machine learning & knowledge extraction

MDPI

Article

Benchmarking Studies Aimed at Clustering and Classification Tasks Using K-Means, Fuzzy C-Means and Evolutionary Neural Networks

Adam Pickens and Saptarshi Sengupta *

Department of Computer Science and Information Systems, Murray State University, Murray, KY 42071, USA; apickens@murraystate.edu

* Correspondence: sengupta.sap@gmail.com; Tel.: +1-270-809-3982

Abstract: Clustering is a widely used unsupervised learning technique across data mining and machine learning applications and finds frequent use in diverse fields ranging from astronomy, medical imaging, search and optimization, geology, geophysics, and sentiment analysis, to name a few. It is therefore important to verify the effectiveness of the clustering algorithm in question and to make reasonably strong arguments for the acceptance of the end results generated by the validity indices that measure the compactness and separability of clusters. This work aims to explore the successes and limitations of two popular clustering mechanisms by comparing their performance over publicly available benchmarking data sets that capture a variety of data point distributions as well as the number of attributes, especially from a computational point of view by incorporating techniques that alleviate some of the issues that plague these algorithms. Sensitivity to initialization conditions and stagnation to local minima are explored. Further, an implementation of a feedforward neural network utilizing a fully connected topology in particle swarm optimization is introduced. This serves to be a guided random search technique for the neural network weight optimization. The algorithms utilized here are studied and compared, from which their applications are explored. The study aims to provide a handy reference for practitioners to both learn about and verify benchmarking results on commonly used real-world data sets from both a supervised and unsupervised point of view before application in more tailored, complex problems.

Keywords: evolutionary neural networks; particle swarm optimization; k-means; fuzzy c-means; optimization

Citation: Pickens, A.; Sengupta, S. Benchmarking Studies Aimed at Clustering and Classification Tasks Using K-Means, Fuzzy C-Means and Evolutionary Neural Networks. *Mach. Learn. Knowl. Extr.* **2021**, *3*, 695–719. https://doi.org/10.3390/make3030035

Academic Editors: Basabi Chakraborty and Saptarsi Goswami

Received: 26 July 2021
Accepted: 24 August 2021
Published: 31 August 2021

1. Introduction

The concept of machine intelligence (M.I.) has been studied and expanded upon over the course of many years. There are a wide variety of applications that the various methods and algorithms belonging to the discipline can be applied to, and as technology grows and improves, more techniques have become feasible to implement given a spurt in computational capacity. The ability to classify data into groups of interest has a wide range of applications in many fields from medical imaging to astronomy. In addition to that, it is particularly important to businesses that rely on field data to make decisions about resource allocation. It is thus important to verify their effectiveness to ensure the validity of the end results generated from metrics returned by the algorithms. This work attempts to explore classification and clustering algorithms, with a focus on the successes and limitations for each. To do so, a set of benchmarking data sets of varying sizes and complexities are taken and run through implementations of the algorithms, and the performance indices are calculated and recorded for later analysis. The algorithms explored in this work are K-Means and Fuzzy C-Means as well as an evolutionary feed-forward neural network with a fully connected topology particle swarm optimization (PSO), which serves as the weight update scheme. K-Means is a widely known algorithm and has been utilized in

many different situations. While it is known for being flexible and simple to implement, it is extremely sensitive to its initialization parameters and tends to become stuck in local optima [1]. The Fuzzy C-Means (FCM) algorithm takes a soft clustering approach, utilizing a membership function that allows a data point to belong to multiple clusters at once. Similar to K-Means however, models can often become stagnant and fall into local optima [2]. Branching out somewhat from clustering algorithms is the classifier utilizing a feed-forward neural network with the PSO optimizer instead of traditional gradient descent. Neural networks can have promising performance for pattern recognition and classification problems, as they are able to approximate a function that can represent the patterns found in the input data [3]. Unless adequate care is taken, however, the network can end up over-fitting or under-fitting the data, and its optimizer may be trapped in local minima. To remedy this, it is necessary to properly propagate the node weights at the various iterations of the training process. This is achieved by using a gradient-free swarm algorithm viz. the PSO, which fine-tunes the network weights. While PSO can be used in conjunction with other algorithms, it is also able to act as a standalone optimizer. PSO works well for finding optimization solutions globally, as it can sample a large region of the search space within a given computational budget, but it must be parametrized properly to avoid trapping in a local optimum [4]. There are a number of parameters that influence the functionality of the algorithm, and while this can allow it to be reworked to better fit an individual data set, incorrect tuning can also cause its performance to suffer. Thus, hybridizing PSO with another classifier algorithm can help eliminate the weaknesses of both [1] for computational applications as in [5].

There have been several studies that have looked at integrating swarm optimizers within feed-forward neural networks in order to maximize performance gains irrespective of the nature of the cost function across a variety of swarm intelligence algorithms [6–8]. In general, swarm optimizers are a collection of guided random search techniques that draw inspiration from natural or physical processes and use multi-agent group dynamics to formulate laws of motion and convergence [9–14]. These algorithms are distributed in nature and are well-known for their efficacy in approximately solving real-world engineering problems such as job shop scheduling [15], vehicle routing [16,17], digital filter design [18,19], portfolio optimization [20,21], and more.

The goal of this work is to explore the performance of these algorithms and comment on the successes and limitations of each, and by doing so, form a better understanding of how they can influence the results so that when they are applied to real world problems, the meaning behind the results is better illuminated. A flowchart representing the organization of the paper can be found in Figure 1. The subsequent sections are structured as follows: Section 2.1 details the K-Means algorithm. Section 2.2 does the same for the Fuzzy C-Means algorithm. Section 2.3.1 describes a feed-forward neural network. Section 2.3.2 describes the PSO algorithm. Section 3 details the experimental setup. Section 4 covers the results and discussion of the experimentation, and finally, Section 5 contains the conclusions reached and possible future work in the area.

Objective of the research: To put the main objective of the experiment; in other words, the goal is to take a more intimate look at the set of algorithms and attempt to understand not just how they function, but what that functionality means for the situations to which they are applied. Every algorithm has strengths and weaknesses that can be studied and worked around, but by understanding these strengths and weaknesses, a new perspective can be gained on how they can apply to the results. While it is easy enough to apply the algorithm and take the results at face value, understanding how those results came about allows us to make more sense of how the data set is interacting with the algorithm and if any biases or misrepresentations are influencing our ability to understand the model. This is not an easy task, however. Not all algorithms can be broken apart easily and making sense of any information gleaned from doing so can be difficult given the complexity that goes into these algorithms. Furthermore, the data being analyzed play a large part in this process, so a decent understanding of the data sets is also incredibly important. It is

hoped that this work will serve as a tutorial for practitioners looking at data clustering and will influence their choice of model selection, parameter choices, and expected results for certain popular data sets.

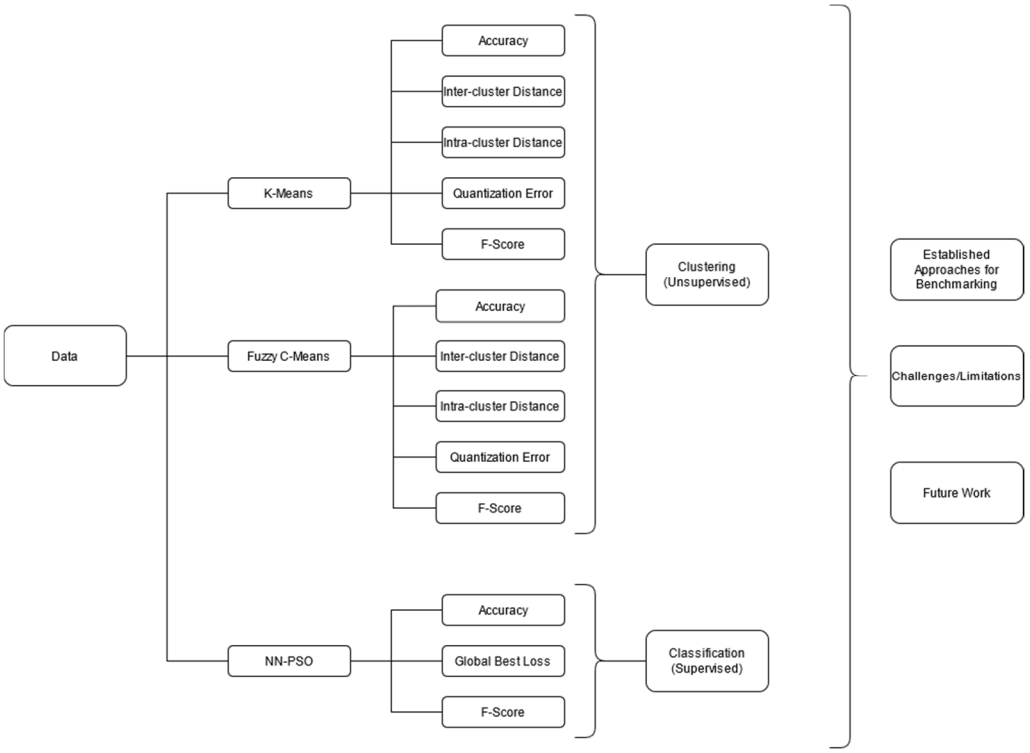

Figure 1. Organization of the paper.

2. Materials and Methods

2.1. K-Means Algorithm

The K-Means algorithm is a well-known clustering algorithm that has been widely applied in many different computational problems [5,22,23]. It is an iterative algorithm, which means that it runs continuously from a given starting point and generates new approximate solutions by adding improvements to the results from previous runs. As K-Means is a clustering algorithm, its goal is to assign all data points from a given data set into a predefined number of clusters. To do this, it attempts to minimize the Euclidean distance between each of the data points belonging to the distinct cluster groups. The Euclidean distance between two points (in Euclidean space) is simply the length of the line segment between the two chosen points. It can be computed from the Cartesian coordinates of the chosen points using the Pythagorean theorem and is also the Pythagorean distance.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{1}$$

p, q are two points in the Euclidean n-space, and q_i and p_i are the Euclidean vectors initiated at a chosen origin. By minimizing the Euclidean distance between points in the group, a higher similarity score can be determined from the groupings. Furthermore, the points in the cluster groups are represented by the cluster centroid, which can be determined by calculating the average of all of the points in the cluster, much like how the center of mass is computed in an n-body problem. Given this context, it can be said that the K-Means algorithm attempts to determine the positions of the centroids that give us the best results in terms of grouping data points that share similar signatures or that are members of the same cluster.

There are many benefits to using the K-Means algorithm [24]. First off, it is comparatively simple to implement and understand. Given that there are only a handful of calculations needed for the vanilla version of the algorithm and that those calculations are simple to both understand and implement, the algorithm can be put together in a functional way much more easily than many other similar algorithms. Second, execution is fast. As one may expect from the simple design, the iterations of the algorithm can be run very quickly, leading to quick convergence. K-Means is also a very flexible algorithm. Its structure makes data sets of all sizes usable for the purpose of clustering, making it perfect for situations where many data sets of varying sizes and complexities need to be worked with.

K-Means does have its drawbacks, however [24]. The algorithm does not determine the optimal set of clusters on its own, relying on the user to provide them. It is also very sensitive to the initialization of parameters. Changes in how the cluster centroids are initialized or how the data set is ordered or formatted can end up changing the results put forward by the algorithm. The algorithm also assumes that the clusters in the data will be spherical, which means that its effectiveness is much more limited when working with data sets with varied cluster shapes.

