CS 70 - Foundations Of Applied Computer Science

Carter Kruse

Reading Assignment 2

Matrices

CS70 - Chapter 3

Definition: An $n \times m$ matrix is a way to package individual vectors together as a single rectangular array of m columns and n rows.

$$A := [\boldsymbol{a_1} \cdots \boldsymbol{a_m}] = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \cdots & & \cdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}$$

A matrix that has the same number of rows as columns (i.e. $n \times n$) is called a *square* matrix. A matrix with more rows than columns is called a *tall* matrix, and one with more columns than rows is called a *wide* matrix.

Since an $n \times 1$ matrix contains a single column it is the same thing as an *n-column vector*. A $1 \times m$ matrix contains a single row and is called a *row vector*.

Definition: The matrix *transpose* (denoted by a superscript ^{T}) flips the rows and columns: $A_{ij}^T = A_{ji}$. The matrix size will change after transpose: an $n \times m$ matrix becomes an $m \times n$ matrix.

Definition: A matrix A is *symmetric* if it is equal to its own transpose: $A^T = A$. Only square matrices can be symmetric.

Definition: The *zero matrix* is a matrix of all zeros and is typically denoted 0. If it is not clear form context, we can write $0_{n \times m}$ to specify an $n \times m$ matrix of zeros.

Definition: By packing the standard $\mathbb{R}^n$ basis vectors $\boldsymbol{e_1}, \ldots, \boldsymbol{e_n}$ as columns of a matrix, we obtain a square *identity* matrix $I_{n \times m}$ with ones along the diagonal and zeros everywhere else.

Since matrices are just a collection of vectors, multiplying a matrix A by a scalar β just scales each of the columns (elements):

$$\beta A := [\beta \boldsymbol{a_1} \cdots \beta \boldsymbol{a_m}] = \begin{bmatrix} \beta a_{11} & \cdots & \beta a_{1m} \\ \cdots & & \cdots \\ \beta a_{n1} & \cdots & \beta a_{nm} \end{bmatrix}$$

If we have two matrices A and B of the same size, then we define their sum by adding their corresponding column vectors (which is simply element-wise addition):

$$A + B := [\boldsymbol{a_1} + \boldsymbol{b_1} \cdots \boldsymbol{a_m} + \boldsymbol{b_m}] = \begin{bmatrix} a_{11} + b_{11} & \cdots & a_{1m} + b_{1m} \\ \cdots & & \cdots \\ a_{n1} + b_{n1} & \cdots & a_{nm} + b_{nm} \end{bmatrix}$$

Matrix Multiplication: For two matrices A and B to be compatible for multiplication, the number of columns in the first matrix must equal the number of rows in the second.

Definition: A row vector times a column vector gives the *inner product*. The dimensions of the two vectors have to be equal.

Definition: A column vector times a row vector gives the *outer product*. The dimensions of the two vectors do not have to be equal. An *n-dimensional column vector* times an *m-dimensional row vector* gives an $n \times m$ matrix.

System Of Linear Equations: We can express any system of linear equations as $A\boldsymbol{x} = \boldsymbol{b}$, where A and $\boldsymbol{b}$ are known, and the unknowns are packed together as the elements of the vector $\boldsymbol{x}$. The column view of matrix-vector multiplication gives us a useful interpretation of this equation.

Definition: The *column space* of a matrix $A_{n\times m}$ is the span (set of all possible linear combinations) of its columns (interpreted as vectors in $\mathbb{R}^n$).

Definition: The row space of a matrix $A_{n\times m}$ is the span of its rows (interpreted as vectors in $\mathbb{R}^m$).

Definition: The dimension of the column space of a matrix (the *column rank*) is the number of linearly independent columns. Likewise, the dimension of the row space of a matrix (the *row rank*) is the number of linearly independent rows.

Theorem: A remarkable fact from linear algebra is that the number of linearly independent columns (column rank) in a matrix equals the number of linearly independent rows (row rank). This number is the *rank* of the matrix and sometimes denoted $rank\,(A)$.

Definition: The *inverse*, denoted A^{-1} of a square matrix $A \in \mathbb{R}^{n\times n}$, is a matrix that satisfies: $A^{-1}A = I_{n\times n} = AA^{-1}$.

Not every matrix has an inverse. When its inverse does not exist, a matrix is said to be singular/non-invertible. When an inverse for A exists, it is unique and A is called invertible/non-singular.

Definition: The *inverse* of a 1×1 matrix (a scalar number) is just its reciprocal, and it exists as long as the number is non-zero.

