Lecture 6: Matrix Properties and Manipulations [*]

10-606 Mathematical Foundations for Machine Learning

1 Slicing and stacking matrices

Given a matrix ℓ, we sometimes need to refer to smaller matrices or vectors formed by keeping some rows or columns from ℓ and crossing out others. These smaller matrices or vectors are called *slices*. If $I \subseteq \{1 \ldots m\}$ and $J \subseteq \{1 \ldots n\}$ are sets of indices, then we'll write $\ell_{I,J}$ for the slice formed by keeping only the elements ℓ_{ij} with indices $i \in I$ and $j \in J$. For example, if

$$\ell = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad I = \{1,3\} \quad J = \{1,2\}$$

then

$$\ell_{I,J} = \begin{pmatrix} 1 & 2 \\ 7 & 8 \end{pmatrix}$$

We will sometimes use colon notation for index sets: in a mathematical expression, `start:end` represents the set of integers from `start` to `end`, and `start:skip:end` represents the set of integers from `start` to `end`, skipping by `skip` between successive elements. For example, $3 : 5$ means $\{3,4,5\}$, and $1 : 2 : 8$ means $\{1,3,5,7\}$. Just a colon on its own is shorthand for the entire set of indices in some dimension. So for example, $\ell_{:,3}$ is column 3 of ℓ, while $\ell_{1:2,:}$ is rows 1 and 2.

In the other direction, it's often useful to construct a matrix by gluing together smaller pieces. This is called *stacking*. For example, if

$$A \in \mathbb{R}^{2\times 2} \qquad B \in \mathbb{R}^{2\times 3} \qquad C \in \mathbb{R}^{3\times 2} \qquad D \in \mathbb{R}^{3\times 3}$$

then we can form a 5×5 matrix by stacking:

$$X = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

The original matrices are now slices of X: for example, $X_{1:2,3:5} = B$.

[*]Compiled on Thursday 8th June, 2023 at 01:54

Knowledge check

1. **Select one:** Using the matrix X above, what does the slice $X_{:,1:2}$ represent?
   a) A
   b) C
   c) $\begin{pmatrix} A \\ C \end{pmatrix}$
   d) None of the above
   - **Answer**: c, $:, 1:2$ represents the first two columns for X, which corresponds to the stack of A on top of C.
2. **Select all that apply:** Using the matrices A, B, C and D above, which of the following are valid ways to stack these matrices?
   a) $\begin{pmatrix} A & B \\ D & C \end{pmatrix}$
   b) $\begin{pmatrix} D & C \\ A & B \end{pmatrix}$
   c) $\begin{pmatrix} A & C \\ B & D \end{pmatrix}$
   d) None of the above
   - **Answer**: a and b, swapping the order of the stacked rows and columns preserves the dimensions but in c, we are trying to stack matrices of incompatible sizes, e.g., $\begin{pmatrix} B & D \end{pmatrix}$ has two rows in the first three columns (B) but three in the last three columns (D).

2 Inverses

Let U and V be inner product spaces, and let $L \in U \to V$ be a linear function. We define the *inverse* function $L^{-1} \in V \to U$ by the property

$$\mathbf{y} = L\mathbf{x} \quad \Leftrightarrow \quad \mathbf{x} = L^{-1}\mathbf{y}$$

or equivalently

$$\mathbf{x} = L^{-1}L\mathbf{x},\ \mathbf{y} = LL^{-1}\mathbf{y} \qquad \forall \mathbf{x} \in U,\ \mathbf{y} \in V$$

Unlike the adjoint, the inverse doesn't always exist. If it does, we say L is *invertible*. If L is invertible, then $(L^{-1})^{-1} = L$.

An inverse can only exist if U and V have the same dimension. If U and V have different dimensions, we might be able to find a *left inverse* or a *right inverse*. These act like inverses when they are applied in the correct order:

$$\mathbf{x} = L^{\text{left}}L\mathbf{x} \quad \forall \mathbf{x} \in U$$
$$\mathbf{y} = LL^{\text{right}}\mathbf{y} \quad \forall \mathbf{y} \in V$$

If L is not invertible, it is called *singular*. However, we might still want a function that acts like an inverse; such an operator is called the *pseudoinverse*, $L^\dagger$. Formally, we define $L^\dagger$ so that, if $\mathbf{x} = L^\dagger\mathbf{b}$, then the error $\|L\mathbf{x} - \mathbf{b}\|$ is as small as possible.

