# Graph sparsification for Kernel Methods

**Hugo Dupré**
MVA Master's degree
École Normale Supérieure
Cachan, France
hugo.dupre@ensae.fr

**Joseph Lam**
MVA Master's degree
École Normale Supérieure
Cachan, France
joseph.lam@ensae.fr

## Abstract

Data sparsification is the process that takes as input a distance/similarity matrix between data points and outputs a new version of it with less non-zero elements. A good sparsifier reduces the complexity of the data (density), while maintaining its underlying structure. As a result, this can improve the data quality by removing the effect of noisy data correlations, and thus assist any machine learning task that follows. In this paper, two methods of sparsification will be considered. One is a proposed algorithm based on Mean Shift and the distribution of distances from a data point. The other is entropic affinities which gives good results with a reduced number of hyperparameters.

## 1 Introduction

Several tools for density analysis are only efficient in one dimension, whereas, most of the time, feature spaces of interest are multidimensional. The proposed algorithm considers data points as "viewers" in an analog way as Kalogeratos and Likas ([1]) and it focuses on the distribution of distances between the viewer and the other data points. In particular, we apply a kernel-based method, the mean shift algorithm, in order to find the modes of the distribution of distances. The only assumption relies on the unimodality of the distribution of members of a cluster. Finally, the similarity matrix is sparsified based on the positions of the points with respect to one another and the positions of the modes.

Gaussian affinities are commonly used in graph-based methods such as spectral clustering or nonlinear embedding. In this framework, hyperparameter tuning is a key to good performance: the bandwidth $\sigma$ is usually set to some rule of thumb or estimated via cross-validation. This method can be computationally prohibitive, and pretty frustrating. Moreover, the best results of the algorithm may not be achieved for a single value of $\sigma$ for all the points, but rather for a separate bandwidth for every data point, in which case the existence of automatic procedure is vital. Hinton and Roweis (2003) introduced a way to set the scale individually for each point so that it has a distribution over neighbors with a desired perplexity, or effective number of neighbors. This gives very good affinities that adapt locally to the data. Also, it reduces the number of hyperparameters in the final model which is preferable.

## 2 Definition of Mean Shift

We would like to find denser regions in the feature space, which correspond to the modes of the probability density function. In order to find these regions, we work with a non parametric estimate of the density function thanks to kernel density estimation.

Firstly, a kernel $K$ is defined as a function satisfying the following requirements:

- $\int_{\mathbb{R}^d} K(x)dx = 1$

- $K(x) \geq 0$ for all x

Then, given $(x_i)_{i \in \{1..n\}}$ the observed points in the feature space, the kernel density estimate is:

$$\hat{f} : x \to \frac{1}{n\sigma^d} \sum_{i=1}^{n} K(\frac{x - x_i}{\sigma})$$

where $\sigma^2$ is the bandwidth parameter, $K$ the kernel and $d$ the dimension of the feature space.

The estimation is sensitive to changes in $\sigma^2$ and the way to determine this parameter is further discussed in the part on entropic affinities.

Then the modes of the density are located at zeros of $\nabla \hat{f}$.

Once the density function has been estimated, it is possible to find local maxima using gradient ascent methods.

$$x_{t+1} = x_t + \eta \nabla \hat{f}(x_t) \tag{1}$$

Now suppose, $K$ is radially symmetric, we can define $k$ such that:

$$K(x) = c_{k,d} k(||x||^2) \tag{2}$$

where $c_{k,d}$ is a normalization constant. Then $g = -k'$ gives:

$$\nabla f(x) = 2\frac{c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (x_i - x) g(||\frac{x - x_i}{\sigma}||^2) \tag{3}$$

$$= 2\frac{c_{k,d}}{nh^{d+2}} [\sum_{i=1}^{n} g(||\frac{x - x_i}{\sigma}||^2)][\frac{\sum_{i=1}^{n} x_i g(||\frac{x - x_i}{\sigma}||^2)}{\sum_{i=1}^{n} g(||\frac{x - x_i}{\sigma}||^2)} - x] \tag{4}$$

Now, $\sum_{i=1}^{n} g(||\frac{x - x_i}{\sigma}||^2)$ is proportional to a density estimate of the $(x_i)$ at $x$ with kernel proportional to $x \to g(||x||^2)$. And the second term is the mean shift:

$$m_\sigma(x) = \frac{\sum_{i=1}^{n} x_i g(||\frac{x - x_i}{\sigma}||^2)}{\sum_{i=1}^{n} g(||\frac{x - x_i}{\sigma}||^2)} - x \tag{5}$$

The mean shift gives away the direction of the gradient.

