Improving K-means clustering algorithm with the intelligent water drops (IWD) algorithm

Hamed Shah-Hosseini

Faculty of Electrical and Computer Engineering,
Shahid Beheshti University,
G.C., Tehran, Iran
E-mail: h_shahhosseini@sbu.ac.ir
E-mail: tasom2002@yahoo.com

Abstract: In this paper, the K-means algorithm for data clustering is improved by the swarm-based nature-inspired optimisation algorithm, the intelligent water drops (IWD) algorithm. The K-means algorithm is an iterative algorithm in which the number of clusters is given in advance. Although the K-means is fast to converge, it is sensitive to the initial conditions. As a result, it is often trapped in local optimums. The IWD algorithm, which mimics the actions and reactions between natural water drops in real rivers, is modified to implicitly embed in itself the main processes of the K-means algorithm. The modified algorithm called IWD-KM is tested with several well-known datasets for clustering, and its performance is compared with the K-means algorithm. The experimental results show the superiority of the proposed IWD-KM algorithm.

Keywords: clustering; K-means; metaheuristic; intelligent water drops; IWD; chaos.


Biographical notes: Hamed Shah-Hosseini received his BS in Computer Engineering from the University of Tehran. He also obtained his MS and PhD both in Computer Science (AI) from the Amirkabir University of Technology, all with high honours. He is now with the Faculty of Electrical and Computer Engineering, Shahid Beheshti University, Tehran, Iran. His research interests include computational intelligence especially neural networks, evolutionary computation, swarm intelligence, and computer vision. He has introduced the Time-Adaptive Self-Organizing Map (TASOM) networks for both stationary and non-stationary environments. Moreover, he has introduced a bio-inspired swarm-based optimisation algorithm called ‘intelligent water drops’ algorithm (or IWD algorithm), which is based on actions and reactions occurring in natural rivers. In addition, he presented a Generalised Taylor’s (G-Taylor’s) theorem and a new metaheuristic called Galaxy-based Search Algorithm (GbSA) for continuous optimisation.
1 Introduction

Data clustering has widely been used in the field of computational intelligence for applications such as pattern recognition (Theodoridis and Koutroumbas, 2003), vector quantisation (Lia et al., 2002), web mining (Eirinaki and Vazirgiannis, 2003), and knowledge discovery (Fayyad et al., 1996).

Clustering algorithms may be grouped into two general categories: hierarchical and partitioning. Besides, there are other categories such as graph-based, fuzzy-based, or neural network-based clustering algorithms (see Xu and Wunsch, 2005 for a detailed review). The K-means algorithm, which is also called the generalised Lloyd algorithm (GLA), is a partitioning clustering algorithm. The number of clusters, $K$, is given to the K-means algorithm in advance, and the K-means partitions the data points into $K$ distinct clusters. Indeed, the K-means is first initialised with cluster centres randomly chosen from the dataset or randomly generated. Then, the cluster centres are updated in a way to reduce some objective measure. After a number of iterations, the cluster centres converge such that each cluster centre is a representative of the data points that belong to the cluster.

In spite of the fact that the K-means converges fast, it often converges to local optima (Khan and Ahmad, 2004). Moreover, the K-means is sensitive to initial conditions. This is due to the fact that the initial cluster centres are chosen randomly. Thus, each time the K-means runs, it may converge to cluster centres different from the previous run. Recently, metaheuristics have been used to improve the performance of the K-means.

Metaheuristics refer to general purpose algorithms applicable to different optimisation problems, which usually need few modifications for adaptation to different problems. Particle swarm optimisation (PSO) and genetic algorithms are among the well-known metaheuristic algorithms. Metaheuristics alone may be used for clustering (Fathian et al., 2007; Karaboga and Ozturk, 2011; Shelokar et al., 2004; Pacheco, 2005). Although they usually find optimal or near-optimal solutions, they perform slowly to converge since the search space for data clustering is huge, and the clustering is an NP-complete problem (Garey et al., 1982). As a result, it is favourable to combine faster clustering algorithms with metaheuristics to have both fast convergence and better optimal cluster centres.

A combination of K-means, Nelder-Mead simplex search, and PSO has been used as a hybrid method for clustering (Kao et al., 2008). Krishna and Murty (1999) introduced ‘genetic K-means algorithm’, which uses a modified mutation operator in the genetic algorithm plus the K-means, for clustering. Laszlo and Mukherjee (2007) used a novel crossover operator for selecting centres to seed the K-means method for clustering. Moreover, a quantum-inspired genetic algorithm has been introduced for the K-means (Xiao et al., 2010). In addition, metaheuristics have also been combined with other iterative clustering algorithms (Jiang et al., 2010; Güngör and Ünler, 2008).

