



WPI



VESTIGO
VENTURES

Developing Models to Visualize & Analyze User Interaction for Financial Technology Websites

Project Team

Amanda Ezeobiesi Computer Science
Guy Katz Industrial Engineering
Alissa Ostapenko Computer Science and
Mathematical Sciences

Project Advisor

Professor Michael Ginzberg
Foisie Business School

Project Co-Advisors

Professor Rodica Neamtu
Department of Computer Science

Professor Sara Saberi
Foisie Business School

Professor Jon Abraham
Department of Mathematical Sciences

This report represents the work of WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on its website without editorial or peer review. For more information about the projects program at WPI, please see:

<http://www.wpi.edu/academics/ugradstudies/project-learning.html>

Abstract

Vestigo Ventures manually processes website traffic data to analyze the business performance of financial technology companies. By analyzing how people navigate through company websites, Vestigo aims to understand different customer activity patterns. Our team designed and implemented a tool that automatically processes clickstream data to visualize different customer activity within a website and compute statistics about user activity. This tool will provide Vestigo insight on the effectiveness of their clients' website structures and help them make recommendations to their clients.

Acknowledgements

Our team would like to extend our sincere gratitude to the individuals mentioned below, from Vestigo Ventures, Cogo Labs, and Link Ventures, as well as our advisors for their support and encouragement throughout the duration of this project:

Vestigo Ventures

Ian Sheridan, a managing director and co-founder of Vestigo Ventures, who made this project possible. He established the initial connection with WPI and checked in on us from time to time.

Frazer Anderson, a data-driven strategist, an investment analyst at Vestigo Ventures, and our project sponsor. He provided us with the resources needed and support, enabling our project's success.

Cogo Labs

Rob Fisher, the newly appointed Chief Executive Officer at Cogo Labs, was the mastermind behind our project. His thoughts, guidance, and consistent feedback were vital to the success of this project.

Daniel Brady, an analytics manager at Cogo Labs, helped us to fully understand the competitive intel database and user interface that held the company's various databases.

Link Ventures

Todd Federman and D'Mitri Joseph, technical fellow and software engineer, respectively, at Link Ventures helped us integrate our API into the existing business analysis tool, making our project a fully functioning application that anyone with proper access to the tool can use.

Our Advisors

Professor Michael Ginzberg, the advisor for the Fintech/Wall Street Project Center, ensured that the project was conducted smoothly by giving us access to the resources we needed prior to and during the project.

We thank our major project advisors, Professors Rodica Neamtu (Computer Science), Sara Saberi (Industrial Engineering), and Jon Abraham (Mathematical Science) who supported our team throughout the entire duration of the project. Their enthusiasm, kind attention, and diligent guidance increased our passion for the project and allowed the MQP to be as impactful as possible.

Executive Summary

Introduction

Vestigo Ventures, a venture capital firm investing in financial technology companies, aims to understand the different ways that people interact with their clients' websites and how many visitors make a purchase. However, Vestigo's clients have diverse website structures and varying definitions of what a purchase is, making it difficult for the firm to use a generalized visualization technique. Our goal was to create an easy-to-use tool that automatically processes internet traffic data to provide Vestigo, and similar companies interested in website performance analysis, insight into the effectiveness of a company website. We developed the *Website Private Investigator (WPI)*, an Application Programming Interface (API) that builds an interactive graph illustrating how people navigate through a given website. Moreover, our tool calculates statistics about customer interaction, such as the percentage of visits that start or end at a certain page within the company's website, allowing Vestigo analysts to understand customer activity patterns in depth.

In the following subsections, we discuss our project management approach, highlight the architecture, features, and performance analysis of our tool. In addition, we provide recommendations and takeaways from our project experience.

Methodology

We organized our work in four sprints, each two weeks long. Each sprint comprised of multiple meetings within the team and with our faculty advisors. Moreover, we consulted our company sponsor, and other knowledgeable employees from Vestigo's partnering company, Cogo Labs. Cogo is an incubator of internet companies and provided the internet traffic data we worked with to develop the *WPI*. We used the feedback from our meetings to guide our project work and to iteratively develop our tool in four phases:

- (1) understand the dataset and experiment with visualization techniques using Python libraries,
- (2) develop the API using Python and Github for version control,
- (3) document our API, and
- (4) deploy the API into production with a user interface (UI).

At the conclusion of our project, we produced a command-line interface (API) version of the *WPI*, as well as a deployed version complete with a graphical UI. To better visualize the components of our tool, we present our work in its deployed form.

Website Private Investigator - Architecture and Features

Phase 1: Gathering Data

To use our tool, an analyst must first query Cogo’s internet traffic database with a company website and start date of interest. We provided a query template that Vestigo can follow to easily gather the data in the format that the *WPI* expects. After running the query, an analyst needs to download the results to a file, producing a dataset of Uniform Resource Locators (URLs) people visited while browsing a company website during an input date range.

Phase 2: Building the Graph

Starting a Job

As illustrated in **Figure 1**, The *Website Private Investigator* expects an input data file, company website name, and a data range within the dataset for processing. The start date must be specified, however, the end date is optional; by default, *WPI* will process all of the data present in the data file from the start date. Clicking “Execute Now” starts a job to build an interactive graph.

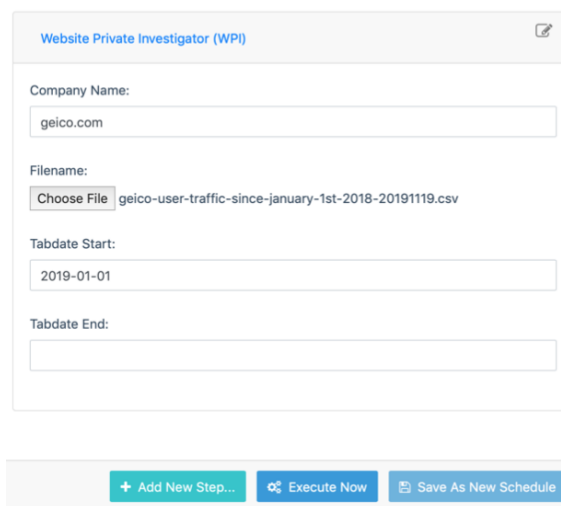
The screenshot shows the 'Website Private Investigator (WPI)' web interface. It has a light gray header with the title and a small icon. Below the header, there are four input fields: 'Company Name' with 'geico.com' entered, 'Filename' with a 'Choose File' button and the filename 'geico-user-traffic-since-january-1st-2018-20191119.csv' displayed, 'Tabdate Start' with '2019-01-01' entered, and 'Tabdate End' which is empty. At the bottom of the form, there are three buttons: '+ Add New Step...', 'Execute Now', and 'Save As New Schedule'.

Figure 1: Specifying input arguments to the Website Private Investigator.

Processing the Data

Cleaning the Data

To remove user-specific information while retaining general activity patterns, we reduce each URL to only the company domain name and the website path. For example, if given the URL <https://www.wpi.edu/admissions/graduate/how-to-apply?itemId=item-27>, we simplified this to www.wpi.edu/admissions/graduate/how-to-apply.

Building User Paths

Our tool uses *pandas* to separate the dataset by unique visits to build user flow paths. A visit is defined by a unique combination of user ID and tabdate, and each flow path is a list of URLs. To better understand which pages people start and end their browsing activity, we group URLs into start pages, intermediate pages, and exit pages.

Clustering Similar URLs and Computing Statistics

To further summarize the different browsing information within the dataset, we use *diffli* to cluster start, intermediate, and exit URLs by string similarity. Finally, using Python packages *scipy* and *math*, we calculate the percentage of visits that landed on each start, intermediate, and end page.

Interacting with the Graph

After several minutes, an interactive graph will load in the browser, as shown in **Figure 2**. We developed the graph using Plotly (Python 3 and JavaScript) and NetworkX (Python 3), open-source graph visualization packages. Each cluster in the start, intermediate, and exit groups is a node in the graph, and edges connect nodes according to the paths built during the preprocessing step. Start web pages are colored green and placed at the top of the graph, exit web pages are red and placed at the bottom, and intermediate webpages are blue, placed between start and exit pages, to better visualize the different components of the different user flow paths.

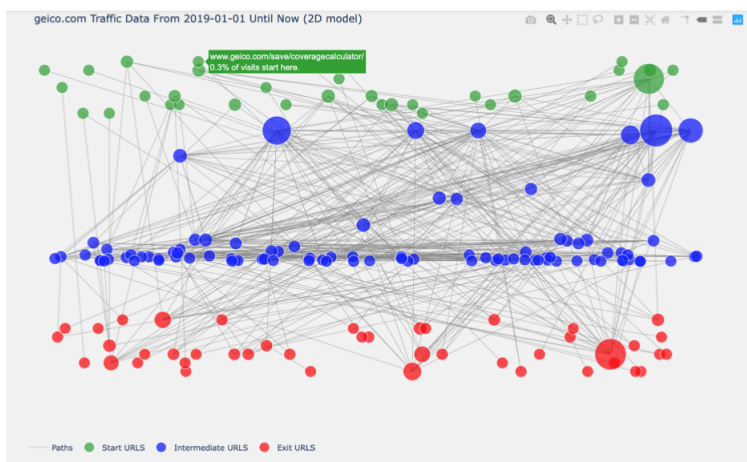


Figure 2: Interactive graph showing user traffic flow for geico.com

Hovering

As illustrated in **Figure 2**, hovering over any node will show the webpages represented by the node, as well as the percentage of visits that pass through these webpages.

