A Novel Approach to Dynamic Unsupervised Clustering

Christopher Heinlen, Mark Volpi
Lone Star Analysis
Addison, TX
cheinlen@lone-star.com, mvolpi@lone-star.com

Randal Allen
Lone Star Analysis
Orlando, FL
rallen@lone-star.com

ABSTRACT

A criticism of Artificial Intelligence and Machine Learning (AI/ML) techniques is brittleness. This means that an AI/ML technique may perform well when analyzing a known data set, but performance will degrade when new data is introduced. While this degradation is apparent when new data belongs to an existing category, degradation is even more pronounced when data from a previously unseen category is introduced. Additionally, many AI/ML solutions do not have a mechanism to detect outliers or noise; instead, they either force these data points into categories where they do not belong or require them to be excluded entirely.

The system described in this paper, Dynamic, Unsupervised Clustering by Algorithmic Triangulation (DUCAT), does not have these shortcomings. It is designed to handle streaming data sets and to add new classification types as new data types are observed. It is not limited to a certain number of categories, and it does not need to be retrained as new data types are introduced. Additionally, it can detect noise and categorize it as such.

Clusters are found using spatial relationships between data points to determine whether a grouping exists within a dataset. Once a cluster is found, the system defines a region in space that the cluster occupies, allowing for new data points to be checked for cluster inclusion. This approach has been verified by segmenting a dataset of simulated radar pulses into a series of portions, which are individually fed into the system. The efficacy of the system is shown by a confusion matrix comparing the assigned labels to truth and displaying classification cohesion across portions.

This system was originally developed to find and identify novel radar pulses in order to offload classification and noise filtering work. This paper will explore alternative applications to scenarios beyond radar classification.

ABOUT THE AUTHORS

Christopher Heinlen is a System Engineer at Lone Star Analysis with experience in AI/ML development, particularly within the realm of unsupervised clustering. He has also conducted significant research into signal processing within the domains of radar, PNT, seismology, and others. He graduated Summa Cum Laude with a B.S. in Mechanical Engineering from the University of Texas at Arlington.

Randal Allen is the Chief Scientist of Lone Star Analysis. He is responsible for applied research and technology development across a wide range of M&S disciplines and manages intellectual property. He maintains a CMSP with NTSA. He has published and presented technical papers and is co-author of the textbook, "Simulation of Dynamic Systems with MATLAB and Simulink." He holds a Ph.D. in Mechanical Engineering (University of Central Florida), an Engineer's Degree in Aeronautical and Astronautical Engineering (Stanford University), an M.S. in Applied Mathematics, and a B.S. in Engineering Physics (University of Illinois, Urbana-Champaign). He serves as an Adjunct Professor/Faculty Advisor in the MAE department at UCF, where he has taught over 20 aerospace-related courses.

Mark Volpi is the Director of Intelligence Solutions for Lone Star Analysis. He is focused on the growth, management, and development of programs intended to solve problems faced by the United States Intelligence Community and their allies. He has twenty years' experience in problem solving, system engineering, software development, real-time systems, and SIGINT processing systems. He has worked with international teams, and his experience has ranged from placing antennas in the desert to being PM on a multi-million-dollar contract. Mark received both his B.S. and his M.S. in Physics from Texas A&M, with an interest in information theory.

A Novel Approach to Dynamic Unsupervised Clustering

Christopher Heinlen, Mark Volpi
Lone Star Analysis
Addison, TX
cheinlen@lone-star.com, mvolpi@lone-star.com

Randal Allen
Lone Star Analysis
Orlando, FL
rallen@lone-star.com

INTRODUCTION & BACKGROUND

Artificial Intelligence and Machine Learning (AI/ML) techniques are generally brittle; that is, they are prone to failure if there are any discrepancies between training and application (Choi, 2021). This means that an AI/ML technique may perform well when analyzing a discrete data set, but that performance will fall apart when new, nonconforming data is added to the model. This degradation is apparent when new data belonging to existing categories is injected into the model, but it is even more pronounced with new, previously unseen types of data. Because of this, traditional AI/ML solutions, including neural nets, generally need to be retrained if a new category of data is added to the system. Additionally, many AI/ML solutions do not have a mechanism to detect outliers or noise; instead, they force these data points into categories where they do not belong. These shortcomings can be avoided by applying the system described in this paper. This system was specifically designed to handle streaming, infinite data sets and to dynamically add new classification types as new data types are seen. It is not limited to a certain number of classification types, and it does not need to be retrained as new data types are introduced, making it efficient and adaptable. Additionally, it can detect outliers and noise and categorize them as such.