Finally, going back to the gradient ascent method for finding local maxima:

$$x_{t+1} = x_t + m_\sigma(x_t) \tag{6}$$

and the initial point is taken equal to a data point for each one of them, so that we might find the closest mode associated to each data point.

Thus, for a correct choice of $\sigma$, we find all the modes. Indeed, a mode is associated with a dense region. Hence, for each mode of the estimated density function, there exists a data point which will allow the algorithm to find that mode.

A setback of Kernel Density Estimation arises in high dimension because of the curse of dimensionality. Our methodology holds thanks to the application to the distribution of the distances from a viewer, which is a one-dimensional vector.

## 3 Sparsification with Mean Shift from the point of view of viewers

Let us consider a sparsification method exploiting Mean Shift applied to one-dimensional vectors. Indeed, for each data point, the vector of similarities from the point is a representation of the complete

data set. From this remark, we would like to use the information from such representations in order to sparsify the similarities matrix.

The result given by the mean shift algorithm that we are going to use is the positions of the modes of the distribution. A criterion is built based on the information on modes: whenever two data points are separated by at least two modes, their similarity is set to 0.

In order to tune the sparsification effect, the sparsification method can be modified into a voting procedure. Instead of directly setting the similarity to 0, a vote should be added whenever a viewer considers that two data points should be separated. Once a threshold for the number of votes has been reached, the considered similarity is set to 0. Trivially, we notice that direct sparsification is a special case of the voting framework for a threshold of 1 vote.

Some thought has been put towards an optimal selection of viewers. Indeed, reducing the number of viewers considered is a great step towards reducing the complexity of the algorithm.

We introduce the concept of a good viewer. Two ideas allow us to judge the quality of a viewer. The number of modes seen by a viewer and the number of votes it would issue. The number of modes seen by a viewer is given by the number of clusters found by the mean shift algorithm. And the number of votes a viewer would issue is found following the sparsification procedure described above. Both criteria are useful in order to discriminate good viewers from bad viewers.

But the selection of a viewer already begins with the sampling of a candidate. The naive baseline method is choosing a viewer via uniform sampling over the data points. Morally, we would like to cover the space as much as possible with as few viewers as possible. That is the reason why an algorithm is devised following the design of K-means++ initialization introduced in [2].

The first candidate is uniformly sampled among all data points. Let $L_c$ be the set of candidates that have already been considered. We would like to sample the next candidate far enough from those that have already been considered. Now, the pairwise similarities between the points are to be exploited. Thus, we define weights to sample from, which are low when the potential candidates are close to past candidates, and high otherwise.

The precise method to aggregate the similarities into weights depend on the kernel that is used. As a rule of thumb, if the kernel has exponentially decreasing tails (like gaussian affinities), the log of the similarity from the potential candidate to the past candidates should be summed with a minus sign. With slowly decreasing tails, the sum of the inverse of the similarities should be appropriate. Finally, with a flat kernel, only points out of range of any past candidate should have nonzero uniform weights.

Since the goal of our undertaking is the sparsification for kernel-based methods and we have proposed the Mean Shift algorithm on viewers which is a kernel-based method itself, our next step will be to consider the choice of bandwidth for gaussian kernels. This will lead to the study of a sparsification method using varying bandwiths: entropic affinities.

## 4 Definition of entropic affinities

The considered approach is proposed by Vladymyrov et Al. ([3]). For a given data set $(x_1, .., x_n) \in (\mathbb{R}^d)^n$, consider an isotropic kernel density estimator of width $\sigma$. Then, for a given point $x \in \mathbb{R}$, we can derive the posterior distribution of the point with respect to the data set. It is a discrete distribution $p(x, \sigma)$ such that $\forall j \in \{1, .., n\}$:

$$p_j(x, \sigma) = \frac{K\big(\|\frac{x-x_j}{\sigma}\|^2\big)}{\sum_{k=1}^{n} K\big(\|\frac{x-x_k}{\sigma}\|^2\big)} = \frac{K\big((\frac{d_j}{\sigma})^2\big)}{\sum_{k=1}^{n} K\big((\frac{d_k}{\sigma})^2\big)} \tag{7}$$

