A tensor is an array with more than two axes (whereas a matrix only has two)

The addition of a vector and matrix yields another matrix

the jth element of b is added to the jth element of each row in A

This is known as broadcasting

**2.2**

Elementwise-product, or Hadamard product, is multiplying each corresponding element

C = AB <= Cij is the dot product of row i of a and col j of B



But matrix multiplication is not commutative

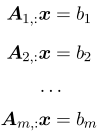


Given that the dot product of two vectors is a scalar, and aT = a for a scalar,





Each row of A and element of b provide their own constraints

or



**2.3**

For all vectors of x which exist in Rn, x times the identity matrix is itself





Theoretically, the same inverse matrix can be used to solve for different values of B, but in practice it is not used since A-1 can only be represented with limited precision on a digital computer

**2.4**

In order for A-1 to exist, there must be one solution for every value of b; it is possible to have 0 solutions or infinitely many solutions for some values of b, but not anywhere in between for a particular b

This is because if you have two solutions, x and y, then

 is also a solution for any real alpha

Think of the columns of A as specifying different directions we can go in starting at the origin, with each element of x specifying how far we should travel in these directions

This is a linear combination

 where ci is some scalar coefficient

The span of a set of vectors is the set of all points obtainable by linear combinations of the original vectors

Ax = b is therefore testing whether b is in the span of A, known as the column space

For Ax = b to have a solution for all values of b in RM , the column space of A must be all M

For the column space of A to be all of RM, A must have at least m (n >= m) columns - otherwise you get a plane or a subset of solutions in RM.

n >= m is necessary for every point to have a solution, but does not satisfy the condition on its own

To have a solution for every b there must be exactly m linearly dependent columns

For the matrix to have an inverse, there must be at most one solution for each value of b

* The matrix must have m columns so that there is only 1 way to parametrize the solution

The matrix must be square (m = n), and all the columns must be linearly independent (since there are only m) ; square matrix with linearly dependent matrix is singular

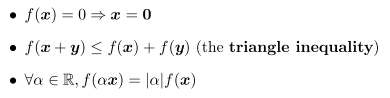
If A is not square or is singular, then matrix inversion cannot be used to solve the equation

Measure the size of a vector by a LPnorm, given by

where p is a real number greater than 1

Norms map vectors to non-negative values ; intuitively, they measure the distance from the origin to point x

Rigorously, the norm satisfies the following



For all real numbers a, the function applied to the scaled vector equals the function applied to the vector scaled up by a factor of a

The squared L2 norm can be calculated as xTx

The squared L2 norm is mathematically more efficient to use than the regular L2 norm since the derivatives of the squared L2 norm only depend on the corresponding element of x, while the derivatives of the regular L2 norm depend on the entire vector

L2 norm increases very slowly near the origin, so it may be more desirable to use L1 norm which is simply 

The L1 norm is used when the difference between zero and nonzero elements is important

L0 norm - # of non-zero entries

The Linf norm simplifies to the absolute value of the element w/ the largest magnitude in the vector

The size of a matrix is measured with a Frobenius norm, analogous to the L2 norm

Dot product in terms of norms: 

**2.6 Special Matrices and Vectors**

Diagonal matrix has nonzero entries along the main diagonal

diag(v) is a square diagonal matrix whose diagonal entries are given by the vector x

Diagonal matrices are computationally efficient to multiply by and find the inverse of

Not all diagonal matrices need to be square - you can remove or add some columns / rows of zeros when taking the product Dx and still be computationally efficient

Symmetric matrix equals its own transpose, often because the function which generated the elements does not depend on the order of the elemnts (e.g Aij = Aji for distance)

Unit vector has a unit norm



A vector x and y are orthogonal if xTy = 0 => if zero, they are at 90 deg to each other

In RN, only n vectors can be mutually orthogonal

If the vectors have unit norm and are orthogonal, they are orthonormal

Orthogonal matrix is a square matrix whose rows are mutually orthonormal and whose cols are mutually orthonormal:

[Orthogonal matrix example](https://miro.medium.com/max/3296/1*kyg5XbrY1AOB946IE5nWWg.png)

which implies that 

Orthogonal matrices have a very cheap inverse to compute

Orthogonal matrices must have *orthonormal* rows

**Eigendecomposition**

Just like how numbers have universal properties which can be explored by decomposing them into prime factors, matrices can be decomposed to show their functional properties

