

Clustering

John Vorsten

May 2020

1 Introduction

Clustering is breaking sets of objects into groups (or clusters) so that objects within groups are similar to each other. A *clusterer* is a algorithm or function that is responsible for grouping objects. It requires a *distance measurement* between objects and an *aggregation function* to decide which group an object belongs in. The aggregation function requires an assumption to be made about the data it operates on [Charrad et al., 2014a]. Without the assumption it would not be able to divide data into groups. Usually that assumption comes in the form of

- the desired number of clusters
- a density lower limit for objects in a cluster
- a maximum distance between objects to fit in a cluster
- some other form of distance limit or statistical metric

So, partitioning a set of examples into clusters depends on the the assumption used to calculate similarity within a cluster. If that assumption is not good, then the aggregation won't be useful. If the assumption is good, then the aggregation might output useful groups.

In practice, the assumption used to partition data (number of clusters, or density threshold) is subjective. In a dataset,

- a Variance and density are affected by the size of the feature space
- b Natural clusters could overlap, or be skewed in one direction (non spherical)
- c Clusters might be abstract
- d The number of clusters that minimize some variance metric might not be the number of clusters that the user expects

This means a clustering scheme needs a way to validate its output, which also validates the assumption. The process of evaluating results of a clusterer is

known as *cluster validity* [Charrad et al., 2014a].

Pat and Charrad et al. [2014a] mention a few ways to measure cluster validity (find the most optimal number of clusters or density assumption). This project will determine the optimal number of clusters based on how a cluster validity metric changes as the *number of clusters* assumption is varied.

This article will cover choosing *clusterers*, including its *distance measurements*, *aggregation functions*, and *metrics* used to validate the output number of clusters on a unique dataset. By the end, I hope to have a framework that will automatically find the best number of clusters in my dataset, cluster it into groups, and maybe even predict the best clusterer to use based on the data characteristics.

2 Problem Statement

Deciding on the number of clusters that should fit a dataset is hard. Imagine if the data has more than 3 dimensions - it is impossible to visualize without losing dimensionality. Estimating a benchmark cluster density is hard (especially in high dimensional spaces). It gets worse if multiple datasets need to be clustered and the data's density distribution varies across datasets. Removing human input and automating the 'number of clusters' decision, again, adds complexity. This blog refers to this type of clustering as (without a human inputting the assumption) *unsupervised clustering*.

This project's goal is to optimally choose the best number of clusters that fit a dataset. Later, the clustered examples will be fed into a ranking/multi-instance classification machine for bag classification. The un-clustered input is a large amount of examples with between two to several hundred examples per set. The examples comes from databases of building automation controllers, and each example has many features. The examples within a dataset can naturally be divided into groups by a trained human (more below). Hopefully, a combination of an aggregation function and distance metric can be found that optimally partition datasets without human input.

3 Data Description

Features include *NAME*, *TYPE*, *SLOPE*, [...]. Figure 1 shows typical features of a dataset :

By observing the name feature, a trained human could identify the number of clusters. The name features are organized hierarchically, with most points sharing a common suffix (samc), and lower abbreviations can represent location, equipment type, and device type. Generally, all instances that belong under one cluster (or 'system') are named similarly except for the ending tag. For example, in Figure 1 are (4) different systems, and each system share similar name feature

NAME	TYPE	SLOPE
samc.ctph.ahu22.ccv	LAO	0.003255
samc.ctph.ahu22.ccvfb	LAI	0.003906
samc.ctph.ahu22.combo	LDI	0
samc.ctph.ahu22.ead	LAO	0.000326
[...]		
samc.ctph.ahu23.ccv		
samc.ctph.ahu23.ccvfb		
samc.ctph.ahu23.combo		
samc.ctph.ahu23.ead		
[...]		
samc.l04.cv01.hwf		
samc.l04.cv01.hxr		
samc.l04.cv01.hxs		
samc.l04.cv01.stmv01		
[...]		
samc.l05.ahu24.ccv		
samc.l05.ahu24.ccvfb		
samc.l05.ahu24.combo		
samc.l05.ahu24.ead		

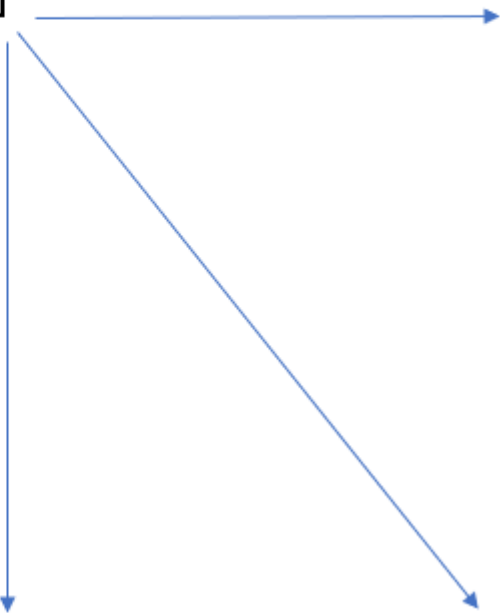


Figure 1: Example objects with features

suffixes except for the ending tag.

