# Naïve Bayes Analysis

- NB algorithm is easy to understand and works fast. It also performs well in multi class prediction, such as when the target class has multiple options beyond binary yes/ no classification. NB can perform well even in case of categorical input variables compared to numerical variable( s).
- We need to compute prior probability. Suppose the data gathered for the last one year showed that during that period there were 2500 customers for R, and 1500 customers for M. Thus the default (or prior) probability for the next customer to be for R is 2500/ 4000 or 5/ 8.
- Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

  In a classification problem, our hypothesis (h) may be the class to assign for a new data instance (d).

  One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge.

  Bayes' Theorem is stated as:

  P(h|d) = (P(d|h) * P(h)) / P(d)

  Where

  - **P(h|d)** is the probability of hypothesis h given the data d. This is called the posterior probability.
  - **P(d|h)** is the probability of data d given that the hypothesis h was true.
  - **P(h)** is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
  - **P(d)** is the probability of the data (regardless of the hypothesis).

  You can see that we are interested in calculating the posterior probability of P(h|d) from the prior probability p(h) with P(D) and P(d|h).

After calculating the posterior probability for a number of different hypotheses, you can select the hypothesis with the highest probability. This is the maximum probable hypothesis and may formally be called the maximum a posteriori (MAP) hypothesis.

This can be written as:

MAP(h) = max(P(h|d))

or

MAP(h) = max((P(d|h) * P(h)) / P(d))

or

MAP(h) = max(P(d|h) * P(h))

The P(d) is a normalizing term which allows us to calculate the probability. We can drop it when we are interested in the most probable hypothesis as it is constant and only used to normalize.

Back to classification, if we have an even number of instances in each class in our training data, then the probability of each class (e.g. P(h)) will be equal. Again, this would be a constant term in our equation and we could drop it so that we end up with:

MAP(h) = max(P(d|h))

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

This is a useful exercise, because when reading up further on Naive Bayes you may see all of these forms of the theorem.

# Naive Bayes Classifier

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values.

It is called *naive Bayes* or *idiot Bayes* because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value P(d1, d2, d3|h), they are assumed to be conditionally independent given the target value and calculated as P(d1|h) * P(d2|H) and so on.

This is a very strong assumption that is most unlikely in real data, i.e. that the attributes do not interact. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold.

## Representation Used By Naive Bayes Models

The representation for naive Bayes is probabilities.

A list of probabilities are stored to file for a learned naive Bayes model. This includes:

- **Class Probabilities**: The probabilities of each class in the training dataset.
- **Conditional Probabilities**: The conditional probabilities of each input value given each class value.

## Learn a Naive Bayes Model From Data

Learning a naive Bayes model from your training data is fast.

Training is fast because only the probability of each class and the probability of each class given different input (x) values need to be calculated. No coefficients need to be fitted by optimization procedures.

### Calculating Class Probabilities

The class probabilities are simply the frequency of instances that belong to each class divided by the total number of instances.

For example in a binary classification the probability of an instance belonging to class 1 would be calculated as:

P(class=1) = count(class=1) / (count(class=0) + count(class=1))

In the simplest case each class would have the probability of 0.5 or 50% for a binary classification problem with the same number of instances in each class.

### Calculating Conditional Probabilities

The conditional probabilities are the frequency of each attribute value for a given class value divided by the frequency of instances with that class value.

For example, if a "*weather*" attribute had the values "*sunny*" and "*rainy*" and the class attribute had the class values "*go-out*" and "*stay-home*", then the conditional probabilities of each weather value for each class value could be calculated as:

- P(weather=sunny|class=go-out) = count(instances with weather=sunny and class=go-out) / count(instances with class=go-out)
- P(weather=sunny|class=stay-home) = count(instances with weather=sunny and class=stay-home) / count(instances with class=stay-home)
- P(weather=rainy|class=go-out) = count(instances with weather=rainy and class=go-out) / count(instances with class=go-out)
- P(weather=rainy|class=stay-home) = count(instances with weather=rainy and class=stay-home) / count(instances with class=stay-home)

## Make Predictions With a Naive Bayes Model

Given a naive Bayes model, you can make predictions for new data using Bayes theorem.

MAP(h) = max(P(d|h) * P(h))

Using our example above, if we had a new instance with the *weather* of *sunny*, we can calculate:

go-out = P(weather=sunny|class=go-out) * P(class=go-out)
stay-home = P(weather=sunny|class=stay-home) * P(class=stay-home)

We can choose the class that has the largest calculated value. We can turn these values into probabilities by normalizing them as follows:

P(go-out|weather=sunny) = go-out / (go-out + stay-home)
P(stay-home|weather=sunny) = stay-home / (go-out + stay-home)

If we had more input variables we could extend the above example. For example, pretend we have a "*car*" attribute with the values "*working*" and "*broken*". We can multiply this probability into the equation.

For example below is the calculation for the "go-out" class label with the addition of the car input variable set to "working":

go-out = P(weather=sunny|class=go-out) * P(car=working|class=go-out) * P(class=go-out)

# Gaussian Naive Bayes

Naive Bayes can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution.

This extension of naive Bayes is called Gaussian Naive Bayes. Other functions can be used to estimate the distribution of the data, but the Gaussian (or Normal distribution) is the easiest to work with because you only need to estimate the mean and the standard deviation from your training data.

### Make Predictions With a Gaussian Naive Bayes Model

Probabilities of new x values are calculated using the Gaussian Probability Density Function (PDF).

