An Algorithm for Clustering Animals by Species Based upon Daily Movement

David M. Curry\(^a\),* 

\(^a\)Department of Engineering Management and Systems Engineering, Missouri University of Science and Technology, Rolla, Missouri 65409

Abstract

An algorithm is presented for clustering individual animals by species based solely upon the daily movements of the individual animals. This is particularly challenging due to the highly erratic nature of the animals' movement. The variance in the scale and frequency of movement between individuals within a species is often greater than the difference between species. Existing clustering algorithms including hierarchical, \(k\)-means, and spectral were tried, but they failed to accurately distinguish between species or to cluster individuals of the same species together. Also, some of these algorithms require a priori knowledge of the number of species (clusters). The algorithm presented here addresses this problem by creating separation through a distance metric based upon ranking and then clustering based upon commonality in rankings. The algorithm performed well, demonstrating the ability to both distinguish between species and to cluster together individual of the same species. Furthermore, it assumes no a priori knowledge of the number of clusters expected.

© 2014 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/3.0/).

Keywords: clustering; sequential data clustering; algorithm; adjusted rand index; ARI

* Corresponding author. Tel.: +1-636-699-0997
E-mail address: dmcfh3@mst.edu
1. Introduction

Clustering, since it lacks reinforcement, is often difficult to apply to problems that may otherwise appear trivial. The problem of identifying animals as belonging to the same species or different species based upon their movement is one such problem. Two questions are at the forefront:

- Is this innately possible, and if so
- Can these movements be differentiated by an unsupervised method?

First, identifying of the pertinent aspects of animal movement is in order. Animals often stay in a local area for a relatively long period of time and then move to another area that suits them. This behavior creates largely similar data where most distances are small and the direction traveled varies widely for local movement and the distance when moving to another area may be many orders of magnitude greater. Because these characteristics of the animals pose special challenges, a new algorithm is proposed to specifically address these difficulties.

The data used validate the algorithm is from a public repository of animal tracking data called the “Movebank Data Repository”. The specific tracking data utilized was chosen based species with at least four individuals having at least 250 days of data apiece. Six such species were found and the first four animals from each species meeting the minimum data collection period were used. Therefore a total of twenty-four animals where used, four each from the six species. The species are:

- European gadwall [1][2],
- Montagu's harrier [3][4],
- European mallard [5][6],
- Cory's shearwater [7][8],
- Turkey vulture [9][10], and
- Burchell's zebra [11][12].

These animals’ movements are shown in Fig. 1 by species. Fig. 2 illustrates the difficulty posed by the large range of distance values combined with most values being densely packed near zero when using methods such as k-means.

![Fig. 1. Distance and bearing data over a 251 day period for all 24 animals by species.](image)
2. Data

The raw data from the Movebank Data Repository is formatted as comma-separated values (CSV) files. Each file potentially comes from a different original source and the specific formatting varies widely. Despite the differences, each file is usually composed of all the individuals in the particular study with a unique ID (tag) so that they can be differentiated from the other individuals. The amount of data for an individual tends to vary greatly even within the same study. Furthermore, the data is in degrees of latitude and longitude, therefore it was necessary to convert the data. The haversine formula was chosen because it is simple to implement and is accurate to within 0.5%.

The proposed algorithm expects the data to be in the form of an $m \times n$ matrix $X$, where $x_{i,j}$ is the weighted ordered pair $(w_1d, w_2\theta)$, $w_1$ is the distance weighting, $w_2$ is the bearing weighting, and $d$ and $\theta$ are the distance and bearing for the $i^{th}$ animal on the $j^{th}$ day.

3. Algorithm

The proposed algorithm seeks to create separation between the points that are near one another and to compress the distance of the points that are far away. It is hoped that this will counter the difficulties due to the large difference in scale found in the data. Each animal’s data will be treated as a sequence of distance and bearing values. The inspiration for treating this data sequentially came from Xu and Wunsch [13] and the algorithm also draws from Guralnik and Karypis [14] in the sense that it uses similarity of subsequences as a distance metric. It is hoped that by allowing individual animals to be clustered based on subsequences, differences in individual behavior within each species will be countered.