Website Private Investigator Special Features

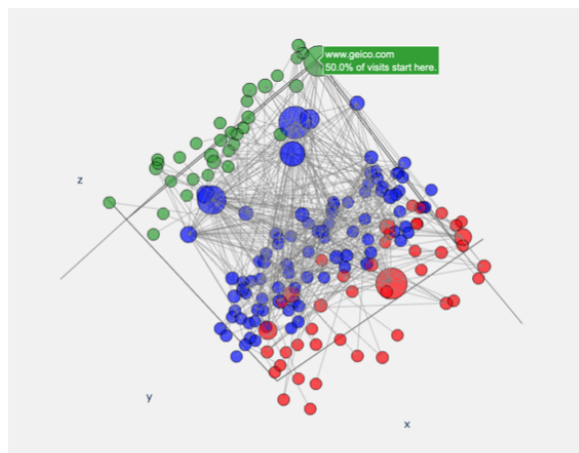


Figure 3: User Traffic Graph in Three Dimensions

Displaying the graph in Three Dimensions

The two-dimensional graph often has many overlapping nodes. For an analyst to view the graph from different angles, we added a 3D toggle which visualizes the data in three dimensions, as illustrated in **Figure 3**.

Highlighting Adjacent Nodes

An analyst may want to see which pages a user went to immediately before or after a particular page, especially if this is an exit page (such as a purchase page) or an entry page (such as a login page). The highlighting features allows an analyst to click on a node of interest to see the

nodes immediately connected to it. **Figure 4** illustrates the possible pages of people went to immediately after checking their account pages on *Geico.com*.

Highlighting User Paths

Moreover, it may be useful to see all pages that people viewed if they passed through a particular webpage. As with *Highlighting Adjacent Nodes*, an analyst can click on a node to highlight all potential pages on a path containing this page. This way, the analyst can identify possible entry, intermediate, and exit points that people could have traveled to before, for example, confirming a purchase.

Keyword Search

Clustering is not a precise technique, and a webpage of interest may be clustered together with less relevant pages. To isolate visit statistics for particular pages of interest, an analyst can use the *WPI*'s keyword search functionality to search for webpages containing a particular term, such as *confirm*. **Figure 5** illustrates the keyword search functionality.

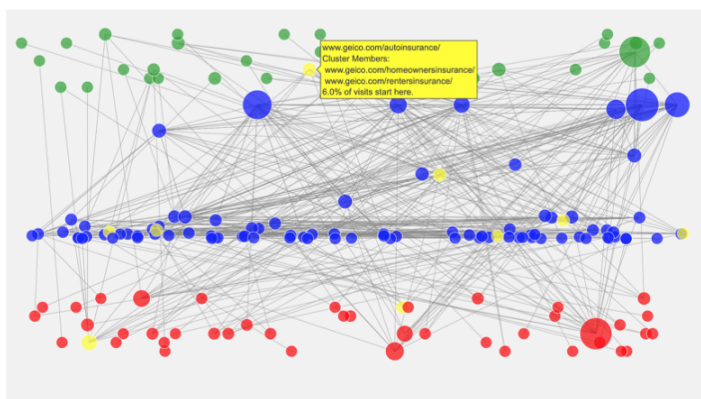


Figure 4: Highlighting pages immediately adjacent to insurance-related pages

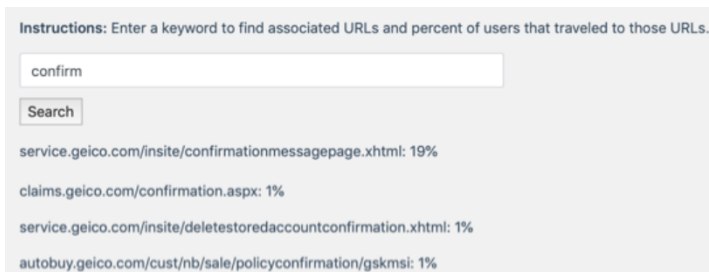


Figure 5: Keyword search of geico.com for 'confirm'

Experimental Analysis

To evaluate our clustering methods, as well as explore financial technology website statistics more in-depth, we used techniques from mathematics and industrial engineering.

Clustering Evaluation

There are many techniques for grouping text data, however, we focus our analysis on Agglomerative Clustering and Gestalt Pattern Matching. Table 1 details our approaches to these techniques.

Agglomerative Clustering	Gestalt Pattern Matching
To capture local character-to-character differences between URLs, we compute the edit distances between each pair of URLs in a dataset. We used scikit-learn’s implementation of agglomerative clustering to group URLs.	We use <i>difflib</i> to compute the Gestalt ratio between URL pairs and grouped together URLs above a threshold ratio. The Gestalt ratio ranges from 0 to 1, where 1.0 indicates a perfect match. We experimented with different ratios between 0.6 and 1.0. The Gestalt ratio reflects sequence-level comparisons between URLs.

Table 1: Clustering techniques

To evaluate our methods, we manually labeled URLs collected from four websites, one of which is *geico.com*. We compared the outputs of the agglomerative clustering and gestalt pattern matching to our labels by computing *Adjusted Rand Index (RI)* and *V-measure*, which are defined in **Table 2**.

Adjusted Rand Index	V-measure
<p><i>Rand Index (RI)</i></p> $\frac{a + d}{a + b + c + d}$ <p>a: same cluster, same label b: same cluster, different label c: different cluster, same label d: different cluster and label</p> <p><i>Adjusted Rand Index (ARI):</i></p> $\frac{RI - Expected(RI)}{Max(RI) - Expected(RI)}$	$V_{\beta} = (1 + \beta) \frac{h * c}{\beta * h + c}$ <p><i>h</i>: homogeneity (a cluster should have only members of the same class) <i>c</i>: completeness (all class samples should be in the same clusters) β: <i>beta</i>, harmonic mean weight of <i>h</i> and <i>c</i></p>

Table 2: Clustering evaluation metrics

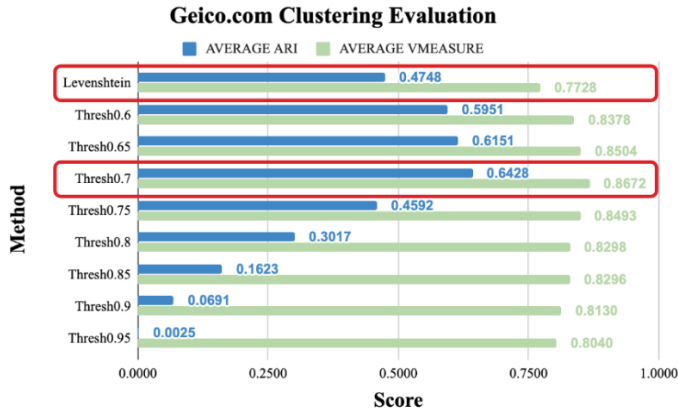


Figure 6: Clustering Evaluation Comparison

In general, Gestalt pattern matching with a threshold around 0.70 and 0.75 outperforms agglomerative clustering for URL grouping in financial technology websites. We programmed the tool to use a threshold of 0.75 based on our experimental results. **Figure 6** illustrates the ARI and V-measure of clustering URLs in *Geico.com* user traffic data.

Linear Regression

Vestigo requested to find ways to calculate the customer conversion rate of a website, or the percentage of visits that end in a customer purchase. Although it is difficult to compute exact conversion rates, we used the WPI tool to approximate the statistics. Using the financial company Wells Fargo as an example, we created a multiple linear regression of the conversion rate. The data we used had been collected over a period of 50 week. The statistical information for each week was collected from the *WPI* along with an approximated conversion rate, which was the percentage of visits that traveled through a web page with the keyword ‘billpay.’

Plotted in **Figure 7** is the multiple linear regression conversion rate compared to the approximated conversion. We can solve for Wells Fargo’s conversion rate using the following equation:

$$\text{Conversion Rate} = -0.06 * (\text{unique visits}) + 0.09 * (\text{unique users}) + 0.15 * (\text{percent return users}) - 0.66 * (\text{average pages in a visit}) + 4.49$$

The equation is not accurate, with an r squared value of only 0.12. However, the general trend of the week to week customer conversion rate is similar to that of the approximated conversion.

