Clustering and Topological Data Analysis: Comparison and Application

Kara Combs
Applied Research Solutions
Kcombs@appliedres.com

Trevor J. Bihl
Air Force Research Laboratory
Trevor.Bihl.2@us.af.mil

Abstract

Clustering is common technique used to demonstrate relationships between data and information. Of recent interest is topological data analysis (TDA), which can represent and cluster data through persistent homology. The TDA algorithms used include the Topological Mode Analysis Tool (ToMATo) algorithm, Garin and Tauzin's TDA Pipeline, and the Mapper algorithm. First, TDA is compared to ten other clustering algorithms on artificial 2D data where it ranked third overall. TDA had the second-highest performance in terms of average accuracy (97.9%); however, its computation-time performance ranked in the middle of the algorithms. TDA ranked fourth on the qualitative "visual trustworthiness" metric. On real-world data, TDA showed promising classification results (accuracy between 80-95%). Overall, this paper shows TDA is a competitive algorithm performance-wise, though computationally expensive. When TDA is used for visualization, the Mapper algorithm allows for unique alternative views especially effective for visualizing highly dimensional data.

1. Introduction

Essential to turning data into useful information is the ability to cluster, classify, and/or predict (Rokach & Maimon, 2005). There are many types of clustering/classification algorithms, but one algorithm that has risen in popularity recently is topological data analysis (TDA). Unique to TDA is the consideration of the "shape" of the data typically through persistent homology; this allows for the input data to be visualized differently and more effectively as the data dimensionality increases (Chazal & Michel, 2021). However, when TDA is employed, it is often without a comparison to a baseline clustering method (Singh, Mémoli, & Carlsson, 2007) (Garin & Tauzin, 2019). The motivation to explore TDA is predominantly due to the lack of comparative research on clustering and the need to handle highly dimensional "big data." Of interest herein is how TDA compares to other clustering algorithms and the advantages/disadvantages of its application. Typical comparison metrics includes quantitative measures (e.g., accuracy); however, previous literature suggests the promise of heuristic evaluation for ease of interpreting visual diagrams (see (Combs, et al., 2020)).

This paper focuses on three research questions to contribute to the literature on TDA and its usage in clustering:
RQ1: How can clustering algorithms be compared?
RQ2: How does TDA compare to traditional methods?
RQ3: How can TDA results be interpreted and best utilized for clustering?
To answer these questions, this paper explores three dimensions of utility for clustering: accuracy of predicted clusters to known clusters, computation time, and a qualitative comparison based on the newly proposed and defined "visual trustworthiness." After discussing several popular clustering methods and background TDA, TDA's performance is compared to 10 other clustering algorithms in terms of accuracy, completion time, and visual trustworthiness on six artificial data sets. Next, TDA is used for classification of the real-world Palmer Penguins data set using the Topological Mode Analysis Tool (ToMATo) algorithm and the MNIST digits data set through Garin and Tauzin's TDA pipeline. The Palmer Penguins and MNIST digits data set are visualized and interpreted through Mapper, as implemented in the KeplerMapper Python library. Finally, this paper ends with conclusions and future work directions.

2. Background

The tasks of clustering and classification have a long history between their techniques, methods, and metrics; clustering is popular because it can be conducted in supervised or unsupervised approaches (Rokach & Maimon, 2005). Some well-known cluster methods are discussed in terms of how they work as well as their advantages and disadvantages. The topological data analysis (TDA) algorithm will then be compared to these well-known methods.

URI: https://hdl.handle.net/10125/102731
978-0-9981331-6-4

2.1. Common Clustering Algorithms

The clustering methods discussed in this section include k-means, affinity propagation, mean shift, spectral clustering, ward hierarchical, agglomerative clustering, density-based spatial cluster of applications with noise (DBSCAN), ordering points to identify cluster structure (OPTICS), balanced iterative reducing and clustering using hierarchies (BIRCH), and Gaussian mixture. These methods were based on those implemented by sci-kit learn (see (Pedregosa, et al., 2011)).

2.1.1. Mini-batch K-means. Arguably the most famous and common clustering method is k-means, which was first described in (Lloyd, 1957). The algorithm works by minimizing the distance between each point and its cluster across *k* total clusters (where *k* is determined by the user) until a stopping point is reached (specific value, number of iterations, etc.) (Ghosh & Liu, 2009). There are many variations to the k-means algorithm; however, mini-batch (see (Sculley, 2010)) is one of the more popular versions due to its ability to decrease computation time without sacrificing the quality of requests compared to the original algorithm (Scikit-learn, 2022).

2.1.2. Affinity Propagation. A more recent algorithm developed in (Frey & Dueck, 2007) is affinity propagation, which uses the most representative points within the data set, dubbed "exemplars," based on sending messages between points. Unlike k-means, the number of clusters is determined by the algorithm rather than the user (Scikit-learn, 2022). Affinity propagation's only two inputs are *preference* (which directly affects the number of exemplars) and *damping factor* (which affects the message rate); one of its major drawbacks is the computation time required.

2.1.3. Mean Shift. Initially developed in (Fukunaga & Hostetler, 1975), mean shift is commonly used for computer vision problems. The number of clusters is identified by the algorithm but influenced by the *bandwidth* parameter, which can be directly set by the user or estimated by a separate function; similar to affinity propagation, the mean shift algorithm will converge when it has reached a minimum or the change in centroid placement is insignificant (Scikit-learn, 2022).

2.1.4. Spectral Clustering. Instead of compactness of data, spectral clustering performs best on connected data (Singh A. , 2010). Spectral clustering utilizes matrices' eigenvectors to cluster the points using another clustering method such as k-means or k-nearest neighbor (KNN) (Von Luxburg, 2007). The number of clusters is pre-determined by the user and spectral clustering is best applied to data with few clusters (Scikit-learn, 2022).

