Computing for Data Sciences

Lecture 10 - 11

Definitions:

A norm is a function that assigns a strictly positive length or size to a vector in a vector space. Similarly a matrix norm is a natural extension of the notion of a vector norm to matrices.

Matrix Norm:

Let $R^{m\times n}$ denote the vector space containing all matrices with *m* rows and *n* columns with entries in R.

A matrix norm is a vector norm on $R^{m\times n}$. That is, if $\|A\|$ denotes the norm of the matrix *A*, then,

$.\|A\| \geq 0$

$.\|A\| = 0$ iff $A = 0$

$.\|\alpha A\| = |\alpha|\|A\|$ for all α in R and all matrices in A in $R^{m\times n}$

$.\|A + B\| \leq \|A\| + \|B\|$ for all matrices A and B in $R^{m\times n}$

Additionally, in the case of square matrices (thus, $m = n$), some (but not all) matrix norms satisfy the following condition, which is related to the fact that matrices are more than just vectors.

There are three important types of matrix norms. Let A be some matrix

1. **Induced norm:** The induced norm corresponding to the p-norm for vectors is:

$$\|A\|_p = \sup_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$

In the case p = 1 and p = ∞, the norms can be computed as:

$$\|A\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}|,$$

$$\|A\|_\infty = \max_{1\leq j\leq n} \sum_{j=1}^{n} |a_{ij}|$$

2. **Element-wise norm:** It is like unwrapping A into long vector, then measuring its vector norm. It treats an *m x n* matrix as vector of size *mn* and use one of the familiar vector norm.
   E.g. using p-norm for vector

$$\|A\|_p = \|vec(A)\|_p = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^p \right)^{1/p}$$

3. **Schatten norm:** It measures the vector norm of the singular values of A. If the singular values are denoted by σ_i, the Schatten p-norm is defined by

$$\|A\|_p = \left(\sum_{i=1}^{\min\{m,n\}} \sigma_i^p \right)^{1/p}$$

So the matrix norm can be generalized to the $L_{p,q}$ norm as:

$$\|A\|_{p,q} = \left(\sum_{j=1}^{n} \left(\sum_{i=1}^{m} |a_{ij}|^p \right)^{q/p} \right)^{1/q}$$

Frobenius Norm

Frobenius norm or the **Hilbert–Schmidt norm** is defined by $L_{p,q}$ norm (Element-wise norm) for $p = q = 2$, or by Schatten norm for p=2 as:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{trace(A^T A)} = \sqrt{\left(\sum_{i=1}^{\min\{m,n\}} \sigma_i^p \right)}$$

Where A^T denotes the conjugate transpose of A, σ_i are the singular values of A, and the trace function is used. The Frobenius norm is similar to the Euclidean norm on R^n.

Singular Value Decomposition (SVD):

In the previous lectures we have covered the concepts, formulation and geometrical interpretation of SVD. We can summarize SVD quickly through figure 1 and figure 2 as:

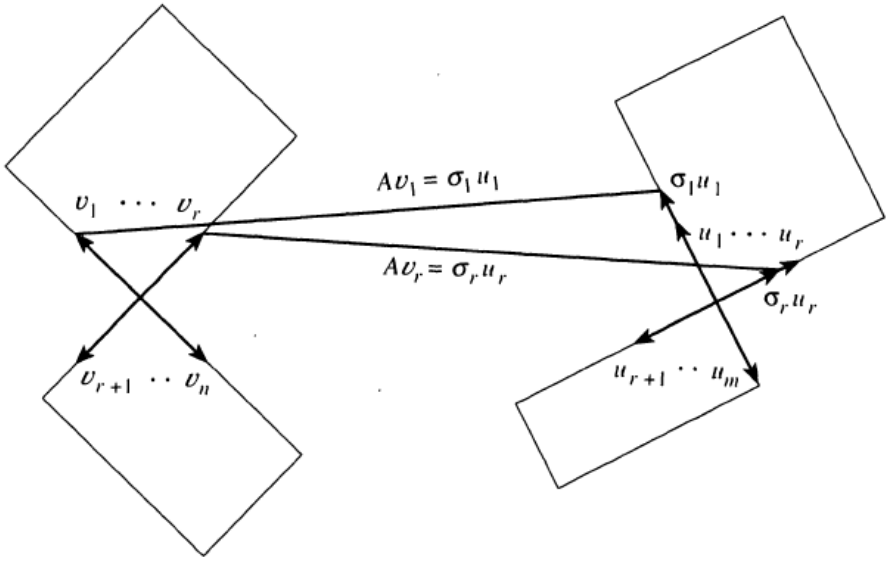


Figure 1: Orthonormal basis that diagonalize A

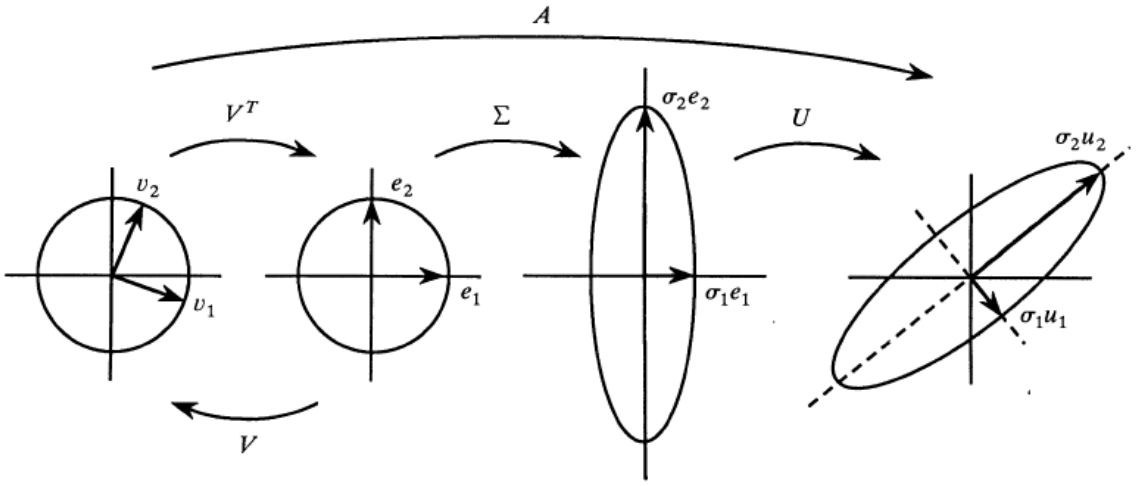


Figure 2: Geometrical interpretation of SVD

The figures clearly explained what SVD does and what are all u_i , v_i, *and* σ_i and in which subspace they belong.

In this lecture we will explore the idea of using SVD in finding the best k-dimensional subspace with respect to set of points from n data points in a d-dimensional space.

Best-Fit Subspaces and Singular Value Decomposition (SVD):

Think of the rows of an $n \times d$ matrix A as n data points in a d-dimensional space and consider the problem of finding the best k-dimensional subspace with respect to the set of points. Here best means minimize the sum of the squares of the perpendicular distances of the points to the subspace.

Let's begin with a special case where the subspace is 1-dimensional, namely a line through the origin. The best fitting k-dimensional subspace is found by repeated applications of the best fitting line algorithm, each time finding the best fitting line perpendicular to the subspace found so far. When k reaches the rank of the matrix, a decomposition of the matrix, called the *Singular Value Decomposition (SVD)*, is obtained from the best fitting lines.

The singular value decomposition of a matrix A is the factorization of A into the product of three matrices,

$$A = UDV^T,$$

where the columns space of U and V are orthonormal and the matrix D is diagonal with positive real entries. In many applications, a data matrix A is close to a low rank matrix and a low rank approximation to A is desired. The singular value decomposition of A gives the best rank k approximation to A, for any k.

