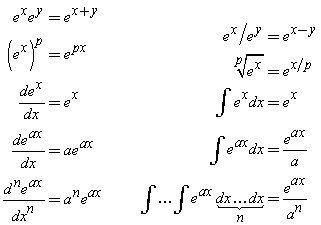
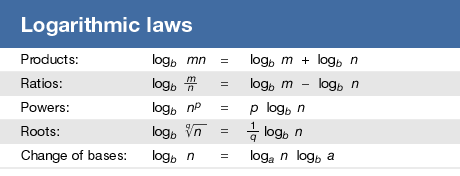
# Math Basics





Graphical user interface, text, application

Description automatically generated

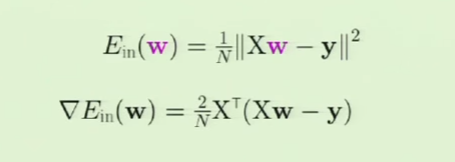
Table

Description automatically generated

A picture containing text, whiteboard

Description automatically generated

**Derivative of matrix norms**



**Derivative of Determinants**

****

# Linear Algebra

**Basics of Linear Transforms**

Linear transform: All grid lines are parallel and evenly spaced after a transform.

*“Think of matrices as functions. It’s always a useful approach”.* CS229 prof

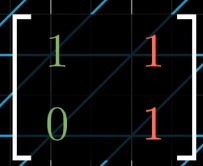
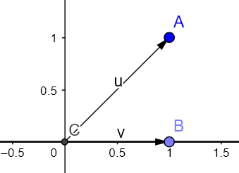
The four subspaces:

1. Row space C(AT) column space of A transposed
2. Null space N(A)
3. Column space C(A)
4. Left nullspace N(AT)

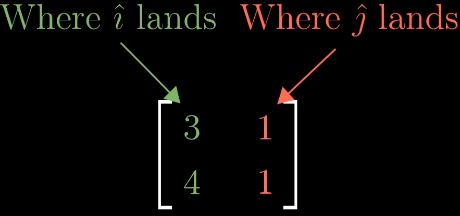
Rank-nullity theorem: let T: V-> W be a linear transform between two vector spaces: rank(T) + Nullity(T) = dim V

Null-space can contain one element which si the zero vector

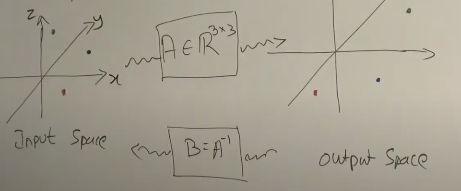
Column vectors: The new location of the basis vectors. Column space

Rank. Means the number of dimensions in the output. Also called the column space because it’s where the basis vectors land. The span of the column = the column space.



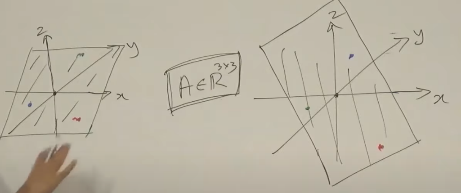
Full rank, unique 1:1 mapping between input and output space



Rank deficient there exists a lower dimension subspace, which must pass through the origin, and a corresponding matrix in the output matrix that also passes through the origin, with a 1:1 mapping.

**Properties of Linear Transforms**

Symmetric. A = AT. Has important properties: their Eigen values are **real** and their eigenvectors are **orthogonal** 



Quadratic form: xTAx, safe to assume A=AT. We use quadratic form to define PSD and PD. **The quadratic form is a scalar** if x is nx1, xT is 1xn, A is nxn so: (1xn)(nxn)(nx1) is (1x1). **Else, non-scalar**

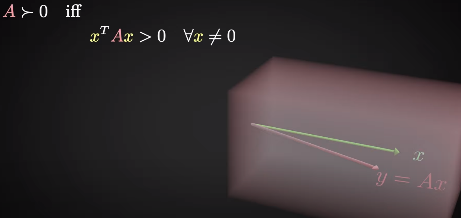
Positive Definite. Analogous to finding if a number is greater > zero. But how can we declare if a matrix is ‘positive’?

PSD if eigenvalues >= 0. If eigen values are negative then the direction is reversed

xTAx > 0

eigenvalues > 0. If negative then scaling direction is reversed so reject.

Given y=Ax, matrix A is PD if the dot product of y and x is > 0 for all x != 0. In other words, the transformed vector y is always on the same side as x. Why is PD useful? Tells you that global maximum exists!



Positive Semi-Definite if the dot product can equal zero, then at least one transformed vector y exists that is orthogonal to the input vector x.

* xTAx >= 0
* eigenvalues >= 0
* PSD have square roots. Analogous to positive scalar numbers.





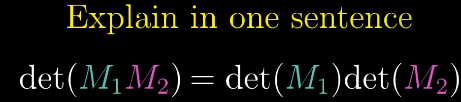
PSD programming was the hottest thing in mathematical programming in the 90s. It’s a special case of convex programming. Why is PSD useful? Lyapunov function. A linear dynamical system is **asymptotically stable** iff there exists a lyapunov function!

NP-Hard. No algo can solve efficiently.

traceSum of all elements in the diagonal of matrix. Only defined for square nxn matrix. It is used to [measure complexity](https://www.quora.com/What-is-a-trace-as-in-trace-of-a-matrix-and-why-is-it-used) of any linear machine learning model 

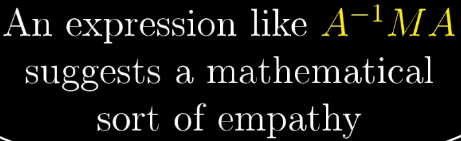
Determinant (scalar): measures how much a linear transform squishes or stretches a space. This tells us how *any* region in space will scale after the transform because all the grid space transformations are linear Determinants can be negative, which means the ‘orientation is flipped’.

* Determinant = product of all Eigen values
* Determinant = volume of output shape / volume of input shape
* Determinant of non-full rank matrix = 0
* Determinant = 0 means no inverse



Null-space. The space of all vectors that land on zero-vector. Full rank matrix only has one solution: the origin, rank deficient matrix have lots of vectors that land on zero.

Inverse matrix: reverse a transformation to find the original vector. Must be square. AA-1 = I

Change of Basis vectors are the coordinates of the grid. M is a transform in our language, A is transform to the basis of Jennifer’s.