Definition: The *inverse* of a 2×2 matrix:

$$A := \begin{bmatrix} a & b \\ c & d \end{bmatrix} \text{ exists, and is } A^{-1} := \left(\frac{1}{ad-bc}\right)\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

if the quantity $(ad - bc)$ in the denominator, called the *determinant*, is non-zero.

Definition: A square matrix $Q \in \mathbb{R}^{n\times m}$ is an *orthogonal matrix* if its columns q_i are unit length ($q_i^T q_i = 1$) and perpendicular to one another ($q_i^T q_j = 0$ for $i \neq j$), i.e., the columns form an orthonormal basis for $\mathbb{R}^n$.

Definition: If Q is an orthogonal matrix, then $Q^T = Q^{-1}$. The *inverse* of an orthogonal matrix is its transpose.

Gaussian Elimination/LU

CS70 - Chapter 4

To solve equations of the form $A\boldsymbol{x} = \boldsymbol{b}$, when A is $n \times n$, we may use Gaussian elimination as a simple solution.

The key idea behind Gaussian elimination is to transform A and the corresponding elements of $\boldsymbol{b}$ into an equivalent system that is particularly easy to solve. This proceeds in two stages: we first perform "forward elimination" to transform the system into "upper-triangular" form and then solve this equivalent system using a technique called "backward substitution."

Definition: The "forward elimination" state relies on just two types of *elementary row operations*:

- Elimination: Adding a multiple of one row to another row.
- Permutation: Swapping two rows.

We operate on one "pivot" element along the main diagonal at a time.

Definition: The "back substitution" allows us to solve a system backwards, bottom to top.

By counting the number of nested for-loops above, we see that the majority of the complexity comes from the forward elimination pass, which is $O\left(n^3\right)$. Once the system is in upper triangular form, all the hard work is done, and backwards substitution is only $O\left(n^2\right)$.

Typically we use a technique called partial pivoting. If the current pivot element A_{kk} is zero, we will swap row k with row l that has the maximum absolute value $|A_{lk}|$ from rows $k+1$ through n. Note that we need to swap the elements in $\boldsymbol{b}$ as well when we swap two rows.

We can also swap two columns instead of two rows to get a non-zero pivot. This is called full pivoting. In this case, we do not need to swap the elements in $\boldsymbol{b}$, but we need to swap elements in $\boldsymbol{x}$.

Definition: A matrix is in *row echelon form* if:

- All rows containing only zeroes are at the bottom.
- The leading coefficient (or pivot) of a nonzero row is always strictly to the right of the leading coefficient of the row above it.

The row echelon form allows us to easily extract the *rank* of the matrix. The rank of the matrix is 2 if there are two non-zero rows (or, equivalently, there are two pivot columns). This also tells us that the columns only span a 2-dimensional subspace of $\mathbb{R}^3$, making the system singular (if it is a 3×3 matrix). The pivot columns form a basis for the column space of a matrix.

To find the inverse of an $n \times n$ matrix A, we are interested in finding an $n \times n$ matrix X that satisfies $AX = I_n$.

If for some reason we do explicitly need the inverse of a matrix, we can run Gaussian elimination on this system and the solution vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ form the columns of the inverse A^{-1}.

LU Decomposition: As discussed above, the decisions taken by Gaussian elimination depend on only the elements in the matrix A and not on any particular vector $\boldsymbol{b}$. If we know ahead of time that A will remain fixed, it can be useful to pre-compute the steps of forward elimination so that we can quickly solve the system using backwards substitution any time we are presented with a particular $\boldsymbol{b}$. This is the idea behind the LU factorization/decomposition.

If a matrix A can be reduced to upper-triangular form using forward elimination without any pivoting (exchanging rows or columns), then A can be expressed as the product of two matrices: $A = LU$, where U is the upper-triangular matrix obtained from forward elimination and L is a unit lower-triangular matrix (with ones along, zeros above, and non-zero entries only below the diagonal).

How will an LU decomposition help us solve linear systems? Once we have $A = LU$, we can solve $A\boldsymbol{x} = \boldsymbol{b}$ in two steps:

- Define $\boldsymbol{y} := U\boldsymbol{x}$ and solve $L\boldsymbol{y} = b$ for $\boldsymbol{y}$ using forward substitution.
- Now with $\boldsymbol{y}$ fixed, solve $U\boldsymbol{x} = \boldsymbol{y}$ for $\boldsymbol{x}$ using backward substitution.