Contents

Chapter 1

Spectral Clustering for Data Streams

1.1 Introduction

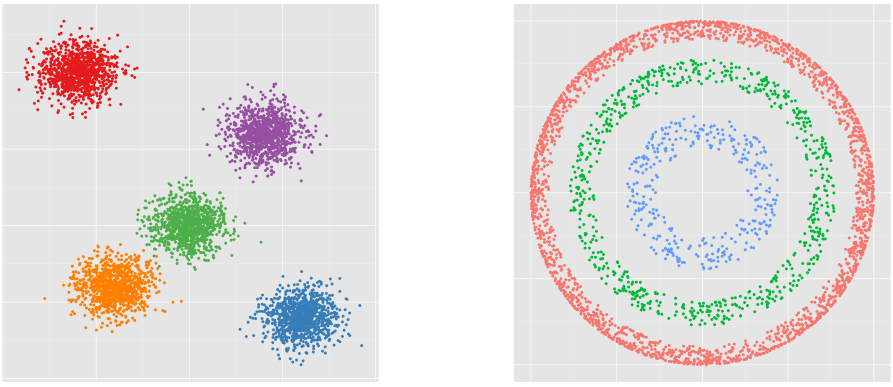
1.2 Spectral Clustering Background

In this section we motivate spectral clustering and introduce the spectral clustering algorithm by first noting the link between spectral clustering and graph partitioning problems.

1.2.1 Motivation

As discussed in Section 1.1 the goal of clustering algorithms is to partition data $X = \{x_1, \ldots, x_n\}, x_i \in \mathbb{R}^d$, into k disjoint classes such that each x_i belongs to exactly one class. Data sets can have underlying true clusters of all shapes and sizes, they can be spherical and convex as in Figure 1.2.1a or connected but non-convex as in Figure 1.2.1b.

Data which is convex may be simple to cluster as the gaps between clusters are easy for simple clustering algorithms like k-means to identify. Connected but non-convex data sets



(a) Convex clusters (b) Non-convex clusters

Figure 1.2.1: Examples of different types of clusters

can be much more challenging than convex data sets, and can cause some simple clustering algorithms to fail. The reason that centroid based clustering algorithms such as k-means struggle with data sets like that shown in Figure 1.2.1b is that k-means clusters the data based on how similar they are to cluster centroids.

Spectral clustering clusters data based on how similar they are to all other data points, which can lead to good quality segmentation on even these difficult cases. We do not formally address what is meant by similarity here, but will define this fully in Section 1.2.3.

The similarity between data points can be neatly represented in a *similarity graph*. We can then restate the clustering problem as a graph partitioning problem where we wish to find a partition of the graph such that the edges between different groups have low weights (which corresponds to data points in cluster being dissimilar) and edges within a group have high weights (the data points are similar).

In order to introduce spectral clustering we first discuss introduce some graph notation and discuss graph cut problems. We will then describe the spectral clustering algorithm, and

discuss in more detail the notion of similarity.

1.2.2 Graph cut problems

Data can be represented as a similarity graph, $G = (V, E)$ where each vertex $v_i \in V$ represents a data point x_i. The graph can then be described by an *adjacency matrix*. Adjacency matrices are a way of depicting the graph structure, showing which vertexes are connected by edges and which are not by creating a binary $n \times n$ matrix. Figure 1.2.2 depicts two similarity graphs. Their corresponding adjacency matrices are given in equation (1.2.1). A value of 1 in cell (2,3) implies that vertexes v_2 and v_3 are connected by an edge. Note that both of the example adjacency matrices given below are symmetric, which can be expected as we are dealing with undirected graphs.

Figure 1.2.2: Two similarity graphs

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{1.2.1}$$

RD: Formatting -shouldn't list matricies as equations.

The *weighted adjacency matrix* of a similarity graph is the matrix $W = (w_{ij})_{i,j=1,\ldots,n}$. The weight w_{ij} is the similarity between vertexes v_i and v_j. If $w_{ij} = 0$ this means that the vertexes

v_i and v_j are not connected by an edge. Again the weighted adjacency matrix, which is also called an affinity matrix will be symmetric $w_{ij} = w_{ji}$.