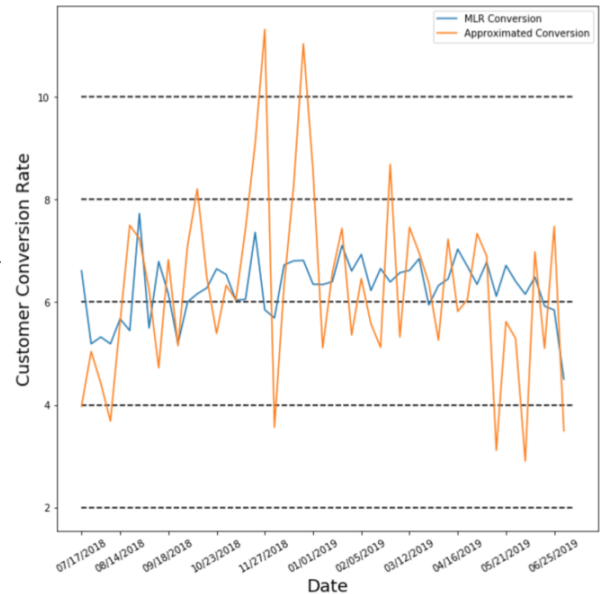


Figure 7: Time series linear regression conversion vs. approximated conversion

Recommendations & Conclusions

Our tool works best for small data sizes. We recommend that users of the tool use at most 50 MB of data to ensure graph creation under one hour. On average, *WPI* will process datasets of approximately 10,000 rows in under 60 seconds. Furthermore, having at least 500 rows of data will ensure a good quality of the graph. While this might limit collecting data for time series regression models, it will ensure that the graph has enough data to provide useful information. Overall, the project allowed the team to apply interdisciplinary knowledge gained from classroom study to real-world data. The team learned how to work win a business setting, with diverse groups of people, and how to create business focused applications which can be used by analysts now and in the future.

Table of Contents

ABSTRACT.....	1
ACKNOWLEDGEMENTS.....	2
EXECUTIVE SUMMARY.....	3
TABLE OF CONTENTS.....	9
LIST OF FIGURES.....	10
LIST OF TABLES.....	11
*** Sections 1 – 8 have been redacted per the request of our project sponsor. ***	
REFERENCES.....	13
APPENDIX A: Redacted	
APPENDIX B: A Hitchhiker’s Guide to the Clustering Galaxy.....	17

List of Figures

Figure 1 - Specifying Input Arguments to the Website Private Investigator.....	4
Figure 2 - Interactive Graph Showing User Traffic Flow for geico.com.....	5
Figure 3 - User Traffic Graph in Three Dimensions	5
Figure 4 - Highlighting Pages Immediately Adjacent to Insurance Related Pages	6
Figure 5 - Keyword Search of geico.com for ' <i>confirm</i> '	6
Figure 6 - Clustering Evaluation Comparison	7
Figure 7 - Time Series Linear Conversion vs. Approximated Conversion	8

List of Tables

Table 1 - Clustering Techniques **7**

Table 2 - Clustering Evaluation Methods **7**

The rest of the paper is removed by request from the sponsoring company.

References

“About-Cogo Labs.” *Cogo Labs*, <https://www.cogolabs.com/about>.

Albert, Bill, and Donna Tedesco. “Clickstream Data.” *Clickstream Data - an Overview* | *ScienceDirect Topics*, <https://www.sciencedirect.com/topics/computer-science/clickstream-data>.

Allahyari, Mehdi, et al. “A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques.” *ArXiv.org*, Cornell University, 28 July 2017, <https://arxiv.org/abs/1707.02919v2>.

Anderson, Frazer. Personal Interview. 9 Sept. 2019.

Black, Paul E. "Ratcliff/Obershelp pattern recognition." Dictionary of algorithms and data structures 17, 2004.

Bonaccorso, Giuseppe. Mastering Machine Learning Algorithms. Packt Publishing Limited, 2018.

Bonnin, Rodolfo, and Claudio Delrieux. Machine Learning for Developers: Uplift Your Regular Applications with the Power of Statistics, Analytics, and Machine Learning. Packt, 2017.

Brady, Daniel. Personal Interview. 22 October 2019.

Chen, James. “Venture Capital Definition.” *Investopedia*, Investopedia, 29 Sept. 2019, <https://www.investopedia.com/terms/v/venturecapital.asp>.

“Conversion Funnel.” *Wikipedia*, Wikimedia Foundation, 11 Oct. 2019, en.wikipedia.org/wiki/Conversion_funnel.

“Data Analysis.” *Data Analysis - Pearson's Correlation Coefficient*, University of the West of England, Bristol, 2019, <http://learntech.uwe.ac.uk/da/Default.aspx?pageid=1442>.

Desai, Falguni. “The Evolution Of Fintech.” *Forbes*, Forbes Magazine, 9 Feb. 2016, <https://www.forbes.com/sites/falgunidesai/2015/12/13/the-evolution-of-fintech/>.

Fisher, Robert. Personal Interview. 2 Oct. 2019.

“GEICO At A Glance.” GEICO, <https://www.geico.com/about/corporate/at-a-glance/>.

Gilleland, Michael. “Levenshtein Distance, in Three Flavors.” *Levenshtein Distance, in Three Favors*, Merriam Park Software,

<https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/LevenshteinDistance.htm>.

Han, Jiawei, et al. *Data Mining: Concepts and Techniques*. Elsevier, 2012.

“Home-Vestigo.” *Vestigo Ventures*, <https://www.vestigoventures.com/>.

Hunter, John, et al. “*Installation.*” Matplotlib, 2019, <https://matplotlib.org/>.

“*Hypertext Transfer Protocol (HTTP) Status Code Registry.*” Hypertext Transfer Protocol (HTTP) Status Code Registry, www.iana.org/assignments/http-status-codes/http-status-codes.xhtml.

Jinka, Preetam. “*Slow Queries? Move Fast to Fix Them.*” Database Monitoring Tools from VividCortex, www.vividcortex.com/blog/slow-queries-move-fast-fix-them.

Kaushik, Avinash. “Web Analytics 2.0: The Art of Online Accountability and Science of Customer Centricity.” *O'Reilly | Safari*, Sybex, <https://learning.oreilly.com/library/view/web-analytics-20/9780470529393/ch08.html>.

Keohane, Dennis. “*David Blundin's Cogo Labs' Formula for Startup Success.*” VentureFizz, 13 July 2018, <https://venturefizz.com/stories/boston/david-blundins-cogo-labs-formula-startup-success>.

Markus, Justas. “*What Is Conversion Funnel? - Learn How to Optimize Your Conversions.*” Oberlo, Oberlo Dropshipping App., 30 Oct. 2019, www.oberlo.com/ecommerce-wiki/conversion-funnel.

“*NetworkX.*” NetworkX, 2019, <https://networkx.github.io/>.

“*Mission of NumFOCUS.*” NumFOCUS, 2019, <https://numfocus.org/community/mission>.

“*NumPy.*” NumPy, 2019, <https://numpy.org/>.

“*Python Data Analysis Library.*” Pandas, 2019, <https://pandas.pydata.org/>.

“*Getting Started with Plotly.*” Getting Started with Plotly | Python | Plotly, <https://plot.ly/python/getting-started/>.

Robinson, Edward, and Julie Verhage. “*Quicktake: Fintech.*” *Bloomberg.com*, Bloomberg, <https://www.bloomberg.com/quicktake/financial-technology-companies-disrupt-comfy-banks-quicktake>.

“*Learn.*” Scikit, <https://scikit-learn.org/stable/>.

Sheil, Humphrey, et al. “*Predicting Purchasing Intent: Automatic Feature Learning Using Recurrent Neural Networks.*” 21 July 2018, doi:arXiv.1807.08207.

“*Sklearn.metrics.v_measure_score.*” Scikit, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_measure_score.html.

Straders, Anne. “*What Is Fintech? Uses and Examples in 2019.*” *TheStreet*, 8 Mar. 2019, <https://www.thestreet.com/technology/what-is-fintech-14885154>.

“*Assumptions of Linear Regression.*” Statistics Solutions, Statistics Solutions, 2019, <https://www.statisticssolutions.com/assumptions-of-linear-regression/>.

“*Techniques for Improving the Performance of SQL Queries under Workspaces in the Data Service Layer.*” IBM Knowledge Center, www.ibm.com/support/knowledgecenter/en/SSZLC2_9.0.0/com.ibm.commerce.developer.doc/refs/rsdperformanceworkspaces.htm.

“*The Consumer Decision Journey.*” McKinsey & Company, www.mckinsey.com/business-functions/marketing-and-sales/our-insights/the-consumer-decision-journey.

“*URL Components Explained.*” URL Components Explained - Tealium Learning Community, 18 Oct. 2016, community.tealiumiq.com/t5/iQ-Tag-Management/URL-Components-Explained/ta-p/5573.

“VC Vestigo Ventures.” Massinvestor Venture Capital and Private Equity Database, <https://massinvestordatabase.com/publicfirm.php?name=Vestigo+Ventures>.

