NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS
DEPARTMENT OF MATHEMATICS
GRADUATE PROGRAM IN LOGIC, ALGORITHMS AND COMPLEXITY

On the meaningful instances of clustering

MSc Thesis of

Eleni Bakali

Advisor: prof. D. Achlioptas

$\mathrm{M}\prod \lambda\forall$
Athens
2014

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ
ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΛΟΓΙΚΗΣ ΚΑΙ ΘΕΩΡΙΑΣ ΑΛΓΟΡΙΘΜΩΝ ΚΑΙ ΥΠΟΛΟΓΙΣΜΟΥ

On the meaningful instances of clustering

Διπλωματική Εργασία της

Ελένης Μπακάλη
Α.Μ. 201019

Επιβλέπων Καθηγητής: Δ. Αχλιόπτας

Τριμελής Επιτροπή:
Δ.Αχλιόπτας, Καθηγητής
Ε.Ζαχος, Καθηγητής
Δ.Φωτάκης, Επικ.Καθηγητής

ΜΠ λ∀
Αθήνα, 15 Δεκεμβρίου 2014

....
Ελένη Μπακάλη

Copyright © Ελένη Μπακάλη, 2014.
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν το συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών.

Η παρούσα Διπλωματική Εργασία

εκπονήθηκε στα πλαίσια των σπουδών

για την απόκτηση του **Μεταπτυχιακού Διπλώματος Ειδίκευσης**

στη

Λογική και Θεωρία Αλγορίθμων και Υπολογισμού

που απονέμει το

Τμήμα Μαθηματικών

του

Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών

Εγκρίθηκε την 15 Δεκεμβρίου 2014 από Εξεταστική Επιτροπή αποτελούμενη από τους:

	Ονοματεπώνυμο	**Βαθμίδα**	**Υπογραφή**
1.	Αχλιόπτας Δ.	Καθηγητής	....
2.	Ζάχος Ε.	Καθηγητής	....
3.	Φωτάκης Δ.	Επικ. Καθηγητής	....

Abstract

Clustering is a problem with many different definitions, approaches and applications, but not well defined mathematically. Especially it is not clear how to define meaningfulness, and how to determine if a solution is meaningful, in the sense that it reveals some existing inherent in the data structure.

When we refer to clustering via optimization of some objective functions, it is usually a task performed efficiently, despite that most existing objective functions are NP-hard.

We will present some existing results showing that “meaningful” instances can be solved efficiently. In these papers is made apparent (implicitly or explicitly) a connection between structure in the data, and the behavior of the objective function over the space of solutions.

We will propose a method exploiting this connection, that could decide for each pair {objective function, dataset}, if it is “meaningful” the particular dataset to be clustered by optimizing (or approximating) this particular objective function.

Keywords: clustering, clusterability

Περίληψη

Το clustering (ομαδοποίηση δεδομένων) είναι ένα πρόβλημα με πολλούς διαφορετικούς ορισμούς, προσεγγίσεις και εφαρμογές, αλλά όχι καλά μαθηματικά ορισμένο. Ιδιαιτερα είναι ασαφές το κατά πόσο η λύση που παίρνουμε έχει νόημα, από την άποψη ότι φανερώνει μια υπάρχουσα εσωτερική δομή στα δεδομένα.

Όταν αναφερόμαστε στο clustering μέσω βελτιστοποίησης κάποιας αντικειμένικής συνάρτησης, αυτό συνήθως γίνεται γρήγορα, παρόλο που οι περισσότερες συναρτήσεις που χρησιμοποιούνται είναι NP-hard.

Θα παρουσιάσουμε σύντομα κάποια υπάρχοντα αποτελέσματα που δείχνουν ότι τα instances που έχουν "νόημα" λύνονται και γρήγορα. Από αυτά γίνεται σαφής μία σύνδεση μεταξή εσωτερικής δομής ενός instance (dataset), και της συμπεριφοράς της αντικειμενικής συνάρτησης πάνω στο χώρο των λύσεων.

Θα προτείνουμε μια μέθοδο που εκμεταλλεύεται αυτή τη σύνδεση, για να αποφανθεί για κάθε ζεύγος {αντικειμενική συνάρτηση, dataset}, αν το dataset έχει "νόημα" να ομοδοποιηθεί με αυτή τη συνάρτηση.

Λέξεις κλειδιά: clustering, clusterability

Acknowledgments

I want to thank prof. Dimitris Achlioptas for his guidance and inspiration. I want to thank Panos Theodoropoulos, prof. Stathis Zachos, prof. Dimitris Fotakis, prof. Aris Pagourtzis and Petros Pantavos for their help, support and useful conversations. I want to thank my family and my friends that always support me.

Contents

Chapter 1

Introduction

Clustering is the task of grouping objects of a data set in such a way that similar objects are grouped together, while dissimilar objects belong to different groups.

It is not uniquely specified as a mathematical problem, and so there is a great variety of approaches, based on different intuitions about what clusters are (convex regions, dense areas, connected components of a graph, multivariate distributions, subspaces etc.), and how should someone find them in a automated way. All approaches assume, but not explicitly, that there is a specific structure in the data set, which the algorithm is going to reveal. Though, many times the algorithms are used even in cases where we do not know if there really is the particular structure that the algorithm is intended to work for, or equivalently we don't know if we have chosen the right model to describe the data, or equivalently we don't know if the instance is rather a random instance (at least from the point of view that we are viewing it).

Some approaches aim at directly optimizing some objective function, which in general is an NP-hard to optimize function, and so approximation algorithms are used. Some other are heuristics based on several spatial intuitions, and the resulting clustering is evaluated afterwards with some criterion. (And there are some other approaches, that do not belong to either of the above cases, but we are going to restrict ourselves only to these.)

We want to give a necessary condition (and a method in fact), to determine whether a given data set has an inherent clusterable structure, compatible with a given clustering-objective function or evaluating criterion, and so to answer to the question whether it is meaningful to optimize this function, or equivalently whether the clustering that will occur will be meaningful in some sense.

This method exploits a relation between structure in the data and the behaviour of the objective function over the space of solutions.

This relations has been apparent in many papers, which prove for certain objective functions that the "natural" instances, which occur in practice, are easy. Those papers differ in the way they define what a "natural" instance is. We will give a brief survey of these papers.

Our approach to characterize an instance as having a structure compatible with the objective function, is the following. There should be a unique global optimum, and all the near-optimal (i.e. approximate) solutions, should also not differ in too many points from the optimum, and from each other. Otherwise it would not be meaningful to try to find the optimum of this function, or even to approximate it.

For that reason we need to look at the behavior of the function in the whole space of solutions. The idea is to take many samples from the space of solutions with probability which is higher for the good solutions (i.e. those with good objective value), and smaller for the bad ones, and see how often two points belong to the same cluster. If the instance has the property that we defined above, then for most of the pairs of points, the frequency (probability) that they belong to the same cluster will tend to 1 or 0, as the number of samples goes to infinity. In the opposite, if there are more than one good solutions with very similar objective value, but which differ in many points, then these frequencies will be far from 1 or 0 for many points.

In particular we define the Boltzmann distribution over the space of solutions as the probability distribution according to which we take the samples, i.e. for each partitioning σ,

$$\text{Pr}(\sigma) = \frac{e^{-\beta f(\sigma)}}{Z_\beta},$$

where f is the objective function (we will also call it "energy" function), β is a positive real parameter, and Z_β is a normalizing factor.

There is a last issue that we need to clarify. The Boltzmann distribution depends on the parameter β. When β tends to 0, the Boltzmann distribution tends to be uniform. In that case, the probability of two points to belong to the same cluster will tend to $1/2$ for all pairs. On the other hand, when β tends to infinity, the Boltzmann distribution tends to give all the probability mass to the global optimum of the function. In that case, the probability of two points to belong to the same cluster will tend to either 1 or 0, depending on whether the globally optimal solution puts them in the same cluster or not. As β changes, if the transition from one of these two extreme distributions to the other happens smoothly or suddenly, reveals whether there is some clusterable structure in the data set, (there may be more sudden transitions although). As β starting from large values, it gets smaller, the range in which we demand for these probabilities to keep being close to 1 or 0, should depend on the approximation factor that we can tolerate when trying to optimize the corresponding function, or in other words the factor for which the corresponding solutions can be considered sufficiently good.

Finally we study several algorithms to compute efficiently the matrix with the mentioned above probabilities.

Chapter 2

Clustering Analysis

In this section we will give a quick view of the field of clustering analysis. One main problem is that clustering is not a mathematically well posed problem. We will describe the various attempts that have been done to systematize its study, we will mention some issues that remain non-fronted, and we will explain the contribution of our work to one of these problems.

There are (at least) three axes concerning clustering analysis. The development of clustering algorithms, the evaluation of clusterings given a dataset, and the evaluation of the clusterability of a dataset with respect to some criterion. Before someone tries to deal with each of them, they should be axiomatized in a way that defines exactly what is the problem that we need to solve in each case.

In this section we will present the existing from bibliography axiomatization for the first two aspects that we mentioned, and we will concentrate on the third one, the axiomatization of meaningfulness.

In this thesis we are not going to develop another algorithm for clustering, nor are we going to deal with the question of determining whether some criterion (objective function) is suitable for clustering in general (i.e. if it resembles what we commonly mean by the word "clustering"). Instead we are going to deal with the clusterability of a dataset with respect to some given criterion, in other words with the question whether there is some inherent structure in the data, compatible with the structure that the given criterion aims to reveal.

We believe that this is (non-trivially) related to the computational complexity of clustering. Theoretically clustering is generally hard (NP-hard), but in practice it is an easy task, and the heuristics used are very efficient. This contradiction between theory and practice can be explained by the fact that in practice we usually have some extra information about the structure that we try to reveal.

Although the problem of optimizing or approximating some clustering objective function, and the problem of reconstruction the inherent clustering are two different tasks, we believe that they are closely connected. Our conjecture is that clustering a dataset w.r.t. some criterion is meaningful, iff it is easy to optimize (or approximate).

We will try to examine the truth of this assertion, after we provide evidence from the bibliography.

We refer to [17], [16], [32] for detailed surveys on clustering analysis.

2.1 Definition

Clustering is the task of grouping objects of a data set in such a way that similar objects are grouped together, while dissimilar objects belong to different groups.

The objects are usually described as vectors $\mathbf{x} = (x_1, \dots, x_n)$, and the corresponding coordinates are usually called attributes. The values of the attributes are either numerical, continuous or discrete, either categorical.

It is not uniquely specified as a mathematical problem, and so there is a great variety of approaches, based on different intuitions about what clusters are (convex regions, dense areas, connected components of a graph, multivariate distributions, subspaces etc.), and how should someone find them in a automated way. All approaches assume, but not explicitly, that there is a specific structure in the data set, which the algorithm is going to reveal.

2.2 The importance of the distance measure

In clustering, the solution (the grouping) does not, and should not, depend on the absolute positions of the data in the corresponding space, neither on the precise interpretation of them. It only depends on the relative positions of them. In particular, their pairwise distances are of great importance, and the choice of the similarity / distance measure is a difficult task of its own. The question of whether there is some clusterable structure in the data, is always considered with respect to the particular chosen distance measure.

Some common choices of distance measure are the following.

1. The L_p norm: $D(\mathbf{x}, \mathbf{y}) = (\sum_{1 \leq i \leq n} (x_i - y_i)^p)^{1/p}$, which is extension of the Euclidean distance.

2. The Mahalanobis distance, when $\mathbf{x}, \mathbf{y}$ are considered as random vectors from the same distribution with covariance matrix S: $D(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T S^{-1} (\mathbf{x} - \mathbf{y})}$. This is also an extension of the euclidean distance, since they coincide for $S = \mathbb{I}$. Note that this measure is scale invariant! (a desirable property for clustering).

3. The inner product, or angle between the two vectors.

4. The Hamming distance, i.e. the number of disagreeing coordinates.

5. The edit or Levenshtein distance, for strings.

6. The Dynamic Time Warping distance and the Longest Common Subsequence distance (LCSS), for time series.

7. The Jaccard coefficient for sets: $J(A, B) = |A \cap B| / |A \cup B|$

8. The discrete metric $D(x, y) = 1$ if $x = y$, else 0.

9. Kernels, which are matrices that give for each pair of points, the distance they would have, if they were interpreted in a different space (of either bigger or smaller dimension).

Most of them are metrics (i.e. $D(x, y) = D(y, x), D(x, y) \geq 0, D(x, y) = 0 \Leftrightarrow x = y$, and the triangle inequality holds), but this is not a necessary property.

2.3 Various Models and Approaches

2.3.1 Cluster Models

Depending on the different characterizations of a cluster and its properties, different approaches exist. Some common cluster notions are the following.

- Convex or spherical regions in the data space
- Dense regions in the data space
- Subspaces of the data space
- Groups with objects close to each other
- Groups generated by some underlying probability distribution
- Cliques or highly connected components of a graph

2.3.2 Characterizations of Clusterings

We can independently distinguish clusterings, according to their desired properties, as follows.

- Hard clustering. Each object belongs to a cluster or not.
- Soft or fuzzy. Each object belongs to each cluster with some probability.
- Strict. Each object belongs to exactly one cluster.
- Strict with outliers. Some objects are considered as noise and don't belong to any cluster.
- Overlapping. Objects can belong to more than one cluster, but there are no associated probabilities.
- Hierarchical. Clusters are refined hierarchically, and each object belonging to a child cluster, belongs to its parent cluster, too.

2.3.3 Clustering algorithms

There are several different approaches for clustering. Some of then aim at optimizing some objective function. Other aim at detecting directly groups that satisfy the characteristics of some underlying cluster model, without optimizing explicitly any objective function. However the resulting clustering is evaluated afterwards (see section 2.4). In other words the first kind of algorithms do not assume explicitly the existence of a particular structure, but assume that this structure should optimize the particular objective function. The second kind of algorithms seek for particular structures, without reference to any objective function.

For example, the first basic category of algorithms, hierarchical algorithms, proceed repeatedly, either by starting from one cluster containing all data points, and then separating it to smaller, and these consecutively to smaller, either by merging small clusters to bigger, starting from singleton clusters. In this way a hierarchy of clusters results, such that each point belonging to a cluster, belongs to all its ancestors too.

The second basic category is density based algorithms, which detect areas that are more dense in the data space, and are separated from each other by sparser areas.

The third type of algorithms are optimization algorithms, like k-means.

There are also many other approaches based on statistical methods, neural networks and other, but we are not going to deal with them. We concentrate on optimization algorithms.

In the next chapter we will present the attempts to determine axiomatically the properties that an algorithm should meet to be appropriate for clustering.

In the following section we present some commonly used objective functions.

2.3.4 Clustering objective functions

Some commonly used objective functions for clustering are the following.

1. k-means: minimize $\sum_{c\in C}\sum_{x\in c} d^2(x,\sigma_c)$, over all k-partitions C, where c denotes a cluster, σ_c is the center of cluster c, i.e. the point in the underlying space X that minimizes $\sum_{x\in c} d^2(x,\sigma_c)$, d is the distance measure defined on space X. Essentially σ_c is the center of mass of points in c. The problem is NP-hard, but there exists an approximation scheme.

2. k-median: minimize $\sum_{c\in C}\sum_{x\in c} d(x,\sigma_c)$, over all k-partitions C, where c, d, σ_c, are defined as before.

3. k-center: minimize the maximum radius of a cluster

4. diameter cost: minimize the maximum diameter of a cluster

5. correlation clustering: given a similarity matrix (with no transitive similarities), minimize the conflicting entries, or maximize the agreeing entries

6. min sum: minimize all the intra-cluster distances

7. worst pair ratio: maximize $\frac{split_c(X)}{width_c(X)}$ over all k-partitions of X, where $split_c(X)$ the minimum distance between two clusters, and $width_c(X)$ the maximum distance between two points in the same cluster.

8. variance ratio: maximize $\frac{B_C(X)}{W_C(X)}$ over all k-partitions of X, where $B_C(X)$ the variance of the cluster centers, and $W_C(X)$ the average (over all clusters) variance of points in the same cluster.

9. strict separation: minimize $\max\limits_{x\in X} \frac{\max_{x\sim y}\|x-y\|}{\min_{x\not\sim y}\|x-y\|}$, over all clusterings of X.

10. medoids or exemplars: Let X be the data set, find a subset S of X, minimizing $\sum_{x\in X} d(x,\sigma_x)$, where σ_x is the point in S nearest to x (called the exemplar of x). The dissimilarity measure d has the property that $d(x,x)$ is not zero for every x, else the solution would be trivial: every point would be the exemplar of itself.

11. Max-Sum: maximize $\sum_{c\in C}\sum_{i,j\in c} s(i,j)$, over all k-partitionings C, where c denotes a cluster, and s is a similarity measure (instead of distance). It is NP-hard for $k > 2$. For $k = 2$ it is equivalent to minimum cut, thus in P. For $k > 2$ there exists a $2-2/k$ approximation algorithm (iterative minimum cuts) [Vazirani 2001].

Note Note that an algorithm that operates by optimizing some objective function, should satisfy the axioms [21] for clustering algorithms (see chapter 3). In [5] the authors translated these axioms to properties of the objective function, to be suitable for clustering (to be precise they refer to a normalized version of objective functions). In the next chapter we will present this attempt as well (see section 3.2).

2.4 Evaluation / Validation

Sometimes when the clustering is found with a heuristic algorithm, we want to evaluate it afterwards. This evaluation is called *external* and is being done using some predefined structure, or labeled objects.

The *internal* evaluation is performed using evaluating functions, like the objective functions that some algorithms try to optimize directly.

But when an algorithm optimizes directly an objective function, the absolute value of a particular clustering, may not give useful information about its quality.

The objective functions, since they are not scale invariant, cannot give a measure of the quality of a clustering in the following sense. A non normalized objective function cannot compare clusterings between different instances, and even for the same instance the optima of two different objective functions cannot be compared to each other.

To overcome this difficulty, normalized quality measures are used to evaluate clusterings.

In [5] the authors determine some properties-axioms that a clustering quality measure should satisfy (see chapter 3). We note again that when we refer to measures that occur as normalized versions of clustering objective functions, then these axioms determine if an objective function is suitable for clustering (in general, regardless of the particular instance).

This set of axioms is satisfied by many quality measures. We give some of them.

2.4.1 Examples

One way to obtain clustering quality measures, is by normalizing a clustering objective function. For example

- Let L be a clustering objective function, and C_{all} the clustering consisted by a single cluster. The L-clustering quality of a clustering C over (X, d) is

$$L - CQ(C, X, d) = \frac{L(C_{all}, X, d)}{L(C, X, d)}$$

Three commonly used validity indices are the following. They identify compact and well separated clusters.

Davies Bouldin index For clusterings with centroids DB index is defined as

$$\text{DB} = \frac{1}{k}\sum_{i=1}^{m} \max_{i \neq j}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$$

where k is the number of clusters, c_i is the centroid of cluster i, σ_i is the average distance of all points in cluster i to its centroid, and $d(c_i, c_j)$ is the distance between c_i and c_j. Small values of this index indicate good clusterings.

Modified Hubert Γ statistic For centroid based clusterings Γ index is defined as

$$\Gamma = \frac{1}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} P(i,j)Q(i,j)$$

where N is the number of points in the dataset, P is the proximity matrix, $M = N(N-1)/2$ and Q is an $N \times N$ matrix s.t. $Q(i,j)$ is the distance between the centroids of the clusters to which i and j belong.

Observe that this index counts only distances between different clusters, since ...... is zero for i, j in the same cluster. It also gives more weight to distances between points belonging to clusters far apart from each other.

The normalized $\hat{\Gamma}$ index is defined as

$$\hat{\Gamma} = \frac{1}{\sigma_P\sigma_Q}\frac{1}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(P(i,j)-\mu_P)(Q(i,j)-\mu_Q)$$

where μ_P, μ_Q, σ_P, σ_Q are the means and variances of P, Q.

If we have compact and well separated clusters, then P and Q will be close in agreement, and the values of this index will be high.

Dunn index (Dunn 1974) Let $d(i,j)$ be a function that measures the distance between clusters i and j (for example it can be the distance of their centroids if they are center based, or the minimum distance between two points, one from each cluster), and let $d'(k)$ be a function measuring the intra-cluster distance of cluster k (for example it could be the diameter of the cluster, or the sum of distances from all points to the center). The D index is defined as

$$\mathrm{D} = \min_{1\le i\le n}\left\{\min_{1\le j\le n, i\ne j}\left\{\frac{d(i,j)}{\max_{1\le k\le n} d'(k)}\right\}\right\}.$$

where n is the number of clusters.

Large values of this index indicate compact and well separated clusters.

This index is sensitive to noise.

