# AUTOMATIC DETERMINATION OF THE NUMBER OF CLUSTERS USING SPECTRAL ALGORITHMS

*Guido Sanguinetti, Jonathan Laidler and Neil D. Lawrence*

Department of Computer Science
University of Sheffield
211 Portobello Street
Sheffield S1 4DP

## ABSTRACT

We introduce a novel spectral clustering algorithm that allows us to automatically determine the number of clusters in a dataset. The algorithm is based on a theoretical analysis of the spectral properties of block diagonal affinity matrices; in contrast to established methods, we do not normalise the rows of the matrix of eigenvectors, and argue that the non-normalised data contains key information that allows the automatic determination of the number of clusters present. We present several examples of datasets successfully clustered by our algorithm, both artificial and real, obtaining good results even without employing refined feature extraction techniques.

The software used in our experiments is available for download from

`http://www.dcs.shef.ac.uk/~guido/`.

## 1. INTRODUCTION

Spectral clustering has come to the fore in recent years as a powerful approach to a range of clustering problems. Its ability to identify non convex clusters makes it ideal for a number of applications, including image segmentation and speech recognition [1, 2, 3]. Typically spectral clustering (for example see [4]) involves constructing an affinity matrix from the data (which depends on a scale parameter $\sigma$), and requires the prior knowledge of the number of clusters present, as well as some system to choose the distance parameter $\sigma$.

In this paper we present a modified algorithm which does not require the user to know the number of clusters a priori. The algorithm is based on some simple geometric properties of the eigenvectors of the affinity matrix: in particular, we show how the positivity of the affinity matrix implies that the rows of the eigenvectors will cluster along radial lines. This can be exploited, using a modification of the $K$-means algorithm, to detect whether the number of clusters selected is less than the true number, and allows one to iteratively obtain the number of clusters.

The plan of the paper is as follows: in Section 2 we review one of the standard approaches to spectral clustering, that proposed by [4]. In Section 3 we introduce our algorithm, highlighting the crucial differences from [4]. In Section 4 we demonstrate the proposed algorithm on some illustrative data sets.

## 2. SPECTRAL CLUSTERING

Spectral clustering initially was inspired by the concept of graph partitioning [1]. While it can be shown that discrete graph partitioning is an NP hard problem, the continuous relaxation is tractable and reduces the problem to a spectral decomposition of the graph's Laplacian. This approach is elegant and has theoretically sound bases, however it needs to be performed hierarchically to provide more than two groupings, as it can be shown that the error made in the relaxation may become unacceptably high on considering more than one eigenvector.

An alternative point of view, bypassing the graph partitioning altogether and generalising to arbitrary numbers of clusters, was proposed in [4], building on work of [5]. This algorithm will be our main reference in the following and we will refer to it simply as Spectral Clustering.

In essence the Spectral Clustering algorithm is a way of grouping $N$ data points (taken to be $d$-dimensional vectors) into a predefined number of clusters, $K$. The starting point is to construct an *affinity matrix* $A$ from the data, which is an $N \times N$ matrix encoding the distances between the various points. This is often done in practice by fitting an RBF kernel to the data. The affinity matrix is then normalised to form a matrix $L$[1] by conjugating with the diagonal matrix $D^{-\frac{1}{2}}$ which has as entries the square roots of the sum of the rows of $A$. This is to take into account the different

[1] There are several variations in the definition of $L$: some authors prefer to use $I - L$, some others set to zero the diagonal entries in $A$. These differences do not significantly alter the algorithm.

**Algorithm 1** The spectral clustering algorithm suggested by *Ng et al.*.

1. Given a dataset consisting of $N$ $d$-dimensional vectors $X \in \Re^{N \times d}$ to be partitioned in $K$ clusters, construct the affinity matrix

$$A_{ij} = \exp\left[-\frac{|X(i,:) - X(j,:)|^2}{\sigma^2}\right]$$

and normalise it to obtain

$$L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}},$$

where $D = diag\left(\sum_{j=1}^{d} A_{ij}\right)$ is the diagonal matrix whose entries are the sum of the rows of $A$. $\sigma$ is a distance parameter which can be thought intuitively as how closely we are looking at the point configuration; it is usually set by the user, although automated procedures such as discrete searches have been proposed [4].

