Unsupervised Similarity Learning for Spectral Clustering

Anonymous authors
Paper under double-blind review

Abstract

Spectral clustering has been popularized due to its ability to identify non-convex boundaries between individual clusters. However, it requires defining a similarity metric to construct the Laplacian matrix. Instead of predefining this metric upfront, we propose to learn it by finding the optimal parameters of a kernel function. This learning approach parameterizes the data topology by optimizing a similarity function that assigns high similarity values to a pair of data that share discriminative features and vice versa. While some existing approaches also learn the similarity values, they rely on hyperparameters to do so. However, these hyperparameters cannot be validated in an unsupervised setting. As a result, suboptimal hyperparameter values can lead to detrimental performance. To circumvent this drawback, we propose a method that eliminates the need for hyperparameters by learning the optimal parameter for a similarity metric used in spectral clustering. This enables unsupervised learning of the similarity metric while performing spectral clustering. The method's capability is verified on several benchmark datasets with a large scale of non-convexity. Our method outperforms SOTA approaches on accuracy and normalized mutual information measures up to 10% when applied to popular image and text datasets.

1 Introduction

Spectral clustering (SC) (von Luxburg, 2007) is a very effective method to cluster data with non-convex separation boundaries. In this approach, the individual data points can be considered as nodes of a fully connected graph where the edge weights are the similarity values. The underlying assumption for SC is that for every pair of data in the training set, the similar ones should have a high similarity score and vice versa. Thus, the objective for SC becomes minimizing the sum of edge weight between clusters and maximizing the sum within clusters (cf. fig. 1).

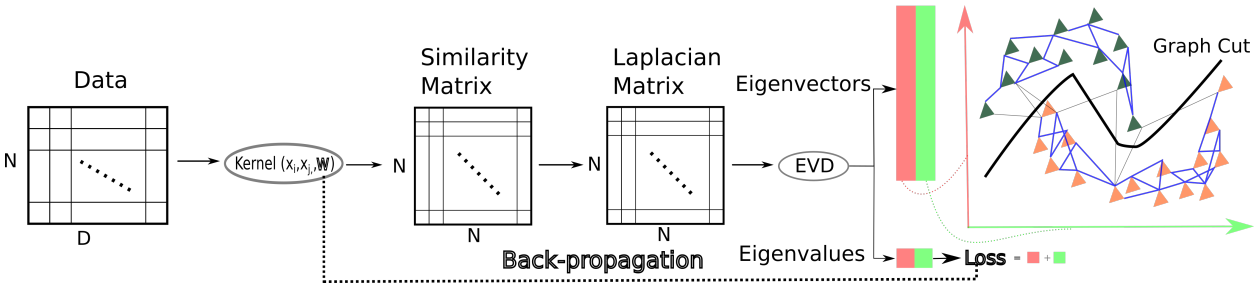


Figure 1: Learning the parameters of the kernel function requires the optimal graph cut (GC) in the connected graph defined by the given data. The sum of the first K eigenvalues is a numerical assessment of the graph-cut (GC), which is minimized via gradient descent.

The performance of SC is primarily attributed to the manually chosen parameters of the *similarity metric*. However, an incorrectly chosen similarity metric can be detrimental to the clustering performance as it might not emphasize enough the similar pairs relative to the dissimilar ones (cf. fig. 2c). Hence, instead of hand-picking the parameters of the similarity metric, one can try to learn instead (cf. fig. 1).

Methods like Kang et al. (2017) aim to learn the similarity metric by defining an objective function. However, it features one or more hyperparameters in the objective function. Unlike supervised learning, validating hyperparameter values through an annotated validation set is not possible in an unsupervised learning setup. Given the need for an extensive exploratory and exploitative search in supervised hyperparameter optimization, it is unlikely that optimal parameters will be selected due to the suboptimal range of values being considerably larger than the optimal range in real-world scenarios.

If we apply the same principle to an unsupervised setting, randomly selecting hyperparameters would likely result in suboptimal values. This, in turn, would result in incorrect similarity scores affecting the downstream clustering performance (Fan et al.). To mitigate this drawback, we propose learning a parameterized similarity metric without introducing any hyperparameter. To do so, *we assume the similarity metric to be a radial basis function (RBF) kernel and try to find its optimal bandwidth.* Since the RBF kernel projects the data into the infinite-dimensional space (Shashua, 2009), its bandwidth enables modulation of the most important dimensions. Moreover, the parameter search space reduces from the number of every possible data pair (when no assumption is made) to the number of parameters in the kernel function.

Finding the optimal parameter of the similarity metric involves minimizing the graph cut (GC) of a connected graph (cf. fig. 1). Numerical computation of the GC directly on the graph is possible only through the Laplacian matrix (von Luxburg, 2007). We propose using the sum of the first K eigenvalues from the Laplacian matrix as an empirical assessment of the GC (cf. fig. 1) where K is the number of clusters. In this study, the initial number of clusters (i.e., K) is predetermined; however, the unsupervised estimation of the cluster count is possible by identifying density peaks within the data topology, as demonstrated in Rodriguez & Laio (2014). We find the optimal parameter of the kernel using gradient descent is possible by back-propagation through the eigenvalue decomposition (EVD) (cf. fig. 1).

Therefore, this method suggests an unsupervised approach to learn similarities that determines the optimal weights for the edges of a fully connected graph, on which SC is then applied. We assessed the clustering performance of our method on various image and text datasets, demonstrating a consistent accuracy improvement over alternative approaches.

The contributions of this work are the following.
1. We introduce a novel unsupervised approach to similarity learning aimed at optimizing spectral clustering.
2. This technique has been tested and proven to provide improved clustering performance on standard benchmark datasets for images and texts.

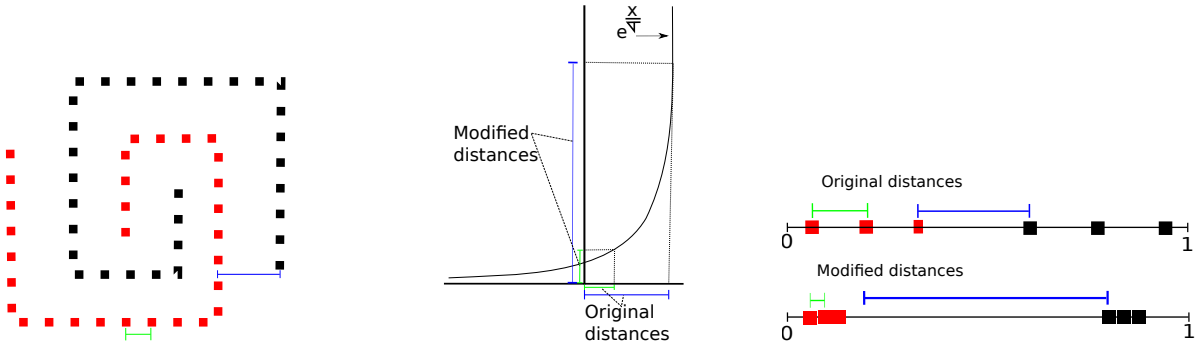
The proposed method learns the parameter governing the similarity metric and mitigates the need for auxiliary hyperparameters. By learning instead of handpicking the parameters of the similarity metric one can guarantee a better data-driven clustering performance.

2 Method

The proposed method works directly on the raw data (e.g., data with the topology as in fig. 2a) and tries to further emphasize any existing separation within the data points (cf. fig. 1). Accordingly, the optimal similarity function would lower the smaller pairwise distances so that all the data points would eventually collapse to a single point (cf. fig. 2c). At the same time, the more pronounced distances are amplified (cf. fig. 2c). To do so, we choose to exponentiate these pairwise distances in a controlled manner (cf. eq. (1) and fig. 2b). The smaller the distance between two data points, the higher their similarity and vice versa. As a result, the bigger distances are exponentiated towards higher values much more than the smaller ones (cf. fig. 2b). Since the new transformed distances are normalized within a unit vector, increasing the bigger magnitudes would render the smaller ones towards even smaller values and vice versa (cf. fig. 2c).

The RBF kernel enables a controlled exponential modulation of the distances via its bandwidth σ as in eq. (1).

$$K(x,y) = e^{\frac{-d(x,y)}{\sigma^2}} = e^{\frac{-|x-y|^2}{\sigma^2}} = \sum_{n=0}^{\infty} \frac{(-|x-y|^2)^n}{(\sigma^2)^n n!} \tag{1}$$



(a) (b) (c)

Figure 2: From the given data in fig. 2a, the gap between each cluster is amplified. To amplify the gaps between each cluster, every distance is modulated using the exponential function of the RBF kernel fig. 2b. As an effect of this, bigger distances are made even bigger, and smaller distances are made even smaller after normalization, as shown in fig. 2c. Upon normalization to a unit vector, the magnitude of smaller distances is reduced, whereas that of larger ones is increased fig. 2c.

The popularity of RBF is attributed to its ability to map data to infinity-dimensional space (Shashua, 2009). By properly tuning the RBF bandwidth σ, one can smoothly truncate the infinite-dimensional expansion (cf. eq. (1)) and select the dimensions to be incorporated in the comparison. In other words, σ adjusts the amount of modulation for the similarity metric. Using the RBF kernel as a generic similarity formulation, the task reduces to optimization of σ that yields the most representative distance from a general formulation. To amplify the distance $d(x, y)$ as presented in eq. (1), the parameter σ must be set to a value less than one (i.e., $\sigma \leq 1$) where $\sigma = 0$ is not allowed. However, to ensure the convexity for the problem at hand, the search space for the parameter σ is reduced to the range of values $\Sigma = (0, 1/2]$. *Traditionally, in a spectral clustering setting, σ is a hyper-parameter which is handpicked. Instead, we propose learning σ directly from the given data.*

2.1 Learning operation

In SC, the objective remains identical to other clustering techniques; similar data are projected together while the dissimilar ones are far apart (von Luxburg, 2007). This objective is equivalent to the GC (cf. fig. 1) when data are considered nodes of the connected graph while the edges are the pair similarity values. Thus, to construct an objective function that is amenable to the gradient-based method, one has to consider the data projections (f) and the similarity values ($A_{i,j} \in R, \forall i, j \in \{1, .., N\}$) simultaneously. Utilizing the general formulation for GC (von Luxburg, 2007) (cf. eq. (2)) as the objective function, it is possible to get an empirical assessment of clustering performance.

$$\mathrm{GC}[A, f] = \sum_{i,j,j<i} A_{i,j}(f_i - f_j)^2 = f^T L f. \tag{2}$$

Where L in eq. (2) is the Laplacian matrix. The formulation in eq. (2) reduces the task to learning the similarity matrix ($A_{[N,N]}$) and the representation of the data (f). As a result, the computation of GC on the graph reduces to matrix multiplication. Since the utilized similarity is parameterized solely by the RBF bandwidth σ, we optimize the computed similarity matrix by tuning σ (cf. eq. (3)).

$$\arg\min_{\sigma,f} \mathrm{GC}[A(\sigma), f(A)] \rightarrow \frac{\partial^2 \mathrm{GC}[A(\sigma), f]}{\partial \sigma \partial f} = 0. \tag{3}$$

