EXPERIMENT 8

**Aim**: Implementation of Page Rank Algorithm

**Theory**:

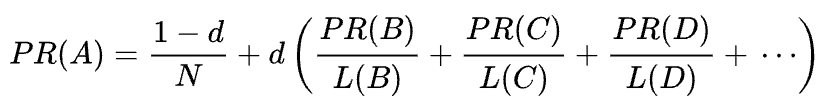
PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages. According to Google:

*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.*

**Working**:

Each link from one page (A) to another (B) casts a so-called vote, the weight of which depends on the collective weight of all the pages that link to page A. And we can't know their weight till we calculate it, so the process goes in cycles.

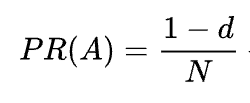
The mathematical formula of the original PageRank is the following:



Where A, B, C, and D are some pages, L is the number of links going out from each of them, and N is the total number of pages in the collection (i.e. on the Internet).

As for d, d is the so-called damping factor. Considering that PageRank is calculated simulating the behavior of a user who randomly gets to a page and clicks links, we apply this damping d factor as the probability of the user getting bored and leaving a page.

As you can see from the formula, if there are no pages pointing to the page, its PR will be not zero

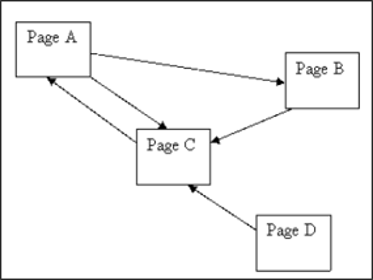


As there’s a probability that the user could get to this page not from some other pages but, say, from bookmarks.

**Code**:

| import numpy as np  def page\_rank\_algorithm(graph,damping\_factor):  outgoing = dict()  incoming\_nodes = dict()  coefficients = dict()  # Outgoing Nodes  for i in range(len(graph)):  outgoing[i]=0   for i,node in enumerate(graph):  for edge in node:  if edge:  outgoing[i] += 1   # Incoming Nodes  for i in range(len(graph)):  temp=[]  for node in graph:  if node[i]:  temp.append(node)  incoming\_nodes[i] = temp   # Coefficient Matrix  for i,node in enumerate(graph):  temp = []  for j,other\_node in enumerate(graph):  if other\_node in incoming\_nodes[i]:  temp.append(damping\_factor\*(1.0/outgoing[j]))  elif i == j:  temp.append(-1)  else:  temp.append(0)  coefficients[i] = temp   coefficients\_list = []  for key,value in coefficients.items():  coefficients\_list.append(value)   constant\_matrix = []  for i in range(len(graph)):  constant\_matrix.append(damping\_factor-1)   pageranks = np.linalg.solve(np.array(coefficients\_list),np.array(constant\_matrix))    print()  for i,rank in enumerate(pageranks):  print('Page Rank of {} is {:.4f}'.format(chr(65+i), rank))  def main():  n = int(input('Enter the number of nodes : '))   d= float(input('Enter the damping factor : '))    graph = []  print('Enter Adjacency Matrix with terms separated by a space : ')  for i in range(n):  temp\_list = input().split(' ')  graph.append(list(map(int,temp\_list)))    page\_rank\_algorithm(graph,d)  main() |
| --- |

**Graph**:



**Output**:

| Enter the number of nodes : 4 Enter the damping factor : 0.85 Enter Adjacency Matrix with terms separated by a space :  0 1 1 0 0 0 1 0 1 0 0 0 0 0 1 0  Page Rank of A is 1.4901 Page Rank of B is 0.7833 Page Rank of C is 1.5766 Page Rank of D is 0.1500 |
| --- |

**Conclusion:**

Page Rank algorithm is one of the first algorithms in the history of Google search engine and is used to rank web pagesIt is a Web Structure Mining algorithm. Page Rank calculated is based on the incoming links (Backlinks). A dampening factor is used to avoid the rank sink problem. It's only drawback is that it favours old pages rather than the new ones but is still a widely used algorithm because of its efficiency.

