A quantitative approach to Consistency Theorem in Clustering

G14PJA

Mathematics 3rd Year Project

Autumn 2018/19

School of Mathematical Sciences

University of Nottingham

Zehui Li

Supervisor: Dr. Yves van Gennip

Project code: XX P99

Assessment type: Review

I have read and understood the School and University guidelines on plagiarism. I confirm that this work is my own, apart from the acknowledged references.

Abstract

Contents

1 Introduction **4**

1.1 Related work 5

1.2 Our contribution 5

2 Review of Kleiberg's work **5**

2.1 Priliminaries 6

2.2 Missing Proofs for single linkage 7

2.3 Why modify the consistency property? 10

3 A framework to investigate Γ-transformation **12**

3.1 Our approach 12

3.2 Generate data points by Γ-transformation 13

3.3 Measure the difference between partition results 14

4 Simulation Results **15**

4.1 Simulation for K-mediods 16

4.2 Simulation for Complete Linkage 18

4.3 Implication and limitation of the simulation 20

5 Prediction With Machine Learning methods **21**

5.1 Approach 22

5.2 Implementation and Experiment results 23

6 Conclusions **23**

A Remaining Proofs for Single-linkage **24**

B Pyhton and Matlab code **27**

1 Introduction

Clustering analysis can be defined as a process of segmenting the data points into several subsets, or clusters, with the goal of making the data points within a cluster to be similar to each other, while the data points in distinct clusters to be different. Clustering has been widely used in many fields, such as pattern recognition, bio-informatics and image processing, however, most of the study toward the uniform notion of clustering only stop at the very general level. The algorithms to achieve the clustering task are called clustering algorithm: depending on the definition of the clusters and the way to find the clusters, these clustering algorithms differ from each other significantly. In 2003, Kleinberg [1] published a highly influential paper, in which he set up a general framework to study clustering algorithms as a whole, and proposed three properties that any clustering algorithms could have.

Clustering algorithms fall into three categories [2]: combinatorial algorithms, mixture modelling, and model seeking, the algorithms in each category follows different underlying principal. The advantage of Kleinberg's framework is that it can be applied to all these clustering algorithms regardless of these principal. The core of the framework - three properties proposed by Kleinberg - are called scale invariance, richness and consistency respectively. Scale invariance states that if the distance(dissimilarity) between the data points is multiplied by a positive number, the clustering algorithm should partition the data into the same clusters as before. Richness requires that for any given partition of the data points, it will be possible to come up with a pair wise distance between the data points, so that the clustering algorithm can produce the given partition. Finally, a clustering algorithm satisfy the consistency theorem if we decrease distances between the data points within a cluster, increase the distance between the cluster, the algorithm should produce the same partition. The most important conclusion from Kleinberg's paper is that there is no clustering algorithm which could satisfy three properties at the same time.

Following the impossibility theorem proposed by Kleinberg, numerous relaxation methods on the axiomatic system are proposed in recent years. In particular, relaxation of

consistency theorem is the focus of this paper. The consistency theorem proposed by Kleinberg, while reasonable and simple, it give a relative strict restriction on clustering algorithms - it require the clustering algorithm to give the same partition results even when the data set is perturbed profoundly (we will explain why this theorem is not sensible in more details in the following section). In this paper, we start from reviewing the work of Kleinberg, adding the missing proof to statements made in his paper, and do simulations on the computer to show the validity of these statements. The rest of the paper will focus on the study of consistency theorem - we come up with a quantitative framework to investigate the consistency property of clustering algorithm. and identify the highly separable distribution of partition results given legitimate perturbation. Finally, we tried several machine learning method to capture the relation between the extend of perturbation and change of partition results. It turns out the methods can capture the model with a very high accuracy and f-measure. At the end of the paper, we try to construct a quantity to measure the extend of perturbation and explore correlation between the the measurement and the change of partition.

1.1 Related work

1.2 Our contribution

2 Review of Kleiberg's work

This section begins by introducing the mathematical notations used for clustering, and give the formal definition of scale invariance, richness and consistency theorem. After having these definition/knowledge in mind, we will move on to prove three statements about a very simple clustering algorithm - single linkage. Then we will present a process of showing the scale invariance property using computer simulation(in python). Finally, we will discuss why consistency theorem is a more strict restriction compared to the others, Which gives us motivation to explore the ways to change consistency theorem.

2.1 Priliminaries

Every clustering algorithm can be denoted by a "clustering function" f, the input of this function is a set S consisting of n data points and the pairwise distances among them. Each points in set S is represented by a integer, so $S = \{1, 2, 3, ..., n\}$ with $n >= 2$. One the other hand,there are multiple ways to represent the pairwise distances, for example, for $S = \{1, 2, 3, ..., N\}$, a $N \times N$ distance matrix M can be used to represent the distances, in which each entry $M_{i,j}$ refer to the distance between points i and j. However, in our case, instead of using the distance matrix, we will use *distance function* to denote the pairwise distances, which is more convenient when dealing with the theorems we defined below. *Distance function* is define as a function d: $S \times S \to \mathbb{R}_{\geq 0}$ with symmetric property, thus, $d(i, j)$ equals $d(j, i)$,and both represent the distance between the points $i, j \in S$. In particular, $d(i, i) = 0$ for any $i \in S$. Distance function are not required to be *metrics*, in other words, d don't need to satisfy triangle inequality, but adding such restriction will not affect results we have below.

Naturally, *clustering function* f take a data Set S and a distance function d as the inputs, and output the a partition Γ of S, where $\Gamma = \{C_1, C_2, ..., C_k\}$, each of the cluster C_k contains some of the data points in S. For example, let $S = \{1, 2, 3, 4, 5\}$, then we could have $f(S, d) = \Gamma = \{\{1, 2\}, \{3, 4, 5\}\}$. For simplicity, we can also write $f(S, d)$ as $f(d)$ without explicate referring to data set S. These three properties - scale invariance, richness and consistency - are all defined around this clustering function f.

Definition 2.1. *Scale-Invariance. f satisfy Scale-Invariance $\iff$ For any given distance function d and any $\alpha > 0$, $f(d) = f(\alpha \cdot d)$*

Scale Invariance simply requires that the clustering algorithm don't rely on the fixed quantity to cluster the data set.