Instead of finding both σ and f simultaneously, an alternation between each parameter is adopted (cf. algorithm 1). We utilize a Lagrangian multiplier to restrict the domain of the data representation f to a unit magnitude $ff^T = 1$.

$$\frac{\partial\big\{\mathrm{GC}[A(\sigma), f] + \lambda(f^T f - 1)\big\}}{\partial f} = 0 \rightarrow Lf = \lambda f \tag{4}$$

From eq. (4), one can derive f and λ as the eigenvalue (λ) and eigenvectors (f) of the Laplacian matrix L. Since the RBF kernel is symmetric (cf. eq. (1)), the resulting Laplacian matrix L is also symmetric upon which EVD produces real and non-negative eigenvalues ($\lambda_i \geq 0, \forall i \in 1, .., N$), and orthonormal eigenvectors ($f^T f = 1$) (Strang, 2006). Utilizing the computed data representation (f), it is possible to rewrite GC as in eq. (5).

$$\mathrm{GC}[A(\sigma), f(A)] = f^T L(\sigma) f = f^T \lambda(\sigma) f = \lambda(\sigma) \underbrace{f^T f}_{f^T f = 1} = \lambda(\sigma) \tag{5}$$

In the case of multiple different orthonormal eigenvector embeddings, the GC reduces to the sum of the corresponding eigenvalues λ as in eq. (6):

$$\mathrm{GC}[A(\sigma), f_{i,..,K}(A)] = \sum_{i=1}^{K} \lambda_i(\sigma) \tag{6}$$

Finding the optimal σ requires minimization of the new loss (cf. eq. (6) and fig. 1) through gradient descent (cf. eq. (7)).

$$\frac{\partial \mathrm{GC}[A(\sigma), f_{i,..,K}(A)]}{\partial \sigma} = \sum_{i=1}^{K} \frac{\partial \lambda_i(\sigma)}{\partial \sigma} = 0 \tag{7}$$

Since eq. (2) contains solely linear matrix operations, the optimal solution for f results in a closed form. Despite the intrinsic non-convex nature of SC (von Luxburg, 2007), we introduce a convex approach for determining the optimal σ in the interval $\Sigma = (0, 1/2]$ (cf. theorem 1).

Theorem 1. *The loss function loss* $= \sum_{i=1}^{K} \lambda_i(\sigma)$ *is convex in* $\Sigma = (0, 1/2], \forall f \in R^N$.

Proof. The convexity is easily shown by proving that the result of differentiating twice $\mathrm{GC}[A(\sigma), f(A)]$ w.r.t σ (cf. eq. (8)) is always non-negative.

$$\frac{\partial^2 \mathrm{GC}[A(\sigma), f_{i,..,K}(A)]}{\partial \sigma^2} = \sum_{i=0}^{k} \frac{\partial^2 \{\lambda_i(\sigma)\}}{\partial \sigma^2} \geq 0 \tag{8}$$

Differentiating the eigenvalues of a symmetric matrix (i.e., $d\lambda = f dL f^T$ (Petersen & Pedersen, 2008))

$$\frac{\partial^2 \mathrm{GC}[A(\sigma), f_{i,..,K}(A)]}{\partial \sigma^2} = \sum_{i=0}^{k} \frac{\partial^2 \{\lambda_i(\sigma)\}}{\partial \sigma^2} = \sum_{i=0}^{k} f_i \frac{\partial^2 L(\sigma)}{\partial \sigma^2} f_i^T \geq 0 \tag{9}$$

Given that $L(\sigma)$ is itself symmetric, its differentiation w.r.t $\sigma \in \Sigma$ is also another real symmetric Laplacian matrix (cf. lemma 1). Since any real symmetric Laplacian matrix is a positive semi-definite (PSD) (von Luxburg, 2007), its Rayleigh quotient is always positive (cf. lemma 1). Hence, even for eigenvector f , the result is always non-negative $f_i \frac{\partial^2 L(\sigma)}{\partial \sigma^2} f_i^T \geq 0, \forall i \in \{1, .., K\}$ resulting in $\sum_{i=0}^{k} f_i \frac{\partial^2 L(\sigma)}{\partial \sigma^2} f_i^T \geq 0, \forall f \in R^N$. □

To be able to prove theorem 1 we need lemma 1.

Lemma 1. *The second derivative of the Laplacian matrix* $\left[L^{''}(\sigma) = \frac{\partial^2 L(\sigma)}{\partial \sigma^2}\right]$ *is positive semi-definite (PSD)* $\forall \sigma \in \Sigma = (0, 1/2]$.

Proof. Initially once can easily observe that because $L(\sigma) = L^T(\sigma)$ is symmetric its second derivative is also symmetric $L^{''}(\sigma) = L^{''T}(\sigma)$ (cf. corollary 1 in Appendix). Since the domain of σ is defined by $\Sigma = (0, 1/2]$, it follows that the second derivative of L constitutes an alternate Laplacian matrix, (cf. lemma 2).

Given the new Laplacian matrix (i.e., $L^{''}(\sigma)$) is symmetric, its eigenvalues are real values (Strang, 2006). As any other Laplacian matrix, $L^{''}(\sigma)$ can be written as a product of incident matrix (IM) as $L^{''}(\sigma) = \text{IM}^T \times \text{IM}$. Whenever the eigenvalues are $L^{''}(\sigma)$ are real numbers, one can safely say that $\forall x, x^T L x = \rho \in R$. Reusing $L = \text{IM}^T \times \text{IM}$ into $x^T L x = x^T \text{IM}^T \times \text{IM} x = ||\text{IM}x||^2 = \rho$ one can conclude that $\rho \geq 0$. Hence, the new Laplacian matrix is PSD. □

To be able to prove lemma 1 we need lemma 2.

Lemma 2. $L^{''}(\sigma)$ *is another Laplacian matrix* $\forall \sigma \in \Sigma = (0, 1/2]$.

Proof. To prove that the second derivative of the Laplacian matrix L is another Laplacian matrix, one must initially demonstrate that each diagonal entry is the negative sum of the non-diagonal entries in its corresponding row (or equivalently, its column), owing to the symmetry of the matrix. Since the matrix is symmetric, we choose row-wise, equivalent to column-wise, and vice versa. Furthermore, it suffices to show for just one row; without the loss of generality, it is equivalent to the rest of the rows.

$$\frac{\partial^2 L_{j,j}(\sigma)}{\partial \sigma^2} = \frac{\partial^2 \sum_{i=0}^{N-1} A_{j,i}(\sigma)}{\partial \sigma^2} = \sum_{i=0}^{N-1} \frac{\partial^2 A_{i,j}(\sigma)}{\partial \sigma^2} \tag{10}$$

Given that RBF kernel $K(\sigma) = e^{\frac{-d(x,y)}{\sigma^2}}$ is convex in Σ (cf. lemma 3) its second derivative is non-negative $\frac{\partial^2 K(\sigma)}{(\partial \sigma)^2} > 0, \forall \sigma \in \Sigma$. This results in two important implications:

Firstly, the diagonal values for $L^{''}_{i,i}(\sigma)$ are always positive since $\frac{\partial^2 A_{i,j|i\neq j}(\sigma)}{\partial \sigma^2} > 0, \forall i \in [0, N-1], \forall \sigma \in \Sigma$ in eq. (10).

Secondly, off diagonal of entries of $L^{''}_{i,j|i\neq j}(\sigma) = -\frac{\partial^2 K(\sigma)}{(\partial \sigma)^2} < 0, \forall \sigma \in \Sigma$

Hence, together with eq. (10), one can conclude that the new matrix is a symmetric Laplacian matrix (cf. eq. (11)).

$$L^{''}(\sigma) = L^{''}(\sigma)^T = \frac{\partial^2 L(\sigma)}{\partial \sigma^2} \tag{11}$$

□

To be able to prove lemma 2 we need lemma 3.

Lemma 3. *The RBF kernel* $K(\sigma) = e^{-\frac{d(x,y)}{\sigma^2}}$ *is convex* $\forall \sigma \in \Sigma = (0, 1/2]$.

Proof. To prove this convexity, we need to prove that $\frac{\partial^2 K(\sigma^2)}{\partial \sigma^2} \geq 0, \forall \sigma \in \Sigma$.

$$\frac{\partial^2 K(\sigma^2)}{\partial \sigma^2} = 2e^{\frac{-d(x,y)^2}{\sigma^2}} \frac{d(x,y)^2}{(\sigma^2)^2} \Big\{1 + 2d(x,y)^2 - 4\sigma^2\Big\} \tag{12}$$

The first two terms in eq. (12) are always non-negative $\left(e^{\frac{-d(x,y)^2}{\sigma^2}} \geq 0, \frac{d(x,y)^2}{(\sigma^2)^2} \geq 0, \forall \sigma, d(x,y) \in R^+\right)$. However, the last term $\left(\left\{1+2d(x,y)^2-4\sigma^2\right\} \geq 0, \forall \sigma \leq \frac{1}{2}, \forall d(x,y) \in R^+\right)$ is guaranteed to be non-negative. Hence the proof.

□

The new algorithm, built on top of the SC setup (von Luxburg, 2007), adopts a two-stage operation within each iteration, starting by vanilla SC setup followed by the σ update (cf. algorithm 1). In the initial phase, the parameter σ is held constant, allowing for the computation of the first K eigenvectors f, which serve as the embeddings for the dataset. Subsequently, the second phase maintains these computed eigenvectors f fixed while it refines the value of σ. Both stages, however, are unified in their objective to optimize the graph cut (cf. eq. (2)).

To introduce the scale invariance in the Euclidean distance matrix ($E^{[N,N]}$ in algorithm 1), its values are normalized so that its minimum value becomes zero and its maximum value becomes one. To ensure that optimization happens within a convex set, the RBF bandwidth always starts at $\sigma = \frac{1}{2}$. As the denominator involves the square of σ, both positive and negative values of σ of the same magnitude have the same modulation effect in the RBF kernel eq. (1). Hence, we consider only positive values for σ, as its negative values do not introduce any extra impact. To mitigate the impact of the initialization of the learning rate, its initial value is set to one (i.e., $lr = 1$). Consequently, the initial update on σ is driven solely by its gradient. We decimate the learning rate if σ assumes a value that is not in Σ.

The parameter σ is inherently data-dependent, adapting its value to the specifics of the dataset under analysis. The gradient of σ acts as a guiding mechanism to ensure its convergence, tending toward zero as σ nears the optimal point. Although reducing the magnitude of σ does lead to a lower graph cut value in spectral clustering, setting σ too close to zero is not advisable. This setting results in the edge weights of the graph converging to zero. Consequently, each data point becomes isolated into its own cluster, effectively obliterating the underlying data structure and preventing the detection of inherent clusters.