Eigen-decomposition decomposes a matrix into a set of eigenvectors and eigenvalues

An eigenvector of a square matrix A is a nonzero vector v such that multiplication by A only changes the scale of v

[Example](https://i.ytimg.com/vi/bqIwOQYOepU/maxresdefault.jpg)

Lambda is the eigenvalue

Left eigenvector: 

If v is an eigenvector of A, then so is any rescaled vector sv

* sv also has the same eigenvalue
* Therefore, we only look at unit eigenvectors

Suppose a matrix A has n linearly independent eigenvectors and has n corresponding eigenvalues

Concatenate the eigenvectors to form a matrix V w/ one eigenvector per column ; concatenate the eigenvalues to form a vector ƛ, each element being an eigenvalue

The eigendecomposition is given by

Constructing matrices with specific eigenvectors and eigenvalues allows us to stretch space in desired directions (e.g across a diagonal axis instead of x,y, or z)

We often want to decompose matrices into eigenvectors and eigenvalues

Sometimes, an eigendecomposition may exist but involves complex numbers

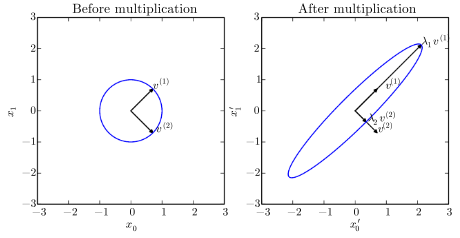
Every real symmetric matrix can be decomposed into an expression using only real-valued eigenvectors and eigenvalues

Q is an orthogonal matrix composed of eigenvectors of A, and is a diagonal matrix

The eigenvalue  corresponds to the eigenvector in column i of Q, Q:, i

Since Q is orthogonal, A can be considered as scaling space by lambdai in direction vi

Ex of a matrix A being applied to V, a matrix of eigenvectors



While any symmetric matrix A has an eigendecomposition, it may not be unique

If any two eigenvectors share the same eigenvalues, then any set of orthogonal vectors in their span are also eigenvectors w/ that same eigenvalue, and we can choose a Q using those eigenvectors instead

Sort the eigenvalues in descending order

Eigendecomposition is unique only if all the eigenvalues are unique

Eigendecomposition tells us useful facts about the matrix - it is singular only if any of the eigenvalues are 0

If where the norm of x is 1, whenever x equals the eigenvector of A, f takes on the value of the corresponding eigenvalue

Maximum value of f within the constraint region is the maximum eigenvalue

Positive definite matrix if its eigenvalues are all positive

Positive semidefinite if they are all positive or 0

For all x, in a positive definite matrix

Positive definite matrices are useful in deep learning since they allow us to solve optimization equations

[Example](https://miro.medium.com/max/2292/1*c0YCmZSf9L0YIII-Xk3Zzw@2x.jpeg)

**Singular Value Decomposition**

Singular value decomposition (SVD) allows us to factorize a matrix into singular vectors and singular values

SVD is defined for every matrix, making it more applicable than eigendecomposition

SVD is similar, except A is a product of three matrices



U and V are orthogonal matrices, and D is a diagonal matrix (not necessarily square)

Columns of U - left singular vectors ; columns of V - right singular vectors ; values along diagonal of D - singular values of matrix A

[Example](https://2.bp.blogspot.com/-QMQY1jKx5Xo/Wy0GwFCOOwI/AAAAAAAAE1M/mPuhZRxw6cMUzecrsV2F-ZXmuGi9KlMpQCLcBGAs/s1600/Singular%2BValue%2BDecomposition%2BExample.png)

Left singular vectors of A are the eigenvectors of AAT, and the right singular vectors of A are the eigenvectors of ATA

Nonzero singular values of A are the square roots of the eigenvalues of ATA

SVD allows us to "invert" non-square matrices

**Moore Penrose Pseudoinverse**

Suppose we want a left inverse matrix B to solve Ax = y such that x = By

It may not be possible to design a unique mapping from A to B

If A is taller than it is wide, then this equation may have no solution

Moore-Penrose pseudoinverse

which equals 

U, D, and V are the singular value decompositions ; the pseudoinverse D+ of a diagonal matrix D is obtained by taking the reciprocal of its nonzero elements and then transposing