The standard implementation scheme of the K-Means algorithm is structured below [1]:

1. Randomly initialize the k cluster centroids in space.

$$Z = \{z_1, z_2, \ldots, z_k\} \tag{2}$$

2. Repeat until a termination condition is satisfied.

Assign each data point in space to the cluster centroids that have the closest distance to the point. The Euclidean distance of data point y_p to the centroid in d-dimensional space is given as:

$$D(y_p.z_j) = \sqrt{\sum_{i=1}^{d} (y_{pi} - z_{ji})^2} \tag{3}$$

Recalculate the cluster centroids based on the definition of centroid. As such, the centroid for cluster j is determined as follows:

$$z_j = \frac{1}{n_j} \sum_{\forall y_p \in C_j} y_p \tag{4}$$

where C_j is the subset of data points belonging to the cluster j, and n_j is the number of data points in this cluster.

The clustering process terminates when one of the following conditions is satisfied:

1. The number of iterations exceeds a predefined maximum.
2. When change in the cluster centroids is negligible.
3. When there is no cluster membership change.

An interested reader may take a look at J. MacQueen's seminal work on the methods for the classification and analysis of multivariate observations [5] for a deeper understanding of the foundations of the K-Means algorithm.

2.2. Fuzzy C-Means Algorithm

Similar to the K-Means algorithm, the Fuzzy C-Means (FCM) algorithm attempts to group a set of N objects into C clusters [25,26]. In other words, the objective is to divide the object set $D = \{D_1, D_2, D_3, \ldots D_N\}$ into C clusters $(1 < C < N)$ with $\Omega = \{\Omega_1, \Omega_2, \Omega_3, \ldots, \Omega_c\}$ representing the cluster centers. Every data point being in-

vestigated is assigned to a cluster, the centroids of which are randomly initialized. This assignment is done using the membership function μ_{ij} defined in [2]:

$$\mu_{ij} = \frac{1}{\sum_{r=1}^{C} \left(\frac{d_{ij}}{d_{rj}} \right)^{\frac{2}{m-1}}} \tag{5}$$

$d_{ij} = \|x_i - y_j\|$ represents the distance between the ith center and the jth data point, $d_{rj} = \|x_r - y_j\|$ is the distance between the rth center and the jth data point, and $m \, \epsilon \, [1, \, \infty)$ is a fuzzifier. The FCM algorithm uses an iterative gradient descent, an optimization algorithm used to find the extremum of a function, to compute centroids. The centroids, in turn, are updated using the following equation [27]:

$$x_i = \frac{\sum_{j=1}^{N} \mu_{ij}^{m} y_j}{\sum_{j=1}^{N} \mu_{ij}^{m}} \tag{6}$$

The objective function to be minimized by the algorithm can be formulated as the sum of membership weighted Euclidean distances:

$$\varphi = \sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij}^{m} \left(\|x_i - y_j\| \right)^2 \tag{7}$$

Through the recursive calculation of Equations (5) and (6), the algorithm can be stopped once a termination condition is met. Similar to other algorithms that utilize gradient descent, FCM can stagnate to local optima in a multidimensional search space. Therefore, the utilization of a stochastic optimization approach to compute cluster centroids can help mitigate this problem to a great extent [28,29].

2.3. Feed Forward Neural Network with a Particle Swarm Optimization (PSO) Driven Weight Update Scheme

For this work, an implementation of a neural network using particle swarm optimization as the weight optimization function was utilized, and while the two are different topics, as they are intrinsically linked in this work; this section will contain descriptions of both.

2.3.1. Feed-Forward Neural Network

A neural network consists of several connected functional units or neurons that come together as parallel information processing units in order to solve classification and regression tasks. Working together, these units can separate the input space into a discrete number of classes, i.e., they can approximate the underlying function that maps inputs to outputs. For this work, a feed-forward neural network is used. This is a multi-layered network of neurons consisting of an input and output layer of nodes along with several hidden layers that link them. The hidden layers learn the intermediate mappings that eventually are propagated to form the final mapping in the form of the output. Within a hidden layer, each neuron is connected to the outputs of a subset (or all) of the neurons in the previous layer each multiplied by a number called a weight. Neural networks learn by changing their weights to approximate a function representative of the input patterns. Instead of traditional gradient descent, the PSO algorithm is used as the optimizer to fine tune the network weights as the neural network learns the optimal decision boundary on the benchmarking data sets. A simple model of the discussed network utilizing a fully connected PSO is shown in Figure 2.

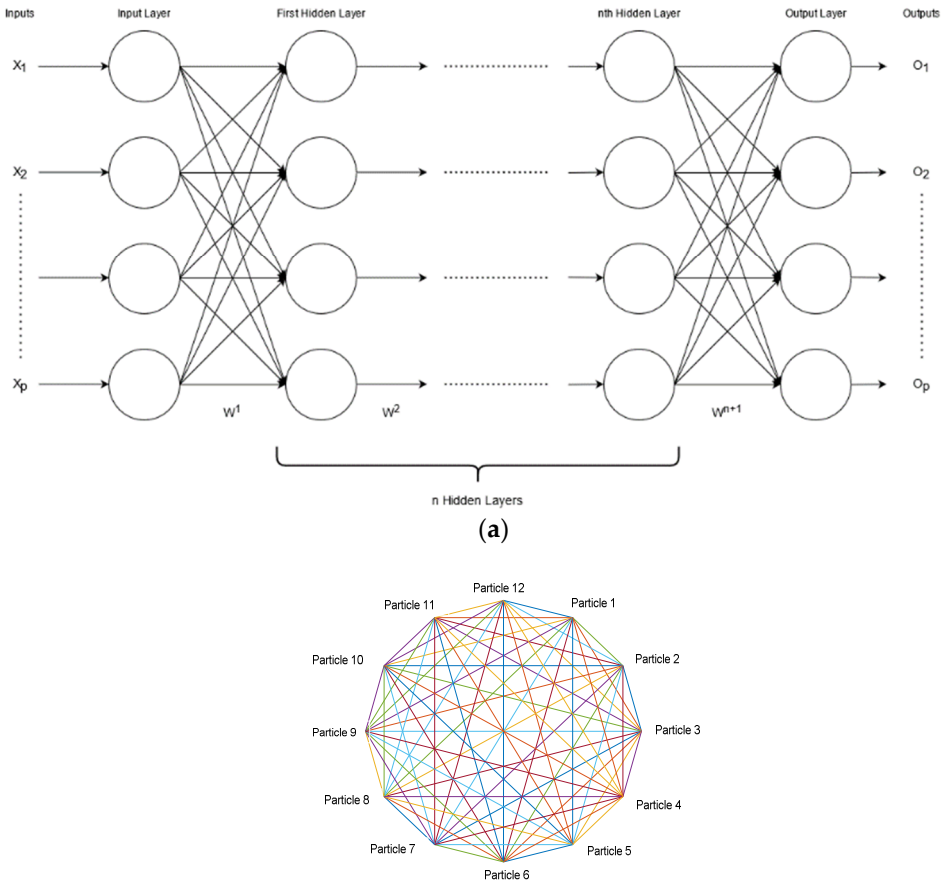

Figure 2. (**a**) A feed-forward neural network with n hidden layers, p input variables, and q output variables, with weights W; (**b**) fully connected topology [30] in PSO where each particle is connected to all other particles in the swarm (shown using 12 particles).

While there are various types of propagation in neural networks, the feed-forward network only passes the data between nodes in one direction, toward the output nodes. In this type of algorithm, the first step is to randomly initialize the weights before tuning the values to minimize the error. This is followed by passing the data forward between layers, applying the weights and evaluating the predictions. These steps will repeat for each layer. The output for the layers in the network can be represented by the following equation [31]:

$$X_{i+1} = f_i(W_i X_i + b_i) \tag{8}$$

In this equation X_i is the input and f_i is the activation function in layer i. The output of said layer, X_{i+1}, becomes the input for the subsequent layer. Once trained, the network should be able to approximate a function that can fulfil the design goal for the algorithm [31].

There are several problems that can arise if the network is not properly trained. Things such as overfitting, results being trapped by local minima, and the vanishing gradient problem can all affect the results depending on how the algorithm is structured. For this reason, many techniques such as altering the weight initialization technique have been used to help mitigate some of these issues [31].

2.3.2. Particle Swarm Optimization (PSO)

The particle swarm optimization (PSO) algorithm is a metaheuristic global optimization paradigm that has received prominence over the last two decades owing to its easy implementation with efficient performance over multidimensional problems that cannot be solved using traditional deterministic algorithms. Its design was inspired by the movement of a flock of birds [1]. For the scope of the algorithm, each "bird" would be represented by a particle, while the "flock" is represented by the swarm. To utilize the swarming mechanisms of the PSO, each particle is initialized by the algorithm and is then placed in the search space. After this initialization, the program is run iteratively, with each particle utilizing a velocity calculation to determine how they "fly" through the search space. In this computation, three different parameters are used. First, the inertial weight of the particle, which represents how much the previous direction of movement influences the next. Second, the social weight, which determines how much the global best position influences the particles' movement. Finally, the cognitive weight, which determines how much the location of an individual particle's personal best location influences its movement. The equations for calculating the velocity and position after each iteration is as follows [2]:

$$v_{ij}(t+1) = \omega v_{ij}(t) + C_1 r_1(t)\big(p_{ij}(t) - x_{ij}(t)\big) + C_2 r_2(t)\big(p_{gj}(t) - x_{ij}(t)\big) \quad (9)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (10)$$

C_1 and C_2 are constants representing the social and cognitive weights, r_1 and r_2 are independent and identically distributed random numbers between 0 and 1. x_{ij} and v_{ij} represent the position and velocity, respectively, of the *ith* particle in *ith* dimension, whereas $p_{ij}(t)$ and $p_{gj}(t)$ are the personal best (pbest) and global best (gbest) positions. In the velocity update scheme, ω represents the inertial weight of the *ith* particle, and the following two terms add in guided vectors towards areas of attraction (also known as local attractors) in the particle's direction of movement. The personal best update uses a greedy update scheme with regard to a cost minimization goal, modeled using the following equation:

$$f(x_i(t+1)) < f(p_i(t)) \Rightarrow p_i(t+1) = x_i(t+1)\, else\ p_i(t+1) = p_i(t) \quad (11)$$

In this equation, set f is the cost, and p_i is the personal best of the particle. The global best (p_g) is the element with the lowest cost result from the set of personal bests of a particular particle. While PSO can cover a large hyperplane of the search space, finding the right values of the parameter set that lead to the best outcome is a time-consuming task that requires careful analysis.

