Power Spectral Clustering

Aditya S Challa, Sravan Danda, B S S Daya Sagar, Laurent Najman

Noname manuscript No.
(will be inserted by the editor)

Power Spectral Clustering

Aditya Challa · Sravan Danda · B. S. Daya Sagar · Laurent Najman

Received: date / Accepted: date

Abstract Spectral clustering is one of the most important image processing tools, especially for image segmentation. This specializes at taking local information such as edge weights and globalizing them. Due to its unsupervised nature, it is widely applicable. However, traditional spectral clustering is $\mathcal{O}(n^{3/2})$. This poses a challenge, especially given the recent trend of large datasets. In this article, we propose an algorithm by using ideas from $\Gamma-$convergence, which is an amalgamation of Maximum Spanning Tree (MST) clustering and spectral clustering. This algorithm scales as $\mathcal{O}(nlog(n))$ under certain conditions, while producing solutions which are similar to that of spectral clustering. Several toy examples are used to illustrate the similarities and differences. To validate the proposed algorithm, a recent state-of-the-art technique for segmentation - multiscale combinatorial grouping is used, where the normalized cut is replaced with the proposed algorithm and results are analyzed.

Keywords Spectral clustering, $\Gamma-$convergence, MST-based clustering, Multiscale combinatorial grouping, Image segmentation.

Aditya Challa,
Systems Science and Informatics Unit, Indian Statistical Institute, 8th Mile Mysore Road, Bangalore, India.
E-mail: aditya.challa.20@gmail.com

Sravan Danda
Systems Science and Informatics Unit, Indian Statistical Institute, 8th Mile Mysore Road, Bangalore, India.
E-mail: sravan8809@gmail.com

B. S. Daya Sagar
Systems Science and Informatics Unit, Indian Statistical Institute, 8th Mile Mysore Road, Bangalore, India.
E-mail: bsdsagar@yahoo.co.uk,bsdsagar@isibang.ac.in

Laurent Najman
Université Paris-Est, LIGM, Equipe A3SI, ESIEE, France.
E-mail: laurent.najman@esiee.fr

1 Introduction

Spectral clustering has been widely popular due to its usage in image segmentation [32]. It plays an important role in globalizing local information in the recent state-of-the-art method for segmentation - multiscale combinatorial grouping [30]. Although convolution neural networks form the current state-of-the-art for image segmentation [26], this can be attributed to the availability of huge labelled datasets. There exists domains where data is not easy to obtain, such as hyperspectral image datasets, where unsupervised techniques can be very useful. In methods such as those described in [35], even after using convolution neural nets, spectral clustering is used as the last step for segmentation.

An overview of spectral clustering procedures can be found in [38]. One of the reasons for the popularity of the spectral clustering procedure is its ability to detect non-convex clusters. Spectral clustering method proceeds by constructing an edge-weighted graph from the data and use the eigenvectors of the Laplacian of the graph. However, the calculation of the eigenvectors is computationally expensive, and is prone to errors. Several efforts were made to increase the speed and accuracy of the spectral clustering methods [42, 10, 17, 18], including parallelizing [34]. In [30], the authors propose to reduce the size of the similarity matrix to speed up the normalized cuts procedure for image segmentation [32].

Another clustering method which allows to detect non-convex clusters is that of *Maximum Spanning Tree (MST) based clustering* [43]. This method proceeds by constructing an MST on the edge-weighted graph and by removing the edges with least similarity. The main advantage of this method is that it is fast and scales well for big data. However, these methods also have some

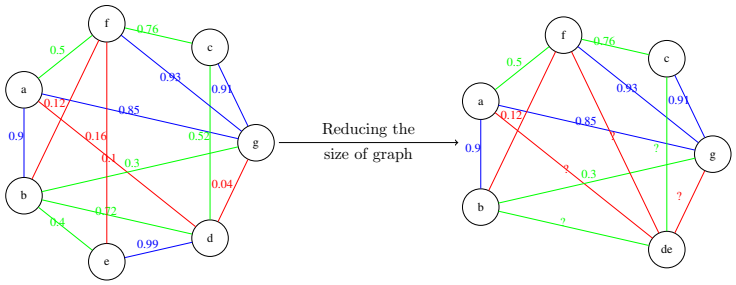


Fig. 1 Figure illustrating the motivation of the article. On the left we have an example graph with the edge weights denoting the similarities. The objective is to perform clustering such as spectral clustering. Intuitively, it is easy to infer that nodes d and e would belong to the same cluster as they have the similarity of 0.99. Hence, there is a scope to reduce the computations required. However, it is not obvious to decide on which similarity values should be considered high enough so as to merge the corresponding incident nodes. Moreover, assume we have indeed reduced the graph size by merging nodes, as in figure on the right. It is not clear what the edge weights in the contracted graph would be? These questions are answered using the power watershed framework.

ambiguity in clusters and results in degenerate solutions. (MST-based clustering is discussed in detail in section 2.)

In this article, the aim is to develop an algorithm which uses efficient MST-based clustering within a cluster and more computationally expensive spectral clustering near the borders of the clusters. This is achieved by computing the limit of minimizers (a.k.a $\Gamma-$limit) of spectral clustering. We obtain a faster version of spectral clustering while preserving the properties of spectral clustering, referred to as *power spectral clustering* or *Power Ratio cut* or more simply *PRcut*. The study of $\Gamma-$limits is referred to as the $\Gamma-$convergence. $\Gamma-$convergence is widely used in the areas of computer vision and in the field of calculus of variations (See chapter 5 of [6]). In [11], the authors proposed *Power Watershed* - $\Gamma-$limit of the energy function in [33] and demonstrated its similarity with watershed transformation [12,13]. It turns out that the Power Watershed performs better in several cases compared to the frequently used seeded segmentations. In [27], the power watershed framework was extended and a simple generic algorithm was proposed to calculate the limit of minimizers under some conditions. Other applications of the power watershed framework can be found in [16, 15,40]. The methods described in [16,15,40] use power watershed framework to reduce the size of the graph. In this article, we propose to use power watershed framework for spectral clustering - by computing a $\Gamma-$limit of spectral clustering. This article is a significant extension of the conference version [8]. The following are main contributions of this article -

1. A novel algorithm for clustering, *PRcut*, is proposed. Using power watershed framework, it is shown that PRcut is faster than spectral clustering while giving similar results. Also, efficient algorithm for implementing PRcut is proposed.
2. Several properties of PRcut are exhibited by using toy examples and compared with MST-based clustering and spectral clustering.
3. To conclusively show that PRcut can be used in place of spectral clustering, application to multiscale combinatorial grouping (MCG)[30] is demonstrated. Specifically, the spectral clustering step in MCG is replaced with the proposed method and empirically shown that PRcut is faster while preserving the overall accuracy of the method.
4. The concept of *discretization of the weights* is introduced. This adds further flexibility to the PRcut, in the sense that in the base case PRcut is equivalent to spectral clustering. Thus, PRcut can be considered an extension of spectral clustering as well.
5. Applications to hyperspectral images and high dimensional datasets are also discussed.

The outline of the article is as follows - In section 2, various concepts required for the rest of the article - spectral clustering, MST clustering and $\Gamma-$convergence, are introduced. In section 3, we start with a generic algorithm to calculate a $\Gamma-$limit and characterize different parts of the algorithm to obtain an implementable version. In section 4, we further increase the efficiency by identifying the relation between eigenvectors and connected components of the Laplacian. In section 5, we explore the PRcut in detail. We compare PRcut with MST-based clustering and spectral clustering using several toy examples to illustrate the differences and similarities. To validate the claim that PRcut is similar to classic spectral clustering, we use multiscale combinatorial grouping (MCG). MCG [30] was a state-of-the-art algorithm for segmentation which used spectral clustering as a step in the pipeline. We replace this step with

the PRcut algorithm and compare the results. It can be deduced that the PRcut algorithm is faster than spectral clustering while giving similar results. Also shown are applications to hyperspectral image data.

Gist of the article: As stated earlier, in this article we develop an algorithm which combines the efficiency of MST-based clustering within a cluster and accuracy of spectral clustering at the boundaries. This is achieved by considering the $\Gamma-$limit of the spectral clustering, To compute the said $\Gamma-$limit we use the power watershed framework developed in [27]. The solution thus obtained is referred to as power spectral clustering or PRcut.

To understand the redundancies Power Spectral Clustering exploits, consider a simple graph as shown in figure 1. The edge weights shown indicate the similarities between the end points of the edge. Intuitively, it is clear that computations are not required to infer that the nodes d and e would belong to the same cluster since they have a similarity of 0.99. Similarly, e and f does not belong to the same cluster since their similarity is 0.1.

However, it is not clear as to where to make a clear distinction? That is, picking a threshold. Moreover, assume we have contracted the edge (d, e) considering a single node for d and e, as shown in figure 1. It is not clear what the new edge weights would be? These questions are implicitly answered when one considers the limit of minimizers, which is obtained using the power watershed framework. As it would be illustrated in this article, the power watershed limit results in algorithm 4.

2 Background

The main mathematical structure used in this article is that of an edge-weighted graph - $\mathcal{G} = (V, E, W)$. Given an image I, the set of vertices is taken as the pixels in I, the set of edges is given by the 4-adjacency relation. The edge weights $W : E \to \mathbb{R}^+$ are constructed by using the pixel values. The edge weights are assumed to reflect the similarity between two pixels/points.

Note that both spectral clustering and MST clustering operate on edge-weighted graphs. Thus, these methods can be used on any data which can be represented as edge-weighted graphs. For data in general euclidean space, k-nearest neighbors graph is used [38]. In this article, small toy examples of general data are also used for better analysis.

D is a diagonal matrix, $diag(d_1, d_2, \cdots, d_n)$, such that

$$d_i = \sum_j w_{ij} \tag{1}$$

The *Laplacian* of a graph is defined by

$$L = D - W \tag{2}$$

We know that the Laplacian is a symmetric positive-semi-definite matrix, and hence has non-negative real eigenvalues. The eigenvalues are represented by $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. The corresponding eigenvectors are denoted by $\{e_1, e_2, \cdots, e_n\}$. Let $A \subseteq V$. Denote

$$\mathbf{1}_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}$$

$\mathcal{G}_{\geq w}$ denotes the *thresholded* graph, with vertex set V and edge set, $E_{\geq w}$, consisting of only those edges whose weights are greater than or equal to w.

2.1 Spectral Clustering

Spectral clustering methods work by projecting the data onto a subspace, so that similar points are close by and dissimilar points are far apart in the projected subspace. There are 3 steps which form the core of spectral clustering methods-

1. Given a set of points $\{x_i\}$ (dataset), construct a graph, $\mathcal{G}$, with each data point as a vertex.
2. Construct the Laplacian for the obtained graph and calculate the first m eigenvectors of the Laplacian. The value of m is fixed based on the number of clusters required. Let H be the matrix such that the i^{th} column of H is the i^{th} eigenvector e_i.
3. Using rows of the matrix H as new representation of the points x_i, use classical clustering methods such as k-means to obtain the final clusters.

The Laplacian in (2) is known as an unnormalized Laplacian. Some works also consider the normalized Laplacians, L_1, L_2 as well [32, 29], where,

$$L_1 = \mathbf{I} - D^{-1}W; \qquad L_2 = \mathbf{I} - D^{-1/2}WD^{-1/2} \tag{3}$$

Why does spectral clustering work? The idea behind working of spectral clustering methods can be intuitively understood using two results - (i) Suppose the graph $\mathcal{G}$ has m connected components, $\{A_1, A_2, \cdots, A_m\}$. Then the basis of the vector space spanned by the first m eigenvectors is $\{\mathbf{1}_{A_1}, \mathbf{1}_{A_2}, \cdots, \mathbf{1}_{A_m}\}$ and (ii) The eigenspaces of a matrix and its perturbation are 'close' [38]. Assume that the similarity between points from two different clusters is small, and similarity between points from same cluster is high. Then, from the above results, the first few eigenvectors of the Laplacian are close to the indicator of the clusters. Hence simple clustering methods work well on this projected data.

