Ministry of Education and Science of the Russian Federation
Peter the Great St. Petersburg State Polytechnic University
Institute of Computer Sciences and Technologies
Graduate School of Cyber-Physical Systems and Control

Course Project

Pattern Recognition Methods for Object Identification
Discipline: Intellectual Computing
15 May 2017

Student Group: 13541/8

________________________ Christopher W. Blake

Professor

________________________ Kuchmin A.Y.

St. Petersburg
2017

Contents

Introduction

The task is to identify features, objects, and patterns within a set of data. The intention is to use these identified objects to later find additional features and objects across multiple datasets. As such, various pattern recognition methodologies are reviewed and compared, eventually leading to a sample program developed in C#.

A brief overview of pattern recognition theory is provided, and then the identified task-relevant category of clustering is expanded in further detail. From the sub-category of clustering, five methods are described, including their advantages and disadvantages. Afterwards, their individual relevance to solving the required task is reviewed, and one method is chosen.

Finally, a sample C# program using hierarchical clustering is developed. Using this software, a user draws shapes or symbols repeatedly on a canvas. The images are compared to each other and clusters are produced. Sample results with comments are produced, leading to a recommendation for future work and usage of hierarchical analysis in object identification.

Background

Pattern Recognition

The realm of pattern recognition is a rather mixed grouping of classification, clustering, and prediction methodologies. The general process intends to identify patterns and regular occurring structures in data. This basic process usually involves the conversion to feature vectors from the data and then further comparison of these vectors for similarities. The analysis process may be supervised or unsupervised as well statistical or non-statistical.

In some methodologies, the data and predicted patterns are further extrapolated to make predictions. Many various categories of pattern recognition exist, depending the task to be solved. A short un-exhaustive list of these categories is shown below with brief descriptions.

Pattern recognition method categories (common)

1. Classification – identification of previously identified patterns in data.
2. Clustering – discovery of new patterns in data, and then classification.
3. Ensemble Learning – generation of hypothesis by combining various methodologies.
4. Sequence Labeling, Real Numbers – prediction of real numbers (through filters)
5. Sequence Labeling, Categorical – prediction of categories
6. Regression – prediction of real numbers (numerically)

Other similar/overlapping fields include

1. Pattern Matching – exact pattern without variation (Ex: Regular Expressions)
2. Machine Learning
3. Data mining
4. Knowledge Discovery in Databases (KDD)

Feature Vector Components

1. Instance – piece of input data for which an out value is generated.
2. Vector – an array description of all known features/characteristics of the instance.
3. Feature – single descriptive item of the instance. Typical categories include:

a. Categorical – ex: male/female
b. Ordinal – ex: large/medium/small
c. Integer-Value – ex: item count
d. Real-Value – ex: measurements of blood pressure

Clustering

Clustering is the act of grouping sets of data together without prior knowledge of what groups may exist in the data. Each of these clusters of data include a common denominator, usually defined by the feature vector. The typical cluster forms are distance from a centroid, density of an area, and statistical intervals. Additionally, the degree and type of membership can vary. Finally, these clusters are identified through a multi-objective optimization problem and iterative process.

Clustering Models (common)

1. Connectivity
2. Centroid
3. Distribution – ex: Expectation-maximization algorithm
4. Density – ex: DBSCAN and OPTICS
5. Subspace - Biclustering (co-clustering)
6. Group – does not provided a refined model. (just group info)
7. Graph-based – cliques using nodes and edges

Cluster membership

A single instance of a feature vector, from the data, can belong to a cluster in multiple ways. First there is the degree of its member and second, the membership type. The degree of membership is typically referred to as soft or hard. The membership type is more complex.

Membership degree:

1. Soft membership – comparable with fuzzy logic where the object partially belongs to the cluster.
2. Hard membership – the instance is either a member or it is not a member (boolean).

Membership type

The membership type is typically further divided in a structural way. The typical types are listed below.

1. Strict partitioning – an instance must be in a cluster, but only one cluster.
2. Strict partitioning with outliers – an instance may be in one cluster or un-clustered.
3. Overlapping – an instance may belong to multiple clusters
4. Hierarchical – the instance may belong to multiple clusters, through inheritance.
5. Subspace – overlapping but only within identified subspaces of the data.

Review – Clustering Algorithms

Many different algorithms exist for clustering of data. For example, as of the writing of this report, the list of algorithms on Wikipedia has 39 different methods shown. Due to time constraints, only the most popular or recently developed methods will be explored.

Density-based spatial clustering of applications with noise (DBSCAN)

DBSCAN is a density-based algorithm developed in 1996 and expanded in 2013. In 2014, it was awarded the "Test of Time" award.

A set of points are compared by locating an initial core (A), which has a minimum number of elements connected by a radius (ε). This core then grows according to the same radius. It grows until edges are found where the new points do not connect to additional points. (Figures 1 & 2)

Like typical clustering methods, the distance between points is defined through a feature vector and associated distance function.

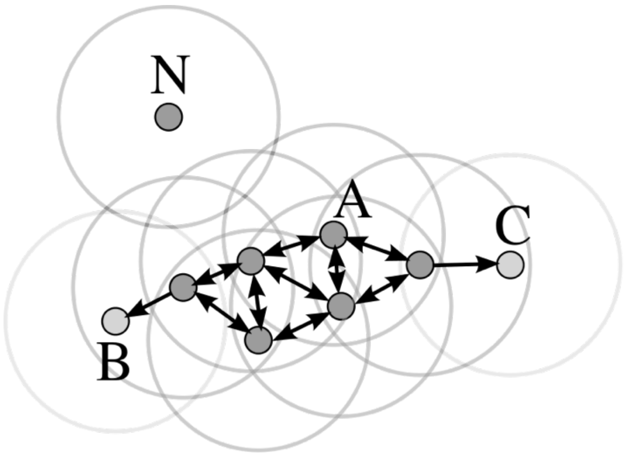


Figure 1: SBSCAN example. "A" is the core. "B" and "C" are edges. "N" is an outlier

Algorithm

1. Get input and configuration parameters
    a. Data Set
    b. ε (eps) – Distance Parameter
    c. minPts – Minimum number of points to form a "dense" region. (core)
2. Choose a random unvisited start point.
3. Retrieve the points neighbors within the distance ε.
4. Check if there are enough points to start a cluster, using minPts.
    a. If not enough points, label it as noise.
5. Repeat steps 3 and 4 for all newly added points. Continue until no new points are added.
6. Return to step 2. If all points have been visited, end the process.

