CLUSTERING NETWORKS WITH NODE ATTRIBUTES USING BREGMAN DIVERGENCES

Felipe Schreiber Fernandes

Tese de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Daniel Ratton Figueiredo
Maximilien Dreveton

Rio de Janeiro
Novembro de 2023

CLUSTERING NETWORKS WITH NODE ATTRIBUTES USING BREGMAN DIVERGENCES

Felipe Schreiber Fernandes

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Daniel Ratton Figueiredo
Maximilien Dreveton

Aprovada por: Prof. Daniel Ratton Figueiredo
Prof. Maximilien Dreveton
Prof. Andressa Cerqueira
Prof. Valmir Carneiro Barbosa

RIO DE JANEIRO, RJ – BRASIL
NOVEMBRO DE 2023

Schreiber Fernandes, Felipe

Clustering Networks with Node Attributes using Bregman Divergences/Felipe Schreiber Fernandes. – Rio de Janeiro: UFRJ/COPPE, 2023.

XI, 76 p.: il.; 29, 7cm.

Orientadores: Daniel Ratton Figueiredo

Maximilien Dreveton

Tese (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2023.

Referências Bibliográficas: p. 71 – 76.

1. Network clustering. 2. Bregman Divergences. 3. Statistical Inference. I. Ratton Figueiredo, Daniel *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedico à minha família e amigos.

Agradecimentos

Gostaria de agradecer aos meus pais, Sofia e Eduardo, por apoiarem a continuação dos meus estudos.

Aos meus amigos que proporcionaram momentos de descontração, sem os quais não seria possível esse trabalho.

Aos meus orientadores, Maximilien Drevton e Daniel Ratton, pela compreensão, paciência e dedicação na pesquisa.

Igualmente gostaria de agradecer aos professores Bernardo Freitas e Heudson Mirandola do Instituto de Matemática que me apresentaram ao mundo da Geometria Diferencial e Otimização.

Ao PESC, por me aceitar e acreditar no meu potencial.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

CLUSTERING NETWORKS WITH NODE ATTRIBUTES USING BREGMAN DIVERGENCES

Felipe Schreiber Fernandes

Novembro/2023

Orientadores: Daniel Ratton Figueiredo
Maximilien Dreveton

Programa: Engenharia de Sistemas e Computação

O clustering de rede aborda o problema de identificação de conjuntos de nós (comunidades) que possuem padrões de conexão semelhantes. No entanto, em muitos cenários, os nós também têm atributos que geralmente estão correlacionados com a estrutura de cluster. Assim, as informações da rede (arestas) e informações do nó (atributos) podem ser aproveitados em conjunto para projetar algoritmos de clustering de alto desempenho. Sob um modelo geral para a rede e atributos do nó, este trabalho apresenta um algoritmo de clustering iterativo que maximiza a verossimilhança conjunta, assumindo que a distribuição de probabilidade das interações na rede e os atributos dos nós pertencem a famílias exponenciais. Este modelo cobre uma ampla gama de possíveis interações (por exemplo, arestas com pesos) e atributos de nó (por exemplo, distribuições não gaussianas), bem como redes esparsas, ao mesmo tempo que explora a conexão entre famílias exponenciais e divergências de Bregman permitindo uma expressão elegante para a função de log-verossimilhança. Extensos experimentos numéricos usando dados sintéticos indicam que o algoritmo proposto supera algoritmos clássicos que aproveitam apenas informações de rede ou apenas atributos bem como algoritmos de última geração que também aproveitam ambas as fontes de informação. A avaliação preliminar utilizando conjuntos de dados reais também indica a superioridade da proposta abordagem na detecção de comunidades. As contribuições deste trabalho fornecem insights sobre as técnicas práticas para inferir rótulos de comunidade em redes atribuídas a nós.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

CLUSTERING NETWORKS WITH NODE ATTRIBUTES USING BREGMAN DIVERGENCES

Felipe Schreiber Fernandes

November/2023

Advisors: Daniel Ratton Figueiredo
Maximilien Dreveton

Department: Systems Engineering and Computer Science

Network clustering tackles the problem of identifying sets of nodes (communities) that have similar connection patterns. However, in many scenarios, nodes also have attributes that are often correlated with the clustering structure. Thus, network information (edges) and node information (attributes) can be jointly leveraged to design high-performance clustering algorithms. Under a general model for the network and node attributes, this work presents an iterative clustering algorithm that maximizes the joint likelihood, assuming that the probability distribution of network interactions and node attributes belong to exponential families. This model covers a broad range of possible interactions (e.g., edges with weights) and node attributes (e.g., non-Gaussian distributions), as well as sparse networks, while also exploring the connection between exponential families and Bregman divergences allowing for an elegant expression for the log-likelihood function. Extensive numerical experiments using synthetic data indicates that the proposed algorithm outperforms classic algorithms that leverage only network or only attribute information as well as state-of-the-art algorithms that also leverage both sources of information. Preliminary evaluation using real datasets also indicate the superiority of the proposed approach in detecting communities. The contributions of this work provide insights into the practical techniques for inferring community labels on node-attributed networks.

Contents

List of Figures

Chapter 1

Introduction

1.1 Network Clustering

Many domains of science have made efforts to study systems composed of several elements. In this sense, not only the elements individually, but also the interaction between them is of great importance towards a broad understanding of the object of study. In this sense, graphs emerge as a powerful abstraction capable of representing different relationships. In a graph, vertices represent entities and edges encode the relation between them. Some examples are: computer network communication, where vertices represent endpoints, commuters and routers, and edges represent the links connecting them, or biological networks where vertices may represent human tissue proteins and edges the interactions between them.

Furthermore, in many scenarios the structure is closely related with the function performed: at proteins network, OhmNet [16] have shown capable of, given the network structure and the representations found in an unsupervised manner, predict the labels (protein roles at each human tissue), in bitcoin Elliptic network [17] it was possible, from an supervised approach, predict whether the transaction was illicit or not. To summarize, Network Science aims to study these complex relationships and infer the role of an element given his structural position and has been gaining a lot of importance over the last two decades.

In order to motivate the community detection problem, one can cite two contexts in which it plays an important role: social networks and biological networks. The first one is very intuitive, since people tend to interact more with others in the same social context, e.g., coworkers in a company are more likely to be friends than with people from other companies. The second, although less intuitive, aims to study how cellular networks are organized and correlated with biological functions, e.g., proteins that belongs to the same human tissue are more likely to interact with each other. Next, one example to illustrate each domain is shown.

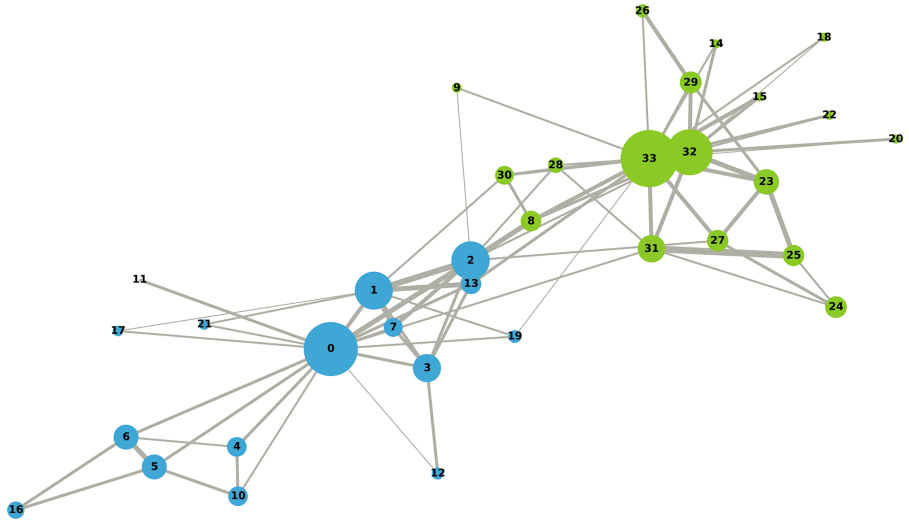


Figure 1.1: Zachary Karate Club. Taken from [1]

A very common network in literature is the Zachary's Karate Club. This network is composed by 34 vertices and 78 edges. Everybody in the club knew each other, but the sociologist Wayne Zachary annotated the links between people who displayed regular interactions outside the club. So one may ask what makes this network so interesting. There was a divergence between the president of the club and the instructor, hence students followed one or the other, splitting the community in two. This splitting unveiled the underlying community structure and is directly related with the connections in the graph.

Another popular application of community detection is in metabolic networks. In this sense, Ravasz and collaborators [2] made the first attempt to identify groups in such systems. In the experiment, they not only found community structures, but also shown their hierarchical organization, which is depicted in Figure 1.2, taken from [18]. There, each color is associated with predominant biochemical class of the molecule. Also, one can see that the groups found have some degree of overlapping, which can be seen as the generalization of the problem of finding hard communities. Indeed, communities play an important role in human diseases, such that proteins of the same disease are likely to interact. This finding was crucial to develop many hypothesis in health sciences.

In this context, community detection is a fundamental problem, since it unveils the inherent network structure. It consists of finding subsets of nodes where nodes within each subset are strongly linked with respect to nodes outside the subset. Formally, communities are described as a locally dense connected subgraph in a network. This relies on two hypothesis:

- Connectedness: Each community is composed of connected subgraphs.
- Density: Members inside the same community are more likely to have connec-

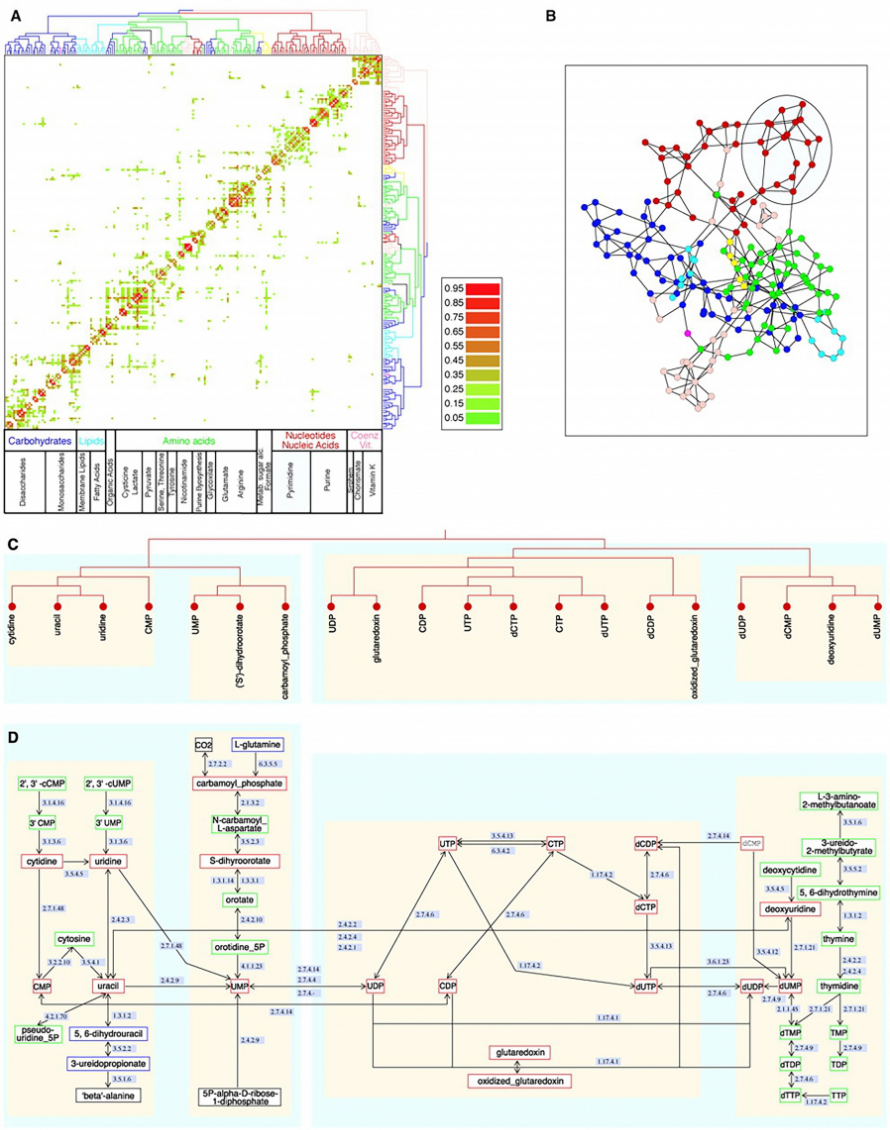


Figure 1.2: Hierarchical clustering of molecules. Taken from [2]

tions with members from the community than with others outside the community.

Hence, several definitions are consistent with these hypothesis, like:

- Maximum Cliques: Groups of elements that form a complete subgraph
- Strong Community: The subgraph C forms a strong community if, for each vertex $i \in C$, the number of connections with other nodes inside the community, also called internal degree, is greater than the number of connections outside, called external degree. Let $k_i^{int}(C)$ be the internal degree of node i in partition C, and $k_i^{ext}(C)$ the external degree. Hence: $k_i^{int}(C) > k_i^{ext}(C)$.
- Weak Community: The subgraph C forms a weak community if $\sum_{i \in C} k_i^{int}(C) > \sum_{i \in C} k_i^{ext}(C)$

Community detection is also known as graph partitioning, a classic combinatorial problem in graph theory. For example, the bisection problem consists in finding two subsets of nodes of equal size such that the edge cut size between them is minimized. In this work we consider the problem of detecting communities with and without overlap. Vertices may belong to multiple communities simultaneously, for example, a person can have her group of family members, group of friends from school and group of friends from college, and so on, which turns the objective even more challenging.

1.2 Data Clustering

In the era of big data, one may be interested in extracting patterns on the data, i.e., finding trends that are useful for business, medicine, and so on so forth. For this end, computational intelligence and data mining disciplines have emerged as valuable tools for data analysis and new knowledge discovery, and in this thesis we focus on the problem of partitioning data into subgroups. Normally, one may divide the problem in two main approaches. The first, as known as supervised task, is when labels for each data point is provided, and an algorithm find partitions by minimizing an objective function that depends on such labels. The second, as known as unsupervised task, occurs when one aims to find partitions based solely on the set of observations, without any sort of labels. In this work we focus on the latter, using cluster analysis to find partitions where data points are somehow similar in a broad sense.

For example, one can analyze the problem formulation of fuzzy c means. Given $X = \{x_1, x_2, x_3, ..., x_n\}$, being $x_k, k \in \{1, 2, ..., n\}$, a characteristics vector $\in \mathbf{R}^P$ such that $x_k = [x_{k1}, x_{k2}, ..., x_{kp}]$, the fuzzy clusterization algorithm consists in finding a

partition $A = \{A_1, A_2, ..., A_c\}$ that best represents the data, i.e., minimizes the following objective function:

$$J_m = \sum_{i=1}^{n} \sum_{j=1}^{c} A_{ij}^m ||x_i - v_j||^2 \tag{1.1}$$

Here A is a matrix, and $A_{ij} = A_j(x_i)$ represents the belonging of point x_i in cluster j. On the other hand, J_m measures the dissimilarity between x_i points and the centers v_j. It is worth remembering that c is an input variable that determines the amount of clusters. In the following chapters, the notion of similarity, as well as dissimilarity, is better explored using a class of functions known as Bregman divergences. In this way, we further define an objective function so that we can algorithmically find partitions with some characteristics.

1.3 Contribution

While graph clustering and data clustering have been studied independently for a long time, it is only recently that they have been unified into a single formulation. In attributed networks, each node in the network is associated with vector $\in \mathbf{R}^P$. Hence, nodes can be clustered using only the node attributes (data clustering), only the network structure (graph clustering), or a combination of both (joint clustering). This work focus on the latter, where both attribute information and network structure are considered together. In particular, the following contributions are made:

- Two clustering algorithms using pseudo-likelihood assuming exponential families
- Extensive algorithm evaluation and comparison with others
- Preliminary evaluation considering real datasets

The remainder of this document is organized as follows: In Chapter 2 we present background and related works. In Chapter 3 we bring a theoretical perspective behind the Bregman divergences. In Chapter 4 we present the framework along with some variations. In Chapter 5 the methodology used is presented along with the evaluation procedure. In Chapter 6 we end with the conclusion and show future works.

Chapter 2

Background and Related Works

Communities play an important role in many fields, since in many contexts the network is closely related to the function of a vertex. However, it is necessary to discuss what we mean by the word structure and what kind of problems arise from it. The main basis behind the relationship between structure and function is that in many areas it is observed that vertices which display similar characteristics tend to form connections, in contrast with vertices that are far from each other in the network. So one may ask, how are the vertices mixed/connected? Is there a non-trivial community structure? Hence it is also necessary to specify a metric to capture this pattern. A common metric present in the literature is the assortativity coefficient [19]. Let n_{ij} be the number of edges between nodes that display a characteristic i and nodes that display a characteristic j and e_{ij} the average, i.e., n_{ij} normalized by the total amount of edges, m:

$$\begin{aligned} e_{ij} &= n_{ij}/m \\ \sum_{ij} e_{ij} &= 1 \\ \sum_{j} e_{ij} &= d(i) \\ \sum_{i} e_{ij} &= d(j), \end{aligned} \tag{2.1}$$

where $d(i)$ is the total amount of edges incident to vertices that display characteristic i and $d(j)$ the total amount of edges incident to vertices that display characteristic j. Now one can easily derive the *assortativity coefficient* by comparing the observed e_{ij} with the expected value if the configuration model is assumed, where the edges are positioned by random:

$$r = \frac{\sum_i e_{ii} - \sum_i d(i)d(i)}{1 - \sum_i d(i)d(i)} \tag{2.2}$$

The denominator of the equation above is just a normalization factor, which represents the maximum possible value, i.e., when there are only edges between vertices of same type. From the equation above, if $r = 0$ then the connections are totally random, otherwise if $r > 0$ then there is a trend to form connections between nodes with similar characteristics.

Now, the next question one may be wondering to answer is: Which nodes are in each community? This latter one is the focus of the present work, and can be divided in two sub categories: Non-overlapping and overlapping community detection.

2.1 Non Overlapping Community Detection

In this approach the objective is to find a partition $P = \{A_1, ..., A_k\}$ that divides the nodes in k non-overlapping subsets, where each node belongs to one single community. In this sense, many works have arisen in the literature and traditional methods are broadly divided in divisive and agglomerative. The former works as follows: at each iteration nodes (or subsets) that are similar according to a specific metric are grouped together, until all the nodes are in the same cluster or a termination condition is reached. A popular one in this category is the Girvan-Newman algorithm [20][21]. The latter process begins with all nodes belonging to a single community, and then removes edges one by one until all nodes are considered its own community, producing a dendogram, which contains information about hierarchical communities, e.g. the famous Ravasz Algorithm [2].

Other two common methods are, respectively, the modularity and cut optimization. However the second will be introduced in Subsection 2.2 where we discuss Overlapping Community Detection, since it can be easily extended to that case. The former seeks to maximize the following objective function:

$$M = \frac{1}{2m} \sum_{k} \sum_{i,j \in V_k} \left(A_{ij} - \frac{d_{ij}}{2m} \right) \tag{2.3}$$

In the equation 2.3 m is the total number of edges in the network, A_{ij} is the total number of edges inside the partition V_k and d_{ij} is the expected amount of edges under the configuration (random) model. The configuration model is also a synthetic network, and can be defined as a network where each pair of N nodes is connected with probability p (Gilbert definition [22]), or as a network of N nodes with L random links (Erdős and Rényi definition [23]). In this way, the modularity represents how strong are the interactions, or how far we are from the configuration model. Common algorithms for this problem are the following, proposed by Newman [24] and Louvain[25]:

- Newman: First, each vertex is assigned with a different partition. Next, compute the modularity gain of merging two partitions for each pair. Merge the pairs that result in highest gain. Save the partition and modularity value for the respective step. Return to step 2, computing modularity gain, and repeat the process until there is only one partition. Return the partition that has the highest value of modularity. The complexity of this algorithm is $O((n+m)m)$.

- Louvain: It's also an heuristic algorithm, and very similar to Newman, but works in $O(m)$ time. First every vertex is assigned with a different partition. Next, compute the modularity gain of moving the node to its neighbor partition, for every pair. Merge the pair that results in greater gain. Create a new network were the merged parts become a single vertex. Repeat step 2 until there is not any improvement.

Alternatively to the modularity function, Louvain can work with any partition quality function $\mathcal{H}(G, P)$, such as the Constant Potts Model, defined as:

$$\mathcal{H}(G, P) = \sum_{C \in P} \left[E(C, C) - \gamma \binom{||C||}{2} \right], \tag{2.4}$$

with E(C,D) denoting the number of edges between nodes in communities C and D.

However, it is known that Louvain algorithm suffers from a couple of problems, which induce badly connected communities. First, it can find communities that are internally disconnected, as pointed out in [3]. Consider the following example in figure 2.1: At a certain point of Louvain we can move a node (with label zero in figure) that was acting like a bridge internally to another community such that increases modularity. Hence, once the disconnected community becomes a node in a new aggregated graph, it remains disconnected unless merged with another community that will act like a bridge. Second, the communities found by Louvain may contain significant substructure, i.e., subgraphs that are densely connected. With these problems in mind, Leiden algorithm [3] tries to mitigate such shortcomings. It works as stated in algorithm 1.

In Leiden, the first step, called MoveNodesFast, tries to move every node to another community, but is more efficient than Louvain because considers a queue only with nodes which neighbor's community has changed. The second step, called RefinePartition, merges nodes from partion P, but instead of greedily merging, it assigns a probability of merging to community C that is proportional to $\exp \Delta H$, where ΔH is the increase of partition quality function. Finally, the last step is aggregation phase, which builds a new graph based on the partition $P_{refined}$. Further details and guarantees may be found in their article, Appendix A.2.

Algorithm 1 Leiden

Require: : graph G, partition P
  do
    $P \leftarrow MoveNodesFast(G, P)$
    $done \leftarrow |P| = |V(G)|$
    if not done **then**
      $P_{refined} \leftarrow RefinePartition(G, P)$
      $G \leftarrow AggregateGraph(G, P_{refined})$
      $P \leftarrow \{\{v | v \subseteq C, v \in V(G)\} | C \in P\}$
  while not done

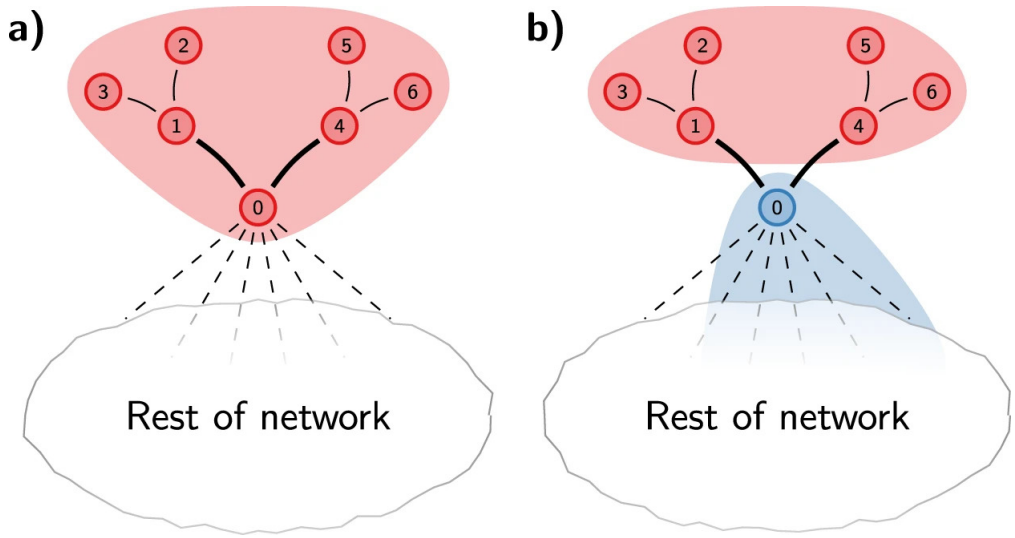


Figure 2.1: Louvain step may induce badly connected communities. Taken from [3]

2.2 Overlapping Community Detection

Here the problem is harder. Given a network, the goal is to determine the communities, just like the previous question, but here each vertex may belong to one or more communities. This is still an object of research, but two main algorithms are CFinder[26] and LinkClustering [27]. The former relies on the definition that a community is the set of k-cliques that shares $k-1$ neighbors, where k is a parameter of the algorithm. The latter is based on clustering links (edges) that are somehow similar. Consider an edge e_{uv} that connects vertices u and v and e_{uw} that connects vertices u and w respectively. A similarity measure between them can be given by the following equation:

$$S(e_{uv}, e_{uw}) = \frac{|n_+(v) \bigcap n_+(w)|}{|n_+(v) \bigcup n_+(w)|} \tag{2.5}$$

In the equation 2.5 $n_+(v)$ is the list of the neighbors of node v, including itself. Hence it measures the fraction of common neighbors that node v has in common with node w divided by the total number of neighbors. In the literature it is known

as Jaccard index.

Next, the Generalized Spectral Clustering method is presented, which is a procedure to represent nodes in Euclidean Space followed by soft clustering.

