# Local Cluster Cardinality Estimation for Adaptive Mean Shift

Étienne Pepin (etienne.pepin78@gmail.com)

August 17, 2025

**Abstract**

This article presents an adaptive mean shift algorithm designed for datasets with varying local scale and cluster cardinality. Local distance distributions, from a point to all others, are used to estimate the cardinality of the local cluster by identifying a local minimum in the density of the distance distribution. Based on these cardinality estimates, local cluster parameters are then computed for the entire cluster in contrast to KDE-based methods, which provide insight only into localized regions of the cluster. During the mean shift execution, the cluster cardinality estimate is used to adaptively adjust the bandwidth and the mean shift kernel radius threshold. Our algorithm outperformed a recently proposed adaptive mean shift method on its original dataset and demonstrated competitive performance on a broader clustering benchmark.

## 1 Introduction

The mean shift algorithm moves all data points towards a local average. Many methods have been implemented over the years to determine which points contribute to this local average. The most basic one simply requires a bandwidth parameter, while adaptive mean shift methods, such as Comaniciu and Meer 2002, change the bandwidth parameter locally to adapt to local data, often with a local density estimation. In this paper, we introduce a method that estimates the cardinality of local clusters and uses this information to compute cluster parameters directly, rather than relying on local density estimation. We hypothesize that this method is more robust to variations in both cluster size and density. We compared our method with a recent adaptive mean shift algorithm and found that our approach achieves superior performance on the paper's original dataset and performs competitively on a general clustering benchmark. The code used to produce those results is available online[1].

The local cluster cardinality is estimated using distance distributions, which are the distribution of distances from one point to other points. Our algorithm searches for the minimum density between the leftmost mode (representing the local cluster) and the rightmost mode of the distance distributions. In our model, this point delineates the boundary of the local cluster and determines its cardinality. The estimated cardinality is then used to calculate statistics on the local cluster, such as its distance variance. Finally, this information is used during the mean shift process to adapt the bandwidth and define a distance threshold for selecting points to include in the local average.

We tested our algorithm against the results of a recent adaptive mean shift algorithm (Ren et al. 2014) and achieved better performance on most of the datasets they used. We also compared our algorithm with the benchmarks from Gagolewski 2022, and the results were mixed, with our algorithm performing better on some datasets. One distinct advantage our algorithm has over the algorithms included in this benchmark is that it does not require the number of clusters as an input.

[1] https://github.com/pEtienn/Adaptive-mean-shift-based-on-local-cluster-cardinality-estimation

## 2 Distance Distributions

A distance distribution is the distribution of distances from one point to other points. Distance distributions can be derived mathematically for a particular distribution. For example, the isotropic Gaussian distribution in $d$ dimensions has a chi distribution with $d$ degrees of freedom (see Section 8.3 in the Appendix). Figure 1 illustrates how the chi Cumulative Distribution Function (CDF) is derived by integrating a Gaussian density over an $n$-sphere (2-sphere in the Figure) of a given radius.

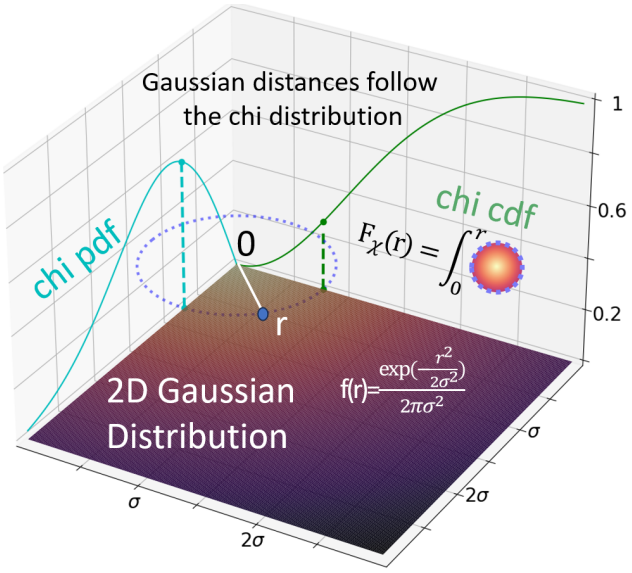


**Figure 1:** Distance distribution visualization

Figure 1 is also representative of the way many distance distributions behave. As the number of dimensions tends towards infinity, the distance distributions of independent and identically distributed (i.i.d.) distributions tend towards a normal distribution (see Section 8.2 in the Appendix).

Our example above considered a distance distribution between the center point of the distribution and the points it generated. A similar concept applies to distance distributions between any pair of points, as described below.

Let $Y$ be the distance distribution between the random variable vector $\mathbf{V}$ and $\mathbf{Q}$ such that $Y = \mathbf{d}(\mathbf{V}, \mathbf{Q})$. This is the distribution of distances between two points sampled at random, which is defined for the Euclidean squared distance by the following:

**Definition 1.** *Let* $\mathbf{V}$ *and* $\mathbf{Q}$ *be two d-dimensional distributions. Then, the squared Euclidean distance distribution* $Y^2$ *between the 2 random points* $\mathbf{V}$ *and* $\mathbf{Q}$ *is:*

$$Y^2 = ||\mathbf{V} - \mathbf{Q}||^2 = \sum_i (V_i - Q_i)^2 \tag{1}$$

which is the definition used in Angiulli 2018. Assuming $\mathbf{V}$ and $\mathbf{Q}$ follow the same distribution, then $Y^2$ has twice the variance of $\| \mathbf{V} \|^2$, among other properties. Using this definition, we show that normal distributions also have a distance distribution between points that follow the chi distribution, as shown in Section 8.3 in the Appendix.

### 2.1 Distance Distributions of a Dataset

Let $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n\} \mid \mathbf{X}_i = [X_{i,1}, \ldots, X_{i,d}]$ denote the dataset. The ordered distance distribution from each random sample to all the others is:

$$\begin{bmatrix} \mathbf{Y}_1 \\ \vdots \\ \mathbf{Y}_n \end{bmatrix} = \begin{bmatrix} Y_{1,(1)} & \ldots & Y_{1,(n_1)} & \ldots & Y_{1,(n-1)} \\ \vdots & & & & \vdots \\ Y_{n,(1)} & \ldots & Y_{n,(n_n)} & \ldots & Y_{n,(n-1)} \end{bmatrix},$$

where $Y_{1,(1)} \leq Y_{1,(2)} \leq \cdots \leq Y_{1,(n_1)} \leq Y_{1,(n-1)}$ and $\mathbf{Y}_1 = \text{sort}([\parallel \mathbf{X}_1 - \mathbf{X}_2 \parallel, \parallel \mathbf{X}_1 - \mathbf{X}_3 \parallel, \ldots, \parallel \mathbf{X}_1 - \mathbf{X}_n \parallel])$. There are $n$ distance distributions $\mathbf{Y}_i$, each containing a local distance distribution of varying unknown length $n_i$ generated by a local cluster, followed by distances to other clusters. $n_i$ is unknown and $Y_{i,(j)}$ stands for the $j$th order statistic of the $i$th distance distribution from $\mathbf{X}_i$ and is always positive.

In practice, some points from other clusters may be closer than certain points within the same cluster. We will introduce some strategies to mitigate this problem.

## 3 Local Cluster Cardinality Estimator

We define the cluster cardinality estimation task as follows: given a distribution of ordered distances $\mathbf{Y}$ from a point $\mathbf{x}$, estimate the cardinality $\hat{n}$ of the cluster to which $\mathbf{x}$ belongs. In this section, we present our new method, which was used in our experiments, followed by potential improvements.

### 3.1 Our Method: Minimum of the $\gamma$ Function

To simplify this task, we make the hypothesis that in the context of mean shift clustering, the distance distributions from each observation can be approximated by bimodal distributions. Our algorithm aims to find the minimum density between the two modes, whose position serves as our estimate of the local cluster cardinality. Figure 2 shows this process. The samples were generated from a bimodal distribution, and we apply the $\gamma$ function to estimate its true density.

