A Weighted Adaptive Mean Shift Clustering Algorithm

Yazhou Ren* Carlotta Domeniconi† Guoji Zhang‡ Guoxian Yu§

Abstract

The mean shift algorithm is a nonparametric clustering technique that does not make assumptions on the number of clusters and on their shapes. It achieves this goal by performing kernel density estimation, and iteratively locating the local maxima of the kernel mixture. The set of points that converge to the same mode defines a cluster. While appealing, the performance of the mean shift algorithm significantly deteriorates with high dimensional data due to the sparsity of the input space. In addition, noisy features can create challenges for the mean shift procedure.

In this paper we extend the mean shift algorithm to overcome these limitations, while maintaining its desirable properties. To achieve this goal, we first estimate the relevant subspace for each data point, and then embed such information within the mean shift algorithm, thus avoiding computing distances in the full dimensional input space. The resulting approach achieves the best-of-two-worlds: effective management of high dimensional data and noisy features, while preserving a nonparametric nature. Our approach can also be combined with random sampling to speedup the clustering process with large scale data, without sacrificing accuracy. Extensive experimental results on both synthetic and real-world data demonstrate the effectiveness of the proposed method.

Keywords: mean shift, subspace clustering, feature relevance estimation, curse of dimensionality.

1 Introduction.

Clustering is the key step for many tasks in data mining. The clustering problem concerns the discovery of homogeneous groups of data according to a certain similarity measure. Traditional clustering algorithms (i.e., k-means) have two inherent disadvantages: the number of clusters must be fixed a-priori, and the generated clusters are restricted to have a spherical or elliptical shape [15].

In contrast, the mean shift procedure, a nonparametric clustering technique, can overcome these two drawbacks. It achieves this goal by performing kernel density estimation, and iteratively locating the local maxima of the kernel mixture. The set of points that converge to the same mode defines a cluster [7, 15]. The key parameter of mean shift is the bandwidth. The original mean shift procedure uses a fixed bandwidth, while the adaptive mean shift [8] sets a different bandwidth value for each point.

While appealing, the performance of the mean shift algorithm significantly deteriorates with high dimensional data due to the sparsity of the input space. Noisy features can also challenge the search of dense regions in the full space. An example is given in Fig. 1. The three classes are generated by two-dimensional Gaussians embedded in a three-dimensional space. As we show in Section 4, the mean shift algorithm does not perform well in this scenario.

Local-sensitive hashing (LSH) has been used to reduce the computational complexity of adaptive mean shift, which is quadratic in the number of points [15]. Freedman *et al.* [13] proposed a fast mean shift procedure based on random sampling. However, these techniques still compute distances between points, and performs mean shift, in the full space.

As Fig. 1 shows, clusters may exist in different subspaces, comprised of different combinations of features. This is often the case with high-dimensional data. Subspace clustering techniques have been developed to detect clusters of data, as well as the subspaces where the clusters exist [19, 20, 24]. In this work we leverage ideas from the subspace clustering literature to address the aforementioned limitations of the mean shift procedure. In particular, we develop a technique to estimate the relevant subspace for each point, and then embed such information within the mean shift search process. This results in a methodology that effectively handles high dimensional data and noisy features, while preserving a nonparametric nature. Our approach can also be combined with random sampling to speedup the clustering process with large scale data, without sacrificing accuracy. The main contributions of this paper are summarized as follows:

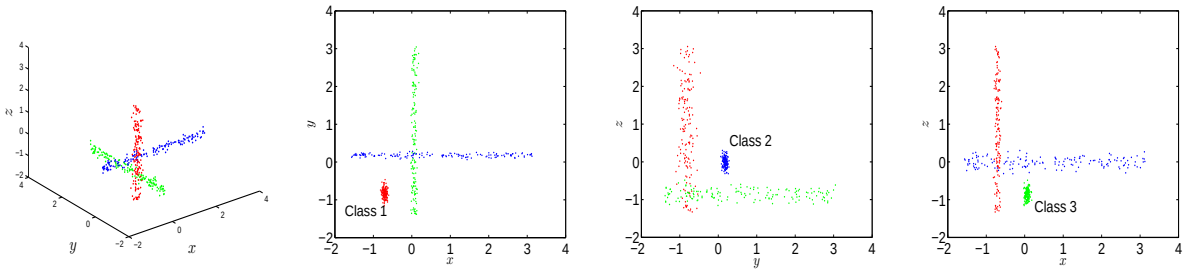
- To the best of our knowledge, this is the first

*School of Computer Science and Engineering, South China University of Technology. Email: yazhou.ren@mail.scut.edu.cn.

†Department of Computer Science, George Mason University, USA. Email: carlotta@cs.gmu.edu.

‡School of Sciences, South China University of Technology. Email: magjzh@scut.edu.cn.

§College of Computer and Information Science, Southwest University, China. Email: gxyu@swu.edu.cn.



(a) Full view (b) *xoy* plane (c) *yoz* plane (d) *xoz* plane

Figure 1: Toy1

time that subspace clustering is embedded in the mean shift technique. The proposed method has the ability to deal with high-dimensional and large scale data.

- We provide the necessary theoretical analysis.
- Extensive experiments on both simulated and real-world data demonstrate the effectiveness of the proposed method.

The rest of this paper is organized as follows. In Section 2, we review related work on adaptive mean shift and subspace clustering. In Section 3, we introduce our methodology. Section 4 presents the empirical evaluation and analysis. Conclusions and future work are provided in Section 5.

2 Related Work.

Traditional mean shift methods [7, 8, 13, 15] estimate the density of the data and compute the *mean shift vector* in the full feature space. As a consequence, high-dimensional spaces present a challenge to the mean shift procedure due to the *curse of dimensionality*. In general, the presence of noisy features can hinder the density estimation process.

To address the issue posed by high-dimensional data, and by the presence of noisy features in general, subspace clustering techniques have been widely studied [19, 20, 24]. Subspace clustering methods assume that each cluster is embedded in a subspace. Two major search strategies have been developed to explore such subspaces: top-down and bottom-up [24].