In this paper, a newly introduced metaheuristic, called ‘intelligent water drops’ (IWD) algorithm is modified to improve the performance of the K-means algorithm. The IWD algorithm was first introduced by Shah-Hosseini (2007) for the travelling salesmen’s problem (TSP) (Msallam and Hamdan, 2011). It has also been used for other applications including the multi-dimensional knapsack problem (MKP) (Shah-Hosseini, 2008), n-queen puzzle (Shah-Hosseini, 2009b), trajectory planning (Duan et al., 2009), and multi-level thresholding of grey-level images (Shah-Hosseini, 2009a; 2012).
The rest of the paper is organised as follows: next section reviews the K-means clustering algorithm. Section 3 summarises the IWD algorithm. Section 4 introduces the proposed IWD for the K-means, named ‘IWD-KM’. Experimental results are included in Section 5. The final section, Section 6, contains the concluding remarks.

2 A review on K-means clustering

Consider we have a set of $M$ data points $X = \{x_1, x_2, \ldots, x_M\}$ such that each data point (also called object, pattern, or input vector) is a $d$-dimensional vector, $x_j \in \mathbb{R}^d$ where $x_j = [x_{j1}, x_{j2}, \ldots, x_{jd}]^T$. The goal is to partition the data points into $K$ distinct subsets $C = \{C_1, C_2, \ldots, C_K\}$ such that:

- $\forall i, C_i \neq \emptyset$
- $\bigcup_{i=1}^{K} C_i = X$
- $\forall i, j, C_i \cap C_j = \emptyset, i \neq j$

The above partitioning is obtained by minimising a criterion function. The sum of squared error function is widely used as a criterion function for partitional clustering.

The K-means falls into the category of partitional clustering algorithms. The K-means may be specified in the following steps:

1. Initialise cluster centres $M_k = \{m_1, m_2, \ldots, m_K\}$ by randomly sampling $K$ input data vectors and considering them as the cluster centres.
2. Assign each data vector $x_j$ to its nearest-cluster centre $m_{kj}$
   \[ k(j) = \arg\min_k (x_j - m_{kj}) \]  

3. Update the cluster centres $M_k$ as the mean vectors of their associated data vectors:
   \[ m_k = \frac{1}{|J_k|} \sum_{j \in J_k} x_j \quad \text{for} \quad k = 1, 2, \ldots, K \]  

   where $J_k = \{j \mid k(j) = k\}$.
4. Go to step 2 until convergence.

The K-means attempts to minimise the following objective function:

\[ f(X, M) = \sum_{j=1}^{N} \|x_j - m_{k(j)}\|^2 \]  

where $k(j)$ is defined in equation (1). It is mentioned that K-means converges fast but may lead to a local optimum. Therefore, the IWD algorithm is used here to reduce its drawbacks. Next section reviews on the IWD algorithm.
3 The IWD algorithm

The IWD algorithm tries to model the behaviour of natural water drops flowing in rivers in order to optimise an objective function of a given problem. The problem is represented by a graph $G(N, E)$ where $N$ denotes the nodes of the graph and $E$ denotes its edges. Memory in the IWD algorithm is distributed and it is the amount of soil on the edges of the graph $G$. Generally, a number of IWDs are placed randomly on the nodes of the graph and then they travel along the edges of the graph from one node to another until they complete their solutions to the problem. In fact, each IWD constructs a solution to the problem gradually through visiting nodes while modifying the amount of soil of the edges connecting the visited nodes. As time elapses, the edges of better solutions contain less soil than other edges of the graph. Moreover, at the end of each iteration of the algorithm, the total-best solution (TBS), which is the best solution found so far, is updated by the solutions obtained by all the IWDs. Again, newly created IWDs are distributed randomly on the graph beginning a new iteration of the algorithm and the whole process mentioned above is repeated.

The IWD algorithm has two types of parameters: Static parameters and Dynamic parameters. Static parameters remain constant during the process of the IWD algorithm. Dynamic parameters are reinitialised after each iteration of the IWD algorithm.