The algorithm begins by calculating the minimum distance between any two $q$-dimensional points between each row creating an $m \times m$ matrix $D$ of these distances where $D_{i,j}$ is the minimum distance between rows $i$ and $j$. Then each row in $D$ is sorted in ascending order and $S_{i,j}$ is set to the final ranking of each element $D_{i,j}$ after sorting. Next, the matrix of cycle lengths (what will be used as the final distance metric), $D'$ is calculated. This is done by adding...
the position of $S_{i,j}$ in row $i$ to the position of $S_{i,j}$ in row $S_{i,1}$ and subtracting two. This yields the number of elements in the cycle from $S_{i,j}$ back to itself. This is illustrated in Fig 3. Therefore, each $D'_{i,j}$ is set to the cycle length of $S_{i,j}$. Treating each row of $D'$ as a set, the largest common, non-trivial subsets formed by the first element and working outward are then found between each of the rows and a cluster $C_i$ is formed by each grouping of such rows (but necessarily by the elements in common between the rows), where $i$ is the number of the cluster formed. Finally, each row that is not a part of a cluster is added to the cluster or first element of an unclustered row that has the shortest cycle length to it.

![Diagram](image)

Fig. 3. An illustration of how the sorted matrix $S$ is formed from the distance matrix $D$. An example demonstrating the cycle length calculation for an element of the sorted distance matrix $D'$ is also shown.

4. Results

The proposed algorithm was implemented using MATLAB® and compared against a spectral algorithm [15] and MATLAB’s included implementations of $k$-means and hierarchical clustering using the Adjusted Rand Index (ARI) [16]. The distance was given a weighting of ten times that of bearing ($w_1=10$, $w_2=1$) to compensate for the difference in magnitude for the majority of the data. A time period of thirty days ($q=30$) was used for the testing. As expected, the proposed algorithm performed slowly compared to the others, requiring 136.5 seconds whereas spectral clustering took 0.3 and the others less than 0.1 seconds. However, the ARI was significantly better despite the fact that the other algorithms were given the true number of clusters a priori. The ARI for the proposed algorithm was 0.436 compared to 0.203 for spectral, and 0.033 for $k$-means and hierarchical. Also, the number of clusters calculated by the proposed algorithm was eight and compares well to the true number of clusters which is six.
Algorithm 1 Proposed Algorithm, Part 1

Require: $X$ \hspace{0.5cm} \triangleright m \times n$ matrix of tracking data

Require: $q$ \hspace{0.5cm} \triangleright$Time period in days

1: procedure CLUSTER($X, q$) \hspace{0.5cm} \triangleright Cluster data in $X$

2: $D \leftarrow 0$ \hspace{0.5cm} \triangleright $m \times m$ minimum distance matrix

3: for $i \leftarrow 1, m - 1$ do \hspace{0.5cm} \triangleright Compute matrix $D$

4: for $j \leftarrow i + 1, m$ do \hspace{0.5cm} \triangleright $D$ is symmetric so do upper half

5: \hspace{1cm} $p_1 \leftarrow (X_{i,1}, X_{i,2}, \ldots, X_{i,q+1})$ \hspace{0.5cm} \triangleright First point in row $i$

6: \hspace{1cm} $p_2 \leftarrow (X_{j,1}, X_{j,2}, \ldots, X_{j,q+1})$ \hspace{0.5cm} \triangleright First point in row $j$

7: \hspace{1cm} $D_{i,j} \leftarrow \|p_2 - p_1\|$ \hspace{0.5cm} \triangleright Distance between points on rows $i$ and $j$

8: for $k \leftarrow 1, n - q$ do \hspace{0.5cm} \triangleright Find minimum distance

9: \hspace{1cm} for $l \leftarrow 2, n - q$ do \hspace{0.5cm} \triangleright Already have the first point

10: \hspace{2cm} $p_1 \leftarrow (X_{i,k}, X_{i,k+1}, \ldots, X_{i,k+q})$ \hspace{0.5cm} \triangleright Point in row $i$

