Continuous Monitoring in Wireless Sensor Networks

Leon Willems (1288563)
l.l.m.willems@student.tue.nl

July 4th, 2023

Contents

1 Introduction

Every master's student is faced with doing a thesis at the end of their studies. Saying there's a thousand different possible topics to choose from is to make an understatement. There's already a dozen research clusters, and within each cluster all the researchers have their own expertise and interests. Looking more closely at a single topic, a lot of open work is to be done. But what is wise to do? What could a student do in half a year? It's a fine line between not choosing a too simple question, yet not making things too difficult.

Here, we'll be diving deeper in the research cluster **Algorithms, Geometry and Applications**, the specialization **Massive Data**, topic **Clustering in Big Data Models**. Clustering is to divide a dataset into k groups where the members of a group show some kind of similarity. Often data is too big to fit on one machine, or arrives as a data stream, so either multiple machines are necessary, or one machine keeps a summary of the dataset.

We will be combining two algorithms: two-round communication for the MPC model, and one-pass for the insertion-only streaming model, to tackle the continuous monitoring problem. On the one hand, we will prove theoretically that we can maintain a $(1+\epsilon)$-coreset. On the other hand, the algorithms will be implemented and tested against other state-of-the-art algorithms on

their performance, by running various experiments on synthetic and real data. The applications are within Wireless Sensor Networks. If the time permits, research will be done in getting rid of one parameter: can the number of outliers z be determined by an algorithm? Normally it's a user-defined parameter, that can influence the final clustering a lot, and is not even always known.

1.1 Motivating Example

Suppose we are interested in recognizing different schools of fish that reside in some underwater area. We know what fish exist in this area, but not exhaustively. Ergo, there can be others that appear every once in a while. We can install underwater sensors to measure acoustics, water displacement and temperature. Similar measurements can be clustered together, all indicating that a certain school of fish swim by. Suppose we know there are k different schools of fish, we wish to distinguish these k different groups of measurements. All other fish or objects that float by, are considered as outliers.

Underwater sensors are quite recourse constrained, because of physical conditions. They live in harsh environments, and have a harder time communicating with each other. Suppose we have m sensors at different locations, continuously measuring their surroundings. They keep a clustering, or a summary, of their data. If, at any point, the end-user is interested in knowing the current state, a base station should gather all the sensors' information, and perform a final clustering, yielding the schools of fish.

Because of the resource constraints, the sensors themselves should keep a small as possible summary. Next, the communication with the base station should be as little as possible, as energy usage during communication is quite high. Thus, algorithms should be devised that are compact, efficient, accurate, and keep communication to a minimum.

2 Main paper

The paper this thesis will start with is by de Berg, Biabani, and Monemizadeh (2023). It has wonderful theoretical results for constructing mini-balls in nearly all big data settings: MPC, insertion-only streaming, and fully dynamic streaming. The same setting but in the sliding-window model has been researched by de Berg, Monemizadeh, and Zhong (2021).

As research is never complete, some open work after these results is as follows:

- Are the algorithms implementable?
- If so: how do they compare against other state of the art algorithms on real and synthetic datasets?
- Can we derive an algorithm for the fully dynamic streaming setting but in doubling metric space?

2.1 Continuous monitoring

What if we have multiple machines, and the data come in as streams? We consider m machines, where each machine M_i receives a data stream via an update p_i^t at time t. We have a coordinator machine M_c that can, at any arbitrary point in time $t = T$, ask the machines to yield some statistic, clustering or summary. At that moment, the machines will send their information to the coordinator, which will calculate something of interest.

This problem considers a combination of the two-round communication MPC algorithm from the main paper, and the one-pass insertion-only streaming algorithm. The challenge lies in ensuring a $(1 + \epsilon)$-coreset is maintained, with given storage bounds. The coreset should at all times be maintained, as the coordinator might ask for it at any point in time.

An extra implication is introduced when, in a data stream, an update can include a deletion as well. In our motivating example, it would be that a particular school of fish is not in the sensor's vicinity anymore.

3 Wireless Sensor Networks

Wireless Sensor Networks are considered to be among the most rapidly evolving technological domains thanks to the numerous benefits that their usage provides. As a result, from their first appearance until the present day, Wireless Sensor Networks have had a continuously growing range of applications.

A Wireless Sensor Network (WSN) is a group of spatially dispersed sensor nodes, which are interconnected by using wireless communication. A sensor node is an electronic device which consists of a processor along with a storage unit, a transceiver module, one or more sensors, along with an analog-to-digital (ADC) converter, and a power source, which normally is a battery.