Eigen vectors. Most vectors get knocked off their span after a transformation. Those that stay on the same original span don’t and are called Eigen vectors.

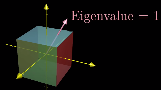
Eigen value. They are the scale factor for Eigen vectors.

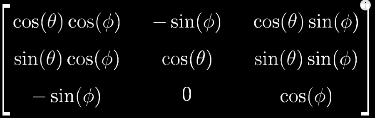
* Eigen values are only defined for square matrices.
* **If A is symmetric then eigen** values are real and eigenvectors are orthonormal

Spectrum: collection of Eigen values. For most matrix operations, is reflected by the spectrum.

Spectral theorem: for every square matrix that is A=AT there exists real **Eigen values and orthonormal eigen vectors**. Important because applies to

* Hessians
* Covariance matrix
* Kernel

Why it’s useful. Consider 3D rotation. If you find Eigen vector then you found the axis of its rotation. The corresponding eigenvalue is 1. Much easier to think about rotating about axis of rotation vs. its rotation matrix. 



The better way to get at the heart of a linear transform rather than thinking about its basis vectors is to think about its Eigen vectors.

Find Eigen vector by knowing after the linear transform, which squishes it into a lower dimension such that determinant = 0

 => 

Diagonal matrix all basis vectors are eigen vectors and all diagonals are the eigen values

OrthogonalUTU = I

**Operations**

Three types of multiplications

1. Vector-vector multiplication
2. Inner product = dot product. Result is scalar
3. Outer product results in matrix. Why is it useful?

**Inner vs. outer product**

* Inner product is the generalisation of the dot product.
* Inner products -> scalars.
* Outer products don’t output scalars

[[1]](#footnote-1)

**Care**: xTx != xxT if x = [1;2;3] you can get an inner our an outer dot product.

A picture containing text

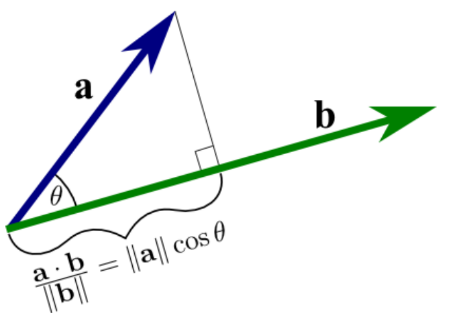
Description automatically generated

**Matrix-vector multiplication**

* A (m x n) x(n x 1) = vector (m x1)

**Matrix-matrix multiplication**

* A (m x n) x B (n x p) = C (m x p)

Projection is not the same as dot product. It’s the component of a vector.

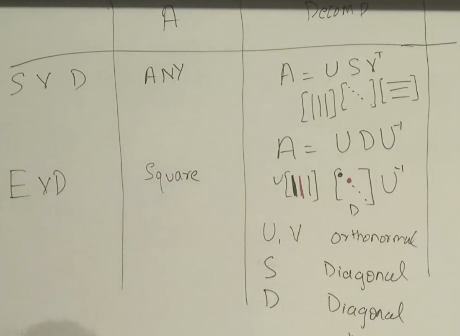
**Decomposition**

Single Value Decomposition The singular matrix values are all real. Rotate, scale, rotate another amount.

* Sledge hammer, can take any matrix A

Eigen Value Decomposition: rotate such that eigen vectors align with axes, scale, then undo the rotation.

* More conditions: Need square symmetric matrix



**Calculus**

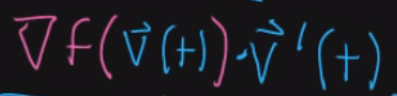


Chain rule



**Chain rule**

Dot product between gradient of v and vector derivative v. Interpret as the directional derivative

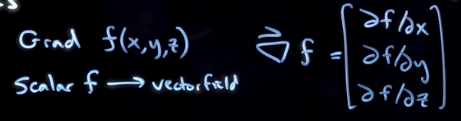
****

**Product rule** ∇xTAx = (∇xxT)Ax + xT(∇xAx)

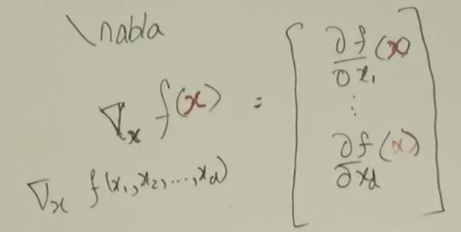
**Vector field** is the solution to a partial differential equation. The vector field also induces a new dynamical system, whereby if you dropped a particle into a vector field, it ‘sees’ the field dynamics, which you can use to predict its behaviour.

Div how much +div flowing away, -div is sucking in. Curl +curl is spin.

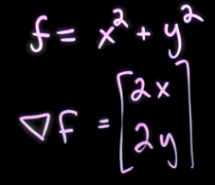
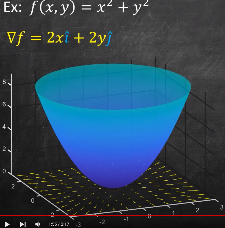
**Gradient** (subset of a jacobian). Takes a scalar f and returns a vector field. There’s a gradient value for every point in space! Think of it as fluids! It tells you which direction the temperature is increasing locally the fastest. *“Follow the gradient direction to get to the place the fastest!”*



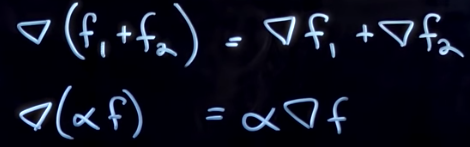
You can write grad in a few ways. Still the same thing because the output of a gradient is a scalar.



Example of gradient is the paraboloid. The gradient is a vector field. *f* in this case is the height of the paraboloid. Large gradient means its steep. Low gradient means its shallow. 0 gradient is no change in any direction.

 ****

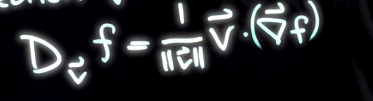
Grad is a linear operator. Steve Brunton says this property is very important because superposition holds.



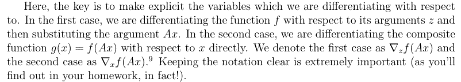
We use the gradient to compute the directional derivative. The derivative of *f* in a direction *v*. We literally just take the dot product. If dot product is zero then the gradient isn’t changing.

****

Typically should see it normalised because it doesn’t matter what the magnitude of v is. Just want to know the change ratio.