Top-down algorithms first find an initial clustering in the full feature space, and then iteratively improve the clustering results by evaluating the subspaces of each cluster [24]. Representative methods are PROCLUS [1], ORCLUS [2], FINDIT [27], and δ-Clusters [28]. Bottom-up approaches initially find dense regions in a low dimensional feature space. Such regions are then combined to form clusters in spaces with higher dimensionality. Examples of methods that belong to this category are CLIQUE [3], ENCLUS [6], MAFIA [16], CBF [5], CLTree [22], and DOC [25].

Typically, in subspace clustering a subspace is defined as a collection of features, and each feature of a subspace participates with equal strength. In contrast, some approaches estimate the relevance of features, and assign weights to features that reflect their importance. The resulting techniques are often called *soft subspace clustering*. LAC [11] and COSA [14] are two representative algorithms of this kind.

LAC is a k-means-like algorithm. It iteratively adjusts weights assigned to features within each cluster, and generates new clusters until convergence. Unlike LAC, COSA specifies a weight vector for each point, rather than for each cluster. COSA iteratively explores the neighborhood of a point to estimate the local relevance of each feature. Such relevance is eventually embedded in a pairwise weighted distance measure, used in combination with hierarchical clustering [18].

Recently, Halite [9], a fast and scalable subspace clustering algorithm has been proposed. It has been shown that Halite achieves accuracy values which are competitive against state-of-the-art techniques. Halite's strength is its linearity or quasi-linearity in time and space w.r.t. data size and dimensionality.

3 Methodology.

Let $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ denote the data set, where n is the number of points and d is the dimensionality of each point $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})^T, i = 1, 2, \ldots, n$. A clustering $C = \{C_1, C_2, ..., C_{k^*}\}$ partitions $\mathcal{X}$ into k^* disjoint clusters, i.e., $C_i \cap C_j = \emptyset$ $(\forall i \neq j,\ i, j = 1, 2, \ldots, k^*)$, and $\cup_{k=1}^{k^*} C_k = \mathcal{X}$. In the following, we first give a brief review of Adaptive Mean Shift [8].

3.1 Adaptive Mean Shift (AMS).

To perform kernel density estimation, AMS sets a different bandwidth $h_i = h(\mathbf{x}_i)$[1] for each data point $\mathbf{x}_i$. The density estimator for $\mathbf{x}$ is then defined as

$$\hat{f}_K(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h_i^d} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_i}\right) \tag{3.1}$$

[1] h_i can be set to the distance between $\mathbf{x}_i$ and its k-th nearest neighbor [12].

where K is a spherically symmetric kernel. In this paper, we use a Gaussian kernel. The *profile* of a kernel K is defined as a function $\kappa : [0, +\infty) \to \mathbb{R}$ such that $K(\mathbf{x}) = \kappa(\|\mathbf{x}\|^2)$. Then, the sample point estimator (3.1) becomes

$$\hat{f}_K(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i^d}\kappa\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right) \tag{3.2}$$

By taking the gradient of $\hat{f}_K(\mathbf{x})$ we obtain

$$\nabla\hat{f}_K(\mathbf{x}) = \frac{2}{n}\left[\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right)\right]\times \underbrace{\left[\frac{\sum_{i=1}^{n}\frac{\mathbf{x}_i}{h_i^{d+2}}g(\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\|^2)}{\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}g(\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\|^2)} - \mathbf{x}\right]}_{\textit{mean shift vector}} \tag{3.3}$$

where $g(x) = -\kappa'(x)$, provided that the derivative of κ exists. The first part of Eq. (3.3) is a constant, and the factor in bracket is the *mean shift vector*, which always points towards the direction of the greatest increase in density. Using the mean shift vector, a sequence of estimation points $\{\mathbf{y}_t\}_{t=1,2,\ldots}$ is computed

$$\mathbf{y}_{t+1} = \frac{\sum_{i=1}^{n}\frac{\mathbf{x}_i}{h_i^{d+2}}g(\|\frac{\mathbf{y}_t-\mathbf{x}_i}{h_i}\|^2)}{\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}g(\|\frac{\mathbf{y}_t-\mathbf{x}_i}{h_i}\|^2)} \tag{3.4}$$

$\mathbf{y}_1$ is chosen as one of the points $\mathbf{x}_i$. The point that $\{\mathbf{y}_t\}_{t=1,2,\ldots}$ converges to is considered as the mode of $\mathbf{y}_1$. Points with the same mode are grouped in the same cluster.

3.2 Weighted Bandwidths. Eq. (3.4) shows that $\mathbf{y}_{t+1}$ is computed using the Euclidean distance between $\mathbf{y}_t$ and each point in $\mathcal{X}$, provided that $g(x)$ and all h_is are fixed. Points close to $\mathbf{y}_t$ have a large influence in the computation of $\mathbf{y}_{t+1}$, while points far from $\mathbf{y}_t$ play a smaller role. But in real data, the Euclidean distance is strongly affected by noisy or irrelevant features, and it may become meaningless in high-dimensional spaces due to the *curse of dimensionality*.

To address this issue, we first estimate the relevance of features for each $\mathbf{x}_i$, and then compute the distance of $\mathbf{x}_i$ to $\mathbf{y}_t$ in the resulting subspace of relevant features. Specifically, we represent the relevance of features for a point $\mathbf{x}_i \in \mathcal{X}$ as a nonnegative weight vector $\mathbf{w}_i = (w_{i1}, w_{i2}, \ldots, w_{id})^T$, where $\sum_{l=1}^{d} w_{il} = 1$. The entry w_{il} denotes the relevance of the l-th feature for $\mathbf{x}_i$. The larger the value w_{il} is, the more relevant the l-th feature is for point $\mathbf{x}_i$. Hence, $\mathbf{w}_i$ defines a *soft subspace* to which point $\mathbf{x}_i$ belongs. Using $\mathbf{w}_i$, each feature participates with the corresponding relevance in the computation of the distance of $\mathbf{x}_i$ to a point $\mathbf{x} \in \mathbb{R}^d$:

$$D_{\mathbf{w}_i}(\mathbf{x}_i, \mathbf{x}) = \sum_{l=1}^{d} w_{il}|x_{il} - x_l|/s_l \tag{3.5}$$

where s_l is the average l-th attribute distance of all pairs of points in $\mathcal{X}$; it is a measurement of the "closeness" on this attribute[2]:

$$s_l = \frac{1}{\binom{n}{2}}\sum_{i<j}|x_{il} - x_{jl}| \tag{3.6}$$

Note that, given two points $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, the two distances $D_{ij} = D_{\mathbf{w}_i}(\mathbf{x}_i, \mathbf{x}_j)$ and $D_{ji} = D_{\mathbf{w}_j}(\mathbf{x}_j, \mathbf{x}_i)$ are different, since in general $\mathbf{w}_i \neq \mathbf{w}_j$. D_{ij} represents the distance of $\mathbf{x}_i$ to $\mathbf{x}_j$, within the subspace of $\mathbf{x}_i$, whereas D_{ji} is the distance of $\mathbf{x}_j$ to $\mathbf{x}_i$, within the subspace of $\mathbf{x}_j$. Given $\mathbf{w}_i$, the k-neighborhood of $\mathbf{x}_i$ is defined as

$$S_i = \{\mathbf{x}_j \in \mathcal{X} | D_{ij} \leq D_{i(k)}, j \neq i\} \tag{3.7}$$

where $D_{i(k)}$ is the k-th smallest value of the weighted distances $\{D_{ij}\}_{j=1}^{n}$. The optimal weight vector $\mathbf{w}_i$ is the one that minimizes the following error function:

$$\begin{aligned} E_1(\mathbf{w}_i) &= \frac{1}{k}\sum_{\mathbf{x}_j \in S_i} D_{ij} = \frac{1}{k}\sum_{\mathbf{x}_j \in S_i}\sum_{l=1}^{d} w_{il}|x_{il} - x_{jl}|/s_l \\ &= \sum_{l=1}^{d} w_{il}\left(\frac{1}{k}\sum_{\mathbf{x}_j \in S_i}|x_{il} - x_{jl}|/s_l\right) = \sum_{l=1}^{d} w_{il}G_l \end{aligned} \tag{3.8}$$

subject to $\sum_{l=1}^{d} w_{il} = 1$, where $G_l = \frac{1}{k}\sum_{\mathbf{x}_j \in S_i}|x_{il} - x_{jl}|/s_l$. $E_1(\mathbf{w}_i)$ measures the dispersion of the k-neighborhood of $\mathbf{x}_i$. We seek to find a suitable $\mathbf{w}_i$ that minimizes such dispersion.

By minimizing E_1, the optimal $\mathbf{w}_i$ vector will assign weight one to the attribute with the smallest G_l value, and weight zero to all the other attributes. To avoid this trivial solution, we add the negative entropy of the weight distribution for $\mathbf{x}_i$ in Eq. (3.8), and minimize

$$\begin{aligned} E_2(\mathbf{w}_i) &= E_1(\mathbf{w}_i) + \alpha\sum_{l=1}^{d}(w_{il}\log w_{il}) \\ &= \sum_{l=1}^{d}(w_{il}G_l + \alpha w_{il}\log w_{il}) \end{aligned} \tag{3.9}$$

subject to $\sum_{l=1}^{d} w_{il} = 1$. $\alpha \geq 0$ is a parameter of the procedure. We can solve this constrained optimization problem by introducing the Lagrange multiplier λ and minimizing the resulting error function:

$$E(\mathbf{w}_i, \lambda) = \sum_{l=1}^{d}(w_{il}G_l + \alpha w_{il}\log w_{il}) + \lambda(1 - \sum_{l=1}^{d} w_{il}) \tag{3.10}$$

By setting the partial derivatives of $E(\mathbf{w}_i, \lambda)$ w.r.t. w_{il} $(l = 1, 2, \ldots, d)$ and λ to zero, we obtain

[2] $s_l = 0$ if and only if the values of all data points on the l-th attribute are the same. In this case, this attribute provides no discriminative information and should be deleted in a preprocessing step. Then, $s_l > 0$ holds and Eq. (3.5) is always meaningful. In addition, s_l is a fixed value for each data set $\mathcal{X}$.

$$\frac{\partial E}{\partial \lambda} = 1 - \sum_{l'=1}^{d} w_{il'} = 0 \tag{3.11}$$

$$\frac{\partial E}{\partial w_{il}} = G_l + \alpha \log w_{il} + \alpha - \lambda = 0 \tag{3.12}$$

From Eq. (3.12):

$$w_{il} = \exp\left(\frac{-G_l + \lambda - \alpha}{\alpha}\right) = \frac{\exp\left(-\frac{G_l}{\alpha}\right)}{\exp(1 - \frac{\lambda}{\alpha})} \tag{3.13}$$

Combing Eqs. (3.11) and (3.13):

$$1 - \frac{1}{\exp(1 - \frac{\lambda}{\alpha})} \sum_{l'=1}^{d} \exp\left(-\frac{G_{l'}}{\alpha}\right) = 0 \tag{3.14}$$

It follows:

$$\exp(1 - \frac{\lambda}{\alpha}) = \sum_{l'=1}^{d} \exp\left(-\frac{G_{l'}}{\alpha}\right) \tag{3.15}$$

Substituting this expression in Eq. (3.13), we obtain

$$w_{il} = \frac{\exp(-\frac{G_l}{\alpha})}{\sum_{l'=1}^{d} \exp(-\frac{G_{l'}}{\alpha})} \tag{3.16}$$

Eq. (3.16) computes the optimal $\mathbf{w}_i$ for a fixed S_i. Intuitively, larger weights are assigned to features with a smaller G_l value, which signifies a smaller dispersion along the l-th attribute within the k-neighborhood of $\mathbf{x}_i$. Once $\mathbf{w}_i$ is updated, a new S_i can be obtained using Eq. (3.7). As such, we can progressively improve the feature weight distribution and the neighborhood for each point. At convergence (proved below), we have the optimal $\mathbf{w}_i$ and S_i for each $\mathbf{x}_i$. We then set the bandwidth h_i for $\mathbf{x}_i$ to be equal to $D_{i(k)}$. The algorithm is summarized in Algorithm 1.

