Swarm Based Clustering Methods Using Ant Colony Optimization and Particle Swarm Optimization

Aiden Bickford aidan.bickford@gmail.com
Nate Norberg natenorberg@gmail.com

Abstract

This paper compares traditional clustering algorithms to swarm based algorithms on a clustering problem. The traditional algorithms used are the k-Means algorithm and a competitive neural network. The swarm based algorithms we are comparing are an adaptation of Any Colony Optimization (ACO) and Particle Swarm Optimization (PCO). For this paper, we created an implementation of these algorithms in Java and evaluated them on various data sets.

1. Introduction

Clustering is a common and valuable problem to solve. It is similar to classification where an algorithm learns to distinguish between a set of different classes or *labels* based on the characteristics of the data sets. With classification, the algorithms are trained with data sets that give the correct class label. This is called *supervised learning* After the algorithm is trained, it can predict future classes based off of what it has already learned.

The difference between classification and clustering is that clustering is an *unsupervised learning* problem. This means that there are no pre-specified class labels for the algorithms to learn. The goal of clustering is to find elements that are similar and group them together into their own class. The effectiveness of the clustering is measured by a combination of measuring how close together elements of the same cluster are (known as *intra-cluster distance*) and how far apart elements of one cluster are from the neighboring clusters, (known as *inter-cluster distance*).

2. Background

2.1 K-Means

K-means is a fairly simple method for clustering. It takes an iterative approach to find the centers of the clusters. To begin, the algorithm randomly generates k points in n dimensional space to be the initial centers of the clusters, where n is the number of features in the input data. From here, each data point is associated with the centroid that it is closest to. Once all of the points have been assigned to a cluster, the centers are recalculated to be the center of gravity for the cluster. To find the new center, all the points that are associated as part of that cluster are averaged and the result becomes the new centroid. From there, the process repeats and each point is assigned to the closest centroid with the new value and the centroid coordinates are updated again. This process is repeated until none of the

points change to be part of a different cluster. At this point, the algorithm is said to be converged.

2.2 Competitive Learning

The competitive learning algorithm clusters data with the use of a neural network. This network is very simple, compared to other neural networks such as Multi Layer Perceptrons. It consists of an input layer and an output layer with each node of the input layer connected to each node of the output layer. Each edge in this graph has a weight value, which could also be treated as a coordinate in that specific dimension. All of these weights are randomly assigned at the start of the algorithm. Therefore, the network starts out by representing a set number of randomly generated cluster centers, much like k-means.

This network learns as data is passed through it. Each node in the output layer takes the squared difference between each of the input edges and the associated weights and sums it together. This value represents the difference between the input and the center of the cluster. The output node that has the lowest distance value is deemed the winner and the input point is associated with that cluster. This network is a *winner-take-all* algorithm which means that only one output neuron can fire. Also, only the weights into this node are updated after the network is run. The weights are updated by moving the weights closer to the input values.

$$\Delta w_i = \eta \times (x_i - w_i)$$

This serves to minimize the distance between that point and the center in future runs while also moving the center farther away from less similar points.

2.3 Ant Colony Optimization

Ant colonies have been an inspiring source for novel approaches to problem solving. Models of their complex behavior and organization has shown to be a valuable approach to optimization problems. Using ant colonies to cluster data was inspired by the movement and organization of corpses within the ant colonies, and is known as cemetery organization. The data points are randomly placed on a two dimensional grid, the ants then traverse the grid at random picking up and dropping the data points with a calculated probability. The probability of an ant picking up or dropping an item is based on the similarity of the other data points surrounding it, such that the probability of a point being put down increases as it is brought closer to similar points. As this process is continued the data points migrate with the help of the ants towards clusters of related points.

2.4 Particle Swarm Optimization

Particle swarm algorithms are based off of the movement of birds in flocks, and fish in schools. A population of particles is created where each particle represents a potential solution. The particles then move towards an improved solution through communication with the rest of the swarm of particles. This allows the particles to converge on points informed by their personal best, and the performance of the other particles in the swarm. The algorithm implemented in these experiments communicated on a global level, where all the particles were influenced by the best global individual.