2.2.1 Generalized Spectral Clustering

The backbone of Generalized Spectral Clustering [28] method relies on the spectral graph theory and, more precisely, at the properties of Graph Laplacians. A common way to represent graphs is by adjacency matrices, which essentially contains all the information necessary. However there are other ways to represent graphs, such as performing several transformations in adjacency matrices, which are called Laplacians. One of such transformations is the unnormalized Laplacian, and is given by:

$$L = D - A, \tag{2.6}$$

where D is the degree diagonal matrix and A is the adjacency matrix. This transformation has some interesting properties, such as:

- L is symmetric ($L^T = L$) and positive definite ($x^T Lx \geq 0, \forall x \in R^{|V|}$)
- $x^T Lx = \sum_{u-v \in E}(x[u] - x[v])^2$
- L has $|V|$ non negative eigenvalues

These properties are convenient when we talk about graph cuts and finding communities. Let's first see the case in which one is interested in finding two non-overlapping communities in a graph. In this case the problem can be reduced to finding the minimum cut. Let A_1 denote a cluster and $\bar{A}_1 = A_2$ its complementary, i.e., $A_1 \bigcup \bar{A}_1 = V$, and $A_1 \bigcap \bar{A}_1 = \emptyset$. Then the minimum cut problem can be expressed by the following objective function, considering K = 2:

$$cut(A, \bar{A}) = \frac{1}{2} \sum_{k=1}^{K} |(u, v) \in \{E\} : u \in \{A_k\}, v \in \{\bar{A}_k\}| \tag{2.7}$$

The equation 2.7 means simply that one is interested in minimizing the number of edges between communities. However, imagine that there is a vertex with degree equal 1. Then by the above objective function this vertex could be considered a community while the rest of the network another. This is a problem since one would like to see a non trivial solution where the communities contain a sufficient large amount of nodes. Hence, a common a approach is to minimize the ratio cut

instead, which penalizes communities formed by a small number of nodes:

$$RatioCut(A, \bar{A}) = \frac{1}{2}\sum_{k=1}^{K}\frac{|(u,v) \in \{E\} : u \in \{A_k\}, v \in \{\bar{A}_k\}|}{|A_k|} \tag{2.8}$$

The above equation can be rewritten in function of the Laplacian. Consider a vector $a \in \mathcal{R}^{|V|}$ such that:

$$a[u] = \begin{cases} \sqrt{\frac{\bar{A}}{A}} & if\ u \in A \\ -\sqrt{\frac{A}{\bar{A}}} & if\ u \in \bar{A} \end{cases} \tag{2.9}$$

Now, applying the second property of the Laplacian we have that:

$$\begin{aligned} a^T La &= \sum_{u,v \in \{E\}} (a[u] - a[v])^2 \\ &= \sum_{(u,v):u \in A, v \in \bar{A}} (a[u] - a[v])^2 \\ &= \sum_{(u,v):u \in A, v \in \bar{A}} \left(\sqrt{\frac{|\bar{A}|}{|A|}} - (-\sqrt{\frac{|A|}{|\bar{A}|}})\right)^2 \\ &= cut(A, \bar{A})\left(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2\right) \\ &= cut(A, \bar{A})\left(\frac{|\bar{A}| + |A|}{|A|} + \frac{|A| + \bar{A}}{|\bar{A}|}\right) \\ &= |V| RatioCut(A, \bar{A}) \end{aligned} \tag{2.10}$$

So it can be seen that the Ratio Cut minimization problem can be written in terms of the Laplacian, since the $|V|$ multiplying factor doesn't change the optimal solution. To summarize, and adding the constraints relative to the vector a, the final objective function is:

$$\begin{aligned} min_{A \subset V} a^T La \\ a \perp 1 \\ ||a||^2 = |V| \\ \text{a defined as in Equation 2.9} \end{aligned} \tag{2.11}$$

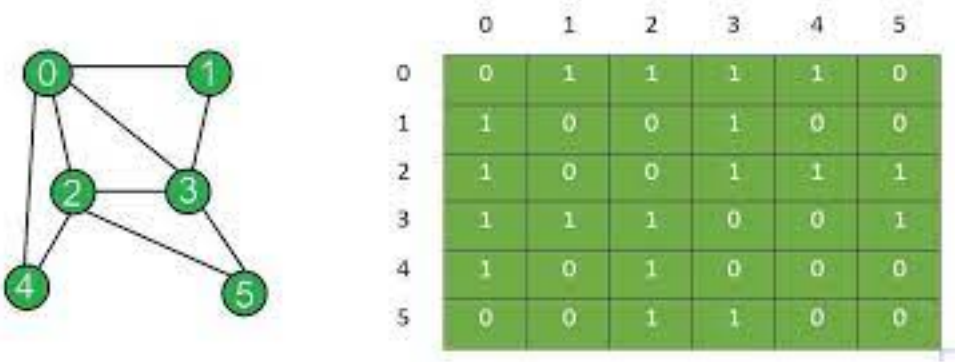
Removing the the last constraint, i.e., allowing a assume any continuum value:

$$\begin{aligned} min_{a \in \mathbf{R}^{|\mathbf{V}|}} a^T La \\ a \perp 1 \\ ||a||^2 = |V| \end{aligned} \tag{2.12}$$

Note, however, that the constraint removal allowed a relaxation of the problem.

If that equation was taken into account, the function to optimize would be NP-Hard because we would be minimizing over a discrete set. Finally, the solution of the above objective function is given by the second smallest eigenvector of the Laplacian, according to the Rayleigh-Ritz theorem. Up to now only the separation into two clusters was considered. In order to generalize to an arbitrary amount K of clusters, one can obtain the K smallest eigenvectors of the Laplacian, represent each vertex by the row of the matrix U formed by the concatenation of the K smallest eigenvectors and then apply any clusterization method, like k-means, on the embeddings obtained.

For example, let $G(V, E)$ be a graph of $|V|$ vertices, $|E|$ edges and A be the adjacency matrix of G such that $A(u, v) = 1$ if u is neighbor v and 0 otherwise. Hence in graph of Figure 2.2, the Laplacian is given by:



	0	1	2	3	4	5
0	0	1	1	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	1	1
3	1	1	1	0	0	1
4	1	0	1	0	0	0
5	0	0	1	1	0	0

Figure 2.2: Adjacency Matrix example

$$L = \begin{bmatrix} 4 & -1 & -1 & -1 & -1 & 0 \\ -1 & 2 & 0 & -1 & 0 & 0 \\ -1 & 0 & 4 & -1 & -1 & -1 \\ -1 & -1 & -1 & 4 & 0 & -1 \\ -1 & 0 & -1 & 0 & 2 & 0 \\ 0 & 0 & -1 & -1 & 0 & 2 \end{bmatrix}$$

It can be seen that L have the following eigenvalues and eigenvectors:

$\lambda_0 = 0, v_0 = \langle 1, 1, 1, 1, 1, 1 \rangle$

$\lambda_1 = 4, v_1 = \langle -1, 1, -1, -1, 1, 1 \rangle$

$\lambda_2 = \frac{-\sqrt{13}+7}{2}, v_2 = \left\langle 0, -1, \frac{\sqrt{13}-3}{2}, \frac{-\sqrt{13}+3}{2}, 1, 0 \right\rangle, v_3 = \left\langle \frac{-\sqrt{13}+3}{2}, -1, \frac{\sqrt{13}-3}{2}, 0, 0, 1 \right\rangle$

$\lambda_3 = \frac{\sqrt{13}+7}{2}, v_4 = \left\langle 0, -1, \frac{-\sqrt{13}-3}{2}, \frac{\sqrt{13}+3}{2}, 1, 0 \right\rangle, v_5 = \left\langle \frac{\sqrt{13}+3}{2}, -1, \frac{-\sqrt{13}-3}{2}, 0, 0, 1 \right\rangle$

The lowest eigenvalue is $\lambda_2 = \frac{-\sqrt{13}+7}{2}$. If we would like to cluster the vertices in $k = 2$ groups, we would have the following U matrix, composed of $\left[v_2^T, v_3^T\right]$:

$$U = \begin{bmatrix} 0 & \frac{-\sqrt{13}+3}{2} \\ -1 & -1 \\ \frac{\sqrt{13}-3}{2} & \frac{\sqrt{13}-3}{2} \\ \frac{-\sqrt{13}+3}{2} & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Finally, the vertex 0 would be represented in $\mathbf{R^2}$ by the point which corresponds to row 0, i.e., $(0, \frac{-\sqrt{13}+3}{2})$, vertex 1 by row 1 (-1, -1), and so on. The last step is apply a clusterization tecnique, such as Gaussian Mixture Models [29], in order to allow community overlapping.

2.2.2 Stochastic Block Model

Before going into other clustering algorithms, it's important to describe some assumptions about the network which are commonly made by them. Hence, in the following subsection we will describe the Stochastic Block Model, the generative network model and an inferencial model with such assumption.

First let's consider the data generation. Let k be the number of communities, and n_k be the number of nodes inside of community k, where $n_k, k \in \mathbf{N}$. Let also Θ be a $k \times k$ matrix such that Θ_{rs} is the probability of an edge leaving the group (community) r and reaching group s, and Θ_{rr} is the probability of existing an edge between nodes inside the same community. Hence, since we desire to evaluate the algorithm at the situation where there is an evident community structure, we would like that $\Theta_{rr} > \Theta_{rs}, \forall r, s \in 1, 2, .., k$.

In this work, we consider that $n_k = n$ and $\Theta_{rr} = p_{\text{in}}$, i.e., the number of nodes is the same in every community and the diagonal of the matrix is constant for every community. Off the matrix diagonal we have $\Theta_{rs} = \Theta_{sr} = p_{\text{out}}$, which means that the probability of existing an edge between two communities is the same, and also given by a constant p_{out}.

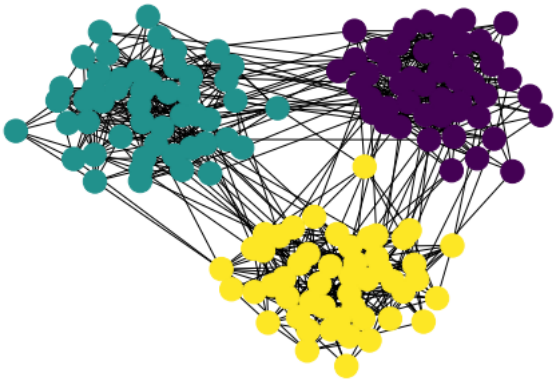


Figure 2.3: Stochastic Block Model with $n = 50$, $k = 3$, $p_{\text{in}} = 0.2$, $p_{\text{out}} = 0.01$

In Figure 2.3 we have an example of the model with parameters $n = 50$, $k = 3$, $p_{\text{in}} = 0.2$ and $p_{\text{out}} = 0.01$. Hence, the matrix Θ is given by:

$$\Theta = \begin{bmatrix} 0.2 & 0.01 & 0.01 \\ 0.01 & 0.2 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix} \tag{2.13}$$

Furthermore, given the model, one may be interested in finding the expected value of the degree of a vertex inside a community:

$$\mathbb{E}[d_i | i \in V_k] = \sum_{j=1}^{k} n_j \Theta_{ij} \tag{2.14}$$

From the example in Figure 2.3, we have that:

$$\mathbb{E}[d_i | i \in V_k] = 50 \times 0.2 + 50 \times 0.01 + 50 \times 0.01 = 11 \tag{2.15}$$

Now let's see how can we use inference. In order to find the communities, one may solve by taking MLE and finding the parameters that are most likely associated with a particular instance. For a network with n nodes and k communities, the parameter to be found are $\theta \in R^{kxk}$, a matrix that describes the probability of connections within and between communities, and a vector of assignments, denoted by z, that associates each node to a specific cluster. Hence, considering the adjacency matrix A, the framework models the probability of existing and edge as:

$$P(A_{ij} = 1 | z) \sim Bernoulli(\theta_{z_i z_j}) \tag{2.16}$$

Since the problem is NP-Complete, no exact algorithm is known to find the optimal structure in polynomial time. To tract such optimization task, some procedures relies on EM, belief propagation, and MCMC accept-reject sampling [30] [31] [32].

Let's analyse [32] proposal carefully, since it's later used by other approaches. Let Z be an assignment matrix, such that $Z_{ik} = 1 \iff i \in k$ denoting that node i belongs to community k, and 0 otherwise. Under the stochastic block model, the conditional log likelihood is given by:

$$\log \mathcal{L}_A = \log(P(A|z, \Theta)) = \frac{1}{2} \sum_{i \neq j} \sum_{k,l} Z_{i,k} Z_{j,l} [a_{ij} \log \theta_{kl} + (1 - a_{ij}) \log(\theta_{kl})] \quad (2.17)$$

They adopt a variational approximation - since $P(Z|A)$ is not tractable - that aims optimize a lower bound of $\log \mathcal{L}_A$, denoted by:

$$\mathcal{J}(R_A) = \log \mathcal{L}_A - KL[R_A(\cdot), P(\cdot|A)],$$

where $P(Z|A)$ denotes the true conditional distribution and R_A is the approximation with respect to the indicator variables Z. The approximation R_A considers the variables $Z_1, \ldots, Z_n$ i.i.d. and has the following form:

$$R_A(Z) = \prod_i h(Z_i; \tau_i),$$

where $\tau_i = (\tau_{i1}, ..., \tau_{iK})$ and $h(\cdot; \tau)$ the multinomial distribution with parameter τ. Let's derive the **E-step**.

Given θ and π the prior such that $\mathbb{P}(Z_{ik} = 1) = \pi_k$, we want to find the optimal variational parameters $\tau_i = \arg\max_{\tau_i} \mathcal{J}(R_A)$. We begin rewriting $\mathcal{J}(R_A)$ as:

$$\mathcal{J}(R_A) = \sum_Z R_A(Z) * \log P(Z, X) - \sum_Z R_A(Z) * \log R_A(Z)$$

$$= \sum_i \sum_q \tau_{iq} \log \pi_q + \frac{1}{2} \sum_{i \neq j} \sum_{q,l} \tau_{iq} * \tau_{jl} * \log(\theta_{ql}^{a_{ij}} * (1 - \theta_{ql})^{1 - a_{ij}}) - \sum_i \sum_q \tau_{iq} \log \tau_{iq}$$

In order to maximize the equation above we can take the derivative with respect to τ_{iq}. It's worth remembering that we must consider the constraint $\sum_q \tau_{iq} = 1, \forall i$. Hence the Lagrangian has the following form:

$$\sum_i \sum_q \tau_{iq} \log \pi_q + \frac{1}{2} \sum_{i \neq j} \sum_{q,l} \tau_{iq} * \tau_{jl} * \log(\theta_{ql}^{a_{ij}} * (1 - \theta_{ql})^{1 - a_{ij}}) - \sum_i \sum_q \tau_{iq} \log \tau_{iq}$$
$$+ \sum_i [\lambda_i \sum_q (\tau_{iq} - 1)],$$

and by taking the derivative wrt τ_{iq} and making equal to zero we have:

$$\log \pi_q + \sum_{i \neq j} \sum_l \tau_{jl} \log(\theta_{ql}^{a_{ij}} * (1 - \theta_{ql})^{1 - a_{ij}}) - \log \tau_{iq} + 1 + \lambda_i = 0 \rightarrow$$

$$\tau_{iq} \propto \pi_q \prod_{j \neq i} \prod_l [\theta_{ql}^{a_{ij}} * (1 - \theta_{ql})^{1 - a_{ij}}]^{\tau_{jl}}$$

Now let's derive the **M-step**.

Given the variational parameters τ_{iq}, we want to find θ and π that maximizes the Lagrangian of $\mathcal{J}(R_A) + \lambda(\sum_q \pi_q = 1)$. By taking the derivatives, one will find that:

$$\pi_q = \frac{1}{N} \sum_i \tau_{iq}$$

$$\theta_{ql} = \frac{\sum_{i \neq j} \tau_{iq} \tau_{jl} a_{ij}}{\sum_{i \neq j} \tau_{iq} \tau_{jl}}$$

This finishes the MLE proposed and proofs needed.

2.3 Data Clustering

When we are presented with data lying on $\mathbf{R}^n$ other than a graph, too many procedures exist to find structured groups. This section aims to briefly present two common algorithms for clustering, KMeans and Gaussian Mixture Model, which will later be discussed under the geometric umbrella.

2.3.1 KMeans

Perhaps the oldest and most known algorithm for such task is the KMeans. It works as follows:

- First, the number of clusters K is defined.
- Assign the initial cluster centroids.
- Compute, for each data point, the centroid with smallest distance. Assign the data point to the cluster corresponding to closest centroid.
- Update each centroid by taking the mean of data points belonging to the respective cluster
- Repeat last two steps until the number of iterations is satisfied or convergence.

As it can be seen, KMeans is very simple and scalable. For initialisation of cluster centroids, they can be taken randomly or accordingly to kmeans plus plus (k++ for short). The later is often preferable in order to produce well separated clusters, and is the default procedure of many packages, such as scikit learn. Such procedure in described in Algorithm 2.

Algorithm 2 Kmeans++

Require: : data points X
Randomly select the first centroid
do
For each data point compute its distance from the nearest, previously chosen centroid
Select the next centroid from the data points such that the probability of choosing a point as centroid is directly proportional to its distance from the nearest, previously chosen centroid
while k centroids have not been sampled

2.3.2 Gaussian Mixture Model

Now we discuss Gaussian Mixture Models (GMM). The intuition behind latent models is that the data has, beyond the multivariate random variables $\mathbf{X}$, hidden ones, which we call $\mathbf{Z}$. In the clusterization problem not only these hidden variables can be interpreted as the clusters that each vertex belongs to, but also its state can be seen as the cause of the observable ones. Hence, let K be a model parameter that specifies the amount of clusters, and let z_i be a discrete hidden variable so $z_i \in \{1, 2, ..., K\}$. We are interested in compute $p(z_i = k|x_i, \theta)$, which is the posterior of sample i belonging to cluster k given the parameter θ of the model and the observable variables x_i. This value can be computed using Bayes rule:

$$r_{ik} = p(z_i = k|x_i, \theta) = \frac{p(z_i = k|\theta)p(x_i|z_i = k, \theta)}{\sum_{k'=1}^{K} p(z_i = k'|\theta)p(x_i|z_i = k', \theta)} \tag{2.18}$$

In equation 2.18 r_{ik} is known as the **responsibility** that cluster k has over sample i, such that $\sum_k r_{ik} = 1, \forall i \in 1, 2, ..., n$. In mixture models we have k base distributions. The probability of observing x_i given it belongs to community k is $p(x_i|z_i = k, \theta_k) = p_k(x_i|\theta_k)$. Hence, the probability of observing a sample x_i is obtained doing a weighted sum of each one of the base distributions:

$$p(x_i|\theta) = \sum_{k=1}^{K} \pi_k p_k(x_i|\theta_k), \tag{2.19}$$

where π_k is the mixture coefficient such that $0 \leq \pi_k \leq 1$ and $\sum_k \pi_k = 1$. In the specific case of GMM we have each base distribution as a Multivariate Gaussian

with mean vector μ_k and covariance matrix Σ_k. Hence the model has the form of $\left[p(x_i|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \sum_k)\right]$. Furthermore, the uncertainty of the partition for a sample i can be given by $1 - \max_k r_{ik}$. If this uncertainty is low enough, the soft clustering problem can be reduced to the hard clustering by doing:

$$z_i^* = \arg\max_k r_{i\_k} = \arg\max_k \log p(x_i|z_i = k, \theta) + \log p(z_i = k|\theta) \tag{2.20}$$

So far we supposed that the model parameters were known, however their estimation is non trivial. Take the log likelihood in the case where all variables are observed:

$$l_c(\theta) \triangleq \sum_{i=1}^{N} \log p(x_i|\theta) \tag{2.21}$$

In this case the parameters θ of the model can be easily optimized via maximum likelihood estimator. But for the model of latent variables we have that:

$$l_c(\theta) = \sum_{i=1}^{N} \log \left[\sum_{z_i} p(x_i, z_i|\theta) \right] = \sum_{i=1}^{N} \log \left[\sum_{z_i} q(z_i) \frac{p(x_i, z_i|\theta)}{q(z_i)} \right], \tag{2.22}$$

where $q(z_i)$ is an arbitrary distribution over the hidden variables. Since one cannot pass the log inside the sum, it is a function hard to optimize. A common approach adopted is the Expectation Maximization algorithm. It consists of two steps, as the name suggests, Expectation and Maximization, which are repeated until convergence:

- Expectation - At this step we seek to maximize Equation (2.22) with respect to the posterior $q(z_i)$. Remembering from the Jensen's inequality, one can find a lower bound that's easier to optimize then Equation (2.22). Since $\log(x)$ is a concave function, one can apply the inequality, that gives:

$$\begin{aligned} l_c(\theta) = \sum_{i=1}^{N} \log \left[\sum_{z_i} q(z_i) \frac{p(x_i, z_i|\theta)}{q(z_i)} \right] &\geq \sum_{i=1}^{N} \sum_{z_i} q(z_i) \log \left[\frac{p(x_i, z_i|\theta)}{q(z_i)} \right] \\ &= \sum_{i=1}^{N} \sum_{z_i} q(z_i) \log \left[p(x_i, z_i|\theta) \right] - q(z_i) \log(q(z_i)) \\ &= \sum_{i=1}^{N} \sum_{z_i} q(z_i) \log \left[p(x_i, z_i|\theta) \right] + \mathbb{H}(q(z_i)), \end{aligned} \tag{2.23}$$

  where $\mathbb{H}(q(z_i)) = -\sum_k p(z_i = k) \log p(z_i = k)$ denotes the entropy. Observing the equation above, one may ask which distribution q_i is the best. Intuitively, the best q_i is the one that results in the tightest lower bound. In order to

obtain the best q_i, we can rewrite Equation 2.23 as follows:

$$\begin{aligned} L(\theta, q_i) &= \sum_{i=1}^{N} \sum_{z_i} q(z_i) \log \left[\frac{p(x_i, z_i|\theta)}{q(z_i)} \right] \\ &= \sum_{i=1}^{N} \sum_{z_i} q(z_i) \log \left[\frac{p(z_i|x_i, \theta) p(x_i|\theta)}{q(z_i)} \right] \\ = \sum_{i=1}^{N} \sum_{z_i} q(z_i) \log \left[\frac{p(z_i|x_i, \theta)}{q(z_i)} \right] &+ \sum_{i=1}^{N} \sum_{z_i} q(z_i) \log \left[p(x_i|\theta) \right] \\ &= \sum_{i=1}^{N} -\mathbb{KL}(q(z_i)||p(z_i|x_i, \theta)) + \log \left[p(x_i|\theta) \right] \end{aligned} \tag{2.24}$$

Since we want to maximize the lower bound, we can set the distribution $q_i(z_i) = p(z_i|x_i, \theta)$, which makes the Kullback Leibler divergence equals zero. But θ is unkown, so instead we have $q_i^t(z_i) = p(z_i|x_i, \theta^t)$, where θ^t denotes the estimate of parameters at iteration t. This is the output of E step: $q_i^t(z_i) = p(Z|X, \theta^t) = r_{ik}$. Now, back to Equation 2.23 we have that the second term is constant wrt θ. So, at the maximization step the parameters will be updated as follows:

$$\theta^{t+1} = \arg\max_{\theta} Q(\theta, \theta^t) = \arg\max_{\theta} \mathbb{E}_{q_i^t}[\log p(x_i, z_i|\theta)] \tag{2.25}$$

- Maximization - At this step we seek to maximize the expected log-likelihood $\mathrm{Q}(\theta, \theta^{t-1})$ wrt π_k and θ_k:

$$\begin{aligned} Q(\theta, \theta^{t-1}) &\triangleq \mathbb{E}\left[\sum_i \log p(x_i, z_i|\theta) \right] \\ &= \sum_i \mathbb{E}\left[\log \left[\prod_k \left(\pi_k p(x_i|\theta_k) \right)^{\mathbb{I}(z_i=k)} \right] \right] \\ &= \sum_i \sum_k \mathbb{E}[\mathbb{I}(z_i = k)] \log[\pi_k p(x_i|\theta_k)] \\ &= \sum_i \sum_k p(Z|X, \theta^{t-1}) \log[\pi_k p(x_i|\theta_k)] \\ &= \sum_i \sum_k r_{ik} \log \pi_k + \sum_i \sum_k r_{ik} \log p(x_i|\theta_k) \end{aligned} \tag{2.26}$$

With respect to π_k we have that the maximum value of Q must respect the constraint that $F(\pi_k) = \sum_k \pi_k = 1$. Hence, by taking the Lagrangian, we have that the function we seek to maximize is given by $\mathcal{L} = \sum_i \sum_k r_{ik} \log \pi_k - \lambda \left(\sum_k \pi_k - 1 \right)$, and to find the solution it suffices to solve the following equation

$\nabla\mathcal{L} = 0$:

$$\frac{\partial\mathcal{L}}{\partial\pi_k} = \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik}\frac{1}{\pi_k} - \sum_{k}^{K}\lambda\pi_k = 0 \tag{2.27}$$

Since for a specific k all the other terms are 0, then we have:

$$\frac{\partial\mathcal{L}}{\partial\pi_k} = \sum_{i=1}^{N} r_{ik}\frac{1}{\pi_k} - \lambda = 0 \tag{2.28}$$

By multiplying the above result by π_k:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\pi_k} = \sum_{i=1}^{N} r_{ik} - \lambda\pi_k = 0 \\ \sum_{i=1}^{N} r_{ik} = \lambda\pi_k\end{aligned} \tag{2.29}$$

Finally, summing over k both sides of the equation:

$$\sum_{k=1}^{K}\sum_{i=1}^{N} r_{ik} = \sum_{k=1}^{K}\lambda\pi_k \tag{2.30}$$

Remembering the constraint that $F(\pi_k) = \sum_k \pi_k = 1$ and for a given data i the sum of the responsibilities $\sum_k r_{i\_k} = 1$:

$$N = \lambda \tag{2.31}$$

Plugging the result $\lambda = N$ at the equations we have that:

$$\begin{aligned}\sum_{k=1}^{K}\left[\sum_{i=1}^{N} r_{ik}\right]\frac{1}{\pi_k} - \sum_{k}^{K} N = 0 \\ \left[\sum_{i=1}^{N} r_{ik}\right]\frac{1}{\pi_k} - N = 0 \\ \sum_{i=1}^{N} r_{ik} = N\pi_k \\ \pi_k = \frac{\sum_i r_{ik}}{N}\end{aligned} \tag{2.32}$$

We solved for π_k, now with respect to θ_k, or μ_k and Σ_k since it is a multivariate gaussian distribution $\mathcal{N}(x|\mu,\Sigma) \triangleq \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}}\exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]$,

and replacing Σ^{-1} for Λ, we have that the log likelihood is given by:

$$l(\mu, \Sigma) = \log p(D|\mu, \Sigma) = \frac{N}{2} \log |\Lambda| - \frac{1}{2} \sum_{i=1}^{N} (x_i - \mu)^T \Lambda (x_i - \mu) \tag{2.33}$$

By doing the substitution $y_i = x_i - \mu$, applying the chain rule and remembering that $\frac{\partial}{\partial a}(a^T A a) = (A + A^T)a$:

$$\frac{\partial}{\partial \mu}(x - \mu)^T \Sigma^{-1} (x - \mu) = \frac{\partial}{\partial y_i} y_i^T \Sigma^{-1} y_i \frac{\partial}{\partial \mu} = -1 \left(\Sigma^{-1} + \Sigma^{-T}\right) y_i \tag{2.34}$$

Plugging this last result at the previous equation:

$$\begin{aligned} \frac{\partial}{\partial \mu} l(\mu, \Sigma) &= -\frac{1}{2} \sum_{i=1}^{N} -2\Sigma^{-1}(x_i - \mu) \\ &= \Sigma^{-1} \sum_{i=1}^{N} (x_i - \mu) = 0 \\ \hat{\mu} &= \frac{1}{N} \sum_{i=1}^{N} x_i = \hat{x} \end{aligned} \tag{2.35}$$

Now taking the MLE with respect to Λ:

$$\begin{aligned} l(\Lambda) &= \frac{N}{2} \log |\Lambda| - \frac{1}{2} \sum_i tr\left[(x_i - \mu)(x_i - \mu)^T \Lambda\right] \\ &= \frac{N}{2} \log |\Lambda| - \frac{1}{2} tr\left[S_\mu \Lambda\right] \end{aligned} \tag{2.36}$$

Where $S_\mu \triangleq \sum_{1=1}^{N} (x_i - \mu)(x_i - \mu)^T$. Now taking the partial derivatives and remembering that $\frac{\partial}{\partial A} log|A| = A^{-T}$,$\frac{\partial}{\partial A} tr(BA) = B^T$:

$$\frac{\partial}{\partial \Lambda} l(\Lambda) = \frac{N}{2} \Lambda^{-T} - \frac{1}{2} S_\mu^T = 0 \tag{2.37}$$

But since $\Lambda^{-T} = \Lambda^{-1} = \Sigma$:

$$\hat{\Sigma} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})(x_i - \hat{\mu})^T \tag{2.38}$$

Which gives the empirical covariance.

