UEQMS: UMAP Embedded Quick Mean Shift Algorithm for High Dimensional Clustering

Abhishek Kumar[1, 3], **Swagatam Das**[2], **Rammohan Mallipeddi**[3]
[1]IAI, TCG Creast, Kolkata, India-700091
[2]Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata, Indian-700108
[3]Department of Artificial Intelligence, Kyungpook National University, Daegu, Republic of Korea - 41566
abhishek.kumar.eee13@iitbhu.ac.in, swagatam.das@isical.ac.in, mallipeddi@knu.ac.kr

Abstract

The mean shift algorithm is a simple yet very effective clustering method widely used for image and video segmentation as well as other exploratory data analysis applications. Recently, a new algorithm called MeanShift++ (MS++) for low-dimensional clustering was proposed with a speedup of 4000 times over the vanilla mean shift. In this work, starting with a first-of-its-kind theoretical analysis of MS++, we extend its reach to high-dimensional data clustering by integrating the Uniform Manifold Approximation and Projection (UMAP) based dimensionality reduction in the same framework. Analytically, we show that MS++ can indeed converge to a non-critical point. Subsequently, we suggest modifications to MS++ to improve its convergence characteristics. In addition, we propose a way to further speed up MS++ by avoiding the execution of the MS++ iterations for every data point. By incorporating UMAP with modified MS++, we design a faster algorithm, named UMAP embedded quick mean shift (UEQMS), for partitioning data with a relatively large number of recorded features. Through extensive experiments, we showcase the efficacy of UEQMS over other state-of-the-art algorithms in terms of accuracy and runtime.

Introduction

Automatic detection of the meaningful groups or clusters in a dataset is a fundamental task in unsupervised learning. Mean Shift (MS) (Cheng 1995) emerged as a non-parametric mode-seeking approach to detect the dense regions in a point cloud. Using a sequence of such momde-seeking iterates through kernel density estimation, MS automatically estimates the number of clusters in a dataset. Due to its simplicity and effectiveness, MS and its variants were widely used in applications such as object tracking (Comaniciu, Ramesh, and Meer 2003; Leichter, Lindenbaum, and Rivlin 2010; Ning et al. 2012; Kumar et al. 2022), unsupervised image and video segmentation (Park, Lee, and Park 2009; Zhou, Wang, and Schaefer 2011; Tao, Jin, and Zhang 2007; Paris 2008; Paris and Durand 2007), and general image processing (Barash and Comaniciu 2004; Bigdeli et al. 2017).

For low-dimensional clustering and segmentation tasks, the popularity of MS exceeded those of other well-studied algorithms, such as k-means (Zhao, Deng, and Ngo 2018; Ismkhan 2018), Felzenszwalb method (Felzenszwalb and Huttenlocher 2004), simple linear iterative clustering (SLIC) (Achanta et al. 2012), and Spectral clustering (Huang et al. 2020). On the other hand, each iteration in MS takes $O(n^2)$ time, making it computationally expensive. In other words, the computational complexity associated with finding each point's neighboring data points is quadratic in the number of data points. Although MS has superior performance in image segmentation, significant computational cost limits its use in high-resolution image segmentation. To address this issue, a faster variant of MS, called MeanShift++ (MS++), was recently proposed in (Jang and Jiang 2021), in which the input space is partitioned into grid cells, and each data point updates its position towards the approximated mode by referencing the data points within its neighboring grid cells. MS++ is ideal for handling extremely large datasets with low dimensionalities, such as in image segmentation. In (Jang and Jiang 2021), MS++ was shown to attain a speedup of 1000 times compared to the vanilla MS. Since MS++'s runtime is dependent on $\mathcal{O}(3^d)$ for a d-dimensional dataset, it becomes extremely slower for $d \geq 5$ datasets. Therefore, its application is only limited to low-dimensional clustering applications. To address this issue, one may think of integrating a dimensionality reduction technique as a pre-processing step to MS++.

Nonnegative matrix factorization (Lee and Seung 1999), Principle Component Analysis (Pearson 1901), and Canonical Correlation Analysis (Andrew et al. 2013) are commonly used unsupervised dimensionality reduction (DR) methods. Such models suffer from limitations over the representational abilities and cannot identify the non-linear hidden relations among data points in the data very efficiently (Guo, Lin, and Ye 2021). On the other hand, popular non-linear dimensionality reduction techniques can be distinguished by their emphasis on preserving global structure and local structure of the data. Isomap (Tenenbaum, Silva, and Langford 2000) is a well-known DR technique that emphasizes more on preserving global structure of the data, while t-SNE (Van der Maaten and Hinton 2008) emphasizes more on preserving global structure of the data. Although UMAP (McInnes, Healy, and Melville 2018) emphasizes more on preserving local structure of the data, it can also preserves global structure more effectively. In recent years, autoencoders (AE) (Tschannen, Bachem, and Lucic 2018)

have been frequently employed as a DR method for clustering due to their powerful representational power. Unsupervised autoencoders are competent in acquiring meaningful encodings of input data. As a way of better understanding, we study the effectiveness of popular DR methods in conjunction with the proposed clustering approach in the Illustrative Example.

In this paper, we first theoretically establish the convergence argument for MS++. At first, we demonstrate that the algorithm may converge to undesired and non-critical points. Still, by making simple modifications, convergence to a local optimum of the data density function (corresponding to the cluster centroids) can be guaranteed within a finite number of iterations. Next, we introduce a deflation variant of MS++, called Quick MeanShift (QMS), that is guaranteed to have lower time complexity compared to the original one without sacrificing clustering accuracy. Considering the simplicity and fast runtime of QMS, we extend its application to high-dimensional datasets by incorporating a dimensionality reduction technique UMAP, with QMS. The resulted algorithm is named UEQMS. There are the following advantages of using UMAP:

i) It extracts the low-dimensional features, less than 3, from high-dimensional datasets more effectively and efficiently than others (as shown in Fig. 1), while preserving the local and global structure of the data.

ii) The extracted features are less complex and noisy than the original ones for clustering. Consequently, this improves the clustering accuracy as performance of MS is improved with decreasing number of insignificant features (as shown in Table 1).

iii) Due to utilization of UMAP, UEQMS is 10^5 times faster than MS and other density-based clustering algorithms for high-dimensional datasets with better clustering accuracy.

