Denoising Spectral Clustering Through Latent Data Decomposition

Guided Research Report

Stephan Rabanser
Department of Informatics
Technical University of Munich
rabanser@in.tum.de

Oleksandr Shchur
Department of Informatics
Technical University of Munich
shchur@in.tum.de

Stephan Günnemann
Department of Informatics
Technical University of Munich
guennemann@in.tum.de

ABSTRACT

Spectral Clustering is a popular clustering technique, which is especially effective for complex shaped data. One drawback however is its high sensitivity to noise in the data. We present a new perspective on this issue by modeling the observed data as a latent decomposition of pure data and corrupting noise. This decomposition can be learned either by relying on external information about the quality of specific edges in the similarity graph or by directly learning the latent spectral embedding in a noise-robust fashion. Our tests on synthetic and real data show promising improvements in clustering quality over other related approaches in many instances. However, both algorithms still offer room for improvement in order to improve general robustness properties.

1 INTRODUCTION

Clustering is considered one of the most fundamental tasks in data analytics/mining an machine learning [1, 14]. The core objective of clustering is to divide a given dataset into subgroups such that points belonging to the same subgroup are similar and points belonging to different subgroups are dissimilar. In machine learning, the clustering problem is also commonly referred to as one of the most prominent examples of latent variable estimation (unsupervised learning), whose main goal is to find hidden structure in the data.

Since the concept of clustering itself cannot be associated with one single algorithm, a number of different approaches have been developed over the past century [6]. Each of these algorithms has its own set of strengths and weaknesses, relies on different data assumptions, and should be chosen depending on the task at hand.

One of the most prominent and successful approaches to clustering to date is called *Spectral Clustering* [19]. The key idea of spectral clustering is to construct an affinity graph of the data (in most cases, this corresponds to a N-nearest-neighbor construction), whose spectrum (set of eigenvalues), is then used to transform the data to a new vector space in which the data is easier to cluster. The actual clustering is then performed in this new vector space by using the widely known K-means algorithm [10].

However, despite its wide applicability, simplicity, and importance to data clustering, one key limitation of spectral clustering is rarely addressed: it is highly sensitive to noisy data, which means that even small amounts of noise can lead to large performance hits in clustering quality. Since data collected in real world applications is rarely free from noise, the development of robust methods plays a key role in current machine learning research and hence also requires noise-tolerant clustering techniques.

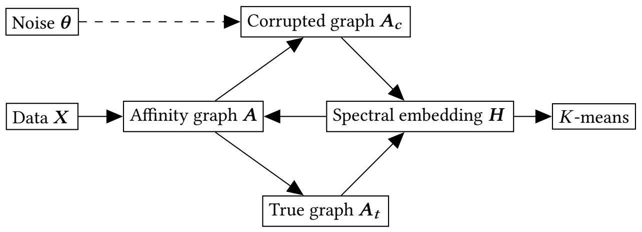

Figure 1: Proposed model from RSC [3]. This model assumes true data, but tries to account for noise through a latent decompsition of the affinity graph $A = A_t + A_c$, where the noise is indirectly captured as part of A_c.

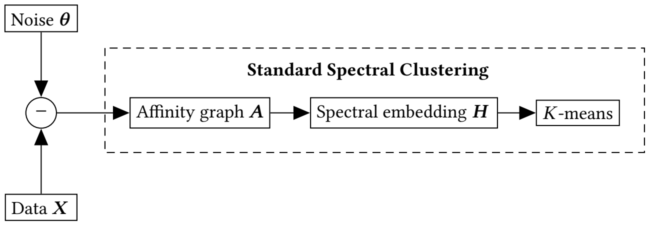

Figure 2: Proposed model of this paper. This model assumes assumes corrupted data $X = X_p + \theta$ consisting of pure data and noise. After removing the noise, standard spectral clustering can be used.

While there are different ways of modeling data corruption for clustering purposes, we intend to directly formalize our natural intuition of decomposing the observed data into the (pure) data and the corrupting noise. As we will see in Section 3, lots of current robust spectral clustering techniques do not model the noise as being part of the data, but rather as being part of the affinity graph (see example of such an algorithm in Figure 1). While this difference might seem minor at first sight, it accounts for a number of different problems. Not only does this not directly correspond to the basic intuition, but it also requires careful adaptations of the clustering algorithm as a whole. In contrast, we model the noise as being part of the observed data (see Figure 2). Therefore, we aim at learning this latent decomposition before the affinity graph is created. This formulation more accurately corresponds to the original problem we want to solve and also allows for standard (spectral) clustering algorithms to be carried out on the pure data.

The remainder of this paper is structured as follows: we start by introducing a few formal preliminary concepts in Section 2. Then, we discuss related works in Section 3. Our algorithmic solution to the data-noise decomposition, being the main contribution, is then outlined in Section 4. Consequently, we present a few experimental results and discuss them as part of Section 5. We close with a brief summary of the main results and an outlook for future work in Section 6.

