}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T \implies \mathbf{V}^T\mathbf{A}^T = \mathbf{\Sigma}^T\mathbf{U}^T$. So $\mathbf{A}^T\,\vec{v}_i = \sigma_i\,\vec{u}_i$.

So we have that (for $i \leq r$):

$$\mathbf{X}\begin{bmatrix}\vec{u}_i\\ \vec{v}_i\end{bmatrix} = \begin{bmatrix}\mathbf{0}_n & \mathbf{A}\\ \mathbf{A}^T & \mathbf{0}_m\end{bmatrix}\begin{bmatrix}\vec{u}_i\\ \vec{v}_i\end{bmatrix} = \begin{bmatrix}\mathbf{A}\,\vec{v}_i\\ \mathbf{A}^T\,\vec{u}_i\end{bmatrix} = \begin{bmatrix}\sigma_i\,\vec{u}_i\\ \sigma_i\,\vec{v}_i\end{bmatrix} = \sigma_i\begin{bmatrix}\vec{u}_i\\ \vec{v}_i\end{bmatrix}$$

$$\mathbf{X}\begin{bmatrix}\vec{u}_i\\ -\vec{v}_i\end{bmatrix} = \begin{bmatrix}\mathbf{0}_n & \mathbf{A}\\ \mathbf{A}^T & \mathbf{0}_m\end{bmatrix}\begin{bmatrix}\vec{u}_i\\ -\vec{v}_i\end{bmatrix} = \begin{bmatrix}-\mathbf{A}\,\vec{v}_i\\ \mathbf{A}^T\,\vec{u}_i\end{bmatrix} = \begin{bmatrix}-\sigma_i\,\vec{u}_i\\ \sigma_i\,\vec{v}_i\end{bmatrix} = -\sigma_i\begin{bmatrix}\vec{u}_i\\ -\vec{v}_i\end{bmatrix}$$

Meanwhile for $i > r$, $\mathbf{X}$ will have the eigenpairs: $(0, \begin{bmatrix}\vec{u}_i\\ \vec{0}\end{bmatrix})$ and $(0, \begin{bmatrix}\vec{0}\\ \vec{v}_i\end{bmatrix})$

For $i > r$, $\mathbf{A}\,\vec{u}_i = \vec{0}$ and $\mathbf{A}^T\,\vec{v}_i = \vec{0}$. So we can use similar reasoning as above to prove these eigenvalue-eigenvector pairings.

In total, we have $2r + (n-r) + (m-r) = n + m$ eigenvalues of $\mathbf{X}$.

So, method 2 for finding an SVD of $\mathbf{A}$ is to solve for the eigenvalues and eigenvectors of $\mathbf{X}$.

Issue: the eigenvalues of $\mathbf{X}$ come in pairs. Thus, algorithms for caculating the eigenvalues of $\mathbf{X}$ such as the power method will fail.

Common solution: calculate the eigenvalues of $\mathbf{X} + \mu\mathbf{I}$ where μ is a random shift. Then subtract μ from each eigenvalue you find.
This works because if $\mathbf{X}\,\vec{v} = \lambda\,\vec{v}$. Then $(\mathbf{X} + \mu\mathbf{I})\,\vec{v} = \mathbf{X}\,\vec{v} + \mu\,\vec{v} = (\lambda + \mu)\,\vec{v}$. Thus adding $\mu\mathbf{I}$ to $\mathbf{X}$ has the effect of adding μ to all of $\mathbf{X}$'s eigenvalues.

This doesn't always fix the problem but assuming $\sigma_1 \neq \sigma_2$, then at least there will be a dominant eigenvalue of $\mathbf{X} + \mu\mathbf{I}$.

Lecture 19: 2/28/2024

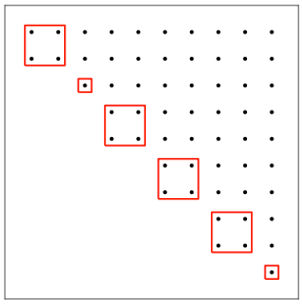
QR-iteration: (say we are given a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$)
Let $\mathbf{A}_0 = \mathbf{A}$.
For $k = 0, 1, \ldots$
Compute a QR decomposition of $\mathbf{A}_k$: $\mathbf{A}_k = \mathbf{Q}_{k+1}\mathbf{R}_{k+1}$.
Then define $\mathbf{A}_{k+1} = \mathbf{R}_{k+1}\mathbf{Q}_{k+1}$

Just by sight, this is a very expensive algorithm flops wise. So what's the point?