Definition 2.2. *Richness. f satisfy Richness $\iff$ For any given partition Γ of S, $\exists d$ such that $f(d) = \Gamma$*

This property is called richness, because the by feeding in the clustering algorithm different distance functions d, we can reach any possible partition of the given data set S. The third property is called consistency, and it contains more details than the first

two. The basic idea of consistency is that, if we perturb the data in a desirable way, our algorithm should produce the same partition Γ. We first give a formal definition to the "desirable perturbation", calling it Γ *-transformation.*

Definition 2.3. *Given a partition* $\Gamma = \{C_1, C_2, ..., C_m\}$ *on data set* S, d' *is a* Γ *-transformation of* $d \iff$ *For any points* $i, j \in C_k$, $d'(i,j) <= d(i,j)$*; and if* $i \in C_k, j \notin C_k$, $d'(i,j) >= d(i,j)$.

It may seems a little messy at the first glance, but we can interpret Γ *-transformation* as a specific way to perturb the dataset. Suppose we have a data set at the beginning, then we apply a clustering algorithm on this data set, and obtain several clusters. We will squash the points within the same cluster together, and move the points in one cluster away from the other clusters. The resulting distance, will be d' in our definition above. And consistency property simply require that, if we apply the clustering algorithm on the perturbed version of data set, we will still have the same points assigned to the same clusters.

Definition 2.4. *Consistency.* f *satisfy consistency* $\iff$ *Given that* d' *is* Γ*-transformation of distance function* d, $f(d) = f(d')$

Kleinberg's framework starts from abstraction: it abstracts clustering algorithm into a clustering function f, then defines several properties to analyse the function. Once we have this framework, there are two direction to continue the study: the first is to come down to specific algorithms and study whether or not this clustering algorithm satisfy these properties; the other way is to modify the existing properties or add new properties into this framework. Both approaches are mentioned in this paper: We study **single linkage** clustering algorithm in the next section, then the rest of the paper will seek ways to modify the **consistency** theorem.

2.2 Missing Proofs for single linkage

Single linkage is a bottom-up hierarchical clustering algorithm, it begin with each clusters representing a single group, then at each step, it will merge the two "nearest" (with least

dissimilar) clusters into a single cluster, where dissimilar is defined in following manner [2]. The algorithm will terminate until some termination condition is satisfied.

Definition 2.5. *Let G,H represent two clusters, d is the distance function of the data set, then dissimilar d_{SL} is defined as:* $d_{SL}(G, H) = \min_{i \in G, i' \in H} d_{ii'}$

An alternative way to describe single linkage is to treat clustering as a Graphs Construction process [3]. Tuple (S, d) naturally form a complete Graph $G(S, d)$, whose node set is the data set S, and weigh of edges between nodes $i, j \in S$ is the distance function d_{ij}.Single linkage will first order the edges in the non-decreasing order, then for each iteration, it will take one edge from the ordered list,then terminate when the termination condition is satisfies. At this point, all the picked edges form a new partially connected graph G_c, where the node set is the still the data set, but edge set is a set formed by all picked edges. We will use this graph perspective in the following proofs.

By controlling the "termination conditions", we can construct three single linkage algorithms, such that each of the them can satisfy two properties out of Scale-invariance, Richness and Consistency. Three stop conditions are listed below[1].

- *k-cluster termination condition.* Stop adding edges when the partially connected graph G_c consists of k connected components.
- *distance-r termination condition.* Only add edges of weight at most r.
- *scale-α termination condition.* Let ρ^* denote the maximum pairwise distance; i.e $\rho^* = \max_{i,j} d(i, j)$. Only add edges of weight at most $\alpha\rho^*$.

Each of the termination condition is a trade-off between three properties: for example, single linkage with *distance-r* termination condition has a built-in scale, so it will not satisfy the scale-invariance property. But this algorithm indeed satisfy Richness and Consistency. Similarly, we have the following theorem:

Theorem 2.1. *For any $\alpha >= 1$, and any $n >= 3$, single-linkage with the scale-α termination condition satisfies Scale-Invariance and Richness.*

Theorem 2.2. *For any $k >= 1$, and any $n >= k$, single-linkage with the k-cluster termination condition satisfies Scale-Invariance and Consistency.*

Theorem 2.3. *For any $r > 0$, and any $n >= 2$, single-linkage with the distance-r termination condition satisfies Richness and Consistency.*

Here we present the proof of *Theorem 2.1* and enclosed the proof of *Theorem 2.2* and *2.3* in the appendix A

Proof. theorem 2.1

Given data set S and distance function d, let $\rho^* = \max_{i,j} d(i,j)$, and f be the single-linkage with scale-α termination condition.

Let's first prove that f satisfy Scale-invariance property. Assume another distance function d', and d' satisfy that for $\forall\ i,j \in S, d'(i,j) = \beta d(i,j)$, where $\beta > 0$. Let $\Gamma = f(S,d)$, $\Gamma' = f(S,d')$ respectively. If we can show that $\Gamma = \Gamma'$, we will prove f satisfy Scale-invariance. Following the Graph interpretation of single-linkage, the resulting partition can be represented by a partially connected Graph $G_c(S,E)$, where the node set is data set S, and edge set E consists of picked edges. Let $G_c(S,E) = \Gamma = f(S,d)$, and $G'_c(S,E') = \Gamma' = f(S,d')$. Now, if we can prove that $G_c(S,E) = G'_c(S,E')$, we are done. To prove $G'_c = G_c$, we only need to prove that edge set $E = E'$, because G and G' have the same node set.

Let's look inside E and E': Due to "scale-α" termination condition, edge Set E will contain all the edges which has weights smaller than $\alpha\rho^*$. Formally, $d(e_i) <= \alpha\rho^*$, for $e_i \in E$; $d(e_j) > \alpha\rho^*$, for $e_j \notin E$. Similarly, E' will only contain the edges which are smaller or equal to the "threshold value", let's denote this value by ρ_2^*. So for $e_i \in E'$, $d'(e_i) <= \rho_2^*$, and for $e_j \notin E'$, $d'(e_j) > \rho_2^*$. But $d'(.) = \beta d(.)$ and $\rho_2^* = \max_{i,j} d'(i,j) = \max_{i,j} \beta d(i,j) = \beta \max_{i,j} d(i,j) = \beta\rho^*$. If we substitute the values of $d'(e)$ and ρ_2^* with $\beta d(e)$ and $\beta\rho^*$ in the inequality for E', we will have $\beta d(e_i) <= \beta\alpha\rho^*$, for $e_i \in E'$, which is equivalent to the constraints for elements in E, thus E' has the same elements as E, which indicate that $\Gamma = \Gamma'$, and we are done with the proof of Scale-invariance.