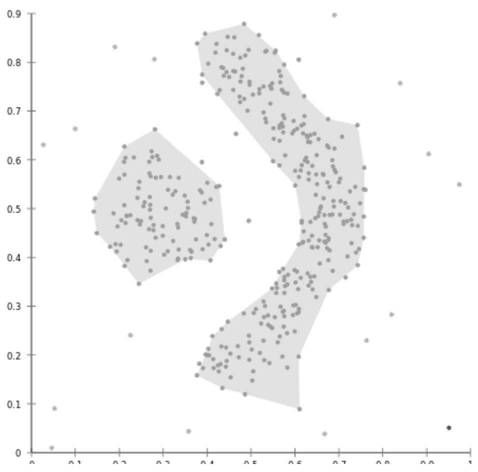


Figure 2: Example results of DBSCAN. Notice that a non-convex region can be formed.

Advantages

1. Does not require a specified number of clusters.
2. Can produce non-convex clusters, and clusters surrounded by other clusters.
3. Robust to noise and outliers.
4. Requires only two parameters: radius and minimum points per core.
5. Insensitive (mostly) to the order of the points.
6. Designed for databases with region queries.

Disadvantages

1. Partial deterministic problems – shared edges may jump between clusters.
2. The quality is dependent on the distance function of the feature vector.
3. Difficulties in high dimensional space, if based on Euclidean distance.
4. Cannot cluster well when clusters have large differences in density.
5. Choosing the radius can be difficult if the data is not well understood.

Applications and Usage Areas

1. Apache Commons Math
2. ELKI – Knowledge discovery application
3. R (programming language)
4. Scikit-Learn - python programming language library
5. SPMF – Java implementation

Expectation-Maximization

The expectation-maximization algorithm derives from statistics, is iterative, and attempts to find the maximum likelihood of the model (in the data). This is because no equations exist that can be solved directly. The data may be subject to unidentified parameters or even missing data points.

The clustering process involves a two-step process: the expectation step (E) and the maximization step (M). These two steps work in an iterative back-and-forth nature to provide refined information to the other, which eventually converges. (Figure 3)

Algorithm

1. Get input
   a. A set of data (X) is generated from a statistical model.
   b. Unobserved latent data - missing values (Z).
   c. Vector of unknown parameters (θ).
   d. Likelihood function (L).
2. Calculate the expectation step (E), the expected value of the log likelihood function with respect to Z and X using the current estimate of θ.

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \operatorname{E}_{\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)}}\left[\log L(\boldsymbol{\theta};\mathbf{X},\mathbf{Z})\right]$$

3. Calculate the maximization step (M), find the parameters that maximize the following equation.

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{\arg\max}\ Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$$

4. Repeat steps 2 and 3 until the parameter vector (θ) converges.

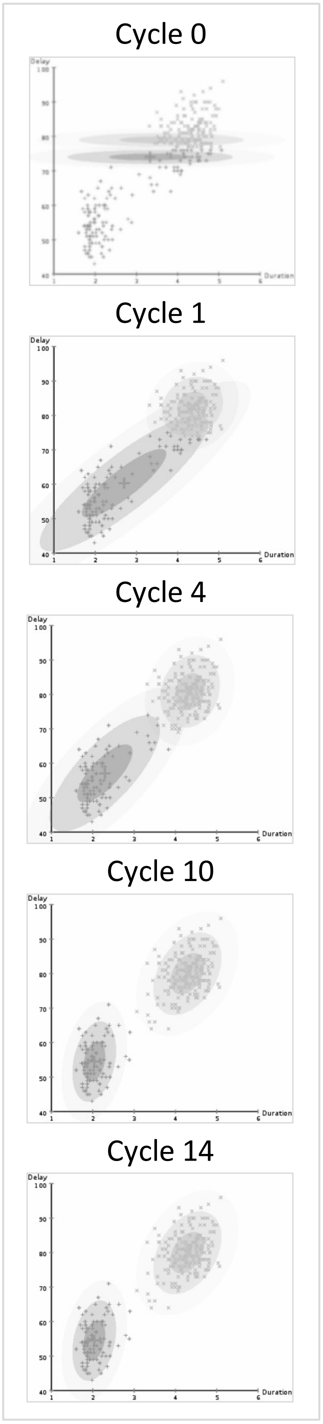


Figure 3: Convergence example of the Expectation-Maximization algorithm

Advantages

1. Handling of missing data.
2. Handling of unknown parameters.
3. Especially useful when the likelihood is of the exponential class.

Disadvantages

1. Complexity
2. Dependency on statistical models.
3. No guarantee of convergence.
4. Many variants for different applications.

Applications and Usage Areas

1. Natural language processing
2. Baum-Welch algorithm
3. Inside-outside algorithm
4. Psychometric, for estimating parameters of item response theory models.
5. Medical image reconstruction
6. Identification of natural vibration properties in structural systems (STRIDE algorithm).

Hierarchal

Hierarchal clustering or (HCA) attempts to merge or split data in a greedy manner. By this process, a hierarchy is produced at different levels, providing different levels of cluster membership.

The bottom-up approach, known as agglomeration, starts with each data point as its own cluster. The clusters are then continuously paired to form the next layer of the hierarchy.

The top-down approach, known as division, starts with all data points in one cluster. The cluster is continuously split, producing more next layers of the hierarchy.