2.1.1 Different formulations of spectral clustering

The spectral clustering can also be analyzed in an optimization framework. Given a similarity graph $\mathcal{G} = (V, E, W)$, consider the *graph cut* measure

$$cut(A_1, A_2, \cdots, A_m) = \frac{1}{2}\sum_{i=1}^{m} W(A_i, \overline{A_i}) \tag{4}$$

where

$$W(A, B) = \sum_{i \in A; j \in B} w_{ij} \tag{5}$$

$\overline{A}$ denotes the complement of A in the vertex set V. m denotes the size of the partition required which is equal to the number of clusters. Note that $cut(.,.)$ measures how similar the clusters are by taking the sum of the weights of the edges connecting distinct clusters, and hence can be used to partition a graph. In practice, however, minimizing the $cut(.,.)$ does not give good results, since it generally separates one vertex, and gives degenerate solutions. Also, minimizing $cut(.,.)$ for $m \geq 3$ is NP-hard [36]. Thus, it was proposed to use a variation of the above cost function. *Ratio cut* [38] is given by

$$Rcut(A_1, A_2, \cdots, A_m) = \frac{1}{2}\sum_{i=1}^{m} \frac{W(A_i, \overline{A_i})}{|A_i|} \tag{6}$$

Here $|A_i|$ is the cardinality of set A_i. Note that ratio cut penalizes small clusters, and hence avoids degenerate solutions. Let $\mathcal{G}$ be a graph constructed from the data set, and let L denote its corresponding Laplacian. It is shown that minimizing the $Rcut(.,.)$ for m clusters is approximately equivalent to solving the optimization problem in (7) [38].

$$\begin{aligned} \underset{H \in \mathbb{R}^{n \times m}}{\text{minimize}} \quad & Tr(H^t L H) \\ \text{subject to} \quad & H^t H = \mathbf{I} \end{aligned} \tag{7}$$

where, $\mathbf{I}$ is the identity matrix. From the *Rayleigh-Ritz* theorem, it is known that the solution to optimization problem in (7) is obtained by considering the first m eigenvectors of L as columns of H [25].

Another variation of the $cut(.,.)$ cost function, proposed in [32] is the *Normalized cut* (Ncut) cost function given by

$$Ncut(A_1, A_2, \cdots, A_m) = \frac{1}{2}\sum_{i=1}^{m} \frac{W(A_i, \overline{A_i})}{vol(A_i)} \tag{8}$$

where $vol(A_i) = \sum_i d_i$. It can be shown that for m clusters, $Ncut(.,.)$ is approximately equivalent to the solution of the following optimization problem[38].

$$\begin{aligned} \underset{H}{\text{minimize}} \quad & Tr(H^t L H) \\ \text{subject to} \quad & H^t D H = \mathbf{I} \end{aligned} \tag{9}$$

It is known that the solution to this optimization problem is obtained by taking the first m eigenvectors of $\mathbf{I} - D^{-1}W$ as columns of H [25].

2.2 Maximum Spanning Tree Clustering

Another method of clustering which can detect non globular clusters in the data is *maximum spanning tree based clustering*. Let $\mathcal{G} = (V, E, W)$ be a connected similarity graph. A *spanning tree*, T, is a connected acyclic subgraph of $\mathcal{G}$ whose vertex set is V. Each spanning tree can be assigned a numerical value by taking the sum of the weights of the edges in the tree,

$$w(T) = \sum_{e \in E(T)} w(e) \tag{10}$$

A spanning tree with the maximum weight is known as the Maximum Spanning Tree (MST). There are several clustering methods based on MST [43]. Although MST-based clustering is not so prevalent in clustering general data, in the context of image segmentation, slight variations of MST-based clustering was shown to be very useful, thanks to its equivalence to watersheds (as described in [12,13]). A generic MST-based clustering algorithm is

1. Given a similarity graph $\mathcal{G} = (V, E, W)$, construct an MST, T, and sort the edges of T according to the weights.
2. If the required number of clusters is equal to m, add edges starting from highest weight until the number of components m are reached. The ties are broken arbitrarily.

MST-based clustering is categorized under graph based clustering, and is related to the hierarchical approaches. In fact, MST-based clustering as described above is equivalent to single link hierarchical clustering.

The main problems with MST-based clustering are - (i) It is prone to noise and outliers, (ii) In practice, it gives small clusters or degenerate solutions which are not so meaningful, and (iii) Arbitrariness in breaking of ties results in non uniform solutions. In the later sections it will be seen that the proposed method, PRcut, is similar to MST-based clustering and does not suffer from these problems.

2.3 Γ-convergence and Power Watershed Framework

Let $\min\{F_i(x) : x \in X\}$ be a sequence of minimum problems. A question of interest is the limiting behavior of the minimizers of this family as $i \to \infty$. Ideally this would be substituted by a single minimum problem

$\min\{F(x) : x \in X\}$ which captures the limiting behavior. Few of the advantages of such a substitution are - 1) This gives an approximate solution to the family of minimizers which are much harder to calculate than the limit, 2) Dependence on a parameter is nullified, and 3) This results in a new method which would present a different model which was previously modeled with F_i. The theory of $\Gamma-$convergence focusses on understanding the conditions under which such a substitution is possible [6].

In this article, however, we are interested in calculating a $\Gamma-$limit of the spectral clustering discussed above. Recently the authors in [11] calculated $\Gamma-$limit of the seeded random walker cost function [33], and proposed *Power watershed* seeded segmentation. This $\Gamma-$limit involves an MST, and hence links the random walk segmentation with MST clustering. In [27], a generic algorithm was proposed to calculate a $\Gamma-$limit, which we review here.

Let $0 < \alpha_1 < \alpha_2 < \cdots < \alpha_k \leq 1$ and $Q_i(x)$ is continuous for all i. Consider the cost function

$$Q^{(p)}(x) = \sum_{i=0}^{k} \alpha_i^p Q_i(x) \tag{11}$$

The problem is to calculate the limit of minimizers of $Q^{(p)}(x)$ as $p \to \infty$. Observe that the function itself converges to 0 at every point if $\alpha_k < 1$, and hence minimizers of the limit would be the whole space. Let C be a compact set. For the sake of simplicity, we assume that we are interested in finding the solutions in C. Define

$$M_k = \arg\min_{x \in C} Q_k(x) \tag{12}$$

i.e, M_k is the set of minimizers for Q_k. Now recursively define, for $i = k-1, \cdots, 1$.

$$M_i = \arg\min_{x \in M_{i+1}} Q_i(x) \tag{13}$$

Observe that we have the following relation between M_i, $i = 1, \cdots, k$.

$$M_i \subseteq M_j \qquad \text{for all } 1 \leq i < j \leq k \tag{14}$$

Theorem 1 ([27]) *Let X^* be the union of the sets of minimizers of $Q^{(p)}$ (as defined in (11)) for all p. Then every limit point of X^* belongs to the set M_1.*

The above theorem can be interpreted in terms of *scale*. If one interprets each α_i as a scale, then theorem 1 states - $\Gamma-$limit (limit of minimizers) belongs to the set of solutions which minimizes all the cost functions at different scales α_i starting from the largest scale α_k. The main consequence of theorem 1 is that one can now formulate algorithm 1 to calculate a $\Gamma-$limit [27].

Algorithm 1 Generic Algorithm to Compute $\Gamma-$limit.

Input: Function $Q^{(p)}(x) = \sum_{i=1}^{k} \alpha_i^p Q_i(x)$, where $0 \leq \alpha_1 < \alpha_2 < \cdots < \alpha_k \leq 1$.
Output: M_1
1: $M_k = \arg\min Q_k(x)$ where $x \in C$
2: **for** i from $k-1$ to 1 **do**
3: Compute $M_i = \arg\min Q_i(x)$ where $x \in M_{i+1}$
4: **end for**

Theorem 1 ensures that a $\Gamma-$limit belongs to the output of algorithm 1. The converse is not true in general, i.e, all solutions obtained from algorithm 1 need not be a $\Gamma-$limit. However, it can be shown that they are 'equivalent', in the sense that the value of the cost function is same for all p, and hence in the limit as well.

Proposition 1 *Let x^* be a $\Gamma-$limit for $Q^{(p)}$. Let $\widehat{x}$ be a solution obtained from algorithm 1. Then we have that, for all p*

$$Q^{(p)}(x^*) = Q^{(p)}(\widehat{x}) \tag{15}$$

The proof of the proposition 1 is straightforward. Although algorithm 1 might not calculate a $\Gamma-$limit, we have that it calculates equivalent solution as far as the cost function is concerned.

In the special case when $Q(x)$ is taken to be the ratio cut cost function, we have the converse as well. This is described in theorem 2.

To summarize, we have computed the $\Gamma-$limit for the cost functions of the form in (11). This resulted in a generic algorithm 1. This is referred to as *Power Watershed framework* in the rest of this article. The limit obtained by using the algorithm 1 is also referred to as *Power Watershed limit*. Detailed analysis of the Power Watershed and its relation to existing works can be found in [27].

3 $\Gamma-$limit of spectral clustering

In several cases discrete models are used for modeling the data, while the data might be continuous. This is especially true in the case of graph-based models [11]. Thus a *discretization scheme* is required for using discrete models in real data. We introduce the definition of a discretization scheme which is then used for calculating a $\Gamma-$limit.

Let $\mathcal{G} = (V, E, W)$ be a graph. Recall that $W : E \to \mathbb{R}^+$.

Definition 1 (Discretization Scheme) A discretization scheme is a decomposition of W, i.e for all edges e, we have

$$W(e) = \widehat{W}(e)\omega(e) \tag{16}$$

where $\widehat{W} : E \to \mathbb{Z}^+$ and $\omega : E \to \mathbb{R}^+$, with a condition that if $W(e_1) < W(e_2)$, then $\widehat{W}(e_1) < \widehat{W}(e_2)$.

An example of a discretization scheme is given by taking the least integer function.

$$W(e) = \lfloor W(e) \rfloor \frac{W(e)}{\lfloor W(e) \rfloor} = \widehat{W}(e)\omega(e) \tag{17}$$

Here $\widehat{W}(e) = \lfloor W(e) \rfloor$ and $\omega(e) = W(e)/\lfloor W(e) \rfloor$. Intuitively, $\widehat{W}$ is the discrete version of W and ω is error with respect to the discretization. It is also clear that there are several discretization schemes possible.

As another example, one can consider the following decomposition of the weights

$$W(e) = \lfloor cW(e) \rfloor \frac{W(e)}{\lfloor cW(e) \rfloor} = \widehat{W}(e)\omega(e) \tag{18}$$

where c is an arbitrary positive constant.

An *exponentiated graph* is denoted by

$$\mathcal{G}^{(p)} = (V, E, W^{(p)}) \tag{19}$$

where $W^{(p)}(e) = (\widehat{W}(e))^p \omega(e)$. Accordingly we can define $D^{(p)}$, a diagonal matrix,

$$[D^{(p)}]_{ii} = \sum_j W_{ij}^{(p)} \tag{20}$$

and a Laplacian $L^{(p)} = D^{(p)} - W^{(p)}$.

