Spectral Clustering Survey

HU, Pili*

May 14, 2012†

Abstract

Abstract. Sources can be found in [11].

*hupili [at] ie [dot] cuhk [dot] edu [dot] hk

†Last compile:May 15, 2012

Contents

1 Introduction

Spectral Clustering(SC) was used in several disciplines long ago. For example, computer vision[22]. Spectral Embeding(SE) was also widely discussed in the community[6]. Outside spectral community, the machine learning community also developed many linear or non-linear Dimensionality Reduction(DR) methods, like Principal Component Anslysis (PCA), Kernel PCA (KPCA)[21], Locally Linear Embedding (LLE)[18], etc. Other technique like Multi-Dimensional Scaling(MDS) was successfully used in computational psychology for a very long time[5], which can be viewed as both "embedding" or "dimensionality reduction".

According to our survey, although those methods target at different problems and are derived from different assumptions, they do share a lot in common. The most significant sign is that, the core procedure involves Eigen Value Decomposition(EVD) or Singular Value Decomposition(SVD), aka "spectral". They all involve an intermidiate step of embedding high-dimensional / non-Euclidean / non-metric points into a low-dimensional Euclidean space (although some do not embed explicitly). In this case, we categorize all these algorithms as Spectral Embedding Technique(SET).

1.1 A Sample Spectral Clustering Algorithm

There are many variations of SC. They all work under certain conditions and researchers don't have a rule of thumb so far. Before we analyze their procedure and justification, we present a simple but workable sample algorithm(**Alg 1**).

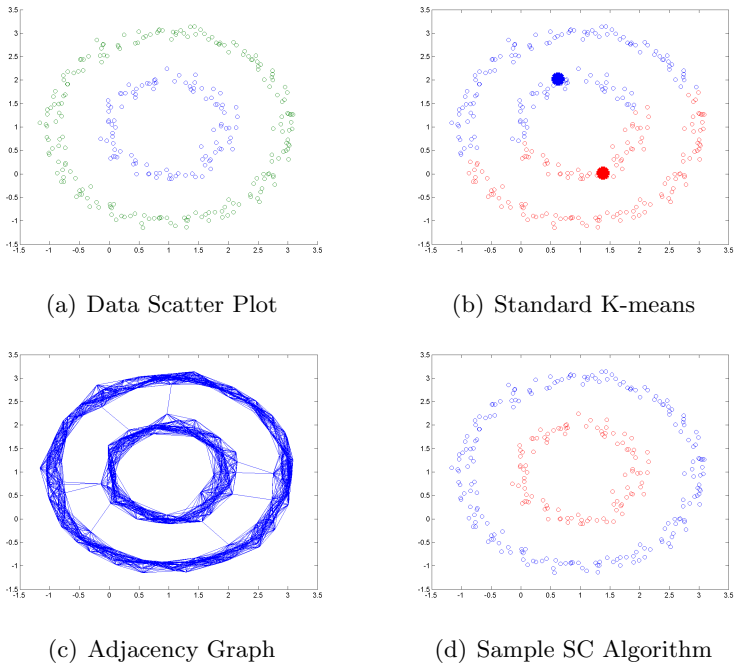
Algorithm 1 Sample Spectral Clustering

Input: Data matrix $X = [x_1, x_2, \ldots, x_N]$; Number of Clusters K.
Output: Clustering $\{C_i\}$: $C_i \in V$ and $\cap_i C_i = \emptyset$ and $\cup_i C_i = V$.

1: Form adjacency matrix A within ϵ-ball.
2: Solve $A = U\Lambda U^{\mathrm{T}}$, indexed according the eigenvalue's magnitude.
3: $Y \leftarrow$ first K columns of U.
4: Cluster Y's rows by K-means.

In **Alg 1**, the ϵ-ball adjacency graph is constructed as follows. First create one vertex for each data point. If for two points i, j satisfy $||x_i - x_j|| < \epsilon$, connect them with an edge. In this simple demonstration, we consider an unweighted graph, i.e. all entries of A are 0(disconnected) or 1(connected).