2.4 Data and network clustering

So far we've considered only the network structure in order to find meaningful communities. Real world data, in the other hand, commonly consists of mixed sources of information, comprising both features and network structure. In this section, we seek to bring two algorithms that leverages both sources: Attributed Stochastic Block Model [4] and Contextual Stochastic Block Model [5].

2.4.1 Attributed Stochastic Block Model

The proposal of this work is very straightforward and simple. It seeks to find the same parameters of the Stochastic Block Model, θ and vector assignments z (or a binary indicator matrix Z such that $Z_{ic} = 1$ if node i belongs to community c), but also the parameters of a Gaussian Mixture Model, μ and Σ, associated with the attributes of each class. Their main assumption is that adjacency matrix is conditionally independent from the attributes, giving the following probabilistic graphical model in figure 2.4.

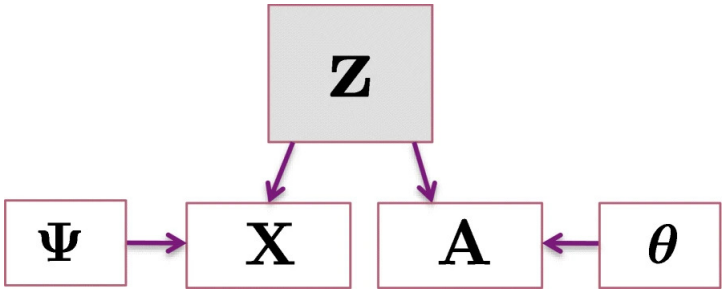


Figure 2.4: Modeling community membership in terms of attributes and connectivity. Node-to-community assignments specified by Z are determined in terms of adjacency matrix information, A and attribute matrix information, X. A and X are assumed to be generated from a stochastic block model and a mixture of multivariate Gaussian distributions, parameterized by θ and Ψ, respectively. Taken from [4]

In order to find such parameters they adopt the MLE procedure, in an Expectation Maximization way, alternating between estimating $p(Z_{ic} = 1|X, A)$ and the parameters Ψ and Θ. Since the attributes matrix X are conditionally independent from the network observation A, the total log-likelihood can be computed as a sum of two parts, one that depends solely on the network and another on the attributes: $P(A, X, Z) = \mathcal{L} = \mathcal{L}_A + \mathcal{L}_X$.

The attribute likelihood of a model consisting of K communities is given by: $\mathcal{L}_X = P(X|\Psi) = \sum_{i=1}^{N} \log\{\sum_{j=1}^{K} \pi_c \mathcal{N}(x_i|\mu_c, \Sigma_c)\}$, where μ_c and Σ_c are the means and covariance matrix respectively. Hence $\Psi = (\mu_1, \mu_2, \ldots, \mu_k, \Sigma_1, \Sigma_2, \ldots, \Sigma_k)$. By taking a_i as the connectivity profile of node i, the likelihood of the network is given by: $\mathcal{L}_A = \frac{1}{2} \sum_{i \neq j} \sum_{k,l} Z_{i,k} Z_{j,l} [a_{ij} \log \theta_{kl} + (1 - a_{ij}) \log(\theta_{kl})]$, the same as in [32].

The inference procedure, once defined the likelihood functions, works as follows. The **E-step** consists of computing $r_{ic} = P(Z_{ic} = 1|X_i, a_i)$, which taking Bayes formula gives $p(z|X, A) = p(X, A|z)p(z)/p(X, A)$. Now, assuming conditional independence, it is equivalent to:

$$\frac{p(X, A|z)p(z)}{p(X, A)} = \frac{p(X|z)p(A|z)p(z)}{p(X)p(A)}$$

$$r_{ic} = \frac{P(X_i|Z_{ic} = 1)P(a_i|Z_{ic} = 1)\pi_c}{\sum_k P(X_i|Z_{ik} = 1)P(a_i|Z_{ik} = 1)\pi_k} \tag{2.39}$$

At the **M-Step** they seek to estimate $\theta_c, \mu_c, \Sigma_c$, for every community, which are given by:

$$\mu_c = \frac{\sum_{i=1}^N r_{ic} X_i}{\sum_{i=1}^N r_{ic}} \tag{2.40}$$

$$\Sigma_c = \frac{\sum_{i=1}^N r_{ic}(X_i - \mu_c)(X_i - \mu_c)^T}{\sum_{i=1}^N r_{ic}} \tag{2.41}$$

$$\theta_{ql} = \frac{\sum_{i \neq j} r_{iq} r_{jl} a_{ij}}{\sum_{i \neq j} r_{iq} r_{jl}} \tag{2.42}$$

The first two equations where demonstrated in previous section. The last one is essentially the result shown in [32].

2.4.2 Contextual Stochastic Block Model

To begin with, it's another algorithm that also assumes that the network comes from a Stochastic Block Model and the covariates come from Gaussian distributions. However, they design an iterative algorithm rather than relying on MLE.

The full Algorithm 3 is stated bellow. It's very simple. First they consider

Algorithm 3 CSBM

Require: : $A \in R^{NxN}$, $X \in R^{nxd}$, $K \in N*$, $\sigma > 0$, $Z(0) \in \{0, 1\}^{nxK}$ a membership matrix and $T \geq 1$.

do

Estimate model parameters

$n_k = |C_k^t|$

$W^t = Z^t(D^t)^{-1}$

$\Pi^t = (W^t)^T A W^t$

$\mu_k = (W_{k:}^t)^T X$

Refine the partition by solving for each node:

$z_i^{t+1} = \arg\min_k ||A_{i:}W^t - \Pi_{k:}^t \sqrt{\Sigma_k^t}||^2 + \frac{||X_i - \mu_k^t||^2}{\sigma^2}$

Build Z^{t+1} matrix from the assignments z

while $0 \leq t \leq T - 1$

an initialization of membership matrix Z. Next step is to estimate the parameters of the model: the number of nodes in cluster k (n_k), the partition W, the SBM probability matrix Π and cluster covariate means μ_k. Once defined the parameters, a refinement of the partition is considered is done. It consists of assigning each node to the closest community, according to the distance function given by $||A_{i:}W^t - \Pi^t_{k:}\sqrt{\Sigma^t_k}||^2 + \frac{||X_i - \mu^t_k||^2}{\sigma^2}$. The first part measures the distance between current estimated connectivity profile and the estimated probabilities of SBM. The second term depends only on the covariates, for which $\sigma_k = \sqrt{\sum_i ||X_i - \mu_k||^2/N}$.

Their algorithm mainly consists of two versions, one that considers the SBM variance as well and another that doesn't. The main difference is if the term Π is rather multiplied by a a factor Σ or doesn't. In the second approach, the factor Σ is given by $\text{diag}(\frac{n^t_k}{\Pi_{kk}})$. They also propose other factors of Σ.

Another important key of their algorithms is that they are all initialisation sensitive. Hence, they propose the following initialization procedure. They provide a

Algorithm 4 EM on graph embedding and covariates (EM-Emb)

Require: : adjacency matrix A, covariates X, number of communities K
Compute $U_K \in R^{nxK}$ the matrix formed by the eigenvectors associated with the top-K eigenvalues (in absolute order) of A
Merge the columns of U_K with the columns of X to obtain a matrix X'
Cluster the rows of X' by using an EM algorithm for GMM

comparison considering different initialisations of the algorithm in figure 2.5 bellow.

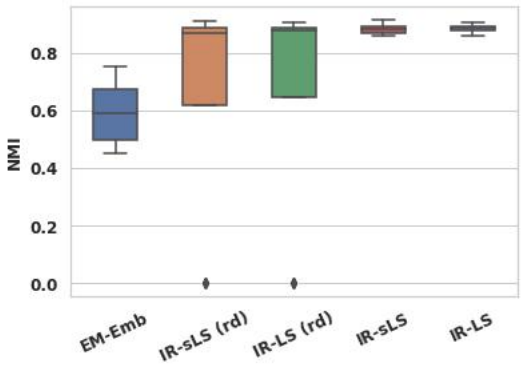


Figure 2.5: Random initialisation (rd) vs EM-emb. Extracted from [5].

Chapter 3

Bregman Divergences

What is a Bregman divergence? What kind of space does it spans? What is the relationship between Bregman divergences and exponential families? How to derive clustering algorithms with such assumptions? It is to answer such questions that in this chapter we bring the mathematical formalism behind Bregman divergences and their geometrical properties under the umbrella of differential geometry. Much of this theoretical analysis is due to the works of Banerjee [8] and Shun-ichi Amari [7]. In the fisrt part of this chapter we bring general concepts of information geometry in an intuitive way for non mathematically inclined readers and, later, we show how they are useful to pattern recognition.

3.1 Concepts of Information Geometry

3.1.1 Manifold

To begin with, we intent to present the manifold definition in an intuitive manner. One can think about manifolds as many planes glued together with smooth intersection. Such union may induce a curved topology. For example, a sphere is a manifold such that in each point we have a tangent plane that is equivalent to a two dimensional Euclidean space. Formally, "an n-dimensional manifold M is a set of points such that each point has n-dimensional extensions in its neighborhood"[1].

Since each point of the manifold is locally equivalent to an n-dimensional Euclidean space, one can define a local coordinate system (or coordinate chart) associated with each point. We will follow the same notation as used by Amari, such that the n-coordinate system is the noted by $\xi = (\xi_1, \ldots, \xi_n)$. The coordinate system depends on the point p. We will denote the coordinate system associated with point p as ξ_p, and the tangent plane in p as $T_p\mathcal{M}$. Hence, in the general case a unique global coordinate system doesn't exist. It's also worth mentioning that the coordi-

[1] Taken from [7], page 3

nate system is not locally unique either. Let ζ be another coordinate system such that exists an one-to-one correspondence to ξ_p. If that holds, then the point p in a manifold $\mathcal{M}$ can be represented by both coordinate systems:

$$\xi_p = f(\zeta_1, \ldots, \zeta_n)$$
$$\zeta_p = f^{-1}(\xi_1, \ldots, \xi_n)$$

3.1.2 Tangent Plane

A differentiable function $\alpha : R \to \mathcal{M}$ is a differentiable curve in $\mathcal{M}$. Take a small piece of this curve, $\alpha(-\epsilon, \epsilon) \to S$, such that $\alpha(0) = p \in \mathcal{M}$. Let f be a differentiable function of the manifold to $\mathcal{R}$. The tangent vector of α in t=0 is the operator $\alpha'(0)\colon f(\mathcal{M}) \to \mathcal{R}$ given by:

$$\alpha'(0)f = \frac{d(f \circ \alpha)}{dt}\Big|_{t=0}$$

Thereby, a tangent vector in p is the vector with t=0 of a curve $\alpha(t)\Big|_{-\epsilon}^{\epsilon} \to S$ such that $\alpha(0) = p \in S$. The set of all tangent vectors in p is denoted by $T_p\mathcal{M}$.

Let x be a parametrization that takes an open set U in R^n and takes to the manifold $x\colon U \to \mathcal{M}$. If we take the tangent vector of the manifold coordinate curves we will find the basis vectors of the tangent plane $B_p = \{(\frac{\partial}{\partial x_1})_p, \ldots, (\frac{\partial}{\partial x_n})_p\}$. For short, we will use $e_i \approx \frac{\partial}{\partial x_i}$ to denote the coordinate vector associated with the ith curve.

Suppose we want to derive a function $f(\mathcal{M}) \to \mathcal{R}$ along a curve α. We can represent the point $p = \alpha(0)$ as $p = x(q)$, and the function we want to derive is $(f \circ x)(q) = (f \circ x)(x_1, \ldots, x_n)$, with $q = (x_1, \ldots, x_n) \in U$ and $(x^{-1} \circ \alpha)(t) = (x_1(t), \ldots, x_n(t))$. Hence in a more general sense we have:

$$\begin{aligned}\alpha'(0)f = \frac{d(f \circ \alpha)}{dt}\Big|_{t=0} &= \frac{d(f \circ x)(x^{-1} \circ \alpha)(t)}{dt}\Big|_{t=0} \\ &= \frac{df \circ x(x_1(t), \ldots, x_n(t))}{dt} \\ = \sum_i x_i'(0)\frac{\partial}{\partial x_i}(f \circ x)(q) &= \sum_i x_i'(0)\left(\frac{\partial}{\partial x_i}\right)_p f\end{aligned}$$

From above equations, we have that any tangent curve can be expressed in terms of the parametrization x:

$$\alpha'(0) = \sum_i x_i'(0)\left(\frac{\partial}{\partial x_i}\right)_p$$

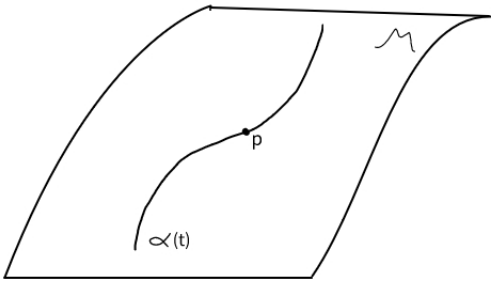


Figure 3.1: A curve on manifold. The set of all tangent vectors in p form the tangent plane $T_p\mathcal{M}$

3.1.3 Charts

A chart ϕ is a mapping from a subset $\mathcal{U}$ of $\mathcal{M}$ to $R^n(\phi : \mathcal{U} \to R^n)$.

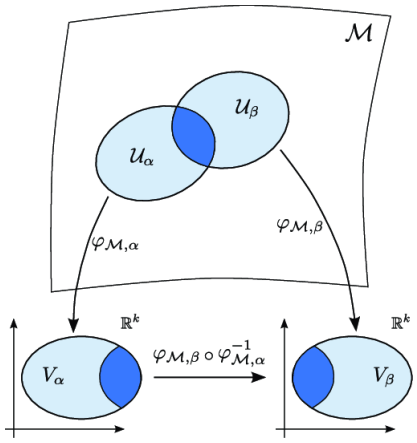


Figure 3.2: Charts on a manifold. Extracted from [6]

A manifold $\mathcal{M}$ is said to be smooth when the change of charts, and sub sequentially the change of coordinates, is smooth, i.e., many times differentiable. Consider the figure 3.2. We have two charts $\phi_{\mathcal{M},\alpha}$ and $\phi_{\mathcal{M},\beta}$, which taking a subset $\mathcal{U} \in \mathcal{M}$ and taking them to an Euclidean coordinate system. Note that they have an intersection. If we wish to find a mapping from one coordinate system to another, in the intersection, we must take the following actions: $V_\alpha \to \mathcal{M} \to V_\beta$, which is mathematically defined as $\phi_{\mathcal{M},\beta} \circ \phi^{-1}_{\mathcal{M},\alpha}$. When such mapping is differentiable, the manifold is differentiable.

3.1.4 Divergence

First of all, a divergence d is a function that takes two points P and Q and measures their dissimilarity. In general, it is not a symmetric function, $d[P, Q] \neq d[Q, P]$, so it can't be interpreted as a distance. P and Q are points of a manifold, so one may use the notation $d[P, Q] = d[\xi_P, \xi_Q]$ to emphasize that they may have different coordinate systems.

Any divergence must follow these properties:

- Non-negativity: $d[P, Q] \geq 0$
- $d[P, Q] = 0 \iff P = Q$

A manifold $\mathcal{M}$ is said to be Riemannian when a positive-definite matrix $\mathrm{G}(\xi)$ is defined on $\mathcal{M}$ and the square of the local distance between two nearby points ξ and $\xi + d\xi$ is given by $ds^2 = 2d[\xi, \xi + d\xi] = \sum_{ij} g_{ij} d\xi_i d\xi_j$. As we will see later, one can define a Riemmanian metric from the divergence, which gives us a Riemannian manifold $\mathcal{M}$.

Bregman Divergences

The Bregman divergences form a special class. They are derived from a convex function ϕ, so we will use the notation d_ϕ to emphasize that. Also, they satisfy the following properties:

- Convexity: d_ϕ is convex with respect to the first argument
- Linearity: $d_{\phi_1+\phi_2}(x, y) = d_{\phi_1}(x, y) + d_{\phi_2}(x, y)$
- Linear separation: An hyperplane contains the set of all points such that $d(x, \mu_1) = d(x, \mu_2)$ for $\mu_1, \mu_2 \in ri(S)$
- Duality: The convex function ϕ and it's conjugate ψ given by Legendre transform satisfy:

$$d_\phi(\mu_1, \mu_2) = \phi(\mu_1) + \psi(\theta_2) - \langle \mu_1, \theta_2 \rangle = d_\psi(\theta_2, \theta_1)$$

- Generalized Pythagorean Theorem. The following inequality holds:

$$d_\phi(x_1, x_2) + d_\phi(x_2, x_3) \leq d_\phi(x_1, x_3)$$

  The equality is attained when the dual geodesic connecting x_1, x_2 is orthogonal to the geodesic connecting x_2, x_3.

Also, Bregman Divergences are derived from a convex function ψ. A strictly convex function satisfies the Jensen's inequality:

$$\lambda\psi(\xi_1) + (1-\lambda)\psi(\xi_2) \geq \psi(\lambda\xi_1 + (1-\lambda)\xi_2)$$
$$\lambda\psi(\xi_1) + (1-\lambda)\psi(\xi_2) = \psi(\lambda\xi_1 + (1-\lambda)\xi_2) \iff \xi_1 = \xi_2,$$

for any ξ_1, ξ_2 and $0 < \lambda < 1$.

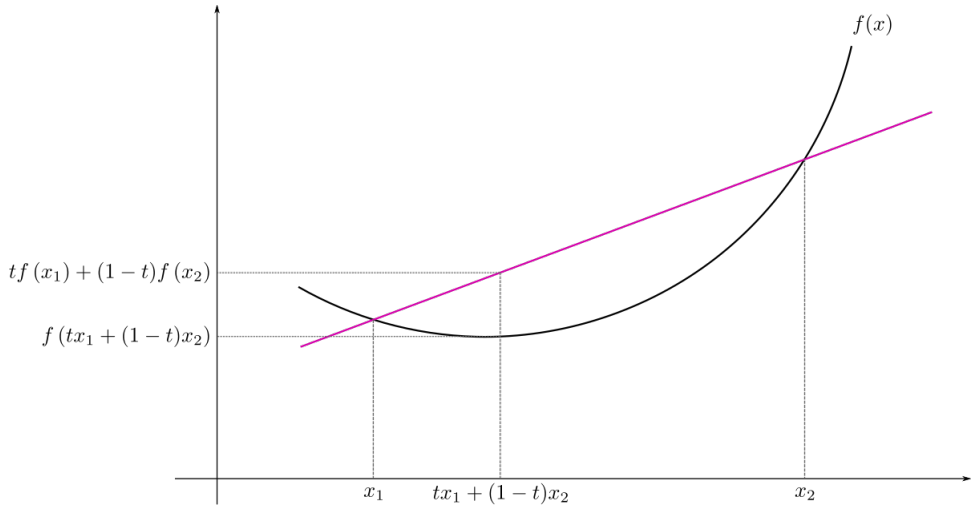


Figure 3.3: Convex function and Jensen inequality

Let's see a geometric interpretation. First, consider a convex function, without loss of generality, as in Figure 3.4. We have the plane equation tangent to ξ_0 given by $z = \psi(\xi_0) + \nabla\psi(\xi_0)(\xi - \xi_0)$. Since ψ is convex, the tangent plane is always bellow the function, i.e., doesn't intersect at any other point ($\psi(\xi) \geq z(\xi)$). We seek now to measure how far the point ξ is from the supporting plane in ξ_0. This gives us the Bregman divergence, which has this following form:

$$D[\xi, \xi_0] = \psi(\xi) - \psi(\xi_0) - \nabla\psi(\xi_0)(\xi - \xi_0)$$

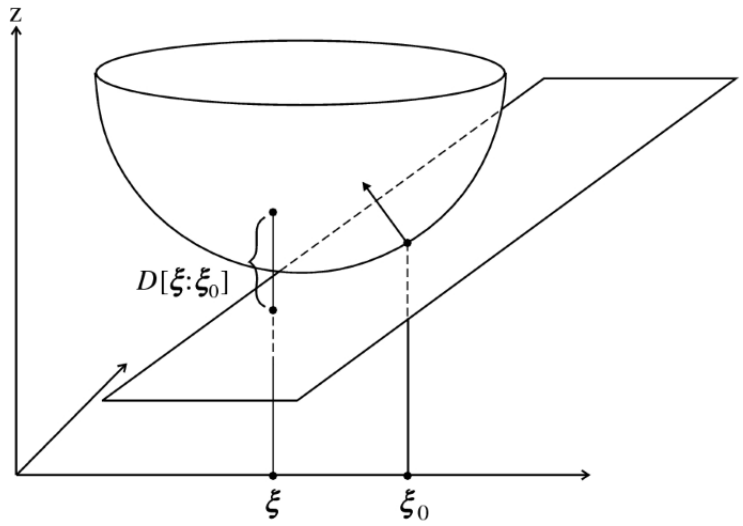


Figure 3.4: Tangent plane at point ξ_0. Taken from [7]

Let's go deeper in the structure derived from a Bregman divergence. We begin defining a transformation that will give us another coordinate system - so far our coordinate system is given by ξ. The **Legendre Transform** will kick in for this

purpose. It is defined as:

$$\xi^* = \nabla\psi(\xi),$$

hence the transformation between ξ^* and ξ is one-to-one and differentiable, because the gradient vector in each point of the surface is different. We want to show that these coordinate systems are coupled to each other, and more strictly that they form a dual basis. Consider for a moment the following function $\phi = \psi^*(\xi^*)$:

$$\psi^* = \sup_{\xi'}\{\xi' \cdot \xi^* - \psi(\xi')\} = \xi \cdot \xi^* - \psi(\xi) \tag{3.1}$$

We want to show that this function is the dual of ψ. To see that, we differentiate such function. Considering also that ξ is some function of ξ^*, this gives:

$$\nabla\psi^*(\xi^*) = \xi + \frac{\partial\xi}{\partial\xi^*}\xi^* - \nabla\psi(\xi)\frac{\partial\xi}{\partial\xi^*}$$

But we also have $\xi^* = \nabla\psi(\xi)$. Hence:

$$\nabla\psi^*(\xi^*) = \xi + \frac{\partial\xi}{\partial\xi^*}\nabla\psi(\xi) - \nabla\psi(\xi)\frac{\partial\xi}{\partial\xi^*}$$
$$\xi = \nabla\psi^*(\xi^*)$$

Now we have the dualistic structure defined. The Legendre Transform can be used always when a change of coordinates is needed. ψ^* is commonly called by the Legendre dual of ψ.