Illustrative Example To show UMAP's effectiveness with QMS, before discussing our proposals, let's examine some preliminary results on MNIST. Specifically, we assess QMS's performance with several popular dimension reduction methods: tSNE (Van der Maaten and Hinton 2008), PACMAP (Wang et al. 2021), AE (Tschannen, Bachem, and Lucic 2018) and UMAP (McInnes, Healy, and Melville 2018). The results of this experiment in terms of accuracy (ACC), Normalized Mutual Index (NMI), Adjusted Rand Index (ARI), and runtime (RT) are depicted in Fig. 1

During clustering, we first apply dimensionalty reduction techniques to extract two features for each data point of the MNIST. Thereafter, we apply QMS on these extracted features for clustering. As shown in Fig. 1, we can see that the UMAP with QMS provides better clustering results than other dimensionalty reduction techniques with a comparable runtime. As a result, the UMAP with QMS algorithm, named UEQMS, is faster and more accurate than all other contenders. Furthermore, we compare the performance of UEQMS with other density-based clustering algorithms and MS variants in conjunction with UMAP. We present the results of this experiment in Table 1. As shown in this table,

Alg	NMI	ARI	RT	Alg	NMI	ARI	RT
MS	0.45	0.38	3E+8	QS++	0.49	0.22	5E+8
MeS	0.51	0.38	4E+8	UEMS	0.88	0.87	491
DB	0.25	0.10	5E+9	UEMS++	0.83	0.85	74
DB++	0.24	0.01	4E+8	UEαMS++	0.83	0.85	68
QS	0.49	0.22	5E+8	UEQMS	**0.89**	**0.88**	**44**

Table 1: *Comparison of Algorithms on MNIST data.* MS: Mean shift (Cheng 1995), MeS: Medoid shift (Sheikh, Khan, and Kanade 2007), DB: DBSCAN (Khan et al. 2014), DB++: DBSCAN++ (Jang and Jiang 2019), QS: Quick shift (Vedaldi and Soatto 2008), QS++: Quick shift++ (Jiang, Jang, and Kpotufe 2018), UEMS: UMAP with mean shift (Cheng 1995), UEMS++: UMAP with meanshift++ (Jang and Jiang 2021), UEαMS++: UMAP with α-meanshift++ (Park 2021a), NMI: normalized mutual information (Estévez et al. 2009), ARI: Adjusted rand index (Santos and Embrechts 2009), RT: runtime in seconds.

UEQMS performs better than other contenders in terms of clustering accuracy and runtime. It is worth noting that UEQMS is 1E+07 times faster than conventional algorithms. UEMS++ and UEαMS++ provide lower clustering accuracy than UEQMS due to noisy clusters (clusters with less than 0.05% data points). UEQMS tackles this issue using proposed modifications. The main contributions of this paper are as follows:

- In this work, we first define the MS++ kernel function. Then, we show that MS++ may converge to a non-stationary point as it is using a non-smooth kernel function.
- To address the convergence issue, a simple modification to the steps of MS++ is proposed ensuring that it will always converge to a local minimum of the kernel function.
- Finally, we propose quicker version of the MS++, named as Quick Mean Shift (QMS), by incorporating the proposed modification in MS++ and a noise reduction technique to improve the clustering accuracy. Furthermore, this algorithm improves the runtime by eliminating duplicate computations.
- Finally, we integrate the dimensionality reduction technique UMAP inside the QMS framework to cluster high-dimensional datasets.

Background and Notations

The following is the idea behind MS algorithms for data clustering: Let's consider dataset $\{x_i\}_{i=1}^{M} \subseteq \mathbb{R}^d$ derived from the probability density function (PDF) $p(z)$. We expect PDF $p(z)$ to have k modes if the dataset contains k clusters. When we apply a gradient-descent-based optimization algorithm to data point x_i as an initial seed, and it converges to $p(z)$'s j-th mode, then data point x_i can be placed in the j-th cluster associated with k-th mode.

Kernel Density Estimation

Unfortunately, PDF $p(z)$ is not available in practice for a real-world dataset. Consequently, we must first estimate the

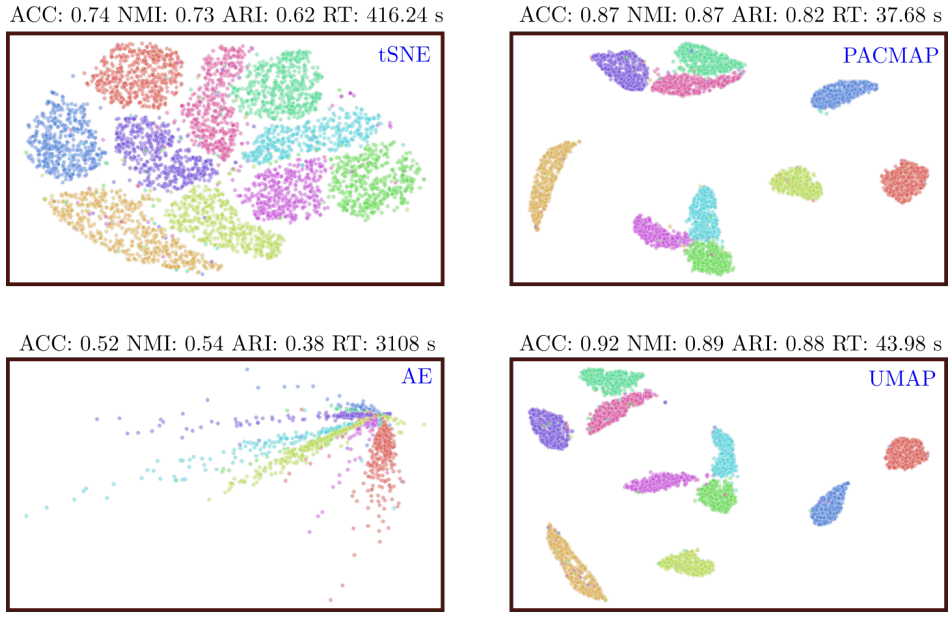

Figure 1: *Comparison of different dimensionality reduction techniques integrated with QMS on the MNIST dataset.* UMAP with QMS returns qualitatively good clustering results with speedups of 10x and 80x over tSNE and AE, respectively. Meanwhile, PACMAP requires less time than UMAP, but its clustering accuracy is not upto the mark.

density of the data set. The Kernel Density Estimator (KDE) is the most popular method for density estimation in the mean shift paradigm. Consider a kernel function $K(z)$ that has the following properties. Given that

$$K(z) \geq 0 \text{ and } \int K(z)dz = 1, \tag{1}$$

the corresponding KDE can be defined as:

$$\hat{p}(z) = \frac{1}{M}\sum_{i=1}^{M} K(z - x_i). \tag{2}$$

There are two popular kernel choices: the Gaussian kernel

$$K_G(z; w) = \frac{c}{w^d} \exp\left(-\frac{||z||^2}{2w^2}\right) \tag{3}$$

and the Epanechnikov kernel

$$K_E(z, w) = \frac{c}{w^d}\left[1 - \frac{||z||^2}{w^2}\right]_+, \tag{4}$$

where c represents the normalizing constants used to ensure that the kernel integrates into one, and parameter w is called bandwidth for controlling the variance of the kernel.

MeanShift++

Using KDE $p(z)$, the MS algorithm seeks to find modes of $\hat{p}(z)$ via the following series of iterations (weighted mean):

$$z^{(t+1)} \leftarrow \frac{\sum_{i=1}^{M} g\left(z^{(t)}, x_i; w\right) x_i}{\sum_{i=1}^{M} g\left(z^{(t)}, x_i; w\right)}, \tag{5}$$

where $g(.)$ is the shadow of the kernel function $K(.)$. For the Gaussian kernel $K_G(.)$, $g(z, x; w) = \exp(||z - x||^2/2w^2)$ and similarly $g(z, x; w) = \mathbb{1}\left(||z - x||^2 < w^2\right)$ is defined for the Epanechnikov kernel.