Fig 1 demonstrates the result of our sample SC algorithm, compared with standard K-means algorithm. **Fig 1(a)** shows the scatter plot of data. It is composed of one radius 1 circle and another radius 2 circle, both centered at (1,1). **Fig 1(b)** shows the result of standard K-means working on Euclidean distance. **Fig 1(c)** shows the graph representation, where the



(a) Data Scatter Plot (b) Standard K-means

(c) Adjacency Graph (d) Sample SC Algorithm

Figure 1: Demonstration of Sample SC Algorithm

adjacency graph is formed by taking a ϵ-ball and $\epsilon = 0.7$ in the example. **Fig 1(d)** shows the output of **Alg 1**. It's obvious that standard K-means algorithm can not correctly cluster the two circles. This is a known major weakness of K-means(in Euclidean): When clusters are not well separated spheres, it has difficulty recovering the underlying clusters. Although K-means works for this case if we transform the points into polar coordinate system(see [11] for code), the solution is not universal. However, in this example, our sample SC algorithm can separate the two clusters, probably because the eigenvectors of adjacency matrix convey adequate information.

A precaution is that **Alg 1** does not always work even in this simple case. Nor have we seen this algorithm from formally published works (so far), let alone justifications. This algorithm is only to show the flavour of spectral clustering and it contains those important steps in other more sophisticated algorithms. Readers are recommended to learn von Luxburg's tutorial[25] before reading the following sections. Since that paper is very detailed, we'll present overlapping topics concisely.

1.2 Spectral Clustering Taxonomy

When the readers start to survey SC related topics, they will soon find that there are two streams of study:

- **Embedding Like**. An example is presented in **Alg 1**. One of the core

procedure is an embedding of points into lower dimensional Euclidean space. After that, hard cut algorithm like K-means is invoked to get the final result. This stream has a lot in common with SE and DR. In the current article, we focus on this line of research.

- **Partitioning Like**. One early example is the 2-way cut algorithm presented by Shi and Malik in[22]. Later Kannan et al. analyzed the framework further[14]. The core procedure of this type of algorithm is a 2-way partitioning subroutine. By invoke this subroutine on resultant subgraph repeatedly, we can get final K clusters. When the subroutine can guarantee certain quality, the global resultant clustering quality can be guaranteed [14]. The attracting aspect of Kannan's framework is that, we can plugin any subroutine as long as they have quality gurantee in one cut. For example, the eigen vector of left normalized graph Laplacian can induce good 2-way cut in terms of the Normalized Cut[22] (**Section 3.2**). This line of research is more close to Spectral Graph Partitioning(SGP), although those algorithms also get the name of Spectral Clustering.

It's worth to note some work of SGP. In Spielman's[23] and Lau's[15] lecture notes, they both presented a graph partitioning algorithm said to result from Lovász[16]. Chung[7] made this framework clear: First define a function $f(v)$ on all vertices; Then set threshold T to cut vertices into two groups, $\{v|f(v) \leq T\}$ and $\{v|f(v) > T\}$. The question now becomes to find a good heuristic function f. Note that the original bi-partition problem is of complexity $O(2^N)$ and the heuristic based bi-partition problem is of $O(N)$(plus the time to get f). If we define f as the second eigen vector of left normalized graph Laplacian, it is just the algorithm of Shi and Malik[22]. Besides, there can be multiple f_i and the best cut can still be searched in polynomial time (given f_i). For example, in [15] Lau used random walk probability $P_1, P_2, \ldots, P_t$ as f. Some recent research of the heuristic function are Personalized Pagerank[1], Evolving Sets[2], etc.

In the rest of this article, we refer Spectral Clustering to the first type, i.e. the embedding like algorithms.