3. Experimental Setup

3.1. Parameter Settings for PSO

In our analysis of the clustering benchmarks using the K-Means implementation, the centroids were determined through a randomly selected group of points. To ensure adequate variability in the starting points, each run of the algorithm had its starting points determined independently, and the mean and standard deviation of the performance metrics were reported. The K-Means algorithm expects the number of clusters to be set beforehand, so care was taken to do the same for each of the data sets considered. The Fuzzy C-Means implementation is similar to K-Means in how starting centroid locations needs to be determined before the model begins to run. Just as with K-Means, the centroids were randomly chosen; however, unlike K-Means, the starting centroids were randomly chosen locations within the search space of the data set instead of from the existing points. In addition to this, the membership matrix needed to be initialized. Since each data point has three factors determining membership with all clusters, each point was given a corresponding list of factors and was randomly assigned one cluster to have a strong correlation with. The parameters for the feed-forward neural network with particle swarm

optimization (FNN-PSO) for the experiments were the inertia weight ω and the cognitive and social parameters C0 and C1. For the inertia weight, an upper bound and a lower bound were chosen to be the range determinants, and during execution, a randomly generated value inside this range was chosen. This was repeated for each particle in the swarm during the optimization phase of the neural network. To ensure optimal results for all benchmarking data sets, the inertia weight range and the social and cognitive parameters were tailored to each individual data set. Table 1 contains the values used for each data set. The parameter selection was conducted by linearly decreasing the inertia weight between the stated ranges in Table 1 and by conducting a grid search with increments of 0.1 between 0 and 1 to determine the optimum cognitive and social factor values. The neural network parameters also needed to be initialized for each run. To ensure that the data set could be properly processed, the number of input nodes was set to the number of attributes for each individual data set, and to compliment this, the number of output nodes was set to one less than the number of input nodes. As for the number hidden layers, a static 20 was used for all of the data sets. This was done to add a bit more consistency to the repeated runs of each data set.

Table 1. Neural network with PSO parameter information.

Data Set	Inertia Weight Range	Cognitive Factor	Social Factor
Iris	0.7–0.9	0.5	0.7
Breast Cancer	0.9–0.9	0.4	0.1
Seeds	0.7–0.9	0.9	0.1
Mammographic Mass	0.6–0.9	0.1	0.9
Sonar	0.9–0.9	0.2	0.1

While there is overlap between many of the factors utilized for this experiment, given how sensitive the implementation was to each of them, adequate fine tuning was needed to ensure that the algorithm generated the best results. The best example for this collection is the Seeds data set. Unlike K-Means, the neural network with PSO did not have the number of categories/clusters initialized at the start, so the algorithm needed to generate the categories for each run. The Seeds data set contains three categories, and while the outside of this the data itself does not appear to be much more complex than the other sets, obtaining a properly usable set of models from the initialization was much harder for this data set than it was for the other data sets. The Seeds data set took longer than any other data set to find the proper parameters for, and it still was the data set that the model struggled the most with to obtain accurate results. The further the initialization parameters strayed from their optimal values, the more difficult it was to obtain accurate results from the model. At times, the model would end up with a prediction of the incorrect number of categories, usually ending up stagnated on one prediction category.

3.2. Data Sets

The data sets utilized for the benchmarking are five well known ones. All but the Iris data are taken from the UCI Machine Learning Repository [32], with the Iris data set being taken from the scikit-learn library [33]. These are described below:

1. R. A. Fisher's Iris data set covering three species of Iris flowers (Setosa, Versicolor, and Virginica) containing a total of 150 instances with 4 attributes each. The attributes are as follows: sepal length, sepal width, petal length and petal width. The first two attributes i.e. Sepal Length and Sepal Width are shown in Figure 3.

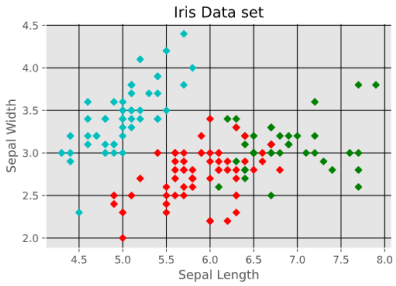

Figure 3. Illustration of the first two attributes of the Iris data set.

2. Breast Cancer Wisconsin (Original) data set consisting of 699 instances and containing 10 attributes. Instances are classified into one of two categories, benign or malignant. The attributes are as follows: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. The first two attributes i.e. Clump Thickness and Uniformity of Cell Size are shown in Figure 4.

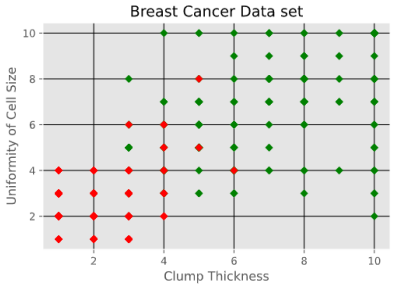

Figure 4. Illustration of the first two attributes of the Breast Cancer data set.

3. Seeds data set, which is used to identify 3 different varieties of wheat: Kama, Rosa, and Canadian. The data set contains 210 instances with 7 attributes each. The attributes are as follows: area, perimeter, compactness, length of kernel, width of kernel, asymmetry coefficient, and length of kernel groove. The first two attributes i.e. Area and Perimeter are shown in Figure 5.

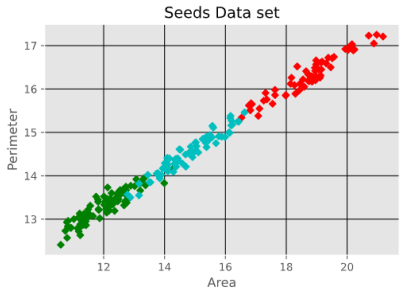

Figure 5. Illustration of the first two attributes of the Seeds data set.

4. Mammographic Mass data set containing 961 instances and is used to classify data into two categories, benign and malignant. Each instance contains 6 attributes based on BI-RADS data and the patient's age. The attributes are as follows: BI-RADS assessment, age, shape, margin, density, and severity. The first two attributes i.e., BI-RADS Assessment and Age are shown in Figure 6.

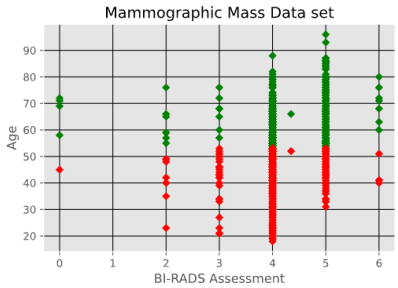

Figure 6. Illustration of the first two attributes of the Mammographic Mass data set.

5. Sonar Data set, which consists of 208 instances with 60 attributes each. The instances are classified into either mines or rocks. The attributes represent the energy within a specific frequency band. The first two attributes i.e. Frequency Band 1 and Frequency Band 2 are shown in Figure 7.

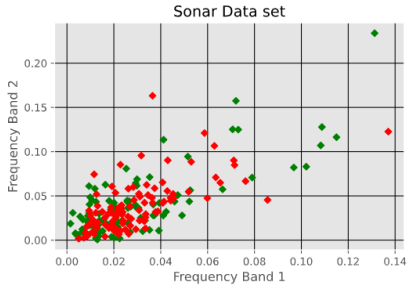

Figure 7. Illustration of the first two attributes of the Sonar data set.

Missing Value Replacement: In the cases of the Breast Cancer data set and the Mammographic Mass data set, the data contained instances where some attribute values were missing and thus needed to be dealt with for the algorithms to function. To this end, the average value for each attribute was calculated, and any missing values were replaced with that average.

3.3. Performance Indices

Measuring the performance indices allows us to determine the effectiveness of a given model to correctly classify instances in each data set. The clustering indices are as follows:

- *Accuracy*: This is also known as the Rand index. The *accuracy* is described as the percentage of correct decisions made by the algorithm. The formula for *accuracy* is as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{12}$$

In this equation, TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and finally, FN is the number of false negatives. A true positive is the outcome in binary classification when the model under test correctly predicts the existence of a condition. A true negative is the outcome when the same model correctly predicts the absence of a condition. A false positive represents an error in binary classification tasks such that the result falsely indicates the presence of a certain condition. On the other hand, a false negative is an error that implies that the result incorrectly fails to indicate the presence of some condition, but it is indeed present.

- *F*-Score: Another means of calculating accuracy. It is calculated using the precision and recall of a model the formula for *F*-Score is as follows:

$$F = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{13}$$

- *Inter-Cluster Distance*: The sum of the distances between each *cluster* centroid. Larger values indicate a greater separation between *clusters*, meaning less overlap between the clusters in the model. The formula for *inter-cluster distance* is as follows [34]:

$$Inter - cluster\ dist = \min\left(\|c_i - c_j\|\right)^2 \tag{14}$$

  where c_j represents the centroid of *cluster* j.
- *Intra-Cluster Distance*: The sum of the distances between individual data points and their respective parent centroid. Smaller values indicate more compact *clusters* and are therefore desired. The formula for *intra-cluster distance* is as follows [34]:

$$Intra - cluster\ dist = \frac{1}{n}\sum_{j=1}^{k}\left|\left|x_i - c_j\right|\right|^2 \tag{15}$$

  k is the number of clusters, c_j represents the centroid j, and n refers to the number of data instances. The section $\left|\left|x_i - c_j\right|\right|^2$ represents the distance between data instances and their respective centroid.
- *Quantization Error*: The sum of the distances between data points and their parent centroid divided by the total number of data points belonging to the cluster and then summed over all *clusters* and averaged for all of the *clusters* in the model. The formula for *quantization error* is as follows [34]:

$$Quantization\ error = \frac{\sum_{j=1}^{k}\left\{\frac{\sum_{i=1}^{N_j}\left|\left|x_i - c_j\right|\right|^2}{N_j}\right\}}{k} \tag{16}$$

In this equation, k is the number of *clusters*, c_j is the centroid of cluster j, N_j represents the number of data points in cluster j, and finally, the section $\left|\left|x_i - c_j\right|\right|^2$ represents the distance between data instances and their respective centroid.

As the PSO neural network is not a clustering algorithm and rather performs a classification task; it uses different indicators to measure performance. As discussed before, these include the global best loss and indices such as *F*-score and accuracy.

The algorithms were implemented in Python 3.9.1 and were executed with an AMD Ryzen 5 3600 6-Core Processor at 3.6 GHz. The experimental results for 50 trials are tabulated and are then analyzed. Table 2 details the data sets used in this work.

Table 2. Data Set Information.

	Number of Instances	Number of Attributes	Number of Categories
Iris	150	4	3
Breast Cancer	699	10	2
Seeds	210	7	3
Mammographic Mass	961	6	2
Sonar	208	60	2

3.4. K-Means Implementation

For the K-Means implementation, the vanilla version of the algorithm was not altered. Before the start of the algorithm, the data set needed to be split into the base data and the classifications associated with each data point. To achieve this, two containers were set, with the first containing a list of the attribute values for each data point with any missing values replaced, and the second containing a list of the corresponding classifications. From this point, the centroids were determined. Originally, the centroids were statically chosen from the data set, but this method was replaced with a randomly selected set of points. While the original method was able to return predictable results, the goal of the experiment

was to study the algorithm's overall performance across a range of choices, and by varying the initialization of the centroids, a much larger range of the attribute space was available to be investigated. The algorithm was set to terminate after a set number of iterations were completed if there were no significant performance gains.

After the algorithm finished each of its runs, the final positions of the centroids were utilized to calculate the performance indices for the run. Once calculated, the results were stored in a separate table to later be collected and averaged. The mean value and the standard deviation over 50 runs were calculated and stored for comparison. The results for the K-Means runs can be found in Table 3.

Table 3. Results from the K-Means Algorithm.