In (Epanechnikov 1969), it was demonstrated that the Epanechnikov kernel minimizes mean square error (MSE) asymptotically, i.e.

$$\int \left(p(z) - \hat{p}(z)\right)^2 dz. \tag{6}$$

However, even with space-partitioning data structures, the computational cost of $g(.)$ for the Epanechnikov kernel is too high, i.e., $\mathcal{O}(M^2dt)$.

MS++ confronts this issue by first partitioning data points into grid cells. Each grid cell is a hypercube with a side of h. Next, MS++ calculates the mean of each data point based on the data points of its 1-neighboring grid cells, which is similar to the Epanechnikov mean shift process. The Epanechnikov mean shift, on the other hand, selects data points from the hyperspherical neighborhood region with w (or bandwidth) radius to calculate the mean. Hence, h must be $\frac{2w}{3}$ to approximate Epanechnikov's mean shift. MS++ calculates the mean much more quickly since each cell can be determined by dividing the point values. As a result, data points of adjacent grid cells are easily found with $\mathcal{O}(M3^dt)$ time complexity. This makes MS++ suitable for applications involving large datasets with low dimensions, such as image segmentation. The basic steps of the MS++ are shown in the supplementary file.

UMAP

UMAP is a recently proposed manifold learning-based dimensionality reduction technique that can both accurately represent local structure and better incorporate global structure (McInnes, Healy, and Melville 2018). The method has

several advantages over the t-SNE method. The UMAP algorithm is well suited to handling large datasets, unlike the t-SNE algorithm. UMAP is also capable of capturing the advantages from both global and local strategies because it emphasizes preserving distances around local neighbourhoods while keeping the global structure.

Three assumptions underlie UMAP, namely that data are distributed uniformly over a Riemannian space, Riemannian metric is locally constant, and the manifold is always locally connected. Therefore, the manifold can be modeled with a fuzzy topological structure based on these assumptions. By seeking the nearest possible fuzzy topological structure corresponding to the projection of the data on a lower dimensional space, we can construct the embedding.

Similar to Isomap (Tenenbaum, Silva, and Langford 2000), UMAP computes the nearest neighbours of points using a k-neighbour based graph algorithm. The UMAP algorithm constructs a weighted k-neighbour graph first, and then computes a low dimensional layout from this graph. Cross entropy is used to optimize this low dimensional layout so that it has as close a fuzzy topological representation as possible to the original.

There are a number of important hyperparameters that affect performance. Firstly, we need to determine how many neighbors should be considered local. A trade-off is made between local structure preservation and global structure capture in terms of granularity. The primary concern is integrating local structure into the embedding, so lower values for neighbours is typically tuned. The second parameter is the dimension of the embedding. We set the embedding dimension to two in this work. The embedding space must also be separated by a minimum amount. If this minimum distance is lower, the true manifold structure will be captured more accurately, but dense clouds may prevent visualization.

Theoretical Analysis

As the kernel function used in MS++ for density estimation is difficult to define numerically, the authors have not provided it in (Jang and Jiang 2021). However, it is necessary to define for rigorous theoretical analysis of MS++. Therefore, we estimate a kernel function in this paper that satisfies all the conditions for being a kernel function, and that fulfills all the characteristics of the MS++. MS++'s kernel function can be defined as follows.

$$K_M(z,x;w) = \frac{c}{w^d}\left(1 - \frac{||z-x||^2}{w^2}\right)\mathcal{I}(z,x;w), \quad (7)$$

where

$$\mathcal{I}(z,x;w) = \mathbb{1}\left(\left\lfloor\frac{z}{h}\right\rfloor - \left\lfloor\frac{x}{h}\right\rfloor \in \{-1,0,1\}^d\right), \quad (8)$$

and $h = \frac{w}{2\sqrt{d}}$. Therefore, the shadow of this kernel function, $g(.)$, can be defined as $g(z,x;w) = \mathcal{I}(z,x;w)$. Here, MS++ estimates the modes of the KDE $\hat{p}(z) = \sum_{i=1}^{M} K_M(z,x_i;w)$ iteratively based on Eqn. (5).

To conduct further analysis, we eliminate constants and scaling in $\hat{p}$ and K_M and change them to f and ϕ respectively.

$$\begin{aligned}\phi(z,x;w) &= \begin{cases}||z-x||^2 & \text{, if } \mathcal{I}(z,x;w)=1,\\ w^2 & \text{, otherwise,}\end{cases}\\ f(z;w) &= \sum_{i=1}^{M}\phi(z,x_i;w).\end{aligned} \quad (9)$$

These changes do not affect the optimization process and all modes of f are equivalent to all modes of $\hat{p}$. *Due to page limitation, we report the proof of all lemmas and propositions in the supplementary file.*

Lemma 1. *Function $f(z)$ is smooth every where except points $z : \max\left\{\frac{x_i}{h} - \left\lfloor\frac{z}{h}\right\rfloor\right\} = 2,\ \forall i = \{1,2,\ldots,M\}$.*

According to the above lemma, $f(z)$ in MS++ is not smooth everywhere. With the help of the following lemma, we can prove that $f(z)$ is strongly convex at smooth points.

Lemma 2. *For each smooth point z of $f(z)$, we have*

$$\begin{aligned}\nabla f(z) &= \sum_{i\in\mathcal{F}} 2(z-x_i),\\ \nabla^2 f(z) &= 2|\mathcal{F}|I,\end{aligned} \quad (10)$$

where $\mathcal{F} = \left\{i : \left\lfloor\frac{z}{h}\right\rfloor - \left\lfloor\frac{x_i}{h}\right\rfloor \in \{-1,0,1\}^d\right\}$ and I is an identity matrix. This means that $f(z)$ is locally convex at all smooth points. Moreover, if $\mathcal{F}$ is not an empty set, $f(z)$ will also be strongly convex.

Since $f(z)$ has some non-smooth points, we use the concept of directional derivative to analyze its properties. The directional derivative of $f(z)$ for a particular direction μ can be expressed as follows.

$$f'(z;\mu) = \lim_{\alpha\to 0}\frac{f(z+\alpha\mu) - f(z)}{\alpha}. \quad (11)$$

The directional derivative can follow the sum rule, therefore

$$f'(z;\mu) = \sum_{i=1}^{M}\phi'(z,x_i;\mu). \quad (12)$$

It is well known that the directional derivative of $f(z)$ for a smooth point z is simply $f'(z.\mu) = \nabla f(z)^T\mu$.

Using the following definition, we can estimate a stationary point for a non-smooth function (Razaviyayn, Hong, and Luo 2013).

Definition 1. For a given z, if $f(z;\mu) \geq 0$ for all μ, then the point z is a stationary point of $f(.)$.

Note that the above definition reduces to $\nabla f(z)^T\mu \geq 0$ for all μ in case of a smooth point z. On the other hand, $\nabla f(z)^T\mu \geq 0$ for all μ implies $\nabla f(z) = 0$, which is the general definition of a stationary point for a smooth function.