4 Data Exploration

4.1 Extract

Extracting information from existing databases is straight forward. There were many SQL databases located on remote servers. From each database, a few tables were selected which hold the bulk of useful features for this project. Extracting data included :

- Copying database files from remote servers to a local hard disc
- Create new table schemas for a new master database
- Iterate through copied database files and copy table data to new master

database

The new database is used as feeder to the Transform process stage.

4.2 Transform

There are many features to choose from the available data. This project focuses on clustering based on the ‘NAME’ attribute. Each instance’s name is a text feature. The name is organized hierarchically delimited by non-alphanumeric characters (‘.’, ‘-’). This section describes transforming the name feature into data that can be clustered.

4.3 Encoding

I transformed the name feature into a one-hot encoded set of text features :

- split feature along delimiter
- construct a vocabulary
- transform to one-hot encoded feature column

The result is a sparse feature space. All names in the dataset vocabulary are included to capture all variance in the data; a pooled feature is not useful for clustering.

One-hot encoding has implications on the clustering algorithm aggregator and distance metric. First, a vector will (depending on objects) vary by 2 directions in its space compared to a vector in the same system. This means the Euclidean distance between two vectors of the same system will be $norm(u - v) = [(u_1 - v_1)^2 + [...] + (u_n - v_n)^2]^{1/2}$. Also, Euclidean distance will increase following $\sqrt{2}, \sqrt{4}, [...]$.

Instances of a similar system should have features distributed “spherically” in their feature space. The clustering algorithm will have to recognize objects with high density in a similar feature space.

An object’s distance in space can be visualized nicely using multidimensional scaling (MDS), which is a dimensionality reduction technique that tries to maintain distances between features. Each systems cluster is mostly isolated from others, and each instance within a cluster is separated from others. Figure 2 represents one database of points that are transformed and reduced to (2) dimensions. There are (3) clearly defined clusters with a large amount of instances, and (2) small clusters with (2) and (3) instances each.

4.4 Load

Raw data was loaded from a SQL Server, transformed, and fed to the next step. Transforming raw data was quick, and the transformed features easily fit in

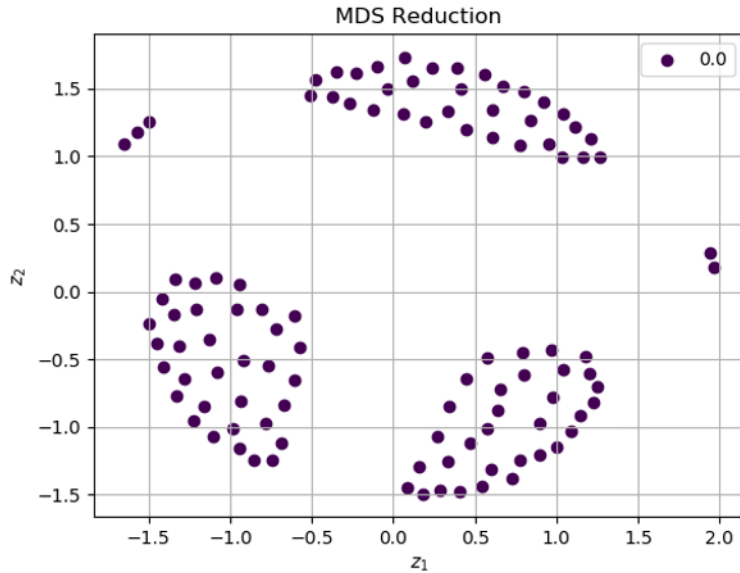


Figure 2: Dimensionality-reduction is a good way to visualize distributions of data across a large instance space

memory. Pipelines were tweaked to meet the clustering goal.

5 Clustering - Aggregation Functions and Evaluation Metrics

5.1 Aggregation Functions

This project uses two classes of aggregation functions :

- Hierarchical
- K-means

K-means is a general-purpose aggregation function, and is the first I found when learning about clustering. K-means attempts to separate examples into n -groups by minimizing the sum of squares between centroids and points assigned to the centroid. It is an iterative process which alternates between

- a first assigning points to a centroid,
- b and then shifting the centroid to minimize the sum of squares criteria [Authors, 2007]

Hierarchical or agglomerative clustering is the process of successively splitting or merging clusters based on a *linkage* criteria between clusters. This paper used *ward* and *average* linkage. Ward linkage criteria seek to minimize sum of squared differences within clusters, and average linkage minimizes average distance within clusters. Both methods are similar to K-means because they try to minimize variance within clusters.

Both average and ward linkage work well on spherical, data. My data is globular, but overlapping. Ideally I would use a linkage that works well with non-globular data, and also use a threshold restriction that prevents overlapping examples from merging to the same cluster.

5.2 Evaluation Metrics

Evaluation metrics are used to quantify the quality of clusters. They help us understand how well the clustering scheme fits a dataset. Evaluation metrics can also help us choose the optimal number of partitions to feed to the aggregation function. In this way an evaluation metric helps us solve the *cluster validity* problem.