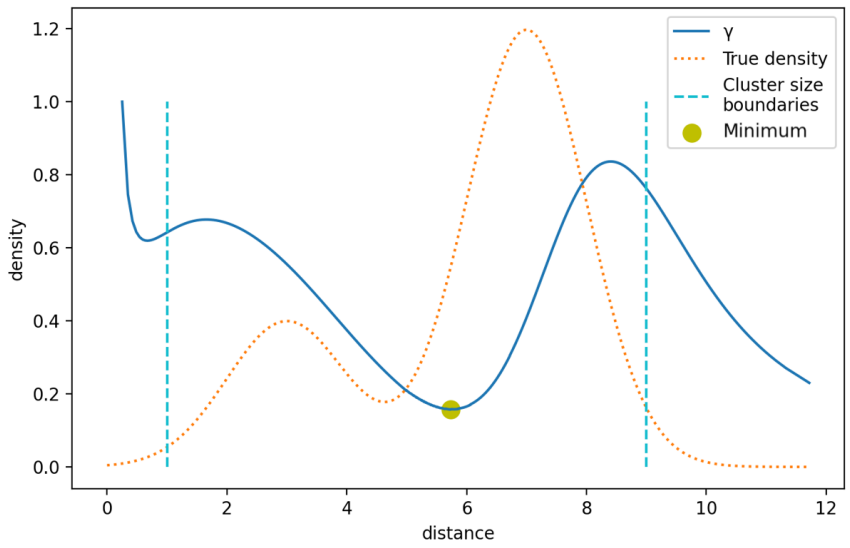


**Figure 2:** Cluster cardinality estimation

In Figure 2, the minimum of $\gamma$ is found at the $i$th ordered sample of the distance distribution, which is then taken as the local cluster cardinality estimate. The definition of the $\gamma$ function is:

$$\gamma(\mathbf{y}_{(k)}) = \frac{s^2}{(\bar{y} - y_{(k)})^2}, \tag{2}$$

where $\mathbf{y}_{(k)} = [y_{(1)}, \ldots, y_{(k)}]$, $\bar{y}$ is the sample mean and $s^2$ is the sample variance

$$\bar{y} = \sum_{1}^{k} y_{(i)}/k \quad \text{and} \quad s^2 = \sum_{1}^{k} (y_{(i)} - \bar{y})^2/k.$$

We calculate $\gamma$ for all distance vectors:

$$\begin{bmatrix} \gamma_{1,(1)} & \cdots & \gamma_{1,(n_1)} & \cdots & \gamma_{1,(n-1)} \\ \vdots & & & & \vdots \\ \gamma_{n,(1)} & \cdots & \gamma_{n,(n_n)} & \cdots & \gamma_{n,(n-1)} \end{bmatrix} := \begin{bmatrix} \gamma(\mathbf{y}_{1,(1)}) & \cdots & \gamma(\mathbf{y}_{1,(n-1)}) \\ \vdots & & \vdots \\ \gamma(\mathbf{y}_{n,(n)}) & \cdots & \gamma(\mathbf{y}_{n,(n-1)}) \end{bmatrix}$$

and estimate the cluster cardinality with

$$\hat{n}_i = \arg\min[\gamma_{i,(minBoundary)} \cdots \gamma_{i,(n_i)} \cdots \gamma_{i,(maxBoundary)}],$$

where $minBoundary$ and $maxBoundary$ are parameters of the cluster size estimation algorithm (see Algorithm 2 in the Appendix). The estimated cluster radius for the $i$th point will be equal to $y_{\hat{n}_i}$, the $n_i$th distance. Since the vector $\mathbf{y}_{(\hat{n}_i)} = [y_{(1)}, \ldots, y_{(\hat{n}_i)}]$ contains only distances generated from the local cluster, other useful statistics can be computed on it, including sample variance, mean, percentiles and mean square distance

$$MSD = \frac{1}{d\hat{n}_i} \sum y_{(i)},$$

where $d$ is the number of dimensions.

Kernel Density Estimation (KDE) methods on the data space are often used in adaptive mean shift, but cluster cardinality estimation has some advantages over it. Cluster cardinality estimation allows estimating statistics restricted to the cluster, while KDE methods measure the density of the data space locally, often using a small fraction of the cluster. For example, let both methods be used on a multivariate i.i.d. Gaussian distribution: our method can estimate the MSD, which for a Gaussian distribution is equal to the factor $\sigma^2$ of its diagonal covariance $\sigma^2 I$. KDEs estimate $f(\mathbf{x})$, the density of the Gaussian distribution at point $\mathbf{x}$.

#### 3.1.1 Details on the $\gamma$ Function and the Minimum Detection

The $\gamma$ function has some properties that make it helpful in estimating the minimum location between two modes compared with 1D KDEs. Experimentally, the $\gamma$ function's minimum value, before the second mode, is near the true location of the minimum density between the two modes. In that range, it behaves like a KDE and has some advantages over it. The $\gamma$ function has high values before the first mode and is scale-invariant. It has been used in the Maximum Likelihood Estimator (MLE) of a truncated normal in Cohen 1959, and to our knowledge, it is the first time it has been used to find local minimums of densities.

The $\gamma$ function answers two problems that might be hard to solve with KDEs.

1. The minimum might be found before the first mode.
2. The bandwidth parameter of the KDE is non-trivial to choose.

The function $\gamma$ addresses the first problem by having high values before the first mode. For instance, when there are only two values, $\gamma(\mathbf{y}_{(2)}) = 1$, which provides an inertia-like effect before reaching the first mode.

The second problem is inherently resolved by using $\gamma$, as it does not require a bandwidth parameter. The scale of the data does not affect the $\gamma$ function since it is scale-invariant as shown below.

*Proof.* Let $Y$ be a scale family such that $Y = \sigma B$. Then, $\gamma(\mathbf{Y}_{(k)}) = \gamma(\mathbf{B}_{(k)})$ since:

$$\gamma(\mathbf{Y}_{(k)}) = \frac{\sigma^2 s^2(\mathbf{B}_{(k)})}{\sigma^2(\bar{B} - B_{(k)})^2}.$$

□

The boundary parameters are used to restrict the minimum finding to the region between the two modes. Typically, the lowest density of a bimodal distance distribution occurs near the right extremity, when distances tend towards infinity. The maximum boundary prevents the algorithm from incorrectly identifying a minimum in the right tail. Its precise value is not critical, as long as it excludes densities lower than the minimum between the two modes.

The $\gamma$ function has not been extensively tested as a KDE. Further experiments and direct comparisons with KDEs are warranted.

## 3.2 Improvements

### 3.2.1 Mode Finding

Instead of setting boundaries via fixed parameters, they could be determined algorithmically. The cardinality estimation algorithm performs best when the two boundaries are located near the modes of the bimodal distance distribution. Those modes can be found separately using mode-finding algorithms, such as mean shift, applied directly to the distance distributions.

For the rightmost mode (distances to points outside the cluster), the mean shift can be initiated with a single seed placed in the right tail, and the bandwidth can be set using the method of Silverman 1998:

$$\begin{aligned} h &= 0.9An^{-1/5} \\ A &= \min(\text{standard deviation, interquartile range}/1.34) \end{aligned} \tag{3}$$

For the left mode (distances to the same cluster), the mean shift can be initiated with one seed in the left tail. The appropriate bandwidth is often harder to determine in this case. For the right boundary case, a bandwidth computed on the whole set of distances tends to be adequate because the right mode usually contains the most points. For the left mode, particularly when it is generated by distances from a very small cluster compared with the whole dataset, computing the bandwidth on the whole dataset can result in over-smoothing and possibly missing the mode. Adaptive mean shift methods could be considered to avoid this issue. The leftmost mode appears less critical than the rightmost mode to locate precisely, since the $\gamma$ function tends to yield high values before the first mode. Therefore, improvements should focus primarily on accurately locating the rightmost mode.

#### 3.2.2 Kernel Density Estimation

We should also consider replacing the $\gamma$ function with a kernel density function to locate the point of minimum density. The difficulty is the same as for finding the left mode. For very small clusters, it is easy to oversmooth the density and then possibly miss a minimum in density.

Finding an appropriate bandwidth for the small distances would also enable more sophisticated methods than locating the minimum between two boundaries. For instance, we could discard the bimodal assumption and instead identify all modes of the distance distribution. In this case, the estimated cardinality would correspond to the minimum density between the first two modes. This approach is more complex and more dependent on finding a suitable bandwidth, but it may perform better in some cases where the current algorithm fails.

#### 3.2.3 Task Definition and Performance Evaluation

To our knowledge, local cluster cardinality estimation is a novel task. Evaluating performance on this task is challenging, as it is intended to be applied to a broad range of multimodal distributions. Consequently, our performance analyses are often exploratory in nature. Clearly defining the task and establishing robust evaluation criteria are important priorities moving forward.

## 4 Adaptive Mean Shift

The mean shift procedure identifies modes by iteratively computing the weighted mean (Eq. 4) of points within a local window, shifting the window to this mean, and repeating the process until convergence. The weighted mean equation is shown below:

$$m(\mathbf{x}, \mathcal{X}) = \frac{\sum_{\mathbf{x}_i \in \mathcal{X}} K(||\mathbf{x}_i - \mathbf{x}||)\mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathcal{X}} K(||\mathbf{x}_i - \mathbf{x}||)}. \tag{4}$$

The adaptive part in our procedure is defined by the Gaussian Kernel $K$, which uses information from local distance distributions to adapt the window size and the bandwidth locally.

### 4.1 Gaussian Kernel

We tested two variations of the Gaussian kernel that utilize information from parameter estimation on distance distributions: a Gaussian kernel and a Gaussian kernel adapted for high dimensions. The Gaussian kernel is presented below:

$$K(x; h, \omega) = \begin{cases} \exp\left(-\frac{x^2}{2h^2}\right) & \text{if } x \leq \omega \\ 0 & \text{if } x > \omega \end{cases} \tag{5}$$

, where $x$ is the distance, $\omega$ is the distance threshold and $h$ is the bandwidth. The distance threshold $\omega$ is equal to $y_{(\hat{n}_i)}$, the $\hat{n}_i$th ordered distance from point $i$. The cluster cardinality limit ($\hat{n}_i$) aims to restrict contributions to the weighted average to points within the same cluster. This prevents a large number of distant points with small contributions from influencing the local mean more than a small number of points from the local cluster, which can occur in datasets containing small clusters.

