Data Stream Clustering Applied to Preprocessed Yugioh Data

Vincent Allen

5/8/2020

cs6405: Clustering Algorithms

About the Author:

After graduating from the Missouri University of Science and Technology with an Electrical Engineering B.S degree in Dec 2014, Vincent joined Boeing as a Software Engineer II. He has since started a Computer Science M.S. degree and plans to graduate in Dec 2020. During his into to AI course, Vince’s [AI](https://github.com/vtad4f/chess-ai) (Volcanic Counter) placed 3rd in the chess tournament that semester. Later on, in his free time he created a [UI](https://github.com/vtad4f/chess-ui) in python to go with it. Growing up Vince enjoyed playing the trading card game Yugioh.

Table of Contents

[Executive Summary 3](#_Toc39780887)

[Introduction 4](#_Toc39780888)

[Project Specifications 7](#_Toc39780889)

[Detailed Design 8](#_Toc39780890)

[Experimental Results 10](#_Toc39780891)

[References 11](#_Toc39780892)

[Works Cited 11](#_Toc39780893)

[Works Consulted 11](#_Toc39780894)

[Appendix A 12](#_Toc39780895)

[Appendix B 13](#_Toc39780896)

# Executive Summary

Before proceeding, we’ll give a brief but relevant introduction to Yugioh. It is a trading card game with thousands of cards and variety of card types. Imagine two children saying back and forth “My alien shoots your dinosaur with a lazer gun”, then “Well my dinosaur uses an anti lazer shield”, then “My alien eats shields” and you have a good idea of what a game of Yugioh is like. There is enough variety in the card effects to make analysis of them difficult. There will always be outliers.

The goal of this project is to use stream clustering to predict the level of a Yugioh monster card based on its attack, defense, and card text. Most monster types have a level/rank between 0 and 12. There isn’t a rule that says higher level monsters should be stronger than lower level monsters (i.e. have higher attack and defense or better effects), but that’s the general trend. In most cases low level monsters are sacrificed to summon high level monsters, so this makes sense. In any case, the interesting aspect of the clustering should be translating the card text into numerical attributes.

With stream clustering we are able to make a prediction as we process every single point of data, instead of having to wait for a training phase to finish. Thus, we are able to watching metrics update dynamically as the algorithm walks the data.

# Introduction

### Preprocessing

Before discussing any data mining algorithms, we should focus briefly on preprocessing data. In real life there will be missing data and decisions will have to be made about how to deal with it. One common approach, called listwise deletion, just means deleting any datapoints with missing attribute data. This is only acceptable if the population is sufficiently large and the data is MCAR. Substitution just means replacing the sample with another sample from the population. Imputation means inserting a new value.

MCAR, or Missing Completely at Random, means that the probability of missing data is equal in all cases.

MAR, or Missing at Random, means that the probability of missing data is equal in certain partitions of the data. I.e. we know why it’s missing, and the reason isn’t one of the attributes we’re recording.

MNAR, or Missing Not at Random, means that the probability of missing data varies for reasons we don’t know.

### Clustering

Clustering is one of many approaches to machine learning. More specifically, it is a method of unsupervised learning – meaning it can be used to analyze data that hasn’t already been labeled/ classified. Clustering algorithms can be applied to a dataset to just ‘see what happens’. Perhaps it will label the data the way we expect it to, or perhaps it will find a new pattern we humans didn’t notice.

As for the approach, similar data points are grouped together in groups called clusters. This similarity is measured by calculating the distance between the attributes of each data point. A good cluster will have less variance. Ideally, a clustering algorithm would minimize the number of standard deviations a point is from the cluster centroid. Note that the centroid is the mean of all the points in the cluster. Consider Figure 1. The data points are colored according to the labels assigned to them when generated.