2 PRELIMINARIES

2.1 Notation

Large non-bold letters, like $I \in \mathbb{N}$, denote integer-valued variables. We use N and D to denote the number of observations and the dimensionality of the data respectively; the number of clusters is denoted by K. Matrices are denoted by large bold letters, like $\boldsymbol{M} \in \mathbb{R}^{I \times J}$, where $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ denotes the observed data matrix and $\boldsymbol{\theta} \in \mathbb{R}^{N \times D}$ the noise matrix. The transposition of a matrix $\boldsymbol{M}$ is denoted by $\boldsymbol{M}^T$. Entries of a matrix $\boldsymbol{M} \in \mathbb{R}^{I \times J}$ are denoted m_{ij}, referring to the entry at the i-th row and the j-th column. The Hadamard product between two matrices $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ is denoted by $\boldsymbol{M}_1 * \boldsymbol{M}_2$. The row-wise maximum of a matrix $\boldsymbol{M}$ can be reduced to a vector $\boldsymbol{m}$ and denoted by

$$\boldsymbol{m} = \max_{\text{per row}} \boldsymbol{M} = \begin{bmatrix} \max\{m_{11}, m_{12}, \ldots, m_{1N}\} \\ \max\{m_{21}, m_{22}, \ldots, m_{2N}\} \\ \vdots \\ \max\{m_{N1}, m_{N2}, \ldots, m_{NN}\} \end{bmatrix}.$$

Note that the row-wise minimum is defined analogously.

2.2 Spectral Clustering

As already briefly introduced in Section 1 and thoroughly described in [19], Spectral Clustering can be broken down into three successive tasks:

(1) Construct the affinity graph of $\boldsymbol{X}$ represented as a symmetric adjacency matrix $\boldsymbol{A} \in \mathbb{R}_{\geq 0}^{N \times N}$. This can either be a fully connected (dense) or a reduced (sparse) graph. While dense graphs naturally come with the advantage of extracting the entire similarity information from the underlying data, they are also more intensive to compute compared to reduced graphs, which only consider local neighborhoods. Quantifying the similarity of two points can be done using kernel functions (the exponential or squared exponential kernels are popular choices) or classical distance metrics (the L_1 or L_2 norm are popular choices). A specific entry in $\boldsymbol{A}$, a_{ij}, will then either correspond to the computed similarity value between data point i and j, i.e. $a_{ij} \in \mathbb{R}_{\geq 0}$, or a binary value indicating whether the two points are similar or dissimilar, i.e. $a_{ij} \in \{0, 1\}$.

(2) Compute the Laplacian matrix $\boldsymbol{L}(\boldsymbol{A})$ and its K first eigenvectors. We introduce two popular Laplacian choices: the unnormalized Laplacian matrix $\boldsymbol{L}_{\text{un}}(\boldsymbol{A})$ can be computed as

$$\boldsymbol{L}_{\text{un}}(\boldsymbol{A}) = \boldsymbol{D}(\boldsymbol{A}) - \boldsymbol{A} \tag{1}$$

where $\boldsymbol{D}(\boldsymbol{A}) = \text{diag}(d_1, ..., d_n)$ with $d_i = \sum_j a_{ij}$ corresponds to the degree matrix of $\boldsymbol{A}$. In contrast, the (symmetric) normalized Laplacian matrix $\boldsymbol{L}_{\text{sym}}(\boldsymbol{A})$ can be computed as

$$\boldsymbol{L}_{\text{sym}}(\boldsymbol{A}) = \boldsymbol{D}(\boldsymbol{A})^{-\frac{1}{2}} \boldsymbol{L}_{\text{un}}(\boldsymbol{A}) \boldsymbol{D}(\boldsymbol{A})^{-\frac{1}{2}}. \tag{2}$$

Independent of the choice for the Laplacian type $\boldsymbol{L}(\boldsymbol{A}) = \boldsymbol{L}_{\text{un}}(\boldsymbol{A})$ or $\boldsymbol{L}(\boldsymbol{A}) = \boldsymbol{L}_{\text{norm}}(\boldsymbol{A})$, it is our goal to minimize the ratio-cut in the affinity graph $\boldsymbol{A}$. An approximation to this optimizing the ratio-cut/normalized-cut, can be obtained by the following trace minimization problem

$$\min_{\boldsymbol{H} \in \mathbb{R}^{N \times K}} \text{Tr}(\boldsymbol{H}^T \boldsymbol{L}(\boldsymbol{A}) \boldsymbol{H}) \quad \text{with} \quad \boldsymbol{H}^T \boldsymbol{H} = \boldsymbol{I} \tag{3}$$

whose optimal solution is given by the first K eigenvectors of $\boldsymbol{L}(\boldsymbol{A})$ stored as columns in $\boldsymbol{H}$. Note that similar minimization problems can be formulated for normalized Laplacians, yet still the optimal $\boldsymbol{H}$ is still given by the first K eigenvectors.

(3) Cluster based on $\boldsymbol{H}$. The spectral embedding of each instance i is given by the i-th row of $\boldsymbol{H}$. The final clustering result can then be obtained by executing K-means on $\boldsymbol{H}$.

2.3 Robust Spectral Clustering (RSC)

In addition to standard spectral clustering, we are also introducing the key concept behind *Robust Spectral Clustering (RSC)* [3], as we make use of RSC as part of our proposed algorithms.

