Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών

Αλγόριθμοι για τα προβλήματα k-means και k-median

Διπλωματική Εργασία

του

Κάβουρα Λουκά

Επιβλέπων: Δημήτρης Φωτάκης
Επίκουρος Καθηγητής Ε.Μ.Π.

Συνεπιβλέποντες: Ευστάθιος Ζάχος
Καθηγητής Ε.Μ.Π.

Αντώνιος Συμβώνης
Καθηγητής Ε.Μ.Π.

Εργαστήριο Λογικής και Επιστήμης Υπολογισμών

Αθήνα, Οκτώβριος 2014

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών

Αλγόριθμοι για τα προβλήματα k-means και k-median

Διπλωματική Εργασία

του

Κάβουρα Λουκά

Επιβλέπων: Δημήτρης Φωτάκης
Επίκουρος Καθηγητής Ε.Μ.Π.

Συνεπιβλέποντες: Ευστάθιος Ζάχος
Καθηγητής Ε.Μ.Π.

Αντώνιος Συμβώνης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 6η Οκτωβρίου 2014.

.....	.....	.....
Ευστάθιος Ζάχος	Αντώνιος Συμβώνης Παγουρτζής	Δημήτρης Φωτάκης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Εργαστήριο Λογικής και Επιστήμης Υπολογισμών
Αθήνα, Ιούλιος 2012

.....
Λουκάς Κάβουρας
Διπλωματούχος Ε.Μ.Π.

Περίληψη

Συσταδοποίηση ονομάζουμε την διαδικασία ομαδοποίησης συνόλου αντικειμένων με τρόπο ώστε αντικείμενα στην ίδια συστάδα να μοιάζουν περισσότερο μεταξύ τους από αντικέιμενα σε άλλες συστάδες. Είναι κύρια διαδικασία της εξόρυξης δεδομένων, της μηχανικής μάθησης και της υπολογιστικής γεωμετρίας. Σε αυτή τη διπλωματική εργασία, εξετάζουμε γνωστά προβλήματα συσταδοποίησης με έμφαση στο πρόβλημα k-means, όπου ο στόχος είναι να διαμερίσουμε n αντικείμενα σε k συστάδες έτσι ώστε να ελαχιστοποιηθεί το άθροισμα τετραγώνων των αποστάσεων των σημείων μέσα στη συστάδα. Παρουσιάζουμε τον αλγόριθμο του Lloyd για το πρόβλημα, ο οποίος έχει αναγνωριστεί ως ένας από τους κορυφαίους δέκα αλγορίθμους στην εξόρυξη δεδομένων. Παρά το ότι ο αλγόριθμος του Lloyd κάνει εκθετικά βήματα στην χειρότερη περίπτωση, συνήθως τρέχει γρήγορα σε πρακτικές εφαρμογές. Το μεγάλο μειονέκτημα του αλγόριθμου είναι ότι υπάρχουν στιγμιότυπα στα οποία βρίσκει αυθαίρετα κακές λύσεις.Ο k-means++ αλγόριθμος αντιμετωπίζει αυτό το πρόβλημα αρχικοποιώντας κατάλληλα τον αλγόριθμο του Lloyd. Ο k-means++ αποδεικνύεται ότι είναι $O(\log k)$-προσεγγιστικός. Στη συνέχεια, εξετάζουμε τον k-means|| αλγόριθμο, ο οποίος είναι εμπνευσμένος από τον k-means++ και μπορεί να παραλληλοποιηθεί αποδοτικά. Στο τελευταίο κεφάλαιο, εξετάζουμε περιπτώσεις, στις οποίες δεν γνωρίζουμε όλη την είσοδο του προβλήματος από την αρχή. Ειδικότερα, εξετάζουμε αλγόριθμους για το k-means πρόβλημα στο streaming μοντέλο, στο οποίο τα δεδομένα είναι πολλά για να αποθηκευτούν στη μνήμη και πρέπει να τα επεξεργαστούμε σειριακά. Στο τέλος, παρουσιάζουμε το facility location πρόβλημα και ειδικότερα τον αλγόριθμο του Meyerson για το online facility location πρόβλημα.

Abstract

Clustering is the task of grouping a set of objects in such a way that objects in the same cluster are more similar to each other than to those in other clusters. It is a main task of data mining, machine learning and computational geometry. In this thesis, we discuss famous clustering problems and we emphasize on the k-means clustering problem, where one seeks to partition n observations into k clusters so as to minimize the within-cluster sum of squares. We present Lloyd's algorithm for the k-means problem, which was identified as one of the top 10 algorithms in data mining. Although Lloyd's algorithm has an exponential running time in the worst case, it usually runs fast in many practical applications. However, the algorithm gives no guarantees and there are natural examples where it may produce arbitrarily bad clusterings. k-means++ algorithm addresses this problem by augmenting Lloyd's algorithm with a simple and intuitive seeding technique. A formal proof shows that k-means++ algorithm is $O(\log k)$ competitive. We also examine the k-means|| algorithm, which is an algorithm inspired by k-means++ algorithm that can be effectively parallelized. In the last chapter, we consider cases where the entire input is not available from the beginning. That is, we study algorithms for k-means in the streaming model, where the data is too large to be stored in main memory and must be accessed sequentially. Finally, we study the facility location problem and discuss the online facility location algorithm of Meyerson.

Keywords

clustering, k-means, k-median, approximation algorithm, online algorithm , streaming algorithm

Ευχαριστίες

Ολοκληρώνοντας την διπλωματική μου, θα ήθελα να ευχαριστήσω την τριμελή επιτροπή που αποτελείται από τους κ.Ζάχο, καθηγητή Ε.Μ.Π., κ.Συμβώνη, καθηγητή Ε.Μ.Π και κ.Φωτάκη, επίκουρο καθηγητή Ε.Μ.Π. για τη στήριξη, που μου παρείχαν, σε ακαδημαϊκό και προσωπικό επίπεδο.

Θα ήθελα ιδιαίτερα να σταθώ στον επιβλέποντα της διπλωματικής μου, κ.Φωτάκη, ο οποίος μου έδωσε την ευκαιρία να ασχοληθώ με το πεδίο των προσεγγιστικών αλγορίθμων. Η συνεργασία μου με τον κ.Φωτάκη με ενέπνευσε και με ενθάρρυνε να συνεχίσω με περισσότερο ζήλο την προσπάθεια μου. Τελος, θα ήθελα να ευχαριστήσω όλα τα μέλη του Εργαστηρίου Λογικής και Επιστήμης Υπολογιστών για την πολύτιμη βοήθεια τους σε ότι χρειάστηκα και για το ευχάριστο κλίμα που έχουν δημιουργήσει στο εργαστήριο.

Contents

1 Introduction **11**

2 Introduction to clustering **15**
2.1 Clustering and cluster models 15
2.2 Requirements 17
2.3 Applications 18

3 Preliminaries **23**
3.1 Problem formulation and variations 23
3.2 Definitions and useful results 27

4 k-means problem offline **31**
4.1 Hardness of the k-means problem 31
4.2 The k-means algorithm 34
4.2.1 Running time 35
4.2.2 Solution quality 36
4.3 The k-means++ algorithm 37
4.4 A parallel implementation 42

5 k-means problem online and streaming **46**
5.1 Streaming algorithms 46
5.1.1 Introduction 46
5.1.2 Streaming k-means on Well-Clusterable Data 47
5.1.3 Fast and Accurate k-means for Large Datasets 55
5.2 The facility location problem 57
5.2.1 Relation with k-median and k-means 57
5.2.2 Online Facility Location 58

Chapter 1

Introduction

Clustering or cluster analysis is the process of grouping a set of objects in such a way that objects in the same group are more similar (in some sense or another) to each other than to those in other groups. The greater the similarity within a group and the greater the difference between groups, the better or more distinct the clustering. Clustering is a central problem in data management with many applications and has a rich history with hundreds of different algorithms published on the subject.

Cluster analysis divides data into groups that are meaningful, useful, or both. In the context of understanding cluster analysis is the study of techniques for automatically finding potential classes, which are the clusters to be formed. This concept plays an important role in how people analyze and describe the world and has practical applications varying from biology and climate to psychology and medicine.

When utility is the goal, cluster analysis is the study of techniques for finding the most representative cluster prototypes. That is, these techniques characterize the cluster in terms of a cluster prototype, i.e. a data object that represents other objects in the cluster. Utility applications are image segmentation, software evolution, recommender systems and many others that we will discuss in the next chapter.

The process of cluster analysis is not one specific algorithm, but a very general task that can be solved by various algorithms that differ in their notion of what constitutes a cluster and in the manner that these clustering algorithms find clusters. By this, we mean that the notion of cluster is not well defined in many applications and that the definition of a cluster depends on the nature of the data and the desired results. So, instead of giving a precise definition of a cluster, we divide the clustering process into "cluster models" where the clustering algorithms that belong to a specific cluster model share common properties.

In chapter 3, we focus on the main problem of this thesis which is the famous k-means problem. In the k-means formulation, one is given an integer k and a set of n data points in R^d and the goal is to choose k centers so as to minimize ϕ, the sum of squared distances between each point and the closest centers. Chapter 3 contains the formal definitions of the k-means problem and some of its variations such as k-median problem and k-center problem. Geometric and analytic properties of these problems are examined so as to further understand

the relationship between these problems. Furthermore, useful results and definitions that will be used in the technical chapters 4 and 5 are presented.

After the necessary preliminaries, we discuss the computational difficulty of k-means problem. We show that k-means is NP-hard even when k is fixed to 2 (2-means). The NP-hardness proof can be divided into two parts. The first part contains a sequence of reductions: a standard restriction of 3SAT is reduced to a special case of NOT-ALL-EQUAL 3SAT and this problem is reduced to a generalization of 2-means. In the second part, we show that it is possible to get back from generalized 2-means to 2-means with the help of a proper embedding.

Although k-means is NP-hard, a very simple heuristic named Lloyd's algorithm or simply k-means algorithm is still used very widely today. Indeed, a 2002 survey of data mining techniques states that it "is by far the most popular clustering algorithm used in scientific and industrial applications" and it was also identified as one of the top 10 algorithms in data mining.

The major advantage of the k-means algorithm is its simplicity: it begins with k arbitrary "centers," chosen uniformly at random from the data points. Each point is then assigned to the nearest center, and each center is recomputed as the center of mass of all points assigned to it. These two steps, the assignment step and the update step, are repeated until the solution does not change between two consecutive rounds.

Regarding the speed of the k-means algorithm, it has been proved that it has a super-polynomial running time in the worst case. However, these worst case instances are very rare and usually the k-means algorithm is fast in practical applications. Another evidence for the empirical speed of the algorithm is the fact that it has polynomial smoothed complexity. Generally, if the smoothed complexity of an algorithm is low, then it is unlikely that the algorithm will take long time to solve practical instances whose data are subject to slight noises and imprecisions.

Despite its empirical speed and simplicity, the k-means algorithm may produce arbitrary bad clustering results. A very intuitive and simple randomized algorithm, dubbed k-means++, addresses this problem by selecting the initial centers more carefully: the first cluster center is chosen uniformly at random from the data points and each subsequent cluster center is chosen from the remaining data points with probability proportional to its squared distance from the point's closest existing cluster center. The intuition behind this approach is that the initial k cluster centers must be relatively spread out. After the selection of the initial centers, the k-means algorithm is applied.

k-means++ initialization gives strong guarantees regarding the quality of the solution produced before k-means algorithm is executed. Arthur and Vassilvitsky have shown that after choosing the k initial centers, k-means++ is $O(\log k)$-competitive to the optimum solution. Their proof is divided into two parts: first they show that k-means++ is constant competitive as long as it chooses each center from a new cluster in an optimal clustering. Then, using a highly non trivial inductive argument, they show that the total error in general is $O(\log k)$. The k-means steps that follow the seeding process can only improve the solution.

However, k-means++ has a major drawback which is its inherently sequential nature. The probability with which a point is chosen to be the ith depends critically on the previous $i-1$ centers. So, there is no obvious way to parallelize k-means++ making it less efficient when applied to big data. This fact motivated the question if it is possible to obtain an algorithm with the approximation guarantees of k-means++ that can be parallelized effectively.

The answer to this question came in 2012 when Bahmani et.al presented the k-means|| algorithm. k-means|| is highly inspired by k-means++, but selects more than one point in each iteration with independent sampling. This allows a parallel implementation of k-means|| in a parallel model of computation, like MapReduce.

We continue considering the k-means problem in the situation where the data is too large to be stored in main memory and must be accessed sequentially, such as from a disk, and where we must use as little memory as possible. In the streaming model points are read sequentially; when the data stream finishes, we must select k of these to designate as facilities.

The first algorithm, we examine, for the streaming variant of k-means works as follows: it "guesses" the cost of optimum, then runs the online facility location algorithm of Meyerson until either the total cost of the solution exceeds a constant time the guess or the total number of facilities exceeds some computed value l. Then, the end of a phase is declared, the guess is increased, the facilities are consolidated via matching, and the algorithm continuous with the next point. When the stream has been exhausted, the algorithm has l facilities, which are then consolidated down to k.

The algorithm achieves a constant-approximation for streaming k-means, stores only $O(k \log n)$ points in memory and runs in $O(nk \log n)$ time. Furthermore, the overall approximation can be improved to an FPTAS by applying a ball k-means step, if the data satisfies a data separability condition. Intuitively, if this condition does not hold, then the data is not well suited for clustering with the given value for k.

Despite its good theoretical performance, the algorithm is not good from a practical standpoint. The reason is that the constants hidden in the asymptotic notation are quite large and they are encoded into the algorithm, making it difficult to argue that the performance should improve for non-worst case inputs.

The second streaming algorithm for k-means presented in this thesis is a simplification of the previous one with the appropriate modifications. This modified streaming algorithm determines better facility cost as the stream is processed, removing unnecessary checks and allowing the user to parametrize what remains. The new algorithm is much more efficient when tested to realistic data.

Next, we consider the famous facility location problem. In the metric variant of this problem, we have a set of facility locations with an opening cost for each potential facility location, and a set of client locations. The goal is to open a subset of the facility locations so as to minimize the facility costs and the assignment costs for the clients, where a clients

assignment cost is its distance from its nearest facility. We then examine the relation of this problem with k-means and k-median.

At last, we study the online facility location algorithm of Meyerson. In the online variant of facility location, clients arrive one-by-one and must be assigned to an open facility upon arrival, without any knowledge about future demands. The decisions of opening a facility at a particular location and of assigning a client to some facility are irrevocable. The algorithm is very simple and intuitive and its approach to the online facility location problem has been the inspiration for the k-means++ algorithm.

In the case of uniform facility costs, the algorithm that achieves a constant approximation is the following. When a new demand point arrives, we measure the distance from this demand to the closest already-open facility. Suppose this distance is δ. With probability $\frac{\delta}{f}$, where f is the facility cost, we will open a new facility at this point. Otherwise, the demand is assigned to the closest open facility.

Chapter 2

Introduction to clustering

This initial chapter is dedicated to introduce the reader to the basics of clustering. We start by defining the clustering process and by categorizing clustering algorithms in terms of their cluster model. The classification of clustering algorithms based on the cluster model is an attempt to overcome the difficulty of defining precisely the general notion of "cluster".

However, there are some requirements that clustering algorithms from all cluster models must satisfy in order to be efficient. We simply mention the most considerable of them, and in chapters 4 and 5, we examine algorithms for the k-means clustering problem which are carefully designed to meet some of these requirements.

