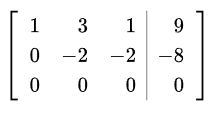
**Linear Algebra**

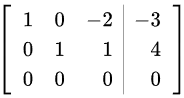
* Gaussian Elimination

Echelon form:



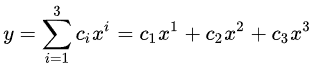
* Gauss-Jordan Elimination

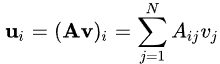
Reduced echelon form:



* Einstein Summation Convention

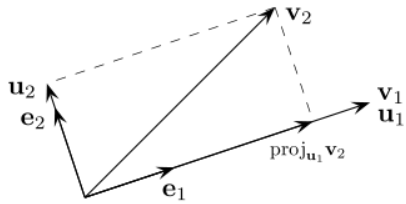
Summation over a set of indexed terms in a formula

is written as 

 is written as 

* Gram-Schmidt Process

Method for orthonormalising a set of vectors



* Determinant

The factor by which the transformed(transformed by the matrix) area is changed(stretched/compressed)

* Eigen vector and eigen value.



An eigenvector or characteristic vector of a linear transformation is a non-zero vector that only changes by a scalar factor when that linear transformation is applied to it i.e. the vector still spans the same line/region before and after transformation. Lambda is the eigenvalue, characteristic value, or characteristic root associated with the eigenvector v i.e. lambda is the size of the eigenvector after transformation.

**Multivariate Calculus**

* Derivative



* Partial Derivative





* Total Derivative

Total derivative of  is 

* Jacobian

Jacobian matrix describes how a small region is transformed when changing from one basis(coordinate) to other

Jacobian determinant tells by what factor the area of the small region got changed when moving from one basis(coordinate) to other

Given a function, its Jacobian will be a vector as only first partial derivative is taken

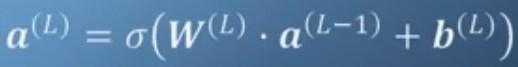
* Hessian

Given a function, its Hessian will be a matrix as the second partial derivative is taken. Hessian matrix will be symmetric matrix(only if the function is continuous)

Jacobin and Hessian appear only in multivariate case

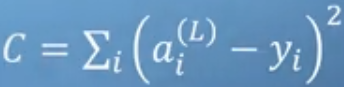
Neural Network

* Output vector is dot product of weight matrix and input vector added with bias vector



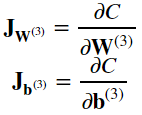
σ denotes activation function, L denotes layer

* Cost function is the square of the difference between the network output(final layer) and actual output

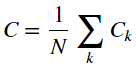


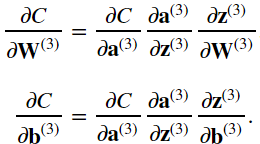


* Backpropagation tries to find the minimum cost w.r.t the weights and bias. So the weights and bias have to be adjusted to get minimum cost. The weights and bias are adjusted in the direction where the gradient of the cost function is minimum. This calls for calculating the Jacobian of the cost function w.r.t weights and bias



Where C is the average cost function over the training set



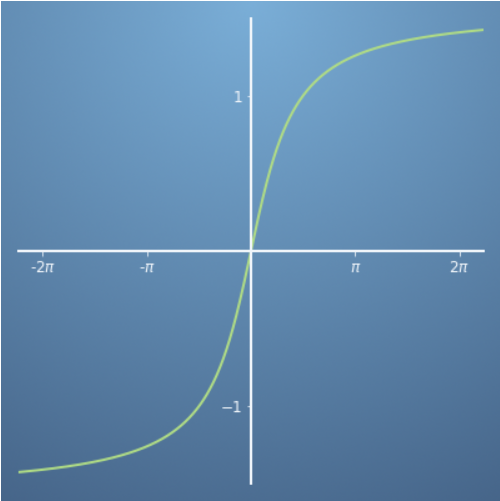


* Taylor Series
* Linearization

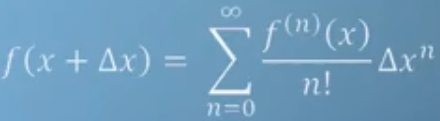
Exact function can be derived from infinite series Taylor series form. To linearize a function, its Taylor series is considered only upto first order and all the remaining terms are treated as error. This would also mean the approximation is second order accurate. Below is the Taylor series approximation for linearization written in a different way.