Currently, we use the standard deviation of the local distance distribution as the bandwidth. Another possibility is to use a method from Silverman 1998, based on minimizing the mean integrated square error (p.85-87). The one-dimensional version of this method is widely used, for example, being part of the default bandwidth estimator for kernel density estimation in scikit-learn (Pedregosa et al. 2011). However, it does not seem that the high-dimensional version of this method is used as much, and it has performed worse in our tests than the standard deviation of the distance distribution.

Next, we show our Gaussian kernel adapted for high dimensions:

$$K(x; h, \omega, a, s, \bar{x}) = \begin{cases} \exp\left(-\frac{(\max[x-(\bar{x}-as),0])^2}{2h^2}\right) & \text{if } x \leq \omega \\ 0 & \text{if } x > \omega \end{cases} \tag{6}$$

where $\bar{x} = \frac{1}{\hat{n}_i}\sum_{i=1}^{\hat{n}_i} x_i$, $s^2 = \frac{1}{\hat{n}_i - 1}\sum_{i=1}^{\hat{n}_i}(x_i - \bar{x})^2$, $a \in \mathbb{R}^+$ and $\{x_1, \ldots, x_{\hat{n}_i}\}$ is the local distance sample. In our experimentation, $a$ is set to 4 to bring the smallest distances to a value slightly above or equal to zero.

Equation 6 is a variation on Eq. 5 that offsets distances closer to zero. The aim is to avoid two issues that arise in high-dimensional spaces, where dimensions are usually a lot larger (see Section 8.2 in Appendix). The first problem is that on the first iteration of the mean shift algorithm, the distance between a point and itself is zero, which assigns it an excessively large weight relative to other points. The algorithm could get stuck at this point or simply take a lot of time to move the initial weighted mean positions away from the original points. Another advantage of this kernel is that it helps prevent computing very small values when distances are very large. Similar to Eq. 5, the $\omega$ threshold ensures that kernel contributions come predominantly from points within the same cluster—likely the most significant enhancement we provide for clustering in high-dimensional settings.

## 4.2 Cluster Cardinality-Based Adaptive Mean Shift

Below is our adaptive mean shift algorithm.

**Algorithm 1** Cluster Cardinality-Based Adaptive Mean Shift

**Input** $\mathcal{X}, (minBoundary \leftarrow 5, maxBoundary \leftarrow n/2), maxIter \leftarrow 250$

**Output** $\mathcal{M}$, **label**

**Step1:** Cluster Cardinality Estimation

$\mathcal{P}_1, (\hat{\mathbf{n}}, \mathbf{h}) \leftarrow$ clusterCardinalityEstimation$(\mathcal{X}, (minBoundary, maxBoundary))$ ▷ See algorithm 2 in the Appendix

**Step2:** Adaptive Mean Shift

**for** $j : 1 \rightarrow maxIter$ **do**

  **for** $\mathbf{p}_{j,i} \in \mathcal{P}_j$ **do**

    $[0, y_{(1)}, \ldots, y_{(n_1)}] \leftarrow$ sort$([\parallel \mathbf{x}_i - \mathbf{x}_1 \parallel, \ldots, \parallel \mathbf{x}_i - \mathbf{x}_{n_1} \parallel])$

    $\hat{n}_k \leftarrow$ getLocalNParameter$(\mathbf{p}_{j,i})$

    $n_{k*} \leftarrow \hat{n}_k \min[(minBoundary + 0.01j * (\hat{n}_k - minBoundary))/\hat{n}_k, 1]$ ▷ area increase, see section 5.1.1

    $h = \text{Var}(y_{(1)}, \ldots, y_{(n_{k*})})$

    $\omega = y_{(n_{k*})}$

    $\mathbf{p}_{j+1,i} \leftarrow \frac{\sum_{\mathbf{x}_i \in \mathcal{X}} K(||\mathbf{x}_i - \mathbf{p}_{j,i}||; h, \omega)\mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathcal{X}} K(||\mathbf{x}_i - \mathbf{p}_{j,i}||; h, \omega)}$

  **end for**

  $\mathcal{P}_{j+1} \leftarrow [\mathbf{p}_{j+1,1}, \ldots]$

  break if $\mathcal{P}_j \approx \mathcal{P}_{j+1}$ and $j > 100$ ▷ Check if the distance between $\mathcal{P}_j$ and $\mathcal{P}_{j+1}$ is very small

  $\mathcal{P}_{j+1} \leftarrow \mathcal{B}$

**end for**

$\mathcal{M}$, **label** $\leftarrow$ clustering$(\mathcal{P}_j)$ ▷ See algorithm 3 in the Appendix

**Step3:** Classify Points With Bad Estimates

$\mathcal{F} \leftarrow \mathcal{X} - \mathcal{P}_1$ ▷ All points with bad estimate. $-$ is the set difference operation.

$\mathbf{label}_{\mathcal{F}} \leftarrow \emptyset$

$\mathbf{s}^2 \leftarrow$ getLocalKParameters$(\mathcal{M})$ ▷ Distance variance of each cluster

**for** $\mathbf{f}_i \in \mathcal{F}$ **do**

  **dist** $\leftarrow ||\mathbf{f}_i - \mathcal{M}||$ ▷ Distances between $\mathbf{f}_i$ and all modes

  $\mathbf{label}_{\mathcal{F}} \leftarrow \mathbf{label}_{\mathcal{F}} \cup \text{argmin} \exp(\mathbf{dist}^2/2\mathbf{s}^2)$

**end for**

**label** $\leftarrow$ **label** $\cup$ $\mathbf{label}_{\mathcal{F}}$

where $\mathcal{X} \in \mathbb{R}^{n \times d}$ is the original dataset, $\mathcal{P}_1 \in \mathbb{R}^{n_1 \times d} \mid n_1 \leq n$ is the dataset without points with bad cardinality estimate, $\hat{\mathbf{n}} = [\hat{n}_1, \ldots, \hat{n}_n] \mid n_i \in \mathbb{N}$ is the cardinality estimate for each point, $\mathbf{h} = [h_1, \ldots, h_n] \mid h_i \in \mathbb{R}$ is the bandwidth for each point, $\mathcal{P}_j \in \mathbb{R}^{n_1 \times d} \mid j > 1$ is the set of shifted points, $K$ is the Gaussian kernel, $\mathcal{M} \in \mathbb{R}^{c \times d}$ is the set of all modes and $\mathbf{label} = [label_1, \ldots, label_n] \mid label_i \in \{1, \ldots, c\}$ is the label of each point corresponding to a mode.

The algorithm takes as input a dataset $\mathcal{X}$ along with minimum and maximum boundary parameters for cluster cardinality estimation, with default values of 5 and $n/2$, respectively. If clusters bigger than $n/2$ are expected, the maximum boundary should be increased. The default minimum is discussed in Section 5.1.

The algorithm works in 3 steps:

1. Cluster Cardinality Estimation
2. Adaptive Mean Shift

3. Classify Points With Bad Estimates

In the **Cluster Cardinality Estimation** step (Section 3 and Algorithm 2 in the Appendix), a cluster cardinality $\hat{n}_i$ is estimated for each point $\mathbf{x}_i$. This is an estimate of the number of points present in the cluster to which $\mathbf{x}_i$ belongs. Points with a cluster cardinality estimate that are affected by the maximum boundary limit are labelled "bad" and are removed from the set of points used during the mean shift. Finally, statistics are computed on the local distance distribution between $\mathbf{x}_i$ and the closest $\hat{n}_i$ points, such as the sample mean, variance, and MSD. The exact statistic needed depends on the kernel used during mean shift, in the above example, the Gaussian kernel (Eq. 5) is used. The cluster cardinality estimation step outputs the set of points with a good cardinality estimate $\mathcal{P}_1$ and the needed parameters for the Gaussian kernel (in this case, $\hat{\mathbf{n}}$ and $\mathbf{h}$).

The **Adaptive Mean Shift** step moves iteratively all points towards their mode. It starts with each point $\mathbf{p}_{1,i} \in \mathcal{P}_1$, computes the weighted average $\mathbf{p}_{2,i}$ in the sphere including $n_{k*}$ points around it and repeats iteratively until it reaches $\mathbf{p}_{1,i}$'s mode or reach the maximum number of iterations. The function "**getLocalNParameter**" takes $\mathbf{p}_{j,i}$ and returns the local space parameter $n_k$. It finds the 5 nearest points in $\mathcal{P}_1$ to $\mathbf{p}_{j,i}$ and returns the median values of the parameter $\hat{n}$ from these five points, effectively regularizing the parameters used by the adaptive mean shift. $n_{k*}$ increases progressively over 100 steps, from $minBoundary$ to $\hat{n}_k$ (see Section 5.1.1). $h$ and $\omega$ are computed on the distance distribution $[y_{(1)}, \ldots, y_{(n_{k*})}]$, delineated by $n_{k*}$.