In the last section of this chapter, we present a large variety of clustering applications. Clustering is applied in many scientific areas varying from medicine to computer science and this indicates the importance of designing clustering algorithms. Furthermore, as data grows larger, the applications demand clustering algorithms that are good in terms of efficiency as well as solution quality.

2.1 Clustering and cluster models

Clustering is the process of organizing objects into groups with the property that objects belonging to a group(cluster) are more similar to each other than objects between different groups. Clustering is one of the classic problems in machine learning, data mining and computational geometry and a common technique for statistical data analysis used in many fields, including pattern recognition, image analysis and information retrieval.

The similarity criterion used to distinguish objects in the same cluster from objects belonging to other clusters differs from application to application. This means that clustering is not one specific algorithm but a general task solved by various algorithms that differ in their notion of what constitutes a cluster and how to efficiently find them. Clustering can be therefore be formulated as a multi-objective optimisation problem.

Although is difficult to give a precise definition of the notion "cluster"[46] we can categorize clustering algorithms in terms of their cluster model. Typical cluster models are:

Connectivity models

Connectivity based clustering, also known as hierarchical clustering, is based on the core idea of objects being more related to nearby objects than to objects farther away. These algorithms connect "objects" to form "clusters" based on their distance.

A hierarchical algorithm is a sequence of partitions in which each partition is nested into the next partition in the sequence. An agglomerative algorithm for hierarchical clustering starts with the disjoint sets of clusters, which places each input data point in an individual cluster. Pairs of clusters are then successively merged until the number of clusters reduces to the desired number. The clusters merged are the ones with the minimum distance. A divisive algorithm for hierarchical clustering reverses the process by starting with all objects in one cluster and subdividing into smaller pieces.

Centroid models

In centroid-based clustering, clusters are represented by a central vector, which may not necessarily be a member of the data set. When the number of clusters is fixed to k, k-means clustering gives a formal definition as an optimization problem: find the k cluster centers and assign the objects to the nearest cluster center, such that the squared distances from the cluster are minimized.

Most k-means-type algorithms require the number of clusters to be specified in advance, which is considered to be one of the biggest drawbacks of these algorithms. Furthermore, the algorithms prefer clusters of approximately similar size, as they will always assign an object to the nearest centroid. This often leads to incorrectly cut borders in between of clusters, which is not surprising, as the algorithm optimized cluster centers, not cluster borders.

Distribution models

The clustering model most closely related to statistics is based on distribution models. Clusters can then easily be defined as objects belonging most likely to the same distribution.

A very famous method for distribution based clustering is the Gaussian mixture models. Here, the data set is usually modelled with a fixed number of Gaussian distributions that are initialized randomly and whose parameters are iteratively optimized to fit better to the data set.

Density models

In density-based clustering[1] clusters are defined as areas of higher density than the remainder of the data set. Objects in these sparse areas - that are required to separate clusters - are usually considered to be noise and border points.

DBSCAN[2] and OPTICS[3] are the most prominent methods in density based clustering. DBSCAN is based on connecting points within certain distance thresholds. However, it only connects points that satisfy a density criterion, in the original variant defined as a minimum number of other objects within this radius. A cluster consists of all density-connected objects (which can form a cluster of an arbitrary shape, in contrast to many other methods) plus all objects that are within these objects' range. Another interesting property of DBSCAN is that its complexity is fairly low - it requires a linear number of range queries on the database - and that it will discover essentially the same results in each run, therefore there is no need to run it multiple times. OPTICS is a generalization of DBSCAN that removes the need to choose an appropriate value for the range parameter and produces a hierarchical result

Graph-based models

A clique, i.e., a subset of nodes in a graph such that every two nodes in the subset are connected by an edge can be considered as a prototypical form of cluster. Relaxations of the complete connectivity requirement (a fraction of the edges can be missing) are known as quasi-cliques.

A "clustering" is essentially a set of such clusters, usually containing all objects in the data set. Additionally, it may specify the relationship of the clusters to each other, for example a hierarchy of clusters embedded in each other. Clusterings can be roughly distinguished as:

- hard clustering: each object belongs to a cluster or not(all-or-nothing).
- fuzzy clustering: in contrast to hard clustering, this is a type of soft clustering in which data elements can belong to more than one cluster, and associated with each element is a set of membership levels. These indicate the strength of the association between that data element and a particular cluster. Fuzzy clustering is a process of assigning these membership levels, and then using them to assign data elements to one or more clusters.

2.2 Requirements

Most of the clustering algorithms must satisfy some or all of the below requirements, depending on the nature of the problem and the goal of the algorithm designer.

- scalability

  A very important property especially when we have to deal with massive data.

- dealing with different types of attributes

  Most clustering algorithms are limited to handling either numerical or categorical attributes, although datasets with mixed types of attributes are common in many applications.

- discovering clusters with arbitrary shape and size

  Many algorithms are only able to detect clusters of a specific shape(spherical) and clusters with similar sizes.

- minimal requirements for domain knowledge to determine input parameters

  We have already mentioned that determining the number of clusters in advance (k-means) is considered as one of the biggest drawbacks for a clustering algorithm.

- ability to deal with noise and outliers
- insensitivity to order of input records
- high dimensionality
- interpretability and usability

2.3 Applications

In this section, we will see a large variety of applications in which clustering is used and we will focus mostly on the computer science applications.

Biology, computational biology and bioinformatics

- Plant and animal ecology
  cluster analysis is used to describe and to make spatial and temporal comparisons of communities (assemblages) of organisms in heterogeneous environments; it is also used in plant systematics to generate artificial phylogenies[48] or clusters of organisms (individuals) at the species, genus or higher level that share a number of attributes.

- Transcriptomics
  clustering is used to build groups of genes with related expression patterns (also known as coexpressed genes). Often such groups contain functionally related proteins, such as enzymes for a specific pathway(a pathway consists of series of chemical reactions occurring within a cell), or genes that are co-regulated. High throughput experiments using expressed sequence tags (ESTs)[49] or DNA microarrays can be a powerful tool for genome annotation[50], a general aspect of genomics[51].

- Sequence analysis
  clustering is used to group homologous sequences into gene families. This is a very important concept in bioinformatics, and evolutionary biology in general.[52]

- High-throughput genotyping platforms
  clustering algorithms are used to automatically assign genotypes. [53]

- Human genetic clustering
  The similarity of genetic data is used in clustering to infer population structures. [54]

Medicine

- Medical imaging
  On PET scans [47], cluster analysis can be used to differentiate between different types of tissue and blood in a three-dimensional image. In this application, actual position does not matter, but the voxel(a value on a regular grid in three-dimensional space) intensity

is considered as a vector, with a dimension for each image that was taken over time. This technique allows, for example, accurate measurement of the rate a radioactive tracer is delivered to the area of interest, without a separate sampling of arterial blood, an intrusive technique that is most common today.

- IMRT segmentation
  Clustering can be used to divide a fluence map into distinct regions for conversion into deliverable fields in MLC-based Radiation Therapy.[55]

Business and marketing

- Market research
  Cluster analysis is widely used in market research when working with multivariate data from surveys and test panels. Market researchers use cluster analysis to partition the general population of consumers into market segments and to better understand the relationships between different groups of consumers/potential customers, and for use in market segmentation, Product positioning, New product development and Selecting test markets. [56]

- Grouping of shopping items
  Clustering can be used to group all the shopping items available on the web into a set of unique products. For example, all the items on eBay can be grouped into unique products. (eBay doesn't have the concept of a a stock keeping unit or SKU, which is a distinct item, such as a product or service, as it is offered for sale that embodies all attributes associated with the item and that distinguish it from all other items)[57] World wide web

- Social network analysis
  In the study of social networks, clustering may be used to recognize communities within large groups of people.[58]

- Search result grouping
  In the process of intelligent grouping of the files and websites, clustering may be used to create a more relevant set of search results compared to normal search engines like Google. There are currently a number of web based clustering tools such as Yippy(a metasearch engine developed by Vivísimo which offers clusters of results)[59]

- Slippy map optimization
  Flickr, an image hosting and video hosting website, as well as other map sites use clustering to reduce the number of markers on a map of photos. This makes it both faster and reduces the amount of visual clutter.[60]

Social science

- Crime analysis
  Cluster analysis can be used to identify areas where there are greater incidences of particular types of crime. By identifying these distinct areas or "hot spots" where a similar crime has happened over a period of time, it is possible to manage law enforcement resources more effectively. [61]

- Educational data mining
  Cluster analysis is for example used to identify groups of schools or students with similar properties. [62]

Computer science

- Software evolution

  Software evolution is the term used in software engineering to refer to the process of developing software initially, then repeatedly updating it for various reasons. The main goal of software evolution is to ensure reliability and flexibility of the system. [63]

  Clustering is useful in software evolution as it helps to reduce legacy properties in code by reforming functionality that has become dispersed. It is a form of restructuring and hence is a way of directly preventative maintenance.

- Image segmentation

  In computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as superpixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.[4, 5] Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

  Some of the practical applications of image segmentation are machine vision, medical imaging, object detection, recognition for example face, iris, and fingerprint recognition, traffic control systems and video surveillance.

  Clustering can be used to divide a digital image into distinct regions for border detection or object recognition. For example, the k-means algorithm which we will examine in the next chapter can be used to partition an image into k clusters.

- Recommender systems

  Recommender systems or recommendation systems (sometimes replacing "system" with a synonym such as platform or engine) are a subclass of information filtering system that seek to predict the 'rating' or 'preference' that user would give to an item.[6, 7]

  Recommender systems have become extremely common in recent years, and are applied in a variety of applications. The most popular ones are probably movies, music, news, books, research articles, search queries, social tags, and products in general. However, there are also recommender systems for experts, jokes, restaurants, financial services,[8] life insurance, persons (online dating), and twitter followers.[9]

Recommender systems sometimes use clustering algorithms to predict a user's preferences based on the preferences of other users in the user's cluster.

Chapter 3

Preliminaries

Clustering is a process widely used in many applications as we have seen in the previous chapter. In the first section of this chapter we focus on the k-means minimization problem and give the formal definition of this problem. Then, we present the most popular variations of k-means problem and examine their geometric and analytic properties. This gives a better understanding of these problems as we briefly discuss the similarities and differences between them.

The second section prepares the reader for the technical chapters that follow this chapter. We give the notation and definitions that will be used frequently and we prove some lemmas and theorems which are applied in the proofs of the next chapters. We start by giving the formal definitions of the problems that are involved in the NP-hardness proof of the k-means problem for the case of 2 clusters in general Euclidean space. Then, we present some algebraic results used in the NP-hardness proof and we end this section by briefly discussing the concept of smoothed complexity.

3.1 Problem formulation and variations

Consider data whose proximity measure is Euclidean distance. For our objective function, which measures the quality of clustering we use the the within-cluster sum of squares (WCSS), also known as scatter. In other words, we calculate the error of each data point, i.e., its Euclidean distance to the closest centroid and then compute the total sum of squared errors. We first give the definition of the norm.

Norm Given a vector space V over a subfield F of the complex numbers, a norm on V is a function $\|\| : V \to R$ with the following properties:

For all $a \in F$ and all $u, v \in V$,

1. $\|v\| \geq 0$ (positivity)
2. $\|av\| = |a|\|v\|$, (absolute homogeneity or absolute scalability)
3. $\|u + v\| \leq \|u\| + \|v\|$ (triangle inequality or subadditivity)

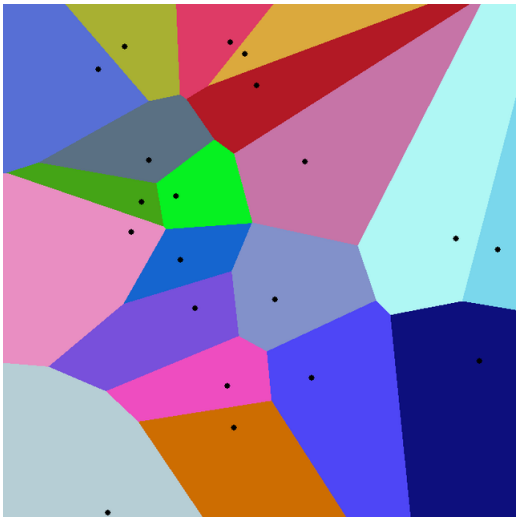
4. If $\|v\| = 0$ then v is the zero vector (separates points)

A vector space on which a norm is defined is called a normed vector space. Now we use the definition of the norm to formally define the Euclidean k-means clustering problem.

Euclidean k-means Given a set X of n observations $(x_1, x_2, .., x_n)$ where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k sets ($k \leq n$) C=$(C_1, C_2, .., C_k)$ so as to minimize the within-cluster sum of squares(WCSS):

$$\arg\min_{C} \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2 \text{ where } \mu_j \text{ is the mean of points in } C_j$$

The result of a k-means clustering can be seen as a partitioning of the Euclidean space into Voronoi cells. In the particular case where the space is a finite-dimensional Euclidean space, there are finitely many points and all of them are different, the Voronoi cells are convex polytopes and they can be represented in a combinatorial way using their vertices, sides, 2-dimensional faces, etc. However, in general the Voronoi cells may not be convex or even connected. An illustration of a Voronoi diagram when Euclidean distance is used follows.



There is a variant of the k-means problem known as the discrete k-means problem where the centers have to be points from X itself. In this case the optima of the k-means problem and its discrete variant are within constant factors of each other.

There are other variants where the objective is to minimize the sum of p-th powers of norms $(\sum_{x \in X} \min_{c \in C} \|x - c\|^p)^{1/p}$ The $p = 1$ case is known as the k-median problem, where instead of calculating the mean for each cluster to determine its centroid, one calculates the median. The norm for this case is called the taxicab norm or the Manhattan norm, since it relates to the distance a taxi has to drive in a rectangular street grid to get from the origin to the point x. The distance derived from this norm is called the Manhattan distance or L_1 distance.

k-median clustering

Input: Finite set $S \subset X$; integer k.

Output: $T \subset S$ with $|T| = k$.

Goal: Minimize cost(T)= $\sum_{x \in S} \|x - T\|_1$

Like k-means clustering, k-median clustering partitions the space into Voronoi cells. We give below an illustration of the Voronoi diagram induced by a k-median clustering of the same 20 points used for the k-means clustering above.

k-center clustering

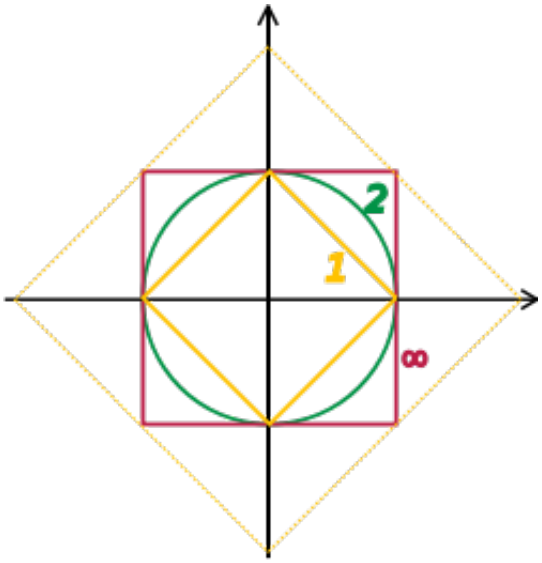
Input: Finite set $S \subset X$; integer k.

Output: $T \subset X$ with $|T| = k$.