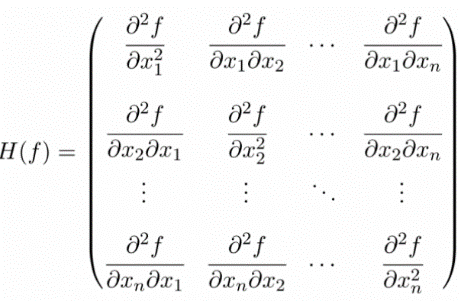
****

Must make explicit the variables which we are differentiation with respect to. You will get different gradients if you differentiate function f wrt to arguments z, and then substitute the argument Ax, vs. if you differentiate a composite function directly.



Hessian akin to second derivation but few caveats to keep in mind. The gradient of a function is a vector, and we cannot take the gradient of a vector – it’s undefined. Hessian is really taking the gradient of each entry of the gradient, not the whole gradient of the vector.

* Positive Definite Hessian = Local Minimum
* Negative Definite Hessian = Local Maximum
* Indefinite Hessian = Saddle point
* Other definiteness = Test is inconclusive



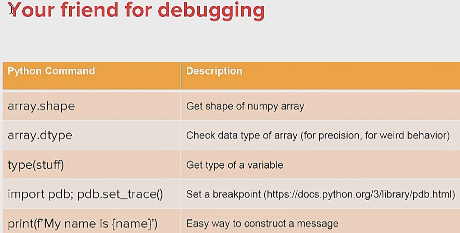
**Reducing computation time**

Spectral theorem goal is to find diagonal of a matrix because it reduces computation. A =UΛUT  “perhaps one of the most important theorems in linear algebra”

conditions: A=AT

numpy





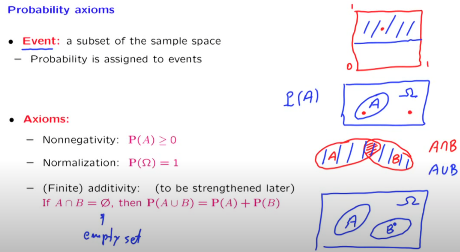
# Probability

**Fundamentals**

Sample space



Outcomes are red



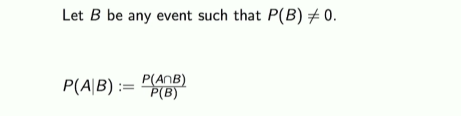
Event vs. outcomes

Outcomes is all possibilities whereas events are outcomes of interests. E.g. roll a die: outcomes: {1,2,3,4,5,6} whereas events are groups of outcomes e.g. even numbers on a die.

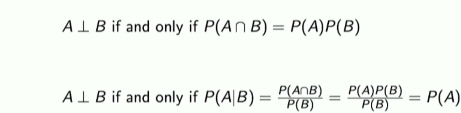
Definitions

* Sample samples are sets.
* Sample space is the set of all possible outcomes.
* Events are sets of outcomes.
* An event is not ‘a thing that happens’ as some stats101 students think (me). The empty set is an event of every experiment: represents the impossible. The sample space is an event of every experiment, represents the certain.
* *“We assign probabilities to* ***events******not outcomes****”*

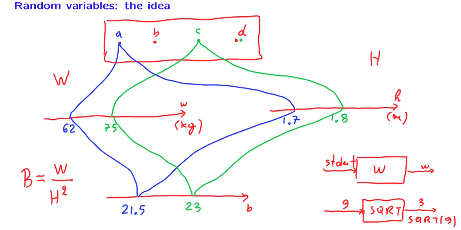
Conditional probability



Independence of events



Random variable *“Random variables are functions already” “is a very poor name, it’s neither random nor a variable. It’s just a* ***function*** *that* ***maps******outcomes*** *to real values.”* We are *not talking* about events.



Cumulative Distribution Function: exists for all random variables

* Fx(-inf) = 0
* Fx(+inf) = 1
* 0 <= Fx(x) <= 1
* Non, decreasing
* P(x1 < X <x2) = Fx(x2) – Fx(x1)

Probability mass function.

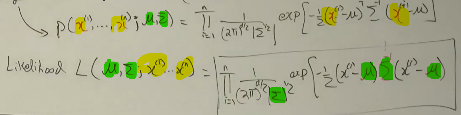
* The probability is given by the height of the PMF value.

Probability Density Function,

* Continuous RV
* Derivative of CDF
* The value returned by the CDF is not the probability. Unlike PMF, the height of the CDF are only defined on intervals, i.e. the area.
* You need to bin. Gaussians are real-values PDFs.

Likelihood function

PDFs can be repurposed. It’s basically the same expression but interpreted differently. In PDF, variables is the data. In Likelihood function, variables are the parameters.

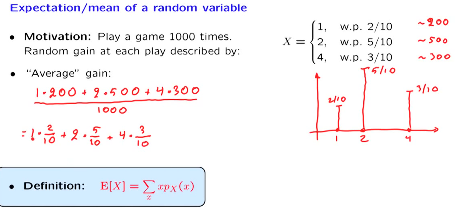


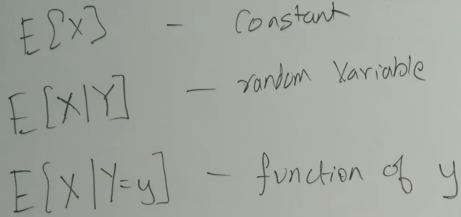


**Discrete Random Variables**

Expectation of a random variable[[2]](#footnote-2)

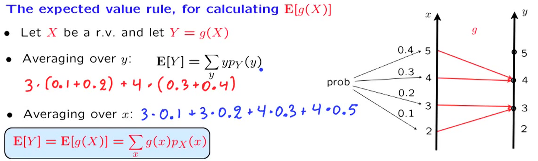
Aka expected value. A concept that is only associated with random variables. Intuition: the centre of mass.



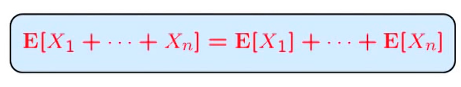


Relevant for homework. Random variable is already a function.

The Expected Value Rule E[g(x)]



Linearity of Expectations

We use this tool to break up complicated random variables and analyse them separately. 



Law of total expectation

This holds true for ANY X and Y, dependent or independent. Purpose is to choose Y deliberately to evaluate E[X].