3. Hypothesis

4. Approach

4.1 Data Sets

The data sets used for this paper are taken from the UCI Machine Learning Repository. These sets have between 5 and 166 attributes. Because these input values vary across many different ranges, we normalized the values between 0 and 1 based off of the respective minimum and maximum values. This allows the random placement of the centers for the different clustering algorithms to more accurately represent the input space.

4.2 Ant Cemetery Organization

The ant colony clustering algorithm uses cemetery organization to cluster the sets of classification data points. The all of the data points were randomly dispersed on a two dimensional grid which was large enough so that the data points took up approximately half of the space. This size was selected after multiple trials as it allowed enough space for the ants and the particles to move, while at the same time not providing so much space that many small clusters would form.

Ants were placed on the grid at random. An ant population the approximate size of one fifth of the number of the population was used. This provided enough ants to perform the clustering while not so many that too many data points would be picked up at once, which would decrease the density calculation, and lower the probability that the ants would ever drop the point.

The ants moved at random to one of the immediate surrounding points on the grid. If the point was occupied with a datapoint, it was picked up with based on the probability of pickup and putdown at following points based on the probability of dropping the point.(Engelbrecht, 2007)

$$P_{pickup} = (\frac{\gamma_1}{\gamma_1 + \lambda})^2$$

$$P_{drop} = (\frac{lambda}{\gamma_2 + \lambda})^2$$

The term λ represents the density of points nearby which are similar to the the point in question. It is calculated measuring the distance between itself and all of the points within a determined visible range of the ant.(Engelbrecht, 2007)

$$\lambda(y_a) = max\left\{0, \frac{1}{n_{\mathcal{N}}^2} \sum_{y_b \in n_{\mathcal{N}} \times n_{\mathcal{N}}(i)} \left(1 - \frac{d(y_a, y_b)}{\gamma}\right)\right\}$$

where $d(y_a, y_b)$ represents the difference between the two points, computed using the euclidean distance. $n_{\mathcal{N}} \times n_{\mathcal{N}}(i)$ represents the visibility of the ant, or the square of area surrounding the ant that it is aware of. This area was linearly increased as the algorithm progressed from a radius of 1 to 3 or 4. (Engelbrecht, 2007)

$$Visibility_{radius} = (1 - Visibility_{radius final})\frac{t_{final} - t}{t_{final}} + Visibility_{radius final}$$

This allowed for the initial clusters to be formed using a lower visibility which minimizes computational cost, and for the refinement of the clusters towards the end of the run time. The values of γ, γ_1, and γ_2 were determined through trial and error to find the optimal values for each data set.

INSERT TABLE HERE

The clusters were removed from the mapping created by the ant using an agglomerative hierarchical clustering algorithm. The algorithm designates each point on the map as its own cluster. Then the two closest clusters are combined decreasing the number of clusters by one. This process is repeated until the target number of clusters has been found. (Handl, 2003)

4.3 Particle Swarm Optimization

4.4 Cluster Metrics

Because the goals of clustering are to minimize intra-cluster distance and maximize inter-cluster distance, we wanted a metric that could evaluate both at the same time. The metric we used for our evaluation was the Davies-Bouldin index, which is given by

$$DB = \frac{1}{n}\sum_{i=1}^{n} max_{i\neq j}(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)})$$

where σ_x is the average distance from each point in cluster x to the center of the cluster, and $d(c_i, c_j)$ is the distance between the centers of clusters i and j. Since we want our clusters to be dense and very separated, a lower Davies-Bouldin index is preferable.

5. Results

6. Discussion

7. Conclusion

References

Eberhart, R., & SHI, Y. (2007). Computational intelligence. Burlington MA: Morgan Kaufman.

Handl, J., Knowles, J. D., & Dorigo, M. (2003, December). On the Performance of Ant-based Clustering. In HIS (pp. 204-213).