Goal: Minimize cost(T)= $\max_{x \in S} \|x - T\|_\infty$

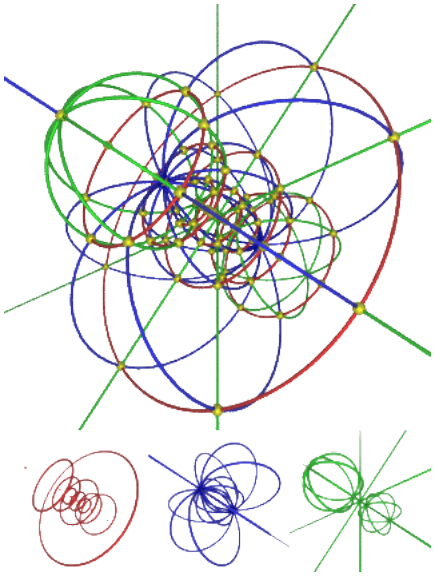
Note that the k-median cost function is more robust to outliers than the k-center cost function. Furthermore a k-median algorithm is forced to pick points from the metric space X that belong to a subset of S, whereas in the k-center problem T is a subset of X.

From a geometrical view, the concept of unit circle (the set of all vectors of norm 1) is different in different norms. The characteristic shape of the unit circle for the Manhattan-norm in R_2 is a diamond(square), for the Euclidean norm it is the unit circle, while for the infinity norm it is a different square. Generally, for any p-norm it is a superellipse with congruent axes.

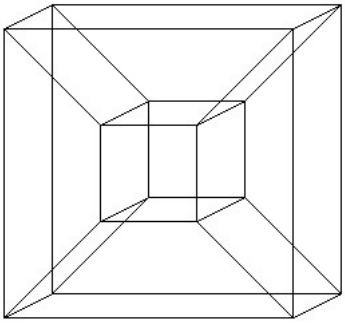


In general metric spaces of higher (finite) dimension we have the following geometrical results:

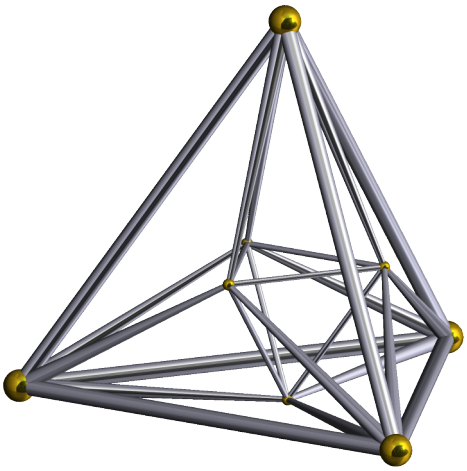
- The set of vectors in R^{n+1} whose Euclidean norm is a given positive constant forms an n-sphere.



- The set of vectors whose infinity norm is a given constant, c, forms the surface of a hypercube with edge length $2c$.



- The set of vectors whose Manhattan norm is a given constant forms the surface of a cross polytope of dimension equivalent to that of the norm minus 1.



3.2 Definitions and useful results

This section contains the formulations of NP-hard problems as well as Schoenberg's theorem which are used in the NP-hardness proof of k-means problem. We also discuss briefly the concept of smoothed complexity and make a brief introduction to online and streaming algorithms.

First, we give the definitions of the three minimizations problems used in the hardness proof of k-means for the case of general Euclidean space d with k=2. We call this problem 2-means.

The first problem is a standard restriction of 3SAT known to be NP-complete.

3SAT

Input: A Boolean formula in 3CNF, where each clause has exactly three literals and each variable appears at least twice.

Output: true if formula is satisfiable, false if not.

The second problem is a special case of NOT-ALL-EQUAL 3SAT, which is also NP-complete.

NAESAT*
Input: A Boolean formula $\phi(x_1, ..., x_n)$ in 3CNF, such that (i) every clause contains exactly three literals, and (ii) each pair of variables x_i, x_j appears together in at most two clauses, once as either $\{x_i, x_j\}$ or $\{\bar{x}_i, \bar{x}_j\}$, and once as either $\{\bar{x}_i, x_j\}$ or $\{x_i, \bar{x}_j\}$.
Output: true if there exists an assignment in which each clause contains exactly one or two satisfied literals; false otherwise.

The final problem we consider is a generalization of 2-means.

Generalized 2-means
Input: An $n \times n$ matrix of interpoint distances D_{ij} .
Output: A partition of the points into two clusters C_1 and C_2, so as to minimize

$$\sum_{j=1}^{2} \frac{1}{|C_j|} \sum_{i,i' \in C_j} D_{ii'}$$

In the generalized 2-means problem the objective function is not the same as in the original formulation of the problem. For this, note that in any optimal solution, the mean μ_j of the points in C_j can be removed entirely from the formulation of the problem. To this end, let X, Y be i.i.d. random draws from C_j . Simple algebra shows $E\|X - Y\|^2 = 2E\|X - EX\|^2$ which implies
$\sum_{i \in C_j} \|x_i - \mu_j\|^2 = \sum_{i,i' \in C_j} \|x_i - x_i'\|^2$
therefore the k-means cost function can equivalently be rewritten as

$$\sum_{j=1}^{k} \sum_{i,i' \in C_j} \|x_i - x_i'\|^2 \ .$$

The next theorem is often used to describe isometric embeddings of Hilbert spaces.

Theorem (Schoenberg) Let H denote the matrix $I - (1/N)11^T$. An $N \times N$ symmetric matrix D can be embedded into l_2^2 if and only if $-HDH$ is positive semidefinite.

Smoothed analysis

Daniel Spielman and Shang-Hua Teng introduced in 2001 the concept of smoothed analysis[19]. Their motivation was the simplex algorithm, which although has a an exponential worst case complexity, in most practical application runs very fast. They showed that if the smoothed complexity of an algorithm is low, then it is unlikely that the algorithm will take long time to solve practical instances whose data are subject to slight noises and imprecisions.

Smoothed analysis is a hybrid of worst-case and average-case analyses that inherits advantages of both, by measuring the expected performance of algorithms under slight random perturbations of worst-case inputs. That is, if $T(x)$ is the time that an algorithm consumes when running with input x, then

- worst case analysis measures $\max_x T(x)$
- average case analysis measures $\operatorname{ave}_x T(x)$
- smoothed analysis measures $\max_x \operatorname{ave}_r T(x + \varepsilon r)$ for $\varepsilon > 0$

Online and Streaming algorithms

In contrast to an offline algorithm, which is given the whole problem data from the beginning, an online algorithm is one that can process its input piece-by-piece in a serial fashion, without having the entire input available from the start. In the same spirit, streaming algorithms are algorithms for processing data streams in which the input is presented as a sequence of items and can be examined in only a few passes (typically just one).

Online and streaming algorithms must make decisions without knowledge of the future, so these decisions may later turn out not to be optimal. Competitive analysis measures the performance of online and streaming algorithms by comparing the relative performance of an online (or streaming) and offline algorithm for the same problem instance. Specifically, the competitive ratio of an online algorithm, is defined as the worst-case ratio of its cost divided by the optimal cost, over all possible inputs.

As we have argued before, besides the running time and the solution quality, the performance of the streaming algorithm is measured by:

- The number of passes the algorithm must make over the stream
- The available memory

We must not confuse streaming algorithms with online algorithms. They both require decisions to be made before all data are available, but they have a main difference. Streaming algorithms only have limited memory available but they may be able to defer action until a group of points arrive, while online algorithms are required to take action as soon as each point arrives.

Chapter 4

k-means problem offline

In the first section of this chapter, we show the NP-hardness of k-means in general Euclidean space even for 2 clusters. The first part of the NP-hardness proof consists of a sequence of reductions where a restriction of 3SAT is reduced to NAESAT* and NAESAT* is reduced to Generalized 2-means(See previous chapter for definitions of these problems). We complete the proof by showing that the distance matrix D of generalized 2-means can be realized by squared Euclidean distances. That is, we show that D can be embedded into l_2^2.

After showing the NP-hardness of k-means problem, we present algorithms for k-means in the offline version i.e., where the space with all the data points are given right from the start. Despite the computational difficulty of the problem, a very simple heuristic-Lloyd's algorithm or simply k-means algorithm-remains the most popular clustering method. Starting with a set of randomly chosen initial centers, one repeatedly assigns each input point to its nearest center, and then recomputes the centers given the point assignment. The algorithm continues until the solution does not change between two consecutive rounds.

k-means is simple and is usually fast in practical applications. It has polynomial smoothed complexity, although the worst case running time can be super-polynomial. Despite its empirical speed, there are examples where k-means can be arbitrary bad in terms of solution quality. k-means++ algorithm addresses this problem by choosing carefully the k-initial centers. It selects the first center uniformly at random, and each subsequent center is selected with probability proportional to its contribution to the overall error given in the previous selections. This procedure achieves already an $O(\log k)$ competitive ratio and then the Lloyd's iterations that follow can only decrease the potential function.

In the last section of this chapter, we present an algorithm which is highly inspired by k-means++. Dubbed k-means||, the algorithm can be effectively parallelized, unlike k-means++, which is inherently sequential. The parallel version of the algorithm can be implemented in the MapReduce model of computation.

4.1 Hardness of the k-means problem

Regarding computational complexity, finding the optimal solution to the k-means clustering problem for n observations in d dimensions is:

1. NP-hard for a general number of clusters k even in the plane[10]

2. NP-hard in general Euclidean space d even for 2 clusters[12, 11]

3. If k and d (the dimension) are fixed, the problem can be exactly solved[22]

The first hardness result was proven by Mahajan, Nimbhorkar and Varadarajan in 2009. Their proof uses a reduction from the planar 3SAT problem and is inspired by a construction used by Megiddo and Supowit in the context of showing the NP-hardness of the k-center and the k-median problem. Given an instance I of planar 3SAT, they construct a planar graph from I, embed it in an integer grid and construct the k-means instance from this grid embedding.

k-means is also NP-hard in general Euclidean space d even for 2 clusters. We will call this restriction of the k-means problem 2-means. There are many NP-hardness proofs for 2-means. In the next lines, we will examine one of them which is due to Sanjoy Dasgupta.

We will divide the proof into two parts. The first part consists of a sequence of reductions involving three problems: a standard restriction of 3SAT well known to be NP-complete, a special case of NOT-ALL-EQUAL 3SAT, which we call NAESAT*, and a generalization of 2-means. We will omit the proof of 3SAT$\preceq$NAESAT*, which can be found in [21]. In the second part, we will see a way to get from the generalization of 2-means problem back to the original 2-means problem.

Now we proceed to the first part of the proof, which is to show the reduction of NAESAT* to Generalized 2-means. For any input $\phi(x_1, ..., x_n)$ to NAESAT*, we efficiently construct a $2n \times 2n$ distance matrix $D(\phi)$ and a threshold $c(\phi)$ such that ϕ satisfies NAESAT* if and only if $D(\phi)$ admits a generalized 2-means clustering of cost at most $c(\phi)$.

For every literal in NAESAT*, we have a corresponding row and column in the distance matrix. Entries of this matrix will be indexed as $D_{\alpha,\beta}$ for $\alpha, \beta \in \{x_1, ..., x_n, \bar{x}_1, ..., \bar{x}_n\}$. We write $\alpha \sim \beta$ to mean that either α and β occur together in a clause or $\bar{\alpha}$ and $\bar{\beta}$ occur together in a clause. We observe that the definition of NAESAT* ensures that every $\alpha \sim \beta$ is generated by a unique clause; it is not possible to have two different clauses that both contain either α, β or $\bar{\alpha}, \bar{\beta}$.
Define

$$D_{\alpha,\beta} = \begin{cases} 0 & \alpha = \beta \\ 1 + \Delta & \alpha = \bar{\beta} \\ 1 + \delta & \alpha \sim \beta \\ 1 & otherwise \end{cases}$$

Here $0 < \delta < \Delta < 1$ are constants such that $4\delta m < \Delta \leq 1 - 2\delta n$, where m is the number of clauses of ϕ . One valid setting is $\delta = 1/(5m + 2n)$ and $\Delta = 5\delta m$.

The next step is to show that ϕ is a "yes" instance of NAESAT* if and only if $D(\phi)$ admits a generalized 2-means clustering of cost at most $c(\phi)$. The intuition behind the construction of $D(\phi)$ is that if ϕ is satisfiable for NAESAT*, then an obvious clustering is to make one cluster for the positive literals and one for the negative literals. So the distances of points (literals) within a cluster must be smaller(1 or $1+\delta$)than the distances of points (literals) between the clusters ($1+\Delta$). Choosing appropriate values for δ, Δ and $c(\phi)$ we have the following lemmas which complete the reduction.

Lemma 4.1 If ϕ is a satisfiable instance of NAESAT*, then $D(\phi)$ admits a generalized 2-means clustering of cost $c(\phi) = n - 1 + 2\delta m/n$, where m is the number of clauses of ϕ.

Proof. The obvious clustering is to make one cluster (say C_1) consist of the positive literals in the satisfying not-all-equal assignment and the other cluster (C_2) the negative literals. Each cluster has n points, and the distance between any two of them can be 1 or $1+\delta$. This fact follows from the definition of the distance matrix and the clustering because if $\alpha = \bar{\beta}$ then the points belong to different clusters. Each clause of ϕ has at least one literal in C_1 and at least one literal in C_2, since it is a not-all-equal assignment. Hence it contributes exactly one $\sim$ pair to C_1 and one $\sim$ pair to C_2. Thus the clustering cost is

$$\frac{1}{2n}\sum_{i,i'\in C_1} D_{i,i'} + \frac{1}{2n}\sum_{i,i'\in C_2} D_{i,i'} = 2\cdot\frac{1}{n}\left(\binom{n}{2} + m\delta\right) = n - 1 + \frac{2\delta m}{n}$$

The following lemma reflects the intuition about the obvious clustering we mentioned in lemma 1. That is, if the cost of the clustering exceeds the threshold $c(\phi)$ then there is a cluster that contains both a variable and its negation.

Lemma 4.2 Let C_1, C_2 be any 2 clustering of $D(\phi)$. If C_1 contains both a variable and its negation, then the cost of this clustering is at least $n - 1 + \Delta/(2n) > c(\phi)$.

Proof. Suppose C_1 has n' points while C_2 has $2n - n'$ points. Since all distances are at least 1 and since C_1 contains a pair of points at distance $1 + \Delta$, the total clustering cost is at least

$$\frac{1}{n'}\left(\binom{n'}{2} + \Delta\right) + \frac{1}{2n-n'}\binom{2n-n'}{2} = n - 1 + \frac{\Delta}{n'} \geq n - 1 + \frac{\Delta}{2n}$$

Lemma 4.3 If $D(\phi)$ admits a 2-clustering of cost $\leq c(\phi)$, then ϕ is a satisfiable instance of NAESAT*.

Proof. Let C_1, C_2 be a 2-clustering of cost $\leq c(\phi)$. By the previous lemma, neither C_1 nor C_2 contain both a variable and its negation. Thus $|C_1| = |C_2| = n$. The cost of the clustering can be written as

$$\frac{2}{n}\left(\binom{n}{2} + \delta\sum_{clauses}(1 \text{ if clause is split between } C_1, C_2;\ 3 \text{ otherwise})\right)$$

Since the cost is $\leq c(\phi)$, it follows that all clauses are split between C_1 and C_2, that is, every clause has at least one literal in C_1 and one literal in C_2. Therefore, the assignment that sets all of C_1 to true and all of C_2 to false is a valid NAESAT* assignment for ϕ.

