Instance Weighting for Partitioning and Graph Based Clustering

Paul Moggridge

University of Hertfordshire

Submitted to the University of Hertfordshire in partial fulfilment of the requirement of the degree of PhD

Supervisors:

Dr. Na Helian

Dr. Yi Sun

Dr. Mariana Lilley

February 2025

Acknowledgements

Firstly, I would like to thank my supervisory team Dr. Na Helian, Dr Yi Sun and Dr Mariana Lilley for supporting me in taking this programme of study. Their care and devotion has always been above and beyond my expectations. Their invaluable advice has transformed my life and for that I am truly grateful.

I would also like to thank to my family. My parents Hayley and Kerry, who have inspired and supported me hugely through my entire journey. Also, my wife, Natalie for her amazing patience, when I have had to devote family-time to my work. Also, my son Finley, who has motivated me to be the best I can be.

Finally, I would like to acknowledge all the people who have been part of this journey, from my grandfather who first introduced me to computers, to my form-tutor at secondary school who first saw potential in me, to the college tutors and support staff who enabled me to continue my studies while recovering from major surgery, to my friends and lecturers who encouraged my studies while at the University of Hertfordshire, and finally my colleagues whose advice kept me motivated throughout.

Abstract

Clustering algorithms often struggle to achieve robust clustering performance on datasets that contain outliers and imbalanced cluster sizes. While k-means and spectral clustering are popular choices, they demonstrate limitations. K-means, a partitioning-based method, while efficient and effective, is sensitive to outliers, leading to inaccurate cluster assignments. Spectral clustering, a graph-based approach, while it is able to model data of arbitrary shape, it performs poorly when clusters have imbalanced instance counts. Traditional instance selection methods address these issues by discarding instances, potentially losing valuable information. This thesis proposes a more nuanced approach by integrating instance weighting into these clustering algorithms. To achieve this, existing literature on instance weighted clustering is reviewed and two novel instance weighted clustering algorithms (LOFIWKM and IWSE) are proposed to demonstrate and evaluate under what conditions instance weighting is effective using both intrinsic and extrinsic clustering accuracy metrics.

LOFIWKM builds on k-means and leverages the Local Outlier Factor measure of outlierness to estimate instance density and adjust centroid placement towards higher density areas. The approach was trialled on synthetic and real-world data to investigate how effective the approach is given different quantities and severities of outliers. The approach is also trialled on a sample of real-world avionics data. My experimental findings demonstrate that LOFIWKM effectively mitigates the influence of outliers, significantly improving clustering performance on datasets with different extents of outlier contamination.

IWSE builds on a spectral clustering ensemble framework and incorporates

density-based weights into the sub-sampling process. The approach is trialled on a variety of synthetic and benchmark datasets to establish when the approach is effective. Then using further synthetic experiments and the MNIST hand written digits dataset the applicability of the method is asserted. My experiments show that IWSE substantially outperforms traditional spectral clustering in terms of clustering performance on datasets with imbalanced clusters that exhibit clear cluster density variations.

Overall, this work contributes novel instance weighting frameworks for partitioning-based and graph-based clustering, offering a robust alternative to instance selection. These approaches are generalizable to other clustering algorithms and can be combined with other techniques, opening new avenues for developing more accurate and robust clustering solutions for challenging real-world data.

Contents

List of Figures

Chapter 1

Introduction

As of 2023, the "Volume of data/information created, captured, copied, and consumed worldwide" is 123 zetabytes and this is projected to more than triple to 394 zetabytes by 2028. Furthermore, as of 2023, 87.9% of the Fortune 1000 and leading global organisations identified "Investments in Data and Analytics are a Top Organizational priority" [1].

To meet this explosion of data, and provide the most sophisticated of the in-demand analytics, is data mining and machine learning. Within machine learning exists unsupervised learning techniques. Most notably data clustering. Through the lens of clustering algorithms, implicit and useful generalisations (groupings) of data can be found. A "good" clustering is defined as one which produces a partitioning of the data which has high intra-cluster similarity and low inter-cluster similarity. As clustering is an unsupervised technique, this does not depend on training with provided labels. Instead, clustering uses

[1] Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2023, with forecasts from 2024 to 2028 (in zettabytes) [Graph], IDC, & Statista, & Various sources, May 31, 2024.
State of data and analytics investment at companies worldwide in 2023 [Graph], Wavestone, & NewVantage Partners, December 25, 2023.

unlabelled data. The partitioning result depends entirely on the data, and the application of some similarity measure within the clustering algorithm. Thus in-practice the quality of clustering results is subjective. Although, a special exception, is semi-supervised clustering, in this case, some constraints (must-link and must-not-link) are provided, akin to labels. However, by comparing against a suggested/intended grouping of the data we can infer clustering performance. At the broadest level clustering algorithms can be grouped into soft and hard. Soft clustering algorithms provide a degree of membership to each cluster for each row of data (instance). While on the other hand, hard clustering algorithms assign instances to one and only one cluster. For an overview of clustering research to date, please refer the seminal surveys of clustering literature [1] [2] and [3]. Furthermore, Aggarwal and Reddy's book titled "Data Clustering" provides a very comprehensive and detailed overview of the topic of clustering [4].

Ezugwu et al. provides a recent survey of state-of-art clustering applications, challenges, and future research prospects [5]. Their systematic literature review found that clustering is useful to a plethora of disciplines: *Web usage, Speech processing, Medical science: Disease onset and progression, Image processing and segmentation, Information retrieval, Aviation and automotive systems, Financial systems and economics, Bioinformatics, Financial systems and economics, Robotics, Text mining, Video surveillance, Marketing, Object recognition and character recognition, Data Mining and Big Data Mining, Dimensionality reduction, Data transfer through network, Urban development, Privacy protection.* As these applications suggest, clustering can be applied to most structured (tabular) and unstructured (multimedia) data.

Since the formal definition of the first clustering algorithms in circa 1950s and 1960s [6, 7], many different types of clustering algorithm have arose. Amongst the earliest are *Partitioning-based* and *Hierarchical-based* clustering. *Partitioning-based* clustering construct various partitions and then evaluates them by some criterion (see Appendix 9.4.2 for a worked example of k-means, a popular partitioning-based algorithm). *Hierarchical-based* clustering creates a hierarchical decomposition of the data using some criterion to partition data points (top-down) or join points (bottom-up), to form clusters. *Density-based* clustering uses density information to associate data points together. *Grid-based* clustering divides the feature space into a number of cells and partitions based on the cells. *Graph-based* clustering constructs a graph representation and partitions it (see Appendix 9.4.3 for a worked example of spectral clustering, a popular graph-based algorithm). Further, this is not an exhaustive list. Despite the huge variety, of the clustering algorithms developed, there remains space for innovation. For example, Ezugwu et al. also highlighted the need for improved, flexible and efficient techniques. My research seeks to advance these techniques.

Arguably the most common approach to increasing the flexibility of clustering algorithms in regard to data quality is instance selection as a preprocessing step. Instance selection (also known as sub-sampling) is a well established technique. It is often used for removing instances that are deemed outliers. Another popular technique is feature weighting (also referred to as "attribute weighting") this is an ongoing area of research. In feature weighting, the features of a dataset are weighted based on various metrics typically related to how much they enhance the accuracy of the main data mining activity. Inspired by instance selection and feature weighting, Instance weighting assigns

a weight to each of the instances in a dataset. To achieve this two parts are required a *weighting scheme* and a *method of application* to integrate the weights into the clustering algorithm or approach. However, choosing the weights, akin clustering itself, is non-trivial.

There are infinitely many ways to assign weights to the instances of a dataset. Instance weighting literature shows some promising results which prompted this work [8, 9, 10]. However, between these promising results there is disagreement on best approach to design the weighting scheme and method of application. Furthermore, there is an opportunity to realise instance weighting a generalised framework, similar to the way feature weighting is understood in current practice and literature. In this context, instance weighting could enhance the applicability of clustering algorithms by overcoming weaknesses in specific clustering algorithms. Enhancing the variety of datasets[2] to which an algorithm can be applied would make it easier to perform clustering, leading towards increased automation.

One interesting area to explore through the lens of instance weighting is outlier accommodation. Outlier accommodation is an appealing problem since from a statistics perspective outlierness can be defined on a scale rather than as boolean property enabling instance weighting. For example, in instance weighting, a lesser weight could be assigned to instances that are noisy or anomalous. A modified clustering algorithm or approach could then utilise these weights such that the lesser weighted instances are less salient in processing. Whilst it is true that some types and severities of outlier should be fully discarded, some types and severity of outliers maybe best partially re-

[2]Consider datasets with: outliers, imbalanced clusters, noise, closely packed clusters, high-dimensionality, high-instance-count, irregular cluster shapes etc.

tained for the clustering process to learn from, especially if data is limited.

Another area for investigation is handling imbalanced data, this is a typical a use case for instance selection techniques (sub-sampling and up-sampling). Similarly to instance selection, instance weighting could be embedded into clustering algorithms and approaches to handle this within the algorithm or approach and enable the discovery of small and large clusters with the need for preprocessing.

In summary, the design of the proposed instance weighting solutions is guided by the following core requirements. Firstly, the solutions must be designed for integration with multiple clustering algorithms. Secondly, the solutions must accommodate outliers by handling up to 10% of the instances as outliers. Here, an outlier is defined as a spatial outlier: an instance that is two standard deviations away from the nearest cluster center and is not density-connected to any cluster. The 10% threshold is based on the average number of outliers typically seen in real-world datasets; for example, in the seminal work by Campos et al., the average percentage of outliers across their list of 23 imbalanced datasets is 7.24% [11]. Thirdly, the solutions must handle a moderate level of cluster imbalance, specifically up to ∼5:1. On the one hand, mild-to-moderate imbalance is common in popular supervised and unsupervised datasets; for example, the Wisconsin Breast Cancer dataset has an imbalance of ∼2:1 [12], and the Bank Marketing dataset has ∼8:1 [13]. On the other hand, addressing extreme imbalances, such as the ∼30:1 seen in the 1200-instance version of the Refuge Glaucoma dataset [14], often requires specialized attention beyond the scope of the general approaches proposed here. Finally, to enhance the variety of datasets to which clustering can be applied and increase automation, the

proposed methods must be robust to slight variations and noise in the dataset. These requirements will directly inform the experimental design of this thesis.

1.1 Aim

The aim of this work is to investigate the integration of instance weighting into different types of clustering algorithms to address data quality issues. The thesis hypothesis is as follows: Instance Weighted Clustering is a valuable tool for increasing clustering performance for data with quality issues.

1.2 Objectives

- Integrate instance weighting into a partitioning-based clustering algorithm.
- Integrate instance weighting into a graph-based clustering algorithm.
- Using synthetic and real-world dataset to simulate data quality issues, in particular, outliers and class imbalance.

1.3 Research Questions

RQ1 How can instance weighting be applied to partitioning-based clustering algorithms for outlier accommodation?

RQ2 How can instance weighting be applied to graph-based clustering algorithms to handle imbalanced data.

RQ3 Under what conditions does instance weighting enhance clustering performance on data characterized by the presence of outliers or class imbalance?

1.4 Contributions to Knowledge

Contribution 1

Linked to RQ1, instance weighting was applied to k-means, a partitioning type algorithm, to enhance the robustness of k-means to outliers. The experimental results found that instance weighting when using a density-based weighting scheme was able to enhance the clustering performance on datasets that contained outliers. Details of this contribution are detailed in Chapters 4 and 5. This important initial finding demonstrates that instance weighting can perform at least well as instance selection.

Contribution 2

Linked to RQ2 instance weighting was applied to bagging-based spectral ensemble. Again, a density-based instance weighting scheme was used. In this algorithm, these weights were used to perturb the sampling of instances within the ensemble and influence the sampling towards better representing the low density / smaller (in terms of instance count) clusters. Experiment results found that the instance weighting when applied to a spectral ensemble enabled smaller clusters to be found. The approach achieves good clustering performance on a variety of datasets especially those with imbalanced clusters (normally very challenging to the spectral clustering algorithm). The approach

also enhances spectral clustering effectiveness for image segmentation. Details of this contribution are described in Chapter 6.

Contribution 3

Linked to RQ3 and continuing on from RQ2. Instance weighted clustering was applied to further imbalanced datasets. To assess limitations and suitability of the instance weighted spectral ensemble approach a series of experiments were conducted. The experimentation isolated the characteristics a dataset should have for my approach to be beneficial. It was found that the approach is most beneficial when datasets include overlapping clusters, moderate to severe imbalance, and contained a variation in density. Some algorithmic and implementation related limitations were identified too, such as the lack of suitability for higher dimensional data. The details of this contribution are described in Chapter 7.

1.4.1 Algorithms Developed

1: Instance weighting for partitioning-based clustering algorithms. Prototype algorithm: Local Outlier Factor Instance Weighted K-Means (LOIWFKM).
2: Instance weighting for an ensemble of graph-based clustering algorithms. Prototype algorithm: Instance Weighted Spectral Ensemble (IWSE).

1.4.2 Thesis Outline

This thesis is organised into eight chapters.

Chapter 2 *Instance Weighted Clustering, a review*, identifies the key themes in literature concerning instance weighted clustering. This chapter identifies gaps, areas to extend, and disagreements in the literature. This chapter motivates and justifies choices made in the following chapters.

Chapter 3 *Methods*, introduces and explains the fundamental clustering techniques which will be extended in the following chapters.

Chapter 4 *Instance Weighting for Partitioning-based Clustering*, describes the proposed LOFIWKM algorithm and demonstrates its ability to accommodate outliers on a simple synthetic dataset, it is shown that LOFIWKM can maintain good clustering performance even when presented with numerous and severe outliers.

Chapter 5 *Instance Weighting for Flight Data Recorder Clustering*, furthers the investigation of LOFIWKM and applies LOFIWKM to a real-world dataset. It is demonstrated that the approach can find clusters in the data despite the addition of artificial outliers. Limitations with the approach are identified and directions for further development are suggested.

Chapter 6 *Instance Weighting for Ensemble Graph-based Clustering*, describes the IWSE approach and demonstrates its effectiveness on a variety of synthetic, image and benchmark datasets. It is shown that instance weighting integrates well with bagging based clustering ensemble.

Chapter 7 *Instance Weighting Clustering for Character Clustering*, further investigates the IWSE approach by trialling on real-world to identify the approaches strengths and weaknesses. This chapter analyses and evaluates the IWSE approach and finds can be highly beneficial to clustering performance, when the right conditions exist.

Chapter 8 *Final Conclusions and Future Work*, summarises the work. It is highlighted how that approaches could be improved and eight areas for future work are identified.

Chapter 2

Instance Weighted Clustering, a review

2.1 Planning

2.1.1 Research Protocol

In this chapter, I conduct a semi-systematic literature review. A semi-systematic approach is adopted as this best suits the challenge presented by the research topic. The research topic of "instance weighting" contains the challenge that "instance weighting" approaches are not always clearly signposted and many different aspects of cluster analysis can be "weighted". Additionally, "instances" (rows of data) are referred to in many different ways based on the application. For example, a study applying clustering to image data, might describe their approach as "pixel" weighted. Utilising a semi-structured approach enables well-labelled studies to be found while allowing literature informally identified to be scrutinised too.

2.1.2 Research Questions

The purpose of the semi-systematic literature review is to establish the position of existing literature on the research questions posed in this study. To answer the research questions the terms mentioned in the questions must first be well-defined.

Partitioning-based clustering, clustering algorithms which partition the data by iteratively updating a model of partitions. (typically using "centroids").
Graph-based clustering, clustering algorithms which partition a graph representation of the data.
Instance weighting, values are assigned to instances which inform the clustering process (in some contexts this is called "boosting").
Outlier accommodation, handling outliers in the clustering process rather then removing them prior to the clustering process.
Imbalanced data, data containing multiple distributions of unequal-cardinality.
Clustering performance, broadly speaking the "accuracy" of the clustering. How meaningful are the clusters? This can be mathematically defined in various ways.
Clustering process, to mean either a clustering algorithm or ensemble of clustering algorithms.

2.1.3 Search Strategy

To identify literature the IEEE Xplore database was utilised along with the search string shown below.

((((”Document Title”:”clustering” OR ”Document Title”:”cluster”) AND (”Document Title”:”partitioning” OR ”Document Title”:”graph” OR ”Document Title”:”boosting” OR ”Document Title”:”boosted” OR ”Document Title”:”spectral”)) OR ”Document Title”:”k-means” OR ”Document Title”:”k-medoids” OR ”Document Title”:”c-means”) AND (((”sample” OR ”instance” OR ”data” OR ”importance” OR ”sampling”) NEAR/10 (”weighted” OR ”weighting”)) OR ”boosting”) AND (”robust” OR ”robustness” OR ”noise” OR ”outlier” OR ”outliers” OR ”outlying” OR ”imbalance” OR ”imbalanced”))

The search string makes the most the of “Command Search” facility. Keywords and their synonyms are organised using the syntax of “AND”, “OR” and “NEAR”. The search string also utilises the ability of search engine to search particular fields, specifically the “Document Title”. The search string has 26 terms utilising the complexity limit imposed by the Command Search feature.

The search string can be broken down into three parts for easy interpretation:

In the “Clustering” part (red): The “Document Title” should contain clustering with the terms “partitioning”, “graph”, “boosting” or “boosted”. Alternatively, the title should mention one of commonly used partitioning or graph clustering algorithms such as “k-means” or “spectral”. The decision was made to ensure the document title included these elements as clustering is the focus of this research. Many studies may use clustering - but do not focus on it. Therefore, it was necessary to preclude their inclusion in the results.

In the “Instance weighting” part (green): Searching anywhere in the docu-

ment. The term “instance” (or a synonym of) must be within 10 words (distance) of “weighting”. Alternatively, the text may mention “boosting” (note “boosted” was not included since the word “boosted” is commonly used descriptively). This part of the search aims to identify well labelled instances of instance weighting. A limitation of this part of the search string is not being able search for all the words that could be used to denote “instance” – as mentioned earlier, the words used to mean “instance” can depend on the application (“customer”, “flight”, “pixel”, “patient”...the list would be extensive).

For the robustness part (blue): Again searching anywhere in the document, the document must contain “robustness”, “noise”, “outliers”, or a synonym of, one of these should be mentioned within the paper to indicate that the paper addresses data quality challenges relating to the research questions.

The keyword NOT was not used, although its application was considered to eliminate documents discussing more common[1] techniques such as “classification” or “feature weighting”. However, clustering is sometimes a preprocessing step for classification (hence classification may be mentioned). Similarly, “feature weighting” is not exclusive of “instance weighting”, both can be used together, and thus may appear together. Henceforth, the decision was made to not utilise “NOT”.

[1] ∼362k papers for the term “classification” versus ∼207k for “clustering” IEEE Xplore (Aug 2024).
∼7k papers for the term “instance weighting” versus ∼70k for “feature weighting” IEEE Xplore (Aug 2024).

2.1.4 Inclusion Criteria

To filter the studies selected by the search string, the following inclusion criteria were produced based on the research questions:

I1 Focuses on partitioning-based or graph-based data clustering algorithms.

I2 Proposes an approach to apply weights to instances to inform the clustering.

I3 Proposed an approach which is robust to noise, outliers or imbalance.

I4 Findings are empirically proven with their own experimental results using intrinsic or extrinsic clustering quality metrics comparing against a traditional (non-weighted) algorithm.

I5 Peer reviewed conference and journal proceedings only.

2.1.5 Exclusion Criteria

E1 Studies not published in English.

E2 Studies published earlier than 2001.

E3 Studies in which the proposed method is not explained, in natural language AND in either, mathematical formulae OR pseudocode OR software.

E4 Studies which exclusively use datasets which are not publicly accessible.

The year 2001 was chosen as by this date the major traditional techniques were established. For example, spectral clustering was developed in 2000. Since this review is concerned with technical details, E3 and E4 were stated to ensure the method described is understandable, workable and repeatable.

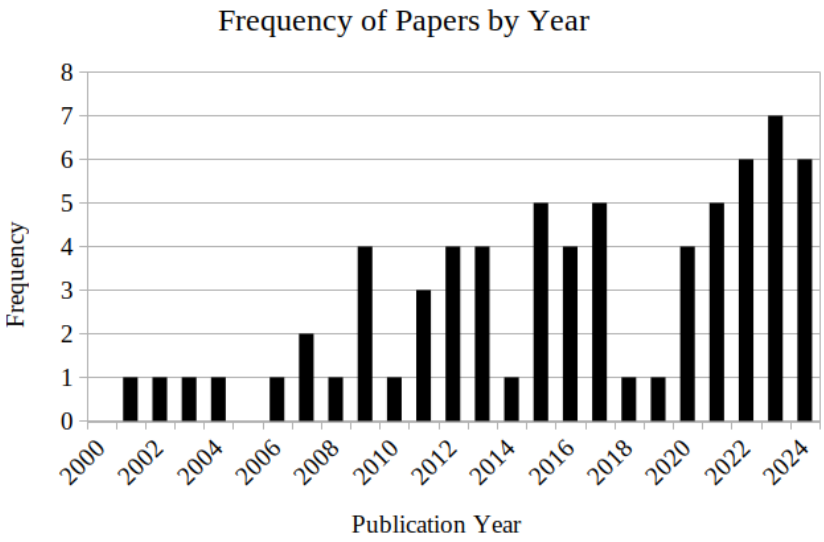


Figure 2.1: Count of papers per year.

2.2 Conducting the Search

The search string selected 60 documents within the IEEE Xplore database. A further 8 papers were added which were the inspiration for this study, or were snowballed from those papers. Figure 2.1 shows the breakdown by year. The full list of documents (including filtering status) can be found Table 9.1 in the Appendix.

The frequency distribution shows an increasing trend of papers on the topic of instance weighting. However, this could be due to the wider increase in Data Science related papers since circa 2010.

Figure 2.3 provides a overview of the filtering process. Filtering was broken down into two stages, screening and eligibility. In the screening stage the inclusion and exclude criteria were applied to the title and abstracts of the documents. The inclusion and exclusion criteria were applied in the order

Figure 2.2: A word cloud of the keywords from the papers.

they are listed. Once a paper is rejected by an inclusion or exclusion criteria it is no longer considered.

The screening and eligibility stages were effective in filtering the papers. In the screening stage some 7 papers were rejected by I1, as they focused on boosting for classification not clustering. These papers were selected by the search as they mentioned clustering in their titles, which had been used typically to inform the weights for boosting classification. A further 14 papers were rejected by I2. Typically, these documents weighted some other aspect clustering, such as the features, views or clusters, rather than the instances. The screening stage followed the principle that if the title and abstract were ambiguous against the inclusion and exclusion criteria, the document was "passed" allowing a more thorough full-text evaluation to take place in the eligibility stage.

In the eligibility screening stage, the full-text of the 47 papers were down-

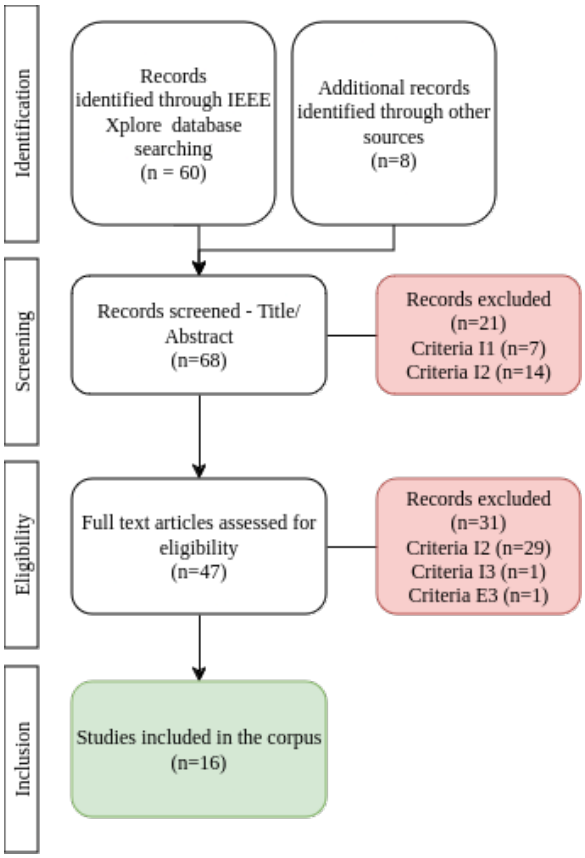


Figure 2.3: PRISMA diagram of filtering process.

loaded and assessed against the inclusion and exclusion criteria. At this point a further 29 papers were rejected by I2, one by I3, and one paper was rejected by E3 due to its brevity.

2.3 Analysis

In the analysis stage, data extraction was completed, a summary of which is shown in Table 9.4. Then codes were developed from the literature, see Table 9.2. Then theme development was completed by organising the codes into a total of 5 themes, see Table 9.3. These themes are summarised and discussed below, leading to the conclusion. See the Appendix for these tables.

Figure 2.4 provides categorisation of the literature in terms of: Type of Clustering Algorithm → Weighting Strategy → Method of Applying the Weights → Data Quality Issue Addressed. The literature found vastly more partitioning-based clustering than graph-based clustering, despite the search string including terms for both. In Figure 2.4 at the root level, all apart from two of the works are included in the light green partitioning-based area (top). No literature was identified that utilised an entirely graph-based clustering approach. The next branching level of Figure 2.4 separates the literature in terms of weighting strategy, this refers to the general philosophy for assigning the instance weights. It can be seen that more than half of the literature found uses a weighting strategy that increases the weight on the archetypal (most inlying) instances. Three works explored using a prototypical (most outlying) weighting strategy and two works explored increasing the weight on both archetypal and prototypical instances, these are labelled "Complex (Both)" in Figure 2.4.

One work used angular information, assigning weight to instances based on their directionality relative to the centroids. Also, one further work, using both Partitioning-and Graph-based clustering assigned weights based on an ensemble process, where more weight is given to instances that are disagreed upon within the ensemble of clusters. The next branching level in Figure 2.4 is the method of applying the instance weights. By far the most common approach was to use the instance weights in the "Centroid Update" phase of the clustering algorithm applied. This step is when centroid positions are calculated based on instances assigned to them. A total of 7 out of the 16 papers utilised this approach. Conversely, 2 papers used the weights in the cluster assignment step. The "Cluster Assignment" step refers to the step where distances are calculated in order to assign instances to clusters. A further 4 papers utilised the instance weights both in the centroid update step and the cluster assignment step. One paper used the weights for neither the update step or the assignment step and instead, used them to inform a merge probability. Finally, at the highest level of the branching in Figure 2.4, it is shown that the most commonly addressed data quality issue is noise, with 7 out of the 16 papers tackling this data quality issue. To much lesser degree, are the logically adjacent issues of outliers and imbalance addressed. Furthermore, on the issue of outliers and imbalance, by observing the nesting in Figure 2.4, it can be seen that increasing the weight on archetypal instances is most commonly used to address data with imbalance and outliers. While increasing the weight on prototypical instances is never used for these data quality issues. This makes logical sense, since increasing the weight on prototypical instances (in essence - outliers) would likely worsen their effect in most cases.

Figure 2.4: A treemap classifying the literature in terms of: Type of Clustering Algorithm → Weighting Strategy → Method of Applying the Weights → Data Quality Issue Addressed. Comparing to size of areas shows the disparity between amount of research found for the different categorisations of the literature. Notice the lack of purely graph-based research.

2.3.1 Weighting Strategy

The discourse in the identified literature finds multiple approaches to instance weighting. Furthermore, there is disagreement in the best fundamental approach to instance weighting.

One school of thought is inspired by boosting for classification, and carries the boosting analogy into clustering. In these studies, increased weight is given to instances that are distant, inconsistently clustered, near boundaries or are far away from cluster centres. [8, 9, 15, 16]. This mirrors boosting for classification where training instances which are wrongly or uncertainly classified are "boosted". In the literature, the boosting analogy for clustering is practically implemented in different ways. In Nock and Nielsen's work data points are iteratively weighted based on their distance from the cluster centres [8]. Similarly in Hammerly and Elkan's research, instance weights are calculated based on distance to the cluster centres [16]. In Lei Gu's research, the angular relationships between data points and cluster centres are used to assign weights. Points are described as either angled between clusters or away from other clusters, and based on this their instance weight is assigned based on their angle *or* their distance from their nearest cluster centre. Points that are angled away (from other clusters relative to their own) or are distant from their own cluster receive a higher weight. In Topchy et al.'s work the boosting analogy is followed very closely. They implement a boosting-style ensemble and increase weight on inconsistency clustered instances. These are the instances that are typically near cluster boundaries. The instance weights control the sampling probability for the proceeding samples in the boosting ensemble [15]. Notably, a general trend of these works is that they do not consider datasets with outliers or imbalanced clusters. They mostly consider noise, tightly-packed

clusters and non-spherical clusters. Thus it is reasonable to hypothesise that these instance weighting strategies may not be so beneficial on datasets with outliers as they may be inclined to assign a high weight to outlying instances, which may skew the clustering model.

The other school of thought draws inspiration from instance selection and thus increases the instance weights on the centrally positioned instances to avoid being influenced by noise and outliers. Such works include [17], [18], [19], [20], [21], [10], [22] and [23]. Each work is unique, but can be broadly categorised into three categories. The first category essentially uses some distance to centroid [17], [18], [20] and [21]. The second category uses an information theory approach from density information [10]. Finally, the third category are algorithms specialised for image segmentation, these use local image patch information, considering a pixel's (an instance's) consistency with its surrounding pixels [19, 22, 23]. Some of these studies include testing on datasets with outliers and demonstrating their effectiveness in this use case [10, 17, 20].

Interestingly, there is a third smaller school of thought that in essence combines aforementioned strategies. Zhai et al.'s approach gives higher weight to both far and near instances [24]. Similarly, Guan et al. gives higher weight to density peaks and connecting areas [25]. Notably, both these methods are *real* ensemble "boosting" methods.

Directly comparing the experimental results is imperfect due to differences between the datasets, differences in the metrics, differences in the optimisation of parameters and differences in the preprocessing. However, it can be observed that across Chen et al.'s 15 datasets an overall average accuracy of 75% is

reported with their best instance weighted approach (P_SFCM) [17]. While in Gu's work using their best approach (SWKMA) across their 11 datasets an overall accuracy of 64% is reported [9]. While the selection of datasets is different, generally Chen et al. chooses complex, noisy and high dimensional datasets (including image datasets). While Gu uses simpler, lower dimensionality datasets. Considering the difference in the datasets, this adds additional merit to 75% achieved by Chen et al. This would suggest that up-weighting central instances has more merit than up-weighting distant instances, especially for noisy image datasets.

2.3.2 Actuation of the Weights

Instance weights have been applied to clustering algorithms in different ways.

In the case of partitioning-based methods, which typically involve iteratively applying a "cluster assignment" step and a "centroid update" step. Instance weights have been applied at both the cluster assignment and centroid update step. The literature is divided between the approaches with some works including [19], [20], [21], [22], [23] and [26] using the instance weights in the "cluster assignment" step. While others applied instance weights in the "centroid update" step, this includes, [8], [9], [10], [17], [18], [19], [21], [22], [23], [26] and [27]. In two of the ensemble based approaches, sampling was used to enact the instance weights [15, 24]. In these cases, the instance weights were used to inform the sampling of instances within the clustering ensembles generative mechanisms. In some respects, this is similar to simple instance selection before applying the clustering approach. Although, here the difference is that many weighted samples are taken as part of the ensemble clustering

approach. In particular, the sampling (informed by the instance weights) is the generative mechanism of the clustering ensemble - rather than a separate preprocessing step.

While no one work provides a direct comparison, reviewing the benefit each strategy yields over a traditional non-instance weighted approaches gives a small insight into effectiveness of the different strategies for applying the instance weights. In Guo et al. an increase in accuracy of $\sim$3% compared to plain k-means was reported, when using instance weighting to inform the "cluster assignment" step [20]. Using the instance weights to inform the "centroid update" step shows the largest benefit to clustering accuracy. Chen et al., reports increases of accuracy of $\sim$3% to $\sim$28% (with a average of $\sim$13%) when comparing their instance weighted approach to k-means [17]. Similarly, Wang and Angelova, find that their instance weighted approach increases clustering performance (measured using Adjusted Random Index) by as much as $\sim$20% compared to Fuzzy C-Means [18]. Furthermore, Gu, observed in increases in accuracy of between $\sim$1% to $\sim$20% (with an average of $\sim$5%) when comparing their instance weighted approach to plain k-means [9]. Topchy et al. finds using a sampling based approach to enact the instance weights resulted in accuracy gains of between $\sim$1% to $\sim$5% [15]. Overall, it seems the approach of using the instance weights to inform the "centroid update" step is most advantageous for accuracy. Although, the literature does not provide a fully clear picture on this. Also, some methods such as using instance weighting to inform sampling in a clustering ensemble is little investigated.

2.3.3 Ensemble Techniques

Another theme that was identified from the literature was the presence of "true" instance weighted clustering ensembles, examples include [15], [24], [25] and [27]. Before conducting the literature search it was known that boosting was an inspiration of instance weighting. However, in conducting the search it was found that the ensemble method of using multiple models had been applied to clustering with instance weighting. The search string did contain a mention of boosting – since it is a metaphor which inspires instance weighting. However, the literature search shows that some works took the metaphor literally, and have implemented ensemble techniques [24, 25]. Implementing "boosting" (traditionally a classification technique) for clustering involves slightly redefining boosting, since, traditionally boosting uses label information, which does not exist in clustering.

Zhai et al. uses a boosting approach on fuzzy clustering which increases the sampling probability of instances that are ambiguously clustered. Instances which are either ambiguous assigned to multiple clusters (in terms of membership degree) or instances which have been assigned to different clusters in previous iterations receive increased instance weight, increasing their probability of being sampled in successive clusterings [24]. Liu et al. uses instance weighted k-means on co-association matrices produced by an ensemble of k-means partitioning. They demonstrate that spectral clustering on a co-association matrix is equivalent to weighted k-means clustering on a binary matrix [27]. This a very different way of applying instance weighting using in it the consensus stage rather than in the generative stage of the clustering ensemble.

Applying instance weighting to an ensemble is a promising option, since ensem-

ble techniques can leverage increasingly abundant compute resources. However, this area also presents a gap as only three papers were identified by the literature search - and each utilises a completely unique approach to each other.

2.3.4 Benefits of Instance Weighting

Several works highlighted or demonstrated that their instance weighted algorithm could handle noise [8], [18], [19], [20], [22], [23] and [27]. Both the level of focus and the definitions of noise varied between the works. Some works investigated Gaussian noise [8], [19] and [23]. A couple of works investigated "Salt and Pepper" noise (specifically in images) [19] and [23]. While other tackled application specific noise, such as: Noise in environment sensors readings (temp., humidity, etc.) [20], Noise in image textures [22], Noise in gene expression data [18].

Two pieces of research focused on outlier accommodation by adding artificial outliers to various datasets. Chen et al. added many outliers (up to 8% of the size of the data) using a uniformly random approach which placed outlying instances at a distance around the dataset [17]. Yu et al. add a single strongly outlying instance to test the robustness of their approach to the presence of an outlier [10].

There was some in investigation into imbalanced clusters too, although, arguably imbalance was never particularly focused on. Wang et al.'s work tested performance of their instance weighted approach with a dataset "two-rectangles" which contained two uniformly random clusters, one with 200 instances and the other with 800 instances [21]. It was shown that their approach

could correctly partition this dataset unlike k-means. In Gu's research multiple datasets with various levels of imbalance were used. One of their experiments uses the "Bensaid-2d" dataset which has three very imbalanced clusters with 18, 141 and 20 instances respectively [9], again performance was tested against k-means showed a significant increase in accuracy of 14% accuracy.

Overall the literature shows how instance weighting approaches have increased the robustness of clustering in the case of noise, outliers and imbalance. It is clear that most of the literature is focused on handling noise and that there is much less research on accommodating outliers and handling imbalance through instance weighting. It seems there is a gap in the literature in handling outliers and imbalance. Positively, some of the research on noise maybe able to enjoy successes against outliers and imbalance - since in essence, noise, outliers and imbalance are all similar. Outliers could be argued to be a extreme case of imbalanced clusters or noise. In addition to robustness gains, one interesting and unexpected finding was that instance weights have been used to expedite the runtime or convergence of clustering algorithms [15, 22, 27]. For example, in Liu et al.'s research an instance weighted k-means algorithm was used to establish the consensus clustering in their ensemble. This reduced the runtime from 30 seconds (using traditional approaches for ensemble consensus) down to just 6 seconds, on the MNIST digits dataset.

2.3.5 Compatibility

Little work discussed instance weighting as it technique in its own right. Only three papers (all regarding partitioning-based clustering algorithms) described instancing weighting as a generalised framework [8, 10, 16]. Only one paper

demonstrated how one approach to instance weighting could be applied across multiple clustering algorithms including k-means, fuzzy c-means, Expectation Maximization, and k-harmonic means [8]. Clearly more research in this space could lead to a highly-compatible instance weighting techniques as better alternatives to instance selection or boosting ensembles.

The other way that compatibility arose during the literature analysis, is instance weighting's compatibility to operate alongside or be integrated with other techniques. For example in Chen et al.'s approach, PCA is iteratively optimised during the clustering, alongside the iterative application of instances weights [17]. In Hamerly and Elkan research, different initialisation strategies were investigated. They trialled Forgy (chooses k data points at random as initial centroid) and Random Partition (assigns each data point to a random centre - then computed the resulting centroids) methods. With their instance weighted approach "H1" they found that the Forgy method of initialization produced the best clustering performance [16]. Wang and Angelova demonstrate how instance weighting can be applied in a high-dimensional feature space, calculated using a kernel method [18]. Makkhongkaew et al. showed how instance weighting can be integrated with feature weighting and semi-supervised clustering [26].

2.4 Conclusion

In conclusion, the themes identified some of the discussions in the literature around instance weighted clustering. The themes themselves identify the key considerations when designing an instance weighted clustering approach.

Along the theme of **Weighting Strategy**, two fundamentally different approaches have been trialled, one approach increases the weight on peripheral (prototypical) instances, while the other increases the weight of archetypal instances. Results in the literature along with logical reasoning suggest that increasing the weight of archetypal instances is most suitable for the aim of robust clustering. There is also a less researched third approach which combines both strategies, certainly this is an interesting gap in the literature. Looking at the **Actuation of the Weights**, this is second ingredient required when designing an instance weighting approach. In the case of partitioning-based clustering the literature presents two approaches for enacting the instance weights: in the "cluster assignment" step *or* in the "centroid update" step. Good success with using the instance weights to inform the "centroid update" in shown in the literature so this appears to be a sensible choice. In the special case of clustering ensembles, the instance weights have been used to inform the sampling probability within the generative mechanism. Generally, results from **Ensemble Techniques** seem promising, but research is limited, so therein lies a gap for further work. Regarding the **Benefits of Instance Weighting**, while less researched than feature weighting, it does have good experimental and theoretical evidence of its benefit, both in the case of noisy and clean datasets. There is a significant gap establishing the benefit of instance weighting upon datasets with outliers and datasets with imbalanced clusters. Finally, in the theme of **Compatibility** it was found that there is some but limited vision of instance weighting as unified framework for applying to across many clustering algorithms.

Now moving to reflecting on the research questions. In partial answer to RQ1

"How can instance weighting be applied to partitioning-based clustering algorithms for outlier accommodation?", the studies show how instance weighting can be designed and be a valid approach to handle noise, offering accuracy gains of ∼20% in some cases. Extending this work to give a more complete answer that is relevant to outliers, is a direction of this work. Reflecting on RQ2 *"How can instance weighting be applied to graph-based clustering algorithms to handle imbalanced data?"*, it seems there is little literature integrating graph-based clustering and instance weighting, thus motivating this work. Finally, regarding RQ3 *"Under what conditions does instance weighting enhance clustering performance on data characterized by the presence of outliers or class imbalance?"*, this review finds evidence of instance weighting being useful for noisy data, gene expression data and image data to name but a few cases. However, there is scope to investigate instance weightings suitability for imbalanced data.

Finally, I make some final remarks and observations. Despite graph-based clustering being a more promising and broadly applicable approach (in terms of handling data of arbitrary shape), there is distinctly less research exploring instance weighted graph-based clustering. Furthermore, around half of the studies failed to compare against instance selection / sampling. So these are areas to include in my work to build upon the previous research. Researchers have not quantified the effectiveness of instance weighting's suitability for outliers and imbalance in as much detail as is possible. Research on the subject has been mostly restricted to comparisons on benchmark datasets either artificial or real-world, and one-off real-world applications, but few have directly investigated the robustness by incrementing properties like outlierness and imbalance while observing the impact on clustering performance.

Chapter 3

Methods

3.1 Partitioning-based Clustering

Partitioning-based based clustering algorithms iteratively optimise a partitioning of the data.

In practice, the common element of partitioning-based algorithms is deciding some initial partitioning/membership of the data then iteratively updating it to minimise some measure of fitness of the partitioning. There are many partitioning-based algorithms proposed, the below list contains a few examples:

- k-means
- k-modes
- Fuzzy k-means clustering
- Mean shift

Possibly the most popular and most widely-used and researched partitioning-based clustering algorithm is k-means. K-means is a very classical clustering

algorithm. The core ideas of the algorithm (error minimisation and representation of clusters via centroids) can be traced back to a paper by Steinhaus in 1956 [6], with MacQueen (publishing in 1967) and Lloyd (publishing in 1982) independently developing the k-means algorithm we recognise today [7, 28]. Arguably, a couple of reasons it remains popular is its simplicity and good performance when data is near spherical and free of quality issues. Due to its simplicity and performance, it is often included in clustering libraries [1].

K-means takes a single input parameter of k (this is the number of clusters to partition the data into) along with a numeric dataset. To initialise the algorithm, k instances are uniformly randomly selected was the locations at which to place centroids. Each centroid will model one cluster/partition.

K-means then iteratively repeats two steps: "cluster assignment" and "centroid update". In the "cluster assignment" step the distance between each instance to each centroid is calculated using Euclidean distance (although other distance metrics such as Minkowski can be used [29]). Using these distances, each instance is then assigned to its nearest cluster. In the second step, "centroid update", for each cluster, the arithmetic mean position of its instances is calculated and this becomes the new centroid position. This process repeats until the error function (see Equation (3.1)) is convergent. In practice, this can be identified by centroids settling at some local minima and thus no instances change cluster assignment from the previous iteration. The error function is given by:

[1] Scikit-learn, SciPy, Spark MLlib, R's stat and cluster packages, Weka and OpenCV to name just a software packages/libraries/module implementing k-means.

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2 \tag{3.1}$$

In equation (3.1), C is the clusters. $|\cdot|$ represents Euclidean distance. Clusters corresponding to the k value are denoted as $C_{i=1...k}$ and μ_i is the arithmetic mean of instances in each cluster, in turn.

Minimising the error function is NP-Hard [30]. To lessen the error from converging to some local minima, several initialisation strategies have been proposed. Arguably most commonly used is running the algorithm multiple times with different initialisations then selecting the result with the lowest error. Although many other initialisation methods exist (for a comparison see [31]).

Despite k-means strengths there are several drawbacks. K-means clustering is sensitive to outliers. Outliers in the dataset influence centroids (and thus the partitioning) away from the centres of the inlying data and this can comprise how well k-means fits the inlying data. Additionally, another drawback is that k-means can have difficulty modelling data where the clusters are non-spherical. Structures in the data such as concentric clusters, concave, skewed or rectangular structures can be poorly modelled by k-means. K-means also requires that the user specifies the k value from prior knowledge or alternatively estimating it using a technique such as the elbow-method (which is can give an ambiguous result). A survey of developments and usage of k-means and be found in [32].

In the Appendix, a worked example of k-means clustering can be found along with a Python implementation.

3.2 Graph-based Clustering

Graph clustering is less common and much younger than partitioning-based clustering, but is becoming increasingly popular. Graph clustering constructs a graph representation of data, which it uses to partition the data. In practice, a graph clustering algorithm can be identified by one of its first steps, which involves applying a function to convert the input data into a graph. While there are less well known graph-based clustering algorithms, most notable are random walk clustering and spectral clustering. Unlike k-means and most other partitioning-based methods, most graph-based clustering methods can fit clusters of arbitrary shape.

Focusing in on spectral clustering, broadly summarising it has three steps. Firstly, a graph is constructed. Secondly, eigenvalues and eigenvectors are calculated. These both describe the nature of the dataset and can optionally be used to embed the data into a space in which the cluster are better separated. Thirdly, the graph is partitioned either directly using the eigenvector information or by applying k-means on the embedded space. Below provides a more detailed view.

Given a dataset of size n, spectral clustering constructs a graph representation $(G = V, E)$ of data points (instances), $x_1, x_2, ...x_n$, where the vertices (V) represent the data points in the dataset, the edges (E) of the graph represent pairwise similarity $s_{ij} = s(x_i, x_j)$ of the instances. The similarity measure can be either 1 or 0 (called an adjacency matrix), or positive real values (called

an affinity matrix). The adjacency or affinity matrix (A) is a $n \times n$ matrix constructed from the graph G defining the similarity of the instances. The adjacency or affinity matrix can be calculated in a number of ways, common choices include a k-nearest neighbours algorithm or a radial basis function. In the case of the k-nearest neighbours, each point is connected to its k nearest neighbours. These connections can be modelled into an adjacency matrix A; where there is a connection between nodes 1 is used, where two nodes are not connected 0 is used. From matrix A, the diagonal/degree matrix D is calculated. Next, either the un-normalised or normalised Laplacian matrix (L) is calculated from $D - A$. Then the eigenvalues$(\lambda_1, \lambda_2, ...\lambda_n)$ and eigenvectors $(l_1, l_2, ...l_n)$ are calculated from L. This process involves, finding the characteristic polynomial of L and solving for the eigenvalues. Then, from the eigenvalues, the non-zero eigenvectors can be solved for. These are then used to partition the graph using a clustering algorithm such as k-means. A simpler alternative way, which can be used to partition the graph when $k = 2$, is the basic Fielder method. In this method, the second eigenvector l_2 associated with the second-smallest eigenvalue λ_2 is selected. This eigenvector l_2 is referred to as fielder vector. The polarity of each value in this vector partitions the corresponding data point. The partitioning of the graph aims to minimise the number of edges connecting the two partitions and keep the size of the partitions similar. While spectral clustering can performance well on variety of datasets, it has some weaknesses. Spectral clustering can fail to find the most appropriate partitions between imbalanced clusters.

An overview of research around spectral clustering can be found in [33]. In the Appendix, a step-by-step example of spectral clustering can be found.

3.3 Clustering Quality Metrics

When evaluating of clustering performance there are fundamentally two approaches: intrinsic and extrinsic quality metrics.

Intrinsic metrics assess clustering results by comparing the resultant partition against an internal model or definition of a "good" cluster structure. The primary advantage of this approach is its independence from ground truth labels. This is crucial for real-world applications where the objective of clustering is precisely to discover the *unknown* cluster structure. However, a significant drawback is that the metric's embedded definition of a "good" clustering may not align with the structural definition implied by the clustering algorithm itself. For example, the definition utilized by the silhouette coefficient is generally compatible with centroid based algorithms like k-means, but it may incorrectly assess a density based clustering result, such as concentric clusters identified by DBSCAN as poor.

Conversely, extrinsic metrics compare the resultant clustering partition directly against a pre-defined ground truth (some known class labels). Extrinsic metrics offer the advantage of direct comparison against a meaningful, known structure, allowing for the evaluation of an algorithm's ability to recover that specific pattern. This allows for straight a forward comparison between different algorithms. A limitation is that the ground truth represents only one valid partition. In many real-world datasets, multiple valid clusterings may exist (e.g., partitioning a dataset of pet images by animal type, by color, or by body position). Both the advantages and drawbacks of these metrics ultimately stem from the inherent subjectivity in defining what constitutes a "good" clustering.

Given this work's objective to investigate and compare clustering approaches, extrinsic metrics will be employed where ground truth data is available. Specifically, the Normalized Mutual Information (NMI) metric is selected as the primary clustering performance measure.

NMI quantifies the statistical dependency between the assigned cluster labels and the ground truth labels, yielding a value between 0 and 1. A score of 1 indicates a perfect correlation (i.e., a complete match between the clustering and the ground truth), whereas a score of 0 signifies that the cluster assignments are statistically independent of the ground truth. NMI is a normalised variant of the original Mutual Information (MI) metric, which is rooted in Claude Shannon's Information Theory. The normalisation step ensures the score is bounded between 0 and 1, simplifying interpretation. Its application in the context of clustering evaluation can be traced back to the 2002 work of Strehl and Ghosh [34]. The reasons for selecting NMI as the key measure for cluster performance in this thesis are:

1. It avoids imposing a specific model on the clustering solution, unlike internal metrics, allowing for the unbiased evaluation of different algorithms.
2. It inherently supports the inclusion of datasets with arbitrary cluster shapes.
3. It's widespread adoption in clustering literature makes it possible compare results with other literature, to some extent.
4. NMI does not require solving the label correspondence problem, which can be computationally expensive when dealing with a high number of

clusters.

A consideration for experiments involving imbalanced cluster distributions is the known sensitivity of NMI to small clusters, which can lead the metric to underestimate cluster performance. Rezaei and Fränti [35] illustrate this point with a experiment, from which this example is drawn: Starting with a ground truth of three equally sized clusters (each N=1000), with an incorrect assignment rate of 20% between the first two clusters, gives an NMI of ∼84%. However, when the size of the third cluster is reduced to N=50 while maintaining the same 20% error rate between the first two clusters, the NMI score drops sharply to ∼67%. This tendency to penalise solutions on imbalanced datasets must be considered when assessing results from imbalanced clustering problems, particularly in the imbalance clustering experiments within this work.

Equation (3.2) shows the calculation of NMI.

$$
\begin{aligned}
H(U) &= -\sum_{i=1}^{|U|} P(i)\log(P(i)) \\
H(V) &= -\sum_{i=1}^{|V|} P(i)\log(P(i)) \\
\mathrm{MI}(U,V) &= \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right) \\
\mathrm{NMI}(U,V) &= \frac{\mathrm{MI}(U,V)}{\mathrm{mean}(H(U),H(V))}
\end{aligned}
\tag{3.2}
$$

In this formulation, $H(U)$ and $H(V)$ represent the entropy (amount of information) of the cluster assignments U and the ground truth labels V, respectively.

The term $MI(U, V)$ is the Mutual Information, where $P(i, j)$ is the joint probability that a data point belongs to cluster i in partition U *and* ground truth class j in partition V. This can be visualised using a contingency table. The marginal probabilities, $P(i)$ and $P'(j)$, represent the *individual* probabilities of an instance belonging to a specific cluster $i \in U$ or a specific class $j \in V$. The final $NMI(U, V)$ is then calculated by normalising the MI by the mean of the two entropies. Worked examples of NMI across a variety of clustering scenarios can be seen in Appendix 9.5.

Chapter 4

Instance Weighting for Partitioning-based Clustering

4.1 Introduction

This chapter begins the empirical research by investigating how can instance weighting be applied to a partitioning-based clustering algorithm for outlier accommodation. Arguably the most popular clustering algorithm is k-means [28]. This partitioning-based algorithm partitions instances into a given number of clusters k. K-means iteratively assigns instances to clusters based on their distance to the centroids of the clusters, the centroids' positions are then recalculated to be the means of instances in their respective clusters.

Jain provides an overview of clustering discussing the key issues in designing clustering algorithms, and points out some of the emerging and useful research directions [32]. Jain's paper outlines six problems / research areas, one of which is "A fundamental issue related to clustering is its stability or consistency. A good clustering principle should result in a data partitioning

that is stable with respect to perturbations in the data. We need to develop clustering methods that lead to stable solutions.". This is the problem my research considers solving through instance weighting. Considering outliers, from a statistics' perspective, outlierness is a scale rather a boolean property, so it makes sense to use weighting rather than selection in response.

Hawkins defines an outlier as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism" [36]. Outlier accommodation enables algorithms to accommodate outliers; it is the opposite of outlier diagnosis, where outliers are identified and removed before processing. Instance Weighting can provide a way for clustering algorithms to accommodate outliers, by adjusting how much to learn from outlying instances. This is important since clustering algorithms, such as k-means can be adversely effected by the presence of outliers in a dataset. Whilst it is true that some types and severities of outlier should be fully discarded, some types and severities of outliers may be best partially retained for the clustering process to learn from. This is especially important when the total number of instances is low.

To evaluate instance weighting for partitioning-based clustering, in this chapter an instance weighted version of k-means will be proposed and evaluated on several synthetic and benchmark datasets with outliers.

4.2 Related Work

Nock and Nielsen's research [8] is inspired by boosting algorithms (from supervised learning) and k harmonic means clustering [37]. They are the first to formalise a boosting based approach, their solution penalises bad clustering accuracy by updating the instance weights. Their algorithm gives more weight to data points that are not well modelled. Their approach could be described as a statistics based approach. Their paper investigates, for which scenarios, instance weighting improves the accuracy of clustering and if instance weighting can reduce initialisation sensitivity. They investigate applying instance weighting on multiple algorithms including k-means, fuzzy k-means, harmonic k-means and Exception Maximisation and prove the applicability of instance weighting to a range of algorithms. Their research shows that instance weighting could speed up the convergence of partitioning-based clustering algorithms. They highlight the growing attention around instance weighted iterative clustering algorithms in unsupervised learning. My research differs by proposing a generalisable method using a density based technique. I also investigate how instance weighting can address the presence of outliers in a dataset.

Sample Weighted Clustering by Jian Yu et al. weights instances using a probability distribution derived from an information theory approach [10]. They point out that there is little research on sample (another name for "instance") weighted clustering compared to feature weighted clustering. Like my work they investigate the benefit instance weighting for datasets with outliers, integrating instance weighting with the popular k-means algorithm. The instance weights are calculated using the maximum entropy principle. Essentially, the instance weights are a probability distribution $p(X_k)$ calculated using

$\exp(-\varsigma \times d_k)/\sum_{k=1}^{n} \exp(-\varsigma \times d_k)$ where ς is tunable parameter between 0 and 1 and d_k is a distortion factor, which is equal to Euclidean distance from point k to its cluster centroid in given iteration. Hence, their algorithm iteratively calculates the weights in each cycle of the clustering algorithm. They highlight that just one outlier can adversely effect the clustering output of k-means, fuzzy c-means and expectation maximisation clustering algorithms. They show that their information theory based instance weighting approach produces robust clustering when outlier(s) are present across a variety of datasets. In addition, it was found that their weighting also made their algorithm less sensitive to initialisation.

Lei Gu's research [9] uses two weighting schemes to weight instances. Their weighting schemes operate per cluster. Instances which are considered either "ambiguous" or "unambiguous" based on their angle relative to a vector between the cluster centre and cluster boundary. Points which are angled such that they within a specific area (defined by an angle) between the boundary and centroid are classed as "ambiguous" points outside this area are classified as "unambiguous". The "ambiguous" points are weighted using a Min-Max Normalised distance from the centre. The "unambiguous" points are weighted using the cosine of the of angle between the vectors: 1) the nearest cluster centre to the given unambiguous instance, and 2) the vectors of the nearest cluster centres to the other clusters in turn. In essence, points that are angled away from other clusters are given a higher weight. Their algorithm outperforms Jain Yu et al.'s algorithm (described in the previous paragraph) for accuracy. Lei Gu's research also considers non-image segmentation based clustering problems.

Hammerly and Elkan's research [16] investigates the k-harmonic means algorithm [37]. K-Harmonics Means builds on the existing k-means algorithm using the harmonic mean rather than the arithmetic mean. The harmonic mean has the advantage that points which are close to multiple centroids (potentially ambiguous points) affect the mean calculation less. Furthermore, the K-Harmonic means algorithm also adds instance weighting and soft membership. The instance weighting in K-Harmonic means recalculated iteratively weighting. The weight for a given instance is the distance from centroids. This means higher weights is given to points which are far from centroids. This loosely follows the boosting analogy from classification where poorly modelled points are upweighted. Hammerly and Elkan show that when they isolate the effects of the instance weighting this approach enhances the clustering performance the case of hard clustering but not soft clustering with a selection of datasets that do not include outliers. A criticism of their approach is that it increases the emphasis placed on any outlying points. They show it is possible to create unified framework for instance weighting partitioning-based algorithms.

In conclusion, successes seen with instance weighting based on the boosting analogy (giving more weight to poorly modelled points) is promising. However, using this analogy risks increasing the impact of outliers. There is some thought around instance weighting being generalised into an approach which can be applied to any partitioning-based, in particular centroid-based algorithm. This motivates discussion around the best way to approach the weighting of an instance for optimal clustering performance across different types of dataset. So far, instance weights have be calculated using various formulas which consider the distance of instances from one or more centroids. However, little research has made use of instance density information to weight

instances. In this work, density was used to define instance weights, with the aim of developing a clustering method that is robust to outliers. The intention is that by reducing the weight outliers have, their misleading information can be disregarded and a better clustering outcome can be achieved.

4.3 Local Outlier Factor

Local Outlier Factor (LOF) is an outlier detection algorithm which provides a measure of outlierness. It is typically used for outlier removal, where a threshold is applied to identify the most outlying instances for removal. LOF works by comparing the density of an instance to that of its neighbours [38]. Equations (4.1), (4.2) and (4.3) show how to calculate the LOF scores as is defined in [38]. A represents the instance for which the local density is being calculated. k represents the number of neighbours to consider. d_{k_NN} is the distance from a given point to its k^{th} furthest point. $N_K(A)$ is the set of k nearest neighbours to A, these in turn are noted as B.

$$d_{Reach,k}(A, B) = \max\{d_{k\_NN}(B), d(A, B)\} \tag{4.1}$$

$$lrd_k(A) := \frac{1}{\left(\frac{\sum_{B \in N_k(A)} d_{Reach,k}(A,B)}{|N_k(A)|}\right)} \tag{4.2}$$

$$LOF_k(A) := \frac{\sum_{B \in N_k(A)} \frac{lrd_k(B)}{lrd_k(A)}}{|N_k(A)|} \tag{4.3}$$

Consider the example dataset in Figure 4.1 (left), the data point at location (5,5) labelled a is moderately outlying. *k-distance* is the distance to the k^{th} furthest point, so if $k = 3$, then k^{th} nearest neighbour of a would be the point

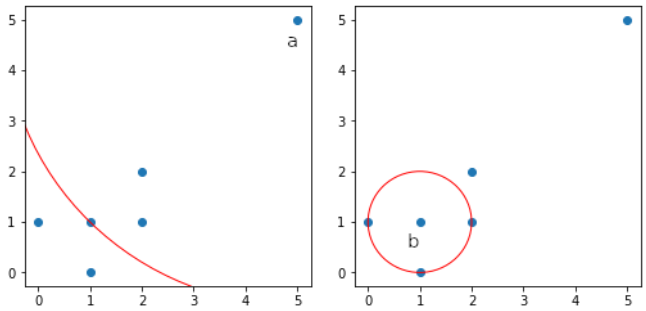


Figure 4.1: Calculating the reachability distance.

at location (1,1) labelled b. If point a is within the k neighbours of point b (See Figure 4.1 (right)), then the $reachability - distance_k(a, b)$ will be the $k - distance$ of b, the distance to the k^{th} further point (2,1) from b. Otherwise, it will be the real distance of a and b. In Figure 4.1 a is **not** within the k neighbours of point b so in this case it is the real distance between a and b.

To get the lrd (local reachability density) for the point a, first the reachability distance of a to all its k nearest neighbours is calculated and the average of that number in taken. The lrd is then simply the inverse of that average. Since a is not the third-nearest point to b (see Figure 4.1 (right)), the reachability distance in this case is always the actual Euclidean distance. A value of $LOF_k(A)$ greater than one indicates a lower density (thus the instance is outlier). A value of $LOF_k(A)$ equal to one indicates similar density to A's neighbours. In Figure 4.2, it can be seen that as point A leaves the cluster and becomes distant, it receives an increasingly high LOF score, highlighting it as an instance as an outlier.