A sensor node uses its sensor(s) in order to measure the fluctuation of current conditions in its adjacent environment. These measurements are converted, via the ADC unit, into relative electric signals which are processed via the node's processor. Via its transceiver, the node can wirelessly transmit the data produced by its processor to other nodes and/or to a selected sink point, reffered to as the Base Station. The Base Station, by using the data transmitted to itself, is able to both perform supervisory control over the WSN it belongs to and transmit the related information to human users or/and other networks.

The collaborative use of a sufficient quantity of such sensor nodes, enables a WSN to perform simultaneous data acquirement of ambient information at several points of interest positioned over wide areas. The inexpensive production of sensor nodes of this kind, which despite their relatively small size, have exceptionally advanced sensing, processing, and communication abilities has become feasible due to continuous technological advances.

3.1 Applications

Such WSNs serve many applications. Kandris, Nakas, Vomvas, and Koulouras (2020) describe six main categories: military, health, environmental, flora and fauna, industrial, and urban. Due to the scope of our research, we'll consider some examples where the data come in with high velocity and with high volume.

The first example is Machinery Health Monitoring within the industrial category. Its objective is to examine the performance of various types of technical equipment and to either detect or predict the occurrence of faults or defects. A situation is described where we monitor oil and gas pipelines. To detect the existence of leakages, sensor nodes are used that monitor the pressure and temperature of the pipeline fluid at many points on the pipelines. The geographical area can be arbitrarily large, where we have many sensors that each collect data many times per second.

Secondly, one important application is Emergency Alerting in the environmental category. Proactive monitoring of the causes of natural disasters can help to avoid them and/or lower their cost. WSNs can, in real time, monitor common disastrous causes to provide alerts in order to lower damage or even prevent disaster. One such instance is a forest fire. They can destroy both animals and vegetation, resulting in huge ecological disasters. Due to the rapid climate change taking place, necessity for proper prevention is extremely pressing. The WSN monitors many parameters such as temperature and humidity. Due to it working in real time, the sensors generate many data points per second, and can be placed in arbitrarily large forests.

3.2 Challenges in WSNs

With rapid growth comes an increase in challenges to overcome. As WSNs are multidisciplinary inventions, we will not consider all challenges and limit our scope to the data analysis side, and others that either influence or are influenced by data analysis methods. Majid et al. (2022) list both challenges and future directions in their survey, of which we will mention a few.

Big data: Devices might produce huge amounts of data, for example, when sensors measure each millisecond.

Limited resources: A sensor node might be severely limited in its capabilities to compute and to store. They are often limited by the total energy available. In that sense, computation are not the bottleneck, but communication is, which costs a lot of power and should thus be reduced as much as possible.

Combining the above, we conclude that we need to build algorithms that do not let the nodes communicate often in the WSN. Thus, we maximize the calculations on the nodes themselves, while still maintaining the (approximate) right information that the sensor saves. Then, when appropriate, the nodes communicate with the Base Station.

Now, we are in the realm of Big Data Models. We maintain a summary of the data on the nodes, where the data exceed the storage space of the nodes. Here we apply insertion-only streaming algorithms, as the measurements come in one by one, and will not be deleted. Then, for communication between all the nodes and the Base Station, we need Massively Parallel Computing (MPC) algorithms.

3.3 Clustering in WSNs

The current research touches on three variants of clustering in WSNs, on different levels of computation. For all situations, we consider an example described by Hassani, Müller, and Seidl (2009). Let m distributed sensor nodes be spread over some area in a jungle, whose target it is to measure the existence of k kinds of wild animals. Any other animal should be considered as an outlier.

Node-level clustering: Regular clustering, per node. Only the data on nodes are clustered, that yield some information to be processed by the WSN. However, this processing does not include clustering anymore. In the mentioned example, we would cluster the animals at node level. Then, we might be interested in how many of these animals pass by every node. After clustering, we only need the sizes of the clusters to answer our question. Thus, communicating once with Base Station is sufficient.

Two-level clustering: Where we are interested in a final clustering over all incoming data points. In line with the example, we have m distributed nodes. Let's say that we are interested in the top k most popular wild animals over the whole area. Each node computes a k-center clustering, send their findings to the Base Station, who calculates a final clustering. Cormode, Muthukrishnan, and Zhuang (2007) mention such a setting, where they are interested in maintaining such a clustering at all times.