2.1.5. Hierarchical Clustering. There are several types of hierarchical/agglomerative clustering, which all attempt to split and merge data into a tree-like structure (Nielsen, 2016). Important parameters include the district metric used and the linkage function (Nielsen, 2016). Single linkage "minimizes the distances between the closest observations of pairs of clusters." Maximum/complete linkage "minimizes the maximum distance between observations of pairs of clusters." Ward linkage (see (Ward Jr., 1963)) minimizes the sum of squared differences within the clusters (Scikit-learn, 2022). Average linkage is often used alternatively to Ward when non-Euclidean district metrics are used.

2.1.6. Density-based Spatial Cluster of Applications with Noise (DBSCAN). First published in 1996, DBSCAN uniquely considers density as its name suggests (Ester et al., 1996). DBSCAN requires two parameters: a minimum number of points per "core sample" (dubbed "minPts") and epsilon (abbreviated to "eps" or ε), the distance the minimum points must lie to be a part of the "core sample." A "core sample" is a point/sample that is in a highly-dense region that is to become a cluster (Scikit-learn, 2022).

2.1.7. Ordering Points to Identify Cluster Structure (OPTICS). Sharing some of its creators with DBSCAN, OPTICS is another density-based algorithm (Ankerst et al., 1999). OPTICS has the same inputs as DBSCAN, but allows ε to be a range of values (Rhys, 2020). The smallest value of ε is called the core distance (ε'), where the minPts is within the radius of the selected "core sample." The "reachability distance" is considered the distance between a core sample and another core sample within ε-distance. This allows for the creation of a "reachability plot" that plots values of ε on the y-axis and the points on the x-axis, which segments' steepness(es) determines whether it is clustered as a core point or noise (Scikit-learn, 2022).

2.1.8. Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH). BIRCH is not distance-based but uses tree structures to cluster data (Zhang et al., 1996). Specifically, a Clustering Feature Tree (CFT) is created, (optionally) condensed into a smaller CFT, and then, used for global clustering. The data is divided into Clustering Feature nodes, which consist of Clustering Feature subclusters (Scikit-learn, 2022). Scikit-learn's implementation has two parameters: branching factor ("limits the number of subclusters within a node") and threshold ("limits the distance between the entering sample and the existing subclusters").

2.1.9. Gaussian Mixture. The Gaussian mixture algorithm is probabilistic, which allows uncertainty to be associated with each data point (Kan, 2017). Dempster et al. (1977)'s expectation-maximization algorithm is used to adjust various unknown parameters associated with a "mixture" of normal distributions the input data follows. (Scikit-learn, 2022). During clustering, the covariance can be spherical, diagonal, tied, or full in scikit-learn's implementation.

2.2. Topological Data Analysis (TDA)

Topological data analysis (TDA) is a relatively new clustering approach with most of its history beginning in the early 2000s (Chazal & Michel, 2021). At the heart of TDA is identifying structure often through persistent homology, which is a particularly useful way to visualize multi-dimensional data (Wasserman, 2018). There are several iterations of algorithms that utilize TDA, which are primarily based around three aspects algebraic topology: simplicial complexes, persistence barcodes/diagrams, and/or Betti numbers/curves.

Before the application of persistent homology, data is converted into simplicial complexes commonly visualized through a Czech or (Vietoris-)Rips diagram (Ghrist, 2008). Figure 1 shows a point cloud of 6 points on the left (orange oval-like figures) and a multi-colored Rips diagram on the right below each sub-figure's label, (a)-(d). Looking at the point clouds, each individual point (which are classified as a 0-simplex) within has a diameter, symbolized ε, shown in the orange area around it, which increases with each Figure 1(a)-1(d). In the case two data point's radius touch or overlap, they are "adjoined" and considered to be a 1-simplex. This relationship is shown in the multi-colored figures where a line connects two points (best shown in on the left of Figure 1(b)). When three points radius' touch it becomes a 2-simplex (yellow triangle), four connected points become a 3-simplex (green quadrilateral), five

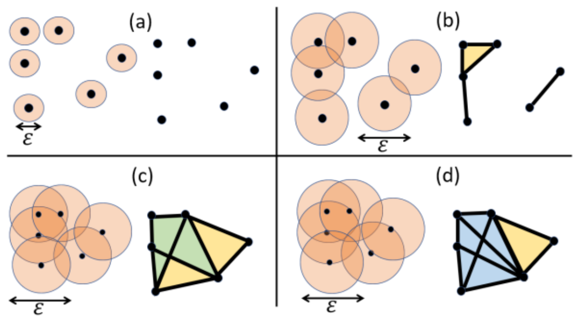


Figure 1. Rips diagram creation

connected points become a 4-simplex (blue irregular pentagon), and so on. As the radius increases the shape will become more interconnected and produce more simplexes of higher degrees until all points are connected.

After the simplicial diagram (the Rips diagram(s) shown in Figure 1) has been used to re-visualize the data, of next consideration is the point cloud's persistence which is described through its "homology dimensions," often symbolized as H_k (where k is replaced by the numerical dimension) (Munch, 2017). A persistence barcode/diagram has the ability to describe point cloud "features" and the aforementioned dimensions are not equivalent to a Rips diagram's simplexes. In persistence homology, "features" refer individual points, connected parts, or loops found within a given simplicial complex. The 0th dimension represents the number of "connected parts" which are individual points or set of points "connected" to one another in the given simplicial complex. (Munch, 2017). The 1st dimensions represent loops (also sometimes called holes) A persistence barcode/diagram is able to visualize when these features are initially formed (called their "birth") and the loops cease to exist because they are "filled in" (called their "death") (Amezquita, Quigley, Ophelders, Munch, & Chitwood, 2020).