One of properties that makes LOF ideal is that the LOF algorithm can work on datasets with clusters of different densities and instance count, as long as the number of k neighbours is below the number of instances in the smallest

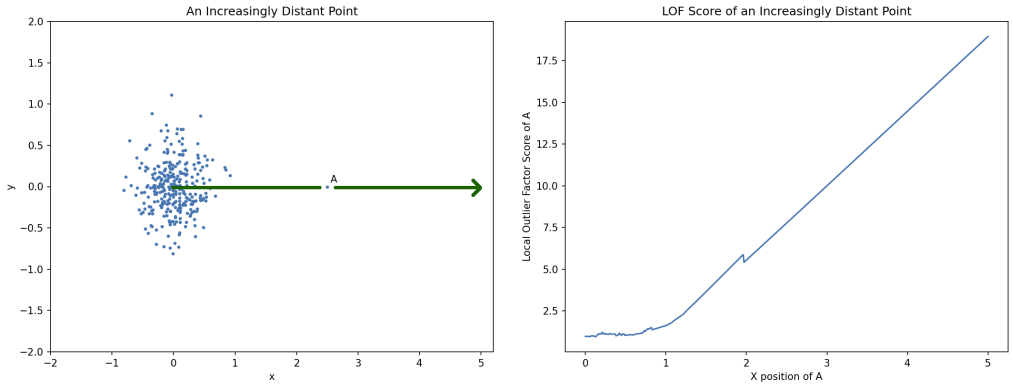


Figure 4.2: Demonstrating the LOF scores.

cluster. This is advantageous since it places little restriction on the dataset to which the weighted clustering algorithm can be applied to. However, one possible drawback to the LOF algorithm is its time complexity of $O(n^2)$, where n is the data size. However, there exists work speeding up LOF using GPU acceleration [39].

4.4 Proposed Methods

Two novel algorithms based on k-means are proposed: Local Outlier Factor Instance Weighted K-Means (LOFIWKM) and Iterative Local Outlier Factor Instance Weighted K-Means (ILOFIWKM). LOFIWKM calculates the weights over the **whole dataset once upon initialisation**, whereas ILOFIWKM calculates the weights **for each cluster upon each iteration**. The weights are generated by executing the LOF algorithm. This is the unique aspect of the approaches, rather than applying a threshold and removing the outliers, instead the outliers are weighted according to their LOF score. These weights are then used when calculating arithmetic means for the positions of the new centroids in the k-means algorithm. In Figure 4.3 the weights are represented

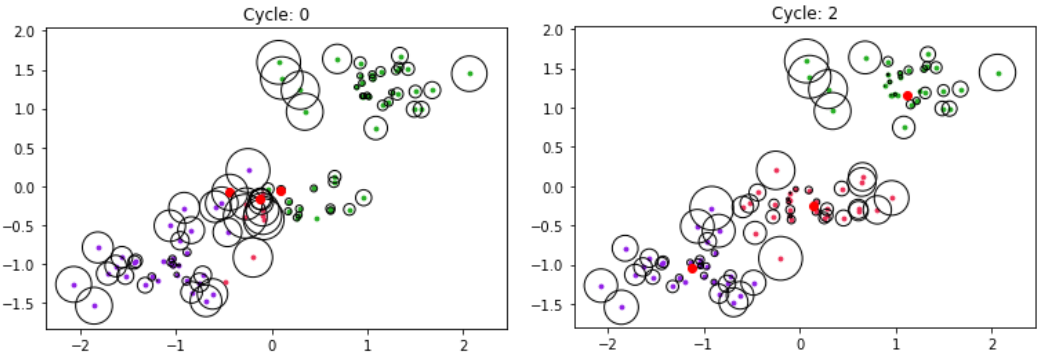


Figure 4.3: The ILOFIWKM algorithm showing how weights change as the algorithm executes. The three red dots are the centroids and radius of black circles shows the outlierness which is inverted to give the instance weight. The smaller coloured dots are the instances of different clusters.

by black circles, where the smaller the circle the higher the weight.

More formally, LOFIWKM, starts by calculating the LOF score of every instance, considering the whole dataset. Taking the whole dataset into consideration, outliers are highlighted relative to the whole dataset using LOF. Then as per k-means, centroids are initialised. However, the algorithm uses a weighted random initialisation based on LOF scores and instance positions. Then as per k-means, instances are assigned to the centroids they are closest to. As standard for k-means, the algorithm iterates until converged (there is no more reassignments of instance between clusters) or the max allowed iterations is met. In each iteration the algorithm calculates the new positions of the centroids based on its' instances, taking a weighted average using normalised inverted LOF scores as weights to moderate the impact of the instance positions on the mean. As in standard k-means, instances are assigned to the new centroid they are nearest to in terms of Euclidean distance. Algorithm 1 shows a formal description of the algorithm where, Dataset of instances = $D_{i=1,...N}$. LOF Scores for each instance in the dataset = $LOF_k(D_i)$. Clusters corresponding the K value entered = $C_{k=1...K}$ a centroid has a position and col-

lection of instances. The number of iterations / k-means cycles is denoted as c.

Algorithm 1 LOFIWKM

Calculate LOF for D
for all $w \in LOF$ **do**
  Assign $\frac{w-min(LOF)}{max(LOF)-min(LOF)}$ **to** w^*
end for
Assign LOF^* **to** LOF
Use LOF weighted random to select K positions from D assign **to** C
for all D_i in D **do**
  Assign D_i **to** k according to $\min(dist(D_i, C))$
end for
Assign 0 **to** c
while C **not** converged **or** $c \leq c^{max}$ **do**
  for all k in C **do**
    Assign $\frac{\sum_{k_N}^{k_0} i \cdot w}{\sum_{k_N}^{k_0} w}$ **to** k
  end for
  for all D_i in D **do**
    Assign D_i **to** k where $\min(dist(D_i, C))$
  end for
  Assign $c + 1$ **to** c
end while

ILOFIWKM operates the same as LOFIWKM up to the end of the iteration step. Then the algorithm recalculates LOF score of every instance by running the LOF algorithm *per cluster* and normalising the LOF scores *per cluster*. Algorithm 2 shows a formal description of the algorithm.

4.5 Experimentation

The purpose of the proposed algorithms is to improve k-means's ability to handle outliers. To evaluate the proposed algorithms, two types of dataset, synthetic and real-world, were experimented upon.

Algorithm 2 ILOFIWKM

Calculate LOF for D
for all w in LOF **do**
  Assign $\frac{w-min(LOF)}{max(LOF)-min(LOF)}$ **to** w^*
end for
Use LOF weighted random to select K positions from D assign **to** C
for all D_i in D **do**
  Assign D_i **to** k according to $\min(dist(D_i, C))$
end for
Assign 0 **to** c
while C **not** converged **or** $c \leq c^{max}$ **do**
  for all k in C **do**
    Assign $\frac{\sum_{k_N}^{k_0} i \cdot w}{\sum_{k_N}^{k_0} w}$ **to** k
  end for
  for all D_i in D **do**
    Assign D_i **to** k where $\min(dist(D_i, C))$
  end for
  for all C **do**
    Partially recalculate LOF for i in k
    for all w in k **do**
      Assign $\frac{w-min(LOF)}{max(LOF)-min(LOF)}$ **to** w^*
    end for
  end for
  Assign $c+1$ **to** c
end while

4.5.1 Synthetic Dataset

Experimental Setup

This first experiment uses a variety of randomly generated datasets to study the extent to which the proposed approaches can accommodate outliers. Two variables, "count of outliers" and "range of outliers" are experimented upon. Furthermore, the proposed approaches are compared against a plain k-means implementation.

The datasets trialled contain two clusters (C_0 and C_1) which are generated from two spherical Gaussian distributions. The y position of both clusters was set to 0 and the x position of C_0 was 0 while the x position of C_1 was set to 1. The covariance of C_0 and C_1 was 0.05 in both x and y directions. C_0 and C_1 each consisted of instances 100 each. Figure 4.4 shows a sample of the data.

To position the probable outliers, the following procedure was used. First, the

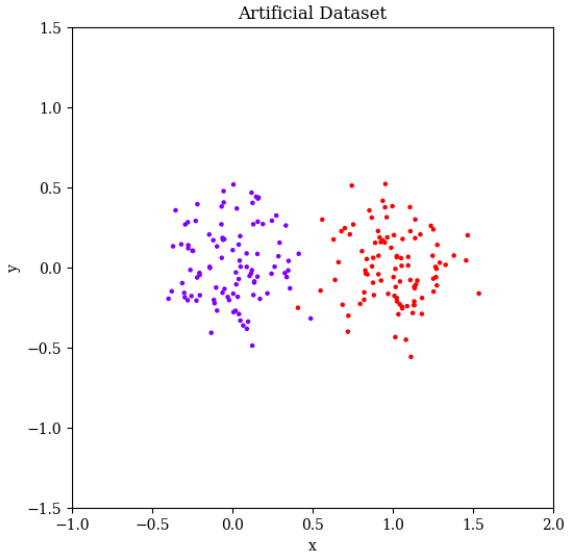


Figure 4.4: A sample of the artificial dataset used showing the 2 clusters.

standard deviation of x and y for a given generation of the dataset is calculated. The mean of these values is then calculated into m. Then, a upper u and lower l bound is defined as multiplier of the mean standard deviations. Using these bounds and Euclidean distance, outliers are then uniformly randomly generated. Figure 4.5 shows a visualisation of the procedure.

This approach has the strength that it provides a simple scale invariant method for defining outliers in terms of mean standard deviations away from the global centre of the dataset. This is helpful since outliers are often defined in terms of standard deviations away from the mean. However, a limitation of this approach is that it should not be used on data where the range of the dimensions varies greatly. Furthermore, a weakness is that uniform random generation does not guarantee that the generated points are distant from each other (hence in places the term "probable outlier" will be used). Hence, this method should be not used to attempt to create a too large number of outliers – through risk of them forming their own logical cluster. To label the

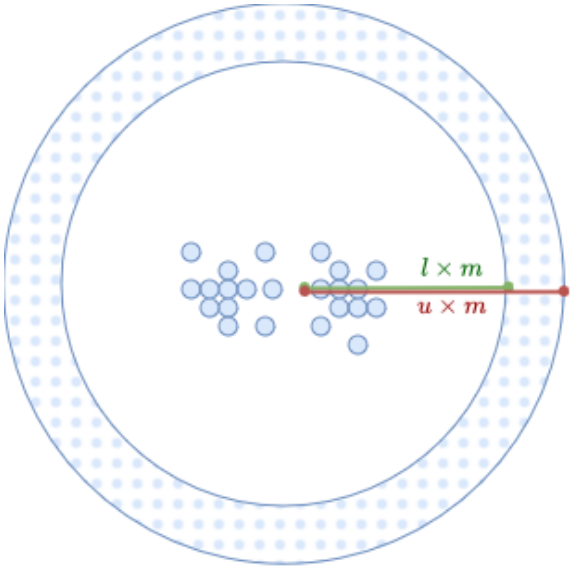


Figure 4.5: A visualisation of the outlier generation procedure, the ring-shaped blue dotted area denotes where instances will uniformly randomly generated to create probable outliers.

generated probable outliers, the labels from C_0 and C_1 are propagated using a KNN procedure with k set to 10. This choice is made such that the outliers do contain some meaning – a premise on which this thesis is based.

For the first experiment, the number of outliers varied from 0 to 25 in increments of 5. The lower bound multiplier l was set to 20 global standard deviations and the upper bound u was set to 22 global standard deviations. Figure 4.6 shows a sample of datasets generated with these settings.

For the second experiment, the lower and upper bound i.e. the range between which outliers are created was varied to incrementally position outliers at an increasing Euclidean distance from the global mean. l was varied from 5 to 30 in increments of 5, and u was set to $l + 5$. Figure 4.7 shows a sample of the datasets generated with these settings.

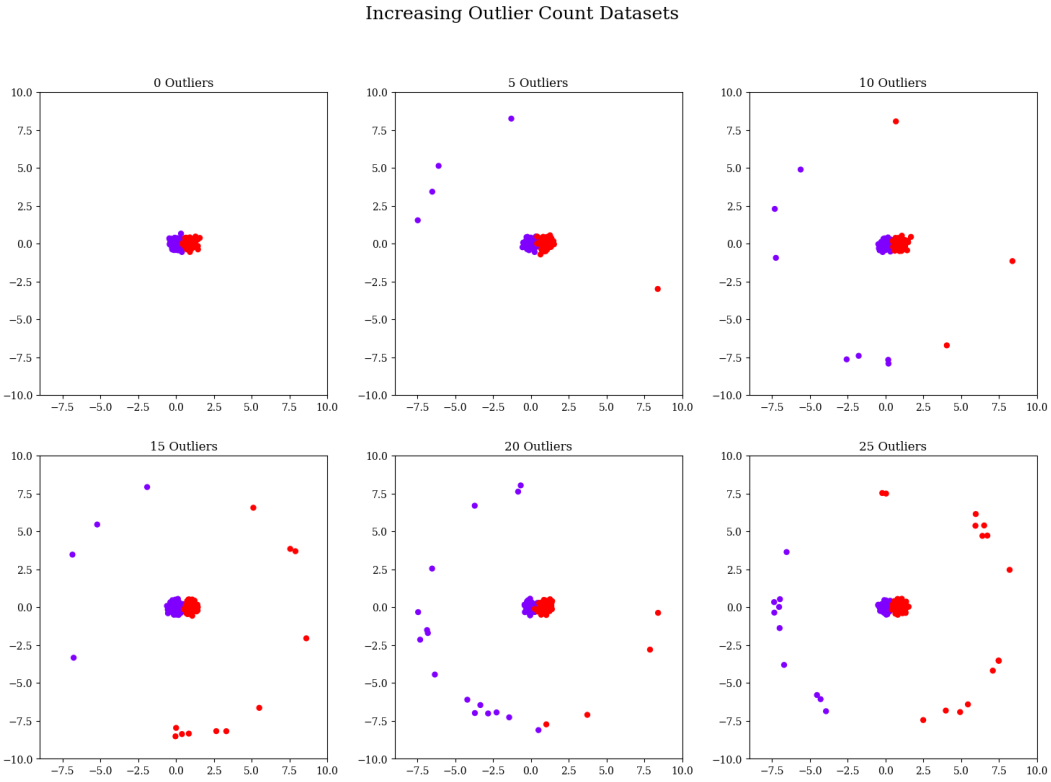


Figure 4.6: A sample of the generated datasets, showing the increasing outlier presence.

For both experiments, the two proposed algorithms LOFIWKM and ILOFIWKM are trialled alongside k-means. For the k-means algorithm, my own Python implementation was used. This is the implementation that was updated to create the novel algorithms. This ensures that the only difference between the k-means implementation and my instance weighted k-means algorithms is the changes described in this paper. For all algorithms the k value was set to the ground truth of 2. For LOFIWKM and ILOFIWKM, the k value of LOF was set to 30 nearest neighbours for density estimation. Prior to execution of the clustering, the generated datasets were min-max scaled between 0 and 1. All experiments are repeated 50 times as both the algorithms and the synthetic dataset generation are stochastic.

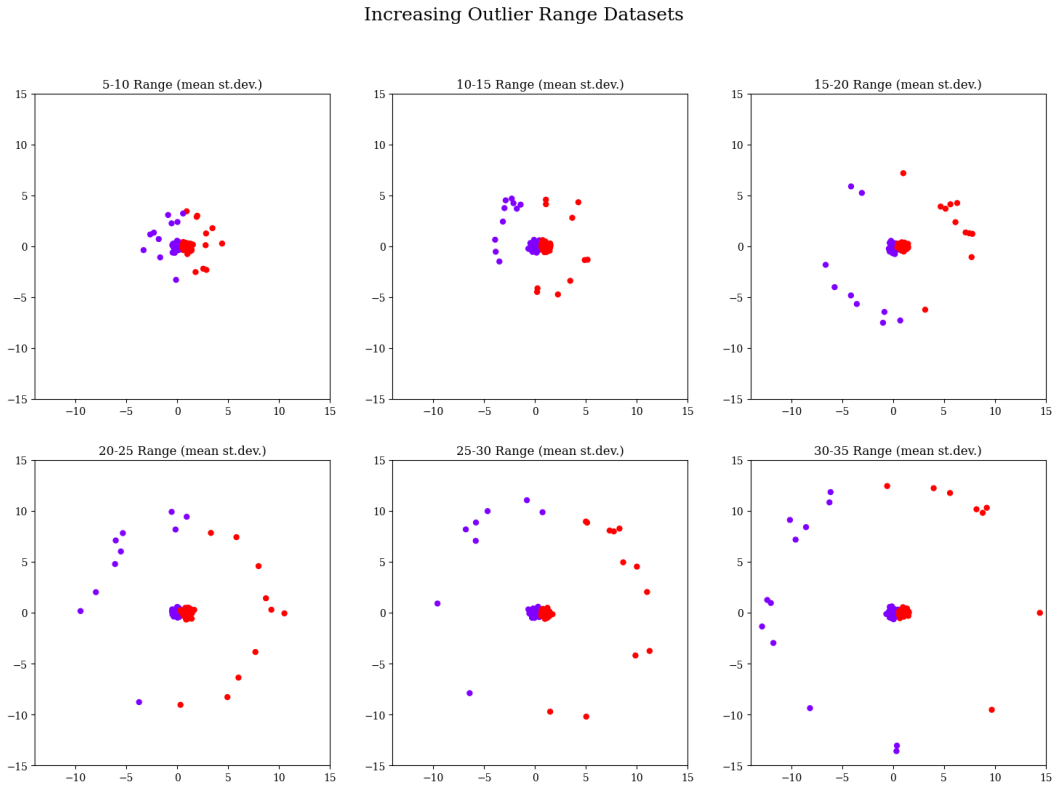


Figure 4.7: A sample of the generated datasets, showing the increasingly distant outliers.

Results and Discussion

The results of the varying the outlier count show strongly positive results for the LOF based instance weighted clustering. The results are presented using the NMI clustering quality metrics.

The results show instance weighted algorithms were able to achieve significantly better clustering accuracy than k-means. In Figure 4.8, when there are no outliers all algorithms perform similarly. However, once 5 or more outliers are introduced, the performance of k-means drops, whereas LOFIWKM and ILOFIWKM remain approximately constant until 15 outliers are added where clustering performance begins to slowly decrease. The results do not show a clear difference in performance between LOFIWKM and ILOFIWKM.

In Figure 4.9, a similar finding but for outlier distance can be seen. As the distance of the outliers from the global mean of the data increases, the accuracy of k-means quickly deteriorates. Conversely, LOFIWKM and ILOFIWKM algorithms are not noticeably effected by the presence of the increasingly distant outliers, only on the most distance value tested (30-35) does LOFIWKM and ILOFIWKM show a decrease. Across both experiments, there is a none-to-minimal benefit to using the iterative weighted version, ILOFIWKM.

An interesting and unexpected observation is that in both experiments LOFIWKM and ILOFIWKM both have a noticeably reduced standard deviation. This indicates that their clustering performance is more consistent. This is interesting because a commonly used strategy [1] to stabilise k-means cluster-

[1] For example, the popular sklearn implementation uses the multiple random initialisation practice.

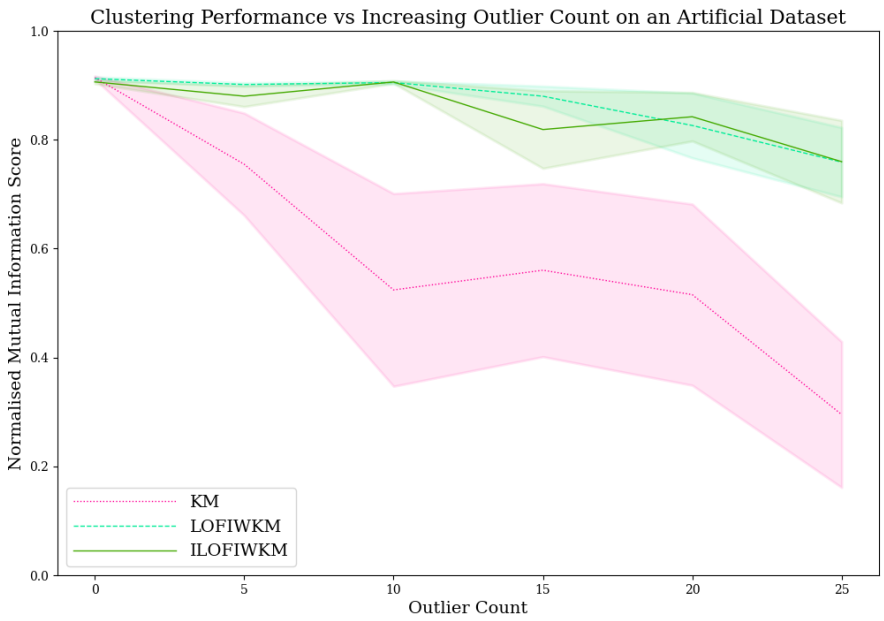


Figure 4.8: The average NMI score and standard deviation of k-means, LOFIWKM and ILOFIWKM on the synthetic dataset with increasing amount of outliers.

ing performance is to randomly initialise and run k-means multiple times and pick the best result. Whereas for LOFIWKM and ILOFIWKM, this arguably wasteful procedure this is not so important.

These findings demonstrate that the proposed instance weighting method can be effective given the presence of outliers in a dataset. Furthermore, the findings show instance weighting can be effective even when the outliers are numerous or have a large magnitude.

A secondary finding is that instance weighting combined with k-means can lessen the variability of the clustering performance. This is a problem normally tackled in research by using different initialisation strategies or initialising and executing k-means multiple times. This small finding suggests that instance

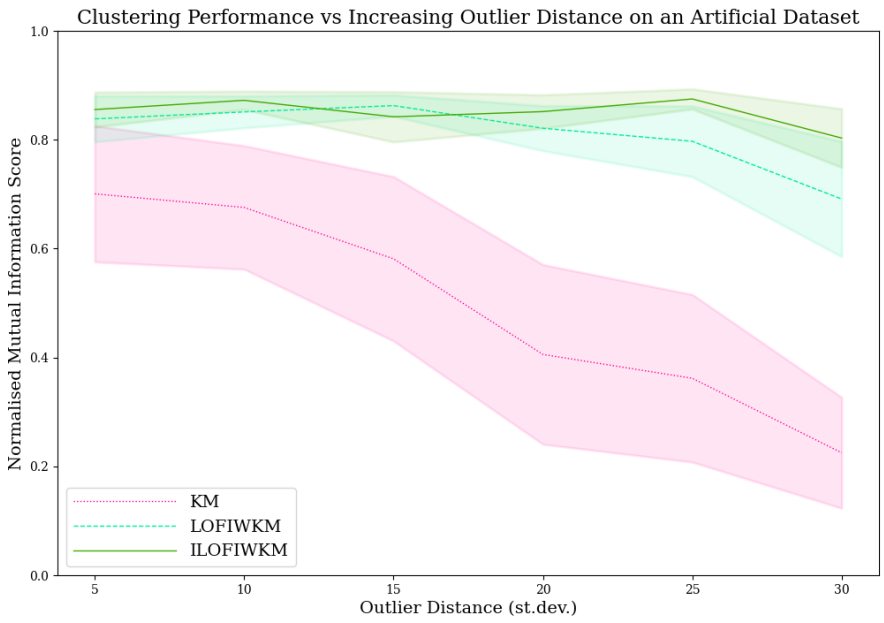


Figure 4.9: The average NMI score and standard deviation of k-means, LOFIWKM and ILOFIWKM on the synthetic dataset with increasingly distant outliers.

weights could be a novel alternative approach to encourage k-means to converge more consistently.

4.5.2 Benchmark Dataset

Experimental Setup

Further experiments are conducted on a real-world dataset containing 210 instances, 7 features and 3 clusters. The dataset presents the measurements of damaged wheat kernels of 3 different varieties. [40] The dataset was obtained via the UCI Machine Learning Repository [41].

To prepare the dataset for clustering some preprocessing steps were taken.

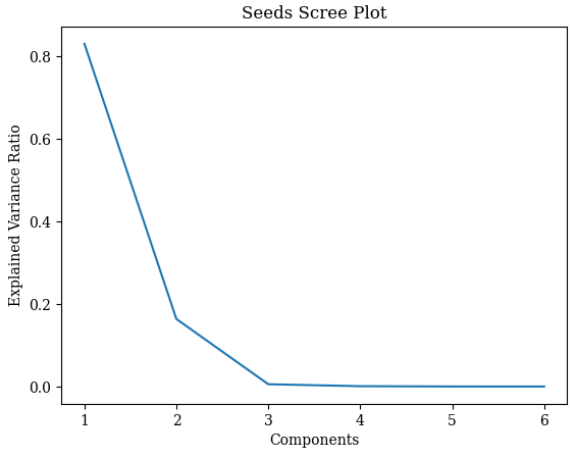


Figure 4.10: UCI Seeds dataset scree plot.

Firstly, the label information was removed. Secondarily, principal component analysis was applied to reduce the dimensionality to two. Figure 4.10, shows the scree plot for the seeds dataset. A scree plot shows the magnitude of the eigenvalues for each principal component, this gives an indication of the variance explained (information) contained in each principal component. The first principal component captures most of the information in the dataset, while successive principal components represent less and less. In Figure 4.10, using just two principal components, it is possible to represent 99.2% of the explained variance of dataset, implying that little information is lost despite compressing the representation down to two dimensions using principal components 1 and 2.

The benefit of reducing the dimensionality is twofold, firstly with fewer dimensions the distance calculations within k-means are more meaningful and thus theoretically the algorithm is more able to identify clusters (as well as faster to execute, although this dataset is trivially small). Secondarily, reducing the dimensionality of the dataset is further advantageous, as the instance weights in my approach are defined using a density-based method which is weakened by the presence of high-dimensionality.

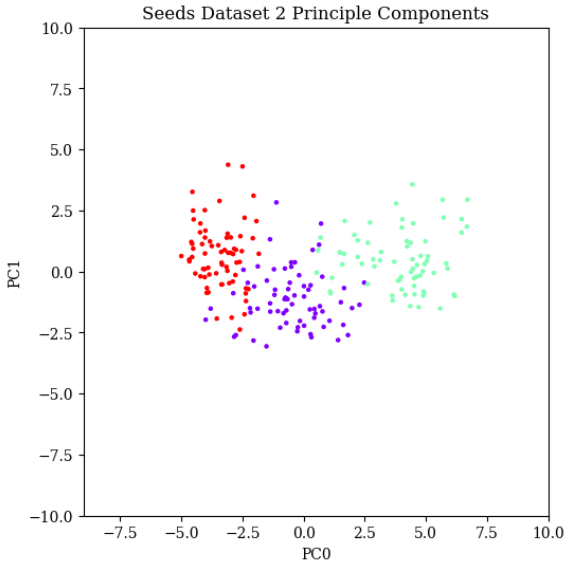


Figure 4.11: A scatter plot of the first two principal components of the Seeds dataset.

Same as above, two experiments are conducted. One with an increasing outlier count and one with increasing outlier distance. The method for adding the outliers and the parameter settings were the same, apart from k, which was set to 3 to mirror the fact this dataset has 3 clusters rather than 2. Figure 4.11 shows and preprocessed Seeds dataset. Figure 4.12 shows the trialled versions of the dataset with the increasing outlier presence. The x axis is PC1 and the y axis is PC2 (axis labels on the subplots were omitted to save space). Figure 4.13 shows the preprocessed Seeds dataset with the increasingly distant outliers. As before, the data was min-max normalised prior to applying clustering. Again, experiments were repeated 50 times.

Compared to the previous artificial dataset, the Seeds dataset has an extra cluster and appears to have a greater degree of overlap between the clusters. Furthermore, the cluster are irregular shapes.

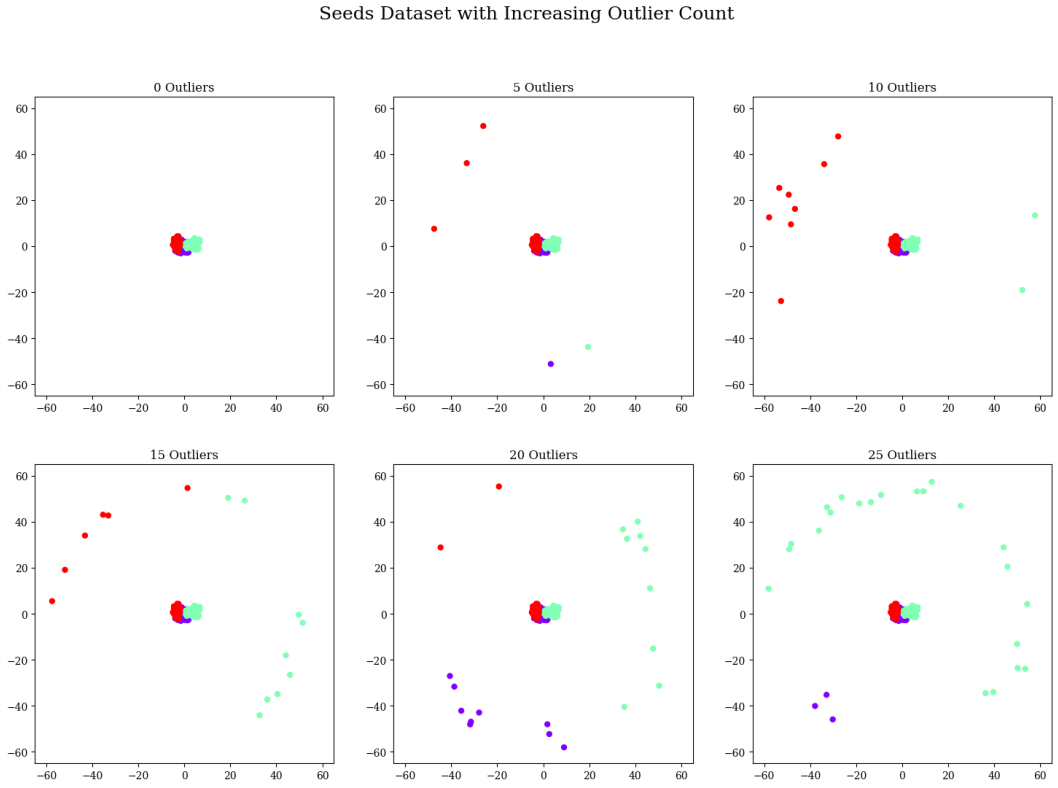


Figure 4.12: The preprocessed Seeds dataset, showing the increasing outlier presence.

Results and Discussion

Figure 4.14 shows the clustering performance of the three approaches. Again, LOFIWKM and ILOFIWKM are much more robust to outlier presence than k-means. Overall, the NMI scores are slightly less than with the artificial dataset of the previous experiments, but this is expected, as it can be seen that the clusters in the Seeds dataset are closer together and thus slightly harder to partition.

Figure 4.15 shows similar patterns to the experiment with the artificial data.

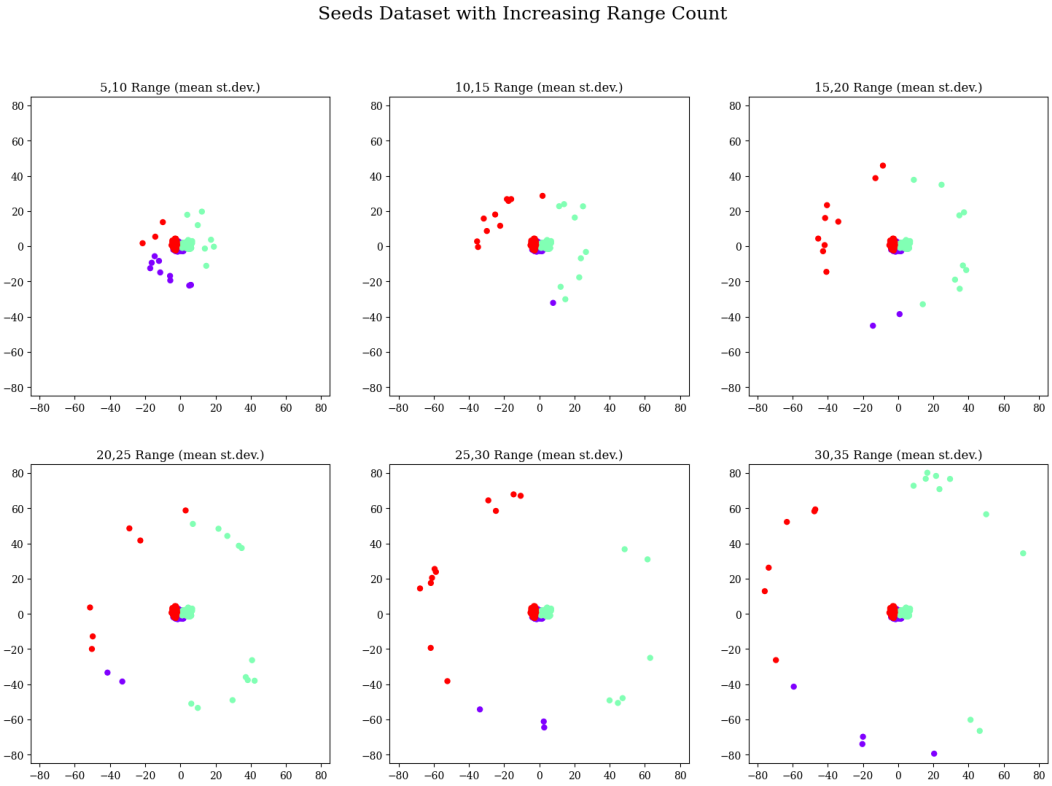


Figure 4.13: The preprocessed Seeds dataset, showing the increasingly distant outliers.

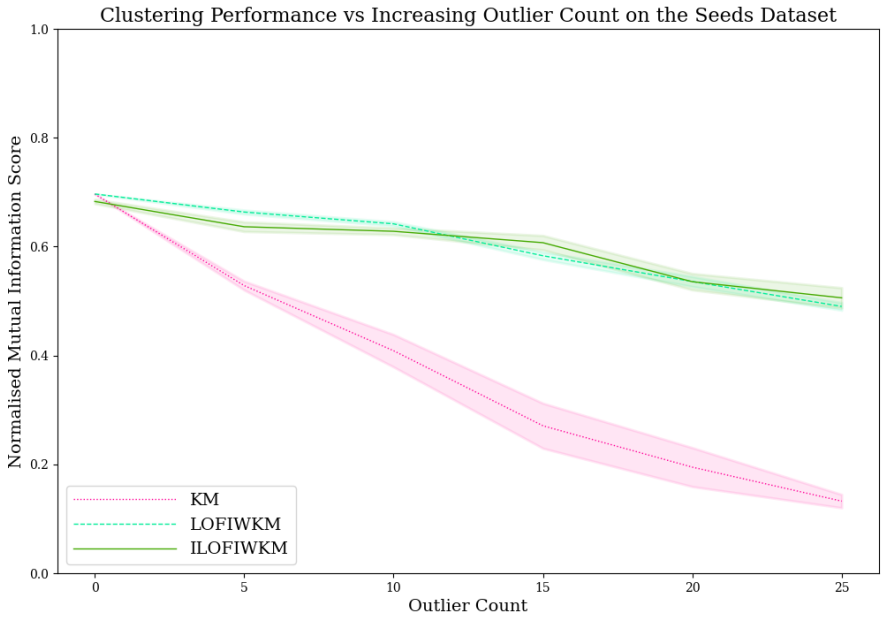


Figure 4.14: The average NMI score and standard deviation of k-means, LOFIWKM and ILOFIWKM on the Seeds dataset with an increasing amount of outliers.

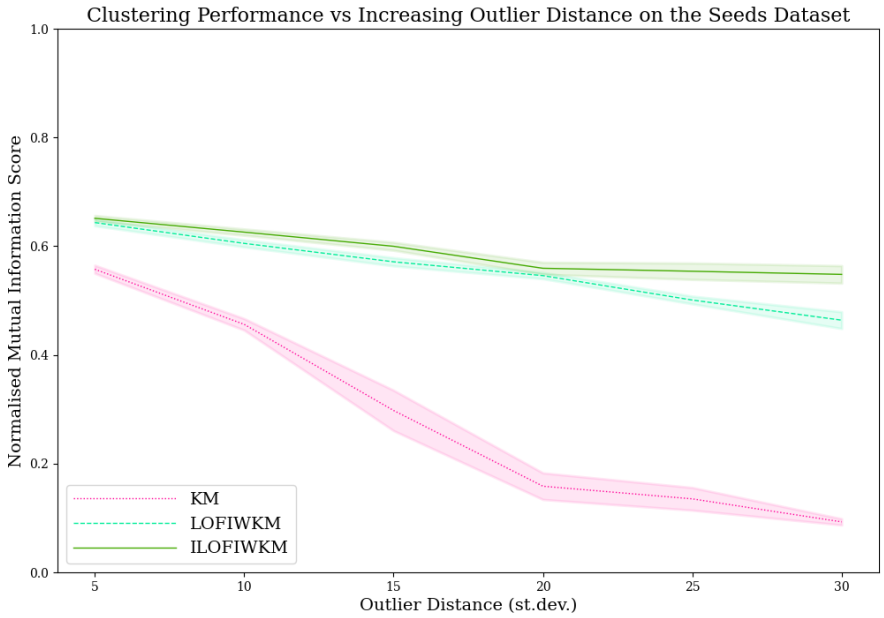


Figure 4.15: The average NMI score and standard deviation of k-means, LOFIWKM and ILOFIWKM on the Seeds dataset with increasingly distant outliers.

Again interestingly, the standard deviation (shown by the filled area around each line), is narrower in LOFIWKM and ILOFIWKM than for k-means. This indicates that the instance weighted methods converge more consistent than plain k-means.

Comparing LOFIWKM and ILOFIWKM, reveals a slight difference in performance when outliers are particularly distant (at more than 20 standard deviations). In Figure 4.15 and (to a lesser extent) 4.9, ILOFIWKM can be observed outperforming LOFIWKM. This could due to the slightly stronger weighting effect produced by ILOFIWKM as it normalises each distribution of LOF values per cluster. This suggests that implementing a parameter to adjust the impact of the weights could be beneficial. This could be added to the LOFIWKM avoid the computational cost of ILOFIWKM.

Alternative results using a different extrinsic clustering quality metric ARI (Adjusted Random Index), are also plotted and included in the Appendix in Figures, 9.1, 9.2, 9.3 and 9.4. They report identical findings but are included for completeness.

4.6 Conclusion

This chapter has demonstrated the extent to which instance weighted clustering can be effective for outlier accommodation, given a simple synthetic dataset. It has been shown a density-based instance weighting approach integrated k-means is able to maintain high clustering performance despite the presence of numerous and severe outliers.

Comparing against existing research, the LOFIWKM approach appears to be comparable to other methods and possibly superior in certain conditions. Comparing against Chen et al. and Yu et al., in their most similar experiments to mine they present an increase in accuracy of 16% and 19% respectively, compared to k-means. While my most similar experiments achieve greater gains compared to k-means. Although, it should be stressed that this is not a direct comparison, so more work would be required, utilising identical metrics and datasets to assert this claim.

Furthermore, an interesting secondary benefit is that the clustering performance is more consistent. Consistently good clustering performance is useful property, since this could reduce the need to use multiple initialisations of k-means.

In these experiments, there is not a consistent benefit to using the iterative version ILOFIWKM (which has a heightened computational cost) hence, it seems LOFIWKM is the most useful technique based on these results.
With the approach demonstrated on synthetic and simple benchmark datasets; the next step is to analyse and evaluate the approach on a more complex real-world dataset. This is attempted in the following chapter.

Chapter 5

Instance Weighting for Flight Data Recorder Clustering

5.1 Introduction

In the previous chapter, it was shown that the LOFIWKM algorithms can be effective for outlier accommodation. However, the experimentation was limited to synthetic and simple benchmark datasets. In this chapter, the investigation of this approach continues. The experimentation is extended to trial the approach on a real-world application which contains outliers. Furthermore, the experimentation aims to compare instance weighting to traditional instance selection methods. This is relevant since the typical approach to handle outliers in production data is to remove them and my approach is pitched as an alternative to this. The aim of this chapter is to support the findings of the previous chapter and investigate the validity of the LOFIWKM approach beyond simple artificial datasets. This investigation will help assert to what extent the instance weighting can accommodate outliers and hence support the answer to RQ1.

To achieve this aim, a Flight Data Recorder (FDR) dataset containing normal and abnormal flights (outliers) is selected. FDR data was chosen, as aviation and automotive systems are one of areas highlighted for further research by Ezugwu et al.'s survey on the application of clustering [5]. This is an interesting and important area to apply clustering which has received less attention than others. Specifically a recently published FDR dataset from National Aeronautics and Space Administration (NASA) was chosen as the real-world data for this evaluation. LOWIWKM will be compared alongside plain k-means and k-means with traditional (instance selection based) outlier removal methods. The results will be analysed to assert if LOFIWKM has any suitability beyond the previous chapters findings with synthetic datasets.

5.2 Related Work

Since the publishing of publicly available FDR datasets, initial literature searches show that a limited number of clustering techniques have been applied to these datasets. This could be because there are numerous challenges with clustering FDR datasets: *outliers, tightly-packed clusters, imbalanced cluster sizes, high-dimensionality.*

One of the most eminent initial works in this area is [42]. Li et al. used the FDR data for the purpose of detecting abnormal flights. This is a highly valuable activity to understand usage of airframes to support proactive maintenance of aircraft. Enabling the enhancement of their safety by reducing accidents due to mechanical failures. Note that mechanical failure accounts as the main cause

for 21% of fatal aviation accidents [1]. Li et al. proposes a new method called Cluster-Based Anomaly Detection (ClusterAD-Flight). Their approach is focused on using clustering techniques to detect abnormal flights. Their cluster analysis was performed on two proprietary FDR datasets one provided by an international airline company, which contained 365 Boeing B777 flights, and another provided by a European airline, which contained 25,519 Airbus A320 landings. Li et al.'s Cluster-AD method begins by transforming the time series data from the Flight Data Recorder (FDR) into high-dimensional vectors (each vector represents a single flight). Ultimately, this yields a high-dimensional dataset, so dimensionality reduction is conducted using PCA. This choice reduces the significant degree of the correlation between the features caused by the aforementioned transformation, and allows the majority of the information in the data to be retained while reducing the number of dimensions. For clustering Li et al. use the DBSCAN algorithm. This density-based clustering algorithm can identify both the clusters and outliers (which represent abnormal flights). The results showed that the Cluster-AD approach outperformed existing techniques at detecting operationally significant anomalies. This included outperforming exceedance detection (ED) the technique currently used by airlines. Through better identification of abnormal flights it is proposed that this could support more proactive safety management in the airline industry.

Similar to Li et al.'s work, Liu et al. also focused on detecting flight abnormalities [43]. Specifically, potentially hazardous "long-landings" of aircraft (when an aircraft touches down further along the runway than is ideal for safety). However, unlike Li et al. the purpose was not for enabling proactive mainte-

[1]Source: http://www.planecrashinfo.com/cause.htm

nance, rather, the purpose was real-time prediction for advanced warning of a long landing on the approach to touch down. Liu et al. used Quick Access Recorder (QAR) data from a commercial fleet of Boeing 737-800. The QAR is similar to the FDR, but provides more detailed data. Part of Liu et al.'s work involved clustering. A Gaussian Mixture Model (GMM) clustering method was used to cluster pilot behaviour during take-off and landing. For preprocessing, the QAR data was simplified to two dimensions, one being the ratio of actual take-off speed to rotation speed (lift-off speed), and the other the ratio of actual landing speed to reference speed (minimum speed that should be maintained during landing). Three clusters of pilots were identified and named "aggressive", "conservative" and "balanced". This clustering information was then used in their classification stage (using XGBoost) to consider the individual pilot's operating characteristics. It was found that taking into account both flight data and pilot behaviour (via the clustering) they were more able to predicting long landings than methods that only considered flight data, potentially enhancing landing safety. This work demonstrates that clustering can be a valuable tool for clustering different flying styles.

Unlike Li et al. and Liu et al., Wang performs clustering on individual flights (rather than many) [44]. Rather than clustering a flight as a whole Wang et al. clusters the epochs of the flight according to their risk of Loss of Control (LOC). To achieve this, Wang et al. purposes an approach called Flight State Deep Clustering Network (FSDCN). Wang et al. uses data which includes eight variables (Airspeed, Roll angle, Climb rate, Roll rate, Angle of attack, Pitch rate, Pitch angle and Yaw rate) in a time series. Wang et al.'s data is from a fighter flight simulator, and the variables recorded are similar to a subset of the data collectable from a FDR. Wang et al. tested their

FSDCN approach on data collected from three simulation flights containing high-difficulty manoeuvrer (such as loops, barrel rolls, s-turns, wingovers, and nosedives) to showcase the application of FSDCN for flight risk evaluation. Their complex approach is designed to extract hidden risk features from raw flight parameters. The FSDCN approach constructs a low dimensional feature space which is clustered using k-means into 5 flight states. Statistical analysis is then used to assign a risk level for each flight state. By comparing the alignment of the clustering results with the points where flight parameters (safe maximums) were exceeded, they asserted validity of FSDCN's clustering result.

The described literature shows a few different ways clustering analysis can be applied to aviation (specifically aeronautical) datasets. Li et al.'s work highlights how outliers (in terms of the quality of landings) exist, and demonstrates that DBSCAN can identify them. While Liu et al.'s work investigates differences between how pilots land aircraft. Both use very similar data of the same nature and source. Thus it is reasonable to theorise that the outliers that Li et al. describes, could present a hindrance to Liu et al.'s work, which is based GMM clustering. Li et al.'s work is an example of outlier identification, which conflicts with the purpose of my work which is outlier accommodation. Hence Liu et al.'s work on clustering types of pilot is more similar to this work. Although it is beyond the scope of this work to integrate classification into the analysis pipeline.

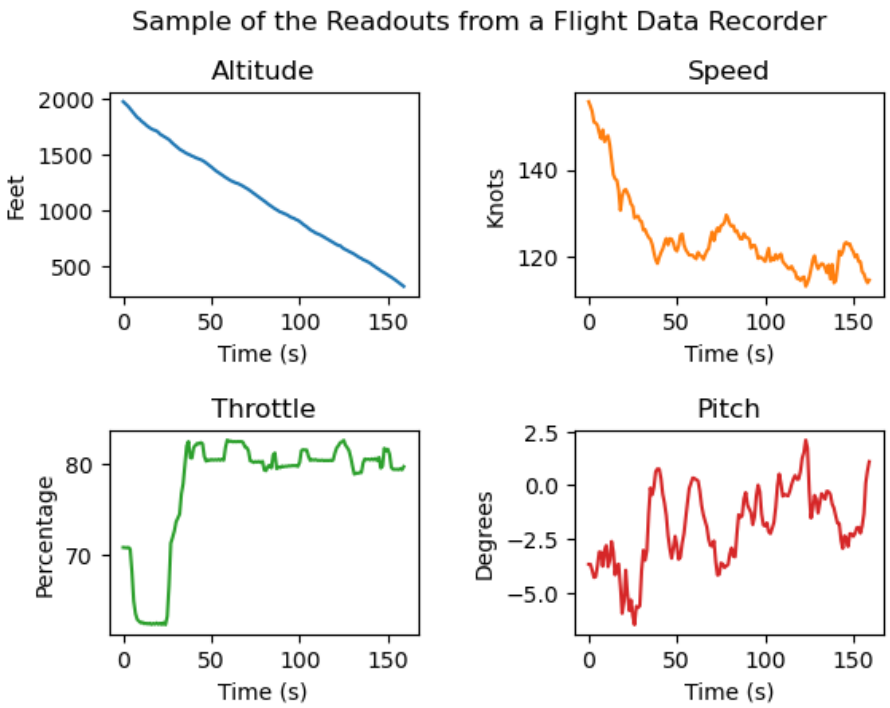


Figure 5.1: A sample of 4 the 19 features recorded by the FDR recorder for 1 of the 99836 flights in the dataset, the plots show the read-outs for the features for the last 160 seconds of flight (before touchdown).

5.3 Experimental Design

To conduct this investigation of the effectiveness of LOFIWKM the "Curated 4 Class Anomaly Detection Data Set" available at NASA DASHlink (**DA**ta mining and **S**ystems **H**ealth) [45] will be used. Figure 5.1 shows a sample of a single instance from the dataset.

Each instance in the FDR dataset represents the last 160 seconds of flight (before touchdown) and has 19 attributes recorded for each second. This means the dataset effectively has 3040 features. The dataset documents an impressive 99,837 flights of several commercial aircraft.

In this dataset the labels identify nominal landings and various types of ab-

normal landings. Naturally, abnormal landings could cause the airframe to endure additional stress and wear and which could lead to early fatigue of components. By identifying such abnormal landings, the servicing could be scheduled earlier as necessary to service the relevant components, before they begin to fail. While earlier work (using DBSCAN) on similar datasets, has shown how clustering can be useful in this application. Clustering was applied to recognise nominal flights and identify abnormal outlying flights, which can present in a number of ways, hence making clustering ideal for this task compared to classification [42].

Ideally, the classes of landing would be treated as cluster labels to compare against, this would allow the use of extrinsic clustering metrics, which offer a more objective comparison. However, this transpired to not be suitable for a few reasons.

When visualising the dataset compressed by PCA to 3 dimensions, see Figure 5.3. Firstly, the various types of anomalous flight labelled in the data are not clear spacial outliers (at least after applying PCA to the dimensionality). All the types of anomalous landings are no longer outlying from the nominal flights in the data. Thus making the LOFIWKM approach unnecessary. Secondarily, the both the nominal flights and various types of anomalous landing are not in cohesive clusters. Instead, the labels span several visually identifiable clusters.

Not reducing the dimensionality is not an option either, as my method relies on LOF which is a kernel-based density estimation and thus becomes increasingly ineffective as more dimensions are used.

While the clustering against the labels is clear not worthwhile investigating, the LOFIWKM approach can still be applied. There is an opportunity to analyse the visually identifiable clusters in the dimensionality reduced data. This can then be assessed using intrinsic metrics and visual assessment of the clustering results. While this does use the data not as intended it does allow the approach to trialled on real-world data in some capacity.

To pursue this aim, before applying the LOFIWKM clustering approach, several items of preprocessing were conducted.

Firstly, one can infer that the selected heading, selected course, true heading are not relevant landing quality (as they are subject only to the runway direction. Hence, these three attributes are removed. Similarly, total pressure and altitude are very highly correlated hence, the decision was made to remove total pressure. Next the data was flattened from a multi-array structure to a single vector per flight.

Then PCA was applied to reduce the dimensionality to 3 dimensions (to enable the visual analysis of the results). Figure 5.2 shows the scree plot. For the first 15 principal components. Taking the first 3 principal components only retains 20.2% of the explained variance in the dataset. Reducing the dimensionality this much is not preferable for optimal analysis of the dataset. However, testing revealed that using much beyond 6 dimensions is futile since the density-based instance weight calculation would be mostly ineffective when given so many dimensions. Figure 5.3 shows the resulting dataset.

Next sampling was applied to select a representative sample of the dataset.

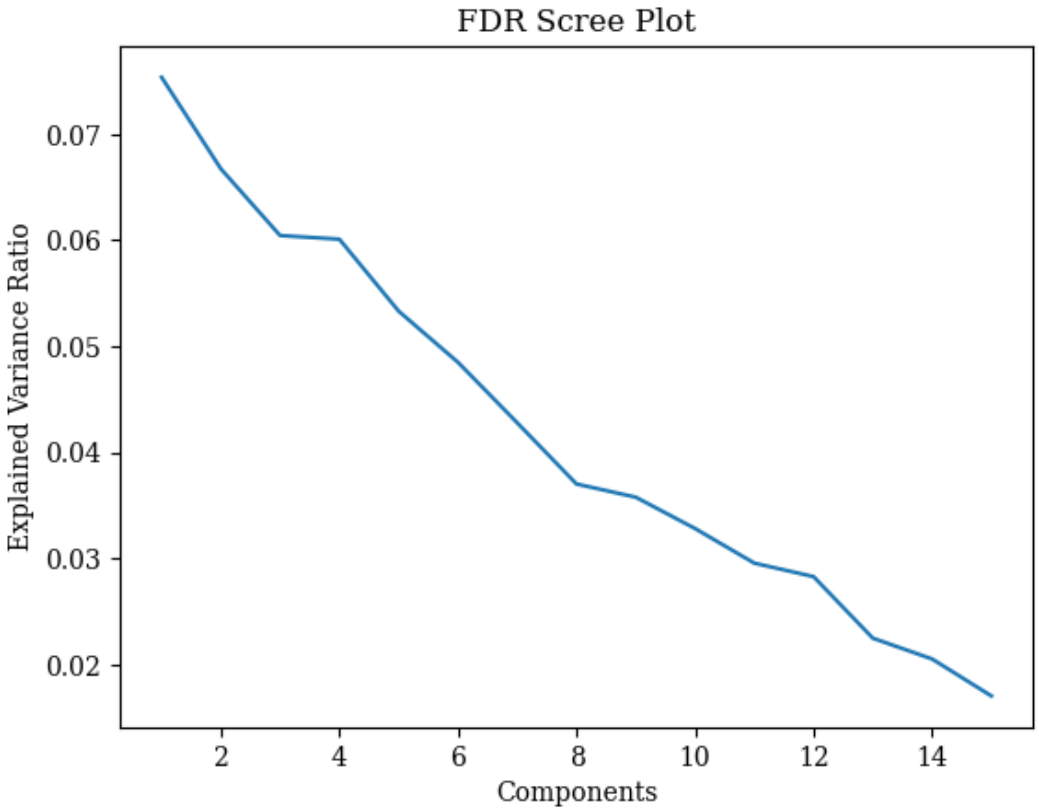


Figure 5.2: Scree plot showing the first 15 principal components of the FDR data.

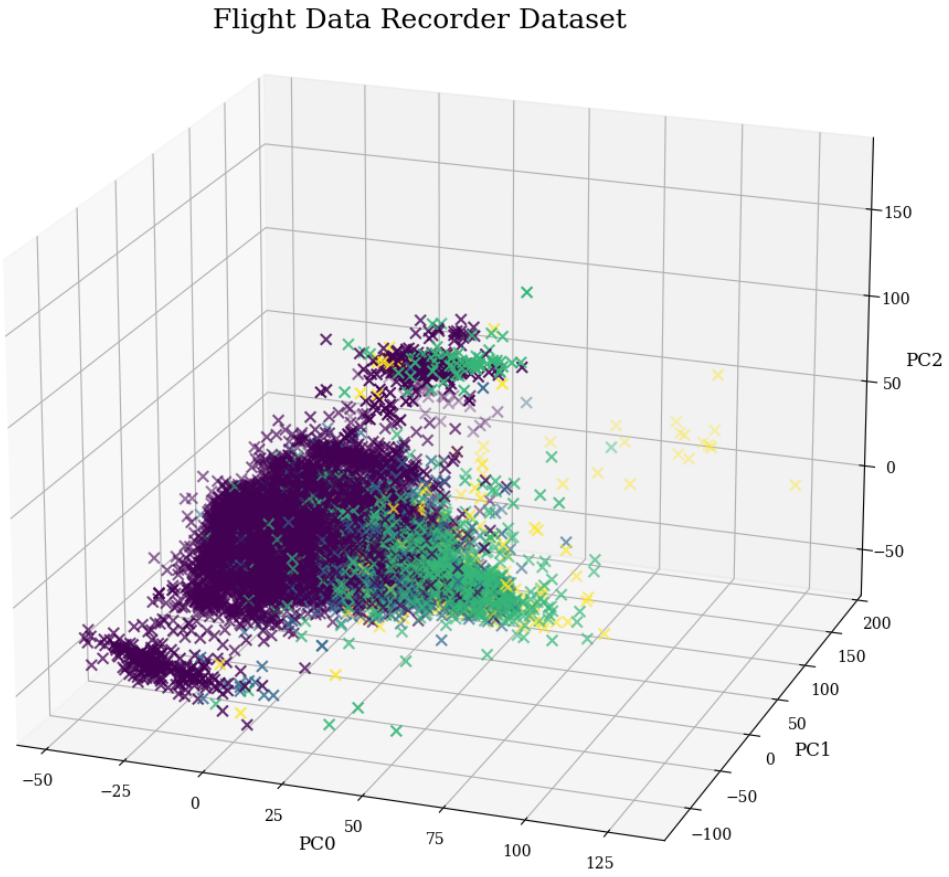


Figure 5.3: The Flight Data Recorder Dataset in 3 principal components. Purple "x" are nominal landings, and "blue", "green" and "yellow" various types of anomalous landings.

As using the entire dataset (99837 instances) would produce a too lengthy runtime. To create the sample, stratified sampling was applied across the 3 dimensions. The strata used equal-width bins of range 50, with 10 instances being randomly sampled from each (3-dimensional) bin. Using this approach approach retained all the visually perceivable clusters in the dataset, while reducing size the of the dataset to a size which the prototype implementations of the algorithms could perform promptly experiments on. Preliminary tests showed that on a sample of $\sim$900 instances k-means would take 13 seconds to execute, while LOFIWKM would take 16 seconds (on average). This would be too much, considering the plan to execute multiple conditions for many repetitions. Furthermore, it was necessary to use a stratified sampling approach in particular, as when using simple random sampling whole clusters would be omitted unless a large sample was taken. Figure 5.4 shows the stratified sample used in this research. The resultant sample contains 331 instances.

The elbow method was used to approximate the number of clusters. The k value of 4 was selected based on reviewing elbow plot and scatter plots.

Additionally, a version with outliers was produced. Outliers were added using aforementioned method. A total of 25 probable outliers were added at a distance of between 5 and 10 mean standard deviations. Figure 5.5 shows the resulting dataset.

To assess if instance weighting could improve performance, k-means was compared against the two instance weighted variants LOFIWKM and ILOFIWKM. For LOFIWKM and ILOFIWKM the parameter LOFnn (the count of neighbouring instances LOF uses to assess if a given point is an outlier) was set to 10.

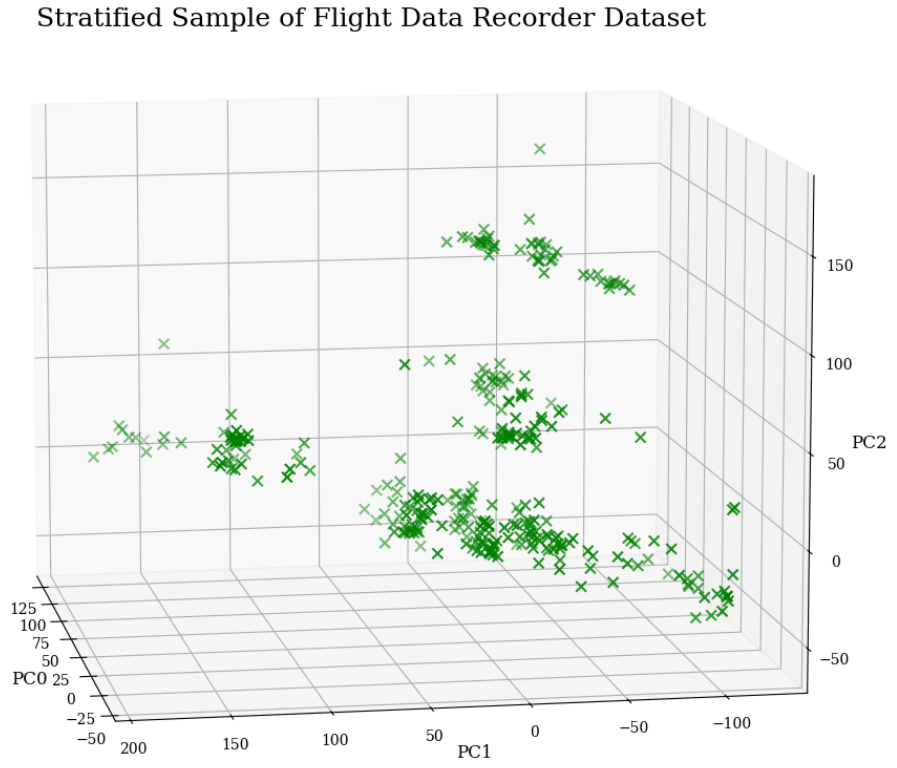


Figure 5.4: The Stratified Sample of the Flight Data Recorder Dataset in 3 principal components.