The IWD algorithm (Shah-Hosseini, 2009b) is specified in the following steps:

1. Initialise static parameters:
   - The problem is represented in a graph $G(N, E)$ with $N_i$ nodes.
   - The quality of the TBS $T^{TB}$ is initially set to the worst value: $q(T^{TB}) = -\infty$ (for maximisation problem).
   - The maximum number of iterations is specified by $\text{iter}_{\text{max}}$.
   - The iteration count $\text{iter}_{\text{count}}$ is initialised with zero.
   - The number of intelligent water drops or IWDs denoted by $N_{IWD}$ is set to a positive integer value.
   - Velocity updating parameters are set as $a_v = c_v = 1$ and $b_v = 0.01$.
   - Soil updating parameters are set as $a_s = c_s = 1$ and $b_s = 0.01$.
   - The local soil updating parameter is set as $\rho_n = 0.1$.
   - The global soil updating parameter is set as $\rho_{IWD} = 0.01$.
   - The initial soil on each edge $e_{ij} \in E$ of the graph connecting nodes $i$ and $j$ is set by $\text{soil}(e_{ij}) = \text{InitSoil}$ where $\text{InitSoil} = 100,000$.
   - The initial velocity of each IWD is set as $\text{InitVel} = 8$.

2. Initialise dynamic parameters:
   - For constraint handling, each IWD possesses a visited node list $V_i(IWD)$, which is initially set to empty: $V_i(IWD) = \{ \}$.
   - Each IWD’s velocity is set to $\text{InitVel}$.
   - All IWDs are set to have zero amount of soil.

3. Spread the IWDs on the nodes of the graph usually randomly.

4. Update the visited node list of each IWD to include the nodes just visited.
Improving K-means clustering algorithm with the IWD algorithm

5 Repeat steps 5.1 to 5.4 for those IWDs with partial solutions.

5.1 For the IWD currently at node \( i \), choose the next node \( j \) by probability \( p_{IWD}^{i,j} \):

\[
p_{IWD}^{i,j} = \frac{f\left(\text{soil}(e_{i,j})\right)}{\sum_{k \neq \text{IWD}(i)} f\left(\text{soil}(e_{i,k})\right)}
\]

such that

\[
f\left(\text{soil}(e_{i,j})\right) = \frac{1}{\varepsilon + g\left(\text{soil}(e_{i,j})\right)}
\]

and

\[
g\left(\text{soil}(e_{i,j})\right) = \begin{cases} 
\text{soil}(e_{i,j}) & \text{if } \min_{\text{IWD}(i,l)} \text{soil}(e_{i,l}) \geq 0 \\
\text{soil}(e_{i,j}) - \min_{\text{IWD}(i,l)} \text{soil}(e_{i,l}) & \text{else}
\end{cases}
\]

Then, add the newly visited node \( j \) to the list \( \text{ve}(\text{IWD}) \).

5.2 For each IWD moving from node \( i \) to node \( j \), update its velocity \( \text{vel}_{IWD}(t) \) by

\[
\text{vel}_{IWD}(t + 1) = \text{vel}_{IWD}(t) + \frac{a_r}{b_r + c_v \cdot \text{soil}^2(e_{i,j})}
\]

where \( \text{vel}_{IWD}(t + 1) \) is the updated velocity of the IWD.

5.3 For the IWD moving from node \( i \) to \( j \), compute the soil \( \Delta\text{soil}(e_{i,j}) \) that the IWD loads from the edge \( e_{i,j} \) by

\[
\Delta\text{soil}(e_{i,j}) = \frac{a_r}{b_r + c_v \cdot \text{time}^2(i, j; \text{vel}_{IWD}(t + 1))}
\]

such that

\[
\text{time}(i, j; \text{vel}_{IWD}(t + 1)) = \frac{\text{HUD}(j)}{\text{vel}_{IWD}(t + 1)}
\]

where the heuristic undesirability \( \text{HUD}(j) \) is defined appropriately for the given problem.

5.4 Update the soil \( \text{soil}(e_{i,j}) \) of the edge \( e_{i,j} \) traversed by the IWD, and also update the soil that the IWD carries \( \text{soil}_{IWD} \) by

\[
\text{soil}(e_{i,j}) = (1 + \rho_n) \cdot \text{soil}(e_{i,j}) - \rho_n \cdot \Delta\text{soil}(e_{i,j})
\]

\[
\text{soil}_{IWD} = \text{soil}_{IWD} + \Delta\text{soil}(e_{i,j})
\]

6 Find the iteration-best solution (IBS) \( T^I \) from all the solutions \( T^{IWD} \) obtained by the IWDs using
\( T^{IB} = \arg \max_{t \in T^{IB}} q(t^{IWD}) \) (10)

where function \( q(.) \) gives the quality of the solution.

7. Update the soils on the paths that form the current IBS \( T^{IB} \) by

\[
\text{soil}(e_{i,j}) = \left( 1 - \rho^{IWD} \right) \cdot \text{soil}(e_{i,j}) - \rho^{IWD} \cdot \frac{1}{(N^{IB} - 1)} \cdot \text{soil}^{IWD} \quad \forall e_{i,j} \in T^{IB} (11)
\]

where \( N^{IB} \) is the number of nodes in solution \( T^{IB} \).