The NP-hardness proof of generalized 2-means clustering problem follows from the combination of lemma 4.1 and lemma 4.3. The next step is to show that the distance matrix $D(\phi)$ can be

realized by squared Euclidean distances. This existential fact is also constructive because the embedding can be obtained in cubic time by multidimensional scaling.[14]

Embeddability of $D(\phi)$

We now show that $D(\phi)$ can be embedded into l_2^2 , in the sense that there exist points $x_\alpha \in \mathbb{R}^{2n}$ such that $D_{\alpha,\beta} = \|x_\alpha - x_\beta\|^2$ for all α, β. To this end, we prove the following corollary derived from the classical theorem of Schoenberg (subsection 3.2).[15]

Corollary An $N \times N$ symmetric matrix D can be embedded into l_2^2 if and only if $u^T D u \leq 0$ for all $u \in \mathbb{R}^N$ with $u \cdot \mathbf{1} = 0$.

Proof. Since the range of the map $\mapsto Hv$ is precisely $\{u \in \mathbb{R}^N : u \cdot \mathbf{1} = 0\}$, we have $-HDH$ is positive semidefinite $\Leftrightarrow v^T HDHv \leq 0$ for all $v \in \mathbb{R}^N \Leftrightarrow u^T Du \leq 0$ for all $u \in \mathbb{R}^N$ with $u \cdot \mathbf{1} = 0$.

Lemma 4.4 $D(\phi)$ can be embedded into l_2^2.

Proof. If ϕ is a formula with variables $x_1, ..., x_n$, then $D = D(\phi)$ is a $2n \times 2n$ matrix whose first n rows/columns correspond to $x_1, ..., x_n$ and the remaining rows/columns correspond to $barx_1, ..., barx_n$. The entry for literals (α, β) is $D_{\alpha\beta} = 1 - \mathbf{1}(\alpha = \beta) + \Delta \cdot textbf1(\alpha = \bar{\beta}) + \delta \cdot (\alpha \sim \beta)$, where $\mathbf{1}(\cdot)$ denotes the indicator function.

Now, pick any $u \in \mathbb{R}^{2n}$ with $u \cdot \mathbf{1} = 0$. Let u^+ denote the first n coordinates of u and u^- the last n coordinates.

$$u^T Du = \sum_{\alpha,\beta} D_{\alpha\beta} u_\alpha u_\beta$$

$$= \sum_{\alpha,\beta} u_\alpha u_\beta (1 - \mathbf{1}(\alpha = \beta) + \Delta \cdot \mathbf{1}(\alpha = \bar{\beta}) + \delta \cdot (\alpha \sim \beta)$$

$$= \sum_{\alpha,\beta} u_\alpha u_\beta - \sum_\alpha u_\alpha^2 + \Delta \sum_\alpha u_\alpha u_{\bar{\alpha}} + \delta \sum_{\alpha,\beta} u_\alpha u_\beta \cdot \mathbf{1}(\alpha \sim \beta)$$

$$\leq (\sum_\alpha u_\alpha)^2 - \|u\|^2 + 2\Delta(u^+ \cdot u^-) + \delta \sum_{\alpha,\beta} |u_\alpha||u_\beta|$$

$$\leq -\|u\|^2 + \Delta(\|u^+\|^2 + \|u^-\|^2) + \delta(\sum_\alpha u_\alpha)^2$$

$$\leq -(1 - \Delta)\|u\|^2 + 2\delta\|u\|^2 n$$

where the last step uses the Cauchy-Schwarz inequality. Since $2\delta n \leq 1 - \Delta$ this quantity is always $\leq$0.

4.2 The k-means algorithm

Clustering is a central problem in data management and has a rich and illustrious history with literally hundreds of different algorithms published on the subject. Even so, a single method remains the post popular clustering method;in fact, it was identified as one of the top 10 algorithms in data mining.[16] This algorithm, usually referred as k-means algorithm

or Lloyd's algorithm was first proposed by Stuart Lloyd in 1957 as a technique for pulse-code modulation, though it wasn't published outside Bell labs until 1982.

Lloyd's algorithm [17] begins with k arbitrary centers, typically chosen uniformly at random from the data points. Each point is then assigned to its nearest center, and each center is recomputed as the center of mass of all points assigned to it. These two steps (assignment and update) are repeated until the process stabilizes.

Algorithm 1: k-means

1: Arbitrarily choose k initial centers $C = \{c_1, c_2, ..., c_k\}$
2: For each $i \in \{1, ..k\}$, set the cluster C_i to be the set of points in X that are closer to c_i than they are to c_j for all j$\neq i$
3: For each $i \in \{1, 2, .., k\}$, set c_i to be the center of mass of all points in C_i.
4: Repeat Steps 2 and 3 until C no longer chances.

Convergence and Termination

In order to prove that Lloyd's algorithm converges we must check that the total error $\phi = \sum_{x \in X} \min_{c \in C} \|x - c\|^2$ is monotonically decreasing. This ensures that no clustering is repeated during the course of the algorithm and since there are at most k^n possible clusterings, the process will always terminate.

It is obvious that step 2 decreases ϕ. For step 3, it is helpful to recall a standard result from linear algebra.

Lemma 4.5 Let S be a set of points with center of mass $c(S)$, and let z be an arbitrary point. Then, $\sum_{x \in S} \|x - z\|^2 - \sum_{x \in S} \|x - c(S)\|^2 = |S| \cdot \|c(S) - z\|^2$

Monotonicity for step 3 follows from taking S to be a single cluster and z to be its initial center.

4.2.1 Running time

In the worst case, k-means can be very slow to converge: in particular it has been shown that there exist certain point sets, even in 2 dimensions, on which k-means takes exponential time, that is $2^{\Omega(n)}$, to converge. We give the high level intuition behind the construction of these points and the complete proof can be found in [18]. The idea is simple and can be related to the saying "Who watches the watchmen".

Consider a sequence of t watchmen $W_0, W_1, ..., W_{t-1}$. A "day" of a watchman W_i $(i > 0)$ can be described as follows : W_i watches W_{i-1}, waking it up once it falls asleep, and does so twice; afterwards, W_i falls asleep itself. The watchman W_0 instead will simply fall asleep directly after it has been woken up. We observe that the days of the watchmen are not synchronized. Now if we begin with a configuration where each watchman is awake (or even just W_{t-1}), it is

clear that W_0 will be woken up $2^{\Omega(t)}$ times by the time that every watchman is asleep. In the construction, we have a sequence of gadgets $G_0, G_1, ..., G_{t-1}$, where all gadgets G_i with $i > 0$ are identical up to a uniform scale factor. Any gadget G_i has a fixed number of points and two centers, and G_i's state (stage of the day) is determined by the partition of its points induced by the current set of clusters. The clustering indicating that G_i "fell asleep" has one center in a particular position S_i^*. This position S_i^* is not an input point, but rather a distinguished point in the plane.

In the situation when G_{i+1} is awake and G_i falls asleep, some points of G_{i+1} will be assigned temporarily to the center of G_i located in S_i^* ; in the next step this center will move so that in one more step the initial clustering (or "morning clustering") of G_i is restored: this models the fact that G_{i+1} wakes up G_i.

Note that since each gadget has a constant number of centers, we can build an instance with k clusters that has $t = \Theta(k)$ gadgets, for which k-means will require $2^{\Omega(k)}$ iterations. Also since each gadget has a constant number of points, we can build an instance of n points and $k = \Theta(n)$ clusters with $t = \Theta(n)$ gadgets. This will imply a lower bound of $2^{\Omega(n)}$ on the running time of k-means.

Although the worst-case running time of k-means algorithm is super-polynomial, very few iterations are usually required in practice. This is corroborated by the fact that the smoothed running time of k-means is polynomial. Smoothed analysis[19] considers the running time of an algorithm after first randomly perturbing the input. Intuitively, this models how fragile worst-case instances are and if they could reasonably arise in practice.

Smoothed complexity of the k-means algorithm

Arthur et al's[20] idea was to prove, first, that the potential after one iteration is bounded by some polynomial and, second, that the potential decreases by some polynomial amount in every iteration (or, more precisely, in every sequence of a few consecutive iterations). To do this, they have proved upper bounds on the probability that the minimal improvement is small.

In order to show the polynomial bound, they reduced the number of cases in the union bound of the n^{3kd} possible clusterings by introducing the notion of transition blueprints. Basically, every iteration of k-means can be described by a transition blueprint. The blueprint describes the iteration only roughly, so that several iterations are described by the same blueprint. Intuitively, iterations with the same transition blueprint are correlated in the sense that either all of them make a small improvement or none of them do. This dramatically reduces the number of cases that have to be considered in the union bound. On the other hand, the description conveyed by a blueprint was still precise enough to allow them to bound the probability that any iteration described by it makes a small improvement.

4.2.2 Solution quality

Unfortunately, the empirical speed and simplicity of the k-means algorithm come at the price of the accuracy. It is guaranteed only to find a local optimum which can often be quite

poor. There are many natural examples for which the algorithm generates arbitrarily bad examples(i.e $\frac{\phi}{\phi_{OPT}}$ is unbounded even when n and k are fixed). Furthermore, these examples do not rely upon an adversarial placement of the starting centers, and the ratio can be unbounded with high probability even with the standard randomized seeding technique. For an illustration of a bad case example see Figure 1.

Improving the accuracy of the k-means method has been the subject of many papers. Inaba et.al[22] where the first to give an exact algorithm for the k-means problem with running time of $O(n^{kd})$. After this work, a lot of PTAS have been developed [23, 25, 24]which have exponential(or worse) dependence on k and that makes them impractical.

Constant competitive algorithms algorithms have been proposed from Kanungo et al.[26], and Mettu and Plaxton[27] . In the next section, we will examine an algorithm for the k-means problem which is $O(logk)$-competitive and has $O(nkd)$ running time.

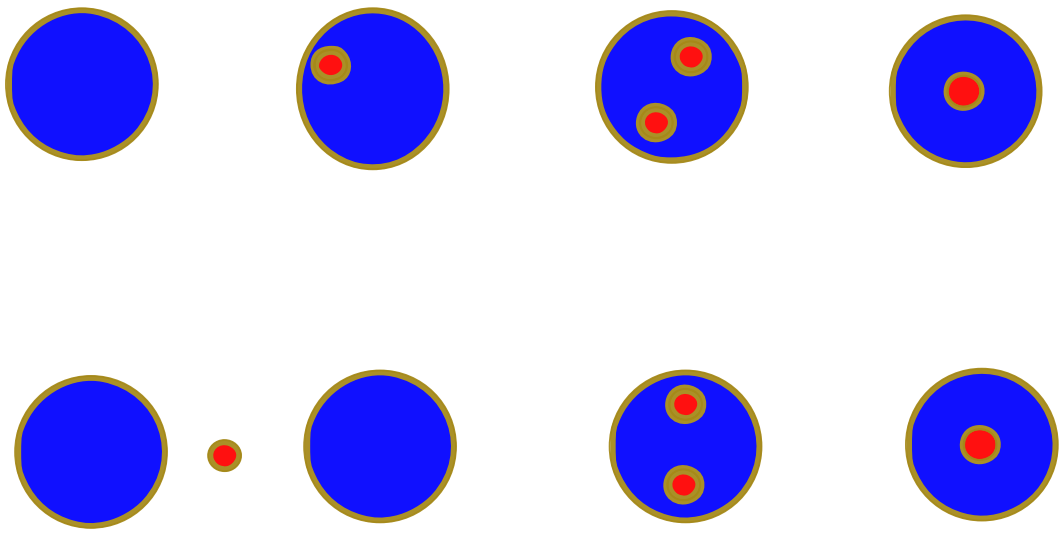


Figure 1: We illustrate a bad example regarding the k-means initialization. k-means initially selects no point from the first cluster, 1 point from the second cluster , 2 points from the third cluster and one point from the last cluster. This results to the clustering below. Note that k-means merges the first 2 clusters. This can result to an arbitrary bad clustering: we can stretch the distance of the first 2 clusters arbitrary.

4.3 The k-means++ algorithm

The topic of this section is the famous k-means++ [28]algorithm which was proposed by David Arthur and Sergei Vassilvitskii in 2007. k-means++ was an attempt to improve the solution quality produced by the standard k-means algorithm, which can be arbitrarily bad as we discussed in the previous section. In 2006, an algorithm, which is essentially identical to k-means++ has been proposed by Ostrovsky et.al[29], although the analysis was quite different.

k-means++ is the algorithm generated by augmenting k-means with a very simple and intuitive randomized seeding technique. The analysis of k-means++ shows with this simple initialization technique, which is similar in spirit to that used by Meyerson for online facility location, k-means becomes $\Theta(logk)$-competitive with the optimal clustering.

The intuition behind k-means++ is that a good clustering must be relatively spread out. So, when selecting a new center, preference should be given to those further away from the previously selected centers. In order to achieve this, the algorithm chooses the centers one by one in a controlled fashion, where the current set of chosen centers will stochastically bias the choice of the next center. Specifically, k-means++ chooses a point p as a center with probability proportional to p's contribution to the overall potential function.

Let $D(x)$ denote the shortest distance from a data point x to the closest center we have already chosen. Then, we define k-means++:

Algorithm 2: k-means++

1: Choose one center uniformly at random from among the data points.
2: For each data point x, compute $D(x)$.
3: Choose one new data point x' as a new center, using a weighted probability distribution where a point x is chosen with probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$
4: Repeat Steps 2 and 3 until k centers have been chosen.
5: Proceed as with the standard k-means algorithm.

Step 3 is called "D^2" weighting.

k-means++ is O(logk)-competitive

Arthur and Vassilvitskii showed that k-means++ is $O(logk)$-competitive to the optimum solution, right after choosing the k initial centers with "D^2" weighting(after step 4). Then, the standard k-means steps can only decrease the potential ϕ.

We will divide the analysis of the performance of k-means into two parts. Before we proceed to the first part, a little bit of notation is needed. We write C_{OPT} to denote the optimal clustering for a given instance of the k-means problem, ϕ_{OPT} denotes the corresponding potential and $\phi(A)$ denotes the contribution of $A \subset X$ to the potential(i.e., $\phi(A) = \sum_{x \in A} \min_{c \in C} \|x - c\|^2$).

In the first part, we will show that k-means++ is constant competitive to the optimum solution provided that after choosing the first center, it chooses each following center from a new cluster in an optimal clustering. That is, we fix an optimum clustering and we assume that k-means++ picks its first center uniformly at random and then it always picks a center from a new optimal cluster (with "D^2"). Then the cost of the clustering will be at most 8 times the cost of the optimal clustering.

We will bound the total error of the first cluster proving the following lemma:

Lemma 4.6. Let A be an arbitrary cluster in C_{OPT}, and let C be the clustering with just one center, which is chosen uniformly at random from A. Then, $E[\phi(A)] = 2\phi_{OPT}(A)$.