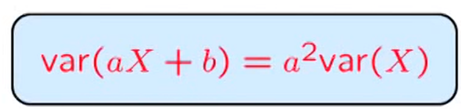
E[X] = E[E[X|Y]]

Variance The square penalises data points that are very far from the mean.





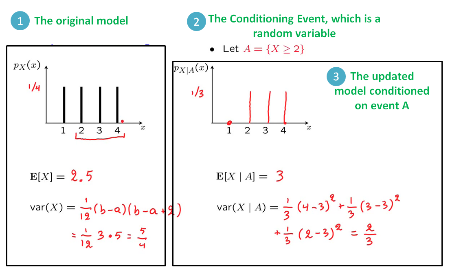
Properties of variance





Conditioning on events

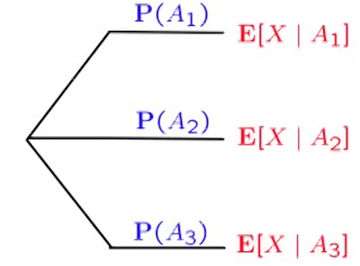
*“To conclude, there is nothing really different with conditional PMFs, expectations and variances except we have to use conditional probabilities throughout instead of the original probabilities”.*

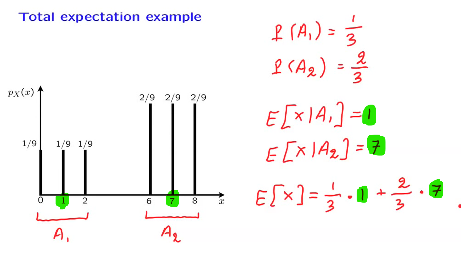


Total expectation theorem

Conditioning is useful because of divide and conquer. It’s the backwards forwards algorithm.

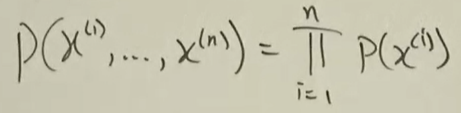


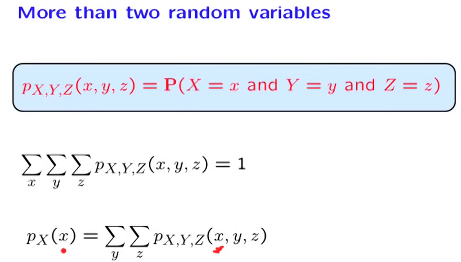




Joint random variables and joint PMFs

Joint probabilities is the multiplication of each of the marginals

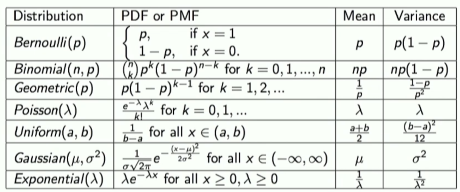
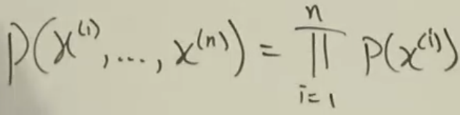




Expected value of joint random variables



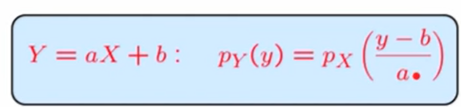
Distributions

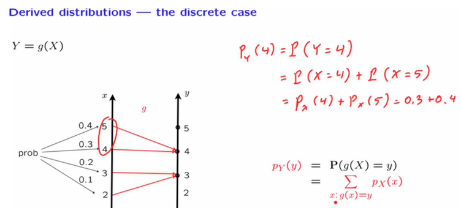


**Derived Distributions**

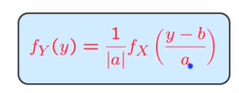
Motivation: We have basic random variables and then we use that to build up more complex models.

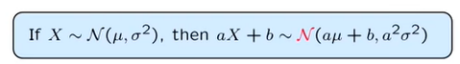
PMF of function of discrete random variable





Linear Function of continuous random variable

Linear Function of normal random variable

Final PDF is also normal

PDF when Y=g(X) where g is monotic

Final PDF is also normal

PDF of function of multiple random variables

Watch the MIT youtube video. I don’t get it.

**Sum of independent RVs, Covariance**

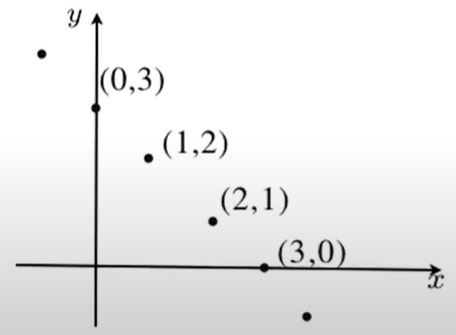
For either discrete and continuous there is a formula and a diagrammatic technique to find this.

Find PMFs of the Sum of Independent Discrete Random Variables

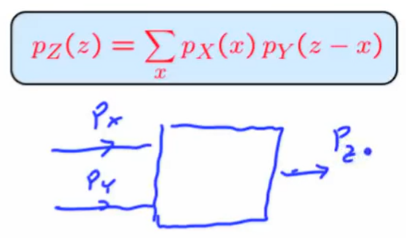
Suppose X, Y are independent and discrete with known PMFs g(X,Y). What is pz(3)?

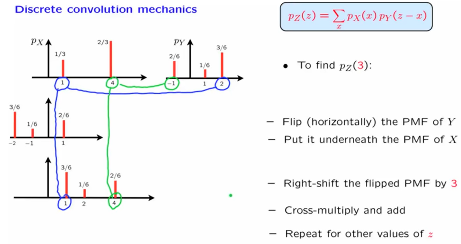
pz(3) = P(X=0, Y=3) + P(X=1,Y=2) + …

pz(3) = Px(0)PY(3) + = Px(1)PY(2) + …

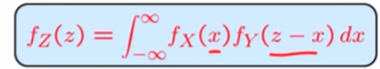


General form is the convolution formula





The Sum of Independent continuous Random Variables



The Sum of Independent Normal Random Variables[[3]](#footnote-3)



When we work with normal RVs, we stay in the realm of normal RVs. The sum is also normal



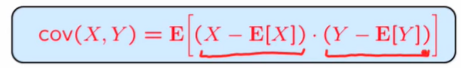
Covariance of two random variables

Tells us how related two random variables are. If covariance tells us whether two random variables move in the same direction together

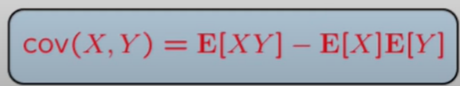
* If independent then covariance always = 0
* But you can have dependent and covariance = 0. So be careful

**But DIFFICULT to interpret qualitatively: normalised, units are squared. More useful is Correlation coefficient which is the dimensionless version of covariance.**

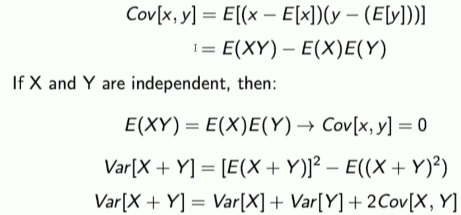
The general forms below



The below version is a reformulation of the above but the MIT video says that this is easier to work with!