The proof for richness is much easier: Given a partition Γ, if we can construct a distance function d such that $f(d) = \Gamma$, we are done. Let's assume we are given a partition $\Gamma = G_c(S,E)$, where $E = \{e_1, e_2, \ldots e_m\}$. To make f produce such a edge set,

we can define d as following:

$$d(e_i) = \begin{cases} \alpha^2, & i \in \{1, 2, 3, \dots m\} \\ 1, & i \notin \{1, 2, 3, \dots m\} \end{cases}$$

In this manner, $\rho^* = \max_{i,j} d(i,j) = 1$, because *theorem 2.1* assumes $\alpha < 1$. Let $f(d) = G(S, E_{new})$, f will put all the elements which are smaller and equal to $\alpha\rho^* = \alpha$ into E. Because $d(e_i) = \alpha^2 < \alpha$, for $i = 1, 2 \dots m$, we have $E_{new} = \{e_1, e_2 \dots e_m\} = E$ □

In this proof, we show how to prove Scale-invariance and Richness. The way to prove Consistency is somewhat similar to Scale-invariance, it starts from treating the partition Γ as a partially connected Graph $G_c(S, E)$, then use the property of f and the relation between d and d' to build the equivalence of two edge set E and E'. The details are put in Appendix A.

2.3 Why modify the consistency property?

Kleiberg's **impossibility theorem** states that it is impossible for any clustering algorithm to have three properties at the same time, in other words, it indicates that we shoud not have unrealistic expectation to have a "perfect" clustering algorithm, but are these three properties desirable? Following the discussion in the blog of Willians[4], we will argue that, consistency property doesn't really reflect our expectation to clustering algorithm.

The idea of consistency theorem is that after we apply Γ-transformation to the data set, the clustering algorithm is not influenced by this perturbation, and can still produce the same partition results. As *figure 1* illustrate, this property seem very intuitive at first glance: if we increase the distance between each cluster or squeezing the points within a cluster together, it should be more obvious to the algorithm that which points should be classified into the same group.

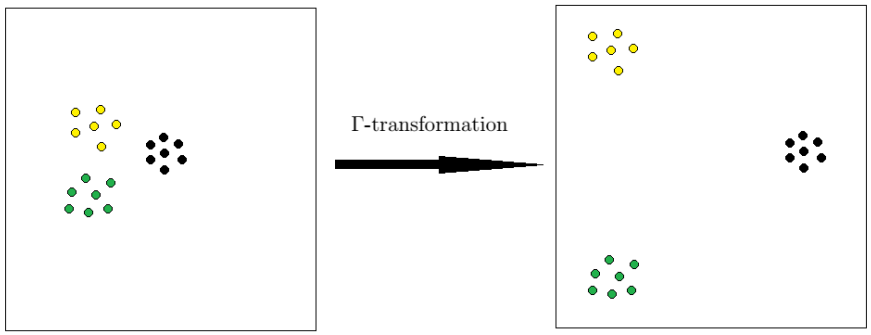

Figure 1: *This is example of* Γ*-transformation for two dimensional data: After the transformation the data points in different clusters are moved away from each other, and the boundary between "clusters" should be more clear than before.*

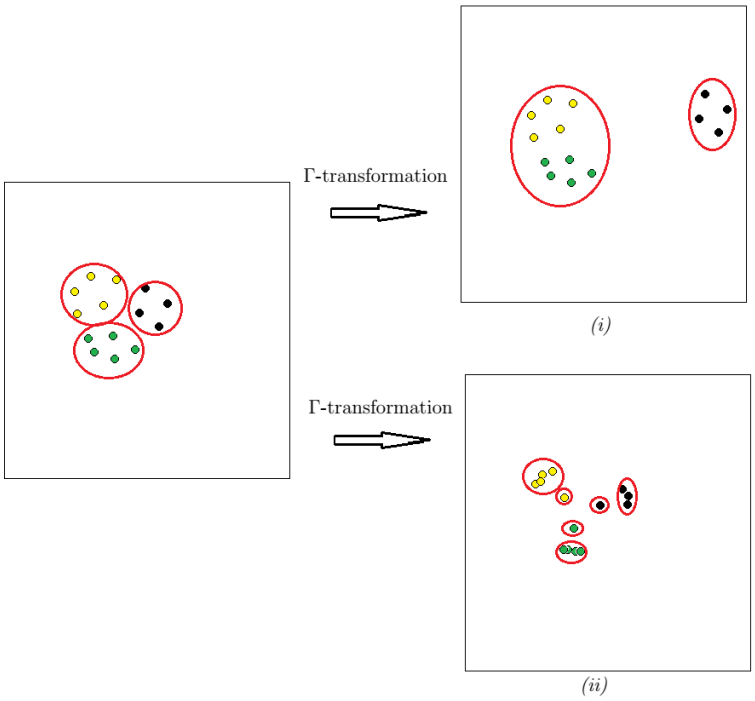

Figure 2: *This is the example of* Γ*-transformation which results in changing the structure of the data: In (i), one of the clusters is pulling away from the other two clusters significantly. In (ii), for each of the cluster, we the shrink all the points together but leave one point out.*

Despite this seemingly reasonable assumption that Γ-transformation always keep the structure of the data the same, actually, in many cases (as *Figure 2* shows), Γ-transformation

will create undesirable transformation, in which the perturbed version of data should be clustered differently. For example, in (i), suppose we move one cluster infinitely away from other clusters, obviously it is a legitimate Γ-transformation, but the resulting data should be partitioned into two clusters. However, consistency theorem, if hold, will require the clustering algorithm to produce three clusters as before. Similarly, in (ii), for each cluster, without breaking the constraints of Γ-transformation, if we shrink all the points together, but leave one points out, it should be more reasonable to partition the data in the way shown in *Figure 2*.

These problems with consistency theorem motivate us to study the property of Γ-transformation, try to figure out in which case does Γ-transformation change the structure of the data set.

3 A framework to investigate Γ-transformation

3.1 Our approach

We are taking a numerical approach to investigate how Γ-transformation change the structure of the data. The general idea is that suppose we have a data set, we first apply some clustering algorithm on this data set, which produce a partition Γ. Then, we apply Γ-transformation to this data set for numerous times, and each time we will obtain a different data set. For a clustering algorithm which doesn't satisfy consistency property, if we apply the algorithm to these data sets, it should produce different partition results. We want to figure out what kind of Γ-transformation will result in partitions which are the same as the partition Γ . To decide whether two partition results are the same, we need have some "measurement", which could measure the difference between two partition, we will discuss various criteria to measure this difference in the third sub-section.