“Vestigo Ventures Closes \$58.9 Million Funding Round.” PR Newswire: Press Release Distribution, Targeting, Monitoring and Marketing, 23 Aug. 2018, <https://www.prnewswire.com/news-releases/vestigo-ventures-closes-58-9-million-funding-round-300701332.html>.

“*Vestigo Ventures Investments.*” CB Insights, <https://www.cbinsights.com/investor/vestigo-ventures-investments>.

Vieira, Armando. “*Predicting Online User Behaviour Using Deep Learning Algorithms.*” 27 May 2016, doi:arXiv.1511.06247.

Wang, Gang, et al. “*Unsupervised Clickstream Clustering for User Behavior Analysis.*” Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI 16, 2016, doi:10.1145/2858036.2858107.

“*What Is SEM? PPC & Paid Search Marketing Explained.*” Search Engine Land, <https://searchengineland.com/guide/what-is-paid-search>.

“*What Is SEO / Search Engine Optimization?*” Search Engine Land, 2019, <https://searchengineland.com/guide/what-is-seo>.

Wooldridge, Jeffrey M. *Introductory Econometrics: a Modern Approach*. Cengage Learning, 2016.

Yeung, Ka Yee and Walter L. Ruzzo. “*Details of the Adjusted Rand index and Clustering algorithms Supplement to the paper*” An empirical study on Principal Component Analysis for clustering gene expression data ” (to appear in Bioinformatics).” 2001.

Zider, Bob. “How Venture Capital Works.” *Harvard Business Review*, 1 Aug. 2014, <https://hbr.org/1998/11/how-venture-capital-works>.

Appendices

B. A Hitchhiker's Guide to the Clustering Galaxy

Introduction

Using data mining techniques, people can process large volumes of data to understand underlying patterns and trends present in datasets. Clustering is an example of an *unsupervised* learning technique; it aims to find natural groupings of elements in a dataset without knowing how they should be grouped. *Supervised* algorithms, on the other hand, use labeled data to learn how to best group objects into pre-determined classes (Akman et al, 2019).

The objective of clustering is to group similar objects in the same cluster while keeping dissimilar objects in different clusters. To compare elements in a dataset, there must be a precise way to define closeness or similarity between elements (Xue and Tian, 2015). For numerical data, we can use a distance metric to define similarity between datapoints. Datapoints that are alike have small distances between them, while dissimilar datapoints are far away from each other.

In the following sections, we illustrate the concept of clustering through a straightforward numerical example on the two-dimensional Cartesian plane. We then discuss different clustering methods, evaluation techniques, and clustering applied to other datatypes, providing a comprehensive introduction to this popular data mining technique.

Introduction and Motivational Example

Figure 1 illustrates 75 points on an X-Y plane. Observing the distances between points, it is easy to see that the data is distributed into five groups, or clusters. After choosing five arbitrary, spaced out center points, 15 points were generated within a fixed radius around each of these central points.

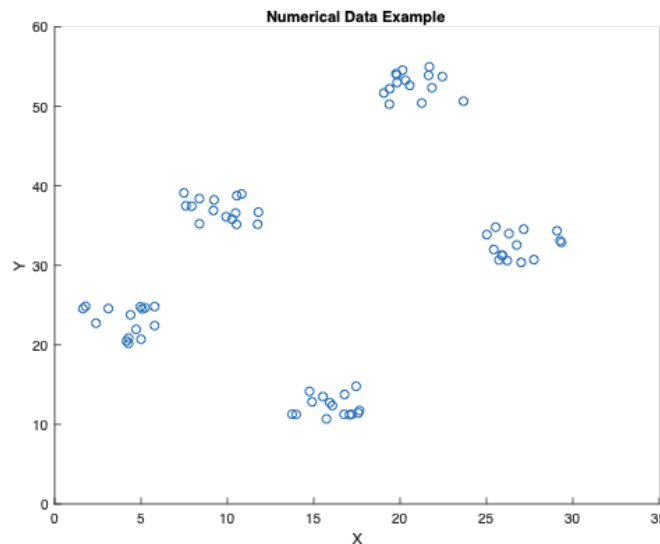


Figure 1: Simple numerical example with five well-separated clusters

Mathematically, there are several distance metrics that can be used to compare the closeness between two points x_i and x_j in n -dimensional space. For simplicity, we will use *Euclidean distance* or straight-line distance. In n dimensions, the Euclidean distance D_{ij} between two points x_i and x_j is defined as:

$$D_{ij} = \left(\sum_{v=1}^n (x_{vi} - x_{vj})^2 \right)^{1/2}$$

where v is one of n dimensions of the points x_i and x_j . In our example, there are only two dimensions to consider (Halabisky, 2012). If $x_i = (a_i, b_i)$ and $x_j = (a_j, b_j)$, then the pairwise distance between them is defined as:

$$D_{ij} = \left(\sum_{v=1}^2 (x_{vi} - x_{vj})^2 \right)^{1/2} = \sqrt{(a_i - a_j)^2 + (b_i - b_j)^2}$$

Applying k-Means Clustering

Now that we have chosen a distance metric (Euclidean distance), there are many different algorithms we can use to identify the natural groups present in the data illustrated in Figure 1. A common approach for clustering numeric data is the *k-Means algorithm*. k-Means is a partitioning algorithm which groups data by computing *centroids* or center points for each cluster of data points (Xue and Tian, 2015). A cluster centroid is simply the average of the points in a cluster. Given a distance metric and a specified number of clusters k , the algorithm iteratively recomputes centroids to minimize the sum of squared distances between each cluster's center and the data points in the cluster. This ensures data points in close proximity to one another are grouped into the same cluster.

To define the *k-Means* algorithm more precisely, let $X = \{x_1, x_2, x_3, \dots, x_N\}$ represent the set of N datapoints to cluster, and let $M^t = \{\mu_1^t, \mu_2^t, \mu_3^t, \dots, \mu_k^t\}$ represent the k cluster centroids computed at each timestep t of the k-Means algorithm (Bonaccorso 2018). In Figure 1, $N = 75$, while $k = 5$. Initially, choose k random points, not necessarily within the dataset X , as cluster centroids. Using Euclidean distance as our distance metric, for each timestep t of the algorithm:

1. Separate each datapoint x_i into one of the k clusters: place x_i into cluster C_j whose centroid has the smallest Euclidean distance to x_i :

$$C^t(x_i) = \operatorname{argmin}_j d(x_i, \mu_j^t),$$

where $i = 1 \dots N$ and $j = 1 \dots k$.

2. For $j = 1 \dots k$, updated cluster centroid μ_j^t by computing the mean point in each new cluster:

$$\mu_j^{t+1} = \frac{1}{N_{C_j}} \sum_{z \in C_j} x_z$$

where N_{C_j} represents the number of datapoints in Cluster C_j .

3. Repeat Steps 1-2 until no data points are reassigned to new clusters. This algorithm is also known as *Lloyd's Algorithm* (Bonaccorso 2018).

k-Means: Important Considerations

k-Means is computationally inexpensive (Rodriguez et al 2019), and it performs well for tasks such as anomaly detection and data segmentation. However, the algorithm is sensitive to initial conditions. Clustering performance depends on the number of clusters specified, the distance metric used, as well as the initial choices for centroid clusters (Singh et al, 2013). Thus, k-Means results may be difficult to reproduce. Moreover, k-Means is not recommended for datasets with outliers, high dimensionality, or in which clusters vary greatly in size and density (k-Means Advantages & Disadvantages, 2020). *k-Medoids* and *k-Means++* are slight modifications of the k-Means algorithm which optimize the choice of initial centroids. Points within the dataset are used as initial centroids, improving the algorithm's computational efficiency and final cluster quality (MATLAB Documentation: kmeans).

Figure 2 shows the result of k-Means clustering applied to the data in Figure 1 using *MATLAB*, with $k = 5$ clusters specified. The algorithm successfully separates the points into distinct clusters, as indicated by the cluster colors. The centroids (mean points of each cluster) are indicated with a black \times . To optimize the algorithm, *MATLAB* employs the *k-Means++* technique to choose cluster centers and uses squared Euclidean distance (similar to Euclidean distance but without a square root) for the distance metric (MATLAB Documentation: kmeans).