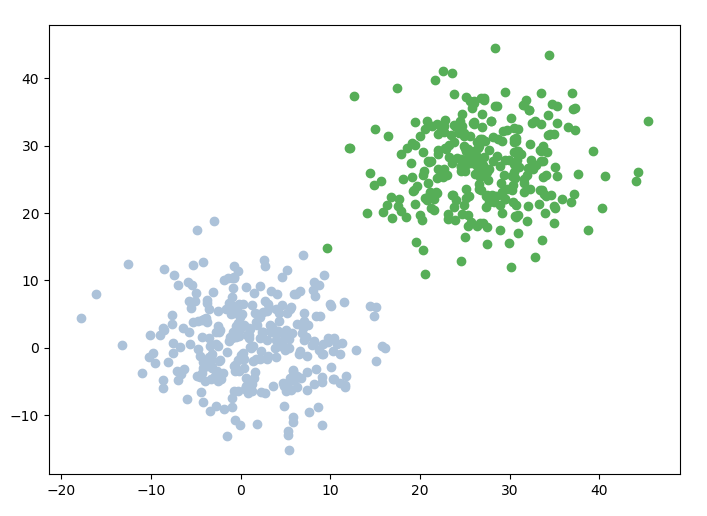


Figure 1

The points are grouped together essentially how we as humans would group them. There are two ‘blobs’ of points, so we would think there should be two clusters. We don’t have to know how many clusters there are, because our minds come up with ‘two’ at a glance. Clustering algorithms can also figure it out without being told by setting a certain threshold for variance. If the points are ‘x’ distance from the centroid, they must not be part of this cluster. See Figure 2 for an example of that distance threshold being set too low. It was too restrictive, so what should have been two clusters turned into several. These figures were generated using a demo file in our repository (see link in Appendix A).

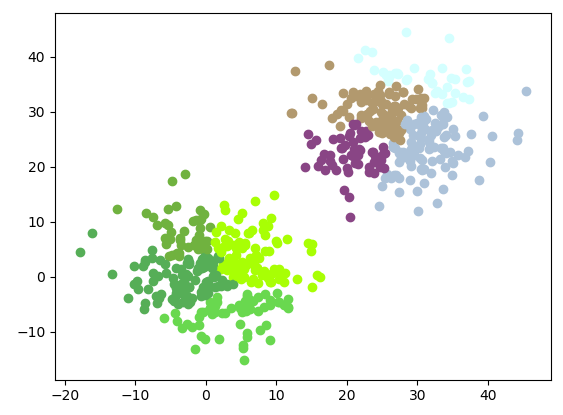


Figure 2

### Hierarchical Clustering

Hierarchical clustering is one of many approaches to clustering. In particular, the relationship between the clusters forms a tree. The leaf clusters merge together to form their parent cluster, and so on. See the dendrogram in Figure 3 (also generated using the demo file in our repo (see link in Appendix A).

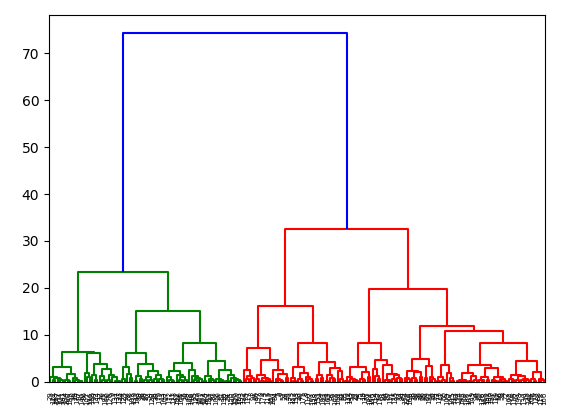


Figure 3

### Data Stream Algorithms

Data stream algorithms in general are useful for analyzing either big data or data that changes or arrives frequently. The concept is that data is received one entry at a time instead of having access to the entire data set from the beginning. This gives us the huge advantage of being able to pause at any moment and have a viable model. It also means that data stream algorithms can attempt to analyze an enormous data set without having to hold the entire thing in memory at once to succeed.

See Figure 4 for a basic Data Stream Clustering algorithm schematic, with example inputs. The training/test data blocks are a reminder that even though clustering is unsupervised it can still be used with labeled data. In fact, the labels it generates could be correlated with the already existing labels and used to measure accuracy (assuming the clustering results aren’t more accurate than the original labels). The serializer would just read one element at a time from the training data set, since the whole set wouldn’t usually be passed to a Data Stream algorithm. The web crawler input is an example of a data stream type that would be especially well suited for input to one of these algorithms.