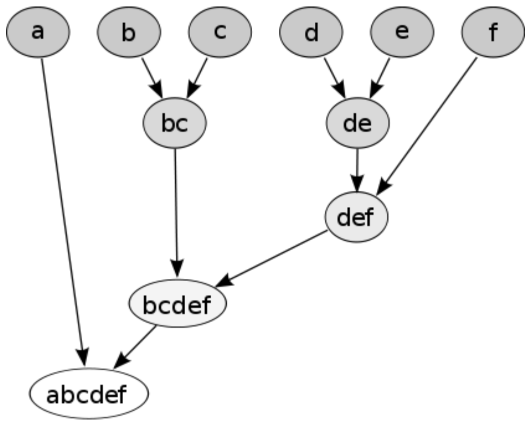


Figure 4: Hierarchy of clusters from ABCDEF data

To perform the clustering, two key components are required: the dissimilarity/distance metric and the linkage criteria.

The distance metric is used to decided where the cluster should be split, or which clusters should be joined. Usually it is of the form of Euclidean distance, but other forms such as squared Euclidean and Manhattan may also be used, as well as many others.

The linkage criteria is a pairwise function that determines the distance between two data points, according to the metric. Typical examples are complete-linkage, single-linkage, average-linkage, centroid-linkage, and minimum energy.

Algorithm (Agglomeration)

1. Get inputs
    a. Data Set
    b. Dissimilarity/Distance function
    c. Linkage Criteria
2. Compute matrix of distances, by comparing each element to each other element.
3. Identify pairs with minimum distances.
4. Increment the layer number and merge the identified pairs into clusters.
5. Update the matrix of distances
    a. Deleting rows/columns related to items of the new clusters.
    b. Add rows/columns for each new cluster.
6. Repeat steps 2 through 5 until all data are in a cluster.

Advantages

1. Any valid measure of distance can be used. Only a matrix of distances is necessary, potentially simplifying processing.
2. Simple to implement
3. Flexibility for adjustment and alteration

Disadvantages

1. Too slow for large datasets

Applications and Usage Areas

1. Open Source
    a. Social Network Visualizer
    b. Cluster 3.0 – GUI for different clustering routines
    c. ELKI – Knowledge discovery application
    d. Octave – Open source alternative to Matlab
    e. Orange – Free data mining suite
    f. R (programming language)
    g. SCaViS –Java computing environment
    h. Scikit-Learn - Python programming language library
    i. Weka – Java data mining suite
2. Commercial
    a. MATLAB – Multi-paradigm programming environment
    b. SAS – Statistical analysis suite
    c. Mathematica – Mathematics focused programming environment
    d. NCSS – Statistical analysis suite
    e. SPSS – Statistical analysis suite
    f. Qlucore – bioinformatics gene explorer
    g. Stata – Statistical analysis suite

K-Means

K-Means clustering is based on vector quantization, where data is partitioned into k number of clusters. Each cluster is represented by a prototype feature vector produced from averaging the cluster members. The data is organized into observations (feature vectors), and these observations go through a two-step process of assignment and update. This process repeats iteratively until no further assignments occur.

The assignment step assigns each observation to a cluster based on its distance from the possible cluster prototypes. The update step calculates a new mean feature vector for all members of the cluster.

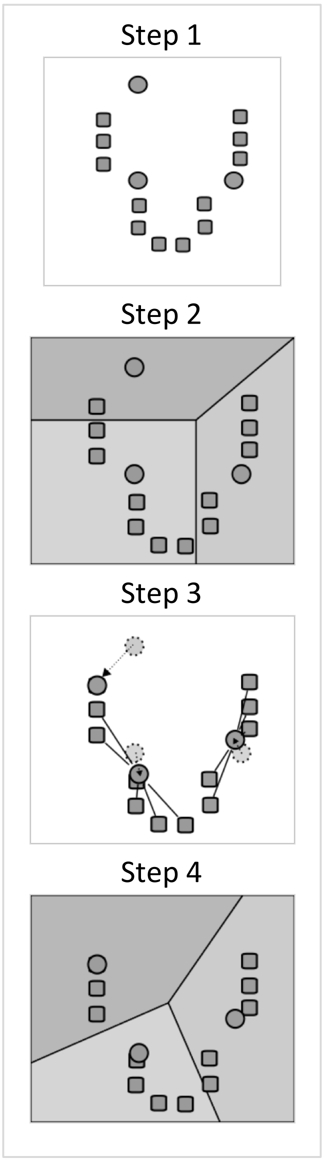


Figure 5: k-means clustering example. Round dots are centroids and square dots are data.

Algorithm (standard)

1. Get input
    a. Set of Data (x)
    b. Number of clusters (k)
2. Generate initial cluster prototypes (m), randomly from data points, or initiate as predefined inputs.
3. Assign each observation to a cluster (S) according to sum of squares (Euclidean distance).
    $$S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\|^2 \le \|x_p - m_j^{(t)}\|^2 \ \forall j, 1 \le j \le k\}$$
4. Update the cluster prototype (m) by averaging members.
    $$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$
5. Repeat steps 2 and 3 until no further assignments occur.

Advantages

1. Very fast
2. Customizable

Disadvantages

1. Must specify the number of clusters in advance.
2. Usage of other distance equations may prevent converging.
3. Many variations and custom alterations

Applications and Usage Areas

1. Areas
    a. Market Segmentation
    b. Computer vision
    c. Geo-statistics
    d. Astronomy
    e. Agriculture
2. Open Source Software
    a. Accord.NET – Scientific computing platform
    b. CrimeStat – mapping of criminal activity
    c. ELKI – Knowledge discovery application
    d. Julia – High-level dynamic programming language
    e. Mahout – Scalable/distributed machine learning platform.

f. MLPACK – C++ library for machine learning
g. Octave – Alternative to Matlab
h. OpenCV – Library for computer vision
i. PSPP – free alternative to SPSS.
j. R (programming language)
k. SciPy – Library for scientific computing in python
l. Spark – Cluster-computing framework.
m. Torch – Scientific computing platform built on the C programming language.
h. Weka – java data mining suite

3. Proprietary Software
    a. Analytic Solver – Engine of Microsoft Excel
    b. Ayasdi – Machine intelligence platform for data mining
    c. MATLAB – Multi-paradigm programming environment
    d. Mathematica – Mathematics focused programming environment
    e. RapidMiner – Environment for data mining and predictive analytics
    f. SAP HANA – Relational database with built in analytics
    g. SPSS – Statistical analysis suite
    i. Stata – Statistical analysis suite

Self-Organizing Map

A self-organizing-map (SOM) is a specific type of artificial neural network (ANN), that is trained via unsupervised learning. It produces low-dimensional representations of a given input space, which represents a training data set. The largest difference from traditional ANNs, is the neuron weights are not updated via error correction, but rather by competitive learning.