We will consider the case where $K$ is the Gaussian kernel. The idea behind entropic affinities is to set $\sigma$ individually for each point $x_i$ of the data set such that the entropy of the distribution $p(x_i; \sigma)$ equals to $\log(K)$, where $K$ is the user-set perplexity parameter. This is equivalent to setting the perplexity of $p(x_i; \sigma)$ to $K$. The perplexity, widely used in natural language processing (Manning & Schütze, 1999), has an intuitive interpretation. A perplexity of $K$ in a distribution $p$ over $n$ neighbors means $p$ provides the same surprise as if we were to choose among $K$ equiprobable neighbors. Thus, applying this to every datapoints yields a collection of bandwidth $(\sigma_1, .., \sigma_n)$ called entropic affinities.

The entropy of the distribution $p(x, \sigma)$, when $K(x) = \exp(-x^2)$ is the Gaussian kernel, is given by the following formula :

$$H(x, \beta) = H(p(x, \sigma)) = \beta \sum_{i=1}^{n} p_i(x, \beta) d_i^2 + \log\big(\sum_{i=1}^{n} \exp(-d_i^2 \beta)\big) \tag{8}$$

Where $\beta = \frac{1}{2*\sigma^2}$ is the precision parameter. It is easier to work with $\beta$ than $\sigma$. The goal of entropic affinities is to find the solution $(\beta_1^*, .., \beta_n^*)$ of the following system :

$$\forall i \in \{1, .., n\} \quad F(x_i, \beta, K) \triangleq H(x_i, \beta) - \log(K) = 0 \tag{9}$$

$F(x, \beta, K)$ is differentiable with respect to $\beta$, which gives :

$$\frac{\partial F(x, \beta, K)}{\partial \beta} = \frac{\partial H(x, \beta)}{\partial \beta} = -\beta(\sum_{i=1}^{n} p_i d_i^4 - \big(\sum_{i=1}^{n} p_i d_i^2\big)^2)$$

The first derivative of $F(x, \beta, K)$ is always negative whenever $\beta > 0$, given that the neighboring points are not equidistant from $x$. This means that $F(x, \beta, K)$ is a monotonically decreasing function of $\beta$ that decreases from $\log(n)$ for $\beta = 0$ to 0 for $\beta \to \infty$. Thus, the system (3) is well defined for any value of $\beta > 0$ and has a unique root $\beta^*(x, K)$ for any $K \in [0, n]$ and any $x \in \mathbb{R}^d$.

This is the proof that entropic affinities are well defined. The conclusion on the existence of the entropic affinities is given by Equation 10.

$$\forall n \in \mathbb{N}^* \ \ \forall (x_1, ., x_n) \in \mathbb{R}^d \ \ \forall K \in [0, n] \ \ \forall i \in \{1, ., n\} \ \ \exists!\, \sigma_i^* \in \mathbb{R}_+^* \ \ H(p(x_i, \sigma_i^*)) = \log(K) \tag{10}$$

Finally, the sparsified graph is created by setting to zero every affinity value less than a specified threshold $\epsilon$.

## 5 Experiments

We coded, from scratch, the entropic affinities method, on Python using the `scipy.optimize.newton` method in order to solve the root-finding problem. The code for it and the proposed method of sparsification using viewers is available on GitHub : `https://github.com/Caselles/Entropic_Affinities`.

We experimented the algorithms on three different classic clustering data sets : moons, circles and blobs, using scikit-learn ([4]). Each data set has two clusters, convex (blobs) or not (moons, circles), and the goal is to compute a sparse graph using different sparsifiers (entropic affinities, K-NN, Mean Shift on viewers), then perform a clustering task. The objective is to study how particular sparsifying methods and particular sparsities affect the clustering task, and more generally the underlying structure of the data. Entropic affinities will be compared with the regular KNN method. The results were very similar on the three data sets, thus we will focus on the circles data set results for Figures and quantitative results. In the following experiments, the focus is set on finding the potential of our algorithms. That is the reason why for Mean Shift on viewers

In Figure 1 two sparse (93% of sparsity) graphs created on the two circles data set are presented (500 data points), one created with KNN and the other with entropic affinities.

Both methods managed to preserve the underlying structure of the data set, while creating a connected and very sparse graph.