There are also validity indices applicable to hierarchical clustering (see [32]).

Note The optimization of these indices may be inefficient, but note that instead of optimizing them, we use them just to evaluate clusterings produced by other algorithms.

See [32] for more details on clustering validity.

2.5 Goals and Applications of Clustering

2.5.1 Two general purposes

- compression
- structure discovery

2.5.2 Applications

Clustering is an important task of exploratory data analysis. Some indicative of its many applications are the following:

- hypothesis generation,
- hypothesis testing,
- prediction based on groups,
- business applications and market research (purchasing patterns, discovery of groups of customers),
- biology and bioinformatics (categorize genes according to their functionality, define taxonomies, find structures inherent in populations),
- spatial data analysis (e.g. satellite images, geographical information systems, medical equipment, image database exploration),
- web mining (groups of document, groups of social networks),
- image segmentation,
- preprocessing step for other algorithms (e.g. classification).

2.6 Evaluating Usefulness

Usefulness depends on the particular purpose that the clustering is intended for. For example if the purpose is just compression of the data, as a pre-process for other tasks, then any clustering may be useful, even if there is not a particular structure inherent in the data.

On the other hand if the purpose is to reveal structure, there should be away to distinguish between structured data and random data. There are some meta-criteria used for this purpose. They don't evaluate a particular clustering of a data set, but they aim at determining the statistical uncertainty in the data.

2.6.1 Metacriteria

Some of these criteria are the following. The first four of them assume an underlying distribution from which samples are drawn, and clustering is performed on the samples. See [34] for references and more details.

- Stability of an algorithm w.r.t. input randomization (for example if we sample from the data-space and then cluster the samples, or if we perturb slightly the input).

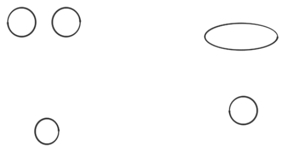
  Stability is usually used to determine the true number of clusters. The idea behind this is that if we have chosen a larger number of clusters than the true one, then the algorithm will "randomly" split some clusters, and similarly if we have chosen a smaller number of clusters, then the algorithm will "randomly" merge clusters, resulting to instability.

For center based algorithms, stability is fully determined by the behavior of the objective function over the space of solutions [6] and specifically on the existence of a unique minimizer.

It is also proved for some center based schemes, that stability indicates the existence of well separated clusters ([3]).

So it turns out that a good behavior of the objective function, as well a good clusterable structure in the data-set, result in stability. However the opposite implication does not hold, and so stability is not suitable for determining the number of clusters or the ground truth clustering.

For example (from [6]) in the following figures, k-means on the first dataset is stable even for $k = 2$, although the clusters are 3, and similarly on the second k-means is stable for $k = 3$, although the clusters are 2.



Another reason that stability isn't an indicator for the right number of clusters is the following. In [6] is noted that the use of stability to identify the number of clusters as stated before, is successful when we have symmetries in the data, however real datasets are not perfectly symmetric, so the objective function has a global minimizer causing an undesirable stability.

However, as it is indicated in [33], the algorithms used in practice don't find the global optimum, but local optima, so we use this disadvantage for good, i.e. with the hope that different initializations will lead to different local optima.

- Convergence of clustering algorithms. Clustering is performed for increasing amounts of data. Then it is checked whether the solutions converge to one particular clustering.
- Generalization bounds. These quantities evaluate how much the results of clustering applied to a small sample of data, deviate from the clustering we would obtain on the full underlying distribution.
- Statistical significance. There are some so called "confidence scores" that evaluate how much the clustering results deviate from what we would obtain on completely random data.
- Behavior of validity indices w.r.t. the parameters of an algorithm. We apply an algorithm with different parameters, and plot the values of an index. For example if the number of clusters k is not predefined, and the plot is monotonically increasing or decreasing with k, then the existence of a "knee" may be an indication for the true number of clusters, as it causes a significant change in the behavior of the index. In random (unstructured) data, there is no such "knee". (see [32] for more on the use of indices)

2.6.2 Our contribution. Evaluating meaningfulness

Given a dataset and a clustering objective function, an important question is whether it is meaningful to cluster by optimizing this objective function, in the sense that the optimum (or approximate optimum) of this function reveals some inherent in the data structure.

We propose axiomatically a property that should be satisfied by a pair of a dataset and a clustering objective function, in order to be meaningful to cluster this particular dataset by optimizing or approximating this particular function.

The **axiom** we propose is strong non-degeneracy.

Let X be a dataset and d a distance measure over X. A clustering objective function takes as input X, d and a clustering C of X and returns a real number.

Definition 1. ***Strong non-degeneracy.*** *We call a clustering objective function (λ, ϵ)-strongly non-degenerate with respect to (X, d), if it has a unique global optimum clustering, and moreover every λ-approximate clustering of X is ϵ-close to the optimum.*

This is a property that is satisfied whenever there is a structure inherent in the dataset, compatible with the objective function. (For example if we have the k-means objective function, a compatible structure could be k convex dense and separated by empty space regions.) If this property does not hold, it is not meaningful to even try to optimize this function, since clustering aims to reveal structure, but this structure cannot be determined uniquely by the optimum of our function.

This property is necessary but not a sufficient to characterize a clustering as meaningful, since there are strongly non degenerate functions that are not suitable for clustering at all. The axioms of clustering quality measures should be satisfied by the normalized objective function, to be suitable for clustering (see chapter 3).

Once we have such a function, the question that arises is whether this function is appropriate for the particular instance (dataset) at hand. The values of (λ, ϵ) is an indicator of the *clusterability* of our dataset with respect to our objective function.

We will propose methods to compute these values explicitly (see chapter 6).

2.7 Open questions and research directions about clustering

Some of the most important research directions in the area of clustering, as expressed in [32] are the following

- Is there a principled way to measure the quality of a clustering on particular data set?
- Can every clustering task be expressed as an optimization of some explicit, readily computable, objective cost function?
- Can stability be considered as a first principle for meaningful clustering?
- How should the similarity between different clusterings be measured?
- Can one distinguish clusterable data from structureless data?
- What are the tools that should be imported from other relevant areas of research?

Chapter 3

Axiomatization of clustering

As we saw in chapter 2, clustering is an intuitive task, not well defined mathematically. There are some attempts to define it through axioms, but this research area is still open. We present some basic approaches.

The first line of research concerns axioms-properties that must be met by a clustering algorithm (or clustering function, since an algorithm is a function that takes a dataset as input and outputs a partitioning).

The second line concerns axioms-properties that a clustering objective function should satisfy (i.e. a function that we optimize in order to determine a clustering).

But these axioms refer to an objective function, regardless of the particular dataset. So an objective function may be proper for clustering in general, but it may not be meaningful for a particular instance.

So finally we propose an axiom that should be satisfied by a pair of a dataset along with an objective function, to be clustering meaningful in terms of structure discovery.

3.1 Axioms for clustering functions / algorithms

No concrete theory. Several attempts. One impossibility theorem.

One of the problems of clustering is that it is not well defined as a computational task. It is not clear what are the properties that a clustering algorithm should satisfy.

Definition 2. *Let S be a set of n objects, and d a distance measure on S. A partitioning function is a function that on input a distance measure on S it outputs a partition of S into disjoint subsets whose union is S.*

There have been several attempts to define the properties that a partitioning function should have to be characterized as a clustering function.

The first attempt was by Kleinberg in [21] where he defined formally three natural properties that should characterize a clustering function, and he proved that it is impossible for any function to satisfy all of them.

To overcome this difficulty, relaxations to these properties have been proposed. However we have some speculations about these approaches, which we explain thereupon.

3.1.1 Kleinberg's Axioms

The three natural properties firstly defined are scale invariance, richness and consistency. Let f be a partitioning function.

Scale Invariance Let $a > 0$ and $d' \equiv a \cdot d$ the distance function s.t. for all i, j $d'(i,j) = a \cdot d(i,j)$, where d is a given distance function.

For any distance function d and any $a > 0$, we have $f(d) = f(a \cdot d)$.

This property is the requirement that the clustering should not depend on the absolute distances between the points, or the units of measurement, but only on their relative pairwise distances.

Richness *Range(f) is the set of all partitions of S.*

This means that for any partition C of S it is possible to construct a distance function d s.t. $f(d) = C$.

Consistency Let C be a partition of S and d, d' two distance functions s.t. for every i, j belonging to the same cluster of C it holds $d'(i,j) \leq d(i,j)$, and for every i, j in different clusters $d'(i,j) \geq d(i,j)$. We will call d' a C-transformation of d.

For any d if $f(d) = C$, then for any C-transformation d' of d, $f(d') = C$.

This means that when given a distance measure d we get clustering C, then if we shrink the distances of points in the same cluster, and / or stretch the distances between points in different clusters, then the clustering should remain the same.

Theorem 3. *For each $n \geq 2$, there is no clustering function f that satisfies Scale Invariance, Richness and Consistency. [21]*

Proof. (sketch) The idea idea of the proof relies on the fact that if we have a function f satisfying consistency and richness, then for every partition C there exist positive real numbers $a < b$ s.t. for all d s.t. $\forall i, j$ in the same cluster $d(i,j) < a$ and $\forall i, j$ in different clusters $d(i,j) > b$, then $f(d) = C$. This contradicts to the property of scale invariance. The contradiction is proved in a rather complicated way. We will try to explain it in a simpler way:

Let A be the partitioning consisting of a single cluster, with all the points together. Let B be any other partitioning. The above mentioned fact implies the existence of numbers (a, b) for A, and (a', b') for B. That means for every d if $\forall i, j, d(i,j) < a$, then $f(d) = A$. (1)

Now take a d conforming to (a', b'). This enforces $f(d) = B$.

Shrink d to d' so that $\forall i, j, d'(i,j) < a$.

This, from scale invariance implies $f(d') = B$, but from (1), implies $f(d') = A$.

(The original proof does not assume A to be the single cluster partition, but it assumes that A is any partition, and B any refinement of A, i.e. a partition in which every cluster is completely contained in some cluster of A.) □

3.1.2 Relaxed Axioms

In [36] Zadeh and Ben David tried to overcome the above impossibility result, by relaxing the richness property, so that not every partition should be a possible outcome of f, but only partitions to k clusters. This of course, limits the number of functions that can be described in this axiomatic framework. They consider functions that depend not only on the distance function d, but also on the parameter k.

k-Richness *Range(f) is the set of all k-partitions of S.*

The Scale Invariance property is the same, but the consistency property is with respect to k.

Theorem 4. *[36] This set of axioms is consistent. For example single linkage clustering (i.e. the minimum spanning tree method) satisfies them.*

3.1.3 Our speculations. Redefining Consistency

The above solution to the problem of solution relies on the fact that consistency and richness together imply the property we mentioned (the existence of the numbers a, b etc.) which contradicts scale invariance. So to overcome this implication they changed the richness property.

However, I think that the problem is not richness, but the property of consistency. Richness is a natural property, but consistency, as stated, is not natural neither reasonable. The reason is that it demands for the clustering function to give the same result even if you change the structure of the clusters.

Consider the following example. Let d be a distance function, f a partitioning function, and $C = f(d)$ the resulting clustering. Let $a = \min_{ij}\{d(i, j)\}$. Consider a cluster c of C. Now Shrink d uniformly so that for every pair i, j in c, $d'(i, j) < a$. Then partition the elements of c into two groups and shrink again uniformly the distances only between pairs of points in the same group. The consistency property requires that $f(d') = C$ again. However this requirement is not natural neither reasonable.

Maybe a more proper definition of consistency could be stated. I suggest the following two.

Recall that a clustering C is a refinement of clustering D, if every cluster of C is contained completely in some cluster of D.

Consistency 1 *A clustering function f is consistent if whenever we shrink the distances of points belonging to the same cluster by a common factor (different for each cluster), or stretch the distances of points not in the same cluster by some other common factor, then the clustering doesn't change.*

Consistency 2 *A clustering function f is consistent if whenever we shrink the distances of points belonging to the same cluster (arbitrarily), then the new clustering should be a refinement of the old one. And whenever we stretch the distances of points not in the same cluster (arbitrarily), then the old clustering is a refinement of the new one.*

Consistency 3 [5] *A clustering function f is consistent if whenever we shrink the distances of points belonging to the same cluster by a common factor (different for each cluster), or stretch the distances of points not in the same cluster in a way such that there exists a set of points, one from each cluster, whose pairwise distances are stretched by some common factor, then the clustering doesn't change.*

Theorem 5. *[5] Consistency 3, Scale invariance, and Richness, is a consistent set of axioms.*

Open question Is Consistency 1 or 2 consistent with scale invariance and richness?

Note These axioms concern a clustering function, and so are also properties that an algorithm for clustering should satisfy. They don't concern an objective function. Next we refer to this issue.

3.2 Axioms for clustering objective functions

Note that an algorithm that operates by optimizing some objective function, should satisfy the axioms for clustering algorithms. In [5] the authors translated these axioms to properties of the objective function, to be suitable for clustering (to be precise they refer to a normalized version of objective functions). We present this attempt here.

Let X be a dataset, d a distance function over X, and C a clustering of X.

Definition 6. *A clustering quality measure m is a function that is given a clustering C of (X, d) and returns a non-negative real number.*

The axioms proposed in [5] for clustering quality measures are the following.

Scale Invariance

Quality measure m satisfies scale invariance if for every clustering C of (X, d), and every positive λ, $m(C, X, d) = m(C, X, \lambda d)$

Isomorphism Invariance

Clusterings C and C' over (X, d) are isomorphic, $C \simeq_d C'$, if there exists a distance preserving isomorphism $\phi : X \to X$, such that x, y belong to the same cluster of C iff $\phi(x), \phi(y)$ belong to the same cluster of C'.

Quality measure m is isomorphic invariant if for all clustering C, C' over (X, d) which are isomorphic, $m(C, X, d) = m(C', X, d)$.

Weak Local Consistency

Distance function d' is a C weakly locally consistent variant of d, where C is a clustering over (X, d) if

- For every cluster C_l of C there is a constant $c_l \leq 1$, such that for all $x, y \in C_l$, $d'(x, y) = c_l d(x, y)$.
- For every x, y not in the same cluster of C, $d'(x, y) \geq d(x, y)$.
- Fore some set of points containing a point from every cluster C_l, there exists a constant $c \geq 1$ such that, for every $p_l, p_{l'}$, $d'(p_l, p_{l'}) = c \cdot d(p_l, p_{l'})$.

Quality measure m is weakly locally consistent if for all clusterings C over (X, d), whenever d' is a C weakly locally consistent variant of d, then $m(C, X, d) \leq m(C, X, d')$.

This axiom requires that if we shrink each cluster uniformly, and expand the clusters uniformly in the sense that there is a representative point in each cluster such that the distances between the representatives expand uniformly, then the quality of the clustering cannot decrease. (We don't require all distances between points in different clusters to expand uniformly. For example in euclidean spaces, if we shrink each cluster uniformly, the distances of pairs of points not in the same cluster, don't change uniformly, yet the new distances are consistent with the clustering, thus its quality cannot decrease.)

Co-final Richness

A distance function d' is a C-consistent variant of d if $d'(x, y) \leq d(x, y)$ for all x, y in the same cluster of C, and $d'(x, y) \geq d(x, y)$ for all x, y in different clusters of C.

Quality measure m satisfies co-final richness if for every pair of non-trivial clusterings C over (X, d) and C' over (X, d'), there exists a C-consistent variant d'' of d such that $m(C, X, d'') \geq m(C', X, d')$.

This means that a clustering can become arbitrarily good by consistent changes of the distance function.

3.3 Axiom for meaningfulness

The previous axioms concerned properties that should be satisfied by an algorithm in order to be suitable for clustering, and the same for objective functions. But these properties do not take into account the particular instance.

We propose axiomatically a property that should be satisfied by a pair of a dataset and a clustering objective function, in order to be meaningful to cluster this particular dataset by optimizing or approximating this particular function.

The **axiom** we propose is strong non-degeneracy.

Let X be a dataset and d a distance measure over X. A clustering objective function takes as input X, d and a clustering C of X and returns a real number.

Definition 7. ***Strong non-degeneracy.*** *We call a clustering objective function (λ, ϵ)-strongly non-degenerate with respect to (X, d), if it has a unique global optimum clustering, and moreover every λ-approximate clustering of X is ϵ-close to the optimum.*

This is a property that is satisfied whenever there is a structure inherent in the dataset, compatible with the objective function. (For example if we have the k-means objective function, a compatible structure could be k convex dense and separated by empty space regions.) If this property does not hold, it is not meaningful to even try to optimize this function, since clustering aims to reveal structure, but this structure cannot be determined uniquely by the optimum of our function.

This property is necessary but not a sufficient to characterize a clustering as meaningful, since there are strongly non degenerate functions that are not suitable for clustering at all. The axioms of clustering quality measures should be satisfied by the normalized objective function, to be suitable for clustering.

Once we have such a function, the question that arises is whether this function is appropriate for the particular instance (dataset) at hand. The values of (λ, ϵ) is an indicator of the *clusterability* of our dataset with respect to our objective function.

We will propose methods to compute these values explicitly.

Chapter 4

Meaningful instances are easy: An overview

Although most objective functions used for clustering or partitioning, are NP-hard in general, it can be conjectured that whenever there is a well clusterable structure in the data-set, then it should be easy to find it, since clustering is indeed easy in practice.

This conjecture is supported by many papers, which prove for certain objective functions that the "natural" instances, which occur in practice, are easy. Those papers differ in the way they define what a "natural" instance is. Although these definitions do not agree on what constitutes a natural instance, they are all cases of instances that can be solved efficiently.

In these papers we can see implicitly or explicitly, a connection between inherent structure in the data, and the behavior of the objective function over the space of solutions. Roughly speaking, good (clusterable) structure translates to good behavior of the objective functions, and the opposite.

4.1 The several definitions of meaningful instances and relations between them (w.r.t. the behavior of the objective functions)

In most cases within the existing bibliography, the clusterability of a dataset is defined in accordance to the behavior of energy in space $\mathcal{X} \times \mathcal{C}$, independently of the precise definition of energy and the metric in space $\mathcal{C}$. All researchers claim that instances with clusterable structure should be solved efficiently, and that in reality these are the kind of instances that come up and not hard ones. Definitions rely on the perception of each researcher on the kind of characteristics identifying a physical / natural input. All researchers propose algorithms solving instances with such characteristics. The authors of the research papers that will be referred to, set as a perquisite for an efficient solution that the instance has these very characteristics (clusterable structure). More explicitly, the existing literature supports the hypothesis that instances are in fact clusterable, thus enabling the implementation of the specific algorithm.

On the contrary, we try to answer to the question "When does an instance have a clusterable structure?".

The definitions Linial et al. [8],[7] define **stability** of X as follows: the solution of minimum energy C remains the same after a small perturbation in X. It also shown that this is equal to the property that the second best solution is much worse than the first one, but not by a multiplicative factor.

They also define a weaker property **local stability** where the solution is a local minimum in its direct neighborhood. The same applies to their definition [11] of (α, γ)-clusterings as well.

Blum et al. [2] define **approximation stability** as follows: all C of minimum or approximately minimum energy, lie close to each other as well as to the target (although it is possible that amongst them there are others of higher energy).

According to Ben David [4] **robustness** of X is defined as follows: energy E has a small slope around the minimum, in other words it remains almost the same after a small perturbation in C (a unique minimum is not presumed).

Linial et al. [8] also define **edge/vertex distinction**: the energy E has a big slope around the optimum. This is the opposite from Ben David's robustness. However this is due to a difference in measuring the distance between clusterings. See at the end of section 4.4 for a discussion on this issue.

Linial et al. also believe that stable and edge distinct instances are rare, but are the only interesting instances, while Ben David believes that the difficult ones are rare and not interesting, and that only robust instances arise in practice.

Finally Linial et al. show that edge distinct instances are stable, but the opposite doesn't necessarily hold.

In addition to the above, Ben David [1] gives the following definitions of **clusterability**: X is k-clusterable when:

(a) the minimum of E energy is considerably small (but not necessarily unique)

(b) the minimum of E energy is considerably reduced if the number of clusters are raised from $k-1$ to k (this could possibly be classified as "big slope for a small change in C").

In stochastic models [13][12] [25] it is considered that a planted clustering exists, which has been subjected to noise. If the noise is fairly small then we can recover it. So an instance is meaningful if the noise is not so big that we lose the initial information. However there are two different tasks related to stochastic models: reconstruction and approximation. Reconstruction is done when possible, without reference to the energy landscape of any particular objective function. Approximation is done with respect to some specific objective function. But as it turns out in [25] approximation doesn't mean reconstruction, i.e. there could be several solutions of equal energy, but different from each other. Only when the noise is very small, the approximation algorithm reconstructs the solution.