Unlike Spectral Clustering, RSC is able to handle noisy data by modeling the assumption of a similarity graph corrupted by noise. A key assumption of RSC is the fact that $\boldsymbol{A}$ can be decomposed into a true $\boldsymbol{A}$, denoted $\boldsymbol{A}_t$, and a corrupted $\boldsymbol{A}$, denoted $\boldsymbol{A}_c$, where

$$\boldsymbol{A} = \boldsymbol{A}_t + \boldsymbol{A}_c \quad \text{where} \quad \boldsymbol{A}_t, \boldsymbol{A}_c \in \{0, 1\}^{N \times N}, \text{symmetric}. \tag{4}$$

Further, RSC assumes that corruptions in the graph are relatively rare ($\boldsymbol{A}_c$ is sparse), that the noise can be bounded by a maximum noise value τ, that each node in $\boldsymbol{A}_t$ is connected to at least M, and that the latent decomposition in Equation (4) can be learned jointly with the spectral clustering process. The latter implies, that RSC can directly rely on the Spectral Clustering objective to learn $\boldsymbol{A}$ and $\boldsymbol{A}_t/\boldsymbol{A}_c$. The objective is then formally given by

$$(\boldsymbol{H}^*, \boldsymbol{A}_t^*) = \arg\min_{\boldsymbol{H}, \boldsymbol{A}_t} \text{Tr}(\boldsymbol{H}^T \boldsymbol{L}(\boldsymbol{A}_t) \boldsymbol{H}) \quad \text{with} \quad \boldsymbol{H}^T \boldsymbol{H} = \boldsymbol{I} \tag{5}$$

where $||\boldsymbol{A} - \boldsymbol{A}_t||_0 \leq 2\tau$ and $||\boldsymbol{a}_{t,i}||_0 \geq M$ with $i \in \{1, \ldots, N\}$.

RSC therefore tries to optimize the spectral embedding $\boldsymbol{H}$ and the true similarity graph $\boldsymbol{A}_t$ jointly. This can be done using an alternating block coordinate-descent optimization scheme. Intuitively speaking, RSC therefore learns which nearest neighbor connections it should keep and which ones it should drop. As we will see in Section 4, we can use this feature of RSC to our advantage.

The general process of RSC is shown in Figure 1.

3 RELATED WORK

Before we start with a detailed treatment of this paper's main contributions, we would like to shed some light on some of the related research work going on in the field of robust clustering methods. Note that a vast majority of the presented approaches rely on the assumption of corrupted affinity graphs.

Noise-robust spectral clustering techniques. Initial works on robust methods for spectral clustering focused on improving the affinity matrix through local similarity scaling which aims at determining the used similarity measures per data instance [21]. Extending on this idea, [11] proposes to further introduce weighting on these individual scalings. [9] explores smoothing of the Laplacian matrix in order to capture the noise present in the underlying data, but requires a full eigendecomposition of the Laplacian. [13] considers the presence of uniform noise in the data and proposes a data wrapping technique which transforms noisy points into a separate cluster distinct from the true observations. Yet another idea for handling corruption is the detection and elimination of noisy features [2, 8, 22]. As already discusses, [3] extracts good/bad neighbor information from the similarity graph and jointly learns the true graph and the corresponding spectral embedding. Further ideas on making Spectral Clustering more robust are presented in [4, 12, 15, 20].

Other noise-robust clustering techniques. Recently, [18] presented a robust clustering technique called RCC which, just like our algorithms, intends to move the data points in such a way that they naturally coalesce into distinct clusters. While RCC does not directly make use of Spectral Clustering, it still tries to extract and optimize a lower-dimensional latent embedding in combination with the clustering through linear least-squares.

4 PROPOSED ALGORITHMS

In contrast to many other works, we do not consider a corrupted similarity graph, but instead model the problem of noisy observations. Concretely, we assume that the data we observe, denoted $\boldsymbol{X}$, is corrupted by noise and hence can be decomposed into the pure data, denoted $\boldsymbol{X_p}$, and the corrupting noise, denoted $\boldsymbol{\theta}$, which can be related using the following equation:

$$\boldsymbol{X} = \boldsymbol{X_p} + \boldsymbol{\theta}. \tag{6}$$

Our ultimate goal is to learn the noise matrix $\boldsymbol{\theta}$, such that we can extract $\boldsymbol{X_p}$ from our observed data as follows

$$\boxed{\boldsymbol{X_p} = \boldsymbol{X} - \boldsymbol{\theta}.} \tag{7}$$

Having obtained the pure data, we can then use ordinary Spectral Clustering.

Note that our initial estimation for $\boldsymbol{\theta}$ will be the zero matrix $\boldsymbol{0}$. In order to learn the latent decomposition from Equation (6), we propose two different algorithms: Repeated Robust Spectral Clustering (RRSC) and Direct Embedding Optimization (DEO).

4.1 Repeated Robust Spectral Clustering (RRSC)

The first algorithm we want to propose to learn the latent decomposition defined in (6), is called *Repeated Robust Spectral Clustering (RRSC)*. As the name already suggests, this algorithm repeatedly applies RSC updates in order to good clusterings. As we have learned in Subsection 2.3, RSC can provide us with estimations of the corrupted and true edges, $\boldsymbol{A_c}$ and $\boldsymbol{A_t}$, of the similarity graph $\boldsymbol{A}$. While $\boldsymbol{A_t}$ already reflects parts of the true complete graph, $\boldsymbol{A_c}$ gives us the edges which are noisy.