Lemma 3. *If the following conditions*

$$\begin{aligned}&\lim_{\alpha\to 0}\left\lfloor\frac{z+\alpha\mu}{h}\right\rfloor - \left\lfloor\frac{x_i}{h}\right\rfloor \in \{-1,0,1\}^d,\\ &\max\left\{\left\lfloor\frac{z}{h}\right\rfloor - \frac{x_i}{h}\right\} = 2,\\ &||z-x_i||^2 \neq w^2,\ \exists i\end{aligned} \quad (13)$$

are met, it is not possible to derive the directional derivative of f with respect to z in direction μ.

The following interesting facts can be derived from Lemma 3.

Proposition 1. *Considering the case where z is a non-smooth point in $f(z)$, z will never be a stationary point for $f(z)$.*

Since the local minimum of the $f(z)$ is also a stationary point, z cannot be the local minimum if it is a non-smooth point of the $f(z)$.

Proposition 2. *In order for a point z^* to be a local minimum of $f(z)$, the following conditions need to be met:*

1. *No x_j exists such that* $\max\left\{\left\lfloor\frac{x_j}{h}\right\rfloor - \left\lfloor\frac{z}{h}\right\rfloor\right\} = 2$,
2. *the set $\mathcal{F} \neq \emptyset$, and*
3. $z^* = \frac{1}{|\mathcal{F}|}\sum_{i\in\mathcal{F}} x_i$.

Therefore, MS++ does not provide the local minimum of the estimated density if one of the conditions listed above does not hold after convergence.

Modified MeanShift++

Prior to proposing the modification, we examine the convergence of MS++.

Lemma 4. *With MS++, the function $f(z)$ is successively minimized at each iteration, therefore $f(z)$ does not monotonically increase as the iteration count increases.*

It is obvious from the Majorization-Minimization interpretation that MS++ tends to converge to a stationary point. However, this interpretation only applicable for a smooth function. Due to the non-smooth nature of $f(z)$ and $\hat{f}(z \mid \breve{z})$, no convergence results exist to prove that MS++ always converges to the stationary point. With a non-zero probability, it can so happen that $z^{(t-1)} = z^{(t)} = \arg\min_{\ddot{z}} \hat{f}(\ddot{z} \mid z^{(t-1)})$ for few x_i, $i \in \left\{i : \max\left\{\frac{x_i}{h} - \left\lfloor\frac{z}{h}\right\rfloor\right\} = 2\right\}$. Based on Propositions 1 and 2, this implies that $z^{(t)}$, even after convergence of MS++, cannot be the stationary point (or local minimum) of $f(z)$.

Algorithm 1: Modified MeanShift++

Require: dataset $X \in \mathbb{R}^d$, $i = 1, \ldots, M$; bandwidth $h > 0$; tolerance η.
Ensure: $\{x_i\}\ i = 1, \ldots, M$.
Initialize $x_i^{(0)} \leftarrow X_i$, $\forall i = 1, \ldots, M$ and $t = 0$;
Initialize empty hash tables $\mathcal{C}$ (stores cell counts) and $\mathcal{S}$ (stores cell sum);
repeat
Step 1: $\mathcal{C}\left(\left\lfloor x_i^{(t)}/h \right\rfloor\right) \leftarrow \mathcal{C}\left(\left\lfloor x_i^{(t)}/h \right\rfloor\right) + 1$, for $i \in \{1, M\}$;
Step 2: $\mathcal{S}\left(\left\lfloor x_i^{(t)}/h \right\rfloor\right) \leftarrow \mathcal{S}\left(\left\lfloor x_i^{(t)}/h \right\rfloor\right) + x_i^{(t)}$, for $i \in \{1, M\}$;
Step 3: $x_i^{(t+1)} \leftarrow \frac{\sum_{v\in\{-1,0,1\}^d} \mathcal{S}\left(\left\lfloor x_i^{(t)}/h \right\rfloor + v\right)}{\sum_{v\in\{-1,0,1\}^d} \mathcal{C}\left(\left\lfloor x_i^{(t)}/h \right\rfloor + v\right)}$ for $i \in \{1, M\}$;
Step 4: If $\sum_{i=1}^M ||x_i^{(t+1)} - x_i^{(t)}|| \geq \eta$ **then**
Step 5: $\mathcal{K}_i \leftarrow \left\{j : \max\left\{\frac{x_j}{h} - \lfloor\frac{x_i}{h}\rfloor\right\} = 2\right\}$ for $i \in \{1, M\}$;
Step 6: $x_i^{(t+1)} \leftarrow \frac{\sum_{j\in\mathcal{K}_i} x_j^{(t)} + \sum_{v\in\{-1,0,1\}^d} \mathcal{S}\left(\left\lfloor x_i^{(t)}/h \right\rfloor + v\right)}{|\mathcal{K}_j| + \sum_{v\in\{-1,0,1\}^d} \mathcal{C}\left(\left\lfloor x_i^{(t)}/h \right\rfloor + v\right)}$ for $i \in \{1, M\}$;
Step 7: end if
Step 8: $t \leftarrow t + 1$, $\mathcal{C} \leftarrow \emptyset$, and $\mathcal{S} \leftarrow \emptyset$;
until $\sum_{i=1}^M ||x_i^{(t)} - x_i^{(t-1)}|| \geq \eta$

To address this issue, we modify the MS++ steps to account for such situations. The main steps of the modified MS++ are depicted in Algorithm 1. As per Algorithm 1, we first pick samples x_j such that $\max\left\{\frac{x_j}{h} - \left\lfloor\frac{z}{h}\right\rfloor\right\} = 2$ for resolving this problem. Then, we re-update the z as the weighted mean of samples x_j together with all samples x_i, $i \in \left\{i : \left\lfloor\frac{z}{h}\right\rfloor - \left\lfloor\frac{x_i}{h}\right\rfloor \in \{-1, 0, 1\}\right\}^d$. This allows us to redefine

$$\begin{aligned} z^{(t)} &= \arg\min_{\ddot{z}} \tilde{f}(\ddot{z}|z^{(t-1)}) \\ &= \sum_{j\in\mathcal{K}} ||z^{(t-1)} - x_j||^2 + \sum_{i\in\mathcal{F}} ||z^{(t-1)} - x_i||^2 \\ &\quad + (M - |\mathcal{F}| - |\mathcal{K}|)\, w^2, \end{aligned} \tag{14}$$

where $\mathcal{K} = \left\{j : \max\left\{\frac{x_j}{h} - \left\lfloor\frac{z^{(t-1)}}{h}\right\rfloor\right\} = 2\right\}$ and $\mathcal{F} = \left\{i : \left\lfloor\frac{z^{(t-1)}}{h}\right\rfloor - \lfloor\frac{x_i}{h}\rfloor \in \{-1, 0, 1\}^d\right\}$.

Next, we demonstrate that modified MS++ (shown in Algorithm 1) calculates a local minimum within a finite number of iterations.