* If a function has rotational symmetry w.r.t origin, it is an odd function. The below function and also Sin is an odd function whereas Cos is an even function



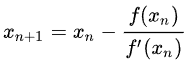
* Multivariate Taylor

 is single variate Taylor

is multivariate Taylor.

X and delta X are vectors. For e.g. for 2D, x = [x, y] and delta\_x = [delta\_x, delta\_y]

* Newton-Raphson method is an iterative method of arriving at a solution to a equation using the gradient

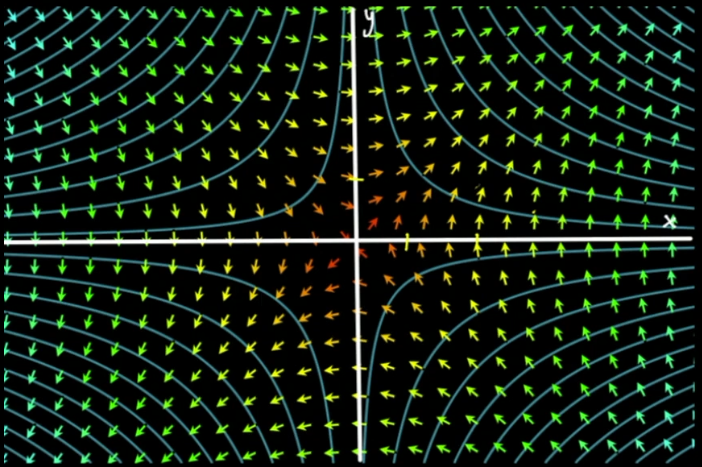


where x\_n is the initail guess of the solution. The above formula is arrived using the line formula to the tangent(slope/gradient) of the function at x\_n

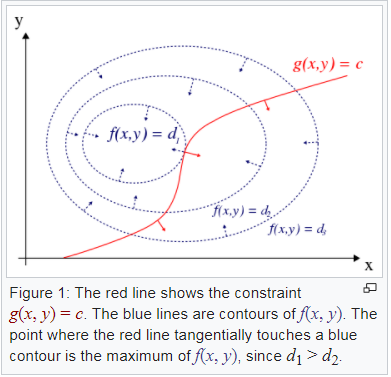
* Gradient points up in the direction of steepest assent. So -ve gardient points down in the direction of steepest descent. Newton-Raphson method moves towards the root/solution which is known and is 0 but in Gradient descent, we have to move towards the minimum value and the minimum value is not always 0. To find a local minimum of a function using gradient descent, we have to take steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point.



* Gradient is the link between linear algebra(vectors in particular) and calculus and we entered into the relms of vector-calculus
* Gradient is always perpendicular to the contour line and it can be thought as getting to the next contour line as fast as it can or increasing the function as fast as it can. E.g. f(x,y) = xy shown below with its contour lines and gradients.



* Directional derivative is gradient of a function along a given vector. It is dot product of gradient and vector.
* Lagrange Multilier: The method of Lagrange multiplier is one of the “constrained optimization” method which is a strategy for finding the local maxima and minima of a function subject to a constarint i.e. subject to the condition that one or more equations have to be satisfied exactly by the independent variables of the function. This is easily solvable using contour lines and gradients. So we have to find the contour of f which is tangential to the contour of g. The point of tangency is the maximam/minimum point.



At the point of the maximum/minimum, the gradient of g will be proportional to gradient of f. So solving will give us the maximum/minimum point(x,y).