Data	Accuracy	Inter Cluster Distance	Intra Cluster Distance	Quantization Error	*F*-Score
Iris	84.3467 ± 11.5433%	9.9448 ± 0.5077	100.9036 ± 9.1452	0.6525 ± 0.0174	0.8435 ± 0.1154
Breast Cancer	95.9484 ± 0.1393%	27.5215 ± 0.0287	3050.4339 ± 1.2969	5.2425 ± 0.0069	0.9595 ± 0.0014
Seeds	89.2952 ± 0.2379%	15.9245 ± 0.1182	313.4652 ± 0.2585	1.4967 ± 0.0003	0.893 ± 0.0024
Mammographic Mass	66.2042 ± 0.7268%	47.8242 ± 0.2174	6993.7479 ± 10.2771	7.2945 ± 0.0048	0.662 ± 0.0073
Sonar	52.6442 ± 3.7956%	2.5068 ± 0.0124	235.1373 ± 0.4677	1.127 ± 0.0056	0.5264 ± 0.038

Trapping in Local Optima

For the K-Means algorithm and many other clustering algorithms, the starting points can have a strong influence on the performance of the model. Just as there are good starting points that allow the algorithm to find appropriate centroids to represent the clusters, there are also starting points that cause the algorithm to incorrectly model the data. The Iris data, as an example, is quite susceptible to poorly initialized centroids. Figure 8 shows a model with centroids that correctly represent the data, and Figure 9 shows a model where the starting centroids cause the model to incorrectly model the data.

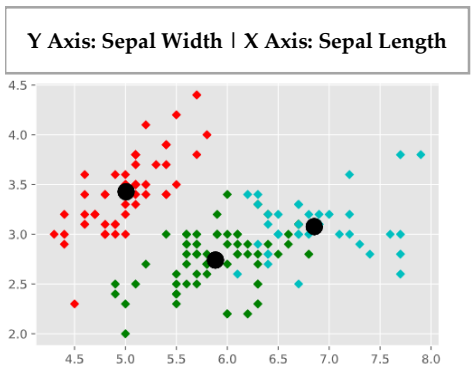

Figure 8. Correctly modeled Iris clusters.

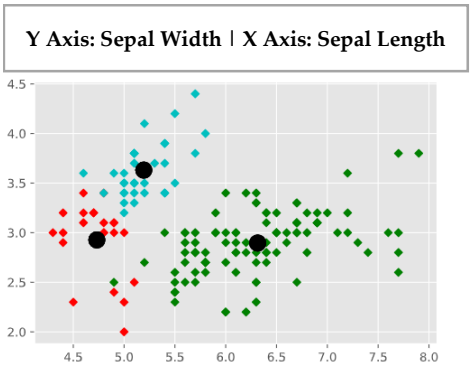

Figure 9. Incorrectly modeled Iris clusters.

In the dimensions of the data, the grouping on the top left of the graph represents a single cluster, while the larger mass is split into the remaining two. However, in a situation where the initialization places two of the centroids in the upper left group, there is a chance that the algorithm will assign two of the clusters to the upper left grouping and leave the remaining centroid to represent the larger grouping in the lower right. This can come as a result of the K-Means algorithm's weakness with regard to getting stuck on local minima [1]. The random initialization of the points is what was used to help mitigate some of the sensitivity to different initialization.

3.5. Fuzzy C-Means Implementation

As with the K-Means algorithm, the base functionality of the algorithm was not altered. Before processing, the data were split into two parts. The first part contained the data points themselves, and the other contained their respective classifications. From there, the centroids were initialized. Unlike K-Means, for the Fuzzy C-Means, implementation of the points was not initialized from points in the data set. Instead, the data were used to set up the range of space where the values are located, and a random set of positions in that space were chosen to act as the centroids. Additionally, unlike K-Means, the membership matrix needed to be initialized. To do so, a membership array was created for each data point in the set, with a number of values equal to the number of classifications/clusters. After the matrix itself was created, each data point was randomly assigned one cluster to have a strong relationship with. As with the other algorithms, the randomness introduced in the initialization facilitated the study of the algorithm's performance over a larger range of the attribute space. The algorithm was set to terminate after a set number of iterations were completed if there were no significant performance gains.

After each run of the algorithm was completed, the final positions of the centroids were used to calculate the various performance indices of the run. After being calculated, the results were stored in a separate table for later collection and averaging. The mean value and standard deviation of over 50 runs were calculated and stored for comparison. The results from the FCM runs can be found in Table 4.

Table 4. Results from the Fuzzy C-Means Algorithm.

Data	Accuracy	Inter-Cluster Distance	Intra-Cluster Distance	Quantization Error	*F*-Score
Iris	88.6667 ± 1.1102e-14%	9.994 ± 3.7006e-15	96.9562 ± 2.309e-14	0.646 ± 9.5505e-17	0.8867 ± 1.1102e-16
Breast Cancer	94.8424 ± 2.2204e-14%	26.9631 ± 7.4012e-15	2681.0984 ± 4.5475e-13	4.7631 ± 1.2243-15	0.9484 ± 2.2204e-16
Seeds	89.5238 ± 1.1102e-14%	16.1425 ± 1.4921e-14	312.5735 ± 4.9555e-14	1.4936 ± 1.3688e-16	0.8952 ± 1.1102e-16
Mammographic Mass	66.875 ± 2.2204e-14%	48.7586 ± 0.572e-14	7023.1509 ± 5.2389e-12	7.3333 ± 4.432e-15	0.6687 ± 2.2204e-16
Sonar	55.2885 ± 0.0000%	1.8598 ± 1.1508e-14	237.3494 ± 4.5848e-13	1.139 ± 1.9135e-15	0.5529 ± 0.0000

3.6. Feed-Forward Neural Network Tuned by a Particle Swarm Optimization Algorithm

While the results of the K-Means and FCM algorithms were promising, to better understand the grouping of the benchmarking data from a predictive analysis point of view, a feed-forward neural network using a PSO optimizer was implemented. Using a supervised learning algorithm such as this that functions quite differently than the clustering algorithms utilized in the experiment allowed us to look at the problem with a different perspective. The results were good, as was expected, but the range of values were notably different than the ones returned by the other algorithms, and as such, different deductions could be made about the functionality of both types of algorithms that were utilized.

As for the implementation itself, similar to the K-Means implementation, the data needed to be split into groups of the data and the classifications. The training data and labels were set up with the raw data and the categories, respectively, and any fixes/replacement of values to the data set were applied. Once finished, the values for all of the necessary nodes were initialized. For this implementation, the dimensionality of the number of inputs, outputs, and hidden nodes were computed, and care was taken to fit the network with the data. The number of input nodes was set to match the number of attributes for

the data set being analyzed, and the number of output nodes was set to be one less than the number of input nodes. For the number of hidden nodes, the number was set to 20 for each data set.

Furthermore, the data sets were split into training and testing groupings, with 70% of the data used to train the model and the remaining 30% used to evaluate the accuracy of the model. After each of the runs, the accuracy, global best loss, and *F*-score were calculated and stored for later use. The results from the 50 runs were stored, and the mean and standard deviation were calculated and stored in Table 5.

Table 5. Results from the FNN-PSO Algorithm.

Data	Accuracy	Global Best Loss	*F*-Score
Iris	99.1556 ± 1.9317%	0.0481 ± 0.0353	0.9916 ± 0.0193
Breast Cancer	94.0381 ± 5.9493%	0.1541 ± 0.109	0.9404 ± 0.0595
Seeds	81.7143 ± 7.4872%	0.4653 ± 0.0995	0.8171 ± 0.0749
Mammographic Mass	78.8264 ± 7.1728%	0.4864 ± 0.0638	0.7883 ± 0.0717
Sonar	71.2666 ± 8.492%	0.546 ± 0.0809	0.7127 ± 0.0849

4. Results and Related Discussion

Tables 3–5 contain the results from the three algorithms used in this experiment. They contain the mean and standard deviation of the performance metrics calculated over 50 runs. Figure 10 plots the accuracy of the K-Means algorithm over its iterations, and Figure 11 plots the clustering model using K-Means in three dimensions. Figure 12 plots the accuracy of the FCM algorithm over its iterations, and Figure 13 plots the clustering model using FCM in three dimensions. Figure 14 plots the loss curve of the FNN-PSO over 1000 iterations, and Figures 15–19 compare the ground truth mapping of the data versus the predicted mappings from the FNN-PSO. Finally, Figures 20–24 model the mean and standard deviation of the accuracy for each distinct data set. All of the colors assigned to the clusters are generated by the clustering and classification algorithms and are not necessarily the same as those of the ground truth when plotted.

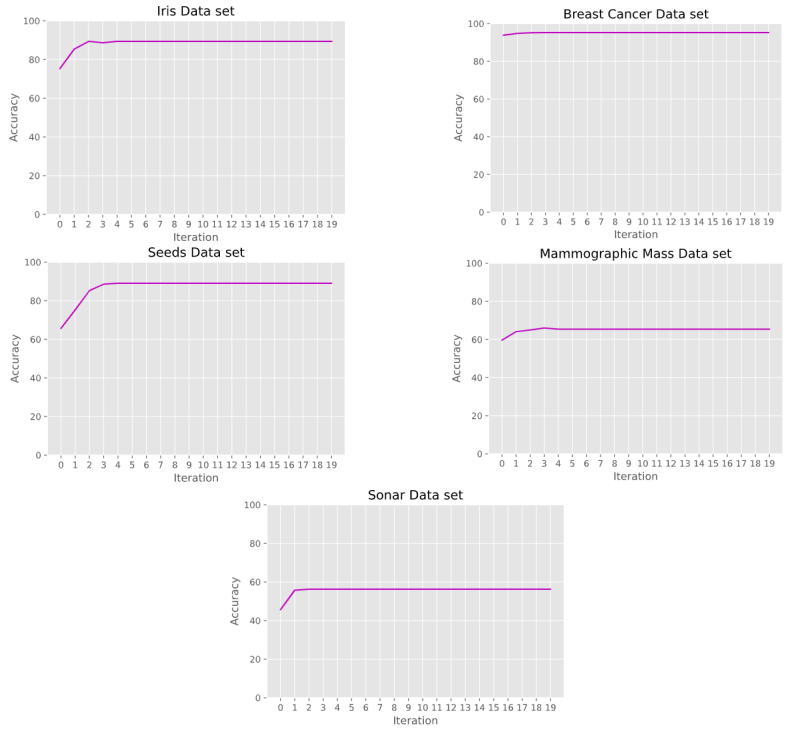

Figure 10. Accuracy using K-Means on the experimental data sets.