Each evaluation metric included in the NbClust software package include some measure of compactness and isolation of the data [Charrad et al., 2014a]. By varying the input clustering assumption (number of clusters) and calculating evaluation metrics on the resulting clusters, we can see how well the ‘number of clusters’ assumption holds. In this way finding the optimal number of clusters is very similar to finding the optimal hyperparameters for some regression or prediction model.

To illustrate this, Figure 4 shows the *C Index* versus number of clusters assumption for test data. The left graph shows raw data, and the right graph shows C Index. The optimal number of clusters is the value which minimizes the C Index. The C Index is minimal at 3 or 5 clusters depending on how the C Index is implemented. It is ambiguous whether 3 or 5 is the correct number of clusters, although the data was generated with 5 clusters.

6 Ranking introduction

In ranking, the goal is to order a list of examples based on the relevance of each example to a question. In this project, examples are clustering schemes, and the question is a dataset. A quality ranking model will optimally sort a list of clustering schemes, such that the clustering scheme that is most likely to predict the desired number of clusters is recommended. Ranking has applications in document retrieval and question answering, but the format works well for this project goal.

Formally, let \mathbf{X} represent the universe of items, and $\mathbf{x} \in \mathbf{X}^n$ represents a list of n items, with $x_i \in \mathbf{x}$ the i th element in \mathbf{x} . Π is a permutation of all items in

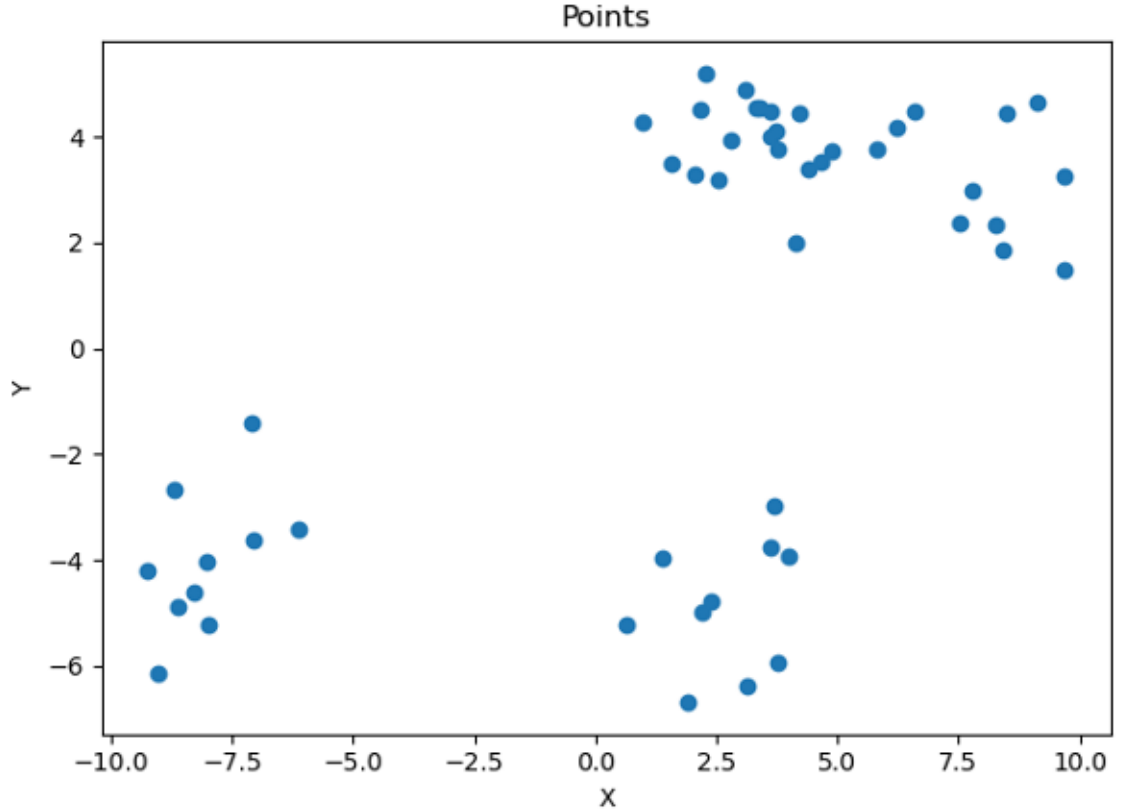


Figure 3: Sample data clusters

the list \mathbf{x} . The ranking function $f : \mathbf{X}^n \rightarrow \Pi^n$ produces a permutation of the list that maximizes the list's utility.

A clustering scheme is a combination of aggregation function and validity index. Many combinations of aggregation functions and validity indices form $\mathbf{x} \in \mathbf{X}^n$. The ranking of \mathbf{x} produces Π which is an ordering which outputs the clustering scheme most likely to predict the correct number of clusters for a dataset.

The training data of m elements is defined as $L^m = \{(\mathbf{x}, \pi) | \mathbf{x} \in \mathbf{X}^n, \pi \in \Pi^n\}$. Each $x_i \in \mathbf{x}$ is really a pair of (database features, clustering scheme), where database features are *context features*, and clustering schemes are *per-item features*. Each $x_i \in \mathbf{x}$ is therefore a feature vector which includes database features and clustering schemes, and each clustering scheme is given a relative ranking to how well it predicts on the database.