11: \hspace{2cm} $p_2 \leftarrow (X_{j,l}, X_{j,l+1}, \ldots, X_{j,l+q})$ \hspace{0.5cm} \triangleright Point in row $j$

12: \hspace{2cm} if $\|p_2 - p_1\| < D_{i,j}$ then

13: \hspace{3cm} $D_{i,j} \leftarrow \|p_2 - p_1\|$ \hspace{0.5cm} \triangleright If $p_1, p_2$ are closer than $D_{i,j}$

14: \hspace{2cm} end if

15: \hspace{1cm} end for

16: \hspace{1cm} end for

17: $D_{j,i} \leftarrow D_{i,j}$ \hspace{0.5cm} \triangleright Symmetric

18: end for

19: for $i \leftarrow 1, m$ do \hspace{0.5cm} \triangleright Compute matrix $S$ of sorted indexes

20: for $j \leftarrow 1, m$ do

21: \hspace{1cm} $k \leftarrow$ ranking of $D_{i,j}$ in ascending order sort of row $i$

22: \hspace{1cm} $S_{i,k} \leftarrow j$ \hspace{0.5cm} \triangleright Row $j$'s rank relative to others in row $i$

23: end for

24: end for

25: for $i \leftarrow 1, m$ do \hspace{0.5cm} \triangleright Compute matrix $D'$ of cycle lengths in $S$

26: for $j \leftarrow 1, m$ do

27: \hspace{1cm} $D'_{i,j} \leftarrow$ CycleLength($S_{i,j}$)

28: end for

29: end for

30: end for
Algorithm 2 Proposed Algorithm, Part 2

31: \( C \leftarrow \emptyset \) \hspace{1cm} \triangleright \text{Set of clusters} \\
32: \( p \leftarrow 0 \) \hspace{1cm} \triangleright \text{Number of clusters} \\
33: \text{for } k \leftarrow \lfloor m/2 \rfloor, 2 \text{ do} \hspace{1cm} \triangleright \text{Exclude trivial clusterings by limiting size} \\
34: \hspace{1cm} \text{for } j \leftarrow 1, m \text{ do} \hspace{1cm} \triangleright \text{Find largest subsets in } S \\
35: \hspace{2cm} \text{for } i \leftarrow 1, m, i \neq j \text{ do} \hspace{1cm} \triangleright \text{i not already clustered} \\
36: \hspace{3cm} \text{if } i \notin C_i \forall C_i \in C \text{ then} \hspace{1cm} \triangleright \text{i also not clustered} \\
37: \hspace{4cm} \hat{U} = \{S_{i1}, S_{i2}, \ldots, S_{ik}\} \hspace{1cm} \triangleright \text{Subset of row } i \\
38: \hspace{4cm} \hat{V} = \{S_{j1}, S_{j2}, \ldots, S_{jk}\} \hspace{1cm} \triangleright \text{Subset of row } j \\
39: \hspace{4cm} \text{if } \hat{U} \neq \hat{V} \text{ then} \hspace{1cm} \triangleright \text{Cluster } i \text{ and } j \\
40: \hspace{5cm} \text{if } j \notin C_i \forall C_i \in C \text{ then} \hspace{1cm} \triangleright \text{j also not clustered} \\
41: \hspace{6cm} p \leftarrow p + 1 \hspace{1cm} \triangleright \text{Increment } p \\
42: \hspace{6cm} C_p \leftarrow \{i, j\} \hspace{1cm} \triangleright \text{Create cluster } C_p \\
43: \hspace{5cm} \text{else} \\
44: \hspace{6cm} t \leftarrow t \ni j \in C_t \hspace{1cm} \triangleright \text{Find cluster with j} \\
45: \hspace{6cm} C_t \leftarrow C_t \cup \{i\} \hspace{1cm} \triangleright \text{Add i to cluster } C_t \\
46: \hspace{4cm} \text{end if} \\
47: \hspace{3cm} \text{end if} \\
48: \hspace{2cm} \text{end for} \\
49: \hspace{1cm} \text{end for} \\
50: \text{end for} \\
51: \text{end for} \\

