Weighted Agglomerative Clustering to Solve Normalized Cuts Problems

Giulio Genovese

Mathematics Department, Dartmouth College, 6188 Kemeny Hall, Hanover, NH 03755, U.S.A.

Abstract. A new agglomerative algorithm is introduced that can be used as a replacement for any partitioning algorithm that tries to optimize an objective function related to graph cuts. In particular, spectral clustering algorithms fall in this category. A new measure of similarity is introduced to show that the approach, although radically different from the one adopted in partitioning approaches, tries to optimize the same objective. Experiments are performed for the problem of image segmentation but the idea can be applied to a broader range of applications.

1 Introduction

Many algorithms have been introduced in recent years that handle the problem of partitioning a given set of elements in a specified number k of groups. Among these, spectral clustering [6–8], weighted kernel k-means [1–3] and non-negative matrix factorization [4, 9] have been proved to try to optimize the same objective function. Such an objective function is related to a special kind of graph cuts, such as ratio cut and normalized cut, that go under the broader name of weighted graph cuts. In this paper, we introduce a new measure of similarity between subsets that gives a mathematical basis to prove that it is possible to devise an agglomerative algorithm that tries to optimize the same mathematical function, adding another mathematically equivalent algorithm whose nature and performance are although substantially different.

This novel algorithm is closely related to linkage clustering algorithms and, in the case in which the objective function is related to ratio cuts, it is equivalent to average linkage [5, Section 10.9.2]. We performed some experiments with objective functions related to normalized cuts, in which case the algorithm resembles a sort of weighted average linkage. Spectral algorithms have been applied to optimize normalized cuts for the purpose of image segmentation. We tried to apply our new algorithm for the same purpose obtaining some preliminary positive results that lead us to conclude that the algorithm is competitive and deserving further study.

2 Normalized and Weighted Cuts

When clustering a dataset $\mathcal{V} = \{x_1, \ldots, x_n\}$ through graph cuts, we think of the elements of our dataset as the vertices of a graph and we associate to each edge the value corresponding to the affinity in between the two vertices. Suppose an affinity matrix A is given, where A_{ij} is the measure of affinity between element x_i and element x_j. We

Genovese G. (2008).
Weighted Agglomerative Clustering to Solve Normalized Cuts Problems.
In *Proceedings of the 8th International Workshop on Pattern Recognition in Information Systems*, pages 67-76

think of a partition $\{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$ as a cut through the edges in between the subsets of the partition. A cut will be optimal, then, according to the objective function we associate to the cut. A bunch of objective functions are popular in the literature. We first introduce the link between two subsets, not necessarily disjoint, as

$$l(\mathcal{A}, \mathcal{B}) = \sum_{x_i \in \mathcal{A}, x_j \in \mathcal{B}} A_{ij}. \tag{1}$$

Then we define the ratio association as an average measure of the clusters coherence and ratio cut as an average measure of cluster affinity.

$$RAssoc(\mathcal{V}_1, \ldots, \mathcal{V}_k) = \sum_{c=1}^{k} \frac{l(\mathcal{V}_c, \mathcal{V}_c)}{|\mathcal{V}_c|}, \quad RCut(\mathcal{V}_1, \ldots, \mathcal{V}_k) = \sum_{c=1}^{k} \frac{l(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{|\mathcal{V}_c|}.$$

Despite the two functions being closely related, an optimal cut for ratio association is not necessarily an optimal cut for ratio cut. To deal with this anomaly, the normalized cut [8] has been introduced as

$$NCut(\mathcal{V}_1, \ldots, \mathcal{V}_k) = \sum_{c=1}^{k} \frac{l(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{l(\mathcal{V}_c, \mathcal{V})}.$$