This project uses TF-Ranking Pasumarthi et al. [2019], which is a open source Tensorflow deep neural network learning framework. Other ranking functions

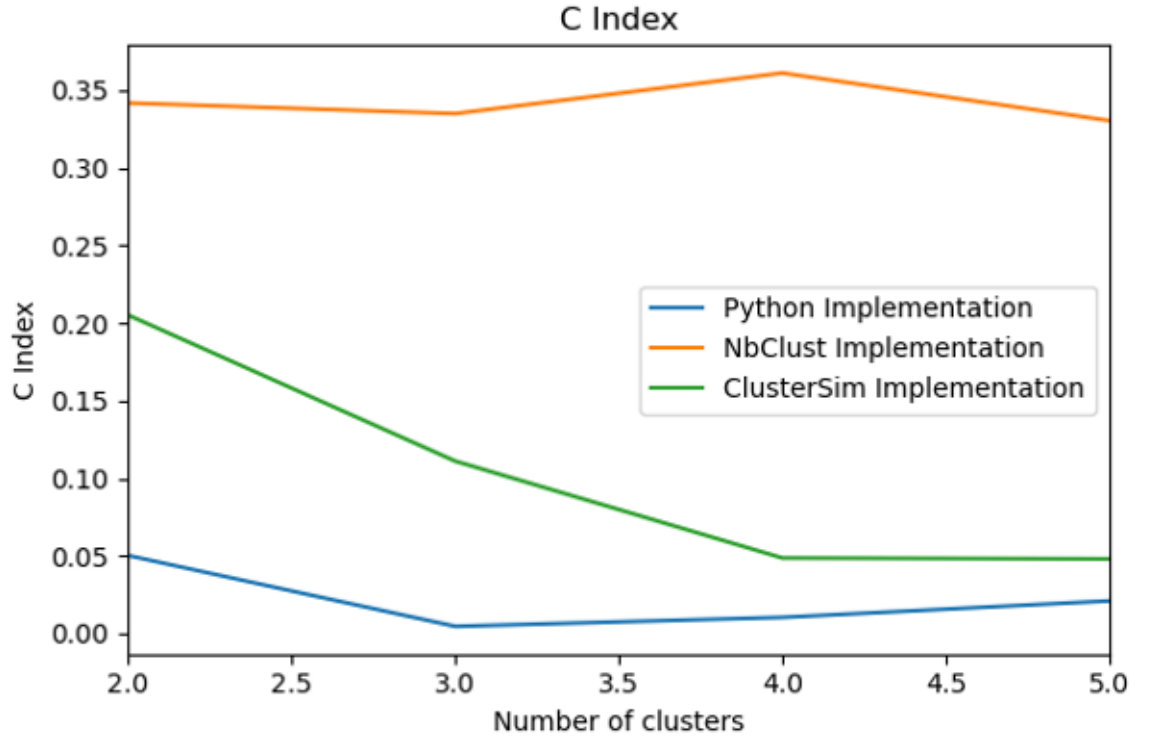


Figure 4: C Index calculated on sample data

could be used, including linear, gradient trees, and support vector machines, but this project only tested a neural network as the scoring function [Ai et al., 2019].

6.1 Metrics

Normalized discounted cumulative gain (NDCG) is the ranking metric used to measure goodness of fit. NDCG measures the cumulative gain of a ranking. Relevant items ranked early in the list add gain, and relevant items ranked last in the list are penalized.

6.2 Training Data features - Context

There are challenges to choosing an optimal number of clusters for this problem.

First, the number of features in each dataset varies between ten to a thousand. This means the closeness and compactness of clusters changes between datasets.

Because each evaluation metric measures some form of geometric or statistical properties, a metric might perform well on a small dataset and poorly on large datasets.

Second, each datasets correct number of clusters varies. How well does a metric perform when the ratio of objects to clusters changes?

Last, the dimensionality of an instances feature space changes between datasets. Again, dimensionality affects the relative density of objects within a space.

This project looks for a relationship between accuracy of validation indices and the features of a dataset that define the datasets geometric properties. For this experiment, the following features of each dataset are used :

1. Number of examples or objects
2. Feature space dimensionality
3. Variance of word lengths
4. Ratio of number of examples to number of features
5. Number of NAME word tokens 1 token long, 2 tokens long ... to 5 tokens long

6.3 Training Data Features - Per Item

Per-item features include information about a) how a dataset is clustered and b) how the optimal number of clusters is predicted. A dataset is clustered with a *clustering algorithm*, and the number of clusters is predicted with a *validity index*.

In addition, dimensionality reduction was alternately applied before clustering data. The full set of per-item features includes the categorical features (Clusterer, index, dimensionality reduction). The full set of categorical features is shown in Appendix A.

6.4 Data Labeling & NbClust

Each dataset was clustered using software packages NbClust [Charrad et al., 2014b] and Scikit-Learn [Pedregosa et al., 2011]. Then, the predicted number of clusters was calculated from evaluation metrics. The label for the (context-feature, per-item feature) pair is the accuracy of the predicted number of clusters to the actual number of clusters. See Figure 5 for an illustration of per-item features, context features, and labels.