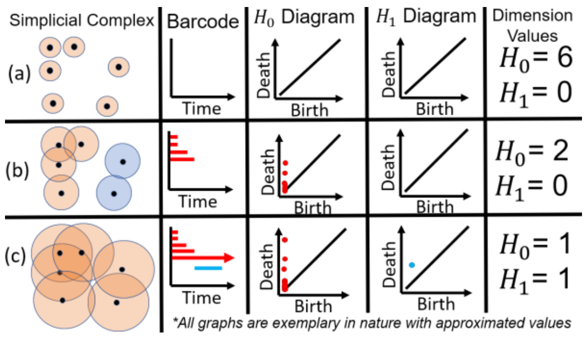


Figure 2. Persistence Barcode and Diagram

Presented with three chronological-consecutive point clouds, their corresponding persistence barcodes, H_0 persistence diagram, H_1 persistence diagram, and dimensions values are presented in Figure 2. Assuming Figure 2(a) is the starting figure, H_0= 6, for each of its 6 points; however, this is usually not reflected in the persistence barcode or diagram. Moving to Figure 2(b), the simplicial complex is color coded to represent its two connected parts (hence, H_0 = 2). These connected parts as well as any previously created connected parts are shown in the persistence barcode and H_0 diagram. The H_1 diagram is still blank because no loops are currently present. Finally, in Figure 2(c), we have a loop (thus H_1 = 1), which is reflected in the blue elements of the persistence barcode and diagrams. A point cloud's H_0 dimension will always be at least 1 because there must be at least 1 "connected part" even if said connected part also forms a loop as in Figure 2(c). A more in-depth example of a simplicial complex with higher dimensions is shown in (Ghrist, 2008).

Though persistence barcodes and diagrams provide another way to summarize information in a point cloud, they still produce significant amount of information. Many TDA algorithms, and other mathematical methods, further utilize the Betti number. A Betti number for a given surface (or shape) is the maximum number of cuts that can be made to the surface without it splitting into two separate pieces (Gardner, 1984). A Betti number is symbolized, β_k, where *k* represents a homology dimension (Munch, 2017). The homology dimension values described earlier are equivalent to Betti numbers as β_0 represents the number of connected parts, β_1 represents the number of loops, and β_2 represents the number of voids/holes. Higher dimensions of Betti numbers exist, but also become increasingly difficult to describe. The Betti number(s) are shown for a point, circle, sphere, torus, and Klein bottle in Figure 3, to better understand what is meant by each homology dimension.

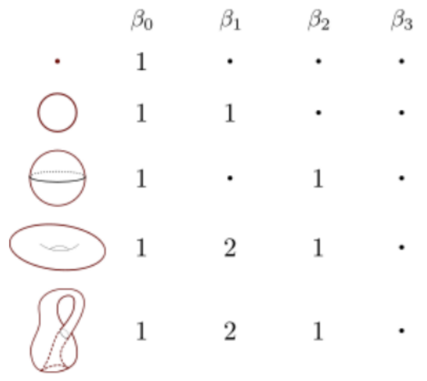


Figure 3. Common Shapes and Their Betti Number(s), taken from (Munch, 2017)

Most TDA methods use the information found in a Rips diagram, persistence barcode, persistence diagram, the Betti numbers, or some combination to alternatively represent and cluster data. There are many libraries dedicated to the application of TDA including Dionysus 1 & 2 (Morozov, 2020), Gudhi (Clement, et al., 2014), giotto-tda (Tauzin, et al., 2021). Of specific interest to this paper is the TDA implementation in the Topological Mode Analysis Tool (ToMATo) algorithm (Chazal, et al., 2013), Garin & Tauzin's TDA pipeline (2019), and Mapper (Singh, et al., 2007).

2.2.1. Topological Mode Analysis Tool (ToMATo) Algorithm. ToMATo is a TDA clustering algorithm that uses persistence barcodes, to allow clusters to "merge" together into their final form (Chazal, et al., 2013). The version of the algorithm used in this study is TDAToolbox (Dindin, 2019), which requires one input, an estimated number of clusters. However, ToMATo treats this as a maximum and may reduce the number of final clusters depending on its results.

2.2.2. Garin & Tauzin's TDA Pipeline. In a different implementation, Garin and Tauzin created a generic pipeline that uses giotto-tda (see (Tauzin, et al., 2021)) to classify and cluster data in point cloud form (Chazal, et al., 2013). Rather than relying on persistence barcodes as ToMATo does, this pipeline uses filtrations to derive the persistent homology, which was originally applied to the MNIST digits data set (Chazal, et al., 2013). To classify data, the training and fitting of the pipeline are followed by Scikit-learn's random forest classifier ensemble method (Chazal, et al., 2013).

2.2.3. Mapper. The Mapper algorithm has three main steps: (1) the placement of input data in one or more bins, (2) inter- and intra-clustering within and between bins, and (3) finally, the creation of a new graphical network where clusters act as vertices and interactions between clusters act as edges (Bihl, et al., 2020). Mapper's most popular Python implement is KeplerMapper, which is a part of the larger suite of libraries, scikit-tda (see (Saul & Tralie, 2019)) (van Veen, et al., 2019).

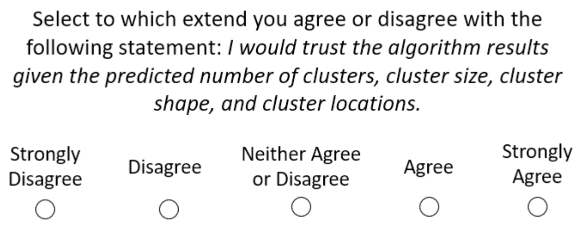
3. Clustering Comparison Framework

Prior works compared clustering methods based on: accuracy, Rand and Jaccard Indexes, and/or accuracy gain (Islambekov & Gel, 2019) (Akcora, et al., 2020) (Lyu, 2022). To answer one of the research questions, this work proposes that the quality of a clustering method can be evaluated across three dimensions: accuracy, computational time, and visual trustworthiness. When used for unsupervised classification, true accuracy is of course unknown; however, the other two dimensions would be available.