8. Update the total best-solution \( T^{TB} \) using the current IBS \( T^{IB} \) by

\[
T^{TB} = \begin{cases} T^{TB} & \text{if } q(T^{TB}) \geq q(T^{IB}) \\ T^{IB} & \text{otherwise} \end{cases} (12)
\]

9. Increment the iteration number by

\[
\text{Iter}_{\text{count}} \leftarrow \text{Iter}_{\text{count}} + 1.
\]

Then, go to step 2 if

\[
\text{Iter}_{\text{count}} < \text{Iter}_{\text{max}}.
\]

10. The algorithm stops here with the TBS \( T^{TB} \).

The constraint handling throughout the IWD algorithm may be tackled in three ways:
One way is to strictly prevent IWDs from constructing solutions that violate the problem’s constraints. Another way is to let the IWDs construct solutions freely but penalise those solutions that violate constraints via reducing the quality value for such solutions. The third way to handle constraints is to let the solutions be constructed freely but they are modified to keep satisfied all the problem’s constraints.

In the next section, the IWD algorithm is modified to be used for clustering while using the K-means algorithm inside itself.

4 The proposed IWD-KM algorithm for clustering

The IWD algorithm is combined with the K-means to cluster input data vectors. The hybrid clustering algorithm IWD-KM is explained in the following subsections. The flowchart of the IWD-KM is shown in Figure 1. In the following, each component of the algorithm as shown in Figure 1 is introduced. It is noted that the K-means is implicitly hidden in the proposed algorithm and it is used in Subsection 4.4 and Subsection 4.5. Moreover, the termination condition used in the IWD-KM is the maximum number of iterations.

4.1 Problem representation

Consider we have \( M \) input data vectors \( X = \{x_1, x_2, \ldots x_M\} \) that is required to be partitioned into \( K \) clusters. Then, a graph with \( M + 1 \) nodes and \( K \times M \) directed edges is
created, which will be used ultimately by the IWDs to construct solutions for clustering. Figure 2 shows an example of the problem’s graph for clustering into three clusters.

**Figure 1** The flowchart of the proposed IWD-KM clustering algorithm

![Flowchart of the proposed IWD-KM clustering algorithm](image)

Every IWD begins its trip from node 1 and ends it by visiting node $M + 1$. A directed edge $e_{i, i+1}(k)$, which connects node $i$ to node $i + 1$, is devoted to the cluster $k$. Since we have $K$ clusters, then there will be $K$ directed edges $e_{i, i+1}(k)$ connecting $i$ to $i + 1$ such that $k = 1, 2, \ldots, K$. 

**Figure 2** The problem’s graph for partitioning $M$ data vectors into three clusters using the IWD-KM

![Graph for partitioning $M$ data vectors into three clusters using the IWD-KM](image)
There needs to be a selection mechanism for the IWD to select the edge that it will use to move to the next node. Next subsection explains this edge selection mechanism.

4.2 Edge selection

Let an IWD residing at node \( i \) and wants to select one of the edges \( e_{i,i+1}(k) \) to visit the next node \( i+1 \). Then, the probability \( P_{\text{IWD}}^{e_{i,i+1}(k)} \) for the IWD to select the edge \( e_{i,i+1}(k) \) is defined by

\[
P_{\text{IWD}}^{e_{i,i+1}(k)} = \frac{f\left(\text{soil}(e_{i,i+1}(k))\right)}{\sum_{l=1}^{K} f\left(\text{soil}(e_{i,i+1}(l))\right)}
\]

where \( f(.) \) has been defined in equation (5).

During visiting nodes and selecting edges, the IWD updates the soil carrying by itself and removing some soil from the currently used edge. The next subsection expresses about this local soil updating.

4.3 Local soil updating

When an IWD leaves node \( i \) by using edge \( e_{i,i+1}(k) \) to arrive at node \( i+1 \), the soil of the IWD, \( \text{soil}_{\text{IWD}} \), and the soil of the used edge, \( \text{soil}(e_{i,i+1}(k)) \), both are updated by

\[
\text{soil}(e_{i,i+1}(k)) = (1 + \rho_n) \cdot \text{soil}(e_{i,i+1}(k)) - \rho_n \cdot \Delta\text{soil}(e_{i,i+1}(k))
\]

\[
\text{soil}_{\text{IWD}}^{\text{IWD}} = \text{soil}_{\text{IWD}} + \Delta\text{soil}(e_{i,i+1}(k))
\]

where \( \Delta\text{soil}(e_{i,i+1}(k)) \) is calculated by

\[
\Delta\text{soil}(e_{i,i+1}(k)) = \frac{a_s}{b_s + c_s \cdot \text{time}^2(i,i+1;\text{vel}_{\text{IWD}}(t+1))}
\]