It is not clear that this holds due to some good property of the energy landscape of such instances. However, if we compute the information theoretical bounds for reconstruction, we observe that when reconstruction is possible, then the planted partition can be found by belief propagation (an algorithm we present later), as in [12]. The same doesn't hold for planted coloring, i.e. there are regions of noise values where reconstruction is possible but hard. This leads us to the conjecture that the cause of hardness in partitioning is mainly the non-reconstructibility, which means the existence of many equally good, but very different from each other solutions.

The formal definitions as well as related results are given in detail in the next sections.

4.1.1 Relations between structure in the data and the behavior of the objective functions

As we will see later in detail, we have the following relations.

- Edge / vertex distinction $\Rightarrow$ stability w.r.t. input perturbations $\Leftrightarrow$ unique optimum much better than any other solution $\Rightarrow$ well separated clusters (for max cut)
- Good clusters and clusterings $\Rightarrow$ local stability (for objectives of the form $\sum_{x \in X} D(x, C_x)$, where D denotes distance of point from cluster).
- Approximation stability $\Rightarrow$ strict separation, i.e. compact and well separated clusters, with at most ϵn ambiguous points-noise (for k-means, k-median, min-sum).
- Strict separation $\Rightarrow$ the target clustering is among at most $2^{\Theta(\frac{1}{\epsilon})}$ possible good clusterings.
- Conjecture: Hardness in partitioning is due to non-reconstructibility, i.e. to the existence of many quite different, but equally good solutions.

4.2 Stability w.r.t. small perturbations

In this group of papers *stability* of X is defined: the solution of minimum energy C remains the same after a small perturbation in X. It also shown that this is equal to the property that the second best solution is much worse than the first one, but not by a multiplicative factor.

Local stability is also defined: the solution is a local minimum in its neighborhood (where two solutions are neighbors if they differ at exactly one point).

Finally *edge / vertex distinction* is defined: the energy has a big slope around the optimum.

4.2.1 Stability and the easy instances of max cut

In [8] the authors define a concept of stability for instances of a NP-hard problem. They support the view that only such stable cases are in fact interesting, though rare in the space of instances, and show that stable instances can be solved fast. The problem they focus on is the MAXCUT, which can be considered as clustering within two clusters.

Definitions

An instance is defined as stable, if the best solution remains the same after a small perturbation of the data.

Definition 8. *Let W be a $n \times n$ symmetric, non-negative matrix. A γ-perturbation of W for $\gamma \geq 1$, is an $n \times n$ matrix W' s.t. $\forall i, j = 1 \ldots n$ $W_{i,j} \leq W'_{i,j} \leq \gamma W_{i,j}$. Let $(S, [n] \setminus S)$ be a maximal cut of W, i.e. a partition that maximizes $\sum_{i \in S, j \notin S} W_{i,j}$. The instance W of the MAXCUT problem is said to be γ-stable if for every γ-perturbation W' of W, $(S, [n] \setminus S)$ is the unique maximal cut of W'.*

This definition is equivalent to the following one, which expresses stability in relation to the solution space of the specific instance without perturbation.

It states that if an instance is γ-stable, then, if we change solution, the edges that currently will be not cut (whilst they were being cut before), are γ times heavier than those that will be now cut and were not cut before.

Definition 9. *Let $\gamma \geq 1$. A graph G with maximal cut $(S, \bar{S})$ is γ-stable (w.r.t. MAXCUT) iff for every vertex set $T \neq S, \bar{S}$, $W(E(S, \bar{S}) \setminus E(T, \bar{T})) > \gamma W(E(T, \bar{T}) \setminus E(S, \bar{S}))$.*

It is easy to see why the two definitions are equivalent, as long as we notice that in order to force someone to chose a different cut C' instead of the optimal C, the best thing to do is to multiply by γ the weight of the edges which get cut by the new cut and do not get cut by the old one. However, if the hypothesis of the above sentence is valid for every different cut C', then every γ-perturbation taken will not change the optimal solution. Therefore the instance is γ-stable. And reversely, if there was a cut C' not satisfying the condition of the sentence, then the already mentioned γ-perturbation will have C' and not C as optimum, therefore the instance is γ-stable.

In addition, an instance is defined to be edge-distinct if its optimal solution differs substantially in objective value from any other solution, as a proportion of the weight of the edges on which the two solutions disagree (i.e. the edges that get cut by exactly one solution).

Definition 10. *Let $(S, \bar{S})$ be a cut in a weighted graph $G = (V, E)$ and $a > 0$. We say that this cut is a-edge-distinct if for any cut $(T, \bar{T})$,*

$$w(E(S, \bar{S})) - w(E(T, \bar{T})) > a \cdot w(E(S, \bar{S}) \triangle E(T, \bar{T})).$$

$\triangle$ is the symmetric difference of two sets, i.e. their union minus their intersection. In this case, it is the set of edges on which the two cuts disagree. We observe that, if C, C', A, B, G are defined as before, then for a-edge-distinct instances it holds that

$$w(C) - w(C') = A - B > a \cdot (A + B) \Leftrightarrow A \geq \frac{1+a}{1-a} B$$

So if a graph is a-edge-distinct, then it is also $\frac{1+a}{1-a}$-stable.

Moreover, an instance is defined to be (vertex) k-distinct, if its optimal solution differs substantially in objective value from any other solution, in relation to the number of vertices where the two solutions disagree. In other words an instance is vertex k-distinct if the slope of the objective function (which is the weight of the cut) is big, in a neighborhood with small Hamming distance from the optimal solution.

Definition 11. *Let $(S, \bar{S})$ be a cut in a weighted graph $G = (V, E)$ and $k > 0$. We say that this cut is k-distinct if for any cut $(T, \bar{T})$,*

$$w(e(S, \bar{S})) - w(e(T, \bar{T})) \geq k \cdot \min\{|S \triangle T|, |S \triangle \bar{T}|\}.$$

We say that a graph is (k, γ)-distinct (w.r.t. MAXCUT) if its maximal cut is k-distinct and γ-stable.

$\triangle$ is the set of elements on which the two sets disagree (the elements that belong to some set but not to both sets), and $\omega(x)$ is the weight of x.

Finally the authors define a weaker notion, the local stability. Is concerns the transfer of a vertex from one side to the other. It is exactly the same definition with stability, but with the requirement to hold just for the T sets that differ from S in exactly one vertex.

If a graph is stable, it is local stable too, but not the opposite. This holds because it may not be an advantage to change side to any single vertex, but this doesn't mean that it wouldn't be an advantage to change many vertices at once.

Definition 12. *Let W be an instance of the MAXCUT problem and let $(S, \bar{S})$ be its optimal partition. We say that W is γ-locally stable if or all $v \in S$*

$$\gamma \cdot \sum_{u \in S} W_{uv} < \sum_{u \in \bar{S}} W_{uv}$$

and for all $v \in \bar{S}$

$$\gamma \cdot \sum_{u \in \bar{S}} W_{uv} < \sum_{u \in S} W_{uv}.$$

Algorithms

Combinatorial

An algorithm for n-stable instances is given in [8], which is roughly based on the fact that if a graph is n-stable, then we can choose the heaviest edges to cut. In particular, the algorithm works for every $\gamma > \min\{\sqrt{\Delta n}, \frac{n}{2}\}$, where Δ is the minimum degree of the graph.

Observe that if a graph is γ-stable, and we connect two vertices belonging to the same side, keeping multiple edges, then the resulting graph is also γ-stable, and the optimal solution is the same.

The algorithm detects vertices that must be in different sides of the cut, and joins them with edges. So a bipartite graphs are constructed, that are joined until a single connected bipartite graph is constructed, and its two sides constitute the cut.

The criterion of choice (which edge will be included) in each step is the following. We choose a connected component c (of the current graph) with the minimal number of vertices. Then we seek for the best component to join it: Remember that every connected component is either a vertex either a bipartite graph. So every two components can be joint in two ways: either we will put in the same side the left set of the first component with the left part of the other, either left with right. (Except if both of them are single vertices, do we just join them with an edge). So for every other subgraph $c' \neq c$, we find which of the two ways is heavier (resulting to a bigger cut). And among all subgraphs $c' \neq c$, we choose to join c with the one that maximizes the current cut.

Proof. The proof of correctness of this algorithm is by induction on the steps of the algorithm, by showing that at each step, the edges that the algorithm chooses, definitely belong to the maxcut. The proof is based on stability. Suppose that the algorithm fails for the first time in some step. I.e. it chooses to connect the smallest c with some other c' in the heaviest possible way (let W be the weight of the edges of this choice), so that these edges do not belong to the max cut. Then the edges which would result if the algorithm connected c with c' in the other way, should necessarily belong to the cut.

Let S be the optimal cut and T the cut resulting if we change sides to the vertices of c. We will show that $A \leq \gamma B$, which contradicts stability, (where A is the weight of edges cut by S and not by T, and B the opposite).

Remember that in the step that the algorithm had chosen c, it had to choose between all other existing subgraphs (let J be their number), with which to join c and how. The two cuts C and T (from the inductive assumption and the construction of T) can differ only on the way that c is connected to each of these subgraphs. Obviously it holds

$$A \leq JW$$

since W was the heaviest possible connection. On the other hand

$$\gamma B \geq \gamma W$$

since as we said, T definitely contains these edges of weight W hat the algorithm chose, and we assumed that they do not belong to S.

It remains to see that $J \leq \gamma = \sqrt{\Delta n} \Leftrightarrow J^2 \leq \Delta n$. This holds because if k is the number of vertices of c, then obviously the number of the other existing components with which it could be connected is at most $k\Delta$. On the other hand, c was also the smallest (in number of vertices) component among all existing components, so their number was at most $\frac{n}{k}$. So $J \cdot J \leq \frac{n}{k} \cdot k\Delta = n\Delta$. $\square$

This algorithms requires very large stability. For example, take a complete graph whose maximum cut consists of two sets of equal size. How would it look like if it was $\sqrt{\Delta n} \simeq n$-stable? Suppose for example that there are 12 vertices separated in two groups of 6 and 6, and between vertices of the same group we have edges of weight μ, and between vertices of different groups we have edges of weight M. It turns out (e.g. by changing group of a single vertex, and computing the difference in the weights of the two cuts) that in order for the instance to be 12-stable, it should hold that $M \geq 10\mu$. This means that the two groups must be very far away from each other, compared to the inner distances of their vertices.

We can reduce γ needed to solve a γ-stable instance, by requiring more properties for the graph to hold, like to have a large minimal degree.

For every γ-stable unweighted graph, with minimal degree $\delta = \Omega(\frac{n}{\log n})$ and $\gamma \geq \frac{4n}{\delta} = O(\log n)$, we can contract vertices having too many common neighbors, because from local stability it turns out that they should belong to the same side of the max-cut. Finally a small graph occurs, in which we can compute exhaustingly the max-cut.

Spectral

The authors in [8] also analyze the spectral partitioning algorithm of Goemans-Williamson: The MAXCUT problem can be formulated as follows

$$\max_{y \in \{-1,1\}^n} \frac{1}{2} \sum_{(i,j) \in E} W_{i,j}(1 - y_i y_j)$$

or equivalently

$$\min_{y \in \{-1,1\}^n} \sum_{(i,j) \in E} W_{i,j} y_i y_j.$$

If we replace the condition $y \in \{-1,1\}^n$ with $y \in \mathbb{R}^n$, $\|y\|^2 = n$. So we have a simple eigenvalue problem (remember that for a symmetric matrix A, the smallest eigenvalue is non negative and equals $\lambda_1 = \min \frac{\mathbf{y}^T A\mathbf{y}}{\mathbf{y}^T\mathbf{y}}$ over all $\mathbf{y} \in \mathbb{R}^n$). The signs of the entries of the eigenvector with the smallest eigenvalue of W reveal the partitioning. The same solution we would take if instead of W we solved the eigenvalue problem on $W + D$ where D is diagonal, which means that we add self loops and so the max-cut of W doesn't change. If W is γ-stable, we have the following lemma.

Lemma 13. *[8] Let W be a γ-stable instance of MAXCUT. Let D be a diagonal matrix, and u an eigenvector corresponding to the least eigenvalue of $W + D$. If $\gamma \geq \frac{\max_{(i,j)\in E}|u_i u_j|}{\min_{(i,j)\in E}|u_i u_j|}$, then the spectral partitioning induced by $W + D$ yields the maximal cut.*

Proof. Without loss of generality, let $\min_{(i,j)\in E}|u_i u_j| = 1$, since we can normalize u. Why do the signs of the entries of u reflect the max-cut of W? Because u minimizes

$$\min_{y\in\mathbb{R}^n}\Big(\sum_{i,j} W_{i,j}y_i y_j + \sum_i D_{i,i}y_i^2\Big)/\|y\|^2$$

so its signs minimize

$$\min_{x\in\{-1,1\}^n}\Big(\sum_{i,j} W_{i,j}\cdot|u_i|x_i\cdot|u_j|x_j + \sum_i D_{i,i}u_i^2\Big)/\|u\|^2$$

which, if $\|u\|^2 = k$ (where $k \geq 1$ after normalization), is equivalent to

$$\min_{x\in\{-1,1\}^n} x(W' + D')x$$

i.e. the sign vector x is the max-cut of $W' + D'$, where $W'_{i,j} = \frac{1}{k}W_{i,j}\cdot|u_i u_j|$ and $D'_{i,i} = \frac{1}{k}D_{i,i}\cdot u_i^2$. So x is the max-cut of W too, since W' is a $\frac{\max|u_i u_j|}{k}$-perturbation of W, which additionally has stability $\gamma \geq \frac{\max|u_i u_j|}{k}$. □

So given γ it suffices to find a D s.t. for the eigenvector u corresponding to the smallest eigenvalue of $W+D$, to hold $\gamma \geq \frac{\max_{(i,j)\in E}|u_i u_j|}{\min_{(i,j)\in E}|u_i u_j|}$. After that we perform spectral partitioning on $W+D$.

Relative to the search of such a D, one solution is the following. Let mc^* be the sign vector giving the max-cut of graph W.

For $D_{ii} = -mc^*\sum_j W_{ij}mc_j^*$ we observe that $(W + D)mc^* = 0$, that is mc^* has eigenvalue 0. If $W + D$ is positive semidefinite, then this will be its smallest eigenvalue. So if

(a) we could find D with some other method, since we don't know mc^*,

(b) $W + D$ was positive semidefinite,

(c) W had the stability required by the previous lemma

then the above extended spectral partitioning algorithm, solves effectively the MAXCUT problem.

The authors prove firstly that if $W + D$ is positive semidefinite, and sufficiently stable, then $W + D$ (and so D too) can be found with some method, and precisely by finding among all positive semidefinite matrices that agree with W in the non diagonal entries, the one with the minimum trace (using the ellipsoid's algorithm).

Secondly they find a sufficient condition for D, so that $W + D$ will be positive semidefinite and γ-stable, for γ sufficient for the spectral algorithm to work. We provide the corresponding lemmas.

Lemma 14. *[8] Let W be a γ-stable instance of Max-Cut, for $\gamma > 1$, and let D be the diagonal matrix $D_{ii} = -mc^* \sum_j W_{ij} mc^*_j$. If $W+D$ is positive semidefinite, then extended spectral partitioning solves Max-Cut efficiently.*

Proof. According to what we already said, it suffices to show how D (or $W + D$) can be found.

We observe that

$$\text{trace}(D) = w(E_{in-maxcut}) - w(E_{not-in-maxcut}) = 2w(E_{in-maxcut}) - w(E).$$

So to find the value of the max-cut it suffices to compute trace(D). The impressing is that the way the authors find the trace, gives the required matrix D too.

We will prove that $m = \min trace(A)$ over $A \in \mathbb{A}$, where $\mathbb{A}$ is the set of all positive semidefinite matrices with $A_{ij} = W_{ij}$ for $\neq j$. (By the way this is the dual of Goemans Williamson relaxation).

This minimum trace is $\leq m$, since as we assumed $W + D \in \mathbb{A}$ and has trace equal to m.

For the opposite we observe that $mc^* A mc^* = -m + trace(A)$, but since A is positive semidefinite, the first quantity is non negative.

The above problem can be solved ([...]) with the help of the theory about ellipsoid's algorithm ([...]).

The solution may be not unique, but with a small random perturbation on W can be unique (without affecting mc^*, due to stability), so equal to $W + D$. $\square$

In the next lemma we denote with $\bar{\delta}$ the least weighted degree (of a vertex) of W, i.e. if i is a vertex and $w(i) = \sum_j W_{ij}$, then $\bar{\delta} = \min_i\{w(i)\}$.

Lemma 15. *[8] Let W be a γ-locally stable instance of Max-Cut, with smallest weighted degree $\bar{\delta}$, spectrum $\lambda_1, \ldots, \lambda_n$, and D diagonal s.t. $D_{ii} = -mc^* \sum_j W_{ij} mc^*_j$, then $W + D$ is positive semi-definite if*

$$2\delta \frac{\gamma - 1}{\gamma + 1} + \lambda_n + \lambda_{n-1} > 0.$$

So if the above lemma is satisfied for an instance, then maxcut can be solved efficiently.

Finally some families of graphs are given, which satisfy the above lemma. They concern graphs with all weighted degrees equal to each other, simple d-regular graphs, simple d-regular expanders, and (k, γ)-distinct d-regular graphs. In each case the stability required for efficiently solving these instances is smaller than the stability required for the first combinatorial algorithm.

In [7] the authors continue the research on stable instances of MAXCUT, giving more results about distinguished, metric, expanding, and dense instances.

4.2.2 Local stability, and good clusters and clusterings

The assumption in [11] is that the data-set follows a probability distribution over a metric space. Clusters are searched without optimizing explicitly some function.

First of all, a good cluster and a good clustering are defined. A good cluster is one s.t. the points inside it are noticeably closer to it that the points outside it (with some notion of distance between point and cluster). A good clustering is one s.t. every point inside it is much closer to this than to any other cluster. Moreover, to avoid trivial cases, like if each cluster consists of a single point, it is required that the clusters are sufficiently big.

More specifically, we deal with points belonging to a metric space (X, d) equipped with some unknown probability distribution P over X. The distance between a point and a cluster depends on this distribution. So the existence of good clusters and good clustering depends on this distribution, along with the metric.

For some cases where good clusters and good clusterings exist, algorithms are given that compute them quickly. But note that the authors consider acceptable for an instance to have many good clustering, even if they are very different from each other.

Definition 16. *Let (X, d) be a metric space and P a probability distribution on X. Let $\Delta(x, A) = E_{y \sim P}[d(x, y) \mid y \in A]$, which is a distance function between a point x and a set A. Another choice could be $\Delta'(x, A) = \inf_{y \in A \setminus \{x\}} d(x, y)$. (Note that Δ depends on P, but we omit it for simplicity of notation).*

*A subset $C \subset X$ is an (α, γ)-**cluster** for $\alpha > 0$, $\gamma > 1$ if $P(C) \geq \alpha$ and for almost every $x \in C$, $y \notin C$*

$$\Delta(y, C) \geq \gamma \cdot \Delta(x, C).$$

*A partition $\mathcal{C} = \{C_1, \dots, C_k\}$ of X is an (α, γ)-**clustering** for $\alpha > 0$, $\gamma > 1$ if for every i $P(C_i) \geq \alpha$, and for every $i \neq j$ and almost every $x \in C_i$*

$$\Delta(x, C_j) \geq \gamma \cdot \Delta(x, C_i).$$

That means that a cluster is good, if outside it there exists a big gap (empty space without points).

Note also that an (α, γ)-clustering, and a partition to (α, γ)-clusters, are not the same thing. For example it may be the case that each point is closer to its own cluster that to others, but between the clusters there is not enough empty space.

Theorem 17. *[11] For every fixed $\gamma > 1$, $\alpha >> 0$, there is an algorithm that finds all (α, γ)-clusterings of a given finite metric space X, and runs in time poly$(|X|)$.*

Proof. (sketch)

The idea is that if α is big enough, then you can take a small sample S of points in X (logarithmically small), according to the distribution P, and this sample w.h.p. will be representative of X. So if you search all the partitions of the sample until you find one that is an (α, γ)-clustering on S, the its extension to X (its "Voronoi diagram"), w.h.p. will be very close (w.r.t. symmetric difference) to an (α, γ)-clustering on X. □

Theorem 18. *[11] There is a polynomial time algorithm that on input a finite metric space X and $\alpha > 0$ finds all $\gamma - clusters$ in X with $\gamma > 3$ and a partition of X into (α, γ)-clusters with $\gamma > 3$, provided one exists. Moreover the latter problem is NP-hard for $\gamma = 5/2$.*

Proof. (sketch).