It is possible to define in an analogous way a normalized association but this is unnecessary since it follows easily that the sum between normalized cut and normalized association is constant. A generalization of these cuts is given by weighted cuts [3]. If we introduce a weight function ω over the subsets of $\mathcal{V}$, that is, a function such that for every two disjoints subset $\mathcal{A}$ and $\mathcal{B}$ of $\mathcal{V}$ it holds

$$\omega(\mathcal{A} \cup \mathcal{B}) = \omega(\mathcal{A}) + \omega(\mathcal{B}), \tag{2}$$

then the weighted association and the weighted cut are defined as

$$WAssoc(\mathcal{V}_1, \ldots, \mathcal{V}_k) = \sum_{c=1}^{k} \frac{l(\mathcal{V}_c, \mathcal{V}_c)}{\omega(\mathcal{V}_c)}, \quad WCut(\mathcal{V}_1, \ldots, \mathcal{V}_k) = \sum_{c=1}^{k} \frac{l(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{\omega(\mathcal{V}_c)}.$$

It turns out that ratio association is a special case of weighted association for which the weight function is just the set counting function, that is, $\omega(\mathcal{A}) = |\mathcal{A}|$ and the same can be said for ratio cut. Also, the normalized cut is a special case of a weighted cut where the weight function corresponds to the graph degree function, that is, $\omega(\mathcal{A}) = l(\mathcal{A}, \mathcal{V})$. Other possible weights can be used but the ones just introduced have been the most popular in the literature.

If the entries of the matrix A where ordered according to the clusters, Fig. 1 would represent graphically the elements of the matrix A that are involved respectively with the weighted cut and the weighted association.

Even if minimizing the weighted cut and maximizing the weighted association are not in general equivalent problem, a strategy to maximize the weighted association is in general sufficient. In fact, consider D the diagonal matrix whose diagonal elements are

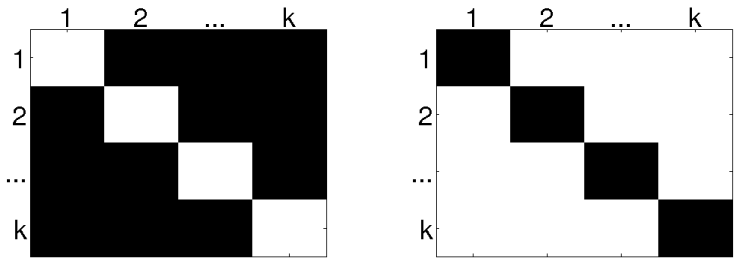

Fig. 1. Weighted cut and weighted association.

the sum of the rows of matrix A. Then define the matrix $A' = D - A$ as a new affinity matrix and define

$$l'(\mathcal{A}, \mathcal{B}) = \sum_{x_i \in \mathcal{A}, x_j \in \mathcal{B}} A'_{ij}, \quad WAssoc'(\mathcal{V}_1, \ldots, \mathcal{V}_k) = \sum_{c=1}^{k} \frac{l'(\mathcal{V}_c, \mathcal{V}_c)}{\omega(\mathcal{V}_c)}.$$

It then follows easily that

$$WAssoc'(\mathcal{V}_1, \ldots, \mathcal{V}_k) = WCut(\mathcal{V}_1, \ldots, \mathcal{V}_k),$$

and therefore a weighted cut can be interpreted as a weighted association for a different affinity matrix. Notice that, even if the matrix A' is guaranteed to be semi-positive definite, its components are mostly negative if the components of A are positive.

Spectral clustering (see [6] for a complete survey) can be seen as an algorithm to maximize weighted association or to minimize weighted cuts. The algorithm is based on an eigenvector decomposition of a matrix strictly related to the affinity matrix A. From this decomposition the elements to be clustered are projected on a lower dimensional vector space where a k-means algorithm separates them. Although, there is no quality guarantee for how well the objective function is optimized. Other approaches, like weighted kernel k-means and non-negative matrix factorization [4], can at least guarantee that the solutions produced satisfy some local optimality conditions. For most weighted cuts the problem of finding the optimal global solution is known to be NP-hard. We are going to introduce an alternative method that optimizes the weighted cut in a completely different way, but first we need to introduce a new way to measure the similarity between two subsets.