We can define an inverse operation on matrices as well: if ℓ and ℓ^{-1} are matrices that satisfy

$$\ell^{-1}\ell\mathbf{x} = \mathbf{x} \qquad \ell\ell^{-1}\mathbf{y} = \mathbf{y} \qquad \forall \mathbf{x} \in U, \mathbf{y} \in V$$

then we say that the ℓ^{-1} is the inverse of ℓ.

As we might hope, matrix inverses are related to linear function inverses: if the matrix ℓ is a coordinate representation of the linear function L, and if L is invertible, then ℓ^{-1} exists and is a coordinate representation of the linear function L^{-1}.

Similarly, there is a matrix representation of the pseudoinverse $L^\dagger$ as well. If the linear function L is not invertible, then none of its representation matrices are invertible (no matter what basis we use). If ℓ is one of these representation matrices, then $\ell^\dagger$ is the corresponding coordinate representation of $L^\dagger$.

Like the adjoint and transpose, both the inverse and pseudoinverse swap the roles of input and output vector spaces: for example, if $L \in U \to V$, then $L^\dagger \in V \to U$.

The inverse is much more useful as a mathematical tool than a computational one. It's rarely a good idea to compute the inverse of a matrix explicitly, since it tends to incur numerical errors. Instead, it's typically better to use an algorithm like Gaussian elimination, which lets us apply the inverse matrix to one or more vectors directly, without ever computing the inverse itself.

If we do need to compute the inverse of a matrix A, we can do so by solving linear systems of equations: if we write $\mathbf{e}_i$ for the ith column of the identity matrix (that is, a 1 in coordinate i and zeros everywhere else) and if $\mathbf{x}$ is the ith column of A^{-1}, then $\mathbf{x}$ satisfies

$$A\mathbf{x} = \mathbf{e}_i$$

This follows from the matrix equation $AA^{-1} = I$, since it is the ith column of that equation. So, we can find $\mathbf{x}$ by solving this linear system, and we can find all of A^{-1} by repeating the process for each $\mathbf{e}_i$. We'll see below how to efficiently solve multiple systems of equations with the same matrix and different right-hand side vectors.

We can do something similar to find left and right inverses: we can use $AA^{\text{right}} = I$ to solve for a column of A^{right} at a time, and we can use $A^{\text{left}}A = I$ to solve for a row of A^{left} at a time.

Knowledge check

1. **Short answer**: If $L \in \mathbb{R}^m \rightarrow \mathbb{R}^n$ with $m < n$, can we find a left inverse of L? What about a right inverse?
   - **Hint:** To get started with this one, think about the meaning of a left and right inverse. Then compare the sizes of $\mathbb{R}^m$ and $\mathbb{R}^n$ and consider the implications for trying to "reverse engineer" the function L from one space to another.
   - **Answer**: If $m < n$ we can get a left inverse but not a right inverse. To see this, observe that $\mathbb{R}^m \subseteq \mathbb{R}^n$. The existence of a left inverse basically asks "after mapping some vector from $\mathbb{R}^m$ (the smaller space) to $\mathbb{R}^n$ (the bigger space), can I figure out what the original vector is?" to which the answer is potentially, if L maps each vector in $\mathbb{R}^m$ to a unique vector in $\mathbb{R}^n$. However, because of their relative sizes and the linearity of L, there will always exist some vector $\mathbf{x} \in \mathbb{R}^n$ that cannot be mapped back to a vector in $\mathbb{R}^m$. Hence, the right inverse cannot exist.
2. **Short answer**: Again, suppose $L \in \mathbb{R}^m \rightarrow \mathbb{R}^n$ but now, assume $m > n$. Now can we find a left inverse of L? What about a right inverse?
   - **Answer**: If $m > n$ we can get a right inverse but not a left inverse. A similar chain of logic to the previous question applies where we consider the relative sizes of $\mathbb{R}^m$ and $\mathbb{R}^n$, noting that in this setting, multiple vectors in $\mathbb{R}^m$ must be mapped to the same vector $\mathbf{x} \in \mathbb{R}^n$ and thus, it will be impossible to come up with a left inverse that always returns the correct original vector in $\mathbb{R}^m$.

3 Matrix patterns

Consider a linear operator $L \in U \to V$ and its matrix representation ℓ under some bases for U and V. Depending on which bases we pick for U and V, the matrix ℓ can look quite different. If we can pick these bases so that many elements of ℓ are zero, then some computational operations involving ℓ become faster. If the nonzero elements follow a simple pattern, then operations can become even faster.