Now, let us create the graph with 93% sparsity for Mean Shift on viewers as in 2. Although the sparsity level is as high as in1, the figure does not show what we would expect from a sparse graph. This means that for K-NN and entropic affinities, the links are gathered more densely along the circles. This hints at the fact that the results will not be as good as for K-NN and entropic affinities. When we impose a sparsity of 98%, the links are reduced to the strongest ones along the circles, with Mean Shift on viewers.

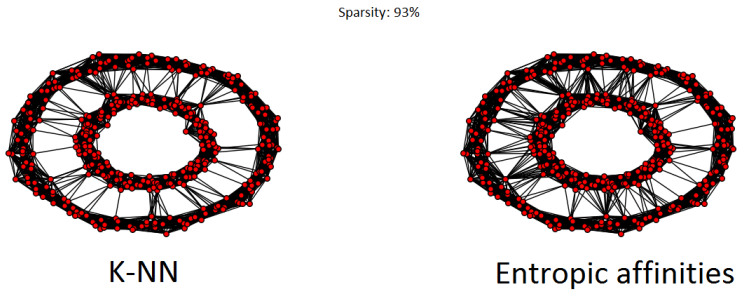


Figure 1: Creating sparse graphs using K-NN and Entropic affinities, on the circles data set

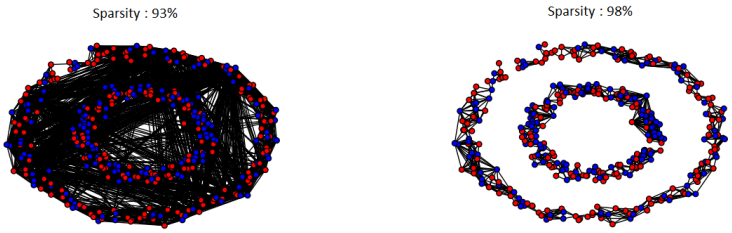


Figure 2: Creating sparse graphs using MeanShift with viewers for different thresholds (13 on the left and 6 on the right). Red color represents viewers.

In order to better understand the entropic affinities, it is interesting to plot statistics about the distribution of the entropic affinities for a range of perplexities. In Figure 3, we can see the evolution of the mean and standard deviation of the entropic affinities, as the perplexity increases. Logically, the bigger the perplexity, the bigger the mean of the entropic affinities : the entropic affinities have to be large in order to provide the same surprise as if we were to choose among a large number $K$ of equiprobable neighbors. This is exactly what we observed. Moreover, the standard deviation also increases as $K$ increases.

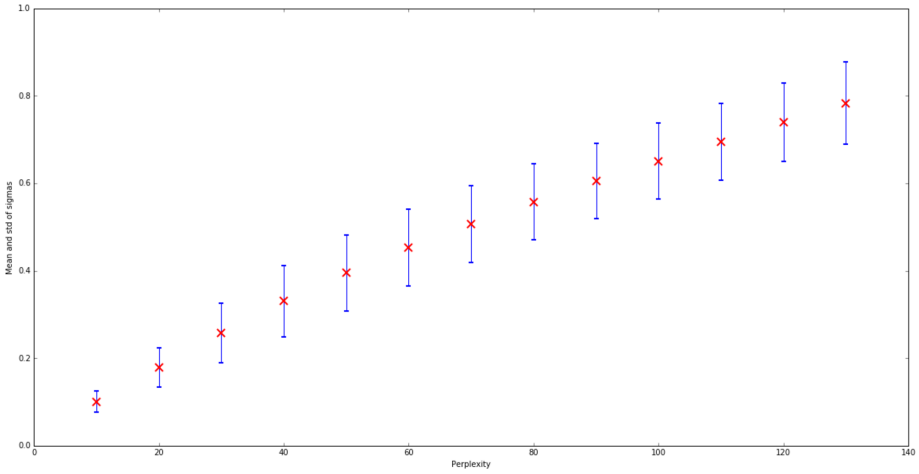


Figure 3: Distribution of the entropic affinities as a function of the perplexity (circles data set)

Next, we can plot the sparsity as a function of the parameters of different methods. Thus we plot it as a function of the entropy (which is $\log_2(K)$) for entropic affinities, the number of nearest neighbors for KNN and the vote threshold for MeanShift. This is what is depicted in Figure 4 and 5:

The decrease in sparsity is linear for the KNN method, and quasi-linear for the entropic affinities method. This is essentially what we expected : the graph is sparser if $k$ or $K$ is small. The evolution