1.3 Linear Algebraic Properties

2 Spectral Clustering Framework

We propose the following framework to absorb currently surveyed variations of SC(and other SET):

1. **Metric Formulation**. This step forms a pairwise metric, upon which an adjacency matrix of (weighted) graph can be constructed. There

are several kinds of input: high-dimensional data (usual case); pairwise proximity input (like MDS, see **Section 4.1**); (weighted) graph (like the input of SGP). If the input of SC is already a graph, this step is omitted. For proximity measures, especially dissmilarity, it is usually first converted to approximte pairwise inner product in Euclidean space. The pairwise inner product is a positive related quantity with similarity(e.g. Jaccard's coefficient for 0/1 vector[26]), and thus suits the notion of weights of graph edges. For high-dimensional data, there are more freedom in the metric formulation stage, like similarity graph[25], geodesic distance[24], etc.

2. **Sepctral Embedding**. With the adjacency matrix built in last stage, this stage embeds all vertices into a lower dimensional Euclidean space. For SC community, this embedding makes clustering structure more clear, so that simple algorithms working in Euclidean space can detect the clusters. For DR community, this embedding reveals the shape of manifolds in their parametric space. The two goals are essentially the same. The core procedure is to do EVD and differences lie before and after EVD:

   - **Matrix Type**. Some authors use graph Laplacian [25, 3, 22] ; others use [17, 6, 14] adjacency matrix.
   - **Normalization**. Both Laplacian and adjacency matrix can be unnormalized, symmetrically normalized, or left(row) normalized[25]. They corresponds to different interpretation and will be explored later.
   - **Scaling**. As is shown in **Alg 1**, we can directly embed vertices using the corresponding row of Y (like [17]). Other alternatives are to scale by square root eigenvalue (like [6]) and scale by eigenvalue (like PCA[4]).
   - **Projection**. For many algorithms, the Y (after scaling) provides an Euclidean space embedding. There are others which further project the rows of Y onto a unit sphere, like [17] and [6].

3. **Clustering**. Based on the embedding result, simple algorithms can be invoked to perform a hard cut on the points. Traditional methods from data mining community are K-means and hierarchical clustering like single/complete linkage[13]. Among those techniques, K-means are the most widely used. A variation of K-means will be proposed later in this article, in order to better fit some angle preserving SET. Simpler hard cut techniques are also possible, e.g. looking at the largest entry of the embedding coordinates[14].

Note that not all of the combinations are justified in published works. We organize them in this way to reveal possibilities from a practitioners

perspective. If some combinations yield good results in practice, we can seek for justifications using tools from spectral graph theory or machine learning.

2.1 Metric Formulation

We rate great importance of metric formulation step, although it is not the main body of SET. There are always many ways of metric formulation given a practical problem. With poorly constructed metric matrix, even the best embedding technique helps little.

2.1.1 High Dimensional Data

For high dimensional data, the following ways can be applied to obtain a graph representation:

- ϵ-ball[25]. If $||x_i - x_j|| < \epsilon$, we connect vertices i, j with an edge.
- k-Nearest-Neighbour(kNN)[25]. For each vertex, we connect it with its k nearest neighbours based on Euclidean distance.
- Mutual kNN(m-kNN)[25]. Note the set of kNN is not symmetric. Multual kNN connects those points who are kNN to each other.
- Complete graph[25]. All vertices are connected with each other. This construction is ususally used with Gaussian kernel weighting below.

After the construction of graph, edges can be weighted in several ways:

- Unweighted[3]. The adjacency matrix is mere 1(connected) or 0(disconnected).
- Gaussian kernel[25], also called heat kernel[3]. Each edge is weighted by $A_{ij} = \exp\{-||x_i - x_j||^2/t\}$, where t is a super parameter controling the decaying rate of similarity.