EXPERIMENT 9

**Aim:** Implementation of HITS Algorithm

**Theory:**

Hyperlink Induced Topic Search (HITS) Algorithm is a Link Analysis Algorithm that rates webpages, developed by Jon Kleinberg. This algorithm is used to the web link-structures to discover and rank the webpages relevant for a particular search.

HITS uses hubs and authorities to define a recursive relationship between webpages. Before understanding the HITS Algorithm, we first need to know about Hubs and Authorities.

Given a query to a Search Engine, the set of highly relevant web pages are called Roots. They are potential Authorities.

Pages that are not very relevant but point to pages in the Root are called Hubs. Thus, an Authority is a page that many hubs link to whereas a Hub is a page that links to many authorities.

**Working:**

In the HITS algorithm, the first step is to retrieve the most relevant pages to the search query. This set is called the root set and can be obtained by taking the top pages returned by a text-based search algorithm. A base set is generated by augmenting the root set with all the web pages that are linked from it and some of the pages that link to it. The web pages in the base set and all hyperlinks among those pages form a focused subgraph. The HITS computation is performed only on this focused subgraph. According to Kleinberg the reason for constructing a base set is to ensure that most (or many) of the strongest authorities are included.

Authority and hub values are defined in terms of one another in a mutual recursion. An authority value is computed as the sum of the scaled hub values that point to that page. A hub value is the sum of the scaled authority values of the pages it points to. Some implementations also consider the relevance of the linked pages.

The algorithm performs a series of iterations, each consisting of two basic steps:

* **Authority update**: Update each node's authority score to be equal to the sum of the hub scores of each node that points to it. That is, a node is given a high authority score by being linked from pages that are recognized as Hubs for information.
* **Hub update**: Update each node's hub score to be equal to the sum of the authority scores of each node that it points to. That is, a node is given a high hub score by linking to nodes that are considered to be authorities on the subject.

The Hub score and Authority score for a node is calculated with the following algorithm:

* Start with each node having a hub score and authority score of 1.
* Run the authority update rule
* Run the hub update rule
* Normalize the values by dividing each Hub score by the square root of the sum of the squares of all Hub scores, and dividing each Authority score by the square root of the sum of the squares of all Authority scores.
* Repeat from the second step as necessary.

**Comparison:**

HITS, like Page and Brin's PageRank, is an iterative algorithm based on the linkage of the documents on the web. However it does have some major differences:

* It is query dependent, that is, the (Hubs and Authority) scores resulting from the link analysis are influenced by the search terms;
* As a corollary, it is executed at query time, not at indexing time, with the associated hit on performance that accompanies query-time processing.
* It is not commonly used by search engines. (Though a similar algorithm was said to be used by Teoma, which was acquired by Ask Jeeves/Ask.com.)
* It computes two scores per document, hub and authority, as opposed to a single score;
* It is processed on a small subset of ‘relevant’ documents (a 'focused subgraph' or base set), not all documents as was the case with PageRank.

**Advantages:**

1. HITS scores due to its ability to rank pages according to the query string, resulting in relevant authority and hub pages.

2. HITS is sensitive to user queries (as compared to PageRank).

3. Important pages are obtained on the basis of calculated authority and hub value.

**Disadvantages:**

1. Since HITS is a query dependent algorithm the query time evaluation is expensive.

2. The rating or scores of authorities and hubs could rise due to flaws done by the web page designer. HITS assumes that when a user creates a web page he links a hyperlink from his page to another authority page, as he honestly believes that the authority page is in some way related to his page (hub).

3. Topic drift occurs when there are irrelevant pages in the root set and they are strongly connected. Since the root set itself contains non-relevant pages, this will reflect on the pages in the base set.

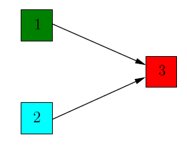
**Algorithm:**

| *G := set of pages for each page p in G do  p.auth = 1 // p.auth is the authority score of the page p  p.hub = 1 // p.hub is the hub score of the page p for step from 1 to k do // run the algorithm for k steps  norm = 0  for each page p in G do // update all authority values first  p.auth = 0  for each page q in p.incomingNeighbors do // p.incomingNeighbors is the set of pages that link to p  p.auth += q.hub  norm += square(p.auth) // calculate the sum of the squared auth values to normalise  norm = sqrt(norm)  for each page p in G do // update the auth scores   p.auth = p.auth / norm // normalise the auth values  norm = 0  for each page p in G do // then update all hub values  p.hub = 0  for each page r in p.outgoingNeighbors do // p.outgoingNeighbors is the set of pages that p links to  p.hub += r.auth  norm += square(p.hub) // calculate the sum of the squared hub values to normalise  norm = sqrt(norm)  for each page p in G do // then update all hub values  p.hub = p.hub / norm // normalise the hub values* |
| --- |