In order to create a graph partition we need to cut the edges in the graph. Non empty subsets of V, A and B will form a partition of the graph G if $A \cap B = \emptyset$ and $A \cup B = V$. The weight of the cut can be calculated by summing the weights of the edges which will be broken when a cut is made. In order to find a good partition of the graph, we wish to choose A and B such that some cut criterion is minimised. The simplest cut criterion is

$$\text{cut(A,B)} = \sum_{i \in A, j \in B} w_{ij}, \tag{1.2.2}$$

where the notation $i \in A$ is short hand to mean the set of indexes $\{i | v_i \in A\}$.

The Minimum cut (Wu and Leahy, 1993) is the cut which minimises equation (1.2.2). This is fairly easy to solve (Stoer and Wagner, 1997) however the minimum cut does not always produce a desirable graph partitioning; it tends to create unbalanced partitions, seperating one vertex from the rest of the graph. To understand why this happens, note that the number of edges cut in mincut will be $|A| \times |B|$ which is minimised by the solutions $|A| = 1$ or $|B| = 1$. In order to avoid this, we can specify that the sets A and B are reasonably large in some way. Two common objective functions used to avoid this issue are the RatioCut (Hagen and Kahng, 1992) and the normalised cut, Ncut (Shi and Malik, 2000).

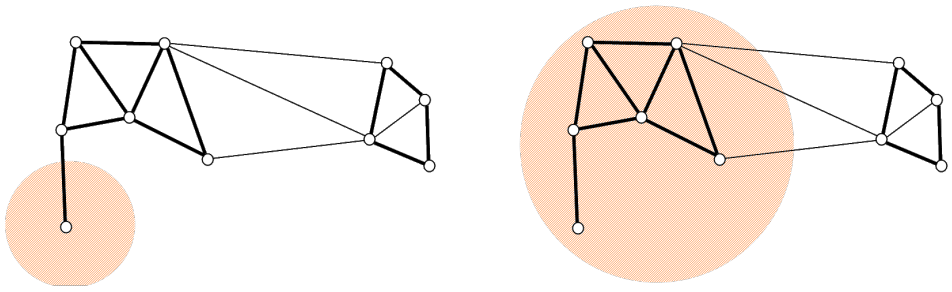
Both RatioCut and Ncut attempt to normalise the weight of the cut by introducing the size of sets A and B. In RatioCut, the size of A is measured by its number of vertexes $|A|$, while in Ncut the size is measured by the weights of its edges $\text{vol}(A) = \sum_{i \in A} d_i$ where $d_i = \sum_{j=1}^{n} w_{ij}$ is the *degree* of a vertex $v_i \in V$. The definitions of RatioCut and Ncut are as

follows,

$$\text{RatioCut(A,B)} = \frac{\text{cut}(A,B)}{|A|} + \frac{\text{cut}(A,B)}{|B|}, \tag{1.2.3}$$

$$\text{Ncut(A,B)} = \frac{\text{cut}(A,B)}{\text{vol}(A)} + \frac{\text{cut}(A,B)}{\text{vol}(B)}. \tag{1.2.4}$$

The main idea in Ncut is that large clusters will increase the denominator $\text{vol}(A)$ and thus decrease $\text{Ncut}(A,B)$. This will encourage splitting the data into fairly evenly sized clusters, and avoid the minimum cut issue of segmented isolated points. This can be seen in Figure 1.2.3 which depicts both the minimum cut and Ncut solutions for a particular graph. The shaded/non-shaded regions represent the partitioning. The minimum cut isolates one vertex from the rest of the graph, whilst the Ncut provides a more balanced and sensible partition.



(a) Minimising the cut. (b) Minimising the normalised cut.

Figure 1.2.3: Two solutions to the bi-partition problem. The partitioning is indicated by shading/non shading of nodes.

Although the partioning has been improved, the previously easy to solve mincut problem has been replaced with minimising the normalised cut which is an NP-hard problem (Wagner and Wagner, 1993). Spectral Clustering is a way to solve relaxed versions of these problems

(von Luxburg et al., 2008). It has been shown that the two graph cut optimisations given in equations (1.2.3), (1.2.4) can be formulated in terms of the spectral decomposition of the graph Laplacian matrices given in equations (1.2.5) and (1.2.6).

$$L = \quad D - W \tag{1.2.5}$$

$$L_{\text{symm}} = \quad D^{-1/2} L D^{-1/2} \tag{1.2.6}$$

Here W is the affinity matrix of the similarity graph, The degree matrix D is defined as the diagonal matrix with the degrees $d_1, \dots d_n$ on the diagonal. The L Laplacian relates to a relaxation of minimising the RatioCut and L_{symm} relates to the relaxation of the Ncut (Chung, 1997). There is no guarantee on the quality of the solution of the relaxed problem compared to the exact solution. There do exist pathalogical cases which are artibuary bad. However, several papers which investigate the quality of the clustering of spectral clustering Spielman and Teng (1996) and Kannan et al. (2004) find spectral clustering to provide good solutions, and is much easier to solve.