Before training, labels were binned into (5) categories. Labels with a value of (5) indicated the most accurate clustering scheme, and labels of (1) indicated the least accurate clustering scheme. There are up to 200 labels per (Context Features, Peritem Features) pair. This means there are up to $200 \div 5 = 40$ labels with value (5), 40 labels with value (4), etc.

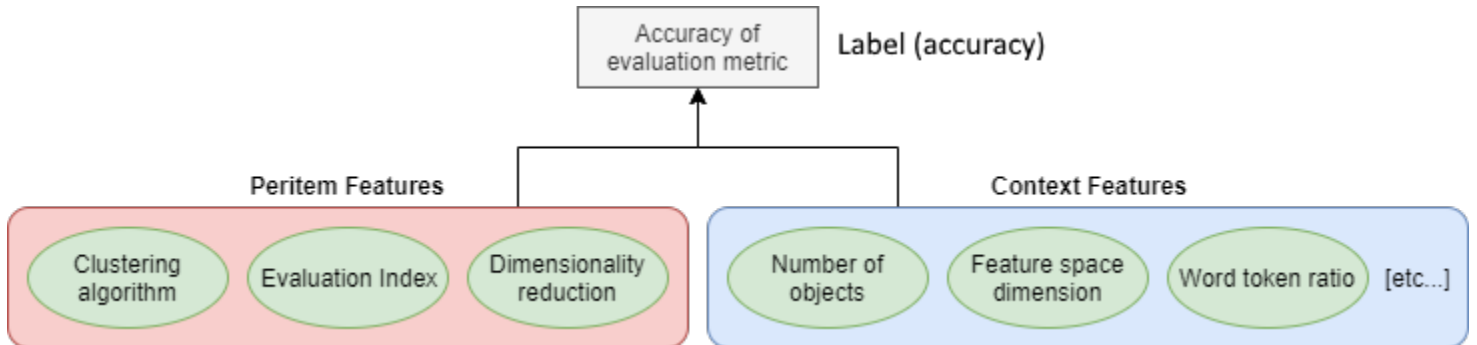


Figure 5: Ranking Per-item features (clustering parameters) and context features (features of the dataset)

6.5 Ranking Results

The ranking scoring model was trained, and NDCG results were recorded (Figure 6). The ranking model showed a small improvement in gain. When measuring NDCG of the top 5 ranked clustering schemes the ranking model is able to improve gain from a minimum of 0.67 to a plateau of 0.75 (Figure 6).

Notice how NDCG improves less when measuring the top 25 items in the ranked list (Figure 7). This indicates there is more ambiguity in the relevance of clustering schemes lower in the list.

7 Conclusion and Improvements

7.1 Conclusion

While the ranking model did improve NDCG modestly, I do not believe the model found a significant relationship between the database features (context features) and clustering schemes (peritem features).

I hypothesize that the ranking model recognized consistently poorly performing validity indices and learned to rank them low. Conversely, the model might recognize validity indices that consistently perform better, and rank those high.

7.2 Improvements

There are a lot of opportunities for improvements in this project, including :

- a. Use different aggregation functions that perform better with unbalanced datasets
- b. Add connectivity constraints to aggregation functions