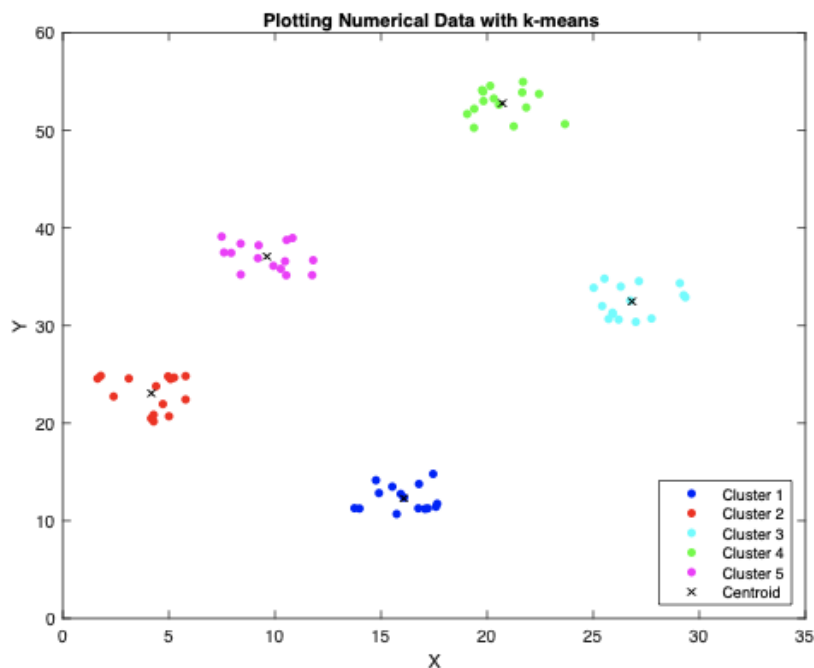


Figure 2: *k-Means* clustering with *MATLAB*

Hierarchical Clustering: A Different Kind of Approach

k-Means is an example of a *partitional* clustering method, grouping datapoints into non-overlapping subsets. Each object is in exactly one subset. *Hierarchical* approaches use similarity metrics to produce *nested* subsets of objects, so that each cluster of objects can contain a sub-

cluster of objects within it (Tan et al, 2008). Common similarity metrics used for hierarchical clustering include single linkage, complete linkage, and average linkage:

- Single linkage defines the distance between two clusters as the distance between the two closest members
- Complete linkage computes the distance between the two farthest members.
- Average linkage forms sub-clusters using the average distance between all members.

The pairwise distance between points can be computed using Euclidean distance.

There are two types of hierarchical methods (Akman et al, 2019), agglomerative and divisive clustering. Agglomerative clustering is a “bottom-up” method that starts with each data point as its own set (cluster). At each time step, sets (sub-clusters) are merged based on a similarity metric. The algorithm terminates when all points are merged into one set, or super-cluster. The divisive technique is a “top down” approach that begins with all data points in the same set. The algorithm repeatedly splits subsets of points until each data point is in its own set (sub-cluster).

A *dendrogram* (Akman et al, 2019) is a diagram that visualizes the hierarchy of sets generated by the hierarchical clustering approach. Figure 4 shows a dendrogram generated in MATLAB by using single linkage with agglomerative clustering, although divisive clustering would generate the same plot. The vertical axis shows the distance between clusters, while the horizontal axis shows the data points corresponding to each leaf on the dendrogram. Each leaf corresponds to several datapoints. Note that hierarchical clustering can be viewed as a sequence of partitional clusterings. Cutting the tree at a certain height produces a partitional clustering of the datapoints (Tan et al, 2008). Cutting the dendrogram in Figure 4 at $y = 3$ yields five clusters of points; for any cluster, the distance between each datapoint and the cluster center, d , satisfies $d \leq 3$ (Ryan Tibshirani 2013). At this height, we ensure that intra-cluster similarity is high since the Euclidean distance between each datapoint is at most 3.

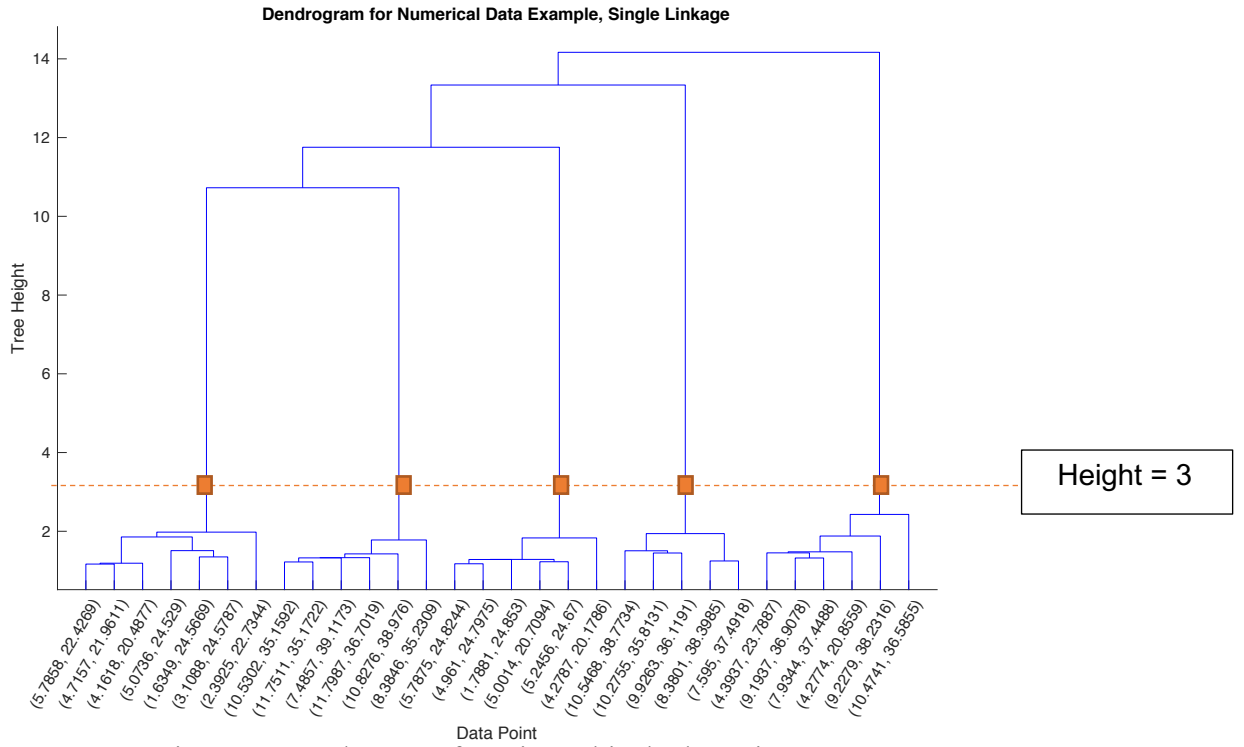


Figure 4: Dendrogram for Hierarchical Clustering

Choosing the Optimal Number of Clusters, k

k-Means clustering requires *a priori* knowledge of k , the number of clusters, to group data points into. However, when working with new, large datasets, the number of clusters may not be known. What is the optimal k , and how do we choose it?

An optimal k for k-Means clustering is one that generates clusters which satisfy two important criteria. One criterion to consider is the within-cluster variation, W , generated by using k clusters (Ryan Tibshirani 2013). We would like similar objects to be in the same cluster; thus, the distances (variation) between objects in a cluster should be small. Mathematically, W is the *Sum of Squared Errors (SSE)* between all data points x_z and their respective cluster centroids μ_j :

$$W = SSE = \sum_{j=1}^k \sum_{x_z \in C_j} \|x_z - \mu_j\|^2$$

where C_j represents the j th cluster, μ_j is the centroid for that cluster, and x_z is a datapoint in that same cluster.

Ideally, we would like to choose a k that not only minimizes within-cluster variation, but it also ensures that similar data points are not grouped in different clusters. In other words, we would like to ensure that between-cluster variation, B , is high while W is low. Between-cluster variation is defined as follows:

$$B = \sum_{j=1}^k N_{C_j} \|\bar{x} - \mu_j\|^2$$

where N_{C_j} is the number of data points in the j th cluster, μ_j is the centroid of the j th cluster, and \bar{x} is the mean data point of the entire dataset:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

A relatively common, although imprecise, method that considers both B and W to approximate the optimal k is the “elbow method.” After plotting SSE against increasing k values, the ideal k is found at the “elbow” of the plot. The “elbow” is not precisely defined. It is the point at which the SSE drops most as the number of clusters increases from $k - 1$ to k (Dangeti, 2017). At this k , there is an elbow-like point on the graph. Thus, the elbow method is a simple heuristic; it may be used as a quick, but possibly inaccurate, approximation when clustering a small dataset.

Figure 5 illustrates a plot generated by computing the SSE for values of k ranging from $k = 1$ clusters to an arbitrary maximum of $k = 10$ clusters; each k value is plotted against its SSE. As the number of clusters increases, the SSE (and thus the value of W) decreases, indicating that our within-cluster variation decreases. We cannot choose a k which minimizes W alone, because W is smallest when each data point is in its own cluster. Thus, we choose a k at the “elbow” of the graph. We know the optimal k is $k = 5$. Looking at the graph, however, it is unclear if we should choose $k = 2$, instead (the SSE decreases most rapidly between $k = 1$ and $k = 2$, forming a sharp elbow-like corner on the graph). The elbow method is thus a “quick and dirty” approach to finding the optimal k , and as Figure 5 illustrates, it may not be very reliable.