When making predictions these parameters can be plugged into the Gaussian PDF with a new input for the variable, and in return the Gaussian PDF will provide an estimate of the probability of that new input value for that class.

pdf(x, mean, sd) = (1 / (sqrt(2 * PI) * sd)) * exp(-((x-mean^2)/(2*sd^2)))

Where pdf(x) is the Gaussian PDF, sqrt() is the square root, mean and sd are the mean and standard deviation calculated above, PI is the numerical constant, exp() is the numerical constant e or Euler's number raised to power and x is the input value for the input variable.

We can then plug in the probabilities into the equation above to make predictions with real-valued inputs.

For example, adapting one of the above calculations with numerical values for weather and car:

go-out = P(pdf(weather)|class=go-out) * P(pdf(car)|class=go-out) * P(class=go-out)

## Fraud Detection Using Naïve Bayes

Naïve Bayes is a mining technique not commonly associated with fraud detection in the scientific literature. In their review of the academic literature on data mining applications in financial fraud detection, Ngai et al [5] indicate very few studies that applied naïve Bayes algorithms for this purpose.
That trend contrasts with Viaene, Dering and Dedene [11] findings, who present a successful application of naïve Bayes algorithm to personal injury protection (PIP) claims for the State of Massachusetts. Their findings suggest that this algorithm can lead to efficient fraud detection systems for insurance claim evaluation support.
Viaene et al [12] also indicates that naïve Bayes algorithms showed comparative predictive performance to more complex and computationally demanding algorithms, such as Bayesian Learning Multilayer Perceptron, least-squares support vector machine and tree-augmented naïve Bayes classification, in a benchmark study of algorithm performance for insurance fraud detection.
The emphasis of research of complex unsupervised algorithms presents to Phua et al [7] a problem for the future. The authors suggest that in order for fraud detection to be successfully implemented in real time applications, less complex algorithms, such as naïve Bayes, have to be considered as the only viable options.

The probability model for a classifier is a conditional model : P(C ? H 1 ,...,H n ) over a dependent class variable C with a small number of outcomes or classes, conditional on several feature variables H 1 through H n .
Using the well-known Bayes' theorem we write:

$$P(C|H_1,\ldots,H_n) = \frac{P(H_1,\ldots,H_n|C)P(C)}{P(H_1,\ldots,H_n)}$$

Because the denominator does not depend on C we are usually interested only in the numerator of the right side fraction. The values of the features are also given and consequently the denominator is constant.
The numerator is equivalent to the joint probability model: P(C,H 1 ,...,H n ).
The problem is that if the number of features is large or when a feature can take on a large number of values, the computation of such a model can be infeasible.
The "naive" conditional independence assumption assumes that each feature is conditionally independent of every other feature: P(H i / C,H j )= P(H i / C).
This strong assumption can be unrealistic in most cases, but empirical studies related to fraud detection show that most frequently the method presents good performance [11]. Zhang [15] explained the superb classification performance of naïve Bayes in real applications, where the conditional independence is rarely true, showing that dependence among attributes usually cancel out each other.
Under these independence assumptions the conditional distribution over the class variable can be expressed as:

$$P(C|H_1,\ldots,H_n) = \frac{1}{Z}\,p(C)\prod_{i=1}^{n} p(H_i \mid C)$$

where Z is a constant if the values of the feature variables are known. P(C) is called the class prior and P(H i ? C) are the independent probability distributions
By using this naïve Bayesian algorithm it is then possible to obtain a probability distribution of objects belonging into classes. Threshold rules can be used to decide when a probability is strong enough to assign an object into a group. Depending on these rules it happens that an object is not assigned to any group or to more than one group, like in fuzzy logic. Naïve Bayes also naturally deals with missing values, what is difficult to achieve using other methods like decision trees or neural networks.

Resulting models are self explainable, unlike other methods like neural networks.
Further information about Bayesian Classifiers can be obtained at [7].

# How Naive Bayes algorithm works?

Let's understand it using an example. Below I have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let's follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

| Weather | Play |
|---|---|
| Sunny | No |
| Overcast | Yes |
| Rainy | Yes |
| Sunny | Yes |
| Sunny | Yes |
| Overcast | Yes |
| Rainy | No |
| Rainy | No |
| Sunny | Yes |
| Rainy | Yes |
| Sunny | No |
| Overcast | Yes |
| Overcast | Yes |
| Rainy | No |

| Frequency Table | | |
|---|---|---|
| Weather | No | Yes |
| Overcast | | 4 |
| Rainy | 3 | 2 |
| Sunny | 2 | 3 |
| Grand Total | 5 | 9 |

| Likelihood table | | | | |
|---|---|---|---|---|
| Weather | No | Yes | | |
| Overcast | | 4 | =4/14 | 0.29 |
| Rainy | 3 | 2 | =5/14 | 0.36 |
| Sunny | 2 | 3 | =5/14 | 0.36 |
| All | 5 | 9 | | |
| | =5/14 | =9/14 | | |
| | 0.36 | 0.64 | | |

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

**Problem:** Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

P(Yes | Sunny) = P( Sunny | Yes) * P(Yes) / P (Sunny)

Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P( Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

# Text Classification Example:

Sentiment Analysis also known as opinion mining is used to find out reviews about a product or topic. Sentiment analysis can be very useful in many ways like finding out the popularity, success of a product etc. Sentiment may vary from one person to the other. Hence, a system is required to collect the data from various web sources like blogs, forums, tweets, comments or posts.