ndcg@5
tag: metric/ndcg@5

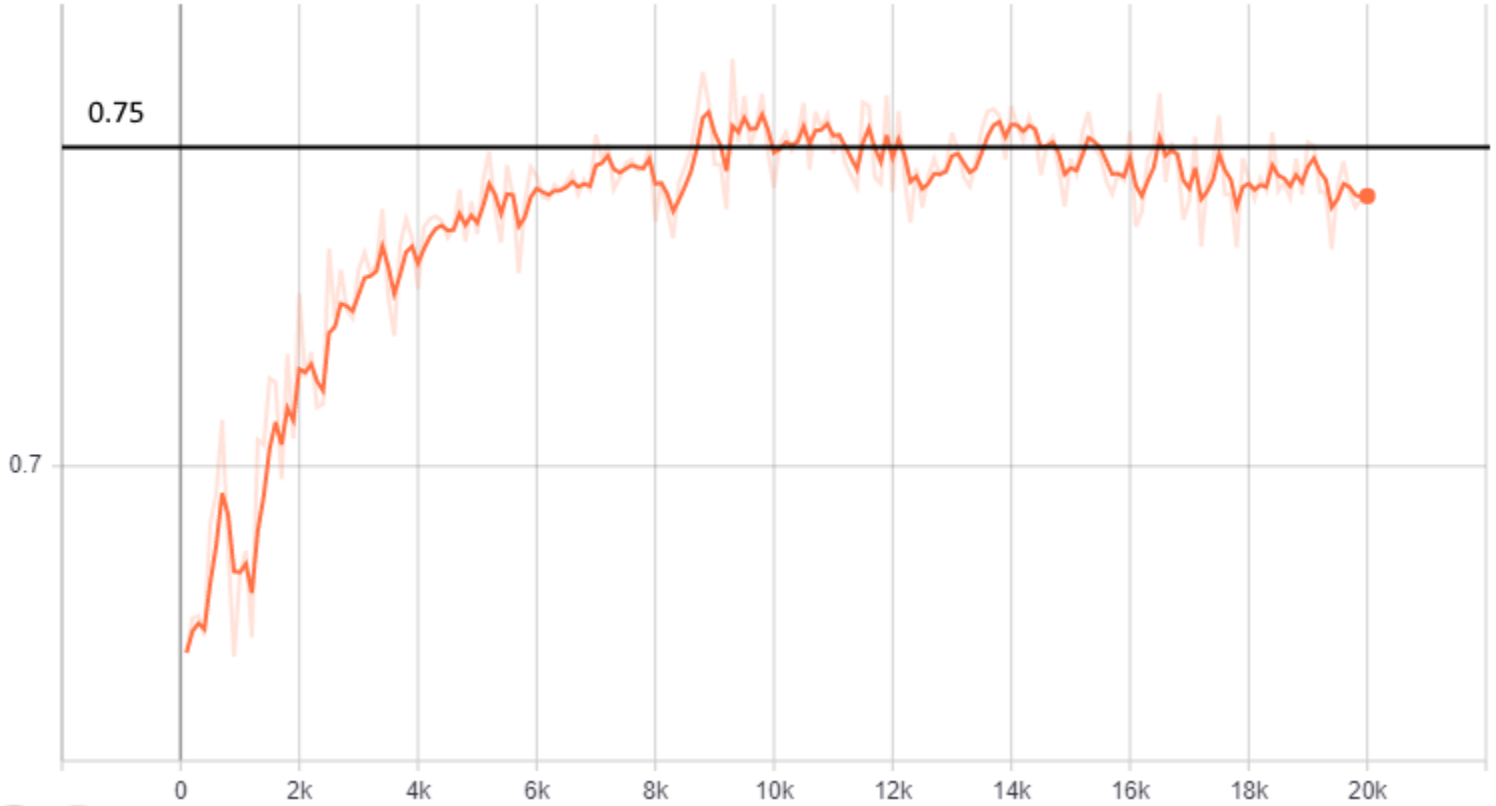


Figure 6: NDCG of top 5 clustering schemes in list

- c. Use a regression or DNN model to predict the best number of clusters based on validity index results directly

Aggregation Functions have personalities and perform slightly differently from each other. Kmeans performs well on evenly spaced data with few clusters, while Agglomerative aggregation performs well with many clusters. Data characteristics to consider include a) number of clusters, b) even cluster sizes, c) cluster geometry (manifolded / flat), d) outlier presence (See Figure 9).

This projects aggregation functions only included Kmeans and Agglomerative, but others like Affinity Propagation, Mean-shift, DBSCAN, or birch may perform better.