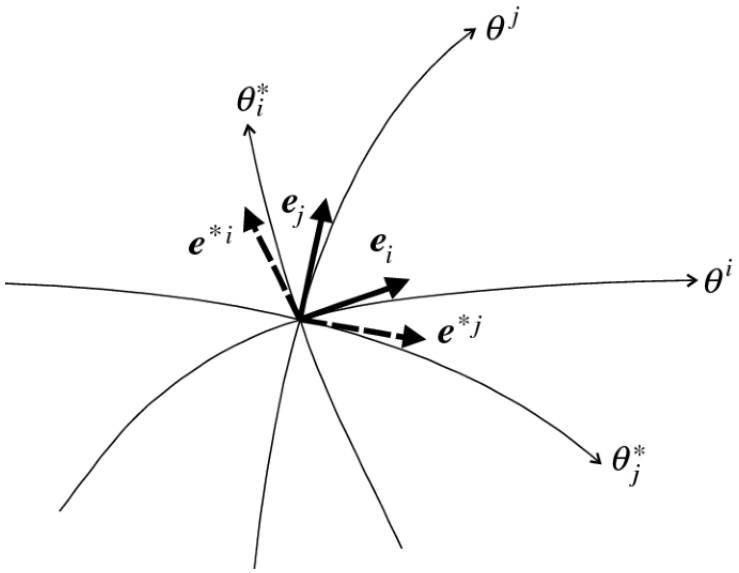


Figure 3.5: Relationship between each coordinate system. The basis vector is the derivative along it's respective curve. Taken from [7].

Finally, we're ready to show how the Riemann metric emerges from the diver-

gence. First, we bring the meaning of such metric. It's, by definition, the product between two coordinate vectors in a tangent plane $T_p\mathcal{M}$:

$$g_{ij}(\xi) = \langle e_i, e_j \rangle$$

In the special case of Euclidean manifold, we have $g_{ij} = \delta_{ij}$. For the Bregman divergences, on the other hand, we can derive

$$g_{ij}(\xi) = \frac{\partial^2 D[\xi, \xi'])}{\partial \xi_i \partial \xi_j} = \frac{\partial^2 \psi(\xi)}{\partial \xi_i \partial \xi_j} = \nabla^2 \psi$$

As we will see later, this is the Fisher Information Matrix, which is central to many important insights in the field of information geometry, such as the so called Cramér-Rao theorem. But what's the relationship between the Hessians $\nabla^2\psi$ and $\nabla^2\phi$? This result is due to Crouzeix [33], who proved that the following equality holds:

$$\nabla^2\psi\nabla^2\phi = g_{ij}(\xi)g^{ij}(\xi^*) = I$$
$$\nabla^2\psi = (\nabla^2\phi)^{-1}$$

In above equation, g^{ij} is the Riemannian metric of the dual space and I the indentity matrix. We denote for now on that the upper index refers to the dual coordinate system. The result shows that the hessian of ψ is the inverse of hessian of ϕ.

3.1.5 Exponential Families

In this subsection we show that there's an one-to-one correspondence between Bregman divergences and regular exponential distributions. To begin with, in the realm of probability manifolds one commonly call the dualistic coordinate systems by the letters $\theta = \xi$ and $\mu = \xi^*$ to denote the natural parameters and expectation parameters. The form of exponential distributions is given by:

$$p_{\psi,\theta} = p(x, \theta) = \exp\{\theta^i h_i(x) + k(x) - \psi(\theta)\},$$

where x is a random variable, $\theta = (\theta^i, \ldots, \theta^n)$ are the natural parameters, h(x) are n linear independent functions of x and $\psi(\theta)$ is just a normalization constant. It's important to note that both h(x) and k(x) do not depend on the parameters θ, and the function ψ does not depend on the variates. We write the expectation parameters as:

$$\mu = \mu(\theta) = \nabla\psi(\theta) = E_{p_{\psi,\theta}}[h(X)] = \int_{R^d} h(x)p_{\psi,\theta}(x)dx$$

Before we start the proof that the equality $\nabla\psi(\theta) = E_{p_{\psi,\theta}}[h(X)]$ holds, we first note that:

$$\begin{aligned}\partial_i \int p(x,\theta)dx &= \int \partial_i p(x,\theta)dx \\ &= \int p\partial_i \ln p dx = E_p[\partial_i \ln p] \\ \text{Since } \int p(x,\theta)dx = 1,\ \partial_i \int p(x,\theta)dx &= 0: \\ E_p[\partial_i \ln p] &= 0\end{aligned}$$

Now we're able to begin the proof:

Proof.

$$\begin{aligned}\ln p &= \theta^i h_i(x) - \psi(\theta) \\ \partial_i \ln p &= h_i(x) - \partial_i\psi(\theta) \\ E_p[\partial_i \ln p] &= E_p[h_i(x)] - \partial_i\psi(\theta) \\ \text{Given that } E_p[\partial_i \ln p] &= 0: \\ 0 &= E_p[h_i(x)] - \partial_i\psi(\theta) \\ \partial_i\psi(\theta) &= E_p[h_i(x)] \\ \nabla\psi(\theta) &= E_p[h(x)]\end{aligned}$$

□

The relationship between these coordinate systems (derived from natural and expected parameters) follows from the Legendre transform, and are commonly denoted as bellow:

$$\begin{aligned}\mu(\theta) &= \nabla\psi(\theta) \\ \theta(\mu) &= \nabla\phi(\mu)\end{aligned}$$

Also, the dual of ψ, ϕ, is written as follows in terms of the coordinate:

$$\phi(\mu) = \langle\theta(\mu), \mu\rangle - \psi(\theta(\mu)), \forall\mu \in int(dom(\phi)). \tag{3.2}$$

Now, how can one establish the relationship between Bregman Divergences and exponential families? Forster and Warmuth [34] prove that:

$$\log(p_{(\psi,\theta)}(x)) = -d_\phi(x, \mu(\theta)) + \log(b_\phi(x)),$$

where b_ϕ is a normalizing function that depends only on x. To show that the

relationship above holds we begin taking the density function of any exponential family, and put other terms that don't depend on distribution parameters inside $p_0(x)$:

$$p_{(\psi,\theta)}(x) = \exp\{\langle x, \theta\rangle - \psi(\theta)\}p_0(x)$$

Now we use the relation $-\psi(\theta(\mu)) = \phi(\mu) - \langle\theta(\mu), \mu\rangle$ from equation 3.2

$$p_{(\psi,\theta)}(x) = \exp\{\langle x, \theta\rangle + \phi(\mu) - \langle\theta(\mu), \mu\rangle\}p_0(x)$$

By rearranging and replacing $\theta = \nabla_\phi(\mu)$

$$= \exp\{\phi(\mu) - \langle x - \mu, \nabla\phi(\mu)\rangle\}p_0(x)$$

$$= \exp\{(-1) * [-\phi(\mu) + \langle x - \mu, \nabla\phi(\mu)\rangle]\}p_0(x)$$

Now we add and subtract $\phi(x)$ in order to write above in terms of a divergence

$$= \exp\{(-1) * [\phi(x) - \phi(\mu) + \langle x - \mu, \nabla\phi(\mu)\rangle] + \phi(x)\}p_0(x)$$

$$= \exp\{-d_\phi(x, \mu)\}b_\phi(x),$$

$$\text{with } b_\phi(x) = \exp\{\phi(x)\}p_0(x)$$

It's important note that, since Legendre Transform is bijective, the pair (ϕ, ψ) is uniquely related to a single exponential distribution, as in Figure 3.6, and ψ denotes the cumulant function. Also, it means that one can write any exponential distribution as a function of either natural parameters or expected ones.

Let's take for example the Gaussian distribution $x \sim \mathcal{N}(\mu, \Sigma)$. We can write as follows:

$$p(x, a, \Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\{-\frac{1}{2}(x - a)\Sigma^{-1}(x - a)\}$$

$$p(x, \theta_1, \theta_2) = \frac{1}{\sqrt{(2\pi)^d}} \exp\{\theta_1 x + \theta_2 x x^T + \frac{1}{4}\theta_1^T\theta_2^{-1}\theta_1 + \frac{1}{2}\log|-2\theta_2|\}$$

The second expression is in the canonical form $p(x, \theta) = \exp\{\theta^i h_i(x) - \psi(\theta)\}b(x)$

with:

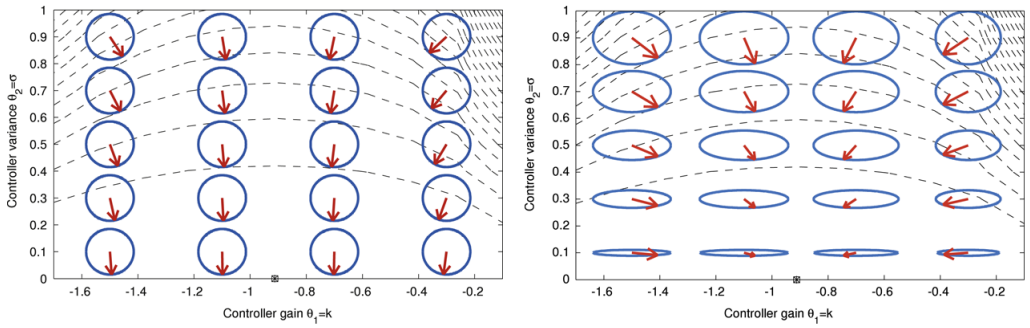
$$\theta = [\theta_1, \theta_2]$$
$$\theta_1 = \Sigma^{-1}a$$
$$\theta_2 = -\frac{1}{2}\Sigma^{-1}$$
$$h(x) = [x, xx^T]^T$$
$$b(x) = \frac{1}{\sqrt{(2\pi)^d}}$$
$$\psi(\theta) = -\frac{1}{4}\theta_1^T\theta_2^{-1}\theta_1 - \frac{1}{2}\log|-2\theta_2|$$
$$a = E[x]$$
$$\Sigma = cov[x]$$

Hence, the vector $[a, \Sigma]$ are the moment (expected) parameters and $[\Sigma^{-1}a, \frac{-1}{2}\Sigma^{-1}]$ the natural (dual) parameters. But one may ask, why do I care about natural parameters? It turns out that it's better to optimize in the natural space rather than in the expected parameter space due to plateaus of the vanilla gradient. Once obtained the parameters in the natural space, one can go back to the moment parameters with the Legendre transform. This idea is used in many results across machine learning algorithms. For example [9] uses natural gradient to obtain optimal policies in a reinforcement learning framework in which the vanilla gradient vanishes in the direction of $\theta = k$ parameter. In [10] the natural gradient is applied to optimize the parameters via MLE in a sensor networks context.

Distribution	$p(\mathbf{x};\boldsymbol{\theta})$	$\boldsymbol{\mu}$	$\phi(\boldsymbol{\mu})$	$d_\phi(\mathbf{x},\boldsymbol{\mu})$
1-D Gaussian	$\frac{1}{\sqrt{(2\pi\sigma^2)}}\exp(-\frac{(x-a)^2}{2\sigma^2})$	a	$\frac{1}{2\sigma^2}\mu^2$	$\frac{1}{2\sigma^2}(x-\mu)^2$
1-D Poisson	$\frac{\lambda^x\exp(-\lambda)}{x!}$	λ	$\mu\log\mu-\mu$	$x\log(\frac{x}{\mu})-(x-\mu)$
1-D Bernoulli	$q^x(1-q)^{1-x}$	q	$\mu\log\mu+(1-\mu)\log(1-\mu)$	$x\log(\frac{x}{\mu})+(1-x)\log(\frac{1-x}{1-\mu})$
1-D Binomial	$\frac{N!}{(x)!(N-x)!}q^x(1-q)^{N-x}$	Nq	$\mu\log(\frac{\mu}{N})+(N-\mu)\log(\frac{N-\mu}{N})$	$x\log(\frac{x}{\mu})+(N-x)\log(\frac{N-x}{N-\mu})$
1-D Exponential	$\lambda\exp(-\lambda x)$	$1/\lambda$	$-\log\mu-1$	$\frac{x}{\mu}-\log\left(\frac{x}{\mu}\right)-1$
d-D Sph. Gaussian	$\frac{1}{\sqrt{(2\pi\sigma^2)^d}}\exp(-\frac{\|\mathbf{x}-\mathbf{a}\|^2}{2\sigma^2})$	$\mathbf{a}$	$\frac{1}{2\sigma^2}\|\boldsymbol{\mu}\|^2$	$\frac{1}{2\sigma^2}\|\mathbf{x}-\boldsymbol{\mu}\|^2$
d-D Multinomial	$\frac{N!}{\prod_{j=1}^d x_j!}\prod_{j=1}^d q_j^{x_j}$	$[Nq_j]_{j=1}^{d-1}$	$\sum_{j=1}^d \mu_j\log(\frac{\mu_j}{N})$	$\sum_{j=1}^d x_j\log(\frac{x_j}{\mu_j})$

Distribution	$\boldsymbol{\theta}$	$\psi(\boldsymbol{\theta})$	dom(ψ)	dom(ϕ)	I_ψ
1-D Gaussian	$\frac{a}{\sigma^2}$	$\frac{\sigma^2}{2}\theta^2$	$\mathbb{R}$	$\mathbb{R}$	$\mathbb{R}$
1-D Poisson	$\log\lambda$	$\exp(\theta)$	$\mathbb{R}$	$\mathbb{R}_+$	$\mathbb{N}$
1-D Bernoulli	$\log(\frac{q}{1-q})$	$\log(1+\exp(\theta))$	$\mathbb{R}$	$[0,1]$	$\{0,1\}$
1-D Binomial	$\log(\frac{q}{1-q})$	$N\log(1+\exp(\theta))$	$\mathbb{R}$	$[0,N]$	$\{0,1,\ldots,N\}$
1-D Exponential	$-\lambda$	$-\log(-\theta)$	$\mathbb{R}_{--}$	$\mathbb{R}_{++}$	$\mathbb{R}_{++}$
d-D Sph. Gaussian	$\frac{\mathbf{a}}{\sigma^2}$	$\frac{\sigma^2}{2}\|\boldsymbol{\theta}\|^2$	$\mathbb{R}^d$	$\mathbb{R}^d$	$\mathbb{R}^d$
d-D Multinomial	$[\log(\frac{q_j}{q_d})]_{j=1}^{d-1}$	$N\log(1+\sum_{j=1}^{d-1}\exp(\theta_j))$	$\mathbb{R}^{d-1}$	$\{\boldsymbol{\mu}\in\mathbb{R}_+^{d-1}, \|\boldsymbol{\mu}\|\leq N\}$	$\{\mathbf{x}\in\mathbb{Z}_+^{d-1}, \|\mathbf{x}\|\leq N\}$

Figure 3.6: Exponential distributions and their expected parameters μ, natural parameters θ and dual pair of functions (ϕ, ψ) associated. Taken from Banerjee[8].



(a) Vanilla policy gradient. (b) Natural policy gradient.

Figure 3.7: Vanilla gradient vs. natural gradient in [9]

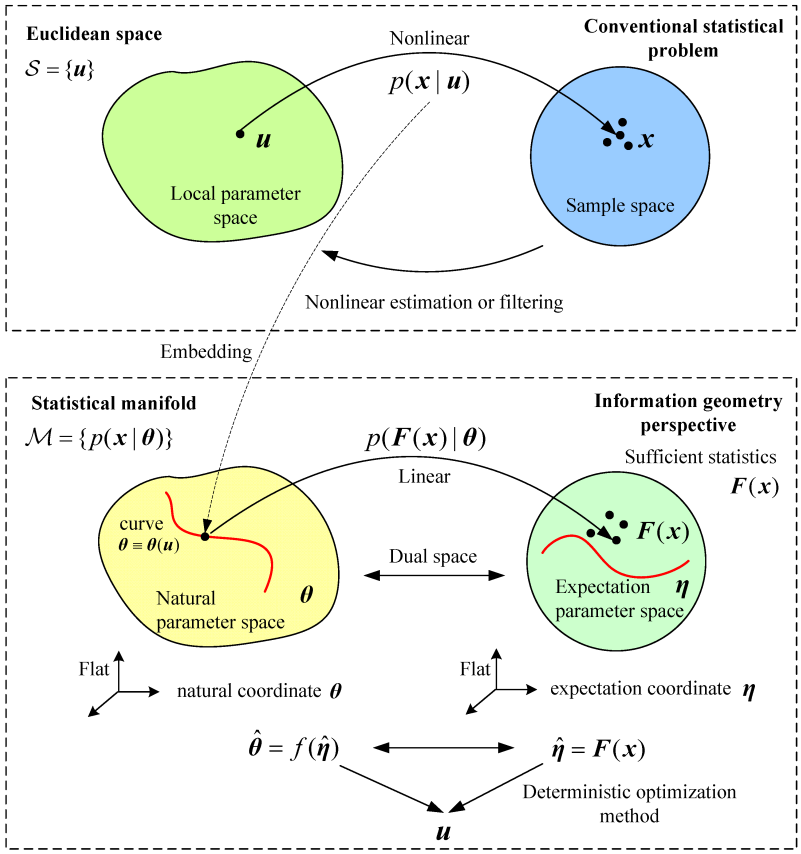


Figure 3.8: Levaraging the geometry of the Riemannian manifold improves the gradient for MLE. Taken from [10]

3.1.6 Fisher Information Matrix

Given any regular statistic distribution parameterized by ξ, $p(x,\xi)$, the Fisher Information Matrix is, by definition:

$$\begin{aligned} F_{ij} &= E_p[\partial_{\xi_i} l(x,\xi)\partial_{\xi_j} l(x,\xi)] \\ &= -E_p[\frac{\partial^2}{\xi_i\xi_j} l(x,\xi)] \\ &= \int \partial_{\xi_i} l(x,\xi)\partial_{\xi_j} l(x,\xi)p(x,\xi)dx, \end{aligned}$$

where $l(x,\xi) = \ln p(x,\xi)$. This defines the unique Riemann metric for any distribution, due to Chentsov's Theorem.

To see that $E_p[\partial_{\xi_i} l(x,\xi)\partial_{\xi_j} l(x,\xi)] = -E_p[\frac{\partial^2}{\xi_i\xi_j} l(x,\xi)]$ holds, we consider the equation $E_p[\partial_j \ln p] = 0$, which by taking the derivative on both sides:

$$\begin{aligned} \partial_i E_p[\partial_j \ln p] = \partial_i \int p\partial_j \ln p dx = 0 \\ = \int p\frac{\partial^2 \ln p}{\partial_i\partial_j}dx + \int \frac{\partial p}{\partial_i}\partial_j \ln p dx \end{aligned}$$

By applying the identity $p\partial_i \ln p = \partial_i p$:

$$\begin{aligned} = \int p\frac{\partial^2 \ln p}{\partial_i\partial_j}dx + \int p\partial_i \ln p\partial_j \ln p dx = 0 \\ E_p[\partial_{\xi_i} l(x,\xi)\partial_{\xi_j} l(x,\xi)] = -E_p[\frac{\partial^2}{\xi_i\xi_j} l(x,\xi)] \end{aligned}$$

In our case of interest, the class of exponential distributions, we have:

$$\begin{aligned} F_{ij} &= \int -\frac{\partial^2}{\xi_i\xi_j} l(x,\xi)p(x,\xi)dx \\ &= \int \frac{\partial^2\psi(\xi)}{\partial\xi_i\partial\xi_j} p(x,\xi)dx \end{aligned}$$

since $\int p(x,\xi)dx = 1$, and that's the only term depending on x:

$$= \frac{\partial^2\psi(\xi)}{\partial\xi_i\partial\xi_j} = g_{ij},$$

which is the result given in subsection 3.1.4. It's also easy to see that $\mathbf{G} = cov[h(x)]$. Since we have $\partial_i \ln p = h_i(x) - \partial_i\psi(\theta)$ and $\partial_i\psi(\theta) = \mu_i$, it follows that $E_p[\partial_{\xi_i} l(x,\xi)\partial_{\xi_j} l(x,\xi)] = E_p[(h_i - \mu_i)(h_j - \mu_j)] = cov[h(x)]$.

In statistical inference, one commonly wishes to infer the model parameters ξ from $p(x,\xi)$ that best fits the data. Considering the estimator as a function of the observed data $\hat{\xi} = f(x_1,\ldots,x_n)$, the bias $b(\xi)$ defines the discrepancy between the

estimation and the true parameter:

$$b(\xi) = E_p[\hat{\xi}] - \xi$$

For a large N, an unbiased estimator satisfies:

$$\lim_{N\to\infty} b(\xi) = 0$$

Now we're ready to state the well-known Crámer-Rao Theorem:

Theorem 1 (Crámer-Rao)**.** *Given an asymptotically unbiased estimator $\hat{\xi}$ the following inequality holds:*

$$V = E[(\hat{\xi}_i - \xi_i)(\hat{\xi}_j - \xi_j)] \succeq \frac{1}{N} G^{-1} \tag{3.3}$$

, in the sense that $V - G^{-1}$ is positive semidefinite. The main result of this theorem is that we have a lower bound for the error covariance matrix of any statistical estimator: the inverse of the Fisher Information Matrix. Moreover, the second part of this theorem states that the lower bound is attained when the estimator is from the exponential family.

3.1.7 Affine Connection

In this subsection we see what kind of structure is induced by the Bregman Divergences. Let $\tau(S)$ be the set of vector fields of class C^∞ in S and $F(S)$ the ring of real functions of class C^∞ defined in S. An affine connection ∇ is hence an application

$$\nabla : \tau(S) \times \tau(S) \to \tau(S),$$

denoted by $(X, Y) \xrightarrow{\nabla} \nabla_X Y$ that connects nearby tangent spaces and satisfies the following properties:

- $\nabla_{fX+gY} Z = f\nabla_X Z + g\nabla_Y Z$
- $\nabla_X (Y + Z) = \nabla_X Y + \nabla_X Z$
- $\nabla_X (fY) = f\nabla_X Y + X(f)Y,$

given that $X, Y, Z \in \tau(S)$ and $f, g \in F(S)$. We can express the field X and Y in terms of local coordinates:

$$X = \sum_i x_i \partial_i$$
$$Y = \sum_j y_j \partial_j$$
$$\text{with } \partial_i = \frac{\partial}{x_i}$$

From above definitions, we can now establish $\nabla_X Y$ as:

$$\nabla_X Y = \sum_{i,j=1}^{n} x_i y_j \nabla_{\partial_i} \partial_j + \sum_{j=1}^{n} X(y_j) \partial_j$$

An important element of differential geometry are the Christoffel symbols, which defines how the vector field $\nabla_{\partial_i} \partial_j$ changes in terms of the coordinate fields ∂_k:

$$\nabla_{\partial_i} \partial_j = \sum_k \Gamma_{ij}^k \partial_k$$

The Levi–Civita connection is the only affine connection (without torsion) such that $X\langle Y, Z\rangle = \langle \nabla_X Y, Z\rangle + \langle Y, \nabla_X Z\rangle$. However, when it comes to Bregman divergences, far more interesting are the dual connections (∇, ∇^*), with $(\Gamma_{ijk}^D, \Gamma_{ijk}^{D*})$. They are defined by taking the third derivative of the divergence:

$$\Gamma_{ijk}^D = -\frac{\partial^2}{\partial \xi^i \partial \xi^j} \frac{\partial}{\partial \xi'^k} D[\xi, \xi']$$
$$\Gamma_{ijk}^{D*} = -\frac{\partial^2}{\partial \xi'^i \partial \xi'^j} \frac{\partial}{\partial \xi^k} D[\xi, \xi']$$

Moreover, they are equal to zero.

$$\Gamma_{ijk}^D = \Gamma_{ijk}^{D*} = 0$$

This means that the basis vector from a tangent plane $T_\xi \mathcal{M}$ doesn't change to another in $T_{\xi'} \mathcal{M}$: in other other words, we have a global coordinate system θ given by the natural parameters of the distribution and another one, μ, given by the expected statistics. Hence, the structure induced by the Bregman divergences is a dually flat manifold.

3.1.8 Other divergences

So far we have studied Bregman divergences and their properties. There are, however, many other functions to measure dissimilarities. In this subsection we bring some other common divergences.

f-divergences

Any divergence that can be written in the form

$$D_f[p,q] = \sum_i p_i f(\frac{q_i}{p_i})$$

, with $f(1) = 0$ and f being a convex and differentiable function is an f-divergence. They are invariant and decomposable. See Amari [7] for the definition of Invariance Criterion.

One important f-divergence is the Hellinger. It measures the distance between probability distributions:

$$H^2(P,Q) = \frac{1}{2}\int_{\mathcal{X}} \left(\sqrt{(}Pdx) - \sqrt{(}Qdx)\right)^2$$

, where $\mathcal{X}$ is the measure space.

α-divergences

The α-divergence has the following form:

$$D_\alpha[p,q] = \frac{4}{1-\alpha^2}(1 - \sum_i p_i^{\frac{1-\alpha}{2}} q_i^{\frac{1+\alpha}{2}}), \alpha \neq \pm -1$$

The KL-divergence is an α-divergence for $\alpha = -1$. The α-divergences is the unique class belonging both to f-divergences and Bregman divergences.

Rényi divergence

Given $\alpha \in (0,1) \bigcup (1,\infty)$ it is defined as

$$D_\alpha[p,q] = \frac{1}{\alpha - 1} \log \int p^\alpha q^{1-\alpha} d\mu$$

, it generalises the KL-divergence.

3.2 Pattern Recognition

Finally we're ready to establish how exponential families are used for clustering. We consider in this section the estimation of a model $M = \{p(x, \xi)\}$ where x is formed by some observed variables y and hidden ones, h, such that $M = \{p(y, h, \xi)\}$. Also, we give a geometric interpretation for the MLE.

3.2.1 Geometric Interpretation of MLE

First, let's take the joint distribution denoted by $q(y, h)$. It is not fully observed, but we can rewrite as a product of the empirical observed distribution $q(y)$ and the conditional $q(h|y)$. The conditional distribution is arbitrary, so we consider all possible candidates lying in the data (sub)manifold:

$$D = \{q(y, h)|q(y, h) = q(y)q(h|y), \sum_h q(h|y_i) = 1, \forall_i\} \tag{3.4}$$

Now we use the empirical distribution, given by:

$$q(y) = \frac{1}{N} \sum \delta(y - y_i) \tag{3.5}$$

Hence, the full distribution is:

$$q(y, h) = \frac{1}{N} \sum \delta(y - y_i)q(h|y) \tag{3.6}$$

Because a convex combination with respect to $q(y, h)$ also belongs to the submanifold D (see eq. 3.7), the data manifold is called m-flat. See [7], page 37, for further definitions of m-flat and e-flat structure of exponential distributions.

$$\lambda_1 q_1(y, h) + \lambda_2 q_2(y, h) = (\lambda_1 q_1(y|h) + \lambda_2 q_2(y|h))q(y) \tag{3.7}$$

Now, how can one relate the manifold given by the model parameters $M = \{p(x, \xi)\}$ and the data manifold D? The following theorem states this relation:

Theorem 2. *The MLE is the minimizer of the KL-divergence from D to M*

Before going into the demonstration, let's first state this useful lemma.