The two quantitative metrics are accuracy and computation time, standard algorithm comparison metrics. Accuracy is considered as the number of data points correctly classified (as determined by the original data labels) divided by the total number of data points. Accuracy is represented by a percent and a higher accuracy means better results. Computation time is defined as how long it took for the algorithm itself to run in seconds. Any time used for data preprocessing and/or graphing is not included in this metric. Ideally computational time should be low meaning the algorithm runs quickly.

This analysis presents "visual trustworthiness" as a qualitative assessment of the clustering results. Visual trustworthiness is based on the heuristic principal of "trustworthiness" formulated in (Combs, et al., 2020). This metric category is subjective and presents the human subject is presented with the question in Figure 4 for each data plot shown in Figure 5. The subject is reliant only on the visual cues shown in the data plots and not presented with any other statistical aspects such as accuracy or computation time. The question aims to understand how the human interprets the quality of the predicted clusters and whether the results seem trustworthy given the predicted:

- number of clusters (Does the predicted number of clusters seem appropriate for the data visually?),
- cluster size (Does the number of data points in each cluster seem unbalanced or not properly distributed given the data?),
- cluster shape (Do the clusters have odd/inappropriate boundaries given the data?), and
- cluster locations (Are the clusters located where you think they should be?).



Select to which extend you agree or disagree with the following statement: *I would trust the algorithm results given the predicted number of clusters, cluster size, cluster shape, and cluster locations.*

Strongly Disagree	Disagree	Neither Agree or Disagree	Agree	Strongly Agree
○	○	○	○	○

Figure 4. Visual Trustworthiness Evaluation

Equating "strongly disagree" to a 1 and "strongly agree" to a 5 on a 5-point Likert scale, the human subject's responses for each data plot were recorded. These values were averaged for each algorithm across all 6 data sets in Figure 5. Since clustering is largely visual in nature, human perception is important and not measurable in typical quantitative metrics. A higher visual trustworthiness score is ideal in this study.

Finally, a ranking from 1-11 was assigned to all the algorithms for each metric, accuracy, computation time, and visual trustworthiness. All three metrics are described by an average for each algorithm looking at its performance across all the data sets; therefore, the ranks were determined based on these averages. For accuracy and visual trustworthiness, a rank of 1 was assigned to the highest value and a rank of 11 was assigned to the lowest value for each algorithm. For computation time, where a lower value is best, a rank of 1 was given to the smallest time and a rank of 11 was given to the algorithm with the largest time. An average of these ranks for accuracy, computation time, and visual trustworthiness were taken and then, ranked as well to give the overall algorithm ranking.

4. Simple Comparison of Methods

Inspired by (Scikit-learn, 2022)'s comparison on novel 2D data, ten clustering algorithms plus TDA were compared on six novel data sets: circles, moons, blobs with different variances, anisotropic, blobs with the same variances, and random. Each data set consisted of 1500 instances and a random state of 10 was used except for the anisotropic data, which used 11. For the non-TDA algorithms, the default parameters used by scikit-learn was used. The TDA algorithm used throughout is ToMATo.

The results of the 11 clustering algorithms are shown graphically (with computation time shown in the bottom right corners) in Figure 5. The accuracy, computational time, and visual trustworthiness scores for each model on all six data sets is presented in Table 1. In the visual trustworthiness section, the number of clusters metric wherein the evaluator answered "Yes" is totaled for each algorithm given its performance on the data sets, which allows for a maximum score of 6. The cluster quality scores were averaged across all 6 data sets for each algorithm. The first author served as the evaluator, so *n*=1 for this study.

The circles' data set had the second-lowest average accuracy (71.6%) and the longest average computational time (1.98s). The results were relatively bipolar with 5 algorithms displaying perfect performance (including TDA); however, the remaining algorithms ranged from 21.8% to 74.4% accuracy. Most algorithms correct estimated two clusters; however, many split the data into unideal half-circle-like clusters.

The moons data set at a 91.2% average accuracy and the longest average computation time of 1.6s. Similar to the circles' data set, the same 5 algorithms showed perfect performance with most of the others in the 80-90% range (except for affinity propagation's 68.4% accuracy). Again, two clusters were identified by most algorithms; however, several misclassified inner edges of the two "moons."

The blobs with variance data set showed an average of 92.4% accuracy and an average computation time of 0.88s, which is approximately middle-of-the-road compared to the other data sets. Performance across the board was fairly high with 8 algorithms in the 90-100% range with the lowest performance being BIRCH at 60%. This is one of two data sets that DBSCAN and OPTICS performed less than 100% on, due to their built-in ability to detect outliers (which are effectively a fourth class) shown in black in Figure 5. Most algorithms identified correctly three clusters, but some of the edges proved to be difficult to cluster. Since this analysis treats the outliers identified by DBSCAN and OPTICS as a fourth class, they were penalized for it in the visual trustworthiness metric.