Algorithm 1 Similarity Learning

procedure (Data $X^{[N,D]}$, Number of clusters K)
  Compute the Euclidean distance matrix $E^{[N,N]}$.
  Scale the Euclidean distance matrix: $E^{[N,N]} = \frac{E^{[N,N]}}{\max E^{[N,N]}}$.
  $\sigma = \frac{1}{2}$. ▷ No modulation effect for RBF kernel as a start.
  $lr = 1$. ▷ Initially, only the gradient updates σ.
  for each $i = 1, 2, \ldots n_{iter}$ **do**
    Vanilla SC Setup
    Affinity Matrix: $A^{[N,N]} \leftarrow e^{-\frac{E^{2\,[N,N]}}{\sigma}}$.
    Degree Matrix: $D^{[j,j]} \leftarrow \sum_{i=0}^{N} A^{[i,j]}, \forall j \in \{1, .., N\}$.
    Laplacian Matrix: $L^{[N,N]} \leftarrow D^{[N,N]} - A^{[N,N]}$.
    First K eigenvalues: $\lambda_{1,..,K} \leftarrow \text{EVD}(L)$
    Our proposal for σ learning
    Compute: $\text{Loss}(\sigma) \leftarrow \sum_{i=0}^{K} \lambda_i(\sigma)$
    Back-propagation: $\frac{\partial \text{L}(\sigma)}{\partial \sigma}$
    Gradient descent: $\sigma_{i+1} \leftarrow \sigma_i - lr \frac{\partial \text{L}(\sigma)}{\partial \sigma}$
    if $\sigma_{i+1} \notin \Sigma$ **then** ▷ Negative σ are excluded.
      $\sigma_{i+1} \leftarrow \sigma_i + lr \times \frac{\partial \text{L}(\sigma)}{\partial \sigma}$ ▷ Correct the σ.
      $lr \leftarrow lr/10$ ▷ Decimate the lr.

This proposed method works solely on the Laplacian matrix setup. Other versions of normalized Laplacian were tried, i.e., symmetric normalized Laplacian $\left(L_{\text{normal}}^{\text{symmetric}} = D^{-\frac{1}{2}} L D^{-1\frac{1}{2}}\right)$, and normalized Laplacian row-wise stochastic version $\left(L_{\text{normal}}^{\text{stochastic}} = D^{-1} L\right)$ but the gradient computation was not updating σ in the

anticipated direction. Furthermore, different kernels did not perform on the setup; therefore, *the RBF kernel is the only one that enables the update of the parameters within the Laplacian setup.*

Complexity analysis: The Euclidean distance matrix is computed once outside the loop at $O(N^2)$ complexity. Since we are looking solely for the first K eigenvectors, the EVD complexity is $O(K \times N^2)$ (Golub & Van Loan). Similarly, the complexity of gradient computation using the back-propagation is $O(K \times N^2)$ since the output of the computational graph has K eigenvalues (Baydin et al.). Over n_{iter} iterations, the total complexity would be $O(n_{iter} \times K \times N^2)$.

3 Related work

To the best of our knowledge, this is the first work that tries to perform similarity metric learning for SC without introducing any hyperparameter in the similarity function or the loss function. The RBF kernel is identified to accommodate a sufficient amount of generality, whereby its bandwidth σ can tune this similarity metric to fit the given data best. Similarity learning is a researched area where previously proposed approaches fall within two main categories.

Neighborhood approach: The neighborhood approach assumes that similar data should be associated with a high similarity score and vice versa (cf. eq. (13)). Model-free similarity learning learns directly every pairwise similarity between data without imposing any particular structure on the metric.

$$\arg\min_{z_i} \sum_{j=1}^{N} (||x_i - x_j||^2 z_{i,j} + \sum_{k=0}^{K} \alpha_k f_k(z_{i,j}) \tag{13}$$

The parameters α_k are a set of regularization modulation, and $f_k(z_{i,j})$ are a set of regularization constraints. This approach does not admit any particular trend of the pairwise similarities $(z_{i,j})$ between data since these values are acquired directly from the optimization process.

Self-expression: Self-expressionism is another approach that assumes a linear expression of individual data relative to the rest (cf. eq. (14)).

$$\arg\min_{Z} (||X - XZ||_F^2 + \sum_{k=0}^{K} \alpha_k f_k(Z) \tag{14}$$

The underlying assumption is that for any individual data, the learned relational parameters $Z_{i,j}$ should be high only for those data pairs with similar discriminative features and vice versa. Under this assumption, the learned coefficients $Z_{i,j}$ are considered pairwise similarity values between individual data. Like the neighborhood approach, this method does not impose any particular formation on the coefficients $Z_{i,j}$.

These methods perform well whenever the underlying assumption incorporated with the regularization term matches the data generation process. *Furthermore, the hyperparameter factor that modulates the impact of the regularization cannot be objectively calibrated, leading to suboptimal performance when chosen randomly.*

One can notice that the learnability of both these methods (cf. eqs. (13) and (14)) is enabled solely by restricting the otherwise vast parameter space through a series of regularization constraints. Furthermore, in the absence of any constraint, the first approach (cf. eq. (13)) would provide all zeros values ($z_{i,j} = 0, \forall i, j \in \{1, ..., N\}$). Similarly, without any constraint, the second approach (cf. eq. (13)) would provide an identity matrix $Z = I$. In both cases, the learned similarities are not of practical use.

Instead, the proposed work assumes the similarity values for every possible data pair to be a function of a predefined distance metric (cf. eq. (1)). Hence, the cardinality of the parameter search space reduces from N^2 for eqs. (13) and (14) down to one (i.e., σ).

However, unlike these approaches, which learn a low-dimension data representation without assuming the number of dimensions, our proposed work assumes that data resides on a K dimensional space apriori.

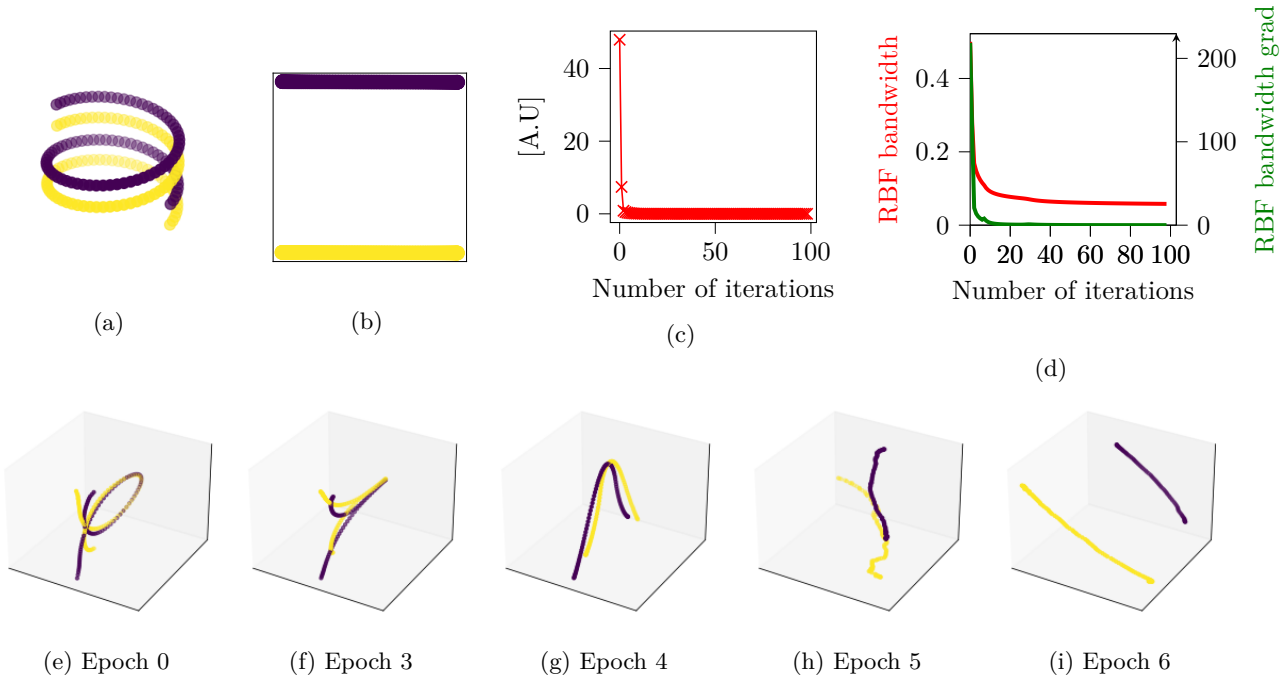
Namely, every data pair is projected and compared into the infinite-dimensional space through the RBF kernel (cf. eq. (1)), and from there, the K dimensions that enable linear separation are selected. Hence, both of these assumptions render any additional constraints obsolete.

Previous methods based upon eqs. (13) and (14) have shown excellent performance in several applications, such as semi-supervised learning (Zhuang et al., 2017; Kamnitsas et al., 2018), face recognition (Zhang et al., 2011), feature selection (Du & Shen, 2015), clustering (Nie et al., 2014).

4 Experiments

We ran a series of experiments on benchmark diagnostic datasets (cf. figs. 3 and 5 to 9 in Appendix) (Fränti & Sieranoja) as well as real-world datasets (cf. table 1) to showcase the method's capabilities at separating highly non-convex shapes (cf. fig. 3a and figs. 5a, 6a, 7a, 8a and 9a in Appendix) into linearly separable clusters (cf. fig. 3b and figs. 5b, 6b, 7b, 8b and 9b in Appendix). In all scenarios, the method tries to amplify the big pairwise distances within a confined space while reducing the small pairwise distances. Hence, the method finds a persisting gap within the topology of the dataset and tries to linearize it in the embedding space (cf. fig. 3b and figs. 5b, 6b, 7b, 8b and 9b in Appendix). One can notice from the optimization trajectory of the gradient for the RBF bandwidth σ to converge towards zero (cf. fig. 3d and figs. 5d, 6d, 7d, 8d, 9d, 10b, 11b, 12b, 13b, 14b, 15b and 16b in Appendix). In the presented cases, the magnitude of σ is progressively reduced until its gradient approaches zero.

Furthermore, the cumulative magnitude of the first K eigenvalues (i.e., loss $= \sum_{i=1}^{K} \lambda_i$) decreases throughout the training process (cf. figs. 3c and 4b and figs. 5c, 6c, 7c, 8c, 9c, 10a, 11a, 12a, 13a, 14a, 15a and 16a in Appendix). Notice that the smaller the magnitude of the first K eigenvalues the more distinct the first K clusters are in a dataset.



(a) (b) (c) (d)

(e) Epoch 0 (f) Epoch 3 (g) Epoch 4 (h) Epoch 5 (i) Epoch 6

Figure 3: Benchmark dataset in fig. 3a. Our method learns a linear separation of these two spirals fig. 3b. In fig. 3b is the eigen-embedding of the benchmark data. In fig. 3c, is the loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training. over iterations. The optimization of kernel bandwidth σ using the gradient descent stabilizes as the σ gradient diminishes fig. 3d. A snapshot of the optimized trajectory is in figs. 3e to 3i

This linearization of the separation boundary reduces the complexity of the downstream tasks one can perform upon these embeddings. Clustering is one popular unsupervised task whose performance is solely

Table 1: Description of the experimented datasets.