Lemma 5. *With modified MS++, the value of $f(z)$ is strictly decreasing till $z^{(t)} = z^{(t-1)}$.*

Theorem 6. *Algorithm 1 terminates at a local minimum of (9) in a finite number of iterations.*

We may observe that, even though we are trying to optimize a non-convex and non-smooth function, we obtain a very strong convergence result: modified MS++ reaches a local minimum after a finite number of iterations.

Proposed Algorithm

In this section, we introduce UEQMS, which qualifies as a faster version of MS++ and can be applied to high-dimensional datasets. First, we describe the main steps of QMS.

MS++ calculates the shifted position of all data points in each iteration. Nevertheless, shifted positions of data points belonging to the same grid have the same value. Therefore, the shifted positions of all data points do not need to be stored for subsequent MS++ iterations. Using unique shifted positions, we can update the shifted position of all data points. Consequently, we only save unique shifted positions with their repetition count in QMS, shown in Algorithm-2. A new hash table is created in Algorithm-2, which keeps the number of repetitions of each unique shifted position. Moreover, the equation used to calculate shifted locations is updated based on the repetition count of each unique shifted position. QMS ultimately produces the same or better result than MS++ since we use the same kernel function and density estimation alongside the same optimization process with the proposed modified step. *The source code of QMS is available online at* https://github.com/abhisheka456/QuickMeanShiftPP.

We compare QMS with MS++ and α-MS++ on various clustering datasets. To provide a fair comparison, we used the same datasets employed in the original MS++ paper (Jang and Jiang 2021). The characteristics of these

Algorithm 2: Quick MeanShift

Require: dataset $X \in \mathbb{R}^d$, $i = 1, \ldots, M$; bandwidth $h > 0$; tolerance η.
Ensure: $\{x_i\}\ i = 1, \ldots, M$.
Initialize $y_i^{(0)} \leftarrow X_i, ny_i^{(0)} \leftarrow 1\ \forall i = 1, \ldots, M$, and $t = 0$;
Initialize empty hash tables $\mathcal{C}$ (stores cell counts) and $\mathcal{S}$ (stores cell sum);
repeat
Step 1: $\mathcal{C}\left(\left\lfloor y_i^{(t)}/h \right\rfloor\right) \leftarrow \mathcal{C}\left(\left\lfloor y_i^{(t)}/h \right\rfloor\right) + ny_i^{(t)}$, for $i \in \{1, M\}$;
Step 2: $\mathcal{S}\left(\left\lfloor y_i^{(t)}/h \right\rfloor\right) \leftarrow \mathcal{S}\left(\left\lfloor y_i^{(t)}/h \right\rfloor\right) + ny_i^{(t)} y_i^{(t)}$, for $i \in \{1, M\}$;
Step 3: $y_i^{(t+1)} \leftarrow \frac{\sum_{v \in \{-1,0,1\}^d} \mathcal{S}\left(\left\lfloor y_i^{(t)}/h \right\rfloor + v\right)}{\sum_{v \in \{-1,0,1\}^d} \mathcal{C}\left(\left\lfloor y_i^{(t)}/h \right\rfloor + v\right)}$ for $i \in \{1, M\}$;
Step 4: If $\sum_{i=1}^{M} ||y_i^{(t+1)} - y_i^{(t)}|| \geq \eta$ **then**
Step 5: $\mathcal{K}_i \leftarrow \left\{j : \max\left\{\frac{y_j}{h} - \lfloor \frac{y_i}{h} \rfloor\right\} = 2\right\}$ for $i \in \{1, M\}$;
Step 6: $y_i^{(t+1)} \leftarrow \frac{\sum_{j \in \mathcal{K}_i} ny_j^{(t)} y_j^{(t)} + \sum_{v \in \{-1,0,1\}^d} \mathcal{S}\left(\left\lfloor y_i^{(t)}/h \right\rfloor + v\right)}{\sum_{j \in \mathcal{K}_i} ny_j^{(t)} + \sum_{v \in \{-1,0,1\}^d} \mathcal{C}\left(\left\lfloor y_i^{(t)}/h \right\rfloor + v\right)}$ for $i \in \{1, M\}$;
Step 7: end if
Step 8: $t \leftarrow t + 1$, $\mathcal{C} \leftarrow \emptyset$, and $\mathcal{S} \leftarrow \emptyset$;
Step 9: Update $\{y_i^{(t)}\}$ and $\{ny_i^{(t)}\}$;
until $\sum_{i=1}^{M} ||y_i^{(t)} - y_i^{(t-1)}|| \geq \eta$

datasets in terms of the number of features, clusters, and data points are reported in the supplementary file. QMS is implemented in Cython and the comparative study is done using the Cython implementation of MS++ as provided in (Jang and Jiang 2021) and α-MS++ implemented by us. The comparison between QMS and MS will not be made here, since MS++ and α-MS++ are faster variants of MS (Jang and Jiang 2021; Park 2021b).

To measure the quality of the clustering results, we use two indices: Adjusted Mutual Information (AMI) (Vinh, Epps, and Bailey 2010) and Adjusted Rand Index (ARI) (Hubert and Arabie 1985). By comparing the calculated labels to the actual labels of the clusters, these two indices are one of the popular ways to evaluate clustering performance (Jang and Jiang 2019).

Here, we validate QMS with MS++ and α-MS++ on 6 standard low-dimensional datasets with a range of data points from 15 to 0.3 millions. In Table 2, the results with respect to AMI, ARI, and runtime are shown for all algorithms. Table 2 shows that QMS provides better results with a 40x and 20x faster runtime compared to MS++, and α-MS++, respectively, on most datasets. Due to the superior clustering quality and runtime performance, QMS outperforms MS++ and α-MS++ on these datasets.

It should be noted that α-MS++ also does not require the calculation of shifted positions for all data points. However, in α-MS++, updating at least one data point of each grid requires shifted positions of all data points. Therefore, α-MS++ utilizes shifted positions of all data points in each iteration. After updating the shifted position of one data point, the shifted position of other data points belonging to the same grid will be updated with the same value without having to recalculate. QMS does not process all data points in each iteration, so it is faster than α-MS++ even though α-MS++ improves the runtime of MS++. Therefore, we can conclude that QMS is the quickest variant of MS++.

As QMS performs better than MS++ and α-MS++, we use QMS as our core clustering algorithm in UEQMS. However, it cannot be applied to datasets with high dimensions. Using UMAP, we extract the low-dimensional features from high-dimensional datasets. The basic steps of UEQMS are as follows:

1. We first use UMAP to the raw data to search for a low-dimensional and better clusterable manifold that preserves local topology.
2. Then, we apply QMS clustering algorithm to discover the clusters.