The fact that the proof relies on, is that when $\gamma > 3$ then the $(\alpha, 3 + \epsilon)$-clusters have a very special structure: they are balls with center one of their points, and moreover, every two clusters are either disjoint, either the one contains completely the other. So a hierarchy of clusters is defined, with tree structure, that can be computed by dynamic programming. Note also that if we define as minimal (α, γ)-clusters those that do not contain a proper subset which is an (α, γ)-cluster too, then there exists only one partition to minimal (α, γ)-clusters, and it can be found quickly.

The Np-hardness for $\gamma = 5/2$ is shown by reduction from 3-D-matching. □

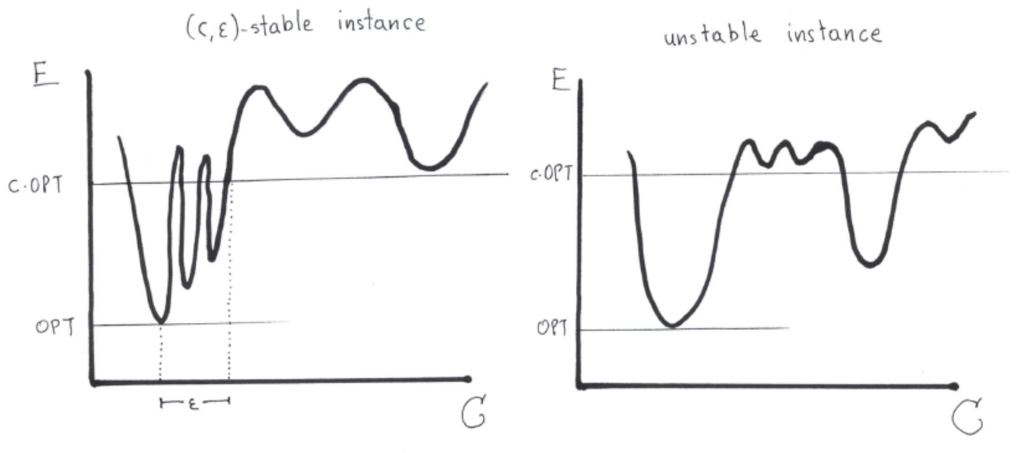


Figure 4.1: stable-unstable instances

Comments

We observe that here the authors don't minimize any function, so we cannot see the form of some corresponding energy landscape. But we can assume that according to their definitions, they subsume that they minimize the function $\sum_{x\in X} \Delta(x, C_x)$, where C_x is the cluster in which x belongs, and where the minimum is taken over all clusterings with $P(C_i) \geq \alpha, \forall i$.

So now the definition of (α, γ)-clusterings is analogue to the definition of γ-locally-stable instances of MAXCUT. I.e. the clustering is such that it is not beneficial to change any single point position (take it from the cluster it belongs and put it to another cluster).

So if we consider as neighborhood structure the one that every two solutions (clusterings) are neighbors iff they disagree to exactly one point, then the instances having an (α, γ)-clustering, have the property that each such clustering is a local minimum and is significantly distinct from its neighbors.

Moreover this holds for a wider neighborhood of every good clustering too, since it is proved in the paper that if there exist many (α, γ)-clusterings, then they differ very much from each other (in number of points).

Finally the authors observe that the (α, γ)-clusterings are stable in the sense that if we perturb a little the metric, then an (α, γ)-clustering remains (α', γ')-clustering for $\alpha' \approx \alpha$, $\gamma' \approx \gamma$. I.e. the good clusterings remain good after a small perturbation on the input.

4.3 Approximation Stability implies Strict Separation

Approximation stability is defined here: all clusterings C of minimum or nearly minimum energy are close to each other, and close to a target clustering (although it is possible that between them there are other clusterings with bigger energy, see figure 4.1).

4.3.1 Approximating Clustering without the approximation

Clustering is often done by optimizing some function. Usually this optimization is NP-hard, so approximate solutions suffice. In this way we presume that the optimum of the function, but also all its approximations, are "close" to the target clustering that we seek for. (Closeness is defined with respect to the fraction of data points in which two clusterings disagree). Moreover, only if this assumption holds for our data, it is meaningful to search for an approximate solution.

This property is strictly defined in [2] as follows.

Definition 19. *We define the distance $dist(\mathcal{C}, \mathcal{C}')$ between two clusterings $\mathcal{C}, \mathcal{C}'$ as the fraction of the points in which they disagree under the optimal matching of clusters in $\mathcal{C}$ to clusters in $\mathcal{C}'$. We say that clusterings $\mathcal{C}$ and $\mathcal{C}'$ are ϵ -close if $dist(\mathcal{C}, \mathcal{C}') < \epsilon$.*

Definition 20. *((c, ϵ)-stability) Given an objective function Φ (such as k-median, k-means, or min-sum), we say that instance $(\mathcal{M}, \mathcal{S})$ is (c, ϵ)-stable for Φ if all clusterings with $\Phi(\mathcal{C}) \leq c \cdot \mathrm{OPT}_\Phi$ are ϵ-close to the target clustering $\mathcal{C}_T$ for $(\mathcal{M}, \mathcal{S})$.*

For instances satisfying this property, the authors of [2] give an algorithm that finds a clustering close to the target. They accomplish that even in cased where the instance is stable for a c s.t. the objective function is hard to approximate within c. Nevertheless, for (c, ϵ)-stable instances, even for $c < c_0$,it is possible to find a solution close to the target, but this solution may not necessarily be a c approximation of the optimum.

The algorithm [2] For the function k-median $\equiv \sum_x \min_i d(x, c_i)$, where c_i the center of the i-th cluster, the algorithm has the following basic idea.

Suppose for simplicity, that the optimum coincides with the target. For each point x, let $w(x)$ be its distance from the center of the cluster it belongs to, $w_2(x)$ its distance from the second closest cluster-center, and $w_{avg} = avg_x w(x) = \mathrm{OPT}/n$.

(c, ϵ)-stability implies that for at most ϵn points we have that $w_2(x) - w(x) < (c-1)w_{avg}/\epsilon$. This means that the real clusters form areas dense and well separates (good points), while between them there exist at most ϵn ambiguous points, i.e. such that $w_2(x) - w(x) < (c-1)w_{avg}/\epsilon$ (bad points), and some (also bad) isolated points, i.e. such that $w(x) > d_{crit}$, for some d_{crit} that we will see later.

The algorithm first discovers the dense areas, defining a cluster for each one, then it put the isolated points into the closest cluster, and it puts the ambiguous anywhere, so we have at most ϵn incorrectly clustered points.

To discover the dense areas we do the following. We define a graph G (threshold graph) in which we connect with an edge every two points having distance $\leq 2d_{crit}$. The value d_{crit} must be s.t. the good points form cliques without common neighbors. So if we set $d_{crit} = (c-1)w_{avg}/5\epsilon$, the good points that belong to the same cluster, form a clique (because of $w(x) < d_{crit}$), and two good points in different clusters do not have common neighbors, (because if for example x, y are good points in different clusters with common neighbor z, then $w_2(x) < |x - z| + |z - y| + w(y) \leq 2d_{crit} + 2d_{crit} + d_{crit} = 5d_{crit}$, which is a contradiction since for the good points it holds that $w_2(x) \geq w(x) + (c-1)w_{avg}/\epsilon = w(x) + 5d_{crit}$).

It remains to discover the cliques. If their size is $\geq b + 2$ where b the number of bad points, then we define a graph H in which we connect two points with an edge, if in G they have more than b common neighbors, and we take as clusters the largest connected components of H.

If the size of the clusters is not so big, we can again have an error $O(\epsilon/(c-1))$ as follows. We find the vertex with the largest degree in H. We put in a new cluster this vertex and its neighbors, and we remove them from H. We repeat $k-1$ times, and in the k-th cluster we put what is left. We omit the proof for the error.

Finally, because in fact we don't know w_{avg}, in the first algorithm we can compute it by assuming initially that is 0, and increasing it gradually until we have cliques of size $\geq b+2$, and at most b bad points. For the general case, with small clusters, this cannot be applied, but we can approximate w_{avg} with a β-approximation algorithm, with the cost of multiplying by β the fraction of incorrectly clustered points, too.

Similar results hold for k-means and min-sum.

Hardness results for stable instances It is proved in [2] that the approximation of k-median with the assumption of (c, ϵ)-stability, is as hard as without it. But we should note that the proof constructs from the given instance a stable one, by adding n/ϵ points very far from each other and from the already existing. So the resulting instance belongs to the case of small clusters instances, so it is possible that the case of big clusters is not hard to approximate. Nevertheless, as we already said, the algorithms we presented can find a solution close to the target, even when the instance is stable for a small c, but not for larger. However it does not seem meaningful to cluster such an instance, since there exist solutions quite different from the optimum, but with comparable energy.

Another theorem in [2] states that it is NP-hard to find a clustering ϵ-close to the optimum, with only the assumption of $(1, \epsilon)$-stability. This means that it does not suffice all the minima to be close to each other, if there exist solutions of very low (but not minimal) energy very far from the target. Besides, as before, it wouldn't be meaningful to search for an approximate solution in this case, and maybe it wouldn't be meaningful to search for any solution at all, for the same reason as before.

4.3.2 Strict separation is weaker than approximation stability

In [3] is shown that approximation stability is more restrictive than the strict separation property, which is the existence of compact and well separated clusters, with at most ϵn ambiguous points between them, even when we have $\epsilon = 0$ ambiguous points.

However it is possible to construct a hierarchy of clusterings using single linkage algorithm (Kruskal's algorithm), s.t. the target clustering is a pruning of this tree.

Moreover it is shown that all good clusterings are at most $2^{\Theta(k)}$, where $k < \frac{1}{2\epsilon}$.

4.4 Robustness

Robustness of X is defined as follows: the energy E should have a very small slope (w.r.t. C) near the minimum, or stated differently, the energy doesn't change much after a small perturbation on C. (No unique minimum is required).

In [4] the authors, like the previous ones, observe that many NP-hard problems can be solved quickly in instances arising in practice. They suggest that maybe this happens because in the space of instances, the hard ones are sparse. They define a measure of "naturality" of inputs, based on the intuition that a natural instance is robust in the sense that small changes on the optimal solution do not cause big changes to the objective function.

Let $\Pi : \mathcal{I} \times \mathcal{T} \to \mathbb{R}$ be the objective function (of gain or loss) of the optimization problem, where $\mathcal{I}$ is the space of inputs, $\mathcal{T}$ is the space of solutions, and let $d : \mathcal{T} \times \mathcal{T} \to \mathbb{R}^+$ be a distance function on the space of solutions. Then

Definition 21.

- *For $I \in \mathcal{I}, T \in \mathcal{T}$*

$$\lambda_{(I,T)} = \sup_{T' \in \mathcal{T}} \frac{\Pi(I,T) - \Pi(I,T')}{\Pi(I,T)d(T,T')}.$$

  I.e. $\lambda_{(I,T)}$ measures the rate by which the gain function changes (normalized by its value at T) as $d(T,T')$ grows (in other words, the slope of the gain function of I around T).

- *For $\rho > 0$ let*

$$\lambda^{\rho}_{(I,T)} = \sup_{T' \in \mathcal{T}, d(T,T') \leq \rho} \frac{\Pi(I,T) - \Pi(I,T')}{\Pi(I,T)d(T,T')}.$$

  The only difference here is that we only care about the slope of the gain function in the ρ-neighborhood of T.

- *An algorithm A is a ρ-approximation for an optimization problem $\mathcal{P}$ w.r.t. a proximity measure d, if for every input I*

$$\Pi(I, A(I)) \geq \sup_{T \in \mathcal{T}} \Pi(I,T)(1 - \rho\lambda^{\rho}_{(I,T)})$$

  Note that the approximation improves as ρ shrinks.

The difference from the usual definition of approximation, is that the error that we allow depends on how robust is the input (i.e. how sharp the extrema of the objective function are). The more robust a solution is (i.e. less sharp), the better is the approximation factor that we get. Even if the optimum is non-robust, but there exists another solution almost optimal and robust, the approximation factor improves (e.g. figure 4.2).

The authors also define the problems 'Best Separating Hyperplane' and 'Densest Open Ball', for which polynomial time ρ-approximation algorithms exist.

Lemma 22. *[4] For every combinatorial optimization maximization problem $\mathcal{P} = (\mathcal{I}, \mathcal{T}, \Pi)$ with a metric d over the space of its solutions, and or every solution $T \in \mathcal{T}$ and $\rho > 0$*

$$\sup_{T \in \mathcal{T}} \inf_{T' \in B(T,\rho)} \Pi(I,T') \geq \sup_{T \in \mathcal{T}} \Pi(I,T)(1 - \rho\lambda^{\rho}_{(I,T)})$$

Theorem 23. *[4] For every $\rho > 0$, the Best Separating Hyperplane problem, as well as the Densest Open Ball problem, have polynomial time ρ-approximation algorithms that find solutions satisfying*

$$\Pi(I, A(I)) \geq \sup_{T \in \mathcal{T}} \inf_{T' \in B(T,\rho)} \Pi(I,T')$$

which means that the solution found by the algorithm can either be close (w.r.t. the metric d) to the optimal, either close to some solution not optimal but more robust (see first figure 4.3), or not even close to any robust solution (see second figure 4.3).

So we have algorithms that find an approximation as better as more robust the input is. But robustness here does not mean existence of a unique optimal solution, not even means that all optimal and near optimal solution should be close to each other.

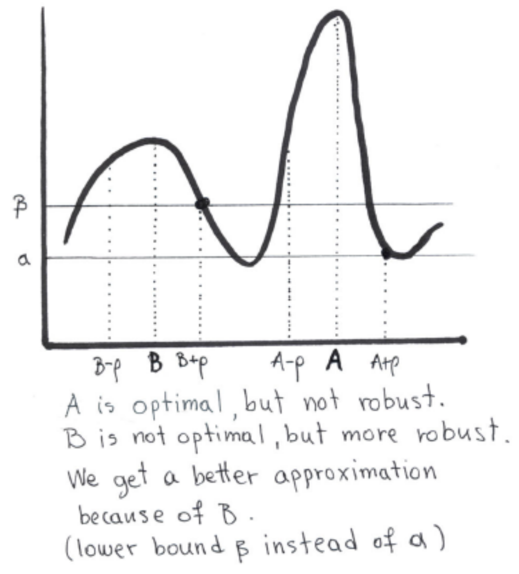

β
α
B-ρ
B
B+ρ
A-ρ
A
A+ρ
A is optimal, but not robust.
B is not optimal, but more robust.
We get a better approximation
because of B.
(lower bound β instead of α)

Figure 4.2: robustness due to a non-optimal solution

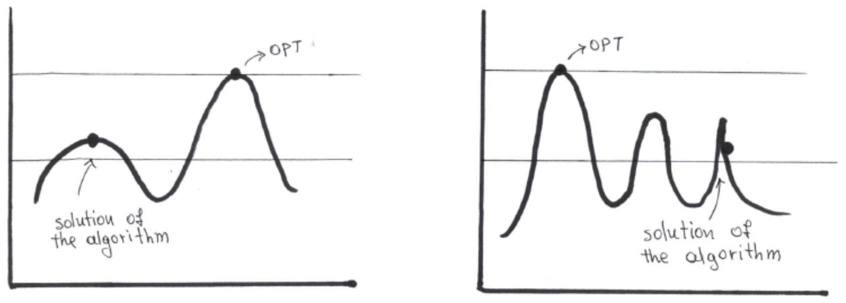


Figure 4.3: possible solutions of algorithms for robust instances

Comment on robustness v.s. edge / vertex distinction Note that robustness seems to be the opposite of edge / vertex distinction, since the first requires small slope near the optimum, while the second requires big slope. What is important is the measure of similarity between clusterings, or the definition of their neighborhood structure. When two clusterings are considered close to each other when the centers of their clusters are close to each other, then for a good clustering the objective function should not change significantly in a neighborhood of it (robustness). On the other hand if clustering similarity is with respect to the number of disagreeing points (hamming distance), i.e. points clustered differently in the two solutions, then a good clustering may change significantly when we move a single point from its cluster to another one, which results in big slope near the optimum (edge / vertex distinction).

4.5 Other notions of clusterability

In [1], the authors examine whether a given data set admits a good clustering. They show that for several notions, when the data have clusterable structure, then the optimal or a near-optimal clustering can be computed quickly. The proofs and the algorithms are different and depend on what definition of clustering is used, namely the objective function that we want to optimize.
They also show that determining whether a dataset has such structure is NP-hard.
They also show that the various definitions of clustering are not equivalent, meaning that there are datasets that are well clusterable under one definition, but not under another.

Definitions

Clusterability can be defined in two ways. One is to consider the function that we want to optimize, and define as clusterability of a set X the optimal value of that function. The following are mentioned:

- worst pair ratio clusterability, with function $WPR(C, X) = \frac{split_c(X)}{width_c(X)}$, where $split_c(X)$ is the minimum distance between two clusters, and $width_c(X)$ is the maximum distance between two points of a cluster. Clusterability of X (for k clusters) is defined as

$$\max\{WPR(C, X) \mid C \text{ is a } k\text{-clustering of } X\}.$$

- variance ratio clusterability, with function $VR(C, X) = \frac{B_C(X)}{W_C(X)}$, where $B_C(X)$ is the variance of the centers of the clusters, and $W_C(X)$ is the average variance of the points inside the clusters. Clusterability of X (for k clusters) is defined as

$$\max\{VR(C, X) \mid C \text{ is a } k\text{-clustering of } X\}.$$

- strict separation clusterability, with function $StrS(C, X) = \max_{x \in X} \frac{\max_{x \sim y} \|x - y\|}{\min_{x \not\sim y} \|x - y\|}$. Clusterability of X is defined as

$$\min\{StrS(C, X) \mid C \text{ is a clustering of } X\}.$$

The second way to define the clusterability of a set X, is through the behavior of the optimization function, near the optimal solution.

- center perturbation clusterability: A dataset X is (ϵ, δ)-CP clusterable for k, if for every clustering C of X that is ϵ-close to some optimal k-clustering of X, $\mathcal{L}(C) \leq (1+\delta)\,\mathrm{OPT}_{\mathcal{L},k}(X)$. $\mathcal{L}$ is the cost function, and ϵ-close refers to moving the centers of the clusters by at most ϵ. So we have clusterability, when after a small perturbation of the centers, the value of the function remains near the optimal value.

- separability clusterability: A dataset X is (k, ϵ)-separable if $\mathrm{OPT}_k(X) \leq \epsilon \cdot \mathrm{OPT}_{k-1}(X)$, where k is the number of clusters. That is, X is k-separable if the cost function improves significantly when we increase the number of clusters by one. (An algorithm is provided only for the case where the cost function is that of k-means).

- target clusterability: A dataset X is $(\mathcal{L}, c, \epsilon)$-clusterable if any near-optimal k-clustering C, where $\mathcal{L}(C) \leq c \cdot \mathrm{OPT}_{\mathcal{L},k}(X)$, has error at most ϵ with respect to the target clustering. That is, X is clusterable when all the low cost solutions are near (using the Hamming distance) to the target (which may be not the minimum cost solution, but has to be near to it). See subsection 4.3.1 for more details on this.

Results

For all these cases there are algorithms [1] for datasets that are clusterable enough.

For the worst pair ratio, if it is > 1 we can determine its exact value, by finding the optimal clustering using the single linkage algorithm, and then computing the worst pair ratio of that clustering.

The variance ratio is optimized by the clustering that optimizes the k-means loss. This happens because $\sigma^2(X) = W_C(X) + B_C(X) \Rightarrow VR_k(X) = \frac{\sigma^2(X)}{W_C(X)} - 1$. For 2-clustering there is a probabilistic approximation algorithm for k-means, and so for the variance ratio as well. The approximation depends on $VR_2(X)$. Computing the exact variance ratio is NP-hard for every k. This happens because k-means (which is NP-hard) reduces to variance ratio.

For $StrS(X)$ we have the following theorem [1]. If $StrS(X) \geq 1$, then we can efficiently construct a tree such that every clustering C for which $StrS(C, X) \geq 1$ is a pruning of this tree. Αυτό δεν το καταλαβαίνω, μήπως εννοεί ≤ 1 ; Επίσης δεν δίνεται κάποιο αποτέλεσμα σχετικά με τη δυσκολία προσδιορισμού του βαθμού clusterability.

For center perturbation clusterability they give an approximation algorithm.

For separability clusterability there is an approximation probabilistic algorithm when the function is the k-means. They also show that determining how separable a set is, is NP-hard by a reduction from k-means.

For target clusterability there is an algorithm that finds a clustering near the target (using the Hamming distance). However, it may be of very high cost.

Comments

Generally, what the authors do here, is looking at various algorithms for clustering and finding what property of the input would guarantee that the algorithm works correctly.