**Code:**

| from math import sqrt  def hits\_algorithm(num\_nodes, graph, iterations):  authority\_scores = dict()  hub\_scores = dict()  for i in range(len(graph)):  authority\_scores[i] = 1  hub\_scores[i] = 1  incoming\_nodes = dict()  for i in range(len(graph)):  temp=[]  for node in graph:  if node[i]:  temp.append(node)  incoming\_nodes[i] = temp  outgoing\_nodes = dict()  for i,node in enumerate(graph):  temp = []  for j,edge in enumerate(node):  if edge:  temp.append(graph[j])  outgoing\_nodes[i] = temp  print()  for k in range(iterations):  print('Iteration : ',k+1)  print('Authority Score')  normalization\_value = 0  for i,node in enumerate(graph):  authority\_scores[i]=0  for j,other\_node in enumerate(graph):  if other\_node in incoming\_nodes[i]:  authority\_scores[i] += hub\_scores[j]  normalization\_value += (authority\_scores[i]\*\*2)  normalization\_value = sqrt(normalization\_value)  for i in range(num\_nodes):  authority\_scores[i] /= normalization\_value  print('{} :{:.2f}'.format(chr(65+i),authority\_scores[i]),end=' | ')  print()  print('Hub Score')  normalization\_value = 0  for i,node in enumerate(graph):  hub\_scores[i]=0  for j,other\_node in enumerate(graph):  if other\_node in outgoing\_nodes[i]:  hub\_scores[i] += authority\_scores[j]  normalization\_value += (hub\_scores[i]\*\*2)  normalization\_value = sqrt(normalization\_value)  for i in range(num\_nodes):  hub\_scores[i] /= normalization\_value  print('{} :{:.2f}'.format(chr(65+i),hub\_scores[i]),end=' | ')  print("\n\n")  def main():  n = int(input('Enter the no of nodes : '))  graph = []  print('Enter Adjacency Matrix : ')  for i in range(n):  temp = input()  temp\_list = temp.split(' ')  graph.append(list(map(int,temp\_list)))  k = int(input('Enter No of Iterations to be performed : '))  hits\_algorithm(n, graph, k)  main() |
| --- |

**Graph:**



**Output:**

| Enter the no of nodes : 3 Enter Adjacency Matrix :  0 0 1 0 0 1 0 0 0 Enter No of Iterations to be performed : 3  Iteration : 1 Authority Score A :0.00 | B :0.00 | C :1.00 | Hub Score A :0.71 | B :0.71 | C :0.00 |   Iteration : 2 Authority Score A :0.00 | B :0.00 | C :1.00 | Hub Score A :0.71 | B :0.71 | C :0.00 |   Iteration : 3 Authority Score A :0.00 | B :0.00 | C :1.00 | Hub Score A :0.71 | B :0.71 | C :0.00 | |
| --- |

**Conclusion:**

Hyperlink-Induced Topic Search (HITS) is a link analysis algorithm that rates Web pages. HITS, like Page and Brin's PageRank, is an iterative algorithm based on the linkage of the documents on the web. However its disadvantages outweigh its advantages and thus it is not commonly used in search engines as compared to the PageRank algorithm which proves to be more efficient on large datasets.

EXPERIMENT 10

**AIM:** Write and Explain one algorithm each on

1.Spatial Association Rules

2.Spatial Classification

3.Spatial Clustering - DBScan

**Theory:**

Spatial data means data related to space which can be the two-dimensional abstraction of the surface of the earth or a man-made space like the layout of a VLSI design, a volume containing a model of the human brain, or another 3d-space representing the arrangement of chains of protein molecules. The data consists of geometric information and can be either discrete or continuous. The explicit location and extension of spatial objects define implicit relations of spatial neighborhood (such as topological, distance and direction relations) which are used by spatial data mining algorithms. Therefore, spatial data mining algorithms are required for spatial characterization and spatial trend analysis.