For example, ℓ is square (the dimensions of U and V are the same) and if all of the nonzeros of ℓ fall on the main diagonal (that is, if $\ell_{ij} = 0$ when $i \neq j$), we say that ℓ is *diagonal*. Writing $\times$ for a possibly-nonzero entry and 0 for an entry that must be zero, a diagonal matrix matches the pattern

$$\begin{pmatrix} \times & 0 & 0 & \dots & 0 \\ 0 & \times & 0 & \dots & 0 \\ 0 & 0 & \times & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \times \end{pmatrix}$$

The simplest diagonal matrix is the *identity matrix I*, which has entries of 1 on the main diagonal and 0 off of it. (That is, all of the $\times$ entries above are 1.) If $U = V$, the identity matrix corresponds to the identity function: that is, the function L that satisfies $L\mathbf{x} = \mathbf{x}$ for all $\mathbf{x} \in U$.

Another useful pattern is *lower triangular*: a square matrix is lower-triangular if it matches the pattern

$$\begin{pmatrix} \times & 0 & 0 & \dots & 0 \\ \times & \times & 0 & \dots & 0 \\ \times & \times & \times & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \times & \times & \times & \dots & \times \end{pmatrix}$$

that is, if $\ell_{ij} = 0$ when $j > i$. An *upper triangular* matrix follows the opposite pattern: the nonzeros are on and above the main diagonal, so that its transpose is lower triangular.

A matrix with lots of zero entries but no particular pattern is called *sparse*. We can save computation by working with just a list of the nonzero entries and their indices, instead of storing and manipulating all of the zeros. Typically, though, we need a high fraction of nonzero entries to make it worthwhile to do this: say at least 80% zeros, and preferably much more. The reason for the penalty is that we spend some storage to explicitly represent the indices of the nonzeros, and we waste some computation because of the less-uniform structure of the list. Fortunately, very sparse matrices are common in some machine learning applications: it's perfectly typical to see matrices where 99.9% of the entries are zero.

If a matrix follows multiple patterns, it can be even more convenient to work with, e.g., a sparse lower triangular matrix is more convenient than a matrix that is just sparse or just lower triangular.

Knowledge check

1. **True or False**: A square matrix can be both upper triangular and lower triangular.
   - **Answer**: True, this is a bit semantic but technically a diagonal matrix is both upper and lower triangular, we just replace the off-diagonal $\times$'s as well (this is because our specification for a lower triangular matrix was "$\ell_{ij} = 0$ when $j > i$" not "$\ell_{ij} = 0$ if and only if $j > i$".

4 Orthogonality

A column-orthogonal matrix is one whose columns are orthogonal vectors. This is equivalent to saying that the matrix $D = A^T A$ is diagonal. A row-orthogonal matrix is one whose rows are orthogonal vectors, so that AA^T is diagonal. A row- or column-orthonormal matrix is one whose rows or columns are orthonormal.

If A is square and its rows are orthonormal, then its columns must also be orthonormal. Similarly, if the columns are orthonormal, the rows must be too. In this case $AA^T = A^T A = I$. We can interpret this equation three ways:

- The dot products of columns of A are either 0 (for distinct columns) or 1 (for the dot product of a column with itself).
- Similarly, the dot products of rows of A are either 0 or 1.
- The inverse of A is A^T.

The first two interpretations come from looking at the matrix equation element by element; the last comes from matching the entire equation to the definition of the inverse.

5 Positive (semi)definite operators and matrices

A linear operator $A \in U \to U$ is called *positive semidefinite* or *PSD* if $\langle \mathbf{x}, A\mathbf{x} \rangle \geq 0$ for all vectors $\mathbf{x} \in U$. It is called *positive definite* if the inequality is strict: $\langle \mathbf{x}, A\mathbf{x} \rangle > 0$ when $\mathbf{x} \neq 0$. We also apply the terms positive semidefinite and positive definite to any matrix representation of A.

Self-adjoint PSD operators have some nice properties as we will see, as do symmetric PSD matrices. In fact, these properties are so nice that some authors include symmetry or self-adjointness in the definition of PSD (we won't do that, since there are also some uses for asymmetric or non-self-adjoint PSD matrices or operators). For example, we can test positive definiteness or semidefiniteness of a symmetric matrix easily using Gaussian elimination; this is called Cholesky factorization in the definite case, and LDL^T factorization in the semidefinite case. Therefore, we can test definiteness of a self-adjoint operator by using Cholesky or LDL^T factorization on any coordinate representation for it.