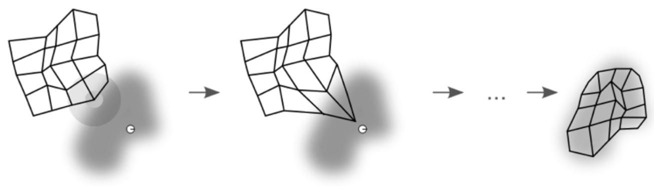


Figure 6: Training process of a SOM. Black grid represents current SOM nodes. White dot (in blob) represents current data point under comparison. Grey dot on grid represents primary node to be adjusted.

A SOM is created in two stages, training and mapping. Training builds the map using provided input examples (like all ANNs). Mapping uses this trained network to automatically classify a new input. As per typical ANNs, the network structure is usually rectangular or hexagonal and each node has an associated weight.

Algorithm

1. Get input parameters
    a. D – target data set
    b. Lambda – iteration limit
    c. Alpha – learning restraint
2. Randomize the map node weights.
3. Select an input vector from D.
4. Cycle through each node in the map
    a. Compare the distance of the map node to the current input vector.
    b. Keep the node that is the most similar (BMU)
5. Update the BMU node and its neighbors by making them more like the current input vector.

6. Repeat steps 3 through 5 until an end condition
    a. The max number of cycles is reached
    b. The map is no longer updating (within a chosen accuracy).

Advantages

1. Easy to understand
2. Work very well
3. Easy to evaluate how good the map is, providing how strong similarities are.

Disadvantages

1. Getting good training data.
2. Similar groups may form separately, rather than in a single group.
3. Selection of good initial approximation.
4. Computationally expensive.

Applications and Usage Areas

1. Meteorology
2. Oceanography
3. Project prioritization & selection
4. Seismic facies analysis (oil and gas exploration)

Algorithm Selection

Task Relevance

Each of the previously described methods is compared for its ability to meet the discussed task, which is to identify features, objects, and patterns within a set of data. A list of comments is shown for each method.

DBSCAN

- Since it is density based, it will likely be good at locating how often similar objects occur once they are identified.
- It will not be capable of identifying objects without a definition of the feature vector. This is not useful for the first layer, where nothing is known about the data.
- No information is provided about membership degree.

Expectation-Maximization

- It is meant for identifying parameters that describe a particular cluster.
- The degree of membership in a cluster is high resolution.
  (deviation from the centroid)

Hierarchal

- The associative nature will be good for translating observations back into their lower structures or simpler components.
- The hierarchal nature supports the idea of both simple objects and complex objects appearing in the data.

K-Means

- Since a specified number of clusters must be given, this is not an option. The data will likely have many very different objects, and new objects may appear at any time.

Self-Organizing Map

- A SOM required initial input vectors, so it is not capable of fully blind discovery.
- This is likely too computationally heavy.
- It may be valuable after the first few layers of identification have taken place.
- Is intended for complex data types. It is too complex for items that only have a few features.

Selection Choice

Given that the source data will be a single stream of unlabeled data, and no information will be provided about the data set, both K-Means and Self Organizing Maps methods are unworthy. K-Means must know the number of clusters beforehand and SOMs intended objects with consistent complexity.

Next, the requirement of object relationship is considered. As data will be compared across future data streams, a clear membership degree needs to be present. Although DBSCAN, may be the best choice for creating the clusters, it provides no information about the membership degree in the cluster. Hence it is ruled out.

Finally, the choice is between Expectation-Maximization and Hierarchical methods. Both provide clear information about degrees of membership. In fact, Expectation-Maximization provides an unnecessarily high resolution level of the membership degree. It requires significantly more calculation effort. Hence, Hierarchical is preferred over Expectation-Maximization because it is simpler and still provides the required results.

The final option is Hierarchal, which appears to be the best fit. The nature of the system is to locate objects that relate to one another and build layered knowledge. This fits with the task of finding layers of objects and patterns. Additionally, the connection of higher-level objects to their lower level components is retained.

Therefore, it is recommended to use the **Hierarchal method** for discovery of previously unidentified features, objects, and patterns.

Sample Program

A sample program using hierarchical clustering has been developed. A user draws symbols repeatedly on a canvas and the symbols are compared to each other, creating clusters. The results of the clustering are shown both graphically and textually, displaying the membership degree and distance. This program is split into two parts, the interface and the hierarchical analysis engine. This is to support future code reuse.