The implementation of these numerical analysis rely on the Python and several libraries including numpy, scikit-learn, etc. We enclose important part of code in Appendix B (The full lists of code can be found on my personal github page).

3.2 Generate data points by Γ-transformation

Clustering algorithms can be classified into two groups by the types of required inputs: for algorithms like k-means, we will need the specific coordinates of the every points in the data set, but for algorithm like single-linkage and k-medoids, they only need the distance functions as the input, then they can produce the partition results. To apply a Γ-transformation on the data set with specific coordinates will be very difficult(actually it is computational impossible on any personal computer even small workspace, when the data set become relatively big[1]). So we restrict our scope to the algorithms like single-linkage and k-medoids, which don't need the coordinates. Then in this case, to apply a Γ-transformation, we simply modify the distance function following the constraints of Γ-transformation.

With the scope restricted, a dataset containing N data points can be denoted by a $N \times N$ distance matrix X, where $X_{i,j}$ refer to the distance between points i and j, and it is symmetric matrix with diagonal elements equal to zero. Suppose we have a dataset X, a clustering function f, and $\Gamma = f(X)$. If we independently apply Γ-transformation to the dataset for n time, we can obtain a set of distance matrices $D = \{X'_1, X'_2, \ldots, X'_n\}$, where each element X'_i is the resulting distance matrix from a random Γ-transformation. Two things need to be noticed: First, every X'_i is produced independently from X. Secondly, Γ-transformation from X to X'_i can make any changes to original distance matrix X, it is a completely random process. We enclose the python code of Γ-transformation in Appendix B, it is a function called **perturb_distance_matrix()**, which take the original distance matrix X and partition Γ as the input, and return a distance matrix produced from some random Γ-transformation.

What is the use of these distances matrices? Well, each of these distance matrices are produced from a "Γ-transformation"; Among these transformations, some of the them will change the structure of the original dataset, in which case we expect to see that

[1]Recall Γ-transformation require that we can only decrease the distance between the points within a cluster, and increase the distance between the point in different point. Suppose we have n points in the data set, then when we move one point, we will need to exam $n-1$ equations to make sure our movement satisfy all the inequality. On the other hand, if we are working on the distance matrix, we don't need to worry about these constraints

$f(X_i') \neq \Gamma$, while in the case where the structure of the dataset is preserved, we will expect $f(X_i') = \Gamma$. Our aim is to target these good Γ-transformations among D, then try to come up with some quantities or models to capture these good Γ-transformations in the unseen cases. To identify "good Γ-transformations", we will apply f to all the elements of distance matrices D, which results in a set of new partition $P = \{\Gamma_1', \Gamma_2' \dots \Gamma_n'\}$, where $\Gamma_i = f(X_i')$. We want to find all $i \in \{1, 2 \dots n\}$ such that $\Gamma_i' = \Gamma$, but how can we compare two partitions? We will look at this problem in next section.

3.3 Measure the difference between partition results

The task here is to compare two partitions of the same dataset, many criteria, such as Jaccard index, Variation of Information distance [5] and Rand index, can be used to describe the difference between two partitions. All these criteria follow a similar idea: the difference between two clusters is defined in terms of the number of points, or pairs of points two partition disagree [6]. In this paper, we will use an variation of a simple criteria - **Adjusted rand index** , this criteria is based on Rand index, but normalized to have better property. Let's first define Rand index.

Given a dataset of n points $S = \{1, 2 \dots n\}$ and two partition of S: $\Gamma = \{C_1, C_2 \dots\}$ and $\Gamma' = \{C_1', C_2' \dots\}$, where $C_1, C_2 \dots$ and $C_1', C_2' \dots$ are non-overlap subsets.

- Let $x = |S^*|$, where $S^* = \{(i, j) | i, j \in C_l \text{ and } i, j \in C_k'\}$
- Let $y = |S^{**}|$, where $S^{**} = \{(i, j) | i \in C_{l_1}, j \in C_{l_2} \text{ and } i \in C_{k_1}', j \in C_{k_2}'\}$

Rand Index, Rd, is defined as:

$$Rd = \frac{x + y}{\binom{n}{2}} \tag{3.1}$$

$\binom{n}{2}$ is the total number of possible choices of pair. Rand index can be interpreted as the probability that Γ and Γ' will agree on a randomly chosen pair. The range of Rand index is $[0, 1]$, when $Rd(\Gamma, \Gamma') = 1$, Γ and Γ' are exactly the same, while $Rd(\Gamma, \Gamma') = 0$ means two partition do not agree on any pair of points. However, Rand index is not often used directly, because it have two obvious drawbacks: For one, in practice, the value of Rand index is ususally in the range between $[0.5, 1]$. Also, its baseline value can be high and does not take a constant value [7].

Due to these reasons, **Adjusted rand index** is introduced, it is proposed in 1985[8]. Adjusted rand index basically is a normalized version of Rand index, it is given by following formula:

$$\text{Adjusted Rand Index} = \frac{\text{Rand Index} - \text{Expected Rand Index}}{\text{Maximum Rand Index} - \text{Expected Rand Index}} \tag{3.2}$$

Expected Rand Index is calculated by assuming the generalized hypergeometric distribution of randomness, we will not go into the details of this distribution. But after this normalization operation, Adjusted Rand index will be a better measurement for the difference between two clusters, it is bounded above by 1 and only equals to 1 if two partition are exactly the same, and close to zero when we randomly label the data into clusters. Furthermore, in practice, the value of the Adjusted Rand Index will be evenly ranging from 0 to 1.

In the simulation, we write our Rand Index function, but make use of the **Adjusted_rand_index()** function in sklearn library for efficiency purpose. Here is the graph showing the property of rand index and adjusted rand index.