To test definiteness of an asymmetric matrix and its corresponding non-self-adjoint operator, we can use the following fact: a matrix A is positive definite (or semidefinite) if and only if $A + A^T$ is. The latter is clearly symmetric, so we can turn to Cholesky or LDL^T factorization.

6 Matrix factorizations

Gaussian elimination is also called LU factorization: it allows us to represent a matrix A as a product $A = LU$ of a lower triangular matrix L and an upper triangular matrix U.

This is one of many possible matrix factorizations, all of which can be really useful. For example, factoring a matrix is often a good way to solve many copies of an equation for different right-hand sides:

$$A\mathbf{x}_1 = \mathbf{b}_1, \quad A\mathbf{x}_2 = \mathbf{b}_2, \quad \ldots$$

We factor A once and then repeatedly use the factorization to solve for the different right-hand sides. If we factor $A = LU$, then for each new vector $\mathbf{b}_i$ we can do two triangular backsubstitutions (for L and then for U) to find $\mathbf{x}_i$; this is much faster than solving $A\mathbf{x}_i = \mathbf{b}_i$ from scratch each time.

In an LU factorization, the matrix U is the result of all of the row operations we apply: for each diagonal element of A, we use row operations to eliminate all of the off-diagonal elements below it, leaving nonzero elements only on and above the main diagonal.

On the other hand, the matrix L encodes our sequence of row operations. A single row operation can be implemented by a matrix that looks like this:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ -2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

All row operations are invertible; for example, the inverse of this one is

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

If we multiply A by the first matrix above, we will subtract twice the first row of A from the fourth row of A. On the other hand, the second matrix above adds twice the first row of A to the fourth row.

Note that both matrices above are lower triangular. Since the product of lower triangular matrices is lower triangular, we can multiply all of the inverses of our row operations together into a single lower triangular factor L.

If we factor a symmetric positive definite matrix this way, we can require $L = U^T$ (which forces the product to be symmetric and definite). This is called a Cholesky factorization. A variant is to pull out the diagonal entries of L and L^T into a diagonal matrix D, resulting in a factorization

$$A = LDL^T$$

where the matrix L is lower triangular and has all diagonal entries equal to 1:

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ \times & 1 & 0 & \dots & 0 \\ \times & \times & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \times & \times & \times & \dots & 1 \end{pmatrix}$$

Since the original matrix is PSD, the entries of D will be nonnegative. We can go back to a Cholesky factorization by multiplying the square root of each diagonal entry into the corresponding row and column of L.

We can also make the same LDL^T factorization for a non-PSD symmetric matrix. But in this case some of the diagonal entries of D will be strictly negative, and so we can't make a Cholesky factorization. This is in fact a great way to test whether a matrix is positive definite: we try to build the LDL^T factorization. If we get all the way to the end with only positive elements of D, we know the matrix is positive definite. If we ever run into a negative element, we know that the matrix is not positive definite.

If we ever run into a zero element, the story is slightly more complicated: we need to try a technique called *pivoting* to avoid having to divide by zero. Pivoting works by rearranging rows and columns of our matrix so that a nonzero element is on the diagonal. If pivoting is impossible, that means we're stuck with only zero elements remaining; so the remaining elements of L and D are zero, and the matrix is PSD but not positive definite.

Knowledge check

1. **Select one**: Let $A = \begin{pmatrix} 2 & 0 \\ 2 & 2 \end{pmatrix}$. Which of the following statements is true?
   a) A is positive definite
   b) A is positive semidefinite but not positive definite
   c) A is not positive semidefinite
   - **Hint**: Try to verify the property that $\langle \mathbf{x}, A\mathbf{x} \rangle > 0$ when $\mathbf{x} \neq 0$ explicitly by defining some general vector $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$
   - **Answer**: a, to see this, let $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ and observe that

$$\langle \mathbf{x}, A\mathbf{x} \rangle = \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 2 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2x_1 + 2x_2 & 2x_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{1}$$

$$= 2x_1^2 + 2x_1x_2 + 2x_2^2 = (x_1 + x_2)^2 + x_1^2 + x_2^2 \tag{2}$$

which is always positive as long as $\mathbf{x} \neq 0$. Hence A is positive definite.