### EXAMPLES OF TEXT CLASSIFICATION

 CLASSES=BINARY
 “spam” / “not spam”
 CLASSES =TOPICS
 “finance” / “sports” / “politics”
 CLASSES =OPINION
 “like” / “hate” / “neutral”
 CLASSES =TOPICS
 “AI” / “Theory” / “Graphics”
 CLASSES =AUTHOR
 “Shakespeare” / “Marlowe” / “Ben Jonson”

### NAÏVE BAYES APPROACH

 Build the Vocabulary as the list of all distinct words that appear in all the documents of the training set.
 Remove stop words and markings
 The words in the vocabulary become the attributes, assuming that classification is independent of the positions of the words
 Each document in the training set becomes a record with frequencies for each word in the Vocabulary.
 Train the classifier based on the training data set, by computing the prior probabilities for each class and attributes.
 Evaluate the results on Test data

# Advantages :

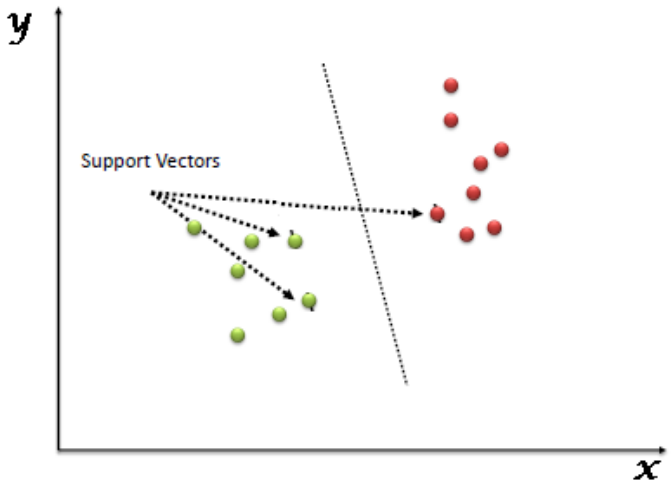
- Easy to implement
- Requires a small amount of training data to estimatethe parameters
- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

# Disadvantages:

- Assumption: class conditional independence, therefore loss of accuracy
- Practically, dependencies exist among variables
  E.g., hospitals: patients: Profile: age, family history, etc.
  Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
- Dependencies among these cannot be modelled by Naïve Bayesian Classifier
- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency". To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from `predict_proba` are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

# What is Support Vector Machine?

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).




Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

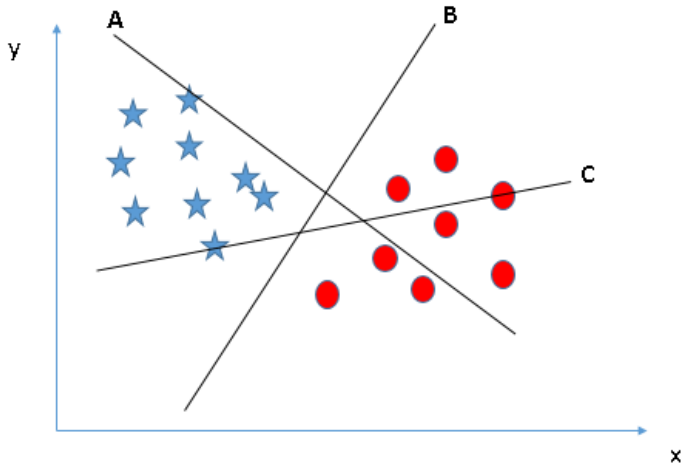
You can look at support vector machines and a few examples of its working here.

# How does it work?

Above, we got accustomed to the process of segregating the two classes with a hyper-plane. Now the burning question is "How can we identify the right hyper-plane?". Don't worry, it's not as hard as you think!

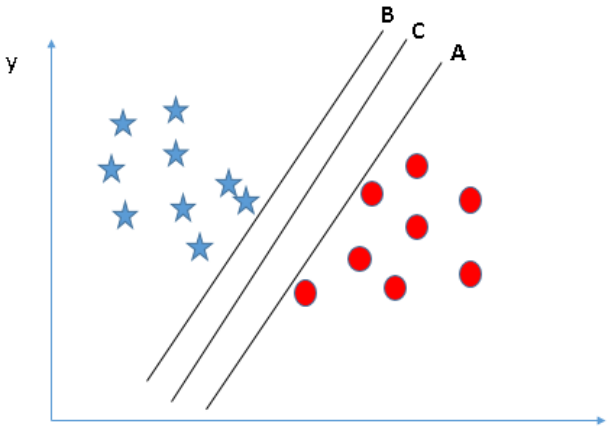
Let's understand:

- **Identify the right hyper-plane (Scenario-1):** Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.

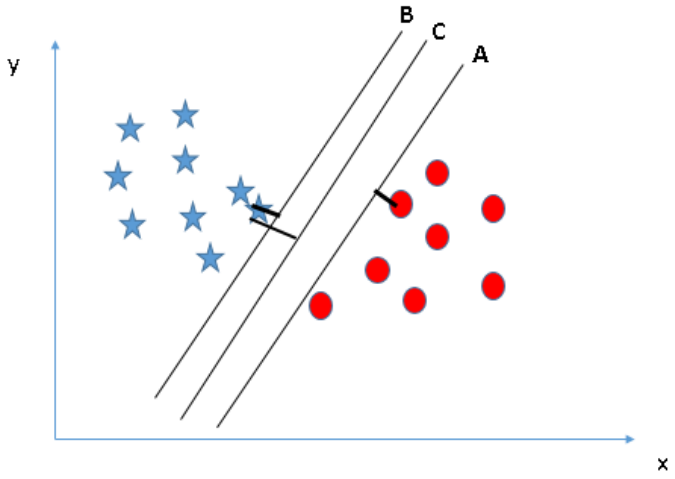



You need to remember a thumb rule to identify the right hyper-plane: "Select the hyper-plane which segregates the two classes better". In this scenario, hyper-plane "B" has excellently performed this job.