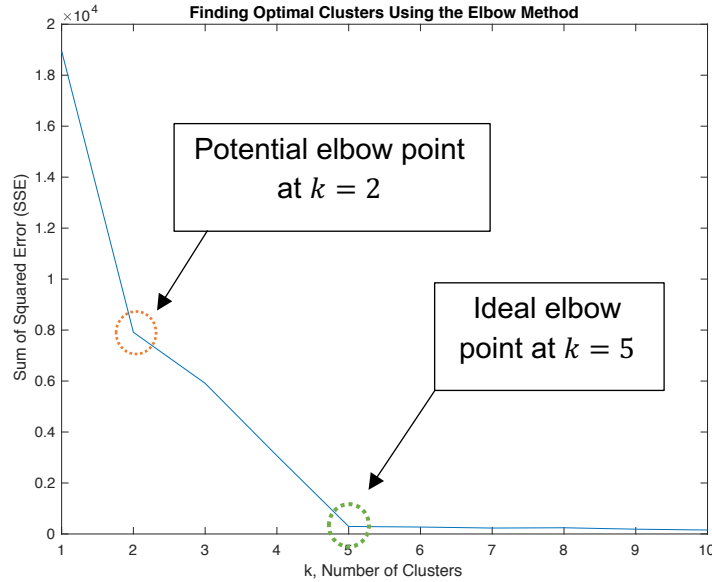


Figure 5: Plotting SSE against k values in the elbow method.

There are more precise approaches for identifying k , namely the Calinski and Harabasz (CH) Index and the Gap statistic. The CH Index (Calinski & Harabasz, 2007) is the same as the F-statistic or F-ratio. applied to cluster analysis, measuring the ratio of between-cluster variance to within-cluster variance:

$$CH(k) = \frac{\frac{B(k)}{k-1}}{\frac{W(k)}{N-k}}$$

The numerator, $\frac{B(k)}{k-1}$ is simply the between-cluster variance divided by its *degrees of freedom*, the number of independent values which are free to vary when computing $B(k)$. Degrees of freedom are computed as *Sample Size* – 1. To visualize this, imagine four students choosing one of four differently colored pens. The last student's pen choice depends on the choices of the three previous students, because there is only pen left to choose. Thus, we can vary how the first three students choose a colored pen, and the degrees of freedom would be three (Degrees of Freedom). Since $B(k)$ involves computing the distances between k centroids and the overall dataset mean, the degrees of freedom are $k - 1$. Similarly, the denominator, $\frac{W(k)}{N-k}$ is the within-cluster variation divided by its degrees of freedom. To compute $W(k)$, we consider each cluster's points independently; thus, the total degrees of freedom is the sum of the degrees of freedom of each cluster. There are k clusters, so the overall degrees of freedom are $N - k$, where N is the total number of points in our sample (Stats: One-Way ANOVA, 1996).

If the datapoints do not fall in natural clusters and are instead distributed equally, we would expect the between-cluster variation to be similar to the within-cluster variation. In this case, the CH index is 1. Otherwise, we would like to find the optimal k which maximizes B while minimizing W . After choosing a maximal number of clusters to consider, K , the CH index computes B and W for values of k from $k = 2$ to $k = K$ (note that we CH Index is undefined at $k=1$). The optimal cluster number is the $k = x$ that maximizes $CH(k)$. Figure 6 plots the CH Index against different k values from 2 to 10. Clearly, the index is highest when $k = 5$, which equals the number of clusters in our dataset.

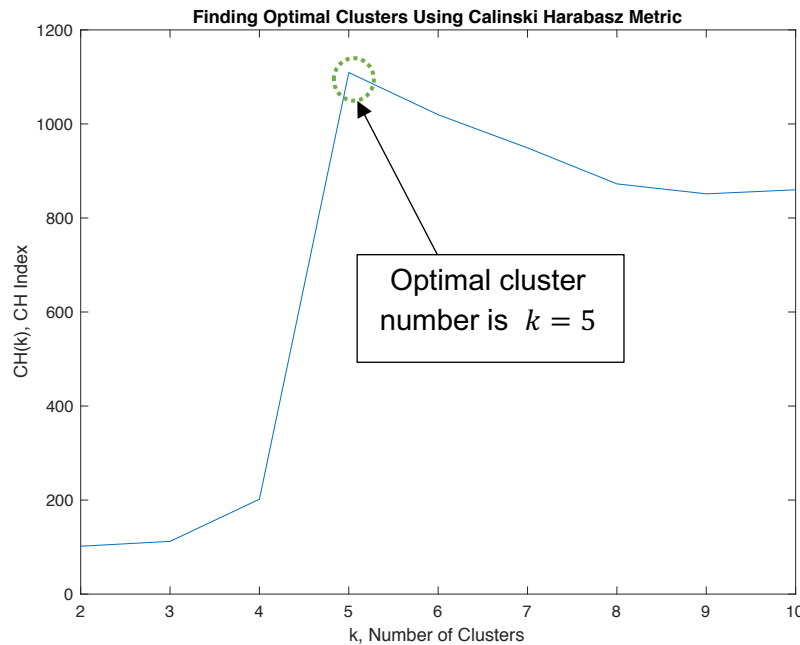


Figure 6: Plotting Calinski Harabasz Index against k

One of the most popular techniques for determining the optimal cluster number is the Gap statistic. Outperforming many other techniques include the CH Index (Mohajer et al, 2011), the Gap statistic was developed to formalize the heuristic “elbow” using statistical concepts (Tibshirani et al, 2000). Instead of considering only raw within-cluster variation W_k , or the SSE , the Gap statistic compares the $\log_e(W_k)$ with the expected value of the $\log_e(W_k)$ of a dataset of points sampled from a uniform distribution on $[0,1]$. The uniform distribution is also called the *reference distribution*. For a given value of k the Gap statistic is computed as:

$$Gap(k) = E_n^*\{\log(W_k)\} - \log(W_k(K))$$

where $E_n^*\{\log(W_k)\}$ is the expected (average) value of the reference distribution. $E_n^*\{\log(W_k)\}$ is obtained by averaging the results of randomly sampling points from the reference distribution using Monte Carlo simulations. Why are logarithms used? Using logarithms was an empirical choice; logarithms are typically used in statistical analysis to make likelihood computations easier (Mohajer et al, 2011).

The smallest k which maximizes the gap between the expected $\log(W_k)$ and the measured $\log(W_k)$ is the optimal one. Formally, this occurs when

$$Gap(k) \geq Gap(k + 1) - s_{k+1}$$

where s_{k+1} is the simulation error resulting from consecutive Monte Carlo simulations. At this point, the measured $\log(W_k)$ is farthest below its expected value.

Figure 7 illustrates a plot of Gap statistics computed at k values from 1 to 10. The plot in Figure 7 peaks at $k = 5$, clearly indicating that the optimal k for our dataset is $k = 5$ (Ryan Tibshirani 2013).

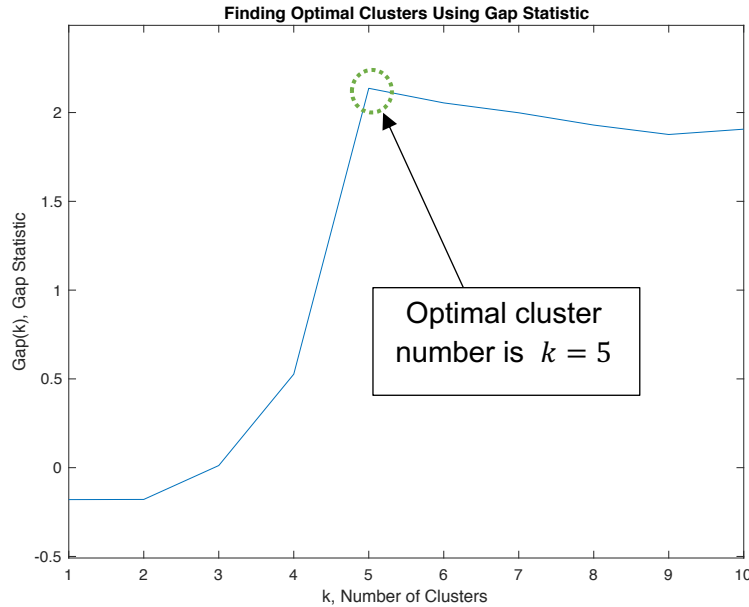


Figure 7: Plotting Gap statistic against k

Figure 8 illustrates the intuition behind the Gap statistic. The expected $\log(W_k)$ of the uniform distribution is shown in red, and the measured $\log(W_k)$ is in blue. We can see that between $k = 4$ and $k = 5$, the measured $\log(W_k)$ decreases rapidly and falls far below the $\log(W_k)$ of the reference distribution. At $k = 5$, the gap between the two distributions is maximized. For $k > 5$, the $\log(W_k)$ begins to decrease at a slower rate than that of the expected distribution because unnecessary clusters are added (Tibshirani et al, 2000). Thus, the ideal cluster number is $k = 5$, as we expect.