4.1.1 Objective. Our main goal is to adapt the edges in $\boldsymbol{A_c}$ in such a way that they are no longer considered corrupted, thereby denoising the complete graph. At the same time, we can make use of $\boldsymbol{A_t}$ to stabilize the adaption of the edges stored in $\boldsymbol{A_c}$. In order to do this, we first need to formalize the notion of degree of similarity. This can be done by constructing a similarity graph based on $\boldsymbol{X_p}$. We propose to construct a full similarity graph using the Euclidian distance as a distance metric, which assigns small values to close (similar) and large value to distant (dissimilar) data points. In order to avoid confusion with other similarity matrices, we denote this similarity matrix by $\boldsymbol{S}$. As $\boldsymbol{X_p}$ is derived from a latent decomposition (see Equation (7)), $\boldsymbol{S}$ has a dependence on the noise $\boldsymbol{\theta}$. In the following we will reference to $\boldsymbol{S}$ using $\boldsymbol{S_\theta}$ to reflect this dependence. Using $\boldsymbol{S_\theta}$ in combination with the similarity matrices from RSC, we can now quantify the similarity for each (corrupted and true) edge.

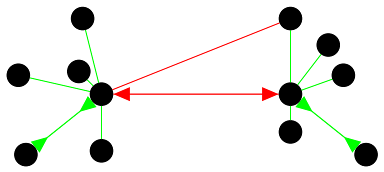

Figure 3: Our objective is to maximize the smallest corrupted edge (red arrowed edge) and to minimize the largest true edge (green arrowed edge).

Ideally, we want to push points connected through corrupted edges further apart. At the same time, it is advisable to let the true edges of a point that is moved due to a corrupted edge guide the point to its correct position. This prevents the point from moving too far, potentially even beyond the structure we want to detect. We therefore propose a local optimization that can be done for each point which is connected to any other point through a corrupted edge. To increase stability, we consider only moving the closest (best) corrupted edge and the most distant (worst) true edge. This procedure is depicted in a simple example in Figure 3. Both edges can be easily computed for all data points and stored as vectors $\boldsymbol{a_{c,b}}, \boldsymbol{a_{t,w}} \in \mathbb{R}^N$ respectively.

$$\begin{aligned} \boldsymbol{a_{c,b}} &= \min_{\text{per row}} [(1 - \boldsymbol{A_c}) * \sigma + \boldsymbol{S_\theta}] \\ \boldsymbol{a_{t,w}} &= \max_{\text{per row}} [\boldsymbol{A_t} * \boldsymbol{S_\theta}] \end{aligned} \tag{8}$$

While selecting the worst true edges is rather simple, selecting the best corrupted edges is more challenging as we want to choose the smallest value larger than zero and zero if there is no value larger than zero. This can be done by inverting the binary values in $\boldsymbol{A_c}$, scaling the remaining entries by a large constant σ and then adding $\boldsymbol{S_\theta}$.

Finally, we aim at minimizing the difference between $\boldsymbol{a_{t,w}}$ and $\boldsymbol{a_{c,b}}$, which implicitly tries to increase the values in $\boldsymbol{a_{c,b}}$ and to decrease the values in $\boldsymbol{a_{t,w}}$. It is important to stress the fact that these operations are only performed as part of the optimization process when the point in question actually is connected to an other point via a corrupted edge. Further, we do not allow the loss to be negative. If both limitations would not be imposed (i.e. if a point is only connected to other points via true edges), many true edges would also be adapted on a continuous basis, which might cause instability.

As we want to capture the noise as part of $\boldsymbol{\theta}$, we can tune $\boldsymbol{\theta}$ such that this difference is minimized, thereby transferring the noise from $\boldsymbol{X}$ to $\boldsymbol{\theta}$. This leads to the following formalized objective function:

$$\boxed{\mathcal{L}_{\boldsymbol{\theta}} = \max\{0, \boldsymbol{a_c}^T(\boldsymbol{a_{t,w}} - \boldsymbol{a_{c,b}})\} \quad \text{where} \quad \boldsymbol{a_c} = \max_{\text{per row}} \boldsymbol{A_c}.} \tag{9}$$

$\mathcal{L}_{\boldsymbol{\theta}}$ can then be minimized by using gradient descent with respect to $\boldsymbol{\theta}$. Once $\mathcal{L}_{\boldsymbol{\theta}}$ is sufficiently minimized, we can then subtract our current noise estimation off the data and execute the same steps again. The entire algorithm stops after a fixed amount of total iterations.

As we originally assumed, just like RSC, that corruptions in the data are sparse and small, we further limit the values in $\boldsymbol{\theta}$ to a fixed maximum value, denoted $\theta_{\max}$. Omitting this restriction could mislead the algorithm into overestimating noise values. While this parameter can be considered a heuristic, we achieved good results by setting it to 10% of the maximum value observed in $\boldsymbol{X}$.