2. Compute the first $K$ eigenvectors of $L$ (the ones with the largest eigenvalues), assemble them in an $N \times K$ matrix $\hat{Y}$ and construct another matrix $Y$ by normalising the rows of $\hat{Y}$.

3. Perform $K$-means (or whatever other clustering algorithm) on the matrix $Y$ (treating each row as a data vector).

spread of the various clusters (points belonging to more rarified clusters will have lower sums of the corresponding row of $A$). It is straightforward to prove that $L$ is positive definite and has eigenvalues smaller or equal to 1, with equality holding in at least one case.

The first $K$ eigenvectors are then computed and arranged as columns in a matrix $\hat{Y}$. The rows of $\hat{Y}$ are then normalised and treated as $K$-dimensional vectors; performing $K$-means on these vectors will return the desired clustering. The algorithm is summarised in algorithm 1.

The reasoning behind this algorithm is very simple. Consider the case when the clusters are widely separated. The matrix $L$ will then be (with the rows ordered by cluster) block diagonal with $K$ blocks. We noticed before that $L$ has at least one eigenvector with eigenvalue 1; in this case, it is clear that it will have exactly one eigenvector with eigenvalue 1 for each block. Therefore, each row in the matrix $Y$ will have exactly one entry equal to 1, all the others being zero, and the data in this $K$-dimensional space will cluster on the unit length vectors on the coordinate axis. We will call the $K$ eigenvectors with eigenvalue 1 the *clustering eigenvectors* and the space they span the *clustering space.*

In general, there will be a rotational ambiguity associated with the selection of the clustering eigenvectors: as $K$ eigenvectors correspond to the eigenvalue 1, any set of mutually orthogonal vectors in the $K$ dimensional clustering space will equally well qualify as an admissible basis of clustering eigenvectors. Therefore, one can only conclude that the rows of the eigenvectors will cluster upon mutually orthogonal points on the $K$ dimensional sphere, rather than on the coordinate axes.

Ng *et al*. in [4] developed a matrix perturbation theory argument to explain why this algorithm might work in more general cases, when the clusters are not widely separated and the matrix $L$ is not block diagonal. Their argument relates the validity of the algorithm with the existence of a large "eigengap", i.e. the $K + 1$th eigenvalue of $L$ needs to be significantly less than 1 to guarantee stability of the selected eigenspace under perturbations.

## 3. AUTOMATIC DETERMINATION OF THE NUMBER OF CLUSTERS

Let us consider again the limiting case of tight, widely separated clusters. We can obtain useful insight into the eigendecomposition of the affinity matrix by recalling that finding the eigenvector corresponding to the largest eigenvalue of a symmetric matrix $A$ is equivalent to the following optimisation problem

$$\max\left(\mathbf{v}^T A \mathbf{v}\right) \text{ subject to } \mathbf{v}^T \mathbf{v} = 1.$$

It is then apparent that, if the matrix $A$ has all positive entries, the entries of the eigenvector $\mathbf{v}$ will all have the same sign (the magnitude of the individual entries will depend on the detailed structure of the cluster).

Therefore, in the case of a block diagonal affinity matrix, the rows of the matrix that has as columns the clustering eigenvectors will cluster along $K$ mutually orthogonal axes in $K$ dimensional space.

This suggests that the normalisation in step 2 of algorithm 1, which placed all the clusters on the unit sphere in the clustering space, might not be necessary after all.

Furthermore, one might ask what happens if the number of clusters selected, $K$, is smaller than the number of clusters present in the dataset. Taking the first $q < K$ eigenvectors, we will be selecting a $q$ dimensional subspace in the clustering space, whose position will in general bear no relation to the clusters.

As the rows of the $K$ eigenvectors clustered along mutually orthogonal vectors, their projections in a lower dimensional space will cluster along radial directions. The general picture will therefore be of $K$ clusters elongated in the radial direction, with possibly some clusters very near the origin (when the subspace is orthogonal to some of the discarded eigenvectors).