When A has more columns than rows, solving the equation using a pseudoinverse provides one of many possible solutions ; it provides the solution with the minimum Euclidean norm ||x||2 among all solutions

When A has more rows than columns, and there is no solution, taking the pseudoinverse gives us the x for which Ax is as close to y in terms of the euclidean norm / distance 

**Trace Operator**

The trace operator gives the sum of all of the diagonal entries of a matrix

Makes it easier to write certain operations such as the Frobenius norm: 

Trace operator is invariant to transpose: 

Trace of a square matrix with many factors (submatrices) is invariant to moving the last factor into the first position, as long as the resulting product is defined

Einstein superscript notation xi => basis vector column of a matrix or index of a vector element

ei => standard basis vector with a 1 at position i

or more generally 

where F is the factor matrices and the notation means multiplying each Fi (summation but multiplication instead)

Invariance to cyclic permutation is true even if the resulting product has a different shapeeven though AB shape is not the same as BA

A scalar is its own trace

**Determinant**

The determinant, noted det(A) maps matrices to real scalars which represent area scaling factors

The determinant is equal to the product of all the eigenvalues of the matrix

If the determinant of the transformation matrix is 1, then the transformation preserves volume / space

**PCA**

Suppose we have a collection of m points {x1 , .. , xm } and we would like to use lossy compression to store them while keeping maximum precision

For each point, we will find a corresponding code vector which has lower dimensionality than the original point, so that it requires less memory

We want to find an encoding function, f(x), which produces a code and a decoding function g(x) which produces the reconstructed input (approximately) given its code

g = Dc is our decoding function where D is a member of Rn by l

PCA constrains the columns of D to be orthogonal to each other and to have unit norm so that there cannot be infinitely many solutions

First we need to figure out how to generate the optimal code point c\* for each input point x

Minimize the distance between x and its reconstruction g(c\*)



Use the squared L2 norm since L2 norm is nonnegative and L2 squared is easier to compute



The function minimized simplifies to

by the definition of the L2 norm

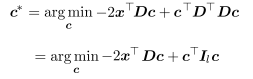
=

since g(c) equals itself

Omit the first term since it does not depend on c\*

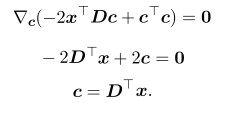


Substitute in the definition of g(c)

due to orthogonality and unit norm constraints



We can now solve using vector calculus

gradient with respect to c

We can optimally encode x using a matrix-vector operation

Apply the encoder function



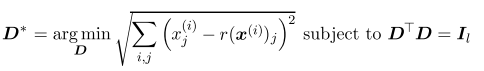
We can also define the PCA reconstruction operation



Now we need to choose an encoding matrix D

Minimize the L2 distance between inputs and reconstructions

Since the same matrix D is used to decode all the points, the points are no longer considered in isolation and we must use the Frobenius norm of the matrix of errors computed over all points



When l = 1, D is just a single vector d, where the problem simplifies to

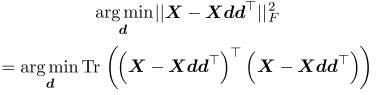
or the more conventional way to write it with the scalars being on t he left of the vector they operate on

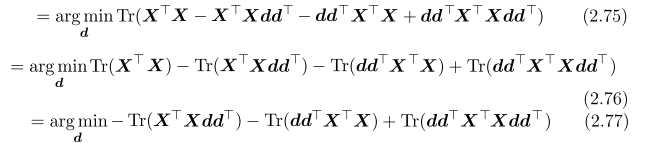
or since a scalar is its own transpose: 

Rather than using a sum of separate example vectors, it is simpler to re-write the problem in terms of a single design matrix

where X is an m by n matrix defined by stacking all the vectors describing the points such that 

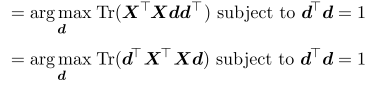
The frobenius norm can be simplified as follows



since terms not involving d don't affect arg min

since we can cycle order of matrices inside a trace





This can be solved using eigendecomposition - the optimal d is given by the eigenvector of XTX corresponding to the largest eigenvalue

In general, the matrix D is given by the l eigenvectors corresponding to the largest eigenvalues

**More on eigenvalues**

Av = lambda \* v

Suppose lambda is the max eigenvalue

The direction which stretches out A the most is given by v, as indicated by a high eigenvalue