We define a $\Gamma-$limit of spectral clustering by the limit of minimizers of the optimization problem (21) as $p \to \infty$.

$$\begin{aligned} \underset{H \in \mathbb{R}^{n \times m}}{\text{minimize}} \quad & Tr(H^t L^{(p)} H) \\ \text{subject to} \quad & H^t H = \mathbf{I} \end{aligned} \tag{21}$$

In the case of the optimization problem theorem (21) as above, teh converse of 1 holds. This is summarized in the following theorem.

Theorem 2 *Given the notation as in theorem 1, and considering the $Q^{(p)}$ to be the optimization problem in* (21)*, we have that any point in M_1 is a limit point of X^*.*

The proof of the above theorem is given in the appendix.

Since the datasets are finite, $\widehat{W}$ takes only finite number of values. Also, since the optimization problem does not change with positive scalar multiples, one can assume without loss of generality that the distinct values may be denoted by $0 < w_1 < w_2 < \cdots < w_k < 1$, indicating the k distinct weights $\widehat{W}(e)$ can take. Denote by $\mathcal{G}_i$ the graph with the vertex set, V, and the edge set consisting of only those edges e such that $\widehat{W}(e) = w_i$. The weight of an edge e in $\mathcal{G}_i$ is given by the function $\omega(e)$. The Laplacian for the graph $\mathcal{G}_i$ is denoted by L_i. Then it is easy to see that

$$L = \sum_{i=1}^{k} w_i L_i \tag{22}$$

and hence,

$$Tr(H^t L H) = \sum_{i=1}^{k} w_i Tr(H^t L_i H) \tag{23}$$

Accordingly, for the exponentiated graphs we have

$$Tr(H^t L^{(p)} H) = \sum_{i=1}^{k} w_i^p Tr(H^t L_i H) \tag{24}$$

Note that (24) is in the form of (11) and hence we can use the generic algorithm 1 to calculate a $\Gamma-$limit.

Let $\overline{M}$ denote the set of all possible m dimensional subspaces of $\mathbb{R}^n$. Each subspace in $\overline{M}$ can be associated with an $n \times m$ matrix H, such that the column space of H is the subspace. Note that this matrix is not unique. Also, let $\mathcal{H}(M)$ denote the set of all matrices whose column space is equivalent to any of the subspaces in the set M. Let $\mathcal{M}(H)$ denote the set of all column spaces of matrices in a set of matrices H. It is easy to see that the following relations hold true.

$$\begin{aligned} &\mathcal{M}(\mathcal{H}(M')) = M' \\ &\mathcal{H}(\mathcal{M}(H')) \supseteq H' \end{aligned}$$

where M' is a set of subspaces and H' is a set of matrices. With this notation, the generic algorithm 1 in the case of ratio cut optimization would be the one in algorithm 2. However, this algorithm is not implementable. One needs to characterize all the solutions to the optimization problem in (7) to obtain an implementable version. This will in turn characterize the M_i and $\mathcal{H}(M_i)$ at every stage i of the algorithm 2.

Algorithm 2 Generic Algorithm to Compute $\Gamma-$limit.

Input: A weighted graph, $\mathcal{G}$, with distinct weights $w_1 < w_2 < \cdots < w_k$. Number of clusters to calculate m.
Output: M_0
1: Set $i = k$, $M_i = \overline{M}$
2: Construct the graph $\mathcal{G}_i$ at level i, and Laplacian L_i.
3: Solve the optimization problem

$$\begin{aligned} \underset{H\in\mathbb{R}^{n\times m}}{\text{minimize}} \quad & Tr(H^t L_i H) \\ \text{subject to} \quad & H^t H = \mathbf{I} \\ & H \in \mathcal{H}(M_i) \end{aligned} \tag{25}$$

4: Let H_i be the set of possible minimizers of the above optimization problem. Set $M_{i-1} = \mathcal{M}(H_i)$.
5: Set i = i - 1
6: **if** $i = 0$ **then**
7: Stop.
8: **return** M_0
9: **else**
10: Goto Step (2)
11: **end if**

3.1 Solutions to ratio cut optimization problem

Given a Laplacian L of dimensions $n \times n$, let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ denote the eigenvalues and $\{e_1, e_2, \cdots, e_n\}$ denote the corresponding eigenvectors. Let $\lambda_{(m)}$ denote the m^{th} smallest eigenvalue. Let A be the matrix obtained by stacking all the eigenvectors of L whose eigenvalue is less than or equal to $\lambda_{(m)}$. Assuming that there are l such eigenvectors, the dimension of A would be $n \times l$. Here m is the number of clusters required. Let l_2 be the number of eigenvectors whose eigenvalue is exactly equal to $\lambda_{(m)}$, and l_1 be the number of remaining vectors. We then have $l_1 + l_2 = l$. Let K be the matrix

$$K = \begin{bmatrix} \mathbf{I}_{l_1\times l_1} & 0 \\ 0 & X_{l_2\times(m-l_1)} \end{bmatrix} \tag{26}$$

where X is any matrix such that $K^t K = \mathbf{I}$. Let Y be any orthogonal matrix. Theorem 3 characterizes the sets $\mathcal{H}(M_i)$ and M_i at every stage.

Theorem 3 *The set of all solutions to the optimization problem* (7) *is the set of all solutions of the form* AKY.

Now, starting at highest level k (edge set consisting only of edges with weight w_k) we have $M_k = \overline{M}$. From theorem 3, we know that all the solutions to the optimization problem (25) for $i = k$ is of the form $A_k KY$, where A_k is the matrix obtained by stacking the eigenvectors for L_k whose eigenvalue is less than or equal to the m^{th} smallest eigenvalue of L_k. At level $k-1$, we have $M_{k-1} = \mathcal{M}(A_k KY)$. Now, we have the following lemma.

Lemma 1 *Given the notation as above, we have*

$$\mathcal{H}(\mathcal{M}(A_k KY)) = \{\textit{matrices of the form } A_k KY\} \tag{27}$$

Thus at level $k-1$ we need to solve the optimization problem,

$$\begin{aligned} \underset{H\in\mathbb{R}^{n\times m}}{\text{minimize}} \quad & Tr(H^t L_{k-1} H) \\ \text{subject to} \quad & H^t H = \mathbf{I} \\ & H \sim A_k KY \end{aligned}$$

($H \sim A_k KY$ is read H is of the form $A_k KY$) which is equivalent to solving the optimization problem

$$\begin{aligned} \underset{K}{\text{minimize}} \quad & Tr(Y^t K^t A_k^t L_{k-1} A_k KY) \\ \text{subject to} \quad & K^t K = \mathbf{I} \end{aligned}$$

Now, since Y is orthogonal, we have

$$Tr(Y^t K^t A_k^t L_{k-1} A_k KY) = Tr(K^t A_k^t L_{k-1} A_k K)$$

Noting the special form of K as in (26), solving the above optimization problem is equivalent to solving the optimization problem

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & Tr(X^t [A_k^t L_{k-1} A_k]_{l_2\times l_2} X) \\ \text{subject to} \quad & X^t X = \mathbf{I} \end{aligned}$$

where, $[A_k^t L_{k-1} A_k]_{l_2\times l_2}$ is the lower-right $l_2 \times l_2$ block. Here l_2 indicates the number of times the eigenvalue $\lambda_{(m)}$ repeats in L_k. Also matrix $[A_k^t L_{k-1} A_k]_{l_2\times l_2}$ is symmetric positive semi-definite. Hence theorem 3 applies. Let $\widehat{A}_{k-1}$ the matrix obtained by stacking the eigenvectors of $[A_k^t L_{k-1} A_k]_{l_2\times l_2}$ whose eigenvalue is less than or equal to the m^{th} smallest eigenvalue of $[A_k^t L_{k-1} A_k]_{l_2\times l_2}$. Then all the solutions are of the form $\widehat{A}_{k-1} KY$. Let

$$A_{k-1} = \begin{bmatrix} \mathbf{I}_{l_1\times l_1} & 0 \\ 0 & \widehat{A}_{k-1} \end{bmatrix} \tag{28}$$

Then, we have that H_{k-1} is the set of all matrices which are of the form $A_k A_{k-1} KY$. This follows from the fact that the matrices K and Y are arbitrary under the constraint that $K^t K = \mathbf{I}$ and Y is some orthogonal matrix. Repeating this procedure, one can can obtain an algorithm to calculate a $\Gamma-$limit. This is summarized in algorithm 3.

Algorithm 3 Algorithm to Compute $\Gamma-$limit for relaxed Ratio-cut.

Input: A weighted (similarity) graph, $\mathcal{G}$, with distinct weights $w_1 < w_2 < \cdots < w_k$. Number of clusters, m.
Output: N_1

1: Set $i = k$, $N_i = \mathbf{I}_n$, $l_1 = 0$, $l_2 = n$ {l_2 indicates the number of eigenvectors at the end whose eigenvalue is the same.}
2: Construct the graph $\mathcal{G}_i$ at level i, and Laplacian L_i.
3: Construct matrix $C = [N_i^t L_i N_i]_{l_2, l_2}$
4: Calculate the first eigenvectors of the generalized eigenvalue problem

$$Cx = \lambda x \tag{29}$$

such that the eigenvalue is less than or equal to $\lambda^* = \lambda_{(m-l_1)}$. Let A be the matrix with the eigenvectors computed, stacked as columns.
5: Construct $\widehat{A}$ as

$$\widehat{A} = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & A \end{bmatrix}$$

6: Set $N_{i-1} = N_i \widehat{A}$.
7: Set l_1 to be the number of eigenvectors with eigenvalue is strictly less than λ^*.
8: Set l_2 to be the number of eigenvectors with eigenvalue is strictly equal λ^*.
9: Set $i = i - 1$
10: **if** $i = 0$ **then**
11: **return** N_1
12: **else**
13: Goto Step (3)
14: **end if**

A heuristic explanation of algorithm 3 is - One starts with a trivial representation of the points, $\mathbf{I}_n$ and at each stage, iteratively refines the representation based on the solutions of the optimization problem at that stage.

4 Implementation

Although algorithm 3 is implementable, note that every stage of the algorithm involves several matrix manipulations and eigenvector calculations. Since eigenvector calculations are not so robust, this algorithm does not work well in practice. Algorithm 4 provides an efficient alternative which is equivalent to algorithm 3.

Recall that a connected component of a graph $\mathcal{G}$ is the maximal subgraph of $\mathcal{G}$ which is connected. Note that in the first iteration of algorithm 3 the eigenvectors in step 4 are the indicators of the connected components. This property holds true for next iterations until the number of components are greater than the required number of clusters. This is formalized in proposition 2.

Proposition 2 *Let $\mathcal{G}$ be a similarity graph. At a given level j, let $\{C_1, C_2, \cdots C_{n_c}\}$ be the connected components of the graph $\mathcal{G}_{\geq w_j}$. And n_c is greater than or equal to the required number of connected components. Also let C be the matrix obtained by stacking the vectors $\mathbf{1}_{C_i}/|C_i|$ in columns. Then*

(a) $Tr(C^t L_i C) = 0$ for all $i > j$
(b) Any solution to the optimization problem (25) at level j is of the form CY where Y is any orthogonal matrix.

Proposition 2 allows us to optimize the first steps of algorithm 3 by considering the connected components instead of calculating the eigenvectors.

Also, recall that a discretization scheme (definition 1) was considered in calculating a $\Gamma-$limit. By suitably altering the discretization scheme, one can assume without loss of generality that $\mathcal{G}_{\geq w_2}$ has at least m (required number of clusters) connected components, and that $\mathcal{G}_{\geq w_1}$ has exactly one connected component.