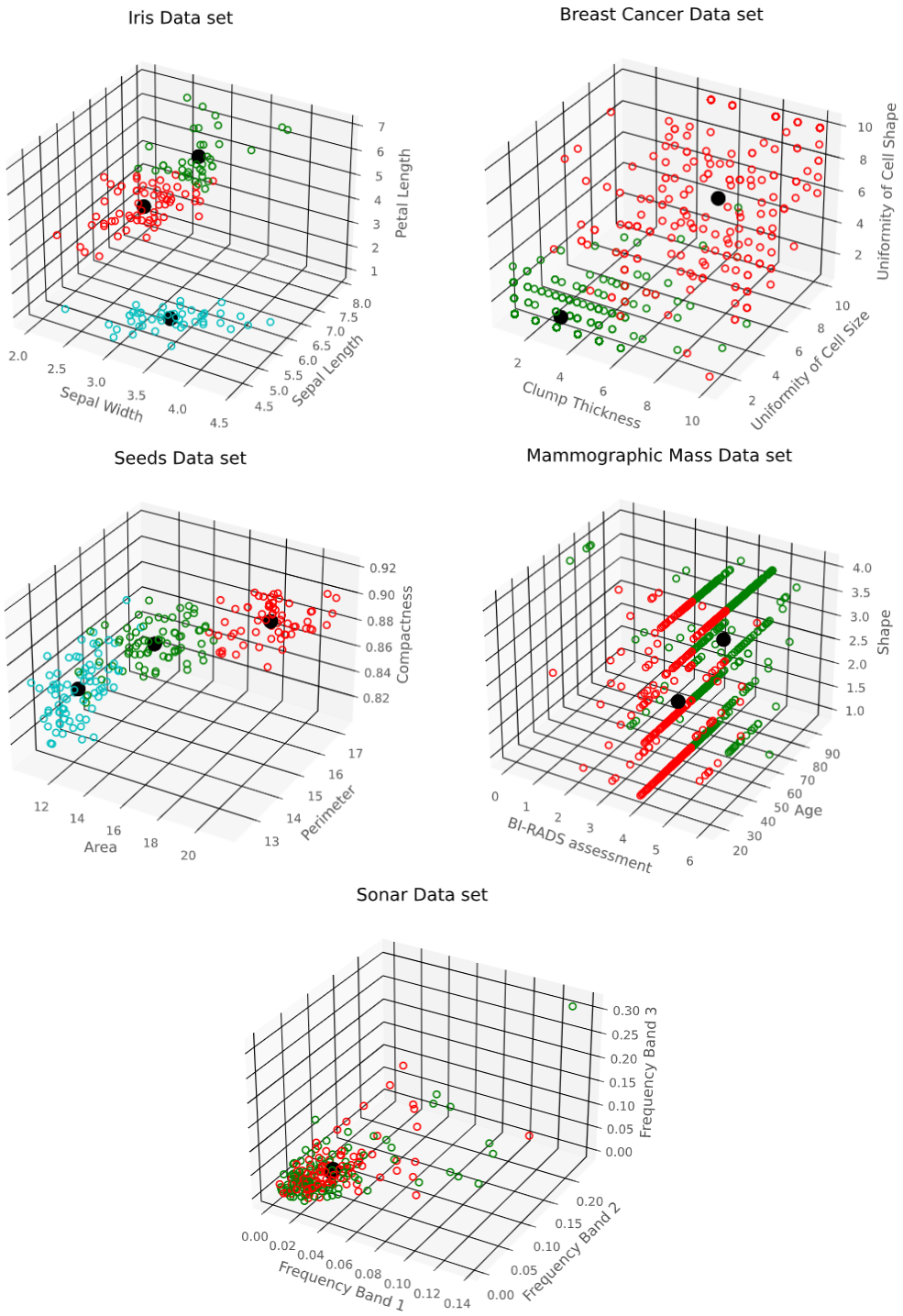

Figure 11. Visualization of the cluster centers using K-Means on experimental data sets.

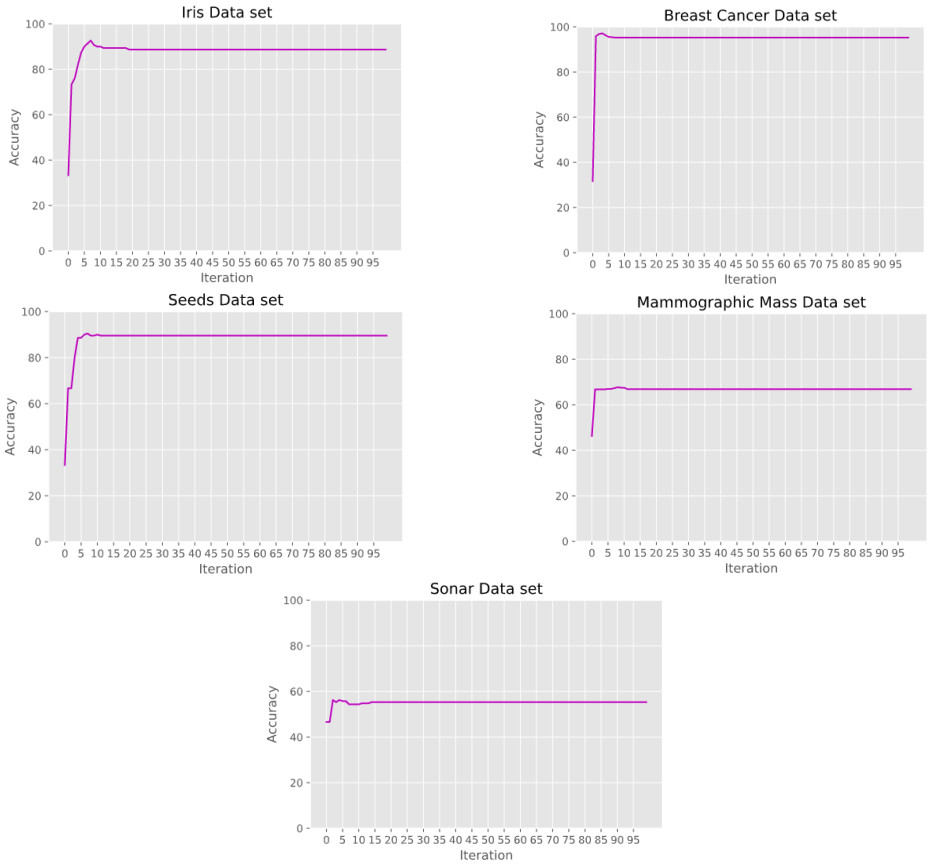

Figure 12. Accuracy using FCM on the experimental data sets.

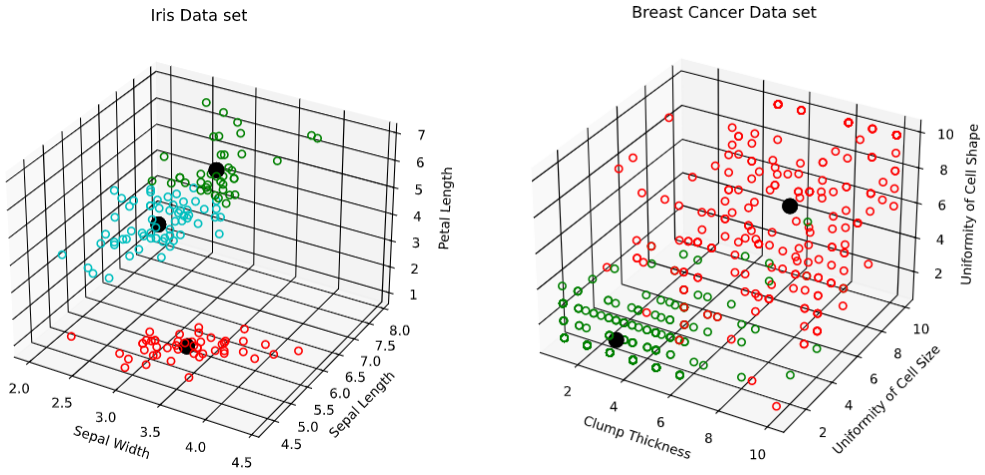

Figure 13. *Cont.*

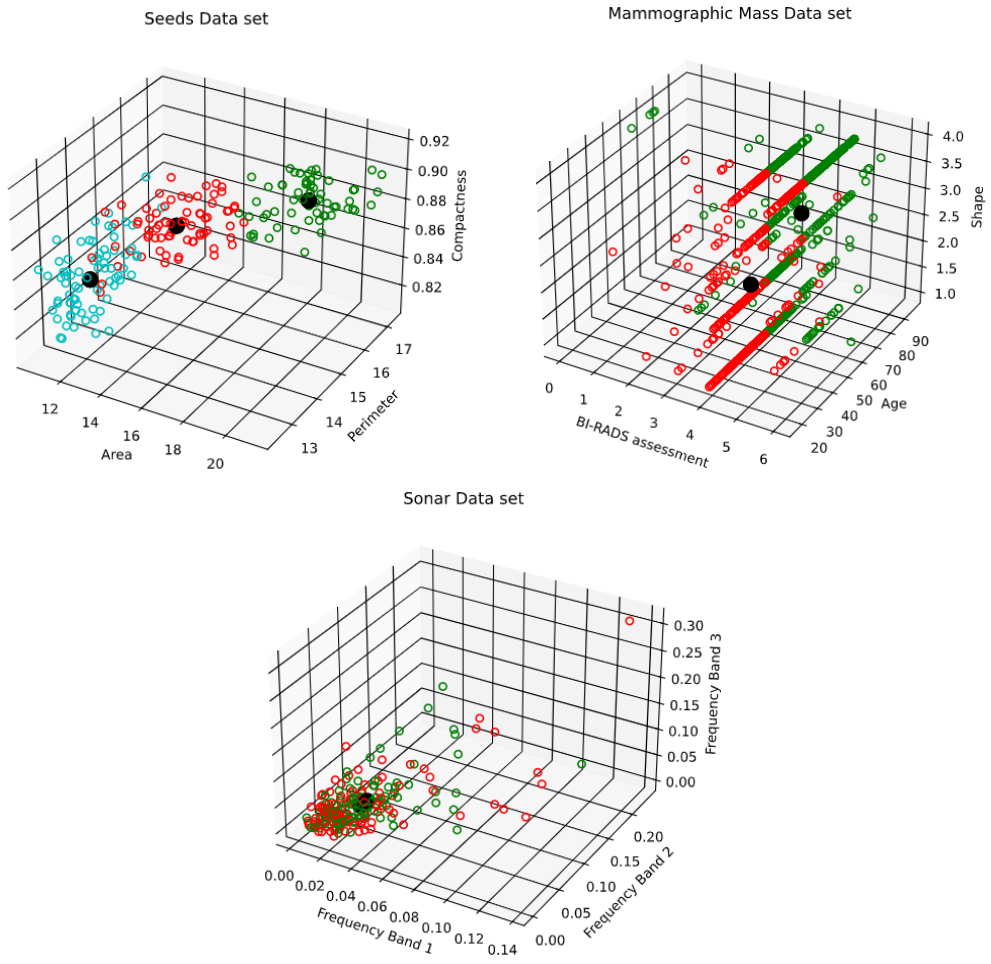

Figure 13. Visualization of the cluster centers using FCM on experimental data sets.

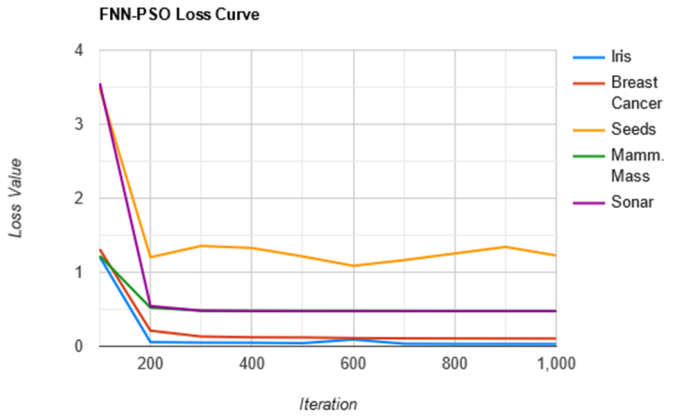

Figure 14. Loss curves of FNN-PSO algorithm for each data set over 1000 iterations.