Lemma 1. *The e-projection from a point of M to D does not alter the conditional distribution $q(h|y)$ and hence the conditional expectation of h.*

The proof of the lemma above can be show as follows:

Proof.

$$KL[q(y,h), p_\xi(y,h)] = \int\int q(y,h) \log\left(\frac{q(y,h)}{p_\xi(y,h)}\right)$$
$$= \int\int q(y)q(h|y) \log\left(\frac{q(h|y)q(y)}{p_\xi(h|y)p_\xi(y)}\right)$$
$$= \int\int q(y)q(h|y)\left[\log\left(\frac{q(h|y)}{p_\xi(h|y)}\right) + \log\left(\frac{q(y)}{p_\xi(y)}\right)\right]$$
$$= \int q(y) \log\left(\frac{q(y)}{p_\xi(y)}\right)\int q(h|y)dh + \int q(y)\int q(h|y)\log\left(\frac{q(h|y)}{p_\xi(h|y)}\right)dhdy$$

Since $\int q(h|y)dh = 1$, the first term doesn't depends on h, which leads to:

$$= \int q(y) \log\left(\frac{q(y)}{p_\xi(y)}\right) + \int q(y)\int q(h|y)\log\left(\frac{q(h|y)}{p_\xi(h|y)}\right)dhdy$$
$$= \int q(y) \log\left(\frac{q(y)}{p_\xi(y)}\right) + \int q(y)KL[q(h|y), p_\xi(h|y)]$$

The minimum above is attained minimizing the KL divergence of the second term, which is attained iff $q(h|y) = p_\xi(h|y)$. This finishes the proof of theorem 2 for the e-step. □

From the above lemma we have that $q(h|y) = p(h|y,\xi)$. It's proof is essentially what we've found in equation 2.24 for Gaussian Mixture Model. We wanted to maximize the lower bound in e-step, and to achieve so we set $q(h|y) = p(h|y,\xi)$. Now we see that this result can be generalized to any distribution.

Now we go through the demonstration of theorem 2 for the m-step.

Proof.

$$KL[D,M] = KL[q(y)q(h|y), p(y,h,\xi)] = \int q(y)q(h|y)\log\frac{q(y)q(h|y)}{p(y,h,\xi)}dydh$$
$$= \int q(y)[\int q(h|y)\log q(h|y)dh - q(y)\int q(h|y)\log p(y,h,\xi)dh]dy$$

We seek to minimize above expression with respect to ξ and $q(h|y)$. To minimize with respect to ξ, we see that only the second term is a function of ξ:

$$\min_\xi KL[D,M] = \min_\xi -\int q(h|y)\log p(y,h,\xi)dh$$
$$= \max_\xi \int q(h|y)\log p(y,h,\xi)dh$$

We must take the derivative of above with respect to ξ and make it equals to zero:

$$\frac{\partial}{\partial \xi} \int q(h|y) \log p(y, h, \xi) dh = 0$$
$$\int \frac{q(h|y)}{p(y, h, \xi)} \frac{\partial p(y, h, \xi)}{\partial \xi} dh = 0$$

Now we use the lemma that got us $q(h|y) = p(y, h, \xi)$, which replacing in last equation:

$$\int \frac{\partial p(y, h, \xi)}{\partial \xi} dh = 0$$

This is exactly the expression for MLE and finishes the proof. The ξ found in this step is called the m-projection from a point in D to M. □

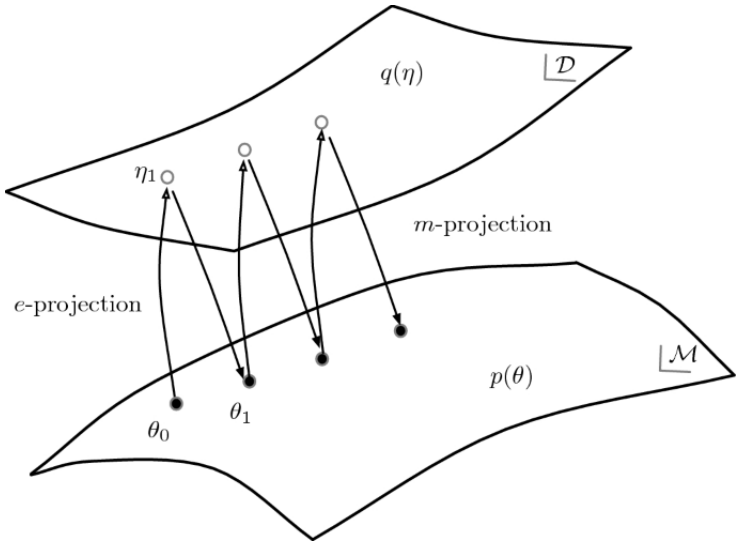


Figure 3.9: Expectation maximization as successive projections between data manifold D and model manifold M. Both M and D are submanifolds of the manifold of all distributions in $P(x) : M, D \subset P(x)$. Because we have a dual flat manifold, the projection is unique. Taken from [11]

Now, one may be wondering, what if instead using KL divergence to make the projection we used another one? That's exactly what Yu Fujimoto and Noboru Murata proposes in [35], called the UM-algorithm, which instead of using the e-projection, they use the u-projection with u denoting the convex function to measure the distance between the manifolds.

3.2.2 Clustering with Bregman Divergences

Now we discuss two procedures, hard and soft Bregman clustering, proposed by Banerjee et al. [8] to find clusters given a general Bregman divergence. The hard

procedure finds a partition in which each point either belongs or not to a certain cluster, while the soft procedure attributes a probability distribution over the clusters.

Let's begin estimating the parameter s given the observations X of a cluster. Consider a random variable $X \in R^d$ following a probability measure p. Given a Bregman divergence d_ϕ, the objective function for finding the optimal cluster representative s is:

$$J_\phi(s) = \min_{s \in R^d} E_p[d_\phi(X, s)] = \min_{s \in R^d} \sum_i^N p_i d_\phi(x_i, s) \tag{3.8}$$

The solution is given $s^* = \mu = E_p[X]$, which is the unique and global minimizer since ϕ is convex.

Proof. To show that the expectation of X is the global minimizer of Equation 3.8, we begin writing the difference between the objective function for any value of s and the optimal $s^* = \mu$ in terms of a divergence:

$$\begin{aligned} J_\phi(s) - J_\phi(\mu) &= E_p[d_\phi(X, s)] - E_p[d_\phi(X, \mu)]) \\ &= E_p[\phi(X) - \phi(s) - \langle \nabla\phi(s), X - s\rangle] - E_p[\phi(X) - \phi(\mu) - \langle \nabla\phi(\mu), X - \mu\rangle] \\ &= -E_p[\phi(s)] - \langle \nabla\phi(s), E_p[X] - s\rangle + E_p[\phi(\mu)] + \langle \nabla\phi(\mu), E_p[X] - \mu\rangle \end{aligned}$$

Since, by definition, $E_p[X] = \mu$, and the expectation $E_p[\phi(\cdot)]$ doesn't depend on X:

$$\begin{aligned} &= \phi(\mu) - \phi(s) - \langle \nabla\phi(s), \mu - s\rangle \\ &= d_\phi(\mu, s) \geq 0 \end{aligned}$$

Since $d_\phi(\mu, s) = 0 \iff \mu = s$, it finishes the proof. □

Another strong result from [36] is that the converse is true, i.e., if $E[X]$ is the minimizer, then d_ϕ must be a Bregman divergence.

Proof. Take two points, $X = \{a, b\}$, such that $p_a + p_b = 1$ and $E_p[X] = ap_a + bp_b$. Since $E_p[X]$ is the minimizer, it holds the inequality:

$$p_a F[a, s] + p_b F[a, s] \geq p_a F[a, \mu] + p_b F[b, \mu]$$

And:

$$p_a \frac{\partial F[a, \mu]}{\partial s} + p_b \frac{\partial F[b, \mu]}{\partial s} = 0$$

Now, if we replace $p_a = (\mu - b)/(a - b)$ and $p_b = 1 - (\mu - b)/(a - b) = (a - \mu)/(a - b)$:

$$\frac{\mu - b}{a - b}\frac{\partial F[a,\mu]}{\partial s} = -p_b\frac{\partial F[b,\mu]}{\partial s} = \frac{\mu - a}{a - b}\frac{\partial F[b,\mu]}{\partial s}$$
$$\frac{1}{\mu - a}\frac{\partial F[a,\mu]}{\partial s} = \frac{1}{\mu - b}\frac{\partial F[b,\mu]}{\partial s}$$

As a consequence of above equality we can conclude that, for any $x \in X$, $\frac{1}{s-x}\frac{\partial F[x,s]}{\partial s} =$ constant, and hence we can write as a function of s only:

$$\frac{1}{s - x}\frac{\partial F[x,s]}{\partial s} = H(s) \tag{3.9}$$

Now, we define $\phi(s)$ as:

$$\phi(s) = \int_0^s \int_0^{s'} H(t)dtds'$$

Which gives $\phi(0) = \phi'(0) = 0, \phi''(s) = H(s)$. Finally we solve the equation 3.9 by integrating both sides:

$$\int \frac{\partial F[x,s]}{\partial s}ds = \int (s - x)H(s)ds = \int sH(s)ds - x\int H(s)$$
$$\text{make the substitution } \int udv = uv - \int vdu \text{ with:}$$
$$\text{u=s; u'=1; v'=H(s); v} = \int H(s)$$
$$= s\int H(s)dy - \int\int H(s)ds - x\phi'(s)$$
$$= s\phi'(s) - \phi(s) - x\phi'(s)$$

Now we have:

$$F(x,s) - F(x,x) = s\phi'(s) - \phi(s) - x\phi'(s) - x\phi'(x) + \phi(x) + x\phi'(x)$$
$$= \phi(x) - \phi(s) + (s - x)\phi'(s)$$
$$= \phi(x) - \phi(s) - (x - s)\phi'(s)$$

Because $F(x,x) = 0$, the non-negativity of F implies that F is always above the tangent plane in s, and hence we have that ϕ must be convex. □

Finally, we're ready to establish the clustering objective function. Given K clusters, denoting μ_j the representative of cluster j C_j, and $Z \in \{0,1\}^{N\times K}$ such

that $Z_{ij} \iff x_i \in C_j$, we have:

$$\min_{Z,\mu} \sum_{j=1}^{K} \sum_{x_i \in C_j} p_i d_\phi(x_i, \mu_j)$$

To address the above problem, they apply the k-means algorithm, generalized by any divergence function. It is stated in algorithm bellow.

Algorithm 5 Bregman Hard Clustering

Require: : Bregman divergence d_ϕ, initial cluster representatives $(\mu_1, \ldots, \mu_K)$.
do
$C_j \leftarrow \emptyset, \forall j \in (1, \ldots, K)$
for each $x \in \mathcal{X}$ **do**
$h_i = \arg\min_h d_\phi(x_i, \mu_h)$
$X_{h_i} \leftarrow X_{h_i} \bigcup x_i$
for each $C_j \in (C_1, \ldots, C_K)$ **do**
$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$
while Not convergence

The algorithm 5 monotonically decreases the loss function. To see that it suffices to note that:

$$\sum_{j=1}^{K} \sum_{x_i \in C_j} p_i d_\phi(x_i, \mu_j^t) \geq \sum_{j=1}^{K} \sum_{x_i \in C_j} p_i d_\phi(x_i, \mu_j^{t+1}) \geq \sum_{j=1}^{K} \sum_{x_i \in C_j^{t+1}} p_i d_\phi(x_i, \mu_j^{t+1})$$

Now let's see the soft algorithm, for which we have to estimate parameters of a mixture distribution. Under the assumption of an exponential family, we seek to maximize the log-likelihood of the distribution. This is essentially the direct application of MLE. Since we've established the probability distribution in the form $exp\{-d\phi(x, \mu)\}b_\phi(x)$, the MLE is the same as minimizing the negative log-likelihood, i.e., minimizing the total divergence in expectation. Hence, in the soft clustering scenario we must estimate the parameters $M = \{\mu_j, \pi_j\}_{h=1}^{K}$ which maximize the likelihood given by:

$$L(M) = P(x|M) = \sum_{j=1}^{K} \pi_j exp\{-d\phi(x, \mu)\}b_\phi(x)$$

With the parameterization of exponential families considering the expected parameters in mind makes the application of EM algorithm straightforward for any Bregman Divergence as follows:

- The expectation **E-step** can be computed by:

$$p(x_i \in C_j) = \frac{\pi_j exp\{-d\phi(x, \mu_j)\} b_\phi(x)}{\sum_{h=1}^{K} \pi_h exp\{-d\phi(x, \mu_h)\} b_\phi(x)}$$

- The maximization **M-step** can be computed by:

$$\pi_j = \frac{1}{N} \sum_i p(x_i \in C_j)$$
$$\mu_j = \frac{\sum_i p(x_i \in C_j) x_i}{\sum_{i=1}^{N} p(x_i \in C_j)}$$

However, in order to apply the algorithm 6, and assuming we are given the natural parameters and ψ instead, we would have to find ϕ first, given by $\phi(x) = \sup_\theta\{\theta \cdot x - \psi(\theta)\}$. Finding $\phi(x)$ in this case is exactly the same as finding the MLE of $P(x|M^*) = p_{(\psi,\theta)}(x) = \exp\{\langle x, \theta\rangle - \psi(\theta)\}p_0(x)$, and wouldn't be computationally efficient. In our algorithm we hence assume that ϕ is given. Another possibility would be changing the algorithm to compute the gradients of Expected lower bound (ELBO) of log-likelihood in the natural space. Hoffman et al. [37] proposes using the natural gradient to maximize the ELBO of the log-likelihood under the variational approach.

Algorithm 6 Bregman Soft Clustering

Require: : Bregman divergence d_ϕ, initial cluster representatives $(\mu_1, \ldots, \mu_K)$.
do
  for each $x_i \in \mathcal{X}$ **do**
    $p(h = j|x_i) = \frac{\pi_j exp\{-d\phi(x,\mu_j)\}}{\sum_{h=1}^{K} \pi_h exp\{-d\phi(x,\mu_h)\}}$
  for each $C_j \in (C_1, \ldots, C_K)$ **do**
    $\pi_j = \frac{1}{N} \sum_i p(h = j|x_i)$
    $\mu_j = \frac{\sum_i p(h=j|x_i) x_i}{\sum_{i=1}^{N} p(h=j|x_i)}$
while Not convergence

Chapter 4

Iterative Algorithms for Clustering

In this chapter we tackle the probabilistic formulation of clustering networks with node attributes using Bregman divergences, as well as a discussion about the regime that allow us obtain the community labels. Therefore, this chapter is divided in two parts. In the first we bring a discussion about the exact recovery threshold, and the second we discuss algorithms in order to achieve the pseudo - since we adopt the variational approach for the network likelihood - MLE.

4.1 Exact recovery in node attributed networks

Consider a population of n objects, called *nodes*, partitioned into $K \geq 2$ disjoint sets, called *blocks* or communities. A node-labelling vector $z = (z_1, \cdots, z_n) \in [K]^n$ represents this partitioning so that z_i indicates the block of node i. The labels (blocks) of nodes are random variables assumed to be independent and identically distributed such that $\mathbb{P}(z_i = k) = \pi_k$ for some vector $\pi \in (0,1)^K$ verifying that $\sum_k \pi_k = 1$. Hence,

$$\mathbb{P}(z) = \prod_{i=1}^{n} \pi_{z_i}. \tag{4.1}$$

The nodes interact in unordered pairs giving rise to undirected edges, and $\mathcal{X}$ is the measurable space of all possible pairwise interactions. Additionally, each node has an attribute that is an element of a measurable space $\mathcal{Y}$. Let $X \in \mathcal{X}^{N \times N}$ denote the symmetric matrix such that X_{ij} represents the interaction between node pair (ij), and by $Y = (Y_1, \cdots, Y_n) \in \mathcal{Y}^n$ the node attribute vector.

Assume that interactions and attributes are independent conditionally on the community labels of the nodes. Let $f_{k\ell}(x)$ denote the probability that two nodes in blocks k and ℓ have an interaction $x \in \mathcal{X}$, and $h_k(y)$ denote the probability that a

node in block $k \in [K]$ has an attribute $y \in \mathcal{Y}$. Thus,

$$\mathbb{P}(X, Y \mid z) = \prod_{1 \le i < j \le n} f_{z_i z_j}(X_{ij}) \prod_{i=1}^{n} h_{z_i}(Y_i). \tag{4.2}$$

In the following, assume that the interaction spaces $\mathcal{X}, \mathcal{Y}$ depend on n, as well as the respective interaction probabilities f, h. In fact, n will increase to infinity while K and π are constant. For an estimator $\hat{z} \in [K]^n$ of z, define the *absolute classification error* as

$$\text{loss}(z, \hat{z}) = \min_{\tau \in \mathcal{S}_K} \text{Ham}(z, \tau \circ \hat{z}),$$

where $\mathcal{S}_K$ is the set of permutations of $[K]$ and $\text{Ham}(\cdot, \cdot)$ is the hamming distance between two vectors. An estimator $\hat{z} = \hat{z}(X, Y)$ achieves *exact recovery* if $\mathbb{P}(\text{loss}(z, \hat{z}) \ge 1) = o(1)$.

4.1.1 Exact recovery threshold in node-attributed SBM

The difficulty of classifying empirical data in one of K possible classes is traditionally measured by the *Chernoff information* [38]. More precisely, in the context of network clustering, let $\text{CH}(a, b) = \text{CH}(a, b, \pi, f, h)$ denote the hardness of distinguishing nodes that belong to block a from block b. This quantity is defined by

$$\text{CH}(a, b) = \sup_{t \in (0,1)} \text{CH}_t(a, b), \tag{4.3}$$

where

$$\text{CH}_t(a, b) = (1 - t) \left[\sum_{c=1}^{K} \pi_c \, \text{D}_t(f_{bc} \| f_{ac}) + \frac{1}{n} \, \text{D}_t(h_b \| h_a) \right] \tag{4.4}$$

is the *Chernoff coefficient* of order t across blocks a and b, and $\text{D}_t(f \| g) = \frac{1}{t-1} \log \int f^t(x) g^{1-t}(x) dx$ is the *Rényi divergence* of order t between two probability densities f, g [39]. The key quantity assessing the possibility or impossibility of exact recovery in SBM is then the minimal Chernoff information across all pairs of clusters. We denote it by $I = I(\pi, f, h)$, and it is defined by

$$I = \min_{\substack{a, b \in [K] \\ a \neq b}} \text{CH}(a, b). \tag{4.5}$$

The following Theorem provides the information-theoretic threshold for exact recovery in node-attributed SBM.

Theorem 3. *Consider model* (4.2) *with* $\pi_a > 0$ *for all* $a \in [K]$. *Denote by* a^*, b^* *the two hardest blocks to estimate, that is* $\mathrm{CH}(a^*, b^*) = I$. *Suppose for all* $t \in (0,1)$, $\lim_{n\to\infty} \frac{n}{\log n}\mathrm{CH}_t(a^*, b^*)$ *exists and is strictly concave. Then the following holds:*

(i) exact recovery is information-theoretically impossible if $\frac{n}{\log n} I < 1$;

(ii) exact recovery is information-theoretically possible if $\frac{n}{\log n} I > 1$.

The proof for Theorem 3 is provided in our recently published paper [40], and some examples are provided below.

Remark 1 (Binary SBM with no attributes)**.** Suppose that $f_{ab} \sim \mathrm{Ber}(\alpha_{ab} n^{-1} \log n)$ where α_{ab} are constants. A Taylor-expansion of the Rényi divergence between Bernoulli distributions leads to

$$I \;=\; (1+o(1))\frac{\log n}{n} \min_{a\neq b} \sup_{t\in(0,1)} \left(\sum_c \pi_c \left[t\alpha_{bc} + (1-t)\alpha_{ac} - \alpha_{bc}^t \alpha_{ac}^{1-t} \right] \right),$$

which indeed coincides with the expression of the Chernoff-Hellinger divergence defined in [41].

Example 1 (Binary SBM with Gaussian attributes)**.** Suppose that $f_{ab} \sim \mathrm{Ber}(\alpha_{ab} n^{-1} \log n)$ and $h_a \sim \mathrm{Nor}(\mu_a \log n, \sigma^2 I_d)$, where α_{ab} and μ_a are independent of n. Then,

$$I \;=\; (1+o(1))\frac{\log n}{n} \min_{a\neq b} \sup_{t\in(0,1)} \left(\sum_c \pi_c \left[t\alpha_{bc} + (1-t)\alpha_{ac} - \alpha_{bc}^t \alpha_{ac}^{1-t} \right] + t\frac{\|\mu_b - \mu_a\|_2^2}{2\sigma^2} \right).$$

In particular, the technical conditions of Theorem 3 are verified if we rule out the uninformative case where all the α_{ab}'s and the μ_a's are equal to each other. Thus, exact recovery is possible if

$$\min_{a\neq b} \sup_{t\in(0,1)} \left(\sum_c \pi_c \left[t\alpha_{bc} + (1-t)\alpha_{ac} - \alpha_{bc}^t \alpha_{ac}^{1-t} \right] + t\frac{\|\mu_b - \mu_a\|_2^2}{2\sigma^2} \right) > 1.$$

Further assuming that $\alpha_{ab} = \alpha 1(a=b) + \beta 1(a\neq b)$ (homogeneous interactions) and $\pi = \left(\frac{1}{K}, \cdots, \frac{1}{K}\right)$ (uniform block probabilities), the expression of I simplifies to

$$I \;=\; (1+o(1))\frac{\log n}{n} \left[\frac{\left(\sqrt{\alpha} - \sqrt{\beta}\right)^2}{K} + \frac{\Delta^2}{8\sigma^2} \right],$$

where $\Delta = \min_{a\neq b} \|\mu_a - \mu_b\|_2$. This last scenario recovers the recently established threshold for exact recovery in the Contextual SBM [42].

Example 2 (Semi-supervised clustering in binary homogeneous SBM)**.** Consider homogeneous binary interactions given by

$$f_{ab} \sim \begin{cases} \mathrm{Ber}(\alpha n^{-1} \log n) & \text{if } a = b, \\ \mathrm{Ber}(\beta n^{-1} \log n) & \text{otherwise,} \end{cases}$$

where α, β are constants independent of n. Consider a semi-supervised model in which the vector of attributes Y is a noisy oracle of the true community labels z. More precisely, for a node i such that $z_i = k$, we have

$$h_k(y) = \mathbb{P}(Y_i = y) \;=\; \begin{cases} 1 - \eta & \text{if } y = 0, \\ \eta_1 & \text{if } y = k, \\ \frac{\eta_0}{K-1} & \text{if } y \in [K] \backslash \{k\}, \end{cases}$$

with $\eta_0 + \eta_1 = \eta$. A bit of algebra shows that exact recovery is possible if

$$\left(\sqrt{\alpha} - \sqrt{\beta}\right)^2 - \frac{2}{\log n} \log\left(1 - \eta + 2(K-1)^{-1/2}\sqrt{\eta_0 \eta_1}\right) > K.$$

When $\eta_0 = 0$ (perfect oracle), the condition simplifies to $\left(\sqrt{a} - \sqrt{b}\right)^2 - \frac{2\log(1-\eta)}{\log n} > K$. Note that the oracle term is non-negligible only if $-\log(1-\eta) \gtrsim \log n$, as previously established [43]. This last condition is very strong, since it implies that the oracle must provide the correct label for almost all nodes, in particular $\eta \gtrsim 1 - 1/n$.

4.2 Bregman clustering of node-attributed networks

4.2.1 Model formulation

We consider the model defined in (4.2) and we suppose that $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \mathbb{R}^d$ $(d \geq 1)$. We also assume that the network is sparse, namely for all blocks $a, b \in [K]$ we have

$$f_{ab}(x) \;=\; (1 - p_{ab})\delta_0(x) + p_{ab} f^*_{ab}(x), \tag{4.6}$$

where f^*_{ab} is a probability density with no mass at zero. Finally, we suppose that the sets of distributions $\{f^*_{ab}\}$ and $\{h_a\}$ belong to two exponential families. More precisely, we have (with respect to the appropriate measure)

$$f^*_{ab}(x) \;=\; e^{<\theta_{ab}, x> - \psi(\theta_{k\ell})} \quad \text{and} \quad h_a(y) \;=\; e^{<\eta_a, y> - \phi(\eta_k)}, \tag{4.7}$$

where $\theta_{k\ell}, \eta_k$ are parameters and ψ, ϕ are two functions. In particular, $\psi(\theta_{k\ell}) = \log \mathbb{E}_{f_{k\ell}} e^{<\theta_{ab}, X>}$ is the cumulant generating function of $f_{k\ell}$. Let us denote $\mu_{k\ell} = \mathbb{E}_{f_{k\ell}}(X) = \nabla\psi(\theta_{k\ell})$ and $\nu_k = \mathbb{E}_{h_k}(Y) = \nabla\phi(\eta_k)$. We recall the following relationships for the log-likelihood of exponential families (see for example [44, Equation 13])

$$\begin{aligned} -\log f^*_{ab}(x) &= d_{\psi^*}(x, \mu_{ab}) - \psi^*(x), \\ -\log h_a(x) &= d_{\phi^*}(x, \mu_a) - \phi^*(x). \end{aligned} \tag{4.8}$$

The following Lemma expresses the log-likelihoods of f_{ab} in terms of Bregman divergences.