In practice, this is assured by taking the union of all classes below the threshold. For instance, suppose initial weights considered are $0 \leq w_1 < w_2 < \cdots < w_k \leq 1$. Assume that if $\mathcal{G}_{\geq w_i}$ has at least m components and $\mathcal{G}_{\geq w_{i-1}}$ has less than m components. Then one can reorganize the weights to be $0 \leq w_1 < w_i < \cdots < w_k \leq 1$. This validates the assumption that $\mathcal{G}_{\geq w_2}$ has at least m (required number of clusters) connected components, and that $\mathcal{G}_{\geq w_1}$ has exactly one connected component.

During the first few iterations, while the number of components in the graph are greater than m, algorithm 3 computes the eigenvectors of the appropriate Laplacians. These eigenvectors are known to be equivalent to indicator vectors of the corresponding connected components. Hence, constructing the matrix $\hat{A}$ constitutes of stacking the indicator vectors of connected components. Using proposition 2, we have that while the number of connected components is greater than m, one can compute the matrix $\hat{A}$ by simply stacking the indicator vectors. This is equivalent to checking if the number of components is greater than m at various thresholds. Thresholding a graph to obtain connected components is equivalent to constructing an MST and removing the lower weight edges. Hence we refer to this as MST phase. This is described in steps 1-4 of algorithm 4.

Using the indicator vectors of the connected components, one can construct the matrix N. This constitutes steps 5-7 of algorithm 4.

Then using all remaining edges, one constructs the Laplacian L_1. The last step (last iteration of algorithm 3) constitutes of constructing the matrix $N^t L_1 N$ and computing the eigenvectors. This constitutes steps 8-11 of algorithm 4.

Algorithm 4 Simplified Efficient algorithm to compute $\Gamma-$limit for relaxed ratio-cut.

Input: A weighted graph, $\mathcal{G}$, with bucketed weights $w_1 < w_2 < \cdots < w_k$. Number of clusters required - m.

Output: N - A representation of the subspace spanned by $\Gamma-$limit of the minimizers.

1: Set $i = k$.
2: **while** Number of connected components of $\mathcal{G}_{\geq w_i}$ is greater than or equal to m **do**
3: Set $i = i - 1$ {We refer to this as an MST-Phase}
4: **end while**
5: Let $\{C_j\}$, $j \in \{1, 2, \cdots, n_c\}$ be the connected components in $\mathcal{G}_{\geq w_i}$.
6: Let I_{C_j} be the vector

$$I_{C_j}(x) = \begin{cases} 1/\sqrt{|C_j|} & \text{if } x \in C_j \\ 0 & \text{othwerwise} \end{cases} \tag{30}$$

7: Construct the matrix N with I_{C_j} as the column vectors.
8: Let $\mathcal{G}_1$ be the graph with the vertex set same as $\mathcal{G}$ and the edges are taken to be all the edges with weight $< w_i$. Let L_1 be the corresponding Laplacian.
9: Let $\overline{L_1}$ given by $N^t L_1 N$.
10: Calculate the first m eigenvectors of $\overline{L_1}$ and construct A using these eigenvectors as columns.
11: return NA.

Observe that one can write the Lapalcian of the original graph (using the same terminology of algorithm 4) as

$$L = L_1 + L_{\geq w_{i_0}} \tag{31}$$

where L_1 is the Laplacian in step 8 of algorithm 4, $L_{\geq w_{i_0}}$ is the Laplacian of $\mathcal{G}_{\geq w_{i_0}}$. i_0 is the value of i in step 5. Also,

$$N^t L_{\geq w_{i_0}} N = 0 \tag{32}$$

Thus, the optimization problem from step 10 of algorithm 4

$$\begin{aligned} \underset{H \in \mathbb{R}^{n \times m}}{\text{minimize}} \quad & Tr(H^t (N^t L_1 N) H) \\ \text{subject to} \quad & H^t H = \mathbf{I} \end{aligned} \tag{33}$$

is equivalent to (since $N^t N = \mathbf{I}$)

$$\begin{aligned} \underset{H \in \mathbb{R}^{n \times m}}{\text{minimize}} \quad & Tr(H^t L_1 H) \\ \text{subject to} \quad & H^t H = \mathbf{I} \\ & H = NX \text{ for some } X \end{aligned} \tag{34}$$

which is equivalent to the optimization problem

$$\begin{aligned} \underset{H \in \mathbb{R}^{n \times m}}{\text{minimize}} \quad & Tr(H^t L H) \\ \text{subject to} \quad & H^t H = \mathbf{I} \\ & H = NX \text{ for some } X \end{aligned} \tag{35}$$

Thus, another interpretation of the $\Gamma-$limit is - It is a solution to the ratio cut optimization problem with an additional constraint of belonging to the column space of N. Observe that the matrix NN^t is a projection matrix onto this subspace. We refer to the output of the algorithm 4 as *Power Ratio cut* or shortly *PRcut*.

Time Complexity of algorithm 4: Let n be the number of data points. Assume that $|V| \sim \mathcal{O}(n)$ and $|E| \sim \mathcal{O}(n)$. Observe that steps 1-4 mimics the construction of minimum spanning tree [14], and hence is $\mathcal{O}(nlog(n))$. In steps 5-7, observe that we are constructing a sparse matrix and hence is $\mathcal{O}(n_c)$. Since N and L_1 are sparse matrices, step 9 is $\mathcal{O}(n)$. Step 10, assuming the worst case scenario of $\overline{L_1}$ being dense matrix, is $\mathcal{O}((n_c)^3)$. Thus the complexity of algorithm 4 is

$$\mathcal{O}(nlog(n)) + \mathcal{O}(n_c^3) \tag{36}$$

Under the additional assumptions that $n_c \sim \mathcal{O}(m)$ and $m \sim \mathcal{O}(1)$ we have that the complexity of algorithm 4 is $\mathcal{O}(nlog(n))$.

Note that n_c depends mainly on the dataset and indirectly on the discretization scheme. This is the only aspect through which the discretization scheme affects the complexity.

In the trivial case of considering the discretized weights as all equal to 1, we have that the PRcut is identical to Ratio cut.

5 Applications and Analysis

It is clear from above that PRcut is an amalgam of MST-based clustering and spectral clustering. It combines 'good' properties of both of these methods. In particular, it is faster than spectral clustering and more accurate than MST-based clustering. All the eigenvector computations are carried out using the SciPy sparse library [22]. The code for generating the results used in this article is available at [7].

Since PRcut is designed to work on weights which take a discrete set of value, one needs a discretization scheme. This discretization scheme is sometimes referred to as *bucketing of weights*. In this article, we use *k-means based bucketing*. Observe that the problem of bucketing weights can be rephrased as - combine the weights into buckets so that weights within a bucket are 'alike'. This is once again a clustering problem in itself and hence one can use any clustering method to cluster the weights. k-means is a simple choice known for its efficiency and simplicity. In this case, each weight is represented by the mean of the cluster to which the weight belongs to. Note that the number of buckets is an important parameter. The greater the number of buckets the faster the algorithm is. However, the correct number of buckets to preserve the quality of the result depends on the domain of application.

Remark (Choosing the number of buckets:) Observe that if the number of buckets is equal to one, then power spectral clustering is identical to the spectral clustering algorithm. As the number of buckets increase, the time taken for the algorithm reduces. If the number of buckets are very large one ends up with MST-based clustering. The ideal number of buckets would be a value somewhere in between and depends majorly on the domain of application. For clustering on BSDS datasets, it has been found that 3 buckets provided an optimal performance.

5.1 Comparison with MST-based clustering

Observe that in algorithm 4, steps 1-4 mimics that of MST based clustering. However in the last step instead of breaking ties arbitrarily, PRcut takes into account the sizes of the clusters. In fact, **PRcut is a specific example of MST based clustering**.

The main difference between PRcut and MST based clustering is - while PRcut uses criterion based on size to pick the final clusters, MST based clustering makes this decision arbitrarily. To illustrate the difference better, consider a simple graph as in figure 2(a). The solid black lines indicate a choice of MST constructed. Figure 2(b) indicates a solution obtained by MST-based clustering. Given the MST from figure 2(a), since MST-based clustering picks the edge randomly, there is $1/2$ chance that figure 2(b) is obtained as a solution. On the other hand, since PRcut takes into consideration the sizes of the clusters, only the solution in figure 2(c) is obtained. Thus PRcut performs better in such cases as in figure 2.

5.2 Comparison with spectral clustering

Notably, PRcut preserves several good properties of spectral clustering as well. In particular, it preserves the property of being able to discover non-globular clusters. For instance, consider the data as in figure 3(a). Figure 3(b) indicates the results obtained using the spectral clustering method. Same result is obtained using Power Ratio cut as well. However, as the noise increases, spectral clustering breaks down (figure 3(d)) while PRcut is slightly more robust (figure 3(c)). This is also illustrated in figure 4.

Another important property of spectral clustering methods is that they penalize dissimilar sized clusters. This effect is preserved for PRcut as well. To verify this we conduct the following experiment - Consider the image in figure 5(a). The image is divided into 3 components - black component on the left, white component on the right and the boundary component in between. This is a synthetic example of the constant gradient at boundaries between classes, referred to as *flatzones*. Clustering the above image we expect the boundary component to be split into two parts - one corresponds to the black component and another corresponds to the white component. Figure 5(b) and figure 5(c) illustrate the result obtained with PRcut and Ratio cut.

As the initial size of the black component varies, due to the above property of spectral clustering, we expect the amount of 'boundary component' allotted to the black component to reduce. This is verified in figure 6 for Rcut. This is also the case for PRcut as well as shown in figure 6. Hence, one can conclude that PRcut preserves the property of penalizing dissimilar sized clusters.

The main difference between spectral clustering methods and PRcut is the run time. Due to preprocessing the data using MST, the size of the data is reduced drastically and hence computing time is saved. As discussed earlier, under some conditions the running time of PRcut increases as $\mathcal{O}(nlog(n))$ while spectral clustering methods take $\mathcal{O}(n^{3/2})$ [32]. This is illustrated in figure 7. Note that for small problems, the performance difference of PRcut and Rcut is negligible. However, as the size of the problem increases, PRcut performs better than Rcut.

5.3 Multiscale Combinatorial Grouping (Segmentation)

One of the main applications of spectral clustering is in the domain of image segmentation. Normalized cuts have been widely used to segment an image into meaningful regions [32,30,41,4]. Normalized cuts is used to globalize the local information. In [30], the authors propose a sequence of steps to obtain segmentation of the images. One of the most important steps in the pipeline is normalized cut. The following lines are taken from [30] -

> The normalized cuts criterion is a key globalization mechanism of recent high-performance contour detectors $\cdots$. Although powerful, such spectral graph partitioning has a significant computational cost and memory footprint that limit its scalability.

In this section, we use exactly the same pipeline as that in [30], but instead of using spectral clustering, we use PRcut. The authors then proposed a heuristic method to calculate the eigenvectors faster, termed as *dNcut*. This method reduces the eigenvalue problem and hence is faster and has less memory requirements, a technique similar to Nystrom's method [18,39].