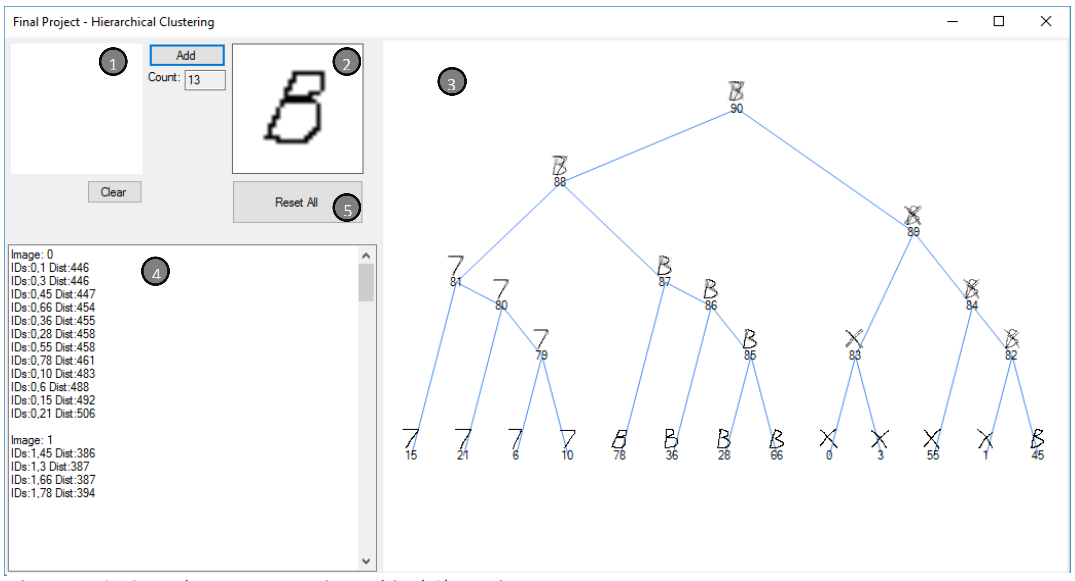
The program engine, “SymbolHierarchalClusterer”, is described in five parts:

1. Fields
2. Classes
3. Algorithm
4. Calculation of Symbolic Image Distance

Interface

The program interface is relatively simple. As such numbered nodes are used to identify each part of the interface and a description is provided.

1. **Drawing Canvas** – The user draws any desired symbol on the white square.
    a. Press the “Add” button to add the drawing to the collection of samples.
    b. Press the “Clear” button to reset the canvas back to white.
2. **Result Canvas** - The user’s drawing is centered in the white square and converted to an image bitmap. This is the result.
3. **Hierarchy Tree –** The entered samples are compared using hierarchical analysis. This is the tree of results, clustering the samples together.
    a. An image of the sample is drawn and its unique ID is shown below.
    b. Cluster images show the conglomerated version of each cluster member.
4. **Detailed Results –** A pairwise comparison of all elements to other elements.
5. **Reset All –** Clears the hierarchy interface and removes all data, allowing the user to start entering new data samples.



```
Image: 0
IDs:0,1 Dist:446
IDs:0,3 Dist:446
IDs:0,45 Dist:447
IDs:0,66 Dist:454
IDs:0,36 Dist:455
IDs:0,28 Dist:458
IDs:0,55 Dist:458
IDs:0,78 Dist:461
IDs:0,10 Dist:483
IDs:0,6 Dist:488
IDs:0,15 Dist:492
IDs:0,21 Dist:506

Image: 1
IDs:1,45 Dist:386
IDs:1,3 Dist:387
IDs:1,66 Dist:387
IDs:1,78 Dist:394
```

Figure 7: C# Sample Program - Hierarchical Clustering

Fields

The engine's fields are used for tracking the user's drawings, storing comparison results, and storing final clustering results. This complete system is stored via three lists.

1. **Symbolic Images List** – A list of sample images created by the user.
2. **Distances List** – A list of distances between all symbolic images.
3. **Cluster Hierarchy List** – The generated connections between symbolic images.

Classes

The lists described above are created from two classes. These classes are used for automating analysis of images, as the user submits them.

1. **Symbolic Image** – A node class for storing an image.
    a. Provides functionality to connect to other nodes through an internal list of children "members".
    b. Fields include
        i. Members – The children nodes that this symbolic image is generated from, if it was generated from other nodes.
        ii. Image Matrix – The 2D array of pixels representing the image.
        iii. ID – a unique identifier for tracking the image.
    c. Construction has two possibilities.
        i. User submitted array of pixels.
        ii. Conglomeration from children nodes. The children's arrays are combined and a grayscale representative image is produced.
    d. Additional methods include
        i. Count Black Pixels – The number of pixels used to generate the image.
        ii. Center Image – Repositions the drawing into the center of the array.
        iii. Calculate Centroid – Calculates the weighted center of the image.
        iv. Convert to Bitmap – Converts from a binary array to a bitmap image.
        v. Convert to Bitmap (Grayscale) – Converts from an integer array to a bitmap image.
        vi. Calculate distance – Calculates the distance comparison of one symbolic image to another. This function is the primary element used for clustering.
2. **Symbolic Image Distance** – An analysis storage class. It contains only three elements.
    a. Symbolic Image A – the first image of a comparison.
    b. Symbolic Image B – the second image of a comparison.
    c. Distance – the comparison results.

Algorithm

The algorithm is only described from entry of an item to the engine, as interactions on the interface are irrelevant. The interface also uses the cluster results to generate a visual hierarchy tree, which is again irrelevant for the analysis engine.

Below is a stepwise explanation of the hierarchical algorithm's process and conversion of a collection of integer arrays into symbolic images and finally into clusters.