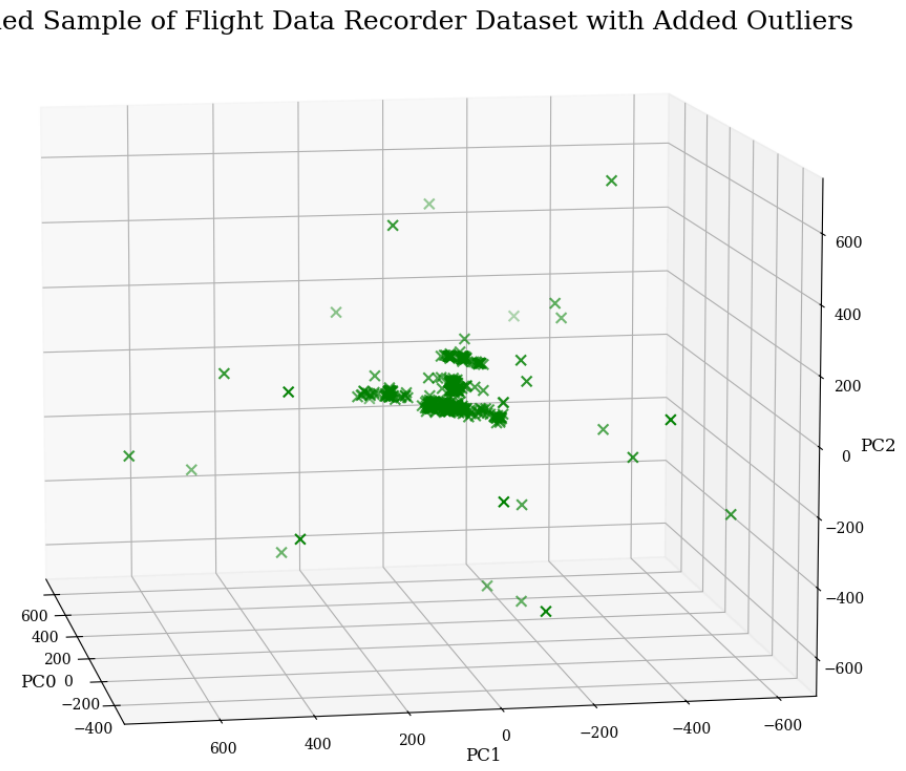


Figure 5.5: The Stratified Sample of the Flight Data Recorder Dataset in 3 principal components with added outliers.

Furthermore, instance selection methods were also trialled. As a most direct comparison, instance selection was performed using LOF (the same density-based outlier detection metric that LOFIWKM and ILOFIWKM use). This method calculates the "outlierness" of each instance and removes the most outlying instances. To avoid a straw-man comparison, three different levels of instance selection were tested, 5, 10, and 15 (instances removed). Furthermore, not exactly instance selection, but an interesting and traditional (and competitive) technique to compare against is Winsorization. This method does not remove instances, but instead confines the data to a user specified percentile range. Instances outside the range are reallocated to the boundary (like clipping signal processing). Again three different values were trialled, clipping data falling in the most extreme 5%, 10% and 15% of each dimension. Each algorithm was executed 300 times to get a reliable result, despite the stochastic nature of the algorithms.

Each algorithm was evaluated used intrinsic clustering metrics, this choice was made as while the dataset does contain labels, the labels denote anomalous landings rather than different modality/clusters/types of landings. Figure 5.3 shows the labels. Purple represents nominal landings, while the other colours represent high-speed, high-path and late-flaps landings. The intrinsic cluster validation methods used were Silhouette Coefficient, Calinski Harabasz Score, Davies Bouldin Score.

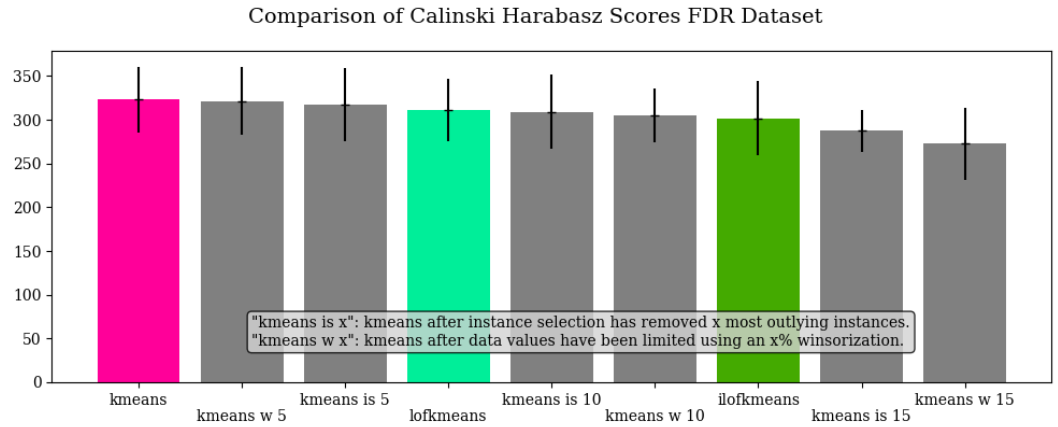


Figure 5.6: Calinski Harabasz scores for the k-means (pink), instance weighted k-means variants (greens) and traditional techniques (grey) on the sample of the FDR dataset.

5.4 Results and Discussion

Figures 5.6 to 5.16 show the results.

Figure 5.6, shows that k-means (pink) performs the best on average, while k-means with winsorization set to 15, performs the worst. The Calinski Harabasz score is the ratio of the between-cluster separation (BCSS) to the within-cluster dispersion (WCSS), normalized by their number of degrees of freedom. Hence the challenge with interpreting this result is that the Calinski Harabasz score offers no reward for ignoring outlying instances. LOFIWKM and ILOFIWKM (greens) both perform moderately compared to the other traditional methods (grey). Performing a two-sample Welch's t-test between k-means and LOFIWKM reveals a p-value of 4.206832×10^{-5}, as the p-value is below 0.05 this implies that they performed significantly differently.

In Figure 5.7, the average Calinski Harabasz score of the approaches are shown for the FDR dataset with added artificial outliers. Again, k-means is shown to perform best. Interestingly the standard deviation of k-means shows that

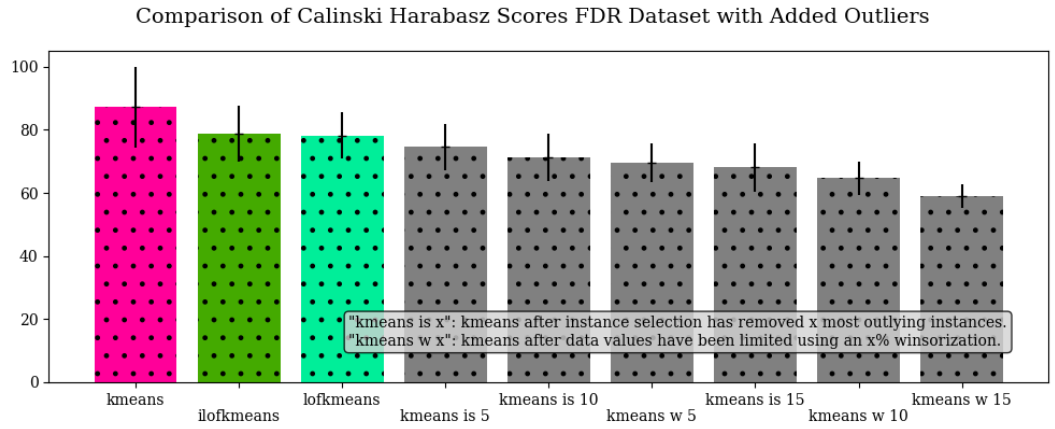


Figure 5.7: Calinski Harabasz Scores for the k-means (pink), instance weighted k-means variants (greens) and traditional techniques (grey) on the sample of the FDR dataset with added artificial outliers.

it performs more variably than the other approaches, matching the findings of the previous experiments. Again, performing a two-sample Welch's t-test between k-means and LOFIWKM reveals a p-value of 2.434058×10^{-23}, showing that there is a more significant difference between the algorithms when outliers are present.

Considering Figure 5.8, in this case k-means performs the best again (lower is better) with the instance weighted methods performing moderately. The Davies Bouldin score assesses the clustering by comparing the average similarity pairwise between the most similar clusters. The similarity is defined as the ratio between inter-cluster and intra-cluster distances. Again this metric has no mechanism for favouring the fitting of inlying instances. However, it takes the average of several measurements, so the result is somewhat more stable than Calinski Harabaz score. Performing a two-sample Welch's t-test between k-means and LOFIWKM reveals a p-value of 9.123486×10^{-16}, implying that these algorithms performed significantly differently.

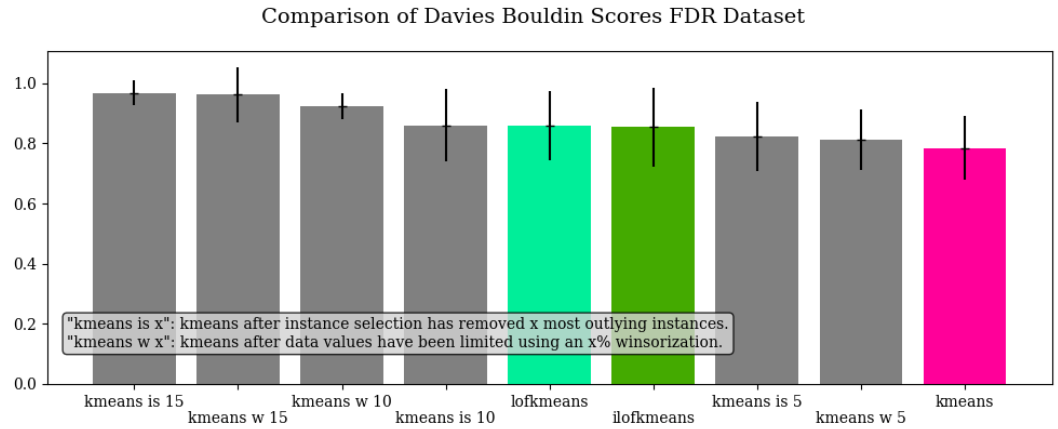


Figure 5.8: Davies Bouldin Scores for the k-means (pink), instance weighted k-means variants (greens) and traditional techniques (grey) on the sample of the FDR dataset.

Figure 5.9 again shows k-means as the best performing algorithm for the FDR dataset with outliers. Welch's t-test between k-means and LOFIWKM reveals a p-value of 9.764259×10^{-46}, again showing that there is a more significant difference between the algorithms when outliers are present.

Considering Figure 5.10, LOFIWKM is the most positive result. The Silhouette Coefficient varies from 1 (meaning the clusters are well separated) to -1 (indicating the clusters not well separated). The Silhouette Coefficient compares the average intra-cluster distance to the average inter-cluster distance. The Silhouette Coefficient still considers all instances equal, regardless of outlierness. However, due to it's design (using multiple averages), it is most stable when used to evaluate clustering with outliers. This metric is still biased against representing the benefit of outlier accommodation, but of the metrics compared, it is theoretically least biased. See Figure 9.11 in the appendix, for an example of the effect of outlier presence on the three intrinsic metrics. Performing a two-sample Welch's t-test between k-means and LOFIWKM reveals a p-value of 2.489144×10^{-17}, implying that they performed significantly

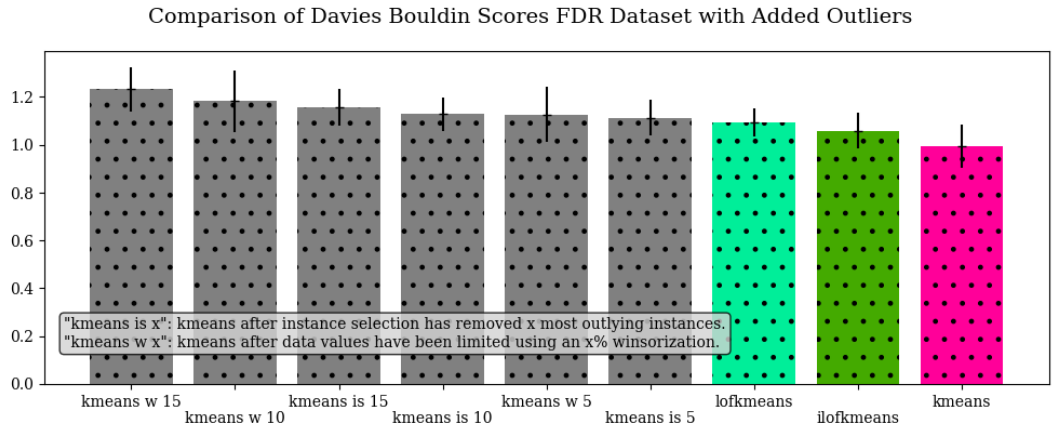


Figure 5.9: Davies Bouldin Scores for the k-means (pink), instance weighted k-means variants (greens) and traditional techniques (grey) on the sample of the FDR dataset with added artificial outliers.

differently.

Finally, in Figure 5.11 on the dataset with additional outliers, again k-means is shown as the best algorithm. Welch's t-test between k-means and LOFIWKM reveals a p-value of 8.17183×10^{-34}, further confirming a more significant difference between the algorithms when the additional outliers are present.

Overall, the different intrinsic metrics each tell a similar story (not forgetting that for Davies Bloudin lower is better). Without additional outliers, the instance weighted algorithms performs moderately. With the additional outliers, the instance weighted methods are second to k-means, which very anomalously performs significantly better.

A general trend across the results, is that k-means with instance selection, or winsorizing, are ordered both in terms of clustering performance and magnitude of their parameter. For example, k-means with instance selection removing 5 instances (referred to as "kmeans is 5" in the Figures) performs most

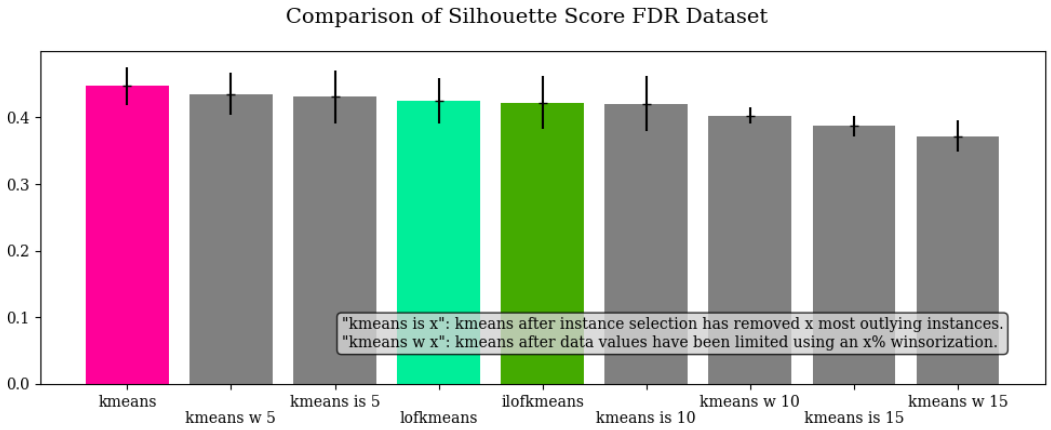


Figure 5.10: Silhouette Scores for the k-means (pink), instance weighted k-means variants (greens) and instance selection (grey) on the sample of the FDR dataset.

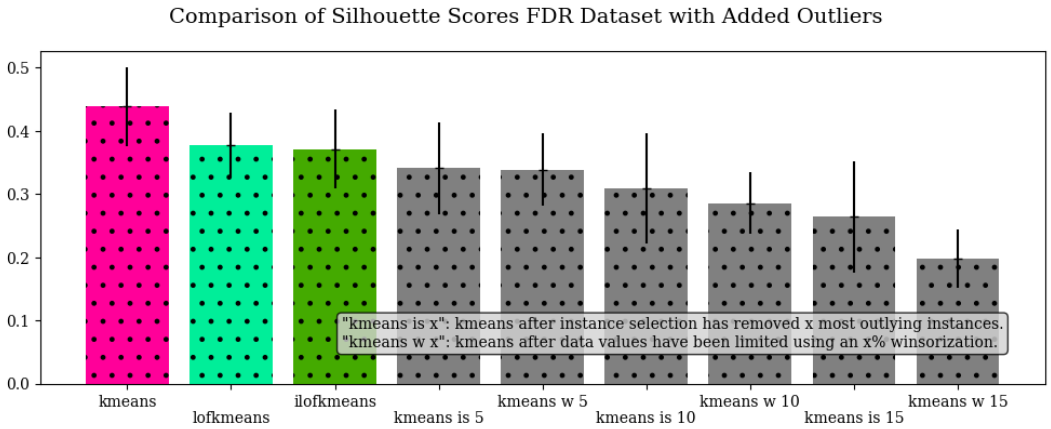


Figure 5.11: Silhouette Scores for the k-means (pink), instance weighted k-means variants (greens) and traditional techniques (grey) on the sample of the FDR dataset with added artificial outliers.

similar to k-means (bar the instance weighting). While "kmeans is 15" (removing 15 instances) performs worst according to the metrics.

Across all metrics, the differences are relatively small between the algorithms on the plain (non-outlier) FDR dataset. While on the FDR dataset with artificial outliers, the differences between the clustering algorithms is more significant. This suggests that the nature of the anomalies in the plain FDR dataset en-masse do not match outlier definitions used in the methods trialled (or at least given the preprocessing conducted).

LOFIWKM and ILOFIWKM routinely come second and third and it is tempting to conclude that instance weighting is better than instance selection based on these results, - but due to the limitations of the metrics, this is not a safe conclusion to draw. The metrics do not treat outliers exceptionally, and henceforth, the models which comprise to fully fit the outliers (at the expense of fitting the inlying data) are preferred. More experimentation would be required to assert instance weighting's superiority over instance selection.

Due to the limited suitability of the intrinsic metrics. Also, in lieu of writing my own intrinsic clustering quality metric (which would be akin to *marking my own homework*), Figures 5.12 to 5.16 allow of the visual inspection of the clustering results. In Figure 5.12, supposedly the best clustering results according to the intrinsic metrics is shown. However, upon inspection of the clustering result manually, it is clear to see that k-means has failed to successfully partition the data. Notice that one cluster (blue) is "expended" on fitting some of the outliers, leaving only 3 clusters to fit the four central clusters and the remaining outliers. It is fair to say that the blue cluster is not fitting the

central data or representing the outliers well and hence this is a in fact a poor clustering. Inspecting Figures 5.13 to 5.16 shows that other methods are not typically behaving in this way. The instance weighting and selection methods are fitting the central four clusters and are mostly capable of avoiding wasting centroids on fitting portions of the outliers.

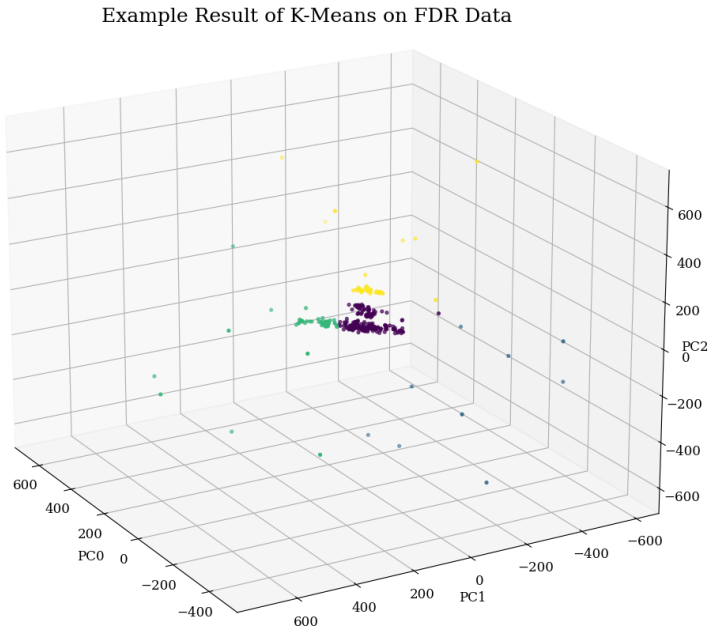


Figure 5.12: A scatter plot showing the clusters found by k-means on a sample of the FDR dataset.

To support this position a random sample of clustering results is plotted in Figures 9.5 and 9.6 in the Appendix. Upon inspection of Figures 9.5 and 9.6, it can be observed that k-means is quite consistently failing to fit the four inlying clusters in the dataset. It can be seen that k-means is confusing two of the visually identifiable clusters by using a cluster to fit some outliers (this is producing the better scores seen in the average intrinsic metrics – while in reality producing subjectively poor models). As a general trend, LOFIWKM and the traditional methods do not often expend a cluster on fitting the out-

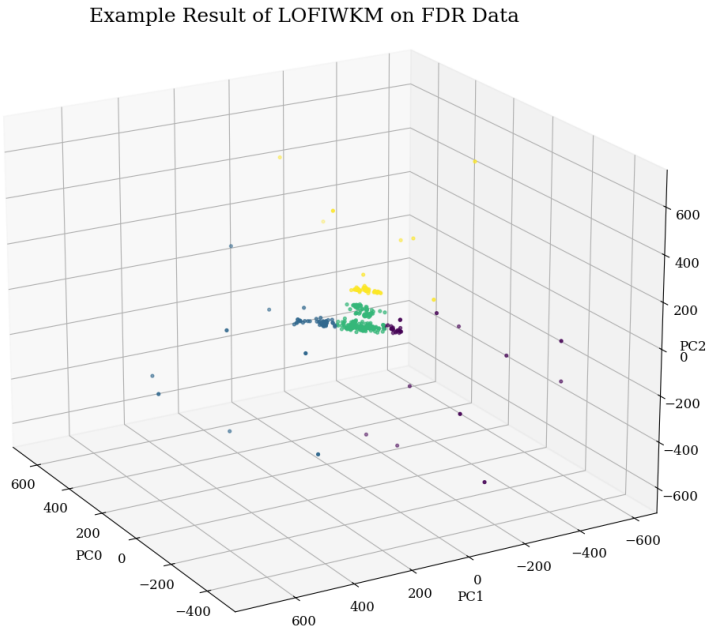


Figure 5.13: A scatter plot showing the clusters found by LOFIWKM on a sample of the FDR dataset.

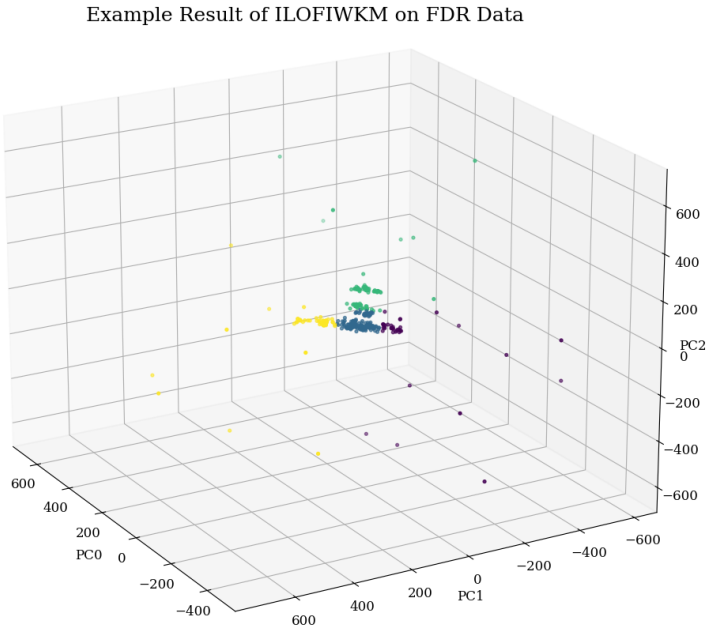


Figure 5.14: A scatter plot showing the clusters found by ILOFIWKM on a sample of the FDR dataset.

liers.

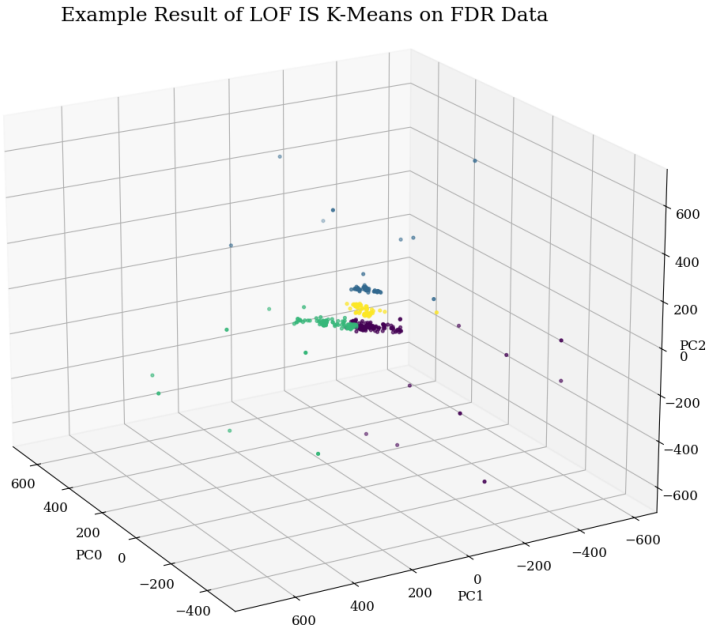


Figure 5.15: A scatter plot showing the clusters found by LOF based Instance Selection + k-means on a sample of the FDR dataset.

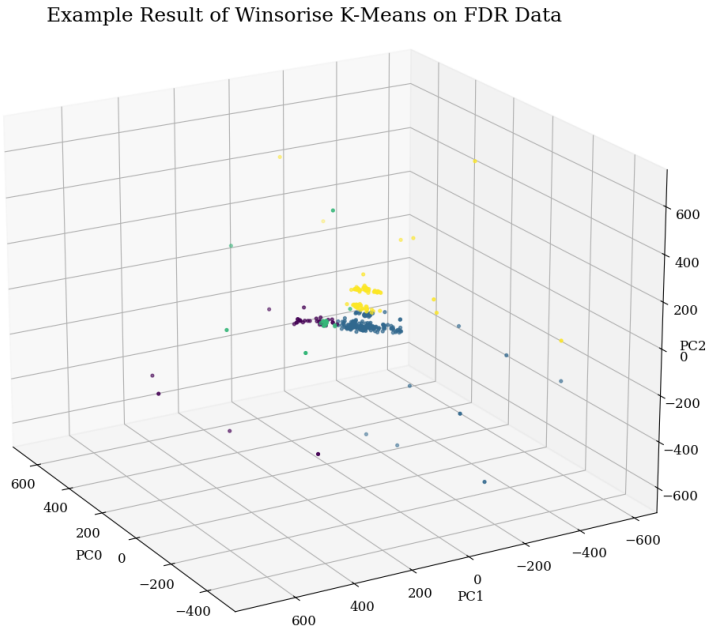


Figure 5.16: A scatter plot showing the clusters found by Winsorisation + k-means on a sample of the FDR dataset.

5.5 Conclusion and Future Work

In conclusion, this chapter has shown that instance weighting can help mitigate the effect of outliers (albeit artificially added outliers) on a real-world dataset in line with the aim originally stated. The instance weighting technique enabled the clustering process to spot real-world data clusters under conditions in which k-means clustering alone would often fail.

An important secondary finding is that the discrepancy between the visual findings and the results from the intrinsic metrics indicates the metrics are not suitable, or at least need to be interpreted differently, when assessing outlier accommodation approaches. This is due to the Calinski Harabasz Score, Davies Bouldin Score and Silhouette Coefficient all favouring fitting *all* the data points regardless of outlierness. A different way to interpret the results is looking at a high score as meaning "has fitted all the data including outliers" and a low score as "possibly has not fitted the outliers". This limitation of the intrinsic metrics make it difficult for this chapter to give a clear picture regarding how instance weighting compares to instance selection. It seems that novel metrics must be developed to confidently assess outlier accommodating clustering algorithms. Some work already exists in this space. For example, Density Based Clustering Validation (DBCV) [46]. However, DBCV is designed for clustering algorithms which identify outliers as part of their clustering output (not outlier accommodation). Thus this is suggested as an area for future work.

In this work, only k-means was investigated in conjunction with the LOF algorithm. However, there are likely more useful combinations. Hammerly and Elkan found that instance weighting did improve the performance of hard

membership function algorithms (i.e. k-means)[16]. But Nock and Nielsen's research suggests that instance weighting is more advantageous for clustering algorithms with soft membership functions such as fuzzy k-means[8]. A recommendation for future work should be to investigate soft membership function algorithms in conjunction with the instance weighting proposed.

My proposed method decreased the weight that outliers have on the clustering outcome. However, this is somewhat in contrast to "boosted" clustering. In boosted clustering more weight is given to poorly fitted instances – and poorly fitted instances would entail outliers. Hence, two questions emerge. 1. How effective is boosted clustering on data with outliers? 2. Would a combination weighting strategy which up-weights poorly fitted data points but does not up-weight outlying ones overcome this?

My modifications were made to a basic version of the k-means algorithm. However, it would be possible to combine the LOF instance weighting with a version of k-means which has more optimisations or is being used in conjunction with wrapper functions. Furthermore, with instance weighting there is the potential to simultaneously apply multiple instance weights and techniques which could increase robustness, applicability and/or accuracy further.

The time complexity of LOFIWKM is equivalent to LOF instance selection $O(n^2)$ plus k-means $O(n)$. However ILOFIWKM is significantly more costly, as it is the complexity of k-means plus the execution of the LOF algorithm per cluster per iteration. Thus it is apparent that ILOFIWKM may be not suitable for large datasets, without optimisation of the LOF algorithm, such as, the research by Alshawabkeh et al.[39]. However, ILOFIWKM shows little no

benefit over LOFIWKM, so a recommendation would be to prefer LOFIWKM for practical application.

Future work also includes testing the algorithms with a more thorough outlier generation process. In this work, instances were added from a uniform distribution, into an area that would make them outlying from existing instances (in terms of standard deviations). However, in the process of adding probable outliers, no approach was taken to distribute them from each other, aside from the randomness. A key observation is that adding true while random outliers (in terms of some definition) is non-trivial.

Currently, my algorithm requires parameter selection of k clusters and the size of the LOF neighbourhood. Other algorithms [9, 10] require some parameter selection exception for the state-of-the-art [47]. For simplicity, it would be better to not include parameters which must be selected and it does seem possible to automate the selection of the neighbour size. However, for flexibility, adding one extra parameter to adjust the impact of the weighting could be advantageous too.

This work investigates clustering flights (specifically the landing of flights), but an interesting area we did not explore is clustering epochs in the flight data, such as is done Wang et al. [44]. Wang et al. considers the risk levels (in terms of chance of LOC), this is a case of imbalanced clustering, since most of the time the risk level is low, and it is only during the apex of manoeuvres that the risk of LOC becomes high , making this an interesting area to explore for future research area around RQ2.

In response to RQ1, it is shown how instance weighting can be applied to k-means (a partitioning-based clustering algorithm) for outlier accommodation, on a real-world dataset. However, unfortunately, in expanding the experimentation to use a real-world dataset and comparing against instance selection, due to limitations of the approach it is not a conclusively positive result, due the comprises/limitation described in this chapter. Rather, the findings highlight directions for further development of the approach and experimental design.

Chapter 6

Instance Weighting for Ensemble Graph-based Clustering

6.1 Introduction

A graph-based algorithm that can handle clusters of arbitrary shape, but cannot robustly handle imbalanced clusters, is the spectral clustering algorithm [48]. In this chapter, this challenge is addressed through the application of instance weighting. The application of instance weighting is a logical choice, since it could be used to reduce the impact of the majority cluster(s), enabling an accurate partitioning. Furthermore, applying ensemble techniques is a proven approach to increase the performance and robustness of clustering [15, 49, 50, 51]. While ensemble clustering does have some disadvantages, such as; computational complexity, sensitivity to the choice of the generative mechanism, and added difficultly when explaining the results. However, when given computational resources and applied effectively, ensemble clustering can

be a very powerful tool. Furthermore, recent promising research and reviews have highlighted the potential of spectral clustering integrated with weighted ensembles [49, 52].

When designing a clustering ensemble, there are three key design decisions to be made. The first decision is the *generative mechanism*. This is the choice of how to generate an ensemble of base clusterings. Typically, the goal of the generative mechanism is to create both diverse and high quality base clusterings [53]. The generative mechanism has significant impacts on performance, and there is a variety of approaches that can be applied, including: different algorithms, different parameters [53, 54], different subsets of instances [50, 55], different subsets of features [51] or a combination of mechanisms [56]. Wu et al. points out that the generative mechanism plays an important role in creating high quality and high diversity base clusterings, which enables the ensemble to outperform clustering alone.

The second decision is the *consensus function* which defines how to combine the outputs of the base clusterings. Typically, the goal of the consensus function is to combine the base clusterings into a single clustering that is higher quality than any of the base clusterings. Boongeon and Iam-On [57] provides a taxonomy of four different types of consensus functions. **Direct approach,** this uses voting to determine the cluster membership of each data point. It is the least complex approach, however there is a need to solve the *label correspondence problem* as clustering algorithms assign arbitrary labels. **Feature-based,** this approach analyses the labels and finds a model describing the results. The model is then used to assign the instances, therefore there is no need to solve the label correspondence problem. **Pairwise-similarity,** this

type scans across all the results and counts the frequency that instances appear in the same cluster. To do this, a $n \times n$ similarity matrix is constructed. The similarity matrices are then merged to create a co-association matrix. Again, this avoids the need to solve the correspondence problem. Finally **Graph based,** this type of consensus function creates a graph from the clustering results. This graph can be constructed from the aforementioned co-association matrix. Graph partitioning methods are then applied to decide the final clustering. The idea of re-framing the problem in this way can be attributed to Strelh and Ghosh's seminal work [34]. They defined three consensus functions: Cluster-based Similarity Partitioning Algorithm (CSPA), Hyper Graph Partitioning Algorithm (HGPA) and Meta Clustering Algorithm (MCLA), which have been broadly adopted by cluster ensemble research since their definition. The simplest of these methods is CSPA. This consensus algorithm takes each co-association matrix from the base clusters and combines them to create a master co-association matrix which can then be partitioned using a graph clustering algorithm. In Strelh and Ghosh's research, the METIS clustering algorithm is used to cluster the resultant graph although any graph clustering algorithm could be used, such as spectral.

The third decision is the ensemble structure; *bagging* or *boosting*. Bagging executes the base clusterings in parallel whereas boosting executes the base clustering sequentially. The boosting approach has the advantage that information discovered by a base clustering can be used to influence the next iteration of base clustering. The bagging approach has the advantage that the base clusterings can run in parallel, which is a useful property given the ubiquity of multi-processor systems. Since, research [50] has shown that both approaches lead to successful clustering ensembles, and that support for distributed pro-

cessing is an increasingly desirable property, in this work the bagging approach will be the focus.

Cluster ensembles can be weighted in various ways, Zhang provides a taxonomy of methods for weighting clustering ensembles and highlights how this can enhance the robustness of the clustering [49]. One approach is weighting the base clusterings. This is where the base clusterings are assessed (typically for clustering performance), then weighted accordingly in the consensus function. Another approach is weighting the features of a dataset, such that each of the columns has more or less influence in the base clustering outcomes. A further approach is weighting instances. In this approach, weights are applied to the instances. Not much research has been done in this area leaving a gap that can be addressed. These instance weights can encode information that the clustering process can utilise to enhance the clustering performance. One such clustering algorithm that could benefit is the aforementioned spectral clustering algorithm. This algorithm is known to have reduced clustering performance when clusters are imbalanced [58]. By utilising the weights to emphasize the low density clusters this limitation could sometimes be overcome. In particular, I will focus on utilising density based instance weighting since this metric could reveal useful information about an imbalance in a dataset.

The combination of ensemble clustering and instance weighting is interesting. Since each independently is known to improve clustering performance and there exists research (see Chapter 6.2.2) that together they can further improve clustering performance. Furthermore, spectral clustering is a natural choice for researching since it is amongst the most versatile and actively researched clustering algorithms. Hence, the aim of this work is to investigate if

instance weighting can be applied usefully within the generative mechanism of a spectral clustering ensemble. This is an important objective since spectral clustering, ensemble clustering and instance weighting are three very promising areas of research within unsupervised learning.

This work presents some initial research into applying instance weights at the generative stage of a bagging spectral ensemble. Experiments with a prototype approach will be used to investigate whether a instance weighted sub-sampling approach based on density could enhance the robustness and clustering performance of spectral clustering for datasets with imbalanced cluster sizes.

The concept behind this approach is that the density based weighting scheme up-samples sparse clusters and down-samples dense clusters. Overall, the clusters would then present as more balanced to the spectral base clusterings. This could improve clustering performance. The aim of this chapter is to investigate if an instance weighted spectral clustering ensemble can to enhance robustness and clustering performance on imbalanced data, as per RQ2.

6.2 Related Work

6.2.1 Variations of Spectral Clustering

Here work modifying spectral clustering to enhance its clustering performance and robustness is explored.

In Nadler and Galun's research, the limitations of the spectral clustering algorithm are explored [59]. They make two contributions to knowledge. The first is demonstrating the limitations with spectral clustering. When spectral clus-

tering creates its affinity matrix for partitioning, there are several methods that can be applied, such as nearest neighbour or a radial basis function. These rely on local distance (typically Euclidean) information. Nadler and Galun identify the use of local information to create global clusters as the issue which impedes spectral clustering from accurately partitioning the data when the data contains "structures at different scales of size and density". Their second contribution is a method for assessing the quality of partitions, which can enhance the clustering performance on datasets with varied cluster sizes (in terms of instance count) and scales. Their coherence measure supports the normalised-cut partitioning within spectral clustering. Their measure works by assessing the top down partitioning of data, indicating if the partitions need further partitioning or not, enabling spectral clustering to better find clusters of different sizes. For example, in a dataset with three clusters (one cluster with many instances and two smaller clusters with fewer instances) their approach would first use spectral clustering to partition the data into two clusters. Under the right conditions the partitioning could separate between the large cluster and the two smaller clusters, then their metric would identify that the partition containing the two small clusters needs further partitioning and would repeat the spectral partitioning on that set to find all three clusters. Nadler and Galun has proven that their measure can be used to augment any graph-based clustering method. Nadler and Galun prove the suitability of their approach on a variety of challenging datasets.

Lucińska and Wierzchoń [60] propose a algorithm called Speculus. Their implementation uses a novel approach to construct the affinity matrix for partitioning. Rather than using a radial basis function or nearest neighbours as normal for spectral clustering, their approach is based on mutual nearest neighbours.

This works by first calculating the nearest neighbours. Then if two instances are both found to be nearest neighbours, then the sum of their ranks that they are nearest neighbours to each other is calculated, (for example $1^{st} + 3^{rd} = 4$) this becomes their "distance", otherwise if two instances are not in each others nearest neighbours then assign a high distance for example infinity or not simply connecting those pairs in the graph representation for clustering. Like [59] their method helps address the weakness of spectral clustering to handle clusters of different scales and sizes but rather than modifying the partitioning process, Lucińska and Wierzchoń's approach introduces a novel way to construct the affinity matrix.

Correa and Lindstrom [61] propose a local-scaled spectral clustering algorithm. Their research attempts to resolve the same problem as [59] and [60]. They propose a sophisticated novel affinity matrix construction. Correa and Lindtrom's research has developed an approach which estimates the scale across the dataset and embeds this information into the affinity matrix. Instead of using k-nearest neighbours approach to construct the affinity matrix (which they demonstrate is sensitive to the choice of k or ϵ (radius)) their approach called β-skeleton uses empty region graphs, which can be adjusted using a parameter β, which is not as sensitive to selection. Essentially, their method defines an empty region from which neighbours are not considered. A benefit is that it can represent clusters with fewer instances that are next to larger clusters. This is unlike k-nearest neighbours where a small cluster could appear as part of the large cluster due a poor selection of k or ϵ. Their method estimates scale (neighbourhood size) automatically using novel Gaussian kernel density estimation based method. Their results on a variety of 2D artificial datasets (Ellipsoids, Gaussian and Noise Rings) and benchmark datasets (Iris,

Wine, Breast Cancer, Ecoli etc.) show the robustness to parameter selection and the good performance in terms of NMI for benchmark datasets. Correa and Lindtrom's experiments on artificial datasets do not show results in terms of NMI for datasets with varied cluster sizes which makes it difficult to be sure of the performance of their method in the case of imbalanced clusters.

In summary, research finds that spectral clustering has some weaknesses in handling data within varying density and datasets with imbalanced clusters sizes. The approaches described have been able to mitigate the limitations of spectral clustering through either redefining the affinity matrix [60, 61] or modifying partitioning process [59].

6.2.2 Ensemble Methods

This subchapter presents research into how ensemble methods can enhance the clustering performance and robustness. In searching for literature, extra consideration was given to papers which utilised weighted methods. This part of the related work has been organised into three paragraphs. The first paragraph explores literature where spectral was not used as the base clustering algorithm and instance weighting was not applied to the generative mechanism. The second paragraph explores literature where spectral was not used as the base clustering algorithm and instance weighting was applied to the generative mechanism. Finally, the third paragraph explores literature where spectral was used as the base clustering algorithm, but where instance weighting was not applied to the generative mechanism.

In this paragraph, literature where spectral was not used as the base clustering algorithm and instance weighting was not applied to the generative mechanism,

is explored. Ayad and Kamal [62] propose a bagging based approach for combining the base clustering results in a clustering ensemble. Their work focuses on consensus functions [34]. Ayad and Kamal's work creates a modified co-association matrix, which is calculated by evaluating the shared neighbours of pairs of instances, using the Jaccard similarity coefficient. Their method called Weighted Shared Nearest Neighbors Graph can then be used in Strelh and Gosh's consensus functions. They use the supra-consensus concept (where CSPA, HGPA and MCLA consensus functions are executed in parallel) and choose the consensus function with the highest Average NMI. They empirically prove the effectiveness of their approach by using their consensus approach within a ensemble using a variety of clustering algorithms as the generative mechanism (k-means, graph partitioning with various distance metrics). They found that using their improved graph in Strelh and Ghoshs' consensus functions achieved a higher f-measure than the mean of the f-measure of the base clusterings. Their selection of clustering algorithms did include a graph partitioning algorithm, but did not include spectral clustering. They found that their approach could handle imbalanced datasets. While related to my work in several ways, unlike my research, their work applies weights to pairs of instances and focuses on the consensus function and does not use spectral clustering. Al-Razgan and Domeniconi [51] propose two bagging based ensemble methods, Weighted Similarity Partitioning Algorithm (WSPA) and Weighted BiPartite Partitioning Algorithm (WBPA). Both ensembles use the LAC (Locally Adaptive Clustering) algorithm as their base clustering algorithm. LAC is an iterative centroid based method developed by the authors, which applies weights to the centroid and features. Their LAC algorithm has a h parameter, this parameter controls the incentive to use more features for the sub-space representation. As the generative mechanism, they vary the h parameter. The

focus of research is on weighted consensus functions to combine the base clusterings. From the LAC algorithm each cluster has a weight and this is used in a weighted distance calculation to create a graph embedding of both the original data and clustering information. Similar to Strelh and Ghosh, Al-Razgan and Domeniconi use the METIS algorithm to partition their graphs for a consensus result. Their experimentation found their WBPA approach to be best. In the WBPA method they create $n \times k \times m$ matrix (where m is the number of bags) with the edge weights representing probability of membership with each cluster. This matrix is interpreted as a bipartite graph and partitioned using METIS algorithm [34] to give the consensus result. Their sub-space clustering approach has the advantage that it lessens the impact of the curse of dimensionality. They identify experimenting with using spectral clustering as the base clustering algorithm as an area of future work. Similar to [62] their work focuses on weighting the consensus function rather than the generative mechanism. Finally, Ren et al. [56] proposes three bagging-based ensemble clustering approaches that feature boosting-style weighting. The algorithms using weighted instances and are called, Weighted-Object Ensemble Clustering (WOEC), Weighted-Object Similarity Partitioning Algorithm (WOSP) and Weighted-Object Hybrid Bipartite Graph Partitioning Algorithm (WOHB). For disambiguation; they use the word "objects" to describe what my paper refers to as "instances". The approaches are designed with the goal of being robust to parameter settings. A bagging approach is used (i.e. base-clusters are executed in parallel) but difficult to cluster instances are given weight (as is typical in boosting). Difficult to cluster instances are identified by the co-association matrix of the base clusterings. More technically, where the normalised co-association matrix (normalised against the number of base clusterings) is 0.5 then there is a disagreement regarding whether these in-

stances are in the same cluster (represented by 1) or not in the same cluster (represented by 0). This is then mapped to a quadratic function with its peak at 0.5 to emphasise uncertain instances. These weights from the quadratic function are then utilised in the consensus phase. For WOSP, the weights are used to recalculate the cluster centres. Across all three approaches for the generative mechanism, Ren et al. created diverse base clusterings by executing k-means with random instance and feature selection. For instances not selected in the sub-sample for each base-clustering, these were assigned to the nearest cluster centre. The WOEC algorithm uses a weighted version of the MCLA algorithm proposed by [34]. The MCLA algorithm treats consensus as a graph clustering problem and uses the METIS clustering algorithm to make the consensus partitioning. Normally, MCLA treats all the instances equally when calculating the similarity graph between clusters. However, in their weighted version the instance weights are taken into account. Hard to cluster instances provide more impact on the similarity calculation. They found that their three algorithms worked equally well for their simulated and real-world datasets. A unique feature of Ren et al.'s work is hybridisation of the ideas of bagging and boosting. A key difference between Ren et al. and this work is that this work assigns weights to instances before any clustering happens whereas in Ren et al. approach, weights are assigned based on the base clusterings and then utilised in the consensus stage. A common factor between these papers is that work has been focused towards the consensus function. Instance weighting was applied, but to influence the consensus function, not the generative mechanism, leaving a gap in area of applying instance weighting the generative mechanism.

In this paragraph, literature where spectral clustering was not used as the base clustering algorithm, but instance weighting was applied to the generative

mechanism, is explored. Parvin et al. [50] conducts many experiments within the space of clustering ensembles and provides an overview of the research into clustering ensembles. While Parvin et al. experiments with both boosting and bagging based ensembles. Their research focused on mostly boosting, their literature review found that both boosting and bagging based approaches led to successful clustering ensembles. Their experiments showed that diversity can be created within the cluster boundaries of an ensemble, using sampling based techniques. For their generative function in their bagging based approach they used non-weighted sample based techniques. While for their boosting techniques they used a disagreement based weighting scheme where instances which were uncertainly partitioned were sampled more frequently in the boosting. Generative functions used k-means as the base clustering algorithm. For the consensus function several approaches were trialled. They found that use of MCLA consensus function resulted in the highest accuracy and NMI in their boosted clustering experiments. Additionally, they found that the CSPA consensus function performed robustly in a number of experiments using boosted clustering ensembles. For their boosting method they found smaller sample size (in region of 20%) along with approximately 10 base partitionings is a generally optimal choice for clustering performance. Their work also compared bootstrap and sub-sampling based approaches and found that sub-sampling is more suitable for accuracy and computing performance. Finally, while Parvin et al. remarked that bagging sampling methods lead to successful clustering ensembles, their experiments boosting, outperformed bagging for clustering performance. Their experiments offered a lot of insights for designing clustering ensembles. Although unlike my work their work focused on boosting and does not explore density based techniques for weighting. Duarte, Fred and Duarte [63] investigated instance weighted ensembles. Their work builds on the

bagging technique of Evidence Accumulation Clustering (EAC) [64]. Through combining weak-clusterers such as k-means, clusters of arbitrary shape can be recognised. EAC is similar to CSPA, in that a $n \times n$ co-association matrix is created and then clustering is applied to the co-association matrix to form the final clusters. In their paper, they propose three boosting based methods based on EAC, these are AdaEAC L (emphasizes low confidence), AdaEAC H (emphasizes high confidence) and AdaEAC U (Emphasizes low and high confidence). They experimented on a variety of artificial datasets and benchmark dataset including "Wine", "Iris" and "Breast Cancer" amongst others. For consensus, they tried using hierarchical clustering with average-linkage and single-linkage methods. Their experiments showed that AdaEAC L, emphasising the low confidence instances accompanied by average-linkage in the boosting process was best. From their experiments with different instance weighting approaches ("L", "H" and "U") they found that "L" worked best. In addition, to the above, Duarte, Fred and Duarte introduces some novel clustering validation measures, ANC (Average Neighbourhood Confidence) and ADNC (Average Dynamic Neighbourhood Confidence). The novel ANC approach compares the neighbourhoods of instances against the co-association matrix to validate if the assignment of the given instance is consistent with its neighbours. They point out that their ADNC validation measure would enable the ensemble to handle clusters of imbalanced sizes best. Similar to my work their approaches are designed to handle data of arbitrary shape. Similar to my work they cluster the co-association matrix to find the consensus clustering. However, different to my research they use boosting, weak-clusterers and a confidence based method for generating the instance weights. Frossyniotis, Likas and Stafylopatis [55] compares boosting and bagging while focusing on boosting. They tested their ensemble methods with k-means and fuzzy

c-means. Frossyniotis, Likas and Stafylopatis's boosted clustering method applies weights to instances over several iterations. They experiment with two different ways of defining the instance weights. A simple approach based on the membership degree to each cluster and more sophisticated approach based on membership and entropy. In each round of clustering the most difficult to cluster instances are upsampled in their weighted bootstrap. For the consensus function a weighted vote is used, therefore, the label correspondence problem has to be solved. Their experiments found that the boosting approaches produced better results than a comparable bagging approach. Their approach enabled their centroid based methods to better handle non-spherical data. Their research is similar to mine as instance weighting is applied for the purpose of sampling the data. However, rather than using a sub-sampling approach a bootstrap approach is used to sample the data and most notably their work focuses on centroid based clustering rather than spectral clustering. In conclusion, research which has utilised instance weighting has mostly focused on boosting type ensembles and has not explored density based weighting.

In this paragraph, literature where spectral clustering was used as the base clustering algorithm, but instance weighting was not applied to the generative mechanism, is explored. In response to the research (such as [50, 65]) highlighting that it is not just the quality of the base clusterings, but also the diversity that has an impact on the quality of clustering ensembles, Fern and Linn investigated the diversity/quality balance [53]. To investigate this, they created three algorithms to select base clusterings which emphasised both quality and diversity. They confirm that concise ensembles of diverse and accurate base clusterings produced the highest NMI. The approach that achieved the best overall performance amongst those they tried, they named "Clus-

ter and Select". The method clusters the clustering results into a number of groups. Then from each group the highest quality clustering is taken. A strength of this approach is that the selection means that it is unlikely that any two of the base clusterings are similar. In their implementation they use the k-means algorithm as the base clustering algorithm and for the generative mechanism they execute k-means with different initialisations, feature subsets, space projections and different k values between 2 and $2 * c$ (where c is the number of excepted clusters). To combine the resultant ensemble they used the CSPA [34] as the consensus function. However, like my work rather than using the METIS clustering algorithm to cluster the co-association matrix they use spectral clustering. It was found that this approach produced robust clustering performance (in terms of NMI), despite degenerate partitions amongst the base clusterings. Fern and Linn do not use instance weighting, but it could have been added to their method as an additional or alternative method to achieve diversity amongst the base clusterings. In one experiment they add spectral clustering to their base clustering algorithms (hence the inclusion of their research into this section of this literature review). To generate diversity in the executions of the spectral clustering algorithm, they vary the parameters used to calculate the affinity matrix, in particular they use a Gaussian kernel to calculate the distances and vary the bandwidth parameter. Similar to the majority of research into clustering ensembles, their work focuses on consensus functions. Huang et al. [54] propose two algorithms U-SPEC (Ultra-scalable spectral clustering) and U-SENC (Ultra-scalable ensemble clustering). Their focus is adapting spectral cluster's scalability and robustness when working with Big Data. This is an important problem since an issue with spectral clustering is that the affinity matrix is $n \times n$, therefore, this can be unmanageable when using larger datasets. Hence, at the core of the U-SPEC method is data

reduction. Their approach hybridises random selection and k-means to produce a smaller dataset. In the "hybrid representative selection" of Huang et al. a random sample of the dataset taken, then k-means is applied to this sample (with a k value much higher than the number of clusters in dataset). The resultant centroids are used as the "representative samples" for clustering. This is supported by an $n \times p$ matrix, where p is the number of representatives. These refer back to the full dataset (by being interpreted as a bipartite graph). In U-SPEC a modified version of spectral is then used to partition the constructed bipartite graph. The U-SENC algorithm is an extended ensemble version of U-SPEC. It utilises multiple U-SENC base clusterings. Their ensemble method generates diversity between the U-SPEC base clusters both by the inherent randomness in the U-SPEC algorithm and running U-SPEC with different k values. The consensus function is based upon stacking the k eigenvectors and applying k-means. Their sophisticated approach enables spectral clustering like performance on ultra large datasets. This is proven by experimentation on datasets with as many as 20 million instances. Finally, most similar to my work, Jia et al. [65] propose a bagging ensemble approach using the spectral clustering algorithm. Their state-of-the-art approach involves creating diverse spectral base clusterings. This is achieved using two main methods. One is a random scaling parameter the other, random sampling in a Nyström approximation of the affinity matrix. The Nyström approximation a very rational choice, as it avoids the computation required to calculate the full $n \times n$ affinity matrix (by approximating it using a sample) while also introducing some useful randomness/diversity. Also, the choice of a random scaling parameter is interesting since, in attempts to improve spectral outside of ensemble research, a non-random scaling parameter was utilised [60, 61]. Similar to [51], there is a degree of selectiveness when combining the base clusterings. Unlike

[51], their selection process is binary rather than weighted. Jia et al. refers to Strehl and Ghosh's work [34] finding that of their three consensus methods CSPA and HPGA produced good accuracy. Although they opted for HPGA due to the computation complexity advantages. Their work is similar as their contribution is in the area of generative mechanisms and they adopt spectral clustering for the base clustering algorithm. The uniqueness of my work is that my approach uses instance weights based on density rather than Nyström approximation. It is clear that their approach would have performance advantages on larger datasets. It is possible my approach could perform better as diversity is more deliberately introduced rather than relying on by-product randomness from Nyström approximation.

In summary, this subchapter presents research into how ensemble techniques have been applied to clustering. The initial work in this area is promising. Furthermore, several works show how instance weighting has been utilised in boosting ensembles and applied in the consensus function. However, despite the search placing an emphasis on spectral ensembles and instance weighting, to the best of my knowledge, there appears to be no research into instance weighted spectral clustering ensembles for creating diversity in the generative mechanism, see Table 6.1.

Table 6.1: Overview of ensemble clustering research with similarities to my work highlighted, notice that no work combines spectral, bagging, instance weighted generative mechanism and CSPA.

Author and Citation	**Year**	**Clustering Algorithm**	**Ensemble Type**	**Generative Mechanism**	**Consensus Approach**
Ayad and Kamal [62]	2003	Several not including spectral	Bagging	Multi-algorithm to create diversity	Supra-consensus using CSPA, HGPA and MCLA on weighted shared nearest neighbours graph.
Frossyniotis, Likas and Stafylopatis [55]	2004	k-means and Fuzzy c-means	Boosting	Instances weighted based on distance to centroids	Weighted Vote
Al-Razgan and Domeniconi [51]	2006	LAC	Bagging	The number of features for the sub-space clustering is differed to create diversity	Weighted Graph + METIS Algorithm
Fern and Linn [53]	2008	k-means + (briefly spectral)	Bagging	Varying the k value	CSPA acting on select subset of base clustering results to emphasise diversity
Jia et al. [65]	2011	Spectral	Bagging	Random Scaling Parameter to create diversity and Nyström approximation of affinity matrix	HPGA
Duarte, Fred and Duarte [63]	2013	k-means	Boosting	Instances weighted based on disagreement	Linkage based methods + a selection other measures
Ren et al. [56]	2013	k-means	Bagging	Diversity via random subsampling and feature selection (with label propagation)	Weighted MLCA + instance weighted techniques
Parvin et al. [50]	2013	k-means	Bagging + Boosting	Instances weighted based on disagreement	Linkage based methods + hyper-graph methods (CSPA, HPGA, MCLA)
Huang et al. [54]	2020	U-SPEC (based on spectral)	Bagging	Inherent randomness in and differing k values to create diversity	k-means on combined eigenvector information

6.2.3 Conclusions from Related Work

Overall, the weaknesses of spectral clustering have been approached in different ways. One way which shows particular promise is ensemble techniques, offering significant improvements in clustering performance across a range of datasets. Ensemble clustering research has mostly focused on boosting based techniques and weighting the consensus function. But bagging is shown to result in good clustering performance too. Furthermore, techniques applying instance weighting to the generative mechanism have only have considered boosting ensembles and disagreement based metrics to the best of my knowledge.

Another conclusion, is that due the nature of clustering, boosting-based ensembles require additional complexity. It could be argued that bagging suits the nature of the clustering problem better. Bagging also has the advantage that it is simpler and it can be executed in parallel on large distributed compute resources. Enhancing its suitability for application to Big Data. When using bagging, a choice between bootstrap and sub-sampling arises. Parvin et al. found that sub-sampling performed favourably over bootstrap for clustering ensembles [50], additionally sub-sampling has compute advantages, so this approach will be adopted in this work.

Furthermore, within bagging based clustering ensembles there are two important design decisions, the generative mechanism and the consensus function [50]. For the generative mechanism it appears that an approach which generates a small, highly diverse ensemble of high quality clusterings produces the best results [53]. For the consensus function, CSPA combines the base clusterings by calculating pair-wise similarity between the cluster memberships in

base clusterings for all instances. This approach has the advantage of being both simple and producing robust performance [50, 65].

Finally, it is known that not all instances in a dataset represent equal information. It also know that data can have structures which can impede the performance of clustering and that spectral clustering is no exception [59]. Hence, the objective of this work is to use instance weighting within an spectral ensemble context to generate diverse base clustering to address the weaknesses of spectral clustering.

6.3 Proposed Approach

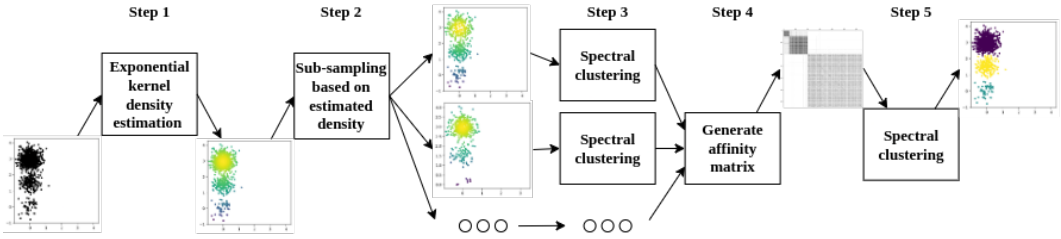


Figure 6.1: Schematic illustration of the IWSE approach.

The first step calculates a weight for each instance, based on an exponential kernel density estimation function using Equation (6.2), the bandwidth value, h is chosen using the Silverman method. The Silverman method here refers to the multivariate generalisation of Silverman's method, as implemented in scikit-learn[1]. In Equation (6.1), n is the number of rows and m is the number of columns of the dataset for clustering.