Heuristics on selecting ϵ, k, t are proposed by many authors, e.g. ch8 of [25], but there is no real rule of thumb. Since the focus of SET study is on the embedding part, most work do not try hard to tweak the construction of similarity graph. We propose other possibilities, which may be helpful to target different practical problems:

- Mahalanobis distance[27]. The connection condition $||x_i - x_j|| < \epsilon$ is substituted by $(x_i - x_j)^{\mathrm{T}}\Sigma^{-1}(x_i - x_j) < \epsilon^2$. Accodingly, when Gaussian kernel is used, the edge weight is given by $A_{ij} = \exp\{-0.5(x_i - x_j)^{\mathrm{T}}\Sigma^{-1}(x_i - x_j)\}$.

- Jaccard's coefficient[26]. It computes the ratio of the intersection and union two sets. It is useful when the high dimensional input coordinates can be interpreted as sets.
- Cosine similarity[28]. It computes the angle between two vectors and is widely used in text mining context.

2.1.2 Proximity

Many real problems has proximity as input. Proximity can be described by similarity or dissimilarity. With pairwise similarity input, we can directly fit the data into following SE procedure. A more interesting problem is to transform dissimilarity into similarity, or at least an equivalent quantity. Decomposing the transformed matrix should be able to yield reasonable embedding(under certain justification).

Denote data matrix by $X = [x_1, x_2, \ldots, x_N]$. Every column x_j corresponds to an n dimensional point. We calculate the pairwise squared distance by $d_{ij}^2 = ||x_i - x_j||^2 = x_i^{\mathrm{T}} x_i + x_j^{\mathrm{T}} x_j - 2x_i^{\mathrm{T}} x_j$. Grouping the N^2 entries into matrix form, we have:

$$D^{(2)} = c\vec{1}^{\mathrm{T}} + \vec{1}c^{\mathrm{T}} - 2X^{\mathrm{T}}X \tag{1}$$

where c is a column vector with $x_i^{\mathrm{T}} x_i$ being the entries. We'll see later (**Section 4.1**) that once a matrix B can be written in the form $B = X^{\mathrm{T}}X$, we have standard ways to decompose it and recover the low dimensional embedding. That is, given dissimilarity measures, we can construct the corresponding inner product form. A standard approach is double centering: [5]

$$J = I - \frac{1}{n}\vec{1}\vec{1}^{\mathrm{T}} \tag{2}$$

$$X^{\mathrm{T}}X = -\frac{1}{2}JD^{(2)}J \tag{3}$$

There are yet two problems left:

- First, not all dissimilarities are distance. One reason is that the real world data is with noise. In such case, random noise will be reduced by SET. Another reason, maybe more frequently seen, is data inconsistency. This is quite often seen in computational psychology when people try to rate the degree of dissimilarity of objects[5]. Those empirical data may break distance laws, like triangle inequality.
- Second, in order to make double centering work, the low dimensional data are required to zero mean, namely $\sum_i x_i = 0$. Actually, we can choose any points as the origin(rather than sample mean), and corresponding forms can be derived. Since our input in this case is

pairwise distance(D or $D^{(2)}$), their is no explicit definition of origin. Or in another word, the effect of embedding is invariant under translation. We can safely locate the embedded sample mean at the origin.

In total, given dissimilarity matrix D, we take the element wise square $D^{(2)}$ and then pass the double centered version $-\frac{1}{2}JD^{(2)}J$ to next stage.

2.1.3 Enhancements

The above sections discussed the basic formulation of an adjacency matrix A. They can be fit direcly into SE procedures. At the same time, people propose some enhancement techniques based on certain assumptions.

Geodesic distance, as is in isomap[24], computes all pair shortest path(geodesic distance) of the adjacency graph first. Then the pairwise geodesic distance is treated as normal distance, D. The standard MDS (**Section 4.1**) can construct an embedding for D. For details, please see **Section 4.2**.

Although we have not found other widely used enhancements in literature, the discussion here do reveal other possibilities. For example, what will be the result if we plug in effective resistance as distance and fit in later SE stage? The effective resistance is closely related with commute time[15]. If it yield good results in practice, justification will not be hard.