3 Similarity Measures

Suppose we are given a set $\mathcal{V}$ with n elements and an $n \times n$ symmetric matrix A whose component A_{ij} represents the affinity between element x_i and element x_j. Suppose also that we are given a weight function ω defined on the subsets of $\mathcal{V}$ and satisfying (2) and the function l defined as in (1). Then a convenient way to measure the similarity between two subsets $\mathcal{A}$ and $\mathcal{B}$ of $\mathcal{V}$ is to define a function S as

$$\mathcal{S}(\mathcal{A}, \mathcal{B}) = \frac{l(\mathcal{A}, \mathcal{B})}{\omega(\mathcal{A})\omega(\mathcal{B})}. \tag{3}$$

This measure turns out to be strictly connected with weighted cuts. To begin with, notice that, in case $\mathcal{A}$ and $\mathcal{B}$ are complementary and disjoint, that is, $\mathcal{B} = \mathcal{V} \setminus \mathcal{A}$, their similarity is a constant multiple of the weighted cut, since

$$\frac{WCut(\mathcal{A}, \mathcal{B})}{\omega(\mathcal{A} \cup \mathcal{B})} = \mathcal{S}(\mathcal{A}, \mathcal{B}).$$

In general it is true more.

Theorem 1. *Given a partition $\{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$ of a set $\mathcal{V}$, the expressions*

$$\frac{WCut(\mathcal{V}_1, \ldots, \mathcal{V}_k)}{(k-1)\omega(\mathcal{V})} \quad \textit{and} \quad \frac{WAssoc(\mathcal{V}_1, \ldots, \mathcal{V}_k)}{\omega(\mathcal{V})}$$

are convex linear combinations of the pairwise similarities $\mathcal{S}(\mathcal{V}_c, \mathcal{V}_d)$ for $c \neq d$ and the self similarities $\mathcal{S}(\mathcal{V}_c, \mathcal{V}_c)$, respectively.

Proof. For the first expression, we have that

$$WCut(\mathcal{V}_1, \ldots, \mathcal{V}_k) = \sum_{c=1}^{k} \frac{l(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{\omega(\mathcal{V}_c)} =$$

$$= \sum_{c=1}^{k} \sum_{d \neq c} \omega(\mathcal{V}_d) \frac{l(\mathcal{V}_c, \mathcal{V}_d)}{\omega(\mathcal{V}_c)\omega(\mathcal{V}_d)} = \sum_{c<d} (\omega(\mathcal{V}_c) + \omega(\mathcal{V}_d)) \mathcal{S}(\mathcal{V}_c, \mathcal{V}_d)$$

and since it holds that

$$\sum_{c<d} (\omega(\mathcal{V}_c) + \omega(\mathcal{V}_d)) = (k-1) \sum_{c=1}^{k} \omega(\mathcal{V}_c) = (k-1)\omega(\mathcal{V}),$$

the convexity follows. For the second expression it is enough to show that

$$WAssoc(\mathcal{V}_1, \ldots, \mathcal{V}_k) = \sum_{c=1}^{k} \frac{l(\mathcal{V}_c, \mathcal{V}_c)}{\omega(\mathcal{V}_c)} = \sum_{c=1}^{k} \omega(\mathcal{V}_c) \mathcal{S}(\mathcal{V}_c, \mathcal{V}_c).$$

Thanks to the previous theorem, if we have a strategy to minimize the pairwise similarities between the k disjoint subsets that will finally form the partition we are looking for, then we also have a strategy to minimize the weighted cut. The following theorem leads to the algorithm we need.

Theorem 2. *If the similarity measure $\mathcal{S}$ is defined as in* (3)*, then it follows that*

$$\begin{cases} \mathcal{S}(\mathcal{A}, \mathcal{B}) > \mathcal{S}(\mathcal{A}, \mathcal{C}) \\ \mathcal{S}(\mathcal{A}, \mathcal{B}) > \mathcal{S}(\mathcal{B}, \mathcal{C}) \end{cases} \Rightarrow \mathcal{S}(\mathcal{A}, \mathcal{B}) > \mathcal{S}(\mathcal{A} \cup \mathcal{B}, \mathcal{C}). \tag{4}$$