Weighted averages from Eq. 4 are computed at the same time to use Python's matrix operations, which is why we check if all points have stopped moving at the same time. The **clustering** function returns the modes $\mathcal{M}$ and the label of each point corresponding to the modes (see Algorithm 3 in the Appendix). Section 5.1.1 contains more details on the gradual mean shift area increase. In the code implementation, clustering is done on each iteration to reduce the size of $\mathcal{P}_{j+1}$.

Finally, the step **Classify Points With Bad Estimates** classifies all points that had a bad cluster size estimation by assigning them to the closest cluster, weighing distances by the clusters' bandwidth.

# 5 Experimentations

Experiments were conducted on a toy dataset containing four clusters designed to test the ability of the algorithm to adapt to scale and to the number of samples.

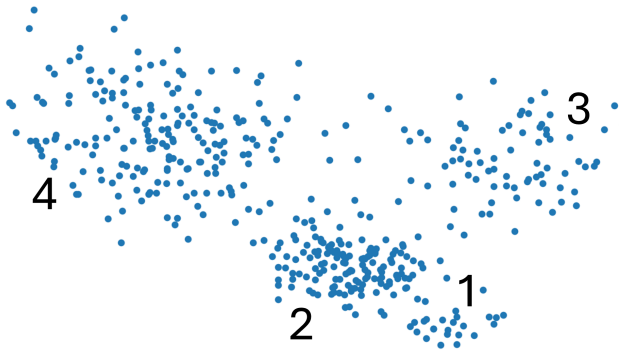


**Figure 3:** Toy dataset clusters

The toy dataset is generated by four isotropic Gaussian distributions with the following properties:

| cluster # | Cardinality | $\sigma$ |
|---|---|---|
| 1 | 25 | 0.7 |
| 2 | 100 | 1 |
| 3 | 75 | 1.5 |
| 4 | 200 | 2 |

**Table 1:** Toy dataset

where $\sigma^2 I$ is the covariance matrix for each distribution. Our algorithm used the following parameters:

| Parameter | Value |
|---|---|
| Minimum boundary | 5 |
| Maximum boundary | $n/2$ |
| Maximum number of iterations | 200 |

**Table 2:** Experimentation parameters

where $n$ is the total number of samples.

## 5.1 Cluster Cardinality Estimation

Cluster cardinality is estimated for each point by finding the minimum density of the distance distribution (see Algorithm 2). Figure 4 shows an example with the distance distribution (left graph) from the pink star (right graph) to all other points. The estimated cluster radius ($\hat{n}$th distance) is indicated by the minimum marker (left graph), and we can see that it accurately delimits points in the blue cluster (on the right).

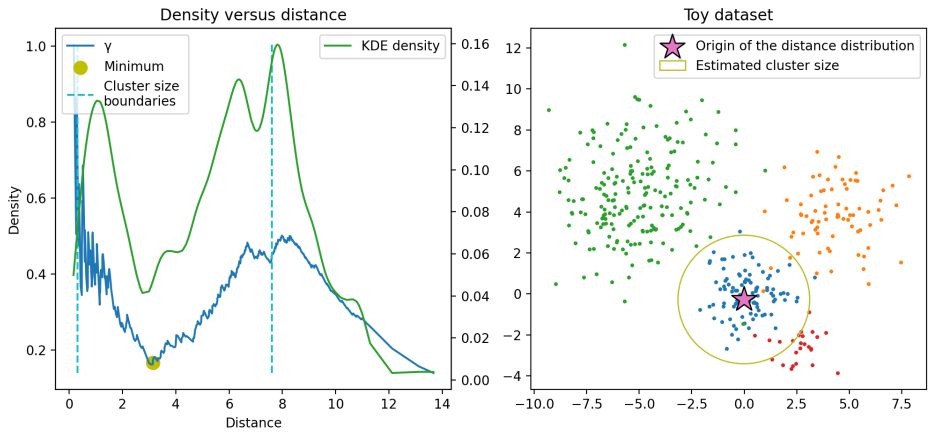


**Figure 4:** Cluster cardinality estimation

Points near the center of a cluster generally receive more accurate cluster cardinality estimates than those located between two clusters. In the latter case, the estimate can be determined by the upper boundary limit, as shown in Figure 5. We classify those estimates as ”bad” and remove those points from the mean shift process.

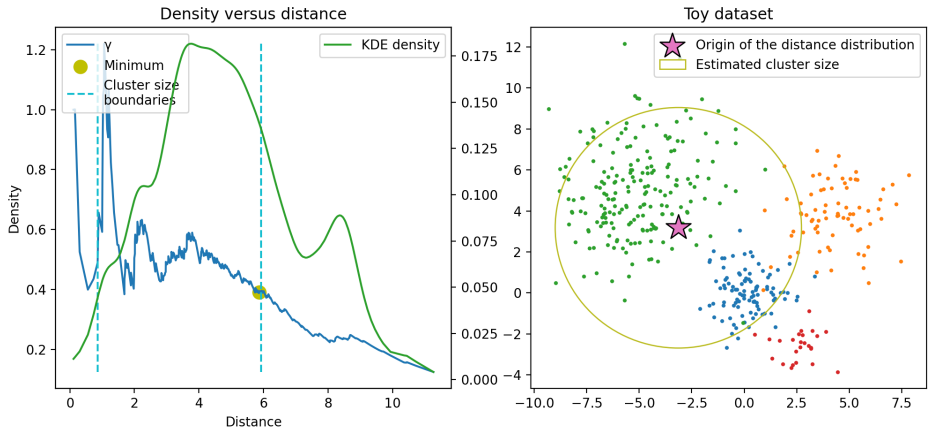


**Figure 5:** Bad cluster size estimation

Both $\gamma$ and KDE functions are shown on Figures 4 and 5 to show how they compare. The $\gamma$ function follows closely the behaviour of the KDE function near the minimum between the two modes in Figure 4, but without having low values before the first mode. This makes the exact choice of the lower boundary less critical; its primary role is to prevent the high variance typically associated with computing statistics on very small samples.

Figure 6 shows the estimated cluster cardinality for all points.

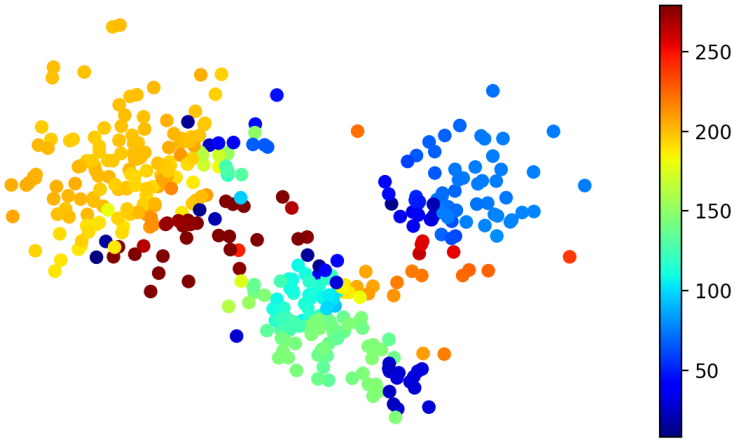


**Figure 6:** Cluster cardinality estimation on toy dataset

Cluster cardinality estimation tends to be more accurate near the center of a cluster and less reliable between clusters, due to the inclusion of nearest neighbors from adjacent clusters in the distance distribution. When these nearest neighbors belong to other clusters, it leads to an overestimation of cluster cardinality.

### 5.1.1 Gradual Mean Shift Area Increase

To counteract the positive bias in cardinality estimates for points far from cluster modes, we gradually increase the mean shift area during the mean shift procedure. The area considered by the kernel is gradually

increased from a small value until it includes a number of points equal to the local cluster cardinality estimate.

Points located farther from the cluster center tend to have positively biased cardinality estimates, causing their mean shift kernel to include points from neighboring clusters. This may shift the weighted mean away from its true cluster, particularly in cases where a large cluster is adjacent to a smaller one, such as clusters 2 and 1 in the toy dataset.

Increasing progressively the mean shift area might allow weighted means to gradually shift towards the right cluster mode, where the cluster cardinality estimate tends to be more accurate. Once the mean shift area reaches its full extent, the weighted means remain near the correct mode. Preliminary results with this approach were promising, although further evaluation is required.

## 5.2 Clustering

In Figure 7, we show clustering of the toy dataset, excluding points that have a bad cluster cardinality estimate (before Step 3 in Algorithm 1). All points marked with a triangle have a bad estimate and were not used during the mean shift. Notice how they are situated mainly between clusters.

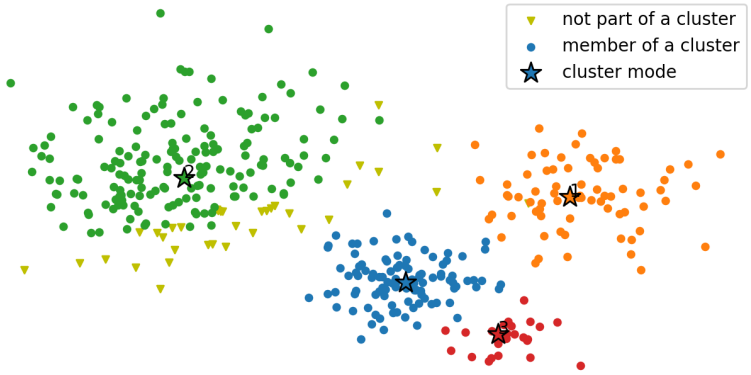


**Figure 7:** Clusters found on toy dataset after mean shift

The classification step assigns them to the nearest cluster, accounting for the scale of the cluster, and results in Figure 8.