Beyond Euclidean Distance

Although we focused on Euclidean distance in our example, any numerical distance metric may be used for clustering. One common distance metric is the *Manhattan* or taxicab distance (Craw 2011). Mathematically, the Manhattan distance between two points x_i and x_j in n -dimensional space is:

$$D_{ij} = \sum_{v=1}^n |x_{vi} - x_{vj}|$$

where $|x_{vi} - x_{vj}|$ is the distance between x_i and x_j in the v th dimension. The distance is called the *Manhattan* distance in reference to Manhattan, New York, where streets are laid in a grid at right angles to each other. Thus, the *Manhattan* distance reflects how far a car would have to drive to get from point A and point B on the grid.

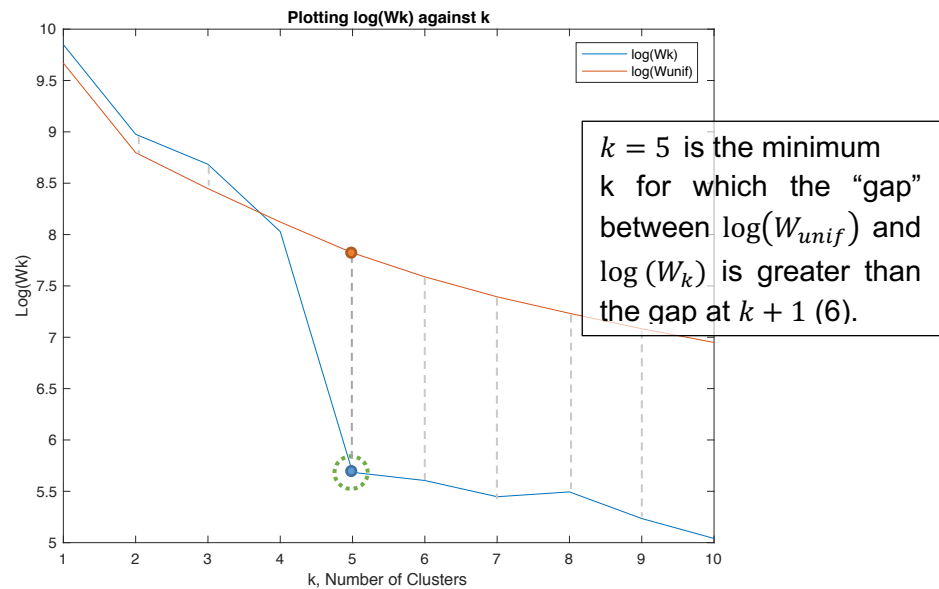


Figure 8: Intuition behind the Gap statistic

Beyond Exclusive (Hard) Clustering

In our simple numerical example, each datapoint belongs to only one cluster. However, *exclusive* or hard clustering may not suit messy, real-world datasets with outliers or complex patterns.

Overlapping or soft clustering (Tan et al, 2008) allows a single datapoint to belong to multiple clusters and is often used for analyzing trends in financial, medical, and scientific datasets.

The *fuzzy c-Means* (FCM) technique (Akman et al, 2019) is a popular extension to the k-Means algorithm, and it often outperforms k-Means clustering on real-world datasets. FCM uses *fuzzy logic* to produce overlapping clusters. Each object has a particular weight ranging between $[0, 1]$ of belonging to a cluster; the sum of an object's membership weights must sum to 1. Similarly to the k-Means algorithm, each iteration of FCM computes optimal cluster centroids. However, each data point has a membership weight of belonging to each centroid; a smaller distance to a centroid yields a higher membership weight. Each centroid is the mean of all points weighted by their membership weights.

Beyond Numeric Data

Distance is clearly defined for numerical data; however, it is more difficult to define the distance between categorical data or text data. In a categorical dataset, each datapoint is described by a set of attributes; the values within each attribute are not inherently comparable (for example, object types). Text data includes strings and text documents. For both data types, the distance metric is problem-dependent, and it should correspond to how people would intuitively group elements in a dataset (Andritsos et al, 2017). After defining and then computing distances, any hard or soft clustering algorithm may be used to find natural groupings and trends in the data.

In a categorical dataset, the distance between points may be represented with the number of overlapping attributes. More overlap in attributes indicates greater similarity between datapoints. Instead of comparing the value of each attribute or character in a string, context-based methods identify groups of attribute values or words that appear together, or “contexts”. In a categorical dataset, we can use contexts to compare values of other attributes not in the context; attributes are similar if they appear in similar contexts (Andritsos et al, 2017).

For text data, we can compare two strings at the character level or at the word (sequence) level. A simple character-based distance is Levenshtein or edit distance (Cohen et al, 2003). The edit distance between two strings is the total number of additions, substitutions, and deletions necessary to transform the first string into the second. A smaller distance indicates more similarity. Character-based distances are best for comparing short strings, such as first and last names. For longer string sequences, such as sentences or phrases, it is more meaningful to compare the words or *tokens* that appear in two strings (Cohen et al, 2003). Strings with common words or groups of words (also called *n-grams*) are more similar to one another.

Extrinsic Techniques for Evaluating Clustering Performance

Although clustering is typically unsupervised, with correct classes unknown, we can sometimes intuitively cluster data to produce ground truth (correct) labels for evaluation. By comparing the outputs of clustering algorithms to the groupings we expect, we can choose the algorithm that performs best for our dataset.

Several scoring functions can be used to assess algorithm clustering performance based on the ground truth labels, including Homogeneity, Completeness, V-Measure (probability-based

methods), and Adjusted Rand Index (a type of pairwise counting method). In the sections below, *clusters* are groups predicted by a clustering algorithm, while ground-truth *classes* are the desired groupings that algorithms are evaluated against.

Evaluating Clusters based on Conditional Entropy (Class Distributions)

Suppose we have N datapoints, each assigned to one of G ground truth classes. Our clustering algorithm will assign each datapoint to one of K predicted clusters. Note that the number of classes may not equal the number of clusters. We will use n_{gk} to indicate the number of datapoints in class g assigned to cluster k (Rosenberg and Hirschberg, 2007).

In each predicted cluster, we would like the class distribution to be skewed to one class (clustered datapoints to belong to a single ground truth class). Likewise, given a class, we would like all points to belong to the same cluster. First, consider the overall class distribution in our dataset, given by the *entropy* (measure of variation) across all of our classes:

$$H(G) = - \sum_{g=1}^G \frac{n_g}{N} * \log \left(\frac{n_g}{N} \right)$$

where n_g is the total number of datapoints in class g and N is the total number of datapoints. Similarly, the overall distribution of datapoints across clusters can be expressed as:

$$H(K) = - \sum_{k=1}^K \frac{n_k}{N} * \log \left(\frac{n_k}{N} \right)$$

To compute the distribution of classes in a given cluster, we can use *conditional entropy*:

$$H(G|K) = - \sum_{k=1}^K \sum_{g=1}^G \frac{n_{gk}}{N} \log \left(\frac{n_{gk}}{n_k} \right)$$

Where n_{gk} is the number of datapoints of class g in cluster k , n_k is the number of datapoints in cluster k , and $\frac{n_{gk}}{N}$ weighs the term proportionally to the total number of points. Mathematically, $\frac{n_{gk}}{N}$ is the probability of a datapoint of class g belonging to cluster k , while $\frac{n_{gk}}{n_k}$ is the probability that a datapoint in cluster k will be of ground truth class g . If all points in the cluster are in the same ground truth class, $\frac{n_{gk}}{n_k}$ is 1 and $\log \left(\frac{n_{gk}}{n_k} \right) = 0$.

If each cluster contains only datapoints of a single class, $H(G|K) = 0$ and we have a perfectly *homogenous* clustering. *Homogeneity*, the measure of uniformity within a cluster, is computed as follows:

$$h = \begin{cases} 1 & \text{if } H(G) = 0 \\ 1 - \frac{H(G|K)}{H(G)} & \text{otherwise} \end{cases}$$

When $H(G) = 0$, we have only one class, so homogeneity is defined to be 1. Otherwise, homogeneity ranges from 0 to 1. Homogeneity is zero when $H(G|K) = H(G)$; this occurs when

we have very “messy” clusters and the class distribution within each cluster matches the overall class distribution.

Grouping each point into its own cluster results in a perfectly homogeneous clustering; however, we would like datapoints of the same class to be in the same cluster.

Completeness is symmetric to homogeneity and assesses whether or not the datapoints within a class are grouped into the same cluster. Completeness is maximized when all datapoints of a class belong in one cluster, and it is computed as follows:

$$c = \begin{cases} 1 & \text{if } H(K) = 0 \\ 1 - \frac{H(K|G)}{H(K)} & \text{otherwise} \end{cases}$$

where $H(K|G)$ represents the spread of each class’s datapoints across all of the predicted clusters. When all datapoints of each class are in the same cluster, $H(K|G) = 0$ and the clustering is perfectly complete. If there is only one predicted cluster, $H(K) = 0$ and completeness is defined to be 1. Similarly to homogeneity, completeness ranges on a scale from 0 to 1.