Dataset	ARI		
	QMS	MS++	α-MS++
Phone Accelerometer	**0.0897** (37.49s)	**0.0897** (1599.34s)	0.0896 (475.58s)
Phone Gyroscope	0.2354 (108.97s)	0.2354 (5324.19s)	**0.2355** (934.28s)
Watch Accelerometer	**0.0913** (18.75s)	**0.0913** (926.59s)	0.0908 (319.29s)
Watch Gyroscope	0.1598 (26.02s)	0.1593 (1247.08s)	**0.1602** (416.85s)
Still	0.79 (0.17s)	0.7899 (8.23s)	**0.7901** (3.12s)
Skin	**0.3264** (0.28s)	**0.3264** (12.58s)	**0.3264** (3.28s)
Dataset	AMI		
	QMS	MS++	α-MS++
Phone Accelerometer	**0.1959** (46.01s)	**0.1959** (2523.85s)	0.1921 (646.27s)
Phone Gyroscope	**0.1835** (46.01s)	**0.1835** (1723.19s)	0.1824 (780.35s)
Watch Accelerometer	**0.2309** (41.04s)	**0.2309** (2500.97s)	0.2301 (658.37s)
Watch Gyroscope	**0.1402** (11.87s)	0.1336 (484.27s)	0.1397 (180.24s)
Still	0.8557 (0.12s)	0.8551 (6.23s)	**0.8599** (2.62s)
Skin	**0.4238** (0.18s)	**0.4238** (9.28s)	0.4234 (2.92s)

Table 2: *Comparison of QMS, MS++, and α-MS++ based on ARI and AMI scores over* 6 *datasets*. These scores are calculated after tuning the bandwidth on each dataset for each algorithm separately. Best scores are reported in bold fonts. Compared to other algorithms, QMS provides the best scores or same score in ARI and AMI scores over majority of the dataset with faster runtime speed. QMS is 40x and 20x faster than MS++ and α-MS++, respectively.

Empirical Performance and Analysis

To validate the effectiveness of the proposed UEQMS algorithm, we conduct extensive experiments in this section. Due to tight page constraints, we have provided a pertinent ablation study, runtime comparisons, and additional experiments in the supplementary document.

Comparison with Automated Clustering Algorithm

In this section, we examine the performance of UEQMS compared to state-of-the-art clustering algorithms on va-

rieties of popular datasets. Our proposed algorithm can be utilized for automated clustering. Therefore, we consider the following algorithms as contenders: Mean-Shift (MS) (Comaniciu and Meer 1999), Weighted Blurring Mean Shift (WBMS) (Chakraborty, Paul, and Das 2020), Medoid-Shift (MeS) (Sheikh, Khan, and Kanade 2007), DBSCAN (DB) (Ester et al. 1996), DBSCAN++ (DB++) (Jang and Jiang 2019), Quick-Shift (QS) (Vedaldi and Soatto 2008), Quick-Shift++ (QS++) (Jiang, Jang, and Kpotufe 2018) and Robust Continuous Clustering (RCC) (Shah and Koltun 2017). We consider 15 real datasets collected from the different repositories such as UCI (Asuncion and Newman 2007), Keel (Alcalá-Fdez et al. 2011), and ASU (Li et al. 2017).

The characteristics of these datasets in terms of the number of features, clusters, and data points are reported in the supplementary file. For a fair comparison, we tune the parameters of all peer algorithms for each problem separately. Although UMAP's parameters affect performance for visualization applications, clustering performance remains consistent. Thus, in all experiments, we used UMAP's default settings except for min_dist. The min_dist is set at 0.0 to create compact clusters. We report the optimal parameter value of UEQMS for real and feature selection datasets in the supplementary file. Outcomes of each algorithm in terms of AMI and ARI values are reported in Table 3. As reported in Table. 3, the UEQMS algorithm shows superior performance compared to other algorithms on most datasets. This case study concludes that UEQMS provides better clusters for real datasets compared to state-of-the-art automated clustering techniques.

Mode-seeking models are severely harmed by high-dimensional datasets, with few useful features and many noisy features (see Table 3). DR essentially transforms high-dimensional feature space into low-dimensional feature space to resolve this issue. While transforming, meaningful properties should not disappear. UMAP is a DR method that can preserve local and global topological structures of the data. It maps nearby points on the manifold to nearby points in the low-dimensional representation and the reverse. Table 4 (supplementary file) shows that MS and DB incorporating UMAP provide better results on HD data than MS and DB. The results show that UMAP provides less noisy embedding.

Comparison with Deep Clustering Algorithm

This section conducts an experimental study on large-scale high dimensional datasets. For this purpose, we consider four image datasets. The same computational protocols stated in the previous section are followed in this study. We report the properties of the datasets in terms of the number of data points, the number of features, and the number of true clusters in the supplementary file. Here, we consider state-of-the-art deep clustering algorithms as contenders: DeepCluster (DC) (Caron et al. 2018), dep clustering networks (DCN) (Yang et al. 2017), deep k-means (DKM) (Fard, Thonet, and Gaussier 2020), deep embedded clustering (DEC) (Xie, Girshick, and Farhadi 2016), improved deep embedded clustering (IDEC) (Guo et al. 2017), soft regularised k-means (SR-KM) (Jabi et al. 2019), varia-

Dataset	Score	MS	WBMS	MeS	DB	UEQMS
ALLAML	AMI	0.00	0.17	0.00	0.00	0.47
	ARI	0.00	0.13	0.00	0.00	0.50
arcene	AMI	0.00	0.07	0.00	0.00	0.23
	ARI	0.00	0.10	0.00	0.00	0.09
GLIOMA	AMI	0.00	0.53	0.00	0.00	0.58
	ARI	0.00	0.40	0.00	0.00	0.45
leukemia	AMI	0.00	0.32	0.00	0.00	0.36
	ARI	0.00	0.27	0.00	0.00	0.43
orlraws10P	AMI	0.00	0.56	0.00	0.00	0.89
	ARI	0.00	0.54	0.00	0.00	0.83
TOX_171	AMI	0.00	0.17	0.00	0.00	0.28
	ARI	0.00	0.11	0.00	0.00	0.33
warpAR10P	AMI	0.00	0.23	0.00	0.00	0.20
	ARI	0.00	0.11	0.00	0.00	0.12
lymphoma	AMI	0.00	0.71	0.00	0.00	0.75
	ARI	0.00	0.72	0.00	0.00	0.74
cacmcisi	AMI	0.13	0.04	0.13	0.27	0.64
	ARI	0.06	0.03	0.09	0.12	0.69
dermatology	AMI	0.02	0.70	0.02	0.03	0.96
	ARI	0.00	0.49	0.00	0.00	0.96
iono	AMI	0.13	0.00	0.11	0.15	0.32
	ARI	0.06	0.00	0.08	0.01	0.34
segment	AMI	0.59	0.03	0.60	0.59	0.70
	ARI	0.44	0.00	0.46	0.36	0.58
vehicle	AMI	0.22	0.27	0.12	0.15	0.25
	ARI	0.06	0.13	0.03	0.03	0.16
vowel	AMI	0.40	0.42	0.40	0.38	0.46
	ARI	0.08	0.25	0.08	0.06	0.22
wdbc	AMI	0.03	0.57	0.01	0.05	0.51
	ARI	0.00	0.56	0.00	0.00	0.61
Dataset	Score	DB++	QS	QS++	RCC	UEQMS
ALLAML	AMI	0.00	0.05	0.00	0.18	0.47
	ARI	0.00	0.04	0.10	0.00	0.50
arcene	AMI	0.00	0.05	0.00	0.11	0.23
	ARI	0.00	0.02	0.10	0.00	0.09
GLIOMA	AMI	0.00	0.14	0.00	0.42	0.58
	ARI	0.00	0.05	0.30	0.00	0.45
leukemia	AMI	0.00	0.03	0.43	0.00	0.36
	ARI	0.00	0.04	0.56	0.00	0.43
orlraws10P	AMI	0.00	0.34	0.82	0.00	0.89
	ARI	0.00	0.13	0.77	0.00	0.83
TOX_171	AMI	0.00	0.01	0.25	0.00	0.28
	ARI	0.00	0.00	0.08	0.00	0.33
warpAR10P	AMI	0.00	0.01	0.18	0.00	0.20
	ARI	0.00	-0.11	0.09	0.00	0.12
lymphoma	AMI	0.00	0.01	0.76	0.36	0.75
	ARI	0.00	0.01	0.73	0.50	0.74
cacmcisi	AMI	0.37	0.18	0.19	0.02	0.64
	ARI	0.42	0.37	0.31	0.16	0.69
dermatology	AMI	0.02	0.68	0.70	0.43	0.96
	ARI	0.00	0.52	0.59	0.62	0.96
iono	AMI	0.09	0.21	0.19	0.03	0.32
	ARI	0.10	0.22	0.16	0.13	0.34
segment	AMI	0.64	0.49	0.61	0.09	0.70
	ARI	0.53	0.25	0.40	0.48	0.58
vehicle	AMI	0.22	0.17	0.26	0.11	0.25
	ARI	0.10	0.10	0.14	0.29	0.16
vowel	AMI	0.39	0.22	0.45	0.05	0.46
	ARI	0.18	0.08	0.22	0.40	0.22
wdbc	AMI	0.02	0.03	0.43	0.12	0.51
	ARI	0.00	0.03	0.56	0.25	0.61