Algorithm 1 Weighted Bandwidth Algorithm

Input: $\mathcal{X}$, k, α, *max_iter*.
Output: h_i and $\mathbf{w}_i$, $i = 1, 2, \ldots, n$.

```
for i : 1 → n do
    t ← 0
    w_i ← (1/d, 1/d, ..., 1/d)
    repeat
        Compute D_{i(k)} and S_i using Eq. (3.7)
        Update w_i using Eq. (3.16)
        t ← t + 1
    until convergence or t = max_iter
    h_i ← D_{i(k)}
end for
return h_i and w_i, i = 1, 2, ..., n.
```

THEOREM 3.1. *Algorithm 1 converges.*

Proof. Suppose $\mathbf{w}_i^{(t)}$ is the weight vector for $\mathbf{x}_i$ at the t-th iteration. The corresponding $S_i^{(t)}$ is obtained using Eq. (3.7). Let $E_2(\mathbf{w}_i^{(t)}, S_i^{(t)})$ denote $E_2(\mathbf{w}_i^{(t)})$, for a given $S_i^{(t)}$. Then the following inequality holds:

$$E_2(\mathbf{w}_i^{(t+1)}, S_i^{(t)}) \leq E_2(\mathbf{w}_i^{(t)}, S_i^{(t)}) \tag{3.17}$$

because $\mathbf{w}_i^{(t+1)}$ is the optimal solution to problem (3.9) when $S_i^{(t)}$ is fixed. Once we have $\mathbf{w}_i^{(t+1)}$, $S_i^{(t+1)}$ can be obtained using Eq. (3.7). It is easy to see from Eq. (3.8) that

$$E_1(\mathbf{w}_i^{(t+1)}, S_i^{(t+1)}) \leq E_1(\mathbf{w}_i^{(t+1)}, S_i^{(t)}) \tag{3.18}$$

because $S_i^{(t+1)}$ consists of the k closest nearest neighbors of $\mathbf{x}_i$ according to $\mathbf{w}_i^{(t+1)}$. Since $E_2(\mathbf{w}_i, S_i) = E_1(\mathbf{w}_i, S_i) + \sum_{l=1}^{d}(\alpha w_{il} \log w_{il})$, and the latter term is a constant when $\mathbf{w}_i$ is fixed, it follows from Eq. (3.18)

$$E_2(\mathbf{w}_i^{(t+1)}, S_i^{(t+1)}) \leq E_2(\mathbf{w}_i^{(t+1)}, S_i^{(t)}) \tag{3.19}$$

From (3.17) and (3.19), we obtain

$$E_2(\mathbf{w}_i^{(t+1)}, S_i^{(t+1)}) \leq E_2(\mathbf{w}_i^{(t)}, S_i^{(t)}) \tag{3.20}$$

That is, $E_2(\mathbf{w}_i^{(t+1)}) \leq E_2(\mathbf{w}_i^{(t)})$. In addition, $E_1(\mathbf{w}_i) \geq 0$ and $\sum_{l=1}^{d}(w_{il} \log w_{il})$ achieves the minimum when equal weights are assigned to all features. Thus, $E_2(\mathbf{w}_i)$ is lower bounded. As a consequence, the sequence of $E_2(\mathbf{w}_i)$ values is monotonically decreasing and converges to a local minimum. □

3.3 Weighted Adaptive Mean Shift (WAMS).

Algorithm 1 provides the bandwidth h_i and the weight vector $\mathbf{w}_i$ for each point $\mathbf{x}_i$. Using $\mathbf{w}_i$, we can compute the weighted distance $D_{\mathbf{w}_i}(\mathbf{x}_i, \mathbf{y}_t)$ between $\mathbf{x}_i$ and $\mathbf{y}_t$, which replaces the Euclidean distance in Eq. (3.4). This gives the following sequence of estimation points $\{\mathbf{y}_t\}_{t=1,2,\ldots}$

$$\mathbf{y}_{t+1} = \frac{\sum_{i=1}^{n} \frac{\mathbf{x}_i}{h_i^{d+2}} g((\frac{D_{\mathbf{w}_i}(\mathbf{x}_i, \mathbf{y}_t)}{h_i})^2)}{\sum_{i=1}^{n} \frac{1}{h_i^{d+2}} g((\frac{D_{\mathbf{w}_i}(\mathbf{x}_i, \mathbf{y}_t)}{h_i})^2)} \tag{3.21}$$

The corresponding mean shift vector is

$$\mathbf{m}(\mathbf{y}_t) = \frac{\sum_{i=1}^{n} \frac{\mathbf{x}_i}{h_i^{d+2}} g((\frac{D_{\mathbf{w}_i}(\mathbf{x}_i, \mathbf{y}_t)}{h_i})^2)}{\sum_{i=1}^{n} \frac{1}{h_i^{d+2}} g((\frac{D_{\mathbf{w}_i}(\mathbf{x}_i, \mathbf{y}_t)}{h_i})^2)} - \mathbf{y}_t \tag{3.22}$$

We call $\mathbf{m}(\mathbf{y}_t)$ the *weighted mean shift vector*. If an estimation point $\mathbf{y}_t$ is located in a sparse area of the full feature space, the density around $\mathbf{y}_t$ is low, especially in high-dimensional data. This will compromise the finding of the proper mode for $\mathbf{y}_t$, when Eq. (3.4) is used. However, there may be points from the same cluster in a certain subspace surrounding $\mathbf{y}_t$. Hence, in this subspace, the area around $\mathbf{y}_t$ is denser. Thus $\mathbf{y}_t$ can converge to a mode in the corresponding subspace, and the negative influence of irrelevant or noisy features is reduced. The weighted distance in Eq. (3.21) serves this goal. We compute $\mathbf{m}(\mathbf{y}_t)$ using the weighted distances to the points $\mathbf{x}_i$, each according to a given subspace. Thus, $\mathbf{m}(\mathbf{y}_t)$ points towards the denser area in a certain subspace. The points that converge to the

same mode are grouped in the same cluster. Algorithm 2 summarizes the WAMS algorithm. It has three phases: Phase A is the mean shift process and provides a mode for each point. Phase B groups together points with the same mode, and gives the clustering result $C = \{C_1, C_2, \ldots, C_{k^*}\}$, where k^* is the number of detected clusters. Phase C computes the average of the weight vectors of points in the same cluster. This average vector represents the *soft subspace* where each cluster exists.

Algorithm 2 WAMS

Input: $\mathcal{X}$, h_i, $\mathbf{w}_i$ $(i = 1, 2, \ldots, n)$, max_iter.
Output: clustering C, modes M, cluster weights W.