The spectral clustering algorithm that we use (Shi and Malik, 2000) uses the symmetric Laplacian L_{symm}. The full spectral clustering algorithm is given in Algorithm 1.

Once the Laplacian has been calculated, one computes the k eigenvectors which correspond to the k smallest eigenvalues of the Laplacian. A matrix $Y \in \mathbb{R}^{n \times k}$ is created, where each column is an eigenvector of the Laplcian, with length n. We can view this matrix Y as an embedding of the original data X into a lower dimensional subspace. When represented in this low subspace the clustering problem is often easier, and can be solved with a simple clustering algorithm such as k-means.

Algorithm 1 NJW spectral clustering algorithm

Input: Data set $X = \{x_1, \ldots, x_n\}$, number of clusters k

Output: k-way partition of the input data

1: Construct the affinity matrix $W = (w_{ij})_{i,j=1,\ldots,n}$

2: Compute the normalised Laplacian matrix $L_{\text{symm}} = D^{-1/2}(D - W)D^{-1/2}$, where D is the diagonal matrix with $D_{ii} = \sum_{j=1}^{n} w_{ij}$.

3: Compute the k eigenvectors of L_{symm}, $v_1, v_2, \ldots, v_k$, associated with the k smallest eigenvalues, and form the matrix $Y = [v_1, v_2, \ldots, v_k]$.

4: Renormalise each row of V to form a new matrix Y.

5: Partition the n rows of Y into k clusters via a general clustering algorithm, such as k-means.

6: Assign the original data point x_i to the cluster $k \iff$ the corresponding row i of the matrix Y is assigned to the cluster k.

Use k-means to cluster the n rows of Y into k clusters. In order to assign cluster labels to the original data, one assigns x_i to the cluster that row Y_i is assigned to.

RD: Show eigenvector embedding plots - DAVE

1.2.3 Choice of affinity matrix

One of the key factors of Spectral Clustering is the affinity matrix $W = (w_{ij})_{i,j=1,\ldots,n}$ which represents the pairwise similarities or distances between all data points x_i and x_j. A popular choice is to use the Gaussian kernel,

$$w_{i,j} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right),\ i,j = 1,\ldots,n, \tag{1.2.7}$$

where the parameter σ controls the width of the local neighbourhoods which we want to model. If x_i and x_j are very close, then $w_{ij} \to 1$, and if they are far apart $w_{ij} \to 0$. A gaussian kernel affinity matrix will have ones along the diagonal and is symmetric ($w_{ij} = w_{ji}$).

The scaling parameter, σ, is usually chosen manually. Ng et al. (2001) automatically choose σ by running their clustering algorithm repeatedly for a number of values and selecting the one which provides least distorted clusters. Zelnik-Manor and Perona (2004) argue that for data which has a cluttered background, or multi-scale data, one global parameter choice for σ is not sufficient. They calculate a localised parameter σ_i for each data point x_i based on its neighbourhood. Using a localised σ_i can deal well with multi-scale data, but requires the user to choose the size of the neighbourhood to calculate σ_i.

If we mainly wish to model the local relationships, using all of the possible pairwise data connections may not be necessary. It is possible to use a weighted k-nearest neighbour

structure (von Luxburg et al., 2008) to build the affinity matrix once corrections have been made to ensure that this matrix is symmetric. Another option is to choose some threshold ϵ and only consider connections between data points whose pairwise distances are smaller than ϵ. This is an ϵ-neighbourhood graph. Although it is possible to weight this graph by ϵ, if we choose ϵ to generate a small ϵ-neighbourhood, then the differences between the weights will be so small that weighting may become eligible. Therefore a simple construction of the ϵ-neighbourhood graph is as shown in equation (1.2.8).

$$w_{ij}^{*} = \begin{cases} 1, & \text{if } w_{ij} < \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{1.2.8}$$

Using this construction will give a sparse affinity matrix instead of a fully connected graph, which will help lower the computational complexity.