- **Identify the right hyper-plane (Scenario-2):** Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?

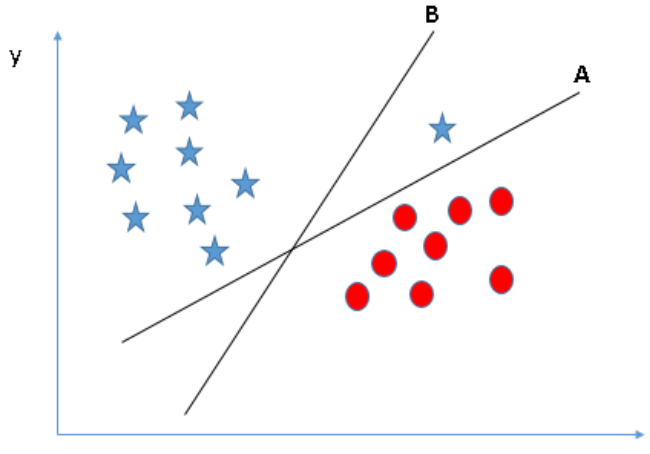



× Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**. Let's look at the below snapshot:



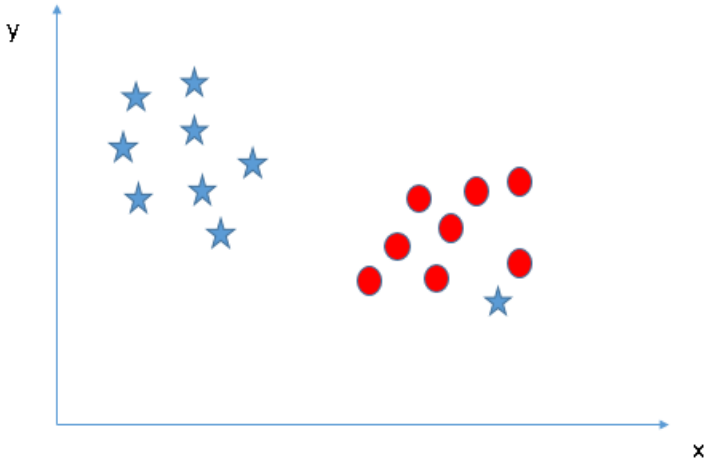

Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

- **Identify the right hyper-plane (Scenario-3):**Hint: Use the rules as discussed in previous section to identify the right hyper-plane



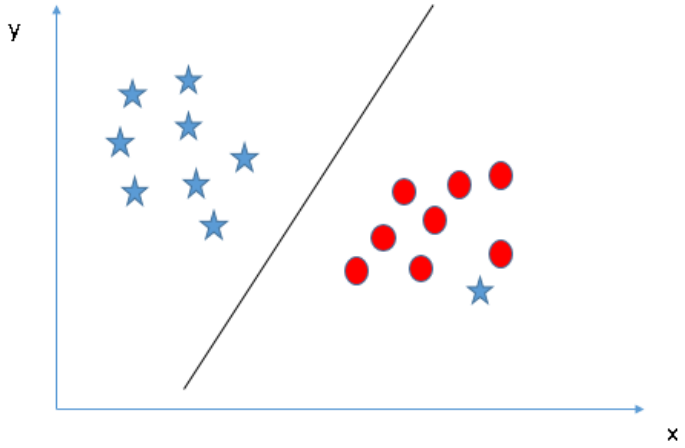

Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A.** But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

- **Can we classify two classes (Scenario-4)?:** Below, I am unable to segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.

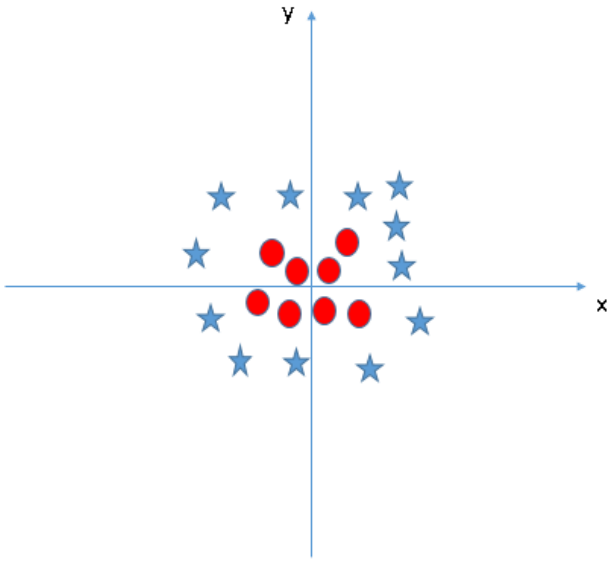



As I have already mentioned, one star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, we can