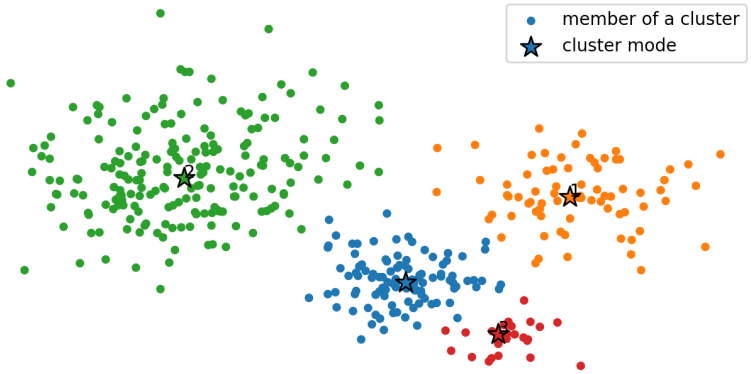


**Figure 8:** Clusters found on toy dataset after classification

### 5.3 Comparison to the Expectation–Maximization Algorithm on the Toy Dataset

The main advantage of our algorithm lies in its ability to handle varying scales and numbers of samples. Many clustering algorithms struggle with sets such as our toy dataset. For example, when applying the EM algorithm on our toy dataset, it has a very high tendency to miss the small cluster, as shown in Figure 9.

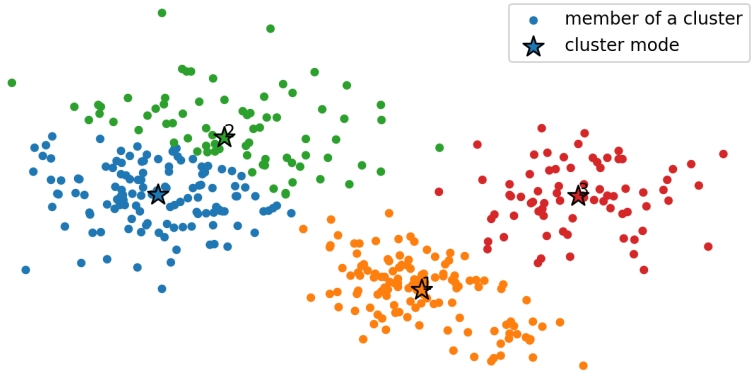


**Figure 9:** Clustering by the EM algorithm

## 6 Comparison to Other Clustering Algorithms

In this section, we will compare our results against a similar clustering algorithm and a benchmark library.

### 6.1 Algorithm: A Weighted Adaptive Mean Shift Clustering Algorithm

The article *Weighted Adaptive Mean Shift Clustering Algorithm* (WAMS) by Ren et al. 2014 presents an adaptive mean shift algorithm in which they "develop a technique to estimate the relevant subspace for each point, and then embed such information within the mean shift search process". We evaluated our algorithm on all datasets used in their experiments, except for the COIL dataset, which we were unable to locate in the form described in their paper.

| Dataset | #points | d | #classes |
|---|---|---|---|
| Iris | 150 | 4 | 3 |
| Yeast | 1136 | 8 | 3 |
| Steel | 1941 | 27 | 7 |
| USPS | 2007 | 256 | 10 |
| CTG | 2126 | 21 | 10 |
| Letter | 2263 | 16 | 3 |
| Image | 2310 | 19 | 7 |
| Pen | 3165 | 16 | 3 |
| Wave | 5000 | 21 | 3 |

**Table 3:** Datasets used in the experiments of Ren et al. 2014

The same preprocessing as in Ren et al. 2014 was done on each of the datasets:

USPS is a handwritten digit database and the 2007 test images were chosen for our experiments. The other eight datasets are from the UCI repository. Iris, Steel (Steel Plates Faults), CTG (Cardiotocography), Image (Image Segmentation), and Wave (Waveform version 1) are all in the original form. The three largest classes of Yeast were selected and letters 'I', 'J' and 'L' were chosen from the Letter database. Pen contains 10,992 samples from 10 classes (digits 0-9) and we selected 3 classes (digits $3, 8, 9$). For each dataset, features were normalized to have zero mean value and unit variance.

We used the Rand Index to evaluate performance, as was done for their experiments. The Rand Index is a measure of similarity between 2 data partitions, going from 0 to 1, which means that the two clusterings are the same. Table 4 contains our results and results presented by Ren et al. 2014.

| Dataset | Cluster Cardinality-Based Adaptive Mean Shift | | WAMS | $k$-means | EM | Sing-l |
|---|---|---|---|---|---|---|
| | Gaussian Kernel | High Dimension Kernel | | | | |
| Iris | **0.9575** | 0.9495 | 0.8060 | 0.8051 | 0.8193 | 0.7771 |
| Yeast | **0.6210** | 0.6041 | 0.6098 | 0.5624 | 0.3713 | 0.3602 |
| Steel | **0.7792** | **0.7794** | 0.7435 | 0.7394 | 0.7168 | 0.2446 |
| USPS | 0.7292 | 0.7459 | **0.9006** | 0.8720 | 0.8225 | 0.1274 |
| CTG | **0.8389** | **0.8387** | 0.8046 | 0.7996 | 0.7487 | 0.2007 |
| Letter | **0.6952** | 0.6925 | 0.6908 | 0.6029 | 0.6074 | 0.3336 |
| Image | 0.8417 | 0.8409 | **0.8820** | 0.8399 | 0.8028 | 0.1531 |
| Pen | **0.8256** | 0.8125 | 0.7198 | 0.7068 | 0.7143 | 0.6852 |
| Wave | **0.6761** | 0.6681 | 0.6695 | 0.6674 | 0.6675 | 0.6467 |

**Table 4:** Comparison against clustering algorithms on real data (Rand Index)

Our algorithm outperforms the compared methods on most datasets, except for the *USPS* and *Image* datasets. The reason for this is unclear, as we do not have access to their implementation for further analysis. The largest differences in results are on *Iris*, *USPS*, and *Pen* datasets.

It is worth noting that selection bias may have influenced the above results, as the datasets could have been chosen to favor the WAMS algorithm over other methods.

Our high-dimensional kernel often performs slightly worse than the Gaussian kernel, except on the USPS dataset, which has the highest number of dimensions by far. Further testing on additional high-dimensional datasets is required to confirm whether the high-dimensional kernel is indeed more effective than the Gaussian kernel in such contexts.

## 6.2 Benchmark: A Framework for Benchmarking Clustering Algorithms

We also evaluated our algorithm using datasets and benchmark results published by Gagolewski 2022. As before, the data is sourced from the UCI repository, although seven of the eight datasets differ from those in the previous comparison. They use a slightly different preprocessing to Ren et al. 2014:

> Remove all columns of zero variance (constant). Centre the data around the centroid (so that each column's mean is 0). Scale all columns proportionally (so that the total variance is 1; note that this is not the same as standardisation: standard deviations in each column might still be different). Add a tiny amount of noise to minimise the risk of having duplicate points.

| Dataset | #points | d | #classes | Gini index |
|---|---|---|---|---|
| Ecoli | 336 | 7 | 8 | 0.65 |
| Glass | 214 | 9 | 6 | 0.48 |
| Ionosphere | 351 | 34 | 2 | 0.28 |
| Sonar | 208 | 60 | 2 | 0.07 |
| Statlog | 2310 | 19 | 7 | 0.00 |
| WDBC | 569 | 30 | 2 | 0.25 |
| Wine | 178 | 13 | 3 | 0.13 |
| Yeast | 1484 | 8 | 10 | 0.63 |

**Table 5:** Datasets in the UCI set by Gagolewski 2022

Table 6 presents the performance of our algorithm using the Gaussian kernel, alongside results from other clustering algorithms included in the UCI benchmark dataset.

| Data | Ours - Gaussian kernel | | GIc | EM | kmeans | Spectral | Birch |
|---|---|---|---|---|---|---|---|
| | $MaxBoundary = 0.5n$ | $MaxBoundary = 0.7n$ | | | | | |
| Ecoli | 0.8520 | **0.8675** | 0.8076 | 0.8467 | 0.8151 | 0.5366 | 0.8308 |
| Glass | 0.6595 | 0.5375 | **0.7123** | 0.6761 | 0.6793 | 0.6681 | 0.6681 |
| Ionosphere | 0.5150 | 0.5277 | 0.6067 | **0.7010** | 0.5889 | 0.5401 | 0.5938 |
| Sonar | 0.5141 | **0.5186** | 0.4978 | 0.5006 | 0.5032 | 0.4976 | 0.4976 |
| Statlog | 0.8784 | **0.9066** | **0.9066** | 0.8450 | 0.8116 | 0.1512 | 0.6499 |
| WDBC | 0.6042 | 0.7289 | 0.6526 | **0.8541** | 0.7504 | 0.5423 | 0.5521 |
| Wine | 0.7067 | 0.7128 | 0.7128 | **0.9220** | 0.7187 | 0.6705 | 0.6515 |
| Yeast | **0.7594** | 0.7385 | 0.7385 | 0.6721 | 0.7421 | 0.2711 | 0.7121 |

**Table 6:** Results on the UCI benchmark dataset (Rand Index)

We tested our algorithm with the maximum boundary parameter set both at $0.5n$ and $0.7n$. GIc is a recent hierarchical clustering algorithm(Gagolewski 2021). All other algorithms are well known and available on scikit-learn (Pedregosa et al. 2011). All algorithms except ours need at least the number of clusters as a parameter. Spectral was used with default parameter $gamma = 1$, Birch had default parameters $treshold = 0.5$ and $branching\ factor = 50$.