1.3 Advanced Spectral Clustering

In the previous section we introduced spectral clustering via graph partioning and discussed options for creating affinity matricies. In this section we review recent methods to deal with using spectral clustering on static large datasets and incremental datasets.

1.3.1 Large-scale Spectral Clustering

Although spectral clustering has been shown to perform well empirically on simple data sets, computational problems arise as the data set size increases. Spectral clustering can be challenging for very large data sets, constructing the affinity matrix W and computing the eigenvectors of L have computational complexity $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ respectively.

The Nyström method (Williams and Seeger, 2001) is a general method for generating good quality low rank approximations of large matricies. The Nyström approximation method for spectral clustering (Fowlkes et al., 2004) randomly samples the columns of the affinity matrix W and approximates the eigen decompostion of the full matrix directly using correlations between the sampled columns and the remaining columns. Effectively this can be thought of as a dial which the user has control over, sampling more columns will provide better results but at a higher computational cost. The downsides with this method are that there are no guarentees, the working memory can be high, and the random sampling of columns may lead to small clusters being under represented or completely missed in the final clustering.

An alternative to the Nyström method is to use a pre-processing technique to reduce the size of the data. A natural way to do this is to select certain representative points to represent the whole dataset. Yan et al. (2009) proposed KASP and RASP, algorithms which use k-means and random forest methods respectively to select q representative points to apply spectral clustering on. Similarly Shinnou and Sasaki (2008) uses also k-means to identify representative points, removes points closes to centers ran SC on remaining points and their committees. Both Yan et al. (2009) and Shinnou and Sasaki (2008) link the clustering of the representative points back to the original points by taking on the value of their closest representative point. As an alternative, Chen and Cai (2011) represents the data as a linear combination of representative points. A sequential reduction algorithm is adopted in Chen et al. (2006); Liu et al. (2007) with an early stopping criteria based on an observation that well seperated data points converge to the final embedding more quickly. However this is only suitable for binary clustering. Other possibilties include random projection with sampling methods (Sakai and Imiya, 2009) and shortest path methods (Liu et al., 2013).

We discuss the KASP algorithm in more detail as it is the most popular speed up method for spectral clusting and it inspired our work in online spectral clustering which is introduced in Section 1.4.

In KASP, k-means is applied with q clusters to the dataset X, where q is chosen such that $k \ll q \ll n$. Therefore each point in X belongs to a cluster $y_j, (j \in 1, \ldots, q)$. Let the centers of these q clusters be $\widehat{y_1}, \ldots, \widehat{y_q}$. These are used as representative points for the whole dataset. Spectral clustering is performed on the representative points, reducing the complexity of the eigendecomposition from $\mathcal{O}(n^3)$ to $\mathcal{O}(q^3)$. Finally, the original datapoints are assigned the cluster label that their closest representative point $\widehat{y_j}$ was assigned in the spectral clustering. The KASP algorithm is given in Algorithm 2.

Algorithm 2 KASP

Input: Data set $X = x_1, \ldots, x_n$, number of clusters k, number of representative points q

Output: k-way partition of the input data

1: Perform k-means with q clusters on $x_1, \ldots, x_n$ to create clusters $y_1, \ldots y_q$.

2: Compute the cluster centroids $\widehat{y_1}, \ldots, \widehat{y_q}$ as the q representative points.

3: Build a correspondence table to associate each x_i with the nearest cluster centroids $\widehat{y_j}$.

4: Run a spectral clustering algorithm on $\widehat{y_1}, \ldots, \widehat{y_q}$ to obtain an k-way cluster membership for each of $\widehat{y_j}, (j \in 1 \ldots q)$.

5: Recover the cluster membership for each x_i by looking up the cluster membership of the corresponding centroid $\widehat{y_j}$ in the correspondence table.