Proof. In fact, consider

$$\mathcal{S}(\mathcal{A} \cup \mathcal{B}, \mathcal{C}) = \frac{l(\mathcal{A} \cup \mathcal{B}, \mathcal{C})}{\omega(\mathcal{A} \cup \mathcal{B})\omega(\mathcal{C})} = \frac{l(\mathcal{A}, \mathcal{C}) + l(\mathcal{B}, \mathcal{C})}{\omega(\mathcal{A} \cup \mathcal{B})\omega(\mathcal{C})} =$$

$$\frac{\omega(\mathcal{A})}{\omega(\mathcal{A} \cup \mathcal{B})} \frac{l(\mathcal{A}, \mathcal{C})}{\omega(\mathcal{A})\omega(\mathcal{C})} + \frac{\omega(\mathcal{B})}{\omega(\mathcal{A} \cup \mathcal{B})} \frac{l(\mathcal{B}, \mathcal{C})}{\omega(\mathcal{B})\omega(\mathcal{C})} = \frac{\omega(\mathcal{A})\mathcal{S}(\mathcal{A}, \mathcal{C})}{\omega(\mathcal{A}) + \omega(\mathcal{B})} + \frac{\omega(\mathcal{B})\mathcal{S}(\mathcal{B}, \mathcal{C})}{\omega(\mathcal{A}) + \omega(\mathcal{B})},$$

so that $\mathcal{S}(\mathcal{A} \cup \mathcal{B}, \mathcal{C})$ is a convex linear combination of values that are both smaller than $\mathcal{S}(\mathcal{A}, \mathcal{B})$, making it smaller as well.

The property (4) is very important since it means that every time we join the two most similar clusters, the pairwise similarities in between the new cluster get smaller. This provides a basis for an agglomerative clustering algorithm. The idea is simple. At the beginning every element forms a different cluster and at every step we join the two clusters for which the similarity measure is the largest until we are left with k clusters. This way we make sure that the largest pairwise cluster similarity gets smaller at every step. Even if this does not guarantee an optimal solution for the weighted cut in the end, which in general is NP-hard to find, it does justify agglomerative clustering as a possible strategy.

Algorithm 1: Weighted agglomerative algorithm.

Input: Set $\mathcal{V} = \{x_1, \ldots, x_n\}$, affinity matrix A, and list of weights.
Initialize $\hat{k} \leftarrow n$ and clusters $\mathcal{V}_i \leftarrow \{x_i\}$.
repeat
  $\hat{k} \leftarrow \hat{k} - 1$
  Find clusters $\mathcal{V}_c$ and $\mathcal{V}_d$ for which $\mathcal{S}(\mathcal{V}_c, \mathcal{V}_d)$ is minimal.
  Join cluster $\mathcal{V}_c$ and cluster $\mathcal{V}_d$.
  Compute similarities between the new cluster and the other clusters.
until $k = \hat{k}$

Agglomerative clustering is a common tool for data clustering. It can be proved that average linkage [5, Section 10.9.2] is equivalent to the agglomerative algorithm previously described where we choose as weight the element counting weight, that is, $\omega(\mathcal{V}_c) = |\mathcal{V}_c|$. Usually agglomerative clustering deals with distances, or dissimilarities, instead of similarities, but the idea is basically the same.

4 Iterative Optimization

If our goal is to find the partition $\{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$ that maximizes the weighted association, then this is equivalent to minimizing the quantity

$$WAssoc(\{x_1\}, \ldots, \{x_n\}) - WAssoc(\mathcal{V}_1, \ldots, \mathcal{V}_k), \tag{5}$$

since the first term is just a constant independent of the partition. Define the divergence between two clusters $\mathcal{A}$ and $\mathcal{B}$ as

$$\mathcal{D}(\mathcal{A}, \mathcal{B}) = \mathcal{S}(\mathcal{A}, \mathcal{A}) - 2\mathcal{S}(\mathcal{A}, \mathcal{B}) + \mathcal{S}(\mathcal{B}, \mathcal{B}).$$