Proof. Let $c(A)$ denote the center of mass of the data points in A. Then
$E[\phi(A)] = \sum_{a_0 \in A} (\sum_{a \in A} \|a - a_0\|^2)$
By Lemma 4, we know that since C_{OPT} is optimal, it must be using $c(A)$ as the center to the cluster A. So,
$\sum_{a_0 \in A} (\sum_{a \in A} \|a - a_0\|^2) = \frac{1}{|A|} \sum_{a_0 \in A} (\sum_{a \in A} \|a - c(a)\|^2 + |A| \cdot \|a_0 - c(A)\|^2) = 2 \sum_{a \in A} \|a - a_0\|^2$,where the last equality follows by applying Lemma 4 again

Now we will bound the total error of the next clusters, in which k-means picks the next centers, assuming that it is always chooses from a new cluster in the optimal clustering. The following lemma completes the first part of the analysis:

Lemma 4.7. Let A be an arbitrary cluster in C_{OPT}, and let C be an arbitrary clustering. If we add a random center to C from A chosen with "D^2" weighting, then $E[\phi(A)] \leq 8\phi_{OPT}(A)$

Proof. We choose some fixed a_0 as our center, given that we are choosing from the new optimal cluster A with "D^2" weighting, with probability $\frac{D(a_0)^2}{\sum_{a \in A} D(a)^2}$. After choosing a_0, a point a will contribute precisely $min(D(b), \|b - b_0\|)^2$ to the potential. Therefore,

$\mathrm{E}[\varphi(A)] = \sum_{a_0 \in A} \frac{D(a_0)^2}{\sum_{a \in A} D(a)^2} \sum_{a \in A} min(D(a), \|a - a_0\|)^2$

The key step here is the triangle inequality. Applying the triangle inequality:

$D(a_0) \leq D(a) + \|a - a_0\|$

By the power mean inequality:

$D(a_0)^2 \leq 2D(a) + 2\|a - a_0\|^2$

Averaging over all a:
$D(a_0)^2 \leq \frac{2}{|A|} \sum_{a \in A} D(a)^2 + \frac{2}{|A|} \sum_{a \in A} \|a - a_0\|^2$

Therefore $E[\phi(A)]$ is at most,

$$\frac{2}{|A|} \cdot \sum_{a_0 \in A} \frac{\sum_{a \in A} D(a)^2}{\sum_{a \in A} D(a)^2} \cdot \sum_{a \in A} min(D(a), \|a - a_0\|)^2$$
$$+ \frac{2}{|A|} \cdot \sum_{a_0 \in A} \frac{\sum_{a \in A} \|a - a_0\|^2}{\sum_{a \in A} D(a)^2} \cdot \sum_{a \in A} min(D(a), \|a - a_0\|)^2$$

In the first expression, we substitute $\min(D(a), \|a - a_0\|^2 \leq \|a - a_0\|^2)$ and in the second expression we substitute $\min(D(a), \|a - a_0\|^2 \leq D(a)^2$.
After this substitutions, $E[\phi(A)]$ is at most

$\frac{4}{|A|} \sum_{a_0 \in A} (\sum_{a \in A} \|a - a_0\|^2) = 8\phi_{OPT}(A)$

The last equality follows from the previous lemma

Lemma 4.6 and Lemma 4.7 complete the first part of the k-means++ algorithm's analysis. We have shown that if clusters are well separated, and we always pick a center from a new optimal

cluster, the algorithm is 8 competitive.

Now we need to show that the total error in general is $O(logk)$. Intuitively, this means that if no points from a cluster are picked, then the cluster does not contribute very much to the total error. The proof involves a highly non-trivial inductive argument and the induction is on the steps of the algorithm.

Lemma 4.8 Let C be an arbitrary clustering. Choose $u > 0$ "uncovered" clusters from C_{OPT} and let X_u denote the set of points in these clusters. Also let $X_c = X - X_u$. Now suppose we add $t \leq u$ random centers to C,chosen with D^2 weighting. Let C' denote the resulting clustering,and let φ' denote the corresponding potential. Then,$E[\phi']$ is at most $(\phi(X_c) + 8\phi_{OPT}(X_u)(1 + H_t) + \frac{u-t}{u}\phi(X_u)$.

Here, H_t, denotes the harmonic sum, $1 + \frac{1}{2} + ... + \frac{1}{t}$.

Proof The notions of "covered" and "uncovered" clusters simplify the analysis of the performance of the algorithm. In the first part, we have shown that k-means++ is constant competitive as long as it chooses centers from each cluster of C_{OPT}. That means that k-means++ is constant competitive as long as all clusters from C_{OPT} are "covered".
We now use induction to show that the "uncovered" clusters(the ones from which k-means++ has not picked a point) from C_{OPT} do not contribute much to the total error. In order to prove this fact, we show that if the result holds for $(t-1, u)$ and $(t-1, u-1)$ then it also holds for (t, u).
Base case For the base case suffices to check $t = 0, u > 0$ and $t = u = 1$.
If $t = 0$ and $u > 0$, the result follows from the fact that $1 + H_t = \frac{u-t}{u} = 1$. If $t = u = 1$ we add a random center to C and there are two possibilities for this center: It belongs to the one uncovered cluster($u = 1$) in C_{OPT} or it does not belong to this cluster in C_{OPT} For the first possibility, the probability of choosing this center is $\frac{\phi(X_u)}{\phi}$ and Lemma 6 guarantees that $E[\phi'] \leq \phi(X_c) + 8\phi_{OPT}(X_u)$.
For the second possibility(choosing from a covered cluster), the contribution to the potential is at most $\frac{\phi(X_c)}{\phi} \cdot \phi$. Obviously, $\phi' \leq \phi$ even if we choose a center from a covered cluster, therefore

$E[\phi'] \leq \frac{\phi(X_u)}{\phi}\phi(X_c) + 8\phi_{OPT}(X_u) + \frac{\phi(X_c)}{\phi} \cdot \phi \leq 2\phi(X_c) + 8\phi_{OPT}(X_u)$
Here $1 + H_t = 1 + 1 = 2$ and the proof for the base case is completed.

Inductive step As in the base case, it is convenient here to consider the case that we choose from a covered cluster and the case we choose from an uncovered cluster. In the first case, we choose our first center again with probability $\frac{\phi(X_c)}{\phi}$. In order to apply the inductive hypothesis in this case $(t-1, u)$ we note again that the new center we have added can only decrease ϕ. Therefore, $E[\phi']$ is at most,

$\frac{\phi(X_c)}{\phi}((\phi(X_c) + 8\phi_{OPT}(X_u)(1 + H_{t-1}) + \frac{u-t+1}{u}\phi(X_u))$. **(1)**

For the second case, suppose we choose our first center from some uncovered cluster A with probability $\frac{\phi(A)}{\phi}$. Let p_a denote the probability we pick $a \in A$ as our center and let ϕ_a denote $\phi(A)$ after we choose our center. We must keep in mind that in this case$(t-1, u-1)$ we must add A to the covered clusters and remove it from the uncovered clusters,so
$\phi(X_c) + 8\phi_{OPT}(X_u) = \phi(X_c + A) + 8\phi_{OPT}(X_u - A) = \phi(X_c) + \phi_a + 8\phi_{OPT}(X_u) - 8\phi(A)$.

The contribution to $E[\phi_{OPT}]$ is at most,

$\frac{\phi(A)}{\phi} \cdot \sum_{a \in A} p_a((\phi(X_c) + \phi_a + 8\phi_{OPT}(X_u) - 8\phi(A)) \cdot (1 + H_{t-1}) + \frac{u-t}{u-1}(\varphi(X_u - \phi(A)))$

$\leq \frac{\varphi(A)}{\phi} \cdot ((\phi(X_c) + 8\phi_{OPT}(X_u)) \cdot (1 + H_{t-1}) + \frac{u-t}{u-1}(\phi(X_u) - \phi(A))$

The last step here follows from Lemma 6 since $\sum_{a \in A} p_a \phi_a \leq 8\phi(A)$ The power mean inequality implies that $\sum_{A \subset X_u} \phi(A)^2 \geq \frac{1}{u} \cdot \phi(X_u)^2$.

If we sum over all uncovered clusters A, then $E[\phi']$ is at most,

$\frac{\phi(X_u)}{\phi} \cdot ((\phi(X_c) + 8\phi_{OPT}(X_u)) \cdot (1 + H_{t-1}) + \frac{1}{\phi}\frac{u-t}{u-1}(\phi(X_u)^2 - \frac{1}{u}\phi(X_u)^2)$

$= \frac{\phi(X_u)}{\phi} \cdot ((\phi(X_c) + 8\phi_{OPT}(X_u)) \cdot (1 + H_{t-1}) + \frac{u-t}{u} \cdot \phi(X_u))$.**(2)**

Now adding the two inequalities **(1),(2)** we have

$2E[\phi'] \leq \frac{\phi(X_c)}{\phi}((\phi(X_c) + 8\phi_{OPT}(X_u)(1 + H_{t-1}) + \frac{u-t+1}{u} \cdot \phi(X_u))$

$+\frac{\phi(X_u)}{\phi} \cdot ((\phi(X_c) + 8\phi_{OPT}(X_u)) \cdot (1 + H_{t-1}) + \frac{u-t}{u} \cdot \phi(X_u))$

$= (\phi(X_c) + 8\phi_{OPT}(X_u))(1 + H_{t-1})(\frac{\phi(X_c)}{\phi} + \frac{\phi(X_u)}{\phi})$

$+\frac{\phi(X_c)}{\phi} \cdot \frac{u-t+1}{u} \cdot \phi(X_u) + \frac{\phi(X_u)}{\phi}\frac{u-t}{u} \cdot \phi(X_u)$

$= (\phi(X_c) + 8\phi_{OPT}(X_u))(1 + H_{t-1})$

$+\frac{(\phi(X_c)+\phi(X_u))(u-t)}{u\phi} + \frac{\phi(X_u)^2(u-t)}{u\phi} + \frac{\phi(X_c)}{\phi}\frac{\phi(X_u)}{u}$

$= (\phi(X_c) + 8\phi_{OPT}(X_u))(1 + H_{t-1})$

$+\frac{\phi(X_u)(u-t)}{u}(\frac{\phi(X_c)}{\phi} + \frac{\phi(X_u)}{\phi}) + \frac{\phi(X_c)}{\phi}\frac{\phi(X_u)}{u}$

$= (\phi(X_c) + 8\phi_{OPT}(X_u))(1 + H_{t-1})$

$+\frac{\phi(X_u)(u-t)}{u} + \frac{\phi(X_c)}{\phi}\frac{\phi(X_u)}{u}$

$\leq (\phi(X_c) + 8\phi_{OPT}(X_u))(1 + H_{t-1} + \frac{1}{u}) + \frac{\phi(X_u)(u-t)}{u}$

Therefore $E[\phi'] \leq 2E[\phi']$ which is at most

$(\phi(X_c) + 8\phi_{OPT}(X_u))(1 + H_{t-1} + \frac{1}{u}) + \frac{\phi(X_u)(u-t)}{u}$

Since $\frac{1}{u} \leq \frac{1}{t}$, the inductive step follows.

It is easy now, to prove the main result of this section.

Theorem 4.1 If C is constructed with k-means++ then the corresponding potential function ϕ satisfies $E[\phi] \leq 8(lnk + 2)\phi_{OPT}$.

Proof Consider the clustering after k-means++ has chosen the first center uniformly at random. Let A denote the C_{OPT} cluster in which we chose the first center(since this is the first center, it will definitely come from a cluster in C_{OPT}). Applying Lemma 6 with $t = u = k - 1$ and with A being the only covered cluster, we have,

$E[\phi_{OPT}] \leq (\phi(A) + 8\phi_{OPT} - 8\phi_{OPT}(A))(1 + H_{k-1})$

and the result follows from Lemma 5, and from the fact that $H_{k-1} \leq 1 + lnk$.

This completes the analysis of the performance for k-means++. Arthur and Vassilvitskii showed that their analysis for k-means++ is tight, in fact they showed that D^2 seeding is no better than $2 \log k$-competitive. The proof is in [28].

4.4 A parallel implementation

In the previous section, we saw a very simple and intuitive initialization technique for k-means, which leads to an improved algorithm in terms of running time and efficiency. In fact, we proved that with k-means++ initialization we are guaranteed to achieve a solution which is $\Theta(logk)$ competitive to the optimal solution(k-means can give an arbitrary bad solution). Moreover, the experiments give evidence that the running time of k-means++ is much better than the running time of k-means, since the seeding technique helps k-means to convergence after a few iterations.

However, k-means++ is not apparently parallelizable. Remember that k-means++ chooses a point to be the ith center with probability that depends critically on the realization of the previous $i-1$ centers, since the previous points determine which points are away in the current solution. This inherently sequential nature of k-means++ is a major drawback considering a massive data scenario.

As datasets grow so does the need for faster algorithms or/and for algorithms that can be parallelized. The growth of data forces a growth of the number of classes into which one wishes to partition the data. For example, clustering millions of points into k=100 or k=1000 is typical, but k-means running time of $O(nkd)$(n:points, k:clusters, d:dimensions) will be very slow in this scenario. The fact that the k-means algorithm can be scaled to massive data with a relative easy way due to its simple iterative nature enhances the need to seek for a faster initialization technique.

Bahmani et al. [34] obtained a parallel version of the k-means++ algorithm and empirically demonstrated its practical effectiveness. They used MapReduce[35] for the parallel implementation of the algorithm, since it easy to implement the rest of the algorithm(Lloyd's iterations) using MapReduce. However, the algorithm can be implemented in a variety of parallel computational models, since the requirements are primitive operations that are readily available in any parallel setting.

The main idea of the algorithm, named k-means||, is that instead of sampling a single point in each pass, k-means|| samples $O(k)$ points in each round and repeats the process for approximately $O(logn)$ rounds. At the end of the algorithm we are left with $O(klogn)$ points that form a solution that is a constant factor away from the optimum. These $O(klogn)$ points are then reclustered into k initial centers for the Lloyd's iteration.

Intuition

Think of random initialization and k-means++ initialization as occurring at two ends of a spectrum. The former selects k centers in a single iteration according to uniform distribution. The latter has k iterations and selects one point in each iteration according to a non uniform distribution that is constantly updated after each new center is selected. We want to reap the benefits from the two worlds, that is to generate an algorithm that runs for few iterations, selects more than one point in each iteration and has provable approximation guarantees. k-means|| finds the sweet spot on the spectrum by carefully defining the number of iterations and the non-uniform distribution itself.

k-means|| algorithm

The main difference between k-means++ and k-means|| is that the latter uses an over-sampling factor $b = \Omega(k)$. The algorithm picks an initial center uniformly at random and computes ψ, the initial cost of the clustering after this selection. It then proceeds in $\log \psi$ iterations, where in each iteration, given the current set C of centers it samples each x with probability $\frac{bd^2(x,C)}{\phi_X(C)}$ where $d(x, C) = \min_{x \in X} \|x - y\|$ and $\phi_X(C) = \sum_{x \in X} d^2(x, C)$. The sampled points are then added to C the quantity $\phi_X(C)$ updated, and the iteration continued. The expected number of points chosen in each iteration is b and at the end, the expected number of points in C is $b \log \psi$ which is typically more than k. To reduce the number of centers, k-means|| assigns weights to the points in C and reclusters these weighted points to obtain k-centers. We now present k-means|| algorithm :

Algorithm 3: k-means||

1: $C \leftarrow$ sample a point uniformly at random from X
2: $\psi \leftarrow \phi_X(C)$
3: **for** $O(\log \psi)$ times **do**
4: $\quad C' \leftarrow$ sample each point $x \in X$ independently with probability
5: $\quad p_x = \frac{bd^2(x,C)}{\varphi_x(C)}$
6: $\quad C \leftarrow$C$\cup$C'
7: **end for**
8: For $x \in C$, set w_x to be the number of points in X closer to x than any other point in C
9: Recluster the weighted points in C into k clusters

Notice that the size of C is significantly smaller than the input size; the reclustering can therefore be done quickly. With MapReduce, since the number of centers is small they can be all assigned to a single machine and any provable approximation algorithm(such as k-means++) can be used to cluster points to obtain k-centers.