Topology management or in-network clustering: As described, energy consumption is a big problem and reducing communication between nodes is vital. Per group of nodes, we assign one node to be the communicator with the Base Station, to reduce total communication. How do we determine groups and their communicator? This is where clustering finds its application: based on location and similarity of measurements, we can choose a group and its centerpoint, ergo the communicator. Shahraki, Taherkordi, Haugen, and Eliassen (2020) describe and review many such algorithms. In the running example, we still have to cluster at node-level. However,

now we'd like to find similar nodes as well.

4 Experiments

For the experimental setup, we take inspiration from other papers with similar research, preferably those with a decent amount of citations. By looking at other papers, we'll be answering the following questions:

- What metrics are used to assess the quality of an algorithm?
- How do other papers compare results with each other?
- How are the number of outliers determined?
- What is the experimental setup?
- On what kinds of datasets do others run their experiments?

For this section, we will go over all the questions and show our findings for three different paper.

4.1 Datasets

Chen, Azer, and Zhang (2018) have a mix of synthetic and real datasets. The synthetic datasets gauss-σ and susy-Σ are constructed respectively by drawing from Gaussian distributions, and by running a Monte Carlo simulation to simulate signal data (Whiteson, 2014).

Real datasets are kddFull[1] and kddSp, where the latter contains 10% of the data of the former. These consider digital connections of a LAN, where the goal is to distinguish between many kinds of attacks or normal conections, thus serving itself perfectly for clustering in a streaming fashion.

The last dataset, 3DSpatial-Δ[2], contains coordinate points with altitudes in North Jutland, Denmark. The exact applications using clustering are unknown.

Just like in the previous paper, Ding, Yu, and Wang (2019) present a mixture of synthetic and real datasets. Their synthetic datasets are all drawn in a hypercube of side lengths 200 from a Gaussian distribution with high varying dimensionality.

The three real datasets are widely used: MNIST (LeCun, Bottou, Bengio, & Haffner, 1998), Caltech-256 (Fei-Fei, Fergus, & Perona, 2004) and CIFAR-10 (McCrary, 1992).

MNIST contains a set of 60,000 training images of handwritten digits, and 10,000 testing images. As there are 10 classes, the problems is perfectly suitable for a clustering problem. Caltech-256 is similar in that it has many images, but now of 256 different objects, serving as a clustering task for bigger k. The third in the same category, CIFAR-10, contains images of 10 classes as well.

Ding, Huang, Liu, Yu, and Wang (2023) consider four real datasets from the UCI KDD archive: Shuttle, Covertype, KDD Cup 1999 and Poker Hand[3]. The number of instances per dataset varies between 43,500 and 4,898,431, and the number of classes between 7 and 23. The dimensionality is not extremely high: at most 54. However, the paper only consider at most 100,000 (random) instances for the bigger datasets.

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[2] https://networkrepository.com/3D-spatial-network.php
[3] http://kdd.ics.uci.edu/

4.2 In the literature

What metrics are used to assess the quality of an algorithm? Chen et al. (2018) test the quality in multiple ways. One is to consider the inherent loss function of either k-median or k-means loss: the cumulative distances to the centers. This is applicable to k-center clustering but using a different loss criterium: the radius of the balls.

The second performance measure is that of outlier detection. Naturally recall, precision and proportion come to mind.

Finally, we are interested in the communication cost and running time, as consequences of the algorithm. Interest is in the number of points that are exchanged between the coordinator and all sites in a round of communication.

Ding et al. (2019) run their experiments with different quality measures in mind. As they are able to run an algorithm that yields the optimal solution, they consider the approximation ratios of their own algorithms. Next to that, the running times are noted, and both metrics contain standard deviations.

Ding et al. (2023) consider, as usual, the running time of all algorithms. Then, the clustering cost $\Phi_\epsilon(X, E)$ is of importance, which is the maximum radius of any of the clustering balls. Both statistics are shown with standard deviations.

How do other papers compare results with each other? Papers often show improvements over others' results, by comparing different measures. We'd like to know what exactly is being compared.

Chen et al. (2018) mainly battle long running times of other state-of-the-art algorithms. Further, the summary sizes are compared, and they provide approximation bounds.

Ding et al. (2019) compare both their implementations against other algorithms, where the metrics are the approximation ratio, and the running time. For the coreset construction, the alternative is a random selection of points. For the final clustering, the alternatives are two baseline algorithms from early work on k-Center clustering, by Charikar, Khuller, Mount, and Narasimhan (2001) and McCutchen and Khuller (2008).