Theorem 3. *The quantity in* (5) *is equal to the sum*

$$\sum_{c=1}^{k} \sum_{x_i \in \mathcal{V}_c} \omega(x_i) \mathcal{D}(\{x_i\}, \mathcal{V}_c). \tag{6}$$

Proof. Rewrite the quantity in (5) as

$$
\begin{aligned}
&WAssoc(\{x_1\},\dots,\{x_n\}) - \sum_{c=1}^{k}\omega(\mathcal{V}_c)\mathcal{S}(\mathcal{V}_c,\mathcal{V}_c) = \\
&= WAssoc(\{x_1\},\dots,\{x_n\}) - 2\sum_{c=1}^{k}\omega(\mathcal{V}_c)\mathcal{S}(\mathcal{V}_c,\mathcal{V}_c) + \sum_{c=1}^{k}\omega(\mathcal{V}_c)\mathcal{S}(\mathcal{V}_c,\mathcal{V}_c) = \\
&= WAssoc(\{x_1\},\dots,\{x_n\}) - 2\sum_{c=1}^{k}\sum_{x_i\in\mathcal{V}_c}\omega(x_i)\mathcal{S}(\{x_i\},\mathcal{V}_c) + \sum_{c=1}^{k}\omega(\mathcal{V}_c)\mathcal{S}(\mathcal{V}_c,\mathcal{V}_c) = \\
&= \sum_{x_i\in\mathcal{V}}\omega(x_i)\mathcal{S}(\{x_i\},\{x_i\}) - 2\sum_{c=1}^{k}\sum_{x_i\in\mathcal{V}_c}\omega(x_i)\mathcal{S}(\{x_i\},\mathcal{V}_c) + \sum_{c=1}^{k}\omega(\mathcal{V}_c)\mathcal{S}(\mathcal{V}_c,\mathcal{V}_c) = \\
&= \sum_{c=1}^{k}\sum_{x_i\in\mathcal{V}_c}\omega(x_i)(\mathcal{S}(\{x_i\},\{x_i\}) - 2\mathcal{S}(\{x_i\},\mathcal{V}_c) + \mathcal{S}(\mathcal{V}_c,\mathcal{V}_c))
\end{aligned}
$$

and the last expression is the same as (6).

The previous expression, by trying to measure separately the contribution of each element, suggests an iterative strategy to optimize the weighted association. If we think of our elements as points in a Euclidean space, and if the weights are all equal to 1 and the divergence $\mathcal{D}$ coincides with the squared distance between the centroids, then the expression coincides with the sum of squared error [5, Section 10.7.1], that is, the objective function that is being minimized by the k-means algorithm.

We can therefore use ideas from the k-means literature to improve a given clustering obtained with the agglomerative approach used in the previous section. In particular, it is possible to generalize a common iterative optimization for k-means [5, Section 10.8] to optimize the weighted association. To this purpose, we first need a lemma.

Lemma 1. *If $\mathcal{A}$ is a cluster and x is an element of $\mathcal{V}$ that does not belong to the cluster $\mathcal{A}$, then the following holds*

$$
\begin{aligned}
&WAssoc(\mathcal{A},\{x\}) - WAssoc(\mathcal{A}\cup\{x\}) = \\
&\qquad \frac{\omega(\mathcal{A})}{\omega(\mathcal{A}\cup\{x\})}\omega(\{x\})\mathcal{D}(\{x\},\mathcal{A}) = \frac{\omega(\mathcal{A}\cup\{x\})}{\omega(\mathcal{A})}\omega(\{x\})\mathcal{D}(\{x\},\mathcal{A}\cup\{x\}).
\end{aligned}
$$

A proof can be found in the appendix. The previous lemma provides an easy rule for improving the weighted association. In fact, suppose we have an element x assigned to cluster $\mathcal{A}$ and suppose we are wondering if it would be better to assign it to cluster $\mathcal{B}$. We just need to check if

$$WAssoc(\mathcal{A},\mathcal{B}) < WAssoc(\mathcal{A}\setminus\{x\},\mathcal{B}\cup\{x\}).$$

By the lemma, this is equivalent to check if

$$\frac{\omega(\mathcal{A})}{\omega(\mathcal{A}\setminus\{x\})}\mathcal{D}(\{x\},\mathcal{A}) < \frac{\omega(\mathcal{B})}{\omega(\mathcal{B}\cup\{x\})}\mathcal{D}(\{x\},\mathcal{B}).$$

Therefore, if we keep updated the element-to-cluster similarities, it is easy to compute the element-to-cluster divergences and to check if the above equation holds. This leads to a natural iterative algorithm.