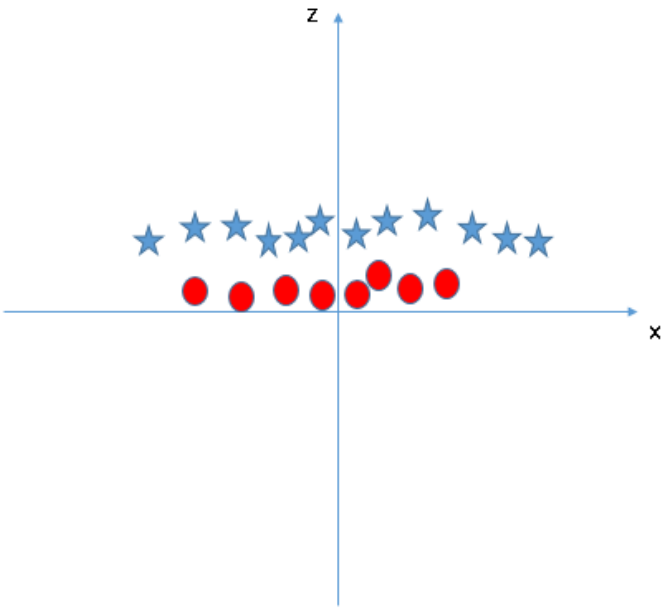
say, SVM is robust to outliers.




- **Find the hyper-plane to segregate to classes (Scenario-5):** In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.




SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:

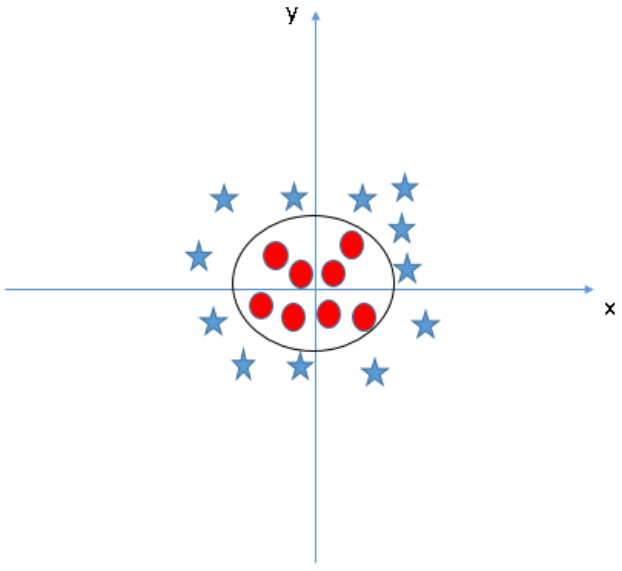



In above plot, points to consider are:

- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

In SVM, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the **kernel trick**. These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you've defined.

When we look at the hyper-plane in original input space it looks like a circle:



Now, let's look at the methods to apply SVM algorithm in a data science challenge.

# Pros and Cons associated with SVM

- **Pros:**
  - It works really well with clear margin of separation
  - It is effective in high dimensional spaces.
  - It is effective in cases where number of dimensions is greater than the number of samples.
  - It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- **Cons:**
  - It doesn't perform well, when we have large data set because the required training time is higher
  - It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping
  - SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library.

More formally, a support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection.[3] Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

Kernel Method

In machine learning, **kernel methods** are a class of algorithms for pattern analysis, whose best known member is the support vector machine (SVM). The general task of pattern analysis is to find and study general types of relations (for example clusters, rankings, principal components, correlations, classifications) in datasets. In its simplest form, the kernel trick means transforming data into another dimension that has a clear dividing margin between classes of data.[1] For many algorithms that solve these tasks, the data in raw representation have to be explicitly transformed into feature vector representations via a user-specified *feature map*: in contrast, kernel methods require only a user-specified *kernel*, i.e., a similarity function over pairs of data points in raw representation.

Kernel methods owe their name to the use of kernel functions, which enable them to operate in a high-dimensional, *implicit* feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the

explicit computation of the coordinates. This approach is called the "**kernel trick**"[2]. Kernel functions have been introduced for sequence data, graphs, text, images, as well as vectors.

Some kernels:

• Linear kernel k(x 1 ,x 2 ) = x T

1 x 2

• Polynomial kernel k(x 1 ,x 2 ) = (1 + x T

1 x 2 )

d

• Contains all polynomial terms up to degree d

• Gaussian kernel k(x 1 ,x 2 ) = exp(−||x 1 − x 2 || 2 /2σ 2 )

• Infinite dimension feature space

## SOCIAL NETWORK ANALYSIS

Social network analysis [SNA] is the mapping and measuring of relationships and flows between people, groups, organizations, computers, URLs, and other connected information/knowledge entities. The nodes in the network are the people and groups while the links show relationships or flows between the nodes. SNA provides both a visual and a mathematical analysis of human relationships.

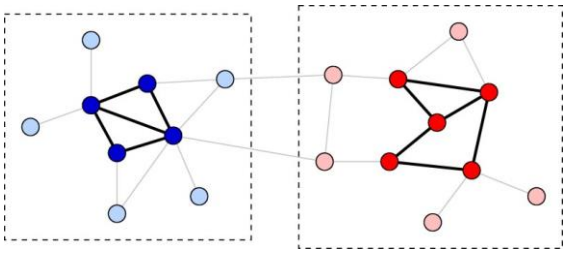
Social network analysis (SNA) is the process of investigating social structures through the use of networks and graph theory.

- There are two major levels of social network analysis: discovering sub- networks within the network, and ranking the nodes to find more important nodes or hubs.
- Computing the relative influence of each node is done on the basis of an input- output matrix of flows of influence among the nodes.
- Entities that many other entities point to are called Authorities.
- Hubs are entities that point to a relatively large number of authorities.