The *yeast* dataset appears in both benchmark datasets, but our algorithm performed better in this case because of the different pre-processing.

The *ionosphere*, *sonar*, and *WDBC* datasets each contain only two classes, making the default maximum boundary of $0.5n$ too low for certain clusters. Increasing this boundary value to $0.7n$ led to improved clustering performance, particularly for the *WDBC* dataset. For most other datasets, clustering performance appeared relatively insensitive to the maximum boundary value. This behavior is by design: the boundary only influences results when adjusting it causes lower-density regions to be included or excluded. As a result, the algorithm is generally tolerant to changes in the boundary value, but such changes may lead to nonlinear and abrupt effects. In future versions of the algorithm, the maximum boundary should be determined algorithmically to eliminate the need for manual tuning.

In general, our algorithm seems competitive with other popular algorithms, particularly since it can be used without specifying the number of clusters.

# 7 Future Work and Conclusion

The current algorithm represents a first functional prototype and offers numerous opportunities for improvement. Cardinality estimation is a critical component of our approach, and additional research is required to refine several aspects, including its mathematical formulation, methodological design, and the behavior of the estimator on complex datasets. KDEs could be used after an initial bandwidth ($\sigma$) estimate has been obtained to improve the density estimation. Cluster boundaries could be determined with mode-seeking algorithms rather than via fixed parameters, for example, using one-dimensional mean shift. Furthermore, different parameters and strategies for the mean shift algorithm, such as the gradual increase in area, should also be tested systematically.

In this article, we introduced a new cluster cardinality estimator based on distance distributions and showed how that information can be used by the adaptive mean shift algorithm to achieve state-of-the-art results on real-world datasets. The proposed cardinality estimator remains a work in progress, with potential for significant improvement and broader applicability beyond clustering tasks. For instance, the cardinality estimator could be applied to a variety of machine learning tasks that require information about small, localized subsets within complex datasets. Currently, some version of a Kernel Density Estimation is often used in such cases. Our method has the advantage of estimating statistics based on distance distributions restricted mainly to a single cluster. In contrast, KDE methods measure local density in the data space, often providing limited insight into the overall cluster. Such KDE-based approaches are often more sensitive to noise and provide less comprehensive insights into the overall structure of individual clusters.

Despite the cardinality estimator being in its early stages, our cluster cardinality-based mean shift algorithm demonstrated competitive clustering performance across a variety of datasets. It outperformed a comparable adaptive mean shift algorithm and performed on par with a broad range of established clustering algorithms, all of which require prior knowledge of the number of clusters. Further improving the cardinality estimator is likely to enhance clustering performance even more.

# 8 Appendix

## 8.1 Algorithms

**Algorithm 2** Cluster Size Estimation

**Input** $\mathcal{X}, (minBoundary, maxBoundary)$
**Output** $\mathcal{P}, (\hat{\mathbf{n}}, \mathbf{h}, \boldsymbol{\omega})$

$j \leftarrow 0$
**for** $\mathbf{x}_i \in \mathcal{X}$ **do**
  $[y_{(1)}, \ldots, y_{(n-1)}] \leftarrow \text{sort}(||\mathbf{x}_i - \mathcal{X}||)$
  **for** $k : minBoundary \rightarrow 1.1 maxBoundary$ **do**
    $\bar{y}_{(k)} \leftarrow \sum_1^k y_{(i)}/k$
    $s^2_{(k)} \leftarrow \sum_1^k (y_{(i)} - \bar{y})^2/k$
    $\gamma_{(k)} \leftarrow \frac{s^2}{(\bar{y} - y_{(k)})^2}$
  **end for**
  $\hat{n}_{i1} \leftarrow \text{argmin}([\gamma_{minBoundary}, \ldots, \gamma_{maxBoundary}]) + minBoundary$
  $\hat{n}_{i2} \leftarrow \text{argmin}([\gamma_{minBoundary}, \ldots, \gamma_{maxBoundary}, \ldots, \gamma_{1.1 maxBoundary}]) + minBoundary$
  **if** $\hat{n}_{i1} = \hat{n}_{i2}$ **then** ▷ True if the cluster size estimate is not affected by the maximum boundary
    $j \leftarrow j + 1$
    $\hat{n}_j \leftarrow \hat{n}_{i1}$
    $\bar{y} \leftarrow \sum_{k=1}^{\hat{n}_j} y_{(k)}/\hat{n}_j$
    $h_j^2 \leftarrow \sum_{k=1}^{\hat{n}_j} (y_{(k)} - \bar{y})^2/\hat{n}_j$
    $\omega_j \leftarrow y_{(\hat{n}_j)}$
    $\mathbf{p}_j \leftarrow \mathbf{x}_i$
  **end if**
**end for**
$\mathcal{P} \leftarrow \{\mathbf{p}_1, \ldots\}$ ▷ Points with good estimate
// parameters of points with good estimate
$\hat{\mathbf{n}} \leftarrow [\hat{n}_1, \ldots]$
$\mathbf{h} \leftarrow [h_1, \ldots]$
$\boldsymbol{\omega} \leftarrow [\omega_1, \ldots]$

**Algorithm 3** Clustering

**Input** $\mathcal{P}$
**Output** $\mathcal{M}$, **label**
$\mathcal{M} \leftarrow \emptyset$ ▷ //Mode
**label** $\leftarrow \emptyset$ ▷ //Class of each points
**for** $\mathbf{p}_i \in \mathcal{P}$ **do**
  $exist \leftarrow False$
  **for** $j : 1 \rightarrow |\mathcal{M}|$ **do**
    **if** $\mathbf{p}_i \approx \mathbf{m}_j$ **then**
      $exist \leftarrow True$
      $label_i \leftarrow j$
      break
    **end if**
  **end for**
  **if** $exist == False$ **then**
    $\mathcal{M} \leftarrow \mathcal{M} \cup \mathbf{p}_i$
    **label** $\leftarrow$ **label** $\cup (|\mathcal{M}|)$
  **end if**
**end for**

## 8.2 Distances in high dimensions

The *curse of dimensionality* is often referred to as a reason that distances should not be used in high-dimensional space. Beyer et al. 1999 mentions that "as dimensionality increases, the distance to the nearest neighbor approaches the distance to the farthest neighbor". We will show in this section that, under some general conditions, this is a misinterpretation of their demonstration. As the number of dimensions increases, the distance between the nearest and farthest neighbors of an isotropic distribution tends toward a constant that depends on the number of samples in the distribution and not the number of dimensions. This theorem is based on the fact that i.i.d. distributions have a distance distribution that tends towards a normal distribution. The insights we bring in this section contribute to the design of our Gaussian kernel.

The following theorem approximates Euclidean distance distributions as a normal distribution as the number of dimensions tends toward infinity. Angiulli 2018 provided the distribution for the squared Euclidean distance; here we derive the Euclidean distance.

**Theorem 1.** *Let $\mathbf{Y}_d$ be any multivariate random distribution such that $\mathbf{Y}_d = [Y_1, Y_2, \ldots, Y_d], y_i \in \mathbb{R}$, $\mathrm{E}[\mathbf{Y}] = \mathbf{0}$ and all $Y_i$ are i.i.d. . Then the distance distribution $X_d = ||\mathbf{Y}_d||$ converges in distribution to a normal distribution such that*

$$\lim_{d\to\infty} X_d \xrightarrow{dist} N(\sqrt{d\mu_{Y^2}}, \frac{\sigma^2_{Y^2}}{4\mu_{Y^2}}) \tag{7}$$

*where $\mu_{Y^2}$ is the expectation of $Y_i^2$ and $\sigma^2_{Y^2}$ is the variance of $Y_i^2$.*

*Proof.* We start by proving it for $X_d^2$ and then for $X_d$. $X_d^2$ tends in distribution toward a normal distribution following the central limit theorem (CLT) such that when $d$ tends toward infinity:

$$\frac{\sqrt{d}\left(\frac{\sum Y_i^2}{d} - \mu_{Y^2}\right)}{\sigma_{Y^2}} \xrightarrow{dist} \mathcal{N}(0,1)$$

$$\sum_{i=1}^{d} Y_i^2 - \mu_{Y^2} d \xrightarrow{dist} \mathcal{N}(0, d\sigma_{Y^2}^2)$$

$$\sum_{i=1}^{d} Y_i^2 \xrightarrow{dist} \mathcal{N}(d\mu_{Y^2}, d\sigma_{Y^2}^2)$$

For example, when $Y_i \sim \mathcal{N}(0,1)$, $X_d^2$ is a chi-squared distribution with $Y_i^2$ having $\mu_{Y^2} = 1$ and $\sigma_{Y^2}^2 = 2$, which makes $X_d^2 \sim \mathcal{N}(d, 2d)$ when d is large.

