Dynamic Data Structures

data decryption and clustering

Loic Niragire

TIM – 8110 Programming Languages and Algorithms

Dr. Phillip David

Table of Contents

1. Introduction

This paper explores clustering and decryption algorithms and their associated dynamic data structures in distributed environments. We also look at various deployment options for efficient performance. Additionally, we discuss potential issues that may arise from each concerned algorithm and provide insights into managing them effectively. Lastly, we present code snippets where deemed necessary.

2. Clustering Algorithms

In simplest terms, clustering problems are concerned with detecting groups, or clusters, within a dataset based on some shared properties (O'Leary, 2009). Classification is another term often used for these problems. However, one key distinction between the two is that clustering is a form of unsupervised learning whereas classification, such as regression, is a supervised learning (Esposito & Esposito, 2020). In practice, however, these two forms of learning are used together. For instance, an e-commerce site may first run a clustering algorithm to find a reasonable number of categories within its customer dataset, then apply a classification algorithm to label a given customer. Or in other words, assign membership to that customer. Similarly, given search parameters, a web crawler may classify a web page, or document, as relevant or irrelevant by performing random walks within a cluster. The clustering algorithms of our focus in this paper address these problems at a large scale. Like most basic algorithms, complications typically grow exponentially; the larger the input size increases and the more distributed they are – problems at the Facebook, Google, or Twitter scale, just to name a few.

Generally, parallel clustering algorithms apply a divide-and-conquer strategy to generate smaller sub-problems, then merge the results to obtain a global solution (Zhang, Wang, & Li, 2015). In a distributed environment, these generated sub-problems are solved on different processors. For the most part, these algorithms differ in how they handle the “divide” and the “conquer” phase. We leverage various data segmentation methods for the “divide” aspect, followed by a merging of parallel results.

Interests in IoT data clustering algorithms have recently gained momentum as portable devices continue to collect more data. These algorithms face unique challenges in part due to network bandwidth and storage capacity. Two main approaches exist for clustering IoT data; centralized or distributed (Brandão, Machado, Goldschmidt, & Choren, 2020). While the centralized approach is simple, it does not scale as the amount of data grows. It also demands a high network bandwidth as it transmits data to a centralized node for clustering. Meanwhile, the distributed approach requires computational capacity as it performs partial mining on the device and sends the outcome to a central node for aggregation. A recently proposed clustering algorithm, “gCluster,” with a linear complexity worst case, aims to reduce network traffic significantly by transmitting summarized data from IoT devices while avoiding computational overhead in the devices (Brandão, Machado, Goldschmidt, & Choren, 2020).

2.1. Machine Learning

Clustering inherently relies on unsupervised, i.e., learning by discovery, machine learning, where we are interested in discovering groups within a given dataset. Typically, this is the first step to understand a dataset in machine learning. As such, clustering models are not deployed in production but instead used as a step to make sense of the available data. Each identified group within a given dataset is referred to as a cluster and assigned a unique identifier – cluster Id. Subsequent analysis involving large feature datasets, therefore, can reference associated cluster Ids instead. Thus, clustering can serve as a data compression technique to simplify large datasets processing by dropping as many rows as possible without altering the knowledge in the dataset (Google Developers, 2021).

Clustering algorithms can be classified into three categories as illustrated in figure 1 below, partition-based, density-based, and hierarchy-based. Table 1 describes commonly used clustering algorithms within these categories.

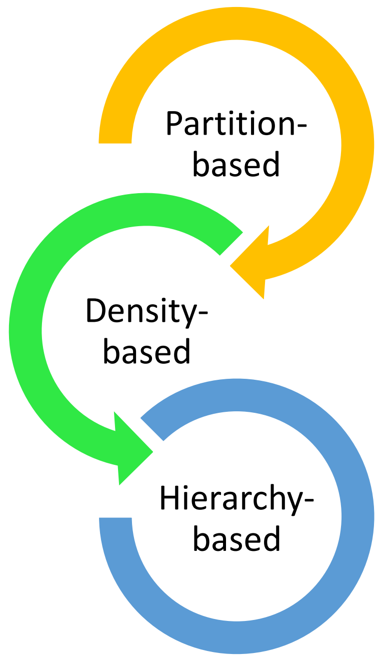


Figure 1 Clustering algorithms categories

Algorithm	*Category*	*Description*
K-Means	Partition-based	Sets a fixed number of clusters and randomly defines their data center.
Mean-Shift	Partition-based	Defines a circular sliding window and shifts the center point of the window to the mean of the points within the radius.
DBSCAN	Density-based	Includes all points within a given data range in a new cluster.
Agglomerative Hierarchical Clustering	Hierarchy-based	Creates a tree of clusters. They are suited for hierarchical data.