The question that arises (which we try to partially answer) is the following. What happens if the input doesn't have this property? Since we don't know whether it does, and for some cases it says that it is NP-hard to determine whether it does.

Furthermore, note that all these definitions of clusterability, don't make impossible the existence of many distinct optimal solutions (except for the case of target clusterability in which it is assumed).

4.6 Stochastic models

In stochastic models [13] [12] [25] it is supposed that there exists some planted clustering with noise. When the noise is sufficiently small, then we can recover it, or at least approximate it.

We can make the following interesting observation. In planted partition, when reconstruction is possible it is also easy (see the diagrams of the experimental results in subsection 4.6.2, and the information theoretical bounds for correlation clustering, which is a special case of planted block model, in subsection 6.8.3). In coloring, in contrast, we have a region of noise levels where reconstruction is possible but hard. So we conjecture that hardness in partitioning is due to the existence of several different good solutions.

4.6.1 Correlation Clustering with Noisy Input

Claire Mathieu, Warren Schudy, SODA 2010

In [25] it supposed that we are given a matrix that tells whether two data points are the same or not. But there is noise applied on them, so the relation is not transitive any more. The objective goal is to find a clustering that violates as few as possible entries of this given matrix.

The problem is NP-hard, and in fact hard to approximate within a constant.

But for the case that we suppose that the instance arose from some base clustering, and after the application of noise, if the probability of noise is relatively small ($p \leq 1/2 - n^{-1/3}$), an approximation algorithm is given. Also for the case that the noise is small ($p \leq 1/3$) and the clusters big, we have an algorithm that recovers the base clustering. Finally if the noise is very small ($p \leq n^{-\delta}/60$), an algorithm is given that finds all big clusters (bigger than $3150/\delta$).

Moreover, these algorithms are average case, so they succeed with high probability (i.e. for all inputs, but not for all). For that reason some certificates are given, to certify for each particular input, if the algorithm succeeded or not. But it is assumed that the instance is generated in this particular way (i.e. in the calculation of the probability of success, it is assumed the probability distribution over the instances which is defined by the corresponding noise model).

The model

The input is generated as follows.

Semi-random model. We start by some arbitrary clustering (ground or base or planted clustering), so we get a graph B, consisted of cliques. Then, the relation of each pair is changed independently with probability p, that is, if there were an edge between them, it is removed, and if there were not, it is included. Finally an adversary selects among all changed pairs (and only them) if he will keep the changes or if he will restore the initial statue.

Fully random model. Here the adversary keeps all the changes caused by noise, i.e. it is like there were no adversary in this model.

Algorithms

SDP with rounding

Let M, N be symmetric $n \times n$ matrices. Let $d(M,N) = \frac{1}{2}\sum_{i,j} |M_{i,j} - N_{i,j}|$. Observe that if M, N are the adjacency matrices of two simple undirected graphs, then d is their Hamming distance, i.e. the number of disagreeing edges.

The correlation clustering problem is, given a graph G, to find between all graphs corresponding to clusterings, one with minimum distance d from G.

The basic algorithm is based on the solution of the following SDP, for a given graph $G = (V, F)$.

$$\min d(X,F) \text{ s.t. } \begin{cases} X_{ii} = 1 & \forall i \\ X_{ij} \geq 0 & \forall i \neq j \\ X_{ij} + X_{jk} - X_{ik} \leq 1 & \forall i \neq j \neq k \\ X \text{ pos. semi-definite} & \end{cases}$$

Observe that if instead of X psd, we searched for an X s.t. $X_{ij} \in \{0,1\}$, then the constraints restrict us to adjacency matrices corresponding to graphs with disjoint cliques (i.e. clusterings).

The rounding of solution X^* resulting from the SDP, is done as follows. We get a random vertex u from V, put it in a new cluster A and we put in the same cluster each other vertex v with probability X^*_{uv}. We set $V := V \setminus A$ and repeat the same procedure in V until it gets empty.

Main Algorithm

We remove edges not belonging to triangles, solve the SDP with rounding, and then we find a maximum matching on the isolated vertices, taking into account the edges that we had removed in the beginning.

Theorems

Theorem 24. *In the semi-random model, when the noise is $p \leq 1/2 - n^{-1/3}$, the main algorithm finds a $1 + O(n^{-1/6})$-approximate solution.*

Theorem 25. *When the noise is $p \leq 1/3$ and the size of the clusters at least $c_1\sqrt{n}$, then w.h.p. the SDP solution is exactly the planted clustering.*

Proof. The proof is based on SDP duality. □

The space of solutions The space of psd matrices P_n is a convex closed cone in $\mathbb{R}^{n \times n}$, but its border is not smooth neither polyhedral. Every matrix of rank less than $n - 1$ (like matrices corresponding to clusterings) is on the boundary, but the boundary is not differentiable in that point.

The solution space of a semidefinite program is a (so called) spectrahedron. It is of the form $K = P_n \cap A$, where P_n is the semidefinite cone, and A is an affine subspace of symmetric matrices (formed by the constraints of the particular problem). We want to minimize a linear function $C^T X$. K is convex, but the minimum is on the boundary, which is not smooth nor polyhedral, so gradient type methods fail, and linear programming methods fail too.

The idea is to approximate the optimum by optimizing in the interior. This is done with the help of $F(Y) = -\log\det(Y)$ which is convex and analytic in the interior of P_n, and tends to ∞

on the boundary. So instead of minimizing $C^T X$, we minimize $F_{\lambda,C}(X) = F(X) + \lambda C^T X$ for some $\lambda > 0$ (which is convex and analytic in the interior too, and infinite on the boundary). As we increase λ we get a better approximation.

Optimization can also be performed in polynomial time exactly by the ellipsoid method [15], but this method is in practice inefficient. It doesn't use intuition about the geometry of the space of solutions.

See [23] for more on semidefinite programming.

4.6.2 The stochastic block model

In [13] [12] is studied the problem of reconstructing the planted partition in the stochastic block model.

It is like the previous model, with the difference that the noise on ones may be different from the noise on zeros in the given matrix.

There is no objective function to optimize, but rather we look for a partition with the maximum possible overlap with the planted partition.

The model Consider the following random experiment. We have n nodes, q labels, n_a a probability distribution over $\{1, \ldots, q\}$, and numbers $0 \leq p_{a,b} \leq 1, \forall a, b \in \{1, \ldots, q\}$. We assign a group label to each node independently, with probability $n_a, \forall a \in \{1, \ldots, q\}$. This process defines a partition of the nodes, which we will call planted partition. Finally we create a graph as following. We add an edge between any two nodes with probability $p_{a,b}$, where a, b are the labels of the two nodes.

The questions

- *Parameter learning:* Given the graph G, what are the most likely parameters $q, \{n_a\}, \{p_{a,b}\}$ that generated it?

- *Inferring the group assignment:* Given the graph G and the parameters $q, \{n_a\}, \{p_{a,b}\}$, what is the most likely assignment to a given node, and how much better is the most likely assignment than a random guess?

  (αν είναι σαν random guess σημαίνει ότι δεν μπόρεσα να ανακτήσω την πληροφορία, σωστά;)

  In particular, the objective of the second question is to find the group assignment $\{q_i\}$ that maximizes the overlap with the original assignment $\{t_i\}$, which is defined as follows

$$Q(\{t_i\}, \{q_i\}) = \max_{\pi} \frac{\frac{1}{N} \sum_i \delta_{t_i, \pi(q_i)} - \max_a n_a}{1 - \max_a n_a}$$

  where π ranges over all the permutations on the q labels.

Comment In [12] [13] the authors give precise answers to the above questions, for sparse networks, i.e. when $n_a = O(1)$ and $c_{a,b} = N p_{a,b} = O(1)$, which means that the edges grow as $O(N)$ instead of $O(N^2)$ when $N \to \infty$. For more dense networks (i.e. when $c_{a,b}$ diverges), the problem is algorithmically easier as shown in [.....].

Inferring the group assignment

Given the parameters $\theta = \{q, \{n_a\}, \{p_{a,b}\}\}$ we create (a) the group assignment $\{q_i\}$, (b) the graph G with adjacency matrix A. We will suppose that we know the parameters θ and the graph G, and given them we will try to infer the group assignment.

The probability distribution over all $G, \{q_i\}$ given θ, according to the process described in the beginning, is

$$P(G, \{q_i\} \mid \theta) = \prod_{i \neq j} \left[p_{q_i,q_j}^{A_{ij}} (1 - p_{q_i,q_j})^{1-A_{ij}} \right] \prod_i n_{q_i}$$

Using Bayes' rule $P(A \mid B) = \frac{P(A,B)}{P(B)}$, and the relation

$$P(A) = \sum_i P(A \mid B = b_i) P(B = b_i) = \sum_i P(A, B = b_i),$$

we get

$$P(\{q_i\} \mid G, \theta) = \frac{P(G, \{q_i\} \mid \theta)}{P(G \mid \theta)} = \frac{P(G, \{q_i\} \mid \theta)}{\sum_{\{t_i\}} P(G, \{t_i\} \mid \theta)}.$$

We will refer to this distribution as the *Boltzmann distribution*, for reasons that we will explain later in the comments.

To find the estimate $\{q_i\}$ of the original group assignment, we compute the marginals of the above distribution for each node, and we assign to this node the label with the maximum probability according to the corresponding marginal. This estimator is called *maximum posterior marginal*, and it is proved in [...] that this is the optimal estimator of the original assignment (in terms of maximizing the overlap), and not the one with the maximum probability according to the Boltzmann distribution.

Comments Note that we can write $P(G, \{q_i\} \mid \theta)$ as $e^{-\log P(G, \{q_i\} \mid \theta)}$.

The quantity $H_1(\{q_i\} \mid G, \theta) \equiv -\log P(G, \{q_i\} \mid \theta)$, and even better the extensive (i.e. proportional to N) quantity

$$H(\{q_i\} \mid G, \theta) \equiv -\log P(G, \{q_i\} \mid \theta) - M \log N =$$
$$-\sum_i \log n_{q_i} - \sum_{i \neq j} \left[A_{ij} \log c_{q_i,q_j} + (1 - A_{ij}) \log(1 - \frac{c_{q_i,q_j}}{N}) \right]$$

is like the Hamiltonian (i.e. energy function) of a generalized Potts model (which is a model of many interacting particles known from statistical physics). The corresponding Boltzmann distribution in unit temperature is

$$\mu(\{q_i\} \mid G, \theta) \equiv \frac{e^{-H(\{q_i\} \mid G, \theta)}}{Z(G, \theta)} \equiv \frac{e^{-H(\{q_i\} \mid G, \theta)}}{\sum_{\{q_i\}} e^{-H(\{q_i\} \mid G, \theta)}}$$
$$= \frac{P(G, \{q_i\} \mid \theta) \cdot N^M}{\sum_{\{q_i\}} (P(G, \{q_i\} \mid \theta) \cdot N^M)} = P(\{q_i\} \mid G, \theta)$$

since N^M is simplified.

The quantity

$$Z(G,\theta) \equiv \sum_{\{q_i\}} e^{-H(\{q_i\}|G,\theta)} = \sum_{\{q_i\}} P(G,\{q_i\} \mid \theta) \cdot e^{M \log N}$$

is called *partition function* in statistical physics, and it is a normalizing factor for the probabilities $e^{-H(\cdot)}$.

It is also defined the *free energy density*

$$\frac{F_N(G,\theta)}{N} \equiv -\frac{\log Z(G,\theta)}{N}$$

which has a well defined limit $f(G,\theta)$ as $N \to \infty$, since the energy is extensive.

Parameter learning

To infer the group assignment, we supposed that we knew the right values of the parameters θ. Now we are going to find the most likely parameters, given the graph G. It holds

$$P(\theta \mid G) = \frac{P(\theta)}{P(G)} P(G \mid \theta) = \frac{P(\theta)}{P(G)} \sum_{\{q_i\}} P(G,\{q_i\} \mid \theta)$$

where $P(\theta), P(G)$ are uniform, since we don't assume any prior knowledge about the parameters or the graph.

So in order to find the most likely parameters, we have to maximize $P(\theta \mid G) \Rightarrow$ it suffices to maximize $P(G \mid \theta) = \sum_{\{q_i\}} P(G,\{q_i\} \mid \theta) \Rightarrow$ it suffices to maximize $Z(G,\theta) = \sum_{\{q_i\}} P(G,\{q_i\} \mid \theta) \cdot N^M$ w.r.t. θ, $\Rightarrow$ it suffices to minimize the *free energy density* $f_G(\theta) \equiv -\frac{\log Z(G,\theta)}{N}$.

But we cannot minimize the free energy density simply by setting to zero the first partial derivative of $f(\theta)$ with respect to each parameter $n_a, c_{ab} \forall a,b \in \{1,\ldots,q\}$, because we have the extra condition that $\sum_a n_a = 1$. Since the free energy density, and the condition, have continuous first partial derivatives w.r.t. $n_a, c_{ab} \forall a,b$, we can find the minimum with the Lagrange Multipliers method, and so we get

$$n_a = \frac{\langle N_a \rangle}{N}, \forall a = 1,\ldots,q$$

$$c_{ab} = \frac{\langle M_{ab} \rangle}{N n_a n_b}, \forall a,b$$

where $\langle \cdot \rangle$ is the expectation with respect to the Boltzmann distribution.

We exploit an expectation-maximization method, i.e. (a) we start with some initial values of the parameters, (b) we compute $\langle N_a \rangle, \langle M_{ab} \rangle$, (c) we redefine n_a, c_{ab} with the new values as they result from the above equations, and we iterate (b),(c) until we reach a fixed point.

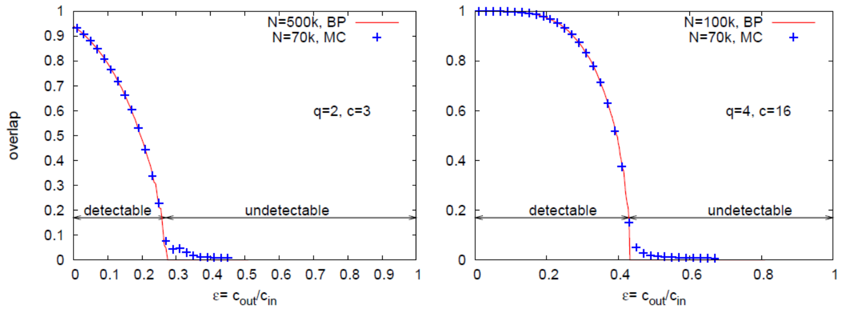


Figure 4.4: planted partitioning with $q = 2$ and $q = 4$

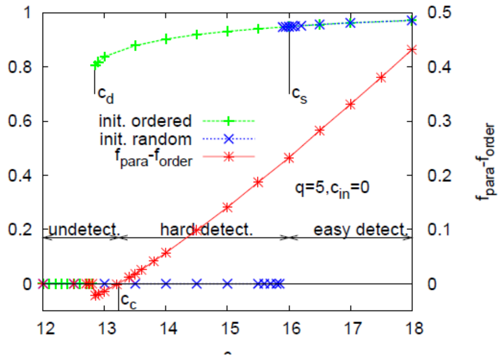


Figure 4.5: planted coloring with $q = 5$

Comments (a) To determine the number of groups q we can run the algorithm with several values until the free energy (as computed by BP) no longer decreases.

(b) Note that the theorem behind the Lagrange multiplier method only gives a necessary, but not sufficient condition for the maximum / minimum. However there are also sufficient conditions (something like the second derivatives conditions).

Belief Propagation for computing the marginals, and the expectations w.r.t. the Boltzmann distribution

The task of computing marginals and expectations w.r.t. the Boltzmann distribution is hard in general. We can use Gibbs sampling, or better Belief Propagation to approximate them in linear time. Linear time suffices for BP, because the graphs generated with this model are w.h.p. locally tree-like.

See chapter 5 for a description of these algorithms.

Experimental results

We give the diagrams with the experimental results, as appeared in [13], in figures 4.4 and 4.5.

The first two diagrams (4.5) refer to the planted partition model with $q = 2$ and $q = 4$ respectively. They give the overlap of the resulting partition with the original, as we vary the value

$c = c_{out}/c_{in}$.

As we expected, when c tends to 1, the graph tends to be completely random, and so inferring the parameters and the group assignments is impossible.

Note that if we tried to minimize the Hamiltonian directly, i.e. to find the most probable assignment, we could detect groups that would not exist (besides that finding the minimum may be hard even in cases where minimizing the free energy is easy).

We observe a second order transition between two phases, an ordered one, where learning the parameters is possible and the planted partition is detectable, and a paramagnetic phase where learning is impossible. The diagrams give the overlap of the resulting partition with the original, as we vary the value $c = c_{out}/c_{in}$.

In the ordered phase the free energy has an attractive global minimum at the true values of the parameters, and they are rapidly inferred by BP.

In the paramagnetic phase the free energy is constant around the true values of the parameters, and also equal to the free energy that would correspond to a completely random graph of the same average degree, so learning is impossible. The marginals computed by BP are $1/q$ for all the q groups, so there is no information about the original partitioning, (the overlap of course is zero).

Inference becomes easy when the paramagnetic fixed point is not locally stable, and the original group assignment is dynamically attractive. So we can compute analytically the point where this happens, if we analyze how a small perturbation of the paramagnetic fixed point propagates with BP. It follows that the detectable phase is when

$$|c_{in} - c_{out}| > q\sqrt{c}.$$

In the planted coloring model, between those two phases, we observe a third one, where the two phases coexist. BP converges to one of the two fixed points of the free energy, depending on the initialization. All but an exponentially small set of initial values lead to the paramagnetic fixed point (blue x's in the diagram 4.5). If the initialization is close to the true group assignments, BP converges to the ordered fixed point (green crosses). These fixed points don't have equal values of the free energy. (See red stars in the diagram).

This is a typical situation of (the so called) mean-field first order phase transition. It is proved [...] that finding the lower free energy phase is very hard in these cases. So, speaking for the stochastic block model, in this case inference is possible, but hard.

The condition $|c_{in} - c_{out}| > q\sqrt{c}$ refers to the transition to the phase where inference is possible and easy (i.e. point c_s in the diagram). In principle, the phase transition from detectable to undetectable phase happens when the ordered fixed point has larger free energy than the paramagnetic one (i.e. point c_c in the diagram).

4.6.3 Planted coloring

Although coloring is a different problem from clustering, it can be viewed as a clustering problem where the given information is about which data should *not* be in the same cluster.

We present here some results of [10] about a planted coloring model, not because it is relevant to clustering, but mainly for the method used, and its analysis, which is related to spectral methods, and could be used for clustering too.

In fact, the results in [10] do not concern only the planted model, but every graph with eigenvalues (of the adjacency matrix) that have a particular form that reflects some existing coloring.

If we consider especially the planted model, it is similar, but not equivalent, to a special case of the stochastic block model, where the probability to include an edge between two nodes of the same group, is 0.

The model

Let $G_{n,d,3}$ be the random graph with vertex set $V = \{1, \ldots, 3n\}$ obtained as follows.

i. Let $V_!, V_2, V_3$ be a random partition of V into three pairwise disjoint sets of equal size.

ii. For any pair $1 < i < j < 3$ independently choose a d-regular bipartite graph with vertex set $V_i \cup V_j$ uniformly at random.

It is proved that $G_{n,d,3}$ w.h.p. can be colored by the BPcol algorithm (which we present later). The proof is a corollary of a more general theorem, stating that BPcol can color graphs that their adjacency matrix has eigenvalues and eigenvectors of a particular form. Such graphs we will call (d, ϵ)-regular. $G_{n,d,3}$ is w.h.p. $(d, 0.01)$-regular.

Definition 26. *We say that a graph $G = (V, E)$ on n vertices is (d, ϵ)-regular if there exists a 3-coloring of G with color classes V_1, V_2, V_3 such that the following is true. Let $\overrightarrow{1}_{V_i} \in \mathbb{R}^V$ be the vector whose entries equal 1 on coordinates $v \in V_i$, and 0 in all other coordinates; then*

R1. *for all $1 < i < j < 3$ the vector $\overrightarrow{1}_{V_i} - \overrightarrow{1}_{V_j}$ is an eigenvector of the adjacency matrix $A(G)$ with eigenvalue $-d$*

R2. *if $\xi \perp \overrightarrow{1}_{V_i}$ for all $i = 1, 2, 3$, then $\parallel A(G)\xi \parallel \leq \epsilon d \parallel \xi \parallel$.*

Theorem 27. *There exist constants $d_0, k > 0$ such that for each $d \geq d_0$ there is a number $n_0 = n_0(d)$ so that the following holds. If $G = (V, E)$ is a $(d, 0.01)$-regular graph on $n = |V| \geq n_0$ vertices, then with probability $\geq kn^{-1}$ over the coin tosses of the algorithm, BPcol(G) outputs a proper 3-coloring of G.*

Corollary 28. *Suppose that $d \geq d_0$ is fixed. W.h.p. a random graph $G = G_{n,d,3}$ has the following property: G is $(d, 0.01)$-regular, and so with probability $\geq kn^{-1}$ over the coin tosses of the algorithm, PBcol(G) outputs a proper 3-coloring of G.*

The algorithm

By the definition of (d, ϵ)-regular graph, it is obvious that it is possible to color such a graph, by computing two eigenvectors with eigenvalue $-d$.