Both KASP and RASP have been shown to perform well empirically on large datasets (Yan et al., 2009), retaining good clustering performance even as the *data reduction ratio* increases. We can express the data reduction ration as $\gamma = \frac{n}{q}$. As in many of the sampling

methods discussed above, in KASP the user has control over the data reduction rate. A larger value of q will give a better performance but at a computational cost. The KASP authors present an upper bound on the misclustering rate given the pertubation to the original data. However as the bound depends on eigengaps and the misclustering rate is bound between 0 and 1, this upper bound can be weak. Finally, the method of assigning datapoints to clusters based on the cluster label of their represntative point can lead to poor segmentations shown in Cao et al. (2014). They propose a local interpolation in their algorithm Local Information-based Fast Approximate Spectral Clustering (Li-ASP) to prevent this poor segmentation issue. They achieve this by assigning datapoints based on a weighted version of their p closest representative points labels, rather than labelling based just on the label of the single closest representative point.

The methods discussed above only address dealing with large datasets which are static. Our aim is to investigate methods which can update the spectral clustering partioning when new datapoints arrive.

1.3.2 Incremental methods for Spectral Clustering

A data stream is a potentially endless sequence of observations obtained at high frequency relative to the available processing and storage capabilities. Datastreams arise in many applications such as online purchases, modelling epidemics and understanding sensor networks. RD: Link papers Re-clustering the graph at each time step whenever new information arrives is not feasible especially if data is arriving rapidly. Therefore there is a need for an online spectral clustering algorithm. To my knowledge, there is not currently a method for spectral clustering of datastreams in a fully online manner. The problem of performing spectral clus-

tering in data streams has been considered, but framed as an evolving network rather than a stream where data appears rapidly. The algorithms for clustering evolving network methods tend to focus on either (i) updating the cluster membership directly (ii) incrementally updating the eigenvectors.

The first method is described in Valgren and Lilienthal (2008), a clustering algorithm for an online topoloical mapping mobile robot. When new points arrive, the spectral clustering is updated directly using a similarity threshold to assign points to clusters. If a new data point is sufficiently far from its closest representative points, it is considered the start of a new cluster, this means that the number of overall clusters must always increase. How? Affinity is shrunk whenever new cluster is added. There are a number of problems with this method. The number of clusters (and therefore the size of the affinity matrix) must always increase. Cannot deal with cluster splits

An incremental update algorithm is proposed in Ning et al. (2007) and Ning et al. (2010). Their algorithm can deal with both additional data points joining the network, and similarity weights changing between existing data points. The algorithm updates the eigenvectors and eigenvalues directly without performing a full eigen-decomposition. The addition of a new data point is treated as a series of n weight changes, where is n is the number of currently observed data points. However the authors recommend a full re-clustering in batch to minimise cumulative errors. There are some issues with update method, mainly that the updating of eigenvectors means that the orthogonality property may be lost - potentially leading to poor cluster detection. Also if the spatial neighbourhoods of often changing vertices are large it can still be computationally difficult as the eigenvector update step involves the inversion of a matrix. Finally the authors recommend a full spectral re-clustering occasionally

to prevent the accumulation of errors in the eigenvectors, this is not feasible in the streaming setting. Generally this method is not suitable for data streaming, as the size of the Laplacian can grown unbounded for an infinite data stream. We did intend to use this algorithm as a competing algorithm in our experiments section RD: ref. However, the computational costs for Ning were so great for datastreaming examples that it was not possible to run the study. (Even when not performing a full reclustering).

Another incremental update algorithm is detailed in Dhanjal et al. (2011) Dhanjal et al. (2014) which approximates the eigen decomposition of the Laplacian incrementally but still requires regular full re-clustering.

Kong et al. (2011) is a bit of a mix between Ning and Valgren.

Other varients include using fuzzy C k-means (Bouchachia and Prossegger, 2012) and a model baseds kernel spectral clustering (Langone et al., 2014).

As discussed, non of the above methods are suitable for a clustering a datastream. We consider an online spectral clustering algorithm for datastreams based on the Clustream model of **?**. The Clustream algorithm is introduced in full in Section 1.4.

1.4 Online Spectral Clustering

1.4.1 The challenges of clustering data streams

1.4.2 Clustream

1.5 Experimentation

1.6 Conclusion

Bibliography

Bouchachia, A. and Prossegger, M. (2012). Incremental spectral clustering. In *Learning in Non-Stationary Environments: Methods and Applications*, volume 9781441980, pages 77–99. Springer New York, New York, NY.

Cao, J., Chen, P., Dai, Q., and Ling, W. K. (2014). Local information-based fast approximate spectral clustering. *Pattern Recognition Letters*, 38(1):63–69.