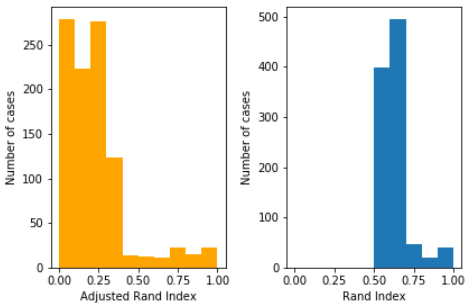

Figure 3: *We randomly generate 1000 pairs of partitions, then compute the Adjusted Rand Index and Rand Index for these pairs. It can be clearly seen that the value of Adjusted Rand Index is distributed across* $[0, 1]$*, while Rand Index without Normalization concentrate on* $[0.5, 1]$

4 Simulation Results

Following the framework proposed in section 3.2, below is the simulation we did for investigating the property of Γ-transformation: We start from the original dataset, rep-

resented by distance matrix X, then let $\Gamma = f(X)$, where f is the selected clustering algorithm we want to study. Apply Γ-transformation on X for multiple times to get the perturbed version of dataset $D = \{X'_1, X'_2 \dots X'_n\}$. Applying f to every element of D will result in a set of partitions $P = \{f(X'_1), f(X'_2) \dots f(X'_n)\} = \{\Gamma'_1, \Gamma'_2 \dots \Gamma'_n\}$. Finally, we calculate corresponding Adjusted Rand Index(ARI) for each of these partitions $\{ARI(\Gamma'_1, \Gamma), ARI(\Gamma'_2, \Gamma) \dots ARI(\Gamma'_n, \Gamma)\}$. *Table-1* present the psudo-code for the simulation steps.

Algorithm 1 Simulation Process

Input: Iteration time n, Clustering function f, Initial distance matrix d_{origin}

Output: A Set containing n Adjusted Rand Index $\{ARI_1, ARI_2 \dots ARI_n\}$, where $ARI_i = ARI(f(d_i), f(d_{origin}))$

1: $i = 1$
2: $\Gamma = f(d_{origin})$
3: **while** $i \leq n$ **do**
4: $d_i = \Gamma$-transformation(d_{origin})
5: $\Gamma_i = f(d_i)$
6: $ARI_i =$ AdjustedRandIndex(Γ_i, Γ)
7: $i = i + 1$
8: **end while**

4.1 Simulation for K-mediods

In the first experiment, f is set to be **k-mediods**. We try $n = 1000$, $n = 10000$ and 20000, *Figure-4* shows the raw frequency of the Adjusted Rand Index in these cases, together with the Adjusted Rand Index under random perturbed data. Compared with the distribution of ARI upon random perturbed data which is the *Figure-4*(a), the distribution of the ARI upon Γ-transformation clearly shows that Γ-transformation has the tendency to preserve the structure of the data. Even more remarkable, as $n \to \infty$, ARI tend to have a fixed **Probability Density Function**. In *Figure-5*, We draw the density plot of the ARI when $n = 10000, 20000 \dots 70000$ (the density is estimated by the sum of multiple Gaussian distribution), and it shows that the density function tends to converge to the

same function.

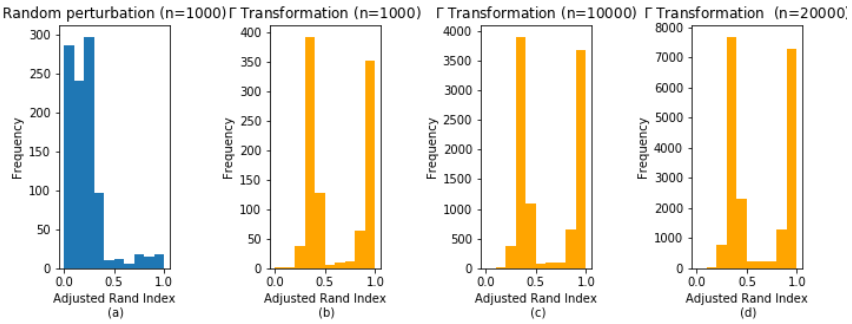

Figure 4: *Histogram of Adjusted Rand Index(ARI) for K-mediods.* n *distance matrices* $\{X'_1, X'_2 \dots X'_n\}$ *are created from* X_{origin} *by* Γ*-transformation,* Y *are created from random perturbation. (b),(c)and (d) shows histogram of* $ARI(f(X'_j), f(X_{origin}))$ *when* $n = 1000, 10000, 20000$ *respectively. (a) shows the histogram of* $ARI(f(Y), f(X_{origin}))$ *when* $n = 1000$.

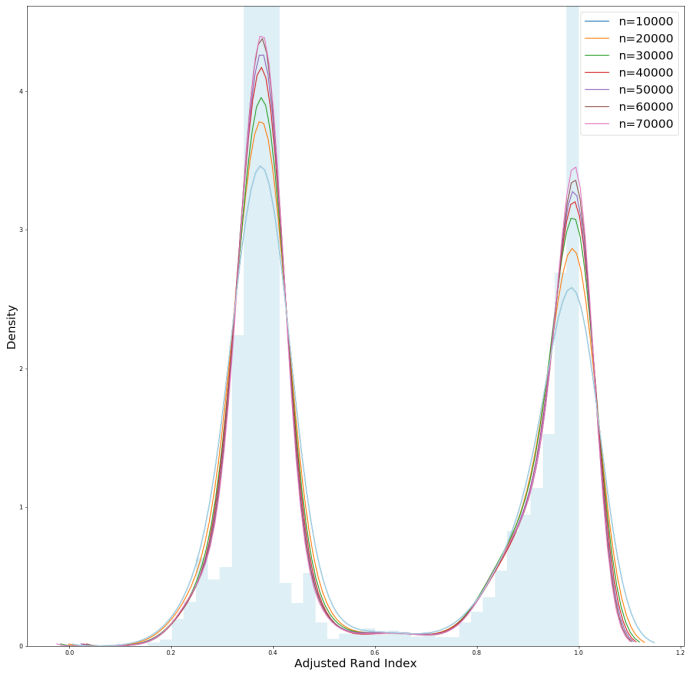

Figure 5: *We estimate the density function of* $ARI(f(X'_j), f(X_{origin}))$ *by the summing up Gaussian Distribution, this graph shows that as we increase the number of distance matrices, the density function converge to a "true" density function.*

4.2 Simulation for Complete Linkage

Complete-linkage is similar to Single-linkage Algorithm we discuss in the section 2.2, it also take a bottom up approach - start from regarding each data points as a cluster, then merge two clusters with least dissimilarity[3]. But the dissimilarity is defined in following manner.

Definition 4.1. *Let G,H represent two clusters, d is the distance function of the data set, then dissimilar* d_{SL} *is defined as:* $d_{SL}(G, H) = \max_{i \in G, i' \in H} d_{ii'}$

Here we let f be Complete-linkage with k-clusters termination condition, and go through the same procedure as we did in previous section. *Figure-6* shows the histogram of the ARI when $n = 1000$ and 4000. Compared to the histogram of K-mediods, it has

heavier tail at $ARI = 1$, which could be interpreted as an evidence for that Complete-linkage are "more consistency". We also try to estimate the probability density function of ARI, but this time, instead of using mixture Gaussian Distribution, we assume the density function has a Gamma-distribution, then use the data to estimate the parameter of it. 7 shows the estimated graph.