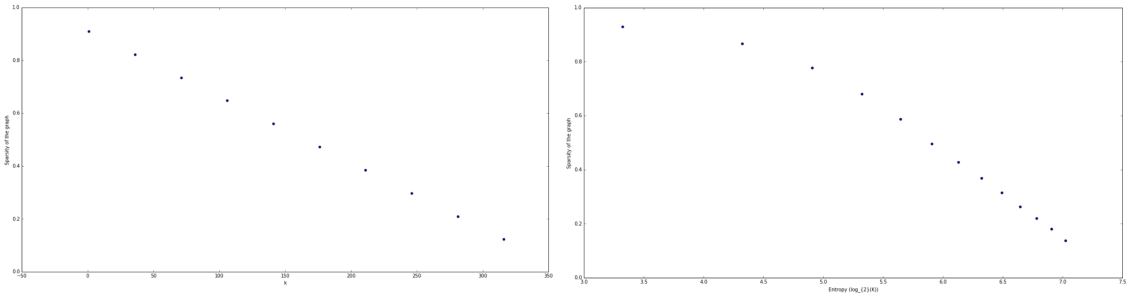

Figure 4: Sparsity as a function of k, and sparsity as a function of the entropy, on the circles data set.

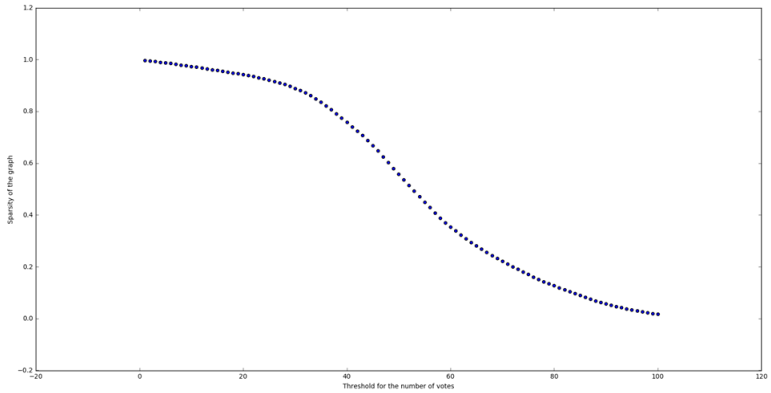


Figure 5: Sparsity with Mean Shift on viewers as a function of the threshold for the votes, on the circles data set.

of sparsity for Mean Shift on viewers, on the other hand, starts slowly and accelerates until it reaches a turning point after which it decelerates. That is the reason why, if a high sparsity level is expected, finetuning the threshold will only lead to small modifications of the sparsity.

Now, we will try to compare, for different sparsities, how well the spectral properties are conserved by each method. For this, we will use two distance criteria.

1. $L_1$-Normalize the 2 vectors of eigenvalues $v$ and $v_s$ ($v$ for the original graph, and $v_s$ for the sparsified one). Then computing a simple dot product between the two vectors measures the similarity of the two distributions. We would rather uses distances than similarities, so we just apply the inverse function to these similarities to get the distance. Hence, the distance is defined by :
$$D_1(G, G_s) = \frac{1}{\langle v | v_s \rangle}$$
2. If $(v_1, .., v_k)$ are the top-k eigenvectors of $L$ (laplacian form of $G$, the original un-sparsified graph) then $d_i = (v_i^T L v_i - v_i^T L_s v_i)$ is a difference in terms of the $i^{th}$ eigenvector, where $L_s$ is the laplacian form of $G_s$ (the original sparsified graph). Thus, the final distance used to compare the spectral properties is :
$$D_2(G, G_s) = \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} (v_i^T L v_i - v_i^T L_s v_i)^2$$

We used both distances, with each method, for a range of sparsities. The results are available in Figure 6.