Ding et al. (2023) show theoretical improvements over other papers. For the experiments, a few algorithms have been compared.

How are the number of outliers determined? It is complicated to objectively count the number of outliers in a dataset. There's many ways to determine what an outlier might be. For synthetic datasets, this might even be harder. We take a look at how other papers tackle this problem.

For their synthetic datasets, and for susy-Δ, Chen et al. (2018) sample points from the original distributions and add shift to make outliers. Thus, the number of outliers is chosen by design. The kddFull dataset contains 23 classes. 3 of these have been chosen as the main data because they make up 98.3% of the points, and the others are outliers.

Ding et al. (2019) have the same strategy for all their datasets: take the original dataset, computing the minimum enclosing ball per cluster, and randomly generate outliers outside the balls. For the synthetic datasets, the number of outliers varies between 2, 4, 6, 8 and 10% of n. To the real dataset, 5% of n are added.

Ding et al. (2023) arbitrarily choose to include 1% on top of the original dataset as number of outliers. They add them by first calculating the minimum enclosing ball of the whole dataset, and then randomly generate point outside of that ball, but within 1.1 times the radius of that ball.

5 Open problems

Many advances have been made, and even more research is still to be done. For this section, we mainly went over reviews and surveys in the two main topics: streaming algorithms and Wireless Sensor Networks. Our focus will be on the intersection of these two ideas. Note that the scope will be too big if we list all challenges, so we will explain the ones most interesting to us more broadly.

5.1 Streaming algorithms

Parameter estimation: As described by Silva et al. (2013), a challenge is to automatically estimate parameters, so to avoid having to choose them beforehand. In our case, two main examples are the number of clusters k, and the number of outliers z.

Benchmarking: There is no clear-cut way yet to compare between streaming algorithms. Silva et al. (2013) mention that effort should be addressed to develop better evaluation criteria, high-quality benchmark data, and a sound methodology for reliable experimental comparison. According to Zubaroğlu and Atalay (2021), there is a lack of high quality benchmark data, and that generating and collecting artificial and real stream datasets and popularizing them is a challenge.

Performance: As the WSNs tend to get bigger, and the data generated by them more voluminous, there is need for scaling up algorithms. Zubaroğlu and Atalay (2021) observe that we should aim to improve on speed performance, to enable handling more data per time unit.

5.2 WSNs

Clustering quality: The importance of quality only becomes apparent when in action. Amutha, Sharma, and Sharma (2021) mention that error rate is still a prevalent issue in WSN clustering, and that quality of clustering will become more critical in evaluating the performance.

Topology: As discussed earlier, clustering can be used for determining the topology of a WSN as well. Recall that a topology is the configuration of nodes and their communicators with the Base Station. This configuration can have significant impact on the energy usage, and Amutha et al. (2021) indicate that this remains a major challenge.

Scalability: As a more overarching challenge, El Khediri (2022) state that the vast majority of clustering protocols are designed to operate only for smaller networks. They will not work on bigger networks, for which more scalable techniques of clustering need to be implemented. Ahmad, Wazirali, and Abu-Ain (2022) observe that often complex ML algorithms are implemented to improve accuracy, without regards for the hardware limitations. We therefore need algorithms that take up little space, and cost less energy.

6 Research proposal

Given the open problem in both streaming algorithms and WSNs, we deem it a good idea to implement the MPC and insertion-only streaming algorithms by de Berg et al. (2023), combine them into a continuous monitoring algorithm, and test it under various circumstances. Given that it's usually unknown beforehand what the parameters k and z are, we're interested in finding a way to automatically predict these in a given situation.

6.1 Goals

We will list the research (sub)questions to be investigated:

- How do the quality and performance of the continuous monitoring algorithm behave against other state-of-the-art algorithms in Wireless Sensor Networks?
  - Subgoal: to build a framework for testing the given algorithm against other algorithms
  - Subgoal: to devise a benchmark by considering multiple metrics, and collecting real and synthetic datasets that work well for data streams
  - Subgoal: to scale the algorithm to Wireless Sensor Networks of arbitrary size in terms of nodes and data points
  - (if the time permits) Subgoal: can we handle deletions in our stream as well? Making it a fully dynamic streaming part of the algorithm
- (if the time permits) Can we implement a model such that it automatically predicts parameters k and z?
  - Subgoal: to devise a model that is trained on predicting what the outliers should be based on an oracle, and using it for predicting the number of outliers for a new data stream
  - Subgoal: to gather knowledge on current k-estimating methods, and implement them

6.2 Research approach

The goals are not to be done in a linear fashion, but rather in some overlapping timelines, and with iterations. Some objectives can only be reached when others have been completed, or at least have been started. A short description of the work that will be done per (sub)goal will follow. The starting times are roughly in chronological order.