There are three overarching branches of machine learning, each of which processes data differently: supervised, unsupervised, and reinforcement learning. Supervised learning is a model created with data whose input and output are known. Supervised learning can be further divided into regression techniques for continuous response prediction and classification techniques for discrete response predictions. Unsupervised learning processes unknown data and employs clustering techniques to identify patterns within that data. This type of learning can be divided into hard clustering and soft clustering. Hard clustering assigns each data point to one and only one cluster, but soft clustering can assign a data point to multiple clusters. Finally, a reinforcement learning model is trained on successive iterations of decision-making, with rewards given based on the results of those decisions. This system utilizes soft, unsupervised clustering to classify streaming data without needing to know the number of classification types within the data stream or any other prior knowledge.

Traditional machine learning deals with a static data set, but many use cases require the ability to classify data points within an endless data stream. Streaming data, as opposed to a static data set, presents distinct challenges for data classification. Ideally, one could see the entirety of a data set, but in real world, real-time scenarios, this is not an option. Hence, it is critical to have the ability to make classifications based on past and current data while also allowing for new classification categories as new data appears. Another challenge of streaming data scenarios is concept drift. Concept drift is a gradual change in the properties or features within a data stream. This can manifest as new classes appearing in a data stream at different points in time or as the features of a particular class morphing over time. But because the system generates dynamic classification types, it can overcome issues stemming from concept drift.

This system was developed due to the limitations seen in other available algorithms. It contains several key features that are not all present in other approaches:

- The ability to discover new class types and record them for future data,
- The ability to find clusters and maintain tight cluster definitions in the presence of large amounts of noise,
- The ability to automatically determine if, and how many, clusters are present in a dataset.

As a result, this system has potential applications in signal classification, image classification, human performance data, and medical data.

SYSTEM DESCRIPTION

The system consists of four distinct modules that work in concert to categorize incoming data and create new category buckets as needed. First, the incoming data is checked against existing categories—if it fits into an existing category, it is classified; if not, it is added to an unclassified pool. When the unclassified pool reaches a threshold, a novel unsupervised clustering method is run on the data to determine if a new category should be created. If a new category (or cluster) is found, the cluster is used to define a shell. New data will be checked against this shell to determine inclusion. Finally, a second pass is used to merge or split clusters and remove noise from them. This data flow is illustrated below in Figure 1. Each of these four steps—checking against existing clusters, unsupervised clustering, shell creation, and a second pass—is described below.

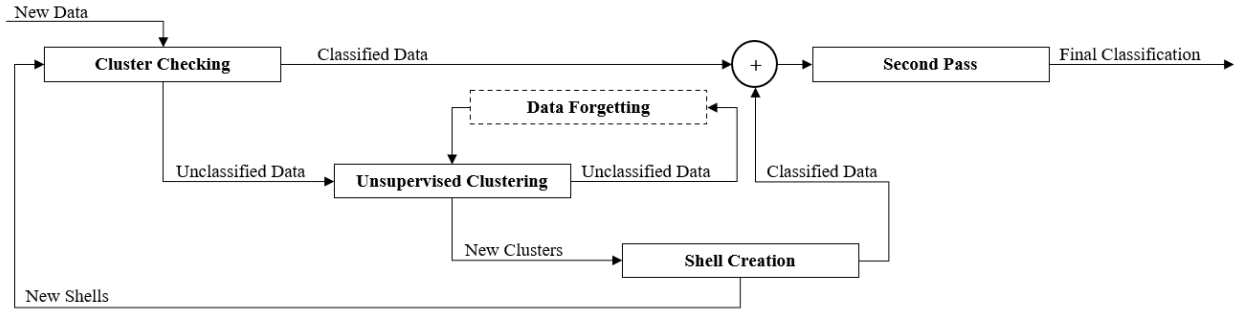

Figure 1. Data Flow

Cluster Check

When a new data point enters the system, it will first be tested against existing categories. This is accomplished by comparing the new point against a cluster's codebook and threshold. The calculation of the codebook and threshold is described in the Shell Creation subsection below. If the new data point is within the threshold distance of any of the codewords in the cluster's codebook, the data point is assigned to that cluster. A point can potentially belong to multiple clusters because categories are defined independently. This independence can result in multiple clusters overlapping. This is intentional, and the second pass, in part, tries to reconcile these overlaps. In the case that a data point does belong to a cluster, that classification is reported. If not, it is added to a pool that goes on to the unsupervised clustering portion of the system.