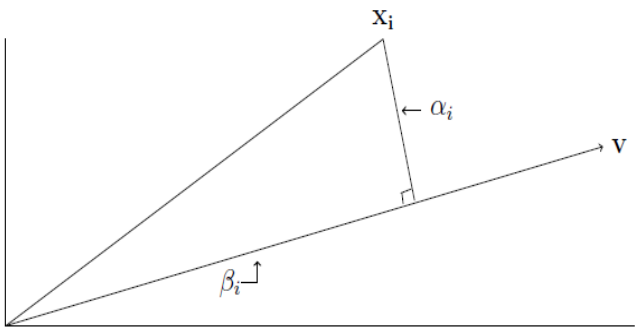
The singular value decomposition is defined for all matrices, whereas the more commonly used eigenvector decomposition requires the matrix A to be square and certain other conditions on the matrix to ensure orthogonality of the eigenvectors.
In contrast, the columns of V in the singular value decomposition, called the *right-singular*

vectors of A, always form an orthonormal set with no assumptions on A. The columns of U are called the *left-singular vectors* and they also form an orthonormal set.
A simple consequence of the orthonormality is that for a square and invertible matrix A,

$$A^{-1} = VD^{-1}U^T.$$

Project a point $a_i = (a_{i1}, a_{i2}. \ . \ . a_{id})$ onto a line through the origin.

$$a_{i1}^2 + a_{i2}^2 + \ldots + a_{id}^2 = (\text{length of projection})^2 + (\text{distance of point to line})^2$$
$$(\text{distance of point to line})^2 = a_{i1}^2 + a_{i2}^2 + \ldots + a_{id}^2 - (\text{length of projection})^2$$



Minimizing $\sum_i \alpha_i^2$ is equivalent to maximizing $\sum_i \beta_i^2$

Figure 3: Projection of the point x_i onto the line through the origin in the direction of v.

$\sum_{i=1}^{n} (a_{i1}^2 + a_{i2}^2 + \ldots + a_{id}^2)$ is a constant independent of the line.

For minimizing the sum of the squares of the distances to the line is equivalent to maximizing the sum of the squares of the lengths of the projections onto the line. Similarly for best-fit subspaces maximizing the sum of the squared lengths of the projections onto the subspace minimizes the sum of squared distances to the subspace.

There are two interpretations of the best-fit subspace:

1. It minimizes the sum of squared distances of the data points to it.
2. It maximizes the sum of projections squared of the data points on it. In other words the subspace contains the maximum content of data among all subspaces of the same dimension.

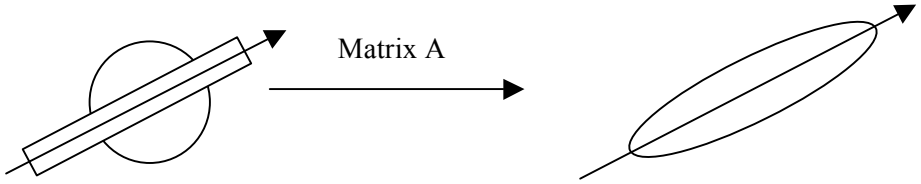
Singular Vectors:

Consider a matrix A of size *m x n*. We already know that matrix alone doesn't do anything. It is a function which operates on vector.

Av: The matrix A do linear transformation (Rotation and Scaling) on vector v.

The matrix A changes the size of n dimensional vector to m dimensional vector and rotates it. Also matrix

doesn't scale the vector equally in every direction. So geometrically the operation of function i.e matrix A can be interpreted as shown below:



Sphere Ellipsoid

The line through the ellipsoid major axis is the line of 1st principal component i.e sphere scaled the most in the direction of 1st principal component or the direction (shown in fig) of sphere in which the matrix A operates the most.