# Discovering Sub Networks

Subnetworks can be identified using

1) Agglomerative Clustering

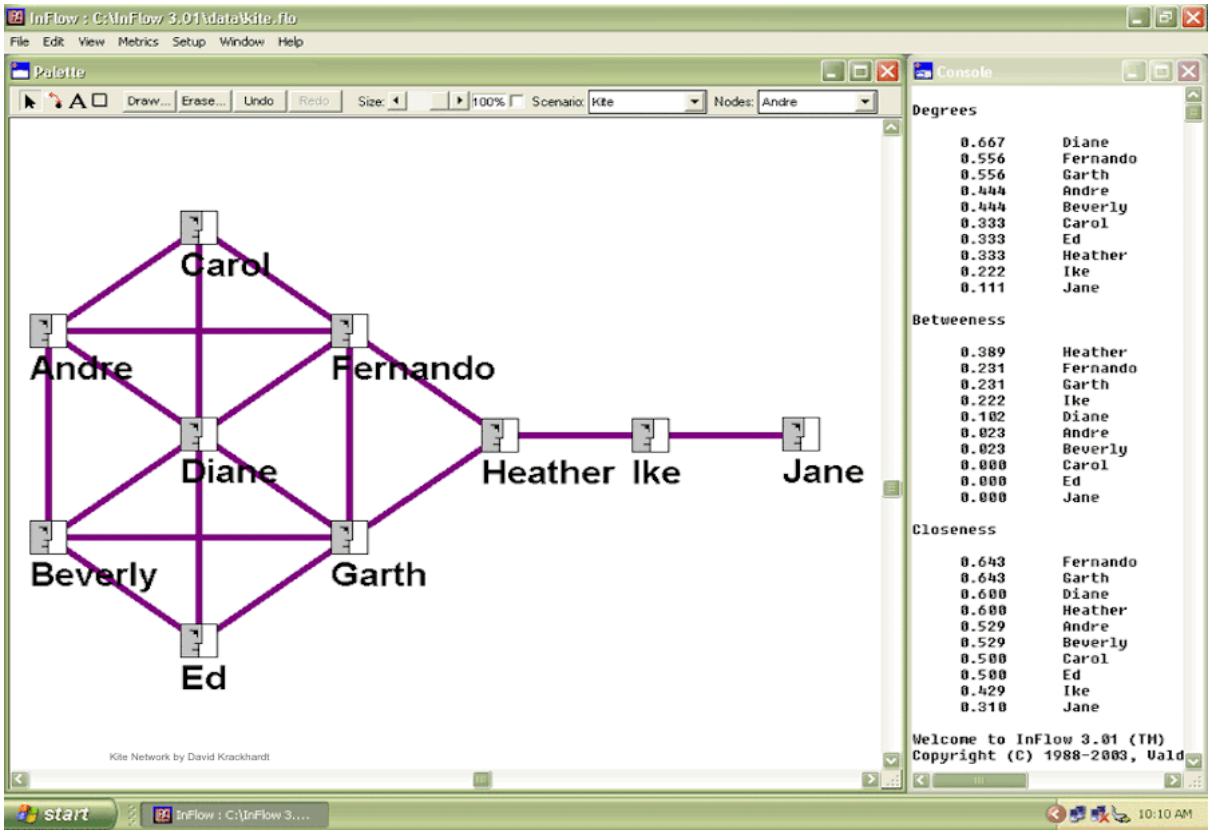


2) Cliques
   The idea of a clique is relatively simple. At the most general level, a clique is a sub-set of a

network in which the actors are more closely and intensely tied to one another than they are to other members of the network. In terms of friendship ties, for example, it is not unusual for people in human groups to form "cliques" on the basis of age, gender, race, ethnicity, religion/ideology, and many other things. The smallest "cliques" are composed of two actors: the dyad.

3) N-Cliques
4) N-Clans
5) K-plexes - to allow that actors may be members of a clique even if they have tiles to all but k other members.
6) K-cores -A k-core is a maximal group of actors, all of whom are connected to some number (k) of other members of the group.
7) Lambda Sets and Bridges- An alternative approach is to ask if there are certain connections in the graph which, if removed, would result in a disconnected structure.
8) Factions-In network terms, actors are said to be equivalent to the extent that they have the same profiles of ties to other actors. It follows that we might define partitions of the network on the basis ofgrouping together actors on the basis of similarity in who they are tied to.
9) Blocks and Cutpoints -One approach to finding these key spots in the diagram is to ask if a node were removed, would the structure become divided into un-connected systems? If there are such nodes, they are called "cutpoints."

# Identifying Importance among nodes




## Degree Centrality

Social network researchers measure network activity for a node by using the concept of degrees -- the number of direct connections a node has. In the kite network above, Diane has the most direct connections in the network, making hers the most active node in the network. She is a 'connector' or 'hub' in this network. Common wisdom in personal networks is "the more connections, the better." This is not always so. What really matters is where those connections lead to -- and how they connect the otherwise unconnected! Here Diane has connections only to others in her immediate cluster -- her clique. She connects only those who are already connected to each other.

## Closeness Centrality

Fernando and Garth have fewer connections than Diane, yet the pattern of their direct and indirect ties allow them to access all the nodes in the network more quickly than anyone else. They have the shortest paths to all others -- they are close to everyone else. They are in an excellent position to monitor the information flow in the network -- they have the best visibility into what is happening in the network.

## Network Centralization

Individual network centralities provide insight into the individual's location in the network. The relationship between the centralities of all nodes can reveal much about the overall network structure.