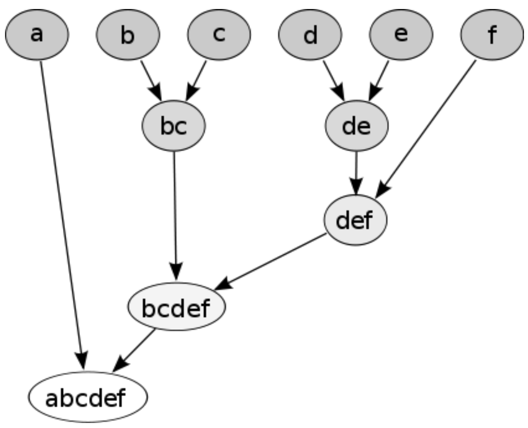
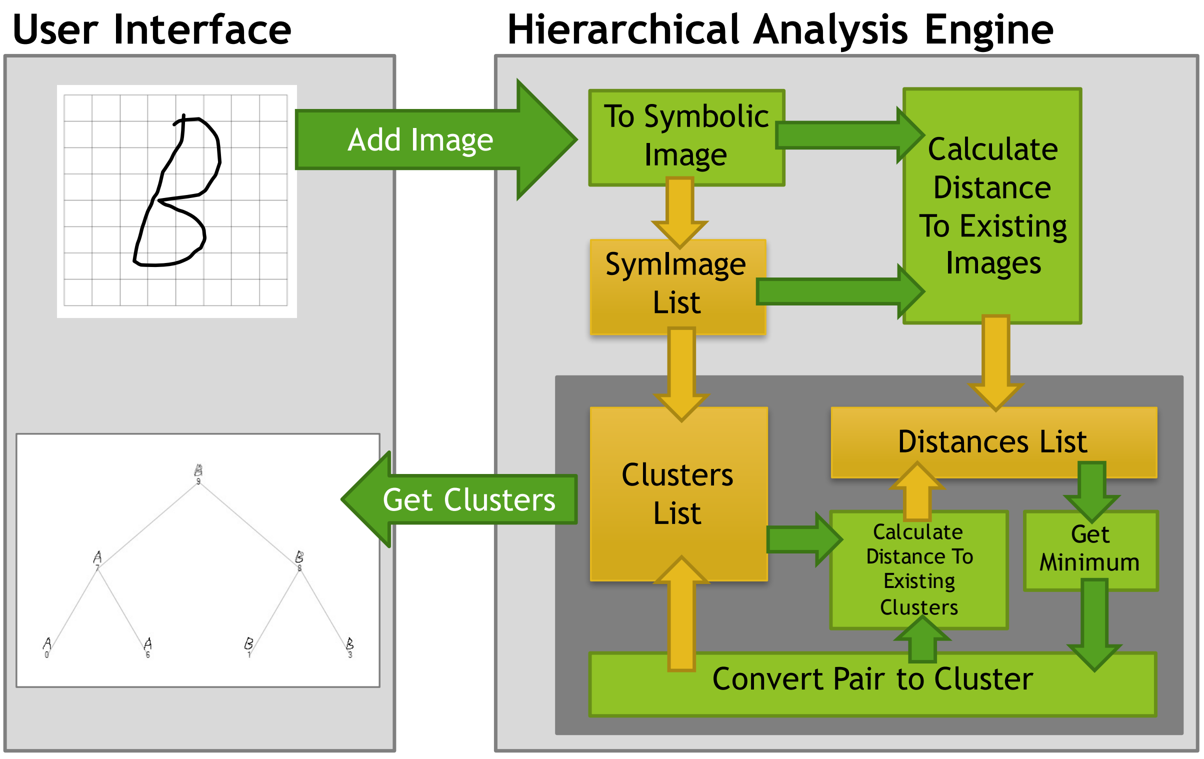


Figure 8: Hierarchy of clusters from ABCDEF data



1. Initial construction of empty tracking and results lists.
    a. Symbolic Images List
    b. Symbolic Image Distances List
    c. Cluster Hierarchy List
2. A new image matrix is submitted to the engine, and it is converted to a symbolic image.
    a. The image is assigned a unique identifier (ID).
    b. The number of black pixels are recorded.
    c. The drawing is centered within the image.
    d. The image centroid is calculated.
    e. A bitmap version of the image matrix is created.
3. The new symbolic image is compared to all existing symbolic images and results are stored in the list of distances.
4. The new symbolic image is stored in the list of symbolic images.
5. Cluster analysis is performed and results are stored in the cluster hierarchy list.

   a. All symbolic images are copied to the clusters list.
   b. A copy of the distances list is created.
   c. The distances list is sorted, smallest to largest.
   d. Beginning of loop: the distance list is counted.
      i. If count is greater than zero, continue to step “e”.
      ii. If count is zero, skip to step “L”.
   e. The first distance entry is retrieved.
      i. Symbolic image A is retrieved.
      ii. Symbolic image B is retrieved.
   f. All distance entries referencing symbolic images A and B are removed.
   g. Symbolic images A and B are removed from the clusters list.
   h. A new symbolic image is created from combining symbolic images A and B.
   i. The new symbolic image is compared to all existing symbolic images in the clusters list.
   j. The new symbolic image is added to the clusters list.
   k. Return to beginning of loop at step “d”.
   l. Save the clusters list as the results and end.

Calculation of Symbolic Image Distance

A pair of symbolic images is compared to generate a distance value. An example of this analysis is provided below for symbolic images A and B. The distance is then normalized against the average number of black pixels between the symbolic images.

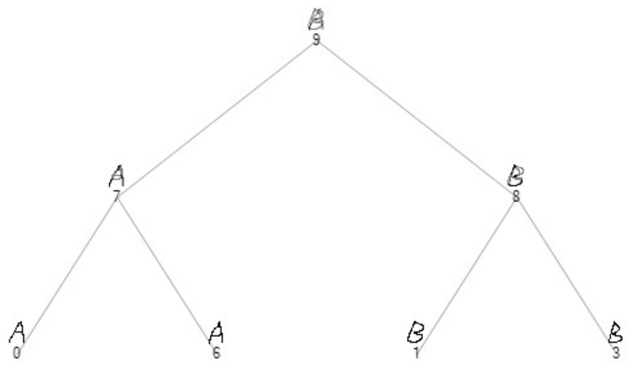
1. Cycle through every black pixel of symbolic image A.
   a. Cycle through every black pixel of symbolic image B.
      i. Calculate the distance between each pair of black pixels of symbolic images A and B, using the below equation.
      $$Distance_i = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2}$$
      ii. Add the distance to a total distance value.
      $$DistanceTotal = \sum Distance_i$$
2. Divide the total distance by the average number of black pixels.
$$DistanceNormalized = \frac{DistanceTotal}{(NumBlackPixels_A + NumBlackPixels_B)/2}$$

Results

Below are a series of tests to compare some logical variations in drawing of symbols. Most situations work, however there are clearly some situations where the clustering is confused.