4.1.2 Normalizing $\boldsymbol{S_\theta}$. While Equation 9 formalizes our intuition, large absolute differences in the values of $\boldsymbol{S_\theta}$ might lead to algorithmic instability. That is why we propose to use the normalized entries of $\boldsymbol{S_\theta}$ as part of our objective. Accounting for this change, $\boldsymbol{a}_{t,w}$ and $\boldsymbol{a}_{c,b}$ turn into

$$\begin{aligned} \boldsymbol{a}_{c,b} &= \min_{\text{per row}} [(1 - \boldsymbol{A}_c) * \sigma + \hat{\boldsymbol{S}}_{\boldsymbol{\theta}}] \\ \boldsymbol{a}_{t,w} &= \max_{\text{per row}} [\boldsymbol{A}_t * \hat{\boldsymbol{S}}_{\boldsymbol{\theta}}] \end{aligned} \tag{10}$$

with $\hat{\boldsymbol{S}}_{\boldsymbol{\theta}} = \text{normalize}(\boldsymbol{S_\theta})$ referring to the normalizing process in which the largest value in $\boldsymbol{S_\theta}$ is set to 1 in $\hat{\boldsymbol{S}}_{\boldsymbol{\theta}}$, the smallest value is set to 0, and all other values are scaled respectively.

4.1.3 Early stopping. Due to the nature of RSC, corrupted edges are also detected in situations where the clustering is already considered good or even ideal (i.e. the NMI is close to 1). As this might lead to points being moved which we would consider as non-corrupted, this behavior can lead to sudden changes in $\boldsymbol{S_\theta}$ which in turn badly affects the clustering process. Since clustering is an unsupervised setting, we do not have access to the cluster assignments during training. As a consequence, we are not able to compute the NMI after every epoch to assess the performance of our model without resorting to ground truth. Therefore, an additional measure is needed to assess the performance of the model given its current parameters.

Based on this problem, we propose to stop the algorithm when RSC only detects corrupted distances which are almost 0 in the embedding space $\boldsymbol{H}$. This happens when the spectral embedding shows highly distinct, concentrated clusters, usually leading to good clusterings.

4.1.4 Objective aggressivity. While the presented objective function performs well on low-dimensional data with small to medium amounts of data points, only optimizing the relationships between minimum and maximum edges can be considered too weak as too little movement is generated. Therefore we further propose a more aggressive objective by averaging distances over true and corrupted edges. Hence, the objective turns into

$$\boxed{\mathcal{L}_{\boldsymbol{\theta}} = \max\{0, \boldsymbol{a}_c^T(\boldsymbol{a}_t - \boldsymbol{a}_c)\} \quad \text{where} \quad \boldsymbol{a}_c = \max_{\text{per row}} \boldsymbol{A}_c} \tag{11}$$

with

$$\boldsymbol{a}_c = \operatorname*{avg}_{\text{per row}} [\boldsymbol{A}_c * \boldsymbol{S_\theta}] \qquad \text{and} \qquad \boldsymbol{a}_t = \operatorname*{avg}_{\text{per row}} [\boldsymbol{A}_t * \boldsymbol{S_\theta}]. \tag{12}$$

A more formal definition of RRSC (using the min/max loss from Equation (10)) is described in Algorithm 1.

```
Read data into $X$;
Define $\theta_{\max}, \sigma, K$;
$\boldsymbol{\theta} \leftarrow \mathbf{0}$;
forall epochs do
    $X_p \leftarrow X - \boldsymbol{\theta}$;
    $A_c, A_t, H \leftarrow \text{RSC}(X_p, K)$;
    $a_{c,H} \leftarrow \text{edges\_in\_embedding}(H, A_c, L_1)$ ;
    if $\max(a_{c,H}) \leq 10^{-15}$ then
        break;
    end
    $S \leftarrow \text{affinity\_graph}(X_p, L_2)$;
    $\hat{S} \leftarrow \text{normalize}(S)$ ;
    $a_c \leftarrow \max(A_c, axis = 0)$;
    $a_{g,w} \leftarrow \max(A_t * \hat{S}, axis = 0)$;
    $a_{c,b} \leftarrow \min((\mathbf{1} - A_c) * \sigma + \hat{S}, axis = 0)$;
    $\mathcal{L} \leftarrow \max\{0, \text{transpose}(a_c) \cdot (a_{g,w} - a_{c,b})\}$;
    forall gradient descent iterations do
        Execute gradient step to minimize $\mathcal{L}$ with respect to $\boldsymbol{\theta}$;
        $\theta \leftarrow \max(\theta, \theta_{\max}) \quad \forall\, \theta \in \boldsymbol{\theta}$;
    end
end
Execute spectral clustering on $X - \boldsymbol{\theta}$;
```

Algorithm 1: The RRSC algorithm.

4.2 Direct Embedding Optimization (DEO)

While we already presented an attractive idea of how to learn noisy clusters with Repeated Robust Spectral Clustering, RRSC only serves as an approximation to a higher goal we would like to achieve: learning the spectral embedding directly from noisy data instead of relying on external information about true and corrupted edges. Recall the basic workings of Spectral Clustering from Subsection 2.2. The main objective is given by the trace minimization problem in Equation (3). Our goal in *Direct Embedding Optimization (DEO)* is to optimize this trace value by tuning the noise matrix $\boldsymbol{\theta}$ accordingly.