[1] `https://github.com/scikit-learn/scikit-learn/blob/c5497b7f7/sklearn/neighbors/_kde.py#L222`

$$h = n \times \left(\frac{m+2}{4}\right)^{\frac{-1}{(m+4)}} \tag{6.1}$$

The weights from the exponential kernel are min-max normalised between 0 and 1, using Equation (6.3). Furthermore, an additional inverted version of the weights is created where a high value indicates low density, seen in Equation (6.4). Within Algorithm 3, the exponential kernel density estimation function can be seen, where n is the number of instances, y is an instance to calculate the weight of, x is each of the instances, and h represents the bandwidth parameter used to determine the weights. In the second step, the approach uses bagging. Bagging was chosen for its computational advantages. When using bagging, a choice between bootstrap (sampling the dataset *with replacement* to get a sample of equal size to the original dataset) and sub-sampling (sampling the dataset *without replacement* to get a smaller dataset) arises. Sub-sampling was chosen as it has been shown to perform favourably for clustering ensembles [50]. The algorithm generates instance weighted sub-samples of the dataset, randomly sized between two user configurable parameters m_{min} and m_{max}. These are executed in parallel (given the appropriate hardware), corresponding to the number of bags parameter M. For example, $M = 32$ would indicate 32 sub-samples are created. For experimental purposes, there are three variations of the weighting scheme. In the "L" variation the low density instances are more likely to be selected. In the "H" variation the high density instances are more likely to be selected. Finally, the "U" variation uniformly randomly switches between the "L" and "H" weighting schemes. In the third step, normalised spectral clustering [66] is utilised for the base clustering of the sub-samples. In the fourth step, once all spectral base clusterings have been executed, consensus takes place. The pair-wise similarities of the

cluster assignments of the instances are accumulated into a co-association matrix. This approach is essentially CSPA [34]. However, instead of using METIS clustering algorithm, (typically used in the CSPA consensus function), spectral clustering is used to produce the final clustering output; this substitution was made because both are similar (in that they are graph partitioning methods), but spectral is more readily available in well-tested libraries. The approach based on CSPA was chosen as it is simple and produces robust performance [50, 65]. Thus in this final step, this co-association matrix is treated as an affinity matrix to which spectral clustering is applied. This provides the final clustering result. Algorithm 3 and Figure 6.1 provide a technical description of my approach. A Python implementation is provided in the Appendix.

$$\rho_K(y) = \sum_{i=1}^{n} exp\left(\frac{-dist(y, x_i)}{h}\right) \tag{6.2}$$

$$\mathbb{W} = \frac{\rho_K - min(\rho_K)}{max(\rho_K) - min(\rho_K)} \tag{6.3}$$

$$\mathbb{W}^* = 1 - \mathbb{W} \tag{6.4}$$

6.4 Experimental Setup

The experiments are arranged as follows, in Experiment A the proposed approaches parameters and suitability for imbalanced data is evaluated on simple dynamically generated artificial datasets. In Experiment B, an initial investigation will be made into the real-world suitability for a image segmentation use case. Finally, in Experiment C, proposed approach is evaluated on artificial and benchmark datasets from other works.

Algorithm 3 Instance Weighted Spectral Ensemble (U)

Input: $\mathbb{X} = \{x_1, ..., x_n\}$, k, k^*, M, m_{min}, m_{max}

Calculate bandwidth value h using Silverman method.
Compute weights $\mathbb{W}$ using $\rho_K(y) = \sum_{i=1}^{n} exp\left(\frac{-dist(x_i,y)}{h}\right)$ for $\mathbb{X}$
Normalise weights $\mathbb{W}$ using equation $\frac{max(x)-min(x)}{x-min(x)}$
Compute inverted weights $\mathbb{W}^*$ using equation $1 - \mathbb{W}$
for $m \leftarrow 1$ *to* M **do**
    Let $r \in \{0, 1\}$ with uniform probability (for switching weighting schemes)
    Let $s \in \{x \in \mathbb{R} | x \geq m_{min}$ *and* $x \leq m_{max}\}$ with uniform probability
    if $r = 1$ **then**
        Let $\mathbb{S} \subset \mathbb{X}$ be a sub-sample of size $n \times s$ using probability $\mathbb{W}$
    else
        Let $\mathbb{S} \subset \mathbb{X}$ be a sub-sample of size $n \times s$ using probability $\mathbb{W}^*$
    end if
    Partition $\mathbb{S}$ into $\mathbb{P} = \{C_1, ..., C_k\}$ using spectral with k and k^*
    Construct $n \times n$ co-association matrix $\mathbb{A}_m$ for $\mathbb{P}$
end for
Let $\mathbb{A}^* = \sum_{m=1}^{M} \mathbb{A}_m$
Partition $\mathbb{A}^*$ into $\mathbb{P}^* = \{C_1, ..., C_k\}$ using spectral with k and k^*

Output: $\mathbb{P}^* = \{C_1, ..., C_k\}$

6.4.1 Experiment A

Setup

To empirically evaluate IWSE, datasets with increasing cluster imbalance were generated. The 2D datasets are generated per run of the experiment to minimise effects from artefacts of the stochastic generation process. Each dynamically generated dataset contains three clusters drawn from normal distributions positioned at (0,0), (0,1.5), and (0, 3), and each cluster has a variance of 0.1 in both x and y dimensions. Cluster sizes were determined using a scaling factor α with values from 1 to 5 in increments of 0.25 where $|C_0| = 50, \quad |C_1| = |C_0| \times \alpha, \quad |C_2| = |C_1| \times \alpha$. So, when $\alpha = 1$ then $|C_0| = 50$, $|C_1| = 50$ and $|C_2| = 50$. When $\alpha = 2$ then $|C_0| = 50$, $|C_1| = 100$

and $|C_2| = 200$. A sample of the datasets can be seen in Figure 6.2.

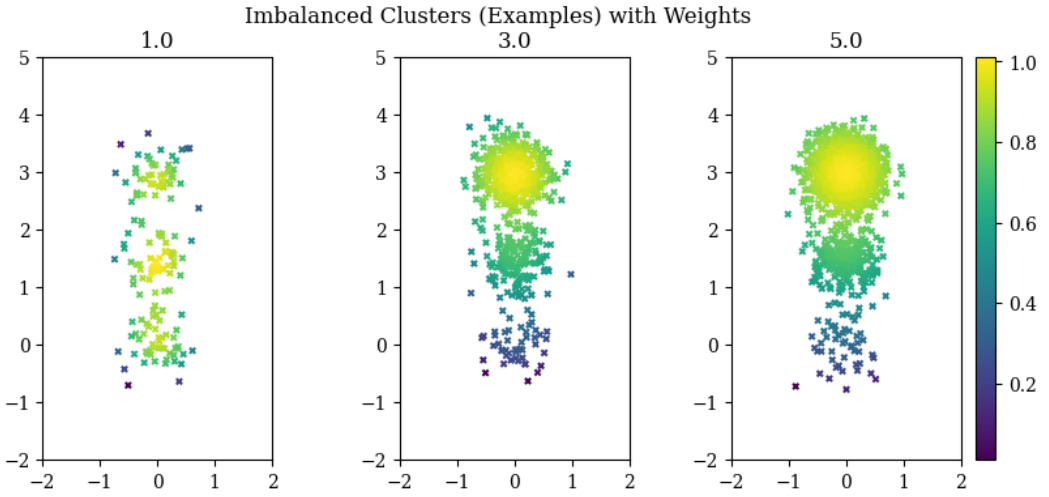


Figure 6.2: A sample of the "imbalance" datasets, the colouration represents the instance weights. The title for each sub-plot shows the "imbalance scaling factor".

For this experiment, multiple versions of the IWSE approach were compared against spectral (S) and spectral ensemble (SER). SER is similar to IWSE in every way, apart from using purely random sampling rather than weighted sampling. Comparing against SER will be useful to see if the instance weighting has any specific benefit. The "H" version (IWSEH) samples preferring the high values from the exponential kernel density function, this means instances in high density locations are more likely to be sampled. The "L" version (IWSEL) is the opposite of the IWSEH, using the instance weights to prefer sampling the low density areas. Finally, the "U" version (IWSEU) combines both approaches.IWSEU uses uniform randomness to switch between the "L" and "H" weighting strategies. This version encourages most diversity in the sub-samples used for the base clustering.

For all algorithms, the k value was set to reflect the ground truth of the dataset

(3) and the k^* nearest neighbours parameter (for calculating spectral clusterings affinity matrix) was set to 9. Where applicable the bags parameter M was set to 32. Increasing M beyond 32 sees a diminishing return in terms of clustering performance for execution time spent. See Appendix Figures 9.13 and 9.14 for a plot of the relationship between M with execution time and clustering performance. k^* was set to 9 based on preliminary experiments which trialled 3, 6, 9, 12. For brevity the experiment details are omitted, but a summary of the findings was that the results are not sensitive to this parameter as long as this parameter is suitably high, in my tests 9 was most suitable.

For the min and max sample size parameters, some hyper-parameter optimisation was completed. Pairs of m_{min} and m_{max} between 10 and 90 in increments of 10 were trialled, see Table 6.2. The m_{min} and m_{max} parameters are interesting to investigate since they control how strongly the instance weighting is enacted. For example, when $m_{min} = 80$ and $m_{max} = 90$, each bag (base clustering) is using a sample of between 80% to 90% of the data, thus samples of data are nearly equivalent to complete data and the instance weighting controlling the weighted random sampling is only removing a few instance per-sample. Whereas, in the case of $m_{min} = 10$ and $m_{max} = 20$, then the instance weighted sampling is removing 90% to 80% of the data, thus it is most likely that only the data with the highest weights remains in each sample.

NMI was used to evaluate the experiments. The labels were based on distribution they were generated by. This extrinsic approach to evaluation will provide a clear picture for how well clustering fits the intended clusters without any artefacts of intrinsic clustering validation.

Table 6.2: The pairs of m_{min} and m_{max} to be trialled.

Range								
10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90
20	10-30	20-40	30-50	40-60	50-70	60-80	70-90	
30	10-40	20-50	30-60	40-70	50-80	60-90		
40	10-50	20-60	30-70	40-80	50-90			
50	10-60	20-70	30-80	40-90				
60	10-70	20-80	30-90					
70	10-80	20-90						
80	10-90							

Results

The results of the hyper-parameter optimisation are presented first. To avoid overloading a single plot with too much information, the clustering performance of each of the pairs m_{min} and m_{max} is presented over several plots by range value.

Firstly, across all plots it can be observed that regardless of choice of m_{min} and m_{max} IWSE's clustering performance in terms of NMI is unaffected until the imbalance scaling factor surpasses 3. Beyond this value, then the selection of m_{min} and m_{max} is clearly important.

For the m_{min} and m_{max} pairs of range 10, it is clear to see that there is an optimal choice for m_{min} and m_{max}. Figure 6.3, shows that extreme pairs with either strong instance weighting (10-20) or weak instance weighting (60-70), (70-80) and (80-90) tend to perform the poorest.

In Figure 6.4, the results of m_{min} and m_{max} with range 20 are shown. Again, the extreme values perform poorly. Also similar, is that the lower values perform the best, with (20-40) and (30-50) performing the best for higher imbalance factor values. It seems that as the imbalance increases, it is necessary to

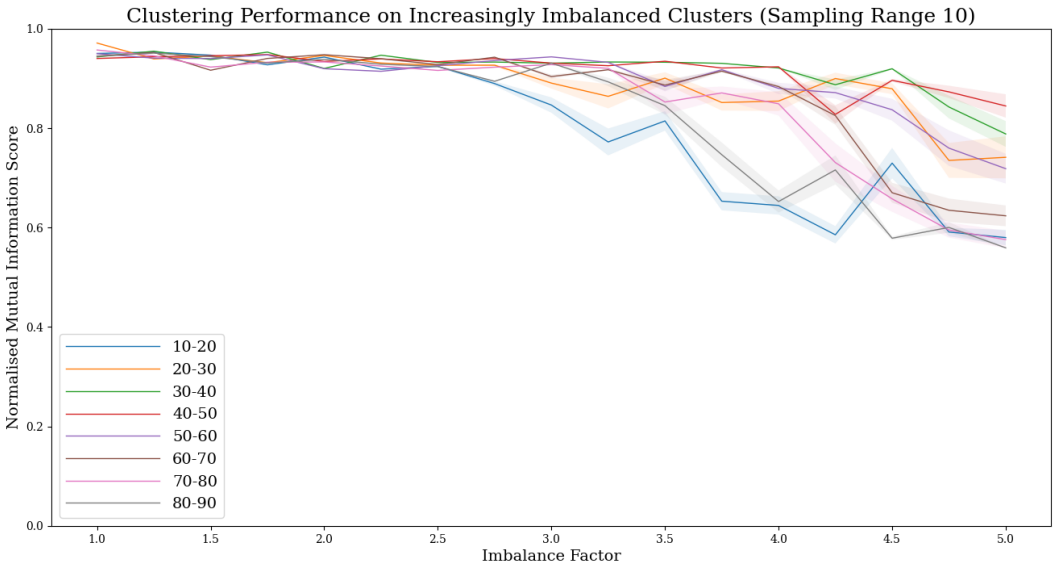


Figure 6.3: The performance IWSEU on the imbalanced data with m_{min} and m_{max} parameters spanning a range of 10.

use lower values for m_{min} and m_{max} (i.e. stronger instance weighting). However, not too low such as (10-20). It is likely that (10-20) is not sampling enough of the data to partition the data well.

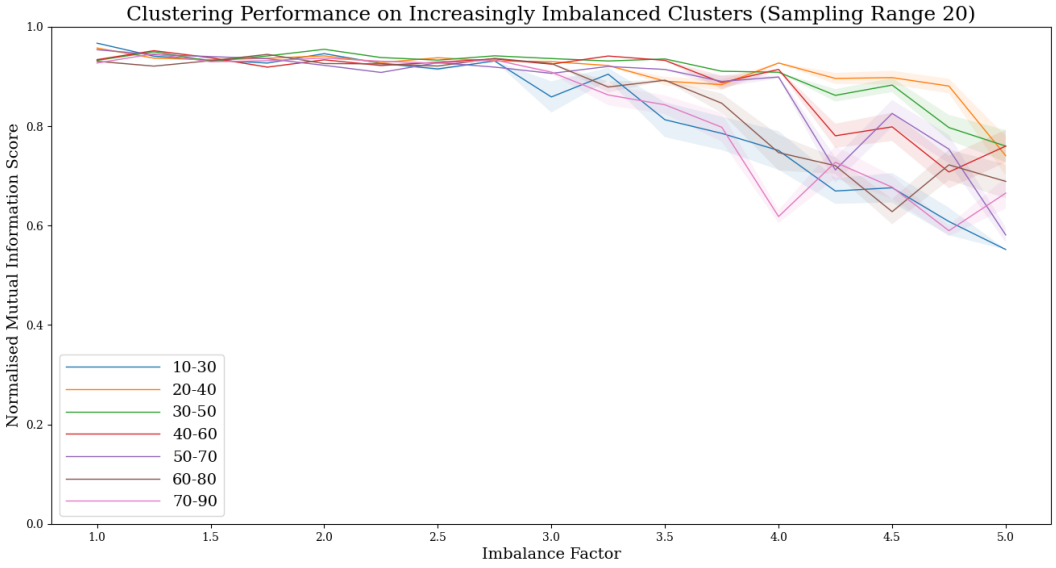


Figure 6.4: The performance IWSEU on the imbalanced data with m_{min} and m_{max} parameters spanning a range of 20.

Interestingly, there is a drop in performance, for the highest imbalance levels seen when the sampling range is 30 or more. This could be because the broader the range is, the less optimised the sampling is. Comparing across the figures this can be observed. In Figures 6.3 and 6.4, some pairs of m_{min} and m_{max} provide very good clustering performance (in terms of NMI) of ∼0.9 for +4.5 imbalance factors, however, in Figures 6.5, 6.6, 6.7, 6.8 accuracy is only ∼0.8 for +4.5 imbalance factors.

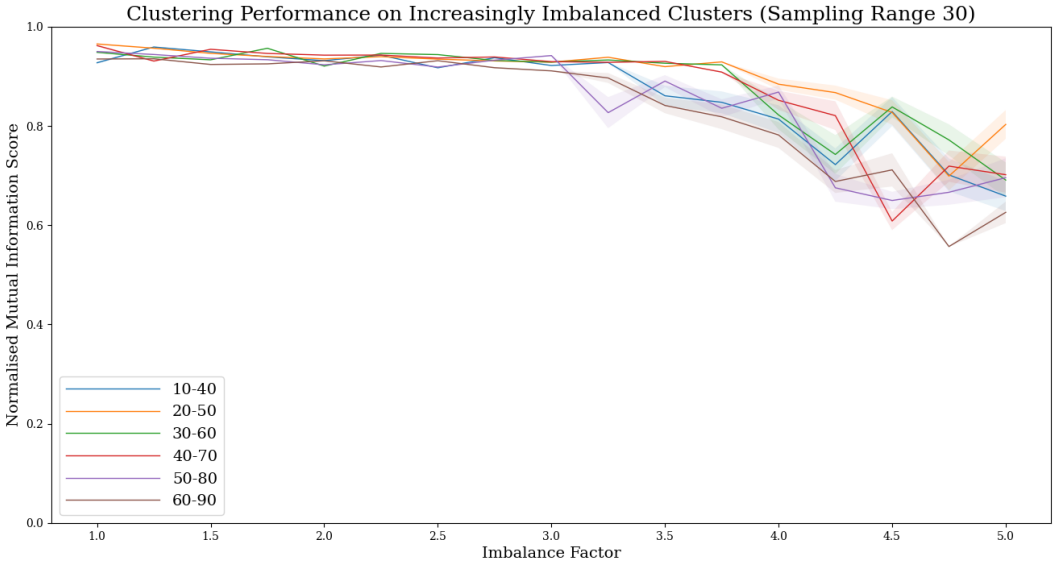


Figure 6.5: The performance IWSEU on the imbalanced data with m_{min} and m_{max} parameters spanning a range of 30.

By calculating the average NMI across each of the imbalance factor values for each pair of m_{min} and m_{max} an overall indication of performance can be ascertained, see Table 9.13 in the Appendix. Based on this, the best performing m_{min} and m_{max} parameter values are (20-30), (10-50), (20-60), (50-60), (10-60), (20-50), (30-50), (20-40), (40-50) and (30-40). As observed from the plots, the best performing m_{min} and m_{max} values have a range of less than or equal to 40 and involve using an m_{max} of ≤60%. The best performing m_{min}

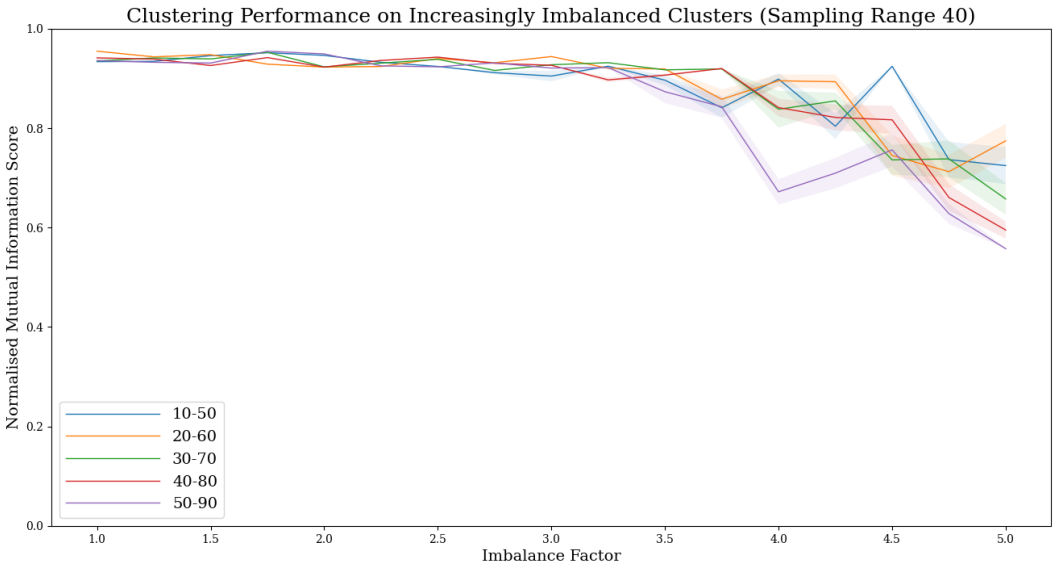


Figure 6.6: The performance IWSEU on the imbalanced data with m_{min} and m_{max} parameters spanning a range of 40.

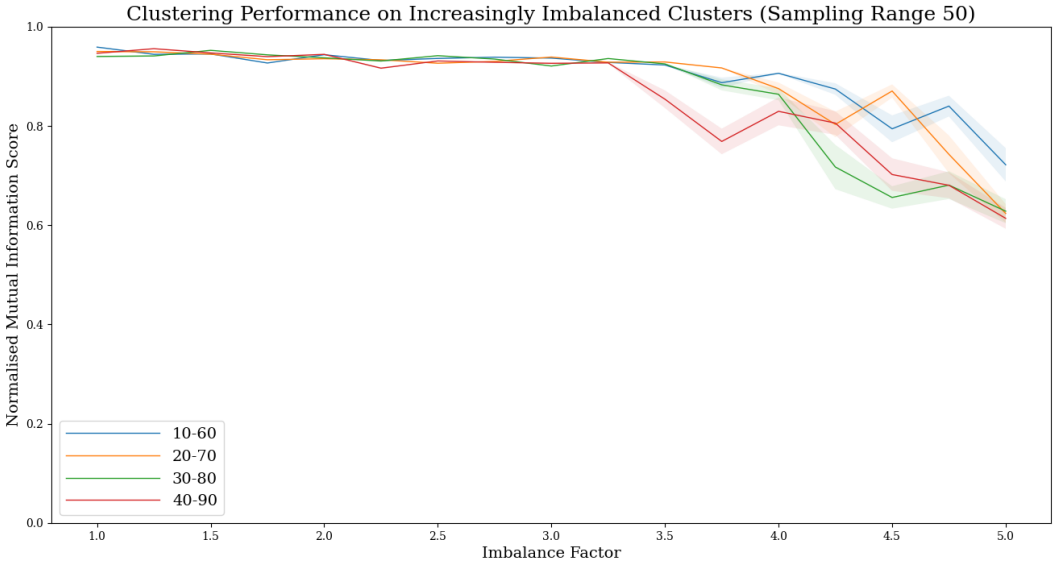


Figure 6.7: The performance IWSEU on the imbalanced data with m_{min} and m_{max} parameters spanning a range of 50.

and m_{max} pairs can be compared in Figure 6.9.

Further aggregating the results by range, shows two interesting correlations, see Table 6.3. Firstly, as the range increases the clustering accuracy decreases,

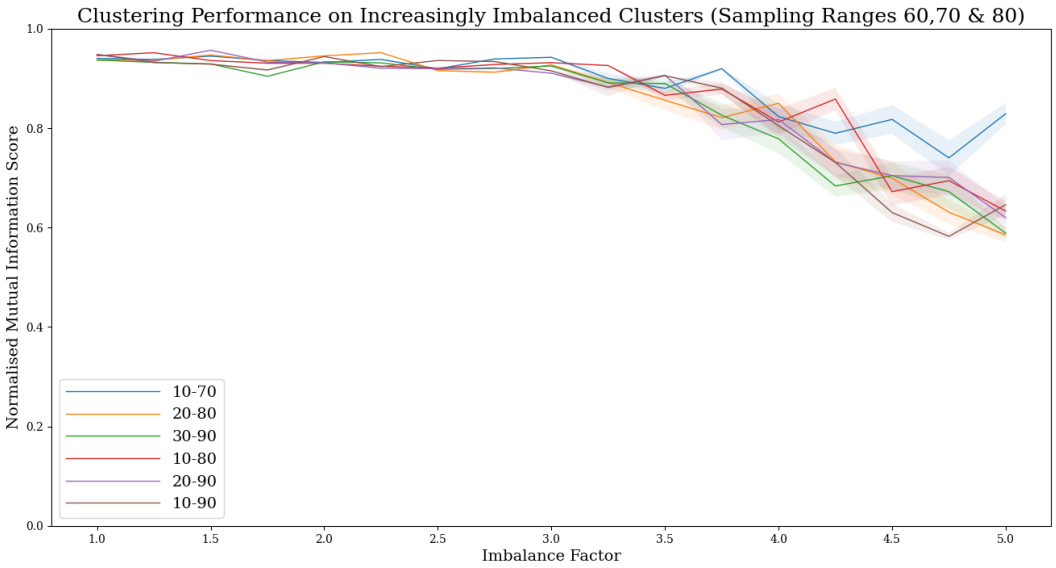


Figure 6.8: The performance IWSEU on the imbalanced data with m_{min} and m_{max} parameters spanning a range of 60, 70 and 80.

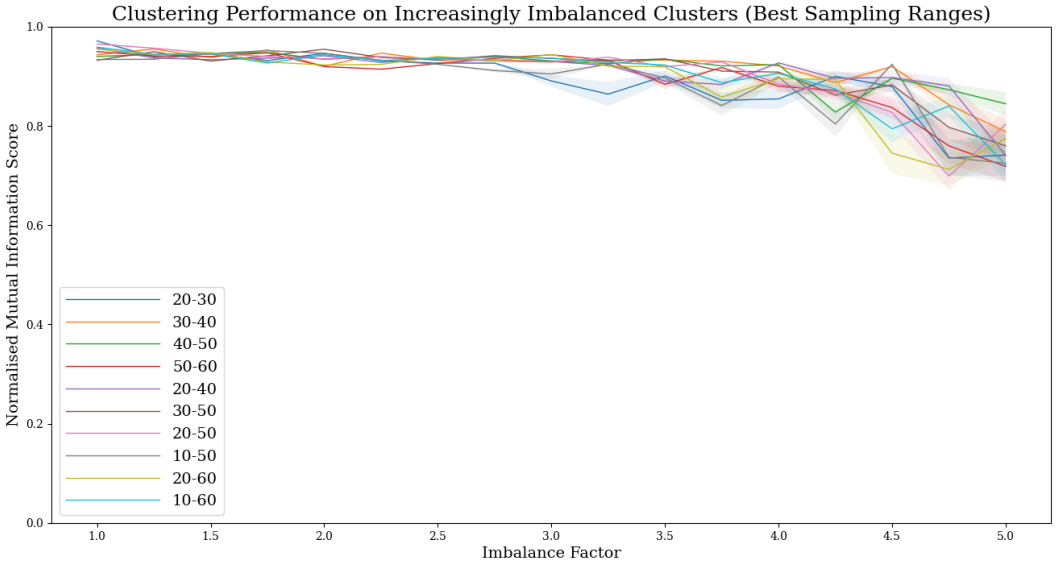


Figure 6.9: The best performing m_{min} and m_{max} parameters for IWSEU on the imbalanced data.

this trend has a Pearson's correlation coefficient of -0.665. The suspected reason for this is that as the range becomes increasingly disperse it is less focused on the optimal sample size – i.e. the optimal level of instance weighting. Secondarily, there is a weak positive correlation (0.317) between range and

Table 6.3: The NMI results of the tested m_{min} and m_{max} pairs across all imbalance factors aggregated by mean and st. dev. by range.

Range	Mean NMI	Standard Deviation NMI
10	0.870524	0.007655
20	0.870523	0.009115
30	0.871842	0.009628
40	0.876535	0.009222
50	0.879414	0.008370
60	0.862021	0.010371
70	0.861184	0.010125
80	0.849350	0.008215

clustering performance, this suggests that smaller ranges perform more stably. Note the range of 80, does not fit the pattern seen, but this may be because it is only the average of one series of results (10-90).

In summary, my tests provide some insights in how to select m_{min} and m_{max} . For this dataset the best choice is 30-40 or 40-50, as this achieves the best overall clustering performance. However, it is inferred that a broader range will be more broadly applicable across a range of datasets. The results, (particularly Table 9.13), show that 30-50, 20-40, or 10-60 all perform very well, but have larger ranges 20, 20, 50 respectively. Having a larger range will enable the IWSE method to extract a greater amount of information from the dataset. Additionally, within that range there is an increased chance of a high-performing sample. Thus, as a suggestion of a default values, for robust performance, m_{min} = 30 and m_{max}=50 are suggested alongside M=32. Moving forwards, these values will be used used across a variety of datasets.

Finally, using the aforementioned parameters for the IWSE algorithms, the IWSE approaches were compared with spectral clustering and a spectral ensemble. Figure 6.10 shows that as the clusters become increasingly imbalanced,

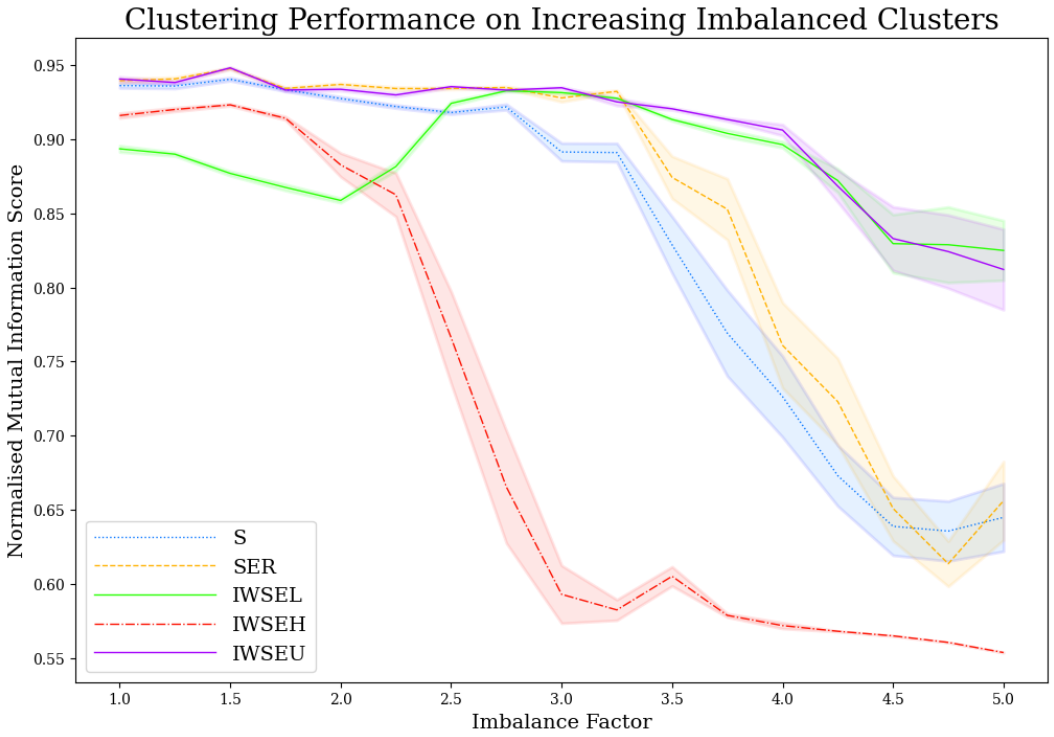


Figure 6.10: IWSEU and IWSEL perform well despite imbalanced clusters.

the performance of spectral clustering and SER drops significantly. When the imbalance ratio reaches 3.75, IWSEL and IWSEU offer superior performance over S or SER. As can be expected, IWSEH performs increasingly poorly as the imbalance increases (due to lack a sampling of the smallest cluster). However, interestingly IWSEU performs similarly to IWSEL and on occasion even better than IWSEL. This is despite incorporating degenerate "H" partitionings. It seems that this could be due to the consensus function benefiting from the degenerate partitionings, as has been observed in other research [53].

6.4.2 Experiment B

Setup

Image segmentation is a popular and important application of clustering algorithms and image segmentation performance can easily be assessed visually.

Yudong He provides a image for assessing segmentation performance[2] for the special case of imbalanced clusters [67]. The image contains a pale wooden surface. On the surface is a black pen, and a number of red, green, blue, orange and yellow round plastic tokens/chips/coins. The image is lit from the right, producing a graduated effect across the surface. Relatively, the areas consumed by the different objects is much less than the background (making this an imbalanced clustering problem). Yudong He finds that segmentation by K-Means, Fuzzy K-Means, Maximum-Entropy Fuzzy Clustering all result in faulty clusterings. These clustering algorithms wrongly partition the wooden surface into two separate clusters. While their proposed method Equilibrium K-Means is able to group the image accurately into 5 clusters: (background), (red/orange tokens), (blue tokens), (yellow tokens) (green tokens and pen).

To trial the proposed algorithm the following setup was used. The aforementioned "chipcoin" image was saved as 64x48 (3072 pixels) JPG format image from the original work and was then loaded using an RGB colour space. Then S, SER, IWSEL and IWSEU were used to segment the image 5 times each. To assess the impact of the instance weighting, settings between the algorithms were kept consistent and sensible values were selected based on the previous experiments findings where relevant. For spectral, k was set to 6, the affinity

[2]Note: Image appears in submissions v1 and v2 of [67].

matrix used 10 nearest neighbours. For the Spectral Ensemble, again k was set to 6, and the affinity matrix used 10 nearest neighbours, the bagging-based ensemble used 32 bags and randomly sampled between 30-50% of the instances per bag. For IWSEL and IWSEU, again k was set to 6, and the affinity matrix used 10 nearest neighbours, the ensemble used 32 bags sampling between 30-50% of instances per bag based on instance weighting based on density estimated using the exponential function, the bandwidth value for the kernel was chosen dynamically using the Silverman method.

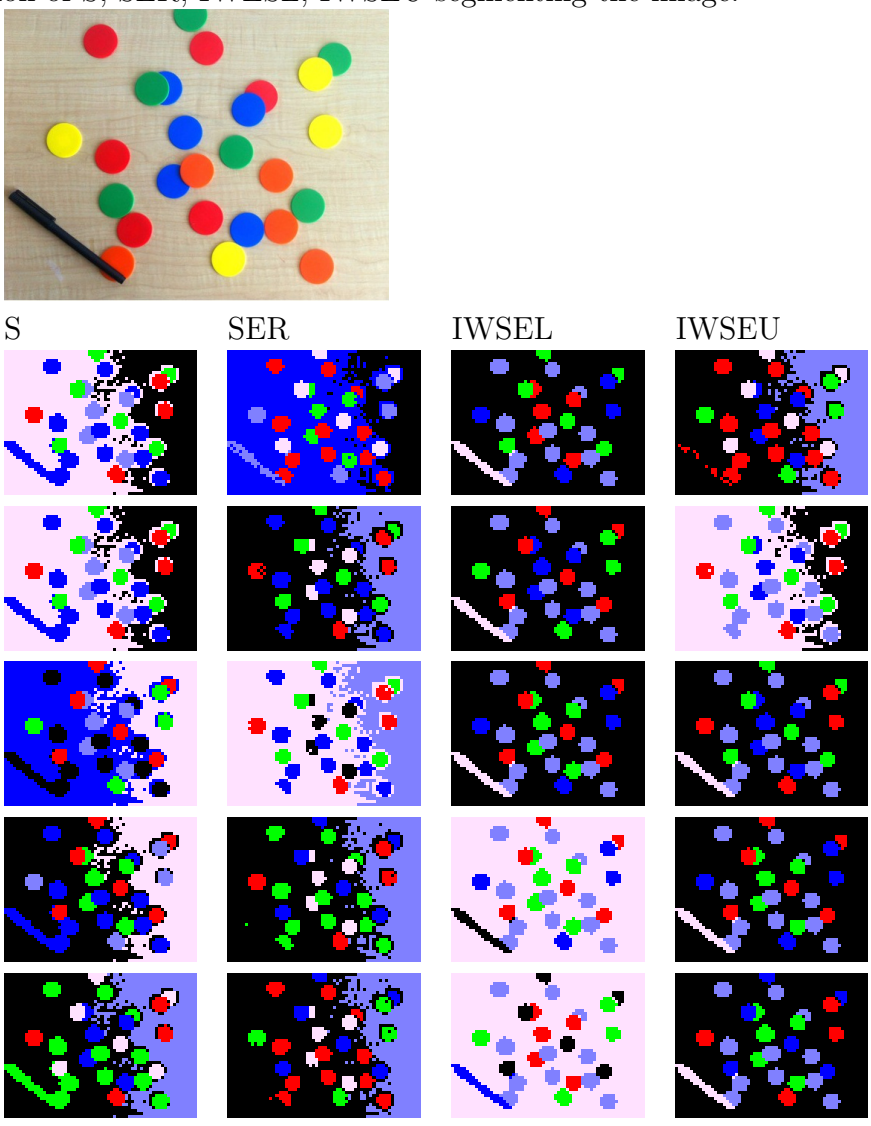
Results

Figure 6.4 shows 5 outputs of the 4 algorithms on the "chipcoin" image. Column S shows the five runs of Spectral clustering, notice that the background is not correctly identified and is subdivided. Also, the pen is mistakenly grouped with the red/orange chip-coins. Column SER shows the five runs of Spectral Ensemble, similar to S, the background and pen are not uniquely identified. Column IWSEL shows the five runs of Instance Weighted Spectral Ensemble in mode "L" (preferring low density instances), notice the background and objects have been segmented well. Column IWSEU shows the five runs of Instance Weighted Spectral Ensemble in mode "H" (preferring high density instances), the result is similar to IWSEL although less reliable. IWSEL is highly effective on this image as it meets the conditions required for it to work well. The background is numerous and varied (low density) while the important objects are relatively consistent (high density) even if they are in lesser in occurrence.

This is promising initial finding and shows that the instance weighting can aid

in image segmentation tasks. Although, it is clear that more experimentation with more images would be necessary to bolster this finding.

Table 6.4: Top: Chip-coin image from [67], notice large amount of graduated background and the imbalanced quantities of chip coins. The columns show 5 execution of S, SER, IWESL, IWSEU segmenting the image.



6.4.3 Experiment C

Setup

Artificial Datasets

A variety of 2D artificial datasets were chosen. 2D datasets were chosen as any issues with clustering results can be diagnosed visually. To test the benefit of the proposed algorithm, datasets including those with imbalance were selected. Additionally, datasets without any strong class imbalance selected too, to assess if the algorithm would have negative consequences when apply upon a balanced dataset. See Table 6.5 for descriptions of datasets.

"2d-20c-no0", "2d-3c-no123" and "2d-4c-no4" contain a class imbalance. "2d-20c-no0" has an imbalance of $\times 4.7$ between the smallest and largest cluster. "2d-3c-no123" has a $\times 4.57$ imbalance and "2d-4c-no4" has a $\times 6.79$. The nature of the imbalance is between the clusters is smooth rather than harshly stepped. The clusters are close to each other, but mostly not overlapping.

"Compound" was selected as another dataset with imbalance. "Compound" has a large imbalance of $\times 9.88$ between the smallest and largest cluster. Again the imbalance is smooth with clusters of various sizes, rather than harshly stepped. This dataset, unlike the previous choices, tests a clustering algorithms ability to handle complex shapes and concentric clusters.

"Jain" also called "half-moons". This is a popular benchmark dataset and includes a minor imbalance of $\times 2.85$ and interlocking clusters which require a clustering model to fit the complex shapes which cannot be separately linearly

(without constructing some representational space).

"Long3" is a relatively simple dataset featuring two skewed clusters with an imbalance of $\times 4$, this dataset is similar to "2d-20c-no0", "2d-3c-no123" and "2d-4c-no4".

"sizes5" is another imbalanced dataset. However, this dataset has a strong imbalance of $\times 9.99$ and the imbalance is harshly stepped. With one extremely large cluster and three relatively very small clusters. The clusters are slightly overlapping.

"Zelnik1", "Zelnik2", "Zelnik3", "Zelnik4", "Zelnik5" and "Zelnik6" were selected as the balanced datasets. They are all relatively balanced, and have imbalance factors of $\times 2.28$, $\times 1.12$, $\times 1.62$, $\times 1.38$, $\times 1.28$, and $\times 1.79$ respectively. "Zelnik1" and "Zelnik6" are concentric. "Zelnik2" and "Zelnik4" contain square clusters embedded with a larger noise cluster. "Zelnik3" is similar to "Jain" but is more balanced and has 3 clusters. "Zelnik5" has four line-shaped clusters of different sizes. As the instance count remains similar between clusters while the size is different the density for each line is different.

"DiscTubes" and "ThreeWells" were replicated from earlier research [15, 59] these datasets are known to challenge weaknesses in the spectral clustering algorithm. "Discs and Tubes" datasets features geometrically defined areas of uniformly distributed instances. For "Discs and Tubes" the entirety of the instances within the area of the shapes are uniformly distributed. In the "ThreeWells" dataset one large distribution is positioned next to two smaller distributions.

Table 6.5: Descriptions of the synthetic datasets trialled.

Name	Instances	Features	Clusters	Dataset Type	Source
2d-20c-no0	1517	2	20 cluster various sizes[3]	Artificial	From [68]
2d-3c-no123	715	2	(264)(370)(81)	Artificial	From [68]
2d-4c-no4	863	2	(421)(86)(294)(62)	Artificial	From [68]
Compound	399	2	(50)(92)(38)(45)(158)(16)	Artificial	From [69]
Jain	373	2	(97)(276)	Artificial	Credit [32]
Long3	1000	2	(800)(200)	Artificial	From [68]
sizes5	1000	2	(769)(77)(77)(77)	Artificial	From [68]
Zelnik1	299	2	(61)(139)(99)	Artificial	From [70]
Zelnik2	303	2	(95)(102)(106)	Artificial	From [70]
Zelnik3	266	2	(118)(73)(75)	Artificial	From [70]
Zelnik4	622	2	(109)(150)(114)(111)(138)	Artificial	From [70]
Zelnik5	512	2	(117)(122)(123)(150)	Artificial	From [70]
Zelnik6	238	2	(100)(82)(56)	Artificial	From [70]
Discs and Tubes	1722	2	(1468)(132)(122)	Artificial	Replicated from [59]
Three Wells	1660	2	(1000)(330)(330)	Artificial	Replicated from [59]

Figure 6.11 shows each of the datasets, with colour being used to show the suggested clustering, and Figure 6.12 shows datasets with the colouration showing the weighting assigned by IWSE.

Where datasets not are replicated, they were downloaded from:
https://github.com/deric/clustering-benchmark/.

Benchmark Datasets

The popular "Iris", "Ecoli", "Wine", "Zoo" datasets were downloaded from the UCI Machine Learning Repository [41]. These datasets were trialled to provide a benchmark against other methods and to assess the suitability of the methods under different conditions. These benchmark datasets are popular choices and are of different instance, feature and cluster numbers. "Iris" contains balanced clusters. While "Ecoli" has imbalanced clusters. "Wine" and "Zoo" are relative high-dimensional, thus may prove challenging for my kernel based approach. Unlike the others, which are continuous, the "Zoo"

[3](106)(97)(43)(70)(85)(50)(85)(108)(65)(108)(33)(103)(23)(71)(88)(85)(84)(75)(53)(85)

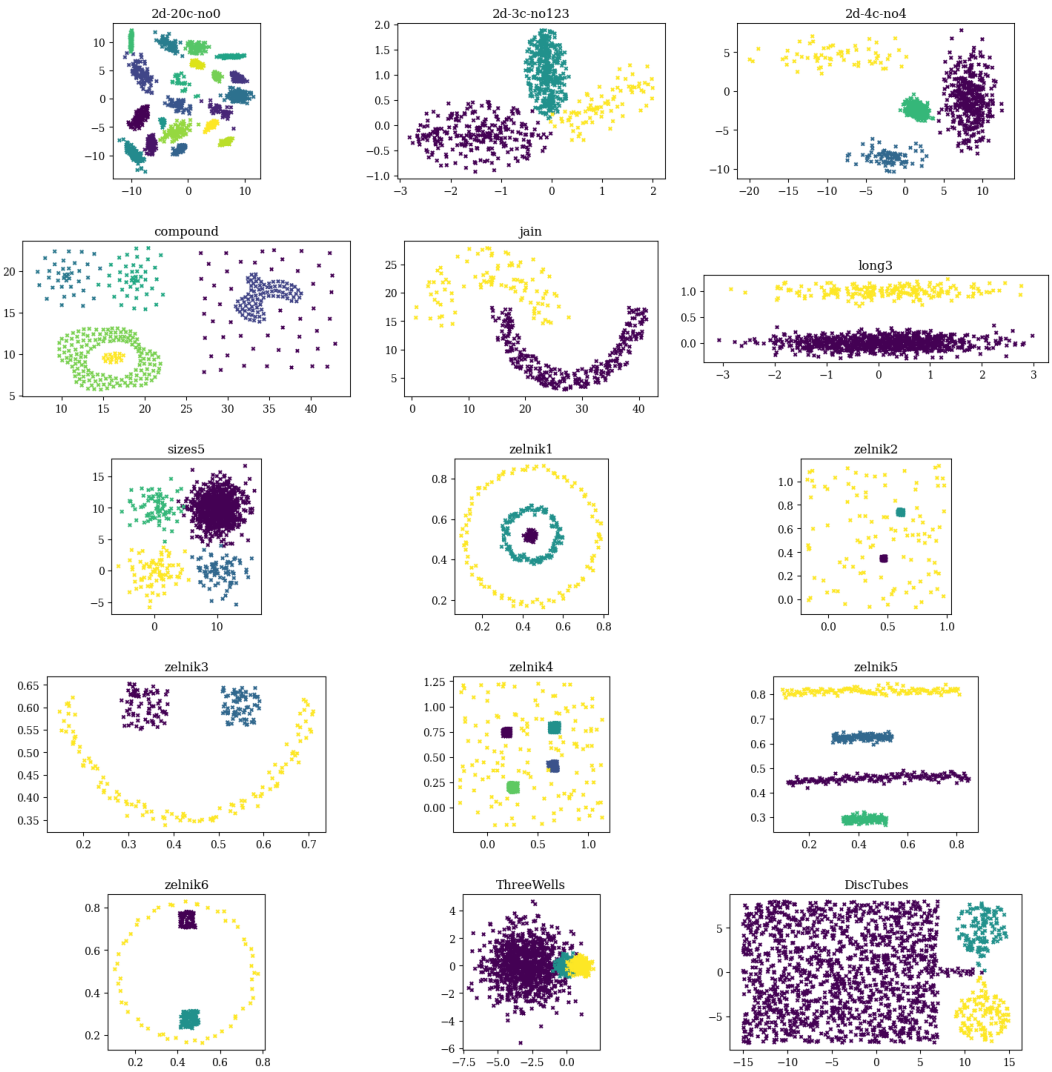


Figure 6.11: The artificial datasets trialled, showing the class labels.

dataset contains several one-hot encoded columns, which can present a challenge to techniques not especially designed for this type of data. See Table 6.6 for a description of the datasets.

The experiment setup was as follows. As in Experiment A, IWESU, IWSEL, IWSEH, SER and S where again compared.

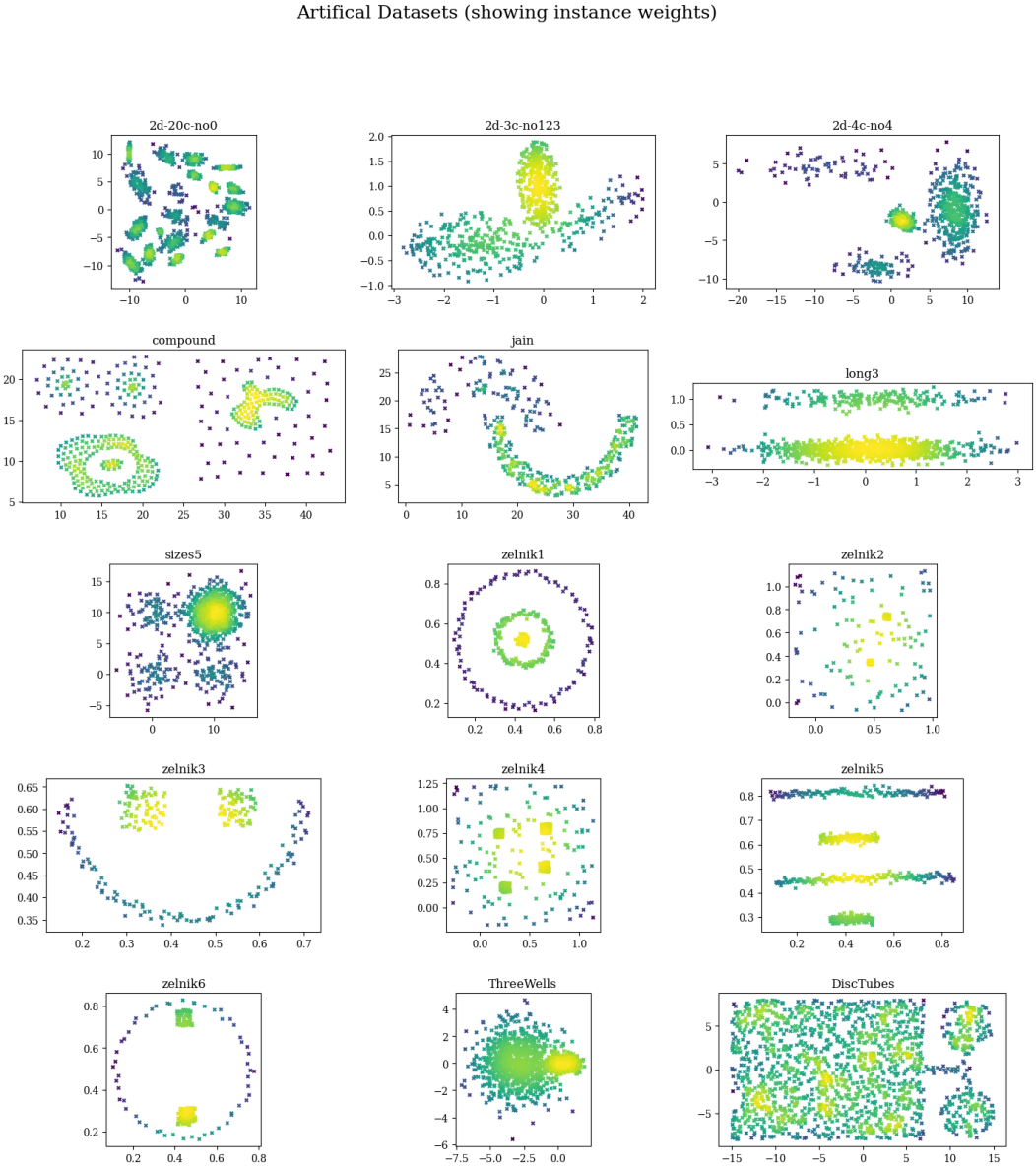


Figure 6.12: The artificial datasets trialled, showing the weighting applied within IWSE.

Clustering was executed with the k value matching the intended number of clusters indicated by the dataset's labels. Spectral clustering was applied with k^* nearest neighbours set to 9 (for constructing the affinity matrix). The ensemble methods used 32 bags.

Table 6.6: Descriptions of the benchmark datasets trialled

Name	Instances	Features	Clusters	Dataset Type	Source
Iris	150	4	(50)(50)(50)	Benchmark	[41]
Ecoli	336	7	(143)(77)(2)(7)(35)(20)(52)	Benchmark	[41]
Wine	178	13	(59)(71)(48)	Benchmark	[41]
Zoo	101	16	(41)(13)(20)(4)(8)(10)(5)	Benchmark	[41]

Additionally, instance selection based methods were evaluated for comparison. One method used was *Stratified Sampling Instance Selection followed by Spectral* (SSIS-S). This used N-Dimensional stratified sampling to take sample of the dataset. This method divides each dimension of the dataset into 5 equal-width ranges, hence given 2-dimensions, this would create 25 bins. From each bin, 10 instance are selected. This sample is then clustered using spectral clustering and the labels from the clustering are then propagated by a KNN kernel configured to use the 9 nearest neighbours. Another instance selection method trialled is *Weighted Random Sampling followed by Spectral* (WIS-S). This approach samples 50% of the data using a weighted random function based on the output of the exponential kernel. This effectively shows the performance of a base clusterer within IWSEL. Finally, *Threshold Instance Selection followed by Spectral* (TIS-S) again uses the same exponential kernel and again takes a 50% sample of the data but instead simply takes the least dense 50% and then applies spectral clustering. After SSIS-S, WIS-S and TIS-S label propagation is applied using a knn kernel with neighbours 9 to label instances that bypassed the clustering. Experiments were repeated 5 times.

Results

Table 6.7: Comparison of IWSE clustering performance (NMI) on imbalanced datasets.

Dataset Algorithm	2d-20c-no0	2d-3c-no123	2d-4c-no4	Compound	Jain	Long3	sizes5	disc-tubes	three-wells
IWSEU	0.93 ± 0.032	0.96 ± 0.004	0.95 ± 0.098	**0.79** ± 0.049	0.70 ± 0.247	**1.00** ± 0.000	0.87 ± 0.148	0.61 ± 0.196	**0.77** ± 0.006
IWSEL	**0.94** ± 0.017	0.96 ± 0.004	**1.00** ± 0.000	0.74 ± 0.008	0.75 ± 0.180	**1.00** ± 0.006	0.79 ± 0.123	0.88 ± 0.217	0.52 ± 0.035
IWSEH	0.85 ± 0.028	0.75 ± 0.009	0.75 ± 0.014	0.72 ± 0.038	0.24 ± 0.008	0.85 ± 0.255	0.70 ± 0.146	0.52 ± 0.007	0.74 ± 0.008
SER	0.93 ± 0.023	0.96 ± 0.014	0.99 ± 0.006	0.76 ± 0.011	0.41 ± 0.050	0.96 ± 0.044	0.60 ± 0.017	0.69 ± 0.240	0.66 ± 0.007
SSIS-S	0.92 ± 0.025	0.71 ± 0.034	0.94 ± 0.094	0.68 ± 0.054	0.65 ± 0.199	**1.00** ± 0.000	**0.94** ± 0.012	0.36 ± 0.342	0.50 ± 0.039
WIS-S	0.89 ± 0.018	0.75 ± 0.012	0.87 ± 0.165	0.73 ± 0.010	0.42 ± 0.053	**1.00** ± 0.000	0.88 ± 0.052	0.86 ± 0.201	0.55 ± 0.006
TIS-S	0.75 ± 0.031	0.96 ± 0.000	0.69 ± 0.000	0.75 ± 0.005	0.51 ± 0.000	**1.00** ± 0.000	0.45 ± 0.000	0.51 ± 0.000	0.30 ± 0.000
S	0.91 ± 0.036	**0.97** ± 0.000	0.83 ± 0.098	0.75 ± 0.000	**1.00** ± 0.000	**1.00** ± 0.000	0.54 ± 0.000	**0.94** ± 0.000	0.60 ± 0.000

Table 6.8: Comparison of IWSE clustering performance (NMI) on approximately balanced datasets.

Dataset Algorithm	Zelnik1	Zelnik2	Zelnik3	Zelnik4	Zelnik5	Zelnik6
IWSEU	**1.00** ± 0.000	0.61 ± 0.007	**1.00** ± 0.000	0.74 ± 0.012	**1.00** ± 0.000	0.67 ± 0.008
IWSEL	0.85 ± 0.175	0.57 ± 0.026	0.97 ± 0.056	0.75 ± 0.002	0.79 ± 0.050	0.66 ± 0.026
IWSEH	0.86 ± 0.209	0.54 ± 0.086	0.92 ± 0.061	0.68 ± 0.004	0.99 ± 0.014	0.49 ± 0.001
SER	0.73 ± 0.243	0.60 ± 0.026	**1.00** ± 0.009	0.72 ± 0.015	0.97 ± 0.077	0.49 ± 0.002
SSIS-S	0.77 ± 0.038	0.68 ± 0.082	0.86 ± 0.094	0.74 ± 0.032	0.81 ± 0.114	0.70 ± 0.018
WIS-S	0.66 ± 0.000	0.62 ± 0.011	0.74 ± 0.013	0.75 ± 0.010	0.78 ± 0.091	0.69 ± 0.004
TIS-S	0.73 ± 0.056	0.55 ± 0.089	0.58 ± 0.000	0.51 ± 0.048	0.68 ± 0.137	0.53 ± 0.032
S	0.81 ± 0.171	**0.87** ± 0.000	0.74 ± 0.145	**0.88** ± 0.000	0.71 ± 0.103	**1.00** ± 0.000

Firstly, considering the results of the 9 imbalanced artificial datasets, seen in Table 6.7: “2d-20c-no0”, “2d-3c-no123”, “2d-4c-no4”, “Compound”, “Jain”, “Long3”, “sizes5”, “Disc-Tubes” and “ThreeWells”.

It can be seen that “2d-20c-no0” and “2d-3c-no123” perform very well using just the plain spectral clustering algorithm, which leaves little room for improvement. For “2d-20c-no0” IWSEL is best performing algorithm with a NMI score of 0.94 , but it is only a narrow improvement on SER scoring 0.93. For “2d-3c-no123” S is best performing algorithm. Minimal to no benefit was found for applying instance weighting to these datasets.

For “2d-4c-no4” spectral performs modestly well (0.83), but leaves room for improvement. IWSEL performs best matching the intended label exactly, but as seen before offers only a marginal improvement on SER scoring 0.99. This shows the benefit is from the application of the ensemble technique rather than instance weighting.

For “Compound” a promising result is seen on imbalanced data, IWSEU outperforms all the other methods including the instance selection techniques. A possible reason why IWSEU worked well in this case is that “Compound” is very varied in terms of distribution, and thus other methods lack the nuance and flexibility to recognise all the different distributions in the dataset.

For “Jain” spectral clustering alone provides a perfect match with the suggested clustering labels, despite the imbalance cluster sizes. IWSEU and IWSEL perform better than the instance selection or the plain ensemble methods, but still significantly worse than spectral alone.

Moving to “Long3” both plain spectral, the instance selection methods and instance weighting are all able to separate this dataset.

Next “sizes5” shows an negative finding, with the stratified sampling based instance selection performing best. This is an extreme case with a $\times 9.99$ imbalance. It is possible that if IWSE's parameters were tuned to account for more extreme imbalance it may perform better.

For “Disc-Tubes” spectral performs the best leaving little room for improvement. While the other methods hinder performance. A possible reason for this negative finding is that this dataset has a lack of a useful variation in density, which logically prevents the density based techniques “IWSE”, “WIS-S” and “TIS-S” from improving upon SER or S respectively. However, it is suspected that is only part of the issue, since all methods perform worse that S, not approximately equal to. The thing that all methods other S have in common is that they take sample(s) of the data. Therefore, it seems likely that therein lies the problem. The sample taken may “fracture” the clusters and thus mislead the partitioning and decrease the clustering performance. Notice that in 6.12, there are misleading density “hotspots” in the uniform randomly noise, which could cause misleading fractures after sampling.

Lastly, for the imbalance datasets “ThreeWells” shows a positive outcome for IWSEU on a highly overlapping dataset. It seems that the combination of instance weighting and the ensemble method has allowed IWSEU to identify a variety of partitions separating different pairs of clusters well. Then by combining this information it has been able to devise an overall partitioning which

is more accurate than its individual components (this can be deduced by comparing with the WIS-S result). This positive result supports the findings of Experiment A, as "ThreeWells" is similar to the generated datasets in Experiment A.

Secondarily, considering the 6 (mostly) balanced datasets: "Zelnik1", "Zelnik2", "Zelnik3", "Zelnik4", "Zelnik5" and "Zelnik6", see in Table 6.8. On these datasets, as there are no severe imbalances, it is not expected that the IWSE approach will excel. These datasets are included to see under what conditions the proposed approach may detract from performance.

"Zelnik1" tests two aspects of a clustering algorithm. Firstly, it tests a clustering algorithms ability to handle concentric clusters and it also tests a clustering algorithms ability to recognise clusters of different densities. For "Zelnik1" IWSEU performs best outperforming all other test techniques. Again, this shows instance weighting and the ensemble technique complimenting each other. Similarly, "Zelnik3" and "Zelnik5" show somewhat similar results (although in their cases, more of the benefit appears to be originating from the use of an ensemble - notice the NMI score of SER). "Zelnik3" and 'Zelnik5" are similar to "Zelnik1", but are less concentric, and not concentric, respectively.