Algorithm 3 Proposed Algorithm, Part 3

52: \text{for } i \leftarrow 1, m, i \neq j \text{ do} \hspace{1cm} \triangleright \text{Aggregate unclustered rows} \\
53: \hspace{1cm} \text{if } i \notin C_i \forall C_i \in C \text{ then} \hspace{1cm} \triangleright \text{i not clustered} \\
54: \hspace{2cm} d \leftarrow D_{i,2} \\
55: \hspace{2cm} i' \leftarrow S_{i,2} \\
56: \hspace{1cm} \text{for } j \leftarrow 2, m \text{ do} \hspace{1cm} \triangleright \text{Find nearest row} \\
57: \hspace{2cm} \text{for } k \leftarrow 1, m \text{ do} \\
58: \hspace{3cm} \text{if } S_{k,j} = i \land D_{k,j} < d \text{ then} \\
59: \hspace{4cm} d \leftarrow D_{k,j} \\
60: \hspace{4cm} i' \leftarrow k \\
61: \hspace{3cm} \text{end if} \\
62: \hspace{2cm} \text{end for} \\
63: \hspace{1cm} \text{end for} \\
64: \hspace{1cm} \text{if } i' \notin C_i \forall C_i \in C \text{ then} \hspace{1cm} \triangleright \text{i' also not clustered} \\
65: \hspace{2cm} p \leftarrow p + 1 \hspace{1cm} \triangleright \text{Increment } p \\
66: \hspace{2cm} C_p \leftarrow \{i, i'\} \hspace{1cm} \triangleright \text{Create cluster } C_p \\
67: \hspace{2cm} \text{else} \\
68: \hspace{3cm} t \leftarrow t \ni i' \in C_t \hspace{1cm} \triangleright \text{Find cluster with i'} \\
69: \hspace{3cm} C_t \leftarrow C_t \cup \{i\} \hspace{1cm} \triangleright \text{Add i to cluster } C_t \\
70: \hspace{2cm} \text{end if} \\
71: \hspace{1cm} \text{end if} \\
72: \text{for } i \leftarrow 1, p \text{ do} \hspace{1cm} \triangleright \text{Add clusters to } C \\
73: \hspace{1cm} C \leftarrow C \cup C_i \\
74: \text{end for} \\
75: \text{return } C \hspace{1cm} \triangleright \text{The clustering of } X \\
76: \text{end procedure}
The clusters each algorithm computed and the true clustering are as follows:

- Actual clusters:
  - 1, 2, 3, 4
  - 5, 6, 7, 8
  - 9, 10, 11, 12
  - 13, 14, 15, 16
  - 17, 18, 19, 20
  - 21, 22, 23, 24

- Proposed algorithm:
  - 1, 2, 3, 11, 12
  - 5, 6, 7, 8
  - 9
  - 4, 10, 18
  - 13, 21, 23
  - 14, 15, 16
  - 19, 20
  - 22, 24

- Spectral:
  - 7, 8, 15, 16
  - 2, 3, 4, 9, 11, 12, 17, 18, 19, 20
  - 5, 6, 10, 14
  - 13, 22
  - 21, 24
  - 1, 23

- \( k \)-means and hierarchical:
  - 16
  - 6
  - 1, 2, 3, 4, 7, 9, 10, 11, 12, 13, 14, 17, 18, 19, 20, 21, 22, 23, 24
  - 8
  - 15
  - 5

5. Conclusions

The proposed algorithm is exceedingly slow compared to the others, but considerably more accurate based upon the Adjusted Rand Index. While the speed penalty appears considerable, it does not have a significant overall
impact considering that a year's worth of data took less than three minutes to cluster on a laptop. This implies that
the algorithm is fast enough for practical use given that it would take decades of data collected on a daily basis
before speed becomes an issue. The intent of this algorithm is the accurate clustering of individual animals by
species and in this aspect, the algorithm shows promise based on its high Adjusted Rand Index. Its ARI
demonstrates that the algorithm's accuracy is not attributable to random chance and that it is superior to the
conventional methods with which it was compared.

References