* Lagrangian: For easier computation in computers, the “constrained optimization” problem is converted to “unconstrained optimization” by using Lagrangian



and solving

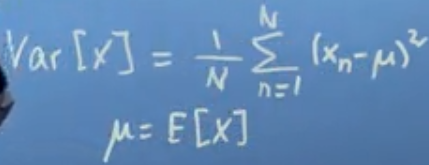


using Newton-Raphson method.

* Linear and non-linear least square fitting method for linear and non linear regressing uses gradient of the square of the difference in model and actual value. Solving for gradient to be zero gives the parameters of the line or curve.

**Principle Component Analysis**

* Mean: average of all the data points.
* Variance: Characterises the variability or spread of datapoints in a data set.



Variance is in square units and mean is in units. Comparing them will be difficult. So the square root of variance, standard deviation, is taken to easily understand the spred of the data points from the mean.

* Covariance: For higher dimensional datasets, we can find variance in each dimension which gives the variation/relation of the dataset in one direction independent of the other directions. Covariance gives the variation/realtion of dataset between different directions.

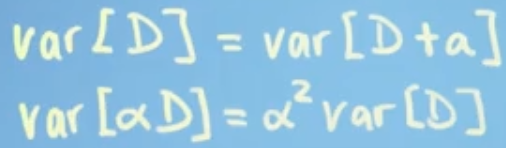


If increasing x increases y, then cov(x,y) is positive and if increasing x decreases y the cov(x,y) is negative. If there is no relation between x and y, then cov(x,y) is zero.

* Effect of linear transformation on mean:

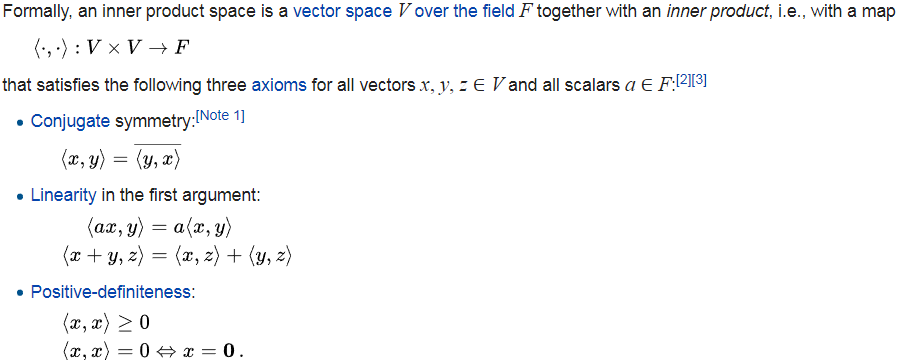


* Effect of linear transformation on variance:

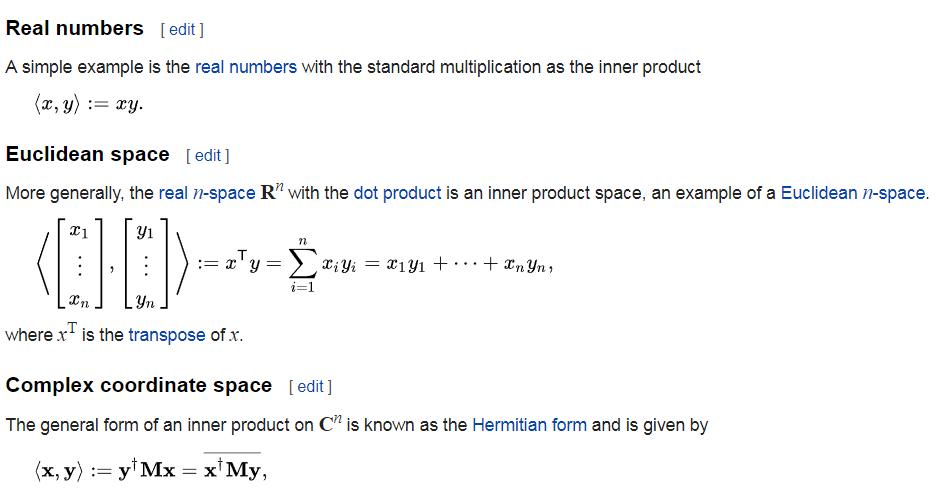


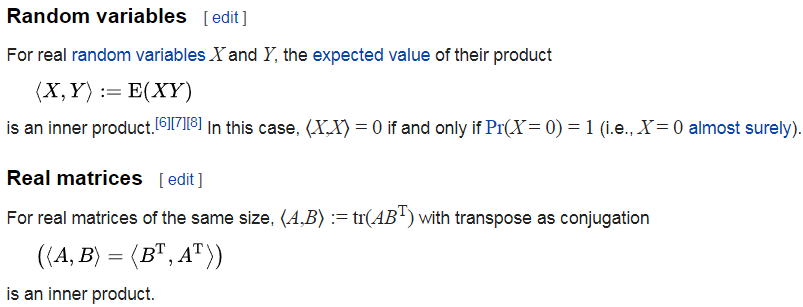


* Inner product: It is the generalization of dot product



E.g.





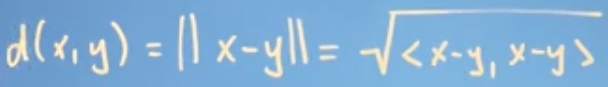
* Triangular inequality:



* Cauchy-Shwarz inequality:

 or equivalently, taking the square root, 

* Distance between two vectors:

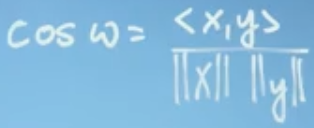


The distance depends on the choice of theh inner product.

If the inner product, M, is an identity matrix, then the distance is Eucledian distance.

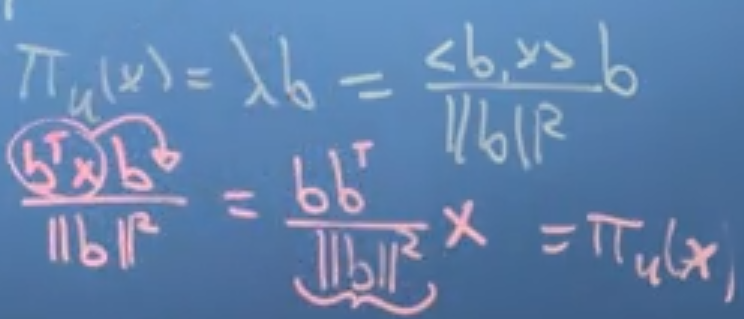
If the inner product, M, is something else, then the distance varies too.

* Angle between vectors: In general case

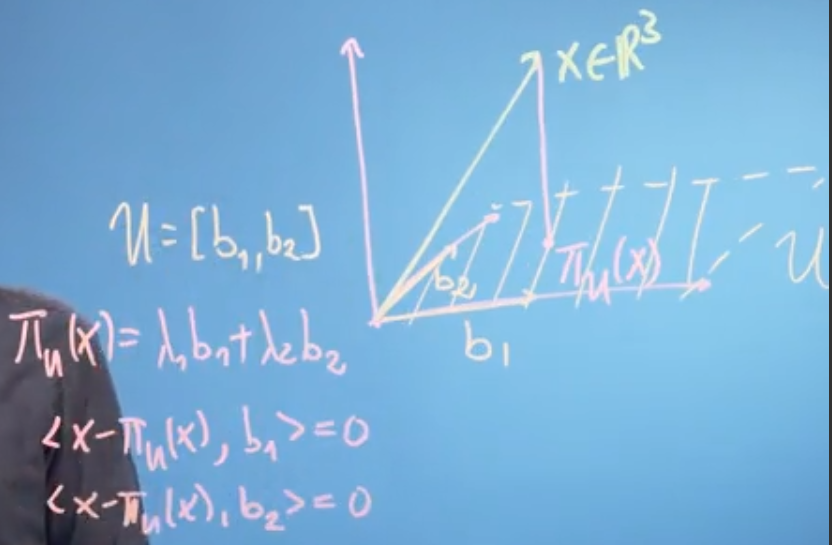


The angle between the vectors don’t remain same when we change the inner product. For e.g. if x=[1,1] and y=[-1,1] are orthogonal in eucledian space where the inner product is dot product but the same x and y are not orthogonal when the inner product is [2, 0; 0, 1]