Spatial data mining or knowledge discovery in spatial databases differs from regular data mining in analogous with the differences between non-spatial data and spatial data. The attributes of a spatial object stored in a database may be affected by the attributes of the spatial neighbors of that object. In addition, spatial location, and implicit information about the location of an object, may be exactly the information that can be extracted through spatial data mining

**1.Spatial Association Rules**

Spatial association means connectedness or relationship between and among variables over space. A single variable may be spatially autocorrelated; that is, values of the variable are somehow connected or related spatially. Many variables may be associated one with another at one or more sites. If there is spatial interaction there is also spatial association. Maps can depict spatial association. A mathematical shorthand technique can be used to represent, in general, measures of spatial association. Scientists test or theorize about variables to determine whether spatial association, either observed or expected, actually can be confirmed. Statistical procedures that have been developed for identifying and measuring the existence of spatial association are outlined.

**Algorithm : Apriori Algorithm**

The Apriori algorithm uses frequent itemsets to generate association rules, and it is designed to work on the databases that contain transactions. With the help of these association rules, it determines how strongly or how weakly two objects are connected. This algorithm uses a breadth-first search and Hash Tree to calculate the itemset associations efficiently. It is the iterative process for finding the frequent itemsets from the large dataset.

This algorithm is mainly used for market basket analysis and helps to find those products that can be bought together. It can also be used in the healthcare field to Frequent itemsets which are those items whose support is greater than the threshold value or user-specified minimum support. It means if A & B are the frequent itemsets together, then individually A and B should also be the frequent itemset.

**Steps for Apriori Algorithm**

Below are the steps for the apriori algorithm:

Step-1: Determine the support of itemsets in the transactional database, and select the minimum support and confidence.

Step-2: Take all supports in the transaction with higher support value than the minimum or selected support value.

Step-3: Find all the rules of these subsets that have higher confidence value than the threshold or minimum confidence.

Step-4: Sort the rules as the decreasing order of lift.

**Advantages of Apriori Algorithm**

* This is easy to understand algorithm
* The join and prune steps of the algorithm can be easily implemented on large datasets.

**Disadvantages of Apriori Algorithm**

* The apriori algorithm works slow compared to other algorithms.
* The overall performance can be reduced as it scans the database for multiple times.
* The time complexity and space complexity of the apriori algorithm is O(2D), which is very high. Here D represents the horizontal width present in the database.

**2.Spatial Classification**

Spatial classification assigns an object to a class from a given set of classes based on the attribute values of the object. It mainly considers the distance, direction, or connectivity relationships among spatial objects.

**Algorithm: KNN Algorithm**

* K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
* The K-NN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories.
* K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suited category by using K- NN algorithm.
* The K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
* K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
* It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
* The KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
* Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know whether it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

**Working**:

The K-NN working can be explained on the basis of the below algorithm:

* Step-1: Select the number K of the neighbors
* Step-2: Calculate the Euclidean distance of K number of neighbors
* Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.
* Step-4: Among these k neighbors, count the number of the data points in each category.
* Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.
* Step-6: Our model is ready.

**Selecting the value of K in the K-NN Algorithm**

Below are some points to remember while selecting the value of K in the K-NN algorithm:

* There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
* A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
* Large values for K are good, but it may find some difficulties.

**Advantages of KNN Algorithm**:

* It is simple to implement.
* It is robust to the noisy training data
* It can be more effective if the training data is large.

**Disadvantages of KNN Algorithm:**

* Always needs to determine the value of K which may be complex some time.
* The computation cost is high because of calculating the distance between the data points for all the training samples.

**3.Spatial Clustering - DBScan**

Spatial Clustering Clustering is a descriptive task that seeks to identify homogeneous groups of objects based on the values of their attributes. In spatial data sets, clustering permits a generalization of the spatial component like explicit location and extension of spatial objects which define implicit relations of spatial neighborhood. Current spatial clustering techniques can be broadly classified into three categories;

* partitional
* hierarchical
* locality-based.