Such that

\[
\text{time}(e_{i,i+1}(k);\text{vel}_{\text{IWD}}(t+1)) = \frac{\text{HUD}(e_{i,i+1}(k))}{\text{vel}_{\text{IWD}}(t+1)}
\]

where the updated velocity of the IWD denoted by \( \text{vel}_{\text{IWD}}(t+1) \) is computed by

\[
\text{vel}_{\text{IWD}}(t+1) = \text{vel}_{\text{IWD}}(t) + \frac{a_v}{b_v + c_v \cdot \text{soil}^2(e_{i,i+1}(k))}
\]

Therefore, an edge with less soil lets the IWD gain more speed than an edge with more soil.

In addition, the heuristic undesirability \( \text{HUD}(e_{i,i+1}(k)) \) represents how much the edge \( e_{i,i+1}(k) \) is undesirable to be chosen by the IWD, and it is related to the problem at hand. For data clustering, \( \text{HUD}(e_{i,i+1}(k)) \) should be defined in a way that reflects how much the
Improving K-means clustering algorithm with the IWD algorithm

The current data vector (denoted by node \(i\)) is preferable to be included in cluster \(k\) than the other clusters. Here, \(HUD(e_{i, i+1}(k))\) is defined by

\[
HUD(e_{i, i+1}(k)) = \sum_{j=1}^{d} (x_{ij} - m_{ij})^2
\]

where \(x_{ij}\) is the jth component of the jth data vector \(x_i = [x_{i1}, x_{i2}, \ldots, x_{id}]^T\). Also, \(m_{ij}\) is the jth component of \(m_k = [m_{k1}, m_{k2}, \ldots, m_{kd}]^T\). It is noted that \(m_k\) is the representative vector of cluster (cluster centre) \(k\). As a result, \(HUD(e_{i, i+1}(k))\) is small when the data vector \(x_i\) is close to the representative vector \(m_k\) whereas \(HUD(e_{i, i+1}(k))\) is great in value if \(x_i\) is far from the cluster representative \(m_k\).

When all the IWDs reach the last node of the problem’s graph, the global soil updating is executed on the edges of the IBS in an IWD algorithm. However, in the proposed IWD-KM, a few processes are executed before the global soil updating is performed. These processes are introduced in the next subsection.

4.4 Cluster centre calculation

When all the IWDs have completed their solutions, based on the cluster assignments found by each IWD, its corresponding cluster centres are computed. Specifically, consider the solution constructed by an IWD is denoted by set \(T^{IWD} = \{t_1^{IWD}, t_2^{IWD}, \ldots, t_M^{IWD}\}\) with \(M\) elements such that \(t_j^{IWD}\) may possess a value from the set \(\{1, 2, \ldots, K\}\). Then, the cluster centres \(m_k^{IWD}\) for the IWD with solution \(T^{IWD}\) are obtained as follows:

\[
m_k^{IWD} = \frac{1}{|J_k^{IWD}|} \sum_{j \in J_k^{IWD}} x_j
\]

where \(J_k^{IWD} = \{j \mid t_j^{IWD} = k\}\). In addition, the symbol \(|\ .\ |\) returns the number of elements of its argument.

4.5 Local chaotic search

When the cluster centres are calculated, then a chaotic-based modification is used to locally change the cluster centres \(m_k^{IWD}\):

\[
Tm_k^{IWD} \leftarrow \omega \cdot m_k^{IWD} + (1 - \omega) \cdot (Chaos - 0.5)
\]

where \(\omega = 0.9999\) and Chaos is a chaotic variable, which is generated by the logistic map:

\[
Chaos \leftarrow 4 \cdot Chaos \cdot (1 - Chaos).
\]

Then, the objective functions for both cluster centres

\[
M^{IWD} = \{m_1^{IWD}, m_2^{IWD}, \ldots, m_K^{IWD}\}
\]
and

\[ T^{IWD}_M = \{ T^{IWD}_1, T^{IWD}_2, ..., T^{IWD}_k \} \]

are calculated by:

\[ f \left( X, M^{IWD} \right) = \sum_{j=1}^{N} \left\| x_j - m^{IWD}_{k(j)} \right\| \] (21)

and

\[ f \left( X, T^{IWD}_M \right) = \sum_{j=1}^{N} \left\| x_j - T^{IWD}_{k(j)} \right\| \] (22)

where

\[ k(j) = \arg \min_k \left( x_j - m^{IWD}_k \right) \] (23)

If

\[ f \left( X, T^{IWD}_M \right) < f \left( X, M^{IWD} \right) \]

then

\[ M^{IWD} \leftarrow T^{IWD}_M, \]

and the solution \( T^{IWD} \) is updated as follows:

\[ t^{IWD}_j = \arg \min_j \left( x_j - m^{IWD}_k \right) \quad \text{for} \; j = 1, 2, ..., M \] (24)

And the local chaotic search is terminated. Otherwise, \( M^{IWD} \) is again changed by the chaotic-based rule of equation (20) and the whole process above is repeated for at most ten times.