* Norm or magnitude of a vector: Norm of a vector is the square root of the inner product of the vector with itself. ||**v**|| = sqrt(<**v**,**v**>). While taking the norm, the given inner product should be used and matrix I should not be assumed as it would be the dot product.
* Projection Matrix when inner product is dot product when projecting a vector on 1-D sub-space U

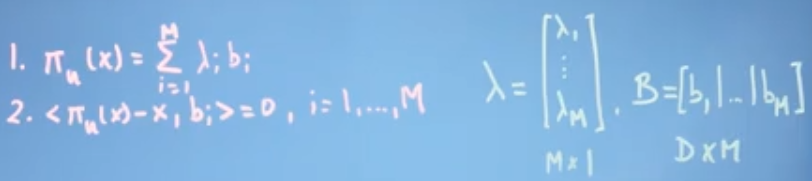


* Projection on 2-D sub-space U

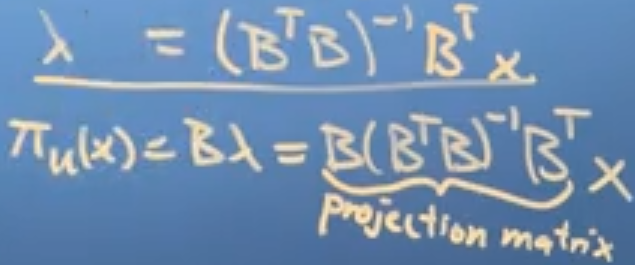


The projected vector will be a linear combination of the basis vectors of the sub-space U and the difference vector between the given vector and its projection will be orthogonal to all the basis vector, i.e. their inner product will be zero.

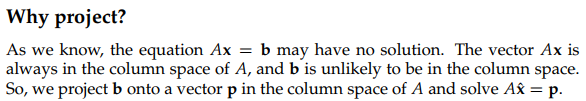
* Generalizing, projection of D-D vector on M-D sub-space U



Solving 1 and 2, and taking dot product as inner product, we get

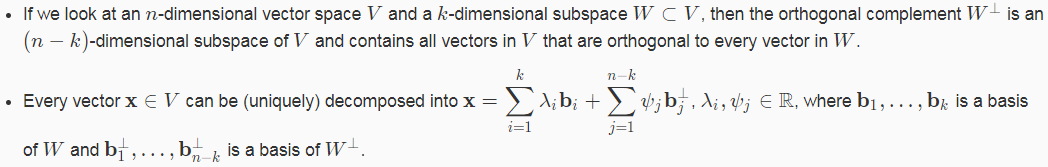


* Application of projection:

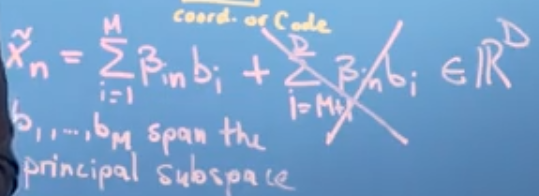


* Study PCA as an algorithm that minimizes average reconstruction errors by orthogonal projections.
* Orthogonal Complement and Orthogonal Decomposition

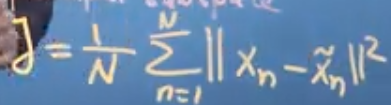
Any vector in n-D vector space can be represented as a linear combination of basis vector of k-D subspace and basis vector of (n-k)-D subspace which is orthogonal to the k-D subspace.