Lemma 2. *Let f_{ab} be a probability density as defined in* (4.6)-(4.7)*. For $x, y \in (0,1)$, let $\mathrm{d}_{\mathrm{KL}}(x,y)$ be the Kullback-Leibler divergences between $\mathrm{Ber}(x)$ and $\mathrm{Ber}(y)$, and let $H(x) = x\log x + (1-x)\log(1-x)$. Then,*

$$-\log f_{ab}(x) = \mathrm{d}_{\mathrm{KL}}\left(x \| p_{k\ell}\right) + c d_{\psi^*}(x, \mu_{ab}) - c\psi^*(x) - H(c),$$

where $c = 1(x \neq 0)$.

Proof. To express $\log f_{ab}(x)$, we first note that

$$\log f_{ab}(x) = \begin{cases} \log(1-p_{ab}) & \text{if } x \neq 0, \\ \log p_{ab} + \log(f^*_{ab}(x)) & \text{otherwise.} \end{cases}$$

This can be rewritten as

$$\log f_{ab}(x) = (1-c)\log(1-p_{ab}) + c\log p_{ab} + c\log(f^*_{ab}(x)),$$

where $c = 1(x \neq 0)$. The result follows by adding and subtracting $H(b)$ in the previous expression and using (4.8). □

Lemma 3. *Let f_{ab} be a probability density as defined in* (4.6)-(4.7)*. Suppose that $p_{ab} = \Theta(\delta)$ for all a, b, where $\delta \ll 1$. We have*

$$(1-t)\,\mathrm{D}_t(f_{ac} \| f_{bc}) = (1+o(1))\left[tp_{ac} + (1-t)p_{bc} - p^t_{ac}p^{1-t}_{bc}\, e^{-J_\psi(\theta_{ab}:\theta_{bc}}\right]$$

where $J_\psi(\theta_{ab}\|\theta_{bc}) = t\psi(\theta_{ac} + (1-t)\psi(\theta_{bc}) - \psi(t\theta_{ac} + (1-t)\theta_{bc})$.

Proof. Using a Taylor expansion, we have

$$\begin{aligned} (1-t)\,\mathrm{D}_t(f_{ac}\|f_{bc}) &= \log\left[(1-t)^t(1-q)^{1-t} + p^t q^{1-t}\int (f^*)^t (g^*)^{1-t}\right] \\ &= tp + (1-t)q - p^t q^{1-t}\int (f^*)^t(g^*)^{1-t} + o(\delta), \end{aligned}$$

and we finish the proof using $\int (f^*)^t (g^*)^{1-t} = J_\psi(\theta_{ab} \| \theta_{bc})$ (see example [45]). $\square$

4.2.2 Clustering by iterative likelihood maximisation

Assuming that X, Y come from the model (4.2), with probability distributions given by (4.6)-(4.7). Let A be a binary matrix such that $A_{ij} = 1(X_{ij} \neq 0)$. We have

$$-\log \mathbb{P}(X, Y \mid z) \;=\; \sum_i \left\{ \frac{1}{2} \sum_{j \neq i} \left[\mathrm{d_{KL}}(A_{ij}, p_{z_i z_j}) + A_{ij} d_{\psi^*} \left(X_{ij}, \mu_{z_i z_j} \right) \right] + d_{\phi^*}(Y_i, \nu_{z_i}) \right\} + c(X, Y)$$

where $c(X, Y)$ is some function of X, Y only. Denoting $Z \in \{0, 1\}^{N \times K}$ the one-hot membership matrix such that $Z_{ik} = 1(z_i = k)$, we observe that $p_{z_i z_j} = \left(Z p Z^T \right)_{ij}$, $\mu_{z_i z_j} = \left(Z \mu Z^T \right)_{ij}$ and $\nu_{z_i} = \left(Z^T \nu \right)_i$. Thus, up to some constants we have

$$\begin{aligned} & -\log \mathbb{P}(X, Y \mid z) \\ =\; & \sum_i \left\{ \frac{1}{2} \sum_j \left[\mathrm{d_{KL}} \left(A_{ij}, \left(Z p Z^T \right)_{ij} \right) + A_{ij} d_{\psi^*} \left(X_{ij}, \left(Z \mu Z^T \right)_{ij} \right) \right] + d_{\phi^*} \left(Y_i, \left(Z^T \nu \right)_i \right) \right\} \\ =\; & \sum_i \left\{ \frac{1}{2} \mathrm{d_{KL}} \left(A_{i\cdot}, \left(Z p Z^T \right)_{i\cdot} \right) + \frac{1}{2} A_{ij} d_{\psi^*} \left(X_{i\cdot}, \left(Z \mu Z^T \right)_{i\cdot} \right) + d_{\phi^*} \left(Y_i, \left(Z^T \nu \right)_i \right) \right\} \end{aligned} \tag{4.9}$$

where by abuse of notation we denote $d_{\psi^*}(A, B) = \sum_i \sum_j d_{\psi^*}(A_{ij}, B_{ij})$ for two matrix A, B.

Following the log-likelihood estimator established in (4.9), we first propose an iterative clustering algorithm that aims at classifying each node in the community maximising $\mathbb{P}(X, Y \mid z_{-i}, z_i = k)$, the likelihood that node i is in the community k given the community labels of the other nodes z_{-i}. Denoting by $Z^{(ik)}$ the membership matrix obtained from Z by replacing the community of node i to k, we have

$$L_{ik}(Z^{(ik)}) \;=\; \frac{1}{2} \mathrm{d_{KL}} \left(A_{i\cdot}, \left(Z p Z^T \right)_{i\cdot} \right) + \frac{1}{2} A_{ij} d_{\psi^*} \left(X_{i\cdot}, \left(Z \mu Z^T \right)_{i\cdot} \right) + d_{\phi^*} \left(Y_i, \left(Z^T \nu \right)_i \right). \tag{4.10}$$

Finally, the probabilities $p = p(X, Z)$ and the means $\mu = \mu(X, Z)$ and $\nu = \nu(Y, Z)$ are given by

$$\begin{aligned} p(X, Z) \;&=\; \left(Z^T Z \right)^{-1} Z^T A Z \left(Z^T Z \right)^{-1} \\ \mu(X, Z) \;&=\; \left(Z^T Z \right)^{-1} Z^T X Z \left(Z^T Z \right)^{-1} \\ \nu(Y, Z) \;&=\; \left(Z^T Z \right)^{-1} Z^T Y. \end{aligned} \tag{4.11}$$

We note that the matrix inverse $\left(Z^T Z \right)^{-1}$ can be easily computed since $Z^T Z$ is a k-by-k diagonal matrices. We summarise this in Algorithm 7.

Algorithm 7 Bregman hard clustering of node attributed SBM

Require: : Node-pairs interactions $X \in \mathcal{X}^{n\times n}$, node attributes $Y \in \mathcal{Y}^n$, convex functions ψ^*, ϕ^*, initial clustering $Z_{init} \in \mathcal{Z}_{n,K}$.

do

Let $Z = Z_{init}$ and compute p, μ, ν according to (4.11)

for each $i = 1, \cdots, N$ **do**

Find $k^* = \arg\max_{k\in[K]} L_{ik}\left(Z^{(ik)}\right)$, where $L_{ik}\left(Z^{(ik)}\right)$ is defined in (4.10)

Let $Z^{\text{new}}_{ik} = 1(k = k^*)$ for all $k = 1, \cdots, K$

Let $Z = Z^{\text{new}}$

Update p, μ, ν according to (4.11)

while Not convergence

Return Node-membership matrix Z

Alternatively, one can state the objective function and try to optimize the expected divergence defined in (4.12), just like in (3.8):

$$\min \sum_{i=1}^{n}\sum_{k=0}^{K}\tau_{i,k}\left[d_\phi(Y_i,\nu_k) + \sum_{j=1}^{n}\sum_{k=0}^{K}\{d_{KL}(A_{i,j},p_{k,l}) + A_{i,j}d_{\psi*}(X_{i,j},\mu_{k,l})\}\right]$$

s.t.

$$p_{k,l} = \frac{\sum_{i\neq j}\hat{\tau}_{ik}\hat{\tau}_{j\ell}A_{ij}}{\sum_{i\neq j}\hat{\tau}_{ik}\hat{\tau}_{j\ell}}, \quad \hat{\mu}_{k\ell} = \frac{\sum_{i\neq j}\hat{\tau}_{ik}\hat{\tau}_{j\ell}X_{ij}}{\sum_{i\neq j}\hat{\tau}_{ik}\hat{\tau}_{j\ell}} \quad \text{and} \quad \hat{\nu}_k = \frac{\sum_i \hat{\tau}_{ik}Y_i}{\sum_i \hat{\tau}_{ik}} \tag{4.12}$$

$$\sum_{k=0}^{K}\tau_{i,k} = 1 \forall i$$

$$\sum_{i=1}^{n}\tau_{i,k} \geq 1 \forall k$$

$$\tau_{i,k} \geq 0$$

However the above formulation is non-convex, since the Bregman divergences are not generally convex with respect to the second argument. This makes common algorithms from Mixed Integer Programming very slow even for small instances and why we adopt EM instead.

Finally, one can also define the soft clustering algorithm. To begin with, we note that equation (4.6) can be written as an exponential family:

$$\begin{aligned} f_{ab}(x) &= (1-p_{ab})^{\delta(A_{ij})}[p_{ab}g_{ab}(X_{ij})]^{\delta(A_{ij}-1)} \\ &= \exp\{\delta(A_{ij})\log(1-p_{ab}) + \delta(A_{ij}-1)[\log(p_{ab}) + \log(g_{ab}(X_{ij}))]\} \\ &= \exp\{-\mathrm{d}_{\mathrm{KL}}(A_{ij},p_{ab}) - A_{ij}d_{\psi^*}(X_{ij},\mu_{ab})\}. \end{aligned} \tag{4.13}$$

Hence, we can thus write the likelihood as follows:

$$\begin{aligned}\mathbb{P}(X, Y \mid z) &= \prod_{1\le i<j\le n} f_{z_i z_j}(X_{ij}) \prod_{i=1}^{n} h_{z_i}(Y_i) \\ &= \exp\{-\sum_{i,j}\left[\mathrm{d_{KL}}(A_{ij}, p_{z_i,z_j}) + A_{ij} d_{\psi^*}(X_{ij}, \mu_{z_i,z_j})\right] - d_{\phi^*}(Y_i, \nu_{z_i})\}.\end{aligned} \tag{4.14}$$

In order to obtain the conditional distribution $p(z|X, Y)$, we can use the Bayes formula $p(z|X, Y) = p(X, Y|z)p(z)/p(X, Y)$. Hence, the E-step of the EM-Algorithm will proceed as follows:

$$\begin{aligned}p(z_i = a|X, Y, z_{-i}) &\propto p(X, Y|z_{-i}, z_i = a)p(z_i = a) \\ &\propto \pi_a \exp\left\{-\sum_{i,j}\left[\mathrm{d_{KL}}(A_{ij}, p_{a,z_j}) + A_{ij} d_{\psi^*}(X_{ij}, \mu_{a,z_j})\right] - d_{\phi^*}(Y_i, \nu_a)\right\}.\end{aligned} \tag{4.15}$$

We note that the inside sum on the nodes i, j remains constant, except for the entries on i when changing its community. This leads us to the following formula, with $\hat{\tau}_{ia} = p(z_i = a|X, Y)$ and a little abuse of notation:

$$\begin{aligned}d_{net}(j) = \mathrm{d_{KL}}(A_{ij}, p_{a,z_j}) + A_{ij} d_{\psi^*}(X_{ij}, \mu_{a,z_j}) + \mathrm{d_{KL}}(A_{ij}, p_{z_j,a}) + A_{ij} d_{\psi^*}(X_{ij}, \mu_{z_j,a}) \\ \hat{\tau}_{ia} \propto \pi_a \exp\{-\sum_j [d_{net}(j)] - d_{\phi^*}(Y_i, \nu_a)\}\end{aligned}$$

In practice, in order to have a stable exponent, we simply add a constant c_i for every node:

$$c_i = \min_a\{\sum_j [d_{net}(j)] + d_{\phi^*}(Y_i, \nu_a)\}$$

$$\hat{\tau}_{ia} \propto \pi_a \exp\{-\sum_j [d_{net}(j)] - d_{\phi^*}(Y_i, \nu_a) + c_i\} \tag{4.16}$$

Finally, given the conditional probabilities, we can update the distribution parameters as follows:

$$\hat{\pi}_k = \frac{1}{n}\sum_i \hat{\tau}_{ik}, \quad \hat{\mu}_{k\ell} = \frac{\sum_{i\neq j} \hat{\tau}_{ik}\hat{\tau}_{j\ell} X_{ij}}{\sum_{i\neq j} \hat{\tau}_{ik}\hat{\tau}_{j\ell}} \quad \text{and} \quad \hat{\nu}_k = \frac{\sum_i \hat{\tau}_{ik} Y_i}{\sum_i \hat{\tau}_{ik}}. \tag{4.17}$$

The steps are described in Algorithm 8. For the convergence we just check if either the number of iterations is reached or the loglikelihood between two sucessive steps is lower than an arbitrary epsilon: $|l(\hat{\tau}_{new}) - l(\hat{\tau}_{old})| \le \epsilon$, where the total loglikelihood is $l(\hat{\tau}) = \sum_i \log \sum_k \hat{\tau}_{ik}$.

Algorithm 8 Bregman soft clustering of node attributed SBM

Require: : Node-pairs interactions $X \in \mathcal{X}^{n\times n}$, node attributes $Y \in \mathcal{Y}^n$, convex functions ψ^*, ϕ^*, initial clustering $Z_{init} \in \mathcal{Z}_{n,K}$.
do
  for each $i = 1, \cdots, N$ **do**
    update τ_{ia} according to (4.16)
  update π,μ and ν according to (4.17)
while Not convergence

4.2.3 Initialisation

To provide a good initialisation for Algorithm 7, we first estimate two clustering $Z_{network}$ (resp. $Z_{attributes}$) obtained by applying Algorithm 7 (starting with a random initialisation) on the network (resp. on the attributes) alone. We note that, given an estimated clustering $\hat{Z}$, we can compute $\hat{\mu}(X, \hat{Z})$ and $\nu(Y, \hat{Z})$ as in (4.11), and hence get estimates $\hat{f}_{ab}$ and $\hat{h}_a$ of the probability densities f_{ab} and h_a. Since Theorem 3 highlights the role of the *Chernoff-Hellinger* divergence to assess the difficulty of clustering, we proceed as follows:

- Use $Z_{network}$ to compute numerically

$$C_{network} = \min_{a\neq b} \sup_{t\in(0,1)} (1-t) \sum_{c=1}^{K} \pi_c \, \mathrm{D}_t \left(f_{bc} \| f_{ac}\right)$$

- Use $Z_{attributes}$ to compute numerically

$$C_{attribute} = \min_{a\neq b} \sup_{t\in(0,1)} (1-t) \left[\frac{1}{n} \mathrm{D}_t \left(h_a \| h_b\right)\right]$$

- Return the $Z_{network}$ if $C_{network} > C_{attributes}$, otherwise return $Z_{attributes}$.

Alternatively one can also initialize using the communities found by spectral clustering in algorithm 9. The idea of using spectral clustering as initialization for Kmeans is debated in [46], which shows that by doing this helps the Kmeans objective function escape from local minima. We hence adapt this idea using the Bregman Divergences to compute a similarity score between data points. For network, one can use as the similarity matrix the graph itself or a known metric from literature, such as Jaccard index. For the data, one can use $\exp\{-d_\phi(y_i, y_j)\}$ for every pair of data points i and j and ϕ the specified divergence.

Since the Bregman Divergences are not symmetric in general, one can make the symmetric version by doing $D_{\phi sym} = d_\phi(y_i, y_j) + d_\phi(y_j, y_i)/2$. One can take the average of both network and attributes similarities to compute a single matrix, however the assumption that the graph and attributes come from different manifolds

work better in practice. That's why it's preferable to compute two distinct laplacian eigenmaps and then concatenate their embeddings. The implementation of such algorithms can be found here [1].

Algorithm 9 Spectral clustering on concatenated matrix

Require: : Observed network data X, attributes Y, a (symmetric) kernel function Φ, number of clusters K.

1: Do some preprocessing on X to obtain $\tilde{X}$ (e.g., compute Jaccard similarity between the neighbourhood of different nodes, normalize by degrees, or do nothing);

2: Let $\tilde{Y} \in \mathbb{R}_+^{n \times n}$ such that $\tilde{Y}_{ij} = K(Y_i, Y_j)$

3: Compute the Laplacian of each matrix, resulting in $\tilde{X}_L$ and $\tilde{Y}_L$

4: (i) Let the eigendecomposition of $\tilde{X}_L$ be $\tilde{X}_L = \sum_{i=1}^{n} \lambda_i u_i u_i^t$, with $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and eigenvectors $u_1, \cdots, u_n$. Denote $U = (u_1, \cdots, u_K) \in \mathbb{R}^{n \times K}$ the leading eigenspace, and $\Lambda = (\lambda_1, \cdots, \lambda_K) \in \mathbb{R}^{K \times K}$ the leading eigenvalues.

5: (ii) Similarly, let $\tilde{Y}_L = \sum_{i=1}^{n} \sigma_i v_i v_i^T$, with $\sigma_1 \leq \cdots \leq \sigma_n$. Denote $V = (v_1, \cdots, v_K)$ and $\Sigma = (\sigma_1, \cdots, \sigma_K)$.

6: (iii) Apply k-means on the rows of $[U\Lambda, V\Sigma] \in \mathbb{R}^{n \times 2K}$, where $[\cdot, \cdot]$ denotes the concatenation between two matrices.

7: Return estimated clusters $\hat{Z}$ obtained by k-means at step (iii).

[1] `https://github.com/FelipeSchreiber/BregmanClustering`

Chapter 5

Evaluation

In this chapter we compare our proposal with others in the literature and with traditional algorithms. In order to achieve this goal, the evaluation was done with both synthetic and real data. Also, a number of metrics were studied. This chapter is hence divided into two parts, one that analyses under synthetic data sets and another the real ones, but first we bring a couple of metrics.

5.1 Metrics

The first thing before thinking about metrics is, what I would like to measure? What would be a perfect score?

Since we're dealing with clustering algorithms, that are unsupervised by nature, the most common approach is to measure the similarity between clusterings, or how close two sets of labelings are. Also, since we're dealing with sets, the ordering doesn't matter in this case, as opposed to classification metrics or recommendation ones. Another aspect is that each set inside a clustering may have different sizes, so we would also like that our metric is unbiased by the size of them. Thereby we present a couple of known ways to compute such similarity, and discuss their shortcomings.

5.1.1 Mutual Information and variants

As the name suggests, it is the normalization of the Mutual Information (MI) score to scale the results between 0 (no mutual information) and 1 (perfect correlation). But, as pointed out by Mark Newman et al. in [47], it's biased. Let's see why.

Let's first describe the mutual information. Mathematically, it's given by:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \ln \frac{|U_i \cap V_j|N}{|U_i||V_j|}$$

, with $|U_i|$ and $|V_j|$ denoting the number of points in cluster U_i and V_j respectively. The main question is, what would be a good value of Mutual Information? A value of 10 can be optimal in an application, but terrible for others. To have a comparable measure, one normally take the normalized version, which is:

$$NMI(U,V) = \frac{MI(U,V)}{avg(H(U),H(V))}$$

The last formulation has some drawbacks. First, it has a bias towards labelings with too many distinct label values. For example, an algorithm that outputs each data point as a community will lead to the highest score of mutual information, while we intuitively expect this output to have no information whatsoever. Secondly, the normalization factor depends on both the ground truth and the output, which may mislead rankings. Another major problem is that the expectation of mutual information is greater than zero, which again is counter intuitive because the random labeling would be expected to be zero. One possible alternative to this last problem is to take the adjusted score [48] instead:

$$AMI(U,V) = \frac{MI(U,V) - E(MI(U,V))}{avg(H(U),H(V)) - E(MI(U,V))}$$

For the other shortcomings many proposals exist, for example normalizing by a measure that depends only on ground truth [47] or take the reduced mutual information which adds a normalization term [49].

5.1.2 Rand Index

The Rand Index [13] is another similarity measure between two clusterings. It consider all pairs of samples and take the ratio between the number of agreeing pairs and total:

$$RI = \frac{\text{Agreeing pairs}}{\text{Total pairs}}$$

In order to factor out the random labelings, normally one subtracts the expected value, giving the adjusted score:

$$ARI = \frac{RI - E[RI]}{max(RI) - E[RI]}$$

5.1.3 Sokal&Sneath

This metric was first introduced in [50] and later debated on [51]. For notation purposes, consider the 2×2 pair matrix K of all sample pairs of two partitions U

and V. Let's discuss the meaning of each entry:

1. K_{11} is the number of pairs in the same cluster
2. K_{00} is the number of pairs that are in different clusters
3. K_{10} is the number of pairs that are in the same cluster in U but in different clusters in V
4. K_{01} is the number of pairs that are in the same cluster in V but in different clusters in U

With this definition in mind, we're to establish the Sokal&Sneath metric:

$$\frac{1}{4}\Big(\frac{K_{11}}{K_{11}+K_{10}}+\frac{K_{11}}{K_{11}+K_{01}}+\frac{K_{00}}{K_{00}+K_{10}}+\frac{K_{00}}{K_{00}+K_{01}}\Big),$$

which is the average of precision, recall and their inverted counterparts.

5.1.4 Pearson Correlation Coefficient

It is one of the most practical indices to measure similarity, although not very used for clustering similarity measure. In terms of K, it is expressed by the following formula:

$$\frac{K_{11}K_{00}-K_{10}K_{01}}{(K_{00}+K_{01})(K_{00}+K_{10})(K_{11}+K_{01})(K_{11}+K_{10})}$$

Interestingly, according to [51] the best agreeing indices are Sokal&Sneath and Pearson. They also satisfy many properties. See image 5.1 bellow.

	Max. agreement	Min. agreement	Symmetry	Distance	Lin. complexity	Monotonicity	Strong monotonicity	Const. baseline	As. const. baseline	Type of bias	Const. baseline
R	✓	✓	✓	✓	✓	✓	✓	✗	✗	⤫	✗
AR	✓	✗	✓	✗	✓	✓	✗	✓	✓		✗
J	✓	✗	✓	✓	✓	✓	✗	✗	✗	↙↙↙	✗
W	✗	✗	✗	✗	✓	✗	✗	✗	✗		✓
D	✓	✗	✓	✗	✓	✓	✗	✗	✗		✗
CC	✓	✓	✓	✗	✓	✓	✓	✓	✓		✗
S&S	✓	✓	✓	✗	✓	✓	✓	✓	✓		✗
CD	✓	✓	✓	✓	✓	✓	✓	✗	✓		✓

Figure 5.1: Metrics properties. R-Rand Index, AR-Adjusted Rand Index, J-Jaccard, W-Wallace, D-Dice, CC-Pearson Correlation, S&S-Sokal&Sneath and CD-Correlation Distance respectivelly. Taken from [12].

	Max. agreement	Symmetry	Distance	Lin. complexity	Monotonicity	Const. baseline
NMI	✓	✓	✗	✓	✓	✗
$\text{NMI}_{\max}$	✓	✓	✓	✓	✗	✗
FNMI	✓	✗	✗	✓	✗	✗
VI	✓	✓	✓	✓	✓	✗
SMI	✗	✓	✗	✗	✗	✓
FMeasure	✓	✓	✗	✓	✗	✗
BCubed	✓	✓	✗	✓	✓	✗
AMI	✓	✓	✗	✗	✓	✓

Figure 5.2: Similarity metric properties. AMI is often preferred over NMI because it's unbiased towards the number of clusters. Taken from [12].

5.2 Datasets

Finally, we're ready to talk about the datasets used to evaluate our model. They are divided between real and synthetic data. Let's see them.

5.2.1 Real data sets

The pytorch geometric [1] provides some citation networks where n papers are linked by m edges. Each paper belongs to one of the K classes and is represented by a keyword denoted by a d-dimensional 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. However, some processing was needed since some nodes are disconnected. We remove them for the sake of simplicity and, due to high dimensionality of the attributes, we also reduce them to 10 most meaningful features, according to the Chi-square test. The choice of this test is just to preserve the original data and simplicity, although other methods could be used, such as Principal Components Analysis (PCA) or t-SNE [52].

- **Cora**: $n = 2708$, $m = 10556$, $d = 1433$ and $K = 7$ (machine learning fields);
- **Citeseer**: $n = 3327$, $m = 9104$, $d = 3703$ and $K = 6$;
- **Wiscosin**: $n = 251$, $m = 515$, $d = 1703$, $K = 5$;
- **Texas**: $n = 183$, $m = 325$, $d = 1703$, $K = 5$;
- **Cornell**: $n = 183$, $m = 298$, $d = 1703$, $K = 5$;

The Chi-square measures the correlation of each feature and the target variable, i.e., the class. We select the 10 highest scores after computing the metric.

[1] `https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html`

Mathematically, it is:

$$\chi^2(D,t,c) = \sum_{e_w \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_w e_c} - E_{e_w e_c})^2}{E_{e_w e_c}},$$

where N is the observed frequency, E the expected frequency, e_w takes value 1 if the paper contains word w and 0 otherwise and e_c takes the value 1 if the paper is in class c and 0 otherwise.