The interesting thing is that it is proved that BPcol, in a way, computes these eigenvectors. We will present the algorithm and a short explanation about the above deduction. For the full proof we refer to [10].

Description of BPcol BPcol tries to compute in each node, the probability of each color (probability over all valid colorings), based on the relation of these probabilities if the graph was a tree.

More specifically, each node V sends to each of its neighbors, its probability $\eta^a_{v \to w}$ to take each color a, given the probability of colors of the other neighbors, except w. This probability equals the probability that none of these neighbors has the color a. So the relation that must be satisfied (because on a tree these probabilities are independent) is

$$\eta^a_{v\to w} = \frac{\prod_{u\in N(v)\setminus\{w\}} 1-\eta^a_{u\to v}}{\sum\limits_{b=1}^{3}\prod_{u\in N(v)\setminus\{w\}} 1-\eta^b_{u\to v}} \tag{4.1}$$

for each edge $\{u, w\}$ and each color $a \in \{1, 2, 3\}$.

The denominator is just a normalizing factor, so that $\sum_a \eta^a_{v\to w} = 1$.

The idea behind BPcol (like all message passing algorithms) is to apply this relation repeatedly, in graphs that are not trees, i.e.

$$\eta^a_{u\to w}(l+1) = \frac{\prod_{u\in N(u)\setminus\{w\}} 1-\eta^a_{u\to v}(l)}{\sum\limits_{b=1}^{3}\prod_{u\in N(u)\setminus\{w\}} 1-\eta^b_{u\to v}(l)} \tag{4.2}$$

starting from some rational initial values $\eta^a_{u\to v}(0)$, until the values converge somewhere (they may converge right, wrong, or not converge at all).

Of course, since the valid colorings are symmetric w.r.t. permutations of colors, the true probabilities are $\forall a \eta^a_{u\to v} = \frac{1}{3}$, which are a fixed point of the above recursion (4.2). However if we start with initial conditions not exactly, but close to $\frac{1}{3}$, then the algorithm converges to other values, which in some cases reveal a coloring. For example if $\chi : V \to \{1, 2, 3\}$, the vector

$$\eta^a_{u\to v} = \begin{cases} 1 & if\ \chi(u) = a \\ 0 & otherwise \end{cases}$$

is a fixed point of (4.2) and reveals the coloring χ.

Proof of theorem

Proof. The analysis of the algorithm approximates the non-linear operator (4.2), denoted $\mathcal{A}$ from now on, with a linear operator $\mathcal{L}$. After a (logarithmic) number of iterations, the messages $\eta^a_{u\to v}$ are dominated by the eigenvectors of $\mathcal{L}$, which is almost identical to the adjacency matrix of the given graph G, so they have similar eigenvectors. And as we already said, the eigenvalues of G reveal the coloring, when G is (d, ϵ)-regular. Moreover, from the property R2 of (d, ϵ)-regular graphs, the second eigenvalue is much smaller from the first ($\leq \epsilon d = 0.01d$), so we have quick convergence.

More specifically, consider the transformations

$$\Delta^a_{u\to w}(l) = \eta^a_{u\to w}(l) - \frac{1}{3} \tag{4.3}$$

and let $\mathcal{B} : \mathbb{R} \to \mathbb{R}$ be the following non linear operator

$$(\mathcal{B}\Gamma)^a_{u\to w} = -\frac{1}{3} + \frac{\prod_{u\in N(u)\setminus\{w\}} 1-\frac{3}{2}\Gamma^a_{u\to v}}{\sum_{b=1}^{3}\prod_{u\in N(u)\setminus\{w\}} 1-\frac{3}{2}\Gamma^b_{u\to v}(l)}, \quad \Gamma \in \mathbb{R}. \tag{4.4}$$

The relation (4.2) is equivalent to

$$\Delta(l+1) = \mathcal{B}\Delta(l). \tag{4.5}$$

When $\|\Delta(l)\|_\infty$ is small, the non linear operator $\mathcal{B}$ can be approximated by the total derivative in 0

$$\mathcal{B}'(\Gamma)^a_{u\to w} = -\frac{1}{2}\sum_{u\in N(u)\setminus\{w\}} \Gamma^a_{u\to v} + \frac{1}{6}\sum_{b=1}^{3}\sum_{u\in N(u)\setminus\{w\}} \Gamma^b_{u\to v} \tag{4.6}$$

which is linear.

Now let $\Xi(l)$ be the sequence $\Xi(0) = \Delta(0), \Xi(l) = \mathcal{B}'^l\Xi(0)$; the "linear approximation" of $\Delta(l)$. Since $\sum_{a=1}^3 \eta^a_{u\to v}(0) = 1$, we have that

$$\sum_{a=1}^{3} \Xi^a_{u\to v}(0) = \sum_{a=1}^{3} \Delta^a_{u\to v}(0) = 0, \text{ for all } (u,v) \in E.$$

Inductively it follows that $\forall l\ \sum_{b=1}^3 \sum_{u\in N(u)\setminus\{w\}} \Xi^b_{u\to w}(l) = 0$. So $\mathcal{B}'$can be simplified to $\mathcal{L} : \mathbb{R} \to \mathbb{R}$

$$(\mathcal{L}\Gamma)^a_{u\to w} = -\frac{1}{2}\sum_{u\in N(u)\setminus\{w\}} \Gamma^a_{u\to v}, \ \ (\{u,v\}\in E, a\in\{1,2,3\}) \tag{4.7}$$

satisfying

$$\Xi(l) = \mathcal{L}^l\Xi(0) = \mathcal{L}^l\Delta(0).$$

Now we just observe that in $\mathcal{L}$ if $\Gamma^a_{u\to v}$ do not depend on v (i.e. they were all the same Γ^a_u) and if moreover the sum was on all neighbors $N(u)$ (and not without $\{w\}$), the we would have the relation $(\mathcal{L}'\Gamma)^a_u = -\frac{1}{2}\sum_{i\in N(u)} \Gamma^a_i$, i.e. $(\mathcal{L})'$ would be exactly the adjacency matrix of G!

For the exact analysis of the algorithm, we refer to [...] in which first the sequence $\Xi(l)$ is analyzed through the eigenvalues and eigenvectors of $\mathcal{L}$, and then the error $\|\Xi(l) - \Delta(l)\|_\infty$ is bounded. □

Proof of corollary

Proof. It suffices to show that $G_{n,d,3}$ is (d,ϵ)-regular w.h.p.

Property R1 is easy to see, if we observe that by construction each vertex $u \in V_i$ has exactly d neighbors in each class $V_j \neq V_i$, so let $\zeta^i = A(G)\overrightarrow{1}_{V_i}$ where

$$\zeta^i_u = \sum_{w\in V_i} a_{uw} = \left\{ \begin{array}{ll} 0 & if\ u\in V_i \\ d & if\ u\notin V_i \end{array} \right., \text{ for all } u\in V.$$

We have that $\zeta^i = A(G)\overrightarrow{1}_{V_i} = d\sum_{j\neq i}\overrightarrow{1}_{V_j}$, so if $1 \le i < j \le 3$ then

$$A(G)(\overrightarrow{1}_{V_i} - \overrightarrow{1}_{V_j}) = -d(\overrightarrow{1}_{V_i} - \overrightarrow{1}_{V_j}).$$

The property R2 is proved with the use of lemmas and techniques for expanders and random regular graphs, and we omit it. □

Appendix Let A be a $n \times n$ adjacency matrix.

$$A\overrightarrow{1} = (\sum_i a_{1i}, \ldots, \sum_i a_{ni})^T$$

If it is δ-regular, the 1 is eigenvector with eigenvalue δ, and it is the largest eigenvalue (A/δ is ergodic, the uniform is stationary)

Another note. If we start L with initial values $\eta(0) = 1/3$, then $\Delta(0) = 0$ and of course, the all-zero vector is a fixed point of L, as the all-$(1/3)$ vector is a fixed point of A.

Chapter 5

Phase Transitions in discrete structures: An overview

The method we propose to determine the meaningfulness of clustering requires the study of the behavior of the objective function over all clusterings, so it is a problem with mathematical formulation similar to problems from statistical physics.

So a promising idea would be to use algorithms developed in this area. The study of this area, its problems, the algorithms used, phenomena and difficulties that appear, will give us an idea of what we could do, and what difficulties we will have to face.

We present a brief overview of this theory.

5.1 Phase Transitions

Let $f : \mathbb{R} \to \mathbb{R}$ be a function. We call *phase transition* a point θ_0 where f is non analytic.

Specifically we have a *first order* phase transition if f is discontinuous in θ_0, and *second order* if f is continuous, but the first derivative in θ_0 does not exist.

If $f(\theta)$ relates to the description of some physical system with some parameter θ (e.g. temperature), then a phase transition corresponds to some physical phenomenon.

For example some physical phenomena are the transition of water from solid state (ice) to liquid in $0^o C$, and the transition of some metal from ferromagnetic to antiferromagnetic phase in the Curie temperature.

The function associated with these phenomena is the so called partition function $Z(\theta)$.

5.2 The Boltzmann Distribution

We can consider a physical system as a set of particles (variables) that interact (constraints).

Let $v_i, i \in \{1, \ldots, N\}$ be the variables, and let them take values on a set A e.g. $\{+1, -1\}$.

Let σ denote a configuration of the system, that is an assignment of values to the variables.

Let $E(\sigma)$ be the *energy* (or *Hamiltonian*), which is a real valued objective function. It gives a real value to each configuration and it is described from the constraints.

The *Boltzmann Distribution* is a probability distribution over all configurations. It is defined as follows.

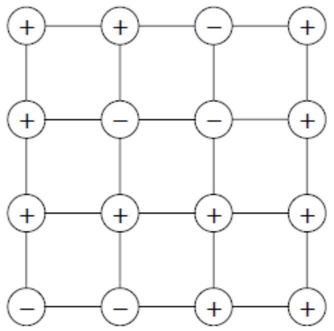
$$\mu_\beta(\sigma) = \frac{1}{Z_\beta} e^{-\beta E(\sigma)}$$

The normalizing factor Z_β is called the partition function.

Note that when $\beta \to 0$, then mu tends to be uniform over all configurations, and when $\beta \to \infty$ the μ tends to give all probability weight to the configurations of least energy. β is also called inverse temperature ($\beta = \frac{1}{k_B T}$, where k_B the Boltzmann constant).

Example. Water. When the temperature is low, the "crystallized" configurations dominate, while in higher temperatures the molecules have more freedom of movement (temperature corresponds to the mobility of the molecules), so we can find the system in many more configurations.

Example. Ising Model We have a grid where each node takes a spin ($\{+, -\}$).



The energy of the system is $E(\sigma) = -\sum_{i \sim j} \sigma_i \sigma_j$.

In low T we have an *ordered* phase, where all spins tend to be the same. In high T we have a *disordered* phase, where all configurations are almost equally probable.

5.3 The Partition Function

In the Boltzmann Distribution $\mu_\beta(\sigma) = \frac{1}{Z_\beta} e^{-\beta E(\sigma)}$, the normalizing factor Z_β is called the *partition function*.

$$Z_\beta = Z(\beta) = \sum_\sigma e^{-\beta E(\sigma)}$$

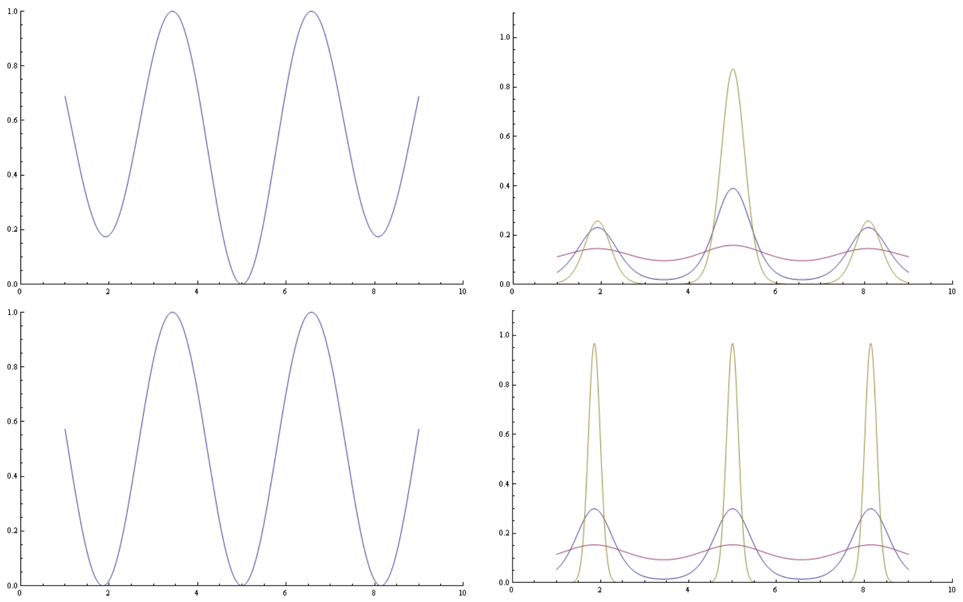
The *free energy density* is

$$F = -\frac{\ln Z}{N} \xrightarrow[N \to \infty]{} f.$$

The behavior of f with respect to β, describes the phenomenon.

For example in the Ising model in 1-dimension f is analytic everywhere, we have no phase transition, the transition from ordered to disordered phase happens smoothly. In the 2-dimensions Ising model we observe a phase transition (sudden magnetization).

Important Note. In each β there is an interval of energies that the corresponding configurations dominate in the Boltzmann distribution (see figures below, on the left there are two energy functions, on the right are the corresponding Boltzmann distributions for three different values of β). For small β (red lines) all configurations have almost the same probability, while for large β (yellow lines) minimum energy configurations dominate (have almost all the probability mass).



Z_β relates to **the number of dominating configurations**:

Let E_{dom} be the average energy of dominating configurations, and Dom the set of dominating configurations.

Let $\epsilon = e^{-\beta E_{dom}}$.

Then $Z_\beta \simeq \sum_{\sigma \in Dom} \epsilon \Rightarrow \frac{Z_\beta}{\epsilon} = |Dom|$.

Note We study f instead of Z because usually Z is exponential in N (E grows with N at least linearly), i.e. $Z = e^{-FN}$. Another reason is for numerical precision.

5.4 Methods from Statistical Physics

We present two kinds of algorithms used in statistical physics, to study physical systems.

The first kind are *Markov chains*, as special cases of the *Metropolis-Hastings* (meta-)algorithm, known as e.g. Glauber dynamics, Heat bath, Gibbs sampling, etc. They are randomized algorithms, and their evolution very much resembles the real evolution of a physical system.

They aim to sample according to the Boltzmann distribution, which is their stationary distribution. They are used to

- study of typical configurations
- compute expectations, marginals, etc.
- approximate the partition function Z_β

The other kind are Message Passing algorithms, which are deterministic distributed algorithms. Some known names are the Cavity Method, Belief Propagation, etc. They aim to directly compute expectations, marginals, Z_β etc.

5.4.1 MCMC-Metropolis

The Metropolis-Hastings [26][18] meta-algorithm is the following.

We define a configuration graph, that is a neighborhood structure between configurations. For example two configurations may be neighbors if they differ in exactly one variable.

Suppose we are in configuration σ_i. We choose a neighbor σ_j uniformly at random. If it is a better configuration, i.e. if $E(\sigma_j) < E(\sigma_i)$, then we go to σ_j with probability $p = 1$. Else we go to σ_j with probability $p = \frac{\mu_\beta(\sigma_j)}{\mu_\beta(\sigma_i)}$ (1), and with probability $1 - p$ we stay in σ_i.

- The stationary distribution is the Boltzmann μ_β.
- The mixing time depends on the energy barriers that may exist, and on β.
- Rule (1) helps to escape from local minima.
- The non trivial part is the design of the neighborhood structure, which gives the rule to select a next configuration. It is different for each problem. Some usual strategies are to change one variable at a time, or one block of variables at a time.

Example. Ising Model Consider the Ising Model on the 2-dim square lattice $\sqrt{n} \times \sqrt{n}$.

There exists a phase transition at β_c.

The Glauber dynamics (Metropolis type Markov Chain) is as follows:

- Choose a vertex $u \in V$ u.a.r.
- Set $u = +$ with probability $Pr[u = +] = \frac{e^{\beta(a-\theta)}}{e^{\beta(a-\theta)} + e^{\beta(\theta-a)}}$

where a and θ are the numbers of neighbors of u with spin '-' and '+' respectively.

It is proved in [24] that the Mixing time of this MC is

- $O(\log n)$ if $\beta < \beta_c$
- $e^{\Omega(\sqrt{n})}$ if $\beta > \beta_c$

So we observe that the physical phase transition is also a "phase transition" with respect to the complexity of the algorithm! (But this may not hold in general.)

5.4.2 Message Passing Algorithms

The energy function E can often be expressed as a sum over some constraints

$$E(\sigma) = \sum_j C_j(\sigma(v_{j1}, \ldots, \sigma(v_{jk}))$$

where C_j are the constraints, and $\forall i = 1, \ldots, k$ v_{ji} are the variables C_j depends on.

For example in the Ising Model for each edge $j = \{u, v\}$, the corresponding constraint is $C_j = -\sigma(u) \cdot \sigma(v)$, and $E(\sigma) = \sum_j C_j$.

So the Boltzmann distribution is factorized over the constraints

$$\mu_\beta(\sigma) = \frac{1}{Z_\beta} \cdot \prod_j e^{-\beta C_j(\cdot)}$$

and the partition function can be expressed as a "Sum-Product":

$$Z_\beta = \sum_\sigma \prod_j e^{-\beta C_j(\cdot)} = \sum_{v_1 \in A} \sum_{v_2 \in A} \cdots \sum_{v_N \in A} \prod_j e^{-\beta C_j(\cdot)}$$

Question Can we compute the above sum quickly?

The sum-product algorithm [22] Let $\oplus, \odot$ be the two operations of a semiring, so the distributive law holds

$$a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c).$$

For example $(+, \cdot)$ on $\mathbb{R}$, $(\max, \cdot)$, $(\min, +)$ etc. In the following we will refer to $(+, \cdot)$ on $\mathbb{R}$, but the results hold for any two such operations.

Let V be a set of variables taking values on Q. The set of configurations is Q^V.

Let $f : Q^U \to \mathbb{R}$ be a function which can be expressed as a "sum-product"

$$f(\cdot) = \sum_{\sigma \in Q^{V \setminus U}} \prod_{j \in J} h_j(\sigma|_{V_j})$$

where $U \subseteq V$, $J \subset \mathbb{N}$, for each j $h_j : Q^k \to \mathbb{R}$, and $V_j \subseteq V$ of size at most k is the set of variables on which h_j depends. For example if we have a function g over n variables that can be expressed as product of functions, then f could be the marginal of g over one variable x, i.e. the sum over all other variables except x, or f could be a quantity like Z_β which does not depend on any of these n variables.

The *factor graph* of f is a graph consisting of nodes corresponding to the variables, one for each $v \in V$, nodes corresponding to the functions (or constraints, or factors), one for each $h_j, j \in J$, and edges connecting a factor node h_j with a variable node v iff h_j depends on v.

If the factor graph is a tree, then due to the distributive law, the computation of f can be decomposed and be executed from the leafs to the root as dynamic programming on trees, by sending messages from node to node, as follows.

Starting from the leaf, each node, after it has received messages from all its children, computes a function of one variable and sends it to its parent, as follows.

If the node corresponds to a factor node h and its parent is a variable node x, then the message from h to x is

$$\mu_{h \to x}(x) = \sum_{\sim\{x\}} (h(\mathcal{N}(h)) \prod_{y \in \mathcal{N}(h) \setminus \{x\}} \mu_{y \to h}(y))$$

where the sum (denoted $\sum_{\sim\{x\}}$) is over all possible values of the vertices on which h depends on, except for x. $\mathcal{N}(h)$ denotes the neighborhood of h i.e. all variables it depends on.

If the node corresponds to variable x and its parent is factor h, then the message from x to h is

$$\mu_{x\to h}(x) = \prod_{h'\in\mathcal{N}(x)\setminus\{h\}} \mu_{h'\to x}(x)$$

where $\mathcal{N}(x)$ is the neighborhood of x, i.e. all factors depending on it.