A very centralized network is dominated by one or a few very central nodes. If these nodes are removed or damaged, the network quickly fragments into unconnected sub-networks. A highly central node can become a single point of failure. A network centralized around a well connected hub can fail abruptly if that hub is disabled or removed. Hubs are nodes with high degree and betweeness centrality.

A less centralized network has no single points of failure. It is resilient in the face of many intentional attacks or random failures -- many nodes or links can fail while allowing the remaining nodes to still reach each other over other network paths. Networks of low centralization fail gracefully.

## Network Reach

Not all network paths are created equal. More and more research shows that the shorter paths in the network are more important. Noah Friedkin, Ron Burt and other researchers have shown that networks have horizons over which we cannot see, nor influence. They propose that the key paths in networks are 1 and 2 steps and on rare occasions, three steps. The "small world" in which we live is not one of "six degrees of separation" but of direct and indirect connections < 3 steps away. Therefore, it is important to know: who is in your network neighborhood? Who are you aware of, and who can you reach?

In the network above, who is the only person that can reach everyone else in two steps or less?

## Network Integration

Network metrics are often measured using *geodesics* -- or shortest paths. They make the (erroneous) assumption that all information/influence flows along the network's shortest paths only. But networks operate via direct and indirect, shortest and near-shortest paths.

We often hear interesting things from various sources in the network. *Different interpretations arrive via different paths*. Therefore, it is important to be on many efficient paths in networks that reach out to various parts of the extended network. Those well integrated in the network of paths have both local and distant information, along with several flavors of it!

## Boundary Spanners

Nodes that connect their group to others usually end up with high network metrics. Boundary spanners such as Fernando, Garth, and Heather are more central in the overall network than their immediate neighbors whose connections are only local, within their immediate cluster. You can be a boundary spanner via your bridging connections to other clusters or via your concurrent membership in overlappping groups.

Boundary spanners are well-positioned to be innovators, since they have access to ideas and information flowing in other clusters. They are in a position to combine different ideas and knowledge, found in various places, into new products and services.

## Peripheral Players

Most people would view the nodes on the periphery of a network as *not* being very important. In fact, Ike and Jane receive very low centrality scores for *this* network. Since individuals' networks overlap, peripheral nodes are connected to networks that are not currently mapped. Ike and Jane may be contractors or vendors that have their own network outside of the company -- making them very important resources for fresh information not available inside the company!

## Client Feedback

When sharing network maps and metrics with clients I explain to them that the maps and metrics are *mirrors*, not *report cards*! The consultant and the client *together* make sense of what the maps/metrics reflect about the organization. The consultant brings external expertise and context, while the client provides internal context about the organizations and its goals. Both are necessary to analyze the networks properly!

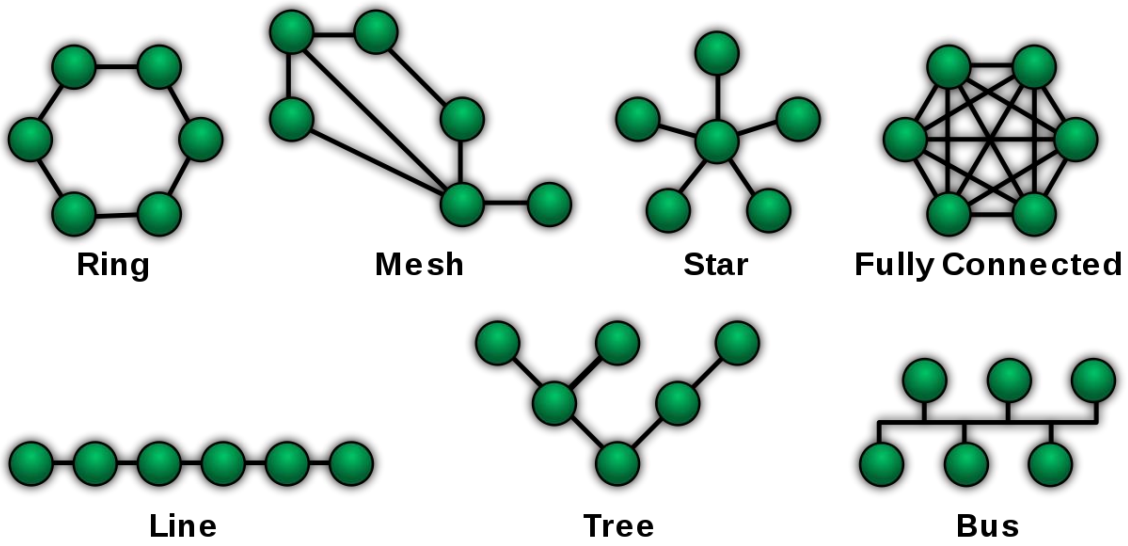
Before sharing the actual network visualizations, I introduce the client to what s/he will see via the Network Discovery Matrix.

| | |
|---|---|
| Positive Expected | Positive Unexpected |
| Negative Expected | Negative Unexpected |



This 2x2 matrix reveals all of the possibilities that the network maps will reveal. The client will see both good news and bad news about the network, along with expected patterns ad unexpected patterns. This matrix helps the client' s managers organize, and make sense of, what they will be seeing.

# Network Topologies




# Applications of SNA

Social network analysis is used extensively in a wide range of applications and disciplines. Some common network analysis applications include data aggregation and mining, network

propagation modeling, network modeling and sampling, user attribute and behavior analysis, community-maintained resource support, location-based interaction analysis, social sharing and filtering, recommender systems development, and link prediction and entity resolution.[47] In the private sector, businesses use social network analysis to support activities such as customer interaction and analysis, information system development analysis,[48] marketing, and business intelligence needs (see social media analytics). Some public sector uses include development of leader engagement strategies, analysis of individual and group engagement and media use, and community-based problem solving.