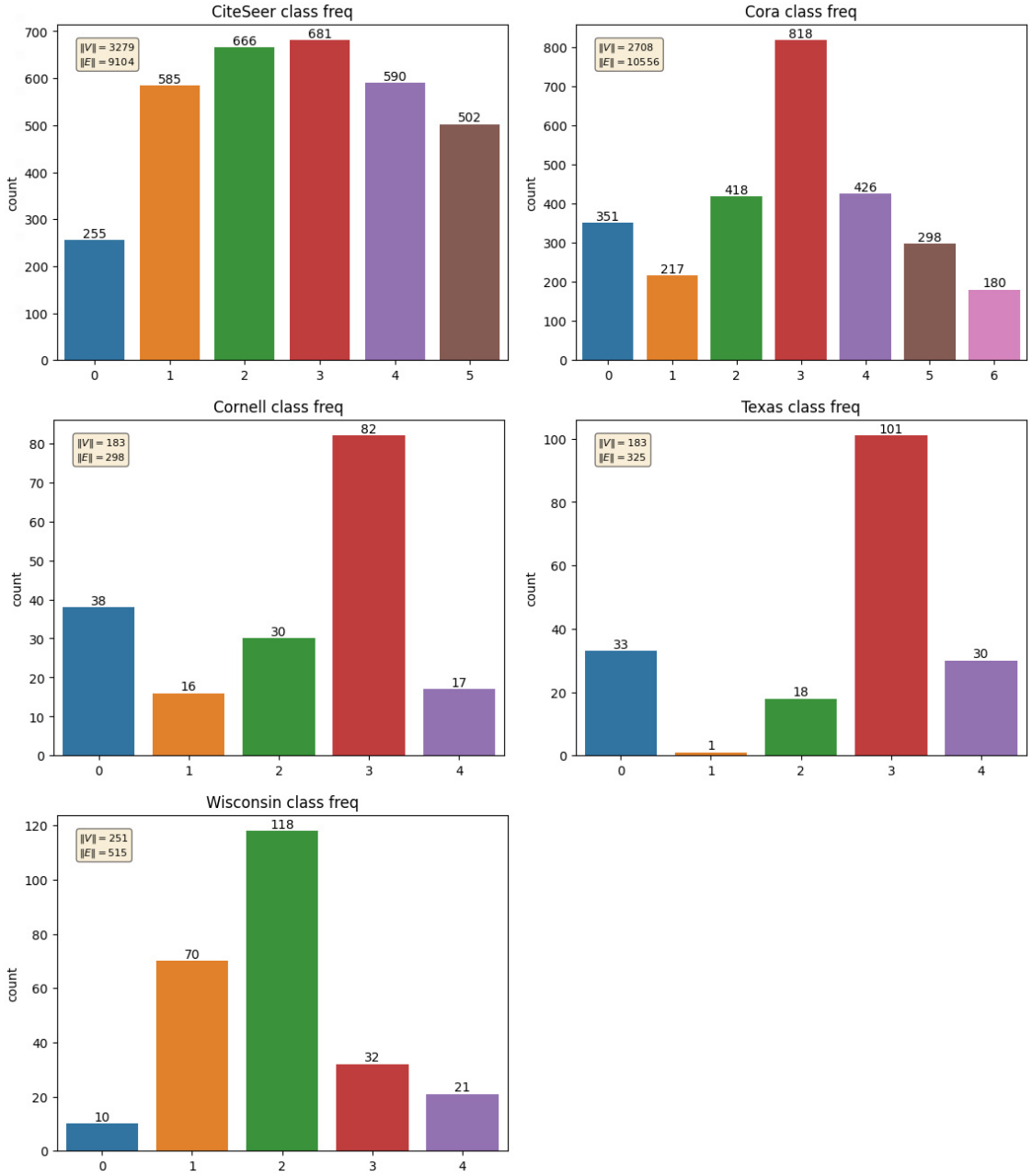


Figure 5.3: Class distribution of each dataset

5.2.2 Synthetic datasets

Since the community recovery depends on both the attributes and the network, our tests were thought to stress the proposed algorithm and his performance when varying the strength of network signal and the attributes signal. The network and attributes are generated independently. The network comes from a Stochastic Block Model, with edge weights coming from a distribution whose mean depends on the node classes of interation. In a broad sense the synthetic datasets have the following parameters to be set:

- p_{in} e p_{out} SBM parameters
- The size of each community
- The distribution from which the attributes and weights come from
- The circle radius
- A $K \times K$ matrix to establish the means of weight distribution between each community pair

Since we make different assumptions about attributes distribution depending on the divergence selected, we also study if there's much performance difference when changing the divergence. As a summary, we perform the following tests:

- We fix the p_{out} SBM parameter and the circle radius to draw the features center. Vary the p_{in} parameter
- We fix the p_{out} and p_{in} SBM parameters. Vary the circle radius
- Draw graph weights from a specific distribution and assume other ones
- Draw attributes from a specific distribution and assume other ones

5.3 Results

5.3.1 Synthetic Datasets

Finally we show a couple of results concerning our algorithms. In Figure Figure 5.5, we compare the performance of Algorithm 7 in terms of exact recovery (fraction of times the algorithm correctly recovers the community of all nodes) with the theoretical threshold for exact recovery proved in the paper (red curve in the plots) in two settings: Figure 5.5a shows binary weight with Gaussian attributes, and Figure 5.5b shows zero-inflated Gaussian weights with Gaussian attributes. Solid black and white squares represent fraction zero (no trial was recovered exactly) and one (all trials were exactly recovered) over 50 trials.

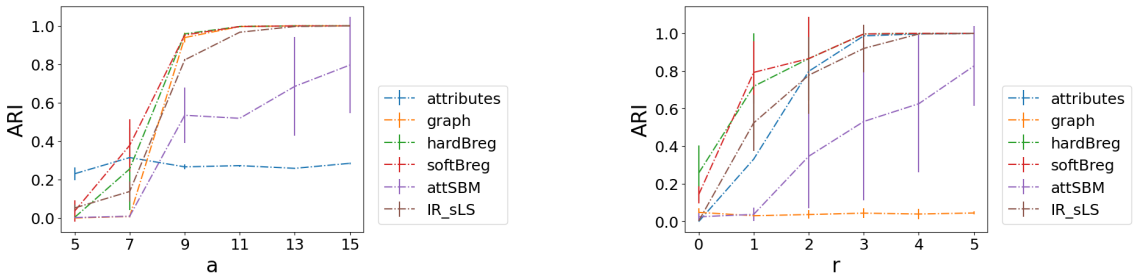
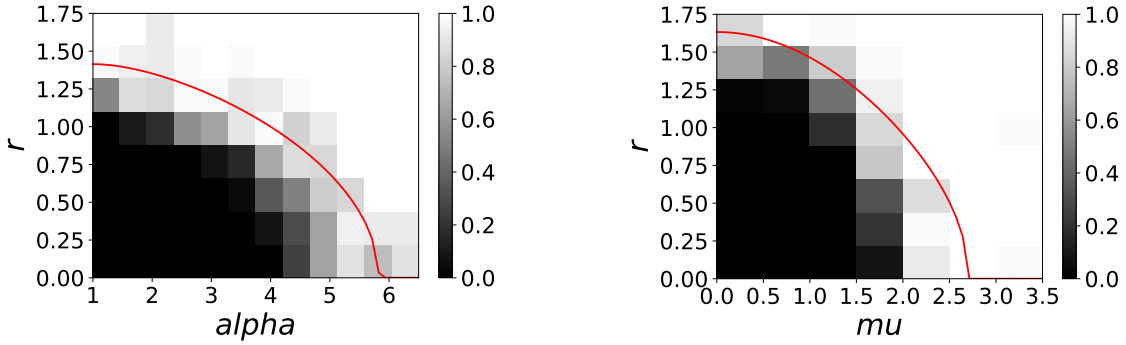


Figure 5.4: (a)We fix $p_{out} = 5\frac{\log n}{n}$ and $r = 1$. Vary $p_{in} = a\frac{\log n}{n}$
(b) We fix $p_{out} = 5\frac{\log n}{n}$, and $p_{in} = 8\frac{\log n}{n}$. Vary radius r.
Both experiments: Gaussian attributes, unweighted graph, n = 600, K = 3



(a) Binary network with Gaussian attributes

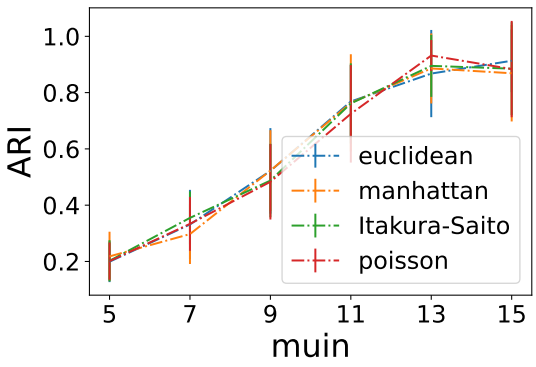
(b) zero-inflated Gaussian weights with Gaussian attributes.

Figure 5.5: Phase transition of exact recovery. Each pixel represents the empirical probability that Algorithm 1 succeeds at exactly recovering the clusters (over 50 runs), and the red curve shows the theoretical threshold.
(a) $n = 500$, $k = 2$, $= \text{Ber}(\alpha n^{-1} \log n)$, $= \text{Ber}(n^{-1} \log n)$. The attributes are 2d-spherical Gaussian with radius $(\pm r\sqrt{\log n}, 0)$ and identity covariance matrix.
(b) $n = 600$, $k = 3$, $= (1-\rho)\delta_0 + \rho\,\text{Nor}(\mu, 1)$, $= (1-\rho)\delta_0 + \rho\,\text{Nor}(0, 1)$ with $\rho = 5n^{-1} \log n$. The attributes are 2d-spherical Gaussian whose means are the vertices of a regular polygon on the circle of radius $r\sqrt{\log n}$.

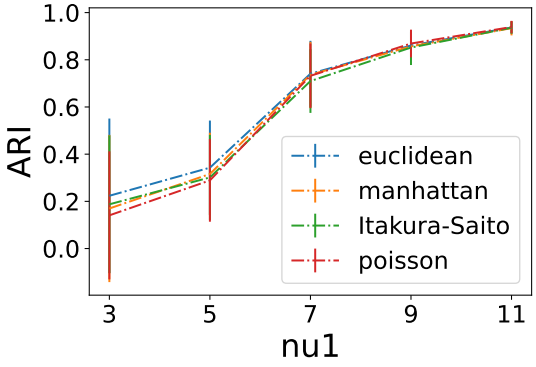
Robustness to the choice of d_{ψ^*} and d_{ϕ^*}

We then show in Figure 5.6 that using a divergence (distribution) for edge weights (Figure 5.6a and node attributes (Figure 5.6b different from the distribution used to generate the data does not impact the results. We note that a similar observation was done in previous papers using Bregman divergence for clustering [14, 44].

Finally, we compare Algorithm 7 with other algorithms presented in the literature. More precisely, in Figure 5.7 we compare Algorithm 7 with the V-EM algorithm of [15] and the algorithm of [14]. Both of these algorithms are designed for dense networks, which explains why Algorithm 7 has overall better performance.

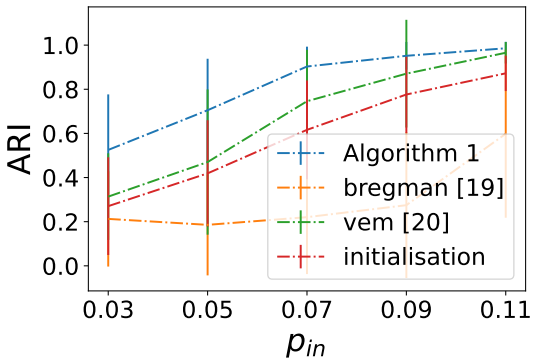


(a) Various d_{ψ^*}. Poisson is the correct model.

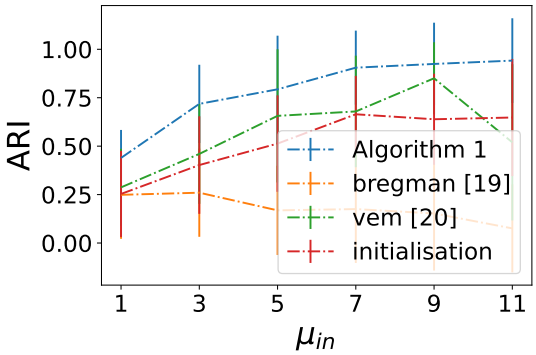


(b) Various d_{ϕ^*}. Poisson is the correct model.

Figure 5.6: Performance of Algorithm 7 when d_{ψ^*} or d_{ϕ^*} do not correspond to the model that generated the data. The different curves show the Adjusted Rand Index (ARI) [13] averaged over 20 realisations with the standard deviations as error bars.
(a) $n = 400$, $k = 4$, $= (1-p)\delta_0(x) + p\text{Poi}(\mu_{in})$ and $= (1-q)\delta_0(x) + q\text{Poi}(5)$, with $p = 0.04$ and $q = 0.01$. Attributes are 2d-Gaussians with unit variances and mean equally spaced the circle of radius $r = 2$.
(b) $n = 400$, $k = 2$, $= (1-p)\delta_0(x) + p\text{Nor}(2,1)$ and $= (1-q)\delta_0(x) + q\text{Nor}(0,1)$, with $p = 0.04$ and $q = 0.01$. Attributes are Poisson with means ν_1 (for nodes in cluster 1) and 3 (for nodes in cluster 2).



(a)



(b)

Figure 5.7: Comparison of Algorithm 1 with algorithms of [14] and [15] Error bars show the standard deviations. Results are averaged over 25 realisations. Attributes are 2d-spherical Gaussian attributes with radius $(\pm 1, 0)$.
(a) $n = 100$, $k = 2$, $= (1 - p_{\text{in}})\delta_0 + p_{\text{in}}\,\text{Poi}(5)$, $= (1 - 0.03)\delta_0 + 0.03\,\text{Poi}(1)$.
(b) $n = 100$, $k = 2$, $= (1 - 0.07)\delta_0 + 0.07\,\text{Poi}(\mu_{\text{in}})$, $= (1 - 0.04)\delta_0 + 0.04\,\text{Poi}(1)$.

5.3.2 Real Datasets

It's known that the k-means objective function is non convex, but a experiment suggested by [46] shows that by initializing with the results from spectral clustering (SC) helps avoidance of local minima, although it alone (SC) isn't the best option. We further do as suggested, and explored the initialization with such method, followed by the MLE optimization routine. As shown in table bellow, we were able to

improve the results a lot with such procedure.

Finally we show a couple of results comparing the algorithms with different initializations. In particular, we study the initialization inspired in the spectral theory. For such experiment, we consider traditional algorithms, such as Leiden/Spectral Clustering for network only, Kmeans and GMM for attributes only, and ours that considers both. For spectral methods considering both sources of information (SC_-jaccard, SC_gaussian_1 and SC_gaussian_2) we do as following:

- SC_jaccard: Computes a network simmilarity matrix using Jaccard index and compute an elementwise average with the attributes simmilarity matrix with euclidean divergence in the formula $\exp\{-d_\phi(y_i, y_j)\}$.

- SC_gaussian_1: Computes a network simmilarity matrix using euclidean distance from adjacency matrix, i.e. $\exp\{-||A_{i:} - A_{j:}||_2^2\}$. Compute an elementwise average with the attributes simmilarity matrix as before.

- SC_gaussian_2: Computes a network simmilarity matrix using euclidean distance from adjacency matrix, i.e. $\exp\{-||A_{i:} - A_{j:}||_2^2\}$. Compute attributes simmilarity matrix as before. The spectral embedding is done independently, instead of previous methods that computes a single vector of length K for each node. The final step is to concatenate both network and attributes embedding, resulting in an embedding of length 2K, and apply a clustering algorithm.

- Hard and Soft clustering + SC: Here we apply using SC_gaussian_2 procedure as initialization, and latter apply the MLE as we've discussed.

We conclude from tables 5.15.2 that in fact applying the spectral initialization the results are improved, as well as it's variance. We note that the spectral method followed by MLE performs better than MLE or SC alone, which is expected from [46] experiments. With that being said, we further highlight the following points:

- According to S&S and CC metrics, the method using Soft clustering with Spectral Clustering for initialisation outperformed the hard clustering counterparts in most of datasets - except for Cornell. The same is true if we analyze the metrics of table 5.1, but since we're dealing with unbalanced data the former metrics are more accurate.

- Surprisingly, in Texas dataset the soft clustering with SC under performed the soft version with default initialization according to NMI and AMI metrics, but not according to others (ARI, S&S and CC). This may be because of bias in mutual information computation.

Table 5.1: Real data experiments: comparison with Normalized Mutual Information, Adjusted Mutual Information and Adjusted Rand Index

NMI	NMI_std	ARI	ARI_std	AMI	AMI_std	algorithm	dataset
0.21346867984328521	0.006959179357894665	0.04479413540185353	0.005369659446045768	0.2113563412092557 5	0.0069790566603647715	both_hard	CiteSeer
0.259581863786	0.0	0.20342453026007018	0.0	0.25792099609969127	0.0	both_hard+SC	CiteSeer
0.2057177812598197	0.009085922866314907	0.0329162978696433	0.005583040602577974	0.20352780437873488	0.009122677164754447	both_soft	CiteSeer
0.2833791032144337	0.0	0.2385932030331515	0.0	0.2818048292215374 3	0.0	both_soft+SC	CiteSeer
0.24877777858489608	0.0	0.1672733317193305	0.0	0.24709597818740087	2.7755575615628914e-17	attSBM	CiteSeer
0.1819581463164466	0.006296970340672214	0.02460972125165921	0.0036210198507557975	0.17965366186418713	0.006316614542094532	leiden(init)	CiteSeer
0.220997661910293	0.03177807037423423	0.13342173106924732	0.03752675767661495	0.2191346656032629	0.03192086866874422	GMM(init)	CiteSeer
0.18988696819854478	0.0	0.08904461595553734	0.0	0.1879098163809906	0.0	kmeans	CiteSeer
0.012793775010289282	1.734723475976807e-18	-0.000623016915563686	0.0	0.008500445718461149	0.0	SC	CiteSeer
0.26304303525973327	0.0	0.18140350178924042	0.0	0.26138914544610764	0.0	SC_jaccard	CiteSeer
0.2599317923771575	0.0	0.18060143478463445	0.0	0.25827614638374313	0.0	SC_gaussian_1	CiteSeer
0.24145296785311604	0.0	0.1630151078812451	0.0	0.23974973355067347	2.7755575615628914e-17	SC_gaussian_2	CiteSeer
0.27785498128988906	0.0207650324303173	0.05861911245130261 4	0.010160345728454964	0.27471778060001649	0.020842078861655646	both_hard	Cora
0.3417992721235633	0.0	0.12249541627871334	0.0	0.33899526513023726	0.0	both_hard+SC	Cora
0.40817991088224 39	0.07082209406986 16	0.2398723221251137	0.07448550143187115	0.40593027762237044	0.07112371187724759	both_soft	Cora
0.4471667745956017	0.0	0.260883098413428	0.0	0.44509426461725143	5.551115123125783e-17	both_soft+SC	Cora
0.30797879718191334	0.0	0.08764904312131717	0.0	0.30499504792253546	0.0	attSBM	Cora
0.36322871063843426	0.023209400067515967	0.18831886377226945	0.018933189074597557	0.3606845787552606	0.023306534490622047	leiden(init)	Cora
0.2743056631982082	0.020222848204216784	0.05778418983687773 3	0.00965157744954359	0.27114723907039006	0.020302220735797728	GMM(init)	Cora
0.2492179911475192 5	0.0	0.04287224332920651	0.0	0.2459603739381418	0.0	kmeans	Cora
0.01378077197743811	1.734723475976807e-18	-0.002880600459865 16	0.0	0.006487747603360862	0.0	SC	Cora
0.3309619309290932	0.0	0.11902880585670625	0.0	0.32827325048305805	0.0	SC_jaccard	Cora
0.33051741067821533	0.0	0.09811415445365605	0.0	0.32776220502680947	0.0	SC_gaussian_1	Cora
0.2918152984372345	0.0	0.06894745133323943	0.0	0.28867755063082234	0.0	SC_gaussian_2	Cora
0.11094695084494213	0.011931796813416317	0.026993491785026674	0.009203995835303801	0.0811139740953358	0.01234501360830146	both_hard	Cornell
0.5509098955046718	0.0	0.489240576894001	5.551115123125783e-17	0.5337959525296577	0.0	both_hard+SC	Cornell
0.21979254856757519	0.06867082577539528	0.07287847196922739	0.04132719432247644	0.19278609285099185	0.07013724886107328	both_soft	Cornell
0.5088401882449745	0.0	0.44645097338778583	0.0	0.4898984141215868	5.551115123125783e-17	both_soft+SC	Cornell
0.5143417710672298	0.0	0.46455091039122304	5.551115123125783e-17	0.4957462872066614	0.0	attSBM	Cornell
0.07549604535943057	0.004029461404109076	-0.006270664213819011	0.00469497159281923	0.043225387696503026	0.004438060917145745	leiden(init)	Cornell
0.4304528869278582	0.020707320765794145	0.36941195137671745	0.01234370750698467	0.40792968373182237	0.022056458138317914	GMM(init)	Cornell
0.45626820799150936	0.0	0.33426136186459 23	0.0	0.4335960930298411	0.0	kmeans	Cornell
0.08217155290042287	0.0	0.014029998338900711	0.0	0.043872352857615975	0.0	SC	Cornell
0.40171522702198664	0.0	0.28324784831172045	0.0	0.38079630539968035	0.0	SC_jaccard	Cornell
0.420696469118474 36	0.0	0.4202301844001294	0.0	0.3991622136905946	0.0	SC_gaussian_1	Cornell
0.4917269616758416	5.551115123125783e-17	0.42219523796592534	0.0	0.47188941033398796	5.551115123125783e-17	SC_gaussian_2	Cornell
0.13014289152043407	0.08116643355980985	0.1358950753610042	0.06803917690089009	0.10068374302457425	0.083862642481019	both_hard	Texas
0.3667595189594293	0.0	0.32582633206400596	0.0	0.3444871880846265	0.0	both_hard+SC	Texas
0.403867579958915	0.007918921931131438	0.3366708594648505	0.07371110877422221	0.38381476733900155	0.008592901600580932	both_soft	Texas
0.3701127897012879	0.0	0.4509939751936239	0.0	0.34740756723620586	0.0	both_soft+SC	Texas
0.3730509643591937	0.0	0.448710892830522	0.0	0.350581649311867	0.0	attSBM	Texas
0.04408124931574434	0.0013042320016693086	0.009405521227490972	0.003038764117383252	0.010324781569764664	0.0010058801381217298	leiden(init)	Texas
0.3926014406173393	0.02549318752200398	0.4364180501902964	0.032529442937379735	0.3688722298029294	0.025714248885746378	GMM(init)	Texas
0.43890884957969123	0.0	0.47808711551595595	5.551115123125783e-17	0.4166903523203039	0.0	kmeans	Texas
0.015408869980161754	0.0	0.006779451645372704	8.673617379884035e-19	0.004710391630037535	0.0	SC	Texas
0.39272825055941557	0.0	0.26606864308219275	0.0	0.3718143818618677	0.0	SC_jaccard	Texas
0.3307901598755094	0.0	0.3576843465536411	0.0	0.3066962881984197	0.0	SC_gaussian_1	Texas
0.3613123759968967	0.0	0.35077469205769934	0.0	0.33896748511678554	0.0	SC_gaussian_2	Texas
0.21305073595140683	0.016074208945793967	0.15569619820019592	0.03285918332253669	0.1936516619407785	0.016396796054837556	both_hard	Wisconsin
0.4144623723744895	0.0	0.4003927116254896	0.0	0.3990953700747937	0.0	both_hard+SC	Wisconsin
0.3360835802651604	0.0236168453307472	0.27765407194970615	0.023415468677856658	0.31999013726365433	0.024416419116558837	both_soft	Wisconsin
0.49274513881409626	0.0	0.4326935905448816	0.0	0.48018272253181904	5.551115123125783e-17	both_soft+SC	Wisconsin
0.48094755335407	0.0	0.433093827984973	0.0	0.4670846673844187	0.0	attSBM	Wisconsin
0.04204616233647699	0.003964627326648053	-0.006794231613112098	0.015808351197907265	0.016202767836663603	0.003943757708596698	leiden(init)	Wisconsin
0.38661693815379183	0.011387021984914 44	0.354822983896564	0.028804439034231168	0.36994761253187 63	0.011607273242143 35	GMM(init)	Wisconsin
0.44218180411986185	0.0	0.36970073266446146	0.0	0.426288413190181	5.551115123125783e-17	kmeans	Wisconsin
0.08753904874115358	0.0	0.04943644344228711	0.0	0.0611052470801942	6.938893903907228e-18	SC	Wisconsin
0.3558502657083752	0.0	0.23054042011933001	0.0	0.339567146497894	0.0	SC_jaccard	Wisconsin
0.37953463699234263	0.0	0.33179352501109804	0.0	0.36334295410462714	0.0	SC_gaussian_1	Wisconsin
0.47718994060731534	5.551115123125783e-17	0.41434495522947035	5.551115123125783e-17	0.4631205605104495	0.0	SC_gaussian_2	Wisconsin

- Apart from Texas dataset in which Kmeans is the best, the methods considering both sources of information did better. In this particular one, Kmeans got S&S score of 0.74, while our soft + SC method is the second best with 0.72. In this same dataset the difference between GMM and Leiden is the biggest, which may suggest a bias favorable to attributes only methods.

- Apart from Cora dataset, all datasets seems to have more attribute information in comparison to network information, since GMM (attributes only method) often does better than Leiden (network only)

- Apart from Winscosin dataset in which according to ARI the attributed SBM is the best, all the metrics reached a consensus of the best performing algorithm on every experiment.