Algorithm 2: Weighted iterative algorithm.

Input: Set $\mathcal{V} = \{x_1, \ldots, x_n\}$, partition $\{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$, list of weights, and element-to-cluster similarities $\mathcal{S}(\{x_i\}, \mathcal{V}_c)$.
for all $x_i \in \mathcal{V}$ **do**
  if $x_i \in \mathcal{V}_c$ and there is a cluster $\mathcal{V}_d$ for which $\frac{\omega(\mathcal{V}_c)\mathcal{D}(\{x_i\},\mathcal{V}_c)}{\omega(\mathcal{V}_c \setminus \{x_i\})} < \frac{\omega(\mathcal{V}_d)\mathcal{D}(\{x_i\},\mathcal{V}_d)}{\omega(\mathcal{V}_d \cup \{x_i\})}$, **then**
    Reassign x_i from cluster $\mathcal{V}_c$ to cluster $\mathcal{V}_d$.
    Update the element-to-cluster similarities and the element-to-cluster divergences.
  end if
end for

Such an algorithm can be run at any time during the running of the previous algorithm. The algorithm is guaranteed to stop since at every step the weighted association of the partition increases. Although, many steps might be required before converging to a local optimum. One way to deal with this is to use weighted kernel k-means, which performs many reassignments at once, but if the similarity matrix $\mathcal{S}(\{x_i\}, \{x_j\})$ is not positive definite, then you are not guaranteed to improve the weighted association and some tricks might be required to enforce positive definiteness [3]. A related algorithm has been implemented inside the `kmeans` function in the MATLAB statistical toolbox where it is left to the user to decide if running it or not once the k-means algorithm has converged. In fact, depending on the number of elements and clusters, it might be more or less appropriate to perform this iterative procedure since it can be time consuming.

5 Experimental Results

Experimental results on images have given positive results. Moreover, the weighted approach using normalized cuts, compared to the unweighted approach using ratio cuts, has showed to be less likely to generate clusters of small sizes.

Fig. 2. First test image.

To show the results of our algorithm, we picked the left picture in Fig. 2 as a test figure. We used as the affinity matrix the one built with Yu's software [10]. In the image we intensified with red the pixels with large weight, as a result of the particular affinity matrix chosen, and with blue the pixels with small weight. We then run the agglomerative algorithm with $k = 6$, the target amount of clusters. The resulting segmentation is shown in the central picture. The value of the normalized cut turned out to be 0.055. Using the iterative optimization algorithm starting from the previous partitioning, we obtained the refined clustering shown in the right picture. The value of the normalized cut dropped, as expected, to 0.046.

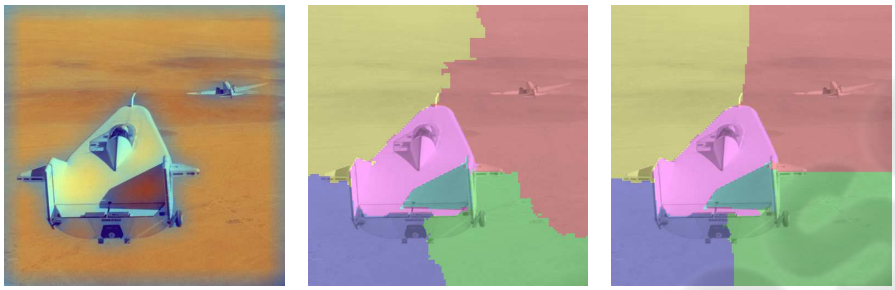

Fig. 3. Second test image.