Note that each message can be computed in constant time, since the sum now is not over all $n = |V|$ variables, but over at most $k-1$ on which h depends (except one).

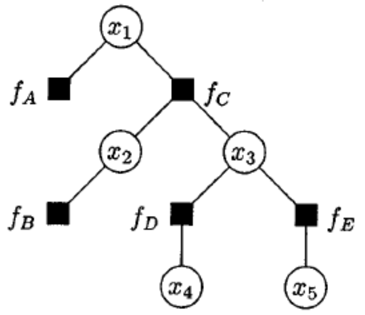
Each message $\mu_{a\to b}(x)$ is a function of one variable, and if Q is finite, then it can be represented as a column matrix m, $1\times|Q|$. Each entry $m(q)$ is the value $\mu_{a\to b}(q)$.

Example (from [22]) Consider the function

$$g(\overrightarrow{X}) = f_A(x_1)f_B(x_2)f_C(x_1,x_2,x_3)f_D(x_3,x_4)f_E(x_3,x_5).$$

Suppose we want to compute the marginal function $g_1(x_1) = \sum_{\sim\{x_1\}} g(\overrightarrow{X})$. The factor graph of g is shown below. The factor graph reveals a way to decompose the computation of g_1, as

$$g_1(x_1) = f_A(x_1)\cdot \sum_{\sim\{x_1\}} (f_B(x_2)\cdot f_c(x_1,x_2,x_3)\cdot (\sum_{\sim\{x_3\}} f_D(x_3,x_4))\cdot (\sum_{\sim\{x_3\}} f_E(x_3,x_5)).$$



Heuristic The idea of belief propagation and other message passing algorithms (e.g. forward / backward algorithm, Viterbi algorithm, turbo decoding, Kalman filter, Certain FFT, cavity method), is to apply this algorithm to factor graphs that are not trees.

We set arbitrary initial values to the messages, and iterate message passing, for all nodes at the same time, until convergence.

It is observed (but not proved) that in many cases it works.

Correctness has been proved for very few cases, as in graphs with only one cycle.

General conditions under which it works, are unknown.

Later on in this chapter we will refer to some ideas (correlation decay, self avoiding walk tree) that may help to understand some cases that it works.

5.5 Symmetry breaking

Often the systems we examine have inherent symmetries, which make the corresponding energy function to be degenerate. An example is the symmetry due to permutations of the values in Q: In coloring it exists, in SAT it does not, in Ising model without external field it exists, in Ising model with external field it does not.

A physical system, in low temperatures selects one of these optimal configurations and "stays" there for ever. This is called symmetry breaking.

When we have symmetry breaking, we observe a phase transition.

5.5.1 Eigenvalues, Mixing time, and Symmetry breaking

From a mathematical point of view, symmetry breaking, is ergodicity breaking, and happens only in the thermodynamic limit (i.e. when $N \to \infty$).

Let P a Markov chain (a stochastic matrix), and π its stationary distribution. That is $P\pi = \pi$, so π is an eigenvector with eigenvalue 1.

It also holds that for every initial probability distribution x, $P^k x \to \pi$ when $k \to \infty$.

Now see how the mixing time relates to the eigenvalues of P: let $\lambda_1, ..., \lambda_n$ be the eigenvalues of P, in order of increasing absolute values, i.e. $\lambda_1 = 1 \leq |\lambda_2| \leq \cdots \leq |\lambda_n|$, and $u_1 = \pi, u_2, \ldots, u_n$ the corresponding eigenvectors.

Consider a vector x decomposed in eigenvectors as

$$x = a_1 u_1 + a_2 u_2 + \cdots + a_n u_n.$$

We have

$$\begin{aligned} P^k x &= a_1 \lambda_1^k u_1 + a_2 \lambda_2^k u_2 + \cdots + a_n \lambda_n^k u_n \\ &= a_1 \pi + a_2 \lambda_2^k u_2 + \cdots + a_n \lambda_n^k u_n \end{aligned}$$

So the mixing time of the Markov chain, i.e. the time it needs to be ϵ-close to π, it depends on $|\lambda_2|$ (how quickly λ_2 will tend to 0).

A Markov Chain is ergodic if it is irreducible and aperiodic. If a MC is ergodic, it has got a unique stationary distribution.

Glauber Dynamics is ergodic, so it has a unique stationary distribution. But if $N \to \infty$ and/or $T \to 0$, ergodicity may break, λ_2 becomes 1 as well, and we have more than one stationary distribution. In fact we have infinite many stationary distributions, since any convex combination of u_1, u_2 is stationary, too.

For example in the Ising model in low temperature,

$$\Pr_{u_1}(\text{all ' + '}) \simeq \frac{1}{2}, \Pr_{u_1}(\text{all ' − '}) \simeq \frac{1}{2}, \Pr_{u_1}(\text{other}) \simeq 0.$$

But when $N \to \infty$, we could also have as stationary u_2 s.t. $\Pr_{u_2}(\text{all ' + '}) \simeq 1, \Pr_{u_2}(\text{other}) \simeq 0$. (Where $\Pr_{u_i}$ denotes the probability according to the distribution u_i.)

5.5.2 Symmetry breaking in practice

In practice symmetry breaking can also happen even in finite (but large) systems, when we do not let the MC to run for sufficient time (poly(N) instead of exp(N) which may be needed), so practically it stacks to a local minimum, or it may happen due to numerical reasons.

With BP we can also observe symmetry breaking. Depending on the initial conditions, BP may converge to the marginals of any stationary distribution. See the experimental results in subsection 4.6.2, for an example of the behavior of BP, depending on the initialization.

5.6 Correlation Decay

Intuitively we can understand that since Belief Propagation and Glauber Dynamics are local algorithms, their success strongly depends on the existence of correlations between vertices that are far from each other in the factor graph.

Both the structure of the factor graph, and the temperature in the Boltzmann distribution, influence the existence of decay of correlations.

For the sake of completeness, we give the definition of correlation decay, without commenting it.

Definition 29. *We say the system has strong spatial mixing if there exist constants b and $a > 0$ s.t. for any two subsets Λ, Ψ where $\Lambda \subseteq \Psi$, any site $u \in \partial\Psi$, and any pair of boundary configurations τ and τ^u that differ only at u, $\|\mu_\Psi^\tau - \mu_\Psi^{\tau^u}\|_\Lambda \leq b|\Lambda| \exp(-a \cdot dist(u, \Lambda))$*

It is proved in [14] that for spin systems on $\mathbb{Z}^d$, that correlation decay is equivalent to the rapid mixing of (block)-Glauber Dynamics.

It is also shown in [35] for the hard core model (and later for other models, too) that when correlation decay holds, it is possible to approximate the partition function, by some method very similar to Belief Propagation applied to the "self avoiding walk tree" of the given graph. We give in the next sections more details about this.

A problem related to correlation decay is the tree reconstruction problem, which is simpler as notion, so we present it below.

5.6.1 The tree reconstruction problem

We consider a d-regular tree T with infinite many nodes, that take values on a finite alphabet $\mathcal{A} = \{1, \ldots, k\}$ Consider the following process. Suppose that the root ρ chooses a value σ_ρ from $\mathcal{A}$ according to some initial distribution, and this value is propagated to all the nodes as follows: Let $M_{k\times k}$ be a stochastic matrix. If the parent u of a node v has the value i, then v takes the value j with probability M_{ij}. The question is whether it is possible to determine the value of the root, given the values of the nodes in depth r, when $r \to \infty$. Or, stated differently, for which T and M does the configuration on the r-th level of the tree contain significant information about the value of the root?

Definition 30. *Let σ_r be the configuration on the r-th level of the tree. Let P_r^l be the conditional distribution of σ_r, given that $\sigma_\rho = l$. Let D_V be the total variation distance between two distributions.*

The reconstruction problem for T and M is solvable if there exist $i, j \in \mathcal{A}$ for which

$$\lim_{r\to\infty} D_V(P_r^i, P_r^j) > 0.$$

A survey on this problem can be found in [27].

Tree reconstruction on Ising model The most understood case of the problem is for the symmetric binary channel, or Ising model on the d-regular tree.

Consider a d-regular tree and let

$$M = \begin{bmatrix} 1-\epsilon & \epsilon \\ \epsilon & 1-\epsilon \end{bmatrix}$$

where $\epsilon = \frac{e^{-\beta}}{e^{-\beta}+e^{\beta}}$.

Theorem 31. *[20][19] Reconstruction is solvable iff $d\lambda_2^2 > 1$, where $\lambda_2 = 1 - 2\epsilon$ is the second eigenvalue of M.*

Proof. (sketch) Let A, B be two probability distributions on probability space X, and $S : X \to \mathbb{R}$ be a random variable.

The statistical distance of A and B is

$$D_V(A, B) = \frac{1}{2}\sum_{x\in X} |A(x) - B(x)|$$

Its easy to see that it is

$$\geq \frac{1}{2}\sum_{x\in X} \frac{(A(x) - B(x))^2}{A(x) + B(x)}$$

From Cauchy Swartz and some algebra we take that it is

$$\geq \frac{(E^A(S) - E^B(S))^2}{E^A(S^2) + E^B(S^2)} \tag{5.1}$$

where E^P denotes the expectation w.r.t. P.

Let now P^+ and P^- be the probability distribution of the nodes in the r-th level of the tree, when the root has value $+$ and $-$ respectively.

Let $S = \#\text{“}+\text{”} - \#\text{“}-\text{”}$ at the r-th level.

By computing the Expectations, we find that

$$(5.1) = \frac{4d^{2r}\lambda_2^{2r}}{4rd^r - 4d^r\lambda_2^{2r} + 2\lambda_2^{2r}d^{2r}} \tag{5.2}$$

If $d\lambda_2^2 < 1$ then $(5.2) \to 0$ when $r \to \infty$.

If $d\lambda_2^2 > 1$ then $(5.2) \to 2 \Rightarrow D_V(P^+, P^-) \to 1$ □

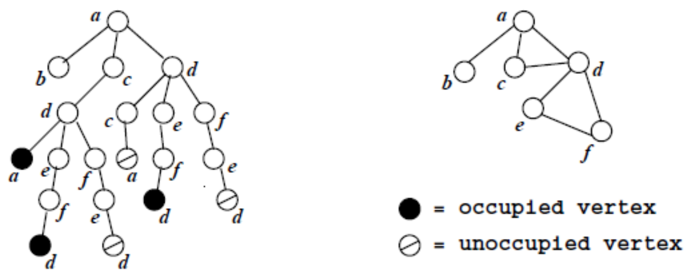
So we see that indeed reconstruction depends on both the temperature (since λ_2 is a function of β), and the structure of the graph (here the degree d).

Some intuition about this result is the following. We can observe that as r tends to infinity, the probability of a vertex to be '$+$' is $1/2$ for both the cases where the root is '$+$' and '$-$'. But when the root is '$+$' this probability is $1/2 + \varepsilon$, while when the root is '$-$' it is $1/2 - \varepsilon$, where ε tends to 0. Whether this difference can be reflected by the nodes in the r-th level, depends on the number of these nodes. The more the better. Of course d must be as larger, as larger is the probability of noise ϵ in M. If noise is small, not many ancestors are needed to keep the information of the root.

Note Note that Belief Propagation on trees is always successful, even in the presence of long range correlations, while this is not true for Glauber Dynamics.

5.7 The Self-Avoiding-Walk tree

Let G be a graph. The self avoiding walk tree ($T_{SAW}(G, u)$) of G rooted at u is a tree with all possible walks starting from u and stopping whenever a vertex is revisited (i.e. a cycle closes). See the next figure for an example and see [35] for details of the construction.



Now consider the hard core model.[1] Whenever a leaf of the T_{SAW} tree corresponds to the closure of a cycle $u_{i_1} u_{i_2} u_{i_3} \cdots u_{i_k} u_{i_1}$ and u_{i_2} is lexicographically smaller (bigger) than u_{i_k}, then fix the value of that leaf to "occupied" ("unoccupied" respectively).

Then it is proved in [35] that the marginal probability of the root u being occupied w.r.t. to the Boltzmann distribution, is the same in the initial graph G and in the $T_{SAW}(G, u)$ tree (with leaves fixed as above).

So as message passing algorithms are exact on trees, we can exactly compute the marginal for each node, through the corresponding T_{SAW} trees.

The problem is that in general the T_{SAW} tree may be exponentially bigger than G.

But if there is correlation decay, then it is shown also in [35] that to approximate the marginals it suffices to look at the tree only up to a logarithmic depth, since the influence of the more distant vertices is negligible.

(We must note that to take the approximation we have to fix properly the nodes on the boundary. See the paper for details.)

[1]In the hard core model there is a graph and the vertices take values 1,0 (occupied / unoccupied) s.t. no two adjacent vertices are both occupied. In other words the valid configurations are the independent sets of the graph.

Chapter 6

Degenerate instances are not meaningful

So far we have seen approaches to define properties that characterize a clustering algorithm, and a clustering objective function. These properties did not take into account the particular instance that we want to cluster.

A natural question is whether a particular instance is compatible with a particular objective function, which means that there is some inherent structure in the data that can be revealed by optimizing or approximating this objective function.

Usually the data are high-dimensional, so we don't have visualization to help us. But as we saw, in the existing bibliography about the meaningful instances of clustering, there is some connections between structure in the data and the behaviour of the objective function over the space of solutions (clusterings). So the idea is to exploit some of these connections to determine the meaningfulness in the data.

In particular our basic intuition is that when someone tries to optimize some objective function, implicitly assumes that all good clusterings are close to the optimum, and to the target clustering (if this exists). If this doesn't hold, then there are several good but different from each other solutions, which means that there is not a unique, compatible with this objective, structure. In other words, this objective does not uniquely defines some structure in the data.

Now, the idea to compare all good solutions, is to construct a matrix, which for each two points of the data set, tells us how often these points belong to the same cluster, when we take into account all good (optimal and approximately optimal) solutions.

Our definition of "closeness" between two clusterings is their hamming distance, i.e. the maximum number of points that these clusterings agree under permutation of cluster labels.

If all good solutions are close to each other, then we have (by definition) that almost all points should be clustered similarly, so all entries of the above matrix will be close to 1 or 0, since almost every pair of points will be in almost all good solutions, either in the same cluster, either not.

If there are several good solutions different from each other, then there will be many pairs of points that in some solutions will be in the same cluster, and in some other will be in different clusters. This will lead to have in the above matrix many entries far from 1 or 0.

We will now give some examples to make more clear these ideas, and to show possible applications of our method. Then we will define in detail the above matrix, and we will discuss several attempts to compute it efficiently.

6.1 Examples-Applications

6.1.1 Examples

We give some examples of pairs (objective function, dataset), in which the matrix we mentioned before, will be not coherent, i.e. there will be many entries far from 1 or 0, and so revealing the existence of many equally good clusterings, instead of one uniquely defined. In these cases clustering with these objectives is not meaningful in terms of structure discovery.

k-means and not convex clusters Suppose out data (in some representation on the euclidean plane) form two concentric circles, and suppose that we know that they are two. If we try to optimize k-means, (or some other center based objective), using the euclidean distances of the points, it is not meaningful, since k-means defines convex clusters. In the case with the concentric circles, all lines passing through the center of the circles, would define equally good solutions.

k-means and wrong number of clusters Suppose that our data form convex clusters, but we don't know how many. If we have chosen as parameter for an algorithm, the right number of clusters, then the optimum solution will be unique. But if the true number of clusters are for example 2, and we chose as parameter the number 3, then one of the real clusters should be split, and there are many different but equally good ways to do this.

k-means, swiss roll, and search of kernel Suppose that our data form convex clusters, but they lye on a manifold of smaller dimension from the dimension of their representation. Such an example is the swiss roll, which is a manifold of dimension 2, in $\mathbb{R}^3$. If we use the euclidean distances of the points, then the real clusters might not be convex in $\mathbb{R}^3$. We should search for the right kernel, that would give us their geodesic distances, i.e. their distances in their intrinsic dimension, and in which dimension they form convex clusters, in a unique good way.

correlation clustering and wrong criterion Suppose that we have handwritten letters ‘‘θ” and ‘‘β”, and suppose that we have chosen a criterion that decides for each two points if they are similar or not (we allow only 1 and 0 values). Finally these relations (1s and 0s) are given as input to correlation clustering. If our criterion was not good, e.g. if we chose as criterion the colour, (i.e. we considered for example orange and yellow as similar, orange and red as similar, red with yellow not similar, and so on) then we wouldn't have a unique way to partition them in two groups. But if the colors were exactly two, too, then we would take a coherent matrix, although the groups are not these that we were looking for. This indicates why our method gives a necessary but not sufficient condition for meaningful clustering.

correlation clustering and blurred images Let us again have 1,0 relations, for images of dogs and cats. If the images are blurred (due to high levels of noise), so much that it would be impossible to tell by sight which animal is in each image, then any partition to two groups would be equally bad.

correlation clustering and random input Consider again 0, 1 relations, that have arose by some critierion good for distinguishing cuts from dogs, but suppose that the real images where fotos of other things. In this case the instance would be like it arose from fotos of cuts and dogs in many different, and of course irrelevant to the reality ways. The instance would look like random, or like very noisy, and we would take a not coherent matrix.

6.1.2 Applications

Some applications of our method could be the following

- Detection of the/some right model, or model parameters
- Detection of the/some right kernel
- Detection of important attributes, or criterion to characterize as similar or dissimilar two input points
- Detection if there is a ground truth, and if there is a lot of noise in the data
- Determine if an instance satisfies conditions that are necessary for some algorithm to work: as we saw in chapter 4, meaningful instances satisfy properties that make an algorithm to work (like stability). But we didn't saw how we can tell if a particular input at hand, satisfies these properties.

6.2 Definition of the Coherence Matrix

We propose a method that takes as input an data-set X and an "energy" function $E : \mathcal{X} \times \mathcal{C} \to \mathbb{R}$ (a clustering objective function), and returns "yes" if it is meaningful to cluster X by optimizing E, else it returns "no". $\mathcal{X}$ is the space of inputs and $\mathcal{C}$ is the space of clusterings (for example for fixed number of clusters, k, $\mathcal{C} = \{1, \ldots, k\}^n$, where n is the number of data points).

The method decides as follows. For a given and fixed X, suppose we choose a clustering C with probability $p(C) = \frac{1}{Z}e^{-\beta E(X,C)}$. Suppose we iterate this many times (say m), and we construct a matrix $T_{n\times n}$, initially with zero entries, in which we add 1 in entry $T_{i,j}$ each time points i and j belong to the same cluster. We return "yes" if the matrix $M = \frac{1}{m}T$ converges to a coherent matrix, in which almost all entries are close to 1 or 0, as m tends to infinity. In other words, this matrix M gives for each pair of data-points the probability of being in the same cluster, according to the Boltzmann distribution.

Note that when β tends to infinity, the Boltzmann distribution tends to give all probability mass to the exactly optimal solutions, while when β tends to zero, the Boltzmann distribution tends to be uniform over all clusterings. The value of β relates to the approximation factor that we want to allow: as we raise β, we take more and more bad (i.e. approximate) solutions into account. Someone should choose a reasonable β (as a function of λ), some value that will allow λ-approximate solutions to have considerable probability mass, for a reasonable λ, according to his desire or intuition.

This method does not decide whether the criterion (objective function) is suitable for clustering, but rather it decides for the particular data whether they have compatible structure with the objective function someone chose.

If the matrix M is coherent, then we know that all good (λ-approximate) clusterings are close to each other (they agree on the classification of most points).

6.3 The Abstraction of the energy and the corresponding factor graph

Now we will define an abstracted energy function, which takes as input the data in the form of a proximity matrix F, i.e. a matrix with the pairwise distances of the data-points, the objective function H (the particular criterion for clustering) and a clustering in the form of a symmetric matrix with 0-1 entries, indicating if two points belong to the same cluster. The output of the abstracted energy is the objective value of the clustering, or infinity if X is not valid, i.e. does not represent a clustering.

6.3.1 A factor graph for an abstracted energy function

Let F be a $n \times n$ proximity matrix. Let $H(X, F)$ be a clustering objective function which takes as input the proximity matrix F, and a clustering in the form of a $n \times n$ adjacency matrix X. Suppose that H can be expressed as a sum of pair-wise functions h, i.e.

$$H(X, F) = \sum_{ij} h(X_{ij}, F_{ij}),$$

where i, j range over $\{1, 2, \ldots, n\}$.