We can write for finding the first principal component as:

$$A\, v_1 = u_1\, \sigma_1$$

OR

$$v_1 = \arg\max_{|v|=1} |Av| \qquad \text{.....(1)}$$

where

v_1 : First singular vector of A.

u_1 : Unit Vector along most scaled direction(gives the direction)

σ_1 : Scaling parameter in most scaled direction(gives the magnitude of scaling or first singular value of A)

arg max|f(*x*)| : gives the value of *x* for which the max|f(*x*)| (if its exists) is attained) and there could be more than one *x* value which give rise to max|f(*x*)|. Here in the context vectors, arg max|Av| give the vector which gives the maximum operated direction.

and the first singular value of A can be written as

$$\sigma_1(A) = |Av_1|$$

But the action of matrix A doesn't stops only in the 1st principal direction. So have to figure out the next most scaled/operated direction. For finding the 2nd principal component, drop the first component dimension i.e. it doesn't mean that drop a column but look for direction which is orthogonal to $v_1$and scaled most after first component.

$$v_2 = \arg\max_{v \perp v_1,\ |v|=1} |Av| \qquad \text{.....(2)}$$

Now for finding the third principal component drop the first and second component i.e. look for

direction which is orthonormal to both v_1 and v_2 and scaled most after second component.

$$v_3 = \arg\max|Av| \quad v \perp v_1, v_2 \quad |v|=1 \qquad \text{.....(3)}$$

and so on until the rank of matrix A.

$$\max_{\substack{v \perp v_1, v_2, \ldots, v_r \\ |v|=1}} |Av| = 0$$

Given rank *r* of matrix, the singular value of *(r+1)th to mth* principal component are zero.

$\sigma_i = 0$ for i = *(r+1)* to *m(dimension of matrix A)*. [*(r+1) <= m*]

Also the order of magnitude of the *ith* singular values are :

$\sigma_1 >= \sigma_2 >= \ldots >= \sigma_k$

This means that *ith* principal component have more information content than the *(i+1)th* principal component.

Now apply the rank *k* approximation algorithm using the k principal components computed before.

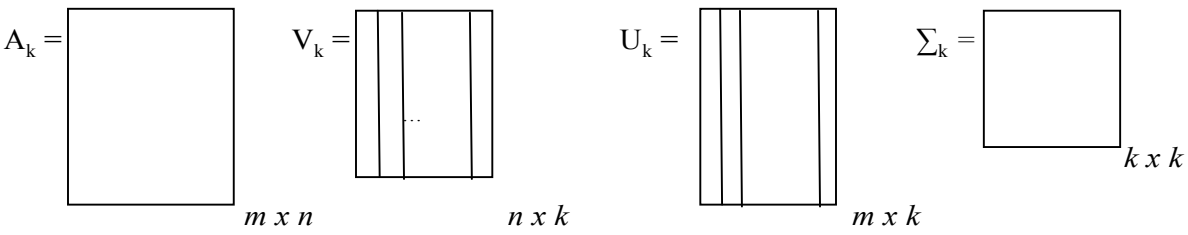
Consider the following:

A_k : *Rank k approximation of the original matrix A.*

V_k : *composition of first k principal components(singular vector) of matrix A.*

U_k : *Composition of first k* u_i *vectors for the corresponding* v_i *vectors.*

$\sum_k$: *Diagonal matrix contains first k singular values of corresponding singular vectors.*



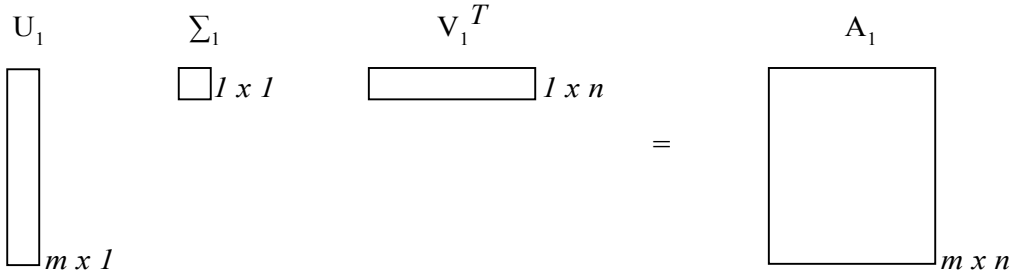
Rank 1 approximation:

For calculation of the rank 1 approximation we can write the A as

$$A_1 = u_1 \sigma_1 v_1^T$$

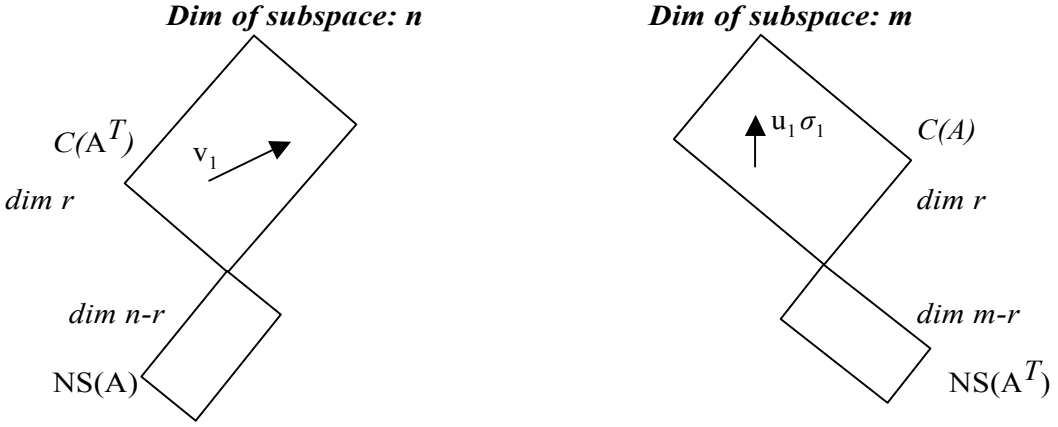
And this A_1 matrix is not the matrix we started with A. This A_1 matrix is rank 1 matrix and Rank

1 approximation of original matrix A.



Here U_1, $\sum_1$ and V_1^T has the vector u_1, σ_1 and v_1^T. So for every u_i, σ_i and v_i^T from equation (1), we calculate the A_1 matrix.

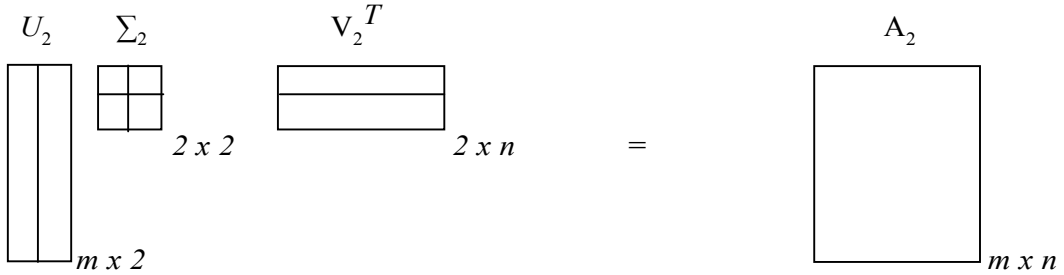
Also we can view the above transformation by matrix A in 4 subspaces as:



So for all v_i and u_i , we generate the A_1 matrix and so end up with '*k*' **rank 1 matrix.** So by finding the set of vectors v_1, we get the rank 1 approximation of original matrix A.

Rank 2 approximation:

Using the equation (2) on previous page, we get the set of vectors v_2 , such that all v_2 are orthonormal to the all set of v_1 vector from equation (1). So here we are approximating the original matrix A using the composition of two principal components v_1 and v_2.



where V_2^T contains composition of $<v_1$ and $v_2>$. For the respective $<v_1$ and $v_2>$, $\sum_2$ contains the $<\sigma_1$ and $\sigma_{2>}$ along the diagonal elements in the *2 x 2* diagonal matrix and U_2 contains the corresponding pairs of $<u_1$ and $u_2>$. So here the matrix A_2 generated is of rank 2 and called the rank 2 approximation of the original matrix A.