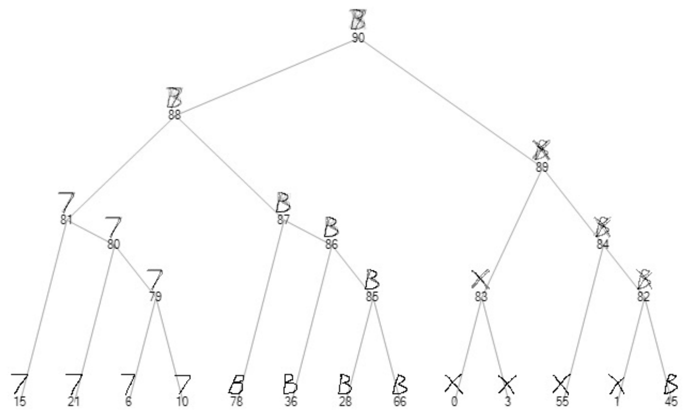
Test 1 – Two Symbols

Simple example of two symbols: A & B. Both are correctly grouped.



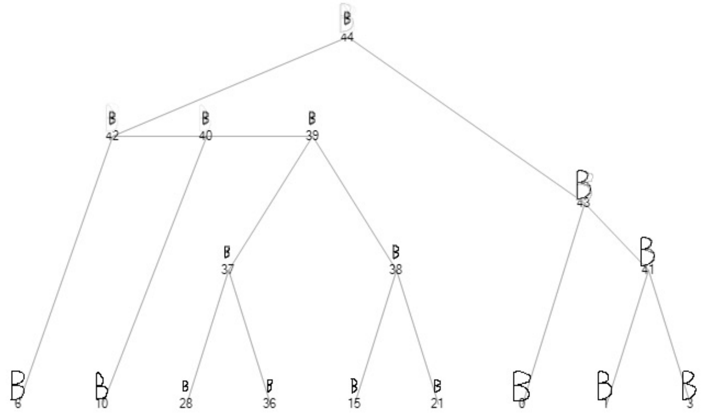
Test 2 – Three Symbols

Three symbols and more sample images. It can be seen that one “B” is confused as an “X”.



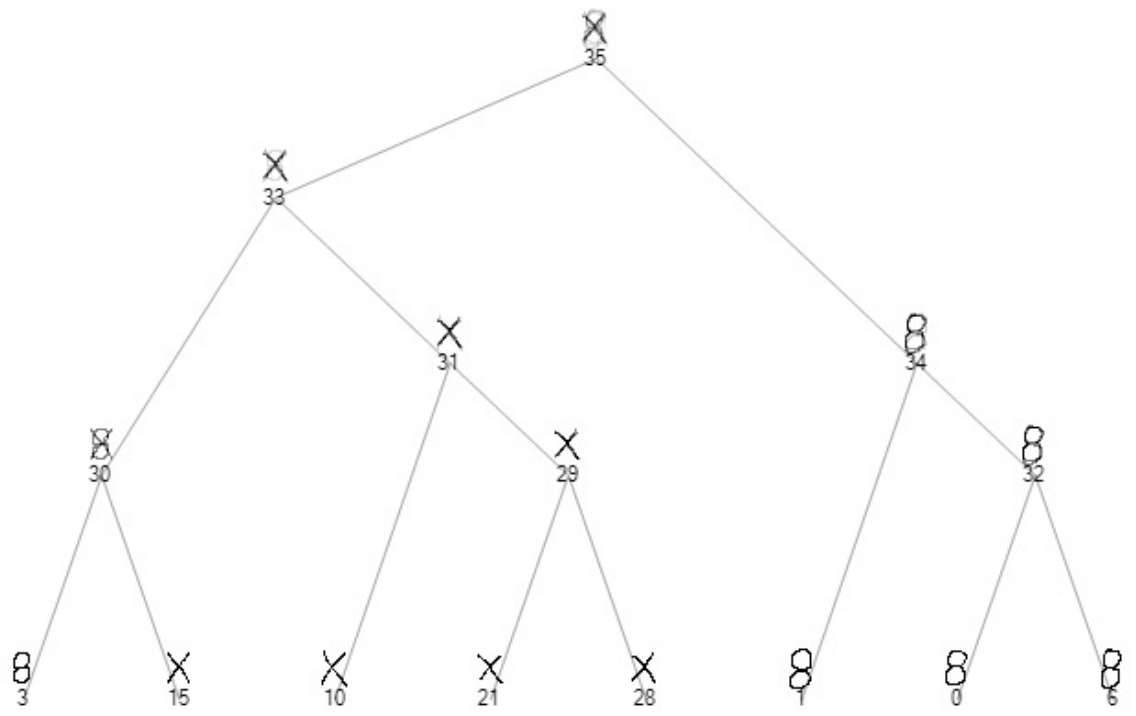
Test 3 – Drawing Size

A “B” was drawn small and large. It can be seen that it mostly sees the difference, but it is still sometimes ambiguous. Possibly, with more samples, it would again group them properly.