Dataset	Size	Nr of Dimensions	Nr of Classes
Yale	165	1024	15
Jaffe	213	676	10
ORL	400	1024	40
COIL20	1440	1024	20
BA	1404	320	36
TR11	414	6429	9
TR41	878	7454	10
TR45	690	8261	10

the capability for learning the best representation of the data topology (von Luxburg, 2007). Hence, to compare the capability of our supervised method, we run K-means + + (Jain, 2010; Arthur & Vassilvitskii) on the eigen-embeddings where the clustering performance would be a proxy evaluator for the learned similarity matrix. We extend to realistic datasets where the models try to find linear separation between data clusters. In such a real data scenario, the proposed method tries to separate the dataset using persistent low-density (i.e., gap) regions throughout the dataset.

Dataset description: The datasets described in table 1 are widely utilized to assess clustering performance as no registration is needed upfront. The first three, JAFFE (Lyons et al., 1998), YALE (of Yale, 1997), ORL (Cambridge, 1994), contain human faces obtained at different illumination conditions, or different facial expressions, or with and without glasses. The last image dataset, COIL20 (Nene et al., 1996)) contains images of toys acquired at different orientations. While the rest of the data (BA, TR11, TR41, TR45) are text corpus (TR). These datasets are curated particularly for clustering algorithms that do not rely on deep learning techniques. Manual pre-registration of the images eliminates the need for translation invariance representation, which is typically achieved through the application of deep clustering techniques. Consequently, these datasets are suitable for testing traditional clustering methods that do not involve CNN-based feature extraction.

Data preparation: The image datasets are assumed to be correctly registered beforehand; therefore, no feature selection is needed upfront. To ensure the equivariance of the scale of the features, standardization ($\hat{x}_{\text{stand}} = \frac{x-\hat{\mu}}{\sqrt{\hat{\sigma}^2+\epsilon}}$) is performed before the computation of the Euclidean distance (cf. algorithm 1).

Once the similarity learning has been completed, we perform K-means++ (Jain, 2010; Arthur & Vassilvitskii) clustering upon the eigen-embeddings. To diminish the effect of the initialization, K-means + + has been run 50 different times until convergence.

Unlike our proposed method, *the competing ones have a hyperparameter(s) dependency that cannot be objectively tuned.* Hence, for a fairer comparison with our method, we report the performance of the competing method as the average across a range of hyperparameters. Moreover, we also report the maximal performance figures of the alternative methods (cf. tables 2 and 3). Two different kernels of choice have been tested across 12 different hyperparameter combinations. RBF kernel eq. (1) across different bandwidth ($\{0.01, 0.05, 0.1, 1, 10, 50, 100\}$) along with a polynomial kernels $K(x, y) = (a + x^T y)^b$ with four different hyperparameter values (i.e., $a = \{0, 1\}$ and $b = \{2, 4\}$). The proposed method is entirely reproducible; therefore, we run just one experiment per dataset and report just one evaluation metric.

Competing methods: We assess the effectiveness of the proposed method in a clustering scenario. Therefore we employ an SC setup in which the spectral embeddings derived from the acquired similarity matrix exhibit better performance compared to standard clustering methods. As competing methods, we utilize a range of clustering methods that incorporate kernel tricks with various hyperparameters, as well as kernel-free similarity learning methods.

Kernel K-means (KKM) (Dhillon et al., 2004) can perform the mean computation in the kernel reproducible space. This method uses a single kernel and demands arbitrary hyperparameter settings that must be

set manually. Robust Kernel K-Means (RKKM) (Du et al., 2015) tries to attain maximum clustering performance by combining multiple kernels on a K-Means setup. Multiple kernel k-means (MKKM) (Huang et al., 2012b) is similar to KKM while aggregating multiple kernels to attain a compounding effect in the final clustering performance.

Spectral Clustering (SC) (Ng et al., 2001) is the first method in the evaluation. Since our proposal is essentially built on top of SC, we can show that it is possible to optimize the RBF kernel, given the data. Therefore, vanilla SC serves as the baseline for the proposed method. Affinity aggregation spectral clustering (AASC) algorithm (Huang et al., 2012a) tries to consolidate multiple affinities matrix to refine the clustering result. Twin Learning for Similarity and Clustering (TLSC) (Kang et al., 2017) is another approach that does not require any kernel function but directly learns the similarity matrix and the clustering indexing. Similarity Learning via Kernel Preserving Embeddings (SLKE) (Kang et al., 2019) is similar to TLSC, which does not learn clustering indicator vectors but solely the similarity matrix. The SLKE algorithm has been experimented with in two distinct regularizations: Sparse SLKE (SLKE-S) and Low-ranked SLKE (SLKE-R). SC is performed on top of the learned similarity matrix via SLKE. The performance of all these methods is conditional not only on the hyperparameters but also on the type of regularization. The aim is to demonstrate the superiority of the proposed method in terms of clustering performance and ease of use.

Recent advancements in deep clustering methods have demonstrated good performance on well-established image datasets. These methods comprise two stages: unsupervised representation learning, followed by the actual clustering step (Ren et al., 2022; Zhou et al., 2022). The bulk of the research in this area concentrates on representation learning. This phase involves organizing the data such that similar items are closely embedded, whereas dissimilar items are spaced further apart. This configuration simplifies the task for conventional clustering techniques. However, our proposed method focuses on the improvement of the clustering technique itself.

Evaluation metric: Unlike classifiers, where the type of classes and their predictions are kept intact, the prediction cluster indicators are permuted in clustering. Therefore, the clustering evaluation reduces to comparing two different types of sets, i.e., ground-truth labels set and cluster indicator. As a result, more than a single metric is required to assess all aspects of the clustering performance normalized mutual information (NMI) and accuracy (ACC) capture different aspects of the performance.

NMI in eq. (15) compares two sets using entropy as a criterion. It measures how much entropy is required to describe the second set using the entropy of the first set. Normalization scales the metric from zero to one.

$$\mathrm{NMI}(X, Y) = \frac{\sum_{x \in X} \sum_{y \in Y} P(x, y) \log \frac{P(x,y)}{P(x)P(y)}}{\max\left\{ -\sum_{x \in X} P(x) \log P(x), -\sum_{y \in Y} P(y) \log P(y) \right\}} \tag{15}$$

In contrast, accuracy (ACC in cf. eq. (16)) measures the pairwise exactness of the predicted clusters with the ground-truth clusters. This metric requires a bi-partite set alignment as in an unsupervised learning setup; the predicted cluster index does not correspond to the ground-truth label value at the individual data level. Therefore, the two sets (X and Y of size n) are aligned through the Kuhn-Munkres algorithm (KMA) (Kuhn, 1955) (*i.e.*, $\hat{Y} = \mathrm{KMA}(Y)$) upfront (cf. eq. (16)).

$$\mathrm{ACC}[X, \hat{Y} = \mathrm{KMA}(Y)] = \frac{\sum_{x \in X, \hat{y} \in \hat{Y}} \delta(x, \hat{y})}{n} \tag{16}$$

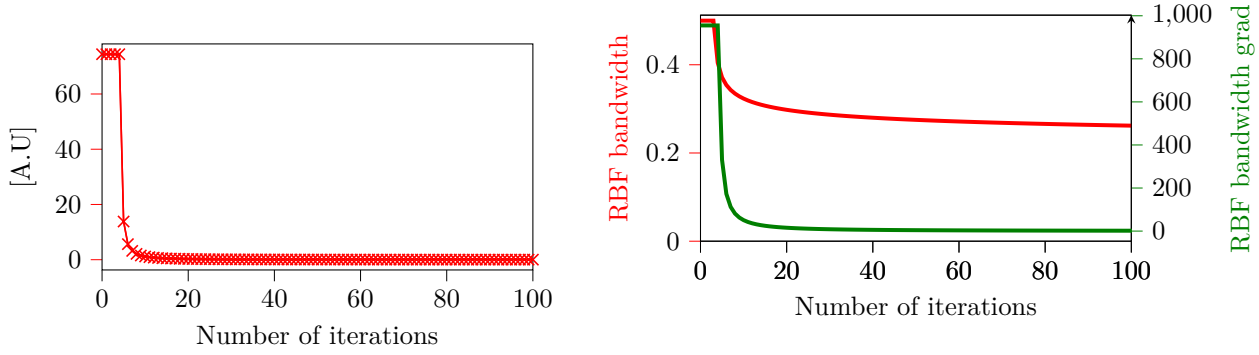
Results: One can notice that the model outperforms the previously reported method on an NMI metric by up to 10% (cf. table 3). This indicates that the entropy of the cluster indexes generated by our similarity matrix is closer to the one actual class index for the data at hand. Furthermore, the amount of improvement (in the NMI term) relative to vanilla SC is quite considerable. This suggests that the proposed method is more effective than the baseline and SOTA methods at identifying gaps in the data topology. NMI makes a more global comparison between true labels and predictions; therefore, we compute clustering accuracy

Table 2: Accuracy (ACC) over the different dataset. The average performance is outside the brackets, while the highest performance from multiple runs is in the bracket for each method. MKKM and AASC results are an aggregation of multiple realizations.

Data	SC	KKM	MKKM	RKKM	AASC	TLSC	SLKE-S	SLKE-R	Ours
YALE	40.53(49.42)	38.97(47.12)	45.70	39.71(48.09)	40.64	45.35(55.85)	38.89(61.82)	51.28(66.24)	**52.62**
JAFFE	54.03(74.88)	67.09(74.39)	74.55	67.89(75.61)	30.35	86.64(99.83)	70.77(96.71)	**90.89**(99.85)	86.38
ORL	46.65(58.96)	45.93(53.53)	47.51	46.88(54.96)	27.20	50.50(62.35)	45.32(77.00)	59.00(74.75)	**67.75**
COIL20	43.65(67.60)	50.74(59.49)	54.82	51.89(61.64)	34.87	38.03(72.71)	56.83(75.42)	65.55(84.03)	**69.79**
BA	26.25(31.07)	33.66(41.20)	40.52	34.35(42.17)	27.07	39.50(47.72)	36.35(50.74)	35.79(44.37)	**43.87**
TR11	43.32(50.98)	44.65(51.91)	50.13	45.04(53.03)	47.15	54.79(71.26)	46.87(69.32)	**55.07**(74.64)	50.96
TR41	44.80(63.52)	46.34(55.64)	**56.10**	46.80(56.76)	45.90	43.18(65.60)	47.91(71.19)	53.51(74.37)	55.69
TR45	45.96(57.39)	45.58(58.79)	58.46	45.69(58.13)	52.64	53.38(74.02)	50.59(78.55)	58.37(79.89)	**63.32**

Table 3: NMI over the different dataset. The average performance is outside the brackets, while the highest performance from multiple runs is in the bracket for each method. MKKM and AASC results are an aggregation of multiple realizations.