"Zelnik2", "Zelnik4" and "Zelnik6" show similar findings. For these datasets, it is most advantageous to simply apply spectral. Unfortunately, applying my approach (albeit where it is not required) does comprise performance in these cases. For "Zelnik2" and "Zelnik4", it is likely the case that DBSCAN/OPTICS, would perform best on this dataset, as these datasets, have a "noise" cluster surrounding well-grouped clusters. The "Zelnik6" result is interesting,

because visually the dataset is similar to "Zelnik1", on which the proposed algorithm performs well, however here it does not. By examining the weighting shown in 6.12, it can be hypothesised that this is similar to "DiscTubes" the instance weighting could be generating unhelpful fractures in the samples.

An important observation is that often IWSE is better WIS-S or SER, this shows the combination of these technique allows them to perform better than either one individually. Demonstrating some promise for the instance weighting technique.

Other general observations, include that stratified sampling is quite an effective technique for handling imbalanced clusters, with a minimum of user intervention. Of the instance selection methods trialled stratified sampling was frequently the best of the instance selection based techniques. Although there is a couple of exceptions to this, notably "Compound" and "disc-tubes'. In these cases, it likely that the bin boundaries create artefacts which misled the clustering. This can be overcome by careful bin-width and bin-sample-size selection. But when the dimensionality is moderate it can be very costly to increase the number of bins (i.e. reduce the bin-width)[4].

Another observation is that SER is not always better than spectral. Ensemble techniques are often praised in the literature as a reliable way to increase clustering performance, but these experiments show that ensembles can also detract from clustering performance, despite reasonable choices. In this case, a possible cause is that the individual samples (at most 50% of the data) fail to capture the representative view of the data.

[4]For example: in 7-Dimensional dimensional space dividing each dimension into 5 equal-width bins creates 5^7 bins, but dividing into 15 bins creates 15^7 (170859375) bins.

Dataset / Algorithm	Normalized Mutual Info Score: Ecoli	Iris	Wine	Zoo
IWSEU	0.64 ± 0.007	0.78 ± 0.030	0.87 ± 0.022	**0.83** ± 0.015
IWSEL	0.65 ± 0.005	0.73 ± 0.009	0.86 ± 0.027	**0.83** ± 0.011
IWSEH	0.57 ± 0.020	**0.83** ± 0.045	0.86 ± 0.018	0.78 ± 0.024
SER	0.56 ± 0.016	0.78 ± 0.007	0.87 ± 0.013	**0.83** ± 0.019
SSIS-S	0.64 ± 0.000	0.78 ± 0.000	X	X
WIS-S	0.65 ± 0.014	0.71 ± 0.076	0.85 ± 0.022	0.75 ± 0.037
TIS-S	**0.69** ± 0.000	0.57 ± 0.000	**0.89** ± 0.000	0.68 ± 0.025
S	0.65 ± 0.000	0.76 ± 0.000	0.88 ± 0.000	0.79 ± 0.000

Considering the imbalanced datasets "Ecoli" and "Zoo". "Ecoli" shows a small success for TIS-S. Instance weighting nor instance selection based on weighting (ensemble or not) does not outperform spectral. This could be due to two factors. Firstly, the Ecoli dataset has 7 features, which would weaken the effectiveness of kernel-based methods. Secondly, some of the clusters are very small (the two smallest clusters are just 2 and 7 instances respectively), causing the random sampling methods to risk omitting them entirely. TIS-S's success may have been due to only removing instances from the largest most dense cluster. "Zoo" is quite unlike all other datasets tested so far as it is makes use of one-hot encoding. On this dataset, ensemble techniques perform best, but an advantage to instance weighting is not seen. A positive takeaway from this is that IWSE does not fail on one-hot encoded data. Note that due to higher number of features in "Zoo" the n-dimensional stratified sample was not feasible.

Regarding the more balanced datasets, "Iris" and "Wine". For "Iris" interestingly, the approach with the greatest clustering performance is IWSEH. This indicates that ignoring the less-dense noise is beneficial. For the "Wine" dataset TIS-S performed best, it is interesting to see that this simple method works well in practice on real-world datasets. Again, due to the higher num-

ber of features, the n-dimensional stratified sample was not feasible for "Wine".

On a positive note, across all the benchmark datasets tested, the application of IWSE is does not decrease performance, when it is not required or necessarily well-suited.

6.5 Conclusion

In summary, all three experiments show that the IWSE method can be effective on imbalanced data. Additionally, Experiment B, shows how IWSE could be useful for image segmentation where the objects in the image are of imbalanced sizes. While Experiment C indicates that IWSE is most beneficial when datasets have a density variation between clusters (as anticipated), note the successes with "Compound", "ThreeWells" and "Zelnik1".

It is believed that this research is the first attempt at creating diversity in the generative mechanism by using instance weighting to sub-sample for a spectral bagging ensemble. The goal of this initial research was to answer RQ2 ("How can instance weighting be applied to graph-based clustering algorithms to handle imbalanced data?"). The chapter has discussed some of the options when designing an instance weighted cluster ensemble and presents an implementation of a promising approach from this discussion. The results, particularly Figure 6.10 demonstrate that IWSE leads to higher clustering performance in terms of NMI than a spectral clustering ensemble when handling simple imbalanced datasets.

A further benefit of the approach is that the bagging element of the design makes execution on distributed hardware trivial. Furthermore, this approach can mostly be implemented using a readily available libraries. However, the IWSE approach does have some drawbacks. Most notably its current CSPA based consensus function is computationally expensive. This because it creates a $n \times n$ matrix comparing the co-membership of instances amongst the base clusterings. Selecting a more computationally efficient consensus function (such as HGPA, as recommended by [65]) could overcome this weakness.

Comparing against instance selection, these experiments do not show a perfectly consistent advantage over instance selection or just plain spectral. Rather, the result is more mixed. It seems the instance weighting technique is not as widely applicable has was theorised, the mixed results suggest that further work is required to mitigate the risk of the IWSE approach performing worse than plain spectral under certain conditions.

An interesting secondary finding is an insight into why the IWSEU method is effective. To illustrate this point, Figure 6.13 shows some of the partitionings created within the IWSEU approach and the resultant accumulated affinity matrix to be used for the final clustering. On the left of Figure 6.13, IWSEU randomly chose the "H" mode, in this weighting scheme the base clusterings confuse the small cluster with the medium cluster, but they do encode some information about the largest cluster that the "L" mode does not capture. In the centre of Figure 6.13, IWSEU randomly chose the "L" mode, in this weighting scheme the base clusterings consistently identify the smallest cluster and achieve good clustering performance. It seems the CSPA method of summing the pairwise similarity across the bags (Figure 6.13 right) then clus-

tering, can handle some degenerate partitionings and even benefit from the information they provide.

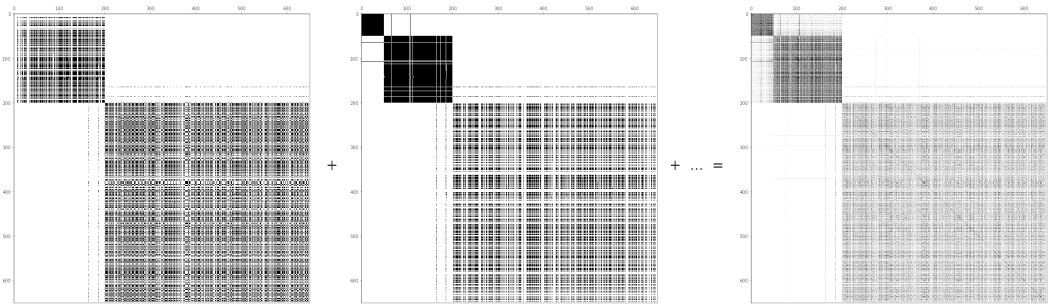


Figure 6.13: Left: a co-association matrix generated by a base clustering in IWSE using mode "H". Centre: a co-association matrix generated by a base clustering in IWSE using mode "L". Right: The sum of the co-association matrices.

Finally and briefly, image segmentation is an important use of clustering for example image segmentation is used in enhancing and annotating medical images (see [71]). In Experiment B, while limited, it was demonstrated how IWSE can be useful in an imbalanced image segmentation use case. This experiment demonstrates some initial promise on real-world data. The practical application of the proposed clustering approach will be further investigated in the followed chapter.

In conclusion, this chapter shows some promising initial research into how instance weights could be used to perturb sub-sampling in the generative mechanism for a spectral bagging ensemble, enhancing its robustness to challenging structures and enhancing clustering performance.

6.6 Future Work

To make IWSE easier to use m_{min} and m_{max} could be simplified, to a mean sample value and co-variance value, this allows the range of sample sizes and their average size to be controlled independently, rather than by adjusting two values together. This would simplify hyper-parameter optimisation for the IWSE approaches.

Looking over the results and weighting the created by the exponential kernel using the Silverman bandwidth choice on the artificial, some cases, such as "2d-3c-no123", "Zelnik5", "Zelnik6" and "disctubes". Tuning the bandwidth manually may be have been beneficial as is possible that the m_{min} and m_{max} choices could have been too small. In Figure 6.12, notice that the weight within a particular clusters is effected by the nearby clusters (see "2d-3c-no123", "Zelnik5" and "Zelnik6" for examples of interactions between clusters) or even the noise in the uniform distribution (see "disctubes" hotspots) these artefacts are potentially causing unwanted sampling behaviour by IWSE. Causing fragmentation in the partitions.

IWSE weights instances, but Zhang [49] highlights the importance of weighting the base clusterings. This could be easily integrated into my method. The base clusterings could be compared in terms of pair-wise similarity, mutual information or other intrinsic quality metrics to establish which should have the greatest impact on the clustering outcome. Naturally, the feature weighting techniques could be integrated too [26, 72].

It is also suggested that weighting methods could combine several metrics to

encode as much information about the dataset as possible [49, 65]. The IWSEU approach only uses density (and inverse density), but could be extended with other weighting schemes to emphasise different aspects of a dataset. This is pertinent because relying on density information alone is imperfect, since density variation does not always accompany a difference in cluster sizes (in terms of instance count). As suggestions to integrate with our approach, methods from previous research have used angular information or information about the connectivity of the clusters [9, 25] to encode information beyond the density and further perturb the samples for the bagging ensemble proposed.

This work focused on the generative mechanism, and used CSPA with spectral clustering for the consensus function. However other works [65] have found that HPGA can produce better results, in terms of clustering performance and computational cost, hence further study into combing the best practices regarding the generative mechanism and consensus function could yield a higher performance algorithm.

Another consideration is multi-view clustering. This is currently a very active research area and very relevant to high-dimensional data collected across a number of modalities, as is typical in Big Data Analysis. Multi-view clustering involves selecting different sub-sets of features to cluster and combines the results. Our method could complement a multi-view ensemble approach. Furthermore, the inherent dimensionality reduction in this technique would allow my kernel based methods to work without suffering from the curse of dimensionality.

Finally, inspired by "Evidence Accumulation Clustering"'s split and merge

strategy, a further improvement could be executing the bags with a k value sometimes higher than the number natural/expected clusters, while still using the natural k value for the consensus clustering. This could possibly enhance the clustering performance of the approach as seen in [63].

Chapter 7

Instance Weighting Clustering for Character Clustering

7.1 Introduction

The previous chapter introduced an instance weighted clustering ensemble framework using the spectral algorithm and showed the approach had some merit on imbalanced simple benchmark and artificial datasets. To further this investigation this chapter conducts more trials with IWSE approach. The aim of this chapter is to answer RQ3 *"Under what conditions does instance weighting enhance clustering performance on data characterized by the presence of outliers or class imbalance?"* assessing the conditions in which IWSE can increase clustering performance on more complex datasets. To assert if the instance weighted spectral ensemble findings based on artificial and diagnostic datasets extrapolate to more complex data.

To achieve this aim, the objective of this chapter is to investigate the application of IWSE upon image data. This is a worthwhile objective since an

increasing multi-media rich world, (thanks to developments such as image and video based social-media platforms, and advanced computer vision embedded into automotive systems, modern data warehouses storing all this data) an ever important area for clustering is image data.

Clustering can be applied to image datasets in two ways: 1. At the image-by-image level, separating images. Torres et al. provides a recent example of this [73]. In their work, images from Synthetic-Aperture Radar (SAR) are cropped into image tiles and clustered to identify different types of vegetation, (specifically crops). 2. The other way in which clustering can be applied to image data is on individual images to segment areas within the image. This can be highly useful medical in a context to enhance or emphasis certain parts of medical images. For example in Sinha et al.'s work, clustering is used to enhance fundus photography[1] for the purpose of identifying damage to blood vessels (typically caused by diabetic retinopathy) [71]. This is accomplished by segmenting the images using clustering. Then using the under utilised green and blue channels (naturally retinae are mostly red) the different areas of the image can be differentiated, making it easier for trained staff to identify pathologies of the retina.

While agriculture and medicine are important areas for data science to be applied, they require domain expertise to investigate thoroughly. On the other hand, the MNIST handwritten digits dataset [74] is much more accessible and does not require specialist knowledge to interpret. The MNIST handwrit-

[1]Fundus photography is a non-invasive medical procedure that uses a specialized camera to capture images of the inside-back of the human eye. This area, called the fundus, includes the retina, macula, and retinal blood vessels. By illuminating the fundus through the pupil (which is dilated using medicated eye drops) and then capturing the reflected light, the images produced are useful for diagnosing and monitoring conditions.

ten digits is very well researched dataset, and highly optimal approaches have been suggested for classifying this dataset. However, the focus of this work not proposing an more optimal strategy for character classification, rather the purpose is to investigate clustering of imbalanced datasets of images. The MNIST handwritten digits dataset complexity is derived from the natural variation in how handwritten digits can be drawn. This should provide a meaningfully complex challenge for the IWSE, while remaining within a simple to reason about domain. Through investigating, if and where the IWSE approach is effective on MNIST then these findings may be translated into other domain requiring specialist knowledge.

7.2 Related Work

In this subchapter, literature related to clustering, images datasets and handling imbalance is investigated. The aim of this literature review is to identify the extent to which imbalance has been investigated and the extent to which any techniques benefit clustering performance. This review will give particular attention to the MNIST handwritten digits dataset, and challenge of separating the digits using clustering.

7.2.1 Clustering Image Datasets

Pei and Ye investigated the application of k-means and Mini-Batch k-means on MNIST digits dataset, with the purpose of evaluating the effectiveness of clustering techniques applied to the MNIST digits dataset [75]. No particular preprocessing was applied, although after the clustering step t-Distributed

Stochastic Neighbour Embedding (t-SNE) was used to visualise results seen in the paper. Using Mini-Batch k-means, 87% accuracy was achieved. This performance surpasses some classification techniques, which have the benefit of the label information for training. This shows that clustering is an effective tool for this task. Furthermore, based on the results, it was found that "0", "6", and "8" were the easiest digits for k-means and Mini-Batch k-means to identify, while "1" was very difficult. It was found that "4", "7" and "9" are hard to distinguish, as are "3" and "5", the author reasonably hypothesises that this is due to their somewhat similar shape. It was concluded that further experimentation with other clustering algorithms would likely surpass the accuracy achieved. Unlike my work this work focuses on partitioning-based clustering, and does not address imbalanced data. As described, this work provides useful insights into the challenge of differentiating the different characters within the MNIST dataset.

Another work performing clustering on the MNIST digits dataset is [76]. Pourmohammad et al.'s research highlights the benefit of preprocessing the digits prior to analysis. The preprocessing steps are as follows, a pixel cut (to crop-out/remove inactive pixels), Principle Component Analysis (PCA), k-means, Linear Discriminant Analysis (LDA) then finally a Bayesian discriminator using Mahalonobis distance is applied to provide the final cluster labels. Interestingly, this approach uses k-means as a preprocessing step. The label from k-means analysis enables the use of LDA. LDA transforms the dataset to have better separation between the clusters. Values between 32-to-96 principal components and 9-to-69 LDA components were experimented with. PCA 32 followed by LDA 29 was recommended for best performance with minimal computational cost. The approach outperforms [75], achieving an error-rate of

3.5% across the full dataset with the best setting found.

Similar to [75] it was again found that “3”, “7” and “9” were difficult to distinguish. Furthermore, Pourmohammad et al. highlights that each character is not unimodal, meaning that, there are several stereotypical forms of each character [76]. It was suggested to cluster the digits that have high error, using a higher k-value to model the different stereotypical ways that certain digits can be written.

Yang et al. applies a deep clustering approach [77]. In deep clustering, the data is embedded into a subspace (using an auto-encoder/decoder) in which it can be more easily separated. In this study spectral clustering is used to cluster the digits. The approach was empirically validated using datasets including, MNIST digits, Fashion MNIST, US Postal Service Digits and the YouTube Faces Dataset. The approach was highly effective and achieved 0.94 NMI score on the full MNIST dataset. This success highlights the value of an effective embedding strategy. The results show that deep clustering is robust against noise, but does not conclude if it is robust to class imbalance.

The studies presented thus far provide evidence that image datasets such as the MNIST digits dataset can separated with a high degree of accuracy using clustering techniques. Two studies also highlight the variable difficulty of separating out certain digits.

7.2.2 Imbalanced Clustering

Similar to [76], Singh and Dhall use clustering as a preprocessing step for classification [78]. Singh and Dhall proposes a method called Cluster-Based

Over-Sampling (CBOS). CBOS finds the centres of distributions in data and calculates the Euclidean distance of a given point to the nearest centre to decide how many points to create around that point. Therefore, their method effectively calculates a weight per-instance. To empirically validate CBOS, the MNIST dataset was used. Four imbalanced datasets were generated. One containing the digits "1" and "4" with an imbalance of 6%. A second containing the digits "2" and "6" with a imbalance of 8%. A third containing the digits "5" and "7" with an imbalance of 12%. Finally, a fourth containing the digits "8" and "9" with an imbalance 10%. They also trialled the SPI dataset (from the SAPA Personality Inventory) and an automotive insurance dataset with a 5% and 6% imbalance respectively, both are structured datasets. Unlike my work they train and use a classification algorithm (DNN) with the label information. It was found that (1, 4) achieved a lowest accuracy of the MNIST datasets trialled and that (5, 7) received the highest accuracy. This is an interesting finding as it is shows that weighting based methods can support the separation of digits with a imbalance of 12%.

Rezaei et al.'s proposed approach StatDEC uses two DNNs, one to assess the imbalance and another perform the clustering [79]. A key contribution is the application of meta-learning for the modelling of the statistics of a dataset. StatDEC uses a "statistic pooling block layer" in a DNN model which considers cardinality, mean and variance. The MNIST digits dataset, as well as three imbalanced image datasets CIFAR-10, CIFAR-100 and Refuge-2 Glaucoma X-ray dataset are utilised to empirically validate the approach. The datasets have an imbalance of 1:10, 1:100, and 1:30. The work investigates the different ways imbalance can occur in multi-distribution datasets (stepped or long-tail). Three different ways to address imbalance are identified. Through modify-

ing the distribution, by adjusting cost-functions and (the authors approach) through statistical representations using meta-learning. On the CIFAR-10 imbalanced dataset, it was found StatDEC increased NMI by 20%.

Yudong He proposes an evolution of the fuzzy k-means algorithm called Equilibrium K-Means (EKM) [67]. Yudong He discusses that the "Uniform Effect" appears in many clustering algorithms. The uniform effect is the tendency of clustering algorithms to produce uniformly sized clusters. In the case of partitioning-based algorithms such as k-means this manifests as the clumping of centroids. This is a serious problem in the case of imbalanced data clustering. The proposed method EKM implements repulsive forces between the centroids to discourage them from clumping together on the same distribution/cluster. EKM was empirically validated using synthetic and real datasets, including the MNIST digits dataset, using digits "0", "2", "3", "4", "5", "6", "7", "8" with a max imbalance of ∼1:7, where "0" has 6900 digits and the other digits have 1000. When handling the MNIST dataset, deep clustering (an auto-encoder and decoder, were jointly trained to reduce reconstruction error) was used to reduce the dataset to 10 features. EKM increased the NMI score on the described imbalanced NMIST digits dataset by 12% compared to Fuzzy K-Means.

Additionally, Yudong He highlights two different ways the problem of imbalance clustering can be overcome; through weighting (as EKM uses) and Multi-prototype clustering.

Another contribution is the production of a diagnostic image for testing clustering techniques for imbalanced datasets. This image contains items (coloured

plastic coins and a pen) on table. There is a class imbalance in terms of area of background (in pixels) compared to the area of the items (in pixels), also there is further imbalance between items themselves. It is shown that k-means, fuzzy k-means and maximum entropy fuzzy clustering cannot accurately segment the image while EKM can successfully segment the items in the image from the background.

You et al. proposes a novel method for defining a subspace for clustering [80]. Their subspace method is designed to represent data well, even in the case of imbalance, noise and big data. The algorithm works by selecting a number of data points using an optimised Farthest-First search with a cost function based on sparse-subspace clustering which was updated to perform better given imbalanced data. The approach was empirically validated using the Extended MNIST (EMNIST) dataset. The EMINST is similar to the MNIST digits dataset however, rather the numbers, it contains the English alphabet. EMINST has some imbalance, $\sim$1:16 between the most frequent letter "e" and the least frequent letter "j". Additionally, the German traffic Sign Recognition Benchmark GTSRB dataset was used. PCA was applied to reduce the dimensionality to 500 dimensions for both datasets. The proposed subspace spectral clustering method performed 10% more accurately than spectral clustering alone on the EMINST dataset.

7.2.3 Conclusion

Together these studies provide insights into how clustering algorithms have been applied to complex image datasets, such the MNIST digits dataset and others. All but [75] emphasize feature reduction has an important step. The studies showing consideration towards feature reduction achieve the best ac-

curacy. In terms of preprocessing, different theories regarding how to do this exist. Deep clustering is utilised by [77, 79, 67] and good results are achieved. Conversely, [76] uses traditional feature reduction methods and also achieves a good accuracy.

There is some literature discussing clustering imbalanced datasets, and a number of methods are proposed. Weighting-based techniques, multi-prototype clustering, modification of the distribution, adjusting cost-functions, and using meta-learning. Current approaches to handling imbalance on the MNIST digits dataset show an increase in NMI score of between 10% to 12% [67, 78, 80]. On other datasets, state-of-the-art methods can increase NMI as much as 20% [79]. Overall, it appears that much of the current literature on imbalanced clustering pays particular attention to k-means and similar methods, however, there is a relatively small body of literature that is concerned with non-partitioning algorithms such as spectral clustering (graph-based) despite this being a more recent and sophisticated technique for clustering complex data, leaving a gap for this work.

7.3 Experimental Design

To address the partial gap found on applying non-partitioning-based algorithms to complex imbalanced datasets and to investigate the effectiveness of instance weighting on a real-world dataset (in particular MNIST handwritten digits) the following steps will be taken:

Intra and inter analysis of MNIST handwritten digits: Individual digits will be compared to other digits of the same class (this will be referred to as "intra-

digit" analysis). Additionally, digits will be compared to digits of other classes (this will be referred to as "inter-digit" analysis). This analysis will inform the design of the experiments and analysis of the results.

Imbalanced clustering experiment using MNIST handwritten digits: Experiments will be conducted to evaluate the effectiveness of spectral clustering and IWSE, on balanced and very imbalanced versions of the MNIST handwritten digits dataset. The goal is ascertain the effectiveness of instance weighting with graph-based algorithms on complex real-world imbalanced data.

Synthetic experiments: Experiments will be used to isolate and generalise the conditions under which instance weighting can enhance clustering performance on imbalanced data. The goal is to support findings from the MNIST dataset by conducting experiments to infer what aspects of the real-world data are influencing the performance of the approach. This information should be useful to understand the required properties a dataset should have for it to be worthwhile to apply the IWSE approach.

Diagnostic segmentation experiment: Further to the above experimentation, IWSE will be trialled for the use case of image segmentation using a diagnostic imbalanced image.

7.4 Investigation of MNIST Digits

The MNIST digits dataset was chosen as it well-researched and has complexity level suitable for meaningfully testing machine learning techniques. The

MNIST handwritten digits dataset contains 70,000 instances, each with 28x28 (784 pixels) greyscale pixels. The dataset is balanced and contains the handwritten digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 in equal quantity. This dataset is traditionally used for classification based tasks and thus includes class labels. The class labels identify which digit each handwritten digit represents. These labels will be considered the ground truth for clustering quality assessment and will only be used to assess clustering performance.

An interesting feature of the MNIST handwritten digits dataset is that some digits can be written in a variety of ways (see Figure 7.10). Ultimately, this adds complexity to the distributions that represent the different digits. This adds a layer of challenge for clustering approaches applied to this dataset.

7.4.1 Intra-Digit Analysis

For the intra-digit analysis the "training" portion of the dataset was sampled, using simple random sampling without replacement.

The first analysis conducted applied the Simple Matching Coefficient (SMC) pixel-wise between all pairs of digits in the sample. See Equation (7.1) where a, b, c, and d are counts representing the frequency of different types of matches between two instances a = 1-1 (match), b = 0-1 (mismatch), c = 1-0 (mismatch), d = 0-0 (match). To enable the SMC to be applied, a threshold function (see Equation (7.2)) was applied to the greyscale pixel values (which vary between 0-255). The result of the threshold function can be seen in Figure 7.1. The sample size was 200 of each class. Sampling 200 digits randomly proved to be more than adequate to ensure repeatable results. The average

SMC is calculated for each of the pairs of digits within the sample of 200 digits representing the same number and this is shown in Figure 7.2.

$$SMC = \frac{b + c}{a + b + c + d} \tag{7.1}$$

$$f(x) = \begin{cases} 1 & \text{if } x \geq 128 \\ 0, & \text{otherwise} \end{cases} \tag{7.2}$$

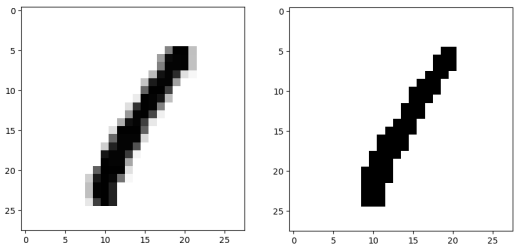


Figure 7.1: An MNIST digit before the threshold function is applied (left) and after the threshold function is applied (right).

Figure 7.2 suggests that "0" and "2" are the most variable digits and that "1" and "7" are the most consistent. However, this measure is partly skewed as a lot of the matching is likely coming from 0 - 0 matches. For example with "1"'s there are less coloured pixels, and thus more blank pixels to match.

The second analysis utilised NMI. As with the previous analysis, NMI is applied pixel-wise after the threshold function has been applied. As before, the values shown in Figure 7.3 represent the average NMI calculated between the digits representing the same number. Again, the sample size was 200 of each class. Sampling 200 digits randomly proved to be more than adequate to ensure repeatable results.

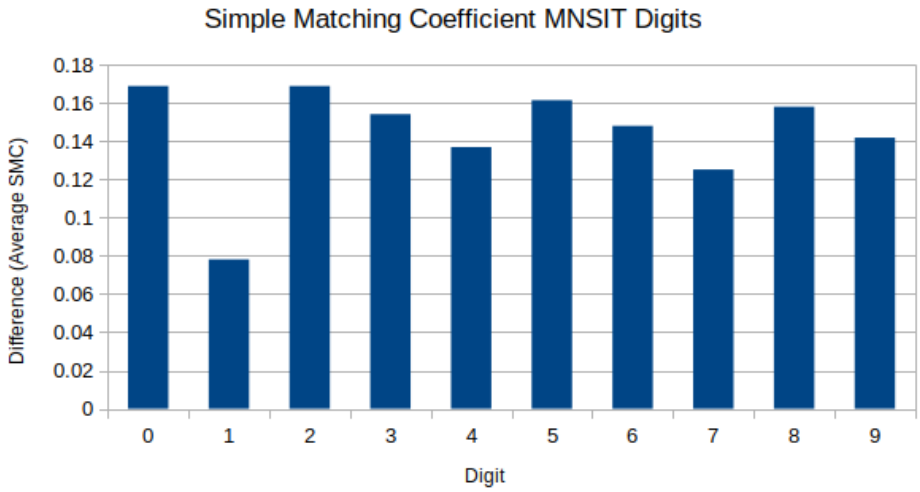


Figure 7.2: A sample of 200 of each class of digit compared per pixel using Simple Matching Coefficient.

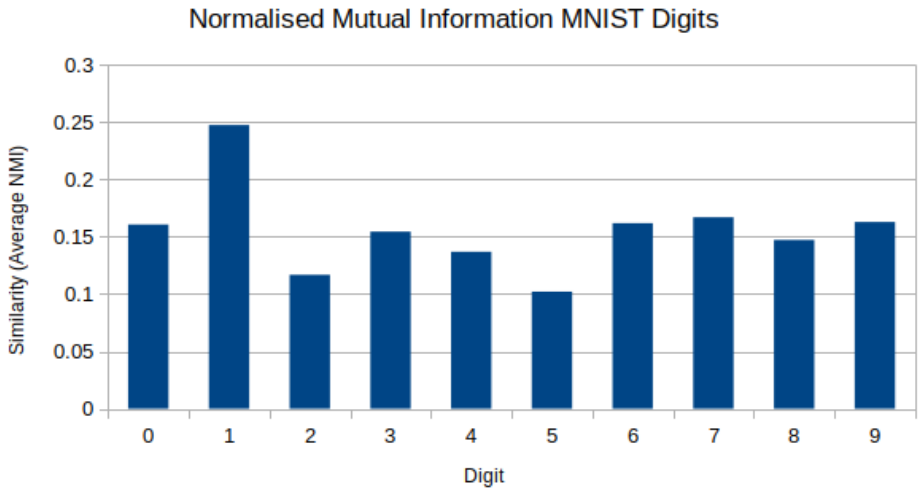


Figure 7.3: A sample of 200 of each class of digit compared per pixel using Normalised Mutual Information.

Unlike with the previous analysis using SMC, NMI is a measure of similarity, thus a high value indicates consistency. In Figure 7.3, it can be seen that "1" is less varied (meaning that "1"'s are most similar with each other), which supports the finding based on SMC. Furthermore, NMI indicates that "2" and

"5" are most varied, this agrees with SMC too.

The third analysis calculates the first and second principal components. Then in this space calculates the 25^{th} and 75^{th} percentiles for each of the digit classes in each of the 2 dimensions. The area of the space within the 25^{th} to 75^{th} percentiles for each digit is then calculated and is shown in Figure 7.4. For this analysis, the sample size was 800 per digit. Sampling 800 of each digit randomly proved to be more than adequate to ensure repeatable results.

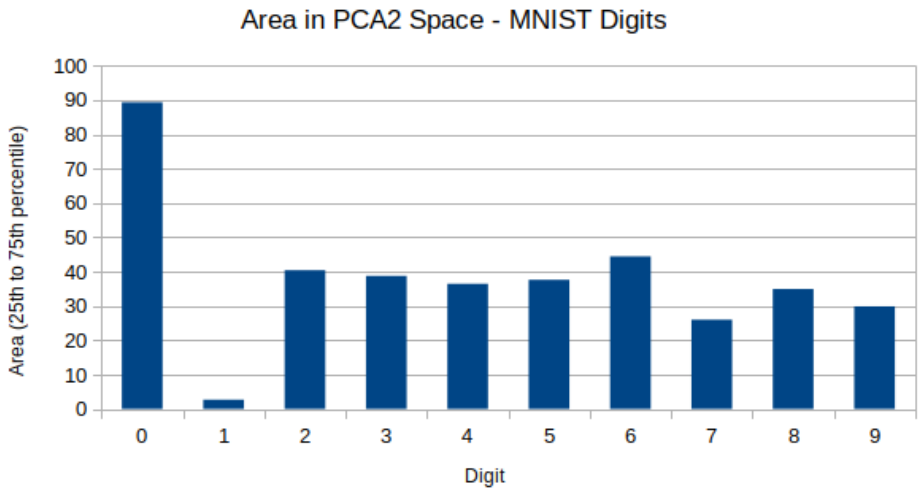


Figure 7.4: Based on a sample of 800 of each class of digit, the average area of the 25^{th} to 75^{th} percentile in PCA2 sub-space.

Figure 7.4 suggests that "0" is very variable and that "1" is very consistent (followed by "7" and "9").

In the forth and final intra-digit analysis, the average density is calculated. All aspects are similar to the previous analysis, however in this analysis, rather than average area of the 25^{th} and 75^{th} percentiles being calculated, the exponential density kernel is used to estimate the density for each instance within

the space. The average density for each class digit is min-max normalised and shown in Figure 7.5.

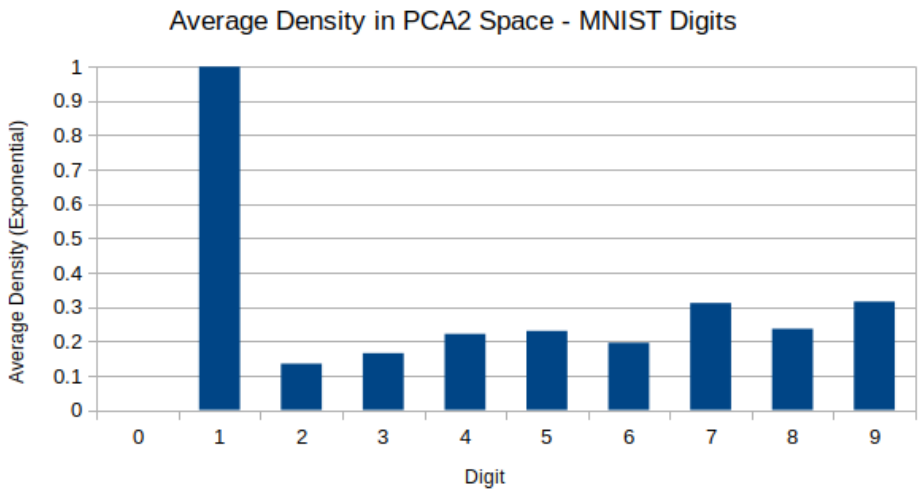


Figure 7.5: Based on a sample of 800 of each class of digit, the athematic mean density of digits in a PCA2 sub-space.

Figure 7.5 suggests that "1" is much more dense than the other digits and therefore less varied. "2", "3", "4", "5", "6", "8" have similar density. "7" and "9" are slightly less varied. While "0" is the most diffuse and varied. In summary, "0" and "1" are different in terms of density and variation from the other digits. While the others have relatively similar levels of density/variation.

7.4.2 Inter-Digit Analysis

As with the intra-digit analysis, the inter-digit analysis uses the "training" portion of the dataset. Simple random sampling without replacement was used to select a sample for analysis.

Firstly, SMC was used pixel-wise to compare samples of 200 instances of each digit from two different classes of digit. As with the intra-digit analysis, to enable the SMC to be applied, a threshold function (see Equation (7.2)) was applied. Randomly sampling 200 of each of the two digit classes compared proved to be more than adequate to ensure repeatable results.

	0	1	2	3	4	5	6	7	8	9	Mean
0		0.20	0.21	0.20	0.21	0.19	0.20	0.20	0.21	0.20	**0.202**
1	0.20		0.16	0.15	0.15	0.15	0.15	0.14	0.15	0.14	0.155
2	0.21	0.16		0.19	0.18	0.19	0.18	0.18	0.18	0.18	**0.185**
3	0.20	0.15	0.19		0.19	0.17	0.19	0.18	0.17	0.18	**0.180**
4	0.21	0.15	0.18	0.19		0.18	0.17	0.15	0.17	0.15	0.172
5	0.19	0.15	0.19	0.17	0.18		0.18	0.17	0.17	0.17	0.175
6	0.20	0.15	0.18	0.19	0.17	0.18		0.18	0.18	0.17	**0.178**
7	0.20	0.14	0.18	0.18	0.15	0.17	0.18		0.17	0.14	0.168
8	0.21	0.15	0.18	0.17	0.17	0.17	0.18	0.17		0.16	0.175
9	0.20	0.14	0.18	0.18	0.15	0.17	0.17	0.14	0.16		0.166

Figure 7.6: A sample of 200 of each digit class compared with each other using a pixel-wise Simple Matching Coefficient.

Figure 7.6 suggests that digit "0" is most distinguishable (highest mismatch) followed by "2", "3" and "6". "1" had the lowest value suggesting that "1" is similar to other digits. Although, this could be skewed by the imbalance of black versus white pixels, as "1" is likely to have more white pixels to match, as it is the character requiring the least ink to write.

The next analysis uses NMI to compare digits. As with the previous analysis, 200 of each digit class is sampled and compared after applying a threshold function.

In Figure 7.7 the shading is inverted to be consistent in meaning with others, as conversely to SMC (for example), a high NMI value indicates similarity and low value indicates disagreement. Figure 7.7 suggests that "0" is the most unique

	0	1	2	3	4	5	6	7	8	9	Mean
0		0.022	0.059	0.069	0.047	0.075	0.071	0.048	0.071	0.056	**0.058**
1	0.022		0.073	0.079	0.051	0.056	0.062	0.064	0.109	0.064	**0.064**
2	0.059	0.073		0.075	0.056	0.047	0.078	0.048	0.082	0.057	**0.064**
3	0.069	0.079	0.075		0.056	0.085	0.059	0.058	0.104	0.07	0.073
4	0.047	0.051	0.056	0.056		0.059	0.077	0.087	0.082	0.122	0.071
5	0.075	0.056	0.047	0.085	0.059		0.064	0.055	0.093	0.079	**0.068**
6	0.071	0.062	0.078	0.059	0.077	0.064		0.045	0.085	0.076	**0.069**
7	0.048	0.064	0.048	0.058	0.087	0.055	0.045		0.076	0.121	**0.067**
8	0.071	0.109	0.082	0.104	0.082	0.093	0.085	0.076		0.098	0.089
9	0.056	0.064	0.057	0.07	0.122	0.079	0.076	0.121	0.098		0.082

Figure 7.7: A sample of 200 of each digit class compared with each other using a pixel-wise Normalised Mutual Information.

digit followed by “1”, “2”, “5”, “6” and “7”. This metric agrees with SMC in that “0”, “2” and “6” are very unique characters. Looking at specific pairs, the most different digits according to NMI is (“0” and “1”) and the most similar digits are (“4” and “9”) closely followed by (“7” and “9”). Anecdotally, this seems correct, (“0” and “1”) are most easily visually distinguishable. While “4” or a “7” can be modified to represent a “9” with minimal ink/modification.

The third inter-digit analysis calculates the first and second principal components, then in this space calculates the Euclidean distance between mean centroids of the different classes of digit. Figure 7.8 shows the results. For this analysis, the sample size was 800 per digit. Sampling 800 of each digit randomly proved to be more than adequate to ensure repeatable results.

Figure 7.8 suggests that “0” followed by “1”, “2” and “7” are most unique universally. With again [0, 1] being most distance from each other, while (“7” and “9”) followed by (“2” and “6”) and (“5” and “8”) are most similar. This partly echoes the previous finding. Again, anecdotally (“2” and “6”) and (“5” and “8”) are somewhat mistakable characters.

	0	1	2	3	4	5	6	7	8	9	Mean
0		18.07	8.55	11.37	13.07	11.18	9.266	16.06	12.13	15.39	**12.786**
1	18.07		9.845	7.901	6.984	6.936	8.98	6.362	6.042	6.128	**8.583**
2	8.55	9.845		2.956	4.677	2.992	0.921	7.72	3.807	6.992	**5.385**
3	11.37	7.901	2.956		1.724	2.668	2.637	4.767	2.646	4.048	4.524
4	13.07	6.984	4.677	1.724		3.675	4.284	3.044	3.169	2.327	4.772
5	11.18	6.936	2.992	2.668	3.675		2.083	6.309	0.986	5.535	4.707
6	9.266	8.98	0.921	2.637	4.284	2.083		7.293	2.94	6.54	4.994
7	16.06	6.362	7.72	4.767	3.044	6.309	7.293		5.52	0.775	**6.428**
8	12.13	6.042	3.807	2.646	3.169	0.986	2.94	5.52		4.754	4.666
9	15.39	6.128	6.992	4.048	2.327	5.535	6.54	0.775	4.754		5.832

Figure 7.8: The distance between digits in a PCA2 space.

Finally, the forth analysis use PCA and t-SNE to produce sub-space in which Euclidean distance is used to access the difference between the digits. Firstly, PCA is used to reduce the dimensionality to 32 dimensions; then t-SNE is applied to reduce the 32 dimensional space down to 2. Here PCA is applied first for computation efficiency and noise suppression, then t-SNE is applied to produce the final result. This approach of using PCA before t-SNE is typical practice when applying t-SNE to high dimensional datasets [81].

	0	1	2	3	4	5	6	7	8	9	Mean
0		143.8	52.09	59.15	91.59	93.53	70.63	139.6	94.99	112.1	**95.273**
1	143.8		114.5	84.73	85.09	70	116.8	66.3	50.94	77.49	**89.955**
2	52.09	114.5		46.93	42.31	90.97	95.03	92.18	76.81	62.99	74.864
3	59.15	84.73	46.93		56.67	44.19	56.81	92.97	36.31	72.24	61.113
4	91.59	85.09	42.31	56.67		90.22	113.5	49.94	65.81	20.76	68.432
5	93.53	70	90.97	44.19	90.22		47.26	109.1	28.02	98.89	74.683
6	70.63	116.8	95.03	56.81	113.5	47.26		145.9	69.15	128.4	**93.721**
7	139.6	66.3	92.18	92.97	49.94	109.1	145.9		81.08	29.8	**89.651**
8	94.99	50.94	76.81	36.31	65.81	28.02	69.15	81.08		71.9	63.889
9	112.1	77.49	62.99	72.24	20.76	98.89	128.4	29.8	71.9		74.955

Figure 7.9: The distance between digits in a PCA32-t-SNE2 sub-space.

Figure 7.9 suggests that “0”, “1”, “6” and “7” are most unique. Again (“0” and “1”) are most distance from each other followed by (“6” and “7”) and (“0” and “7”). As with some of the previous methods (“7” and “9”) and (“4” and “9”) appear again as very similar.

7.4.3 Summary of Intra and Inter Digit Analysis

The intra-digit analysis finds that the digits have different distributions, "1" is highly consistently written and has high density. "7" and "9" have medium density and medium level consistency in how they are written. "2", "3", "4", "5", "6" and "8" have a similar lower level of density and lower level of consistency in how they are written and "0" has the lowest level of density and lowest consistency in how it is written.

Summarising the inter-digit analysis methods applied is that "0", "1", "2" are the most unique, from all digits generally. Across all methods, the general consensus is that ("0" and "1") is the most different and that ("4" and "9") and ("7" and "9") are the most similar/overlapping. Other literature such as Pei and Ye who also found "4", "7", "9" most difficult to distinguish [75].

This analysis of MNIST handwritten digits are useful for studies investigating clustering and classification of MNIST handwritten digits. These findings can aid understanding why algorithms confuse particular digits and succeed in at distinguishing others. In this research these findings will be used to meaningfully select different sub-sets of digits to analyse.

7.5 Imbalanced MNIST clustering

7.5.1 Experimental Design

To investigate how imbalance effects spectral clustering on the MNIST handwritten digits dataset, the following experiment was designed.

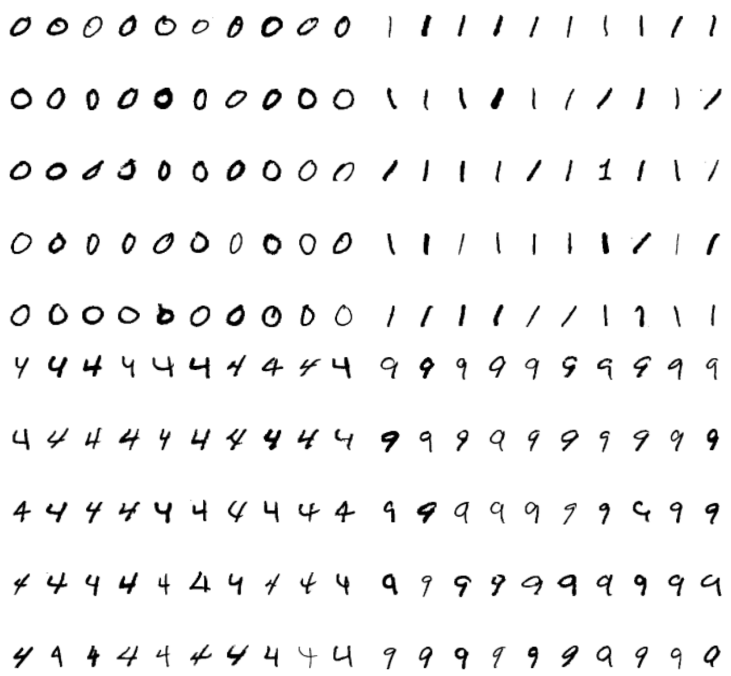


Figure 7.10: Random selection of 50 zeroes, ones, fours and nines from MNIST, zeros and ones are visually very different however, fours and nines are not so different.

Firstly, pairs of digits are carefully chosen, based on the findings of the analysis. Based on the previous analysis, it is possible to categorise different pairs of digits in terms of overlap and density. The intra-digit analysis provided insight in the density of digits. While the inter-digit analysis provided insight into similarity/overlap between digits. For example, ("0" and "1") have very different densities, and are most distant from each other. Hence, ("0" and "1") can be classified as having large difference in density and a small degree of overlap. Ultimately, this allows the construction of a table, categorising pairs of digits that have similar properties. Such a table, makes it possible to see which scenarios have been tested. Table 7.1, shows the categorisation of the digit combinations to be trialled. Note that, not all combination of density/overlap exist within the possible combinations of the MNIST digits, this is limitation of this experiment. Furthermore, some density/overlap com-

binations have many options. The combinations chosen aim to best cover as many density/overlap combinations as possible. The choice to cover only some pairs of digits was made to enable simple understanding and visualisation of the results as well as limiting the computation resources required to execute the experiments.

Table 7.1: The selection of digits to be evaluated, based on the previous analysis.

		Cluster Density Variation			
		Tiny	Small	Medium	Large
Overlap	Small	No such example in MNIST	No such example in MNIST	1-2	0-1
	Medium	2-7	0-3, 0-4, 0-7, 0-8, 0-9	1-3, 1-5, 1-6, 1-8	No such example in MNIST
	Large	2-4, 3-5, 3-7, 5-7, 6-7, 6-8, 7-8, 7-9	0-2, 0-5, 0-6	1-4, 1-7, 1-9	No such example in MNIST

In order to investigate imbalance, the MNIST handwritten digits training dataset was sampled in various ratios of each digit in the combinations. The datasets trialled were sampled with an imbalance ranging from an extra 480% (in steps of 20%) of each digit in the pair while keeping the total number of instances in consistent at 2000. To achieve this, Equation (7.3), is proposed.

Given the combination of ("0" and "1") at 0% imbalance there would be 1000 zeros and 1000 ones (perfectly balanced). At 200% imbalance there would be 1500 zeros and 500 ones, (1000 more zeros (200% of 500) than ones). At 400% imbalance there would be 1666 zeros and 334 ones (1332 more zeros (400% of 334) than ones). Conversely, at -200% imbalance there would be 500 zeros and 1500 ones. At -400% imbalance there would be 334 zeros and 1666 ones. To approximate the integer values for various degrees of imbalance percentage,

Equation (7.3) is devised. Where N is the number of instances. N remains consistent at 2000 in this experiment. i is the imbalance factor (for example 400 (percent)). L is the large side of the ratio, and S is the small side of the ratio. Worked examples are provided in Appendix 9.7. Using the proposed equation enables the experiments to smoothly increment the imbalance using a single variable i while maintaining a consistent N. Prior research has mostly omitted devising such an experiment in an instance weighted clustering context, see Section 2.4.

$$\begin{aligned} L &= \lfloor \frac{N}{2 + (\frac{i}{100})} \times (1 + (\frac{i}{100})) \rfloor \\ S &= N - L \end{aligned} \tag{7.3}$$

To compare the effectiveness of instance weighting against traditional methods, spectral (S), Spectral Ensemble (SER), and Instanced Weighted Spectral Ensemble Union (IWSEU) will be executed on each imbalance condition with a randomly chosen sample of data for 5 repetitions for each of the chosen pairs of digits. Spectral clustering provides a baseline result for graph clustering. Spectral Ensemble (with random weighting) will provide another baseline showing the effectiveness of spectral ensemble clustering and my approach IWSEU, shows the benefit of instance weighting integrated into a spectral clustering ensemble. To assess the impact of the instance weighting, settings between the algorithms were kept consistent. For spectral, k was set to 2, the affinity matrix used 10 nearest neighbours. For the Spectral Ensemble, again k was set to 2, and the affinity matrix used 10 nearest neighbours, the bagging-based ensemble used 36 bags, and uniformly randomly sampled between 30-50% of

the instances per bag. For IWSEU, again k was set to 2, and the affinity matrix used 10 nearest neighbours, the ensemble used 36 bags, sampling between 30-50% of instances per bag using weighted random sampling based on the instance weights from the density estimation using the exponential function. The bandwidth value for the kernel was chosen dynamically using the Silverman method.

Prior to applying each of the clustering algorithms, each of the randomly sampled experimental datasets was standard scaled (z-score normalised). After scaling, PCA was applied to reduce the dataset to either 3 or 6 dimensions, the results of both are presented. A known limitation of applying kernel based density estimation methods is that they fail to recognise dense areas on high dimensional datasets. Hence, only 3 and 6 principal components were trialled. Preliminary tests showed that much beyond 6 dimensions will render the density estimation ineffective. Three principal components was has selected, while loses more information, it benefits from be possible to visual for analysis of the clustering results.

7.5.2 Results and Discussion

For each algorithm, executing on each combination of digits, a series of data is produced, and this information is shown in Figures 9.15 to 9.66. As multiple executions were completed the mean value (the plotted line) and variance (the corresponding shaded area) are shown. Figures 9.15 to 9.66 use a two sample t-test to annotate where there is a significant difference (p-value $<$ 0.05) between IWSEU (purple) and SER (orange) (shown by a vertical green or red line). A vertical green line indicates the IWSEU significantly outper-

formed SER (indicating instance weighting is beneficial), while a vertical red line indicates that SER outperformed IWSEU (indicating instance weighting is detrimental).

Firstly, considering PCA3 results seen in Table 7.2. Across the variety of digit combinations the proposed approach is shown to both decrease and increase clustering performance. In (“0” and “1”), (“0” and “6”), (“0” and “8”), (“0” and “9”), (“1” and “2”), (“1” and “3”), (“1” and “4”), (“1” and “5”), (“1” and “6”), (“1” and “7”), (“1” and “8”), (“1” and “9”), (“5” and “7”) and (“7” and “9”) instance weighting significantly increases clustering performance for some imbalance factor. The most positive result was (“0” and “1”), where the average NMI was increased by as much ∼60%. In this case, instance weighting was the sole factor separating between a unintelligible or accurate clustering. Elsewhere, results showed more modest gains in NMI: (“0” and “6”), (“0” and “8”), (“1” and “3”), (“1” and “8”) in places showed a ∼5% increase, (“5” and “7”) in places showed a ∼10% increase, (“1” and “4”) and (“1” and “5”) in places showed a ∼15% increase, (“0” and “9”), (“1” and “6”), (“1” and “7”), (“1” and “9”) in places showed a ∼20% increase. There was one poor result (“7” and “9”) where all algorithms performed poorly, in this case it seems likely that due to the similarity of digits “7” and “9” and the simplistic preprocessing choices, the selected clustering algorithms were unable to distinguish the digits.

Secondarily, looking at PCA6 results in Table 7.3 the outcome is less positive for instance weighting. There are much fewer digit combinations where IWSEU is able to meaningfully, increase clustering performance. In (“0” and “5”) and (“1” and “7”) NMI in (for certain imbalance levels) showed a ∼5% increase, (“0” and “6”) in places showed a ∼10% increase. Again (“7” and

"9") proved to be too difficult the separate for any of the algorithms. It seems that 6 dimensions is still too few principal components to separate "7" and "9". Overall, using 6 dimensions the decreased the performance gap between IWSEU and SER. But increased the overall performance in terms of NMI. The reduced gap is likely due to reduced effectiveness of IWSEU's kernel based density estimation when presented higher dimensional data.

A general observation is that IWSEU most often performs best when the majority cluster is also the higher density cluster. Notice that in ("0" and "1") and ("0" and "9") in Figures 9.15 and 9.23 respectively, IWSEU significantly outperforms when there are more ones and more nines. Although this not always the case for example see ("0" and "6") and ("0" and "8") in Figures 9.20 and 9.22 respectively.

Broadly comparing the PCA3 and PCA6 results, reveals a general pattern that using 3 less dimensions reduces clustering performance in terms of NMI by about 10%. Using 6 dimensions narrows and in some cases removes the performance advantage of IWSEU. Generally, the best overall results are IWSEU with PCA6. Other research has found that preprocessing plays a significant role in clustering performance. While outside the scope of this work, finding the best approach for the separating the MNIST digits using clustering would likely involve more attention to the preprocessing, to create a a optimal sub-space in which to apply clustering algorithms such as spectral clustering and IWSEU. Some approaches that could address this are using a pixel-cut to manually reduce dimensionality prior to automatic methods (as is done by Pourmohammad et al. [76]). Additionally, You et al. and Yang et al.'s subspace methods would likely to more effective than the Principle Component

Analaysis utilised in this work [77, 80].

To provide an overview of the results and able to discover trends, the results of significance test from Figures 9.15 – 9.66 were summarised into Tables 7.2 and 7.3. This reveals some useful and interesting patterns. When the digits combination has either a medium or large difference in density, or a large overlap between clusters IWSEU often aided performance, represented by the green and blue highlights in Tables 7.2 and 7.3 . IWSEU helped when there is a density difference between the clusters, this is a positive result and in-line the hypothesis. However, an unexpected result is that IWSEU can somewhat help in the case of overlapping clusters. Other researchers [75, 76, 78] found that the overlapping digits are most difficult to separate and thus it appears my approach maybe able help in these cases.

Table 7.2: Summary of results of the imbalanced MNIST digits datasets using **3 principal components**. Green highlight indicates IWSEU aided clustering performance for some imbalance level(s) compared to SER. Blue highlight indicates that IWSEU both aided and hindered clustering performance across the imbalance levels trialled compared to SER. Orange highlight indicates IWSEU hindered clustering performance for some imbalance level(s) compared to SER. Finally grey highlight indicates that IWSEU made no significant difference for any of the imbalance levels trialled.

		Cluster Density Variation			
		Tiny	Small	Medium	Large
Overlap	Small	No such example in MNIST	No such example in MNIST	1-2	0-1
	Medium	2-7	0-3, 0-4, 0-7, 0-8, 0-9	1-3, 1-5, 1-6, 1-8	No such example in MNIST
	Large	2-4, 3-5, 3-7, 5-7, 6-7, 6-8, 7-8, 7-9	0-2, 0-5, 0-6	1-4, 1-7, 1-9	No such example in MNIST

The size of the clustering performance gains seen in this work are similar to

Table 7.3: Summary of results of the imbalanced MNIST digits datasets **using 6 principal components**. Green highlight indicates IWSEU aided clustering performance for some imbalance level(s) compared to SER. Blue highlight indicates that IWSEU both aided and hindered clustering performance across the imbalance levels trialled compared to SER. Orange highlight indicates IWSEU hindered clustering performance for some imbalance level(s) compared to SER. Finally grey highlight indicates that IWSEU made no significant difference for any of the imbalance levels trialled.

		Cluster Density Variation			
		Tiny	Small	Medium	Large
Overlap	Small	No such example in MNIST	No such example in MNIST	1-2	0-1
	Medium	2-7	0-3, 0-4, 0-7, 0-8, 0-9	1-3, 1-5, 1-6, 1-8	No such example in MNIST
	Large	2-4, 3-5, 3-7, 5-7, 6-7, 6-8, 7-8, 7-9	0-2, 0-5, 0-6	1-4, 1-7, 1-9	No such example in MNIST

the gains seen in other works attempting to cluster imbalanced image datasets. While none of the following are a direct comparison, they give an indication to position my work within. Yudong He et al. creates and clusters an imbalanced version of the MNIST digits dataset [67]. Different to my experimentation, their experiment uses 8 out of 10 of the digit classes rather 2 out of 10 of the digit classes. In their experiments their imbalance factor is 600% rather than my 480% between the most frequent and least frequent digits. They find that their approach (called EKM) performs 12% better than the tradition alternative to their algorithm fuzzy k-means. Rezaei et al reports an increase of 20% NMI when using StatDec compared to a traditional alternative (Deep Neural Network) on the CIFAR-10 image dataset (similar to MNIST, but objects in instead of characters) with an imbalance of 900%. You et al trialled the full EMNIST (which similar MNIST digits but contains the letters of the alphabet instead) dataset which has at most 1500% imbalance factor and was able to achieve 10% increase in NMI over spectral.

7.6 Synthetic Experiments

7.6.1 Experiment Design

To enable generalisation of the findings from the previous analysis, and understand the limitations of IWSEU, synthetic experiments where conducted to isolate which properties (in terms of overlap and difference in density) a dataset should have to be effectively clustered by IWSE. Based on the theory, analysis and results converging in Tables 7.2 and 7.3, three synthetic experiments were planned.

The first experiment investigates the impact overlap has upon clustering performance. In Table 7.2, a pattern emerged that suggests that IWSEU is most beneficial when the overlap extent was large. To test this, two Gaussian distributions were used to generate two spherical clusters, C_0 and C_1. In this experiment, the independent variable is the x position of C_1. The x position of C_1 was varied from 1.5 to 0.5 in decrements of 0.01, this moves it from a relatively distant position to a position where it largely overlaps C_0, see Figure 7.11. All other aspects of the experiment were fixed. The covariance of C_0 was 0.01 in both x and y. C_0 contained 1800 instances, and C_1 contained 200 instances. Experiments were repeated 5 times and the mean NMI and it's variance is shown in Figure 7.15.

The second experiment investigates the impact of the difference in density between clusters. Again, in Table 7.2, a pattern could be seen that suggested that IWSEU is most beneficial when density difference between cluster was

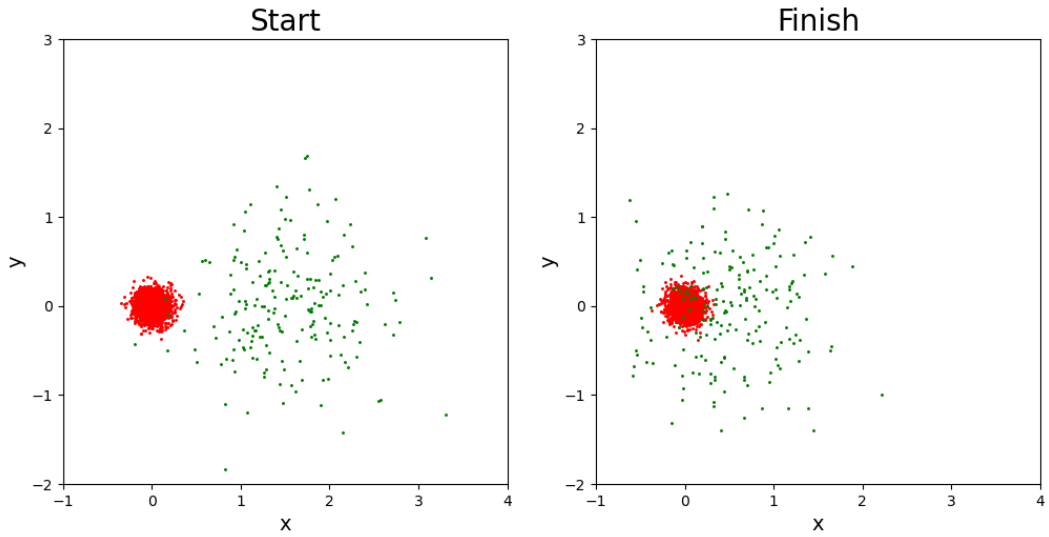


Figure 7.11: Synthetic experiment 1 (overlap extent) – a sample of the start and finish datasets.

large. Hence, in this experiment, there are two clusters C_0 and C_1 which start with equal density. The density of C_1 is linearly decreased by increasing the covariance of it's distribution. Figure 7.13 shows how the density decreases approximately linearly over the experiment. However, increasing the covariance of C_1 naturally increases the overlap. As C_1 grows spacially so it would encroach upon C_0, creating a confounded variable in experiment. To adjust for this, C_1 is moved in the x dimension proportionally in response to the increase in co-variance using $o + (v \times 2)$, where o is a fixed offset of 1.2, and v is covariance of C_1. In Figure 7.12 it can be seen that the C_0 and C_1 are equally adjacent to despite the change in spacial size of C_1. The covariance of C_1 was varied from 0.05 to 0.27 in increments of 0.002. As in the previous experiment, C_0 had 1800 instances and C_1 had 200 instances. Experiments were repeated 5 times and the mean NMI and it's variance is shown in Figure 7.16.