4.6 Global soil updating

After the local chaotic search, once more the cluster centre calculation is performed. Then, the IBS \( T^{IB} \) is found. Afterwards, the soils of the edges forming the solution \( T^{IB} \) are updated by:

\[ \text{soil} \left( e_{i,i+1}(k) \right) = \min \left\{ \max \left( \text{Tempsoil} \left( e_{i,i+1}(k) \right), \text{MinSoil} \right), \text{MaxSoil} \right\} \quad \forall e_{i,i+1}(k) \in T^{IB} \] (25)

where

\[ \text{Tempsoil} \left( e_{i,i+1}(k) \right) = \left( 1 - \rho_{IWD} \right) \cdot \text{soil} \left( e_{i,i+1}(k) \right) - \rho_{IWD} \cdot \frac{\text{soil}^{IWD}_M}{M} \quad \forall e_{i,i+1}(k) \in T^{IB} \] (26)
Here, the global soil updating is bounded by the interval \([\text{MinSoil}, \text{MaxSoil}]\) to prevent underutilisation of any cluster. Experimentally, \(\text{MinSoil} = 50\) and \(\text{MaxSoil} = 100,000\). Moreover, \(\text{Soil}_{IB}^{IWD}\) represents the amount of soil that the IWD with the IBS has gathered from the edges of the graph. In other words, \(\text{Soil}_{IB}^{IWD}\) implicitly represents the quality of the IBS \(T^{IB}\). As a result, there is no need to explicitly calculate the quality of the solutions \(T^{IWD}\). In fact, for any IWD with solution \(T^{IWD}\), the soil that the IWD carries denoted by \(\text{Soil}_{IWD}\) takes the value of function \(q(T^{IWD})\) in equation (12).

When the global soil updating is finished, one iteration of the IWD-KM has ended and another iteration begins with new IWDs. This process continues until the maximum number of iterations is met or when the TBS does not change in consecutive iterations.

Next section employs the proposed IWD-KM for data clustering of some well-known data sets.

5 Experimental results

In this section, the proposed IWD-KM is tested with six well-known datasets downloaded from the UCI machine learning dataset repository (UCI, 2010). The performance of the IWD-KM is compared to the K-means algorithm. It is noted that the experiments are conducted on a Notebook PC with Pentium 4 CPU 2.53GH and 4 GB RAM using Microsoft Windows 7 home edition. The algorithms are written using the C# language in the Microsoft Visual Studio.NET.

The six datasets are introduced briefly as follows:

- Iris dataset: This dataset contains 150 data vectors of three classes. Each class holds 50 instances of the data vectors related to a certain kind of the iris plant. Every data vector excluding its class information is composed of four attributes, which are: sepal length, sepal width, petal length, and petal width, all in centimetres. The class information for each data vector holds the species of the Iris plant, which are Setosa, Versicolour, or Virginica.

- Wine dataset: This dataset includes a chemical analysis of wines grown in the same region in Italy but taken from three different cultivars. The three classes of the wines are represented by 13 attributes, which are:

  1. alcohol
  2. malic acid
  3. ash
  4. alcalinity of ash
  5. magnesium
  6. total phenols
  7. flavanoids
  8. non-flavanoid phenols
Out of the total 178 data vectors, the first class contains 59 data vectors. The second class holds 71 data vectors, and the third class holds 48 data vectors.

- Thyroid dataset: This dataset includes 150 data vectors. Each data vector is composed of five attribute values excluding the class information. The dataset is used to predict whether a patient’s thyroid belongs to the class euthyroidism, hypothyroidism, or hyperthyroidism.

- Wisconsin Breast cancer dataset: It contains 699 instances each composed of 11 attributes. The first attribute is the id number. The last attribute holds the class information. There are two classes: malignant and benign. The value of 2 stands for benign and the value of 4 represents the malignant class. The attributes from two to ten are:
  1. clump thickness
  2. uniformity of cell size
  3. uniformity of cell shape
  4. marginal adhesion
  5. single epithelial cell size
  6. bare nuclei
  7. bland chromatin
  8. normal nucleoli
  9. mitoses.