Phase A. Mean shift process
for $i : 1 \to n$ **do**
  $t \leftarrow 0$
  $\mathbf{y}_0 \leftarrow \mathbf{x}_i$
  repeat
    $\mathbf{y}_{t+1} \leftarrow \frac{\sum_{i=1}^{n} \frac{\mathbf{x}_i}{h_i^{d+2}} g((\frac{D_{\mathbf{w}_i}(\mathbf{x}_i, \mathbf{y}_t)}{h_i})^2)}{\sum_{i=1}^{n} \frac{1}{h_i^{d+2}} g((\frac{D_{\mathbf{w}_i}(\mathbf{x}_i, \mathbf{y}_t)}{h_i})^2)}$
    $t \leftarrow t + 1$
  until convergence or $t = max\_iter$
  mode$(i) \leftarrow \mathbf{y}_t$ //The mode of $\mathbf{x}_i$
end for
Phase B. Clustering
$C \leftarrow \emptyset$ //Clustering result
$k^* \leftarrow 0$ //Number of clusters
for $i : 1 \to n$ **do**
  $exist \leftarrow False$
  for $j : 1 \to k^*$ **do**
    if (**mode**$(i) ==$ **mode**(j)) **then**
      $exist \leftarrow True$ //**mode**(i) already exists
      $C_j = C_j \cup \mathbf{x}_i$
      break
    end if
  end for
  if $(exist == False)$ **then**
    $k^* \leftarrow k^* + 1$ //A new cluster
    $C_{k^*} \leftarrow \{\mathbf{x}_i\}$
    $M(k^*) \leftarrow$ **mode**(i)
  end if
end for
Phase C. Subspace exploration
for $j : 1 \to k^*$ **do**
  $W(j) \leftarrow \frac{\sum_{\mathbf{x}_i \in C_j} \mathbf{w}_i}{|C_j|}$ //Cluster j weight vector
end for
return C, M, W.

3.4 Fast WAMS. The computational complexity of both Algorithms 1 and 2 is quadratic in the number of points n. It is time consuming to perform the mean shift on every point when dealing with large scale data. To speedup the computation we apply a similar idea as in [13]. When n is large, we can randomly select a subset $\mathcal{X}' = \{\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_m\}$ from $\mathcal{X}$, with $m \ll n$. Algorithms 1 and 2 are applied to $\mathcal{X}'$, and provide s'_l $(l = 1, \ldots, d)$, the clustering $C' = (C'_1, C'_2, \ldots, C'_{k*})$ and corresponding weight vectors $\mathbf{w}'_1, \mathbf{w}'_2, \ldots, \mathbf{w}'_m$ for the data points. Then, for each point $\mathbf{x} \in \mathcal{X} \setminus \mathcal{X}'$, we select the point $\mathbf{x}'_i \in \mathcal{X}'$ which minimizes the following weighted distance

$$D_{\mathbf{w}'_i}(\mathbf{x}'_i, \mathbf{x}) = \sum_{l=1}^{d} w'_{il} |x'_{il} - x_l| / s'_l \tag{3.23}$$

That is, we find the closest $\mathbf{x}'_i$ to $\mathbf{x}$ in a certain subspace, thus avoiding computing distances in the full space. We add $\mathbf{x}$ to the cluster which $\mathbf{x}'_i$ belongs to.

4 Empirical Evaluation.

4.1 Data. We conducted experiments on three simulated data sets and ten real data sets to evaluate the performance of the proposed methods. The details of the data used are shown in Table 1.

Table 1: Data used in our experiments

Data	#points	#features	#classes
Toy1	450	3	3
Toy2	300	10	2
Toy3	300	50	2
Iris	150	4	3
COIL	360	1024	5
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

Toy example 1 (Toy1) is shown in Fig. 1. All three classes were generated according to multivariate Gaussian distributions. Each class consists of 150 points. Class 1 (red) was generated on the xoy plane, with mean vector and the covariance matrix equal to $(0, 0)$ and [0.5 0; 0 5], respectively. The values on the z-axis are random values in the range $[0, 80]$. Class 2 (blue) was generated on the yoz plane. The mean vector and the covariance matrix are $(18, 25)$ and [0.5 0; 0 5], respectively. The x-axis values are random values in the range $[-15, 65]$. Class 3 (green) was generated on the xoz plane. The corresponding mean vector and the covariance matrix are $(13, 10)$ and [0.5 0; 0 5], respectively. The y-axis values are random values in the range $[-10, 70]$.

Both toy examples 2 and 3 (Toy2 and Toy3) consist of 2 classes, each made of 150 points. The dimensionalities of Toy2 and Toy3 are 10 and 50, respectively. We first generated two classes using Gaussian distributions on the xoy plane (see Fig. 2). The mean vector and the covariance matrix of class 1 (red) are $(5, 10)$ and [0.5 0; 0 10], while those of class 2 (blue) are $(25, 10)$ and [10 0; 0 0.5]. For Toy2, the remaining 8 features are random values in the range [0,1]. The remaining 48 features of Toy3 are random values in the range [0,1].

COIL contains 100 classes and we selected the first 5 classes, each containing 72 images. USPS is a handwritten digit database and the 2007 test images

Table 2: Comparison against AMS on simulated data

		RI				ARI				NMI			
Data	$k=$	30	50	70	90	30	50	70	90	30	50	70	90
Toy1	WAMS	**0.9469**	**1.0000**	**1.0000**	**1.0000**	**0.8751**	**1.0000**	**1.0000**	**1.0000**	**0.9116**	**1.0000**	**1.0000**	**1.0000**
	AMS	0.7590	0.5990	0.3318	0.3318	0.3622	0.2439	0.0000	0.0000	0.6041	0.4251	0.0000	0.0000
Toy2	WAMS	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	AMS	0.4983	0.4983	0.4983	0.4983	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Toy3	WAMS	**0.9933**	**0.9671**	**1.0000**	**0.9671**	**0.9867**	**0.9342**	**1.0000**	**0.9342**	**0.9711**	**0.8941**	**1.0000**	**0.8941**
	AMS	0.4983	0.4983	0.4983	0.4983	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

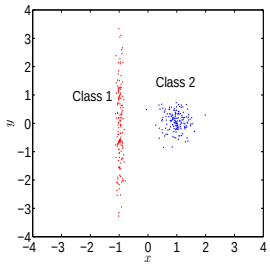


Figure 2: The xoy plane of Toy2 and Toy3

were chosen for our experiments[3]. The other eight data sets are from the UCI repository[4]. 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 data set, features were normalized to have zero mean value and unit variance.