Unsupervised Clustering

Once the pool of unclassified data points reaches a predetermined threshold, the system attempts to find new clusters within those data points. This threshold is a function of the streaming speed of the incoming data and how often the user wants to check for newly forming clusters. This unsupervised clustering method is applicable to high-dimensional data, but for ease of visualization, this document will look at two- and three-dimensional examples. The clustering system does not need any prior knowledge of the data, and it returns groupings (or clusters) of similar data within the complete set. This form of unsupervised clustering does not depend upon prior knowledge, but if the user does have prior knowledge, they can add additional thresholds and discriminators to this process. Additionally, this unsupervised clustering method isolates clusters from surrounding noise so that every point need not belong to a found cluster. Identifying clusters within the unclassified pool is critical, as it allows the system to dynamically categorize data by type and isolate relevant data from noise.

The unsupervised method used in this system relies on the Parzen Window Density Estimation (PWDE) to determine cluster seeds and distance thresholds. The PWDE is defined with the following equation:

$$p(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V}\phi\left(\frac{x_i - x}{h}\right)$$

where ϕ is a window function, h is the window width, V is the volume of the window, n is the number of points in the data set, x is location at which the density estimation is evaluated at, and x_i are the points in the data set. The simplest PWDE uses a hypercube as the window, in this case $V = h^d$, where d is the number of dimensions the data set contains. But while a hypercube provides a simple PWDE implementation, the window function is not restricted to a hypercube and can take on any geometry. The system uses a hypersphere for the window function. In effect, the PWDE slides a window across a search space and determines the number of points within the window at a given point in space. This number is then used to determine a pseudo-density at that given point (Duda, 2001). A two-dimensional data set and its resulting PWDE is shown below in Figure 2.

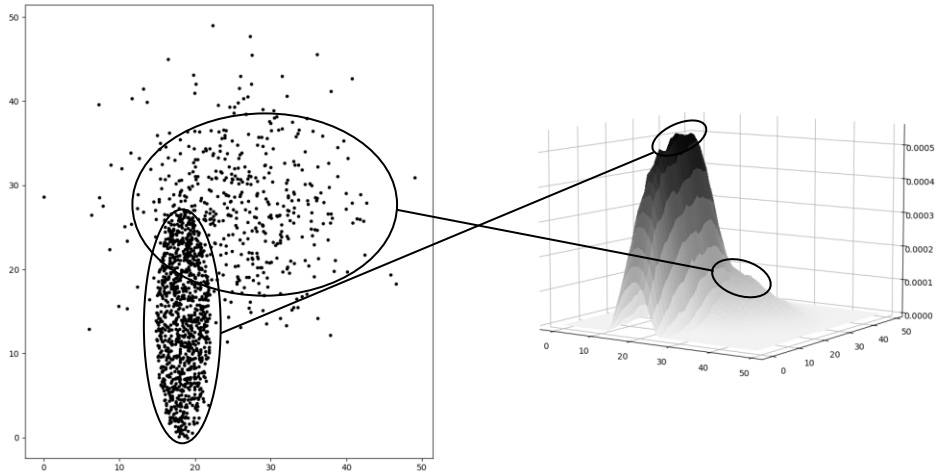

Figure 2. Two-Dimensional Point Set and Resulting PWDE

Fully defining the PWDE for a given search space is a computationally expensive task, especially with high-dimensional problems. A simple grid search becomes exponentially more difficult with every dimension added, so a way of targeting areas of interest is necessary to keep the algorithm fast. By evaluating the PWDE only at the location of points within the dataset, the complexity of the problem becomes tied to the length of the data set rather than the dimensionality of the data, so evaluation at high dimensions becomes feasible.

Once the PWDE is computed, a directional network is created to find the points that represent the peaks of the PWDE. Delaunay triangulation is a type of triangulation that (1) generates simplices between points within a point set and (2) seeks to minimize the maximum interior angle while maximizing the minimum interior angle of the newly generated simplices. In other words, it attempts to avoid slivers and tends towards equilateral simplices (Li, 2004). The Delaunay triangulation of a two-dimensional data set can be seen below in Figure 3. The Delaunay triangulation serves to determine the neighboring points of each point in the set, and the edges of the generated simplices become the connections in the network. The direction for each edge is determined by the PWDE values at each point and flows towards greater PWDE values. With this network, we can determine the PWDE peaks by finding the points within the set that only have incoming connections. Each of these peaks are considered cluster seeds.