**Algorithm 2** Elongated $K$-means algorithm

1. Initialise $K$ centres $\mathbf{c}_{1..K}$ in the data space.
2. For each centre $\mathbf{c}_i$ compute the distance of all points $\mathbf{x}$ from it as follows:
   - if $\mathbf{c}_i^T \mathbf{c}_i > \epsilon$, i.e. if the centre is not very near the origin ($\epsilon$ is a parameter to be fixed by the user),
     $$\text{e-dist}\,(\mathbf{x}, \mathbf{c}_i) = (\mathbf{x} - \mathbf{c}_i)^T M (\mathbf{x} - \mathbf{c}_i)$$
     where $M = \frac{1}{\lambda}\left(I_q - \frac{\mathbf{c}_i \mathbf{c}_i^T}{\mathbf{c}_i^T \mathbf{c}_i}\right) + \lambda \frac{\mathbf{c}_i \mathbf{c}_i^T}{\mathbf{c}_i^T \mathbf{c}_i}$.
     $\lambda$ is the *sharpness* parameter that controls the elongation (the smaller, the more elongated the clusters).
   - If the centre is very near the origin, $\mathbf{c}_i^T \mathbf{c}_i < \epsilon$, the distances are measured using the Euclidean distance.
3. Using this distance measure, assign each point $\mathbf{x}$ to the nearest centre. Update the location of each centre by taking the mean of all the data assigned to it.
4. Return to step 2 and repeat until there is no change in the location of the centres.

Given the elongated nature of the clusters, we modified the $K$-means algorithm in order to downweight distances along radial directions and penalise distances along transversal directions. The *elongated $K$-means* algorithm differs from a standard $K$-means method in that it computes the distances of point $\mathbf{x}$ from the centre $\mathbf{c}_i$ in a different way. This difference is highlighted in algorithm 2

In this way, if a centre is within a cluster, all the points in the cluster will be very near to it, while points in another cluster (i.e. along another radial direction) will be judged to be further from that centre than from the origin.

This clustering algorithm is illustrated on a toy dataset in figure 1. Red plus signs and green diamonds give the clusters found by elongated $K$-means. The blue circles are the means found by ordinary $K$-means. Notice that $K$-means assigns no point to the mean initialised at the origin. The solid ellipse is the curve at elongated distance 1 from the centre of the first cluster; the dashed circle is Euclidean distance 1 from the origin. Clearly the points in the diamond cluster have smaller Euclidean distance from the origin than elongated distance from the mean of the red pluses. Therefore, initialising one centre in the pluses cluster and the other centre in the origin, elongated $K$-means gets the desired result.

Therefore, one can exploit this to construct an algorithm that computes the number of clusters. Having computed

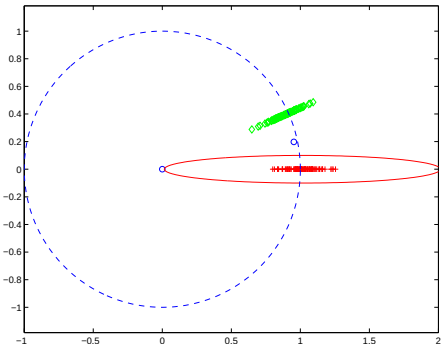


**Fig. 1**. An illustration of elongated $K$-means. Red plus signs and green diamonds give the clusters found by elongated $K$-means. The blue circles are the means found by ordinary $K$-means. The solid ellipse is the curve at elongated distance 1 from the centre of the first cluster; the dashed circle is Euclidean distance 1 from the origin.

**Algorithm 3** Cluster detecting algorithm

1. Form matrix $L$ as in step 1 of algorithm 3. Set $q = 2$.
2. Compute $q$ eigenvectors with greatest eigenvalues. Arrange them in a matrix $Y$.
3. Perform elongated $K$-means with $q+1$ centres on $Y$, initialising the $q+1$-th mean in the origin.
4. If the $q+1$-th cluster contains any data points, then there must be at least an extra cluster; set $q = q+1$ and go back to step 2. Otherwise, end algorithm, output clustered data and number of clusters.

the $L$ matrix as in the Spectral Clustering algorithm 1, one can start by considering only the first two eigenvectors[2]. In this two dimensional space, one will expect $K$ clusters along radial directions. Initialise elongated $K$-means with two centres corresponding to data points in different clusters and one centre in the origin. The algorithm then will drag the centre in the origin towards one of the clusters not accounted for. Compute another eigenvector (thus increasing the dimension of the clustering space to three) and repeat the procedure. Eventually, when you reach as many eigenvectors as the number of clusters present in the data[3], no points will be assigned to the centre at the origin, leaving the last cluster empty. This is the signal to terminate the algorithm. This algorithm is summed up in algorithm 3.