Table 3: *Performance Analysis on Feature Selection and Real Datasets in terms of AMI and ARI values.*

Dataset	Score	DC	DCN	DKM	DEC	IDEC	SR-KM
MNIST	ACC	0.797	0.830	0.840	0.863	0.881	0.939
	NMI	0.661	0.810	0.796	0.834	0.867	0.866
USPS	ACC	0.562	0.688	0.757	0.762	0.761	0.901
	NMI	0.540	0.683	0.776	0.767	0.785	0.912
Fashion	ACC	0.542	0.501	–	0.518	0.529	0.507
	NMI	0.510	0.558	–	0.546	0.557	0.548
pendigit	ACC	–	0.720	–	0.701	0.784	
	NMI	–	0.690	–	0.678	0.723	
Dataset	Score	VaDE	CGAN	JULE	DEPICT	EDESC	UEQMS
MNIST	ACC	0.945	0.950	0.964	**0.965**	0.913	0.913
	NMI	0.876	0.890	0.913	**0.917**	0.862	0.887
USPS	ACC	0.566	–	**0.950**	0.899		0.952
	NMI	0.512	–	**0.913**	0.906	–	0.892
Fashion	ACC	0.578	0.630	0.563	0.392	**0.631**	0.586
	NMI	0.630	0.640	0.608	0.392	**0.670**	0.664
pendigit	ACC	–	0.770	–	–		**0.888**
	NMI	–	0.730	–	–	–	**0.862**

Table 4: *Clustering Performance of Different Deep Clustering Algorithms and UEQMS in terms ACC and NMI.* The mark "–" denotes that the results is unavailable from paper or code.

tional deep embedding (VaDE) (Jiang et al. 2016), clustering with GAN (CGAN) (Mukherjee et al. 2019), joint unsupervised learning (JULE) (Yang, Parikh, and Batra 2016), deep embedded regularized clustering (DEPICT) (Ghasedi Dizaji et al. 2017), and efficient deep embedded subspace clustering (EDESC) (Cai et al. 2022).

The optimal parameters value of UEQMS for image datasets are reported in the supplementary file. We compare the performance with respect to their accuracy (ACC) and NMI values on each dataset. we report the outcomes in Table 4. As shown in Table. 4, UEQMS outperforms other algorithms for the majority of the datasets as ACC and NMI of UEQMS is significantly better than other ones. This study suggests that UEQMS is the best algorithm among considered peer algorithms for large-scale high-dimensional datasets.

Conclusion

In this work, we investigate the convergence of MS++. We observed that it works well but that, as of this writing, there is no theoretical analysis of its performance. Our study paid more attention to the non-smoothness of its kernel function. We show that there are some instances where non-smoothness affects MS++ convergence, and MS++ converges to a non-stationary point. We propose a simple remedy to address this problem. The proposed remedy enhances the MS++ convergence, where its convergence is guaranteed at a local minimum within a finite number of iterations. We also introduce a faster version of MS++, QMS, which avoids duplicate computations in each iteration. Consequently, QMS has lower computational and memory space requirements than MS++ while maintaining the same or better clustering performance.

The application of QMS and MS++ based algorithms is only limited to low-dimensional datasets. In this work, we also address this issue and propose an algorithm named UEQMS. In UEQMS, we first apply UMAP to extract the low-dimensional features of high-dimensional datasets without sacrificing the local and global structure. Then QMS is applied to discover the clusters using these low-dimensional features. The proposed method has been verified to be more accurate than its competitors through experiments on various datasets. Moreover, it's possible to develop a Deep Neural Network-based AE that works within the MS-based framework, where the encoder can be trained using binary cross entropy loss and decoder by reconstruction loss. We consider this work as a potential future research avenue.

Acknowledgments

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2021R1I1A3049810).

References

Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; and Süsstrunk, S. 2012. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11): 2274–2282.

Alcalá-Fdez, J.; Fernández, A.; Luengo, J.; Derrac, J.; García, S.; Sánchez, L.; and Herrera, F. 2011. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17.

Andrew, G.; Arora, R.; Bilmes, J.; and Livescu, K. 2013. Deep canonical correlation analysis. In *International conference on machine learning*, 1247–1255. PMLR.

Asuncion, A.; and Newman, D. 2007. UCI Machine Learning Repository. Available online at https://archive.ics.uci.edu/ml/index.php. Accessed 15 March 2022.

Barash, D.; and Comaniciu, D. 2004. A common framework for nonlinear diffusion, adaptive smoothing, bilateral filtering and mean shift. *Image Vis. Comput.*, 22: 73–81.

Bigdeli, S. A.; Zwicker, M.; Favaro, P.; and Jin, M. 2017. Deep Mean-Shift Priors for Image Restoration. In *NIPS*.

Cai, J.; Fan, J.; Guo, W.; Wang, S.; Zhang, Y.; and Zhang, Z. 2022. Efficient Deep Embedded Subspace Clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1–10.

Caron, M.; Bojanowski, P.; Joulin, A.; and Douze, M. 2018. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, 132–149.

Chakraborty, S.; Paul, D.; and Das, S. 2020. Automated Clustering of High-dimensional Data with a Feature Weighted Mean Shift Algorithm. *arXiv preprint arXiv:2012.10929*.