Combine the MPC and streaming algorithms into a continuous monitoring algorithm. The goal is to write pseudocode that includes both algorithms, and to bridge the gap between the two. A second challenge is to enable a user to demand answers at any points in time. As the original algorithms are already quite flexible and generalizable, this should not pose a problem.

Derive theoretical results from the continuous monitoring algorithm. One of the challenging parts of the thesis. The algorithm should yield a $(1+\epsilon)$-coreset, with clear bounds on the storage, running times and communication costs. This objective should be started early on in the timeline, as it might pose problems and delays if started late.

Implement the continuous monitoring algorithm. Both elements of the algorithm have already been implemented as a toy example. The main challenge is to develop it in a robust, scalable way on a platform that provides streaming and parallel computing possibilities. The code should be neat and interpretable by other programmers.

Build a framework in which to test the continuous algorithm against other state-of-the-art algorithms. For this objective, there needs to be a way to incorporate other algorithms as well. As they are implemented across platforms and coding languages, this will not always be possible. That could mean it's only possible to compare the results against the results mentioned in other papers.

Devise a benchmark and run the experiments. The benchmark will be based on a lot of other literature, combining their approaches and observing what works well and what is popular. Datasets need to be carefully chosen to accurately represent a continuous monitoring situation. The goal is to devise a benchmark that can be re-used by other researchers, to easily compare such algorithms.

Research and implement predicting z. As one of the open problems in research, this is definitely not the easiest of objectives. In the running example, it is unknown what the number of outliers will be. Therefore, if a slightly accurate way can be found to predict the number (continuously), a lot will be gained. A method needs to be devised that can predict z, which is likely a machine learning algorithm, trained using an oracle for different datasets. Ergo, datasets are needed where z is known at any point in time.

Extend to the fully dynamic streaming model. The main paper includes a fully dynamic streaming algorithm that works well in Euclidean space. As most datasets contain d-dimensional samples, this is applicable, and a choice that is grounded. The tricky parts are the sketches used.

Research and implement predicting k. A last goal would be to also predict the number of clusters k. What if there's a school of fish we haven't considered yet? There is a trade-off between considering them as outliers, or as a separate cluster.

6.3 Timeline

The actual thesis will start on September 7th, and ends six months later, on March 7th. Please find a first draft of the timeline in table 1.

References

Ahmad, R., Wazirali, R., & Abu-Ain, T. (2022). Machine Learning for Wireless Sensor Networks Security: An Overview of Challenges and Issues. *Sensors*, *22*(13). doi: 10.3390/s22134730

Amutha, J., Sharma, S., & Sharma, S. K. (2021). Strategies based on various aspects of clustering in wireless sensor networks using classical, optimization and machine learning techniques: Review, taxonomy, research findings, challenges and future directions. *Computer Science Review*, *40*, 100376. Retrieved from `https://doi.org/10.1016/j.cosrev.2021.100376` doi: 10.1016/j.cosrev.2021.100376

Charikar, M., Khuller, S., Mount, D. M., & Narasimhan, G. (2001). Algorithms for facility location problems with outliers. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 642–651. doi: 10.5555/365411.365555

Chen, J., Azer, E. S., & Zhang, Q. (2018). A practical algorithm for distributed clustering and outlier detection. *Advances in Neural Information Processing Systems*, *2018-Decem*(Nips), 2248–2256.

Cormode, G., Muthukrishnan, S., & Zhuang, W. (2007). Conquering the divide: Continuous clustering of distributed data streams. *Proceedings - International Conference on Data Engineering*, 1036–1045. doi: 10.1109/ICDE.2007.368962

de Berg, M., Biabani, L., & Monemizadeh, M. (2023). k-Center Clustering with Outliers in the MPC and Streaming Model. Retrieved from `http://arxiv.org/abs/2302.12811`

de Berg, M., Monemizadeh, M., & Zhong, Y. (2021). *K-Center Clustering With Outliers in the Sliding-Window Model* (Vol. 204) (No. 13). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany. doi: 10.4230/LIPIcs.ESA.2021.13