Since, heuristically PRcut works by reducing unnecessary computation, PRcut is compatible with other

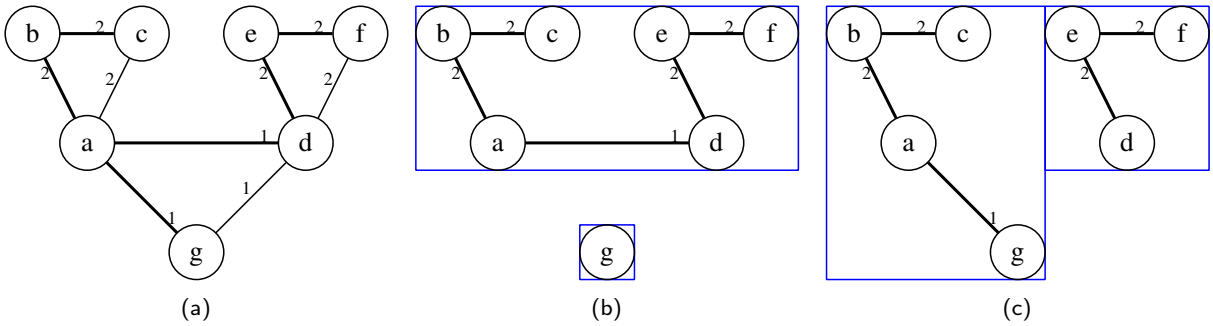


Fig. 2 Simple example illustrating the difference between PRcut and MST-based clustering. (a) An example graph. Solid lines indicate a maximum spanning tree. (b) A result obtained by using MST-based clustering. (c) Result obtained by using PRcut. Observe that PRcut provides better results compared to simple MST-based clustering.

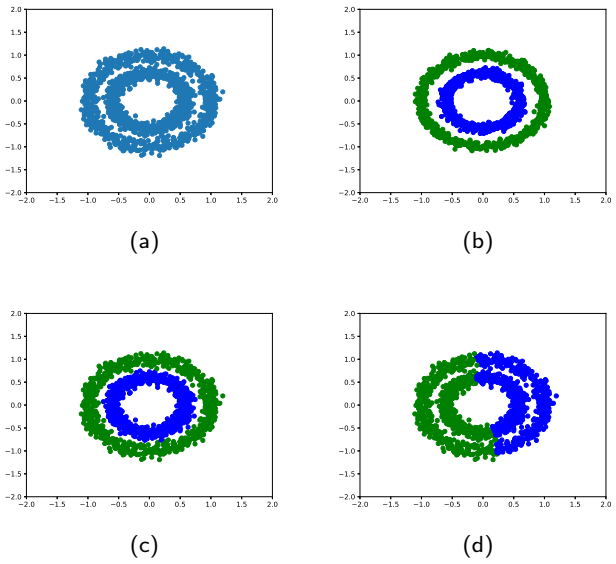


Fig. 3 Toy example illustrating the robustness of PRcut when compared to Ratio cut (Rcut). (a) Two circles dataset [1] with noise=0.07. (b) Result obtained using Ratio cut on the two circles dataset with noise=0.05. Same result is obtained with PRcut as well. (c) Result obtained by using PRcut on two circles dataset with noise=0.07. (d) Result obtained using Ratio cut on two circles dataset with noise=0.07.

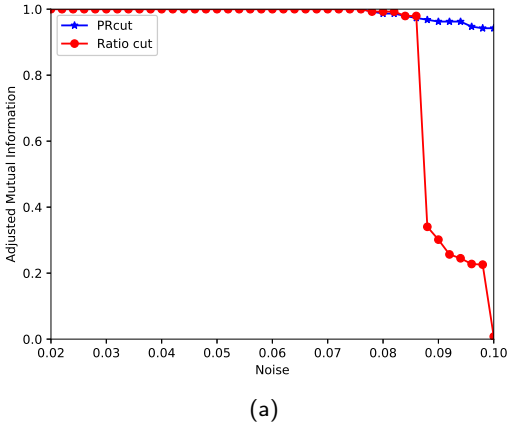


Fig. 4 Plot of the adjusted mutual information plotted vs the noise in the two circles dataset [1]. The values are averaged over 10 iterations. Note that at higher noise levels, PRcut performs better than Ratio cut.

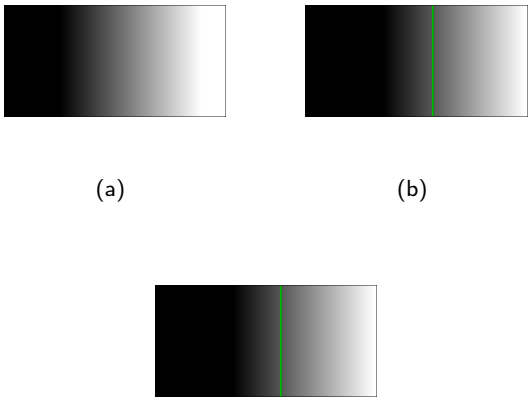
Table 1 Table indicating p-values on BSDS dataset under the hypothesis that the average F_{op} and F_b scores of Ncut and PRcut are equal. The sample size is taken to be 200. Empirical verifications of the normality assumptions are added in the appendix.

(p - values)	F_{op}	F_b
PRcut vs Ncut	0.2902	0.2819
dPRcut vs dNcut	0.3499	0.6884

reduction techniques. Hence, using the same reduction technique as in [30] along with PRcut, we have another technique called *dPRcut*. Also, all these methods have a multiscale version as well.

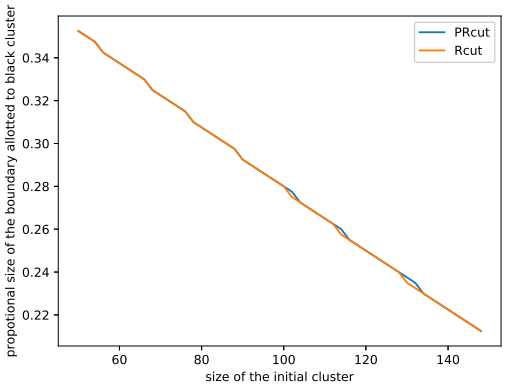
These methods were experimentally compared by using the BSDS500 dataset [4]. Few selected contour saliency maps [28] (also referred to as UCM, ultrametric contour maps in [3]) are shown in figure 8. The precision recall curves were plotted using the techniques described in [31], shown in figure 9. From this, it can be seen that the results are similar. Scatterplots in figures 9 (c), (d), (e), (f) corroborate this observation. Under the null hypothesis that the methods are equivalent, the p-values calculated by using the t-test are given in table 1. It is clear that there is *no* sufficient evidence to claim that one method is superior to the other in terms of accuracies. Note that dPRcut does not vary much compared to dNcut, thus establishing that PRcut is compatible with other reduction techniques as well.

However, PRcut is twice as fast as the Ncut, as shown in figure 10. In fact, the time of PRcut is comparable to $dNcut$ as well. Using the extra layer of approximation in $dPRcut$ and $dNcut$ did not show much of a difference



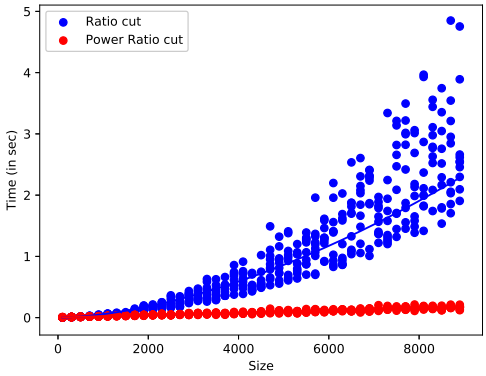
(a) (b) (c)

Fig. 5 (a) A synthetic example of the of a ramp image (constant horizontal gradient). The image consists of 3 components - A black component for $x < t_b$, a white component $x > t_w$ and a smooth transition between t_b and t_w. Here x refers to the horizontal axis. Further details can be found in [7]. (b) Result obtained using PRcut (c) Result obtained using Rcut. Observe that PRcut and Rcut gives similar partitions.



(a)

Fig. 6 Plot to illustrate that both PRcut and Rcut penalize differently sized clusters in the same way. We change the size of black component in figure 5(a) initially before clustering and plot the size of the boundary allotted to the black component after clustering. X-axis indicates the size of the black component initially before clustering. Y-axis indicates the size of the boundary allotted to the black component. Note that, for both PRcut and Rcut, the size of the boundary component allotted to the black component reduces as the initial size of the black component increases. This allows us to conclude that PRcut penalizes dissimilar clusters as well.



(a)

Fig. 7 Time complexity of PRcut vs Rcut as a function of data size on blobs dataset [2] with parameters - $n\_features = 2$, $centers = 2$. Time is measured in seconds. Observe that for small data sizes the difference between PRcut and Rcut is not significant, while for large data sizes it varies considerably. The variance in the timing of the ratio cuts is due to the optimizations of the SciPy sparse library [22].

since the size of the images is limited. However, depending on the size of the image, $dPRcut$ can be twice as fast as $dNcut$.

5.4 Hyperspectral Image Data

Clustering is also referred to as *unsupervised learning*. Solutions to clustering problem, unlike supervised learning, cannot offer black box solutions [23,21]. It is usually the case that several aspects of clustering are dictated by the domain knowledge. One of the main advantages of PRcut is that it offers better control over the clustering procedure than spectral clustering. We now describe the application of spectral clustering to hyperspectral image data. These results are first reported in [9].

A feature of the hyperspectral image data is that most of the data remains unclassified (class 0). Thus the classes are not balanced and hence spectral clustering methods cannot be used directly. Moreover, spectral clustering methods cannot be easily adapted to such cases as well. However, since PRcut has two phases - MST phase and spectral clustering, one can suitably modify the algorithm to requirements. In the case of hyperspectral image data, this is obtained by ignoring the small clusters after the MST phase in PRcut (explained in detail in [8]). This allows to obtain higher accuracy with PRcut compared to Rcut, as observed in case of Salinas dataset in table 2.

We have considered three hyperspectral image datasets - (a) Salinas (512×217) (b) Pavia University (610×340) (c) Pavia Center (1096×715). For each of the datasets,

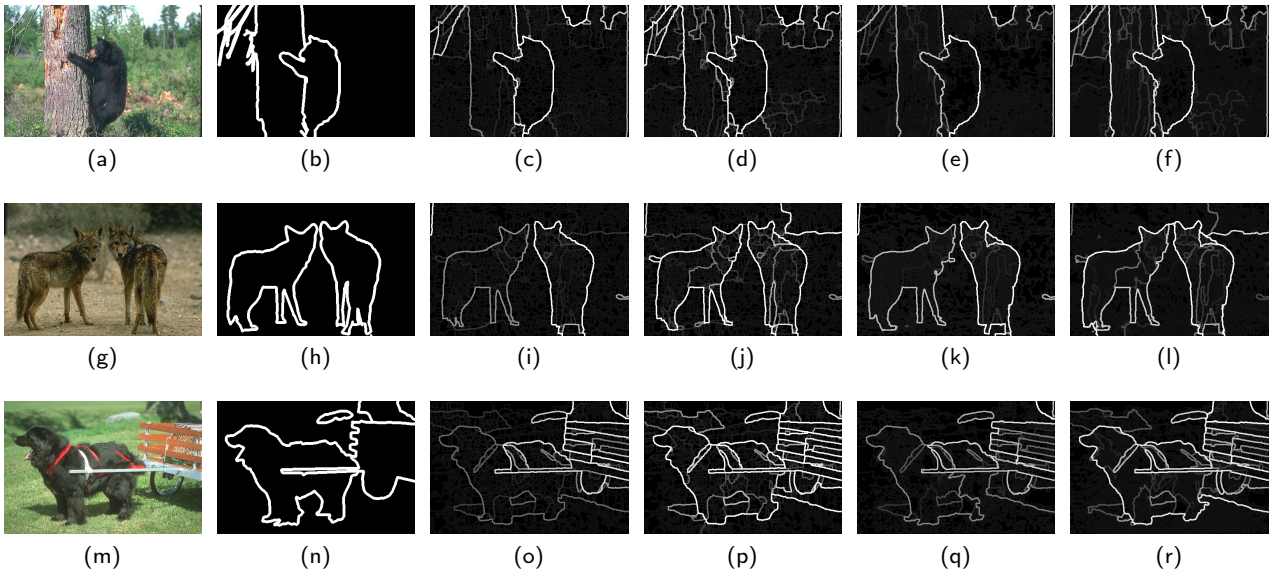


Fig. 8 Selected Results from BSDS500. First-column: Original Image, Second-Column: Groundtruth, Third-Column: PRcut, Fourth- Column: Normalized cut, Fifth-Column : Multiscale PRcut, Sixth-Column : Multiscale Ncut