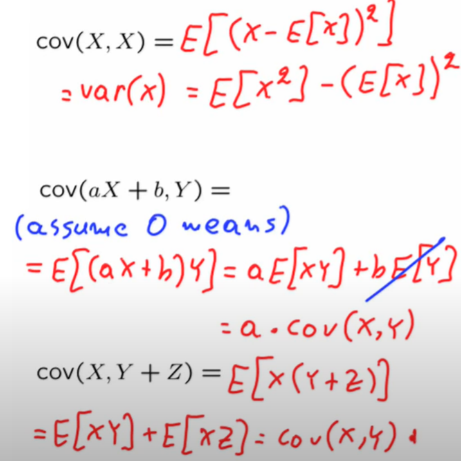


From the Stanford lectures



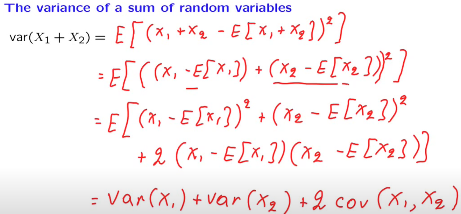
Covariance properties[[4]](#footnote-4)

1. Covariance with itself is the variance
2. Covariances are symmetric
3. Covariances are always PSDs

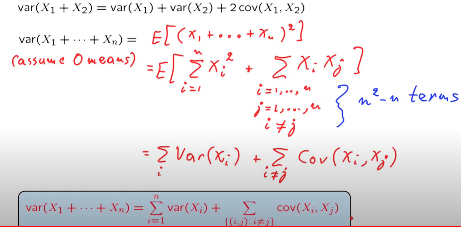


The Variance of sum of random variables

In the case of independence the covariance is 0 and we just have the sum of variances.



And the sum of MANY random variables



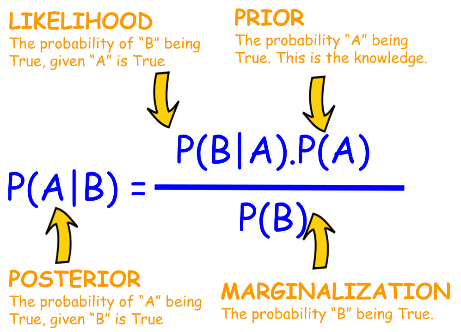
Joint distribution probability of joint distribution of independent random variables

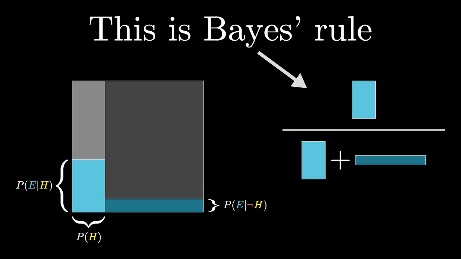
Independence of Random variables

Important because it let’s use decompose joint probabilities into product of marginal, makes machine learning tractable.

Bayes Rule

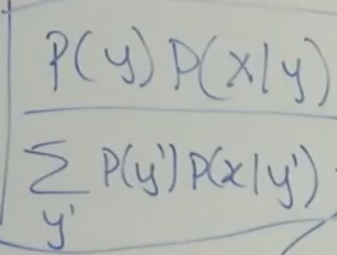
inference technique that looks back at our prior dataset to assess partially observable outcome that we see now.

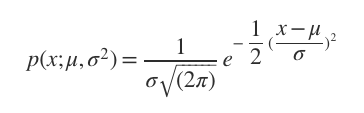




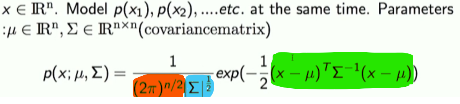
*This is helpful because we often have an asymmetry where one of these conditional probabilities is easy to compute and the other is not.* Furthermore *Bayes' Theorem matters because the math shows that the intuitive understanding of the world is kind of crap.*

It works for conditional probabilities too.



Univariate Gaussian distribution 

Central Limit theorem

Multivariate Gaussian distribution 

Sigma is called the covariance matrix. It generalised the concept of variance to multiple dimensions.

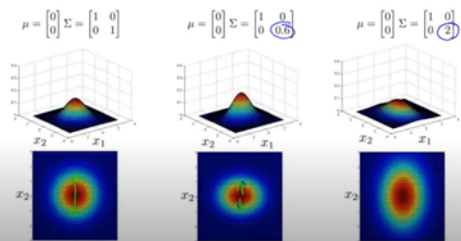
Properties

* The covariance matrix is always symmetric and PSD. Any covariance matrix for any joint distribution is always PSD. It is full rank, which means inverse exists.
* Notation for the square root of the determinant |covariance matrix|0.5
* We also see the quadratic form in (x-mu) covariance matrix -1(x-mu)
* Constant 1/2pin/2 is to normalise everything to 1

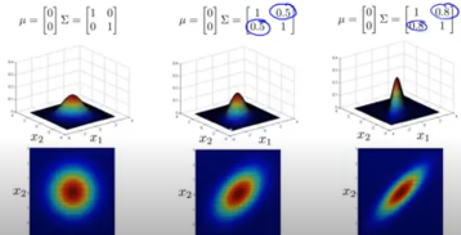
In below example, we have two features x1, x2. The mean is at the origin. Sigma is the covariance matrix. If elements of sigma are diagonal, then we are scaling the probability distribution in the directions of x1, x2.



If the diagonal elements aren’t the same then we stretch or shrink one axis relative to the other.