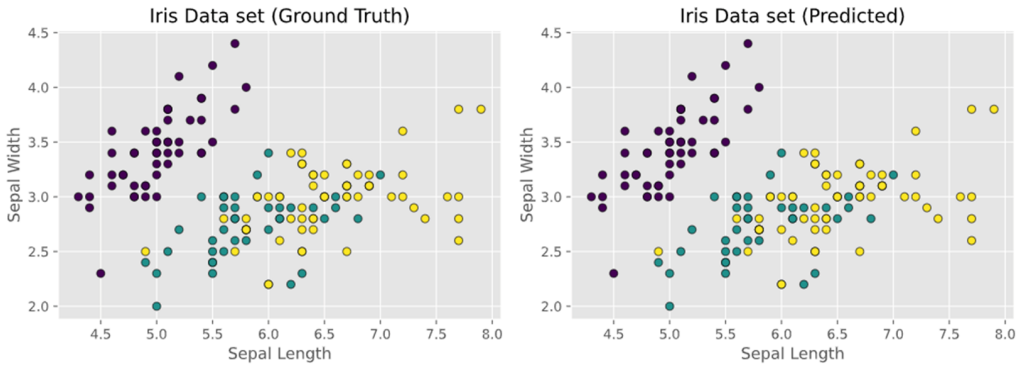

Figure 15. Ground truth vs. predicted labels from FNN-PSO algorithm on Iris.

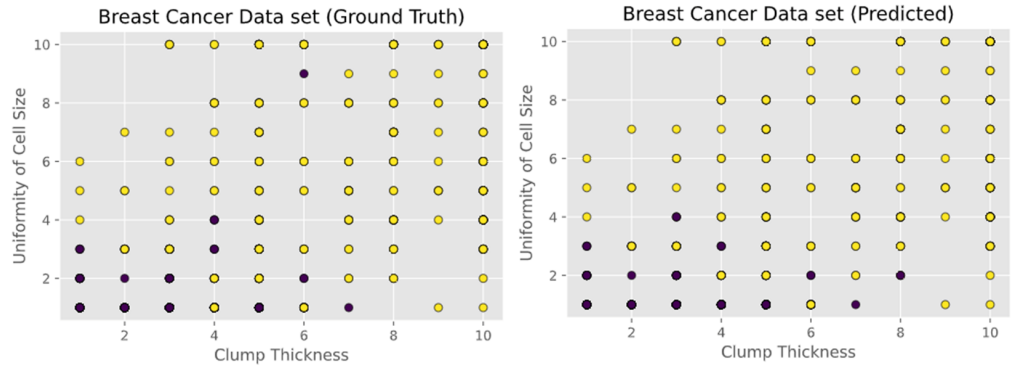

Figure 16. Ground truth vs. predicted labels from FNN-PSO algorithm on Breast Cancer.

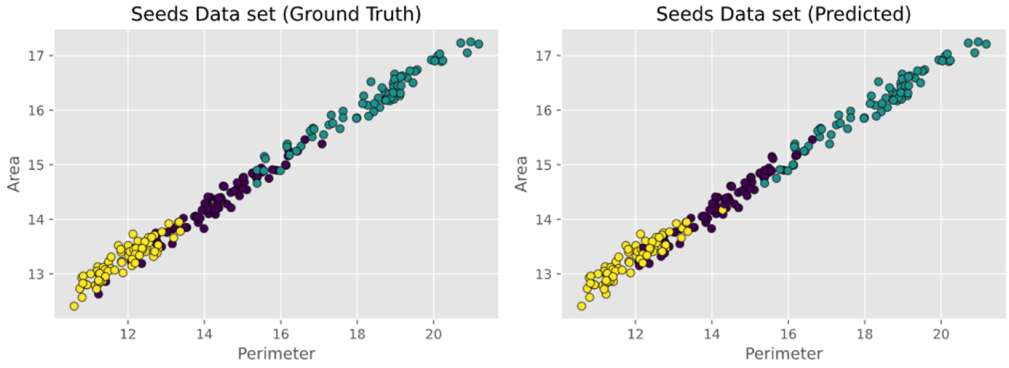

Figure 17. Ground truth vs. predicted labels from FNN-PSO algorithm on Seeds.

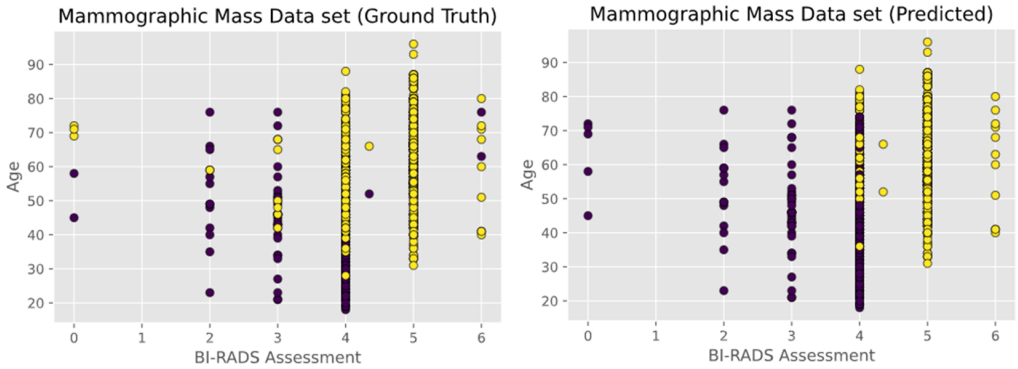

Figure 18. Ground truth vs. predicted labels from FNN-PSO algorithm on Mammographic Mass.

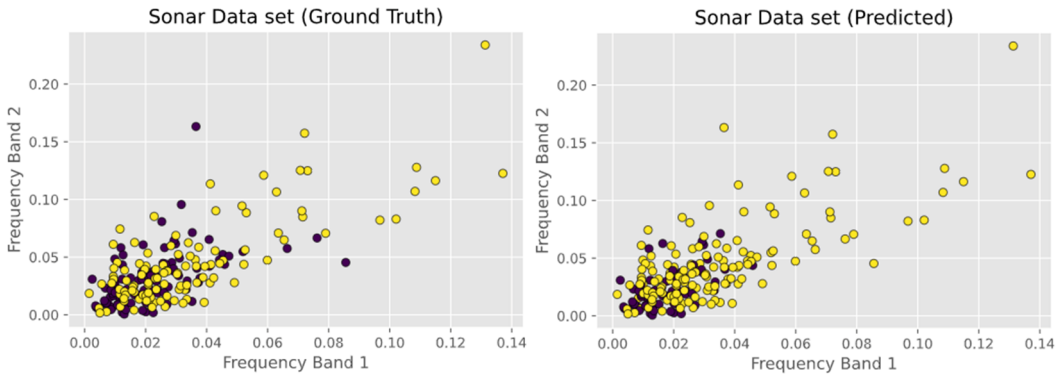

Figure 19. Ground truth vs. predicted labels from FNN-PSO algorithm on Sonar.

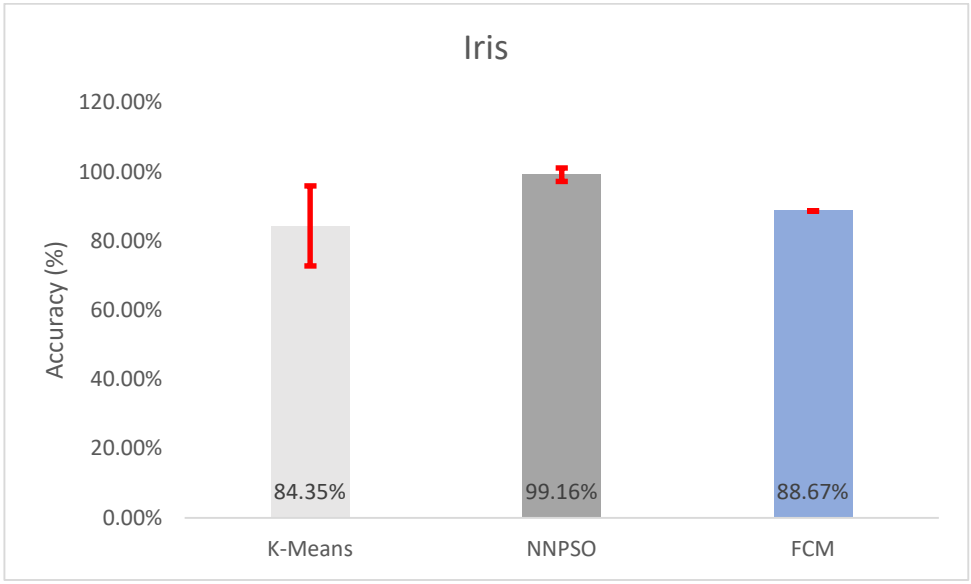

Figure 20. Accuracy of the algorithms on the Iris data set.

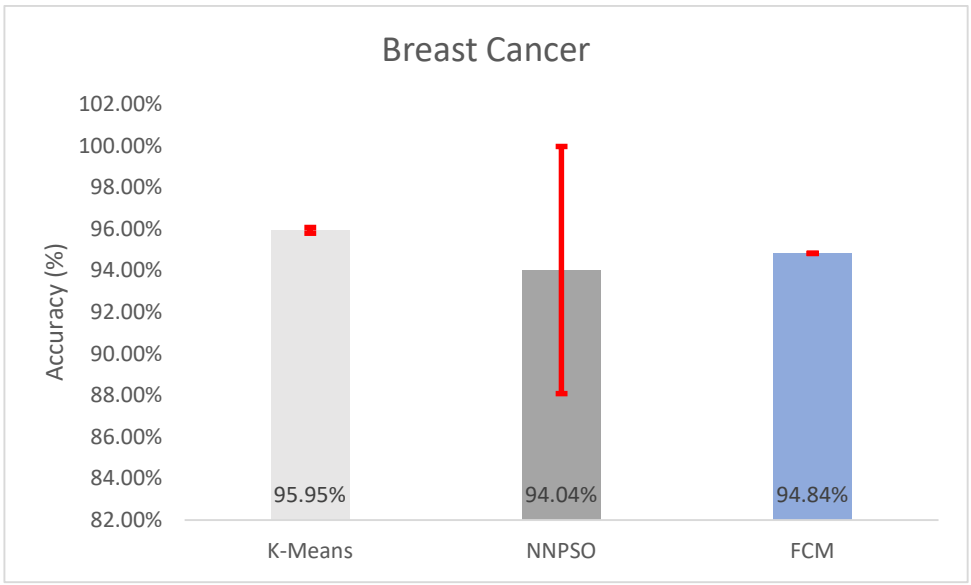

Figure 21. Accuracy of the algorithms on the Breast Cancer data set.

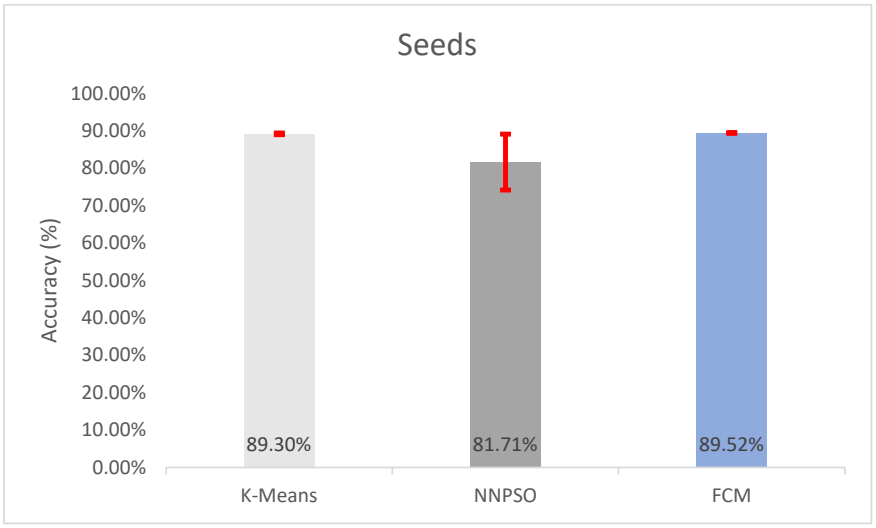

Figure 22. Accuracy of the algorithms on the Seeds dataset.