In the next lines we will briefly discuss the basic features of the MapReduce model and we will se how k-means can be implemented in the MapReduce model.

MapReduce model of computation

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

We could think of MapReduce as a 5-step parallel and distributed computation:

1. **Prepare the Map() input.** The "MapReduce system" designates Map processors, assigns the K1 input key value each processor would work on, and provides that processor with all the input data associated with that key value.
2. **Run the user-provided Map() code.** Map() is run exactly once for each K1 key value, generating output organized by key values K2.
3. **"Shuffle" the Map output to the Reduce processors.** The MapReduce system designates Reduce processors, assigns the K2 key value each processor should work on, and provides that processor with all the Map-generated data associated with that key value.
4. **Run the user-provided Reduce() code.** Reduce() is run exactly once for each K2 key value produced by the Map step.
5. **Produce the final output.** The MapReduce system collects all the Reduce output, and sorts it by K2 to produce the final outcome.

As we have mentioned earlier, the k-means algorithm can easily be implemented in a parallel model of computation like MapReduce due to its simple iterative nature. Hence we can focus only focus on Steps 1-7 of the k-means|| algorithm.

Step 2 We assume that the set of centers is small enough to be held in memory or be distributed among all the mappers. Given such a set C of centers, we can compute $\phi_X(C)$ easily: Each mapper working on an input partition $X' \subseteq X$ computes $\phi_{X'}(C)$ and all mappers send their values to the reducer. Then the reducer simply adds all these values and obtains $\phi_{X'}(C)$

Steps 3-7 We have updated the value $\phi_X(C)$ needed for the iterations in these steps from the previous step. We must take care of steps 4 and 7. Step 4 is simple in MapReduce because each mapper can sample independently. The same argument holds for step 4; each mapper can compute w_x independently.

The assumption we made in Step 2 (small set of centers) is not necessary in MapReduce. Each mapper holds $X' \subseteq X$ and $C' \subseteq C$ can output the tuple $\langle x; \arg\min_{c \in C'} d(x,c) \rangle$, where $x \in X'$ is the key. From this, the reducer can easily compute $d(x, C)$ and hence $\phi_X(C)$.

Chapter 5

k-means problem online and streaming

5.1 Streaming algorithms

5.1.1 Introduction

All the algorithms we have seen so far for the k-means problem assumed that the entire input was available for processing in any form the algorithm designer needed. In this chapter, we focus on algorithms for the k-means problem in the online and streaming setting.

As commercial, social, and scientific data sources continue to grow to an unprecedented rate, it is important that algorithms to process and analyze this data operate in online, or one-pass streaming settings. The goal is to design light-weight algorithms that make only one pass over the data.

We consider the k-means problem in the situation where the data is too large to be stored in main memory and must be accessed sequentially, such as from a disk, and where we must use as little memory as possible. Our algorithms must store very few points at any given time.

Though streaming algorithms had already been studied by Munro and Paterson as well as Flajolet and Martin, the field of streaming algorithms was first formalized and popularized in a paper by Noga Alon, Yossi Matias, and Mario Szegedy.[37] For this paper, the authors later won the Gödel Prize in 2005 "for their foundational contribution to streaming algorithms." There has since been a large body of work centered around data streaming algorithms that spans a diverse spectrum of computer science fields such as theory, databases, networking, and natural language processing.

For these situations, we consider the problem of Euclidean k-means in the streaming model. Points in Euclidean space are read sequentially; when the data stream finishes, we must select k of these to designate as facilities. The cost of the objective function is again the sum of squared distances from each point in the stream to its nearest facility.

5.1.2 Streaming k-means on Well-Clusterable Data

In recent years, there have been series of results that produced PTASs for k-means clustering in the streaming model. That is, they produced $(1+\varepsilon)$-approximation for $\varepsilon > 0$. Since optimizing the k-means objective is Max-SNP hard, all these algorithms must take time exponential in k unless $P = NP$.

In this subsection, we discuss an algorithm that tries to address this problem. That is, an algorithm which has truly polynomial runtime. As we argued in the previous paragraph, one cannot hope to produce a near optimal polynomial algorithm for the k-means problem with no further assumptions on the input.

Recall that in the previous chapter we have seen the proof of the k-means++ $O(\log k)$ competitiveness. We divided the proof into two parts and in the first part we have proved that if the clusters are well separated and we always pick a center from a new optimal cluster then k-means++ is 8-competitive. We have also mentioned that Ostrovsky et al. gave an algorithm essentially identical to k-means++.

Ostrovsky et al. proved that their algorithm is constant competitive, assuming that the input satisfies a data separability condition: the assumption closely reflects how k-means is used in practice and allowed the authors to create a high-quality approximation for k-means clustering in the non-streaming setting with polynomial running time even for large values of k. Their work left open a natural and important question: are similar results possible in the streaming setting?

The algorithm we present here is due to Braverman et al.[38] and achieves a near-optimal approximation algorithm for k-means in high-dimensional Euclidean space with sublinear memory and a single pass, under the same data separability assumption. The algorithm also offers significant improvements in both space and running time over previous algorithms while yielding asymptotically best-possible performance(assuming that the running time must be fully polynomial and $P \neq NP$).

Data separability condition

The data separability condition captures the notion of "meaningful" k-clustering. In order to understand this notion we will give two examples of settings where one would intuitively not consider the data to possess a meaningful k-clustering.

The first example considers cases where nearly optimum cost can be achieved by two very different k-way partitions of the data. In this case, the identity of the optimal partition carries little meaning (for example, if the data was generated by random sampling from a source, then the optimal cluster regions might shift drastically upon resampling).

Alternatively, if a near-optimal k-clustering can be achieved by a partition into fewer than k clusters, then that smaller value of k should be used to cluster the data. If near-optimal k-clusterings are hard to find only when they provide ambiguous classification or marginal benefit (i.e., in the absence of a meaningful k-clustering), then such hardness should not be viewed as an acceptable obstacle to algorithm development. Instead, the performance criteria should be revised.

We will say that the data is well-suited for clustering when $\frac{\phi_{OPT,k}}{\phi_{OPT,k-1}} \leq \sigma^2$, where $\phi_{OPT,k}$ denotes the optimal potential for a k-clustering. With simple words, this condition says that the input to k-means is σ-separable if reducing the number of facilities from k to $k-1$ would increase the cost of the optimum solution by $\frac{1}{\sigma^2}$.

High level Ideas

Before we proceed to the algorithmic steps, we give a high level description of the algorithm. This will make both the understanding of the algorithm and the analysis easier, since the algorithm involves many steps and some of them are complicated. The algorithm is a fully polynomial time streaming algorithm for k-service clustering and can be applied to k-median as well.

A high level description of the algorithm is the following:

1. "Guess" the cost of optimum
2. Run the online facility location algorithm of Meyerson until either the total cost of the solution exceeds a constant time the guess or the total facilities exceed some computed value κ
3. Declare end of phase, increase the guess,consolidate the facilities via matching and continue with the next point
4. When stream is exhausted consolidate facilities down to k

High-probability bounds on the performance of online facility location are produced, showing a constant competitive algorithm with probability $1 - \frac{1}{n}$. Combining this result with the algorithm of Charikar et al.[40] improves the memory bound and processing time per point by a $\Theta(\log n)$ factor, yielding the first streaming constant-approximation for k-median to store only $O(k \log n)$ points in memory. Since the analysis extends to cases where triangle inequality approximately holds, we obtain also a streaming constant-approximation for k-means to store only $O(k \log n)$ points in memory.

The end of phase declaration in Step 3 requires "success" of a randomized algorithm at a particular critical phase; prior phases are guaranteed to have bounded cost. The performance and success probability do not depend on the number of phases. A better than linear bound on the number of phases is obtained by simply requiring each phase to read in at least a logarithmic number of new points.

Algorithm 4 takes as input a data stream of n points and a value k for the desired means. The algorithm is defined in terms of constants β, γ. Precise values for these constants will be given later in the analysis. Step 22 requires a non streaming $O(1)$ approximation algorithm as a subroutine. One candidate is that of Kanungo et al.[26]

Placing points at the front of the data stream is easy to implement with a stack structure. When placing an item at the front of the stream, push it to the stack. When reading from the stream, check first if the stack is empty: if it is not, read by popping from the stack. If the stack is empty, read from the stream as normal. This also allows us to place items with weight on the stream, and we consider each item from the stream to be of weight one.

The algorithm uses implicitly the online facility location algorithm of Adam Meyerson [39]. The algorithm works as follows. We are given a facility cost f. As each point arrives, we measure the service cost δ for assigning that point to the nearest existing facility. With probability $\min\{\frac{\delta}{f}, 1\}$ we create a new facility at the arriving point. Otherwise, we assign the point to the nearest existing facility and pay the service cost δ. In the last subsection of this chapter we discuss the facility location problem and the online facility location algorithm of Meyerson. The reason is that facility location problem is strongly connected with k-median and k-means ,and lots of techniques have been adapted from facility location problem to k-median and k-means problem.

Now we show how the general method described in Algorithm 4 produces a constant approximation for the general class of semi-clustering problems; problems that satisfy the α-approximate triangle inequality. Note that k-median satisfy the triangle inequality and k-means the 2-approximate triangle inequality:

α-APPROXIMATE TRIANGLE INEQUALITY If, for any points a,b,c the following applies: $\alpha[\delta(\text{a,b})+\delta(\text{a,b})]\geq \delta(\text{a,c})$, then we say that α-approximate triangle inequality is satisfied.

In order to prove the constant approximation we will use the following theorem, which produces high probability bounds on the performance of online facility location, showing that the algorithm achieves within constants of its expected behavior with probability $1 - \frac{1}{n}$.

Theorem 5.1 Suppose we run the online facility location of [39] with $f = \frac{L}{k(1+\log n}$ where $L \leq OPT$ and that the service costs satisfy the α-approximate triangle inequality. Then the expected cost is at most $(3\alpha + 1)OPT$ and the expected number of facilities generated by the algorithm is at most $(3\alpha + 1)k(1 + \log n)\frac{OPT}{L}$. Further, with probability at least $1 - \frac{1}{n}$ the service cost is at most $(3\alpha + \frac{2e}{e-1})OPT$ and the number of facilities generated is at most $(6\alpha + 1)k(1 + \log n)\frac{OPT}{L}$.

The main new idea in the proof of theorem 5.1 involves producing a high probability bound on the service cost paid prior to opening facilities in each region. Induction is used to upper bound the actual probability of paying at least a given cost prior to opening the facilities; by setting the target probability appropriately, the chance of exceeding the expected cost by more

Algorithm 4: One pass, constant approximation k-service clustering algorithm

$L_1 \leftarrow 1$
$i \leftarrow 1$
while solution not found **do**
  $K \leftarrow 0$
  $cost \leftarrow 0$
  $f \leftarrow L_i/(k(1+\log n))$
  while there are points still in the stream **do**
    $x \leftarrow$ next point from stream
    $y \leftarrow$ facility in K that minimizes $\delta(x,y)$
    if probability $\min\{\frac{weight(x)\cdot\delta(x,y)}{f}, 1\}$ **then**
      $K \leftarrow K \cup \{x\}$
    else
      $cost \leftarrow cost + weight(x)\cdot\delta(x,y)$
      $weight(y) \leftarrow weight(y) + weight(x)$
    end if
    if $cost > \gamma L_i or |K| > (\gamma-1)(1+\log n)k$ **then**
      break and raise flag
    end if
  end while
  if flag raised **then**
    push facilities in K onto stream
    $L_{i+1} \leftarrow \beta L_i$
    $i \leftarrow i+1$
  else
    Cluster K to yield exactly k facilities
    Declare solution found
  end if
end while

than a constant is exponentially small in the number of regions. Applying Chernoff bounds for the number of facilities completes the proof, which can be found in [38].

We now analyze Algorithm 4 with help of theorem 5.1. Let $\beta = 2\alpha^2 c_{OFL} + 2\alpha$ and $\gamma = \max\{4\alpha^3 c_{OFL}^2, \beta k_{OFL} + 1\}$, where c_{OFL} is the constant factor on the service cost obtained from online facility location with high probability from theorem 5.1 and k_{OFL} be such that online facility location guarantees to generate at most $k_{OFL}k(1 + \log n)\frac{OPT}{L}$ facilities. We assume that $c_{OFL} \geq 2\alpha$ and we can always replace c_{OFL} with a larger value since it is a worst case guarantee.

Define a phase in Algorithm 4 to be a single iteration of the outermost loop. We try reading as many points as we can until either the service cost grows too high or we have too many facilities (line 15). Then the lower bound on OPT (L_i) is too small, so we increase it by a factor β. In a phase, we pay at most $f = L_i/k(1+\log n)$ for a weighted point and there are at most $(\gamma - 1)k(1 + \log n)$ weighted points from the previous phase. Our service cost is at most $(\gamma - 1)L_i$ for these points, so they are successfully clustered. Thus at the start of each phase the stream looks like some weighted points from the previous phase followed by unread points.

Lemma 5.1 Let X' be any subset of points in the stream at the start of phase i. Then the total service cost of the optimum k-service clustering of X' is at most $\alpha \cdot OPT + \gamma\frac{(}{a^2}\beta - \alpha)L_i$

Proof Consider an original point $x \in X$. We say that $y \in K$ represents x in phase l if y is the assigned facility for x or for $x's$ phase $l-1$ representative. The weight of $y \in K$ is the number of points it represents. Moreover, once a point x becomes represented in phase l, it is represented for all future phases.

We examine the cost of the weighted points and the unread points at the start of phase i when using the optimum facilities to serve all of these points. Fix a point $x \in X$ and let us bound the cost due to this point. Let $y_j, ...y_{i-1}$ be x's respective representatives in phases j up through $i-1$. Then the service cost due to x will be $\delta(y_{i-1}, y*)$ where $y*$ is the cheapest optimum facility for y_{i-1}. By α-approximate triangle inequality

$$\delta(y_{i-1}, y*) \leq \alpha\delta(x, y*) + \alpha\delta(x, y_{i-1})$$

$$\leq \alpha\delta(x, y*) + \sum_{l=2}^{i-j} \alpha^L \delta(y_{i-l}, y_{i-l+1}) + \alpha^{i-j}\delta(x, y_j)$$

Thus, summing over all points x in or represented by points in X', and noting that our service cost in phase l is bounded by $\gamma L_l \leq \gamma L_i \frac{1}{\beta^{i-l}}$, gives a total service cost of at most

$$\alpha \cdot OPT + \gamma\alpha L_i \sum_{l=1}^{i-1} (\frac{\alpha}{\beta})^{i-l}$$

$$= \alpha \cdot OPT + \gamma(\frac{\alpha^2}{\beta - \alpha})L_i$$

We have proven that there exists a low cost clustering for the points at each phase, provided we can guarantee that $L_i \leq OPT$. The next step is to show that we terminate at or

before the critical phase(last phase where $L_i \leq OPT$) with high probability.

Lemma 5.2 With probability at least the success probability of online facility location from theorem 5.1, Algorithm 4 terminates before the critical phase.