Table 5.2: Real data experiments: comparison with Sokal&Sneath and Correlation Coefficient metrics

S&S	S&S_std	CC	CC_std	algorithm	dataset
0.5300494085670995	0.003605798597739101	0.05800499648739317	0.006960312854227995	both_hard	CiteSeer
0.6024645307639732	0.0	0.20451698681997663	0.0	both_hard+SC	CiteSeer
0.5232144117101322	0.0036642837741560894	0.044840621275832326	0.0070770617339243795	both_soft	CiteSeer
0.6193347299474888	0.0	0.23864739360285547	0.0	both_soft+SC	CiteSeer
0.5844213646796723	0.0	0.1684168191553715	0.0	attSBM	CiteSeer
0.5181076441692193	0.0027518903892251104	0.035023381509223926	0.005531998673818602	leiden(init)	CiteSeer
0.5705951521357389	0.01779584341522195	0.13944442268353266	0.03577433198904634	GMM(init)	CiteSeer
0.5501554634648966	0.0	0.0979837358951223	0.0	kmeans	CiteSeer
0.4957825650088162	0.0	-0.005115736264428648	0.0	SC	CiteSeer
0.591672006173999	0.0	0.18282033659043864	0.0	SC_jaccard	CiteSeer
0.5911113926174268	0.0	0.18178198469398982	0.0	SC_gaussian_1	CiteSeer
0.5824049230338664	0.0	0.16432613318657788	0.0	SC_gaussian_2	CiteSeer
0.5355439242743328	0.0061474266091625105	0.06892920009476358	0.011922553299823752	both_hard	Cora
0.5685312212956654	0.0	0.13404797824606526	0.0	both_hard+SC	Cora
0.621415931227131	0.036859495152658135	0.24206236261735786	0.07387594997524789	both_soft	Cora
0.6314235770136144	0.0	0.26231353709828864	0.0	both_soft+SC	Cora
0.5517199543626476	0.0	0.10053095995628084	1.3877787807814457e-17	attSBM	Cora
0.6019480804748347	0.009781445312013753	0.20043973452267255	0.019382358842140207	leiden(init)	Cora
0.5350495652676974	0.005821772362574524	0.06796871819880543	0.011293645226999329	GMM(init)	Cora
0.5261459359896783	0.0	0.05068206132120533	6.938893903907228e-18	kmeans	Cora
0.48575253125847667	0.0	-0.019674582817118372	0.0	SC	Cora
0.5649320589310372	0.0	0.12749531617805906	0.0	SC_jaccard	Cora
0.5553561601326535	0.0	0.10815241562893792	0.0	SC_gaussian_1	Cora
0.5415292386372659	0.0	0.08057677787087981	0.0	SC_gaussian_2	Cora
0.5136161379431157	0.004658523136779102	0.02718712041475387	0.00929508769009895	both_hard	Cornell
0.7511393439872041	0.0	0.5009025617377737	0.0	both_hard+SC	Cornell
0.5380532572495016	0.022333799112504136	0.07584356889568729	0.044411102162164914	both_soft	Cornell
0.7309438319978814	0.0	0.46036991971585256	0.0	both_soft+SC	Cornell
0.7390860658705549	0.0	0.47676918498038645	5.551115123125783e-17	attSBM	Cornell
0.49686284854126966	0.0023473121483484737 7	-0.006273686557305158	0.004694379341487168	leiden(init)	Cornell
0.6939331954196233	0.006780905406228025	0.38632904716401184	0.013435312379842194	GMM(init)	Cornell
0.6823871791537457	0.0	0.3628568888150612	0.0	kmeans	Cornell
0.5171808294150825	0.0	0.03244322704356505	0.0	SC	Cornell
0.6416394139538386	0.0	0.2832738178274195	0.0	SC_jaccard	Cornell
0.7132759063445633	0.0	0.42580211957170483	0.0	SC_gaussian_1	Cornell
0.719844223417698	0.0	0.4380657236602191	5.551115123125783e-17	SC_gaussian_2	Cornell
0.5707435162291284	0.03546092916435076	0.14076723826311163	0.07054592332122592	both_hard	Texas
0.6649669857624094	0.0	0.32953213859345204	0.0	both_hard+SC	Texas
0.6692402403185131	0.037481321156439984	0.33832138613528306	0.07484925023694196	both_soft	Texas
0.7271925640211407	0.0	0.4542508823527238	0.0	both_soft+SC	Texas
0.7252013416642535	0.0	0.45026460289542614	0.0	attSBM	Texas
0.5047595761873719	0.001536651275396924	0.0095078798344164	0.0030699512406807154	leiden(init)	Texas
0.7210624240699741	0.01772437750670355	0.4419526337445682	0.03536184989470705	GMM(init)	Texas
0.7424955998664785	0.0	0.4847729373333235	5.551115123125783e-17	kmeans	Texas
0.5514254038717498	0.0	0.04232774777347764	0.0	SC	Texas
0.6373341764479203	0.0	0.2736351617681436	0.0	SC_jaccard	Texas
0.6792995717894718	0.0	0.3585275541853753	0.0	SC_gaussian_1	Texas
0.6784513734530753	0.0	0.3562634535875523	0.0	SC_gaussian_2	Texas
0.5809942156514938	0.016798869015950336	0.16085140140659873	0.03348483159124113	both_hard	Wisconsin
0.7006960666909214	0.0	0.40126208623137005	0.0	both_hard+SC	Wisconsin
0.6395089689635596	0.01183809504444334	0.27884136012871213	0.023627897675137528	both_soft	Wisconsin
0.7199863595576359	0.0	0.4388325192714809	5.551115123125783e-17	both_soft+SC	Wisconsin
0.7166843719353484	0.0	0.43333521810994335	0.0	attSBM	Wisconsin
0.49659724197164373	0.007911212688508773	-0.006804196056809933	0.015820814024841208	leiden(init)	Wisconsin
0.6775386858276904	0.014438321008999727	0.3550483441816756	0.028867977139559426	GMM(init)	Wisconsin
0.6868962279261217	0.0	0.37346426015369016	0.0	kmeans	Wisconsin
0.5257513529173141	0.0	0.05139668278729089	0.0	SC	Wisconsin
0.6174958269000292	0.0	0.23428068693172058	0.0	SC_jaccard	Wisconsin
0.6665143224055772	0.0	0.3328634068610344	0.0	SC_gaussian_1	Wisconsin
0.7071932265519517	0.0	0.41438159940698666	5.551115123125783e-17	SC_gaussian_2	Wisconsin

Chapter 6

Conclusion

In this work we proposed a novel algorithm for community detection in node attributed networks inspired by the Bregman Divergences and exponential families, in an attempt to achieve the maximum likelihood estimator. Real data is not dense in general, so we give an sparse formulation to the problem. Also, we generalized many works by making use of exponential distributions, since most of them mainly assume Gaussian distributions. The task of achieving MLE is an NP-hard combinatorial problem, so we adopted an expectation maximization procedure to tackle the problem. Other studies relied on belief propagation and MCMC procedures.

We compared our solution with others in the literature, an show the superiority of our approach through several experiments with both synthetic and real data by analysing many metrics. Many initialization procedures were studied, and a new one that relies on the spectral theory was proposed. The new initialization proposed empirically improved the results achieved by the MLE approach. Finally, we studied briefly some concepts of information geometry and the dual flat structure induced by Bregman divergences in the parameter space. We also showed how the natural parameters can be useful to the optimization task, since it avoids plateaus, and gave a geometric interpretation for MLE.

6.1 Future work

As future work, one can study other divergences and leverage them to cluster data. Also, instead of minimizing the KL divergence of MLE, one can experiment replacing by other divergences and check the impact of such change in the robustness of solution, i.e., how outliers may affect the final estimator? Other interesting experiments would be test the performance with other benchmarks, such as Lancichinetti-Fortunato-Radicchi (LFR) [53] and Artificial Benchmark for Community Detection [54] benchmarks.

Furthermore, one may also tackle the algorithm scalability. When dealing with

large graphs, equation (4.10) is intractable due to the large number of factors in the sum. In order to compute the divergences with respect to the graph, we consider a stochastic version, defined as follows:

$$\begin{aligned} L_{ik}(Z^{(ik)}) &= \sum_j \left[\mathrm{d_{KL}}\left(A_{i,j}, \left(ZpZ^T\right)_{i,j}\right) + A_{ij} d_{\psi^*}\left(X_{i,j}, \left(Z\mu Z^T\right)_{i,j}\right)\right] + d_{\phi^*}\left(Y_i, \left(Z^T\nu\right)_i\right) \\ &\approx \frac{N}{|S|}\sum_{j\in S} \left[\mathrm{d_{KL}}\left(A_{i,j}, \left(ZpZ^T\right)_{i,j}\right) + A_{ij} d_{\psi^*}\left(X_{i,j}, \left(Z\mu Z^T\right)_{i,j}\right)\right] + d_{\phi^*}\left(Y_i, \left(Z^T\nu\right)_i\right), \end{aligned} \tag{6.1}$$

where S is a set containing the node indexes sampled. By the law of large numbers, the above expression converges to the exact value in (4.10). In order to update the attribute means ν, the Bernoulli parameters p and the expected weight μ for each community pair we do as follows for large graphs:

$$\begin{aligned} p_{a,b}(X,Z) &= \frac{1}{N_a N_b}\sum_{i,j} A_{i,j}\mathbf{1}_{\mathbf{a},\mathbf{b}}(i,j) \approx \frac{N_a N_b}{|S|}\sum_{i,j\in S} A_{i,j}\mathbf{1}_{\mathbf{a},\mathbf{b}}(i,j) \\ \mu_{a,b}(X,Z) &= \frac{\sum_{i,j} X_{i,j}\mathbf{1}_{\mathbf{a},\mathbf{b}}(i,j)}{\sum_{i,j} A_{i,j}\mathbf{1}_{\mathbf{a},\mathbf{b}}(i,j)} \approx \frac{\sum_{i,j} A_{i,j}\mathbf{1}_{\mathbf{a},\mathbf{b}}(i,j)}{\sum_{i,j\in E} A_{i,j}\mathbf{1}_{\mathbf{a},\mathbf{b}}(i,j)}\sum_{i,j\in E} X_{i,j}\mathbf{1}_{\mathbf{a},\mathbf{b}}(i,j) \\ \nu_a(Y,Z) &= \frac{\sum_i Y_i \mathbf{1}_{\mathbf{a}}(i)}{N_a} \approx N_a \frac{\sum_{i\in S} Y_i \mathbf{1}_{\mathbf{a}}(i)}{|S|}, \end{aligned} \tag{6.2}$$

where $\mathbf{1}_{\mathbf{a},\mathbf{b}}(i,j)$ is an indicator function with value 1 when $(z_i, z_j) = (a, b)$, N_x is the size of cluster x, S is random sample of nodes i and j and E denotes a sample from the set of edges. Optionally one could do the update the parameters at time t using a learning parameter α as follows, leading to the mini-batch kmeans:

$$\begin{aligned} target &= N_a \frac{\sum_{i\in S} Y_i \mathbf{1}_{\mathbf{a}}(i)}{|S|} \\ \nu_a(Y,Z)^{t+1} &= \nu_a(Y,Z)^t + \alpha[target - \nu_a(Y,Z)^t] \end{aligned} \tag{6.3}$$

Algorithm 10 Monte Carlo Bregman hard clustering

Require: : Node-pairs interactions $X \in \mathcal{X}^{n\times n}$, node attributes $Y \in \mathcal{Y}^n$, convex functions ψ^*, ϕ^*, initial clustering $Z_{init} \in \mathcal{Z}_{n,K}$.
  Let $Z = Z_{init}$ and compute p, μ, ν according to (4.11)
  do
    $idx\_sample \sim \mathcal{U}(0, N-1)$
    for each $i \in idx\_sample$ **do**
      Find $k^* = \arg\max_{k\in[K]} L_{ik}\left(Z^{(ik)}\right)$, where $L_{ik}\left(Z^{(ik)}\right)$ is defined in (6.1)
      Let $Z_{ik}^{\text{new}} = 1(k = k^*)$ for all $k = 1, \cdots, K$
    Let $Z = Z^{\text{new}}$
    Update p, μ, ν according to (6.2)
  while Not convergence
  Return Node-membership matrix Z

The Monte Carlo version of our algorithm wasn't implemented, so one could

see other aspects such as the sampling size and convergence for this case. Finally, our proposed algorithm optimizes in the expected parameters space, but when dealing with Gaussians of unknown variance for example it would be interesting to see whether optimizing in the natural space is better due to (expected) better convergence properties.

To summarize, possible research directions are:

- Provide an scalable version of our algorithm, and compare with other benchmarks in the literature.
- Use other divergences, so that we can generalize further to other distributions that are not from exponential family, e.g. t-student and Cauchy. Study it's robustness when dealing with outliers. Also, other divergences can be used for a pseudo MLE scheme by alternating projections in data and model manifolds.
- Propose and compare optimization routines in the natural space. Analyze the quality of solution and convergence speed.
- Estimate the number of communities, e.g. by Integrated Completed Likelihood (ICL).
- Assess how small each community must be to still have recovery.

References

[1] FRONZETTI COLLADON, A., NALDI, M. "Distinctiveness centrality in social networks", *PLOS ONE*, v. 15, n. 5, pp. 1–21, 05 2020. doi: 10.1371/journal.pone.0233276. Disponível em: <https://doi.org/10.1371/journal.pone.0233276>.

[2] RAVASZ E, SOMERA AL, M. D. O. Z. B. A. "Hierarchical organization of modularity in metabolic networks", 2002. doi: 10.1126/science.1073374.

[3] TRAAG, V. A., WALTMAN, L., VAN ECK, N. J. "From Louvain to Leiden: guaranteeing well-connected communities", *Scientific Reports*, v. 9, n. 1, mar 2019. doi: 10.1038/s41598-019-41695-z. Disponível em: <https://doi.org/10.1038%2Fs41598-019-41695-z>.

[4] STANLEY, N., BONACCI, T., KWITT, R., et al. "Stochastic Block Models with Multiple Continuous Attributes". 2018.

[5] DESHPANDE, Y., MONTANARI, A., MOSSEL, E., et al. "Contextual Stochastic Block Models". 2018.

[6] KREEFT, J., PALHA, A., GERRITSMA, M. "Mimetic framework on curvilinear quadrilaterals of arbitrary order", 11 2011.

[7] AMARI, S.-I. *Information Geometry and Its Applications*. Springer Publishing Company, Incorporated, 2016. ISBN: 4431559779.

[8] BANERJEE, A., MERUGU, S., DHILLON, I. S., et al. "Clustering with Bregman Divergences", *Journal of Machine Learning Research*, v. 6, n. 58, pp. 1705–1749, 2005. Disponível em: <http://jmlr.org/papers/v6/banerjee05b.html>.

[9] THEODOROU, E., BUCHLI, J., SCHAAL, S. "Reinforcement learning of motor skills in high dimensions: A path integral approach". In: *2010 IEEE International Conference on Robotics and Automation*, pp. 2397–2403, 2010. doi: 10.1109/ROBOT.2010.5509336.

[10] CHENG, Y., WANG, X., MORAN, B. "Optimal Nonlinear Estimation in Statistical Manifolds with Application to Sensor Network Localization", *Entropy*, v. 19, n. 7, 2017. ISSN: 1099-4300. doi: 10.3390/e19070308. Disponível em: <https://www.mdpi.com/1099-4300/19/7/308>.

[11] HINO, H., AKAHO, S., MURATA, N. "Geometry of EM and related iterative algorithms", *Information Geometry*, 11 2022. doi: 10.1007/s41884-022-00080-y.

[12] GÖSGENS, M., TIKHONOV, A., PROKHORENKOVA, L. "Systematic Analysis of Cluster Similarity Indices: How to Validate Validation Measures". In: *International Conference on Machine Learning*, 2019. Disponível em: <https://api.semanticscholar.org/CorpusID:218630055>.

[13] HUBERT, L. J., . A. P. "Comparing partitions", *Journal of Classification*, 1985. doi: 10.1007/BF01908075.

[14] LONG, B., ZHANG, Z. M., YU, P. S. "A probabilistic framework for relational clustering". In: *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 470–479, 2007.

[15] MARIADASSOU, M., ROBIN, S., VACHER, C. "Uncovering latent structure in valued graphs: A variational approach", *The Annals of Applied Statistics*, v. 4, n. 2, pp. 715 – 742, 2010. doi: 10.1214/10-AOAS361. Disponível em: <https://doi.org/10.1214/10-AOAS361>.

[16] ZITNIK, M., LESKOVEC, J. "Predicting multicellular function through multi-layer tissue networks", *CoRR*, v. abs/1707.04638, 2017. Disponível em: <http://arxiv.org/abs/1707.04638>.

[17] WEBER, M., DOMENICONI, G., CHEN, J., et al. "Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics", *CoRR*, v. abs/1908.02591, 2019. Disponível em: <http://arxiv.org/abs/1908.02591>.

[18] BARABÁSI, A.-L., PÓSFAI, M. *Network science*. Cambridge, Cambridge University Press, 2016. ISBN: 9781107076266 1107076269. Disponível em: <http://barabasi.com/networksciencebook/>.

[19] NEWMAN, M. E. J. "Mixing patterns in networks", *Phys. Rev. E*, v. 67, pp. 026126, Feb 2003. doi: 10.1103/PhysRevE.67.026126. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevE.67.026126>.

[20] NEWMAN, M. E. J., GIRVAN, M. "Finding and evaluating community structure in networks", *Physical Review E*, v. 69, n. 2, Feb 2004. ISSN: 1550-2376. doi: 10.1103/physreve.69.026113. Disponível em: <http://dx.doi.org/10.1103/PhysRevE.69.026113>.

[21] GIRVAN, M., NEWMAN, M. E. J. "Community structure in social and biological networks", *Proceedings of the National Academy of Sciences*, v. 99, n. 12, pp. 7821–7826, 2002. ISSN: 0027-8424. doi: 10.1073/pnas.122653799. Disponível em: <https://www.pnas.org/content/99/12/7821>.

[22] GILBERT, E. N. "Random Graphs", *The Annals of Mathematical Statistics*, v. 30, n. 4, pp. 1141 – 1144, 1959. doi: 10.1214/aoms/1177706098. Disponível em: <https://doi.org/10.1214/aoms/1177706098>.

[23] ERDÖS, P., RÉNYI, A. "On Random Graphs I", *Publicationes Mathematicae Debrecen*, v. 6, pp. 290, 1959.

[24] NEWMAN, M. E. J. "Fast algorithm for detecting community structure in networks", *Physical Review E*, v. 69, n. 6, Jun 2004. ISSN: 1550-2376. doi: 10.1103/physreve.69.066133. Disponível em: <http://dx.doi.org/10.1103/PhysRevE.69.066133>.

[25] BLONDEL, V. D., GUILLAUME, J.-L., LAMBIOTTE, R., et al. "Fast unfolding of communities in large networks", *Journal of Statistical Mechanics: Theory and Experiment*, v. 2008, n. 10, pp. P10008, Oct 2008. ISSN: 1742-5468. doi: 10.1088/1742-5468/2008/10/p10008. Disponível em: <http://dx.doi.org/10.1088/1742-5468/2008/10/P10008>.

[26] "Uncovering the overlapping community structure of complex networks in nature and society", *Nature*, 2005. doi: https://doi.org/10.1038/nature03607.

[27] AHN, Y.-Y., BAGROW, J. P., LEHMANN, S. "Link communities reveal multiscale complexity in networks", *Nature*, v. 466, n. 7307, pp. 761–764, Jun 2010. ISSN: 1476-4687. doi: 10.1038/nature09182. Disponível em: <http://dx.doi.org/10.1038/nature09182>.

[28] NG, A., JORDAN, M., WEISS, Y. "On Spectral Clustering: Analysis and an algorithm". In: Dietterich, T., Becker, S., Ghahramani, Z. (Eds.), *Advances in Neural Information Processing Systems*, v. 14. MIT Press, 2002. Disponível em: <https://proceedings.neurips.cc/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf>.

[29] REYNOLDS, D. A. "Gaussian Mixture Models". In: *Encyclopedia of Biometrics*, 2009.

[30] ZHANG, P., KRZAKALA, F., REICHARDT, J., et al. "Comparative study for inference of hidden classes in stochastic block models", *Journal of Statistical Mechanics: Theory and Experiment*, v. 2012, n. 12, pp. P12021, dec 2012. doi: 10.1088/1742-5468/2012/12/P12021. Disponível em: <https://dx.doi.org/10.1088/1742-5468/2012/12/P12021>.

[31] PEIXOTO, T. P. "Efficient Monte Carlo and greedy heuristic for the inference of stochastic block models", *Phys. Rev. E*, v. 89, pp. 012804, Jan 2014. doi: 10.1103/PhysRevE.89.012804. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevE.89.012804>.

[32] DAUDIN, J.-J., PICARD, F., ROBIN, S. "A mixture model for random graphs", *Statistics and Computing*, v. 18, pp. 173–183, 2008. Disponível em: <https://api.semanticscholar.org/CorpusID:8272913>.

[33] CROUZEIX, J. P. "A Relationship between the Second Derivatives of a Convex Function and of Its Conjugate", *Math. Program.*, v. 13, n. 1, pp. 364–365, dec 1977. ISSN: 0025-5610. doi: 10.1007/BF01584350. Disponível em: <https://doi.org/10.1007/BF01584350>.

[34] FORSTER, J., WARMUTH, M. K. "Relative Expected Instantaneous Loss Bounds". In: *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, COLT '00, p. 90–99, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN: 155860703X.

[35] FUJIMOTO, Y., MURATA, N. "A modified EM algorithm for mixture models based on Bregman divergence", *Annals of the Institute of Statistical Mathematics*, v. 59, pp. 3–25, 02 2007. doi: 10.1007/s10463-006-0097-x.

[36] BANERJEE, A., GUO, X., WANG, H. "On the optimality of conditional expectation as a Bregman predictor", *IEEE Transactions on Information Theory*, v. 51, n. 7, pp. 2664–2669, 2005. doi: 10.1109/TIT.2005.850145.

[37] HOFFMAN, M., BLEI, D. M., WANG, C., et al. "Stochastic Variational Inference". 2013.

[38] CHERNOFF, H. "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations", *The Annals of Mathematical Statistics*, pp. 493–507, 1952.

[39] VAN ERVEN, T., HARREMOS, P. "Rényi divergence and Kullback-Leibler divergence", *IEEE Transactions on Information Theory*, v. 60, n. 7, pp. 3797–3820, 2014.

[40] DREVETON, M., FERNANDES, F. S., FIGUEIREDO, D. R. "Exact recovery and Bregman hard clustering of node-attributed Stochastic Block Model". In: *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. Disponível em: <https://openreview.net/forum?id=TjJJmcHw9p>.

[41] ABBE, E., SANDON, C. "Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery". In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pp. 670–688, Los Alamitos, CA, USA, 2015. IEEE Computer Society.

[42] BRAUN, G., TYAGI, H., BIERNACKI, C. "An iterative clustering algorithm for the Contextual Stochastic Block Model with optimality guarantees". In: *International Conference on Machine Learning*, pp. 2257–2291. PMLR, 2022.

[43] SAAD, H., NOSRATINIA, A. "Community detection with side information: Exact recovery under the stochastic block model", *IEEE Journal of Selected Topics in Signal Processing*, v. 12, n. 5, pp. 944–958, 2018.

[44] BANERJEE, A., MERUGU, S., DHILLON, I. S., et al. "Clustering with Bregman divergences." *Journal of machine learning research*, v. 6, n. 10, 2005.

[45] NIELSEN, F. "Chernoff information of exponential families", *arXiv preprint arXiv:1102.2684*, 2011.

[46] DAMLE, A., MINDEN, V., YING, L. "Simple, direct and efficient multi-way spectral clustering", *Information and Inference: A Journal of the IMA*, v. 8, n. 1, pp. 181–203, 06 2018. ISSN: 2049-8772. doi: 10.1093/imaiai/iay008. Disponível em: <https://doi.org/10.1093/imaiai/iay008>.

[47] JERDEE, M., KIRKLEY, A., NEWMAN, M. E. J. "Normalized mutual information is a biased measure for classification and community detection". 2023.

[48] VINH, N. X., EPPS, J., BAILEY, J. "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance", *Journal of Machine Learning Research*, v. 11, n. 95, pp. 2837–2854, 2010. Disponível em: <http://jmlr.org/papers/v11/vinh10a.html>.

[49] NEWMAN, M. E. J. "Modularity and community structure in networks", *Proceedings of the National Academy of Sciences*, v. 103, n. 23, pp. 8577–8582, jun 2006. doi: 10.1073/pnas.0601602103. Disponível em: <https://doi.org/10.1073%2Fpnas.0601602103>.

[50] ROSS, H. H. "Principles of Numerical Taxonomy", *Systematic Biology*, v. 13, n. 1-4, pp. 106–108, 03 1964. ISSN: 1063-5157. doi: 10.2307/sysbio/13.1-4.106. Disponível em: <https://doi.org/10.2307/sysbio/13.1-4.106>.

[51] LEI, Y., BEZDEK, J. C., ROMANO, S., et al. "Ground Truth Bias in External Cluster Validity Indices". 2016.

[52] VAN DER MAATEN, L., HINTON, G. "Visualizing Data using t-SNE", *Journal of Machine Learning Research*, v. 9, n. 86, pp. 2579–2605, 2008. Disponível em: <http://jmlr.org/papers/v9/vandermaaten08a.html>.

[53] LANCICHINETTI, A., FORTUNATO, S., RADICCHI, F. "Benchmark graphs for testing community detection algorithms", *Physical Review E*, v. 78, n. 4, oct 2008. doi: 10.1103/physreve.78.046110. Disponível em: <https://doi.org/10.1103%2Fphysreve.78.046110>.

[54] KAMIŃSKI, B., PRAŁAT, P., THÉBERGE, F. "Artificial Benchmark for Community Detection (ABCD)—Fast random graph model with community structure", *Network Science*, v. 9, n. 2, pp. 153–178, 2021. doi: 10.1017/nws.2020.45.