Again, we first construct a full similarity matrix $\boldsymbol{S_\theta}$ of the data. In contrast to RRSC however, we intend to assign large values to close (similar) and small values to distant (dissimilar) data points. Note that we will normalize the distances, similar to RRSC. If desired, the resulting graph can further be reduced to a K-nearest-neighbor graph by selecting the top-K values from $\boldsymbol{S_\theta}$ per row. After computing the degree matrix from $\boldsymbol{S_\theta}$, we can then derive both the unnormalized and the (symmetric) normalized Laplacian matrices. Recall that the spectral embedding is then given by the first K eigenvectors of the Laplacian, i.e. the eigenvectors corresponding to the K smallest eigenvalues. The final objective function is then given by

$$\boxed{\mathcal{L}_{\boldsymbol{\theta}} = \text{Tr}(\boldsymbol{H}^T \boldsymbol{L}(\boldsymbol{S_\theta}) \boldsymbol{H}).} \tag{13}$$

$\mathcal{L}_{\boldsymbol{\theta}}$ can then be minimized by using gradient descent with respect to $\boldsymbol{\theta}$. Once $\mathcal{L}_{\boldsymbol{\theta}}$ is sufficiently minimized, we can then subtract our current noise estimation off the data to execute spectral clustering on the pure data. Note that the maximum noise value is again capped by $\theta_{\max}$.

A more formal definition of DEO is described in Algorithm 2.

```
Read data into $\boldsymbol{X}$;
Define $\theta_{\max}$, reduce, $K$;
$\boldsymbol{\theta} \leftarrow \mathbf{0}$;
$\boldsymbol{X}_p \leftarrow \boldsymbol{X} - \boldsymbol{\theta}$;
$\boldsymbol{S} \leftarrow$ affinity_graph($\boldsymbol{X}_p, L_2$);
$\hat{\boldsymbol{S}} \leftarrow$ normalize($\boldsymbol{S}$) ;
$\hat{s} \leftarrow e^{-\hat{s}} \quad \forall \hat{s} \in \hat{\boldsymbol{S}}$ ;
if reduce then
    $\hat{\boldsymbol{S}} \leftarrow$ top_k($\hat{\boldsymbol{S}}, K$);
end
$\boldsymbol{D} \leftarrow$ degree($\hat{\boldsymbol{S}}$);
$\boldsymbol{L}_{\text{un}} \leftarrow \boldsymbol{D} - \hat{\boldsymbol{S}}$;
$\boldsymbol{L}_{\text{sym}} \leftarrow \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{L}_{\text{un}} \boldsymbol{D}^{-\frac{1}{2}}$;
$\boldsymbol{e}, \boldsymbol{V} \leftarrow$ eigen_decomp($\boldsymbol{L}_{\text{sym}}$) ;
$\boldsymbol{H} \leftarrow$ first_columns($\boldsymbol{V}, K$) ;
$\mathcal{L} \leftarrow \text{Tr}(\boldsymbol{H}^T \boldsymbol{L}_{\text{sym}} \boldsymbol{H})$;
forall gradient descent iterations do
    Execute gradient step to minimize $\mathcal{L}$ with respect to $\boldsymbol{\theta}$;
    $\theta \leftarrow \max(\theta, \theta_{\max}) \quad \forall \theta \in \boldsymbol{\theta}$;
end
Execute spectral clustering on $\boldsymbol{X} - \boldsymbol{\theta}$;
```

Algorithm 2: The Direct Embedding Optimization algorithm.

Table 1: The datasets we used for assessing the performance of our models.

Type	Name	N	D	Classes	Source
Synthetic	Two moons	600	2	2	[17]
	Three moons	900	2	3	[3]
Real	Iris	150	4	3	[7] [5]
	Banknote	1372	4	2	[5]
	Pendigits	7494	16	10	[5]
	USPS	9298	256	10	[16]

5 EXPERIMENTAL RESULTS

5.1 Datasets and general setup

In order to adequately assess the performance of both algorithms presented in Section 4, we carried out a couple of tests on synthetic and real world data. The used datasets and their key properties are defined in Table 1. As a general performance measure for the clustering quality, we report the Normalized Mutual Information (NMI) for these data sets. In terms of runtime, we observed quadratic runtime in the number of data points for both algorithms.

Note that both algorithms are trained via gradient descent with a learning rate of 0.01 and a maximum of 1000 steps per epoch. The overall results are given in Table 2.

5.2 RRSC

The main RRSC results are given in Figure 4 and Figure 5. Note that for the two moons dataset we make use of Equation (10), while we make use of Equation (11) for real world data sets as the respective loss functions.

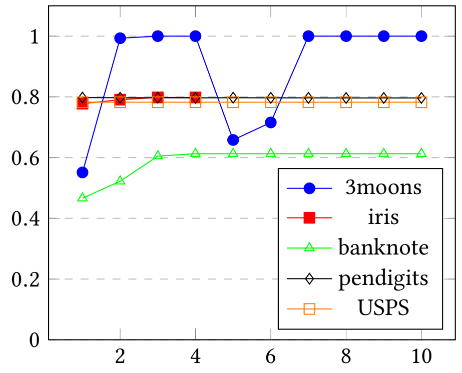

Figure 4: NMI scores for different data sets with the RRSC algorithm over 10 epochs. We can see that RRSC is clearly able to learn the latent noise in the data since it achieves higher NMIs.

Table 2: Overall best NMI results on the datasets from Table 1.