Notice that the rotational ambiguity does not affect the

[2] One might consider starting with only one dimension, but the number of radial directions available in one dimension is too limited for our purposes.

[3] This clearly depends on the setting of the parameters $\sigma$ and $\lambda$.

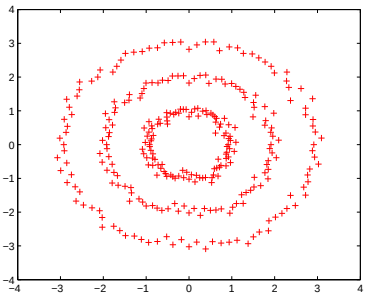


**Fig. 2**. The three circles data set

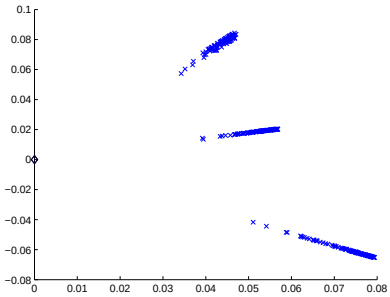


**Fig. 3**. The 2D clustering space: plot of the two dominant eigenvectors for the three circles data set. Notice the presence of three elongated clusters on approximately radial directions.

elongated nature of the clusters, except in the case in which one cluster will end up in the origin (in which it will be assigned to the spherical cluster in the origin).

## 4. IMPLEMENTATION AND EXPERIMENTAL RESULTS

One of the key characteristics of the $K$-means clustering algorithm is its sensitivity to initialisation. We exploit this in step 4 of algorithm 3: if there are no points nearer the origin than the mean of any other cluster, $K$-means will simply return an empty cluster. Therefore, our algorithm is also sensitive to the initialisation of $K$-means. We always initialise the first $q$ means to be points in a cluster identified at the previous step, while the $q+1$th mean is in the origin.

The value of the distance parameter $\sigma$ is empirically chosen somewhere between the intercluster average distance and the intracluster average distance. The sharpness parameter has been fixed to 0.2 in all experiments.

To illustrate the novel features of the algorithm we will work through a case study, clustering the three circles data set. This data set, shown in figure 2, is a good example of when spectral methods are appropriate for clustering, as it consists of non-convex clusters which cannot be separated directly using $K$-means or similar clustering algorithms.

A value of 0.05 was used for $\sigma$ in this example; this was empirically found to produce tight clusters in the clustering space. Initially we set $q = 2$, and thus take just the two eigenvectors of L with largest eigenvalues. This gives us a clustering space which is two-dimensional, visualised in figure 3.

We can see that there are three clear clusters, each lying approximately on a line that passes through the origin. It is this observation that we exploit to find the number of clusters. Recall that our initialisation of the $K$-means is crucial, so we initialise the means as follows. The first mean $\mathbf{c}_1$ is initialised at the data point that is furthest from the origin. The second mean $\mathbf{c}_2$ is initialised at the data point that simultaneously maximises its norm while minimising the dot product with $\mathbf{c}_1$. As we know that there will be at least two clusters in the plane, this guarantees that the second mean is set at a point in a different cluster.

Then we add a third mean $\mathbf{c}_3$ at the origin. Because we are going to do an elongated $K$-means clustering, each mean is considered closer to points that lie along the same radial line than to points that lie off this line. For this reason we find that the first two means will not easily be moved away from the two clusters they started in.