Most of these results are intuitive : the distance between the original graph and the sparsified graph is getting bigger as the sparsity increases ; except for Mean Shift on viewers with $D_1$ which decreases

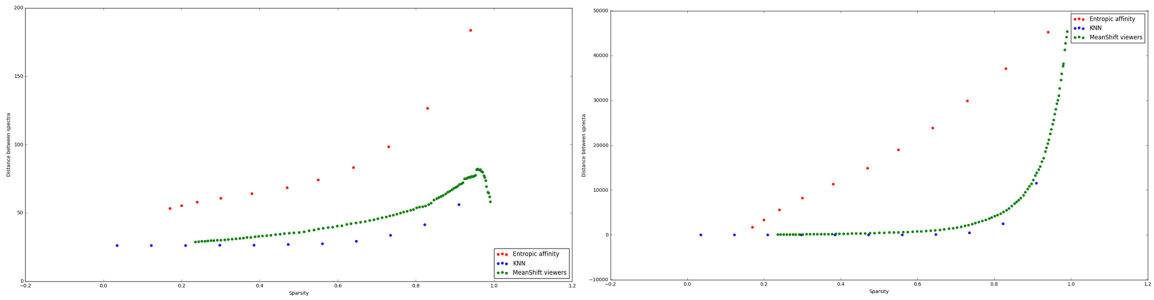


Figure 6: $D_1$ (left) and $D_2$ (right) comparison on the circles data set.

once sparsity gets close to 1. Notice that KNN seems to perform better than the other methods on this data set, although Mean Shift on viewers gives results which are almost as good. The same results were observed on the other toy data sets. And the methods seem to have the same behaviour: the distance increases exponentially once the sparsity is high (close to 90%).

Finally, in order to complete the comparison of the methods, we will compare the spectral clustering results (in terms of accuracy) of the two methods, as a function of sparsity. The results are depicted in Figure 7.

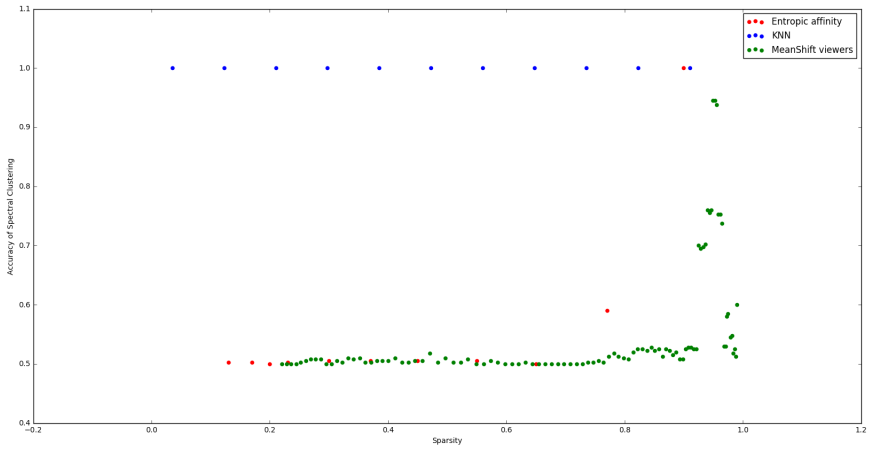


Figure 7: Spectral clustering results comparison on the circles data set.

Since the data are well separated (even though the clusters are non-convex), the clustering results reach excellent levels. Once again KNN seems to perform better, but it is mostly due to the use of a toy data set, it might not stand for real and richer data sets. The important result is that the entropic affinities gives very satisfying results for an appropriate sparsity level. On the other toy data sets, the sparsity level for which entropic affinities allow for good clustering results is larger, which is very encouraging. However, Mean Shift on viewers seems to have a really narrow space for which it gives good clustering results (not as good as the other sparsification methods).

Thus, we can conclude that K-NN has given better results overall. Although Mean Shift on viewers has given promising results in terms of spectral properties conservation, entropic affinities gives a better accuracy for the clustering task. These results should be qualified by the fact that the experiments were run on toy data sets. In particular, we would expect K-NN to fare worse as the dimension increases.

## 6 Discussion

Over the course of our experiments, we have considered another algorithm for viewer-based sparsification. It uses Hartigans' dip test [5], which verifies unimodality, where a function F(t) is unimodal with mode the region $s_m = \{(t_L, t_U) : t_L \leq t_U\}$ if it is convex in $s_L = (-\infty, t_L]$, constant in $[t_L, t_U]$ and concave in $s_U = [t_U, \infty)$. Now for bounded input functions $F, G$, let $\rho(F, G) = \max_t |F(t) - G(t)|$ and let $\mathcal{U}$ be the class of cdfs of all unimodal distributions.