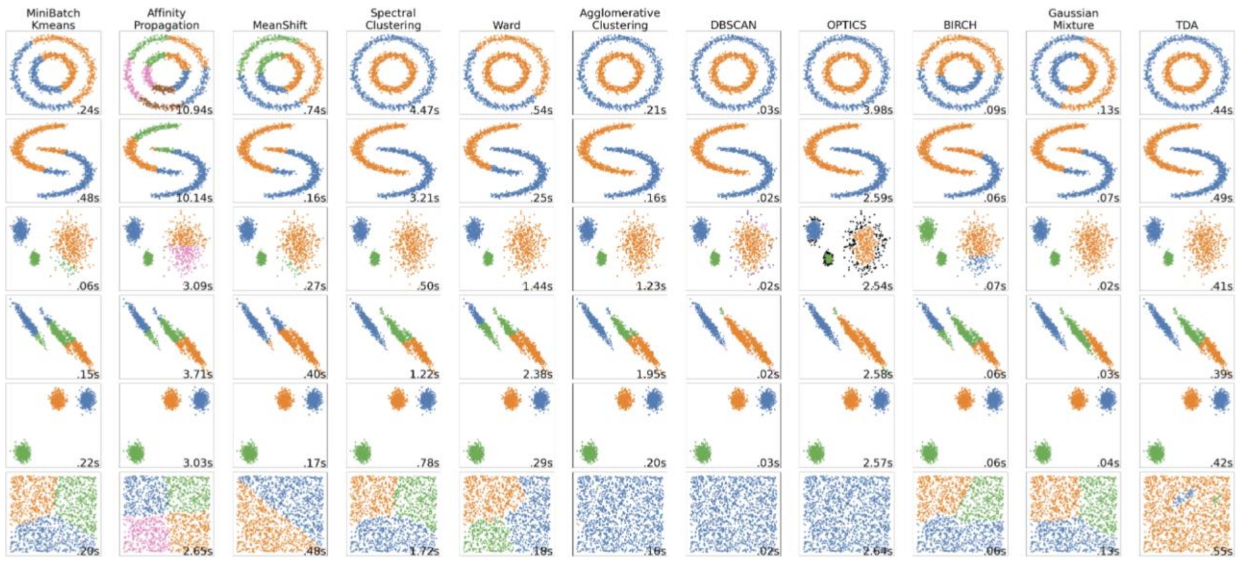


Figure 5. Artificial Data Scatter Plots

Table 1. Accuracy, Computation Time, and Visual Trustworthiness

Algorithm	K-means	AP	MS	SC	Ward	AC	DBSCAN	OPTICS	BIRCH	GM	TDA
Accuracy (%)											
Circles	51.2	21.8	38.5	100	74.4	100	100	100	51.3	50.1	100
Moons	84.2	68.5	87.3	100	90.3	100	100	100	87.3	85.4	100
Blobs w/ Vari.	97.2	84.7	97.9	100	100	100	95.9	80.7	60	100	99.9
Anisotropic	83.9	83.6	65.7	91.2	69.3	88.9	66.3	66.7	83.4	99.2	93.3
Blobs w/o Vari.	100	100	100	100	100	100	100	100	100	100	100
Random	35.6	26.7	50.1	40.1	40	99.6	100	100	42.2	35.7	94.3
Average	75.6	64.2	73.3	88.6	79	98.1	93.7	91.2	70.7	78.4	97.9
Computation Time (s)											
Total	1.35	33.6	2.22	11.9	5.08	3.91	0.14	16.9	0.4	0.42	2.79
Average	0.39	9.6	0.63	3.4	1.45	1.12	0.04	4.83	0.11	0.12	0.78
Visual Trustworthiness (*n*=1)											
Average	3.33	2.17	3.17	4.17	3.5	4.67	4.83	4.67	2.83	4	4.5
Overall Algorithm Ranking											
Rank	7	11	10	6	8	2	1	5	9	4	3

The anisotropic data was a struggle for some of the algorithms, with even the high-achieving ones unable to obtain 100% accuracy. This is one of two data sets with touching clusters (the other being blobs with variance), which most of the algorithms have a difficult time clustering. Impressively, Gaussian mixture, TDA, and spectral clustering all had an accuracy above 90%. However, as shown in Figure 5, many of the algorithms divided up the data such that the clusters ran horizontally rather than at the slight angle the true clusters are. Yet again, the number of clusters did not prove to be too challenging; however, excluding Gaussian Mixture, all the algorithms struggled with the quality of the clusters with most combing the right-most two to some degree.

The blobs with no variance were so clearly dispersed and dense, that all 11 algorithms perfectly predicted the three original clusters. Despite having three clusters, this data set had the second-lowest average computation time of 0.71s. As this was arguably the easiest data set, all eleven algorithms received full credit for both visual trustworthiness metrics.

The final row of the accuracy section, showing the results of the uniform random distribution was unique in the sense that there was only one cluster. This was by far the most difficult data set given its average accuracy of 60.3%. Due to the nature of the data set, algorithm performance is better accessed through the number of clusters identified rather than overall accuracy. DBSCAN and OPTICS correctly assumed one cluster, followed by agglomerate clustering and mean shift assuming two clusters. Five algorithms assumed three clusters and two algorithms (affinity propagation and TDA) assumed four clusters. It is worth noting that despite TDA guessing four clusters, 94.3% of the instances all fell within one large cluster in contrast with many of the other algorithms attempting to have equally-distributed clusters. Yet again, this was a struggle with most algorithms identifying more clusters than what existed. For the cluster quality metric, scores were largely based on the number of clusters identified since the data set was uniformly distributed.

In comparison with the other algorithms, TDA has the second-highest accuracy of 97.9% and the sixth-lowest total computational time at a total runtime of 2.79s for all six data sets. Despite TDA's high accuracy, both agglomerative clustering and DBSCAN obtained a higher accuracy with lower computation times of 2.26s and an impressive 0.13s, respectively. For the visual trustworthiness metrics, TDA was in an 8-way tie for 2nd place for the number of clusters, and ranked 4th in regards to cluster quality. However, real-world data is quite different, which is the next way TDA is evaluated.

5. Topological Data Analysis Application on Real-world Data

In a different assessment of TDA, it is applied to two frequently-used real-world data sets, the Palmer Penguins and the Modified National Institute of Standards and Technology (MNIST) digits data sets. The Palmer Penguins data set has been dubbed the "new" Iris data and details characteristics of three types of penguins (Horst & Hill, 2020). The MNIST data set is a classic data set consisting of handwritten numbers between 0 to 9 (LeCun, et al., 1998).