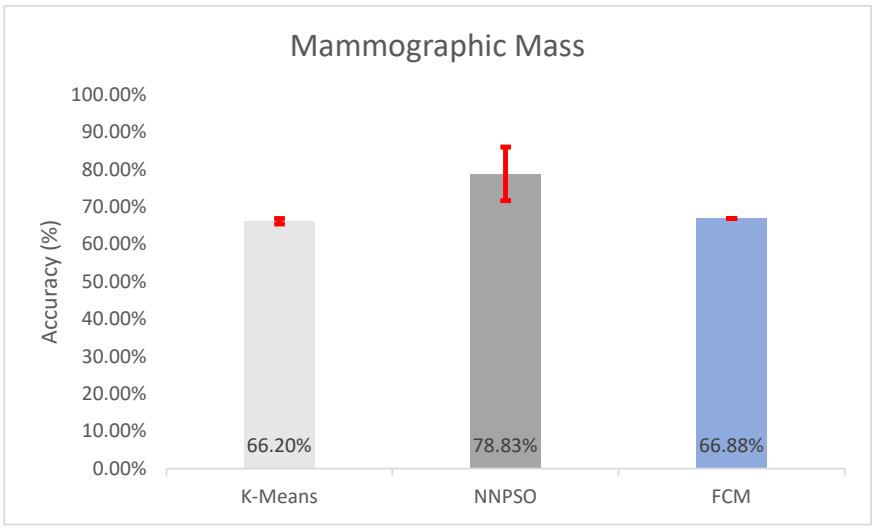

Figure 23. Accuracy of the algorithms on the Mammographic Mass data set.

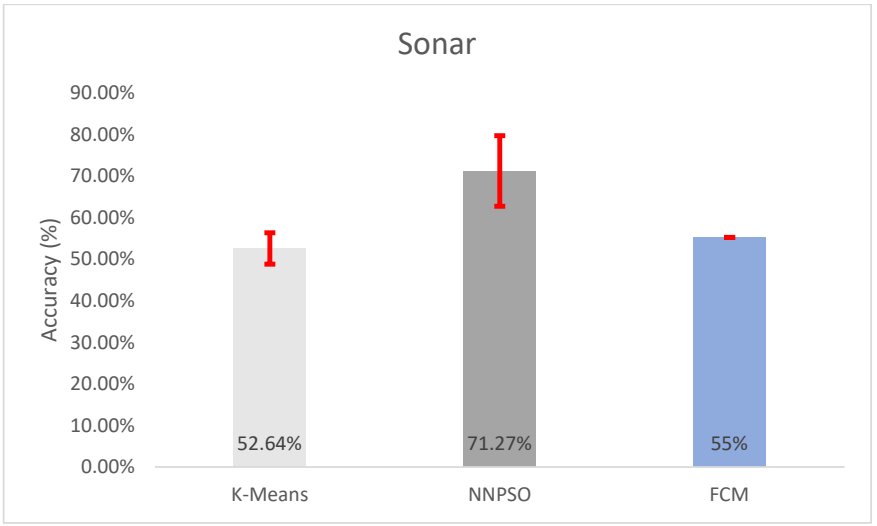

Figure 24. Accuracy of the algorithms on the Sonar data set.

The K-Means algorithm was the first algorithm that was implemented for the experiment, so it was often used as a baseline for comparing the other algorithms. While the performance did not excel in many categories, the results were all within the expected range based on results from other similar experiments and in the literature [2]. One potential problem to note is that the standard deviation for the performance metrics from the Iris results are noticeably higher than the results from the other data sets. This was found to be due to the algorithm's tendency to become stuck at local minima [1], which is detailed more in Section 5. The results from the Fuzzy C-Means algorithm have averages that are more or less similar to the results from K-Means. However, the standard deviation for those values is much lower. Despite randomly initializing the starting centroids, the model always seemed to converge to positions that effectively presented the same performance metrics. It can be inferred that the 'fuzzy' aspect of the model is what is contributing to this, allowing for any particular point to belong to each cluster centroid to a certain degree, but when it came time to classify, the highest weighted relationship with a centroid (no matter how small the difference between the various weights) was what determined which cluster the point belonged to. Fuzzy C-Means also did not seem to share the issue for the Iris data set with K-Means. This could imply that the membership matrix also allows better exploration of the data sets in certain conditions.

As for the neural network, while it does not share most of the performance metrics with the other two algorithms, accuracy and *F*-Score, which it does share, seem to have a significant improvement on most of the data sets. The Mammographic Mass and Sonar data sets in particular have a notable improvement over the two clustering algorithms. The most likely factor in this is the difference in how the algorithm determines the classification for the data points. As the neural network does not rely on the Euclidean distance to determine which points belong to which category and instead uses the approximation function generated by the network, the overlapping of the cluster areas that caused problems for the clustering algorithms is not apparent. It is also worth noting however that the performance of the Seeds data set was notably worse for the network. As shown in Figure 14, the Seeds data set struggles to converge, unlike most of the data sets. Looking at the standard deviations for the accuracies may offer some insight into this discrepancy. The network appears to have more volatility in its predictions then both clustering algorithms, and so it is possible that the setup of the nodes and activation function adds some problematic handling of the data that was not apparent in K-Means or Fuzzy C-Means.

5. Concluding Remarks and Observations from Related Studies

The goal of this work was to investigate the strengths and weaknesses of several clustering and classification algorithms to help validate the performance of each of them and to gain a better understanding of where they might be applicable. The simpler and relatively computationally cheap K-Means is well suited for general tasks but tends to fall short on more complex data sets. Fuzzy C-Means is similarly suited for general tasks and has the bonus of being able to better handle the ambiguity between clusters using a probabilistic membership matrix. Still, the limitations of utilizing a deterministic cluster update scheme can leave it falling short in data sets with overlapping clusters. Furthermore, both algorithms can end up trapped in local optima, creating an inaccurate model. The same problem may be looked at from a classification standpoint, where a feed-forward neural network optimized by PSO performs reliably on all of the data sets considered. The sensitivity to the initialization parameters implies that care must be taken to ensure that the algorithm returns meaningful results for any individual data set. However, there is no guarantee that even the most optimal set of parameters will result in a perfect classification, as that would depend on the complexity of the underlying mapping represented by the dataset.

While there is still work that can be conducted in this area, the experiments done in this paper have achieved some of the insights that were hoped for. While the clustering data sets performed admirably on some data sets, others were only slightly better than guesswork.

This shows that while spatially grouping data can be adequate in some situations, it can only go so far. Furthermore, the improved exploration provided by the "Fuzzy" aspects of FCM did allow it to perform more consistently and even better in some situations than K-Means. Still, this strength did not allow it to overcome the fundamental weakness in spatially grouped data with overlapping clusters. The FNN-PSO had its strengths as well, relying not on spatial clustering, but approximating an equation to represent the data set, meaning overlapping data was not an issue. It did however fall short in the complexity of the implementation, especially with regard to the Seeds data set, where even the most optimal parameters that were found did not allow it to properly approximate the data. Additionally, as with the Breast Cancer data set, it did not perform much better than the simpler clustering algorithms despite this extra complexity. Going forward, it will be important to keep these things in mind, as knowing how the data set works is just as important as finding an algorithm that works well for the data set.

Future work in this area would include moving beyond benchmarking data sets and into more complex data in order to apply some of the insights gained from these introductory experiments. Furthermore, exploring the functionality of the algorithms on a step-by-step basis could give further insights into the exact strengths and weaknesses of the algorithms and their applicability to a large variety of interesting problems.

5.1. Related Work and Performance

While the results from the experiments here have given some of the desired insights, to validate some of these results, it is productive to compare them to the results reported in other, similar experiments. Below is a list of works that cover clustering and classification tasks utilizing the same data sets investigated in this paper.

- "Comparative Analysis of K-means and K-medoids Algorithm on IRIS Data" [35], authored by Kalpit G. Soni and Atul Patel.
- "Self-organizing Maps as Substitutes for K-Means Clustering" [36], authored by Fernando Bação, Victor Lobo, and Marco Painho.
- "Comparison of Machine Learning Methods for Breast Cancer Diagnosis" [37], authored by Ebru Aydındag Bayrak, Pınar Kırcı and Tolga Ensari.
- "Ensemble Decision Tree Classifier for Breast Cancer Data" [38], authored by D. Lavanya and K. Usha Rani.
- "An Application of Deep Neural Network for Classification of Wheat Seeds" [39], authored by Ayşe Eldem.
- "Predicting breast cancer biopsy outcomes from BI-RADS findings using random forests with chi-square and MI features" [40], authored by Sheldon Williamson, K. Vijayakumar, and Vinod J. Kadam.
- "Classification of Sonar Targets Using OMKC, Genetic Algorithms and Statistical Moments" [41], authored by Mohammad Reza Mosavi, Mohammad Khishe, Ehsan Ebrahimi.

5.2. Observations on Iris

In Soni and Patel [35], the K-medoids or partitioning around medoid method was utilized for benchmarking purposes. The K-medoids method was proposed to help reduce the effects of outliers in the K-Means algorithm. In this technique, a set number of random points is chosen before the distances between the data points, and the centroids are calculated. The results of clustering accuracy vs. true classes for the K-means algorithm was 88.7%, and for K-medoids, it was 92%. Bação, Lobo and Painho [36] investigated self-organizing maps (SOM) on the Iris data set. Self-organizing maps were utilized here in a clustering context, and function similar to K-means in how the data patterns were mapped to neurons in an n-dimensional grid and were then grouped based on distance in the output space. The results from this study show that self-organizing maps can perform well and are less prone to local optima than K-means. Essentially, this method allows earlier exploration of the search space and can avoid issues that K-means may have where the data have multiple optima.

5.3. Observations on Breast Cancer

Bayrak, Kirci and Ensari [37] cover the utilization of support vector machines and artificial neural networks on the Wisconsin Breast Cancer (Original) data set. Support vector machines are often used for pattern recognition problems and function by separating the classes with a hyperplane consisting of support vectors of critical samples from each of the classes. The results for the best implementation of a SVM put the classification accuracy at about 96.9957%. In Lavanya and Rani [38], the classification and regression trees (CART) method was tested on the Breast Cancer (Original) data set. This method uses recursive partitioning in a regression tree algorithm hybridized with bagging, a bootstrap aggregation method. Based on the experiments, this hybrid method was able to give a classification accuracy of about 97.85%.

5.4. Observations on Seeds

In Eldem [39], a deep neural network is utilized to test the effectiveness of a model trained on the Seeds data set. For this experiment, both the base data set and a synthetic data set were utilized to test the performance of the model. Multiple models were trained using a combination of these two data sets. First off, a model with only the base data set was trained on a 70–30 split, one trained was on the synthetic data set and tested on the base data set, and finally, a model utilizing a combination of both the base and synthetic data sets combined and utilizing a 70–30 train–test split was determined. The model utilizes categorical cross entropy as its loss function. The results show a 100% success rate across its tests, including the tests utilizing the synthetic data set.