Chen, B., Gao, B., Liu, T.-Y., Chen, Y.-F., and Ma, W.-Y. (2006). Fast spectral clustering of data using sequential matrix compression. *Machine Learning: ECML 2006*, pages 590–597.

Chen, X. and Cai, D. (2011). Large Scale Spectral Clustering with Landmark-Based Representation. *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence Large*, pages 313–318.

Chung, F. R. (1997). *Spectral graph theory*, volume 92. American Mathematical Soc.

Dhanjal, C., Gaudel, R., and Clémençon, S. (2011). Incremental spectral clustering with the normalised laplacian.

Dhanjal, C., Gaudel, R., and Clémençon, S. (2014). Efficient eigen-updating for spectral graph clustering. *Neurocomputing*, 131:440–452.

Fowlkes, C., Belongie, S., Chung, F., and Malik, J. (2004). Spectral grouping using the Nyström method. *IEEE transactions on pattern analysis and machine intelligence*, 26(2):214–25.

Hagen, L. and Kahng, A. (1992). New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085.

Kannan, R., Vempala, S., and Vetta, A. (2004). On Clusterings: Good, Bad and Spectral. *Journal of the ACM*, 51(3):497–515.

Kong, T., Tian, Y., and Shen, H. (2011). A Fast Incremental Spectral Clustering for Large Data Sets. In *2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 1–5. IEEE.

Langone, R., Mauricio Agudelo, O., De Moor, B., and Suykens, J. A. (2014). Incremental kernel spectral clustering for online learning of non-stationary data. *Neurocomputing*, 139:246–260.

Liu, J., Wang, C., Danilevsky, M., and Han, J. (2013). Large-scale spectral clustering on graphs. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 1486–1492.

Liu, T.-Y., Yang, H.-Y., Zheng, X., Qin, T., and Ma, W.-Y. (2007). Fast large-scale spectral clustering by sequential shrinkage optimization. In *Proceedings of the 29th European conference on IR research*, pages 319–330, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). On Spectral Clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, pages 849–856.

Ning, H., Xu, W., Chi, Y., Gong, Y., and Huang, T. (2007). Incremental Spectral Clustering With Application to Monitoring of Evolving Blog Communities. *SIAM International Conference on Data Mining*, pages 261–272.

Ning, H., Xu, W., Chi, Y., Gong, Y., and Huang, T. S. (2010). Incremental spectral clustering by efficiently updating the eigen-system. *Pattern Recognition*, 43(1):113–127.

Sakai, T. and Imiya, A. (2009). Fast spectral clustering with random projection and sampling. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, volume 5632 LNAI, pages 372–384.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

Shinnou, H. and Sasaki, M. (2008). Spectral Clustering for a Large Data Set by Reducing the Similarity Matrix Size. *In proceedings of the Sixth International Language Resources and Evaluation(LREC08)*, pages 201–204.

Spielman, D. A. and Teng, S.-H. (1996). Spectral partioning works: planar graphs and finite element meshes. In *37th Ann. Symp. on Found. of Comp. Science (FOCS)*, pages 96–105.

Stoer, M. and Wagner, F. (1997). A simple min-cut algorithm. *Journal of the ACM*, 44(4):585–591.

Valgren, C. and Lilienthal, A. (2008). Incremental spectral clustering and seasons:

Appearance-based localization in outdoor environments. In *2008 IEEE International Conference on Robotics and Automation*, pages 1856–1861. IEEE.

von Luxburg, U., Belkin, M., and Bousquet, O. (2008). Consistency of spectral clustering. *The Annals of Statistics*, 36(2):555–586.

Wagner, D. and Wagner, F. (1993). Between Min Cut and Graph Bisection. In *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*, pages 744–750. Springer-Verlag.

Williams, C. and Seeger, M. (2001). Using the Nyström Method to Speed Up Kernel Machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688.

Wu, Z. and Leahy, R. (1993). An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113.

Yan, D., Huang, L., and Jordan, M. I. (2009). Fast approximate spectral clustering. In *Work*, page 907, New York, New York, USA. ACM Press.

Zelnik-Manor, L. and Perona, P. (2004). Self-tuning spectral clustering. *Advances in Neural Information Processing Systems 17*, pages 1601–1608.