4.2 Evaluation Measures. Various metrics to assess clustering results exist [4, 21]. We use Rand Index (RI) [17], Adjusted Rand Index (ARI) [17], and Normalized Mutual Information (NMI) [26] as clustering validity indices since the labels of data are known. The label information is only used to evaluate the clustering results, and is not utilized during the clustering process. Both RI and NMI range from 0 to 1, while ARI yields a value between -1 and +1. A larger value of RI/ARI/NMI indicates a better clustering result.

4.3 Experimental settings. The maximum number of iterations max_iter is set to 200 for both Algorithms 1 and 2. We need to specify two additional parameters for Algorithm 1: the number of neighbors k and the parameter $\alpha > 0$. α can affect the weight distribution. When $\alpha \to 0$, Eq. (3.16) tends to assign weight 1 to the feature with the smallest G_l, and weight 0 to all other features. When $\alpha \to +\infty$, all features are assigned equal weights. In all our experiments $\alpha = 0.2$.

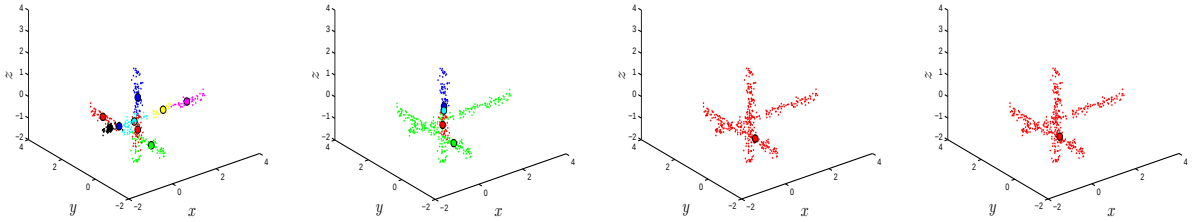
We compared WAMS with AMS [8]. AMS sets the bandwidth for a given point equal to the Euclidean distance of that point to its k-th neareast neighbor. In the experiments, we use the same value of k for both WAMS and AMS. We also performed comparisons against several other clustering algorithms, including three classic algorithms, i.e. k-means [23], EM (with a Gaussian mixture) [10], and Sing-l (single-linkage clustering) [18], and three subspace clustering algorithms, i.e. LAC [11], COSA [14], and the most recent Halite [9]. LAC has a parameter h (see [11] for details), similar to the α of WAMS. We set both to 0.2. COSA first outputs a weighted distance matrix, which is then used for single-linkage clustering. Halite has two versions, one for soft clustering and one for hard clustering. We used the version for hard clustering made available by the authors. k-means, EM, and LAC have a random component. The reported results for these techniques are the average of 100 independent runs. One-sample t-test and paired-samples t-test are used in our experiments to assess the statistical significance of the results. The significance level is 0.05. Note that for WAMS, AMS, and Halite we do not need to specify the number of clusters in advance; for all the other algorithms we set the number of clusters equal to the number of classes.

4.4 Results and Analysis.

4.4.1 Results on Simulated Data. To illustrate the effectiveness of WAMS, we first compared it against AMS [8] on the three toy data sets. We set $k = 30, 50, 70, 90$. Figs. 3 and 4 show the results. As expected, when k increases, fewer modes are found by AMS. When $k \geq 70$, all points converge to the same mode and are grouped in one cluster. This shows the sensitivity of AMS to the choice of k (i.e. bandwidth). A small bandwidth might cause the finding of too many modes (or clusters), while a large one might cause the merging of distinct clusters. AMS finds four modes at $k = 50$. However, the result is still poor because of the noisy features. In contrast, WAMS is robust with respect to k. WAMS found four clusters when $k = 30$,

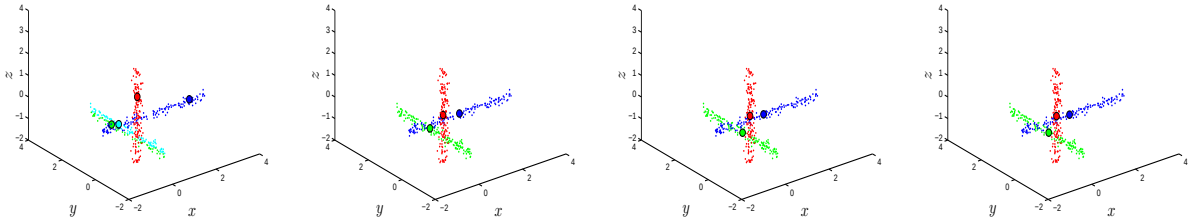
[3] COIL and USPS can be downloaded at `www.cad.zju.edu.cn/home/dengcai`

[4] `http://archive.ics.uci.edu/ml/index.html`



(a) k=30, 9 modes (b) k=50, 4 modes (c) k=70, 1 mode (d) k=90, 1 mode

Figure 3: Clustering results of AMS on Toy1



(a) k=30, 4 modes (b) k=50, 3 modes (c) k=70, 3 modes (d) k=90, 3 modes

Figure 4: Clustering results of WAMS on Toy1

and achieved perfect clustering for $k \geq 50$. This may be due to the fact that WAMS operates in subspaces associated to the points. The k nearest neighbors of a point are used to explore its subspace. The resulting weight vector is resilient to a wide range of k values.

Additional results are shown in Table 2, w.r.t. three clustering evaluations—RI, ARI and NMI. The larger value obtained in each case is highlighted in boldface. We can see that WAMS outperforms AMS by a large margin in each case. In particular, the performance of AMS on Toy2 and Toy3 is extremely poor, due to the noisy features. Note that the zero values of ARI or NMI indicate that all points are grouped in only one cluster. This happens because the data sparsity causes larger distances, and therefore larger bandwidth values. In contrast, WAMS achieves a very good performance on Toy2 and Toy3, despite the large number of noisy features injected in the data.