5.1. Palmer Penguins

Named for the Palmer Archipelago in Antarctica area, the Palmer Penguins size data set consists of the target variable, species, and six other variables: island (nominal; 3 levels), culmen (bill) length (continuous), culmen (bill) depth (continuous), flipper length (continuous), body mass (discrete), and sex (nominal; 2 levels) (Horst & Hill, 2020). There are 344 total instances; however, 11 had missing data and were removed from the data for this study.

Since ToMATo can only be applied to 2D numerical scatter data; three plots comparing the culmen length (on the x-axis) to the remaining, culmen depth (left column of Figure 6), flipper length (middle column), and body mass (right column) were constructed to see how TDA performs. The true clusters represented by the species target variable are shown in the top row of graphs in Figure 6, with the TDA-predicted clusters in the bottom row. TDA produces fairly high accuracies given the clusters: culmen length vs. depth had an 80.2% accuracy, culmen length vs. flipper length had a 95.6% accuracy, and culmen length vs. body mass had an 88.9% accuracy.

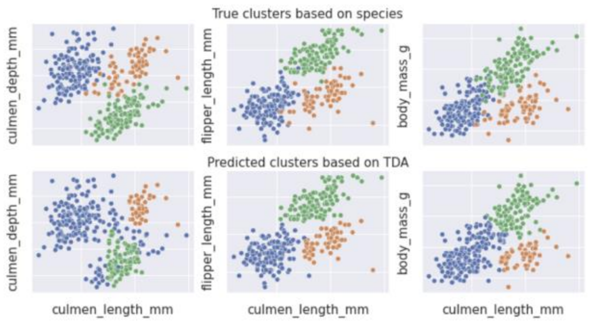


Figure 6. True and TDA-predicted Palmer Penguin Clusters

5.2. MNIST Digits

The MNIST Digits data set consists of 70,000 handwritten single digits split 60,000 for training and 10,000 for testing (LeCun, et al., 1998). Due to limited computational resources, only 10,000 examples were used for training, but all 10,000 examples were used for testing. Despite the small training size, the TDA pipeline performed quite well with an accuracy of 94.82% on the test data. The highest performing individual class was 0, with 97.8% accuracy, with the lowest-performing being 4, with an accuracy of 92.3%. Regarding the latter's sub-par performance (in comparison), 3.7% of 4's was mistaken as 9's, and in vice versa, mistaking a 9 for a 4, was the second most common error with 2.3% occurrence. As discussed in Section 2.2, persistent diagrams and barcodes are based on the number of connected components, loops, and holes in a given figure. Given a "closed" number 4, such as displayed by this font, its format simplifies to one loop and one component, just as a 9 is. Therefore, it is reasonable for the TDA pipeline to mix the two numbers up. Other spots of trouble include confusing 2's and 7's and interestingly, 8's and 9's.

Though TDA's performance based on accuracy was satisfactory, the required computation time is exceptionally long. It took over 14 minutes to train the TDA pipeline (and random forest classifier) and then, and an additional 13 minutes for running the test data. Considering TDA's accuracy is good, but not great (as many models have performed much higher), it is hard to justify the use of this TDA pipeline, given the settings used, for the MNIST data set in terms of practicality.

6. KeplerMapper Visualization & Implementation

The last TDA algorithm of interest is Mapper (see (Singh, et al., 2007)), which is implemented in Python via the KeplerMapper library (van Veen, et al., 2019). KeplerMapper has several hyperparameters (with many different settings) that can drastically change the resulting diagram, which is typically used for data visualization.

6.1. Hyperparameters & Settings

In the KeplerMapper implementation, there are three main hyperparameters: the projection (also called a lens), the distance matrix, the scaler, and the number of cubes (van Veen, et al., 2019). The first three are inputs to KeplerMapper's *project/fit_transform* functions, with the latter an input to the *Cover* class. It also uses a clustering algorithm, which by default was DBSCAN, which uses the parameters, *eps* and *min_samples* described in section 2.1.6.

The projection affects the input data and how it is "projected" to the Mapper algorithm. Unlike ToMATo which needs 2D scatter data and the TDA pipelines which require many filtrations of 2D images, KeplerMapper can "summarize" data with many variables through different projections. These projections can be a Scikit-learn class with the *fit_transform* function in addition to other measures (sum, mean, etc.). Projections can also turn 3D data into 2D data by only considering two variables. For both examples in this section, the t-distributed Stochastic Neighbor Embedding (t-SNE) method was used.

The distance matrix determines how the distances between data instances are calculated. By default, Euclidean distance is used, but this may be any string corresponding to a pairwise distance metric. This is applied before the projection.

The scaler by default is Min-max; however, this field may be left blank if the user wishes for no post-map scaling. Scaling takes place after the data is projected using the districted distance matrix. For our purposes, only the Min-max scaler is used.

The number of cubes, symbolized as *num_cubes*, is the number of hypercubes to be created along each dimension. Data points within the hypercubes are clustered and these clusters become nodes on the final KeplerMapper diagram. These clusters/nodes are connected by lines, called edges when one point appears in multiple clusters/nodes. Optionally, the *Cover* class can be assigned a value for *perc_overlap*, which is the "amount of overlap between adjavent cubes along one dimension." The settings used to generate Figures 8 and 9 are shown in Table 2.