**Algorithm: DBSCAN Algorithm**

Density-Based Clustering refers to unsupervised learning methods that identify distinctive groups/clusters in the data, based on the idea that a cluster in data space is a contiguous region of high point density, separated from other such clusters by contiguous regions of low point density.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a base algorithm for density-based clustering. It can discover clusters of different shapes and sizes from a large amount of data, which is containing noise and outliers.

The DBSCAN algorithm uses two parameters:

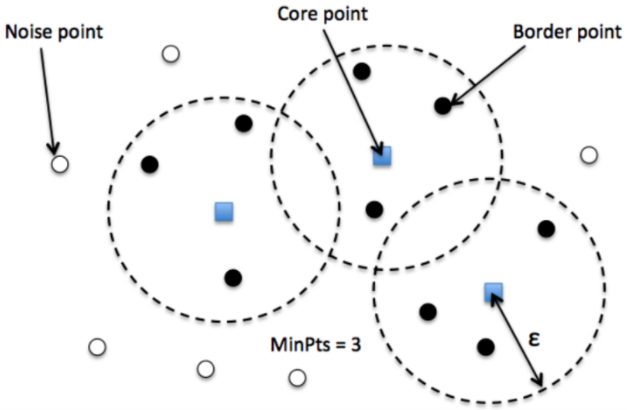
* minPts: The minimum number of points (a threshold) clustered together for a region to be considered dense.
* eps (ε): A distance measure that will be used to locate the points in the neighborhood of any point.

These parameters can be understood if we explore two concepts called Density Reachability and Density Connectivity.

Reachability in terms of density establishes a point to be reachable from another if it lies within a particular distance (eps) from it.

Connectivity, on the other hand, involves a transitivity based chaining-approach to determine whether points are located in a particular cluster. For example, p and q points could be connected if p->r->s->t->q, where a->b means b is in the neighborhood of a.

There are three types of points after the DBSCAN clustering is complete:



* **Core** — This is a point that has at least m points within distance n from itself.
* **Border** — This is a point that has at least one Core point at a distance n.
* **Noise** — This is a point that is neither a Core nor a Border. And it has less than m points within distance n from itself.

**Steps for DBSCAN clustering**

* The algorithm proceeds by arbitrarily picking up a point in the dataset (until all points have been visited).
* If there are at least ‘minPoint’ points within a radius of ‘ε’ to the point then we consider all these points to be part of the same cluster.
* The clusters are then expanded by recursively repeating the neighborhood calculation for each neighboring point

**Parameter Estimation**

Every data mining task has the problem of parameters. Every parameter influences the algorithm in specific ways. For DBSCAN, the parameters ε and minPts are needed.

* **minPts**: As a rule of thumb, a minimum minPts can be derived from the number of dimensions D in the data set, as minPts ≥ D + 1. The low value minPts = 1 does not make sense, as then every point on its own will already be a cluster. With minPts ≤ 2, the result will be the same as of hierarchical clustering with the single link metric, with the dendrogram cut at height ε. Therefore, minPts must be chosen at least 3. However, larger values are usually better for data sets with noise and will yield more significant clusters. As a rule of thumb, minPts = 2·dim can be used, but it may be necessary to choose larger values for very large data, for noisy data or for data that contains many duplicates.
* **ε**: The value for ε can then be chosen by using a k-distance graph, plotting the distance to the k = minPts-1 nearest neighbor ordered from the largest to the smallest value. Good values of ε are where this plot shows an “elbow”: if ε is chosen much too small, a large part of the data will not be clustered; whereas for a too high value of ε, clusters will merge and the majority of objects will be in the same cluster. In general, small values of ε are preferable, and as a rule of thumb, only a small fraction of points should be within this distance of each other.
* **Distance function**: The choice of distance function is tightly linked to the choice of ε, and has a major impact on the outcomes. In general, it will be necessary to first identify a reasonable measure of similarity for the data set, before the parameter ε can be chosen. There is no estimation for this parameter, but the distance functions need to be chosen appropriately for the data set.

**CONCLUSION:**We learnt about spatial data, spatial association rules, classification and clustering. We also studied an algorithm of each type and learnt about the working, advantages and the disadvantages of each one of them.