	Description	Expected Result	Deliverable
1	Algorithm selection	to have a set of algorithms we will compare against ours	collection of state-of-the-art algorithms
2-4	Coding	write pseudocode for and implement the continuous monitoring algorithm	codebase for continuous monitoring algorithm, load in other codebases
3-6	Theoretical	derive theoretical results for continuous monitoring combined algorithm	a sound, proven algorithm with error guarantees and bounds
5-6	Research	to be ready for testing	have the right metrics and datasets, streaming environment to work in
7-8	Framework	a testing environment	first iteration of the benchmark framework
9	First tests	test current setting, without generalization of k and z	results of the first test setting
10-11	Research	to have read literature on k and z estimation methods	writings on current k and z estimation methods
12-13	Coding	have the algorithms ready	codebase including automatic k and z prediction
14	Gather and incorporate feedback	collect feedback from multiple sources, do some more research, incorporate	improvements on all topics
15	Fully dynamic	think of extension to fully dynamic streaming	feasibility conclusion of the extension to fully dynamic streaming
16	Christmas break		
17	Research	investigate and think about the best way to run experiments	setup for the experiments
18	Final experiments	run all the code	results and tables
19-22	Writing	finalizing draft	thesis draft
23	Carnival break		
24-25	Incorporate feedback	to be done with the written part	thesis final version
25-26	Work on presentation	to be ready for the defence	presentation
26	Defence	graduation	thesis

Table 1: Timeline of my thesis.

Ding, H., Huang, R., Liu, K., Yu, H., & Wang, Z. (2023). Randomized Greedy Algorithms and Composable Coreset for k-Center Clustering with Outliers. Retrieved from http://arxiv.org/abs/2301.02814

Ding, H., Yu, H., & Wang, Z. (2019). Greedy strategy works for k-center clustering with outliers and coreset construction. *Leibniz International Proceedings in Informatics, LIPIcs*, *144*. doi: 10.4230/LIPIcs.ESA.2019.40

El Khediri, S. (2022). Wireless sensor networks: a survey, categorization, main issues, and future orientations for clustering protocols. *Computing*, *104*(8), 1775–1837. Retrieved from https://doi.org/10.1007/s00607-022-01071-8 doi: 10.1007/s00607-022-01071-8

Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2004-Janua*(January). doi: 10.1109/CVPR.2004.383

Hassani, M., Müller, E., & Seidl, T. (2009). EDISKCO: Energy efficient distributed in-sensor-network K-center clustering with outliers. *Proceedings of the 3rd International Workshop on Knowledge Discovery from Sensor Data, SensorKDD'09 in Conjunction with the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD-09*, 39–48. doi: 10.1145/1601966.1601975

Kandris, D., Nakas, C., Vomvas, D., & Koulouras, G. (2020). Applications of wireless sensor networks: An up-to-date survey. *Applied System Innovation*, *3*(1), 1–24. doi: 10.3390/asi3010014

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2323. doi: 10.1109/5.726791

Majid, M., Habib, S., Javed, A. R., Rizwan, M., Srivastava, G., Gadekallu, T. R., & Lin, J. C. W. (2022). Applications of Wireless Sensor Networks and Internet of Things Frameworks in the Industry Revolution 4.0: A Systematic Literature Review. *Sensors*, *22*(6), 1–36. doi: 10.3390/s22062087

McCrary, M. B. (1992). Urban multicultural trauma patients. *Asha*, *34*(4).

McCutchen, R. M., & Khuller, S. (2008). Streaming algorithms for k-center clustering with outliers and with anonymity. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5171 LNCS*, 165–178. doi: 10.1007/978-3-540-85363-3_14

Shahraki, A., Taherkordi, A., Haugen, Ø., & Eliassen, F. (2020). Clustering objectives in wireless sensor networks: A survey and research direction analysis. *Computer Networks*, *180*(March). doi: 10.1016/j.comnet.2020.107376

Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., De Carvalho, A. C., & Gama, J. (2013). Data stream clustering: A survey. *ACM Computing Surveys*, *46*(1). doi: 10.1145/2522968.2522981

Whiteson, D. (2014). *SUSY*. UCI Machine Learning Repository. (DOI: https://doi.org/10.24432/C54606)

Zubaroğlu, A., & Atalay, V. (2021). *Data stream clustering: a review* (Vol. 54) (No. 2). Springer Netherlands. Retrieved from https://doi.org/10.1007/s10462-020-09874-x doi: 10.1007/s10462-020-09874-x