Proof. Let i be the critical phase, and let OPT_i be the optimum cost of clustering all the points (weighted or not) seen on the stream at the start of phase i. By lemma 5.1 and the fact that $OPT \leq \beta L_i$, we have

$$OPT_i \leq \alpha \cdot OPT + \gamma(\tfrac{\alpha^2}{\beta-\alpha})L_i$$

$$\leq (\alpha\beta + \gamma\tfrac{\alpha^2}{\beta-\alpha})L_i$$

Theorem 5.1 guarantees that online facility location algorithm yields a solution with at most $\beta k_{OFL}(1+\log n)k$ facilities and of cost at most $c_{OFL}OPT_i$ with high probability. Our definitions for β, γ guarantee that $c_{OFL}OPT_i \leq \gamma L_i$. Furthermore $(\gamma-1)k(1+\log n) \geq \beta k_{OFL}k(1+\log n)$ from definition of γ, so if online facility location "succeeds", the critical phase will allow the online facility location algorithm to run in completion.

Corollary 5.1 With high probability (same as that for online facility location), Algorithm 4 completes the final phase with a solution of cost at most $\frac{\alpha\beta\gamma}{\beta-\alpha} \cdot OPT$. Applying the values for the constants gives an approximation factor of $4\alpha^4 c_{OFL}^2 + 4\alpha^3 c_{OFL}$ provided $4\alpha^3 c_{OFL}^2 + 2\alpha^2 c_{OFL} \geq \beta k_{OFL} + 1$.

Proof. Consider a point $x \in X$ and let $y_j, y_{j+1}, ..., y_{i-1}, y_i$ be x's respective representatives in phases j up through i. The service cost $\delta(x, y_i)$ due to x is at most

$$\alpha^{i-j}\delta(x, y_j) + \sum_{l=1}^{i-j} \alpha^l \delta(y_{i-l}, y_{i-l+1})$$

Summing over all points x, and noting that our service cost in phase l is bounded by γL_l, combined with the knowledge that with high probability, we terminate at a phase where $L_i \leq OPT$ gives a total service cost of at most:

$\alpha\gamma L_i + \alpha^2\gamma L_{i-1} + \alpha^3\gamma L_{i-2} + ... + \alpha^i\gamma L_1$

$\alpha\gamma L_i \sum_{l=0}^{i-1} (\frac{\alpha}{\beta})^l$

$\frac{\alpha\beta\gamma}{\beta-\alpha} \cdot OPT$

Although we have bounded the total service cost, this solution has much more than k facilities. Algorithm 4 takes care of this in line 22, where we apply a a non streaming constant approximation algorithm with ratio c_{KS} to produce an overall approximation of $(\alpha + 4\alpha^5 c_{OFL}^2 + 4\alpha^4 c_{OFL})c_{KS}$.

Theorem 5.2 With high probability, our algorithm achieves a constant approximation to k-service clustering if α-approximate triangle inequality holds for fixed constant α. This uses exactly k facilities and stores $O(k \log n)$ points in memory.

Now we show how to modify Algorithm 4 to improve its running time. The reason is that the algorithm, as presented, has $O(\log_\beta OPT)$ phases in expectation. We take care of this by setting appropriate bounds and balancing facility with service cost. After these modifications the following theorem holds.

Theorem 5.3 For any fixed α, Algorithm 4 can be modified to run in $O(nk \log n)$-time.

Proof. Consider any phase. The end of phase is depends on the service cost and on the number of facilities. Regarding the service cost, the phase starts by reading the weighted facilities from the previous phase and paying a cost of at most $f = \frac{L_i}{k(1+logn)}$ for each, at the end of which the cost is at most $(\gamma - 1)L_i$. Each additional point gives us a service cost of at most $\frac{L_i}{1+k \log n}$,so the phase must read at least $k(1 + \log n)$ additional unread points before it can terminate due to cost exceeding γL_i.

Now suppose that the phase ends due to having too many facilities without reading at least $k(1+\log n)$ additional points. Since each new point can create at most one facility, the previous phase must have had at least $(\gamma-2)k(1+\log n)$ facilities already. Consider an optimal k-service clustering over the set X' of all the weighted points during this phase. Let OPT' denote the total service cost of this solution and OPT'_r denote the optimum total service cost if we are instead restricted to only selecting points from X'. Note that by α-approximate triangle inequality, we have $OPT'_r \leq 2\alpha OPT'$. Thus, by lemma 5.1 we have that $OPT'_r \leq 2\alpha(\alpha + \gamma\frac{\alpha^2}{\beta-\alpha}OPT)$.

Since OPT'_r is only allowed k facilities, it must pay non-zero service cost for at least $(\gamma - 3)(1 + logn)k$ weighted points. Define the nearest neighbor function $\pi : X' \to X'$ where for each point $x \in X'$, $\pi(x)$ denotes closest other point in terms of service costs in X'. Then note that $\Delta_x = weight(x) \cdot \delta(x, \pi(x))$ gives a lower bound on the service cost for x if it is not chosen as a facility. Thus, the sum η of all but the k highest Δ gives a lower bound on OPT'_r. It follows that $\frac{\eta}{2\alpha(\alpha+\gamma\frac{\alpha^2}{\beta-\alpha})} \leq OPT$. We will set L_i to the maximum of this new lower bound and

βL_{i-1}, eliminate $k(1+logn)$ facilities and increase service cost to at most L_i. This guarantees that the next time the number of facilities grows too large we will have read $(klogn)$ new points, bounding the number of phases by $O(\frac{n}{k \log n})$.

Let $\hat{X} \subseteq X'$ denote the set of points with $|\Delta_x \leq \eta[2\alpha(\alpha + \gamma\frac{\alpha^2}{\beta-\alpha})(1 + \log n)k]^{-1}$. Suppose that $|\hat{X}| < 2k(1+\log n)$. The number of points which contribute to η is at least $(\gamma-3)k(1+\log n)$, and at least $(\gamma-5)k(1+\log n)$ of these do not belong in $\hat{X}$. Thus the sum of Δ_x for such points is bounded by $\frac{\eta(\gamma-5)}{2\alpha(\alpha+\gamma\frac{\alpha^2}{\beta-\alpha})} \leq \eta$. Solving this inequality for γ yields $\gamma \leq \frac{2\alpha^2+5}{1-\frac{2\alpha^3}{\beta-\alpha}}$.

Plugging in the values for β and γ along with $c_{OFL} \geq 2\alpha$ and $\alpha \geq 1$ yields a contradiction. Thus $|\hat{X}| \geq 2k(1 + \log n)$. We assume that $\hat{X}$ is even to simplify the analysis. Some points in $\hat{X}$ have their nearest neighbor in $\hat{X} - X'$. For the remaining points in $\hat{X}$, consider the nearest neighbor graph induced by these points. This graph has no cycles of length 3 or longer. Thus, the graph is bipartite and we can find a vertex cover C of size at most $|\hat{X}|/2$. We can add

additional points to C from $\hat{X}$ to get precisely $|\hat{X}/2$ points. Note that all points in $\hat{X} - C$ have a nearest neighbor not in $\hat{X} - c$. Thus, we can remove $\hat{X} - C$ as facilities and increase our service cost by at most

$$\frac{\eta(1+\log n)k}{2\alpha(\alpha+\gamma\frac{\alpha^2}{\beta-\gamma})(1+\log n)k} = \frac{\eta}{2\alpha(\alpha-\gamma\frac{\alpha^2}{\beta-\alpha})} \leq L_i$$

We can compute η in time $O(k^2 \log^2 n)$,$\hat{X}$ in time $O(k \log n)$, and the vertex cover in time linear in $|\hat{X}|$ (using a greedy algorithm; note that it needn't be a minimum vertex cover). Additionally, all these can be computed using space to store $O(k \log n)$ points. The running time for reading a new (unweighted) point is $O(k \log n)$, so the total running time is the time to read unweighted points plus the overhead induced by starting new phases (and reading weighted points). Each of these is at most $O(nk \log n)$.

Summing up, we have produced a streaming constant-approximation algorithm for k-median and k-means problem that has a total running time of $O(nk \log n)$ and stores only $O(k \log n)$ points in memory.

For a σ-separable dataset we can improve the constant approximation algorithm to an $1+\sigma^2$ approximation algorithm for streaming k-means. Braverman et al. showed how to achieve this by applying a single recentering step, called ball k-means step.

The ball k-means step involves selecting the points which are much closer to one of our facilities than to any other (the "ball" of that facility) and computing the center of mass on those points. The idea is that the optimum facilities for such an instance must be far apart; any constant-approximation must include a facility close to each of the optimum ones. Combining these facts gives a one-to-one mapping between facilities and optimums, and they showed that the points which are very close to each of our facilities must therefore belong to distinct optimum ones.

The difficulty here is that we wish to compute the entire solution with only one pass through the data and full ball k-means step requires another pass through the data. Braverman et al. overcame this barrier by showing that it is sufficient to use a smaller random sample of points. They proved that sampling works well for computing center of mass. A random sample of constant size, which is independent of the size of the cluster, provides a constant approximation. Therefore, they produced a suitable random sample of the points belonging to each of the "balls" for the final ball k-means step.

Another difficulty here is that we do not know what the final cluster centers will be until the termination of the stream, making it difficult to sample uniformly from the balls. Instead, they show that the clusters from the solution are formed by adding points one at a time to clusters and by merging existing clusters together. This process permits to maintain at all times a random sample of the points belonging to each of the clusters. Of course, randomly sampling from the points in these clusters is not the same as randomly sampling from the balls in the ball k-means step. However, they then show that the set we are actually sampling from

and the set they are sampling from (the points which are much closer to this particular facility than any other one of our facilities) are roughly (within constants) the same set of points, and that as the separability value σ approaches zero, these sets of points converge and become effectively identical. Random sampling and ball k-means steps proofs can be found in [38].

Putting it all together, the overall result maintains a sample of size $\frac{1}{\varepsilon}$ from each of the clusters at all times. The number of clusters will never exceed $O(k \log n)$, so the total memory requirement is $O(\frac{k}{\varepsilon} \log n)$ points for a chosen constant ε. The approximation factor for the final solution is $1 + O(\varepsilon) + O(\sigma^2)$ for σ-separable data, and the overall running time is $O(nk \log n)$.

5.1.3 Fast and Accurate k-means for Large Datasets

As we have seen in the previous subsection, Algorithm 4 is a very efficient streaming approximation algorithm for k-service clustering from a theoretical standpoint. However, it's performance is not so good for practical purposes. The main issue is that the constants hidden in the asymptotic notation are quite large. The approximation factor is in hundreds, and the $O(klogn)$ memory requirement has so large constants that there are actually more than n facilities for many of the data sets analyzed. Furthermore, these constants are encoded into the algorithm itself, making it difficult to argue that the performance should improve for non worst case inputs.

The algorithm we present in this subsection is a simplification of Algorithm 4, both in design and analysis, and eliminates the large constant factors in the approximation guarantee, the memory requirements and the running time. It improves the manner by which the algorithm determines better facility cost as the stream is processed, removing unnecessary checks and allowing the user to parametrize what remains. The end-of-phase condition based on the total cost is removed, ending phases only when the number of facilities exceeds a number $\kappa \in \Omega(k \log n)$. In addition, the end transition between phases is simplified.

Algorithm 5: Fast streaming k-means (data stream, k, κ, β)

```
Initialize f = 1/(k(1 + log n)) and an empty set K
while some portion of the stream remains unread do
  while |K| ≤ κ and some portion of the stream is unread do
    Read the next point x from the stream
    Measure δ = min_{y∈K} d(x, y)^2
    if probability δ/f event occurs then
      set K ← K ∪ {x}
    else
      assign x to its closest facility in K
    end if
  end while
  if stream not exhausted then
    while |K| > κ do
      Set f ← βf
      Move each x ∈ K to the center-of-mass of its points
      Let w_x be the number of points assigned to x ∈ K
      Initialize K̂ containing the first facility from K
      for each x ∈ K do
        Measure δ = min_{y∈K̂} d(x, y)^2
        if probability w_x δ/f event occurs then
          set K̂ ← K̂ ∪ {x}
        else
          assign x to its closest facility in K̂
        end if
      end for
      Set K ← K̂
    end while
  else
    Run the batch k-means algorithm on weighted points K
    Perform ball k-means on the resulting set of clusters
  end if
end while
```

5.2 The facility location problem

In this section we discuss the famous facility location problem. The facility location problem is a branch of operations research and computational geometry concerned with the optimal placement of facilities to minimize transportation costs while considering factors like avoiding placing hazardous materials near housing and competitors' facilities. The reasons that we discuss this problem are that it is strongly related with k-median and k-means problems and that techniques used in facility location can be applied to cluster analysis.

The simplest and most popular variant of Facility Location is the metric uncapacitated Facility Location problem.

Metric Uncapacitated Facility location

Input: A metric space, a set of facility locations with an opening cost for each potential facility location, and a set of clients locations.

Output: A (sub)set of the facility locations

Goal: Minimize the opening cost for all facilities plus the assignment cost for all clients, where a client's assignment cost is the distance of its location to its nearest facility.

There are many other variants for facility location. Some of them are:

- the space is non metric
- facility costs are uniform
- facilities have capacities

The non-metric facility location can be approximated to within a factor $O(\log n)$[41]. The factor of $O(\log n)$ is tight. This can be shown via an approximation preserving reduction from the set cover problem. The idea of the approximation preserving reduction is that we reduce a problem P to a problem P' so that if an c-approximation algorithm exists for P', then we can get an $f(c)$-approximation algorithm for P, where f is some function. Then if we know that P is hard to approximate within some factor, the reduction implies that P' is also hard to approximate within some factor.

The metric UFL problem is still NP-hard and hard to approximate within factor better than 1.463. The current best approximation algorithm is due to Li [42] . He combined an algorithm by Byrka [43] and an algorithm by Jain et al.[44] to achieve an approximation guarantee of 1.488, which is close to being best possible.

5.2.1 Relation with k-median and k-means

The k-median problem definition in the general case is the following :

k-median problem

Input: A set F of potential facility locations, a set C of clients, a distance metric d over $F \cup C$ and an integer k

Output: A subset $S \subseteq F$

Goal: Minimize $cost(S) = \sum_{j \in C} d(j, C)$, where $d(j, C)$ denotes the distance from j to its nearest facility in C

When $F = C = X$, solution S partitions the set of points into what is known as clusters and thus the objective measures how well X can be partitioned into k clusters. Note here that the difficulty of the k-median problem lies in the hard constraint that only k facilities are allowed to be opened. Without this constraint we could simply open all facilities.

The UFL problem has similar input as k-median but instead of giving an upper bound k on the number of facilities we can open, it specifies an opening cost f_i for each facility $i \in F$. The goal is to open a set of facilities S that minimizes the sum of the opening costs and assignment costs, i.e. $\sum_{i \in S} f_i + cost(S)$.

The connection between UFL and k-median is motivated by basic economic theory: if we let the opening costs of facilities be small then a "good" solution to UFL will open many facilities whereas if we let the opening costs of facilities be large then a good solution will only open few facilities. By appropriately selecting the cost of facilities, one can therefore expect that an algorithm for UFL opens close to k facilities and therefore almost also gives a solution to the k-median problem. This the intuition for the concept of the bi-point solutions, which Jain and Vazirani first exploited to obtain a 6-approximation algorithm for k-median using their 3-approximation primal-dual algorithm for UFL [45]. The factor 3 was later improved by Jain et al.[44] to 2 resulting in a 4-approximation algorithm for k-median.

The current best approximation for k-median also exploited the concept of bi-point solutions (from UFL) and using the novel approach of pseudo-approximation, produced an approximation for k-median with ratio $1 + \sqrt{3} + \varepsilon$.