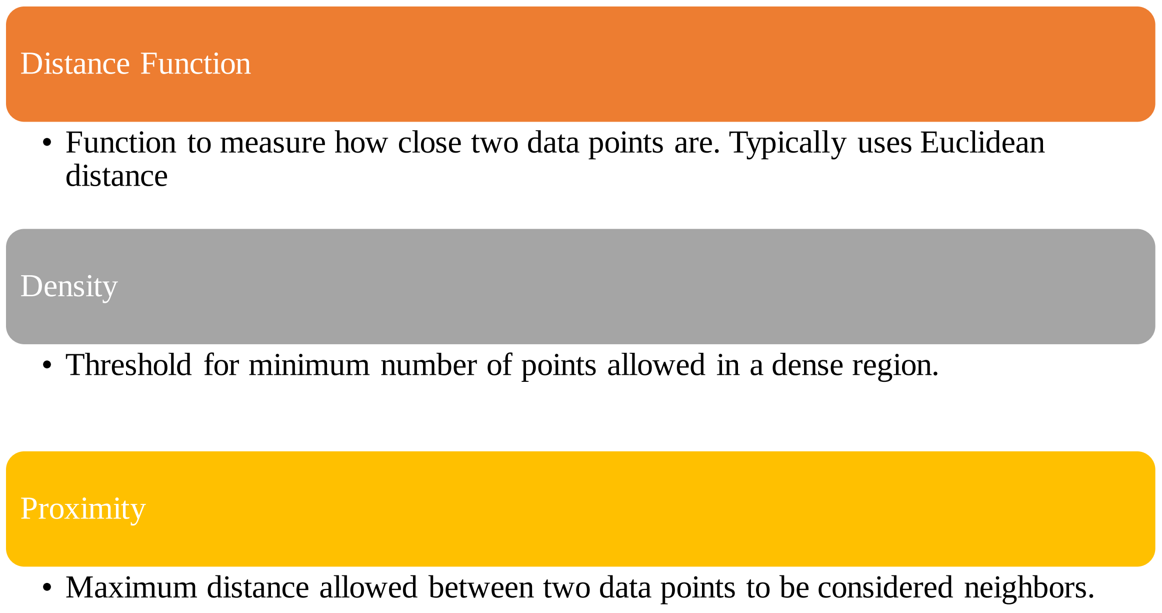
Table 1 Common clustering algorithms

2.1.1. K-Means Algorithm

K-Means algorithm builds *K* clusters in which the variance is as low as possible by placing all data points around a centroid element in each cluster and using the square of the Euclidean distance for measuring proximity (Esposito & Esposito, 2020). Then it iteratively update the initial centroid element until a stopping condition is met. New clusters are created when data points do not fit within the existing centroid.

2.1.2. DBSCAN Algorithm

The acronym DBSCAN is short for “*density-based spatial clustering of applications with noise*.” This algorithm was first presented about 15 years after the K-Means algorithm and aimed at algorithmically determining the number of clusters in a dataset. It works by defining the parameters illustrated in figure 2 below. The algorithm attempts to build a new cluster for each unvisited point and tries to expand on it by including neighborhood points.



Distance Function

- Function to measure how close two data points are. Typically uses Euclidean distance

Density

- Threshold for minimum number of points allowed in a dense region.

Proximity

- Maximum distance allowed between two data points to be considered neighbors.

Figure 2 DBSCAN parameters

Table 2 gives a comparison between the two algorithms discussed above.

ALGORITHM	PROS	CONS
K-MEANS	- Simple and fast	- Sets fixed number of groups - It only works for numeric features. - No effective strategy for choosing K value
DBSCAN	- Does not require a pre-determined number of clusters – K - Detecting outliers	- Poor performance on large datasets or datasets with large differences in densities.