Table 2. KeplerMapper Settings

Figure	*Eps*	*Min_ samples*	*Num_ cubes*	*Perc_overlap*
7	0.3	4	20	0.3
8(a)	0.3	15	35	0.4
8(b)	0.3	5	15	0.52

6.2. Palmer Penguins

Rather than using only the numerical variables as done in Section 4.2, all variables were used in this analysis. However, the nominal variables were represented with numerical labels. The resulting figure is shown in Figure 7. The node size represents how large that node/cluster is and the color represents the "average" true label of the node. It is an "average" because if the node consisted of heterogeneous data (from two or more species), the color would adapt such that the color reflects all of the true labels represented. Ideally, alike items would be clustered together in "chains" or "superclusters," meaning nodes that are further connected by edges. However, in this case TDA identified three "superclusters" which correlate to three classes; however, they do not align well with the data point's true clusters as shown in in the coloring. The Adelie penguins are purple, the Chinstrap penguins are teal, and the Gentoo penguins are yellow; however, if a cluster consists of multiple types the colors correspondingly blend. To address one of the research questions, TDA identifies three "superclusters;" however, they do not accurate correspond with the three classes in this particular view of the data.

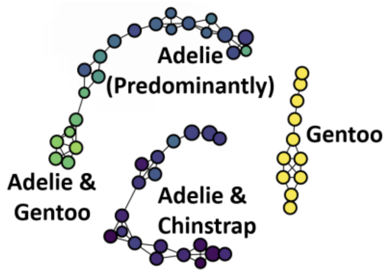


Figure 7. Palmer Penguins KepplerMapper Diagram

6.3. MNIST Handwritten Digits

The creators of KeplerMapper already provided the Python code for exploring the MNIST data set. The resulting figure is shown in Figure 8(a). Except for a few, most of the data has accurately been chained together into "superclusters" even if in different individual clusters (remember the nodes of the graphs are the clusters identified by KeplerMapper). Four numbers, 1, 7, 8, and 9, had multiple "superclusters" but they were all relatively homogenous in terms of representing the desired number. There are a few instances of heterogeneous clusters, such as the stand-alone 7 cluster, which has some 9's in addition to being primarily 7's. Overall, KeplerMapper does a good job of identifying clusters and "superclusters" concerning the data's true labels.

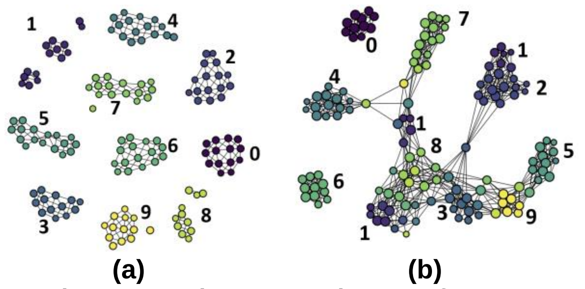


Figure 8. KeplerMapper Diagram of MNIST

Alternatively, by changing KeplerMapper's settings, Figure 8(b) can be constructed from the same data. Figure 8(b) provides a more equivocal diagram. There is one main "supercluster" that connects all the digits with the exception of 0 and 6's superclusters. Clusters are represented by the colored nodes and considering each data point can belong to multiple clusters, this allows for connections to be between nodes, identified by lines. Even within the primary "supercluster" one can see how alike nodes tend to cluster together as best exemplified by the 7 and 2. However, other data points, such as those in the "1" class are often misclassified as 2, 3, and 8.

To address the final research question, TDA allows for many different representations of the same data through its clusters (nodes) and superclusters (connected nodes). These views are facilitated by the adjustment of various clustering, scaling, and projections which is particularly valuable for multi-dimensional data (as such with the MNIST data set) where accurate and complete visual representation can be difficult.

7. Conclusions

This paper presents the use of topological data analysis (TDA) for data classification and visualization. First, background information on clustering and different algorithms were presented. Then, the Topological Mode Analysis Tool (ToMATo) algorithm (Dindin, 2019), Garin & Tauzin's TDA pipeline (2019), and Mapper algorithm (Singh, Mémoli, & Carlsson, 2007) was applied to 2D scatter data (novel and Palmer Penguins (Horst & Hill, 2020) data sets) and/or the MNIST handwritten digits data set (LeCun, Cortes, & Burges, 1998). Lastly, the Mapper algorithm was explored as implemented in the Python library, KeplerMapper.

First, the cluster algorithms were compared through the creation of three metrics: accuracy (was each data point correctly classified?), computational time (how long did classification take?), and visual trustworthiness (how well does classification meet a human's visual expectation?) which all measure different aspects of the selected clustering methods. Wherein, TDA ranked second (out of eleven) in accuracy, sixth in computational time, and fourth in visual trustworthiness. Overall, TDA was third after DBSCAN and agglomerative clustering, respectively, on the artificial simple data comparison study. Then, two real-world data sets, Palmer Penguins and MNIST, were explored through the application of the Mapper algorithm. Mapper's implementation of TDA produces clusters (colorful nodes in Figure 7 and Figure 8) and what's dubbed "superclusters" in this paper. "Superclusters" are essentially clusters of clusters connected by lines to one another. Ideally these "superclusters" correspond to the original class; however, depending on the hyperparameters, this is not always the case. Therefore, TDA results may be directly interpreted in terms of the clusters (as the results for ToMATo and the TDA Pipeline) or need more abstract extraction (as shown with Mapper). Shown throughout the paper, the best TDA algorithm is likely dependent on the data used and the intended end goal. Classification is best done through ToMATo and TDA Pipeline; however, if open to more fuzzy clustering, Mapper is better. Regardless of the specific algorithm, TDA is appropriate for a variety of data sets especially those with high dimensionality.

One clear line of future work would be to compare other popular clustering algorithms to TDA on different data sets, 2D data, images, point clouds, etc. Other areas of interest would be to dive deeper into Mapper's visualization results and see how they can be better optimized for classification.

8. Acknowledgements

The views expressed in this paper are those of the author and do not represent the views of any part of the US Government. This work was cleared for unlimited release under: AFRL-2022-4453.