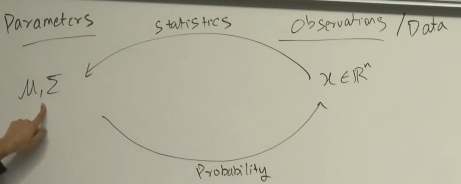
If we have off diagonal entries then we introduce shear.



# Statistics

Probability vs. statistics Probability is inference. Statistics is deriving parameters given data: techniques include Maximum likelihood estimation (most relevant for ML). Probability is using the parameters to make inferences.

Where does ML come in? It does both arrows!

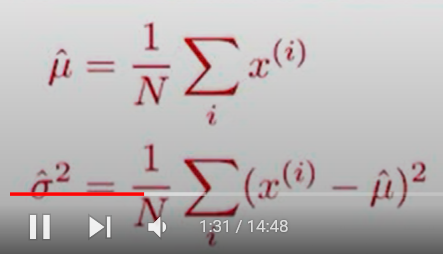


Maximum Likelihood Estimation (MLE) assumption data is sampled independently and identically distributed (IID). Objective is to find **µ** and covariance matrix **Σ**

* *Likelihood of parameters given data*. The **shape of the likelihood** function is given by the data. Integration of likelihood may not necessarily equal to 1. For different sets of data, get different likelihoods of parameters. We choose the parameters based on the highest likelikehood.
* *Probability of data given parameters*. Integration equals 1. The parameters set the shape. This is a completely different function to the likelihood of the parameters function.

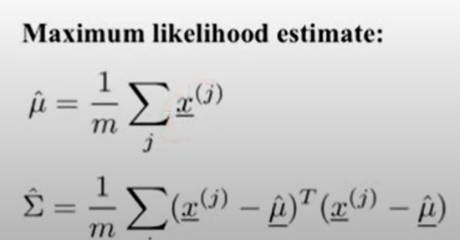


Generic procedure for Univariate

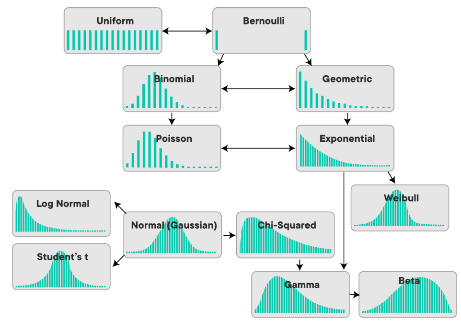


Generic procedure for Multivariate

* Assume the likelihood function distribution of the data. Generally 
* Find argmax **µ** and **Σ** of the likehood function.
* Take the gradient of the likelihood function and set to zero
* Take log of the Gaussian distribution, valid gradient because it is monotonically increasing.



All Distributions



Functional Analysis

The study of functions. Intuition: linear algebra in infinite dimensions. This is a good perspective to have according to Prof. Anand.

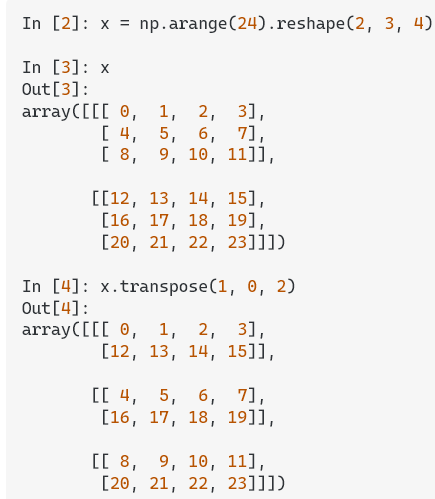
*“think of functions as points in infinite dimensional space”,* where the height of this point is the function itself. Switching between

# Python tricks

**3D matrix multiplication[[5]](#footnote-5).**

3D matrix multiplication is nothing but a collection (or a stack) of many 2D matrices.

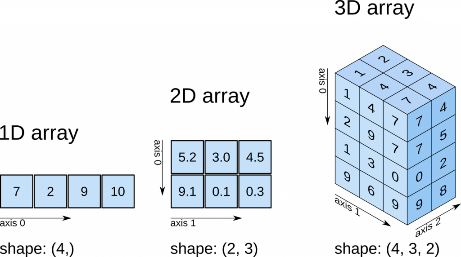
**3D matrix transpose**



**Divide by zero protection**

np.divide(arr1,arr2)

sum axis



# Overleaf tricks

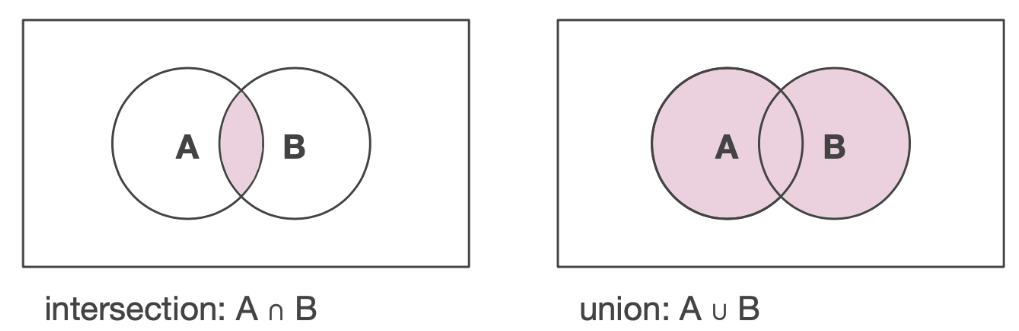
If you set up folders, each with their own main pages, and have a top level main, you need to set the main document.



# Probability tricks

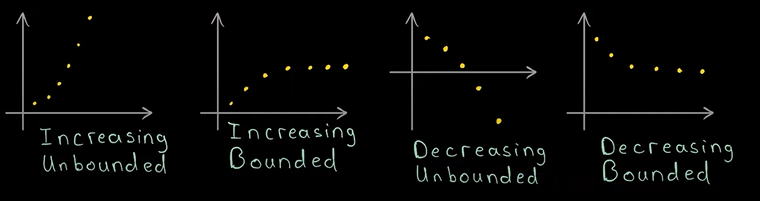
Joint probability P(X,Y) is actually implied by P(x)P(Y|X). Think about it, you start with probability x, then multiply the conditional of y on x. P(X,Y) is not the same as P(X)+P(Y).

I mixed two fundamental concepts: when to add probabilities vs. when to multiply probabilities. Joint probabilities is multiply (intersection). Add (union) probabilities when you want to know whether one event or another could happen.