Rank k Approximation:

Similarly rank k approximation, we choose the combination of *k* principal components and calculate the A_k matrix . The A_k matrix generated have the rank *k.* This matrix is called Rank k approximation of the original matrix A.

If *k* is equal to the rank of matrix A i.e. *r* , then rank k approximation is equal to the original matrix i.e we approximated the exact function A.

$$A_k = A.$$

Distance notion for matrix:

So now comes the notion of distance between the original matrix and the approximated matrix. Earlier the using distance notion we computed the distance between two vectors in a space. But now we have to define the notion of the distance for matrix.

We know that if *k = r(rank of matrix A)*,

$$A_k = A$$

This means that

$$\{ A_k - A\} = 0$$

So there should be a notion of distance between two matrix such that

$$Dist(A, A_k) = 0 \quad , if\ k = r$$

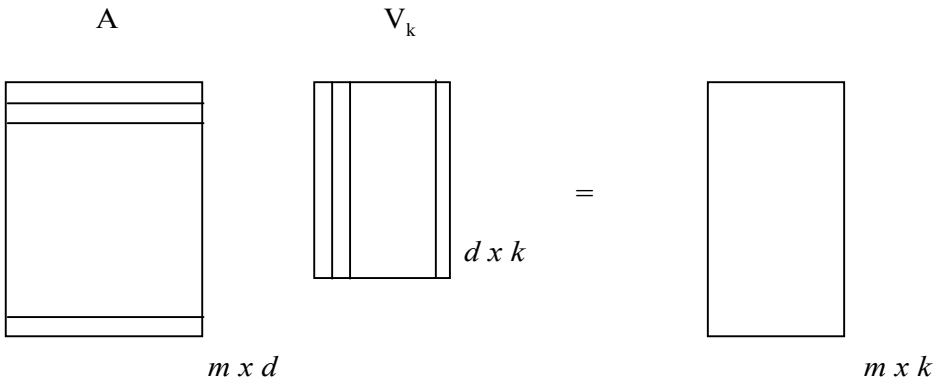
So now we have to use the Frobenius norm.

$$\| A \|_F = \sqrt{\sum_{j,k} a_{jk}^2}$$

Frobenius norm in terms of Rows

Let A be an $n \times d$ matrix with singular vectors $v1, v2, \ldots, v_r$. For $1 \leq k \leq r$, let V_k be the subspace spanned by $v1, v2, \ldots, v_k$. For each k, V_k is the best-fit k-dimensional subspace for A.

Lets see how the matrices look like :



Note that the *n*-vector Av_i is a list of lengths with signs of the projections of the rows of A onto v_i. Think of $|Av_i| = \sigma_i(A)$ as the "component" of the matrix A along v_i. For this interpretation to make sense, it should be true that adding up the squares of the components of A along each of the v_i gives the square of the "whole content of the matrix A". This is indeed the case and is the matrix analogy of decomposing a vector into its components along orthogonal directions.

Consider a row a_j, of A. Since $v_1, v_2 ... v_r$ span the space of all rows of A, $a_j.v = 0$ for all perpendicular to $v_1, v_2 ... v_r$.

Thus , for each row a_j,

$$\sum_{i}^{r} a_j.v_i = |a_j|^2 .$$

Summing over all rows j,

$$\sum_{j=1}^{n} |a_j|^2 = \sum_{j=1}^{n}\sum_{i=1}^{r} (a_j.v_i)^2 = \sum_{i=1}^{r}\sum_{j=1}^{n} (a_j.v_i)^2 = \sum_{i=1}^{r} |Av_i|^2 = \sum_{i=1}^{r} \sigma_i^2(A)$$

But

$$\sum_{j=1}^{n} |a_j|^2 = \sum_{j=1}^{n}\sum_{k=1}^{d} a_{jk}^2 ,$$

i.e. the sum of squares of all the entries of A. Thus, the sum of squares of the singular values of A is indeed the square of the "whole content of A", i.e., the sum of squares of all the entries.