We are looking for an adjacency matrix X, corresponding to a clustering, that minimizes $H(X, F)$ under the following constraints.

$$\min H(X, F) \text{ s.t. } \begin{cases} X_{ii} = 1 & \forall i \\ X_{ij} \in \{0, 1\} & \forall i \neq j \\ X_{ij} + X_{jk} - X_{ik} \leq 1 & \forall i \neq j \neq k \end{cases}$$

The third constraint ensures that the adjacency matrix X corresponds to clustering. So as a first approach, for the factor graph corresponding to this energy function, we introduce a binary variable node X_{ij} for each pair of data points i, j, indicating whether the points i and j belong to the same cluster or not. Nodes X_{ii} are fixed to 1 for all i. We also introduce factor nodes

$$h_{ij}(X_{ij}) = h(X_{ij}, F_{ij}),$$

and factor nodes

$$g_{ijk}(X_{ij}, X_{jk}, X_{ik}) = 0 \; if \; X_{ij} + X_{jk} - X_{ik} \leq 1, \; else \; +\infty.$$

The abstracted energy function is

$$E(H, F, X) = H(X, F) + \sum_{ijk} g_{ijk}(X_{ij}, X_{jk}, X_{ik}) = \sum_{ij} h(X_{ij}, F_{ij}) + \sum_{ijk} g_{ijk}(X_{ij}, X_{jk}, X_{ik}).$$

6.3.2 The Boltzmann distribution over all clusterings and the coherence matrix

The Boltzmann distribution over all clusterings is

$$\mu_\beta^{H,F}(C) = \frac{1}{Z_\beta} e^{-\beta E(H,C,F)}$$

where β is the inverse temperature, Z_β is a normalizing factor, called partition function in statistical physics, H is the given clustering objective function, and F is the given proximity matrix of the data.

Note that if X does not correspond to a clustering, then for some i, j, k, should be $g_{ijk} = +\infty$, so the Boltzmann distribution gives it zero probability.

The coherence matrix is an $n \times n$ matrix M where

$$M_\beta^{H,F}(i,j) = \sum_{C clustering} \mu_\beta^{H,F}(C) C_{ij}.$$

Note that since the relation of "being in the same cluster" is transitive (i.e. if i is in the same cluster with j and j with k, then i is in the same cluster with k), then if the M-matrix is coherent (i.e. almost all entries are very close to 1 or 0), then it will correspond to a graph that consist of separated cliques, which, by the way, is the inherent structure of the data.

6.4 Obtaining the coherence matrix for the abstracted energy function

Now we will study whether it is possible to obtain the coherence matrix M in polynomial time.

One idea is to use Belief Propagation, or sampling with Markov Chains. But there are some difficulties:

- The constraint graph doesn't belong to a category already studied, and has more complicated structure
- We have three parameters to tune: β, the data (by the way, we need a generative model e.g. noise model), and the objective function
- BP and MC don't work for every β in general, neither for every factor graph

Even if we fix the objective function (even if we can fix it without loss of generality), we will have to face the following problems.

In most papers I am aware of, the models have one parameter with respect to which the models are analysed. For some it is the inverse temperature, while in other it is the randomness of the graph (in specific manners). Although we could say that it is the same thing in different names, in our case these two parameters exist both, and play their independent (surely?) role.

When the instances F are generated by a random process, we have two parameters in the model that we study: β, the inverse temperature, and p (noise or some other parameter of the random graph model). Should there exist a phase transition w.r.t. β, for all graphs, or a different for each p?

The success of belief propagation also depends on the two above parameters (more precisely it depends on some -not known in their generality- parameters of the graph.). For example it is exact in the case of a tree, but also it works for graphs with large tree-width, in high temperatures. On the other hand Glauber Dynamics may not succeed in low temperatures even on trees.

We could fix β (depending on the λ, i.e. how much approximate solutions we consider as good), and analyse for which p we obtain a coherent matrix M.

Or we could examine separately for random graphs and structured graphs (fixed p) the coherence of M as a function of β.

Question: Is there an interval of β's values, for which Belief Propagation works, and which is sufficient to distinguish between structured and random input?

6.5 Obtaining the coherence matrix for max-cut

Since Belief propagation succeeds on trees, and on locally treelike graphs without long range correlations, one idea would be to transform the given graph to a tree, (or a graph with no short cycles, and with few big cycles, or a graph with small tree-width, etc.), such that the objective values are preserved, and apply BP to this graph.

For example, it is possible to sparsify a graph in such a way that cuts are preserved.

6.5.1 Sparsification

Given a matrix G, a spectral sparsifier of G is a matrix H with $O(n \log n)$ edges, s.t. for all $x \in \mathbb{R}^n$ it holds

$$x^T L_G x \simeq x^T L_H x.$$

where L_A is the Laplacian of the matrix A, and

$$x^T L_A x = \sum_{i \sim j} a_{ij}(x_i - x_j)^2.$$

Notice that if x indicates a cut, i.e. $x_i = 1$ if i belongs to the first group, and $x_i = 0$ if i belongs to the second group, then the above quantity gives the weight of the cut.

See [31] for some constructions of spectral sparsifiers.

6.6 The coherence matrix for k-means in poly(n) time

The idea is that we can approximate all k-clusterings by some clusterings in a family of polynomial size (polynomial in the size of the dataset). This family consists of all k clusterings with centers in T, where T is defined as follows: Let D be a cover of the whole dataset consisted by balls of radius δ. T is the set of the centers of these balls.

It is proved in [29] that each k-clustering can be approximated to within $4Rk\delta$ by a clustering with centers in T, and the number of these clusterings is $(\frac{4R+\delta}{\delta})^{mk}$, where m is the dimension of the data-space, and R the radius of a ball covering all data points.

So we can compute the coherence matrix by summing over all k-clusterings with centers in T.

Note The number of clusterings is polynomial in the number of data-points, but exponential in their dimension. However we can perform dimensionality reduction, or even use the method to determine if a small dimension is enough for structure discovery.

6.7 The coherence matrix for kernel k-means

6.7.1 Sampling

A quite general model on which we could apply our method, is the kernel k-means.

k-means, and other center-based algorithms, create clusters that are convex, which allows us to define similarity between two clusterings with respect to their centres' closeness, instead of Hamming distance.

The disadvantage of these models is that they allow only convex clusters (more specifically, the convex halls of the clusters' points are disjoint). However there exist instances for which the inherent groups are not convex (e.g. two concentric cycles).

However if we apply some transformation to the data, to represent them differently, then we could have convex clusters in the new representation. Moreover, for each such transformation (map) there is a matrix (called kernel) that gives the pairwise distances of the images of the initial data, and we can use directly the kernel without even knowing the transformation explicitly. So we could use in every case the k-means objective, with distances those given by the kernel.

So the machine learners problem which was to determine the right objective function to cluster a particular data-set, is reduced to finding the right kernel that represents the data as convex clusters in their intrinsic dimension. (This in general may be very difficult, because someone needs to know something about the data, e.g. that they belong to a manifold of smaller dimension, or that they would be better represented in some other coordinate system like polar, or some information of this kind).

So an idea would be to perform sampling with a Metropolis-type Markov chain, where we change in each step one (or more) of the cluster centres, from one position to another one neighbouring position.

We just have to define a neighbourhood structure for the possible positions of cluster centres. For example we could allow only data-points to be possible positions of centres, and define a threshold for the pairwise distances, to consider two points as neighbouring.

6.7.2 Multidimensional Scaling and approximation with balls

Can we adopt the idea with the cover from (simple) k-means, to this setting, where we have kernels instead of euclidean distances between data-points, and we wouldn't know the explicit representation of the centers of the cover?

The answer is that we can apply the previous idea by using first Multidimensional Scaling with the kernel distances, to embed the points in a euclidean space, while preserving the pairwise distances.

Multidimensional scaling

We are given a matrix of squared distances d_{ij}^2 and we are looking for points in a euclidean space with such distances.

Set $a_{ij} = -\frac{1}{2}d_{ij}^2$, $B = HAH$ where $H = \mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^T$ is the centering matrix.

Diagonalize B by singular value decomposition $B = \Gamma\Lambda\Gamma^T$.

B is symmetric and semi-definite, so it has non-negative eigenvalues which coincide with its singular values, so $\Lambda^{\frac{1}{2}}$ exists.

Set $Z = \Gamma\Lambda^{\frac{1}{2}}$, so $B = ZZ^T$.

The rows of Z is the embedded points we were looking for, since $(z_i - z_j)^T(z_i - z_j) = d_{ij}^2$.

Moreover these points are centred around **0**, and the columns of Z are principal components, since $B = cov(Z) \cdot n$.

Note The dimension of this embedding is n, but we can use our method to determine if a small (constant) number of principal components is enough to reveal some inherent structure.

6.8 Our method on Correlation Clustering

Another model which is quite general for qualitative data, and on which we could apply our method, is the correlation clustering model.

In this model we are given a matrix with the relations of every two points as 0 (meaning dissimilar) and 1 (meaning similar). But this relation may not be transitive. We are asked to find a grouping that minimizes the disagreements with the given matrix.

Since the relation is not transitive, it would be natural to consider each instance as obtained from a ground truth clustering, after applying noise on it. An instance is meaningful and can be clustered well, if the noise is not so large that it makes the instance to look random, not even if it looks like it can be clustered well in many different ways.

The usefulness of our method could be to detect if there exists some ground truth, or whether the noise levels are so high that it is not meaningful to search for the ground truth.

This model looks a lot like a generalization of error correcting codes, so we could try to use techniques and theorems from coding theory, or modifications of them.

For example we can compute bounds on the noise levels, over which is impossible to reconstruct the ground truth (no unique solution). We could also compute a bound over which we cannot have a polynomial number of probable solutions. Finally we may could compute a bound over which all probable solutions, even if not poly-mane, are ϵ-close to each other, or we could compute a bound after which there will certainly exist two solutions, more than ϵ-far from each other.

As for Belief propagation, there are also constructions of error correcting codes (although very different from ours), in which Belief propagation is used for decoding very efficiently and proven correctly, despite the existence of small cycles in the corresponding factor graph (LDPC decoding, turbo codes). Maybe we could modify these algorithms for our method.

6.8.1 The model

We are given a matrix $M_{n\times n}$ consisting of zeros and ones, where M_{ij} indicates whether objects i and j should be in the same cluster. We are asked to find a clustering minimizing the violated entries of M. This is a very hard problem, in particular it is hard to approximate within a constant.

We can assume that the matrix M is generated as follows. There is a ground truth clustering which creates a block matrix M' indicating which objects are in the same cluster and which are not. After that, noise is put in the matrix M', creating M, i.e. every entry of M' is changed with probability p.

As we saw, in [25] is given a $(1 + O(n^{-1/6}))$-approximation algorithm for the case that $p < 1/2 - n^{-1/3}$.

6.8.2 The factor graph for correlation clustering

Given a proximity matrix F, we are looking for an adjacency matrix X, corresponding to a clustering, that minimizes $d(X,F) = \frac{1}{2}\sum_{ij}|X_{ij} - F_{ij}|$ under the following constraints.

$$\min d(X,F) \text{ s.t. } \begin{cases} X_{ii} = 1 & \forall i \\ X_{ij} \in \{0,1\} & \forall i \neq j \\ X_{ij} + X_{jk} - X_{ik} \leq 1 & \forall i \neq j \neq k \end{cases}$$

The third constraint ensures that the adjacency matrix X corresponds to clustering. So as a first approach, for the factor graph corresponding to this energy function, we introduce a binary variable node X_{ij} for each pair of data points i, j, indicating whether the points i and j belong to the same cluster or not. Nodes X_{ii} are fixed to 1 for all i. We also introduce factor nodes

$$f_{ij}(X_{ij}) = |X_{ij} - F_{ij}|,$$

and factor nodes

$$g_{ijk}(X_{ij}, X_{jk}, X_{ik}) = 0 \; if \; X_{ij} + X_{jk} - X_{ik} \leq 1, \; else \; +\infty.$$

The energy function is

$$E(X,F) = \frac{1}{2}\sum_{ij} f_{ij}(X_{ij}) + \sum_{ijk} g_{ijk}(X_{ij}, X_{jk}, X_{ik})$$

6.8.3 Correlation Clustering as error correcting code

After observing the diagrams of the experimental results for the planted partition model (section 4.6.2), a natural question is whether the fail of Belief propagation to find a solution with positive overlap with the planted partition, is due to impossibility of reconstruction from an information theoretical point of view. We computed this bound for the correlation clustering model, which is a special case of the stochastic block model, answering the above question in the affirmative.

In this section we are going to prove that, in the case of 2 clusters, recovering the ground truth is impossible when $p = 1/2 \pm \frac{\sqrt{\ln 2}}{\sqrt{2n}}$.

We have to mention that the stochastic block model is a generalization of correlation clustering, where the probabilities of noise for the zero entries $p = b/n$, and for the one entries $q = (1-a)/n$, are different. It is proved in [28] with different techniques, that finding the ground truth is impossible when $(a-b)^2 < 2(a+b)$ (where a/n and b/n are the inter cluster and intra cluster edge probabilities respectively), which implies that when $(1-a)/n = b/n = p$, it is impossible when $p = 1/2 \pm \frac{1}{\sqrt{2n}}$, and this is better from our result by a factor of $\sqrt{\ln 2}$.

Theorem 32. *Biclustering is impossible when* $p = \frac{1}{2} \pm \frac{\sqrt{\ln 2}}{\sqrt{2n}}$.

Proof. Based on Shannon's theory. For details see [30].

We have n^2 space (the matrix M) to express two things: the clustering and the noise.

There are $(2^n - 2)/2 = 2^{n-1} - 1$ different partitions of n points into two non empty sets, so the information of the clustering is $\log(2^{n-1} - 1) \simeq n - 1$.

Lemma. If we have an n-bit string where noise is put on each bit with probability p, the information for the noise is $\simeq H(p)n$, where $H(p)$ is the entropy function $H(p) = -p \log p - (1-p)\log(1-p)$.

Sketch Proof of lemma. For $n \to \infty$ the noisy bits are pn w.h.p. The different noise-strings (i.e. the string whose entries indicate whether we have flipped the corresponding bit of the original string), are roughly $\binom{n}{pn}$, so the information of the noise matrix is roughly

$$\log \binom{n}{pn} \simeq \log \frac{n^n}{(pn)^{pn}((1-p)n)^{(1-p)n}} \simeq n(-p\log p - (1-p)\log(1-p)) = H(p)n.$$

Since in our case noise is put in each of n^2 bits of the clustering matrix, the information for the noise is $H(p)n^2$.

It is impossible to recover the clustering if the total information is more than the size of M, i.e. when

$$n - 1 + H(p)n^2 > n^2 \Leftrightarrow$$

$$n - 1 + H(p)n^2 - n^2 > 0 \Leftrightarrow$$

Set $f(n,p) = n - 1 + H(p)n^2 - n^2$. The Taylor approximation of $f(n,p)$ for p close to $1/2$ is

$$T(n,p) = n - 1 - \frac{2n^2(p-\frac{1}{2})^2}{\ln 2} + O(p - \frac{1}{2})^4.$$

Now

$$T(n,p) > 0 \Leftrightarrow p = \frac{1}{2} \pm \frac{\sqrt{2\ln 2(n^3 - n^2)}}{2n^2} \simeq \frac{1}{2} \pm \frac{\sqrt{\ln 2}}{\sqrt{2n}}.$$

□

6.9 The coherence matrix via semidefinte programming

Note that our model is a generalized semidefinite programming problem too, and we could tackle it as such.

If we consider the theorems in [25], saying that for $p \leq 1/3$ the sdp solution is exactly the planted clustering, and that for small enough noise we can obtain an approximate solution, then we observe the following. If we computed the coherence matrix with these approximate solutions, not with the Boltzmann probabilities, but with the probabilities resulting from the rounding scheme, then this matrix would be (almost) the same with the solution of the sdp X. This holds because the rounding scheme roughly puts two points i and j in the same cluster with probability X_{ij}.

The question is how much do we lose from the real matrix. Are there any approximate solutions (with the same factor of approximation) that we don't take into account with this approach?

Note The same idea applies not only to correlation clustering, but to the generalized energy model too, since it can be formulated as a semidefinite program in a similar way.

Bibliography

[1] Margareta Ackerman and Shai Ben-David. Clusterability: A theoretical study. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, pages 1–8, 2009.

[2] Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Approximate clustering without the approximation. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 1068–1077, 2009.

[3] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. A discriminative framework for clustering via similarity functions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 671–680, New York, NY, USA, 2008. ACM.

[4] Shai Ben-David. Alternative measures of computational complexity with applications to agnostic learning. In *Theory and Applications of Models of Computation, Third International Conference, TAMC 2006, Beijing, China, May 15-20, 2006, Proceedings*, pages 231–235, 2006.

[5] Shai Ben-David and Margareta Ackerman. Measures of clustering quality: A working set of axioms for clustering. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 121–128, 2008.

[6] Shai Ben-David, Ulrike von Luxburg, and Dávid Pál. A sober look at clustering stability. In *Learning Theory, 19th Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 22-25, 2006, Proceedings*, pages 5–19, 2006.

[7] Yonatan Bilu, Amit Daniely, Nati Linial, and Michael Saks. On the practically interesting instances of MAXCUT. In *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, February 27 - March 2, 2013, Kiel, Germany*, pages 526–537, 2013.

[8] Yonatan Bilu and Nathan Linial. Are stable instances easy? In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 332–341, 2010.

[9] Yonatan Bilu and Nathan Linial. Are stable instances easy? *Combinatorics, Probability & Computing*, 21(5):643–660, 2012.

[10] Amin Coja-Oghlan, Elchanan Mossel, and Dan Vilenchik. A spectral approach to analysing belief propagation for 3-colouring. *Combinatorics, Probability & Computing*, 18(6):881–912, 2009.

[11] Amit Daniely, Nati Linial, and Michael Saks. Clustering is difficult only when it does not matter. *CoRR*, abs/1205.4891, 2012.

[12] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *CoRR*, abs/1109.3041, 2011.

[13] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Phase transition in the detection of modules in sparse networks. *CoRR*, abs/1102.1182, 2011.

[14] Martin E. Dyer, Alistair Sinclair, Eric Vigoda, and Dror Weitz. Mixing in time and space for lattice spin systems: A combinatorial view. *Random Struct. Algorithms*, 24(4):461–479, 2004.

[15] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, second corrected edition edition, 1993.

[16] Dimitrios Gunopulos. Cluster and distance measure. In *Encyclopedia of Database Systems*, pages 374–375. 2009.

[17] Dimitrios Gunopulos. Clustering overview and applications. In *Encyclopedia of Database Systems*, pages 383–387. 2009.

[18] W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[19] Y. Higuchi. Remarks on the limiting gibbs state on a $(d+1)$-tree. *RIMS Kyoto Univ.*, 13:335–348, 1977.

[20] H. Kesten and B. P. Stigum. Additional limit theorems for indecomposable multidimensional galton-watson processes. *Ann. Math. Statist.*, 37(6):1463–1481, 12 1966.

[21] Jon M. Kleinberg. An impossibility theorem for clustering. In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 446–453, 2002.

[22] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theor.*, 47(2):498–519, September 2006.

[23] L. Lovász. Semidefinite programs and combinatorial optimization. In BruceA. Reed and CláudiaL. Sales, editors, *Recent Advances in Algorithms and Combinatorics*, CMS Books in Mathematics / Ouvrages de mathématiques de la SMC, pages 137–194. Springer New York, 2003.

[24] F. Martinelli and E. Olivieri. Approach to equilibrium of glauber dynamics in the one phase region. i. the attractive case. *Communications in Mathematical Physics*, 161(3):447–486, 1994.

[25] Claire Mathieu and Warren Schudy. Correlation clustering with noisy input. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 712–728, 2010.

[26] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[27] E. Mossel. Survey: Information flow on trees. *ArXiv Mathematics e-prints*, June 2004.

[28] E. Mossel, J. Neeman, and A. Sly. A Proof Of The Block Model Threshold Conjecture. *ArXiv e-prints*, November 2013.

[29] Alexander Rakhlin and Andrea Caponnetto. Stability of \$k\$-means clustering. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 1121–1128, 2006.

[30] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

[31] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.

[32] Michalis Vazirgiannis. Clustering validity. In *Encyclopedia of Database Systems*, pages 388–393. 2009.

[33] Ulrike von Luxburg. Clustering stability: An overview. *Foundations and Trends in Machine Learning*, 2(3):235–274, 2009.

[34] Ulrike von Luxburg, Robert C. Williamson, and Isabelle Guyon. Clustering: Science or art? In *Unsupervised and Transfer Learning - Workshop held at ICML 2011, Bellevue, Washington, USA, July 2, 2011*, pages 65–80, 2012.

[35] Dror Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 140–149, 2006.

[36] Reza Zadeh and Shai Ben-David. A uniqueness theorem for clustering. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 639–646, 2009.