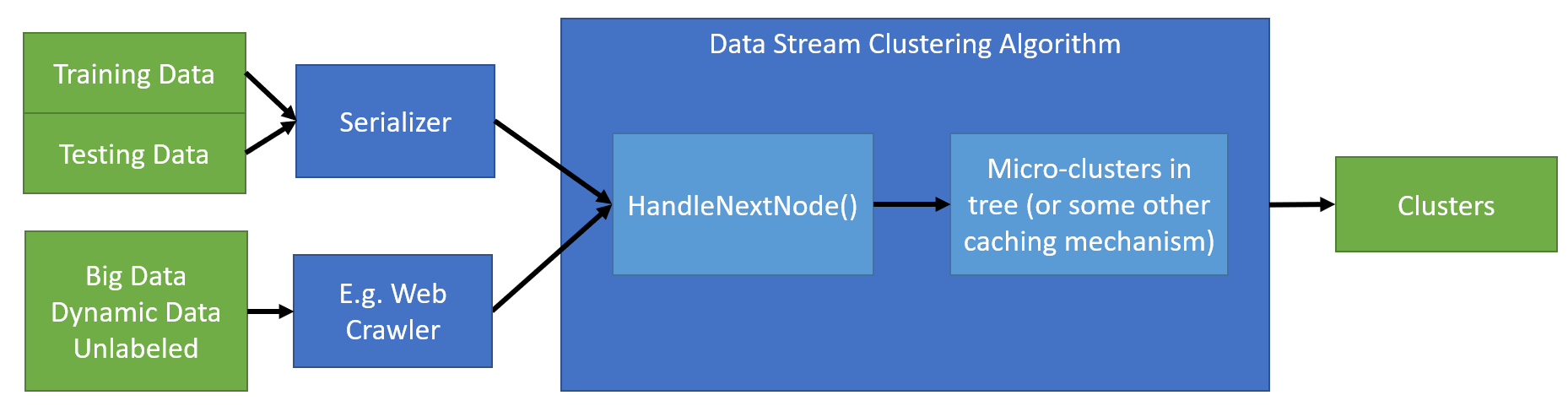


Figure 4

# 

# Project Specifications

Why Yugioh, and why data stream clustering? This particular data set offers a chance to experiment with textual data preprocessing. If we use only the inherently-numeric attributes there is an accuracy of roughly 50%, so it is obvious whether any additional attributes derived from card text have negative, positive, or zero impact. The goal is to improve the accuracy above 50%, and the tool to be used for evaluation will be a data stream clustering algorithm. This type of algorithm allows us to see the accuracy fluctuate as the algorithm parses the data set. It gives us a way to granularly observe the impact of re-ordering our data or including/omitting attributes.

# Detailed Design

In our case, a few data points were (1) labeled incorrectly and (2) missing data. Both issues were fixed manually by looking up the correct information.

Next, we had to deal with numerical data that wasn’t missing but couldn’t be assigned a ‘correct’ value either. Several monsters in the game have placeholder values (e.g. ‘?’,’???’, or ‘X000’) for attack and defense values, and their effects specify the value. The problem is that they are all situationally dependent. E.g. monster X has 600 attack per card in the opponent’s hand. Given that they are all special cases, manual imputation could be an option for a person who knows the game. Though it would be time consuming and extremely biased. The simplest solution is probably to omit them entirely.

As mentioned in the introduction, this approach (listwise deletion) is only acceptable if the dataset is sufficiently large (over 6000 in our case) and MCAR. This ‘missing’ data is definitely MCAR, as there isn’t a specific subset of monsters with variable attack and defense values. Any type of monster with any level could have such an effect.

The original approach was to simply set the attack and defense to 0. This is because several cards in the game actually have a value of 0 instead of something like ‘?’, yet they still have an effect that replaces the 0 value. An example is *Relinquished*, which captures an opponent’s monster and uses its attack and defense values as its own. The question is, how do we tell those cards apart from the ones which actually have a value of 0? Is it worth the time to try and detect these cards by parsing the card text? Not to mention that there are other cards with typical attack and defense values which can be increased or decreased via their effect. If two different people started down that same rabbit hole, they would emerge with two completely different data sets. Thus, we believe the best/least biased action is to not alter the attack and defense data based on card text.