2.2 Spectral Embeding

2.3 Clustering

3 Spectral Clustering Justification

3.1 Random Walk

[von]

3.2 Normalized Cut

[shi]

3.3 Normalized Association

[shi]

3.4 Ratio Cut

[von]

3.5 Conductance

[von]

3.6 Matrix Perturbation

[andrew ng]

3.7 Low Rank Approximation

[matthew brand]

3.8 Density Estimation View

[mo chen, 2010]

3.9 Commute Time

[jihun ham, kernel]. view pseudo inverse of graph Laplacian by commute times on graphs.

3.10 Polarization

[m. brand] unifying view...

4 Other Spectral Embedding Technique

4.1 MDS

4.2 isomap

4.3 Laplacian Eigenmap

4.4 Hessian Eigenmap

4.5 PCA

4.6 LLE

4.7 Kernel PCA

4.8 Kernel Framework

4.9 Graph Framework

5 Conclusion

Acknowledgements

References

[1] R. Andersen and F. Chung. Detecting sharp drops in pagerank and a simplified local partitioning algorithm. *Theory and Applications of*

Models of Computation, ¡¡missing¿¿:1–12, 2007.

[2] R. Andersen and Y. Peres. Finding sparse cuts locally using evolving sets. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 235–244. ACM, 2009.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[4] C.M Bishop. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[5] I. Borg and P.J.F. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Verlag, 2005.

[6] M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

[7] F. Chung. Random walks and local cuts in graphs. *Linear Algebra and its applications*, 423(1):22–32, 2007.

[8] S.W. Hadley, B.L. Mark, and A. Vannelli. An efficient eigenvector approach for finding netlist partitions. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 11(7):885–892, 1992.

[9] B. Hendrickson and R. Leland. Multidimensional spectral load balancing. *Report SAND93-0074, Sandia National Laboratories, Albuquerque, NM*, ¡¡missing¿¿:¡¡missing¿¿, 1993.

[10] Pili Hu. Matrix calculus. GitHub, https://github.com/hupili/tutorial/tree/master/matrix-calculus, 3 2012. HU, Pili's tutorial collection.

[11] Pili Hu. Spectral techniques for community detection on 2-hop topology. GitHub, https://github.com/hupili/Spectral-2Hop, 4 2012. course project of CUHK/CSCI5160.

[12] Pili Hu. Tutorial collection. GitHub, https://github.com/hupili/tutorial, 3 2012. HU, Pili's tutorial collection.

[13] H. Jiawei and M. Kamber. *Data mining: concepts and techniques*, volume 5. San Francisco, CA, itd: Morgan Kaufmann, 2001.

[14] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515, 2004.

[15] Lap Chi Lau. Spectral algorithm lecture notes. http://www.cse.cuhk.edu.hk/ chi/csc5160/index.html, 2012.

[16] L. Lovász and M. Simonovits. The mixing rate of markov chains, an isoperimetric inequality, and computing the volume. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 346–354. IEEE, 1990.

[17] A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[18] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[19] L.K. Saul and S.T. Roweis. An introduction to locally linear embedding. Technical report, NYU, 2000.

[20] L.K. Saul and S.T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *The Journal of Machine Learning Research*, 4:119–155, 2003.

[21] B. Schölkopf, A. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

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

[23] DA Spielman. Spectral graph theory lecture notes. http://www.cs.yale.edu/homes/spielman/561/, 2011.

[24] J.B. Tenenbaum, V. De Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[25] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[26] Wikipedia, Jaccard Coefficient, `http://en.wikipedia.org/wiki/Jaccard_index`

[27] Wikipedia, Mahalanobis Distance, `http://en.wikipedia.org/wiki/Mahalanobis_distance`

[28] Wikipedia, Cosine Similarity, `http://en.wikipedia.org/wiki/Cosine_similarity`

Appendix