9. References

Akcora, C. G., Li, Y., Gel, Y. R., & Kantarcioglu, M. (2020). BitcoinHeist: Topological data analysis for ransomware prediction on the Bitcoin blockchain. *29th International*

Joint Conference on Artificial Intelligence, (pp. 4439-4445). Yokohama.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., & Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. *ACM Sigmod Record, 2*(49-60), 28.

Bihl, T. J., Gutierrez, R. J., Bauer, K. W., Boehmke, B. C., & Saie, C. (2020). Topological data analysis for enchancing embedded analytics for entireprise cyber log analysis and forensics. *53rd Hawaii International Conference on System Sciences.* Grand Wailea.

Chazal, F., & Michel, B. (2021). Introduction to topological data analysis: Fundamental and practical aspects for data scientists. *Frontiers in Artificial Intelligence, 4,* 1-28.

Chazal, F., Guibas, L. J., Oudot, S. Y., & Skraba, P. (2013). Persistence-based clustering in Riemannian manifolds. *Journal of the ACM, 60*(6), 1-38.

Clement, M., Boissonnat, J.-D., Glisse, M., & Mariette, Y. (2014). The Gudhi library: Simplicial complexes and persistent homology. *4th International Congress on Mathematical Software.* Seoul.

Combs, K., Fendley, M., & Bihl, T. (2020). A preliminary look at heuristic analysis for assessing artificial intelligence explainability. *WSEAS Transactions on Computer Research, 8,* 61-72.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological), 39*(1), 1-22.

Dindin, M. (2019). TdaToolbox. Github.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *2nd International Conference on Knowledge Discovery and Data Mining, 96,* pp. 226-231. Menlo Park.

Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. *Science, 315,* 972-976.

Fukunaga, K., & Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory, 21*(1), 32-40.

Garin, A., & Tauzin, G. (2019). A topological "reading" lesson: Classification of MNIST using TDA. *18th IEEE International Conference On Machine Learning and Applications.* Boca Raton.

Ghosh, J., & Liu, A. (2009). K-means. In X. Wu, & V. Kumar (Eds.), *The top ten algorithms in data mining* (pp. 21-36). Boca Raton: Taylor & Francis Group.

Ghrist, R. (2008). Barcodes: The persistent topology of data. *Bulletin of the American Mathematical Society, 45*(1), 61-75.

Horst, A. M., & Hill, A. P. (2020). palmerpenguins: Palmer archipeglago (Antarctica) penguin data. doi:10.5281/zenodo.3960218

Islambekov, U., & Gel, Y. (2019). Unsupervised space-time clustering using persistent homology. *arXiv:1910.11525v1,* 1-9.

Kan, A. (2017). *Clustering. Gaussian Mixture Model.* https://trevorcohn.github.io/comp90051-2017/slides/13_clustering_gmm.pdf

LeCun, Y., Cortes, C., & Burges, C. J. (1998). The MNIST database of handwritten digits. New York City: Courant Institute.

Lloyd, S. P. (1957). *Least squares quantization in PCM.* Atlantic City: Bell Laboratory Tech. Note.

Lyu, M. (2022). Optimal base station network based on topological data analysis. *International Journal of Modeling and Optimization, 12*(1), 8-15.

Morozov, D. (2020, November 24). *Dionysus 2*. (MRZV) Retrieved March 30, 2022, from https://www.mrzv.org/software/dionysus2/

Munch, E. (2017). A user's guide to topological data analysis. *Journal of Learning Analytics, 4*(2), 47-61.

Nielsen, F. (2016). Hierarchical clustering. In *Introduction to HPC with MPI for Data Science* (pp. 221-239). Springer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.

Rhys, H. I. (2020). Clustering based on density: DBSCAN and OPTICS. In *Machine learning with R, the tidyverse, and MLR.* Shelter Island: Manning Publications Co.

Rokach, L., & Maimon, O. (2005). Clustering methods. In *Data mining and knowledge discovery handbook* (pp. 321-352). Boston: Springer.

Saul, N., & Tralie, C. (2019). Scikit-TDA: Topological data analysis for Python. Zenodo.

Scikit-learn. (2022). *Clustering*. (Scikit-learn) Retrieved March 24, 2022, from https://scikit-learn.org/stable/modules/clustering.html#overview-of-clustering-methods

Scikit-learn. (2022). *Gaussian mixture models*. https://scikit-learn.org/stable/modules/mixture.html#mixture

Sculley, D. (2010). Web-scale k-means clustering. *19th international conference on world wide web.* New York.

Singh, A. (2010, November 22). *Spectral Clustering.* https://www.cs.cmu.edu/~aarti/Class/10701/slides/Lecture21_2.pdf

Singh, G., Mémoli, F., & Carlsson, G. (2007). Topological methods for the analysis of high dimensional data sets and 3D object recognition. *Eurographics Symposium on Point-based Graphics, 2.* Prague.

Tauzin, G., Lupo, U., Tunstall, L., Burella Pérez, J., Caorsi, M., Medina-Mardones, A. M., . . . Hess, K. (2021). giotto-tda: A topological data analysis toolkit for machine learning and data exploration. *Journal of Machine Learning Research, 22*, 39-46.

van Veen, H. J., Saul, N., Eargle, D., & Mangham, S. W. (2019). Kepler Mapper: A flexible Python implementation of the Mapper algorithm. *Journal of Open Source Software, 4*(42), 1315.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing, 17*(4), 395-416.

Ward Jr., J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association, 58*(301), 236-244.

Wasserman, L. (2018). Topological data analysis. *Annual Review of Statistics and its Application, 5*, 501-532.

Zhang, T., Ramakrishnan, R., & Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases. *ACM Sigmod Record, 25*(2), 103-114.