Dataset	SC	RSC	NRSC	AHK	RRSC	DEO
Three moons	0.55	**1.0**	0.99	0.57	**1.0**	0.55
Iris	0.78	0.80	0.79	0.53	0.80	**0.86**
Banknote	0.46	**0.61**	0.47	0.52	**0.61**	**0.61**
Pendigits	0.80	0.80	**0.83**	0.80	0.80	0.80
USPS	0.78	**0.85**	0.83	0.77	0.78	0.78

As one can easily observe from Figure 4, the final NMI is either better or the same as the initial estimation from Spectral Clustering and closely aligns with the RSC result. However, we also noticed some examples in which RRSC was not able to improve the NMI score, but instead showed degrading performance. In order to address this, we looked at a couple of different additional metrics, such as

- the amount of change in the similarity matrix $\boldsymbol{S}_{\boldsymbol{\theta}}$
- minimum, maximum, and average corrupted edge lengths in the embedding space
- and minimum, maximum, and average corrupted edge lengths in the data space.

Sadly, we were not able to find a strong correlation between any of these metrics and the reported NMI. While some metrics often provided good estimates for potentially stopping the algorithm early, they also failed in other situations with only slightly different data.

5.3 DEO

The main DEO results are given in Figure 6. The results reported so far give a mixed overall picture. Most interestingly, the results reported on real world data show good NMI values, but DEO seems to fail on the moons datasets.

In order to further study this behavior, we considered

- switching between unnormalized and normalized Laplacian matrices

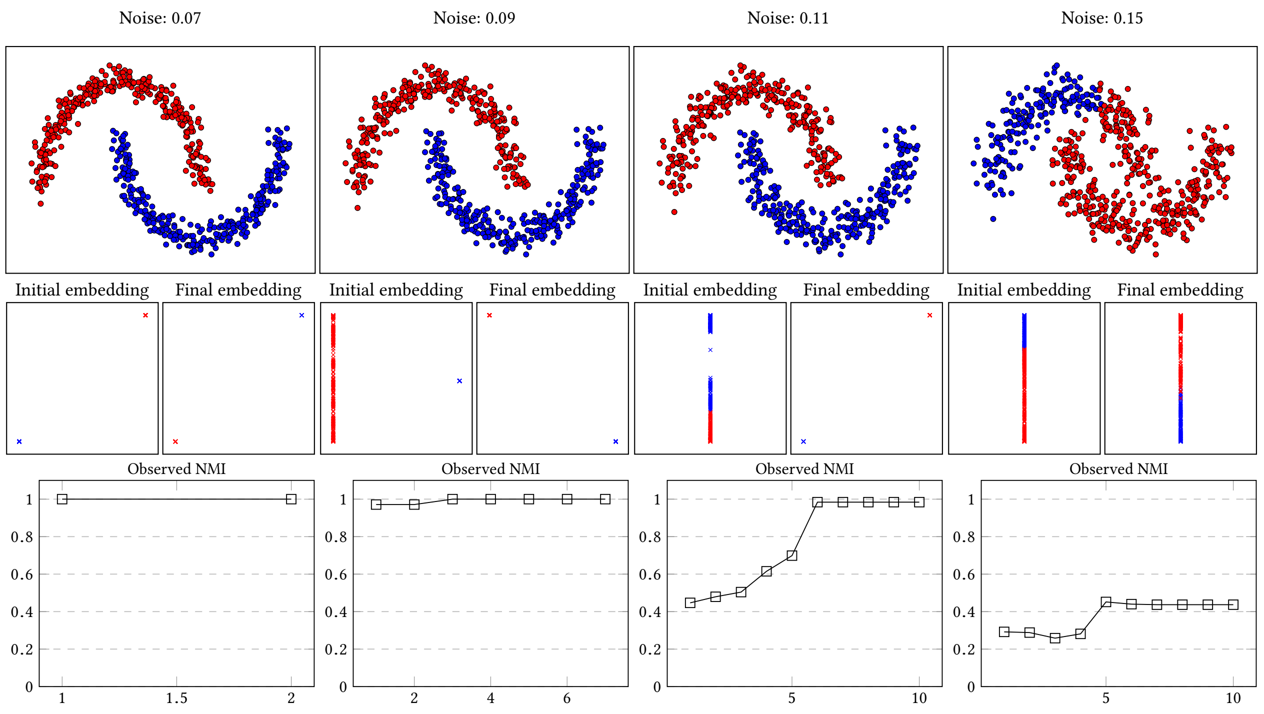

Figure 5: RRSC applied to the two moons dataset (600 datapoints) with different noise values over a maximum of 10 global iterations. The first row shows the final Spectral Clustering result on the learned pure data X_p, while the second row shows the evolvement of the spectral embedding H. The last row quantifies the performance by showing the NMI evolvement. While higher noise values naturally also pose a challenge to RRSC, we can see that RRSC is able to handle fair amounts of noise pretty well.

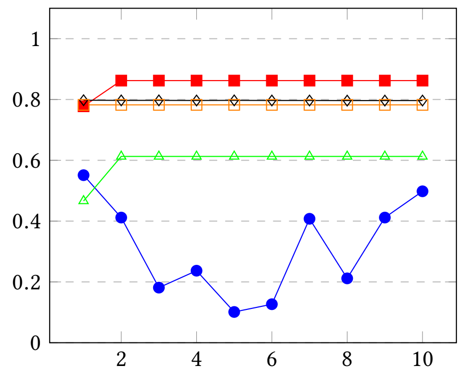

Figure 6: NMI scores for different data sets with the DEO algorithm with $10 \cdot 1000$ gradient descent steps. DEO performs well on real world data, but relatively bad on the moons dataset.