The proof for the square of the distance makes use of the delta method, which states that if there is a sequence of random variables $X_d$ satisfying

$$\sqrt{d}[X_d - \theta] \xrightarrow{dist} \mathcal{N}(0, \beta^2)$$

, then

$$\sqrt{d}[g(X_d) - g(\theta)] \xrightarrow{dist} \mathcal{N}(0, \beta^2[g'(\theta)]^2)$$

for any function $g$ satisfying the property that $g'(\theta)$ exists and is non-zero valued. Continuing from the CLT demonstration:

$$\sqrt{d}\left(\frac{\sum Y_i^2}{\sqrt{d}} - \mu_{Y^2}\sqrt{d}\right) \xrightarrow{dist} \mathcal{N}(0, \sigma_{Y^2}^2 d)$$

let $g(x) = \sqrt{x}$, $\theta = \mu_{Y^2}\sqrt{d}$ and $\beta^2 = \sigma_{Y^2}^2 d$ with $[g'(\theta)]^2 = \frac{1}{4\mu_{Y^2}\sqrt{d}}$, then:

$$\sqrt{d}\left(\frac{\sqrt{\sum Y_i^2}}{d^{1/4}} - \sqrt{\mu}_{Y^2} d^{1/4}\right) \xrightarrow{dist} \mathcal{N}(0, \frac{\sigma_{Y^2}^2\sqrt{d}}{4\mu_{Y^2}})$$

$$\sqrt{\sum_{i=1}^{d} Y_i^2} - \sqrt{\mu_{Y^2} d} \xrightarrow{dist} \mathcal{N}(0, \frac{\sigma_{Y^2}^2\sqrt{d}}{4\mu_{Y^2}})\frac{1}{d^{1/4}}$$

$$\sqrt{\sum_{i=1}^{d} Y_i^2} \xrightarrow{dist} \mathcal{N}(\sqrt{d\mu_{Y^2}}, \frac{\sigma_{Y^2}^2}{4\mu_{Y^2}})$$

Which proves the theorem. □

Theorem 2 shows that distances between neighbors of an i.i.d. distribution tend toward a constant in high dimensions:

**Theorem 2.** *Let $\mathbf{Y}_d = [Y_1, \ldots, Y_d]$ be an i.i.d. random vector. Then, if $X = ||\mathbf{Y}_d||$ and $d \to \infty$, then*

$$\mathrm{E}[X_{(n:n)} - X_{(1:n)}] = c \tag{8}$$

*, where $c$ is a constant independent of $d$ for a given $Y_i$ and $n$.*

*Proof.* From theorem 1, $\lim_{d\to\infty} X \xrightarrow{dist} N(\sqrt{\mu d}, \frac{\sigma^2}{4\mu})$. Let $A \sim N(\sqrt{\mu d}, \frac{\sigma^2}{4\mu})$ and $B \sim N(0, \frac{\sigma^2}{4\mu})$. Then $A = B + \sqrt{\mu d}$ and $A_{(i:n)} = B_{(i:n)} + \sqrt{\mu d}$. Now we calculate the expectation of the difference between the farthest and the closest neighbor

$$\begin{aligned}\mathrm{E}[A_{(n:n)} - A_{(1:n)}] &= E[(B_{(n:n)} + \sqrt{\mu d}) - (B_{(1:n)} + \sqrt{\mu d})] \\ &= E[B_{(n:n)} - B_{(1:n)}]\end{aligned}$$

The quantity $E[B_{(n:n)} - B_{(1:n)}]$ is the expectation of the range (David and Nagaraja 2003, p. 1) and can be calculated numerically easily for any distribution and $n$ value. Since it is independent of $d$, the expectation of the range of the approximation is only dependent on $\sigma^2$, $\mu$, and $n$ and not the number of dimensions. □

Because the distance distribution converges in distribution to a normal distribution with a variance that is independent of the dimension, the difference between the farthest and closest neighbor also converges, even if the mean of the distribution increases.

This brings context to Beyer et al. 1999 where they looked at the ratio of distances between the closest and farthest neighbor. If $\sqrt{\mu d} + B_{(1)}$ represents the closest neighbor, then the ratio of distances when $d$ tends toward infinity can be written in terms of order statistics:

$$\lim_{d\to\infty} \frac{\mathrm{E}[\sqrt{\mu d} + B_{(1)}]}{\mathrm{E}[\sqrt{\mu d} + B_{(n)}]} = 1 \tag{9}$$

since

$$\lim_{d\to\infty} \sqrt{\mu d} = \infty \quad \text{and} \quad \mathrm{E}[B_{(1)}], \mathrm{E}[B_{(n)}] \in \mathbb{R}$$

This confirms the findings of Beyer et al. 1999 for i.i.d. distributions, but also adds context. They mentioned that "as dimensionality increases, the distance to the nearest neighbor approaches the distance to the farthest neighbor". The ratio of distances tends toward 1, but the expected distance between closest and farthest neighbor can stay the same, as is the case for any multivariate distribution with i.i.d. distributions along all axes. This also confirms findings in Aggarwal, Hinneburg, and Keim 2001 where they found that the distance between farthest and nearest neighbor tends toward a constant for the Euclidean norm.

## 8.3 Euclidean Distances

The distribution of Euclidean distances from the mean in normal isotropic multivariate distributions follows a chi distribution with $d$ degrees of freedom and scale parameter $\sigma$. This version of the chi distribution is uncommon, but is referenced in Johnson, Kotz, and Balakrishnan 1994 (p. 452); it is also the scale family of the chi distribution (Casella and Berger 2001 (p. 116)). The chi distribution is related to the normal distribution by the following Theorem:

**Theorem 3.** *Let $\mathbf{Y} \in \mathbb{R}^d$ be any random variable following an isotropic multivariate normal distribution such that its probability density function (pdf) is:*

$$f_{\mathbf{Y}}(\mathbf{y}) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}} e^{\left(-\frac{||\mathbf{y}||^2}{2\sigma^2}\right)}. \tag{10}$$

*The distribution of Euclidean distances $X = ||\mathbf{Y}||$ is the chi distribution with $d$ degrees of freedom and scale parameter $\sigma$ with pdf*

$$f_X(x) = \frac{1}{\Gamma(\frac{d}{2})2^{(d/2)-1}\sigma^d} e^{-x^2/2\sigma^2} x^{d-1}$$

*and cumulative density function (cdf)*

$$P(X < x) = \frac{\gamma(\frac{d}{2}, \frac{x^2}{2\sigma^2})}{\Gamma(\frac{d}{2})}.$$

*Proof.* We will start with the multivariate normal distribution (eq.11) and modify it to obtain the multivariate isotropic Gaussian distribution (eq.10). The multivariate normal distribution is

$$f(\mathbf{x} : \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^d}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right), \tag{11}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the mean, $\Sigma$ is a square covariance matrix, $d$ is the number of dimensions and $|\Sigma|$ is the determinant of the covariance matrix.

Using an average of $\boldsymbol{\mu} = \mathbf{0}$ removes the term without loss of generality since we want to calculate our cdf from the center of the distribution. To get an isotropic normal density, we reduce the covariance matrix to a diagonal matrix with value $\sigma^2$ on the diagonal. This leads to

$$\mathbf{x}^T \Sigma^{-1} \mathbf{x} = \frac{||\mathbf{x}||^2}{\sigma^2}$$

and

$$|\Sigma| = (\sigma^2)^d.$$

We then obtain:

$$f(\mathbf{x} : \sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^d} e^{-||\mathbf{x}||^2/2\sigma^2}, \tag{12}$$

which is eq.10 from theorem 3.

In eq.12, $\mathbf{x}$ is a vector, but since we only use it's norm, it can be used as a variable instead of $\mathbf{x}$ without losing generality. This results in an univariate function of the radial distance $r$ instead of a multivariate one:

$$f(r : \sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^n} e^{-r^2/2\sigma^2}, \tag{13}$$

where $||\mathbf{x}||^2 = r^2$.

We now want to find the cdf of the distribution $X$ of distances from the center of an isotropic multivariate Gaussian distribution. It follows this form $P(X < r; d, \sigma)$: the probability of choosing a point at random from a normal distribution with a distance from the center of $r$ or less. We count the number of points at every distance $r$ and calculate their combined probability density function. To calculate this, we construct an hypersphere with a Gaussian density and calculate its mass. The d-dimensional multivariate Gaussian density defines iso-contours of equiprobable points. In the case of an isotropic multivariate Gaussian density, these lie on the surface of a hypersphere in $\mathbb{R}^d$.