Table 2 Comparison between K-Means and DBSCAN

3. Decryption Algorithms

Cryptography plays an ever-increasing role in the modern digital age by facilitating secure communication over networks through encryption. Various encryption processes or algorithms exist today to transform a given input into an unrecognized form. Decryption is the process used to reverse encrypted content back into its original condition. Thanks to public-key cryptography, users who never communicated before can now do so securely. Asymmetric cryptography algorithms use a shared public key generated by the recipient to encrypt the data before sending it over the network. Upon receipt, the recipient applies their private key, which is never to be shared, to decrypt received data.

Symmetric cryptographic algorithms use the same key for encryption and decryption of data. Some of the popular symmetric cryptographic algorithms in usage since the 1970s include Data Encryption Standard (DES), Triple DES (3DES), International Data Encryption Algorithm (IDEA) and Rivest Cipher 5 (RC5). Although all these are considered weak and broken today, they are faster and easier to validate than asymmetric ciphers. This is mostly due to their lack of authentication abilities and key exchange scaling problems (Grimes, 2020).

3.1. Polynomial Workload Effort

Mathematical integer factorization problems lie at the core of public-key cryptography. The secret is in using the product of two very large prime numbers as the public-key. Suppose A and B are our chosen very large prime numbers, then we would use $C = A x B$ as our public key. Theoretically, given that A and B are sufficiently large, it is tough to solve for them knowing only C. Thus, the workload required to factor large prime equations is what protects associated keys (Grimes, 2020).

3.2. RSA Cryptosystem

A cryptosystem, or cipher system, is defined as an implementation of cryptographic techniques and their associated infrastructure to provide information security services (Tutorialspoint, 2021). Both symmetric key encryption and asymmetric key encryption are considered as types of cryptosystems. Asymmetric keys, public and private, are different but mathematically related. Figure 3 illustrates components of a cryptosystem.

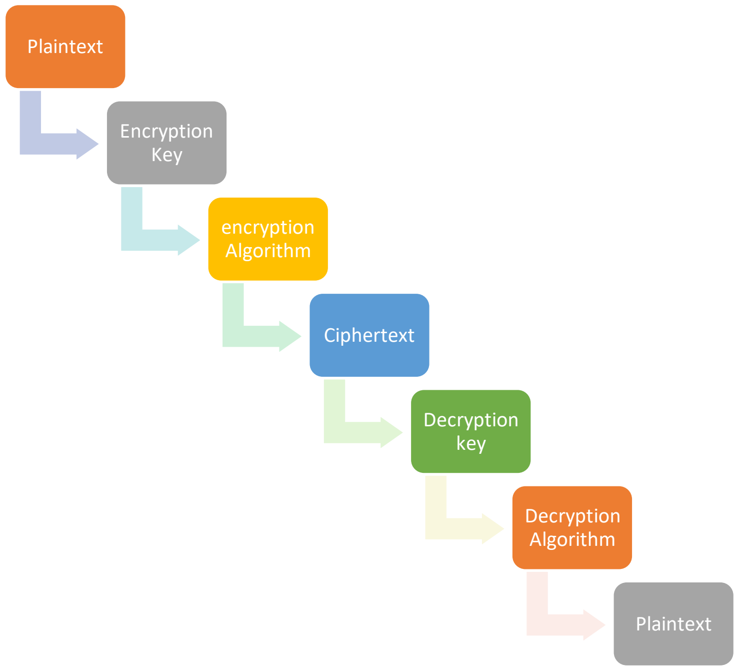


Figure 3 Cryptosystem components

RSA is one the most cryptosystem in usage today. The acronym reflects the three scholars involved in its invention: *Ron Rivest*, *Adi Shamir*, and *Len Adleman*. Figure 4 below illustrates the process of generating encryption and decryption keys in RSA. After private and public keys have been generated, the algorithm relies on modulo arithmetic to encrypt and decrypt the content. Table 3 below gives details of the encryption and decryption process.