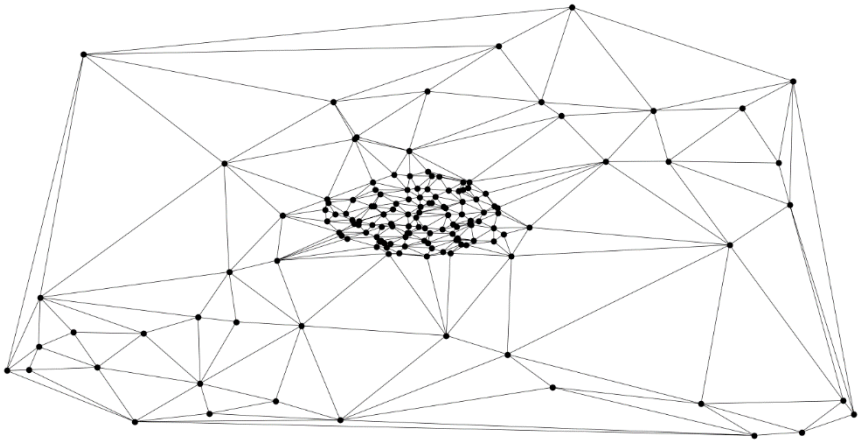

Figure 3. Delaunay triangulation of a cluster and noise.

Using the seeds as starting points, clusters are defined. First, a threshold is determined by the PWDE value at the seed point. This threshold is computed by determining what value of *h* will result in the same pseudo-density value while having only one data point fall within the window. This threshold value is indicative of the spacing of data points at the densest portion of the cluster, and it must be augmented to increase so that it is representative of the entire cluster. With the threshold determined, it is now possible to define the cluster by looking at the subset of the Delaunay triangulation edges that are less than the threshold. The points that remain connected to the seed using this reduced network are considered part of that seed's cluster.

There are two parameters that define whether a cluster is worth reporting or not. The first is the minimum cluster size; this parameter quantifies the minimum number of occurrences needed to define a new data type, or cluster. Any clusters that are smaller than this threshold are reclassified as noise. This parameter can be set at the user's discretion. The second parameter is the maximum cluster percent. This parameter is a set percentage, generally close to 1. If a cluster makes up a greater percentage of the whole dataset than the value of the parameter, the cluster is reclassified as noise. Hence, this parameter prevents a dataset that is comprised of only noise from being classified as one large cluster. Additional methods of culling out clusters can be implemented here if there is any leverageable prior knowledge about the data being analyzed. These additional methods are not necessary for the system and will be unique to a particular use case. For example, if the total possible range of a given feature is known and if clusters are expected to have a standard deviation much smaller than that total range, a threshold could be added to prevent clusters that span the entire range from being generated.

It is possible for one seed's cluster to be a subset of another seed's cluster, even with the two hyperparameters described above. A simple method of rectifying that case is to report the larger cluster; this will prevent the system from being too conservative with its cluster definitions and neglecting fringe points within a cluster. More sophisticated methods for disambiguation have been developed and could also be pursued. For example, if a cluster is identified with multiple seeds, the cluster points can be analyzed in isolation. The shape of the isolated points' PWDE can be used to determine if the cluster needs to be split or not. A uniform or Gaussian PWDE implies a single cluster, while a discontinuity, especially in between seeds, would imply that two disparate clusters were merged. While this approach will lead to a higher degree of confidence in the resulting clusters, this advantage comes at a processing cost. Any additional checks will slow down the system, so the trade-off between speed and a potential increase in accuracy must be weighed.

This system uses a novel approach to unsupervised clustering in order to avoid the shortcomings noted in other existing algorithms. *k*-means is a well-known and popular method, but it requires knowledge of how many clusters exist in a dataset and does not account for noise. *k*-means partitions a data set into a predetermined number of classes, and every data point within the set is assigned to one of the classes. Because of this, it is not compatible with the problems this system sets out to solve. Another algorithm, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and its extension, Hierarchical DBSCAN (HDBSCAN), are both sophisticated and capable algorithms that accomplish

a similar task as the unsupervised clustering employed by this system (Campello, 2013). In fact, in the original iterations of this system, HDBSCAN was used as the unsupervised clustering algorithm, but HDBSCAN's efficacy suffers in datasets with a high degree of noise. To illustrate this point, Figure 4 compares the results of this system's unsupervised clustering algorithm and HDBSCAN both classifying a dataset containing two clusters and noise. This dataset is synthetic and was specifically generated to contain two obvious clusters surrounded by a large amount of noise. Both algorithms can identify the two clusters, but HDBSCAN's cluster definitions include much of the surrounding noise. This system is able to maintain a tight cluster definition.