$\mathbf{A}_{k+1} = \mathbf{Q}_{k+1}^T \mathbf{A}_k \mathbf{Q}_{k+1}$. Thus $\mathbf{A}_{k+1}$ and $\mathbf{A}_k$ are orthogonally similar, which means they have the same eigenvalues as each other.

Theorem: If $\mathbf{A} \in \mathbb{R}^{n \times n}$ has n distinct eigenvalues and $\mathbf{A} = \mathbf{Q}\mathbf{T}\mathbf{Q}^T$ is the real Schur decomposition of $\mathbf{A}$, then $\lim_{k \to \infty} \mathbf{A}_k = \mathbf{T}$.

We're not going to prove convergence here. However, this theorem is useful because the eigenvalues of $\mathbf{T}$ are easy to compute and $\mathbf{T}$ is similar to $\mathbf{A}$. This is because $\mathbf{T}$ will be block-upper-triangular with (2×2) and (1×1) blocks.



(I did not make this image. Rather it was taken from the class textbook on page 235.)

We are not going to prove this but the eigenvalues of $\mathbf{T}$ are the eigenvalues of each block on the diagonal of $\mathbf{T}$.

Still, the above algorithm sucks. If the eigenvalues of $\mathbf{A}$ are close to each other, it converges really slowly. Plus, taking a QR decomposition every iteration is expensive. So how do we improve this?

Idea:

$\mathbf{H} \in \mathbb{R}^{n \times n}$ is called upper-Hessenberg if $h_{i,j} = 0$ for $i > j + 1$.

You can think of this as a weaker version of being upper-triangular.

Theorem: Every $\mathbf{A} \in \mathbb{R}^{n \times n}$ can be decomposed as $\mathbf{A} = \mathbf{Q}\mathbf{H}\mathbf{Q}^T$ where $\mathbf{Q}$ is orthogonal and $\mathbf{H}$ is upper-Hessenberg.

Idea of the algorithm: consider $\mathbf{A} = \begin{bmatrix} a_{1,1} & \vec{y}^T \\ \vec{x} & B \end{bmatrix}$.

Create a Householder reflector $\tilde{\mathbf{Q}}$ such that $\tilde{\mathbf{Q}}\vec{x} = \|\vec{x}\|_2 e_1$.

Next define $\mathbf{Q} = \begin{bmatrix} 1 & \vec{0}^T \\ \vec{0} & \tilde{\mathbf{Q}} \end{bmatrix}$. Then $\mathbf{Q}\mathbf{A}\mathbf{Q}^T = \begin{bmatrix} a_{1,1} & \vec{y}^T\tilde{\mathbf{Q}} \\ \tilde{\mathbf{Q}}\vec{x} & \tilde{\mathbf{Q}}B\tilde{\mathbf{Q}}^T \end{bmatrix}$

Thus, the first column of $\mathbf{QAQ}^T$ has all zero elements after the second row.

Now let $B' = \tilde{\mathbf{Q}} B \tilde{\mathbf{Q}}^T$. Inductively, repeat this algorithm on B'. Having expressed $B' = \tilde{\mathbf{Q}}' H' (\tilde{\mathbf{Q}}')^T$ where H' is an upper Hessenberg Matrix, define $\mathbf{Q}' = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Q}}' \end{bmatrix}$. Then $\mathbf{Q}'\mathbf{QAQ}^T(\mathbf{Q}')^T$ is upper-Hessenberg.

Thus we come up with an improved algorithm for QR-iteration.

1. Find an upper-Hessenberg matrix $\mathbf{H}$ such that $\mathbf{A} = \mathbf{QHQ}^T$.
2. Do QR-iteration on $\mathbf{H}$ since $\mathbf{H}$ is similar to $\mathbf{A}$.

Note that each $\mathbf{H}_k$ obtained by doing QR-iteration on $\mathbf{H}$ is also upper-Hessenberg. So the algorithm converges faster. Also, because $\mathbf{H}$ has so many zero elements, QR-decomposition only needs $\mathcal{O}(n^2)$ flops.

The total flop count of this algorithm is $\frac{10}{3}n^3 + \mathcal{O}(n^2 \cdot N)$ where n is the size of $\mathbf{A}$ and N is the number of iterations performed.

Our textbook for the class was *Introduction to Numerical Linear Algebra* by Christoph Börgers.