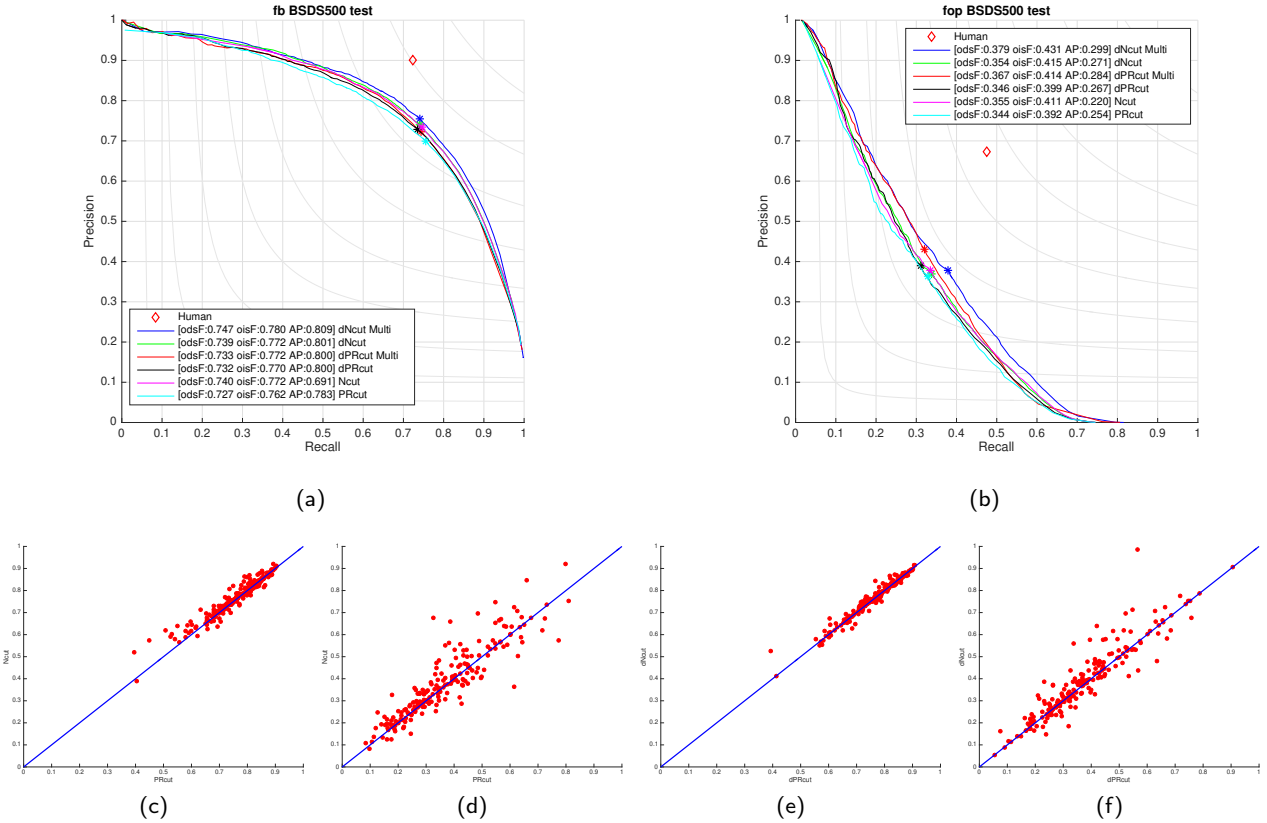
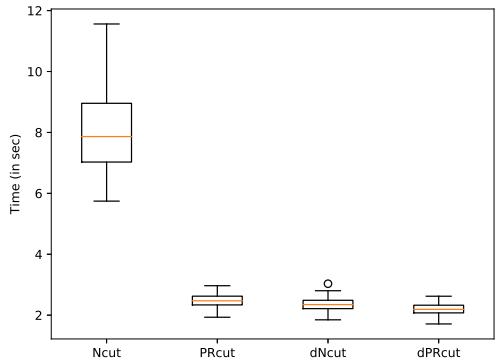


Fig. 9 (a) Shows the precision-recall curves of the measure F_b as defined in [31]. (b) Shows the precision-recall curves of the measure F_{op} as defined in [31]. 'dPRcut Multi' and 'dNcut Multi' indicate the multiscale versions of PRcut and Ncut respectively. (c) Scatter plot of optimal F_b measure on individual images between PRcut and Ncut. (c) Scatter plot of optimal F_{op} measure on individual images between PRcut and Ncut. (e) Scatter plot of optimal F_b measure on individual images between dPRcut and dNcut. (f) Scatter plot of optimal F_{op} measure on individual images between dPRcut and dNcut. Observe from (c)-(f) that there is no major difference in accuracy between PRcut and Ncut measures. All results are calculated on BSDS500 dataset with the MCG algorithm, using either a NCut or a PRCut step.

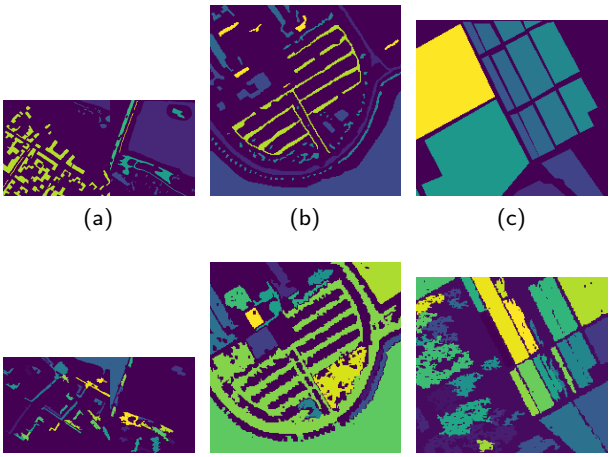
Table 2 Table indicating the average times and accuracy of Ratio cut and Power Ratio cut on hyperspectral image data. The timing is reported in seconds. The accuracy is measured by (37).

	Data	Process Time	Rcut		PRcut	
			Time	Accuracy	Time	Accuracy
Ratio = 0.7	Pavia University	129.33	95.15	0.70	27.38	0.74
	Pavia Center	530.95	435.01	0.76	91.38	0.77
	Salinas	142.59	94.27	0.28	24.92	0.71
Ratio = 0.8	Pavia University	194.14	151.00	0.72	34.06	0.78
	Pavia Center	623.74	535.79	0.78	104.46	0.78
	Salinas	202.98	139.31	0.36	26.88	0.74
Ratio = 0.9	Pavia University	279.42	163.66	0.75	40.71	0.76
	Pavia Center	830.18	633.638	0.79	115.09	0.79
	Salinas	271.56	170.28	0.43	31.63	0.78



(a)

Fig. 10 Box plot of the total time taken by various methods. The time is on Y-axis and is measured in seconds. The time taken is calculated from the input to the final segmented output obtained by MCG on the BSDS500 dataset. Based on average values, observe that MCG using PRcut is 2.23 times faster than MCG using Ncut and MCG using dPRcut is 1.07 times faster than MCG using dNcut. However, considering just the spectral clustering step, dPRcut is 1.5-2 times faster depending on the size of the image. Since the images in BSDS500 are small the difference between MCG using dPRcut and MCG using dNcut is not visible.



(a) (b) (c)

(d) (e) (f)

Fig. 11 Top row: Groundtruth images of Pavia Center, Pavia University and Salinas hyperspectral images. Bottom row: Results using PRcut of Pavia Center, Pavia University and Salinas hyperspectral images

we have considered three ratios - $0.7, 0.8, 0.9$, of their sizes. Each experiment was repeated 10 times and have taken the average. Also, the datasets were over segmented and the accuracy is calculated by assuming that each cluster is labelled with the largest groundtruth label. That is, let $\mathcal{C} = \{\mathcal{C}_i\}$ be classes obtained by clustering and let $\mathcal{C}^* = \{\mathcal{C}_i^*\}$ indicate the groundtruth classes. Let $N(\mathcal{C}_i, \mathcal{C}_j^*)$ be the number of pixels labelled $\mathcal{C}_i$ and $\mathcal{C}_j^*$. The accuracy is then given by

$$accuracy(\mathcal{C}, \mathcal{C}^*) = \frac{\sum_i (\max_j N(\mathcal{C}_i, \mathcal{C}_j^*))}{\text{Total number of pixels}} \tag{37}$$

The timing is measured in seconds. All experiments are done on *Intel(R) Xeon(R) CPU E5620 at 2.40GHz* with RAM size of 16 GigaBytes.

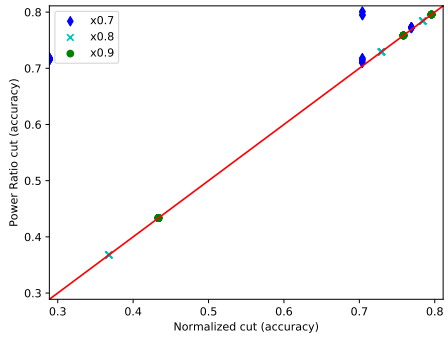
An example of the results obtained of the PRcut on subsets of these images and their groundtruth is shown in figure 11. Average of accuracy and times for each of the methods is shown in table 2. Scatter plots of accuracy and times are also plotted in figure 12. It is easy to see from the table that PRcut is faster and gives better accuracy than ratio cuts (or at least equal).

5.5 High dimensional data

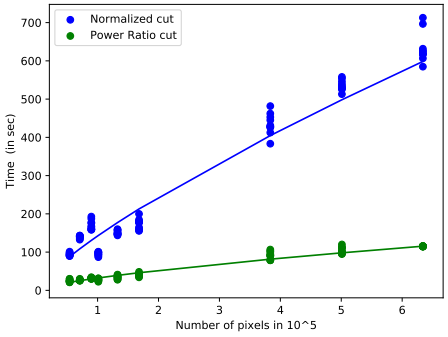
Another characteristic of the data is its dimensionality. In high dimensions, due to the curse of dimensionality [5,19], it is difficult to measure the distances accurately and hence difficult to construct an edge weighted graph. The distance between points become unreliable in higher dimensions. Thus spectral clustering cannot be directly applied for clustering high dimensional data. Usually a preprocessing step is applied to the data to reduce the

Table 3 Table indicating the average times and accuracy of Ratio cut and Power Ratio cut on MNIST dataset. The timing is reported in seconds. The accuracy is measured by (37). ARI indicates Adjusted Rand Index [20], and AMI indicates Adjusted Mutual Information [37].