## Security applications

Social network analysis is also used in intelligence, counter-intelligence and law enforcement activities.

## Textual analysis applications

Large textual corpora can be turned into networks and then analysed with the method of social network analysis. In these networks, the nodes are Social Actors, and the links are Actions. The extraction of these networks can be automated by using parsers.

## Internet applications

Social network analysis has also been applied to understanding online behavior by individuals, organizations, and between websites. Hyperlink analysis can be used to analyze the connections between websites or webpages to examine how information flows as individuals navigate the web. Social network analysis has been applied to social media as a tool to understand behavior between individuals or organizations through their linkages on social media websites such as Twitter and Facebook.[58]

### Key terms

There are several key terms associated with social network analysis research in computer-supported collaborative learning such as: **density**, **centrality**, **indegree**, **outdegree**, and **sociogram**.

- **Density** refers to the "connections" between participants. Density is defined as the number of connections a participant has, divided by the total possible connections a participant could have. For example, if there are 20 people participating, each person could potentially connect to 19 other people. A density of 100% (19/19) is the greatest density in the system. A density of 5% indicates there is only 1 of 19 possible connections.[59]
- **Centrality** focuses on the behavior of individual participants within a network. It measures the extent to which an individual interacts with other individuals in the network. The more an individual connects to others in a network, the greater their centrality in the network.[59]

In-degree and out-degree variables are related to centrality.

- **In-degree** centrality concentrates on a specific individual as the point of focus; centrality of all other individuals is based on their relation to the focal point of the "in-degree" individual.[59]

- **Out-degree** is a measure of centrality that still focuses on a single individual, but the analytic is concerned with the out-going interactions of the individual; the measure of out-degree centrality is how many times the focus point individual interacts with others.[59]
- A **sociogram** is a visualization with defined boundaries of connections in the network. For example, a sociogram which shows out-degree centrality points for Participant A would illustrate all outgoing connections Participant A made in the studied network.[59]

**Unique capabilities**

# Techniques and Algorithms

Algorithms

1) Cliques and Cores
   One of the earliest sociometric concepts is that of a clique (Luce and Perry, 1949). It corresponds to a subgraph of a simple, undirected graph in which every vertex is adjacent to every other vertex. A clique is the idealized version of a cohesive group.

2) Triad Census and Clustering Coefficient
   The triad census is a vector containing the number of occurrences of nonisomorphic subgraphs on exactly three vertices. For an undirected graph, these correspond to four graphs shown in Figure 1. Note that, by combination of different configurations in nonnull dyads, these are refined to 16 such subgraphs in the directed case. The triad census was proposed as a means to assess local structure (Holland and Leinhardt, 1970, 1976) and has been in wide use since.

3) Role Equivalence (Regular Equivalence)
   In contrast to Section Cliques and Cores, we now consider groups of actors defined by common structural roles rather than cohesion. See Borgatti and Everett (1992) and Lerner (2005) for background and an overview.

4) Shortest-Path Centralities
   Actor centrality indices in social networks have been classified along two dimensions (Borgatti, 2005): a diffusion mechanism (shortest paths, random walks, etc.) by which something (information, disease, etc.) spreads in a social network, and the nature of positions along the resulting pathways (beginning, center, etc.).

# Page Rank

PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages. According to Google:

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

It is not the only algorithm used by Google to order search engine results, but it is the first algorithm that was used by the company, and it is the best-known.

The above centrality measure is not implemented for multi-graphs.

***Algorithm***

The PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is evenly divided among all documents in the collection at the beginning of the computational process. The PageRank computations require several passes, called "iterations", through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

**Simplified algorithm**

Assume a small universe of four web pages: A, B, C and D. Links from a page to itself, or multiple outbound links from one single page to another single page, are ignored. PageRank is initialized to the same value for all pages. In the original form of PageRank, the sum of PageRank over all pages was the total number of pages on the web at that time, so each page in this example would have an initial value of 1. However, later versions of PageRank, and the remainder of this section, assume a probability distribution between 0 and 1. Hence the initial value for each page in this example is 0.25.

The PageRank transferred from a given page to the targets of its outbound links upon the next iteration is divided equally among all outbound links.

If the only links in the system were from pages B, C, and D to A, each link would transfer 0.25 PageRank to A upon the next iteration, for a total of 0.75.

$$PR(A) = PR(B) + PR(C) + PR(D).$$

Suppose instead that page B had a link to pages C and A, page C had a link to page A, and page D had links to all three pages. Thus, upon the first iteration, page B would transfer half of its existing value, or 0.125, to page A and the other half, or 0.125, to page C. Page C would transfer all of its existing value, 0.25, to the only page it links to, A. Since D had three outbound links, it would transfer one third of its existing value, or approximately 0.083, to A. At the completion of this iteration, page A will have a PageRank of approximately 0.458.

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3}.$$

In other words, the PageRank conferred by an outbound link is equal to the document's own PageRank score divided by the number of outbound links L( ).

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}.$$ In the general case, the PageRank value for any page u can be expressed as:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)},$$

i.e. the PageRank value for a page u is dependent on the PageRank values for each page v contained in the set Bu (the set containing all pages linking to page u), divided by the number L(v) of links from page v. The algorithm involves a damping factor for the calculation of the pagerank. It is like the income tax which the govt extracts from one despite paying him itself.

Following is the code for the calculation of the Page rank