5.5. Observations on Mammographic Mass

Williamson, Vijayakumar and Kadam [40] investigate the performance of the random forest algorithm on the Mammographic Mass data set. Before the model was run, multiple setup steps were utilized. First, missing value handling was completed using a K-nearest neighbors implementation; then, the values of the data set were standardized using min-max normalization, and finally, the chi-square test of independence was used to determine which attributes should be used for testing. After these pre-processing steps, a classification model using the random forest algorithm was created. With the proposed method, an accuracy of 84.7% was recorded.

5.6. Observations on Sonar

Mosavi, Khishe and Ebrahimi [41] utilized the Sonar data set to benchmark performance of a variety of online multi kernel classifier (OMKC) algorithms. In essence, the functionality of the algorithms proposed for this paper are based on the idea of training the multi kernel classifier and optimizing the linear combination simultaneously. The proposed variations in structure serve to improve both the accuracy and efficiency of operation and utilize stochastic methods for updating the kernel weights. The results of these various models have ranges in accuracy between 61.11% and 79.95%, with the best performance belonging to the implementation utilizing the stochastic selection of the kernels and the stochastic updating of the kernel weights.

While the methods used vary widely, the range of values produced from these works are in range with the ones produced by the experiments in this paper. For reference, one may also look at [42–45] as implementations of a related nature for benchmarking classification and clustering jobs.

Author Contributions: Conceptualization, A.P. and S.S.; methodology, A.P. and S.S.; formal analysis, A.P. and S.S.; investigation, A.P. and S.S.; resources, S.S.; writing—original draft preparation, A.P.; writing—review and editing, A.P. and S.S.; visualization, A.P. and S.S.; supervision, S.S. Both authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The publicly available datasets analyzed in this work are accessible from UCI Machine learning Repository at https://archive.ics.uci.edu/ml/index.php (accessed on 10 February 2021) and the scikit-learn library at https://scikit-learn.org/stable/ (accessed on 10 February 2021).

Acknowledgments: The authors would like to express their appreciation to the colleagues who provided valuable input during the preparation of this manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ahmadyfard, A.; Modares, H. Combining PSO and k-means to enhance data clustering. In Proceedings of the 2008 International Symposium on Telecommunications, Tehran, Iran, 27–28 August 2008; pp. 688–691.
2. Sengupta, S.; Basak, S.; Peters, R.A. Data clustering using a hybrid of fuzzy c-means and quantum-behaved particle swarm optimization. In Proceedings of the 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 8–10 January 2018; pp. 137–142.
3. Sengupta, S.; Basak, S.; Saikia, P.; Paul, S.; Tsalavoutis, V.; Atiah, F.; Ravi, V.; Peters, A. A review of deep learning with special emphasis on architectures, applications and recent trends. *Knowl. Based Syst.* **2020**, *194*, 105596. [CrossRef]
4. Merwe, D.W.; Engelbrecht, A.P. Data Clustering Using Particle Swarm Optimization. In Proceedings of the 2003 Congress on Evolutionary Computation, Canberra, Australia, 8–12 December 2003; pp. 215–220.
5. MacQueen, J. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, 21 June–18 July 1965; Volume 1, pp. 281–297.
6. Tarkhaneh, O.; Shen, H. Training of feedforward neural networks for data classification using hybrid particle swarm optimization, Mantegna Lévy flight and neighborhood search. *Heliyon* **2019**, *5*, e01275. [CrossRef]
7. Zhang, C.; Shao, H. Particle Swarm Optimisation in Feedforward Neural Network. In *Artificial Neural Networks in Medicine and Biology. Perspectives in Neural Computing*; Malmgren, H., Borga, M., Niklasson, L., Eds.; Springer: London, UK, 2000. [CrossRef]
8. Karaboga, D.; Akay, B.; Ozturk, C. Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks. In *Modeling Decisions for Artificial Intelligence*; Torra, V., Narukawa, Y., Yoshida, Y., Eds.; MDAI 2007. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2017; Volume 4617. [CrossRef]
9. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
10. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
11. Sengupta, S.; Basak, S.; Peters, R.A. QDDS: A Novel Quantum Swarm Algorithm Inspired by a Double Dirac Delta Potential. In Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bangalore, India, 18–21 November 2018; pp. 704–711. [CrossRef]
12. Yang, X.S.; Deb, S. Cuckoo Search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
13. Colorni, A.; Dorigo, M.; Maniezzo, V. Distributed Optimization by Ant Colonies. In *Actes de la Première Conférence Européenne sur la vie Artificielle, Paris, France*; Elsevier Publishing: Amsterdam, The Netherlands, 1991; pp. 134–142.
14. Sengupta, S.; Basak, S.; Peters, R.A., II. Chaotic Quantum Double Delta Swarm Algorithm Using Chebyshev Maps: Theoretical Foundations, Performance Analyses and Convergence Issues. *J. Sens. Actuator Netw.* **2019**, *8*, 9. [CrossRef]
15. Sha, D.; Lin, H.-H. A multi-objective PSO for job-shop scheduling problems. *Expert Syst. Appl.* **2010**, *37*, 1065–1070. [CrossRef]
16. Chen, C.; Zhou, K. Application of Artificial Bee Colony Algorithm in Vehicle Routing Problem with Time Windows. In Proceedings of the 2018 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC), Xi'an, China, 15–17 August 2018; pp. 781–785. [CrossRef]
17. Marinakis, Y.; Marinaki, M.; Migdalas, A. Particle Swarm Optimization for the Vehicle Routing Problem: A Survey and a Comparative Analysis. In *Handbook of Heuristics*; Martí, R., Pardalos, P., Resende, M., Eds.; Springer: Cham, Switzerland, 2018.
18. Sengupta, S.; Basak, S. Computationally efficient low-pass FIR filter design using Cuckoo Search with adaptive Levy step size. In Proceedings of the 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), Jalgaon, India, 22–24 December 2016; pp. 324–329. [CrossRef]
19. Dhabal, S.; Sengupta, S. Efficient design of high pass FIR filter using quantum-behaved particle swarm optimization with weighted mean best position. In Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT), Hooghly, India, 7–8 January 2015; pp. 1–6. [CrossRef]
20. Zhu, H.; Wang, Y.; Wang, K.; Chen, Y. Particle Swarm Optimization (PSO) for the constrained portfolio optimization problem. *Expert Syst. Appl.* **2011**, *38*, 10161–10169. [CrossRef]
21. Kumar, D.; Mishra, K. Portfolio optimization using novel co-variance guided Artificial Bee Colony algorithm. *Swarm Evol. Comput.* **2017**, *33*, 119–130. [CrossRef]

22. Basak, S.; Sun, F.; Sengupta, S.; Dubey, A. Data-Driven Optimization of Public Transit Schedule. In *Big Data Analytics*; Madria, S., Fournier-Viger, P., Chaudhary, S., Reddy, P., Eds.; BDA 2019. Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2019; Volume 11932. [CrossRef]
23. Majhi, S.K.; Biswal, S. Optimal cluster analysis using hybrid K-Means and Ant Lion Optimizer. *Karbala Int. J. Mod. Sci.* **2018**, *4*, 347–360. [CrossRef]
24. K-Means Advantages and Disadvantages | Clustering in Machine Learning. 2021. Available online: https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages (accessed on 10 February 2021).
25. Nayak, J.; Naik, B.; Behera, H.S. Fuzzy C-Means (FCM) Clustering Algorithm: A Decade Review from 2000 to 2014. *Blockchain Technol. Innov. Bus. Process.* **2015**, *2*, 133–149.
26. Bezdek, J.C. Modified Objective Function Algorithms. In *Pattern Recognition with Fuzzy Objective Function Algorithms*; Springer US Science & Business Media: New York, NY, USA, 1981; pp. 155–201.
27. Torra, V. On Fuzzy c -Means and Membership Based Clustering. In *International Work-Conference on Artificial Neural Networks*; Springer: Cham, Switzerland, 2015; pp. 597–607.
28. Zhang, J.; Ma, Z. Hybrid Fuzzy Clustering Method Based on FCM and Enhanced Logarithmical PSO (ELPSO). *Comput. Intell. Neurosci.* **2020**, *2020*, 1–12. [CrossRef] [PubMed]
29. Feng, Y.; Lu, H.; Xie, W.; Yin, H.; Bai, J. An Improved Fuzzy C-Means Clustering Algorithm Based on Multi-chain Quantum Bee Colony Optimization. *Wirel. Pers. Commun.* **2017**, *102*, 1421–1441. [CrossRef]
30. Figueiredo, E.M.; Ludermir, T. Investigating the use of alternative topologies on performance of the PSO-ELM. *Neurocomputing* **2014**, *127*, 4–12. [CrossRef]
31. Sengupta, S.; Basak, S.; Peters, R. Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives. *Mach. Learn. Knowl. Extr.* **2018**, *1*, 10. [CrossRef]
32. UCI Machine Learning Repository. Available online: http://archive.ics.uci.edu/ml/ (accessed on 10 February 2021).
33. Scikit-learn Library. Available online: https://scikit-learn.org/stable/ (accessed on 10 February 2021).
34. Patel, G.K.; Dabhi, V.K.; Prajapati, H.B. Clustering Using a Combination of Particle Swarm Optimization and K-means. *J. Intell. Syst.* **2016**, *26*, 457–469. [CrossRef]
35. Soni, K.G.; Patel, A. Comparative Analysis of K-means and K-medoids Algorithm on IRIS Data. *Int. J. Comput. Intell. Res.* **2017**, *13*, 899–906.
36. Bação, F.; Lobo, V.; Painho, M. Self-organizing Maps as Substitutes for K-Means Clustering. In *Computational Science–ICCS 2006*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 476–483. [CrossRef]
37. Bayrak, E.A.; Kirci, P.; Ensari, T. Comparison of Machine Learning Methods for Breast Cancer Diagnosis. In Proceedings of the 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT), Istanbul, Turkey, 24–26 April 2019. [CrossRef]
38. Lavanya, D.; Rani, K.U. Ensemble Decision Making System for Breast Cancer Data. *Int. J. Comput. Appl.* **2012**, *51*, 19–23. [CrossRef]
39. Eldem, A. An Application of Deep Neural Network for Classification of Wheat Seeds. *Avrupa Bilim Ve Teknol. Derg.* **2020**, *19*, 213–220.
40. Williamson, S.; Vijayakumar, K.; Kadam, V.J. Predicting breast cancer biopsy outcomes from BI-RADS findings using random forests with chi-square and MI features. *Multimed. Tools Appl.* **2021**, 1–21. [CrossRef]
41. Mosavi, M.R.; Khishe, M.; Ebrahimi, E. Classification of sonar targets using OMKC, genetic algorithms and statistical moments. *J. Adv. Comput. Res.* **2016**, *7*, 143–156.
42. Hanif, M. Parallelized PSO Clustering; GitHub Repository. 2020. Available online: https://github.com/ms03831/parallelized-PSO-clustering (accessed on 10 February 2021).
43. Shriti, K. Fuzzy C Means Clustering; GitHub Repository. 2020. Available online: https://github.com/ShristiK/Fuzzy-C-Means-Clustering (accessed on 10 February 2021).
44. Ahmad, Z. Train Neural Network (Numpy)-Particle Swarm Optimization (PSO). 2020. Available online: https://medium.com/@zeeshanahmad10809/train-neural-network-numpy-particle-swarm-optimization-pso-93f289fc8a8e (accessed on 10 February 2021).
45. Prateekk94. Fuzzy C-Means Clustering on Iris Dataset. 2020. Available online: https://www.kaggle.com/prateekk94/fuzzy-c-means-clustering-on-iris-dataset (accessed on 10 February 2021).