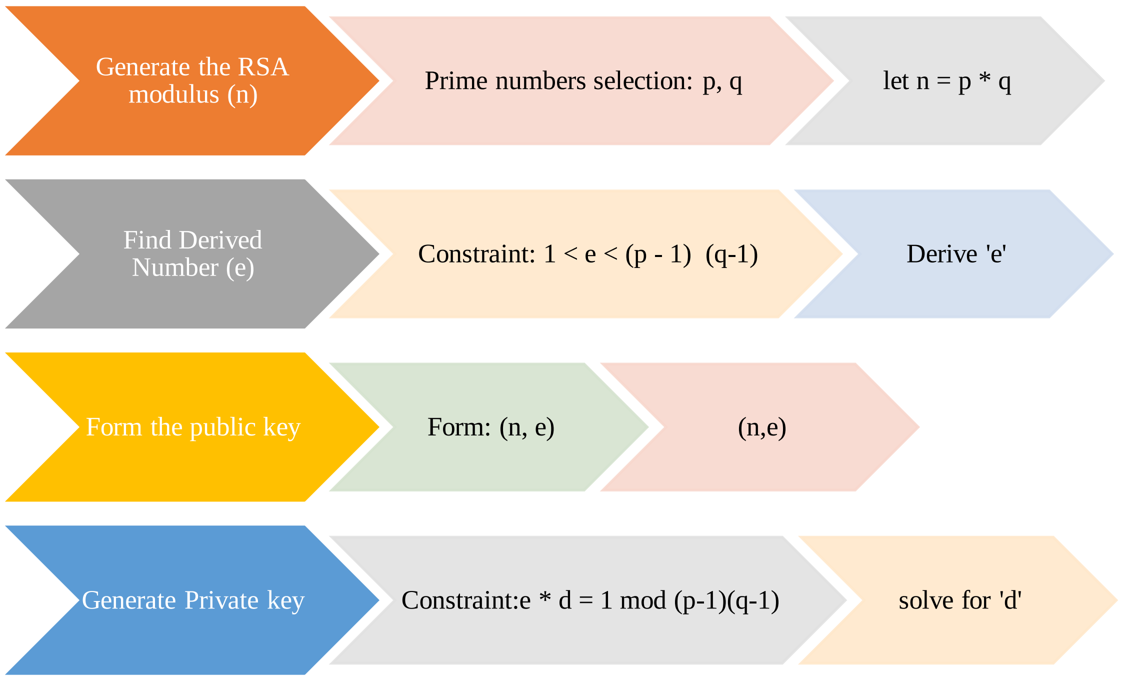

Generate the RSA modulus (n)
Prime numbers selection: p, q
let n = p * q
Find Derived Number (e)
Constraint: 1 < e < (p - 1) (q-1)
Derive 'e'
Form the public key
Form: (n, e)
(n,e)
Generate Private key
Constraint:e * d = 1 mod (p-1)(q-1)
solve for 'd'

Figure 4 RSA Keys generation

	Algorithm Steps
RSA Encryption	- Public key (n, e) - Represent text as a series of numbers less than n - Encrypted text = C = P^e mod n
RSA Decryption	- Encrypted text = C - Decrypted text = C^d mod n - Where d = private key

Table 3 RSA Encryption and Decryption process

Reference List

Brandão, R. d., Machado, G. R., Goldschmidt, R. R., & Choren, R. (2020). A Reduced Network Traffic Method for IoT Data Clustering. ACM Transactions on Knowledge Discovery from Data, Vol 15 (pp. 13-36). ACM.

Du, K.-L. (2010). Clustering: A neural network approach. Neural Networks , 89-107.

Esposito, D., & Esposito, F. (2020). Introducting Machine Learning. Pearson Education.

Google Developers. (2021, March). Clustering in Machine Learing. Retrieved from developers.google.com: https://developers.google.com/machine-learning/clustering/overview

Grimes, R. A. (2020). Cryptography Apocalypse: Preparing for the Day When Quantum Computing Breaks Today's Crypto. Hoboken: John Wiley & Sons.

O'Leary, D. P. (2009). Scientific Computing With Case Studies. Philadelphia: SIAM.

Tutorialspoint. (2021, March 15). Cryptosystems. Retrieved from tutorialspoint.com: https://www.tutorialspoint.com/cryptography/cryptosystems.htm

Zhang, Y., Wang, J. Z., & Li, J. (2015). Parallel Massive Clustering of Discrete Distributions. ACM Transactions on Multimedia Computing, Communications, and Applications (pp. 49-73). Philadelphia: ACM.