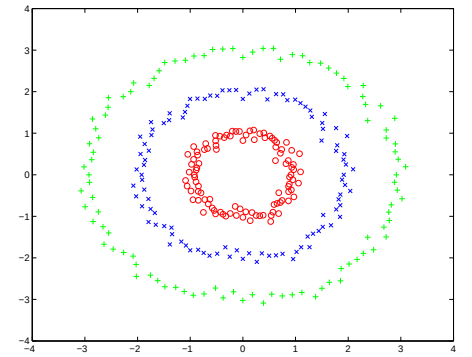
However, as $\mathbf{c}_3$ is set in the origin, distances from it will be measured using the standard Euclidean distance, and this will mean that the points of the third cluster will be assigned to it (as their Euclidean distance from the origin is smaller than their elongated distance from another cluster).

The consequence of this is that $\mathbf{c}_3$ gets dragged towards the third cluster, and we achieve the clustering we desired, as each of the clusters in the clustering space of figure 3 corresponds to one of the concentric circles in figure 2.

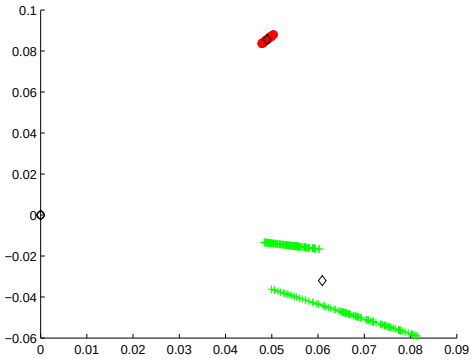
When iterating the algorithm further, all three clusters will have a mean vector initialised at one of their points. Therefore, the assignment rule of $K$-means will force the fourth mean $\mathbf{c}_4$ (initialised at the origin) to have no points assigned to it. This will be taken as the termination signal.

The final result for this data set is shown in figure 4(a). In order to demonstrate the advantages of applying elongated $K$-means rather then a standard implementation of $K$-means, we draw attention to figure 4(b). In this case $\sigma^2$ has been set less optimally so that the clusters appear closer together in the clustering space. Elongated $K$-means is still able to separate these by virtue of them being in different radial lines, but a standard $K$-means clustering groups two of the clusters together.

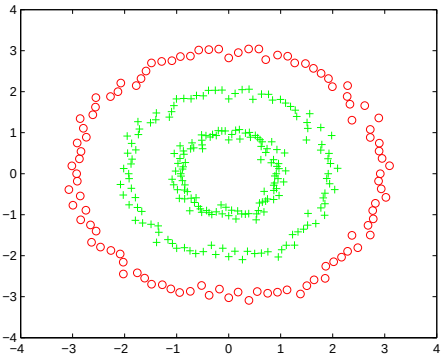
Figure 5 shows two more examples of non-convex clustering tasks, taken from the implementation examples in [4].



(a)

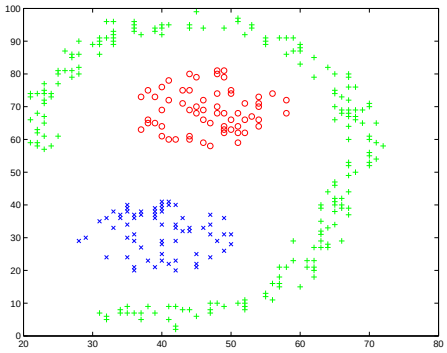


(b)

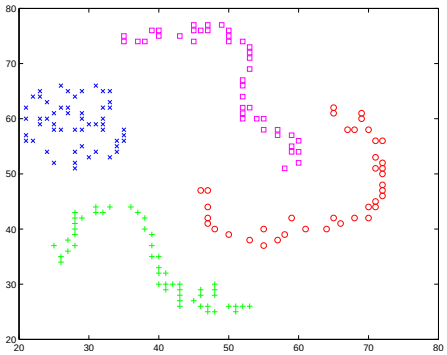


(c)

**Fig. 4**. Panel (a) shows the clustering output for the Three Circles data set. Points belonging to different clusters are shown using different symbols. (b) shows what would happen if we had attempted to cluster with standard $K$-means. Two of the radial clusters are grouped together, resulting in the wrong clustering, as illustrated in (c).