The third experiment investigates the imbalance extent between the clusters. In Figures 9.15 to 9.40, it could be seen that extent to which IWSEU aided

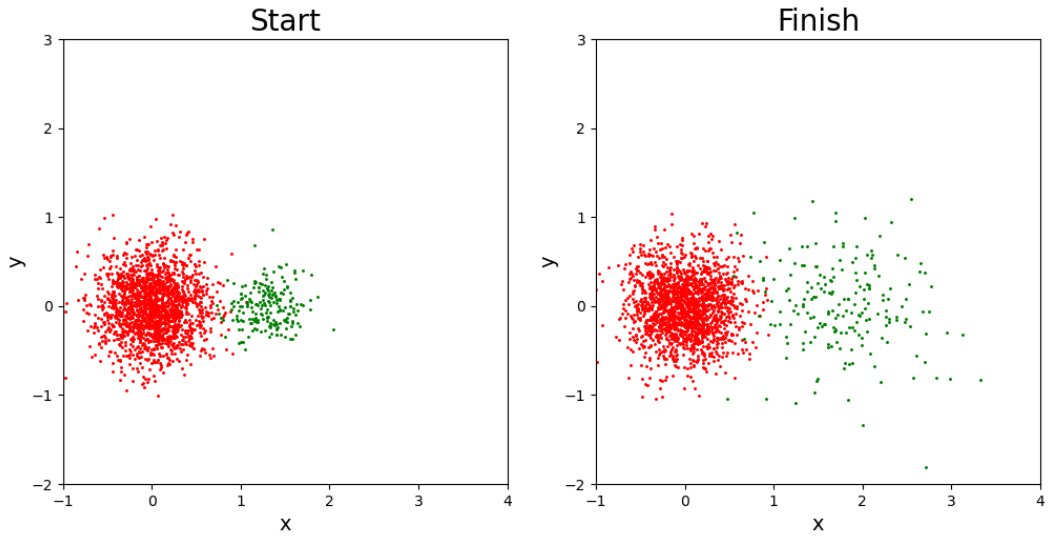


Figure 7.12: Synthetic experiment 2 (density difference) – a sample of the start and finish datasets.

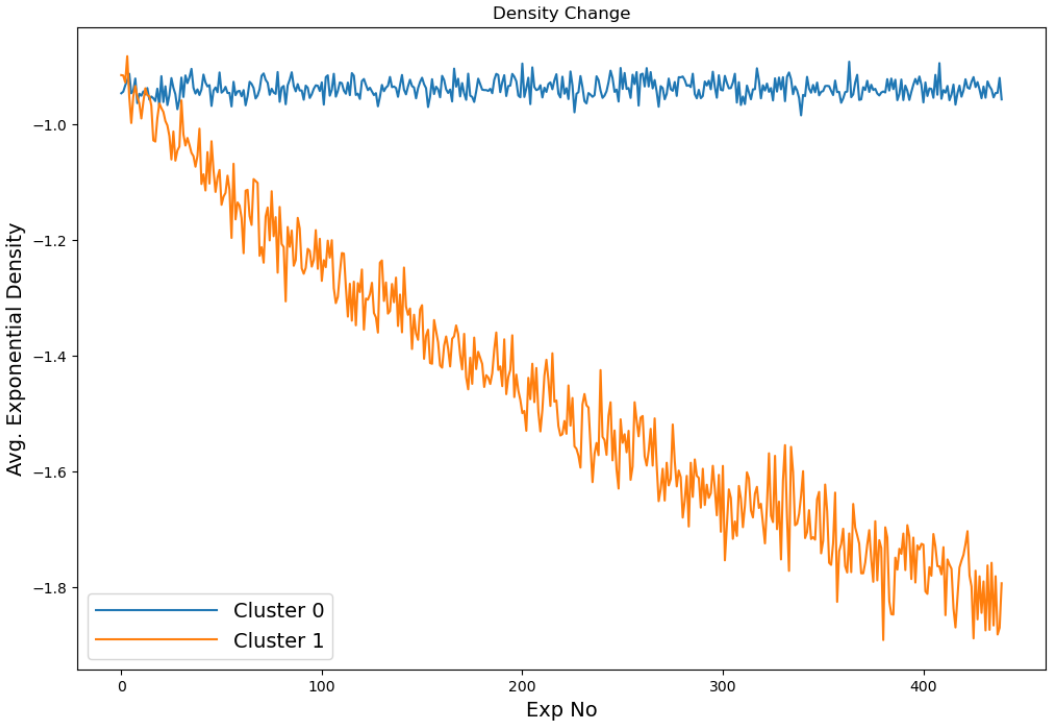


Figure 7.13: Synthetic experiment 2 (density difference) - showing the linear decrease in density.

performance varied across imbalance factors. Generally, it was observed that IWSEU performed best when the majority cluster was the denser cluster (for

example see ("0" and "1")). To investigate the impact of imbalance between the clusters. The imbalance factor between C_0 and C_1 was varied. At the start of the experiment the clusters are perfectly balanced with 1000 instances each. Over the experiment, the size of C_0 is incrementally increased by 10 instances and C_1 was incrementally decreased by 10 instances, until the size of C_1 reached 50 instances, see Figure 7.14. To replicate the conditions in which IWSEU performed well in the MNIST experiments, C_0 (which becomes the more numerous cluster over the experiment) had a co-variance of 0.1 in both x and y, and C_1 had a covariance of 0.2. C_0 and C_1 were separated by a distance of 0.8 in the x dimension. It should be noted that this experiment has some limitations. Since as the numerosity of each cluster changes, so does it's average density, hence the this simple experiment should be interpreted with this in mind. Experiments were repeated 5 times and the mean NMI and it's variance is shown in Figure 7.17.

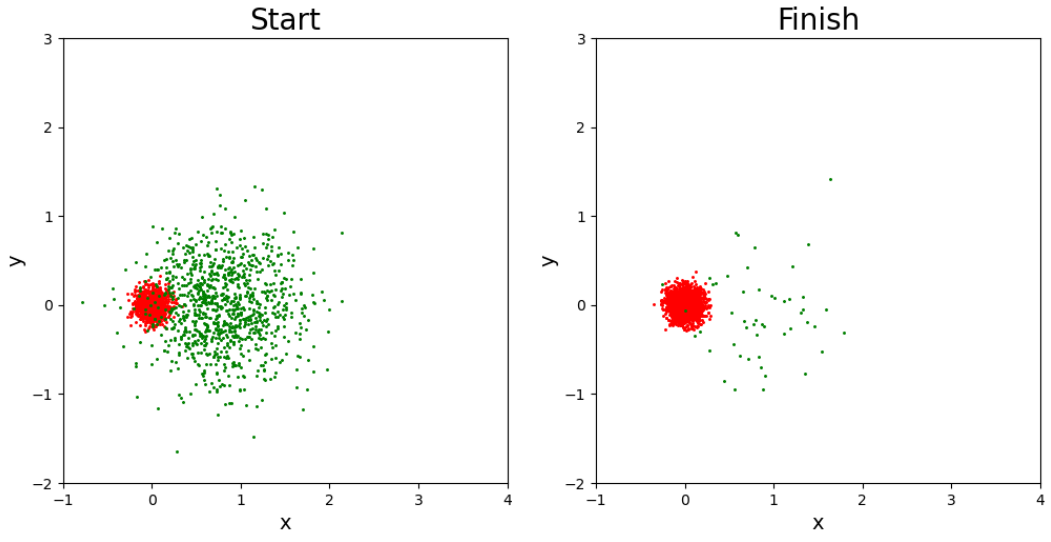


Figure 7.14: Synthetic experiment 3 (imbalance) – a sample of the start and finish datasets.

7.6.2 Results and Discussion

The first experiment investigates how the overlap of clusters effects the clustering performance of instance weighting. Initially, when the clusters are well separated the all algorithms perform well, however as the distance between the cluster decreases (to ∼1.3) and the overlap becomes non-trivial the performance of traditional methods drops, while IWSEL and IWSEU continue to perform well, this trend continues until ∼1.0 where the IWSEU begins to drop and destabilise. For IWSEL this point comes a little later around 0.8. On the right of Figure 7.15, all algorithms converge on a very poor clustering performance, once the C_0 is engulfed by C_1, (this happens at ∼0.7). A possible explanation is that the problem of uniformity (the tendency of some clustering algorithms (such as spectral) to produce clusters of equal size) is non-issue when the clusters are well separated. However, as the clusters increasingly overlapped, instance weighting begins helps to alleviate this (through the weighted sub-sampling balancing the numerosity of the clusters somewhat leading to a more representative partitioning). This outcome aligns with patterns seen in Table 7.2.

The second experiment investigates how the difference in density between clusters impacts clustering performance. Again, inline the pattern seen in Table 7.2, IWSEU and IWSEL perform best when there is a difference in the density between the clusters. Figure 7.16 shows that IWSEU and IWSEL perform approximately equal to SER until the difference in density exceeds ∼40% (as per Figure 7.13) this happens when the covariance of C_1 reaches 0.15. After this point, the instance weighting based techniques are able to perform ∼20% better.

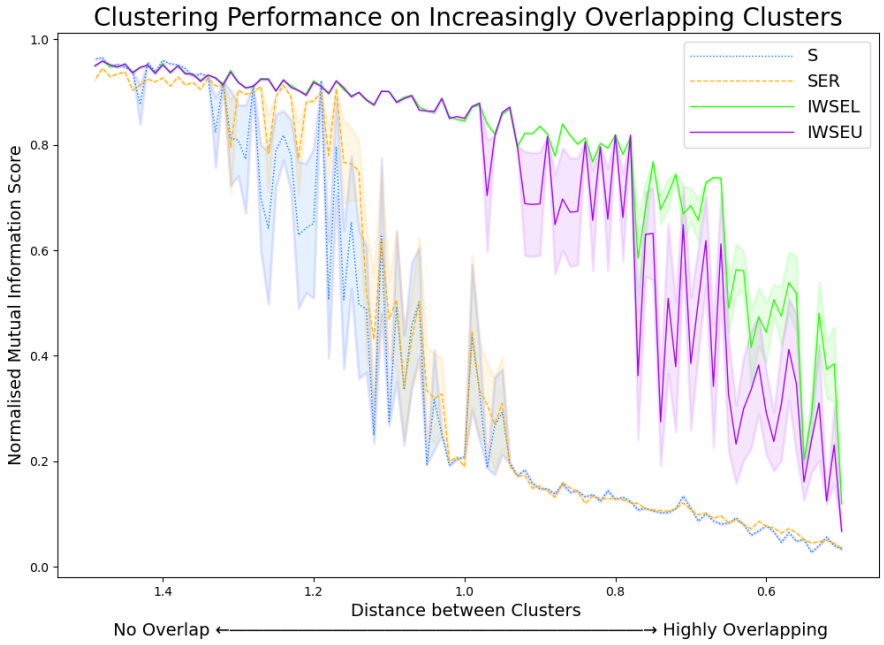


Figure 7.15: IWSEU and IWSEL outperform traditional methods once there is a significant difference in density between the clusters.

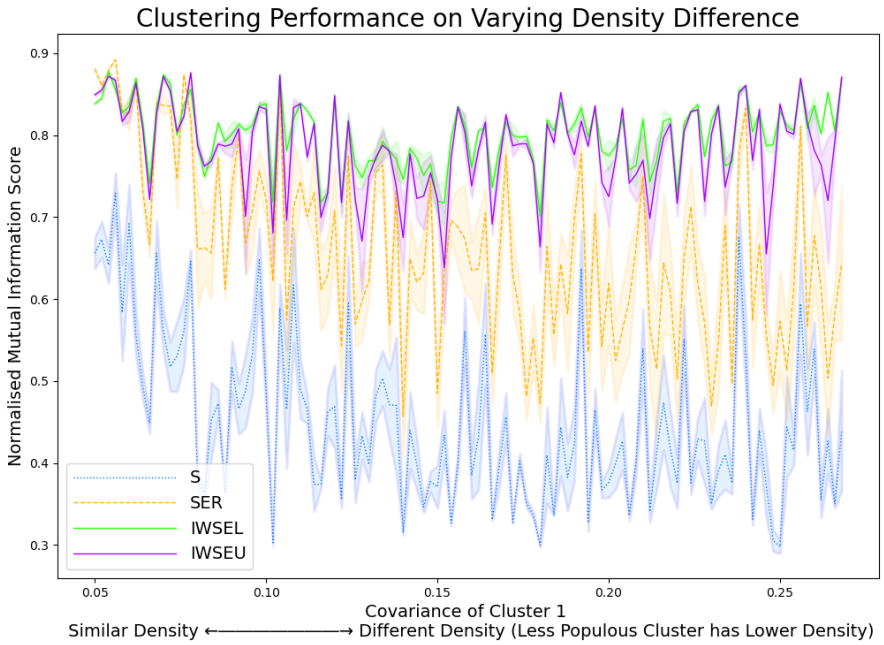


Figure 7.16: IWSEU and IWSEL remains performant despite a significant degree of overlap between imbalanced clusters with different densities.

Finally, the third experiment investigated how a difference imbalance factor effects clustering performance. Initially, when the clusters are balanced, Figure 7.16 shows that initially all algorithms perform well. Between 400% imbalance factor (334:1666) to around 750% imbalance factor (211:1789), the benefit of IWSEU and IWSEL is clear. It seems there is not a simple answer to what degree of imbalance IWSEU and IWSEL will be most beneficial. This experiment indicates that IWSEU and IWSEL is most beneficial at 400%-750% but in the previous MNIST experiments, a variety of ranges where IWSEU helped were observed, see Figures 9.15 to 9.40. A general observation across all experiments is that instance weighting modestly extends clustering performance from the point that traditional methods fail, given the right conditions (in terms of density and overlap).

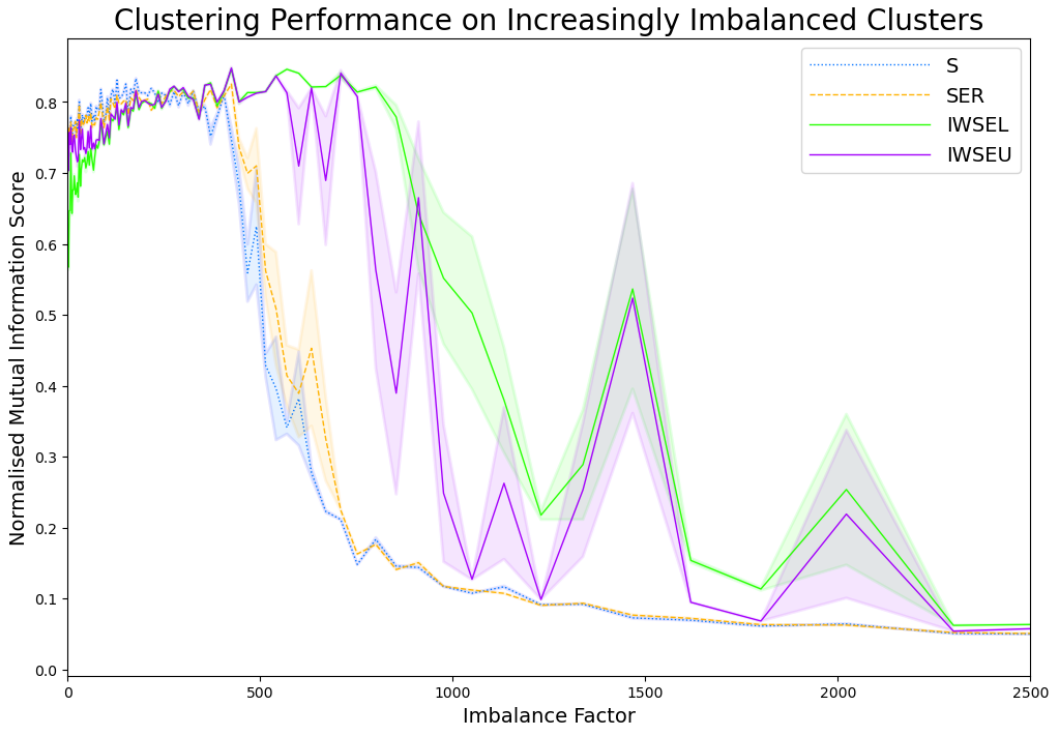


Figure 7.17: IWSEU and IWSEL outperform traditional methods at certain levels of imbalance.

In summary, the synthetic experiments are useful as they confirm some trends suspected and partially observed in the previous MNIST handwritten digits experiments.

Firstly, it was identified that where the clusters are overlapping then IWSEU and IWSEL can in beneficial up to point at which clusters are totally overlapping. Secondarily, a difference in density of ∼40% upwards was necessary in these experiments for instance weighting to benefit clustering performance. Finally, the third experiment, in conjunction with the other experiments, do not conclusively indicate that there is a specific range of imbalance factors on which IWSEU or IWSEL works best, but a general observation is that it can help for a limited period when tradition methods are failing to provide satisfactory clustering performance. The results indicate that when the above conditions present then IWSE may be beneficial.

7.7 Conclusion

This chapter has investigated to what extent and under what conditions the novel framework IWSE can address imbalance by conducting experiments with different degrees of cluster density variation, overlap and imbalance, using the MNIST digits dataset. This chapter provides some insights for answering RQ3 *"Under what conditions does instance weighting enhance clustering performance on data characterized by the presence of outliers or class imbalance?"*. Summarising, the results are mixed and show the framework can be both beneficial and sometimes detrimental to clustering performance. For instance using the approach developed, the following caveats exist. Firstly, the

approach developed has the limitation that it is not suitable for high dimensional datasets. Users will have to utilise preprocessing methods to first reduce the dimensionality to level at which the exponential kernel produces meaningful results. To overcome this issue with the IWSE approach to calculating the density would need to be reconsidered. Secondly, for the IWSE approach, as expected, a variation in density between the clusters is beneficial for the IWSE algorithm to perform well on imbalanced data. Experiments showed that a difference in covariance of 200% (0.05 to 0.15) (in terms of clusters derived from Gaussian distributions) provides adequate information for the instance weighting implement effective sampling to alleviate the "uniform effect". The other aspect related to density is that for the IWSE "U" (and likely "L") variants, the majority cluster should be the denser cluster.

A more surprising result was that Instance Weighting was how beneficial instance weighting was as clusters in the dataset become overlapped. The experiment showed that when the imbalanced clusters are well separated traditional methods are adequate. However, once the clusters are overlapping the imbalance becomes problematic for the traditional methods. The instance weighting approach is able to maintain good clustering performance in this case until up to nearly the point at which the clusters engulf each other. In terms of imbalance and using the parameter setting described, the instance weighting approach was able to address a cluster imbalance of 800%.

In practice, the limitation of kernel density estimation restricted cluster performance by limiting the dimensionality of the datasets input into the approach. The kernel-trick [18] or deep clustering [77, 79, 67] approaches discussed in other work overcome this issue. More research would be required to integrate

these aspects.

Chapter 8

Final Conclusions and Future Work

This thesis set out with the aim of addressing data quality challenges for clustering algorithms using an instance weighting approach. To achieve this aim, two novel clustering approaches, LOFIWKM and IWSE, were developed and analysed. LOFIWKM demonstrated integrating instance weighting into a partitioning-based clustering algorithm. While IWSE demonstrated integrating instance weighting into a graph-based clustering ensemble. The experimentation evaluated the abilities of LOFIWKM and IWSE to handle outliers and imbalanced clusters respectively. In response to the thesis hypothesis, while some aspects of the findings showed negative results, the overall results support the central claim that *"instance weighted clustering is a valuable tool for increasing clustering performance for data with quality issues"*.

Reflecting on my research, there are a number of limitations to my research method and approaches. My research method has focused on using literature and practical experimentation with novel prototype algorithms. However, us-

ing the prototype algorithms limits the generalisability of the findings beyond these prototype algorithms. A more theoretical and mathematical reasoning based approach would lead to findings with better generalisability. Examples of works using this approach include [8], [27] and [17]. A further limitation is the limited pool of datasets used. While in-places depth was achieved, the breadth of experimentation could be expanded to give a broader and clearer picture of the benefits of the proposed approaches. A final limitation which arose was due to the ill-defined nature of clustering. When evaluating the effectiveness of new clustering approaches on datasets without suggested labels and for datasets with data quality challenges. The effectiveness of intrinsic clustering metrics is reduced, since they are not beyond being comprised by the data quality issues themselves. Thus in places the assessment of results involves a degree of subjectivity. These challenges were most notable in Chapter 5.

This thesis identifies the following new knowledge. Firstly, it demonstrates the effectiveness of a density-based weighting scheme integrated into a partitioning-based clustering algorithm for outlier accommodation, in turn addressing RQ1. Secondarily, (again using density-based weighting scheme) how instance weighting be applied to graph-based clustering algorithm to handle imbalanced data. In this case, a clustering ensemble is utilised. A particular merit of this approach is that it does not require adapting the implementation of the clustering algorithm. Meaning that it is practical to implement and adapt into real-world applications. Arguably, the general approach proposed could be taken as a general framework for creating other instance weighting approaches. The proposed approach targetted imbalanced data thus addressing RQ2 and RQ3, but aspects of the framework could be adapted to meet different data quality

issues. Overall, these contributions are significant as they clarify the components and design decisions required to implement instance weighting for two different clustering approaches. Also, a somewhat generalisable framework is presented through the use of the ensemble method.

Finally, 8 directions for future work are identified:

1. This research focused on "how" and "to what extent". But to further the "to what extent" discussion, optimisation and exploration the hyper-parameters could be furthered. It would be interesting to explore if more conclusive hyper-parameter tuning would allow the proposed techniques to handle more extreme outliers / imbalance.

2. Optimisation of the approaches themselves. The proposed approaches mostly are constructed favouring traditional methods and simplicity (to aid understanding of where benefits are emerging from), but integrating more complex and optimal features, into the proposed approach could yield versions of these algorithms that more suitable for a production environment.

3. Both LOFIWKM and IWSE both use density-based weighting schemes, but the literature survey in Chapter 2 showed some other possibilities. Furthermore, it is possible to apply multiple weighting schemes [24]. Multiplexing between different weighting schemes could be a way to address multiple data quality issues, which appears to be a open gap in the research.

4. The exploration of soft clustering combined with instance weighting would be an interesting area to investigate. Nock and Nielsen found that instance

weighting benefited the clustering performance of soft clustering techniques more than hard clustering [8].

5. In this work, semi-supervised clustering was not explored. However, Makkhongkaew et al. showed how instance weighting can be integrated with semi-supervised clustering and presented good results [26]. It would be interesting to research if the instance weights could be used to propagate the information contained by the constraints provided.

6. A key weakness of this work, is the use of methods which do not scale for high-dimensional data. Hence, a future direction of research is towards high-dimensional instance weighting, there is some work in this direction such as Chen et al. who integrate PCA directly into their clustering approach [17]. Also Makkhongkaew et al. who uses feature weighting. Furthermore, Wang and Angelova who use a kernel-based version of FCM [18]. Others explore deep clustering [77, 79, 67].

7. Noise, outliers and imbalance are all similar in nature, however, this work addresses them as separate issues. A future direction could be developing an instance weighting framework to address all these issues.

8. This work explored partitioning-based and graph-based methods, although density-based methods, in particular Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [82] (and Ordering Points To Identify the Clustering Structure (OPTICS)) are effective for clustering data with outliers. One such practical example is Li et al.'s work detecting aircraft landing anomalies [42]. A key benefit of DBSCAN's approach to clustering is its ability to

handle arbitrary and concentric shapes. Different to my work, DBSCAN identifies outliers unlike my outlier accommodation approach. This makes instance weighting for outlier accommodation unnecessary. However, there are a couple weaknesses with DBSCAN that instance weighting could address. For example, DBSCAN struggles to achieve good clustering performance when clusters are tightly packed / highly connected. There is limited research into addressing this weaknesses. In fact, most research proposing variations of DBSCAN focuses on lowering DBSCAN's execution time [83]. However, one work that explored enhancing DBSCAN's robustness proposed a instance weighted DBSCAN variant called Varied DBSCAN (VDBSCAN) [84]. VDBSCAN uses instance weighting to vary ϵ. The instance weights are calculated based on the distance to each instances' K^{th} nearest neighbour. Drawbacks of this method are computational complexity and sensitivity to noise. Also, it is not clear if their method can handle tightly packed clusters. Hence, there is room for future work exploring different weighting schemes to overcome this challenge.

Bibliography

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[2] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[3] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data: Recent advances in clustering*, pp. 25–71, Springer, 2006.

[4] C. C. Aggarwal and C. K. Reddy, *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC, 1st ed., 2014.

[5] A. E. Ezugwu, A. M. Ikotun, O. O. Oyelade, L. Abualigah, J. O. Agushaka, C. I. Eke, and A. A. Akinyelu, "A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects," *Engineering Applications of Artificial Intelligence*, vol. 110, p. 104743, 2022.

[6] H. Steinhaus *et al.*, "Sur la division des corps matériels en parties," *Bull. Acad. Polon. Sci*, vol. 1, no. 804, p. 801, 1956.

[7] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*, 1967.

[8] R. Nock and F. Nielsen, "On weighting clustering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 8, pp. 1223–1235, 2006.

[9] L. Gu, "A novel sample weighting k-means clustering algorithm based on angles information," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3697–3702, IEEE, 2016.

[10] J. Yu, M.-S. Yang, and E. S. Lee, "Sample-weighted clustering methods," *Computers & Mathematics with Applications*, vol. 62, no. 5, pp. 2200–2208, 2011.

[11] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," *Data mining and knowledge discovery*, vol. 30, no. 4, pp. 891–927, 2016.

[12] M. O. S. N. Wolberg, William and W. Street, "sBreast Cancer Wisconsin (Diagnostic)." UCI Machine Learning Repository, 1993. DOI: https://doi.org/10.24432/C5DW2B.

[13] R. P. Moro, S. and P. Cortez, "Bank Marketing." UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5K306.

[14] J. I. Orlando, H. Fu, J. B. Breda, K. Van Keer, D. R. Bathula, A. Diaz-Pinto, R. Fang, P.-A. Heng, J. Kim, J. Lee, *et al.*, "Refuge challenge: A unified framework for evaluating automated methods for glaucoma

assessment from fundus photographs," *Medical image analysis*, vol. 59, p. 101570, 2020.

[15] A. Topchy, A. K. Jain, and W. Punch, "A mixture model for clustering ensembles," in *Proceedings of the 2004 SIAM international conference on data mining*, pp. 379–390, SIAM, 2004.

[16] G. Hamerly and C. Elkan, "Alternatives to the k-means algorithm that find better clusterings," in *Proceedings of the eleventh international conference on Information and knowledge management*, pp. 600–607, 2002.

[17] J. Chen, J. Zhu, H. Jiang, H. Yang, and F. Nie, "Sparsity fuzzy c-means clustering with principal component analysis embedding," *IEEE Transactions on Fuzzy Systems*, vol. 31, no. 7, pp. 2099–2111, 2023.

[18] Y. Wang and M. Angelova, "Weighted kernel fuzzy c-means method for gene expression analysis," in *2012 Spring Congress on Engineering and Technology*, pp. 1–4, 2012.

[19] M. Gong, Y. Liang, J. Shi, W. Ma, and J. Ma, "Fuzzy c-means clustering with local information and kernel metric for image segmentation," *IEEE Transactions on Image Processing*, vol. 22, no. 2, pp. 573–584, 2013.

[20] W. Guo, Z. Huang, Y. Hou, Q. Xiao, J. Jia, and L. Mao, "Environment parameter rating evaluation for smart museum based on improved k-means clustering algorithm," in *2020 Chinese Control And Decision Conference (CCDC)*, pp. 5449–5453, 2020.

[21] Z. Wang, G. Q. Liu, and J. C. Guo, "An improved k-means algorithm based on multiple feature points," in *2009 International Workshop on Intelligent Systems and Applications*, pp. 1–5, 2009.

[22] J. Chen, Z. Li, and B. Huang, "Linear spectral clustering superpixel," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3317–3330, 2017.

[23] Z. Ji, Y. Xia, Q. Chen, Q. Sun, D. Xia, and D. D. Feng, "Fuzzy c-means clustering with weighted image patch for image segmentation," *Applied soft computing*, vol. 12, no. 6, pp. 1659–1667, 2012.

[24] S.-l. Zhai, B. Luo, and Y.-t. Guo, "Fuzzy clustering ensemble based on dual boosting," in *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, vol. 2, pp. 240–244, 2007.

[25] J. Guan, S. Li, X. Chen, X. He, and J. Chen, "Demos: Clustering by pruning a density-boosting cluster tree of density mounts," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 10, pp. 10814–10830, 2023.

[26] R. Makkhongkaew, K. Benabdeslem, and H. Elghazel, "Semi-supervised co-selection: Features and instances by a weighting approach," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3477–3484, 2016.

[27] H. Liu, J. Wu, T. Liu, D. Tao, and Y. Fu, "Spectral ensemble clustering via weighted k-means: Theoretical and practical evidence," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 5, pp. 1129–1143, 2017.

[28] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[29] R. Cordeiro de Amorim and B. Mirkin, "Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering," *Pattern Recognition*, vol. 45, no. 3, pp. 1061–1075, 2012.

[30] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "Np-hardness of euclidean sum-of-squares clustering," *Machine learning*, vol. 75, pp. 245–248, 2009.

[31] S. Harris and R. C. De Amorim, "An extensive empirical comparison of k-means initialization algorithms," *IEEE Access*, vol. 10, pp. 58752–58768, 2022.

[32] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[33] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, "A survey of kernel and spectral methods for clustering," *Pattern recognition*, vol. 41, no. 1, pp. 176–190, 2008.

[34] A. Strehl and J. Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *Journal of machine learning research*, vol. 3, no. Dec, pp. 583–617, 2002.

[35] M. Rezaei and P. Fränti, "Set matching measures for external cluster validity," *IEEE transactions on knowledge and data engineering*, vol. 28, no. 8, pp. 2173–2186, 2016.

[36] D. M. Hawkins, *Identification of outliers*, vol. 11. Springer, 1980.

[37] B. Zhang, "Generalized k-harmonic means," *Hewlett-Packard Laboratoris Technical Report*, 2000.

[38] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM sigmod record*, vol. 29, pp. 93–104, ACM, 2000.

[39] M. Alshawabkeh, B. Jang, and D. Kaeli, "Accelerating the local outlier factor algorithm on a gpu for intrusion detection systems," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, pp. 104–110, 2010.

[40] M. Charytanowicz, J. Niewczas, P. Kulczycki, P. A. Kowalski, S. Łukasik, and S. Żak, "Complete gradient clustering algorithm for features analysis of x-ray images," in *Information technologies in biomedicine*, pp. 15–24, Springer, 2010.

[41] D. Dua and C. Graff, "Uci machine learning repository," 2017.

[42] L. Li, S. Das, R. John Hansman, R. Palacios, and A. N. Srivastava, "Analysis of flight data using clustering techniques for detecting abnormal operations," *Journal of Aerospace information systems*, vol. 12, no. 9, pp. 587–598, 2015.

[43] Y. Liu, R. Sun, and P. He, "Research on the pre-warning method of aircraft long landing based on the xgboost algorithm and operation characteristics clustering," *Aerospace*, vol. 10, no. 5, p. 409, 2023.

[44] G. Wang, H. Xu, B. Pei, and H. Cheng, "Flight risk evaluation based on flight state deep clustering network," *Complex & Intelligent Systems*, vol. 9, no. 5, pp. 5893–5906, 2023.

[45] B. Matthews, "NASA DASHlink curated 4 class anomaly detection data set." `https://c3.ndc.nasa.gov/dashlink/resources/1018/`, 2022. Accessed: 2023-02-18.

[46] D. Moulavi, P. A. Jaskowiak, R. J. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proceedings of the 2014 SIAM international conference on data mining*, pp. 839–847, SIAM, 2014.

[47] K. Efimov, L. Adamyan, and V. Spokoiny, "Adaptive nonparametric clustering," *IEEE Transactions on Information Theory*, vol. 65, no. 8, pp. 4875–4892, 2019.

[48] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, pp. 849–856, 2002.

[49] M. Zhang, "Weighted clustering ensemble: A review," *Pattern Recognition*, p. 108428, 2021.

[50] H. Parvin, B. Minaei-Bidgoli, H. Alinejad-Rokny, and W. F. Punch, "Data weighing mechanisms for clustering ensembles," *Computers & Electrical Engineering*, vol. 39, no. 5, pp. 1433–1450, 2013.

[51] M. Al-Razgan and C. Domeniconi, "Weighted clustering ensembles," in *Proceedings of the 2006 SIAM International Conference on Data Mining*, pp. 258–269, SIAM, 2006.

[52] Z. Wang, S. Zhao, Z. Li, H. Chen, C. Li, and Y. Shen, "Ensemble selection with joint spectral clustering and structural sparsity," *Pattern Recognition*, p. 108061, 2021.

[53] X. Z. Fern and W. Lin, "Cluster ensemble selection," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 1, no. 3, pp. 128–141, 2008.

[54] D. Huang, C.-D. Wang, J.-S. Wu, J.-H. Lai, and C.-K. Kwoh, "Ultra-scalable spectral clustering and ensemble clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 6, pp. 1212–1226, 2019.

[55] D. Frossyniotis, A. Likas, and A. Stafylopatis, "A clustering method based on boosting," *Pattern Recognition Letters*, vol. 25, no. 6, pp. 641–654, 2004.

[56] Y. Ren, C. Domeniconi, G. Zhang, and G. Yu, "Weighted-object ensemble clustering," in *2013 IEEE 13th International Conference on Data Mining*, pp. 627–636, IEEE, 2013.

[57] T. Boongoen and N. Iam-On, "Cluster ensembles: A survey of approaches with recent extensions and applications," *Computer Science Review*, vol. 28, pp. 1–25, 2018.

[58] J. Qian and V. Saligrama, "Spectral clustering with imbalanced data," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3057–3061, IEEE, 2014.

[59] B. Nadler and M. Galun, "Fundamental limitations of spectral clustering," in *Advances in neural information processing systems*, pp. 1017–1024, Citeseer, 2007.

[60] M. Lucińska and S. T. Wierzchoń, "Spectral clustering based on k-nearest neighbor graph," in *IFIP International Conference on Computer Information Systems and Industrial Management*, pp. 254–265, Springer, 2012.

[61] C. D. Correa and P. Lindstrom, "Locally-scaled spectral clustering using empty region graphs," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1330–1338, 2012.

[62] H. Ayad and M. Kamel, "Refined shared nearest neighbors graph for combining multiple data clusterings," in *International Symposium on Intelligent Data Analysis*, pp. 307–318, Springer, 2003.

[63] J. M. Duarte, A. L. Fred, and F. J. F. Duarte, "Adaptive evidence accumulation clustering using the confidence of the objects' assignments," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 70–87, Springer, 2012.

[64] A. L. Fred and A. K. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 6, pp. 835–850, 2005.

[65] J. Jia, X. Xiao, B. Liu, and L. Jiao, "Bagging-based spectral clustering ensemble selection," *Pattern Recognition Letters*, vol. 32, no. 10, pp. 1456–1467, 2011.

[66] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[67] Y. He, "Imbalanced data clustering using equilibrium k-means," *arXiv preprint arXiv:2402.14490v2*, 2024.

[68] J. Handl and J. Knowles, "An evolutionary approach to multiobjective clustering," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 56–76, 2007.

[69] C. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Transactions on Computers*, vol. C-20, no. 1, pp. 68–86, 1971.

[70] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," *Advances in neural information processing systems*, vol. 17, 2004.

[71] S. Sinha, A. K. Bhandari, and R. Kumar, "Low quality retinal blood vessel image boosting using fuzzified clustering," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 6, pp. 3022–3033, 2024.

[72] S. Chowdhury, N. Helian, and R. C. de Amorim, "Feature weighting in dbscan using reverse nearest neighbours," *Pattern Recognition*, vol. 137, p. 109314, 2023.

[73] D. L. Torres, L. C. La Rosa, D. A. B. Oliveira, and R. Q. Feitosa, "Evaluation of unsupervised deep clustering methods for crop classification using sar image sequences," in *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pp. 4240–4243, 2021.

[74] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[75] Y. Pei and L. Ye, "Cluster analysis of mnist data set," in *Journal of Physics: Conference Series*, vol. 2181, p. 012035, IOP Publishing, 2022.

[76] S. Pourmohammad, R. Soosahabi, and A. S. Maida, "An efficient character recognition scheme based on k-means clustering," in *2013 5th international conference on modeling, simulation and applied optimization (ICMSAO)*, pp. 1–6, IEEE, 2013.

[77] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, "Deep spectral clustering using dual autoencoder network," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4066–4075, 2019.

[78] N. D. Singh and A. Dhall, "Clustering and learning from imbalanced data," *arXiv preprint arXiv:1811.00972*, 2018.

[79] M. Rezaei, E. Dorigatti, D. Rügamer, and B. Bischl, "Joint debiased representation learning and imbalanced data clustering," in *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 55–62, IEEE, 2022.

[80] Y. Chong, L. Chi, D. P. Robinson, R. Vidal, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, "A scalable exemplar-based subspace clustering algorithm for class-imbalanced data," in *Proc. European Conference on Computer Vision (ECCV)*, pp. 68–85, 2018.

[81] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[82] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *kdd*, vol. 96, pp. 226–231, 1996.

[83] T. Ali, S. Asghar, and N. A. Sajid, "Critical analysis of dbscan variations," in *2010 international conference on information and emerging technologies*, pp. 1–6, IEEE, 2010.

[84] P. Liu, D. Zhou, and N. Wu, "Vdbscan: varied density based spatial clustering of applications with noise," in *2007 International conference on service systems and service management*, pp. 1–4, IEEE, 2007.

Chapter 9

Appendix

9.1 Literature Review Thematic Analysis Tables

Table 9.1: Papers identified by the literature search, with filtering.

Document Title	Publication Year	Include Title & Abstract (Screening)	Include Full Text (Eligibility)
Spectral Ensemble Clustering via Weighted K-Means: Theoretical and Practical Evidence	2017	PASS	PASS
Weighted Multiview Possibilistic C-Means Clustering With L2 Regularization	2022	FAIL I2	
CUSBoost: Cluster-Based Under-Sampling with Boosting for Imbalanced Classification	2017	PASS	FAIL I2
Fuzzy K-Means with Variable Weighting in High Dimensional Data Analysis	2008	PASS	FAIL I2
Research on Text Categorization of KNN Based on K-Means for Class Imbalanced Problem	2016	PASS	FAIL I2
Adaptive Ensemble Clustering With Boosting BLS-Based Autoencoder	2023	FAIL I2	
Incomplete Multi-View Clustering Based on Dynamic Dimensionality Reduction Weighted Graph Learning	2024	FAIL I2	
A Dissimilarity Measure Powered Feature Weighted Fuzzy C-Means Algorithm for Gene Expression Data	2024	FAIL I2	
Multi-View Feature Boosting Network for Deep Subspace Clustering	2022	FAIL I2	
Context-Aware Hypergraph Construction for Robust Spectral Clustering	2014	PASS	FAIL I2
Feature Weighted Multi-View Graph Clustering	2024	FAIL I2	
Robust guidewire segmentation through boosting, clustering and linear programming	2010	PASS	FAIL I2

Image Segmentation Using Fuzzy C-Means Algorithm Incorporating Weighted Local Complement Membership and Local Data Distances	2016	PASS	FAIL I2
Customer Churn Prediction in the Telecom Sector with Machine Learning and Adaptive k-Means Cluster using Imbalance Data	2024	PASS	FAIL I2
Sparsity Fuzzy C-Means Clustering With Principal Component Analysis Embedding	2023	PASS	PASS
Fuzzy C-means algorithm incorporating local data and membership information for noisy medical image segmentation	2015	PASS	FAIL I2
Fuzzy K-Means Clustering With Discriminative Embedding	2022	PASS	FAIL I2
K-means clustering based on self-adaptive weight	2012	PASS	FAIL I2
Fuzzy Clustering Ensemble Based on Dual Boosting	2007	PASS	PASS
An intelligent Weighted Kernel K-Means algorithm for high dimension data	2009	PASS	FAIL E3
Boosting K-Nearest Neighbour (KNN) Classification using Clustering and AdaBoost Methods	2022	FAIL I1	
On implementing a spectral clustering controlled islanding algorithm in real power systems	2013	PASS	FAIL I2
Smartphone TBI Sensing using Deep Embedded Clustering and Extreme Boosted Outlier Detection	2021	PASS	FAIL I2
Research on Ship Track Clustering Method Based on Optimized Spectral Clustering Algorithm	2021	PASS	FAIL I2
Using Boosting and Clustering to Prune Bagging and Detect Noisy Data	2009	FAIL I1	
Adaptive Graph Representation for Clustering	2022	PASS	FAIL I2
Distributed Weighted Fuzzy C-Means Clustering for Wireless Sensor Network Data Analysis	2023	FAIL I2	
Cluster-Based Boosting	2015	FAIL I1	
DEMOS: Clustering by Pruning a Density-Boosting Cluster Tree of Density Mounts	2023	PASS	PASS
A new brain MRI image segmentation strategy based on wavelet transform and K-means clustering	2015	PASS	FAIL I2
A k-means-based and no-super-parametric Improvement of AdaBoost and its Application to Transaction Fraud Detection	2020	FAIL I1	
Public Sector Corruption Analysis with Modified K-means Algorithm Using Perception Data	2020	PASS	FAIL I2
An Automatic Kernel of Graph Clustering Method in Conforming Clustering Number	2007	PASS	FAIL I2
Weighted Kernel Fuzzy C-Means Method for Gene Expression Analysis	2012	PASS	PASS
Cluster Density Properties Define a Graph for Effective Pattern Feature Selection	2020	FAIL I2	
Multi-Channel Augmented Graph Embedding Convolutional Network for Multi-View Clustering	2023	PASS	FAIL I2
MFWK-Means: Minkowski metric Fuzzy Weighted K-Means for high dimensional data clustering	2013	PASS	FAIL I2
Dempster-Shafer theory of evidence in Single Pass Fuzzy C Means	2013	PASS	FAIL I2
Weighted Spectral Cluster Ensemble	2015	PASS	FAIL I2
A Privacy-Preserving Smart Body Scale with K-Means Anonymization towards GDPR-Compliant IoT	2023	FAIL I2	
Fuzzy C-Means Clustering With Local Information and Kernel Metric for Image Segmentation	2013	PASS	PASS

Weight based movie recommendation system using K-means algorithm	2017	PASS	FAIL I3
Boosting Home WiFi Throughputs via Adaptive DAS Clustering of PLC Extenders	2021	FAIL I1	
Boosting the Computational Performance of Feature-Based Multiple 3D Scan Alignment by iat-k-means Clustering	2012	FAIL I2	
Boosting performance of I/O-intensive workload by preemptive job migrations in a cluster system	2003	FAIL I1	
Bias field estimation and segmentation of MR image using modified fuzzy-C means algorithms	2015	PASS	FAIL I2
LUCID: Author name disambiguation using graph Structural Clustering	2017	PASS	FAIL I2
An Image Segmentation Algorithm Based on Fuzzy C-Means Clustering	2009	PASS	FAIL I2
Sampled Data Debugging via Fuzzy C-Means	2021	FAIL I2	
A novel controlled islanding algorithm based on constrained spectral clustering	2011	FAIL I2	
Boosting Meaningful Dependency Mining with Clustering and Covariance Analysis	2024	FAIL I1	
Initializing FWSA K-Means With Feature Level Constraints	2022	FAIL I2	
Low Quality Retinal Blood Vessel Image Boosting Using Fuzzified Clustering	2024	PASS	FAIL I2
R - Tree Index Construction of Dynamic K-means Algorithm	2018	PASS	FAIL I2
Environment Parameter Rating Evaluation for Smart Museum Based on Improved K-Means Clustering Algorithm	2020	PASS	PASS
Graphic: Graph-Based Hierarchical Clustering For Single-Molecule Localization Microscopy	2021	PASS	FAIL I2
An Improved K-Means Algorithm Based on Multiple Feature Points	2009	PASS	PASS
Convex Hierarchical Clustering for Graph-Structured Data	2019	PASS	FAIL I2
FCAN-MOPSO: An Improved Fuzzy-Based Graph Clustering Algorithm for Complex Networks With Multiobjective Particle Swarm Optimization	2023	PASS	FAIL I2
Hidden Markov Model with Parameter-Optimized K-Means Clustering for Handwriting Recognition	2011	FAIL I2	
Sample-weighted clustering methods	2011	PASS	PASS
On Weighting Clustering	2006	PASS	PASS
Linear Spectral Clustering Superpixel	2017	PASS	PASS
Fuzzy c-means clustering with weighted image patch for image segmentation	2012	PASS	PASS
A Novel Sample Weighting K-Means Clustering Algorithm based on Angles Information	2016	PASS	PASS
Semi-supervised co-selection: features and instances by a weighting approach	2016	PASS	PASS
Adaptive clustering ensembles	2004	PASS	PASS
Alternatives to the k-means algorithm that find better clusterings	2002	PASS	PASS

Table 9.2: The codes linked to data extraction sources.

Code	Data Extraction Source
Boosting	[24, 25, 15]
Bagging	[27]
Weighting used in Assignment Membership Degree	[19, 20, 21, 22, 23, 26]
Weighting used in Centroid Update	[27, 17, 18, 19, 21, 8, 10, 22, 23, 9] [26]
Weights used for Sampling	[24, 15]
Merge Probability	[25]
Outlier Accommodation	[17, 20, 10, 8]
Expediting Runtime/Convergence	[27, 22, 15]
Noise Accommodation	[27, 18, 19, 20, 10, 8]
Imbalanced Clusters	[21]
Distant Instances have Higher Weight	[8, 16]
Complex Weighting Schemes	[24, 25]
Boundary points have Higher Weight	[9, 15]
Distant Instances have Low Weight	[17, 18, 19, 20, 21, 10, 22, 23, 26]
Feature Weighting	[26]
Generalised Framework	[10, 8]
Feature Reduction	[17, 16]
Kernel Method	[18]

Table 9.3: The themes linked to codes.

Theme	Codes
Actuation of the Weights	Weights used in Assignment/Membership Degree
	Weights used in Centroid Update
	Weights used for Sampling
	Weights used for Merge Probability
Ensemble Techniques	Boosting
	Bagging
Benefits of Instance Weighting	Expediting Runtime/Convergence
	Outlier Accommodation
	Noise Accommodation
	Imbalanced Clusters
Weighting Strategy	Distant Instances have Higher Weight
	Complex Weighting Schemes
	Boundary points have Higher Weight
	Distant Instances have Low Weight
Compatibility	Generalised Framework
	Feature Weighting
	Feature Reduction
	Kernel Method

Table 9.4: Data extraction table.

Title and citation	What clustering algorithms were investigated/developed?	How were the weights defined?	How were the weights applied?	For what datasets/applications?	Data quality issues addressed	Clustering Quality Metrics used	Notes – interesting features
Spectral Ensemble Clustering via Weighted K-Means: Theoretical and Practical Evidence [27]	Spectral Ensemble using an Instance Weighted K-Means	Delta (difference) between assignment of instances across the ensemble of clusterings is used to inform the weighting.	When calculating the centroids.	Many Benchmark datasets: breast w, iris, wine, cacmcisi, classic, cranmed, hitech, k1b, la12, mm, re1, reviews, sports, tr11, tr12, tr41, tr45, letter. Real-world data sets from UCI: MNIST, Handwritten Digits, three-Sources, Multilingual, Four-Areas. Sina Weibo (Tweets-like)	Noise, while no specific type of noise mentioned, the selection of datasets includes some noisy partitions. Also, the challenge of big data is partially addressed.	(Intrinsic) Custom measures. (Extrinsic) Normalized Rand Index. In places, the instances weights improved clustering performance by ∼20%. Additionally, runtime and execution complexity is reduced significantly.	Applies an instance weighted k-means at the consensus stage of a clustering ensemble. Demonstrates distributed processing. In two benchmark datasets (breast_w and cacmcisi). Instance weighting made the difference between a total failure to cluster the datapoints and good clustering (for example: 82% vs 7% with UCI breast_w). Instance weights enabled the simplification of the spectral ensemble clustering to a weighted K-means. This reduced runtime.

Sparsity Fuzzy C-Means Clustering With Principal Component Analysis Embedding [17]	Fuzzy C-Means	Sum of all distances to cluster centres from a instance point. These are sorted, for the identification of outliers. A threshold is used to covert the rank, to a weight of either 0 or 1.	The weights of the outliers identified is set to 0. The resulting effect is the outliers have 0 distance to all clusters and thus, do not impact the centroid update step.	15 benchmark datasets: COIL20, USPS, ORL, MNIST, PALM, MSRA, YALE, UMIST, LEUML, dermatology, prostateCancerPSA410ML, lymphoma, Solar, MALDIML, and MLLML. These datasets are from UCL	Synthetic outliers, uniformly random but gapped from the dataset	Accuracy and NMI was used to evaluate the results. Gains of 3% to 28% (average 13%) in accuracy when comparing their instance weighted approach to k-means.	This approach simultaneously applies PCA and clustering.
Weighted Kernel Fuzzy C-Means Method for Gene Expression Analysis [18]	Fuzzy C-Means	Kernel version of the FCM algorithm. A Gaussian function is fitted to describe the local distribution around a cluster, such that noise can be mitigated. Points closer to the centre of the distribution, receive a higher weight. The impact of the weighting to can be adjusted using a parameter. Where 0 is strongly applied and 1 and degrades the clustering to non-weighted.	The weights impact the calculation of the new cluster centroids positions.	Gene Expression Analysis Dataset: Rat CNS and Yeast Cell cycle These are noisy Microarray datasets.	Noise - application specific (the natural noise in gene expression data). They do not describe the nature of the noise. Non-linear partitions and overlapping data.	ARI was used to evaluate the effectiveness of approach. Gains of as much as 20% ARI when using their instance weighted method compared to plain FCM.	By mapping the data to a higher dimensional space, their kernel approach enables FCM to handle non-spherical and overlapped clusters with noise.

Fuzzy C-Means Clustering With Local Information and Kernel Metric for Image Segmentation [19]	Fuzzy C-Means	Based on the intensity of neighbouring pixels. Their trade-off weighting approach has two parameters, a spatial constraint and grey level constraint. Higher weight is given to neighbouring pixels which are close and similar in intensity. Ultimately, noise pixels receive less weight.	The instance weights are used in the assignment step and in the centroid update step.	Image segmentation Simple diagnostic images. Brain MR images. Natural images.	Gaussian noise, Salt and Pepper noise, and Rician noise in greyscale images.	Accuracy Increases in accuracy of ∼5% were observed across the variety of images.	Bandwidth selection for their Gaussian Radial Basis Function kernel is difficult so they use an estimation method. Maps data to a higher dimensional space using the kernel method.
Environment Parameter Rating Evaluation for Smart Museum Based on Improved K-Means Clustering Algorithm [20]	K-Means	Weights are calculated based on distance to centroid. Weight per cluster are calculated to sum up to 1. Instances are ranked in terms of distance away from the centre.	The instance weights are used in the cluster assignment phase.	A mix of real-world and benchmark data is used. Real-world: Museum Environment Recordings UCI Benchmark datasets: Glass and Iris	The work claim to address noise and outliers, is fair to assume their environmental readings contained some natural noise.	Accuracy was calculated using a train-test split. Gains of ∼2% on the Iris dataset and gains of ∼4% on the Glass dataset were observed.	Their algorithm uses a histogram to calculate density information to assist initialisation.

An Improved K-Means Algorithm Based on Multiple Feature Points [21]	K-means	Based on distance to "Feature Points" (multiple points represent a cluster in their implementation)	Instance weights are used in both the cluster assignment phase and in the centroid update phase. Calculates weighted distances to distribute the data points to clusters and to build new feature point sets.	Circle and ring. (Uniformly random instances, concentric clusters) Two rectangles. (Uniformly random instances, imbalanced clusters) UCI Segment Challenge (20 features, 1500 instances)	Imbalanced Clusters. Concentric clusters.	Somewhat limited experiments. Experiments only ran once. Metric is simply "Correct" or "Incorrect".	Their algorithm (MFPKM) selects multiple feature points as initial cluster centres rather than just one. Requires tuning hyperparameters (feature point number and sparse factor) Their approach improved performance on imbalanced clusters.

Fuzzy Clustering Ensemble Based on Dual Boosting [24]	Fuzzy C-means	Iteratively creates new training sets that include both "hard" instances (those that are difficult to cluster) and "fuzzy" instances (those with ambiguous cluster assignments). This is done by updating the probability of selecting each instance based on its clustering performance in previous iterations, based on the membership degree. Up weights "hard" and "fuzzy" instances. In summary, instances that have similarity affinity with all clusters OR are very well clustered are up-weighted.	The instance weights are used to guide sampling probability in their boosting based approach.	The experiments were conducted on an artificial dataset and UCI benchmark datasets: Iris, Wine	Stability/robustness of results was their goal.	Error Rate used to assess results. Their results showed their method decreased error rates by: 9% for Iris, 2% for Wine and 0.7% for x8d5k, compared to FCM. Experiments were repeated 50 times.	The final clustering solution is obtained using a co-association matrix of the assignments from the ensemble.

DEMOS: Clustering by Pruning a Density-Boosting Cluster Tree of Density Mounts [25]	Based on K-means and Density Peak Clustering (Density peak clustering is based on Mean-shift)	High density peaks are given more weight. Also, more weight is given in valleys with strong connectivity, based on number of linking instances. In this dual boosting approach, essentially, high density areas and connecting areas are considered important. Based on local density data.	Weights effect the likelihood of density peaks merging into clusters.	Synthetic datasets: Agg, Flame, Jain, Compound, R15, Spiral, Path-based, D31, S3, and T48k, with instance counts ranging from 240 to 8000. Cluster count ranging from 2 to 31. All are 2d dimensional datasets. Real datasets: Iris, Wine, Segment, Drivedata, Breastcancer, YTF, USPS, MNIST.	Complex shapes Automatically finding the correct number of clusters. Big data.	Very impressive results. Mostly 100% accuracy on the variety of synthetic datasets trialled. Adjusted Mutual Information, Adjusted Random Index, F1 and DGCI scores used to assess performance. Results included 100% ARI on the Jain datasets and 85% ARI on the compound dataset.	A strength of this work is that they are clear on their definition of a cluster. "A cluster is assumed to be a density-connected area with multiple (or a single) density mounts and a relatively large dis-connectivity from density-connected areas of higher densities." Produces a clustering tree (dendrogram). The classification of points is similar to DBSCAN.

Sample-weighted clustering methods [10]	K-Means, Fuzzy C-Means, Expectation Maximisation	Based on the Maximum Entropy Principle (Information theory) a probability distribution is calculated over the dataset to inform the instance weights. Using the maximum entropy principle, noise points receive a smaller weight. Essentially, points that appear to be uniform noise will have a lower entropy and receive less weight. This is based on a distortion factor, which is the squared distance from the cluster centre that an instance is currently a member of. Their approach assigns lower weight to outliers. Using exponential function. Note that Zeta (seen in their Equation 3) is essentially the bandwidth. Instance weights are greater than 0 and together add up to 1.	Weights are used to inform cluster centre positions only.	Synthetic data and the Iris dataset are trialled with a single strongly outlying instance added.	Explicitly mentions robustness to outliers and noise. Evaluates performance against an added artificial outlier.	Average and minimum error rate are used to assess performance. They compare their instance weighted algorithm with different clustering algorithms and with and without instance weighting. The results show a large performance gain, especially with the single outlier datasets.	Their work offers a generalised approach to instance weighting multiple clustering algorithms: "The new clustering framework can be applied to most clustering algorithms." It is implied, centroid-based clustering algorithms only. This work highlights that instance (sample) weighting is less researched than feature weighting.

On Weighting Clustering [8]	K-Means, Fuzzy C-means, EM and K-Harmonic Means	Iteratively weighted approach. Their approach focuses on the harder to cluster points using a probabilistic framework. Points far from cluster centres have higher weights. Boosting inspired.	Instance weights are only used to update centroid positions.	Various synthetic datasets, including normal, uniform and concentric distributions.	Mix of datasets included somewhat noisy and sometimes overlapping Gaussian distributions.	Their results use three metrics including extrinsic and intrinsic metrics, most notably KNM-loss and missed clusters.	The weighted algorithms (apart from fuzzy k-means) performed better as the dimensionality increases. The instance weighted algorithms worked better with the fuzzy (soft membership) clustering algorithms particularly fuzzy k-means. Their describe their method as a framework and show how it can be applied to a selection partitioning-based algorithms. Uses Bregman divergences to calculate weights, which compared to traditional distance measures, offer several advantages, such as robustness to outliers and the ability to handle non-Euclidean data.

Linear Spectral Clustering Superpixel [22]	Spectral Clustering and k-means	Uses both colour and space information in distance calculation. Distance metric is applied as a kernel function in a high dimensional feature space. Simplifies N-cuts component in spectral clustering by replacing with a weighted k-means. The weight of a pixel (instance) is based on it's similarity with all other pixels. Pixels which are similar to all other pixels are given higher weight.	The weighted k-means uses the instance weights in both the assignment and centroid update phases.	Brekley Image segmentation dataset.	Noise in natural images. Interestingly, here the use of weights enables the integrated k-means to perform equivalently to N-cuts, this makes the spectral clustering algorithm more efficient.	State of the art clustering performance – shown by comparison with other improved algorithms. Although still somewhat slower than some algorithms. Accuracy gains of $\sim$3-4%	This is a super-pixel segmentation algorithm. Their algorithm requires the manual specification of k - this the number of superpixels.

Fuzzy C-Means clustering with weighted image patch for image segmentation [23]	Fuzzy C-Means	Uses local spatial information to inform the instance weights. First, the mean and standard deviation of patch around an (instance) pixel is calculated. Next, the exponential kernel is applied to image patch. These weights are then normalised. Pixels that are very different to their neighbours have a low, often 0 weights.	The instance weights are used in both the cluster assignment and centroid update stages. Instances that have a low weight (thus do not belong in their patch) effect the distance calculation and centroid calculation less. Note that, pixels are replaced with a patch of pixels. These will respond differently to different centroids. Their approach is sophisticated.	The MRI images were from BrainWeb.	Gaussian noise, salt-and-pepper noise and a combination of both are applied to synthetic images and Brain MRI images.	Three metrics are used to assess accuracy. To assess the image segmentation performance extrinsically, the percent of the of correctly classified pixels was used. To assess the performance of the clustering intrinsically, the fuzzy partition coefficient and partition entropy were used.	Their algorithm relies on the user specifying an image patch size, which depends on the amount of noise in the image. The authors highlight automating this as future work.

A Novel Sample Weighting K-Means Clustering Algorithm based on Angles Information [9]	K-Means	Instances are considered either "ambiguous" or "unambiguous" based on their angle relative to a vector between the cluster centre and cluster boundary. Points which are angled such that they are within a specific area between the boundary and centroid are classed as "ambiguous", while points outside this area are classified as "unambiguous". The "ambiguous" points are weighted using a Min-Max Normalised distance from the centre. The "unambiguous" points are weighted using the cosine of the of angle between the vectors: nearest cluster centre to the given unambiguous instance AND the vector of the nearest cluster centre to the other clusters in turn. Points that are angled away from other clusters are given a higher weight.	Cluster centres are calculated using the weight.	They use real and synthetic datasets. Two artificial and nine real datasets: Bensaid1, Bensaid2, Tae, Sonar, Seeds, Svmguide4, Column2C, Column3C, Liver, Diabetes, Vehicle.	The data quality issues are not explicitly mentioned, but small imbalanced datasets are used.	Accuracy increases of between 1 to 20% (average 5%) are observed, compared to k-means.	This work is based on the principle that points that are closer to a boundary may be homogenous with a different cluster.