As long as UFL is related with the k-median, it is also related with k-means. The reason is that means and medians only differ in the metric; the way distances are measured. Furthermore, we have discussed a streaming algorithm in the previous section that makes use of the online facility location algorithm of Meyerson [39] and produces a FPTAS for k-means.

In the next subsection, we discuss the online facility location algorithm of Meyerson. It is a simple and intuitive algorithm which has been the inspiration for the k-means++ algorithm of Arthur and Vassilvitskii.

5.2.2 Online Facility Location

The online facility location problem we discuss here is the online variant of the metric uncapacitated facility location (UFL) with the difference that clients are not restricted to open a facility in particular facility locations, but they can open as many facilities they like at any location. The clients arrive one-by-one and must be assigned to an open facility upon arrival, without any knowledge about future demands. The decisions of opening a facility at a particular

location and of assigning a client to some facility are irrevocable. In this subsection, we present the randomized $O(1)$-competitive algorithm of Meyerson [39] in the case where points arrive in random order.

Online facility location problems arise in a variety of telecommunication, networking, and mobile computing applications. For example, suppose we are asked to construct a network. We need to purchase various servers (facilities) and connect each client to one of the servers. The cost to connect a client to a server is linear in the distance between them. Once the network has been constructed, additional clients may need to be added. In this case we must purchase additional cables (assign the clients upon arrival) and possibly new servers in order to accommodate the increase in demand. We would like to minimize the total cost.

We assume that the adversary has limited power. He designs the metric space and the set of demand points (clients), then randomly permutes these points. For adversaries of this kind, we give a randomized $O(1)$-competitive algorithm, even in the case of nonuniform facility costs.

In the case of uniform facility costs the algorithm is straightforward. When a new demand point arrives, we measure the distance from this demand to the closest already-open facility. Suppose this distance is δ. With probability $\frac{\delta}{f}$, where f is the facility cost, we will open a new facility at this point. Otherwise, the demand is assigned to the closest open facility.

The important property of the algorithm is that we can bound the facility cost and the assignment cost due to any demand by δ. That is, if the cost for opening a facility is greater than δ then the demand will be assigned to the already open nearest facility in expectation, paying δ.

We will divide the demand points into good points and bad points to make the analysis easier. Intuitively, bad points are points that are far away from the center of an optimum cluster, in the case where a facility near the optimum cluster is already open. We use the term bad, because these points are more likely to open another facility within the cluster.

Suppose an optimum solution opens k facilities $c_1, ..., c_k$ and call cluster C_i the points that the optimum sends to center c_i. Define $A_i = \sum_{p \in C_i} d_p$, where d_p is the distance from point p to the nearest open facility and $a_i = A_i/|C_i|$. Let γ_p be the cost paid when point p arrives and consider the closest half of the points in C_i to be good and the other half to be bad. Then, the following lemma holds regardless of the order in which the demand points arrive.

Lemma 5.1 The total expected cost of good points $g \in C_i$ is bounded by
$E[\sum_g \gamma_g] \leq 2f + 2A_i + 2\sum_g d_g$

Proof Suppose there exists an open facility which is within $2a_i$ of c_i. By triangle inequality, any point g is within $2a_i + d_g$ of the nearest open center. The total expected cost due to g is the expected facility cost due to g plus the expected assignment cost due to g which are both bounded by $2a_i + d_g$. So $E[\gamma_g] \leq 2(2a_i + d_g)$ and $E[\sum_g \gamma_g] \leq 2A_i + 2d_g$. Now, suppose that no such nearby facility exists. Each good point which arrives has a chance of opening a new facility

and with help of Markov's inequality we see that all good points are within $2a_i$ of c_i. Using a simple potential function argument (or expected waiting time techniques), one can show that the expected assignment cost of good points before opening a facility is bounded by f. Opening a facility also costs f, so the total expected cost of good points is the expected cost before a nearby facility opens ($2f$) plus the cost afterwards ($2A_i + 2d_g$).

We must use a different technique to bound the cost of the bad points. The reason is that if a bad point opens a facility, this facility need not be near the center of the optimum cluster. This means, that points that arrive later may have increased assignment cost.

The key property here is that the above lemma holds regardless of the order that good points arrive. This allows us to assume that many good points have already arrived and then a bad point arrives. The good points have probably opened a facility near the optimum center, so we can bound the cost due to the bad point in terms of the good points. Formally:

Lemma 5.3 For any bad point b of cluster C_i, $E[\gamma_b] \leq 2d_b + \frac{2}{C_i}(f + \sum_g (E[\gamma_g] + 2d_g))$.

Proof Let g be the good point that arrived just before bad point b with probability $\frac{2}{C_i}$. Let x be the distance from optimum center c_i to the nearest open facility, then $E[\gamma_b] \leq 2(x+d_b)$, since the distance from b to the facility is at most $(x + d_b)$ by triangle inequality. Now, $E[\gamma_g] \geq 2(x - d_g)$, since the nearest open center was at least distance x from the optimum center when g arrived. So, $E[\gamma_b] \leq E[\gamma_g] + 2d_b + 2d_g$. To conclude the proof, we observe that there is also a probability of $\frac{2}{C_i}$ that point b arrives before all good points, in which case we pay a cost of at most f for point b.

Theorem 5.4 The algorithm is constant competitive

Proof The proof of the competitiveness is immediate from the previous lemmas. From lemma 5.3 we have that $E[\sum_b \gamma_b] \leq f + \sum_g (E[\gamma_g] + 2d_g) + 2\sum_b d_b$. The total expected cost for a cluster i is the sum of the costs incurred by good and bad points, which is bounded by $f + 2E[\gamma_g] + 2(\sum_g d_g + \sum_b d_b)$. Recalling that the last quantity in the sum is exactly A_i for cluster i and half of the points closest to the optimum cluster are good we have that
$f + 2E[\gamma_g] + 2(\sum_g d_g + \sum_b d_b) \leq f + 2A_i + 2E[\sum_g \gamma_g] \leq 5f + 6A_i + 4\sum_g d_g \leq 5f + 8A_i$.
The optimum pays $f + A_i$, so the algorithm is within 8 of the optimum in expectation.

Bibliography

[1] Kriegel, Hans-Peter; Kröger, Peer; Sander, Jörg; Zimek, Arthur (2011). "Density-based Clustering". WIREs Data Mining and Knowledge Discovery 1 (3): 231–240. doi:10.1002/widm.30.

[2] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise". In Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231. ISBN 1-57735-004-9. CiteSeerX: 10.1.1.71.1980.

[3] Ankerst, Mihael; Breunig, Markus M.; Kriegel, Hans-Peter;Sander, Jörg (1999). "OPTICS: Ordering Points To Identify the Clustering Structure". ACM SIGMOD international conference on Management of data. ACM Press. pp. 49–60. CiteSeerX: 10.1.1.129.6542.

[4] Linda G. Shapiro and George C. Stockman (2001): "Computer Vision", pp 279-325, New Jersey, Prentice-Hall, ISBN 0-13-030796-3

[5] Barghout, Lauren, and Lawrence W. Lee. "Perceptual information processing system." Paravue Inc. U.S. Patent Application 10/618,543, filed July 11, 2003.

[6] Francesco Ricci and Lior Rokach and Bracha Shapira, Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, 2011, pp. 1-35

[7] How Computers Know What We Want — Before We Do

[8] Alexander Felfernig, Klaus Isak, Kalman Szabo, Peter Zachar, The VITA Financial Services Sales Support Environment, in AAAI/IAAI 2007, pp. 1692-1699, Vancouver, Canada, 2007

[9] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Bosagh Zadeh WTF:The who-to-follow system at Twitter, Proceedings of the 22nd international conference on World Wide Web

[10] Mahajan, M.; Nimbhorkar, P.; Varadarajan, K. (2009). "The Planar k-Means Problem is NP-Hard". Lecture Notes in Computer Science 5431.274-285.doi:10.1007/978-3-642-00202-1-24

[11] Aloise, D.; Deshpande, A.; Hansen, P.; Popat, P. (2009). "NP-hardness of Euclidean sum-of-squares clustering". Machine Learning 75: 245–249. doi:10.1007/s10994-009-5103-0.

[12] Dasgupta, S. and Freund, Y. (July 2009). "Random Projection Trees for Vector Quantization". Information Theory, IEEE Transactions on 55: 3229–3242. arXiv:0805.1390. doi:10.1109/TIT.2009.2021326.

[13] Inaba, M.; Katoh, N.; Imai, H. (1994). "Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering". Proceedings of 10th ACM Symposium on Computational Geometry. pp. 332–339. doi:10.1145/177424.178042.

[14] J.B. Kruskal and M. Wish. Multidimensional Scaling. Sage University Paper series on Quantitative Application in the Social Sciences, 07-011, 1978.

[15] I.J. Schoenberg. Metric spaces and positive definite functions. Transactions of the American Mathematical Society, 44:522–553, 1938.

[16] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. Knowl. Inf. Syst., 14:137, 2007.

[17] Stuart P. Lloyd. Least squares quantization in pcm. IEEE Transactions on Information Theory, 28(2):129–136, 1982.

[18] A. Vattani, "k-means requires exponentially many iterations even in the plane," in Proc. of the 25th ACM Symp. on Computational Geometry (SoCG), 2009, pp. 324–332.

[19] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," Journal of the ACM, vol. 51, no. 3, pp. 385-463, 2004.

[20] Arthur, D.; Manthey, B.; Roeglin, H. (2009). "k-means has polynomial smoothed complexity". Proceedings of the 50th Symposium on Foundations of Computer Science (FOCS).

[21] Dasgupta, S. (2008). The hardness of k-means clustering (Technical Report CS2008-0916). University of California, 17 January 2008.

[22] M. Inaba, N. Katoh, and H. Imai, Applications of Weighted Voronoi Diagrams and Randomization to Variance-Based k-clustering, Proc. 10th Ann. ACM Symp. Computational Geometry, pp. 332-339, June 1994.

[23] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. A simple linear time $(1+\varepsilon)$-approximation algorithm for k-means clustering in any dimensions. In FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04), pages 454–462, Washington, DC, USA, 2004. IEEE Computer Society.

[24] Jirí Matousek. On approximate geometric k-clustering. Discrete & Computational Geometry, 24(1):61–84, 2000.

[25] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, pages 291–300, New York, NY, USA, 2004. ACM Press.

[26] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. Comput. Geom., 28(2-3):89–112, 2004.

[27] R.Mettu and C.G.Plaxton, Optimal Time Bounds for Approximate Clustering,Proc.Conf.Uncertainty of Artificial intelligence, 2002.

[28] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In SODA, pages 1027-1035, 2007.

[29] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of Lloyd-type methods for the k-means problem. In FOCS, pages 165-176, 2006.

[30] David Arthur and Sergei Vassilvitskii. k-means++ test code. http://www.standford.edu/ darthur/kMeansppTest.zip.

[31] Phillipe Collard's cloud cover database. ftp://ftp.ics.uci.edu/pub/machine-learning-databases/undocumented/taylor/cloud.data.

[32] KDD Cup 1999 dataset. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[33] Spam e-mail database. http://www.ics.uci.edu/ mlearn/databases/spambase/.

[34] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, Scalable k-means++, Proc. VLDB Endow., 5 (2012), pp. 622–633.

[35] J.Dean and S.Ghemawat.MapReduce: Simplified data processing on large clusters. In OSDI,pages 137-150,2004.

[36] N. Ailon, R. Jaiswal, and C. Monteleoni. Streaming k-means approximation. In NIPS, pages 10-18, 2009.

[37] Alon, Noga; Matias, Yossi; Szegedy, Mario (1999), "The space complexity of approximating the frequency moments", Journal of Computer and System Sciences 58 (1): 137–147, doi:10.1006/jcss.1997.1545, ISSN 0022-0000. First published as Alon, Noga; Matias, Yossi; Szegedy, Mario (1996), "The space complexity of approximating the frequency moments", Proceedings of the 28th ACM Symposium on Theory of Computing (STOC 1996), pp. 20–29, doi:10.1145/237814.237823, ISBN 0-89791-785-5.

[38] Vladimir Braverman, Adam Meyerson, Rafail Ostrovsky, Alan Roytman, Michael Shindler, and Brian Tagiku. Streaming k-means on Well-Clusterable Data. In SODA,2011.

[39] Adam Meyerson.Online facility location. In FOCS,2001.

[40] Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In STOC,2003.

[41] Hochbaum, D. S. (1982). "Heuristics for the fixed cost median problem". Mathematical Programming 22: 148–162. doi:10.1007/BF01581035.

[42] S. Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. In Automata, Languages and Programming - 38th International Colloquium (ICALP), pages 77-88, 2011.

[43] J. Byrka. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. In APPROX '07/RANDOM '07: Proceedings of the 10th International Workshop on Approximation and the 11th International Workshop on Randomization, and Combinatorial Optimization. Algorithms and Techniques, pages 29-43, Berlin, Heidelberg, 2007. Springer-Verlag.

[44] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, STOC '02, pages 731-740, New York, NY, USA, 2002. ACM.

[45] K Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. J. ACM, 48(2):274-296,2001.

[46] Estivill-Castro, Vladimir (20 June 2002). "Why so many clustering algorithms — A Position Paper". ACM SIGKDD Explorations Newsletter 4 (1): 65–75. doi:10.1145/568574.568575.

[47] Bailey, D.L; D.W. Townsend, P.E. Valk, M.N. Maisey (2005). Positron Emission Tomography: Basic Sciences. Secaucus, NJ: Springer-Verlag. ISBN 1-85233-798-2.

[48] "phylogeny". Biology online. Retrieved 2013-02-15.

[49] ESTs Factsheet. National Center for Biotechnology Information.

[50] Stein, L. (2001). "Genome annotation: from sequence to biology". Nature Reviews Genetics 2 (7): 493–503. doi:10.1038/35080529. PMID 11433356.

[51] National Human Genome Research Institute (2010-11-08). "A Brief Guide to Genomics". Genome.gov. Retrieved 2011-12-03.

[52] "The Academic Genealogy of Evolutionary Biology: James F. Crow"

[53] Genotype definition - Medical Dictionary definitions

[54] Population Structure and Eigenanalysis, Nick Patterson, Alkes L. Price, David Reich, s" PLoS Genet 2(12) e190. doi:10.1371/journal.pgen.0020190

[55] Camphausen KA, Lawrence RC. "Principles of Radiation Therapy" in Pazdur R, Wagman LD, Camphausen KA, Hoskins WJ (Eds) Cancer Management: A Multidisciplinary Approach. 11 ed. 2008.

[56] McQuarrie, Edward (2005), The market research toolbox: a concise guide for beginners (2nd ed.), SAGE, ISBN 978-1-4129-1319-5

[57] Malakooti, Behnam (2013). Operations and Production Systems with Multiple Objectives. John Wiley & Sons. ISBN 978-1-118-58537-5.

[58] Wasserman, Stanley; Faust, Katherine (1994). "Social Network Analysis in the Social and Behavioral Sciences". Social Network Analysis: Methods and Applications. Cambridge University Press. pp. 1–27. ISBN 9780521387071

[59] "Yippy.com Site Info". Alexa Internet. Retrieved 2014-04-01

[60] "Recommended Online Video Hosting Services". Groundwire.org. Retrieved 2014-01-19

[61] Boba, Rachel (2005). Crime Analysis and Crime Mapping. Sage Publications. pp. 5–6.

[62] "EducationalDataMining.org". 2013. Retrieved 2013-07-15.

[63] Trung Hung Vo (2007), Software Maintenance