A second example, using the left picture in Fig. 3, was used to show how the refining process can change dramatically the original partition. The initial segmentation is shown in the central picture. The value of the normalized cut was in this case 0.219. Using the iterative optimization algorithm produced the partition shown in the right picture. In this case the value of the normalized cut dropped to 0.172.

Fig. 4. Third test image.

To show another example, we picked the left picture in Fig. 4 and we run again the agglomerative algorithm. The resulting partitioning is shown in the central picture with value 0.189 for the normalized cut. The iterative optimization improved the value of the normalized cut to 0.170, producing the right picture. In all cases the iterative optimization provided a clear visual improvement of the cuts, although it should be

noticed that the amount of pixels that could end up being moved can be very large suggesting that it is important to initiate the algorithm from a partitioning that is already good.

Even if the similarity matrix was sparse, we didn't take advantage of this fact due to lack of time to code a sparse approach for the weighted agglomerative algorithm, so we preferred to omit running times. Although, experiments have shown that the algorithm is fast and practical and a sparse version would definitely make it competitive with a sparse spectral approach. All of the code used in the experiment has been written in MATLAB and C++ and is freely available on the author's website at the URL http://www.math.dartmouth.edu/∼genovese/.

6 Conclusions

A new agglomerative algorithm for clustering data has been proposed. Despite the fact that the algorithm has been proved to try to optimize the same objective function as k-means like algorithms, it is meant more as a complement rather than a substitute to partitioning algorithms. Many practical algorithms that deal with large datasets apply partitioning algorithms after coarsening the data enough to make the approach feasible. The mathematical framework here introduced justifies the weighted agglomerative approach as an algorithm to perform the coarsening.

An important direction of investigation would be the one of changing the weights associated to the elements. We have chosen the weights that best optimize the normalized cut objective function but any kind of weight would have delivered a different and possible algorithm and understanding better this choice might help deciding which algorithm should fit best the problem that is being tackled. In the clustering literature usually it is the case that some elements act as exemplars for other elements and maybe an appropriate weighting is exactly the right way to measure this fact.

Acknowledgements

This work is supported by NIH, grant number NIH R01GM075310-02.

References

1. I. Dhillon, Y. Guan, and B. Kulis, (2004), Kernel k-means: spectral clustering and normalized cuts, In KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 551–556, New York, NY, USA. ACM.
2. I. Dhillon, Y. Guan, and B. Kulis, (2005). A fast kernel-based multilevel algorithm for graph clustering. In KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, pages 629–634, New York, NY, USA. ACM.
3. I. Dhillon, Y. Guan, and B. Kulis, (2007). Weighted Graph Cuts without Eigenvectors A Multilevel Approach, IEEE Trans. Pattern Anal. Mach. Intell., 29(11):1944–1957.
4. C. Ding, X. He, and H. D. Simon, (2005). On the equivalence of nonnegative matrix factorization and spectral clustering. In Proc. SIAM Data Mining Conf., pages 606–610.

5. R. O. Duda and P. E. Hart, and D. G. Stork, (2000), Pattern Classification, 2nd ed., John Wiley and Sons.
6. Ulrike Luxburg, (2007), A tutorial on spectral clustering, Statistics and Computing, 17(4):395–416.
7. A. Ng, M. Jordan, and Y. Weiss, (2001). On spectral clustering: Analysis and an algorithm. In Advances in Neural Information Processing Systems 14: Proceedings of the 2001.
8. J. Shi and J. Malik, (2000), Normalized cuts and image segmentation, Transactions on Pattern Analysis and Machine Intelligence, 22(8):888-905.
9. W. Xu, X. Liu, and Y. Gong, (2003). Document clustering based on non-negative matrix factorization. In SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval, pages 267–273, New York, NY, USA. ACM.
10. Stella Yu, (2003). Computational Models of Perceptual Organization. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. CNBC and HumanID.