Data	SC	KKM	MKKM	RKKM	AASC	TLSC	SLKE-S	SLKE-R	Ours
YALE	44.79(52.92)	42.07(51.34)	50.06	42.87(52.29)	46.83	45.07(56.50)	40.38(59.47)	52.87(64.29)	**62.05**
JAFFE	59.35(82.08)	71.48(80.13)	79.79	74.01(83.47)	27.22	84.67(99.35)	60.83(94.80)	81.56(99.49)	**93.90**
ORL	66.74(75.16)	63.36(73.43)	68.86	63.91(74.23)	43.77	63.55(78.96)	58.84(86.35)	75.34(85.15)	**86.30**
COIL20	54.34(80.98)	63.57(74.05)	70.64	63.70(74.63)	41.87	73.26(82.20)	65.40(80.61)	73.53(91.25)	**85.32**
BA	40.09(50.76)	46.49(57.25)	56.88	46.91(57.82)	42.34	52.17(63.04)	55.06(63.58)	50.11(56.78)	**60.51**
TR11	31.39(43.11)	33.22(48.88)	44.56	33.48(49.69)	39.39	37.58(58.60)	30.56(67.63)	45.39(70.93)	**48.68**
TR41	36.60(61.32)	40.37(59.88)	**57.75**	40.86(60.77)	43.05	43.18(65.50)	34.82(70.89)	47.45(68.50)	51.15
TR45	33.22(48.03)	38.69(57.87)	**56.17**	38.96(57.86)	41.94	44.36(74.24)	38.04(72.50)	50.37(78.12)	55.93



(a) Loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training. (b) RBF bandwidth σ and its gradient during training.

Figure 4: Optimization of the σ on the JAFFE dataset in fig. 4b. Cumulative magnitude reduction of the fist K=10 eigenvalues (loss $= \sum_{i=1}^{K} \lambda_i$) over the iterations for the JAFFE dataset in fig. 4a.

as a complementary metric to provide the comparison at the individual level. The accuracy has improved for nearly all previously proposed methods. Nevertheless, it is important to note that the evaluation of competing methods is based on average accuracy, which means a less accurate clustering assignment is just as possible. As in the case of NMI, our proposed method significantly improves accuracy compared to the SC baseline across all test image datasets. Similarly, optimizing the kernel bandwidth σ follows a similar trajectory as in the benchmark dataset. The method progressively tries to reach an optimal value and decreases the gradient (cf. figs. 4b, 10b, 11b, 12b, 13b, 14b, 15b and 16b in Appendix). Likewise, the cumulative magnitude of the first K eigenvalues progresses towards smaller values as in the case of the

benchmark data experiments (cf. figs. 4a, 10a, 11a, 12a, 13a, 14a, 15a and 16a in Appendix). This behaviour indicates the formation of the clusters as the σ progresses towards more optimal values.

5 Discussion

Similarity learning for SC operates under the assumption that optimizing parameters for the kernel function, such as the eigen-embeddings of the dataset providing the maximum linear separation for K clusters, is possible. Utilizing this assumption enables the proposed method to leverage the discriminative features to infer the global latent structure of the given data. Otherwise, a non-optimal RBF bandwidth σ would either amplify the distances too little, such as no cluster is visible, or too much, such as every point becomes an individual cluster.

Since no labels are provided, similarity learning ought to operate in an entirely unsupervised way without the need for hyperparameters. The hyperparameter cannot be validated using the data or the objective function; their suboptimal values can be detrimental to the performance of clustering as downstream tasks. *To the best of our knowledge, this is the first work that proposes a setup for similarity learning for SC without introducing any hyperparameter during the optimization.* Our proposed work is built on top of the SC setup while utilizing the RBF kernel function to augment the pairwise similarity.

We showcase the method's ability to separate highly non-convex shapes and project the dataset into a linearly separable topology on a series of benchmark datasets. The proposed method is validated on image and text datasets and compared with vanilla SC as a baseline and recent similarity learning models. Our proposed method outperforms competing approaches in both NMI and accuracy metrics. The performance of all the alternative models is strictly dependent on their hyperparameter settings. *Since these hyperparameters are not validated in an unsupervised setting, the optimal performance of the competing methods is never guaranteed.*

The method we propose introduces unsupervised learning of the similarity metric for SC. However, it should be noted that SC itself inherently requires the predefined specification of the number of clusters K within the dataset. Looking forward, an area for further research would be to incorporate methods for estimating the number of clusters autonomously. A promising approach is to identify density peaks within the data's topological structure, an idea inspired by the technique presented in the study by Rodriguez & Laio (2014). Adopting such a strategy could enhance the unsupervised aspect of SC, expanding it from similarity learning to include autonomous cluster number determination.

The proposed method is constrained to the RBF kernel, necessitating precise pairwise image alignment, a condition that is impractical for real-world image datasets. To increase applicability, the approach should incorporate deep learning models, which produce translation-invariant embeddings, thus reducing the need for exact pairwise image alignment.

Lastly, the modulation of the learning rate is currently governed by heuristic principles rather than being informed directly by the data. Nevertheless, there exists an opportunity to transition to a more data-centric approach by employing a model that integrates the curvature information of the loss function to guide the learning rate adjustments.

References

Text REtrieval Conference (TREC) Home Page — trec.nist.gov. `https://trec.nist.gov/`. [Accessed 15-08-2023].

David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. SODA '07.

Atilim Gunes Baydin, Barak A. Pearlmutter, and Alexey Andreyevich Radul. Automatic differentiation in machine learning: a survey. *CoRR*.

AT&T Laboratories Cambridge. Our Database of Faces, 1994.

Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: Spectral clustering and normalized cuts. KDD '04, 2004.

Liang Du and Yi-Dong Shen. Unsupervised feature selection with adaptive structure learning. 2015.

Liang Du, Peng Zhou, Lei Shi, Hanmo Wang, Mingyu Fan, Wenjian Wang, and Yi-Dong Shen. Robust multiple kernel k-means using 2;1-norm. IJCAI'15, pp. 3476–3482, 2015.

Xinjie Fan, Yuguang Yue, Purnamrita Sarkar, and Y. X. Rachel Wang. On hyperparameter tuning in general clustering problemsm. PMLR, 13–18 Jul .

Pasi Fränti and Sami Sieranoja. K-means properties on six clustering benchmark datasets.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press.

Hsin-Chien Huang, Yung-Yu Chuang, and Chu-Song Chen. Affinity aggregation for spectral clustering. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012a.

Hsin-Chien Huang, Yung-Yu Chuang, and Chu-Song Chen. Multiple kernel fuzzy clustering. *IEEE Transactions on Fuzzy Systems*, 2012b.

Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 2010. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

Konstantinos Kamnitsas, Daniel Coelho de Castro, Loïc Le Folgoc, Ian Walker, Ryutaro Tanno, Daniel Rueckert, Ben Glocker, Antonio Criminisi, and Aditya V. Nori. Semi-supervised learning via compact latent space clustering. *CoRR*, 2018.

Zhao Kang, Chong Peng, and Qiang Cheng. Twin learning for similarity and clustering: A unified kernel approach. *CoRR*, 2017.

Zhao Kang, Yiwei Lu, Yuanzhang Su, Changsheng Li, and Zenglin Xu. Similarity learning via kernel preserving embedding, 2019.

H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research*, Logistics Quarterly 2,83–97, 1955.

Michael Lyons, Miyuki Kamachi, and Jiro Gyoba. The Japanese Female Facial Expression (JAFFE) Dataset, April 1998.

Samer A. Nene, Shree K. Nayar, and Hiroshi Murase. Columbia object image library (coil-20). Technical Report CUCS-005-96, February 1996.

Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

Feiping Nie, Xiaoqian Wang, and Heng Huang. Clustering and projected clustering with adaptive neighbors. KDD '14, 2014.

University of Yale. The Yale Face Database, September 1997.

K. B. Petersen and M. S. Pedersen. The matrix cookbook, October 2008. Version 20081110.

Yazhou Ren, Jingyu Pu, Zhimeng Yang, Jie Xu, Guofeng Li, Xiaorong Pu, Philip S. Yu, and Lifang He. Deep clustering: A comprehensive survey, 2022.

Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191): 1492–1496, 2014.

Amnon Shashua. Introduction to machine learning: Class notes 67577, 2009.

Gilbert Strang. *Linear algebra and its applications*. Thomson, Brooks/Cole, Belmont, CA, 2006. ISBN 0030105676 9780030105678 0534422004 9780534422004. URL http://www.amazon.com/Linear-Algebra-Its-Applications-Edition/dp/0030105676.

Ulrike von Luxburg. A tutorial on spectral clustering. 2007.

Lei Zhang, Meng Yang, and Xiangchu Feng. Sparse representation or collaborative representation: Which helps face recognition? In *2011 International Conference on Computer Vision*, 2011.

Sheng Zhou, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, Martin Ester, et al. A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions. *arXiv preprint arXiv:2206.07579*, 2022.

Liansheng Zhuang, Zihan Zhou, Shenghua Gao, Jingwen Yin, Zhouchen Lin, and Yi Ma. Label information guided graph construction for semi-supervised learning. *IEEE Transactions on Image Processing*, 2017.

A Appendix

A.1 Symmetry of the Laplacian Matrix's Second Derivative

Corollary 1. *$L^{''}(\sigma)$ is symmetric.*

Proof. It is crucial to compute the $L^{''}(\sigma)$ using the RBF as a kernel to calculate the second derivative of the RBF itself first.

Using the composition $\frac{d}{dx}\left[f(g(x))\right] = \frac{d}{dg}\left[f(g)\right]\frac{d}{dx}\left[g(x)\right]$ the derivative could be simplified to:

$$\frac{dK(\sigma^2)}{d\sigma} = \frac{dK(\sigma^2)}{d\sigma^2}\frac{d\sigma^2}{d\sigma} = 2\sigma\frac{dK(\sigma^2)}{d\sigma^2} \tag{17}$$

while the second derivative is:

$$\frac{d^2K(\sigma^2)}{(d\sigma)^2} = \frac{d}{d\sigma}\Big\{\frac{dK(\sigma^2)}{d\sigma}\Big\} = \frac{d}{d\sigma}\Big\{2\sigma\frac{dK(\sigma^2)}{d\sigma^2}\Big\} = \frac{d}{d\sigma}\Big\{2\sigma K^{'}(\sigma^2)\Big\} \tag{18}$$

$$\frac{d^2K(\sigma^2)}{(d\sigma)^2} = K^{'}(\sigma^2)\frac{d(2\sigma)}{d\sigma} + 2\sigma\frac{dK^{'}(\sigma^2)}{d\sigma} \tag{19}$$

$$\frac{d^2K(\sigma^2)}{(d\sigma)^2} = 2K^{'}(\sigma^2) + 2\sigma\frac{dK^{'}(\sigma^2)}{d\sigma^2}\frac{d\sigma^2}{d\sigma} \tag{20}$$

$$\frac{d^2K(\sigma^2)}{(d\sigma)^2} = 2\frac{dK(\sigma^2)}{d\sigma^2} + 4\sigma^2\frac{d^2K(\sigma^2)}{(d\sigma^2)^2} \tag{21}$$

where the first derivative of the kernel is:

$$\frac{dK(\sigma^2)}{d\sigma^2} = e^{\frac{-|x-y|^2}{\sigma^2}}\frac{|x-y|^2}{(\sigma^2)^2} \tag{22}$$

and the second derivative of the kernel is

$$\frac{d^2K(\sigma^2)}{(d\sigma^2)^2} = \frac{d^2e^{\frac{-|x-y|^2}{\sigma^2}}}{(d\sigma^2)^2} = \frac{d}{d\sigma^2}\left\{\frac{de^{\frac{-|x-y|^2}{\sigma^2}}}{d\sigma^2}\right\} \tag{23}$$

$$= e^{\frac{-|x-y|^2}{\sigma^2}}\frac{|x-y|^2}{(\sigma^2)^4}\Big\{|x-y|^2 - 2\sigma^2\Big\} \tag{24}$$