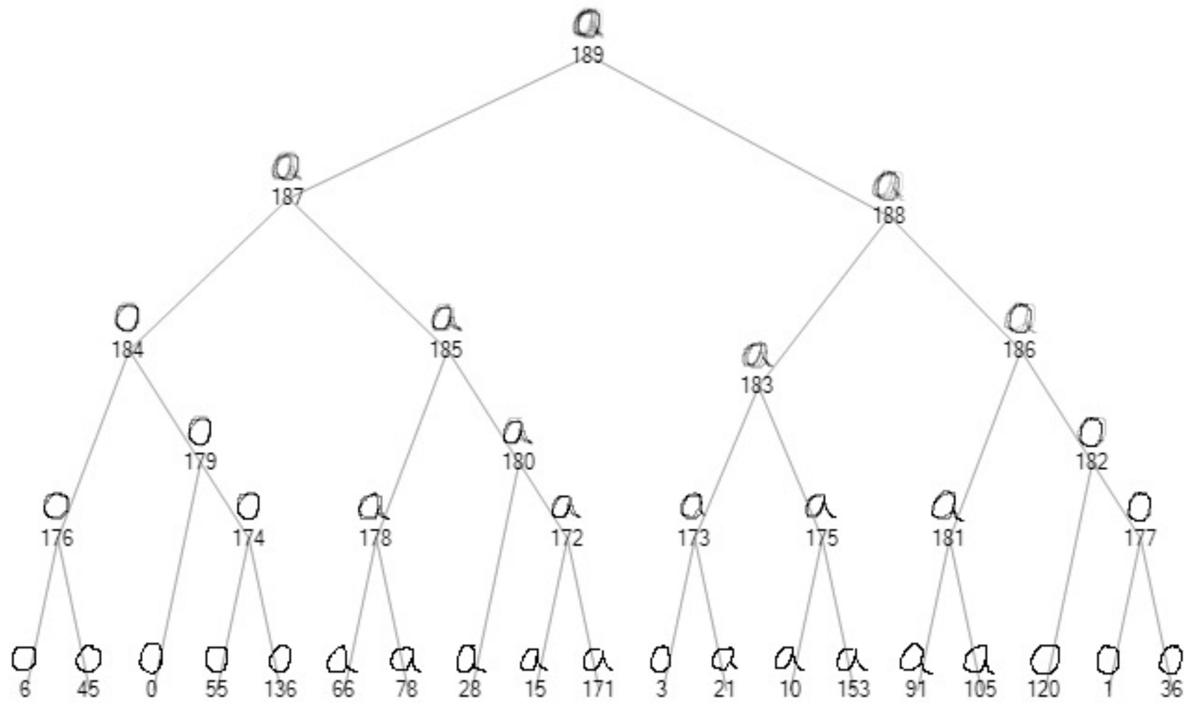


Test 4 - Similar features

An “8” is similar to an “X” in that it is connected at the top and bottom. Interestingly, It was still able to distinguish this difference, with only one error.



The same test was performed between “a” and “o”. It is still mostly correct but there is still significant ambiguity.



Discussion

The hierarchical clustering process should probably only be used if an accurate distance function can be created to represent the data. With simple numeric values in a data stream, this is straight-forward and simple. However, with images for example, a single data object is already quite complex. Hence, defining such a distance function is not a trivial task. It is recommended to use this process for simpler, easily definable data types.

The hierarchical clustering mechanism seems to work as expected. However, the definition of the distance function clearly needs improvement. It needs to account for additional image characteristics.

Conclusion

Five methods of clustering data have been reviewed and compared. Additionally, their relevance to the task of identifying features, objects, and patterns in the data was considered. From these, a single method, Hierarchical Clustering, was identified as the best solution.

A sample program, in C#, has been created to test the process of hierarchical clustering. An interface has been created allowing a user to draw symbols on a canvas. These symbols are then added to a list and compared in a pairwise fashion, generating a hierarchy of clusters and the relative degrees of membership.

The results show that this process seems difficult for more complex datatypes such as images, but it is indeed semi-successful. Four tests were performed using different numbers of shapes, different sizes of shapes, and similar shapes. In most situations, it could correctly group drawn symbols, with only a few errors. It seems that the complexity of the distance equation must be similar to the complexity of the data object.

Because the task is to identify features, objects, and patterns within a simple numeric data stream, it is therefore recommended to utilize hierarchical clustering.

References

1. Bueno, Leandro. (2015). **A self-organizing maps classifier structure for brain computer interfaces**. Biomedical Engineering. Vol 31, Number 3. P 232-240.
2. Campello, R. J. G. B. (2012, October 3). **A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies.** Springer.
3. **Category:Data clustering algorithms**. (n.d.) In Wikipedia. Retrieved March 4, 2017, from https://en.wikipedia.org/wiki/Category:Data_clustering_algorithms
4. **Cluster Analysis**. (n.d.) In Wikipedia. Retrieved March 3, 2017, from https://en.wikipedia.org/wiki/Cluster_analysis
5. **DBSCAN**. (n.d.) In Wikipedia. Retrieved March 5, 2017, from https://en.wikipedia.org/wiki/DBSCAN
6. **Expectation-maximization algorithm**. (n.d.) In Wikipedia. Retrieved March 5, 2017, from https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm
7. **Feature Selection**. (n.d.) In Wikipedia. Retrieved March 3, 2017, from https://en.wikipedia.org/wiki/Feature_selection
8. Germano, Tom (1999, March 23). **Self Organizing Maps**. Retrieved from http://davis.wpi.edu/~matt/courses/soms/
9. **Hierarchal clustering**. (n.d.) In Wikipedia. Retrieved March 5, 2017, from https://en.wikipedia.org/wiki/Hierarchical_clustering
10. Kanungo, Tapas. (2016). **The Analysis of Simple k-Means Clustering Algorithm**.
11. **k-means clustering**. (n.d.) In Wikipedia. Retrieved March 5, 2017, from https://en.wikipedia.org/wiki/K-means_clustering
12. Liang, Dawen. (2015, February 25). **Technical Details about the Expectation Maximization (EM) Algorithm.** Columbia University.
13. **Likelihood function**. (n.d.) In Wikipedia. Retrieved March 5, 2017, from https://en.wikipedia.org/wiki/Likelihood_function
14. Moulavi, Davoud. (2014). **Density-Based Clustering Validation**. Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA, 2014
15. **Pattern Recognition**. (n.d.) In Wikipedia. Retrieved March 3, 2017, from https://en.wikipedia.org/wiki/Pattern_recognition
16. **Pattern Recognition (psychology)**. (n.d.) In Wikipedia. Retrieved March 3, 2017, from https://en.wikipedia.org/wiki/Pattern_recognition_(psychology)
17. **Self-organizing map**. (n.d.) In Wikipedia. Retrieved March 5, 2017, from https://en.wikipedia.org/wiki/Self-organizing_map
18. **Single-linkage clustering recognition**. (n.d.) In Wikipedia. Retrieved March 5, 2017, from https://en.wikipedia.org/wiki/Single-linkage_clustering
19. Tanaseichuk, Olga. (2015, September 24). **An Efficient Hierarchical Clustering Algorithm for Large Datasets**. Austin Publishing Group.