  There are 16 missing attributes in 16 instances of the dataset. In the experiments, we remove these instances from the dataset.

- E. coli dataset: It contains 336 instances related to protein localisation sites in Eukaryotic cells. Each instance holds nine attributes. The first attribute is the sequence name. The rest of the attributes excluding the last one are the attributes which are used for clustering or classification. The last attribute is the symbol showing the class information, which is one of eight location sites.

- Image segmentation dataset: It contains 2,100 instances for test and 210 instances for training. Here, the 2,100 test instances are used. Each instance is composed of 19 attributes. And there are seven classes in the instances.

For the first five datasets, the experiments with the proposed IWD-KM and the K-means are repeated for ten times and the solutions after convergence are reported in Table 1. The results of clustering in terms of the objective function defined in equation (3) are shown in the table.
Improving K-means clustering algorithm with the IWD algorithm

Table 1  The performance of the proposed IWD-KM and the K-means for five datasets in terms of objective function in ten runs

<table>
<thead>
<tr>
<th>Dataset:</th>
<th>Iris</th>
<th>Thyroid</th>
<th>Wine</th>
<th>Wisconsin breast cancer</th>
<th>E. Coli</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. clusters:</td>
<td>Three</td>
<td>Three</td>
<td>Three</td>
<td>Two</td>
<td>Eight</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Best STD</td>
<td>Best STD</td>
<td>Best STD</td>
<td>Best STD</td>
<td>Best STD</td>
</tr>
<tr>
<td>IWD-KM</td>
<td>97.22 0</td>
<td>1,961.75 0</td>
<td>16,530.54 0</td>
<td>2,984.07 0</td>
<td>62.28 0.13</td>
</tr>
<tr>
<td>K-means</td>
<td>97.33 12.58</td>
<td>1,987.93 6.83</td>
<td>16,555.43 869.71</td>
<td>2,988.43 0</td>
<td>62.43 0.14</td>
</tr>
</tbody>
</table>

Note: STD stands for standard deviation.

For four datasets Iris, Thyroid, Wine, and Wisconsin breast cancer, the IWD-KM produces the same clustering. In fact, the standard deviation of the objective function of equation (3) for the four datasets has a value of zero in all ten runs of the IWD-MKP. Only for the E. coli, the standard deviation is 0.13, which is slightly greater than zero.

Another point is that the IWD-KM produces the lowest values for the objective function in ten runs of the algorithm whereas the K-means cannot produce the same objective values in its ten runs.

To be sure of superiority of the IWD-KM, the K-means is run for 1,000 iterations to find the best solution for the Wine dataset using the Matlab function kmeans. It is mentioned that for every run, a random set of initial cluster centres is chosen from the dataset. This 1,000 runs of the K-means took 7.56 seconds to finish and the best result among the 1,000 runs obtains the objective value 16,555.43. In contrast, the IWD-KM finds the solution 16,530.54 in less than 1.52 seconds.

To complete the aforementioned experiment, from two to nine clusters for the Wine dataset are tested by both the K-means with 1,000 runs and the IWD-KM with one run. The results are summarised in Table 2. In two out of the eight cases of the experiments of Table 2, the K-means gets lower values in terms of the objective function. However, in the other six cases, the IWD-KM gets better objective values, which reflects the superiority of the IWD-KM in respect of the K-means even with multiple runs. It is noted that the computation times are also reported in the table.

Table 2  The performance of the IWD-KM with one run against the K-means with 1,000 different runs for the dataset ‘Wine’

<table>
<thead>
<tr>
<th>Dataset: Wine</th>
<th>No. clusters:</th>
<th>Two</th>
<th>Three</th>
<th>Four</th>
<th>Five</th>
<th>Six</th>
<th>Seven</th>
<th>Eight</th>
<th>Nine</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IWDM-KM</td>
<td>23,562.74 (1.22)</td>
<td>16,530.54 (1.52)</td>
<td>12,789.35 (1.57)</td>
<td>10,563.05 (1.76)</td>
<td>8,642.12 (1.93)</td>
<td>7,808.78 (5.16)</td>
<td>7,205.86 (5.55)</td>
<td>5,787.49 (6.4)</td>
</tr>
<tr>
<td></td>
<td>K-means</td>
<td>23,799.98 (7.22)</td>
<td>16,555.43 (7.56)</td>
<td>12,836.64 (10.52)</td>
<td>10,668.18 (11.29)</td>
<td>9,147.13 (11.65)</td>
<td>6,418.93 (12.08)</td>
<td>5,970.43 (13.41)</td>
<td></td>
</tr>
</tbody>
</table>

Note: The numbers in parentheses reflect the times in seconds.