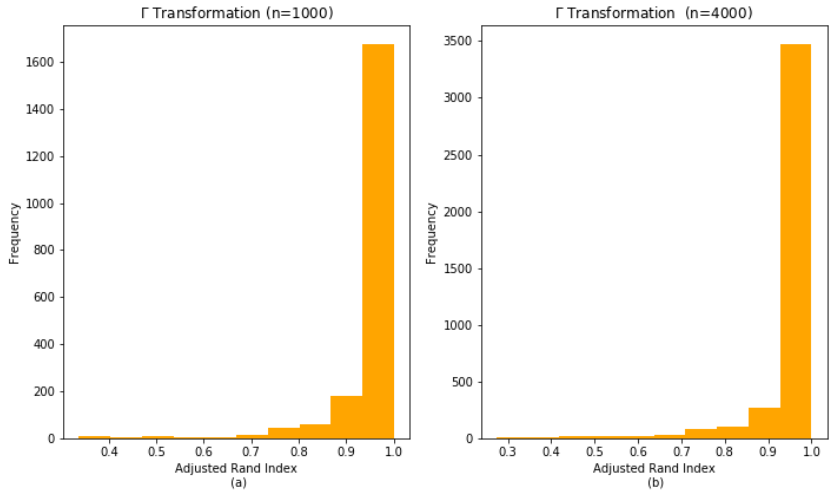

Figure 6: *Histogram of Adjusted Rand Index(ARI) for complete-linkage when* $n = 1000$ *and* 4000

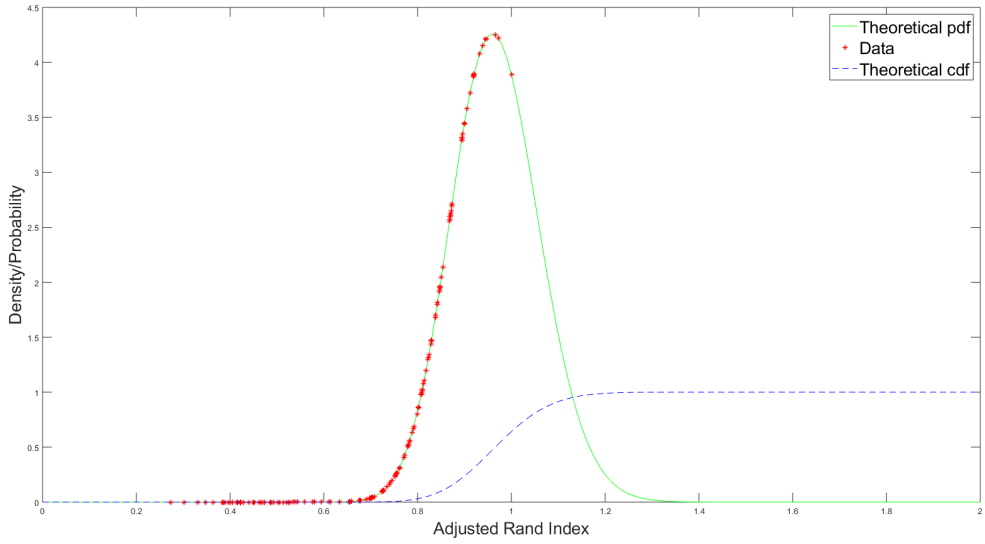

Figure 7: *Estimated density function and cumulated density function of ARI for Complete linkage*

4.3 Implication and limitation of the simulation

There are two important observations we get from the simulation process. The first one is that Adjusted Rand Index(ARI) after Γ-transformation had a very heavy tail at $ARI = 1$, which is consistent with our intuition for Γ-transformation - it shouldn't change the structure of data set. The second observation is that different clustering algorithms will have different ARI distribution, for K-mediods, it has two "peaks" at ARI = 1 and ARI = 0.4, while for Complete-linkage, distribution of ARI can be fitted relative good by Gamma distribution.

For Complete-linkage and K-mediods, Γ transformation lead to a skewed distribution of ARI (The relative frequency of $ARI = 1$ is higher than other value). Although we haven't tried all the clustering algorithm, it is reasonable to make the assumption that there are many other clustering algorithms satisfying this skewed distributed feature. So we formalize our assumption and state it as a refined version of consistency property, calling it β**-Partial Consistency**. Given $0 <= \beta <= 1$, it is defined as following:

Definition 4.2. *β-Partial Consistency. Clustering algorithm f satisfy β-Partial Consistency $\iff$ Given $D = \{d'_1, d'_2 \dots d'_n\}$ contains n Γ-transformation of distance function d, as $n \to \infty$, there exists at least $\beta * n$ number of $d'_i \in D$ satisfying $f(d'_i) = f(d)$*

Recall the original consistency property, it is actually a special case of β-Partial Consistency when $\beta = 1$. The original consistency property is a very strict one, only a small portion of clustering algorithm satisfy this property, but a wider range of clustering algorithms could satisfy β-Partial Consistency. From the simulation, we can see the potential existence of this property for **K-mediods** and **Complete-Linkage**. Figure-8 shows that the relative frequency of $ARI = 1$ could tend to a fixed value as we increase the value of n.

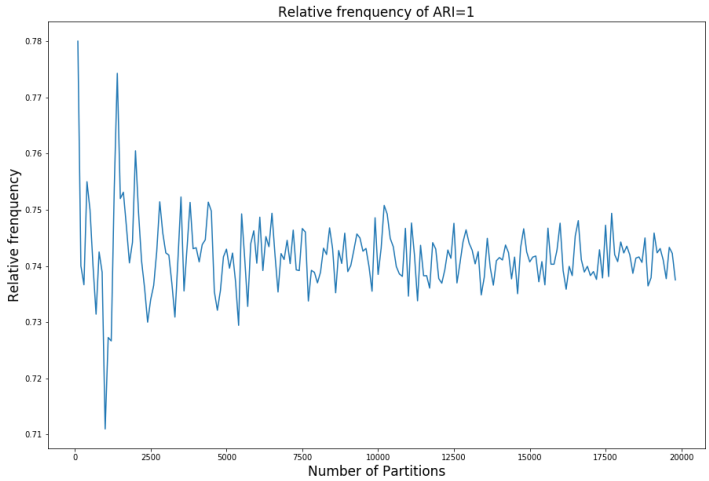

Figure 8: *Change of the relative frequency of ARI = 1. Y-axis shows* ***relative frequency of*** $1 = \frac{\textit{Frequency of ARI=1}}{\textit{Total number of cases}}$ *for Complete-linkage algorithm; X-axis shows the value of n. As n increase, this value tend to a fixed value*$= 0.74$.