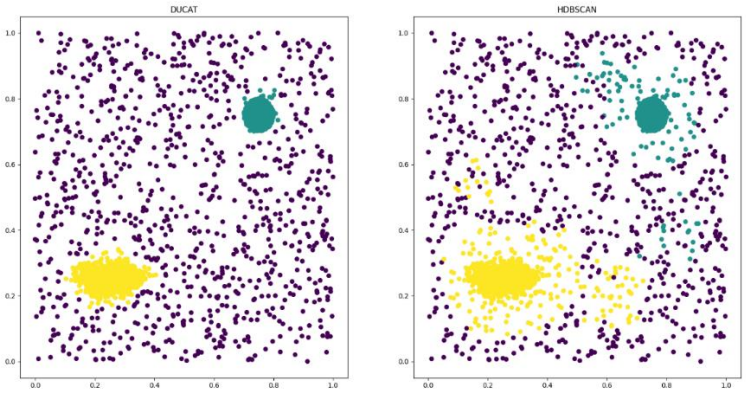

Figure 4. Novel Approach vs. HDBSCAN

Depending on the amount of data being ingested, the length of time the stream is running, and how noisy the data is, the system may need a method of forgetting or removing unclassified points. As the system runs, the unclassified pool will continue to grow as more and more outliers, or noise points, are seen. Left unchecked, the unclassified pool could grow to a size that hampers performance and slows the system, so a method of forgetting may need to be established. This method can take multiple forms, but a simple solution would be a hard cap on either time or size. For example, if a point is older than a threshold, it will be discarded, or if a pool is above a threshold, points will be removed. Alternatively, a soft cap can be implemented where after a certain threshold (either in time or in pool size) a sampling of points is removed, which retains some of the older information in the unclassified pool.

Shell Creation

Once a cluster is identified by unsupervised clustering, the data points that comprise that cluster are used to create a shell that new data points can be compared against. First, a threshold distance is determined based on the density of the cluster being analyzed. If a new point is within that threshold distance of any point within the cluster, it is likely part of the cluster. However, it would be computationally inefficient to calculate the distance of a new point from every point that makes up an existing cluster, so the system needs a method of generalizing the cluster so it can run quickly enough to keep up with real time data. Vector quantization is a method of reducing a large set of vectors down to a small, representative codebook. This codebook consists of a predetermined number of vector codewords that spatially represent the entirety of the original cluster. By utilizing a set codebook length, the processing time for a new data point will remain consistent regardless of how large the initial cluster definition is. Additionally, the codebook length can be adjusted to suit the application: a shorter codebook will result in faster processing speeds, while a longer codebook will maintain the original shape of the cluster with a higher degree of fidelity. The codebooks for two irregularly shaped clusters can be seen below in Figure 5. The codewords are shown in red, and the original data points are shown in blue.

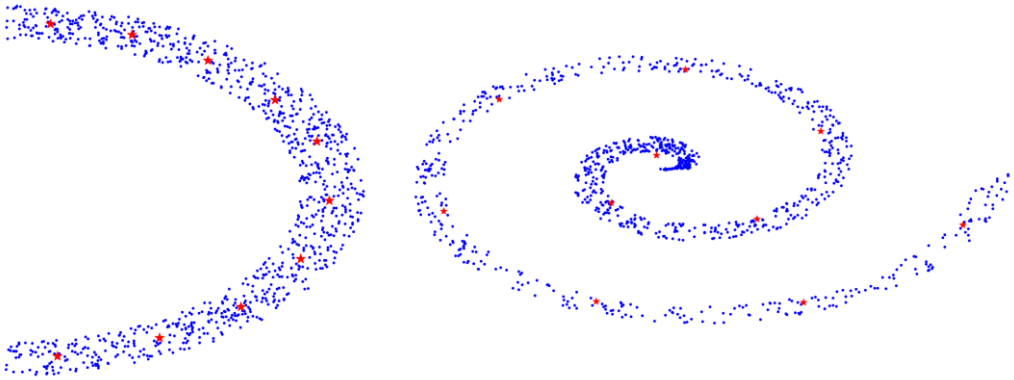

Figure 5. Clusters and Their Codewords

Once a codebook is generated, the distance threshold is determined by evaluating the distance between each codeword and its nearest neighbor. The largest of these distances is used as the threshold for the cluster. The result of this is a set of n-dimensional spheres, each with a radius of the threshold, centered on each of the codes. A point that falls within any of these spheres will be classified as part of the newly formed cluster.

Second Pass

The steps described above are sufficient to categorize data if the data is linearly separable in the dimensions being analyzed, but that is not always the case. For example, consider two radar signals that are identical in every way except their pulse repetition interval (PRI). Analyzing the pulses individually would result in the two signals being classified as the same thing, but we can analyze the resulting aggregate to determine that there are two distinct PRIs present in the cluster. Additionally, in the case of very noisy data, we may see that a significant amount of noise is being classified as part of actual clusters due to spatial proximity. If the actual members of the cluster are related to each other in time, analysis of the aggregate can help cull out the noise data points that do not actually belong to the cluster. In cases where it is necessary or useful, a second pass can reevaluate the clusters generated and the inclusion of points within those clusters in order to merge clusters, split clusters, or aid in noise discrimination.