ndcg@25
tag: metric/ndcg@25

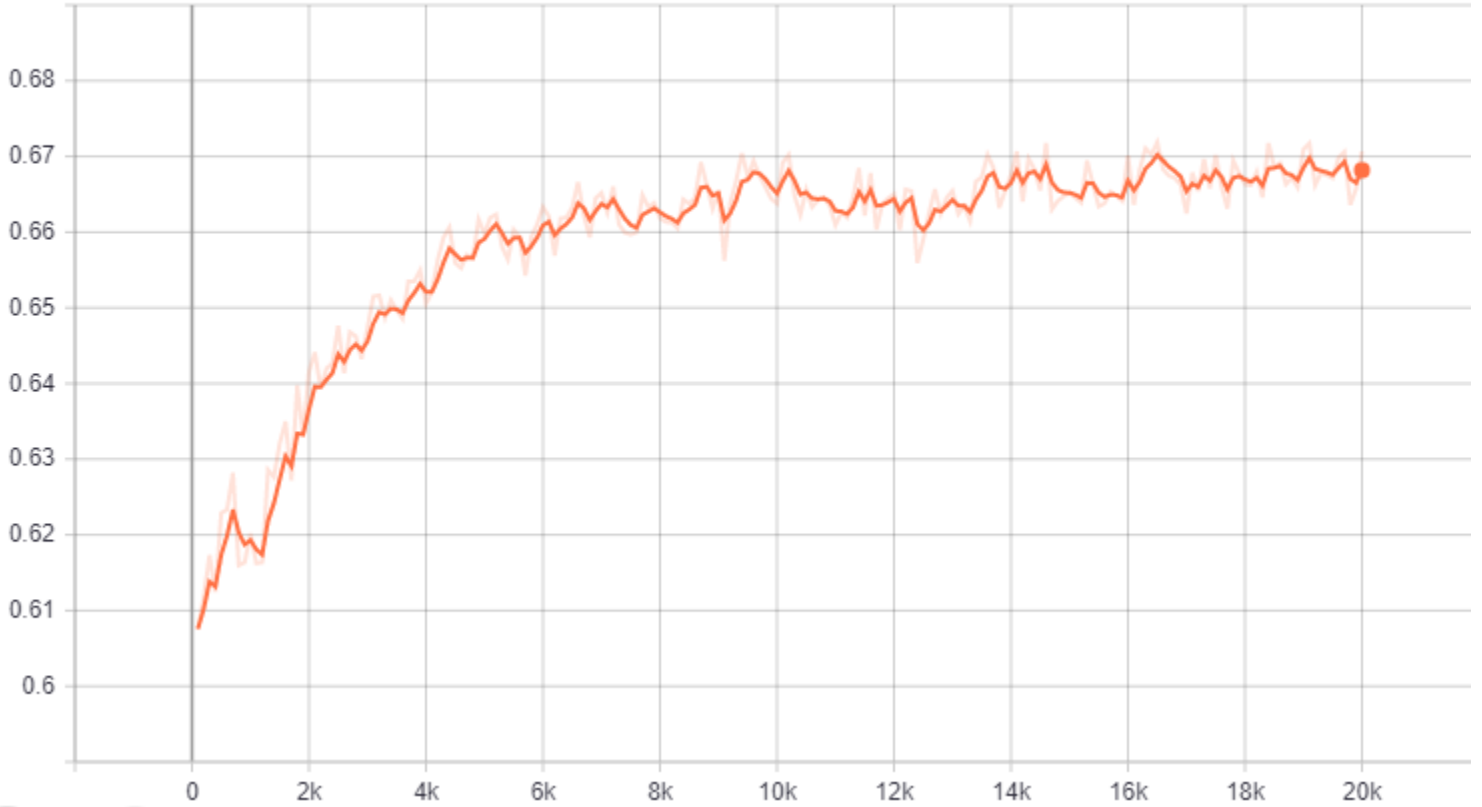


Figure 7: NDCG of top 25 clustering schemes in list

Connectivity constraints are distance thresholds that prevent clusters from being merged that are not adjacent. Because this projects text data is highly structured and hierarchical, connectivity constraints could give a better estimate of cluster boundaries than a density metric. A connectivity constraint could be made to disallow clustering of text features that are not directly adjacent when the text is sorted alphabetically (Figure 9).

I believe that most of this projects data could be better clustered with carefully crafted connectivity constraints instead of density estimates.

Direct inference with DNN / regression When the 'optimal' number of clusters is chosen from a validity index, the optimal number of clusters occurs at some function of the validity index. Generally, if \mathbf{X} is the validity index vector, where $x_i \in \mathbf{X}$ is the i th validity measurement, the goal is to find a function