Shannon Entropy:

In information theory, **entropy** (more specifically, **Shannon entropy**) is the expected value (average) of the information contained in each message received. 'Messages' don't have to be text; in this context a 'message' is simply any flow of information. The entropy of the message is its amount of uncertainty; it increases when the message is closer to random, and decreases when it is less random. The idea here is that the less likely an event is, the more information it provides when it occurs. This seems backwards at first: it seems like messages which have more structure would contain more information, but this is not true.

For example: Suppose there is a Machine 1 and 2, generating the sequence of A, B, C, D symbols with different probability of generation of each symbol in both machines i.e. AABDCAC and we ask : Which machine is producing more information?

To counter this situation **Claud Shannon** ask the same question in different way i.e. if you have to predict next symbol generating from both the machine, how many questions you would expect to ask?

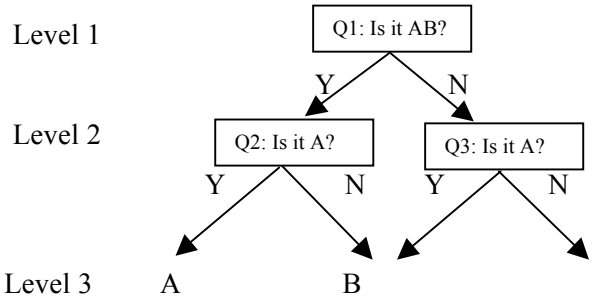
Also the event which have less probability of occurrence have more information and hence require more questions to gain certainty. Hence the information content is inversely proportional to probability of occurrence of that event.

Machine 1:

Probability of generation of symbol A or B or C or D = 1/4

All symbol are equally likely to occur.

P (symbol = A) = P (symbol = B) = P (symbol = C) = P (symbol = D)



After the Q1 we pick any one of A or B as both are equally likely and ask the Q2

because Q2 lead to increase in probability P(symbol = A) from 1/4 in level 2 to 1 in level 3 or decrease the amount of uncertainty or gain of information.

Expected No. of Questions per symbol = 2

We doesn't ask Q3 as one step down the Q3 have no change in uncertainty or no gain of information. P(symbol = A) which remains 0 from level 2 to level 3 in Q1-Q3 path.

Machine 2:

P (Letter = A) = 1/2

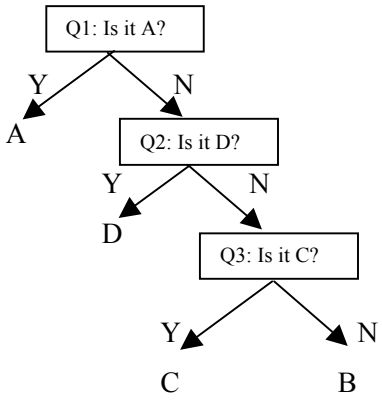
P (Letter = B) = 1/8

P (Letter = C) = 1/8

P (Letter = D) = 1/4

Here the probability of occurrence of each symbol is different so we have to ask our questions differently.

So we start by asking about the symbol which have less information or less uncertainty.



Expected no of question for each symbol in Machine 2 is equal to 1.75.

So what is expected number of questions signifies that if we have 100 symbols to identify from both machines then we require 200 questions for Machine 1 and 175 questions for Machine 2.

This means that machine 2 is producing the less information than because there is less uncertainty of occurrence of symbols. Claud Shannon called this measure of uncertainty as **Entropy represented by letter H.** The unit for entropy is based on the uncertainty of fair coin flip and called the **BIT** which is equivalent to fair bounce.

$$\mathbf{H} = \sum_{i=1}^{n} p(i) log(\frac{1}{p(i)})$$

Where :

log here is logarithm base 2,

p (i): probability of occurrence of symbol i.

So entropy is maximum when all outcomes are equally likely and if any predictability is introduced then entropy must go down or reduction in amount of information.

References:

1. https://en.wikipedia.org/wiki/Matrix_norm
2. https://en.wikipedia.org/wiki/Entropy_(information_theory)
3. https://www.khanacademy.org/computing/computer-science/informationtheory