The mass of the d-dimensional hypersphere can be calculated with the following formula:

$$M(r) = \int_0^r density(r)\, dV$$

For the hypersphere density we use eq.13. Which results in the cdf:

$$F(r) = P(X < r) = \int_0^r \frac{1}{(\sqrt{2\pi}\sigma)^d} e^{-\frac{x^2}{2\sigma^2}} dV, \tag{14}$$

where $V$ is the n-dimensional volume (content) of the hypersphere. Let $S_d$ be the hyper-surface area of a d-dimensional hypersphere of unit radius. From the wolfram[2]:

$$S_d = \frac{2\pi^{d/2}}{\Gamma(\frac{d}{2})}$$

and

$$V = \int_0^r S_d x^{d-1} dx = \frac{S_d r^d}{d}.$$

Which we use to calculate $dV$:

$$\begin{aligned} V &= \frac{2\pi^{d/2} r^d}{d\Gamma(\frac{d}{2})} \\ dV &= \frac{V}{dr} = \frac{2\pi^{d/2} r^{d-1}}{\Gamma(\frac{d}{2})} \end{aligned} \tag{15}$$

We can now combine eq.14 and 15 to calculate the cdf:

$$P(X < r) = \int_0^r \frac{2\pi^{d/2}}{\Gamma(\frac{d}{2})} \frac{1}{(\sqrt{2\pi}\sigma)^d} e^{-x^2/2\sigma^2} x^{d-1}\, dx, \tag{16}$$

we move the terms we can out of the integral,

$$= \frac{2}{2^{d/2}\Gamma(\frac{d}{2})\sigma^d} \int_0^r e^{-x^2/2\sigma^2} x^{d-1}\, dx,$$

then modify the terms of the integral in order to obtain the lower gamma function $\gamma$ starting with a variable substitution:

$$\frac{x^2}{2\sigma^2} = y \rightarrow \begin{cases} \frac{x}{\sigma^2} dx = dy \\ x = \sqrt{2y\sigma^2}. \end{cases}$$

[2] https://mathworld.wolfram.com/Hypersphere.html

The upper limit of the integral goes from $r$ to $\frac{r^2}{2\sigma^2}$:

$$= \frac{2\sigma^2}{2^{d/2}\Gamma(\frac{d}{2})\sigma^d} \int_0^r e^{-x^2/2\sigma^2} x^{d-2} \frac{x}{\sigma^2}\, dx = \frac{2\sigma^2}{2^{d/2}\Gamma(\frac{d}{2})\sigma^d} \int_0^{\frac{r^2}{2\sigma^2}} e^{-y} (\sqrt{y2\sigma^2})^{d-2}\, dy$$
$$= \frac{2^{d/2}\sigma^d}{2^{d/2}\Gamma(\frac{d}{2})\sigma^d} \int_0^{\frac{r^2}{2\sigma^2}} e^{-y} y^{d/2-1}\, dy$$

By definition of the $\gamma$ function we obtain:

$$= \frac{1}{\Gamma(\frac{d}{2})} \gamma(\frac{d}{2}, \frac{r^2}{2\sigma^2})$$
$$P(X < r) = \frac{\gamma(\frac{d}{2}, \frac{r^2}{2\sigma^2})}{\Gamma(\frac{d}{2})} \tag{17}$$

You can change variable $r$ to $x\sigma$, where $x$ is the Mahalanobis distance:

$$P(X < x) = \frac{\gamma(\frac{d}{2}, \frac{x^2}{2})}{\Gamma(d/2)} \tag{18}$$

Eq.17 is the cdf of the Chi distribution. It is related to the chi-square distribution, which is the distribution of the sum of $d$ squared standard normal distribution. We will work with eq.17 instead to keep generality. For the rest of this document we call the distribution having this cdf the Chi scale family distribution and denote it this way: $X \sim \chi(d, \sigma)$.

We can obtain the pdf of the chi distribution from eq.16 by deriving it:

$$f(x) = \frac{1}{2^{d/2-1}\Gamma(\frac{d}{2})\sigma^d} e^{-x^2/2\sigma^2} x^{d-1} \tag{19}$$

We now concluded that the distribution of distances to the center of an isotropic Gaussian distribution follows a chi scale family distribution. □

**Theorem 4.** *Let $\mathbf{Y}, \mathbf{Q} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 I_d)$ and $X = ||\mathbf{Y} - \mathbf{Q}||$. Then, $X$ is a distribution of Euclidean distances between 2 points randomly sampled from $\mathcal{N}(\boldsymbol{\mu}, \sigma^2 I_d)$ and follows a chi distribution with d degrees of freedom and scale parameter $2\sigma^2$ noted $\chi(d, 2\sigma^2)$.*

*Proof.* By the sum of normal random variable property, if $\mathbf{A} = \mathbf{Y} - \mathbf{Q}$, then $\mathbf{A} \sim \mathcal{N}(0, 2\sigma^2 I_d)$ and by theorem 3 $||A|| \sim \chi(d, 2\sigma^2)$. □

Theorem 4 confirms results presented in Thirey and Hickman 2015 where they found that the distance distribution between points of a normal distribution with parameter $\sigma = 1$ follows a distribution with pdf:

$$f(x, d) = \frac{2^{1-d}}{\Gamma\left(\frac{d}{2}\right)} \exp\left(\frac{-x^2}{4}\right) x^{d-1} \tag{20}$$

This is the same result as using Theorem 3 and theorem 4 to calculate the point-to-points distance pdf in a Gaussian distribution with $\sigma = 1$:

$$f(x, d) = \frac{1}{\Gamma\left(\frac{d}{2}\right) 2^{(d/2)-1} \sqrt{2}^d} \exp\left(\frac{-x^2}{2\sqrt{2}^2}\right) x^{d-1} \tag{21}$$

which is the pdf of the chi($d$, $\sqrt{2}$) distribution. Apart from confirming previous results, theorem 4 also links the interpoint distance distribution with the chi distribution and defines the relation between the sigma parameter of the normal distribution and its effect on the interpoint distance distribution.

# References


Aggarwal, Charu C., Alexander Hinneburg, and Daniel A. Keim (2001). "On the Surprising Behavior of Distance Metrics in High Dimensional Space". en. In: *Database Theory — ICDT 2001*. Ed. by Gerhard Goos et al. Vol. 1973. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 420–434. ISBN: 978-3-540-41456-8 978-3-540-44503-6.

Angiulli, Fabrizio (Apr. 2018). "On the behavior of intrinsically high-dimensional spaces: Distances, direct and reverse nearest neighbors, and hubness". In: *Journal of Machine Learning Research* 18, pp. 1–60.

Beyer, Kevin et al. (1999). "When Is "Nearest Neighbor" Meaningful?" en. In: *Database Theory — ICDT'99*. Ed. by Catriel Beeri and Peter Buneman. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 217–235. ISBN: 978-3-540-49257-3.

Casella, George and Roger Berger (2001). *Statistical Inference*. Anglais. 2 édition. Australia ; Pacific Grove, CA: Duxbury Press. ISBN: 978-0-534-24312-8.

Cohen, A. Clifford (1959). "Simplified Estimators for the Normal Distribution When Samples Are Singly Censored or Truncated". In: *Technometrics* 1.3. Publisher: [Taylor & Francis, Ltd., American Statistical Association, American Society for Quality], pp. 217–237. ISSN: 0040-1706.

Comaniciu, D. and P. Meer (May 2002). "Mean shift: a robust approach toward feature space analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.5. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 603–619. ISSN: 1939-3539.

David, H. A. and H. N. Nagaraja (2003). *Order statistics*. 3rd ed. Hoboken, N.J: John Wiley. ISBN: 978-0-471-38926-2.

Gagolewski, Marek (July 2021). "genieclust: Fast and robust hierarchical clustering". In: *SoftwareX* 15, p. 100722. ISSN: 2352-7110.

— (Dec. 2022). "A framework for benchmarking clustering algorithms". In: *SoftwareX* 20, p. 101270. ISSN: 2352-7110.

Johnson, Norman Lloyd, Samuel Kotz, and N. Balakrishnan (1994). *Continuous univariate distributions*. 2nd ed. Wiley series in probability and mathematical statistics. New York: Wiley. ISBN: 978-0-471-58495-7 978-0-471-58494-0.

Pedregosa, Fabian et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.85, pp. 2825–2830. ISSN: 1533-7928.

Ren, Yazhou et al. (Apr. 2014). "A Weighted Adaptive Mean Shift Clustering Algorithm". en. In: *Proceedings of the 2014 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, pp. 794–802. ISBN: 978-1-61197-344-0.

Silverman, B. W. (1998). *Density estimation for statistics and data analysis*. Monographs on statistics and applied probability 26. Boca Raton: Chapman & Hall/CRC. ISBN: 978-0-412-24620-3.

Thirey, Benjamin and Randal Hickman (Aug. 2015). "Distribution of Euclidean Distances Between Randomly Distributed Gaussian Points in n-Space". In: *arXiv:1508.02238 [math]*. arXiv: 1508.02238.