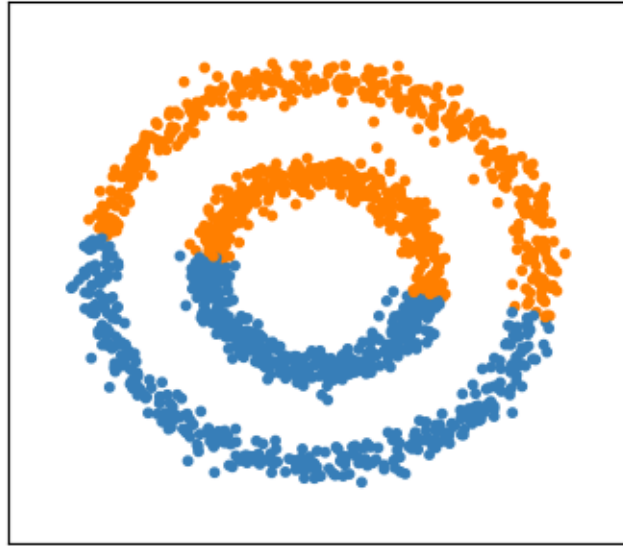


Figure 8: Kmeans clustering on non-flat data

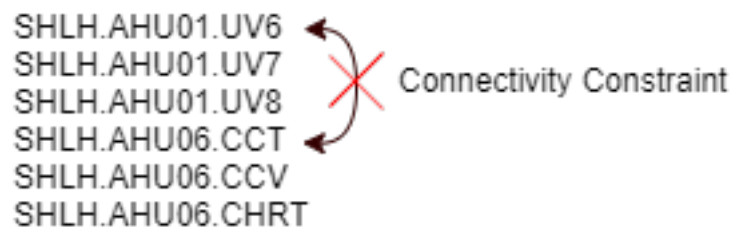


Figure 9: Connectivity Constraint

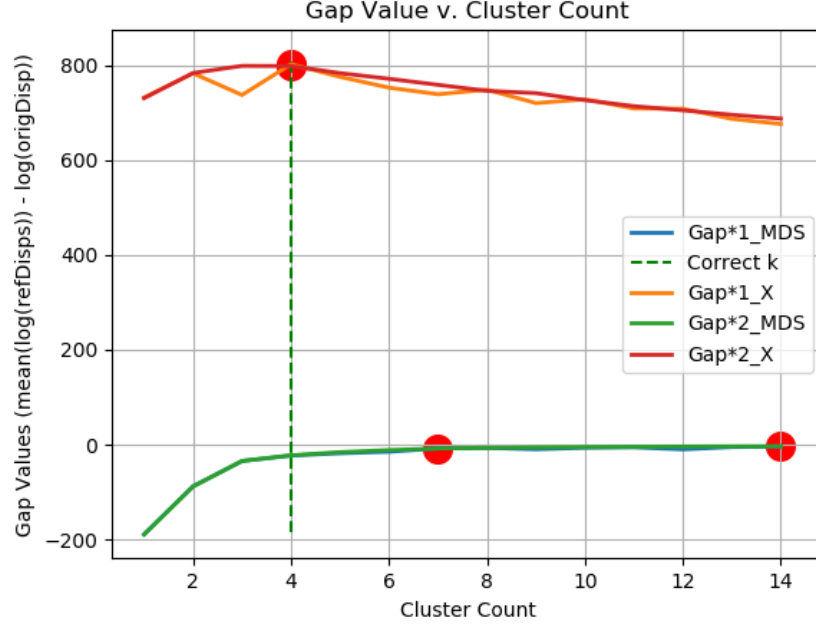


Figure 10: Gap statistic versus number of clusters

$f : \mathbf{X} \rightarrow \mathbb{Z}$ where \mathbb{Z} is the set of all real integers. The function f should maximize or minimize some sort of criteria. Because most validity indices measure cluster density or between-cluster spacing, it is natural for the function f to maximize cluster density or minimize entropy (Figure 10). Sometimes the function f is the function that maximizes the second-difference (usually called the 'knee') of measurements.

I wonder if applying a general DNN or sequential classifier could infer the correct number of clusters better than a simple min/max. This problem could be treated similar to the multiclass classification problem, where the input vector \mathbf{X} is the validity index vector, and the correct number of clusters is the index of the maximum softmax activated logit (Figure 11).

References

Pattern Recognition. Academic Press, 4 edition.

Q. Ai, X. Wang, S. Bruch, N. Golbandi, M. Bendersky, and M. Najork. Learning groupwise multivariate scoring functions using deep neural networks. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR '19*, page

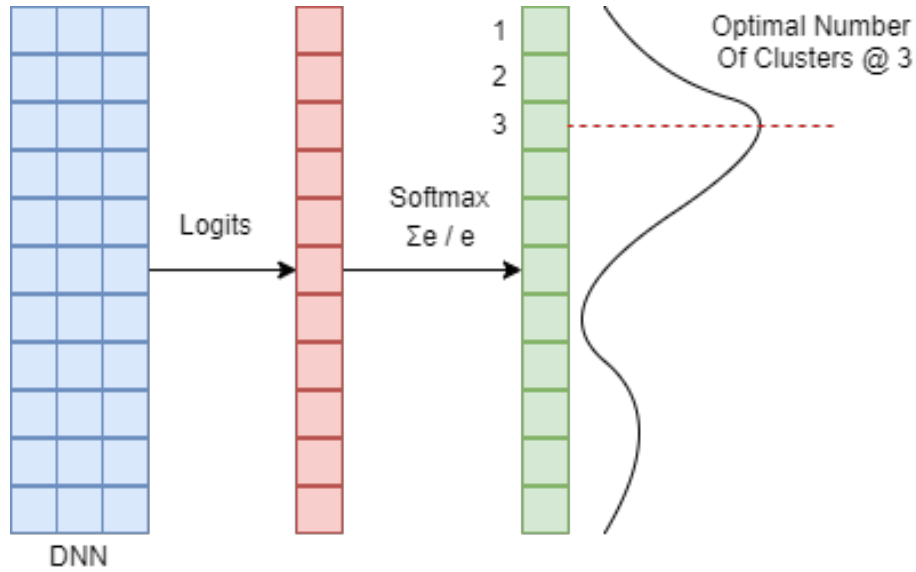


Figure 11: Proposed DNN predicting optimal number of clusters

85–92, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368810. doi: 10.1145/3341981.3344218. URL <https://doi.org/10.1145/3341981.3344218>.

T. V. Authors. *K-means fundamentals*, 2007. URL <https://www.vlfeat.org/api/kmeans-fundamentals.html>.

M. Charrad, N. Ghazzali, V. Boiteau, and A. Niknafs. Nbclust : An r package for determining the relevant number of clusters in a data set. *Journal of Statistical Software*, 61(6), 2014a.

M. Charrad, N. Ghazzali, V. Boiteau, and A. Niknafs. NbClust: An R package for determining the relevant number of clusters in a data set. *Journal of Statistical Software*, 61(6):1–36, 2014b. URL <http://www.jstatsoft.org/v61/i06/>.

R. K. Pasumarthi, S. Bruch, X. Wang, C. Li, M. Bendersky, M. Najork, J. Pfeifer, N. Golbandi, R. Anil, and S. Wolf. Tf-ranking: Scalable tensorflow library for learningto-rank. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2970–2978, 2019. doi: 10.1145/3292500.3330677.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn:

Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

A Per-item and Context Features

All clustering indices used in this project

[KL, CH, Hartigan, CCC, Marriot, TrCovW, TraceW, Friedman, Rubin, Cindex, DB, Silhouette, Duda, PseudoT2, Beale, Ratkowsky, Ball, PtBiserial, Frey, McClain, Dunn, Hubert, SDindex, Dindex, SDbw, `gap_tib`, `gap_star`, `gap_max`, Scott]

Clustering algorithms used in project

[average, kmeans, ward.D, Ward.D2]