Semi-supervised co-selection: features and instances by a weighting approach [26]	Custom	Initially the instance weights are randomly generated. They are then iteratively optimised according to an objective function: For each instance, the distance from each cluster according to each weighted feature. Their approach assigns higher weight instances closer to centroids.	Instance weights inform the feature weights. The instance weights inform the assignment and the centroid positions, the higher the weight the more an instance attracts the centroid.	UCI: Dermathology Lymphoma Multiple Ovarian Semeion Sonar Some datasets used a large number of features 403.	High dimensional datasets	Accuracy. Often significantly better than non-weighted approaches.	Performs both feature and instance weighting. They claim that prior to their work none have previously combined instance and feature weighting approaches. Semi-supervised approach. Their Beta hyperparameter allows for tuning of the weighting. A interesting advantage of instance weighting which they point out is that the final weighting shows the importance of instances to the clustering result.

Adaptive clustering ensembles [15]	K-Means	Designed to focus on problematic areas of the feature space. Uses a boosting based ensemble. The generative mechanism for the ensemble is resampling. Sampling probability for each data point dynamical depending on the consistency of it's previous assignments in the ensemble. Their adaptive sampling favours points from regions close to the decision boundaries. Their clustering consistency index requires solving the label correspondence problem. Essentially, a high weight is given to points that inconsistently assigned.	Weights decide the sampling (with replacement) probability.	Artificial and real-world datasets: Galaxy, Half-rings, Wine, 3-gaussains, Iris, LON.	Challenging cluster structures. Non-spherical clusters.	Their instance weighting approach produced $\sim 1-5\%$ gains in accuracy. Experiment results show instance weighting is better than random sampling.	They also compared against instance selection. They found that it is best to use a larger k value, than the actual k (that is used in the consensus stage, to provide the final result) for the clustering in the ensemble. Experiments demonstrate faster convergence. MCLA, CSPA and EM where used for the ensemble consensus function.

Alternatives to the *k*-means algorithm that find better clusterings [16]	K-Harmonic Means	The instance weights for a given instance is the distance from centroids. Higher weight is given to points which are far from centroids. This loosely follows the analogy of boosting. Naturally, the harmonic mean is less influenced by points that are close to multiple centroids.	Weights are taken into consideration assigning instance membership to clusters.	The experiments are conducted with two different datasets, BIRCH 100 clusters two dimensions 10,000 instances and the Pelleg dataset, with 2, 4 and 6 dimension versions each with 50 clusters, 2500 instances, they adjust the dataset slightly to make the clusters more separated, which makes centroid jumping between clusters harder but separation easier.	Challenging structures.	To assess the quality of the clustering algorithm, in their main experiment they use square of the *k*-means objective function rather than objective function of the algorithm tested, this is an intrinsic measure. They use the square root to exaggerate the severity of poor solutions. KHM is shown to work better than KM.	Their work uses the harmonic mean rather than the arthritic mean and has both instance weights and soft membership added. The paper makes the point that varying instance weights is similar to boosting. They point out dimensionality reduction helps KHM, but don't compare this in their experiments. Their include some suggestion of generalising instance weighting into a unified framework. They investigated initialisation methods to get stable/robust clustering performance. The paper tests with two different initialisation strategies, Forgy (chooses random data points to place the cluster centres on) and Random Partition (assigns data points to random centres then calculates the positions of the centres to initialise the centroids).

9.2 ARI Results for LOFIWKM Experiments

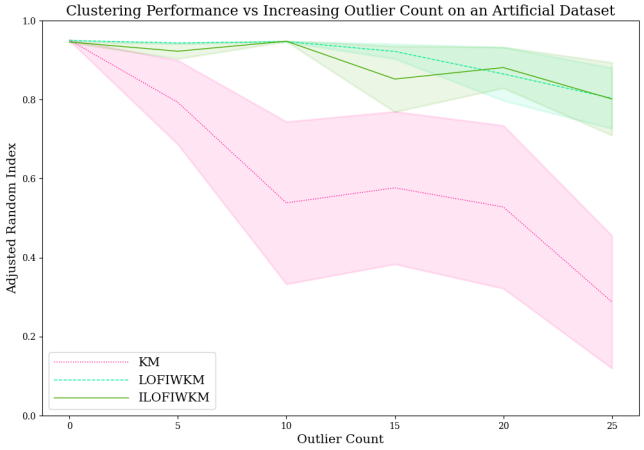


Figure 9.1: The average ARI score and standard deviation of k-means, LOFIWKM and ILOFIWKM on the synthetic dataset with an increasing amount of outliers.

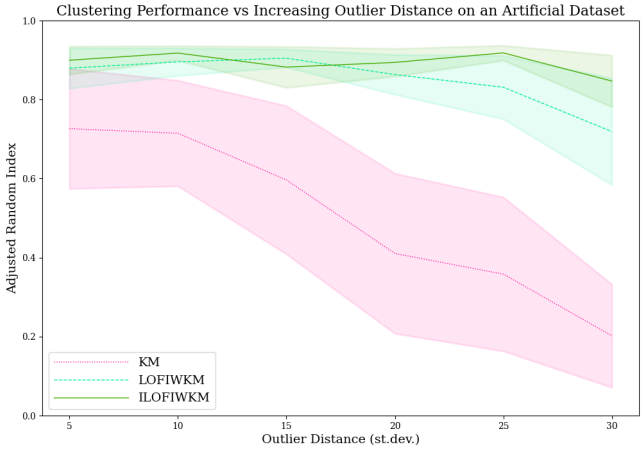


Figure 9.2: The average ARI score and standard deviation of k-means, LOFIWKM and ILOFIWKM on the synthetic dataset with increasingly distant outliers.

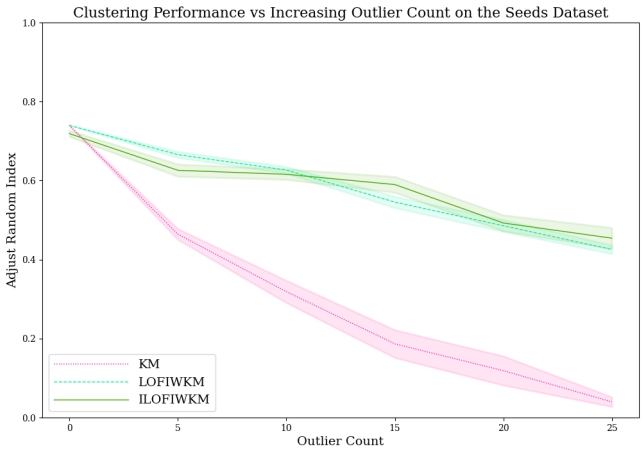


Figure 9.3: The average ARI score and standard deviation of k-means, LOFIWKM and ILOFIWKM on the Seeds dataset with an increasing amount of outliers.

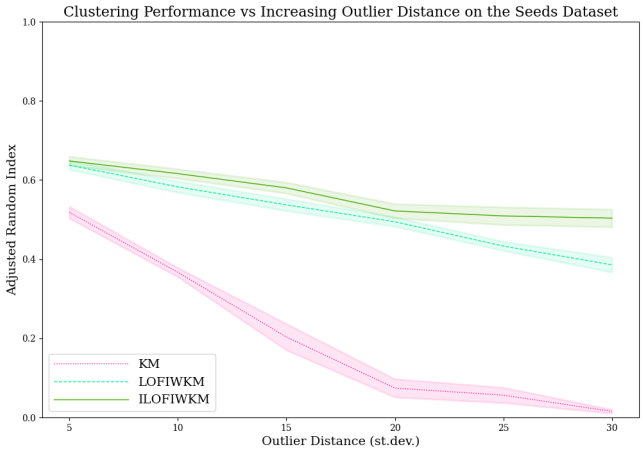


Figure 9.4: The average ARI score and standard deviation of k-means, LOFIWKM and ILOFIWKM on the Seeds dataset with increasingly distant outliers.

9.3 Clustering Results of FDR Dataset

These plots show a random sample of the results of the different approaches trialled on the FDR dataset. A total of six runs of each technique is seen across Figure 9.5 and 9.6.

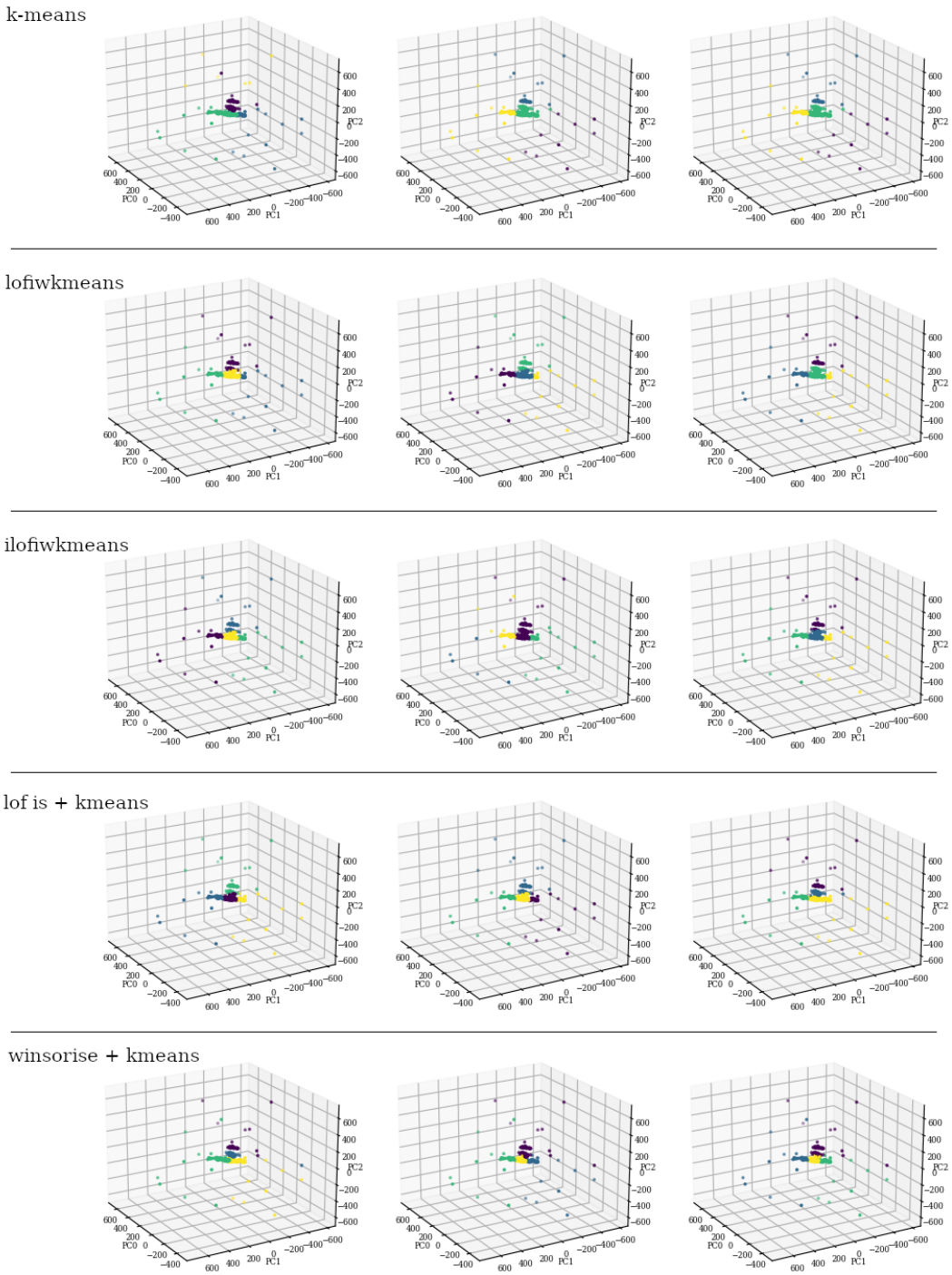


Figure 9.5: Scatter plots with colouration showing the clusters found by each of the approaches trialled across three runs.

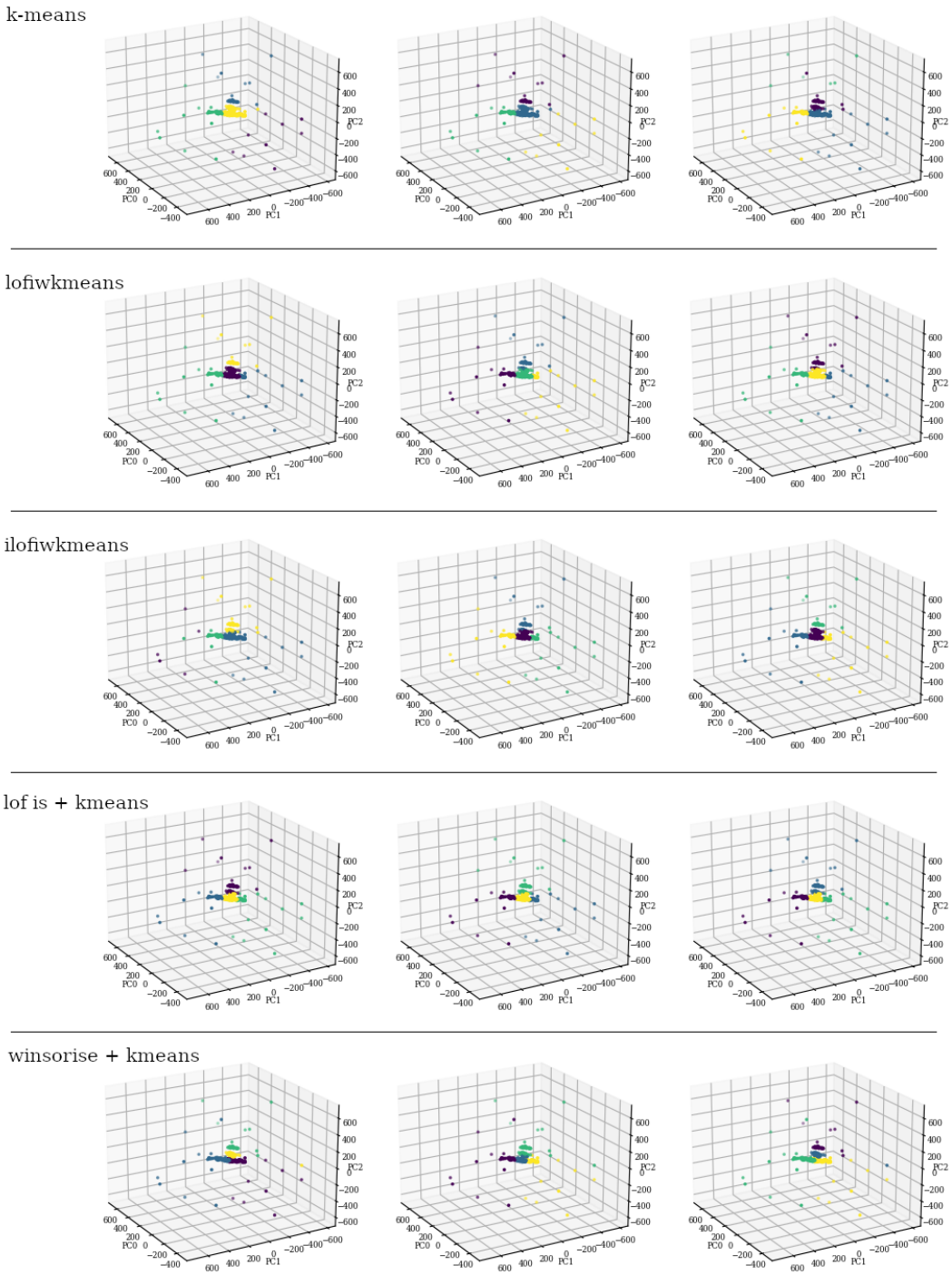


Figure 9.6: Scatter plots with colouration showing the clusters found by each of the approaches trialled across another three runs.

9.4 Worked Examples of Clustering Algorithms

9.4.1 Example Dataset

Id	x	y
P1	20	10
P2	21	8
P3	10	12
P4	9	13
P5	16	12
P6	8	15
P7	18	12

Table 9.5: The values of the example dataset.

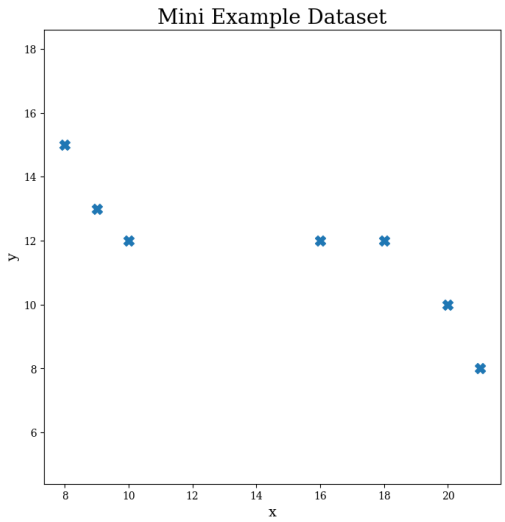


Figure 9.7: Scatter plot of the example dataset.

9.4.2 K-means Example

Assignment step, (first iteration).

$$d(C_1, P1) = \sqrt{(20-20)^2 + (10-10)^2}$$
$$d(C_1, P1) = \sqrt{(0)^2 + (0)^2}$$
$$d(C_1, P1) = \sqrt{0+0}$$
$$d(C_1, P1) = \sqrt{0} = 0.0$$

$$d(C_2, P1) = \sqrt{(21-20)^2 + (8-10)^2}$$
$$d(C_2, P1) = \sqrt{(1)^2 + (-2)^2}$$
$$d(C_2, P1) = \sqrt{1+4}$$
$$d(C_2, P1) = \sqrt{5} = 2.236$$

$$d(C_1, P2) = \sqrt{(20-21)^2 + (10-8)^2}$$
$$d(C_1, P2) = \sqrt{(-1)^2 + (2)^2}$$
$$d(C_1, P2) = \sqrt{1+4}$$
$$d(C_1, P2) = \sqrt{5} = 2.236$$

$$d(C_2, P2) = \sqrt{(21-21)^2 + (8-8)^2}$$
$$d(C_2, P2) = \sqrt{(0)^2 + (0)^2}$$
$$d(C_2, P2) = \sqrt{0+0}$$
$$d(C_2, P2) = \sqrt{0} = 0.0$$

$$d(C_1, P3) = \sqrt{(20-10)^2 + (10-12)^2}$$
$$d(C_1, P3) = \sqrt{(10)^2 + (-2)^2}$$
$$d(C_1, P3) = \sqrt{100+4}$$
$$d(C_1, P3) = \sqrt{104} = 10.198$$

$$d(C_2, P3) = \sqrt{(21-10)^2 + (8-12)^2}$$
$$d(C_2, P3) = \sqrt{(11)^2 + (-4)^2}$$
$$d(C_2, P3) = \sqrt{121+16}$$
$$d(C_2, P3) = \sqrt{137} = 11.705$$

$$d(C_1, P4) = \sqrt{(20-9)^2 + (10-13)^2}$$
$$d(C_1, P4) = \sqrt{(11)^2 + (-3)^2}$$
$$d(C_1, P4) = \sqrt{121+9}$$
$$d(C_1, P4) = \sqrt{130} = 11.402$$

$$d(C_2, P4) = \sqrt{(21-9)^2 + (8-13)^2}$$
$$d(C_2, P4) = \sqrt{(12)^2 + (-5)^2}$$
$$d(C_2, P4) = \sqrt{144+25}$$
$$d(C_2, P4) = \sqrt{169} = 13.0$$

$$d(C_1, P5) = \sqrt{(20-16)^2 + (10-12)^2}$$
$$d(C_1, P5) = \sqrt{(4)^2 + (-2)^2}$$
$$d(C_1, P5) = \sqrt{16+4}$$
$$d(C_1, P5) = \sqrt{20} = 4.472$$

$$d(C_2, P5) = \sqrt{(21-16)^2 + (8-12)^2}$$
$$d(C_2, P5) = \sqrt{(5)^2 + (-4)^2}$$
$$d(C_2, P5) = \sqrt{25+16}$$
$$d(C_2, P5) = \sqrt{41} = 6.403$$

$$d(C_1, P6) = \sqrt{(20-8)^2 + (10-15)^2}$$
$$d(C_1, P6) = \sqrt{(12)^2 + (-5)^2}$$
$$d(C_1, P6) = \sqrt{144+25}$$
$$d(C_1, P6) = \sqrt{169} = 13.0$$

$$d(C_2, P6) = \sqrt{(21-8)^2 + (8-15)^2}$$
$$d(C_2, P6) = \sqrt{(13)^2 + (-7)^2}$$
$$d(C_2, P6) = \sqrt{169+49}$$
$$d(C_2, P6) = \sqrt{218} = 14.765$$

$$d(C_1, P7) = \sqrt{(20-18)^2 + (10-12)^2}$$
$$d(C_1, P7) = \sqrt{(2)^2 + (-2)^2}$$
$$d(C_1, P7) = \sqrt{4+4}$$
$$d(C_1, P7) = \sqrt{8} = 2.828$$

$$d(C_2, P7) = \sqrt{(21-18)^2 + (8-12)^2}$$
$$d(C_2, P7) = \sqrt{(3)^2 + (-4)^2}$$
$$d(C_2, P7) = \sqrt{9+16}$$
$$d(C_2, P7) = \sqrt{25} = 5.0$$

Assigning each instance its nearest cluster produces the below partitioning.

Instance	Cluster
P1	C_1
P2	C_2
P3	C_1
P4	C_1
P5	C_1
P6	C_1
P7	C_1

Centroid update step (first iteration).

$$Cx_1^* = \frac{(20+10+9+16+8+18)}{6} = 13.5$$
$$Cy_1^* = \frac{(10+12+13+12+15+12)}{6} = 12.333$$

$$Cx_2^* = \frac{(21)}{1} = 21.0$$
$$Cy_2^* = \frac{(8)}{1} = 8.0$$

Assignment step, (second iteration).

$$d(C_1, P1) = \sqrt{(13.5-20)^2 + (12.333-10)^2}$$
$$d(C_1, P1) = \sqrt{(-6.5)^2 + (2.333)^2}$$
$$d(C_1, P1) = \sqrt{42.25 + 5.443}$$
$$d(C_1, P1) = \sqrt{47.693} = 6.906$$

$$d(C_2, P1) = \sqrt{(21-20)^2 + (8-10)^2}$$
$$d(C_2, P1) = \sqrt{(1)^2 + (-2)^2}$$
$$d(C_2, P1) = \sqrt{1+4}$$
$$d(C_2, P1) = \sqrt{5} = 2.236$$

$$d(C_1, P2) = \sqrt{(13.5-21)^2 + (12.333-8)^2}$$
$$d(C_1, P2) = \sqrt{(-7.5)^2 + (4.333)^2}$$
$$d(C_1, P2) = \sqrt{56.25 + 18.775}$$
$$d(C_1, P2) = \sqrt{75.025} = 8.662$$

$$d(C_2, P2) = \sqrt{(21-21)^2 + (8-8)^2}$$
$$d(C_2, P2) = \sqrt{(0)^2 + (0)^2}$$
$$d(C_2, P2) = \sqrt{0+0}$$
$$d(C_2, P2) = \sqrt{0} = 0.0$$

$$d(C_1, P3) = \sqrt{(13.5 - 10)^2 + (12.333 - 12)^2}$$
$$d(C_1, P3) = \sqrt{(3.5)^2 + (0.333)^2}$$
$$d(C_1, P3) = \sqrt{12.25 + 0.111}$$
$$d(C_1, P3) = \sqrt{12.361} = 3.516$$

$$d(C_2, P3) = \sqrt{(21 - 10)^2 + (8 - 12)^2}$$
$$d(C_2, P3) = \sqrt{(11)^2 + (-4)^2}$$
$$d(C_2, P3) = \sqrt{121 + 16}$$
$$d(C_2, P3) = \sqrt{137} = 11.705$$

$$d(C_1, P4) = \sqrt{(13.5 - 9)^2 + (12.333 - 13)^2}$$
$$d(C_1, P4) = \sqrt{(4.5)^2 + (-0.667)^2}$$
$$d(C_1, P4) = \sqrt{20.25 + 0.445}$$
$$d(C_1, P4) = \sqrt{20.695} = 4.549$$

$$d(C_2, P4) = \sqrt{(21 - 9)^2 + (8 - 13)^2}$$
$$d(C_2, P4) = \sqrt{(12)^2 + (-5)^2}$$
$$d(C_2, P4) = \sqrt{144 + 25}$$
$$d(C_2, P4) = \sqrt{169} = 13.0$$

$$d(C_1, P5) = \sqrt{(13.5 - 16)^2 + (12.333 - 12)^2}$$
$$d(C_1, P5) = \sqrt{(-2.5)^2 + (0.333)^2}$$
$$d(C_1, P5) = \sqrt{6.25 + 0.111}$$
$$d(C_1, P5) = \sqrt{6.361} = 2.522$$

$$d(C_2, P5) = \sqrt{(21 - 16)^2 + (8 - 12)^2}$$
$$d(C_2, P5) = \sqrt{(5)^2 + (-4)^2}$$
$$d(C_2, P5) = \sqrt{25 + 16}$$
$$d(C_2, P5) = \sqrt{41} = 6.403$$

$$d(C_1, P6) = \sqrt{(13.5 - 8)^2 + (12.333 - 15)^2}$$
$$d(C_1, P6) = \sqrt{(5.5)^2 + (-2.667)^2}$$
$$d(C_1, P6) = \sqrt{30.25 + 7.113}$$
$$d(C_1, P6) = \sqrt{37.363} = 6.113$$

$$d(C_2, P6) = \sqrt{(21-8)^2 + (8-15)^2}$$
$$d(C_2, P6) = \sqrt{(13)^2 + (-7)^2}$$
$$d(C_2, P6) = \sqrt{169 + 49}$$
$$d(C_2, P6) = \sqrt{218} = 14.765$$

$$d(C_1, P7) = \sqrt{(13.5-18)^2 + (12.333-12)^2}$$
$$d(C_1, P7) = \sqrt{(-4.5)^2 + (0.333)^2}$$
$$d(C_1, P7) = \sqrt{20.25 + 0.111}$$
$$d(C_1, P7) = \sqrt{20.361} = 4.512$$

$$d(C_2, P7) = \sqrt{(21-18)^2 + (8-12)^2}$$
$$d(C_2, P7) = \sqrt{(3)^2 + (-4)^2}$$
$$d(C_2, P7) = \sqrt{9 + 16}$$
$$d(C_2, P7) = \sqrt{25} = 5.0$$

Assigning each instance its nearest cluster produces the below partitioning.

Instance	Cluster
P1	C_2
P2	C_2
P3	C_1
P4	C_1
P5	C_1
P6	C_1
P7	C_1

Centroid update step (second iteration).

$$Cx_1^* = \frac{(10+9+16+8+18)}{5} = 12.2$$
$$Cy_1^* = \frac{(12+13+12+15+12)}{5} = 12.8$$

$$Cx_2^* = \frac{(20+21)}{2} = 20.5$$
$$Cy_2^* = \frac{(10+8)}{2} = 9.0$$

Assignment step, (third iteration).

$$d(C_1, P1) = \sqrt{(12.2 - 20)^2 + (12.8 - 10)^2}$$
$$d(C_1, P1) = \sqrt{(-7.8)^2 + (2.8)^2}$$
$$d(C_1, P1) = \sqrt{60.84 + 7.84}$$
$$d(C_1, P1) = \sqrt{68.68} = 8.287$$

$$d(C_2, P1) = \sqrt{(20.5 - 20)^2 + (9 - 10)^2}$$
$$d(C_2, P1) = \sqrt{(0.5)^2 + (-1)^2}$$
$$d(C_2, P1) = \sqrt{0.25 + 1}$$
$$d(C_2, P1) = \sqrt{1.25} = 1.118$$

$$d(C_1, P2) = \sqrt{(12.2 - 21)^2 + (12.8 - 8)^2}$$
$$d(C_1, P2) = \sqrt{(-8.8)^2 + (4.8)^2}$$
$$d(C_1, P2) = \sqrt{77.44 + 23.04}$$
$$d(C_1, P2) = \sqrt{100.48} = 10.024$$

$$d(C_2, P2) = \sqrt{(20.5 - 21)^2 + (9 - 8)^2}$$
$$d(C_2, P2) = \sqrt{(-0.5)^2 + (1)^2}$$
$$d(C_2, P2) = \sqrt{0.25 + 1}$$
$$d(C_2, P2) = \sqrt{1.25} = 1.118$$

$$d(C_1, P3) = \sqrt{(12.2 - 10)^2 + (12.8 - 12)^2}$$
$$d(C_1, P3) = \sqrt{(2.2)^2 + (0.8)^2}$$
$$d(C_1, P3) = \sqrt{4.84 + 0.64}$$
$$d(C_1, P3) = \sqrt{5.48} = 2.341$$

$$d(C_2, P3) = \sqrt{(20.5 - 10)^2 + (9 - 12)^2}$$
$$d(C_2, P3) = \sqrt{(10.5)^2 + (-3)^2}$$
$$d(C_2, P3) = \sqrt{110.25 + 9}$$
$$d(C_2, P3) = \sqrt{119.25} = 10.92$$

$$d(C_1, P4) = \sqrt{(12.2 - 9)^2 + (12.8 - 13)^2}$$
$$d(C_1, P4) = \sqrt{(3.2)^2 + (-0.2)^2}$$
$$d(C_1, P4) = \sqrt{10.24 + 0.04}$$
$$d(C_1, P4) = \sqrt{10.28} = 3.206$$

$$d(C_2, P4) = \sqrt{(20.5 - 9)^2 + (9 - 13)^2}$$
$$d(C_2, P4) = \sqrt{(11.5)^2 + (-4)^2}$$
$$d(C_2, P4) = \sqrt{132.25 + 16}$$
$$d(C_2, P4) = \sqrt{148.25} = 12.176$$

$$d(C_1, P5) = \sqrt{(12.2 - 16)^2 + (12.8 - 12)^2}$$
$$d(C_1, P5) = \sqrt{(-3.8)^2 + (0.8)^2}$$
$$d(C_1, P5) = \sqrt{14.44 + 0.64}$$
$$d(C_1, P5) = \sqrt{15.08} = 3.883$$

$$d(C_2, P5) = \sqrt{(20.5 - 16)^2 + (9 - 12)^2}$$
$$d(C_2, P5) = \sqrt{(4.5)^2 + (-3)^2}$$
$$d(C_2, P5) = \sqrt{20.25 + 9}$$
$$d(C_2, P5) = \sqrt{29.25} = 5.408$$

$$d(C_1, P6) = \sqrt{(12.2 - 8)^2 + (12.8 - 15)^2}$$
$$d(C_1, P6) = \sqrt{(4.2)^2 + (-2.2)^2}$$
$$d(C_1, P6) = \sqrt{17.64 + 4.84}$$
$$d(C_1, P6) = \sqrt{22.48} = 4.741$$

$$d(C_2, P6) = \sqrt{(20.5 - 8)^2 + (9 - 15)^2}$$
$$d(C_2, P6) = \sqrt{(12.5)^2 + (-6)^2}$$
$$d(C_2, P6) = \sqrt{156.25 + 36}$$
$$d(C_2, P6) = \sqrt{192.25} = 13.865$$

$$d(C_1, P7) = \sqrt{(12.2 - 18)^2 + (12.8 - 12)^2}$$
$$d(C_1, P7) = \sqrt{(-5.8)^2 + (0.8)^2}$$
$$d(C_1, P7) = \sqrt{33.64 + 0.64}$$
$$d(C_1, P7) = \sqrt{34.28} = 5.855$$

$$d(C_2, P7) = \sqrt{(20.5 - 18)^2 + (9 - 12)^2}$$
$$d(C_2, P7) = \sqrt{(2.5)^2 + (-3)^2}$$
$$d(C_2, P7) = \sqrt{6.25 + 9}$$
$$d(C_2, P7) = \sqrt{15.25} = 3.905$$

Assigning each instance its nearest cluster produces the below partitioning.

Centroid update step (third iteration).

Instance	Cluster
P1	C_2
P2	C_2
P3	C_1
P4	C_1
P5	C_1
P6	C_1
P7	C_2

$$Cx_1^* = \frac{(10+9+16+8)}{4} = 10.75$$
$$Cy_1^* = \frac{(12+13+12+15)}{4} = 13.0$$

$$Cx_2^* = \frac{(20+21+18)}{3} = 19.667$$
$$Cy_2^* = \frac{(10+8+12)}{3} = 10.0$$

Assignment step, (fourth iteration).

$$d(C_1, P1) = \sqrt{(10.75-20)^2 + (13-10)^2}$$
$$d(C_1, P1) = \sqrt{(-9.25)^2 + (3)^2}$$
$$d(C_1, P1) = \sqrt{85.562 + 9}$$
$$d(C_1, P1) = \sqrt{94.562} = 9.724$$

$$d(C_2, P1) = \sqrt{(19.667-20)^2 + (10-10)^2}$$
$$d(C_2, P1) = \sqrt{(-0.333)^2 + (0)^2}$$
$$d(C_2, P1) = \sqrt{0.111 + 0}$$
$$d(C_2, P1) = \sqrt{0.111} = 0.333$$

$$d(C_1, P2) = \sqrt{(10.75-21)^2 + (13-8)^2}$$
$$d(C_1, P2) = \sqrt{(-10.25)^2 + (5)^2}$$
$$d(C_1, P2) = \sqrt{105.062 + 25}$$
$$d(C_1, P2) = \sqrt{130.062} = 11.404$$

$$d(C_2, P2) = \sqrt{(19.667 - 21)^2 + (10 - 8)^2}$$
$$d(C_2, P2) = \sqrt{(-1.333)^2 + (2)^2}$$
$$d(C_2, P2) = \sqrt{1.777 + 4}$$
$$d(C_2, P2) = \sqrt{5.777} = 2.404$$

$$d(C_1, P3) = \sqrt{(10.75 - 10)^2 + (13 - 12)^2}$$
$$d(C_1, P3) = \sqrt{(0.75)^2 + (1)^2}$$
$$d(C_1, P3) = \sqrt{0.562 + 1}$$
$$d(C_1, P3) = \sqrt{1.562} = 1.25$$

$$d(C_2, P3) = \sqrt{(19.667 - 10)^2 + (10 - 12)^2}$$
$$d(C_2, P3) = \sqrt{(9.667)^2 + (-2)^2}$$
$$d(C_2, P3) = \sqrt{93.451 + 4}$$
$$d(C_2, P3) = \sqrt{97.451} = 9.872$$

$$d(C_1, P4) = \sqrt{(10.75 - 9)^2 + (13 - 13)^2}$$
$$d(C_1, P4) = \sqrt{(1.75)^2 + (0)^2}$$
$$d(C_1, P4) = \sqrt{3.062 + 0}$$
$$d(C_1, P4) = \sqrt{3.062} = 1.75$$

$$d(C_2, P4) = \sqrt{(19.667 - 9)^2 + (10 - 13)^2}$$
$$d(C_2, P4) = \sqrt{(10.667)^2 + (-3)^2}$$
$$d(C_2, P4) = \sqrt{113.785 + 9}$$
$$d(C_2, P4) = \sqrt{122.785} = 11.081$$

$$d(C_1, P5) = \sqrt{(10.75 - 16)^2 + (13 - 12)^2}$$
$$d(C_1, P5) = \sqrt{(-5.25)^2 + (1)^2}$$
$$d(C_1, P5) = \sqrt{27.562 + 1}$$
$$d(C_1, P5) = \sqrt{28.562} = 5.344$$

$$d(C_2, P5) = \sqrt{(19.667 - 16)^2 + (10 - 12)^2}$$
$$d(C_2, P5) = \sqrt{(3.667)^2 + (-2)^2}$$
$$d(C_2, P5) = \sqrt{13.447 + 4}$$
$$d(C_2, P5) = \sqrt{17.447} = 4.177$$

$$d(C_1, P6) = \sqrt{(10.75 - 8)^2 + (13 - 15)^2}$$
$$d(C_1, P6) = \sqrt{(2.75)^2 + (-2)^2}$$
$$d(C_1, P6) = \sqrt{7.562 + 4}$$
$$d(C_1, P6) = \sqrt{11.562} = 3.4$$

$$d(C_2, P6) = \sqrt{(19.667 - 8)^2 + (10 - 15)^2}$$
$$d(C_2, P6) = \sqrt{(11.667)^2 + (-5)^2}$$
$$d(C_2, P6) = \sqrt{136.119 + 25}$$
$$d(C_2, P6) = \sqrt{161.119} = 12.693$$

$$d(C_1, P7) = \sqrt{(10.75 - 18)^2 + (13 - 12)^2}$$
$$d(C_1, P7) = \sqrt{(-7.25)^2 + (1)^2}$$
$$d(C_1, P7) = \sqrt{52.562 + 1}$$
$$d(C_1, P7) = \sqrt{53.562} = 7.319$$

$$d(C_2, P7) = \sqrt{(19.667 - 18)^2 + (10 - 12)^2}$$
$$d(C_2, P7) = \sqrt{(1.667)^2 + (-2)^2}$$
$$d(C_2, P7) = \sqrt{2.779 + 4}$$
$$d(C_2, P7) = \sqrt{6.779} = 2.604$$

Assigning each instance its nearest cluster produces the below partitioning.

Instance	Cluster
P1	C_2
P2	C_2
P3	C_1
P4	C_1
P5	C_2
P6	C_1
P7	C_2

Centroid update step (fourth iteration).

$$Cx_1^* = \frac{(10 + 9 + 8)}{3} = 9.0$$
$$Cy_1^* = \frac{(12 + 13 + 15)}{3} = 13.333$$

$$Cx_2^* = \frac{(20 + 21 + 16 + 18)}{4} = 18.75$$
$$Cy_2^* = \frac{(10 + 8 + 12 + 12)}{4} = 10.5$$

Assignment step, (fifth iteration).

$$d(C_1, P1) = \sqrt{(9 - 20)^2 + (13.33 - 10)^2}$$
$$d(C_1, P1) = \sqrt{(-11)^2 + (3.33)^2}$$
$$d(C_1, P1) = \sqrt{121 + 11.089}$$
$$d(C_1, P1) = \sqrt{132.089} = 11.493$$

$$d(C_2, P1) = \sqrt{(18.75 - 20)^2 + (10.5 - 10)^2}$$
$$d(C_2, P1) = \sqrt{(-1.25)^2 + (0.5)^2}$$
$$d(C_2, P1) = \sqrt{1.562 + 0.25}$$
$$d(C_2, P1) = \sqrt{1.812} = 1.346$$

$$d(C_1, P2) = \sqrt{(9 - 21)^2 + (13.33 - 8)^2}$$
$$d(C_1, P2) = \sqrt{(-12)^2 + (5.33)^2}$$
$$d(C_1, P2) = \sqrt{144 + 28.409}$$
$$d(C_1, P2) = \sqrt{172.409} = 13.13$$

$$d(C_2, P2) = \sqrt{(18.75 - 21)^2 + (10.5 - 8)^2}$$
$$d(C_2, P2) = \sqrt{(-2.25)^2 + (2.5)^2}$$
$$d(C_2, P2) = \sqrt{5.062 + 6.25}$$
$$d(C_2, P2) = \sqrt{11.312} = 3.363$$

$$d(C_1, P3) = \sqrt{(9 - 10)^2 + (13.33 - 12)^2}$$
$$d(C_1, P3) = \sqrt{(-1)^2 + (1.33)^2}$$
$$d(C_1, P3) = \sqrt{1 + 1.769}$$
$$d(C_1, P3) = \sqrt{2.769} = 1.664$$

$$d(C_2, P3) = \sqrt{(18.75 - 10)^2 + (10.5 - 12)^2}$$
$$d(C_2, P3) = \sqrt{(8.75)^2 + (-1.5)^2}$$
$$d(C_2, P3) = \sqrt{76.562 + 2.25}$$
$$d(C_2, P3) = \sqrt{78.812} = 8.878$$

$$d(C_1, P4) = \sqrt{(9 - 9)^2 + (13.33 - 13)^2}$$
$$d(C_1, P4) = \sqrt{(0)^2 + (0.33)^2}$$
$$d(C_1, P4) = \sqrt{0 + 0.109}$$
$$d(C_1, P4) = \sqrt{0.109} = 0.33$$

$$d(C_2, P4) = \sqrt{(18.75 - 9)^2 + (10.5 - 13)^2}$$
$$d(C_2, P4) = \sqrt{(9.75)^2 + (-2.5)^2}$$
$$d(C_2, P4) = \sqrt{95.062 + 6.25}$$
$$d(C_2, P4) = \sqrt{101.312} = 10.065$$

$$d(C_1, P5) = \sqrt{(9 - 16)^2 + (13.33 - 12)^2}$$
$$d(C_1, P5) = \sqrt{(-7)^2 + (1.33)^2}$$
$$d(C_1, P5) = \sqrt{49 + 1.769}$$
$$d(C_1, P5) = \sqrt{50.769} = 7.125$$

$$d(C_2, P5) = \sqrt{(18.75 - 16)^2 + (10.5 - 12)^2}$$
$$d(C_2, P5) = \sqrt{(2.75)^2 + (-1.5)^2}$$
$$d(C_2, P5) = \sqrt{7.562 + 2.25}$$
$$d(C_2, P5) = \sqrt{9.812} = 3.132$$

$$d(C_1, P6) = \sqrt{(9 - 8)^2 + (13.33 - 15)^2}$$
$$d(C_1, P6) = \sqrt{(1)^2 + (-1.67)^2}$$
$$d(C_1, P6) = \sqrt{1 + 2.789}$$
$$d(C_1, P6) = \sqrt{3.789} = 1.947$$

$$d(C_2, P6) = \sqrt{(18.75 - 8)^2 + (10.5 - 15)^2}$$
$$d(C_2, P6) = \sqrt{(10.75)^2 + (-4.5)^2}$$
$$d(C_2, P6) = \sqrt{115.562 + 20.25}$$
$$d(C_2, P6) = \sqrt{135.812} = 11.654$$

$$d(C_1, P7) = \sqrt{(9-18)^2 + (13.33-12)^2}$$
$$d(C_1, P7) = \sqrt{(-9)^2 + (1.33)^2}$$
$$d(C_1, P7) = \sqrt{81 + 1.769}$$
$$d(C_1, P7) = \sqrt{82.769} = 9.098$$

$$d(C_2, P7) = \sqrt{(18.75-18)^2 + (10.5-12)^2}$$
$$d(C_2, P7) = \sqrt{(0.75)^2 + (-1.5)^2}$$
$$d(C_2, P7) = \sqrt{0.562 + 2.25}$$
$$d(C_2, P7) = \sqrt{2.812} = 1.677$$

Assigning each instance its nearest cluster produces the below partitioning.

Instance	Cluster
P1	C_2
P2	C_2
P3	C_1
P4	C_1
P5	C_2
P6	C_1
P7	C_2

As no instances switch between clusters, the algorithm has converged. The final clustering result with Voronoi diagram can be seen in Figure 9.8.

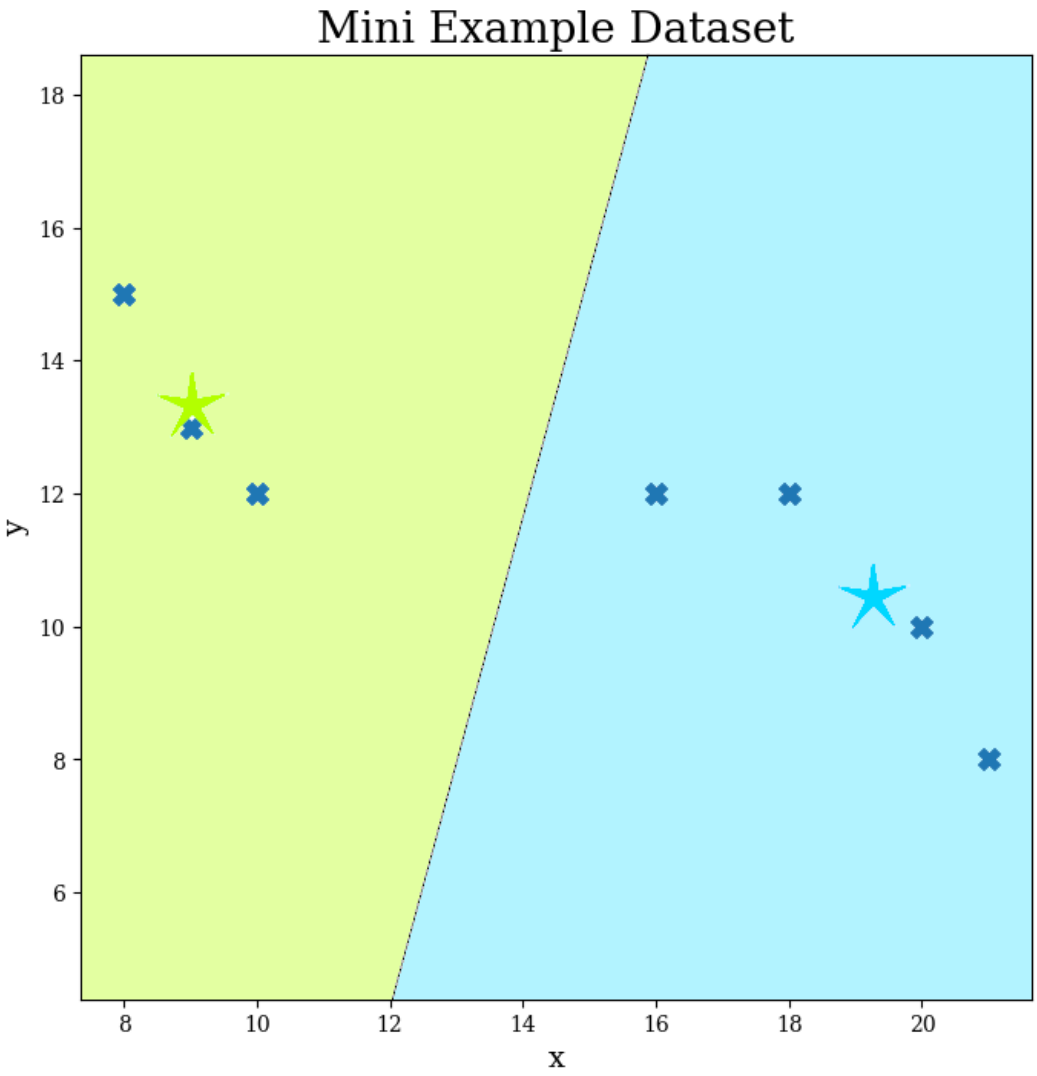


Figure 9.8: The example dataset clustered by k-means.

9.4.3 Spectral Example

Firstly, using 2 nearest neighbours using Euclidean distance on the example dataset produces a graph, see 9.9. The distance calculations have been omitted for brevity.

Note that, on this toy dataset using specifically *2* nearest neighbours does produce two separate graphs, this could be manually detected at this stage. But for the purpose of the demonstration, let's continue with the spectral cluster-

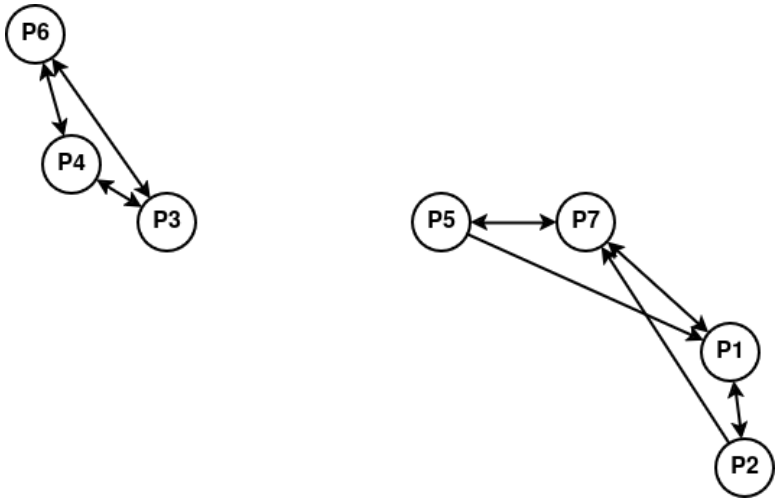


Figure 9.9: The graph derived from the example dataset using 2 nearest neighbours.

ing method.

The connectivity of the graph in Figure 9.9 can be modelled by a symmetric matrix, A.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

To partition matrix A, first the degree matrix is calculated. This is simply the sum of each row, placed on the diagonal.

$$D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

A is then subtracted from D to produce a symmetric matrix L.

$$L = D - A$$

$$L = \begin{bmatrix} 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ -1 & 2 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 2 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 2 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 2 & 0 & -1 \\ 0 & 0 & -1 & -1 & 0 & 2 & 0 \\ -1 & -1 & 0 & 0 & -1 & 0 & 3 \end{bmatrix}$$

L has a determinant of 0. Multiply this matrix by the λ times I. Where I is the identity matrix (the "do nothing" matrix) to and retain a determinant of 0.

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\lambda I = \begin{bmatrix} \lambda & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda \end{bmatrix}$$

$$det(L-(\lambda I)) = det\left(\begin{bmatrix} 3-\lambda & -1 & 0 & 0 & -1 & 0 & -1 \\ -1 & 2-\lambda & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 2-\lambda & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 2-\lambda & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 2-\lambda & 0 & -1 \\ 0 & 0 & -1 & -1 & 0 & 2-\lambda & 0 \\ -1 & -1 & 0 & 0 & -1 & 0 & 2-\lambda \end{bmatrix}\right) = 0$$

Next, the eigenvalues $\lambda_{1..n}$ are found. To do this the characteristic polynomial is found and solved for the eigenvalues. For the 7×7 example arranging the formula to find the characteristic polynomial and then solving for the 7 eigen-

values is very complex (although finding the 2 eigenvalues of a 2×2 matrix is quite manageable by hand). Below shows the characteristic polynomial (calculated by wolfram-alpha - not by hand).

$$-\lambda^7 + 16\lambda^6 - 101\lambda^5 + 314\lambda^4 - 480\lambda^3 + 288\lambda^2$$

Once the eigenvalues are obtained, the non-zero eigenvectors can then be solved for ensuring that $L\overrightarrow{v} = \lambda\overrightarrow{v}$ is true. Again this is non-trivial to by hand.

Thankfully using python, the eigenvalue and eigenvector pairs can be calculated. [1]

```
# calculate eigenvalues and eigenvectors
vals, vecs = np.linalg.eig(L)

# sort both based on eigenvalue
vecs = vecs[:,np.argsort(vals)]
vals = vals[np.argsort(vals)]
```

Table 9.4.3 shows the resultant values:

[1] William Fleshman's 2019 post on Towards Data Science titled *Spectral Clustering- Foundation and Application* demonstrates this clearly: `https://towardsdatascience.com/spectral-clustering-aba2640c0d5b`.

Eigenvalue λ_x	$\overrightarrow{v}$ Eigenvector
0	$\begin{bmatrix} 0 \\ 0 \\ -0.577 \\ -0.577 \\ 0 \\ -0.577 \\ 0 \end{bmatrix}$
0	$\begin{bmatrix} -0.5 \\ -0.5 \\ 0 \\ 0 \\ -0.5 \\ 0 \\ -0.5 \end{bmatrix}$
2	$\begin{bmatrix} 0 \\ -0.707 \\ 0 \\ 0 \\ 0.707 \\ 0 \\ 0 \end{bmatrix}$
3	$\begin{bmatrix} 0 \\ 0 \\ -0.408 \\ 0.816 \\ 0 \\ -0.408 \\ 0 \end{bmatrix}$
3	$\begin{bmatrix} 0 \\ 0 \\ 0.513 \\ 0.293 \\ 0 \\ -0.807 \\ 0 \end{bmatrix}$
4	$\begin{bmatrix} 0.866 \\ -0.289 \\ 0 \\ 0 \\ -0.289 \\ 0 \\ -0.289 \end{bmatrix}$
4	$\begin{bmatrix} -0.127 \\ -0.361 \\ 0 \\ 0 \\ -0.361 \\ 0 \\ 0.85 \end{bmatrix}$

Table 9.6: The eigenvalues their corresponding eigenvectors for the example dataset.

These can be checked for correctness. Lets check the Eigenvalue of 2 follows $L\overrightarrow{v} = \lambda\overrightarrow{v}$ should be true.

$$\begin{bmatrix} 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ -1 & 2 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 2 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 2 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 2 & 0 & -1 \\ 0 & 0 & -1 & -1 & 0 & 2 & 0 \\ -1 & -1 & 0 & 0 & -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ -0.707 \\ 0 \\ 0 \\ 0.707 \\ 0 \\ 0 \end{bmatrix} = 2 \cdot \begin{bmatrix} 0 \\ -0.707 \\ 0 \\ 0 \\ 0.707 \\ 0 \\ 0 \end{bmatrix}$$

On the left side of the equation, there is matrix multiplication:

$$\begin{bmatrix} (3 \times 0) + (-1 \times -0.707) + (0 \times 0) + (0 \times 0) + (-1 \times 0.707) + (0 \times 0) + (-1 \times 0) \\ (-1 \times 0) + (2 \times -0.707) + (0 \times 0) + (0 \times 0) + (0 \times 0.707) + (0 \times 0) + (-1 \times 0) \\ (0 \times 0) + (0 \times -0.707) + (2 \times 0) + (-1 \times 0) + (0 \times 0.707) + (-1 \times 0) + (0 \times 0) \\ (0 \times 0) + (0 \times -0.707) + (-1 \times 0) + (2 \times 0) + (0 \times 0.707) + (-1 \times 0) + (0 \times 0) \\ (-1 \times 0) + (0 \times -0.707) + (0 \times 0) + (0 \times 0) + (2 \times 0.707) + (0 \times 0) + (-1 \times 0) \\ (0 \times 0) + (0 \times -0.707) + (-1 \times 0) + (-1 \times 0) + (0 \times 0.707) + (2 \times 0) + (0 \times 0) \\ (-1 \times 0) + (-1 \times -0.707) + (0 \times 0) + (0 \times 0) + (-1 \times 0.707) + (0 \times 0) + (3 \times 0) \end{bmatrix} = \begin{bmatrix} 0 \\ -1.414 \\ 0 \\ 0 \\ 1.414 \\ 0 \\ 0 \end{bmatrix}$$

On right right of the equation, their is scalar multiplication:

$$2 \cdot \begin{bmatrix} 0 \\ -0.707 \\ 0 \\ 0 \\ 0.707 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -1.414 \\ 0 \\ 0 \\ 1.414 \\ 0 \\ 0 \end{bmatrix}$$

Now the graph can be partitioned. The Fiedler vector is the eigenvector corresponding to the second smallest eigenvalue, in this case, this is the vector

containing: -0.5, -0.5, 0, 0, -0.5, 0, -0.5. Here, the polarity of values indicates their class assignment. Assigning negative values to cluster 0 (green) and positive values to cluster 1 (blue), a suitable partitioning of the graph is obtained, see Figure 9.10.

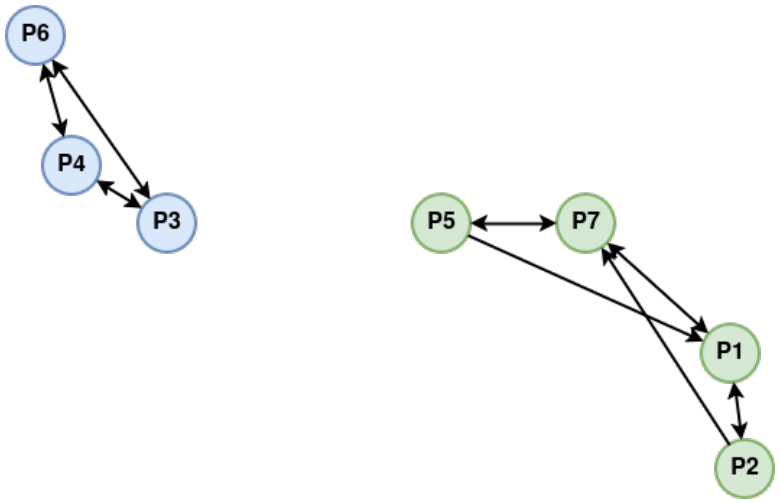


Figure 9.10: The partitioned graph.