To compare WAMS against other clustering algorithms, we average the results of WAMS under the four values of k being tested. The results are given in Table 3. In each row, the significantly best value is highlighted in boldface. In general, a better RI value corresponds also to better ARI and NMI values. Hence, hereinafter, only RI values are reported due to space limit. COSA achieves a perfect score on Toy1, but it fails on Toy2 and Toy3, due to the noisy features. Additional experiments in Section 4.4.3 show that WAMS on Toy1 gives a perfect score for a wide range of k values. On Toy2 and Toy3, WAMS significantly outperforms all the competitive methods.

Table 3: Comparison against clustering algorithms on simulated data (RI)

Data	WAMS	Subspace clustering			k-means	EM	Sing-l
		LAC	COSA	Halite			
Toy1	0.9867	0.6486	**1.0000**	0.3318	0.6467	0.9604	0.5729
Toy2	**1.0000**	0.6416	0.4984	0.4983	0.6626	0.9407	0.4984
Toy3	**0.9819**	0.5055	0.4984	0.4983	0.5039	0.5049	0.4984

4.4.2 Results on Real Data. For each data set, we test the following values of k: $k_1 = 0.6 \times \sqrt{n}$, $k_2 = \sqrt{n}$, $k_3 = 2 \times \sqrt{n}$, and $k_4 = 3 \times \sqrt{n}$, where n is the number of points. Table 4 shows the results comparing WAMS and AMS. On Image and Pen, AMS gives the best result when $k = k_1$, but its performance quickly deteriorates as k increases. A similar trend is observed for Steel, CTG, Letter, and Wave data sets. This shows again the sensitivity of AMS on the value of k. In particular, on COIL, Yeast and USPS, AMS fails to output a reasonable clustering result for all the k values. In contrast, WAMS has a stable behavior and outperforms AMS in most cases.

Table 5 shows the results against the other clustering methods. For WAMS, we report the average RI of the results obtained with the four values of k being tested. In each row, we highlight the statistically significant best results. It is interesting to see that WAMS always achieves the best performance, except on Iris and COIL. It turns out that when $k = k_1$ and $k = k_2$, WAMS still performs very well on both Iris and COIL. For Iris, $k_3 = 24$ and $k_4 = 37$, and each of the three classes contains only 50 points. For COIL, $k_3 = 38$ and

Table 4: Comparison against AMS on real data (RI)

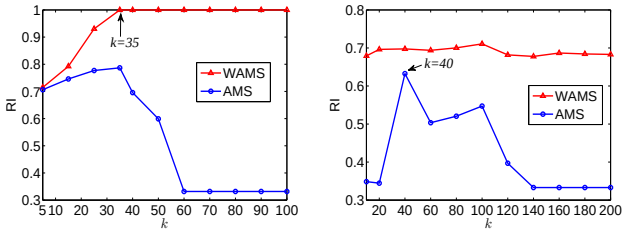
Data	$k =$	k_1	k_2	k_3	k_4
Iris	WAMS	**0.8440**	**0.8275**	0.7763	0.7763
	AMS	0.8030	0.7247	0.7763	0.7763
COIL	WAMS	**0.7997**	**0.7760**	**0.6629**	**0.6777**
	AMS	0.1978	0.1978	0.1978	0.1978
Yeast	WAMS	**0.6347**	**0.6014**	**0.5983**	**0.6050**
	AMS	0.3543	0.3543	0.3543	0.3543
Steel	WAMS	**0.7506**	**0.7315**	**0.7499**	**0.7418**
	AMS	0.6431	0.5376	0.4821	0.2217
USPS	WAMS	**0.8990**	**0.9030**	**0.9029**	**0.8975**
	AMS	0.1089	0.1089	0.1089	0.1089
CTG	WAMS	**0.8114**	**0.8034**	**0.8078**	**0.7959**
	AMS	0.4894	0.4732	0.1602	0.1602
Letter	WAMS	**0.6913**	**0.6959**	**0.7007**	**0.6753**
	AMS	0.6757	0.6218	0.5457	0.3331
Image	WAMS	0.8811	**0.8927**	**0.8962**	**0.8580**
	AMS	**0.8860**	0.8785	0.7734	0.5914
Pen	WAMS	0.7114	**0.7107**	**0.7384**	**0.7188**
	AMS	**0.7301**	0.6476	0.6457	0.4224
Wave	WAMS	**0.6862**	**0.6440**	**0.6790**	**0.6689**
	AMS	0.6425	0.6105	0.3333	0.3333

Table 5: Comparison against clustering algorithms on real data (RI)

Data	WAMS	Subspace clustering			k-means	EM	Sing-l
		LAC	COSA	Halite			
Iris	0.8060	0.8076	0.7764	0.3535	0.8051	**0.8193**	0.7771
COIL	0.7291	**0.7980**	0.6825	0.1978	**0.7979**	0.7529	0.2179
Yeast	**0.6098**	0.5706	0.3557	0.3620	0.5624	0.3713	0.3602
Steel	**0.7435**	**0.7413**	0.2242	0.2217	0.7394	0.7168	0.2446
USPS	**0.9006**	0.8750	0.1162	0.1089	0.8720	0.8225	0.1274
CTG	**0.8046**	0.8006	0.1697	0.5307	0.7996	0.7487	0.2007
Letter	**0.6908**	0.6074	0.3336	0.5147	0.6029	0.6074	0.3336
Image	**0.8820**	0.8544	0.3676	0.1812	0.8399	0.8028	0.1531
Pen	**0.7198**	0.7143	0.3385	0.6020	0.7068	0.7143	0.6852
Wave	**0.6695**	0.6675	0.3335	0.3358	0.6674	0.6675	0.6467

$k_4 = 57$, while each class only has 72 points. In both cases k_3 and k_4 are large w.r.t. the class sizes, resulting in lower RI values. As shown in Table 5, LAC is the second best performer. A drawback of LAC is that it needs the number of clusters in input, while WAMS does not. Halite is faster than WAMS, but we found that its accuracy is quite poor across all data.