There are multiple approaches that could be taken during this step. Features that are left out of the earlier stages can be leveraged, a reduced feature set can be utilized, or an analysis of the same feature set within the confines of a single category can be conducted. The nature of the second pass depends on the type of data being processed and any prior knowledge about the incoming data.

For Signals Intelligence (SIGINT) data, the time of arrival (TOA) of a data point is a feature that will generally not be useful during the previous steps of the system, but it can be leveraged here in the second pass. By looking at the TOA of the data points within a given category, similarities in the time or the intervals of incoming data points can be analyzed. Outliers can be reclassified as noise, and if multiple distinct groupings form from this analysis, categories can be split. Additionally, if two neighboring categories share a similar TOA and interval, those categories can be merged.

APPLICATIONS & VERIFICATION

The system described above was developed to classify radar pulses, but it can be applied to many other use cases where unsupervised clustering techniques are used today. It is well suited for streaming data that contains distinct classes and is comprised of data points consisting of some number of real valued features. Furthermore, it can be applied in scenarios where the user does not know what they are looking for and thus lacks any training data. The system also has the advantage of working without any knowledge of how many classes will appear, setting it apart from methods like *k*-means. This system is ideal for any scenarios in which a user is trying to find unknown patterns used by an adversary.

Efficacy of the system is shown through two simulations:

1. Radar use case using a synthetic dataset of radar pulse descriptor words (PDWs). Synthetic PDW datasets are used to measure detector performance, as they are critical in regression testing algorithms and in validating performance of systems used by warfighters.
2. The MNIST (Modified National Institute of Standards and Technology) dataset.

Both datasets are processed in batches to simulate a data stream and emergent clusters.

Radar PDW Dataset

The radar PDW dataset used for this simulation is comprised of 23,000 simulated radar pulses. The dataset consists of four distinct pulse types along with randomly generated noise points. The system was used to categorize the pulses based on a subset of the metadata contained in a PDW—specifically, the center frequency (FC), pulse width (PW), and bandwidth (BW). These values can be seen below in Table 1.

Table 1. Pulse Centroids

	FC (MHz)	PW (µs)	BW (MHz)	Total Pulses
Signal 1	8	5	2	6,000
Signal 2	4	8	7	6,000
Signal 3	6	1	5	3,000
Signal 4	2	3	3	3,000

The pulses were generated with a Gaussian distribution centered around the above values with a standard deviation of 0.1, and 5,000 noise points were generated with a uniform distribution between zero and ten in each of the dimensions. To simulate a data stream, the dataset is divided into twenty-three 1,000 PDW batches that are analyzed sequentially. Signal one first appears in batch one, signals two and four first appear in batch two, and signal three is introduced in batch three., The remaining batches have a random combination of one, some, or all four signals. Noise points are present in every batch. Figure 6 below shows the complete dataset plotted with each of the features as a spatial dimension. The colored points represent each of the four signals, and the gray points represent the noise points.

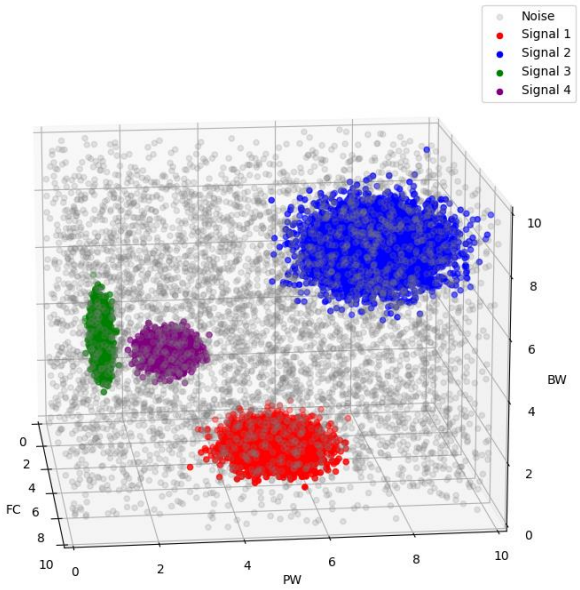

Figure 6. PDW Dataset

The results of this simulation can be seen below in the confusion matrix in Table 2. The system performs very well with this dataset, classifying each of the clusters with an accuracy of over 99% and correctly identifying 86% of the noise points.