* Idea behind PCA:

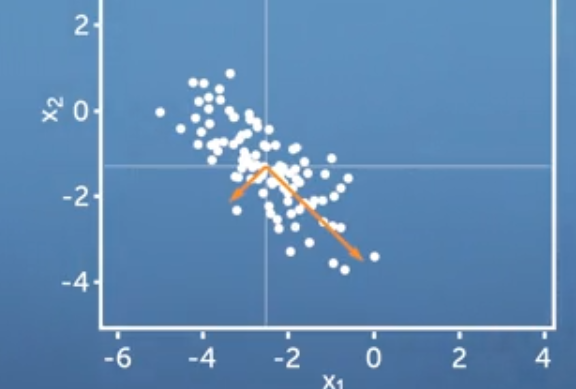


The second term of the orthogonal decomposition is removed. This imples that a D-Dimensional vector is represented as a M-Dimensional vector in M-Dimensional subspace of RD. This also implies that only M co-ordinates(βs) are required to represent the vector thus leading to compression while storing/transmitting data. The problem thus requires us to find the co-ordinaates βs and the orthonormal basis bs such that the average squared resonstruction error is minimized. The error is given as below:

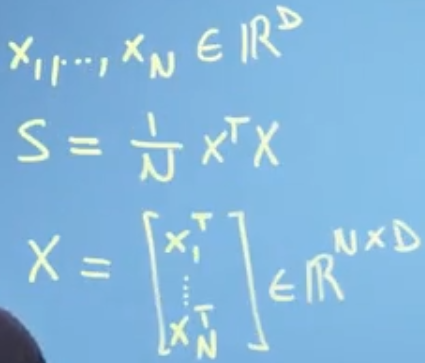


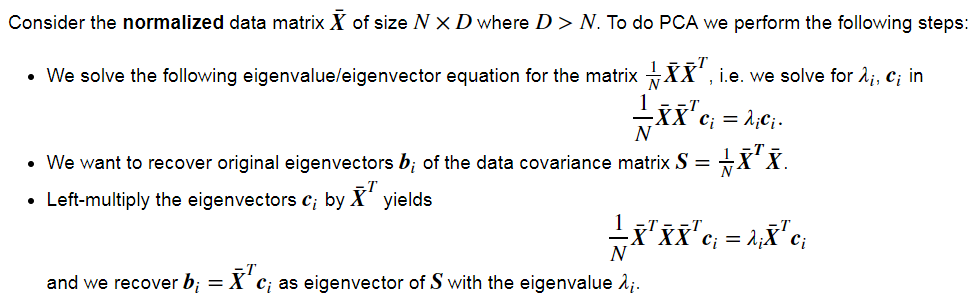
The error is minimized by solving the partiala derivative of the error with respect to β and b equated to zero. Βand b is related to the error via xn.

* For PCA, the data should be projected onto the dimensions in which the data variance is more and the less varaince dimension can be ignored. The basis vector of the largest varaince dimension will be the eigen vector and the corresponding variance will be the eigen value

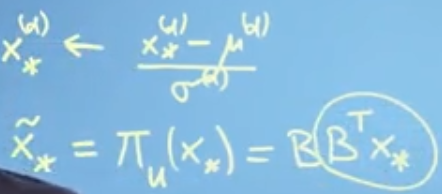


The above graph shows that the data should be projected onto the dimension spanned by the longer orange basis to reduced the average recontruction error.

* PCA algorithm:
*  xi is a datapoint with D dimensions. Assume there are N datapoints. These are put into a data matrix X whose dimension would be NxD.
* It is good to zero-center the data(subtract by the mean) in every dimension. This avoids numerical problems(E.g If all data is in order of 108, multplying the numbers will lead to very large numbers)
* Divide every dimension of the centered data by the corresponding standard deviation. This makes the data unit free and gaurantees that the variance of the data in every dimension is 1.
* Find the data co-varaince matrix, S. The formula for S above works efficiently only if N > D i.e. we have number of datapoints more than the dimension of the datapoint. If N < D, then



* Then compute the eigen values and eigen vector of the data co-variance matrix
* Project the data points onto the principal subspace that is spanned by the eigen vectors that belong to the largest eigen values.



B is the matrix that contains the eigen vectors, that belong to the largest eigen values, as columns and BTx­­\* are the co-ordinates of the projection with respect to the basis of the principal sub-space.

Note: The projection matrix is B(BTB)-1BT but as we have assumed B to be orthonormal, BTB will be identity matrix.