Moreover, to test the IWD-KM with larger datasets, the dataset ‘Segmentation’ with 2,100 input vectors is clustered by both the K-means and the proposed IWD-KM for different number of clusters where the results are expressed in Table 3. Here, the IWD-KM uses 100 IWDs to cluster the dataset. The K-means is run for 7,000 times for
clustering into two and three clusters and the best clustering in terms of the objective function is reported in the table. For clustering into four, five, six, seven, eight and nine clusters, the K-means is run for 2,000 times and the only the best results are written in the table. In addition, the computation times for both K-means and the IWD-KM are stated in parentheses in Table 3. Although the times taken to stop the IWD-KM are lower than those of the K-means, the objective values obtained by the IWD-KM are much lower than those of the K-means.

Table 3  The performance of the IWD-KM with one run against the K-means with 7,000 different runs for creating two and three clusters, and 2,000 different runs for the rest of clustering with the dataset ‘segmentation’ holding 2,100 data vectors

<table>
<thead>
<tr>
<th>Dataset: Segmentation</th>
<th>No. clusters: Two</th>
<th>Three</th>
<th>Four</th>
<th>Five</th>
<th>Six</th>
<th>Seven</th>
<th>Eight</th>
<th>Nine</th>
</tr>
</thead>
<tbody>
<tr>
<td>IWD-KM</td>
<td>211,235.54</td>
<td>174,318.5</td>
<td>161,837.59</td>
<td>151,155.17</td>
<td>143,347.18</td>
<td>135,456.00</td>
<td>129,374.56</td>
<td>126,040.27</td>
</tr>
<tr>
<td></td>
<td>(108.04)</td>
<td>(43.30)</td>
<td>(50.00)</td>
<td>(47.88)</td>
<td>(48.42)</td>
<td>(57.45)</td>
<td>(60.84)</td>
<td>(60.04)</td>
</tr>
<tr>
<td>K-means</td>
<td>211,241.22</td>
<td>174,399.64</td>
<td>168,973.28</td>
<td>154,807.89</td>
<td>144,491.45</td>
<td>143,011.10</td>
<td>134,507.83</td>
<td>126,154.29</td>
</tr>
<tr>
<td></td>
<td>(108.07)</td>
<td>(314.16)</td>
<td>(54.57)</td>
<td>(86.67)</td>
<td>(117.48)</td>
<td>(132.69)</td>
<td>(145.19)</td>
<td>(129.63)</td>
</tr>
</tbody>
</table>

Note: The numbers in parentheses reflect the times in seconds.

It is mentioned that other modifications of the K-means have been introduced but with a price of an increase in computational complexity. The reader is referred to Hansen et al. (2005).

The performance of the IWD-KM in terms of the objective function versus iteration for the experiments of Table 1 is shown in Figures 3 to Figure 7. In all the experiments, the number of the IWDs is ten unless expressed otherwise. Both the IBS and the TBS are shown in the figures. It is seen that the curves of the IBS are always close to those of the TBS. This fact experimentally demonstrates that the IWD-KM for data clustering is convergent. However, the iteration number that the IWD-KM converges is different for the datasets.

Figure 3  The objective values of the IBS and TBS versus the number of iterations for Iris dataset using the IWD-KM
Figure 4  The objective values of the IBS and TBS versus the number of iterations for Wine dataset using the IWD-KM

Figure 5  The objective values of the IBS and TBS versus the number of iterations for Thyroid dataset using the IWD-KM

Figure 6  The objective values of the IBS and TBS versus the number of iterations for Wisconsin breast cancer dataset using the IWD-KM
In terms of computation time, one run of the K-means converges faster with fewer iterations than the proposed IWD-KM. Therefore, there is still much work to be done to converge faster than the K-means and producing better results with zero standard deviations.

6 Conclusions

In this paper, the IWD algorithm was combined by the processes used in the K-means to create a hybrid clustering algorithm. The proposed algorithm called IWD-KM was tested with several well-known datasets. The intention was to alleviate the drawbacks of the K-means which are sensitive to initial conditions and getting trap into local optimums. The proposed IWD-KM produced better clustering results whereas it showed zero or near zero sensitivity to initial conditions. However, the K-means is faster to converge and in the future the IWD-KM should be modified to converge faster. Moreover, other versions of the K-means algorithm may be used in combination with the IWD algorithm. It is also possible to use the IWD algorithm solely for data clustering without using the K-means.

References


Improving K-means clustering algorithm with the IWD algorithm