Combining eq. (22) and eq. (23) with eq. (21) results into:

$$K^{''}(\sigma^2) = \frac{d^2K(\sigma^2)}{(d\sigma)^2} = 2e^{\frac{-|x-y|^2}{\sigma^2}}\frac{|x-y|^2}{(\sigma^2)^2}\Big\{1 + 2|x-y|^2 - 4\sigma^2\Big\} \tag{25}$$

As a result of differentiation, the RBF kernel transforms into another different similarity metric eq. (26). The new similarity metric is not a kernel since it contains the Euclidean distance, which is not a kernel function.

$$K^{''}(x,y) = K^{''}(y,x) \tag{26}$$

As a result of symmetricity(cf. eq. (27)), the new matrix resulting from the differentiation of the Laplacian would still be another symmetricity matrix.

$$\frac{\partial^2 L(\sigma)}{(\partial\sigma)^2} = \left\{ \frac{\partial^2 L(\sigma)}{(\partial\sigma)^2} \right\}^T \tag{27}$$

□

A.2 Experiments on benchmark dataset

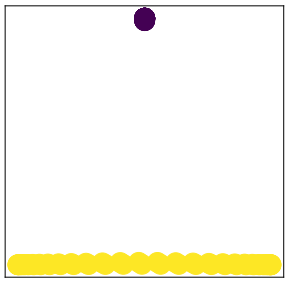
Herein, we present a series of sanity checks on popular benchmark clustering datasets (cf. table 4). These benchmark datasets serve as the sanity check for any clustering algorithm. The purpose of these experimentations is to showcase the capability of the method to separate clusters under different types of convexity. Apart from different types of separation, the proposed method also handles an asymmetric number of data per cluster as in fig. 8. For each tested dataset, we present the final eigen-embeddings, the loss (loss $= \sum_{i=1}^{K} \lambda_i$) over iterations together with the trajectory of the RBF bandwidth and its gradient. Last but not least, a visual representation of the eigen-embeddings is provided to illustrate how the method facilitates the linear separation of individual clusters.

The convexity of the method persists for each dataset, as it is evident from the always decreasing magnitude of the RBF bandwidth gradient (figs. 5d, 6d, 7d, 8d and 9d). Notice that although RBF bandwidth always starts at $\sigma = 0.5$, its starting gradient varies across different types of datasets.

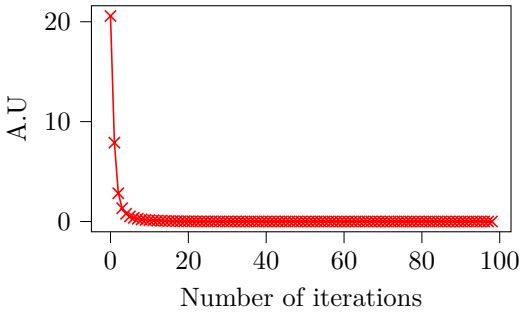
Table 4: Description of the benchmark datasets.

Dataset	**Size**	**Nr of Dimensions**	**Nr of Classes**
Two spirals (cf. fig. 3)	500	3	2
Two rings (cf. fig. 5)	500	3	2
Three spirals (cf. fig. 6)	312	2	3
Two circles (cf. fig. 7)	500	2	2
Two (asymmetric) half moons (cf. fig. 7)	373	2	2
Two (symmetric) half moons (cf. fig. 8)	500	2	2

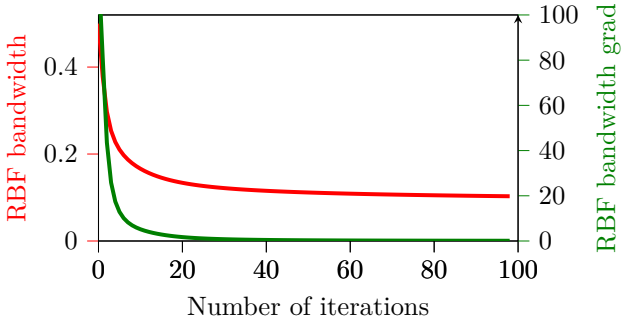
(a) Benchmark dataset in 3D.



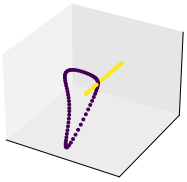
(b) Eigen-embeddings of the benchmark dataset.



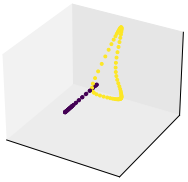
(c) Loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training.



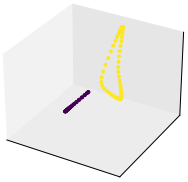
(d) RBF bandwidth σ and its gradient during training.



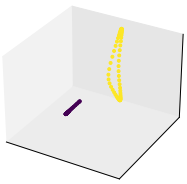
(e) Epoch 0



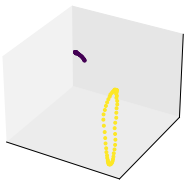
(f) Epoch 5



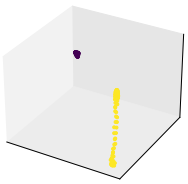
(g) Epoch 15



(h) Epoch 20



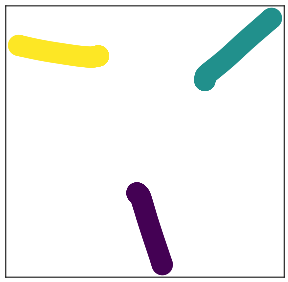
(i) Epoch 25



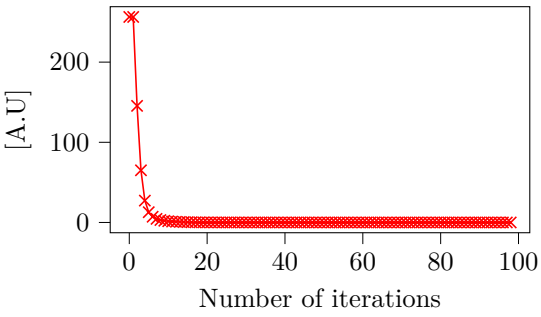
(j) Epoch 25

Figure 5: Benchmark dataset two rings in 3D (cf. fig. 5a). Despite the shape not being convex, the method learns a linear separation of these two rings (cf. fig. 5b). The loss values (loss $= \sum_{i=1}^{K} \lambda_i$) decrease consistently with the number of iterations (cf. fig. 5c). Optimizing kernel bandwidth σ using the gradient descent stabilizes as the σ gradient diminishes (cf. fig. 5d). A snapshot of the optimized trajectory is in figs. 5e to 5j.

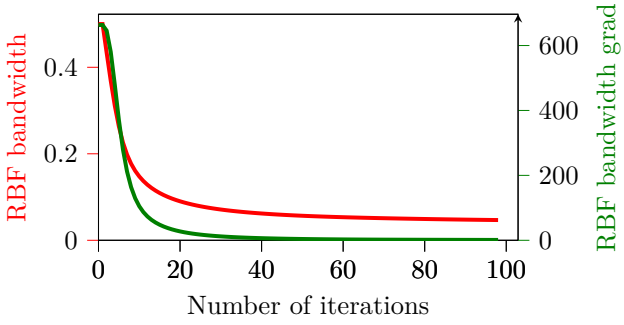
(a) Benchmark dataset in 2D.



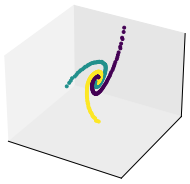
(b) Eigen-embeddings of the benchmark dataset.



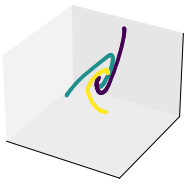
(c) Loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training.



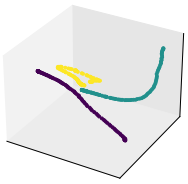
(d) RBF bandwidth σ and its gradient during training.



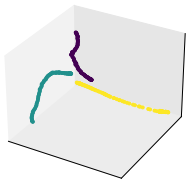
(e) Epoch 0



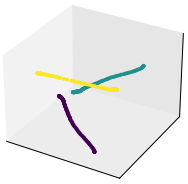
(f) Epoch 30



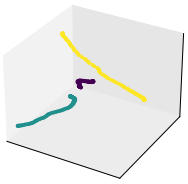
(g) Epoch 50



(h) Epoch 60

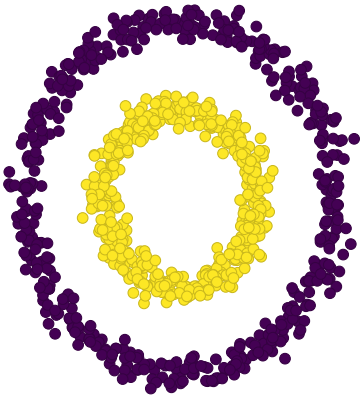


(i) Epoch 70

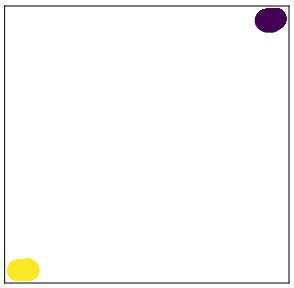


(j) Epoch 80

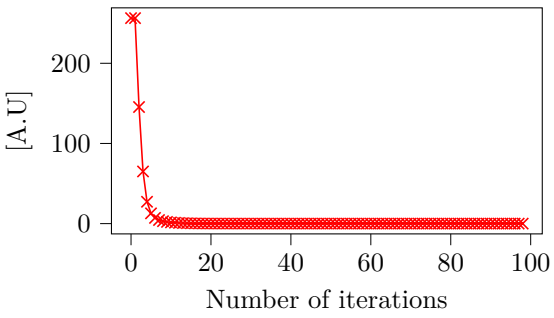
Figure 6: Benchmark dataset three spirals in 2D (cf. fig. 6a). Despite the shape not being convex, the method learns a linear separation of these three spirals (cf. fig. 6b). The loss values (loss $= \sum_{i=1}^{K} \lambda_i$) decrease consistently with the number of iterations (cf. fig. 6c). Optimizing kernel bandwidth σ using the gradient descent stabilizes as the σ gradient diminishes (cf. fig. 6d). A snapshot of the optimized trajectory is in figs. 6e to 6i.