Table 2. PDW Confusion Matrix

	Cluster A	Cluster B	Cluster C	Cluster D	Noise
Signal 1	99.57%				0.43%
Signal 2		99.20%			0.80%
Signal 3			99.37%		0.63%
Signal 4				99.90%	0.10%
Noise	3.96%	7.38%	1.26%	1.12%	86.28%

MNIST Dataset

The MNIST dataset is a classic dataset comprised of 28x28 pixel images of handwritten digits (LeCun, n.d.). This dataset has been used as a benchmark classification test for decades. As an additional test of the system's efficacy, the classification of a processed and batched version of MNIST will be presented. To convert the MNIST images into a compatible form and for ease of visualization, UMAP (Uniform Manifold Approximation and Projection) was used to reduce the images from their pixel format into a three-dimensional space (McInnes, 2020). This embedding can be seen below in Figure 7, where the plot shows the three derived features that UMAP generated. Each digit is plotted according to the color bar to the right of the image.

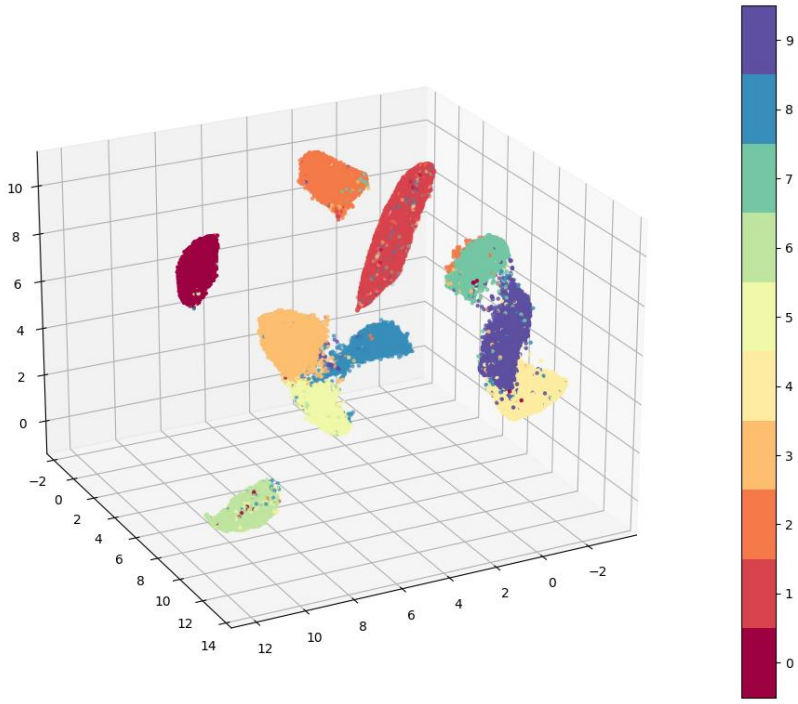

Figure 7. Embedded MNIST Digits

Since this embedding was generated without labels, it did not result in perfectly separated clusters, and some amount of overlap remained in the dataset. The data was then split into batches of 1,000 to simulate a data stream. The first three batches consisted of zeros, sixes, and sevens; batches four through six contained zeros, ones, and eights; batches seven through nine consisted of fives, sevens, and nines; batches ten through twelve were made up of twos, threes, and fours; and the remaining 58 batches were a combination of every digit. The results of the simulation can be seen below in Table 3. The columns represent the labels generated by the system, and the rows represent the actual labels. For example, 99.25% of the actual zeros and 0.01% of the actual ones were classified as a zero by the system.

Table 3. PDW Confusion Matrix

	0	1	2	3	4	5	6	7	8	9	Noise
0	**99.25%**	0.09%	0.06%	0.03%	0.03%	0.12%	0.33%	0.06%	0.01%	0.07%	0.00%
1	0.01%	**99.01%**	0.41%	0.00%	0.24%	0.00%	0.05%	0.23%	0.03%	0.22%	0.00%
2	0.64%	0.70%	**95.78%**	0.30%	0.14%	0.06%	0.17%	1.85%	0.29%	0.13%	0.01%
3	0.06%	0.18%	0.50%	**96.32%**	0.14%	5.66%	0.03%	0.67%	4.34%	0.35%	0.00%
4	0.04%	0.69%	0.01%	0.01%	**97.73%**	0.00%	0.32%	0.18%	0.03%	9.07%	0.00%
5	0.22%	0.06%	0.02%	3.42%	0.22%	**96.74%**	0.93%	0.08%	8.47%	0.49%	0.06%
6	0.38%	0.25%	0.01%	0.00%	0.10%	0.41%	**98.82%**	0.00%	0.09%	0.00%	0.00%
7	0.03%	1.03%	0.33%	0.01%	0.48%	0.01%	0.00%	**97.33%**	0.00%	1.25%	0.04%
8	0.26%	1.14%	0.19%	1.51%	0.81%	2.34%	0.42%	0.37%	**94.18%**	0.88%	0.09%
9	0.27%	0.20%	0.04%	1.39%	22.65%	0.19%	0.06%	1.98%	0.80%	**95.69%**	0.03%

Each digit was identified in the first batch they appeared in, and every digit was classified with at least 94% accuracy.