This simulation process has several limitations: First of all, we only work on two of the clustering algorithms, so the simulation results could differ for other clustering algorithm. Secondly, we haven't proved that these two algorithms satisfy **Partial Consistency**, currently, it is only an assumption from simulation results, and mean to provide a direction for future study. Thirdly, our python implementation of Γ-transformation can only deal with distance matrix but not specific coordinates due to the limitation of computational resources.

5 Prediction With Machine Learning methods

For the clustering algorithm without consistency, in many cases, they can still produce the same partition on the perturbed dataset(perturbation means Γ-transformation). We would like to know when these clustering algorithm could produce the same partition. Put it more formally, the problem is: Give d' is the Γ-transformation of d, and particular clustering algorithm f, we would like to know whether $f(d) = f(d')$ holds without calculating $f(d)$ and $f(d')$. How can we do that? We solve this problem by regarding it as

a canonical classification task in machine learning setting. Details of model building and model evaluation are given in the following subsections, here we first state the application this method.

Suppose we have a lot of datasets $\{d_1, d_2 \dots d_n\}$which need to be processed by a clustering algorithm, we know that all of these datasets are the Γ-transformation of Dataset d, the clustering algorithm don't satisfy consistency property, but satisfy Partial Consistency. By the assumption we made in section 4, we know that many datasets will have the same partition results as dataset d, if we have a classifier C, and it can tell us which dataset d_i will have the same partition results as d, then we can skip these datasets. In this manner, we will save a lot of time and computation resources. Note, what motivate us to do this is the Partial Consistency assumption, if only a small amount of datasets have the same partition results as d, we will not bother using learning algorithm to train this model.

5.1 Approach

This method make use of the simulation process from *Section 4* to generate data. Given clustering algorithm f and initial dataset d, we can use the simulation process to generate multiple Γ-transformation of d: $\{d_1, d_2 \dots d_n\}$, then calculate corresponding Adjusted Rand Index:$\{ARI(f(d_1), f(d)),\ ARI(f(d_2), f(d)) \dots ARI(f(d_n), f(d))\}$. Now we have each pair (d_i, ARI_i) as a training sample, to make it a classification problem, we convert the Adjusted Rand Index to Binary label using η function, which is defined as following:

$$\eta(x) = \begin{cases} 0, & x < 1 \\ 1, & x = 1 \end{cases}$$

So formally, we have the training set $\{(d_1, \eta(ARI_1)), (d_2, \eta(ARI_2)) \dots (d_n, \eta(ARI_n))\}$. By fitting these data to learning algorithms, we can obtain different classifiers.

We don't go into the details about learning algorithms, we use decision tree, Support Vector Machine and Random Forest in this paper, which are all standard algorithms used in industry for small dataset.

5.2 Implementation and Experiment Results

6 Conclusions

A Remaining Proofs for Single-linkage

In this appendix, we provide the proof for *Theorem 2.2* and *Theorem 2.3*.

Proof. theorem 2.2

Given data set S and distance function d, and f be the single-linkage with k-clusters termination condition. Scale-Invariance of f is easy to prove, suppose another distance function d' satisfies

$$\forall i, j \in S, d'(i,j) = \beta d(i,j), where \beta > 0 \tag{A.1}$$

Following the same idea of proof 2.2, Let

$$\Gamma = f(S,d) = G_c(S,E) \tag{A.2}$$

$$\Gamma' = f(S,d') = G'_c(S,E') \tag{A.3}$$

If we can prove that edge sets $E = E'$, then $\Gamma = \Gamma'$ and scale invariance is proved.

$$E = \{e_1, e_2 \dots e_m\} \tag{A.4}$$

One key property of single linkage is that

$$d(e_i) <= d(e_k), \text{for } \forall e_i \in E, e_k \in E^{\complement} \tag{A.5}$$

On the other hand, E' is created by picking element with minimum $d'(e)$ until k clusters are formed. By Equation A.1, we have

$$d'(e_i) <= d'(e_k), \text{for } \forall e_i \in E, e_k \in E^{\complement} \tag{A.6}$$

So all the edges in E have to be included in E', and at this point, k clusters are formed, so the algorithm has to terminate. Thus, E' contains the exactly the same elements as E, which means Scale-Invariance is proved.

The proof for consistency is more subtle. We first need to define what do we mean by equivalent Edge sets.

Definition A.1. *two edge sets E_1 and E_2 are equivalent $\iff$ E_1 and E_2 represent the same partition to the dataset.*

Here is a example, in figure-9, $E_1 = \{e_1, e_2, e_8, e_9\}$ and $E_2 = \{e_2, e_5, e_7, e_8\}$ are **equivalent**, because both of them represent the same partition $\Gamma = \{\{1, 2, 3\}, \{4, 5, 6\}\}$.

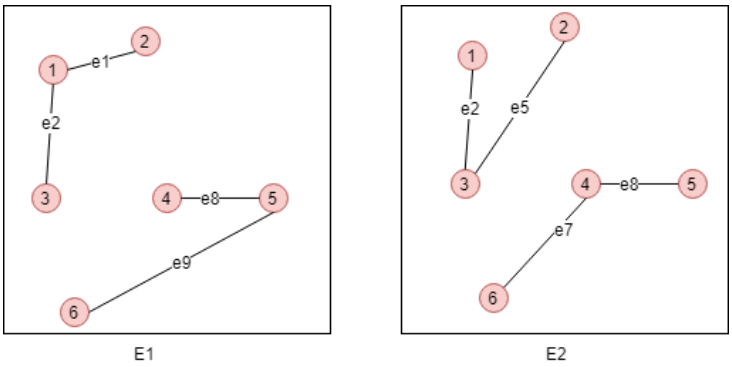

Figure 9: *E_1 and E_2 are two equivalent Edges sets*

Let d'' be the Γ-transformation of original distance function d, and $\Gamma_2 = f(S, d'') = G_c(S, E'')$. If we can prove that E and E'' are a equivalent, then we will be able to prove $\Gamma'' = \Gamma$, thus the consistency of f. With k-clusters termination condition, f will continue to pick the edges with small weight until k sub-graphs $\{G_1^{sub}, G_2^{sub} \dots G_k^{sub}\}$are formed. Then all the edges can be classified into two groups: in the first group, the edges connect two nodes within the same sub-graph; in the second group, the edges connect two nodes belongs to different sub-graphs. Formally,

$$M = \{(i, j) | i, j \in G_l^{sub}\} \tag{A.7}$$

$$M^{\complement} = \{(i, j) | i \in G_{l_1}^{sub}, i \in G_{l_2}^{sub}\} \tag{A.8}$$