Cheng, Y. 1995. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8): 790–799.

Comaniciu, D.; and Meer, P. 1999. Mean shift analysis and applications. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, 1197–1203. IEEE.

Comaniciu, D.; Ramesh, V.; and Meer, P. 2003. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5): 564–577.

Epanechnikov, V. A. 1969. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1): 153–158.

Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X.; et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, 226–231.

Estévez, P. A.; Tesmer, M.; Perez, C. A.; and Zurada, J. M. 2009. Normalized mutual information feature selection. *IEEE Transactions on neural networks*, 20(2): 189–201.

Fard, M. M.; Thonet, T.; and Gaussier, E. 2020. Deep k-means: Jointly clustering with k-means and learning representations. *Pattern Recognition Letters*, 138: 185–192.

Felzenszwalb, P. F.; and Huttenlocher, D. P. 2004. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2): 167–181.

Ghasedi Dizaji, K.; Herandi, A.; Deng, C.; Cai, W.; and Huang, H. 2017. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision*, 5736–5745.

Guo, W.; Lin, K.; and Ye, W. 2021. Deep Embedded K-Means Clustering. In *2021 International Conference on Data Mining Workshops (ICDMW)*, 686–694. IEEE.

Guo, X.; Gao, L.; Liu, X.; and Yin, J. 2017. Improved deep embedded clustering with local structure preservation. In *Ijcai*, 1753–1759.

Huang, D.; Wang, C.-D.; Wu, J.-S.; Lai, J.-H.; and Kwoh, C.-K. 2020. Ultra-Scalable Spectral Clustering and Ensemble Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 32(6): 1212–1226.

Hubert, L.; and Arabie, P. 1985. Comparing partitions. *Journal of classification*, 2(1): 193–218.

Ismkhan, H. 2018. I-k-means-+: An iterative clustering algorithm based on an enhanced version of the k-means. *Pattern Recognition*, 79: 402–413.

Jabi, M.; Pedersoli, M.; Mitiche, A.; and Ayed, I. B. 2019. Deep clustering: On the link between discriminative models and k-means. *IEEE transactions on pattern analysis and machine intelligence*, 43(6): 1887–1896.

Jang, J.; and Jiang, H. 2019. DBSCAN++: Towards fast and scalable density clustering. In *International Conference on Machine Learning*, 3019–3029. PMLR.

Jang, J.; and Jiang, H. 2021. MeanShift++: Extremely Fast Mode-Seeking With Applications to Segmentation and Object Tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4102–4113.

Jiang, H.; Jang, J.; and Kpotufe, S. 2018. Quickshift++: Provably good initializations for sample-based mean shift. In *International Conference on Machine Learning*, 2294–2303. PMLR.

Jiang, Z.; Zheng, Y.; Tan, H.; Tang, B.; and Zhou, H. 2016. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*.

Khan, K.; Rehman, S. U.; Aziz, K.; Fong, S.; and Sarasvady, S. 2014. DBSCAN: Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, 232–238. IEEE.

Kumar, A.; Ajani, O. S.; Das, S.; and Mallipeddi, R. 2022. GridShift: A Faster Mode-seeking Algorithm for Image Segmentation and Object Tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8131–8139.

Lee, D. D.; and Seung, H. S. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755): 788–791.

Leichter, I.; Lindenbaum, M.; and Rivlin, E. 2010. Mean Shift tracking with multiple reference color histograms. *Comput. Vis. Image Underst.*, 114: 400–408.

Li, J.; Cheng, K.; Wang, S.; Morstatter, F.; Trevino, R. P.; Tang, J.; and Liu, H. 2017. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6): 1–45.

McInnes, L.; Healy, J.; and Melville, J. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Mukherjee, S.; Asnani, H.; Lin, E.; and Kannan, S. 2019. Clustergan: Latent space clustering in generative adversarial networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 4610–4617.

Ning, J.; Zhang, L.; Zhang, D. D.; and Wu, C. 2012. Robust mean-shift tracking with corrected background-weighted histogram. *Iet Computer Vision*, 6: 62–69.

Paris, S. 2008. Edge-Preserving Smoothing and Mean-Shift Segmentation of Video Streams. In Forsyth, D.; Torr, P.; and Zisserman, A., eds., *Computer Vision – ECCV 2008*, 460–473. Berlin, Heidelberg: Springer Berlin Heidelberg.

Paris, S.; and Durand, F. 2007. A Topological Approach to Hierarchical Segmentation using Mean Shift. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8.

Park, H. 2021a. α-meanshift++: Improving meanshift++ for image segmentation. *IEEE Access*, 9: 131430–131439.

Park, H. 2021b. α-MeanShift++: Improving MeanShift++ for Image Segmentation. *IEEE Access*, 9: 131430–131439.

Park, J. H.; Lee, G. S.; and Park, S. Y. 2009. Color image segmentation using adaptive mean shift and statistical model-based methods. *Computers & Mathematics with Applications*, 57(6): 970–980. Advances in Fuzzy Sets and Knowledge Discovery.

Pearson, K. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11): 559–572.

Razaviyayn, M.; Hong, M.; and Luo, Z.-Q. 2013. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2): 1126–1153.

Santos, J. M.; and Embrechts, M. 2009. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *International conference on artificial neural networks*, 175–184. Springer.

Shah, S. A.; and Koltun, V. 2017. Robust continuous clustering. *Proceedings of the National Academy of Sciences*, 114(37): 9814–9819.

Sheikh, Y. A.; Khan, E. A.; and Kanade, T. 2007. Mode-seeking by medoidshifts. In *2007 IEEE 11th international conference on computer vision*, 1–8. IEEE.

Tao, W.; Jin, H.; and Zhang, Y. 2007. Color Image Segmentation Based on Mean Shift and Normalized Cuts. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(5): 1382–1389.

Tenenbaum, J. B.; Silva, V. d.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500): 2319–2323.

Tschannen, M.; Bachem, O.; and Lucic, M. 2018. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Vedaldi, A.; and Soatto, S. 2008. Quick shift and kernel methods for mode seeking. In *European conference on computer vision*, 705–718. Springer.

Vinh, N. X.; Epps, J.; and Bailey, J. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11: 2837–2854.

Wang, Y.; Huang, H.; Rudin, C.; and Shaposhnik, Y. 2021. Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMap, and PaCMAP for Data Visualization. *J. Mach. Learn. Res.*, 22(201): 1–73.

Xie, J.; Girshick, R.; and Farhadi, A. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, 478–487. PMLR.

Yang, B.; Fu, X.; Sidiropoulos, N. D.; and Hong, M. 2017. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, 3861–3870. PMLR.

Yang, J.; Parikh, D.; and Batra, D. 2016. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5147–5156.

Zhao, W.-L.; Deng, C.-H.; and Ngo, C.-W. 2018. k-means: A revisit. *Neurocomputing*, 291: 195–206.

Zhou, H.; Wang, X.; and Schaefer, G. 2011. *Mean Shift and Its Application in Image Segmentation*, 291–312. Berlin, Heidelberg: Springer Berlin Heidelberg.