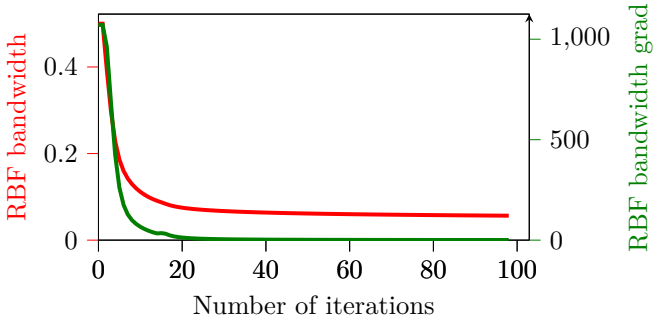
(a) Benchmark dataset in 2D.



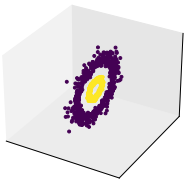
(b) Eigen-embeddings of the benchmark data.



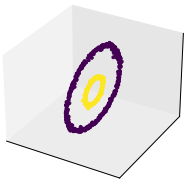
(c) Loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training.



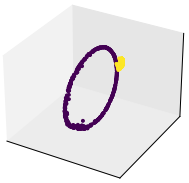
(d) RBF bandwidth σ and its gradient during training.



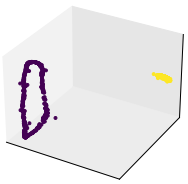
(e) Epoch 0



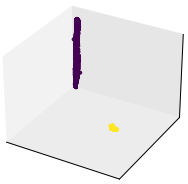
(f) Epoch 10



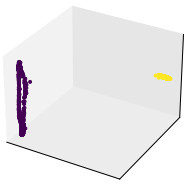
(g) Epoch 15



(h) Epoch 20

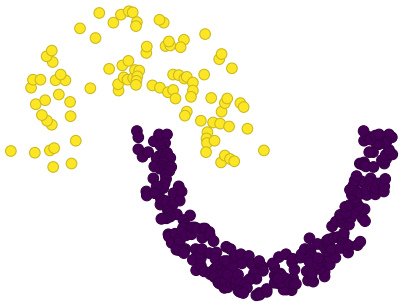


(i) Epoch 25

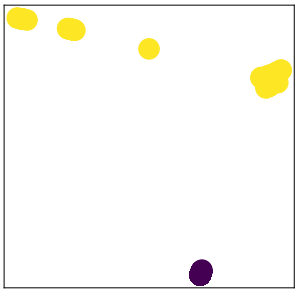


(j) Epoch 30

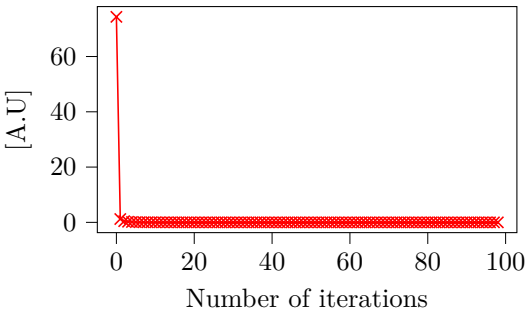
Figure 7: Benchmark dataset two rings in 2D (cf. fig. 7a). Despite the shape not being convex, the method learns a linear separation of these two rings (cf. fig. 7b). The loss values (loss $= \sum_{i=1}^{K} \lambda_i$) decrease consistently with the number of iterations (cf. fig. 7c). Optimizing kernel bandwidth σ using the gradient descent stabilizes as the σ gradient diminishes (cf. fig. 7d). A snapshot of the optimized trajectory is in figs. 7e to 7j.



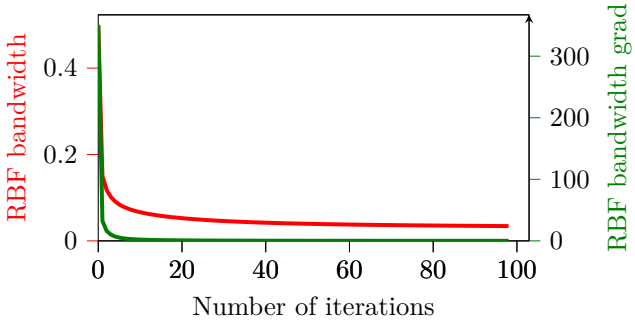
(a) Benchmark dataset in 2D.



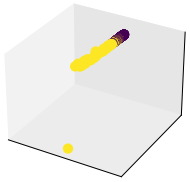
(b) Eigen-embeddings of the benchmark dataset.



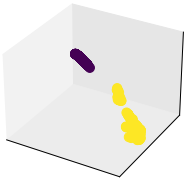
(c) Loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training.



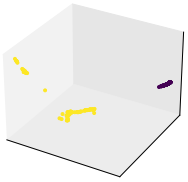
(d) RBF bandwidth σ and its gradient during training.



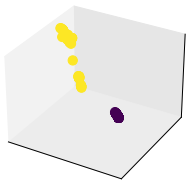
(e) Epoch 0



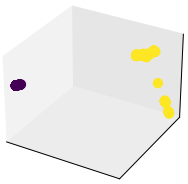
(f) Epoch 25



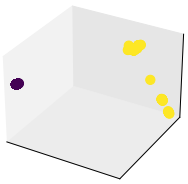
(g) Epoch 30



(h) Epoch 50

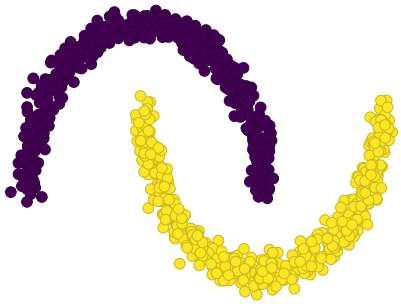


(i) Epoch 70

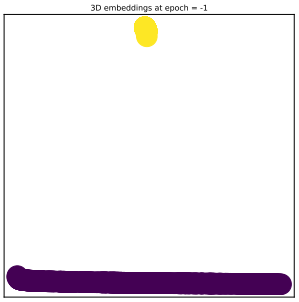


(j) Epoch 98

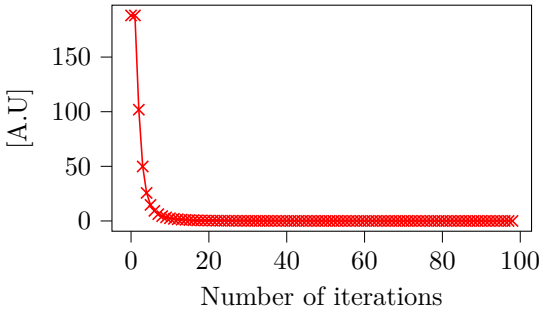
Figure 8: Benchmark dataset two (asymmetric) half moons in 2D (cf. fig. 8a). Despite the shape not being convex, the method learns a linear separation of these two half moons (cf. fig. 8b). The loss values (loss $= \sum_{i=1}^{K} \lambda_i$) decrease consistently with the number of iterations (cf. fig. 8c). Optimizing kernel bandwidth σ using gradient descent stabilizes as the σ gradient diminishes (cf. fig. 8d). A snapshot of the optimized trajectory is in figs. 8e to 8j.



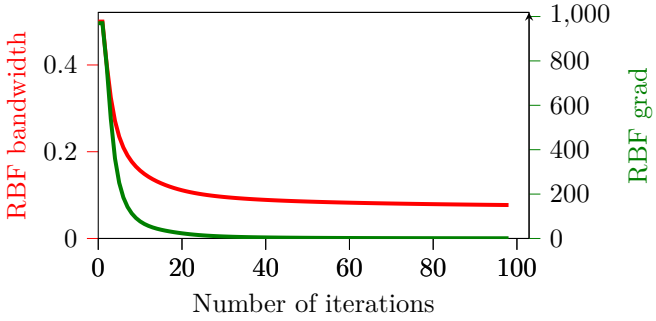
(a) Benchmark dataset in 2D.



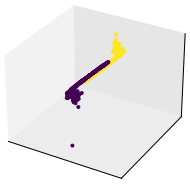
(b) Eigen-embeddings of the benchmark dataset.



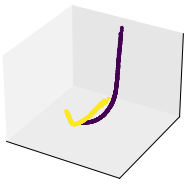
(c) Loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training.



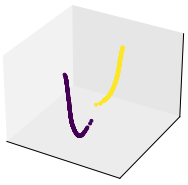
(d) RBF bandwidth σ and its gradient during training.



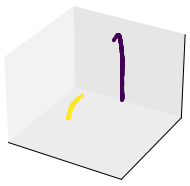
(e) Epoch 0



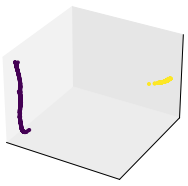
(f) Epoch 10



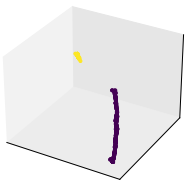
(g) Epoch 20



(h) Epoch 30



(i) Epoch 40

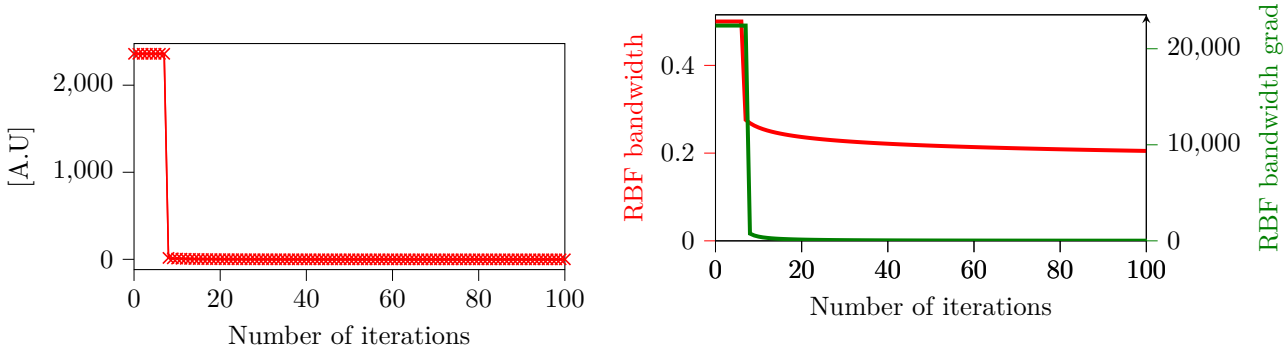


(j) Epoch 50

Figure 9: Benchmark dataset two moons in 2D (cf. fig. 9a). Despite the shape not being convex, the method learns a linear separation of these two moons (cf. fig. 9b). The loss values (loss $= \sum_{i=1}^{K} \lambda_i$) decrease consistently with the number of iterations(cf. fig. 9c). Optimizing kernel bandwidth σ using gradient descent stabilizes as the σ gradient diminishes (cf. fig. 9d). A snapshot of the optimized trajectory is in figs. 9e to 9j.