	Data Size	Process Time	Rcut				PRcut			
			Time	Accuracy	ARI	AMI	Time	Accuracy	ARI	AMI
Dim = 10	4318	0.56	2.03	0.54	0.40	0.52	1.33	0.54	0.40	0.51
	8489	1.29	15.36	0.56	0.42	0.54	7.48	0.56	0.42	0.54
	12634	2.17	33.61	0.56	0.43	0.55	26.11	0.56	0.43	0.55
Dim = 15	4318	0.73	2.03	0.56	0.45	0.57	1.38	0.56	0.45	0.57
	8489	1.97	12.52	0.57	0.46	0.59	7.90	0.57	0.47	0.59
	12634	3.18	36.13	0.58	0.49	0.61	25.49	0.58	0.49	0.61
Dim = 20	4318	0.94	2.86	0.57	0.46	0.58	1.87	0.57	0.46	0.58
	8489	2.96	11.96	0.58	0.49	0.61	9.84	0.58	0.49	0.61
	12634	4.34	32.66	0.59	0.51	0.63	24.23	0.59	0.51	0.63



(a)



(b)

Fig. 12 (a) Scatter plot of the accuracy of normalized cuts vs power ratio cuts on hyperspectral image data. The red line indicates the line $x = y$. As one can see, all the points lie above or on the line $x == y$. Hence power ratio cut results are better than normalized cuts (or at least equal). (b) Plots of executing times vs the number of pixels in the image. It is easy to observe that PRcut scales much better compared to normalized cuts.

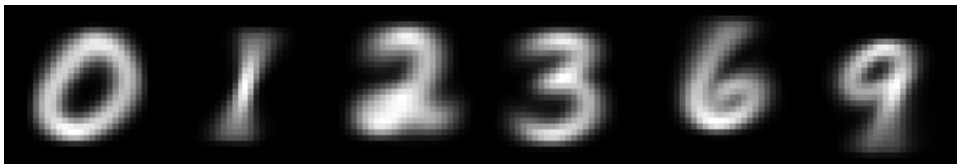
dimension, such as principal component analysis (PCA). The reduced data is then used for further processing.

Here the MNIST dataset [24] is used to experiment with high dimensions. The number of data points are 42000 and each data point has a dimension of 784. The dataset if first processed with PCA to reduce the dimensionality. Figure 13 shows the results on the selected classes. Table 3 shows the results obtained with PRcut and spectral clustering on randomized samples of the datasets, with varying dimension. It is clear from these results that PRcut is faster than Rcut while preserving the precision of the results.

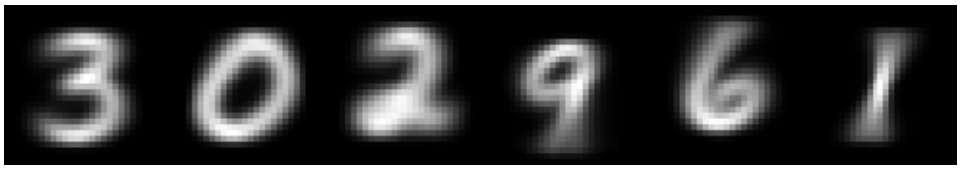
6 Conclusion

To summarize, we have proposed a faster alternative to the spectral clustering - Power Ratio cut or PRcut. This was obtained by considering a $\Gamma-$limit of the spectral clustering. Intuitively PRcut uses maximum spanning tree to reduce the size of the dataset and then solves the appropriate eigenvalue problem. This results in a much faster algorithm than spectral clustering. Also, by considering a $\Gamma-$limit several good properties of spectral clustering such as - penalizing unequal cluster distribution, ability to detect non-convex clusters is preserved. PRcut is compared with both MST-based clustering and spectral clustering considering several toy examples. Its efficiency as an alternate to Ncut is exhibited by suitably replacing Ncut in the MCG pipeline with PRcut and comparing the results on BSDS. Results of comparison with Ncut on hyperspectral image datasets are also provided. The code to generate the results in this paper is available at [7].

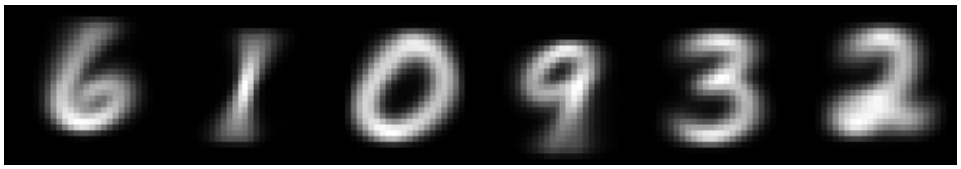
Note that, in this article we have considered a $\Gamma-$limit of the ratio cut. One can similarly probe the question of - "$\Gamma-$limit of normalized cut". It is relatively easy to extend the algorithm 4 replacing ratio-cut with normalized cut. However, the solutions to the normalized cut problem need not belong to a bounded set as $p \to \infty$. Hence the theory developed here must extended to incorporate normalized cuts. This is a direction of future research.



(a)



(b)



(c)

Fig. 13 Results obtained on selected classes of the MNIST dataset. (a) Average of the groundtruth classes. (b) Average of clusters obtained by Rcut (c) Average of clusters obtained by PRcut.This figure provides a visual confirmation that the results obtained by both PRcut and Rcut are the same.

7 Appendix

Proof of Theorem 3

Proof We first show that any matrix H of the form AKY is a solution to (7). We have

$$\begin{aligned} Tr((AKY)^t L(AKY)) &= Tr(Y^t(AK)^t L(AK)Y) \\ &= Tr((AK)^t L(AK)) \\ &= Tr(K^t(A^t LA)K) \end{aligned}$$

since Y is an orthogonal matrix. Now, $A^t LA$ is of the form

$$\begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ 0 & \cdots & \cdots & 0 \\ 0 & 0 & \cdots & \lambda_m \end{bmatrix}$$

So, we have

$$\begin{aligned} Tr(K^t(A^t LA)K) &= \lambda_1 + \lambda_2 + \cdots + \lambda_{l_1} + \lambda_{(m)} Tr(X^t \mathbf{I} X) \\ &= \lambda_1 + \lambda_2 + \cdots + \lambda_m \\ &= \min Tr(H^t LH) \end{aligned}$$

Since, the minimum value of $Tr(H^t LH)$ is equal to $\lambda_1 + \lambda_2 + \cdots + \lambda_m$ [25]. To show the other side, note that the set of all the solutions of (7) is

$$\{H \in \mathbb{R}^{n \times m} \mid Tr(H^t LH) = \lambda_1 + \lambda_2 + \cdots + \lambda_m\}$$

Let $\overline{A} = [e_1, e_2, \cdots e_n]$, i.e the matrix obtained by stacking all the eigenvectors of L in columns. Then any H can be written as $\overline{A}Z$ where $Z^t Z = \mathbf{I}$. Now, note that since $\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$ can be arbitrary, we need to have that

$$Z_{ij} = 0 \text{ if } \lambda_i > \lambda_{(m)}.$$

Thus we can ignore the lower part of matrix Z. If $A = [[e_1, e_2, \cdots e_l]]$, then we have that H is of the form AZ where Z is a $l \times m$ matrix such that $Z^t Z = \mathbf{I}$. Now let,

$$Z = \begin{bmatrix} Z_{l_1,l_1} & Z_{l_1,m-l_1} \\ Z_{l_2,l_1} & Z_{l_2,m-l_1} \end{bmatrix}$$

Also, note that A^tLA is of the form

$$\begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ 0 & \cdots & \cdots & 0 \\ 0 & 0 & \cdots & \lambda_l \end{bmatrix}$$

Since Z should satisfy $Tr(Z^tA^tLAZ) = Tr(H^tLH) = \lambda_1 + \lambda_2 + \cdots + \lambda_m$, we need to have that Z_{l_1,l_1} is full rank, and $Z^t_{l_1,l_1}Z_{l_1,l_1} = \mathbf{I}$. Hence Z must be of the form KY where K is a matrix of the form (26) and Y is any orthogonal matrix. Hence all the solutions to (7) must be of the form AKY.

Proof of lemma 1

Proof One side follows from the relation

$$\mathcal{H}(\mathcal{M}(H')) \supseteq H'$$

For the other side, let H be some matrix which spans the same subspace as $\widehat{H} = A_kKY$. Then there exists an orthogonal matrix Q such that $HQ = \widehat{H}$. Hence $H = \widehat{H}Q^t = A_kKYQ^t$. This is still in the form A_kKY where Y is some orthogonal matrix.

Proof of Proposition 2

Proof To prove (a), note that since C_i is a connected component of $\mathcal{G}_{\geq w_j}$, it is also a union of connected components of $\mathcal{G}_{\geq w_i}$ for all $i >= j$. Also, we know that if L denotes a Laplacian of the graph, then $L\mathbf{1}_{C_i} = 0$ if C_i is a connected component in the graph. Hence proved.

Since (a) is true, we know that C is a solution to (25) at level j. Thus all matrices of the form CY are solutions. Now, since $Tr(C^tL_iC) = 0$ for all $i >= j$, any vector c belonging to the column space of the solution must satisfy $c^tL_ic = 0$ for all $i >= j$. This implies that c belongs to the column space of the 0 eigenvectors, which are indicators of connected components. Hence c must belong to the column space of the indicators of connected components for each $\mathcal{G}_i$, $i >= j$. This implies that c belongs to the column space of indicators of connected components of $\mathcal{G}_{\geq w_j}$, and hence belongs to the column space of C. Hence proved.

Proof of theorem 2

Proof For the sake of simplicity, we consider x to be a 1d vector. Recall that

$$Q^{(p)}(x) = Tr(x^tL^{(p)}x) \tag{38}$$

such that $x^tx = 1$ and $x^t\mathbf{1} = 0$. We refer to this domain as $\lceil$.

Now, assume that $\hat{x}$ is a point belonging to M_1. We now show that $\hat{x}$ is also a limit point. From [25],

$$\|Q^{(p)}(\hat{x}) - \min_{x\in\lceil} Q^{(p)}(x)\| \to 0 \text{ as } p \to \infty \tag{39}$$

On the other hand, consider a ball around $\hat{x}$, $B(\hat{x}, \epsilon)$. Then,

$$\min_{y\in B(0,\epsilon)} Q^{(p)}(\hat{x}+y) = \hat{x}^tL^{(p)}\hat{x} + 2\hat{x}^tL^{(p)}y + y^tL^{(p)}y \tag{40}$$

Taking $y = -\epsilon\hat{x}$,

$$\min_{y\in B(0,\epsilon)} Q^{(p)}(\hat{x}+y) \leq \hat{x}^tL^{(p)}\hat{x}(1 - 2\epsilon + \epsilon^2) \tag{41}$$

From (39) and (41), for p large enough,

$$\min_{y\in B(0,\epsilon)} Q^{(p)}(\hat{x}+y) \leq \min_{x\in\lceil} Q^{(p)}(x) \tag{42}$$

In other words, for every ϵ ball around $\hat{x}$, there exists a minimizer of $Q^{(p)}(x)$. Hence, $\hat{x}$ is a limit point.

Checking normality for t-test in Table 1

To obtain p-values in table 1 we have assumed (a) normality of samples means and (b) χ^2 for sample variances. Note that these assumptions are sufficient to perform the t-test. Here, we verify these assumptions empirically.

To verify these assumptions, using bootstrap technique, we generate several samples of size 30. On these subsamples, sample means and variances are computed. Then using probability plots we empirically verify the distribution to be normal and χ^2 respectively. Figure 14 show the probability plots and the R^2 values for various quantitites.

Acknowledgements AC and SD would like to thank Indian Statistical Institute. LN would like to acknowledge the funding received from - ANR-15-CE40-0006 CoMeDiC, ANR-14-CE27-0001 GRAPHSIP research grants and Programme d'Investissements d'Avenir (LabEx BEZOUT ANR-10-LABX-58). BSDS would like to acknowledge the support received from the Science and Engineering Research Board (SERB) of the Department of Science and Technology (DST) with the grant number EMR/2015/000853, and the Indian Space Research Organization (ISRO) with the grant number ISRO/SSPO/Ch-1/2016-17.