(a)

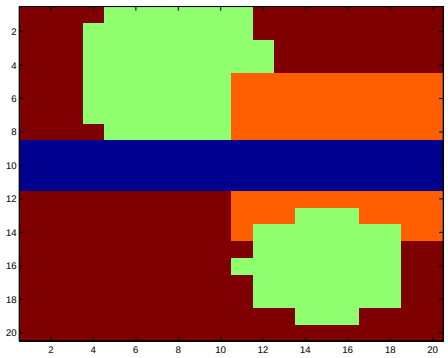


(b)

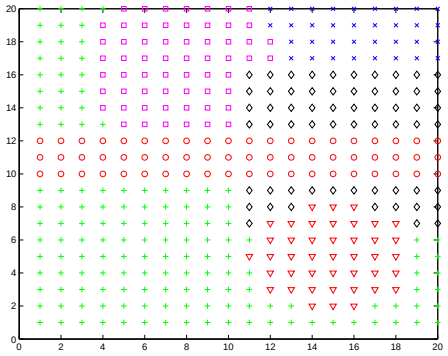
**Fig. 5**. Two more examples of non-convex clusters successfully clustered by our algorithm. In (a) the parameter $\sigma$ was set to 1, while in (b) it was set to 0.5. The number of clusters was determined automatically. The different clusters found are indicated by using different symbols and colours.

In figure 6 an artificially constructed image is segmented. The image was turned into greyscale and the intensity was then scaled to have the same range as the spatial coordinates. The input points were then treated as three dimensional vectors, the third coordinate being given by the intensity. The experiment shows how, for $\sigma = 2$, the algorithm grouped the points by intensity only, while for $\sigma = 1$ more detailed structure was picked up. This is not surprising, as the number of clusters present in a data set depends on $\sigma$.

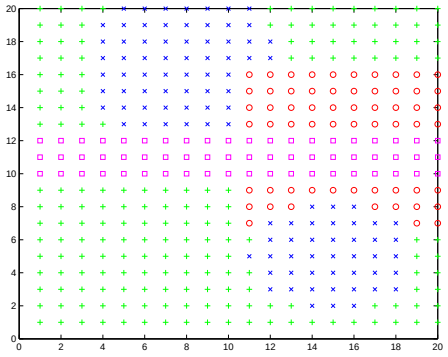
It should be pointed out that modern image segmentation techniques use very refined feature extraction tools as preprocessing steps. These may involve texture information filters for images [1]. It is interesting, though, that, even without any feature extraction, such a simple algorithm manages to produce somewhat sensible groupings.



(a)



(b)



(c)

**Fig. 6**. Simple image segmentation example. The image in (a) was turned to greyscale and then clustered with our algorithm. In (b) the parameter $\sigma$ was set to 1, while in (c) it was set to 2. As it might be expected, the number of clusters found in (b) was greater. The groupings found are indicated by using different symbols and colours.

## 5. DISCUSSION

We analysed the standard Spectral Clustering algorithm, as proposed in [4]. By considering the spectral properties of the affinity matrix, we suggested a modification of the algorithm that exploits some of these properties in order to achieve an automatic selection of the number of clusters present in a dataset.

While the algorithm is largely heuristic, it does seem to perform very well on a range of examples, reliably producing the expected number of clusters in the case of non-convex clusters. It also gives reasonable results when applied to more difficult artificial examples of image segmentation.

**Acknowledgments** GS and NDL acknowledge support from a BBSRC award 'Improved processing of microarray data with probabilistic models'.

## 6. REFERENCES

[1] Jianbo Shi and Jitendra Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[2] Guy J. Brown and Martin P. Cooke, "Computational auditory scene analysis," *Computer Speech and Language*, vol. 8, pp. 297–333, 1994.

[3] Francis R. Bach and Michael I. Jordan, "Blind one-microphone speech separation: A spectral learning approach," in *Advances in Neural Information Processing Systems*, Cambridge, MA, 2005, vol. 17, MIT Press.

[4] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, Thomas G. Dietterich, Sue Becker, and Zoubin Ghahramani, Eds., Cambridge, MA, 2002, vol. 14, MIT Press.

[5] Yair Weiss, "Segmentation using eigenvectors: A unifying view," in *International Conference on Computer Vision*, 1999.