Appendix

A proof of lemma 1 is given.

$$WAssoc(\mathcal{A},\{x\}) - WAssoc(\mathcal{A}\cup\{x\}) =$$
$$= \omega(\mathcal{A})\mathcal{S}(\mathcal{A},\mathcal{A}) + \omega(\{x\})\mathcal{S}(\{x\},\{x\}) - \omega(\mathcal{A}\cup\{x\})\mathcal{S}(\mathcal{A}\cup\{x\},\mathcal{A}\cup\{x\}).$$

For the first equality rewrite the last expression as

$$\begin{aligned}
&\omega(\mathcal{A})\mathcal{S}(\mathcal{A},\mathcal{A}) + \omega(\{x\})\mathcal{S}(\{x\},\{x\}) - \omega(\mathcal{A})\mathcal{S}(\mathcal{A},\mathcal{A}\cup\{x\}) \\
&- \omega(\{x\})\mathcal{S}(\{x\},\mathcal{A}\cup\{x\}) = \\
&= \omega(\mathcal{A})\mathcal{S}(\mathcal{A},\mathcal{A}) + \omega(\{x\})\mathcal{S}(\{x\},\{x\}) - \frac{\omega(\mathcal{A})^2\mathcal{S}(\mathcal{A},\mathcal{A}) + \omega(\mathcal{A})\omega(\{x\})\mathcal{S}(\mathcal{A}),\{x\}}{\omega(\mathcal{A}\cup\{x\})} \\
&- \frac{\omega(\mathcal{A})\omega(\{x\})\mathcal{S}(\{x\},\mathcal{A}) + \omega(\{x\})^2\mathcal{S}(\{x\},\{x\})}{\omega(\mathcal{A}\cup\{x\})} = \\
&= \frac{\omega(\mathcal{A})}{\omega(\mathcal{A}\cup\{x\})}\omega(\{x\})(\mathcal{S}(\{x\},\{x\}) - 2\mathcal{S}(\{x\},\mathcal{A}) + \mathcal{S}(\mathcal{A},\mathcal{A})) = \\
&= \frac{\omega(\mathcal{A})}{\omega(\mathcal{A}\cup\{x\})}\omega(\{x\})\mathcal{D}(\{x\},\mathcal{A}).
\end{aligned}$$

For the second equality rewrite the previous expression as

$$\begin{aligned}
&\omega(\mathcal{A}\cup\{x\})\mathcal{S}(\mathcal{A}\cup\{x\},\mathcal{A}) - \omega(\{x\})\mathcal{S}(\{x\},\mathcal{A}) \\
&+ \omega(\{x\})\mathcal{S}(\{x\},\{x\}) - \omega(\mathcal{A}\cup\{x\})\mathcal{S}(\mathcal{A}\cup\{x\},\mathcal{A}\cup\{x\}) = \\
&= \frac{\omega(\mathcal{A}\cup\{x\})^2\mathcal{S}(\mathcal{A}\cup\{x\},\mathcal{A}\cup\{x\})}{\omega(\mathcal{A})} - \frac{2\omega(\mathcal{A}\cup\{x\})\omega(\{x\})\mathcal{S}(\{x\},\mathcal{A}\cup\{x\})}{\omega(\mathcal{A})} \\
&+ \frac{\omega(\{x\})^2\mathcal{S}(\{x\},\{x\})}{\omega(\mathcal{A})} + \omega(\{x\})\mathcal{S}(\{x\},\{x\}) - \omega(\mathcal{A}\cup\{x\})\mathcal{S}(\mathcal{A}\cup\{x\},\mathcal{A}\cup\{x\}) = \\
&= \frac{\omega(\mathcal{A}\cup\{x\})}{\omega(\mathcal{A})}\omega(\{x\})(\mathcal{S}(\{x\},\{x\}) - 2\mathcal{S}(\{x\},\mathcal{A}\cup\{x\}) + \mathcal{S}(\mathcal{A}\cup\{x\},\mathcal{A}\cup\{x\})) = \\
&= \frac{\omega(\mathcal{A}\cup\{x\})}{\omega(\mathcal{A})}\omega(\{x\})\mathcal{D}(\{x\},\mathcal{A}\cup\{x\}).
\end{aligned}$$