Speaking of card text, it will actually contribute the bulk of our preprocessing logic. Since it isn’t inherently numeric, we’ll need to translate the text into one or more numeric attributes. The goal is to group monsters based on level, and monsters with good effects are comparable with ones that have high attack or defense stats. I.e. we need find a way to detect and measure ‘good’ effects.

There is one thing to note before we dive in. One type of monster called ‘Normal’ has text but it serves no functional purpose. It’s just lore. Thus, Normal monster text will be omitted during the preprocessing. Now, on to translating card text to numeric values.

The first attempt is a simple word count. Let’s assume that the effects with the most words are strongest. This is definitely incorrect. Consider *Exodia the Forbidden One* which has the 40-word effect “If you have [...] in addition to this card in your hand, you win the Duel.” This is a very good effect for being so short, as there are just a handful of cards in the game that allow a player to win outright without using monsters to attack their opponent.

*This section will form the bulk of the report. Here you should include design decisions and tradeoffs as well as any detailed technical drawings such as circuit diagrams, flow charts, etc. If these are particularly large, they may be placed in an appendix, but should be referenced in this section.*

*It is here that you really get into the details of why your project is designed the way it is. Tradeoffs are made in a number of areas and a good way to organize this section is to figure out what the most important tradeoffs are and explain each of them with a few paragraphs.*

*This is a section that will evolve as the project nears completion, but its writing could be started at the beginning of the semester. I’ll be happy to give feedback on whatever you are able to produce for this section during the semester, while recognizing that some elements will be subject to change or impossible to write up until the final project is completed.*

# 

# Experimental Results

*At some point you will have to determine whether your project works. This section should detail the design of the testing experiments, the results of the testing, and comments regarding whether the project does what it is supposed to do. Elements that are hard to test and aspects of the project that do not pass the tests should be highlighted. The design of the experiments is worth some space as well, as there are design tradeoffs and decisions to be discussed here as well as for the project itself. Any data should be included in tables or charts in this section or in an extra appendix (in addition to the one described below) if it is particularly cumbersome.*

# References

## Works Cited

**[1]** M. Ghesmoune, M. Lebbah, and H. Azzag, “State-of-the-art on clustering data streams,” Big Data Analytics, vol. 1, no. 1, 2016. ([link](https://bdataanalytics.biomedcentral.com/articles/10.1186/s41044-016-0011-3))

**[2]** S. Marsland, J. Shapiro, and U. Nehmzow, “A self-organising network that grows when required,” Neural Networks, vol. 15, no. 8-9, pp. 1041–1058, 2002. ([link](https://www.sciencedirect.com/science/article/abs/pii/S0893608002000783))

**[3]** P. Kranen, I. Assent, C. Baldauf, and T. Seidl, “The ClusTree: indexing micro-clusters for anytime stream mining,” Knowledge and Information Systems, vol. 29, no. 2, pp. 249–272, 2010. ([link](https://www.researchgate.net/publication/220283749_The_ClusTree_Indexing_micro-clusters_for_anytime_stream_mining))

**[4]** S. Mansalis, E. Ntoutsi, N. Pelekis, and Y. Theodoridis, “An evaluation of data stream clustering algorithms,” Statistical Analysis and Data Mining: The ASA Data Science Journal, vol. 11, no. 4, pp. 167–187, 2018. ([link](https://www.researchgate.net/publication/325977106_An_evaluation_of_data_stream_clustering_algorithms))

**[5]** A. Amini and T. Y. Wah, “Density Micro-Clustering Algorithms on DataStreams: A Review,” Lecture Notes in Engineering and Computer Science, vol. 1, 2011. ([link](https://www.researchgate.net/publication/50864247_Density_Micro-Clustering_Algorithms_on_Data_Streams_A_Review))