4.4.3 Sensitivity Analysis of Parameter k. We tested the sensitivity of WAMS w.r.t. k on Toy1 and Letter. The tested ranges are $[5, 100]$ and $[10, 200]$, respectively. Fig. 5 gives the results. On Toy1, the performance of both AMS and WAMS increases for larger k values, and reaches the peak at $k = 35$. Then, the performance of AMS drops sharply, and reaches the minimum when $k \geq 60$, which indicates that AMS groups all points in one cluster. In contrast, WAMS is stable for $k > 35$. Similarly, for Letter the performance of AMS deteriorates when $k > 40$, while WAMS is stable throughout. This confirms the robustness of WAMS w.r.t. the parameter k.



(a) Toy1 (b) Letter

Figure 5: Sensitivity analysis of parameter k (RI)

Table 6: Evaluation of F-WAMS (RI)

Data	WAMS	F-WAMS			
		40%	20%	10%	5%
Letter	0.6959	0.6944	0.6959	0.6852	0.6749
time(sec)	516.0	88.6	18.2	4.4	1.3
Image	0.8927	0.8932	0.8836	0.8607	0.8269
time(sec)	860.6	174.6	32.3	6.8	2.1
Pen	0.7107	0.7243	0.7223	0.7231	0.7248
time(sec)	1623.9	307.5	64.2	14.7	3.7
Wave	0.6440	0.6467	0.6454	0.6411	0.6429
time(sec)	9664.6	1190.9	259.4	65.5	18.7

4.4.4 Evaluation of F-WAMS. In this section, we evaluate the performance of F-WAMS on the four largest data sets: Letter, Image, Pen, and Wave. For each data set, we selected four random samples of sizes 40%, 20%, 10%, and 5% of the original data set (the balance between classes was preserved). We set $k = \sqrt{n}$ for WAMS and $k = \sqrt{m}$ for F-WAMS, where m is the number of points of the corresponding sample. For F-WAMS we report the average RI values of 20 independent runs. Table 6 shows the results. The reported time (seconds) of F-WAMS is also the average value of the 20 runs. Experiments were performed in Matlab R2010a on an Intel® Core™ i7-4700MQ processor with 8 GB RAM using Windows 8.

The running time of F-WAMS reduces sharply as the size of the data decreases. On Letter and Wave, F-WAMS achieves a similar performance as WAMS. F-WAMS performs slightly worse as the size of the sample decreases on Image. On Pen, F-WAMS outperforms WAMS. This is mainly because sampling can reduce the influence of noisy data. Overall, F-WAMS reduces the time complexity without sacrificing accuracy, showing good potential to deal with large scale data in real-life tasks.

5 Conclusion and Future Work.

We have introduced WAMS, a nonparametric clustering algorithm that explores the existence of clusters in subspaces. The effectiveness of the proposed methods is demonstrated through extensive experiments. In summary, we have shown that (1) WAMS reduces

the negative influence of noisy features, and can find meaningful clusters; (2) WAMS explores the subspace where each cluster exists; and (3) WAMS is robust to k (the neighborhood size). Like AMS, our approach can find clusters of irregular shapes, and does not require the a-priori specification of the number of clusters. Finally, F-WAMS has shown good potential to scale nicely with large data. We also observe that the mean shift procedure can largely benefit from a parallel implementation, a direction we plan to pursue.

Acknowledgments

This paper is partially supported by grants from the Natural Science Foundation of China (61101234), Fundamental Research Funds for the Central Universities of China (XDJK2014C044 and XDJK2013C123), the Doctoral Fund of Southwest University (No. SWU113034), and the China Scholarship Council (CSC).

References

[1] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park. Fast algorithms for projected clustering. In *SIGMOD*, pages 61–72, 1999.

[2] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *SIGMOD*, pages 70–81, 2000.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD*, pages 94–105, 1998.

[4] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Prez, and I. Perona. An extensive comparative study of cluster validity indices. *PR*, 46(1):243–256, 2013.

[5] J. W. Chang and D. S. Jin. A new cell-based clustering method for large, high-dimensional data in data mining applications. In *Proceedings of the ACM Symposium on Applied Computing*, pages 503–507, 2002.

[6] C. H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *SIGKDD*, pages 84–93, 1999.

[7] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *TPAMI*, 24(5):603–619, 2002.

[8] D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *ICCV*, pages 438–445, 2001.

[9] R. L. Cordeiro, A. J. Traina, C. Faloutsos, and C. T. Jr. Halite: Fast and scalable multiresolution local-correlation clustering. *TKDE*, 25(2):387–401, 2013.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journals of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[11] C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan, and D. Papadopoulos. Locally adaptive metrics for clustering high dimensional data. *DMKD*, 14(1):63–97, February 2007.

[12] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, second edition, 2000.

[13] D. Freedman and P. Kisilev. Fast mean shift by compact density representation. In *CVPR*, pages 1818–1825, 2009.

[14] J. H. Friedman and J. J. Meulman. Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society*, 66:815–849, 2004.

[15] B. Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimensions: A texture classification example. In *ICCV*, pages 456–463, 2003.

[16] S. Goil, H. Nagesh, and A. Choudhary. MAFIA: Efficient and scalable subspace clustering for very large data sets. Technical report CPDC-TR-9906-010, Northwestern University, 2145 Sheridan Road, Evanston IL-60208, June 1999.

[17] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[18] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[19] H. P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM TKDD*, 3(1):1:1–1:58, 2009.

[20] H. P. Kriegel, P. Kröger, and A. Zimek. Subspace clustering. *WIREs DMKD*, 2(4):351–364, 2012.

[21] C. Legny, S. Juhsz, and A. Babos. Cluster validity measurement techniques. In *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, pages 388–393, 2006.

[22] B. Liu, Y. Xia, and P. S. Yu. Clustering through decision tree construction. In *Proceedings of the international conference on Information and Knowledge Management*, pages 20–29, 2000.

[23] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. University of California Press, 1967.

[24] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *ACM SIGKDD Explorations Newsletter*, 6(1):90–105, 2004.

[25] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *SIGMOD*, pages 418–427, 2002.

[26] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *JMLR*, 3:583–617, 2002.

[27] K. G. Woo, J. H. Lee, M. H. Kim, and Y. J. Lee. FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology*, 46(4):255–271, 2004.

[28] J. Yang, W. Wang, H. Wang, and P. S. Yu. δ-clusters: Capturing subspace correlation in a large data set. In *ICDE*, pages 517–528, 2002.