Monotone

* Increasing an+1 >= an
* decreasing an+1 <= an



Diagram

Description automatically generated

Graphical user interface, text, application

Description automatically generated

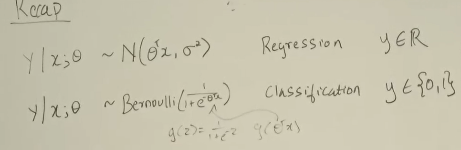
A picture containing table

Description automatically generated

# Discriminative

## Linear Regression

* Linear because thetaTx. The data type of y value informs our choice of the distribution. It doesn’t make sense to use Bernoulli for real values of y and it doesn’t make sense to make
* Taking the moments means to find the mean and variance



Add matrix intercept column

Required for continuous variables. I am dumb ☹ <https://stats.stackexchange.com/questions/440242/statsmodels-logistic-regression-adding-intercept>

## Logistic Regression

Hypotheses such that always 0 or 1

Text, letter

Description automatically generated

Log Loss Function

Average empirical loss for logistic function

**Text, letter

Description automatically generated**

Derivative of log loss function

Text

Description automatically generated

Text

Description automatically generated

Same update rule as perceptron and linear regression, the only difference is htheta(x). Not a coincidence

Hessian of log loss function

Gradient and Hessian for coding

**Text, letter

Description automatically generated**

Stochastic ascent



Perceptron



Newton’s method

Alternative to gradient descent. The intuition for Newton’s method: to find min of loss function, find when its derivative is zero.

Newton’s method is a **root finding method**. It converges faster than gradient descent.

Procedure

1. Approximate is a linear approximation of the first derivative @ a point
2. Solve for the root of that linear function.

A picture containing text, whiteboard

Description automatically generated

**Newton’s Raphson method**

The vector version of newton’s method. Alpha is step size. The decision criteria for choosing Newton Raphson vs. GD is how big is dimension of Hessian. If too large, then Newton Raphson will be slow O(dimension of Hessian).

Limitations

* Newton Raphson will only take you nearest stationary point which can be local minimum OR maximum (yikes)
* You may not have invertible Hessian

A picture containing text, whiteboard

Description automatically generated

these two perspectives are useful for gaussian and gradient boosting.

## Generalised Linear Models

# Generative

## MLE

**Continuous Multivariate**

* Assume a likelihood function distribution of the data (Bernoulli, poisson, uniform, normal etc.)
* Generally we assume gaussian:
* Find argmax **µ** and **Σ** of the likehood function.
* Take the gradient of the likelihood function and set to zero
* Take log of the Gaussian distribution, valid gradient because it is monotonically increasing.

**Discrete (Bernoulli)**

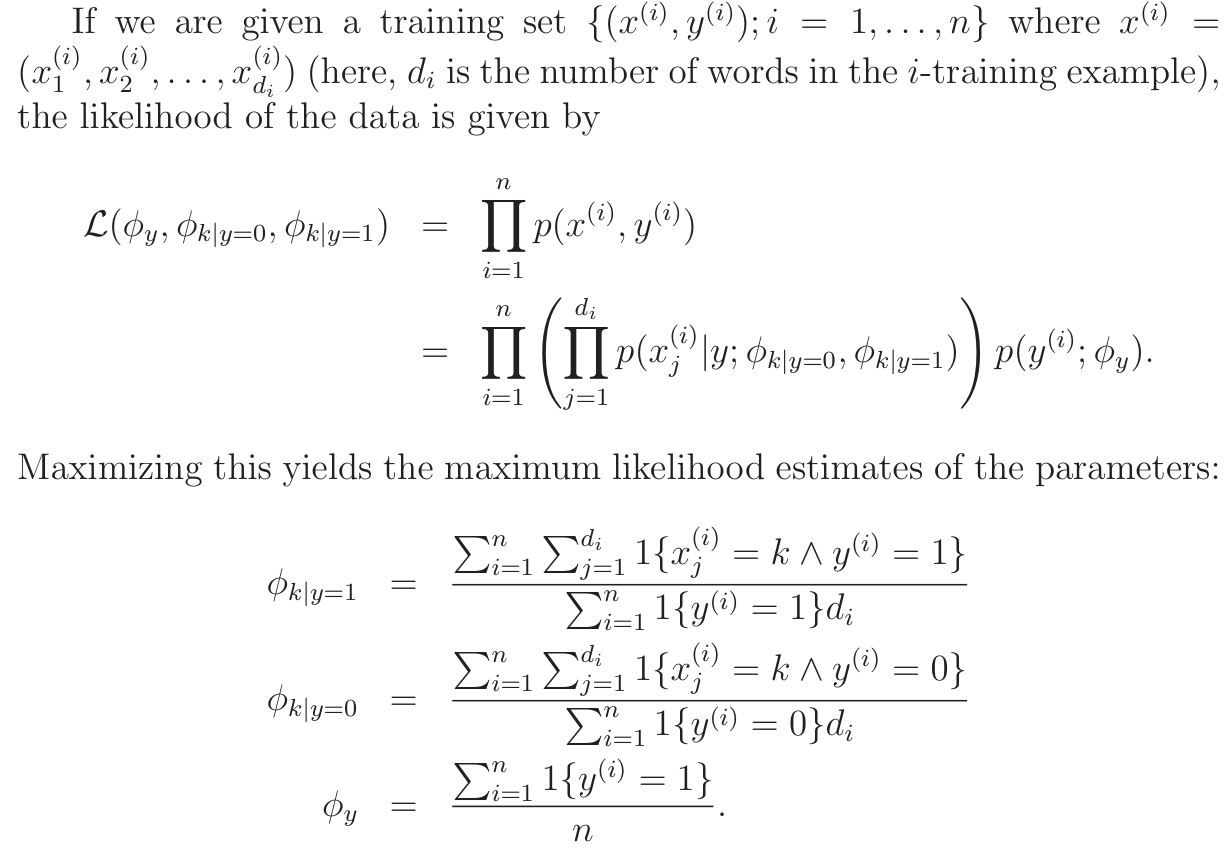
From CS229 notes 2, p10

Text

Description automatically generated

**Discrete (Multinominal)**

CS229, p14, notes2

****

## GDA

Assumptions

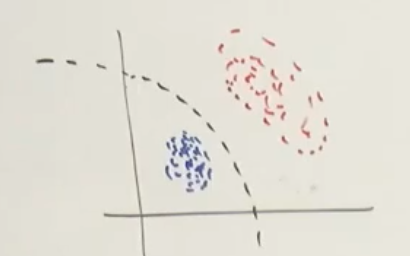
* Classes share the same covariance matrix.
* The data is gaussian.