- using different ranges for the eigenvectors selected from the Laplacian
- and fixing $\boldsymbol{H}$ between epochs via discrete updates.

So far, none of these adaptations however provided consistent improvements to the algorithm. In addition, we encountered numerical issues in some instances, which we were not able to fully eliminate.

6 CONCLUSIONS AND FUTURE WORK

As part of this paper, we have introduced a new approach of decomposing observed noisy data into pure data and corrupting noise. We have introduced two algorithms which are able to learn this latent decomposition. Repeated Robust Spectral Clustering (RRSC) achieves this by relying on the good and bad edges information provided by RSC in order to move points connected by a corrupted edge further apart while at the same time keeping the points relatively close to their true neighborhoods. As part of RRSC, we have presented two different loss functions which either provide smooth but slow improvements or faster bust less stable ones. In contrast, Direct Embedding Optimization (DEO) aims at learning the spectral embedding directly from the data.

While the results provided by RRSC already outperform popular spectral clustering techniques in most cases, we still need to further explore additional early stopping criterions for RRSC. Also, we will need to revisit the DEO approach as it still shows inconsistent results at the current point in time. However, we are confident that the general direction for this idea is promising and will further investigate algorithmic optimizations to this approach in the future. At the moment, both algorithms are not making use of sparse matrices, which we want to incorporate to improve runtime and memory usage.

REFERENCES

[1] Charu C Aggarwal and Chandan K Reddy. 2013. *Data clustering: algorithms and applications*. CRC press.

[2] Francis R Bach and Michael I Jordan. 2004. Learning spectral clustering. In *Advances in neural information processing systems*. 305–312.

[3] Aleksandar Bojchevski, Yves Matkovic, and Stephan Günnemann. 2017. Robust Spectral Clustering for Noisy Data: Modeling Sparse Corruptions Improves Latent Embeddings. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 737–746.

[4] Hong Chang and Dit-Yan Yeung. 2008. Robust path-based spectral clustering. *Pattern Recognition* 41, 1 (2008), 191–203.

[5] Dua Dheeru and Efi Karra Taniskidou. 2017. UCI Machine Learning Repository. (2017). http://archive.ics.uci.edu/ml

[6] BS Everitt, S Landau, M Leese, and D Stahl. 2011. Cluster analysis. 2011. *Arnold, London* (2011).

[7] Ronald A Fisher. 1936. The use of multiple measurements in taxonomic problems. *Annals of human genetics* 7, 2 (1936), 179–188.

[8] Stephan Gunnemann, Ines Farber, Sebastian Raubach, and Thomas Seidl. 2013. Spectral subspace clustering for graphs with feature vectors. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 231–240.

[9] Hao Huang, Shinjae Yoo, Hong Qin, and Dantong Yu. 2011. A robust clustering algorithm based on aggregated heat kernel mapping. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 270–279.

[10] Anil K Jain. 2010. Data clustering: 50 years beyond K-means. *Pattern recognition letters* 31, 8 (2010), 651–666.

[11] Wanzeng Kong, Sanqing Hu, Jianhai Zhang, and Guojun Dai. 2013. Robust and smart spectral clustering from normalized cut. *Neural Computing and Applications* 23, 5 (2013), 1503–1512.

[12] Xi Li, Weiming Hu, Chunhua Shen, Anthony Dick, and Zhongfei Zhang. 2014. Context-aware hypergraph construction for robust spectral clustering. *IEEE Transactions on Knowledge and Data Engineering* 26, 10 (2014), 2588–2597.

[13] Zhenguo Li, Jianzhuang Liu, Shifeng Chen, and Xiaoou Tang. 2007. Noise robust spectral clustering. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 1–8.

[14] James MacQueen and others. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1. Oakland, CA, USA, 281–297.

[15] Benjamin A Miller, Michelle S Beard, Patrick J Wolfe, and Nadya T Bliss. 2015. A spectral framework for anomalous subgraph detection. *IEEE Transactions on Signal Processing* 63, 16 (2015), 4191–4206.

[16] Sam Roweis. 2016. Data for MATLAB hackers. (2016). https://cs.nyu.edu/~roweis/data.html

[17] scikit-learn developers. 2017. Sklearn Datasets Make Moons. (2017). http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html

[18] Sohil Atul Shah and Vladlen Koltun. 2017. Robust continuous clustering. *Proceedings of the National Academy of Sciences* 114, 37 (2017), 9814–9819.

[19] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.

[20] Xiaoqian Wang, Feiping Nie, and Heng Huang. 2016. Structured doubly stochastic matrix for graph based clustering: Structured doubly stochastic matrix. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1245–1254.

[21] Lihi Zelnik-Manor and Pietro Perona. 2005. Self-tuning spectral clustering. In *Advances in neural information processing systems*. 1601–1608.

[22] Xiatian Zhu, Chen Change Loy, and Shaogang Gong. 2014. Constructing robust affinity graphs for spectral clustering. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 1450–1457.