It is worth noting that there is a hard ceiling to the possible accuracy the system could achieve. As can be seen in Figure 7, the embedding is not perfect. Some data points are embedded into regions that do not match the digit that they represent; the system is classifying the embedding and thus can only ever be as good as the UMAP results are. Furthermore, note that the summation of each row does not equal 100%. This is because the cluster shells are created independently and can potentially overlap. Nine and four have significant overlap due to their clusters containing a shared edge; five, eight, and three share a similar overlap due to their clusters being closely entwined. This spatial proximity can be seen in Figure 7. This overlap is a potential area to target with the second pass, in which an additional feature like TOA could be leveraged to split the overlapping clusters.

CONCLUSIONS & FUTURE WORK

A flexible system that allows for classification of a data stream that contains an unknown number and type of classes was presented. The algorithm was originally conceived to aid in the discovery of never-before-seen signals, such as reserved modes (aka war modes), which may be encountered in a real conflict. While that was the original use case for the system, it has many other applications. It can be applied to any streaming data that contains some number of classes and whose data points consist of some number of real-valued features. According to Google Scholar, the original paper for DBSCAN has over 28,000 citations, showing a definite interest and need for an unsupervised classification method.

System improvement is ongoing, and refinements and enhancements to the algorithm are constantly being developed. The current implementation utilizes an external library for the calculation of the Delaunay triangulation, and this library is severely limited at higher dimensions. The complexity worst-case boundary of a n-dimensional Delaunay triangulation is $O(n^{d/2})$, which has obvious limitations for high-dimensional datasets. A different approach to forming the Delaunay triangulation is needed to perform in high dimensions. There are existing algorithms that purport to improve upon this worst-case boundary that need to be explored, and developing a purpose-built algorithm specifically for this system is another option. If an approach cannot be found to improve the complexity of the triangulation in higher dimensions (i.e., greater than 50), then a high-dimensional alternative will need to be developed for the portions of the system that rely on the triangulation.

Within that same vein, the triangulation tends to fail when a dimension contains few unique values. For example, an integer feature with a small range will cause problems in the triangulation of the dataset. A method of handling this type of data needs to be added to the system as well. This could take the form of an "OR" gate at the head of the system in the case of a binary feature that feeds into separate pipelines. Alternatively, a method of adding a small amount of jitter could be implemented to create separation between data points sharing the same value in that feature dimension while also maintaining their spatial proximity.

Result visualization and cluster reporting is another area that needs to be further refined. An algorithm that automatically provides a user with real-time cluster statistics and relationships between clusters is a necessary addition for a fully realized and deployed version. Visualization of clusters in space is a simple problem for two- and three-

dimensional datasets, but for higher-dimensional datasets, a sophisticated algorithm will be needed to succinctly display cluster parameters.

REFERENCES

Campello, R. J. G. B., Moulavi, D., Sander, J., (2013). Density-based clustering based on hierarchical density estimates. In Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (Eds.), *Advances in knowledge discovery and data mining. PAKDD 2013. Lecture notes in computer science*, (Vol. 7819). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37456-2_14

Choi, C. Q., (2021). 7 revealing ways AIs fail: Neural networks can be disastrously brittle, forgetful, and surprisingly bad at math. *IEEE Spectrum*, 58(10), 42-47. https://doi.org/10.1109/MSPEC.2021.9563958

Duda, R. O., Hart, P. E., Stork, D. G., (2001). Parzen Windows. In *Pattern classification* (2nd ed.), (pp. 164-174). Wiley.

LeCun, Y., Cortes, C., Burges, C. (n.d.). *The MNIST database of handwritten digits.* http://yann.lecun.com/exdb/mnist/

Li, Z., Zhu, Q., Gold, C., (2004). Triangular irregular network formation from irregularly distributed data. In *Digital terrain modeling: Principles and methodology*, (pp. 79-80). CRC Press.

McInnes, L., Healy, J., Melville, J., (2018). *UMAP: Uniform manifold approximation and projection for dimension reduction*. https://doi.org/10.48550/arXiv.1802.03426