The line is a straight line given the above assumptions

A white paper with writing on it

Description automatically generated with medium confidence

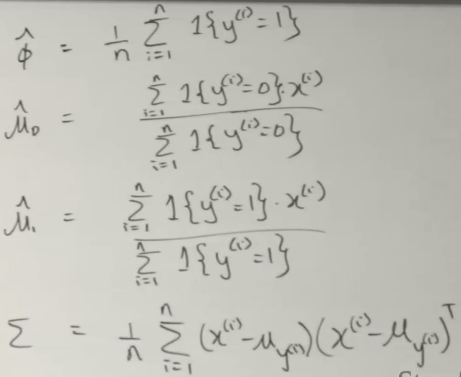
If different covariance, distribution of classes different. Separation line is curved. *“Logistic regression with polynomial features”*



Logistic regression is more robust. Logistic regression is always your first choice in real life. GDA is more efficient only if the assumptions hold true.

Procedure

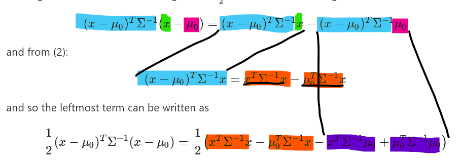
1. Write log likelihood function. We take log because it’s a monotonically increasing function.
2. Solve for arg max (MLE)



A white paper with black writing

Description automatically generated with low confidence





If you have GDA model, its posterior is always a logistics model. Not true the other way.

## Naïve Bayes

# Kernels

Gives us a way to compute dot product in features space without even knowing what this space is and what is phi[[6]](#footnote-6).

## Kernel perceptron

# Deep Learning

## Regularisation

## Relationship between MAP, MLE, Bayes theory

[**https://www.youtube.com/watch?v=y7KJcCltR5Y**](https://www.youtube.com/watch?v=y7KJcCltR5Y)

## Relationship between KL Divergence and Cross-Entropy

[**https://www.youtube.com/watch?v=SxGYPqCgJWM**](https://www.youtube.com/watch?v=SxGYPqCgJWM)

# Learnings from Past Exams

## Math

* (Ax)2 = (Ax)TAx
* (ABC)T= CTATBT

## Statistics

* Exponential family doesn’t have infinite entropy
* Represent P(y < u) as CDF(u). CDFs always ‘downwards’

## Lin regression

## GDA

* Whole family of generative models have posterior distributions that can be written in the form of the logistic function

## Linear models

* Decision boundary is P(y=1|data) > P(y=0|data). To find the parameters w and b, take the log and simplify into the linear form.
* Decision boundary is -b when h(x) = wTx +b

## Loss functions

* Logistic loss more generalisable than exponential loss because its less sensitive to mistaken levels as it grows only linearly for mis-classifications rather than exponentially

## GLMs And Exponential Fam

* Logistic loss more generalisable than exponential loss because its less sensitive to mistaken levels as it grows only linearly for mis-classifications rather than exponentially

## Training

* Fewer training examples = underfit

## Convergence

* Newton’s method could converge faster than SGD because it could take fewer steps
* If learning rate is fixed, parameter \theta can never converge exactly to optimal \theta\*. The learning rate needs to decrease to zero

## Outliers

* Logistic regression is affected by outliners. Because all training examples are considered
* SVM is not affected.

## Bias vs. variance

* Bias if model is insufficiently complex
* Variance if test results are sensitive to training data.

## Regularisation

## MAP and MLE

* MAP adds regularisation
* MLE only works for regression. Counting is for discrete.

## Gaussian Processes (GPs)

* Have a closed form solution on the kernel matrix is evaluated
* But can’t be trained by gradient ascent or SGD or SGA because GPs are a Bayesian approach which means no loss function to minimise. We condition on the observed data and construct a posterior distribution.

## Naïve Bayes

* Bernoulli vocab with d words and 3 classes creates 3d+2 parameters, 2 for p(y) and 
* Multinominal (based on msg) where vocab is d words and three classes results in 2 parameters for p(y) and d-1 parameters for each of the 3x multinomials
* NB multivariate and multinominal, 2 classes, vocab size V, email length L: num params = 2-1 classes x 2 classes x V words = 1+2V params
* How many look ups of p(xj|c) on average to make a single spam or not spam classification
  + Multivariate NB involves 2V lookups (1xV for each class)
  + Multinomial involves 2L lookups
* Which method badly affected by stop words? Multinominal since P(x|) is factored din every time x appears, whereas multivariate NB only considers whether x appears at all.
* In NB, we assume xi is indep of all other features xj. True or false? False, in both cases xi and xj are conditionally independent of y.
* NB is a linear classifier

## Kernels / SVMs

* Only SVMs can linearly separate perfectly separated data
* Kernel perceptron works for non-linear (by adding more dimensions to the dataset) whereas linear classifiers don’t of course
* Mixtures use GDA. Don’t use SVMs.
* SVMs don’t output probabilities

## MDPs

* Num of distinct polices for S states and A actions is SA polices

## K-means

* Can be combined with dradient descent update rule via distrortion function

## EM algorithm

* EM is not guaranteed to monotonically increase with each iteration

1. <https://www.inf.ed.ac.uk/teaching/courses/cfcs1/lectures/cfcs_l10.pdf> [↑](#footnote-ref-1)
2. <https://ocw.mit.edu/courses/res-6-012-introduction-to-probability-spring-2018/resources/expectation/> [↑](#footnote-ref-2)
3. <https://www.youtube.com/watch?v=aGbP_7yAiEk> [↑](#footnote-ref-3)
4. <https://www.youtube.com/watch?v=RQKJBpaCCeo> [↑](#footnote-ref-4)
5. # <https://www.geeksforgeeks.org/numpy-3d-matrix-multiplication/#:~:text=A%203D%20matrix%20is%20nothing,between%20their%20row%2Fcolumn%20vectors>.

   [↑](#footnote-ref-5)
6. <https://stats.stackexchange.com/questions/152897/how-to-intuitively-explain-what-a-kernel-is> [↑](#footnote-ref-6)