**[6]** L. Liu and T. Peng, “Clustering-based topical Web crawling using CFu-tree guided by link-context,” Frontiers of Computer Science, vol. 8, no. 4, pp. 581–595, 2014. ([link](https://link.springer.com/article/10.1007%2Fs11704-014-3050-9))

## Works Consulted

Articles / Websites:

* <https://www.guru99.com/supervised-vs-unsupervised-learning.html>
* <https://www.quora.com/Is-unsupervised-learning-a-better-approach-than-supervised-learning-models>
* <https://www.researchgate.net/post/what_is_micro-cluster_in_data_stream_clustering_algorithm>
* <http://www.personal.psu.edu/pum10/tois-tan.pdf>
* <http://www.rroij.com/open-access/an-approach-to-build-a-web-crawler-using-clustering-based-kmeans-algorithm-14-22.pdf>
* <https://en.wikipedia.org/wiki/R-tree>
* [<https://nces.ed.gov/training/datauser/PSS_02.html?dest=PSS_02_S0170.html>](https://nces.ed.gov/training/datauser/PSS_02.html?dest=PSS_02_S0170.html)
* [<https://www.theanalysisfactor.com/causes-of-missing-data/>](https://www.theanalysisfactor.com/causes-of-missing-data/) *(broken link since original visit)*

Source Code:

* <https://github.com/Spark-clustering-notebook/coliseum/wiki/G-Stream>
* <https://github.com/huawei-noah/streamDM>
* <https://cran.r-project.org/web/packages/clustree/index.html>

Videos:

* <https://www.youtube.com/watch?v=B-NN-weo5e4>

# Appendix A

You can find the source code for this project here:

<https://github.com/vtad4f/stream_clustering/tree/master/src>

Some experimentation was performed with an [R implementation of gStream](https://cran.r-project.org/web/packages/gStream/index.html). However, with just one look at the API and the demo it is obvious that data cannot be fed into the algorithm one node at a time.

In contrast, this [python implementation](https://github.com/scikit-multiflow/scikit-multiflow) of several streaming algorithms explicitly defines data stream classes with next\_sample() functions. See this [demo script](https://github.com/scikit-multiflow/scikit-multiflow/blob/master/src/skmultiflow/demos/_test_knn.py) which uses kNN to predict one element of data at a time. Here are a few lines from the file:

**while** n\_samples **<** max\_samples**:**

X**,** y **=** stream**.**next\_sample**()**

my\_pred **=** knn**.**predict**(**X**)**

**if** y**[**0**]** **==** my\_pred**[**0**]:**

my\_corrects **+=** 1

n\_samples **+=** 1

Now, we are free to take this same granular approach with our implementation, but there is a higher-level import from the same package which displays a graph with the metrics of our choice. The following is an excerpt from our implementation, with all the optional parameters removed for brevity:

**from** skmultiflow**.**data **import** FileStream

**from** skmultiflow**.**lazy**.**knn **import** KNN

**from** skmultiflow**.**evaluation **import** EvaluatePrequential

stream **=** FileStream**(**'data/stream1.csv'**)**

stream**.**prepare\_for\_use**()**

mdl **=** KNN**()**

evaluator **=** EvaluatePrequential**(**metrics**=[**'accuracy'**])**

evaluator**.**evaluate**(**stream**=**stream**,** model**=**mdl**)**

This is straightforward, and we can watch the accuracy update gradually as the algorithm walks the dataset. That being said, we’ve done some preprocessing work to translate the card text into numeric attributes, and to normalize the data. It was all trivial to implement, so we will only include the high level functions here:

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

"""

BRIEF Parse and assemble a single csv with star columns

https://www.yugiohcardguide.com/level/0.html

all the way through .../12.html

Note that the page contents have been copied manually

"""

header\_row**,** rows **=** ReadAll**()**

Write**(**header\_row**,** rows**,** 'stars.csv'**)**

ProcessText**(**rows**)**

FixNumeric**(**rows**)**

random**.**shuffle**(**rows**)**

Write**([**

'attrib1'**,** # atk

'attrib2'**,** # def

'attrib3'**,** # word ct

'class' # level

**],** rows**,** 'stream1.csv'**)**

# Appendix B

See additional file for the annotated bibliography.