References

1. scikit-learn datasets. `http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make$\_$circles.html`. Accessed: 2017-12-12

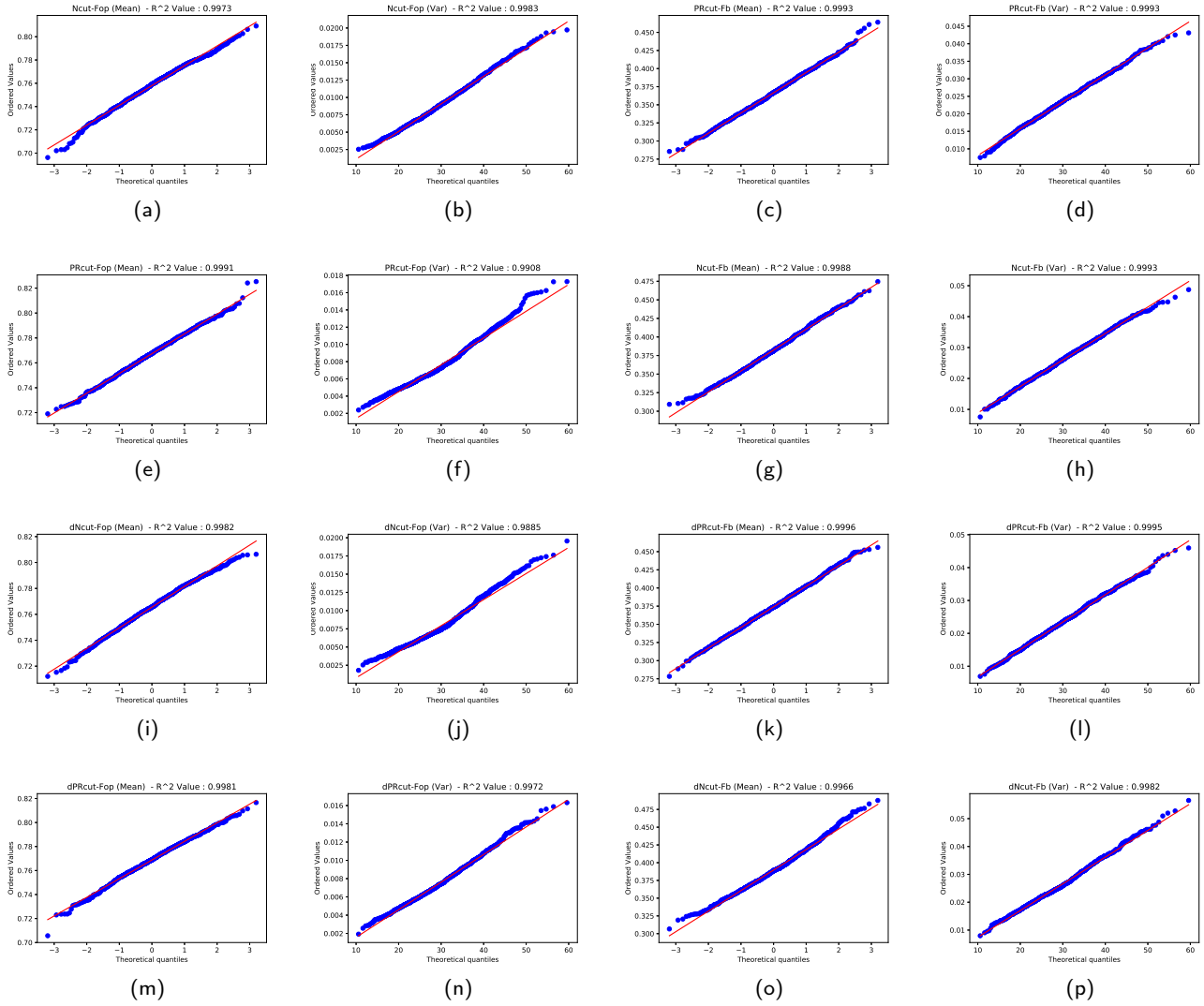


Fig. 14 Probability plot of sample data against the quantiles of a specified distribution - Normal for sample means and χ^2 for sample variance. The plots are generated using bootstrap samples of size 30. The data consists of means and variances of the measures F_{op} and F_b of the methods PRcut, dPRcut, Ncut, dNcut. Figure titles at the top of each plot indicate the values being plotted along with the R^2 values of the fit.

2. scikit-learn datasets. `http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make$\_$blobs.html`. Accessed: 2017-12-12
3. Arbelaez, P.: Boundary extraction in natural images using ultrametric contour maps. In: Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on, pp. 182–182. IEEE (2006)
4. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE transactions on pattern analysis and machine intelligence **33**(5), 898–916 (2011)
5. Bellman, R.E.: Adaptive control processes: a guided tour. Princeton university press (2015)
6. Braides, A.: Gamma-convergence for Beginners, vol. 22. Clarendon Press (2002)
7. Challa, A.: Power spectral clustering. `https://github.com/ac20/Power-Spectral-Clustering` (2018)
8. Challa, A., Danda, S., Daya Sagar, B.S., Najman, L.: An Introduction to Gamma-Convergence for Spectral Clustering, pp. 185–196. Springer International Publishing, Cham (2017). DOI 10.1007/978-3-319-66272-5_16. URL `https://doi.org/10.1007/978-3-319-66272-5_16`
9. Challa, A., Danda, S., Daya Sagar, B.S., Najman, L.: Power Spectral Clustering on Hyperspectral Data. In: International Geoscience and Remote Sensing Symposium. IEEE, Forth Worth, United States (2017). URL `https://hal.archives-ouvertes.fr/hal-01484896`
10. Chen, J., Fang, H.r., Saad, Y.: Fast approximate knn graph construction for high dimensional data via recursive lanczos bisection. Journal of Machine Learning Research **10**(Sep), 1989–2012 (2009)
11. Couprie, C., Grady, L., Najman, L., Talbot, H.: Power watershed: A unifying graph-based optimization framework. IEEE transactions on pattern analysis and machine intelligence **33**(7), 1384–1399 (2011)
12. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: Minimum spanning forests and the drop of

water principle. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(8), 1362–1374 (2009)
13. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: Thinnings, shortest path forests, and topological watersheds. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(5), 925–939 (2010)
14. Cousty, J., Najman, L., Kenmochi, Y., Guimarães, S.: Hierarchical segmentations with graphs: quasi-flat zones, minimum spanning trees, and saliency maps. Journal of Mathematical Imaging and Vision (2017). URL https://hal.archives-ouvertes.fr/hal-01344727
15. Danda, S., Challa, A., Sagar, B.S.D., Najman, L.: Revisiting the isoperimetric graph partitioning problem. IEEE Access **7**, 50636–50649 (2019). DOI 10.1109/ACCESS.2019.2901094. URL https://doi.org/10.1109/ACCESS.2019.2901094
16. Danda, S., Challa, A., Sagar, B.S.D., Najman, L.: Some theoretical links between shortest path filters and minimum spanning tree filters. Journal of Mathematical Imaging and Vision **61**(6), 745–762 (2019). DOI 10.1007/s10851-018-0866-1. URL https://doi.org/10.1007/s10851-018-0866-1
17. Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means: spectral clustering and normalized cuts. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 551–556. ACM (2004)
18. Fowlkes, C., Belongie, S., Chung, F., Malik, J.: Spectral grouping using the nystrom method. IEEE transactions on pattern analysis and machine intelligence **26**(2), 214–225 (2004)
19. Friedman, J., Hastie, T., Tibshirani, R.: The elements of statistical learning, vol. 1. Springer series in statistics New York (2001)
20. Hubert, L., Arabie, P.: Comparing partitions. Journal of classification **2**(1), 193–218 (1985)
21. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM computing surveys (CSUR) **31**(3), 264–323 (1999)
22. Jones, E., Oliphant, T., Peterson, P., et al.: SciPy: Open source scientific tools for Python (2001–). URL https://docs.scipy.org/doc/scipy/reference/sparse.html
23. Kleinberg, J.M.: An impossibility theorem for clustering. In: Advances in neural information processing systems, pp. 463–470 (2003)
24. Lecun, Y., Cortes, C.: The MNIST database of handwritten digits URL http://yann.lecun.com/exdb/mnist/
25. Lütkepohl, H.: Handbook of Matrices, 1 edn. Wiley (1997)
26. Maninis, K., Pont-Tuset, J., Arbeláez, P.A., Gool, L.V.: Convolutional oriented boundaries: From image segmentation to high-level tasks. CoRR **abs/1701.04658** (2017). URL http://arxiv.org/abs/1701.04658
27. Najman, L.: Extending the PowerWatershed framework thanks to Γ-convergence. SIAM Journal on Imagine Sciences **10**(4), 2275–2292 (2017). DOI 10.1137/17M1118580. URL https://hal.archives-ouvertes.fr/hal-01428875
28. Najman, L., Schmitt, M.: Geodesic saliency of watershed contours and hierarchical segmentation. IEEE Transactions on pattern analysis and machine intelligence **18**(12), 1163–1173 (1996)
29. Ng, A.Y., Jordan, M.I., Weiss, Y., et al.: On spectral clustering: Analysis and an algorithm. Advances in neural information processing systems **2**, 849–856 (2002)
30. Pont-Tuset, J., Arbelaez, P., Barron, J.T., Marques, F., Malik, J.: Multiscale combinatorial grouping for image segmentation and object proposal generation. IEEE transactions on pattern analysis and machine intelligence **39**(1), 128–140 (2017)
31. Pont-Tuset, J., Marques, F.: Supervised evaluation of image segmentation and object proposal techniques. IEEE transactions on pattern analysis and machine intelligence **38**(7), 1465–1478 (2016)
32. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on pattern analysis and machine intelligence **22**(8), 888–905 (2000)
33. Sinop, A.K., Grady, L.: A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In: Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pp. 1–8. IEEE (2007)
34. Song, Y., Chen, W.Y., Bai, H., Lin, C.J., Chang, E.: Parallel spectral clustering. Machine Learning and Knowledge Discovery in Databases pp. 374–389 (2008)
35. Turaga, S.C., Briggman, K.L., Helmstaedter, M., Denk, W., Seung, H.S.: Maximin affinity learning of image segmentation. In: Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, A. Culotta (eds.) Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada., pp. 1865–1873. Curran Associates, Inc. (2009). URL http://papers.nips.cc/paper/3887-maximin-affinity-learning-of-image-segmentation
36. Vazirani, V.V.: Approximation algorithms. Springer Science & Business Media (2013)
37. 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 **11**(Oct), 2837–2854 (2010)
38. Von Luxburg, U.: A tutorial on spectral clustering. Statistics and computing **17**(4), 395–416 (2007)
39. Williams, C.K., Seeger, M.: Using the nyström method to speed up kernel machines. In: Advances in neural information processing systems, pp. 682–688 (2001)
40. Wolf, S., Bailoni, A., Pape, C., Rahaman, N., Kreshuk, A., Köthe, U., Hamprecht, F.A.: The mutex watershed and its objective: Efficient, parameter-free image partitioning. IEEE transactions on pattern analysis and machine intelligence pp. 1–1 (2020)
41. Xiaofeng, R., Bo, L.: Discriminatively trained sparse code gradients for contour detection. In: Advances in neural information processing systems, pp. 584–592 (2012)
42. Yan, D., Huang, L., Jordan, M.I.: Fast approximate spectral clustering. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 907–916. ACM (2009)
43. Zahn, C.T.: Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Transactions on computers **100**(1), 68–86 (1971)