Then the dip statistic of a cdf F is given by:

$$dip(F) = \min_{G \in \mathcal{U}} \rho(F, G) \tag{11}$$

In Hartigan ([5]), it is demonstrated that dip values of uniform distributions are stochastically larger than other unimodal distributions. Thus, the class of uniform distributions $\mathcal{U}$ is the most appropriate for the null hypothesis of the unimodality test. Then the p-value is computed comparing the dip value of the observations and the dip value of samples from the uniform distribution over $[0, 1]$:

$$pval = \frac{\sum_{r=1}^{b} \mathbb{1}_{\{dip(F_n) \leq dip(U_n^r)\}}}{b} \tag{12}$$

where $F_n$ is the empirical cdf given by the $n$ observations, $U_n^r$ the $r$-th empirical cdf given by $n$ observations sampled from the uniform distribution and $b$ the number of sets of observations sampled from the uniform distribution.

This p-value allows us to determine if the null hypothesis, that $F_n$ is unimodal, is accepted or rejected in favor of $F_n$ being multimodal. If $F_n$ is determined to be unimodal, then we can test whether it was generated from a uniform distribution using Kolmogorov-Smirnov goodness-of-fit test [6].

An algorithm exploiting Hartigan's dip test has been attempted in order to find all the modes of the distribution. It can be described as follows:

**Algorithm 1** Detection of modes using Hartigan's dip test.

Set $\alpha_1$ and $\alpha_2$ as thresholds for the p-values.
Initialize $(x_L, x_U) = \mathbb{R}$.
**procedure** RECURSION($(x_L, x_U)$)
    Apply the dip test algorithm in $(x_L, x_U)$. A mode is found in $[t_L, t_U]$.Take it at $\frac{t_L + t_U}{2}$.
    Determine $pval_{dip}$ of the unimodality test using $b$ sets of samples from a uniform distribution.
    **if** $pval_{dip} \geq \alpha_1$ **then**
        The distribution is unimodal.
        Apply a Kolmogorov-Smirnov test to determine if the distribution is uniform.
        **if** $pval_{KS} \geq \alpha_2$ **then**
            The distribution is not uniform.
            **return** $\frac{t_L + t_U}{2}$.
        **else**
            **return** NONE.
        **end if**
    **else**
        **return** $\{$RECURSION($(x_L, t_L)$), $\frac{t_L + t_U}{2}$, RECURSION($(t_U, x_U)$)$\}$
    **end if**
**end procedure**

However, the algorithm demonstrates poor results, because limiting the space using $t_L$ and $t_U$ is not restricting enough, and this leads us to consider dense regions as modes, although they simply are artifacts left by a mode that has been found previously.

Besides, we show that using Hartigan's test for unimodality is hopeless for our task. Indeed, let us describe the design of our experiment. Two different setups are compared, both consist of three clusters generated by gaussian distributions. We assume there is a point x in one mode. The difference between the two setups lies in the position of the centers of the two other clusters with respect to x.

In the first setup, the centers are equally distant from x, and not in the second setup. Then we can compare the dip and p values from the point of view of x in both setups, and we repeat the experiment, counting how many times one value is greater than the other.

During our experiment, out of a thousand repetitions, the dip value of the first setup was always larger and the p value lower. Hence, although Hartigan's dip test is a measure of multimodality, it does not increase with the number of modes, which makes it ineffective for our task.

## 7 Conclusion

Several graph sparsification methods have been studied. One of the proposed methods, Mean Shift on viewers, is very promising in terms of conservation of spectral properties, although the sensitivity to the threshold on the number of votes for the clustering task remains a major setback. Entropic affinities seem to be a good method which tunes the bandwidth automatically. Even though the parameter K still needs to be tuned, it reduces the number of hyperparameters to tune by providing an intuitive and efficient concept. Depending on the dataset, it may be the optimal solution : further studies can be done on this subject.

## References


[1] A. Kalogeratos and A. Likas, "Dip-means: an incremental clustering method for estimating the number of clusters," in *Advances in neural information processing systems*, pp. 2393–2401, 2012.

[2] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035, Society for Industrial and Applied Mathematics, 2007.

[3] M. Vladymyrov and M. A. Carreira-Perpinán, "Entropic affinities: Properties and efficient numerical computation.," in *ICML (3)*, pp. 477–485, 2013.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] J. A. Hartigan and P. Hartigan, "The dip test of unimodality," *The Annals of Statistics*, pp. 70–84, 1985.

[6] M. A. Stephens, "EDF statistics for goodness of fit and some comparisons," *Journal of American Statistical Association*, vol. 69, no. 347, pp. 730–737, 1974.