Ideally, we would like our clusters to be homogenous *and* complete. V-measure is the weighted harmonic mean of homogeneity and completeness:

$$V = \frac{(1 + \beta) * h * c}{(\beta * h) + c}$$

where β is the weighting parameter used to give more importance to homogeneity or completeness. Note that $\beta = 1$ yields the unweighted harmonic mean of homogeneity and completeness:

$$V = \frac{(1 + 1) * h * c}{(1 * h) + c} = \frac{2 * h * c}{h + c} = \frac{2}{\frac{h + c}{h * c}} = \frac{2}{\frac{1}{h} + \frac{1}{c}}$$

In general, we have:

$$V = \frac{(1 + \beta) * h * c}{(\beta * h) + c} = \frac{(1 + \beta)}{\frac{(\beta * h) + c}{hc}} = \frac{(1 + \beta)}{\frac{\beta * h}{hc} + \frac{c}{hc}} = \frac{(1 + \beta)}{\frac{\beta}{c} + \frac{1}{h}}$$

If we would like to give more importance to homogeneity, we would set $\beta < 1$ to decrease the value of $\frac{\beta}{c}$. However, if we would like to weigh completeness more, we would set $\beta > 1$ to increase the value of $\frac{\beta}{c}$.

Homogeneity, completeness, and V-measure can be calculated for any clustering setup, regardless of the dataset size, the number of clusters and classes, and the algorithm used.

Evaluating Clusters Using Pairs of Datapoints

Instead of considering the variability of datapoints within each class and cluster, we can use all possible pairs of datapoints to compare our predicted clusters against the ground truth classes. The predicted clusters *agree* with the ground truth classes when either the two points are clustered together in both groupings or the two points are in different clusters in both groupings. Otherwise,

there is *disagreement*: the points may be clustered together in one grouping, but clustered separately in the other.

One type of clustering evaluation metric based on this idea is the Rand Index (Rand 1971), which simply takes the ratio of agreements to all possible pairs of datapoints:

$$\frac{n_{11} + n_{00}}{\binom{N}{2}}$$

where N is the number of datapoints in our dataset ($N \geq 2$), n_{11} is the number of agreeing pairs in which datapoints are clustered together, n_{00} is the number of agreeing pairs in which datapoints are split into separate clusters, and $\binom{N}{2}$ represents the total number of possible pairs of datapoints.

Rand Index can be viewed as the probability of extracting an agreeing pair from all pairs of datapoints, and it ranges from 0 to 1. However, notice that Rand Index is only zero when the number of agreements, $n_{11} + n_{00}$, is zero (Vinh et al, 2009). This occurs in the extreme case when every datapoint is in its own cluster in one grouping, while in the other grouping, all datapoints are in a single cluster. If we had a random grouping for our ground truth and another random grouping generated by our algorithm, we would like our scoring metric to output a consistent, low value. However, the Rand Index does not have a constant value in the case of random partitions.

To account for chance partitioning, the Adjusted Rand Index (ARI) was developed based on the expected value of $n_{11} + n_{00}$ (Vinh et al, 2009). The adjusted index is expressed as follows:

$$ARI = \frac{Index - Expected\ Index}{Max\ Index - Expected\ Index}$$

where Index is $n_{11} + n_{00}$. The Expected Index is based on the worst-case scenario, occurring when the ground truth labels are completely independent of the predicted clustering.

Define c_{ij} as the number of objects that are in both class i and cluster j . Then the number of pairs of agreeing objects in class i and cluster j is $\binom{c_{ij}}{2}$. Moreover, define the number of objects in class i as a_i and the number of objects in cluster j as b_j . Then the ARI, above, becomes:

$$ARI = \frac{\sum_{i,j} \binom{c_{ij}}{2} - \sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} / \binom{N}{2}}{\frac{1}{2} [\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - \sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} / \binom{N}{2}}$$

where $\sum_{i,j} \binom{c_{ij}}{2} = n_{11} + n_{00}$ (the number of agreeing pairs) and $\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} / \binom{N}{2}$ is the expected (worst) case if each clustering is independent (random).

The maximum value of agreements, $\frac{1}{2} [\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}]$, occurs when we have a perfect clustering: all class objects are grouped into the same cluster.

Thus, when we have two random clustering assignments, the Index equals the Expected Index, and ARI takes on a constant value of 0. In the best case, the Index is at its maximum value, ARI is 1, and we have a perfect clustering.

Although these methods are useful ways to evaluate clustering techniques, they require *labeled* data. V-measure and ARI are also called *external* indices because they evaluate clustering performance using “answers” given by another source. When labels are not available, *internal* indices must be used. Internal indices include Sum of Square Errors and the Calinski-Harabasz Index, which we also used to find the optimal number of clusters.

Conclusion

Clustering is a very active area of research that is continuously evolving, especially as more and more data is collected each day. Illustrating the intuition of clustering through numerical data, we provide a comprehensive overview of methods for optimizing, evaluating, and extending numerical clustering techniques. This appendix provides a foundation for further exploration of clustering algorithms on diverse, real-world datasets.

References (Appendix B)

- Akman, Olcay, et al. "Data Clustering and Self-Organizing Maps in Biology." *Algebraic and Combinatorial Computational Biology*. Academic Press, 2019. 351-374.
- Andritsos P., Tsaparas P. Categorical Data Clustering. In: Sammut C., Webb G.I. (eds) *Encyclopedia of Machine Learning and Data Mining*. Springer, Boston, MA, 2017.
- Bonaccorso, Giuseppe. *Mastering Machine Learning Algorithms: Expert Techniques to Implement Popular Machine Learning Algorithms and Fine-Tune Your Models*. PACKT Publishing, 2018.
- Caliński, Tadeusz, and Jerzy Harabasz. "A dendrite method for cluster analysis." *Communications in Statistics-theory and Methods* 3.1 (1974): 1-27.
- Cohen, William W., Pradeep Ravikumar, and Stephen E. Fienberg. "A Comparison of String Distance Metrics for Name-Matching Tasks." *IIWeb*. Vol. 2003. 2003.
- Craw S. *Manhattan Distance*. In: Sammut C., Webb G.I. (eds) *Encyclopedia of Machine Learning*. Springer, Boston, MA, 2011.
- Dangeti, Pratap. *Statistics for Machine Learning*. PACKT Publishing, 2017.
- "Degrees of Freedom." UT Austin Statistics Online Support, sites.utexas.edu/sos/degreesfreedom/. Accessed 3 February 2020.
- Halabisky, Brian. *Euclidean Distance in 'n'-Dimensional Space*. 2012. Stanford HLab.
- "k-Means Advantages and Disadvantages." Google Developers, developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages. Accessed 29 January 2020.
- "MATLAB Documentation: kmeans." MathWorks Help Center. www.mathworks.com/help/stats/kmeans.html. Accessed 5 February 2020.
- Mohajer, Mojgan, Karl-Hans Englmeier, and Volker J. Schmid. "A comparison of Gap statistic definitions with and without logarithm function." *arXiv preprint arXiv:1103.4767*, 2011.
- Rand, William M. "Objective Criteria for the Evaluation of Clustering Methods." *Journal of the American Statistical Association*, vol. 66, no. 336, 1971, pp. 846–850. JSTOR, www.jstor.org/stable/2284239. Accessed 12 Feb. 2020.
- Rosenberg, Andrew, and Julia Hirschberg. "V-measure: A conditional entropy-based external cluster evaluation measure." *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning*

(EMNLP-CoNLL). 2007.

Rodriguez, Mayra Z., et al. "Clustering algorithms: A comparative approach." PloS one 14.1, 2019.

Singh, Archana, Avantika Yadav, and Ajay Rana. "K-means with Three different Distance Metrics." International Journal of Computer Applications 67.10, 2013.

"Stats: One-Way ANOVA." Math-130: Introduction to Statistics Lecture Notes. University of Richland, 1996. people.richland.edu/james/lecture/m170/ch13-1wy.html

Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. "Cluster analysis: basic concepts and algorithms." Introduction to data mining 8 (2006): 487-568.

Tibshirani, Robert, Guenther Walther, and Trevor Hastie. "Estimating the number of clusters in a data set via the gap statistic." Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63.2 (2001): 411-423.

Tibshirani, Ryan. "Clustering 3: Hierarchical clustering (continued): Choosing the number of clusters," 2013.

Vinh, Nguyen Xuan, Julien Epps, and James Bailey. "Information theoretic measures for clusterings comparison: is a correction for chance necessary?." Proceedings of the 26th annual international conference on machine learning. 2009.

Xu, Dongkuan, and Yingjie Tian. "A comprehensive survey of clustering algorithms." Annals of Data Science 2.2 (2015): 165-193.