Let $E = \{e_1, e_2 \dots e_m\}$, then we have $E \subseteqq M$ and

$$d(e_i) <= d(e_j), \quad \forall e_i \in E, e_j \notin E \tag{A.9}$$

After the Γ-transformation, we will have

$$d''(e_i) >= d(e_i), \quad \forall e_i \notin M \tag{A.10}$$

Combine equation A.9 and A.10 we have

$$d''(e_i) >= d'(e_j), \quad \forall e_i \notin M, e_j \in E \tag{A.11}$$

E'' will pick edges with small weight to include, but equation A.11 indicate that edges which are not in M(the edges between clusters), can never be selected, because the by the time these edges are selected, the edges in E either have been all included in E'', in that case, there are k clusters are formed, the selection process terminate,so $E = E''$, or the selection process terminate earlier, in this latter case, because no edges between the clusters are selected, E'' and E have to be the equivalent Edges set to form k sub-graphs. Thus $\Gamma = \Gamma''$. The consistency is proved. □

Proof. theorem 2.3

We first prove richness, for a given Partition $\Gamma = G_c(S, E)$, where $E = e_1, e_2 \dots e_m$. To make f produce such a Edge set, we can simply define d as following:

$$d(e_i) = \begin{cases} r, & i \in \{1, 2, 3, \dots m\} \\ r+1, & i \notin \{1, 2, 3, \dots m\} \end{cases}$$

Let $f(d) = G(S, E_{new})$. Because f is the single-linkage with distance-r termination condition, it will put all the edges of weight smaller or equal to r into Edge set, so $E_{new} = \{e_1, e_2 \dots d_m\} = E$

To prove the consistency of f, we can follow the same idea as we did in last proof. For Edges in E, we have

$$d(e_i) > r, \quad \forall e_i \notin E \tag{A.12}$$

And because of equation A.11, we have

$$d''(e_i) >= d(d_i) > r \quad \forall e_i \notin M \tag{A.13}$$

So these Edges connecting between the clusters can never be included in the Edge Set, On the other hand, we have

$$d''(e_j) <= d(e_j) <= r \quad \forall e_i \notin E \tag{A.14}$$

So E have to at least include these edges, which form the same clusters as Γ, and whether or not include the other inter-connected edges will in E'' will not change the fact that E'' and E are equivalent Edge set. So $\Gamma'' = \Gamma$, we prove the consistency.

□

B Pyhton and Matlab code

Below is the code for Γ-transformation. We also enclose the matlab code for fitting the data to Gamma-distribution

```
import numpy as np
import random
from math import factorial
# take the similarity matrix, partition result
def perturb_distance_matrix(in_matrix,partition):
    matrix = np.matrix.copy(in_matrix)
    #take the index of first,second ... cluster
    partition_list = list()
    for i in np.unique(partition):
        index_i = np.where(partition==i)
        if len(index_i[0])!=1:
            #In one cluster: Randomly select the x pair of points
            temp_upper_bound = factorial(len(index_i[0]))/(factorial(len(index_i[0])-2)*2)
            loop_time = random.randint(1,temp_upper_bound+1)
            #Randomly reduce the distance between each pair
            for j in range(1,loop_time+1):
                pair = np.random.choice(index_i[0],2,False)
                matrix[pair[0],pair[1]]=  matrix[pair[1],pair[0]]= random.uniform(0, 1)*matrix[pair[1],pair[0]]
            #store the partition into the list
        partition_list.append(index_i[0])
    #Ramdonmly select x' pair of clusters
    temp_upper_bound = factorial(len(partition_list))/(factorial(len(partition_list)-2)*2)
    loop_time = random.randint(1,temp_upper_bound+1)
    #increate the distance between clusters with a random ratio
    for i in range(1,loop_time+1):
        cluster_pair = random.sample(partition_list,2)
        #create a ramdom generated matrix - then do elementwise multiplication
```

```
28          increase_coefficient = np.multiply((np.random.rand(cluster_pair[0].
    shape[0],cluster_pair[1].shape[0])+1),(np.random.randint(4,size=(
    cluster_pair[0].shape[0],cluster_pair[1].shape[0]))+1))
29          temp_multiplier = np.multiply(matrix[np.ix_(cluster_pair[0],
    cluster_pair[1])],increase_coefficient)
30          matrix[np.ix_(cluster_pair[0],cluster_pair[1])]= temp_multiplier
31         # the points are symmetric to each other, so do the corresponding
    changes
32          matrix[np.ix_(cluster_pair[1],cluster_pair[0])] = np.matrix.
    transpose(temp_multiplier)
33
34      return matrix
```

Listing 1: Γ-transformation()

```
1 % Fit gamma pdf to data
2 data = table2array(foo)
3 ab   = gamfit(data);
4 % Write the functional form of the pdf with the estimated parameters
5 a    = ab(1);
6 b    = ab(2);
7 gpdf = @(x) x.^(a-1)/(b^a*gamma(a)).*exp(-x/b);
8 % Plot the fitted pdf and the data
9 x    = 0:0.01:2;
10
11 plot(x,gpdf(x),'-g',data,gpdf(data),'*r',x,gamcdf(x,a,b),'--blue')
12 xx=xlabel('Adjusted Rand Index')
13 xx.FontSize = 20;
14 yy=ylabel('Density/Probability')
15 yy.FontSize = 20;
16 lgd=legend('Theoretical pdf','Data','Theoretical cdf')
17 lgd.FontSize = 20;
```

Listing 2: fitting the data to Γ distribution

References

[1] Jon M Kleinberg. An impossibility theorem for clustering. In *Advances in neural information processing systems*, pages 463–470, 2003.

[2] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[3] D Manning Christopher, Raghavan Prabhakar, and Schutza Hinrich. Introduction to information retrieval. *An Introduction To Information Retrieval*, 151(177):5, 2008.

[4] Alex Williams. Is clustering mathematically impossible, 2015.

[5] Marina Meilă. Comparing clusterings by the variation of information. In *Learning theory and kernel machines*, pages 173–187. Springer, 2003.

[6] Ulrike Von Luxburg et al. Clustering stability: an overview. *Foundations and Trends® in Machine Learning*, 2(3):235–274, 2010.

[7] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010.

[8] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.