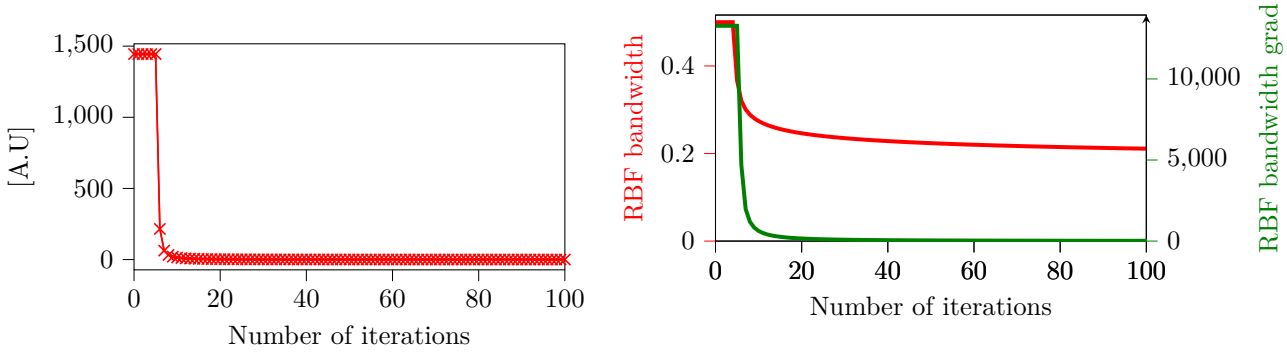
A.3 Optimization trajectory on the image and text datasets

For each tested dataset, we present the loss (loss $= \sum_{i=1}^{K} \lambda_i$) over iterations (cf. figs. 10a, 11a, 12a, 13a, 14a, 15a and 16a) together with the trajectory of the RBF bandwidth and its gradient (cf. figs. 10b, 11b, 12b, 13b, 14b, 15b and 16b). The convexity of the method persists for each dataset, as is evident from the always-decreasing magnitude of the RBF bandwidth gradient. Notice that although RBF bandwidth always starts at $\sigma = 0.5$, its starting gradient varies across different types of datasets.



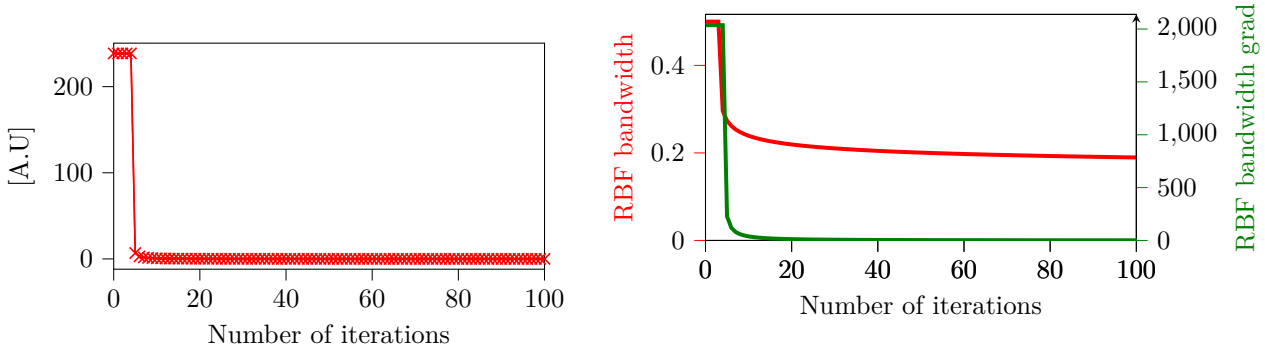
(a) Loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training. (b) RBF bandwidth σ and its gradient during training.

Figure 10: The loss values (loss $= \sum_{i=1}^{K} \lambda_i$) over the iterations for the COIL20 dataset in fig. 10a.Optimization of the σ on the JAFFE dataset in fig. 10b.



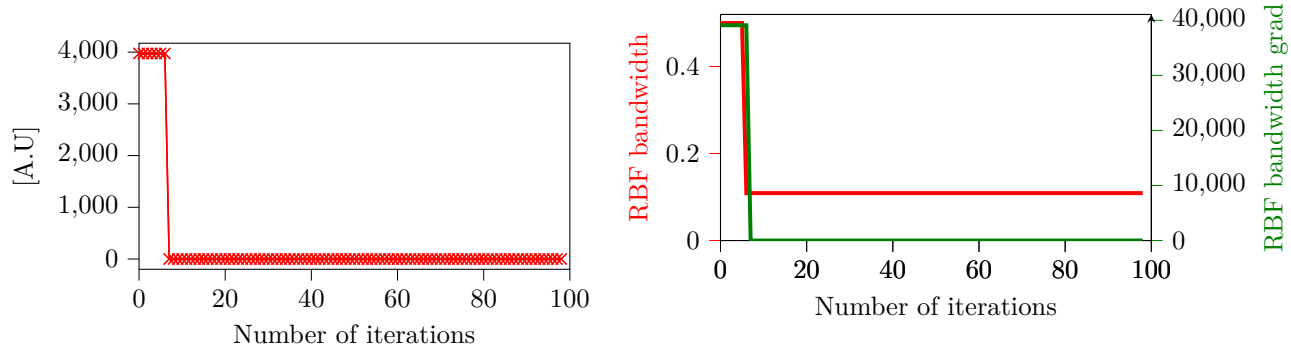
(a) Loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training. (b) RBF bandwidth σ and its gradient during training.

Figure 11: The loss values (loss $= \sum_{i=1}^{K} \lambda_i$) over the iterations for the ORL dataset in fig. 11a.Optimization of the σ on the ORL dataset in fig. 11b.



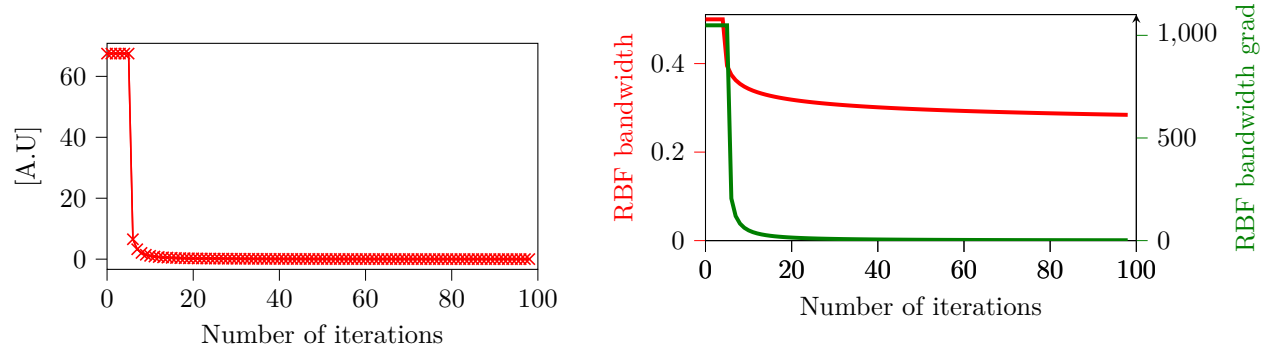
(a) Loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training. (b) RBF bandwidth σ and its gradient during training.

Figure 12: The loss values (loss $= \sum_{i=1}^{K} \lambda_i$) over the iterations for the YALE dataset in fig. 12a.Optimization of the σ on the YALE dataset in fig. 12b.



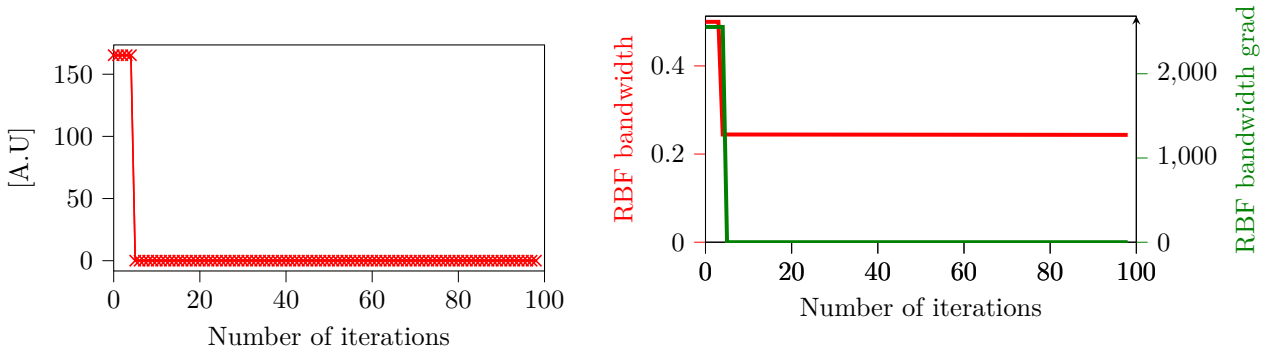
(a) Loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training. (b) RBF bandwidth σ and its gradient during training.

Figure 13: The loss values (loss $= \sum_{i=1}^{K} \lambda_i$) over the iterations for the BA dataset in fig. 13a.Optimization of the σ on the BA dataset in fig. 13b.



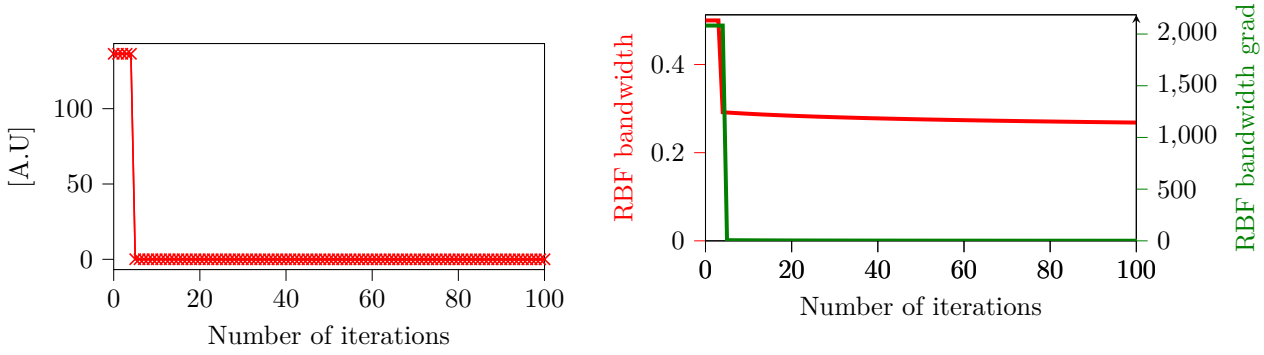
(a) Loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training. (b) RBF bandwidth σ and its gradient during training.

Figure 14: The loss values (loss $= \sum_{i=1}^{K} \lambda_i$) over the iterations for the TR11 dataset in fig. 14a.Optimization of the σ on the TR11 dataset in fig. 14b.



(a) Loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training. (b) RBF bandwidth σ and its gradient during training.

Figure 15: The loss values (loss $= \sum_{i=1}^{K} \lambda_i$) over the iterations for the TR41 dataset in fig. 15a.Optimization of the σ on the TR41 dataset in fig. 15b.



(a) Loss function (loss $= \sum_{i=1}^{K} \lambda_i$) over during training. (b) RBF bandwidth σ and its gradient during training.

Figure 16: The loss values (loss $= \sum_{i=1}^{K} \lambda_i$) over the iterations for the TR45 dataset in fig. 16a.Optimization of the σ on the TR45 dataset in fig. 16b.