9.5 Worked Examples of NMI

9.5.1 NMI Formulae

$$H(U) = -\sum_{i=1}^{|U|} P(i)\log(P(i))$$

$$H(V) = -\sum_{i=1}^{|V|} P(i)\log(P(i))$$

$$\mathrm{MI}(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right)$$

$$\mathrm{NMI}(U,V) = \frac{\mathrm{MI}(U,V)}{\mathrm{mean}(H(U),H(V))}$$

9.5.2 Example of NMI score on Poor Clustering

When there is no correlation between the ground truth (U) and the cluster labels (V), the NMI score is 0.

$$U = [0, 0, 0, 0, 1, 1, 1, 1]$$

$$V = [1, 0, 1, 0, 1, 0, 1, 0]$$

$$H(U) = -((\frac{4}{8}) \times log_2(\frac{4}{8})) + ((\frac{4}{8}) \times log_2(\frac{4}{8})))$$

$$H(U) = -((0.5 \times -1) + (0.5 \times -1))$$

$$H(U) = 1$$

$$H(V) = -((\frac{4}{8}) \times log_2(\frac{4}{8})) + ((\frac{4}{8}) \times log_2(\frac{4}{8})))$$

$$H(V) = -((0.5 \times -1) + (0.5 \times -1))$$

$$H(V) = 1$$

Table 9.7: The joint and marginal (grey) probabilities for U compared to V.

		i = 0	i = 1	
j	0	$\frac{2}{8}$	$\frac{2}{8}$	$\frac{4}{8}$
	1	$\frac{2}{8}$	$\frac{2}{8}$	$\frac{4}{8}$
		$\frac{4}{8}$	$\frac{4}{8}$	

$$\text{MI}(U,V) = (\frac{2}{8} \times log_2(\frac{\frac{2}{8}}{\frac{4}{8} \times \frac{4}{8}})) + (\frac{2}{8} \times log_2(\frac{\frac{2}{8}}{\frac{4}{8} \times \frac{4}{8}})) + (\frac{2}{8} \times log_2(\frac{\frac{2}{8}}{\frac{4}{8} \times \frac{4}{8}})) + (\frac{2}{8} \times log_2(\frac{\frac{2}{8}}{\frac{4}{8} \times \frac{4}{8}}))$$

$$\text{MI}(U,V) = (0.25 \times log_2(1)) + (0.25 \times log_2(1)) + (0.25 \times log_2(1) + (0.25 \times log_2(1))$$

$$\text{MI}(U,V) = 0 + 0 + 0 + 0$$

$$\text{MI}(U,V) = 0$$

$$\text{NMI}(U,V) = \frac{0}{(\frac{1.0+1.0}{2})}$$

$$\text{NMI}(U,V) = \frac{0}{1}$$

$$\text{NMI}(U,V) = 0$$

9.5.3 Example of NMI score on a Moderate Clustering

A discrepancy between the ground truth and the cluster labels produces a moderate NMI score.

$$U = [0, 0, 0, 0, 1, 1, 1, 1]$$

$$V = [0, 0, 1, 0, 1, 1, 1, 1]$$

$$H(U) = -((\frac{4}{8}) \times log_2(\frac{4}{8})) + ((\frac{4}{8}) \times log_2(\frac{4}{8})))$$

$$H(U) = -((0.5 \times -1) + (0.5 \times -1))$$

$$H(U) = 1$$

$$H(V) = -((\frac{3}{8}) \times log_2(\frac{3}{8})) + ((\frac{5}{8}) \times log_2(\frac{5}{8})))$$

$$H(V) = -((0.375 \times -1.415) + (0.625 \times -0.678))$$

$$H(V) = 0.9544.$$

Table 9.8: The joint and marginal (grey) probabilities for U compared to V.

		i = 0	i = 1	
j	0	$\frac{3}{8}$	$\frac{0}{8}$	$\frac{3}{8}$
	1	$\frac{1}{8}$	$\frac{4}{8}$	$\frac{5}{8}$
		$\frac{4}{8}$	$\frac{4}{8}$	

$$\text{MI}(U,V) = (\frac{3}{8} \times log_2(\frac{\frac{3}{8}}{\frac{3}{8} \times \frac{4}{8}})) + (\frac{1}{8} \times log_2(\frac{\frac{1}{8}}{\frac{5}{8} \times \frac{4}{8}})) + (\frac{0}{8} \times log_2(\frac{\frac{0}{8}}{\frac{4}{8} \times \frac{3}{8}})) + (\frac{4}{8} \times log_2(\frac{\frac{4}{8}}{\frac{4}{8} \times \frac{5}{8}}))$$

$$\text{MI}(U,V) = (0.375 \times log_2(2)) + (0.125 \times log_2(0.4)) + (0 \times log_2(0) + (0.5 \times log_2(1.6))$$

$$\text{MI}(U,V) = 0.375 + -0.165. + 0 + 0.339.$$

$$\text{MI}(U,V) = 0.5487949406953986$$

$$\text{NMI}(U,V) = \frac{0.5487949406953986}{(\frac{1.0+0.95443}{2})}$$

$$\text{NMI}(U,V) = \frac{0.5487949406953986}{0.977217}$$

$$\text{NMI}(U,V) = 0.5615896374043826$$

9.5.4 Example of NMI score on Perfect Clustering

The best possible NMI score is 1. This is seen when the ground truth and cluster labels match exactly.

$$U = [0, 0, 0, 0, 1, 1, 1, 1]$$

$$V = [0, 0, 0, 0, 1, 1, 1, 1]$$

$$H(U) = -((\frac{4}{8}) \times log_2(\frac{4}{8})) + ((\frac{4}{8}) \times log_2(\frac{4}{8})))$$

$$H(U) = -((0.5 \times -1) + (0.5 \times -1))$$

$$H(U) = 1$$

$$H(V) = -((\frac{4}{8}) \times log_2(\frac{4}{8})) + ((\frac{4}{8}) \times log_2(\frac{4}{8})))$$

$$H(V) = -((0.5 \times -1) + (0.5 \times -1))$$

$$H(V) = 1$$

Table 9.9: The joint and marginal (grey) probabilities for U compared to V.

		i = 0	i = 1	
j	0	$\frac{4}{8}$	$\frac{0}{8}$	$\frac{4}{8}$
	1	$\frac{0}{8}$	$\frac{4}{8}$	$\frac{4}{8}$
		$\frac{4}{8}$	$\frac{4}{8}$	

$$\text{MI}(U, V) = (\frac{4}{8} \times log_2(\frac{\frac{4}{8}}{\frac{4}{8} \times \frac{4}{8}})) + (\frac{0}{8} \times log_2(\frac{\frac{0}{8}}{\frac{4}{8} \times \frac{4}{8}})) + (\frac{0}{8} \times log_2(\frac{\frac{0}{8}}{\frac{4}{8} \times \frac{4}{8}})) + (\frac{4}{8} \times log_2(\frac{\frac{4}{8}}{\frac{4}{8} \times \frac{4}{8}}))$$

$$\text{MI}(U, V) = (0.5 \times log_2(2) + (0 \times log_2(0) + (0 \times log_2(0) + (0.5 \times log_2(2))$$

$$\text{MI}(U, V) = 0.5 + 0 + 0 + 0.5$$

$$\text{MI}(U, V) = 1$$

$$\text{NMI}(U,V) = \frac{1}{(\frac{1+1}{2})}$$
$$\text{NMI}(U,V) = \frac{1}{1}$$
$$\text{NMI}(U,V) = 1$$

9.5.5 Example of NMI score on Alternative Perfect Clustering

It is not necessary to solve label correspondence problem.

$$U = [0,0,0,0,1,1,1,1]$$
$$V = [1,1,1,1,0,0,0,0]$$

$$H(U) = -((\frac{4}{8}) \times log_2(\frac{4}{8})) + ((\frac{4}{8}) \times log_2(\frac{4}{8})))$$
$$H(U) = -((0.5 \times -1) + (0.5 \times -1))$$
$$H(U) = 1$$

$$H(V) = -((\frac{4}{8}) \times log_2(\frac{4}{8})) + ((\frac{4}{8}) \times log_2(\frac{4}{8})))$$
$$H(V) = -((0.5 \times -1) + (0.5 \times -1))$$
$$H(V) = 1$$

Table 9.10: The joint and marginal (grey) probabilities for U compared to V.

		i = 0	i = 1	
j	0	$\frac{0}{8}$	$\frac{4}{8}$	$\frac{4}{8}$
	1	$\frac{4}{8}$	$\frac{0}{8}$	$\frac{4}{8}$
		$\frac{4}{8}$	$\frac{4}{8}$	

$$\text{MI}(U,V) = ((\frac{0}{8} \times log_2(\frac{\frac{0}{8}}{\frac{4}{8} \times \frac{4}{8}})) + \frac{4}{8} \times log_2(\frac{\frac{4}{8}}{\frac{4}{8} \times \frac{4}{8}})) + (\frac{4}{8} \times log_2(\frac{\frac{4}{8}}{\frac{4}{8} \times \frac{4}{8}}) + (\frac{0}{8} \times log_2(\frac{\frac{0}{8}}{\frac{4}{8} \times \frac{4}{8}})))$$

$$\text{MI}(U,V) = (0.5 \times log_2(0) + (0 \times log_2(2) + (0 \times log_2(2) + (0.5 \times log_2(0))$$

$$\text{MI}(U,V) = 0 + 0.5 + 0.5 + 0$$

$$\text{MI}(U,V) = 1$$

$$\text{NMI}(U,V) = \frac{1}{(\frac{1+1}{2})}$$

$$\text{NMI}(U,V) = \frac{1}{1}$$

$$\text{NMI}(U,V) = 1$$

9.5.6 Example of NMI score on Imbalanced Data (Most Likely Error)

The NMI score accounts for imbalanced distributions of the class labels. Assuming errors randomly are randomly distributed, there is greater chance of an error occurring in a position in V where 0 should be as per the ground truth U. Hence, NMI penalises the score less compared to the mistake being in the opposition direction.

$$U = [0, 0, 0, 0, 0, 0, 1, 1]$$

$$V = [0, 0, 1, 0, 0, 0, 1, 1]$$

$$H(U) = -((\frac{6}{8}) \times log_2(\frac{6}{8})) + ((\frac{2}{8}) \times log_2(\frac{2}{8})))$$

$$H(U) = -((0.75 \times -0.4150374992788438) + (0.25 \times -2.0))$$

$$H(U) = 0.8113.$$

$$H(V) = -((\frac{5}{8}) \times log(\frac{5}{8})) + ((\frac{3}{8}) \times log_2(\frac{3}{8})))$$

$$H(V) = -((0.625 \times -0.678) + (0.375 \times -1.415))$$

$$H(V) = 0.9544.$$

Table 9.11: The joint and marginal (grey) probabilities for U compared to V.

		i = 0	i = 1	
j	0	$\frac{5}{8}$	$\frac{0}{8}$	$\frac{5}{8}$
	1	$\frac{1}{8}$	$\frac{2}{8}$	$\frac{3}{8}$
		$\frac{6}{8}$	$\frac{2}{8}$	

$$\text{MI}(U, V) = (\frac{5}{8} \times log_2(\frac{\frac{5}{8}}{\frac{5}{8} \times \frac{6}{8}})) + (\frac{0}{8} \times log_2(\frac{\frac{0}{8}}{\frac{5}{8} \times \frac{2}{8}})) + (\frac{1}{8} \times log_2(\frac{\frac{1}{8}}{\frac{3}{8} \times \frac{6}{8}})) + (\frac{2}{8} \times log_2(\frac{\frac{2}{8}}{\frac{3}{8} \times \frac{2}{8}}))$$

$$\text{MI}(U, V) = (0.625 \times log_2(1.\overline{3})) + (0 \times log_2(0)) + (0.125 \times log_2(0.\overline{4}) + (0.25 \times log_2(2.\overline{6}))$$

$$\text{MI}(U, V) = 0.2594. + 0 + -0.1462. + 0.3538.$$

$$\text{MI}(U, V) = 0.4669.$$

$$\text{NMI}(U,V) = \frac{0.4669.}{(\frac{0.9544.+0.8113.}{2})}$$
$$\text{NMI}(U,V) = \frac{0.4669.}{0.8829.}$$
$$\text{NMI}(U,V) = 0.5289.$$

9.5.7 Example of NMI score on Imbalanced Data (Less Likely Error)

Conversely, to the above example, again assuming errors randomly are randomly distributed, there is lesser chance of an error occurring in a position in V where 1 should be as per the ground truth U. Hence, NMI penalises the score more compared to the mistake being in the opposition direction.

$$U = [0,0,0,0,0,0,1,1]$$
$$V = [0,0,0,0,0,0,0,1]$$

$$H(U) = -((\frac{6}{8}) \times log_2(\frac{6}{8})) + ((\frac{2}{8}) \times log(\frac{2}{8})))$$
$$H(U) = -((0.75 \times -0.4150374992788438) + (0.25 \times -2.0))$$
$$H(U) = 0.8113.$$

$$H(V) = -((\frac{7}{8}) \times log(\frac{7}{8})) + ((\frac{1}{8}) \times log_2(\frac{1}{8})))$$
$$H(V) = -((0.875 \times -0.1923) + (0.125 \times -3))$$
$$H(V) = 0.5436.$$

Table 9.12: The joint and marginal (grey) probabilities for U compared to V.

		i = 0	i = 1	
j	0	$\frac{6}{8}$	$\frac{1}{8}$	$\frac{7}{8}$
	1	$\frac{0}{8}$	$\frac{1}{8}$	$\frac{1}{8}$
		$\frac{6}{8}$	$\frac{2}{8}$	

$$\text{MI}(U,V) = (\frac{6}{8} \times log_2(\frac{\frac{6}{8}}{\frac{7}{8} \times \frac{6}{8}})) + (\frac{1}{8} \times log_2(\frac{\frac{1}{8}}{\frac{7}{8} \times \frac{2}{8}})) + (\frac{0}{8} \times log_2(\frac{\frac{0}{8}}{\frac{1}{8} \times \frac{6}{8}})) + (\frac{1}{8} \times log_2(\frac{\frac{1}{8}}{\frac{1}{8} \times \frac{2}{8}}))$$

$$\text{MI}(U,V) = (0.75 \times log_2(1.1429.)) + (0.125 \times log_2(0.5714.)) + (0 \times log_2(0)) + (0.125 \times log_2(4))$$

$$\text{MI}(U,V) = 0.1445 + -0.1001. + 0 + 0.25$$

$$\text{MI}(U,V) = 0.2936.$$

$$\text{NMI}(U,V) = \frac{0.2936.}{(\frac{0.8113.+0.5436.}{2})}$$

$$\text{NMI}(U,V) = \frac{0.2936.}{0.6774.}$$

$$\text{NMI}(U,V) = 0.4334.$$

9.6 Comparison of Intrinsic Measures against an Outlier

In Figure 9.11, the colour represent the labels from a "perfect" clustering. Left: The scores executed on the dataset and labels with no outlier. Right: one point is move into an outlying position, again the scores are shown. Notice that the huge 7.4x flucation in the Calinski Harabasz score, compared to the (relatively) more modest changes in the other scores.

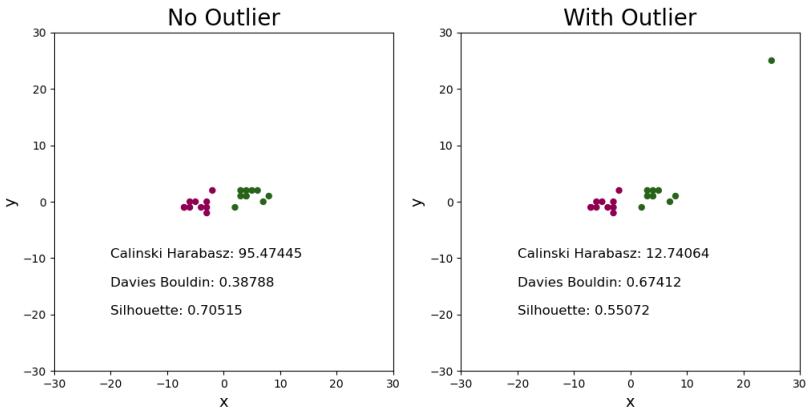


Figure 9.11: Comparison of how Calinski Harabasz, Davies Bouldin, Silhouette Coe. are effected by the presence of an outlier in the data.

9.7 Worked Example of Imbalance Ratio Calculation

Given a dataset size N of 2000 and given an imbalance (percentage) target i = +400%.

$$L = \lfloor \frac{N}{2 + (\frac{i}{100})} \times (1 + (\frac{i}{100})) \rfloor$$

$$S = N - L$$

$$L = \lfloor \frac{2000}{2 + (\frac{4}{100})} \times (1 + (\frac{4}{100})) \rfloor$$

$$L = \lfloor \frac{2000}{6} \times (1 + 4) \rfloor$$

$$L = \lfloor 333.\overline{3} \times 5 \rfloor$$

$$L = \lfloor 1666.\overline{6} \rfloor$$

$$L = 1666$$

$$S = 2000 - 1666$$

$$S = 334$$

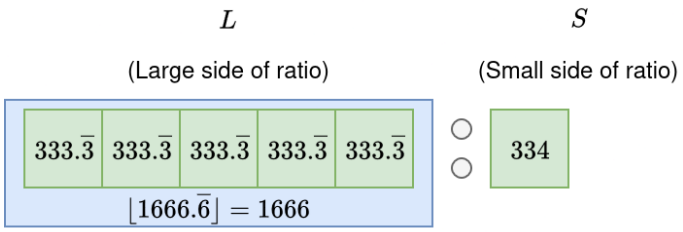


Figure 9.12: A visual representation of the calculated instance counts for the imbalanced dataset.

The solution, $1666 : 334$ can be thought of as the ratio of: $(334 + 400\%) : 334$.

Here is a another example, this time with $i = +100\%$.

$$L = \lfloor \frac{2000}{2 + (\frac{1}{100})} \times (1 + (\frac{1}{100})) \rfloor$$
$$L = \lfloor \frac{2000}{3} \times (1 + 1) \rfloor$$
$$L = \lfloor 666.\overline{6} \times 2 \rfloor$$
$$L = \lfloor 1333.\overline{3} \rfloor$$
$$L = 1333$$
$$S = 2000 - 1333$$
$$S = 667$$

The solution, $1333 : 667$ can be thought of as the ratio of: $(667 + 100\%) : 667$.

Here is another example with $i = 0\%$.

$$L = \lfloor \frac{2000}{2 + (\frac{0}{100})} \times (1 + (\frac{0}{100})) \rfloor$$
$$L = \lfloor \frac{2000}{2} \times (1 + 0) \rfloor$$
$$L = \lfloor 1000 \times 1 \rfloor$$
$$L = \lfloor 1000 \rfloor$$
$$L = 1000$$
$$S = 2000 - 1000$$
$$S = 1000$$

Of course, with 0% imbalance factor, and an even number of instances, the ratio is perfectly balanced $1000 : 1000$.

9.8 IWSE Bags Parameter

The results in Figures 9.13 and 9.14 show that for the values of M tested, generally the higher M values yielded better clustering performance for the "Three Sizes" and "Ball Line" datasets.

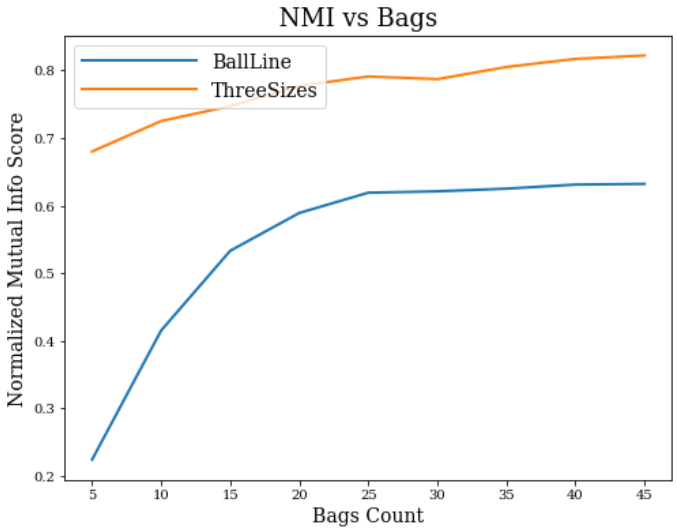


Figure 9.13: The impact on NMI when adjusting the bags count.

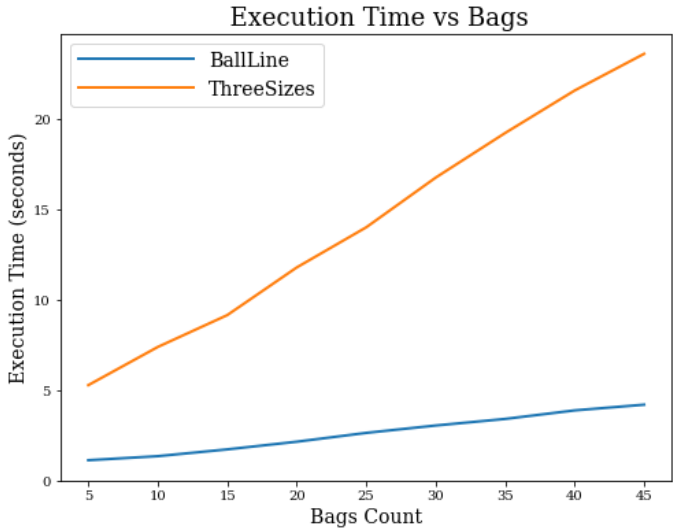


Figure 9.14: The impact on execution time when adjusting the bags count.

Figure 9.14 also shows that execution time has a almost linear relationship with the M parameter (0.99 Pearson coefficient). This makes it trivial to approximate the execution time once you have executed two runs with different values of M parameter for a given dataset.

Another finding from Figure 9.13 is that while more bags tends to increase clustering performance, the NMI tends to plateau. Comparing with literature, these results contradict [50] as in their bagging experiments they found that increasing the number of base partitioning sometimes decreased clustering performance in terms of NMI.

A final insight from this experiment is that a stopping condition could be introduced. The base clustering and consensus function could be executed in batches and once the clustering outcomes are not changing more than some threshold, the execution could be stopped, potentially saving computational resources.

9.9 IWSE mn-mx Parameter

Table 9.13: Overall clustering performance of each mn-mx pair.

	mn-mx	Range	Mean NMI	Standard Deviation NMI
0	10-20	10	0.803938	0.010669
7	80-90	10	0.822080	0.007340
8	10-30	20	0.829483	0.014354
14	70-90	20	0.832231	0.011013
20	60-90	30	0.837999	0.009589
32	30-90	60	0.844852	0.010259
25	50-90	40	0.844869	0.009746
6	70-80	10	0.848947	0.009808
35	10-90	80	0.849350	0.008215
31	20-80	60	0.851372	0.010901
13	60-80	20	0.851907	0.010792
19	50-80	30	0.853923	0.011257
34	20-90	70	0.855437	0.011224
29	40-90	50	0.859561	0.010915
28	30-80	50	0.866337	0.008847
33	10-80	70	0.866930	0.009026
5	60-70	10	0.868767	0.006549
15	10-40	30	0.869577	0.011954
12	50-70	20	0.873468	0.007117
24	40-80	40	0.874291	0.007715
18	40-70	30	0.879356	0.008483
23	30-70	40	0.882014	0.009516
17	30-60	30	0.887799	0.009619
11	40-60	20	0.888517	0.008541
27	20-70	50	0.889830	0.006979
30	10-70	60	0.889839	0.009955
1	20-30	10	0.890095	0.011182
21	10-50	40	0.890168	0.009519
22	20-60	40	0.891330	0.009611
4	50-60	10	0.894980	0.007636
26	10-60	50	0.901927	0.006742
16	20-50	30	0.902397	0.006865
10	30-50	20	0.908275	0.005600
9	20-40	20	0.909780	0.006387
3	40-50	10	0.917547	0.003953
2	30-40	10	0.917840	0.004104

9.10 Imbalanced MNIST Digits Clustering Results

9.10.1 Three principal components

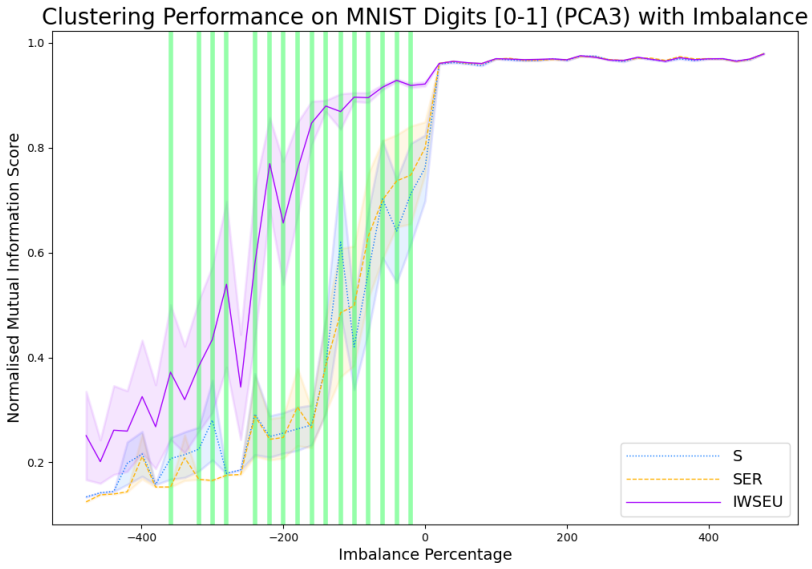


Figure 9.15: NMI score for S, SER, IWSEU on across an imbalance of -480% (more ones) to +480% (more zeros) of either digit, using 3 principal components.

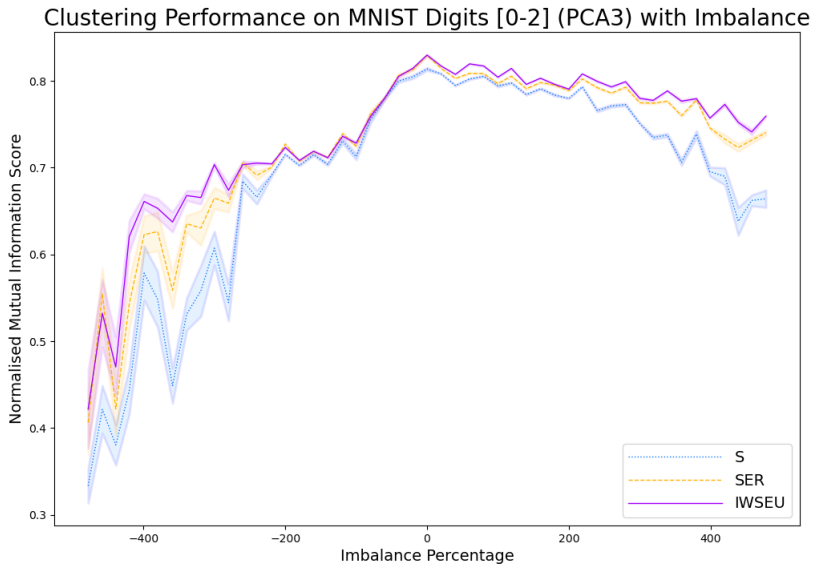


Figure 9.16: NMI score for S, SER, IWSEU on across an imbalance of -480% (more twos) to +480% (more zeros) of either digit, using 3 principal components.

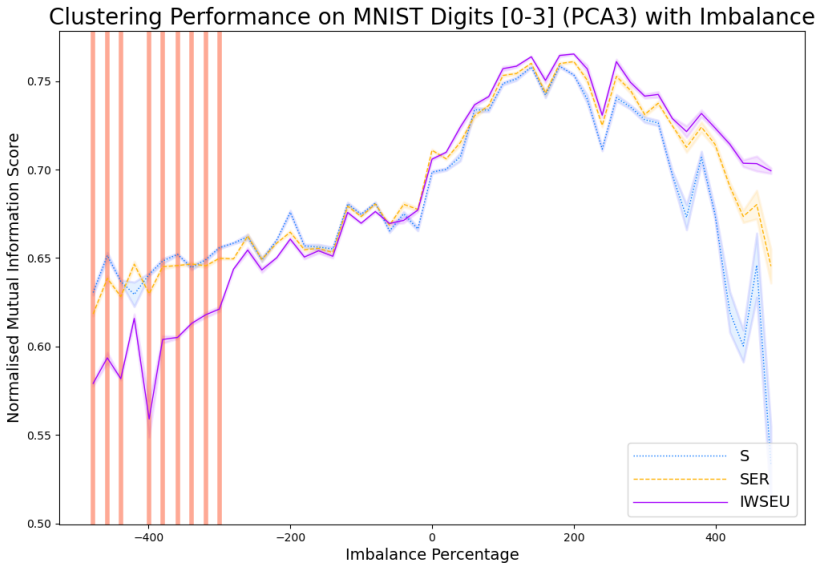


Figure 9.17: NMI score for S, SER, IWSEU on across an imbalance of -480% (more threes) to +480% (more zeros) of either digit, using 3 principsl components.

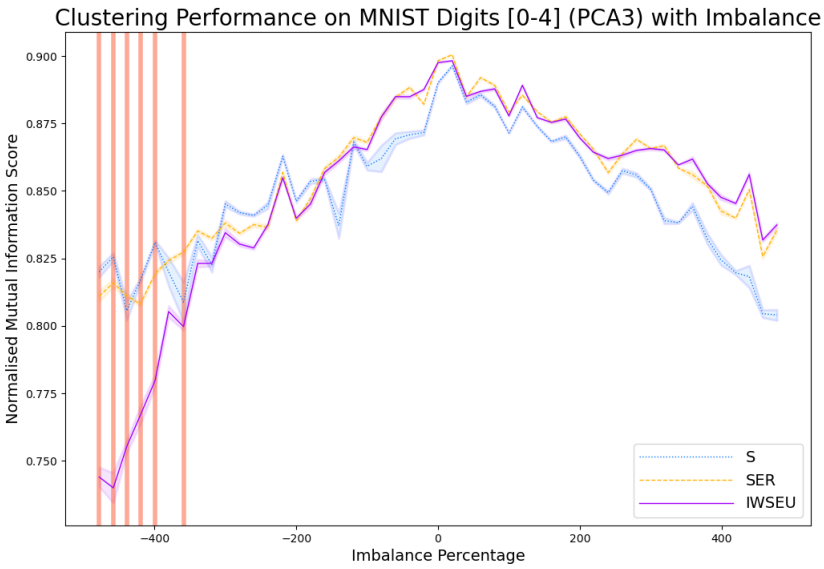


Figure 9.18: NMI score for S, SER, IWSEU on across an imbalance of -480% (more fours) to +480% (more zeros) of either digit, using 3 principal components.

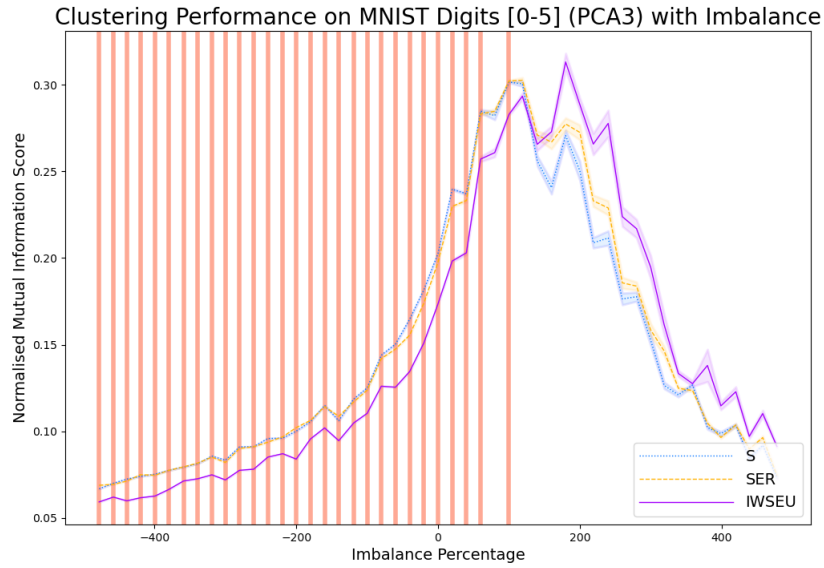


Figure 9.19: NMI score for S, SER, IWSEU on across an imbalance of -480% (more fives) to +480% (more zeros) of either digit, using 3 principal components.

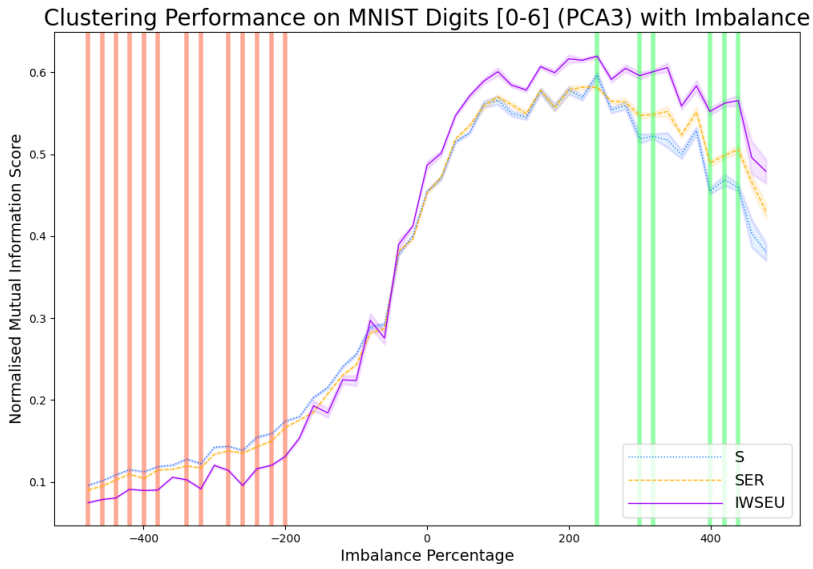


Figure 9.20: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sixes) to +480% (more zeros) of either digit, using 3 principal components.

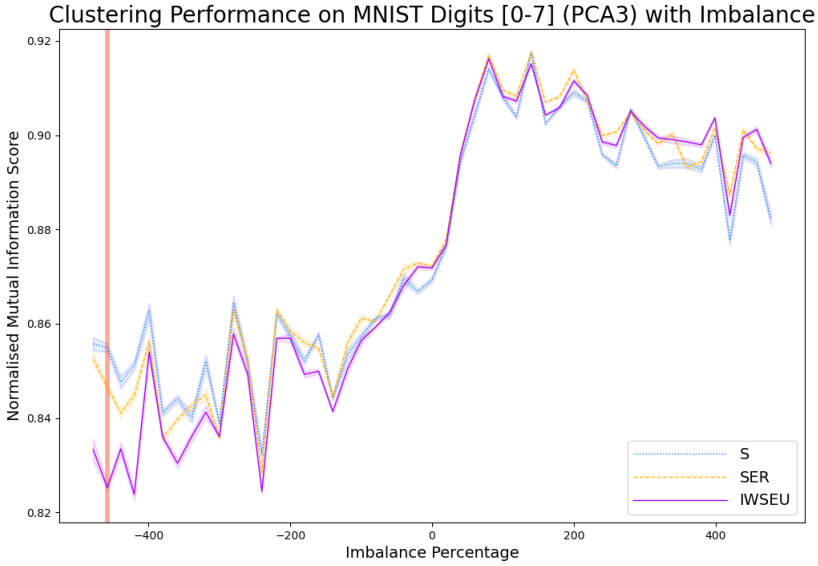


Figure 9.21: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sevens) to +480% (more zeros) of either digit, using 3 principal components.

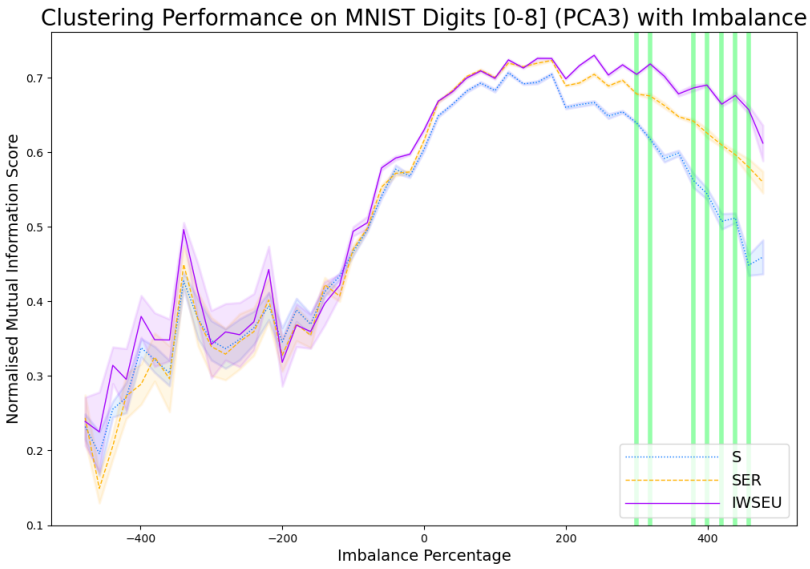


Figure 9.22: NMI score for S, SER, IWSEU on across an imbalance of -480% (more eights) to +480% (more zeros) of either digit, using 3 principal components.

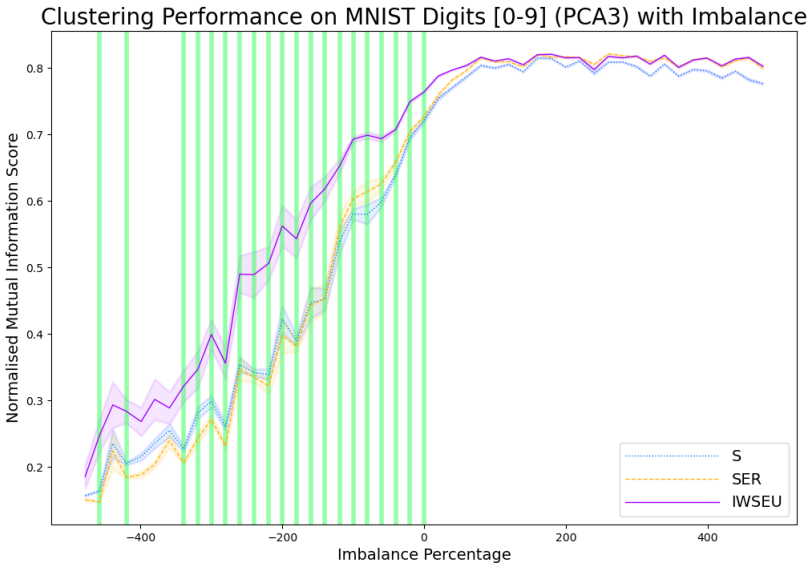


Figure 9.23: NMI score for S, SER, IWSEU on across an imbalance of -480% (more nines) to +480% (more zeros) of either digit, using 3 principal components.

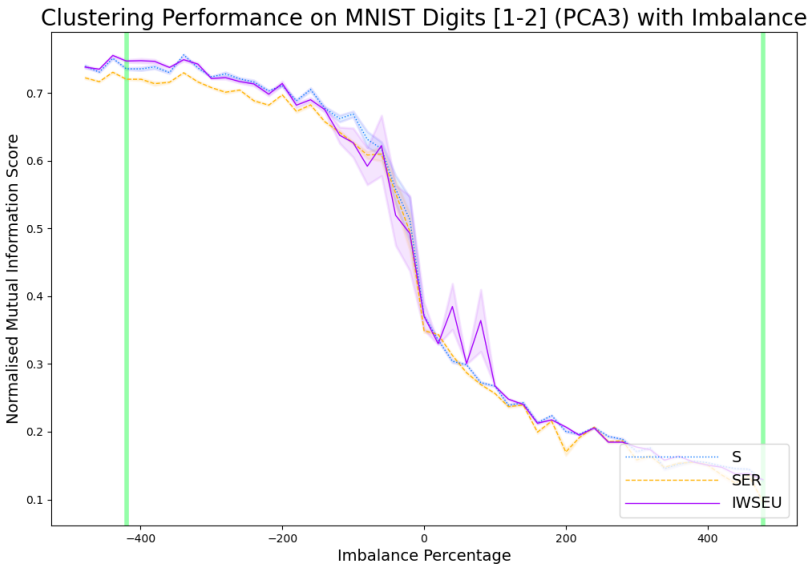


Figure 9.24: NMI score for S, SER, IWSEU on across an imbalance of -480% (more twos) to +480% (more ones) of either digit, using 3 principal components.

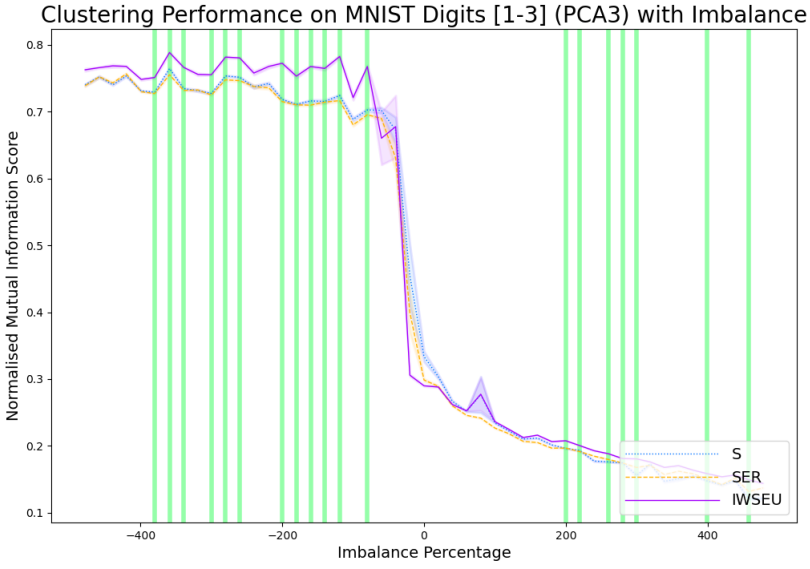


Figure 9.25: NMI score for S, SER, IWSEU on across an imbalance of -480% (more threes) to +480% (more ones) of either digit, using 3 principal components.

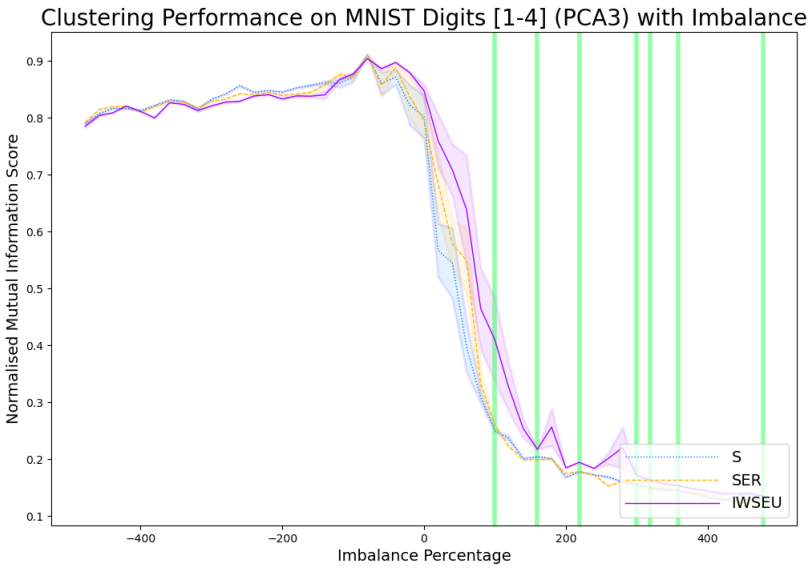


Figure 9.26: NMI score for S, SER, IWSEU on across an imbalance of -480% (more fours) to +480% (more ones) of either digit, using 3 principal components.

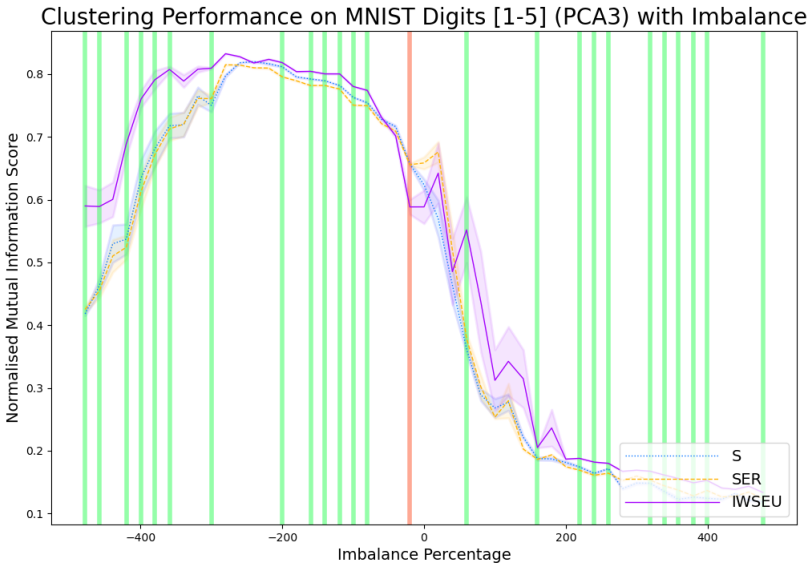


Figure 9.27: NMI score for S, SER, IWSEU on across an imbalance of -480% (more fives) to +480% (more ones) of either digit, using 3 principal components.

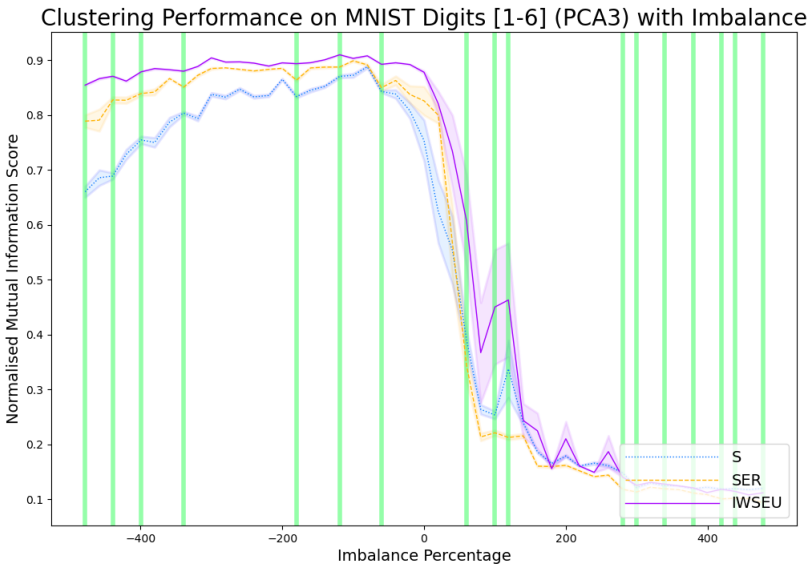


Figure 9.28: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sixes) to +480% (more ones) of either digit, using 3 principal components.

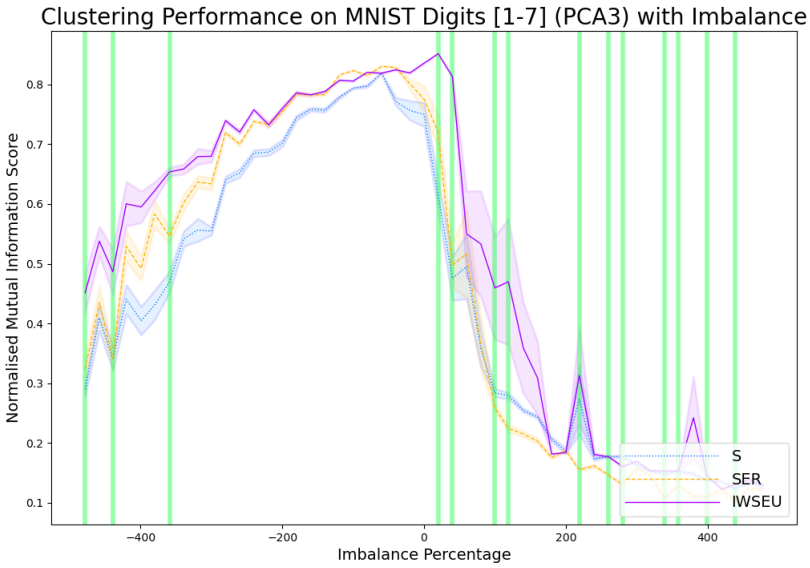


Figure 9.29: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sevens) to +480% (more ones) of either digit, using 3 principal components.

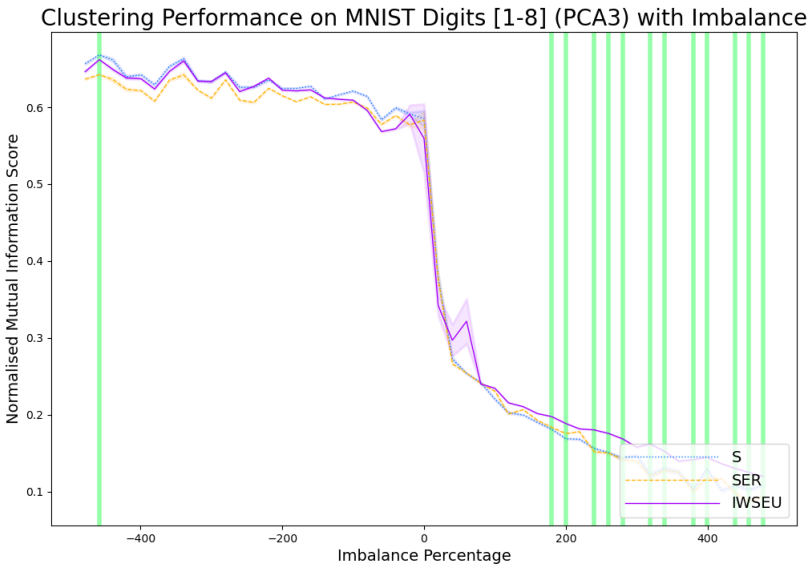


Figure 9.30: NMI score for S, SER, IWSEU on across an imbalance of -480% (more eights) to +480% (more ones) of either digit, using 3 principal components.

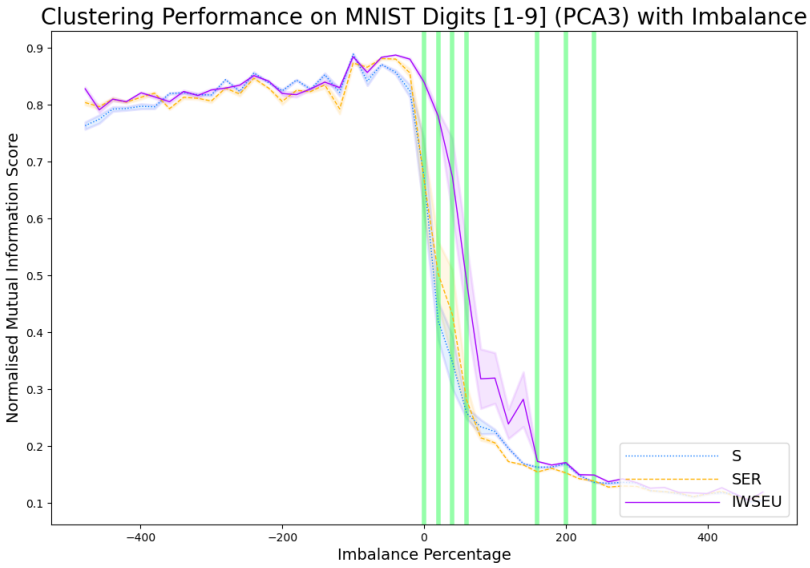


Figure 9.31: NMI score for S, SER, IWSEU on across an imbalance of -480% (more nines) to +480% (more ones) of either digit, using 3 principal components.

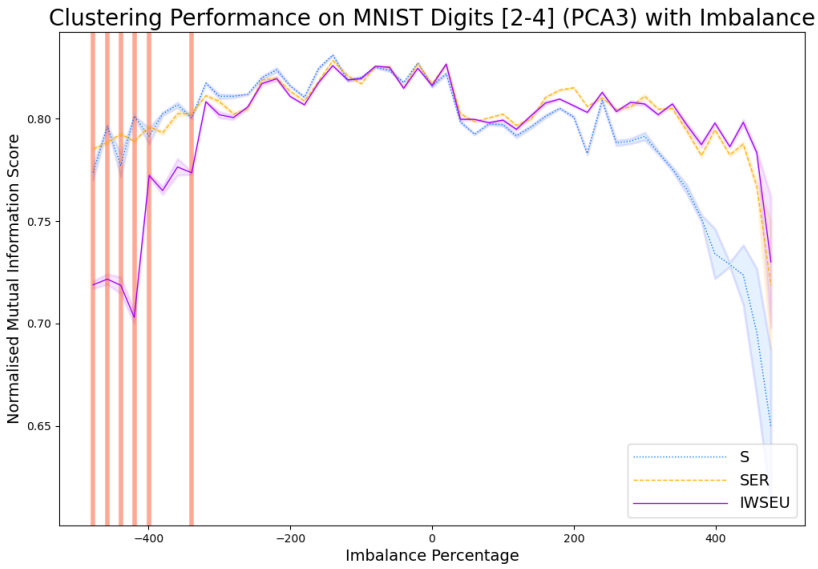


Figure 9.32: NMI score for S, SER, IWSEU on across an imbalance of -480% (more fours) to +480% (more twos) of either digit, using 3 principal components.

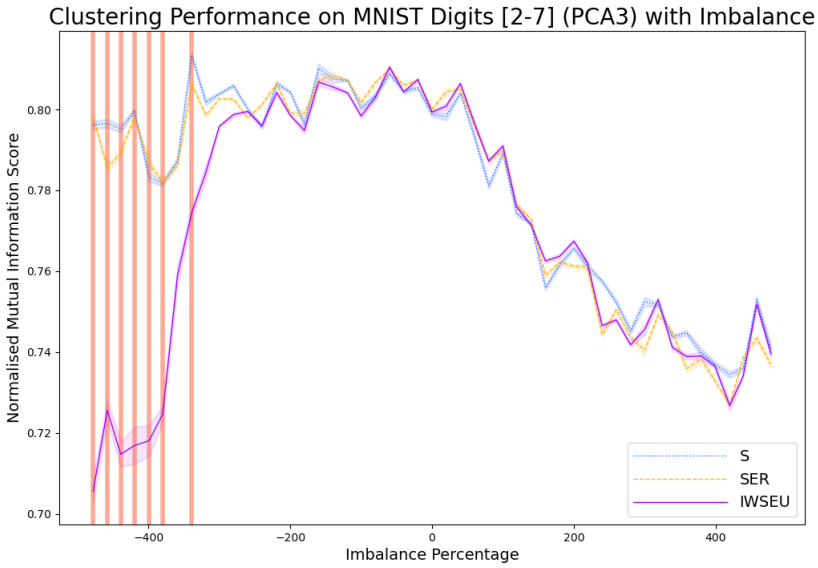


Figure 9.33: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sevens) to +480% (more twos) of either digit, using 3 principal components.

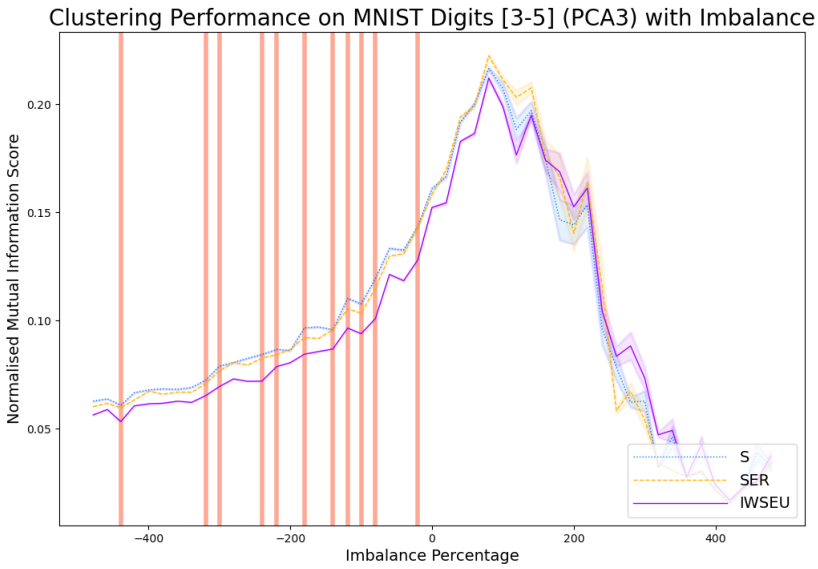


Figure 9.34: NMI score for S, SER, IWSEU on across an imbalance of -480% (more fives) to +480% (more threes) of either digit, using 3 principal components.

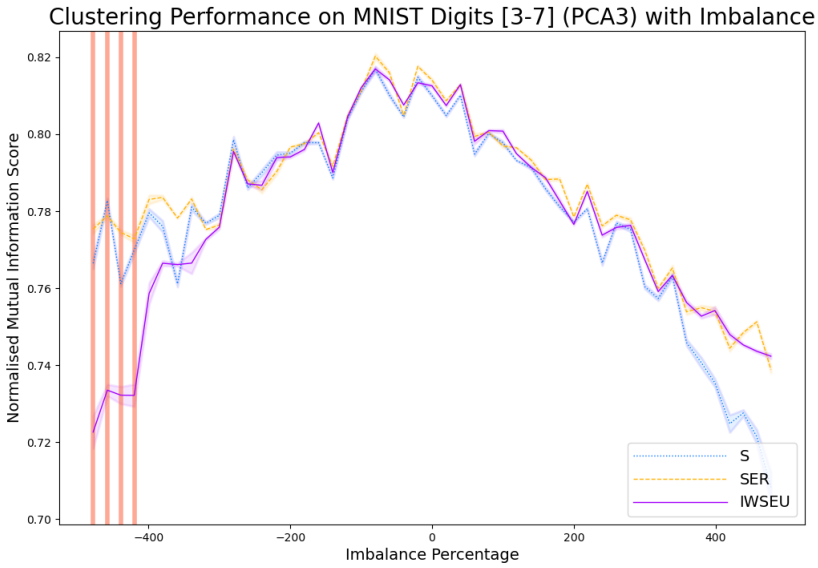


Figure 9.35: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sevens) to +480% (more threes) of either digit, using 3 principal components.

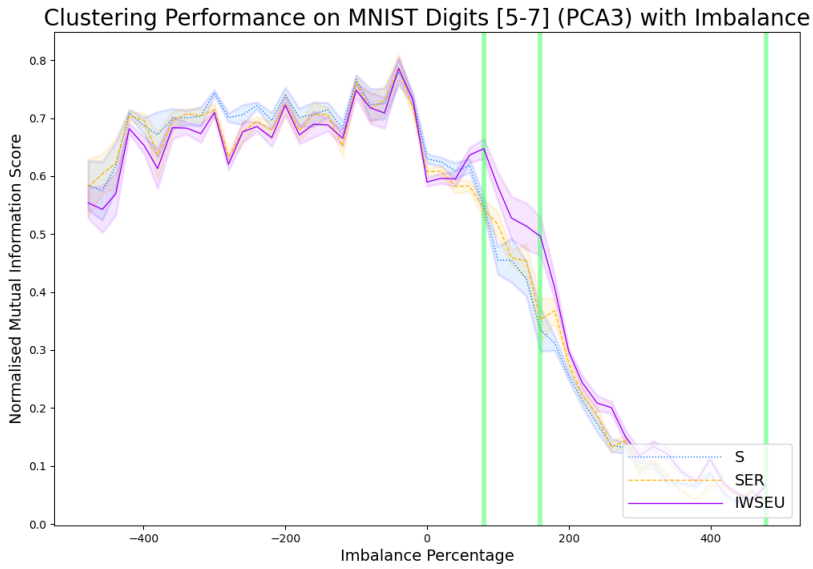


Figure 9.36: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sevens) to +480% (more fives) of either digit, using 3 principal components.

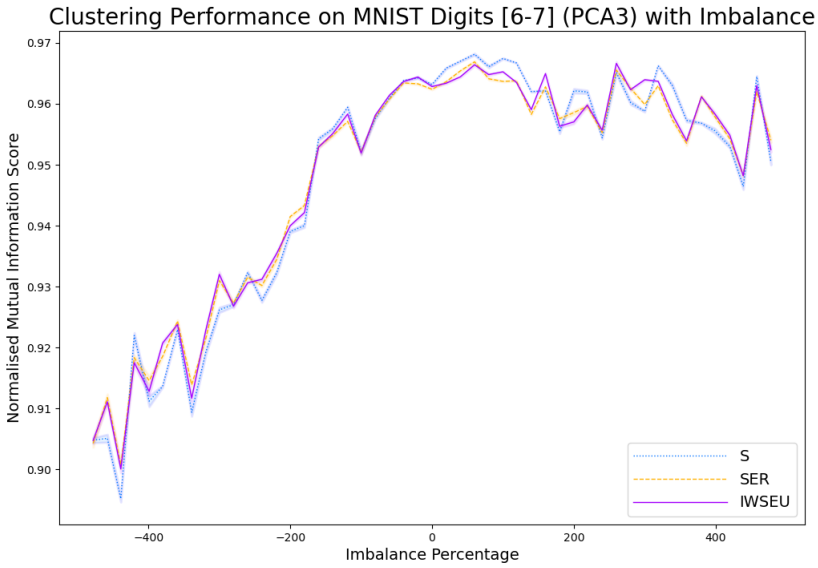


Figure 9.37: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sevens) to +480% (more sixes) of either digit, using 3 principal components.

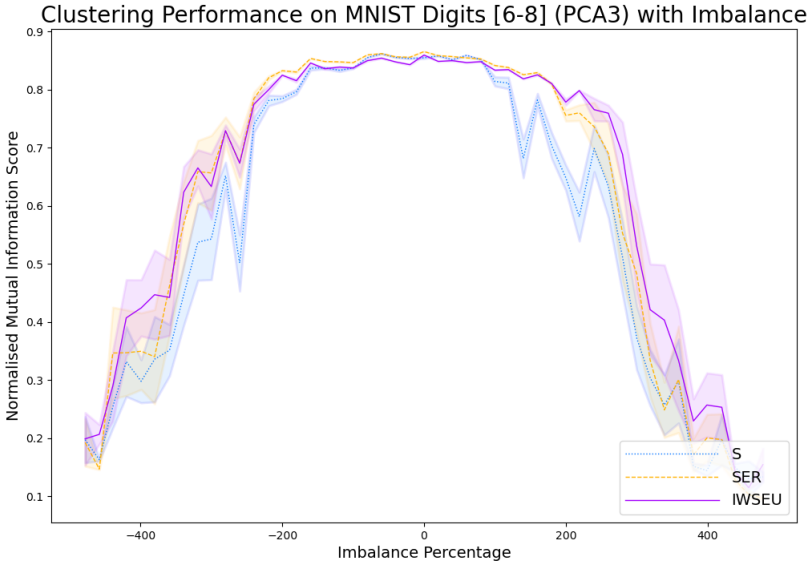


Figure 9.38: NMI score for S, SER, IWSEU on across an imbalance of -480% (more eights) to +480% (more sixes) of either digit, using 3 principal components.

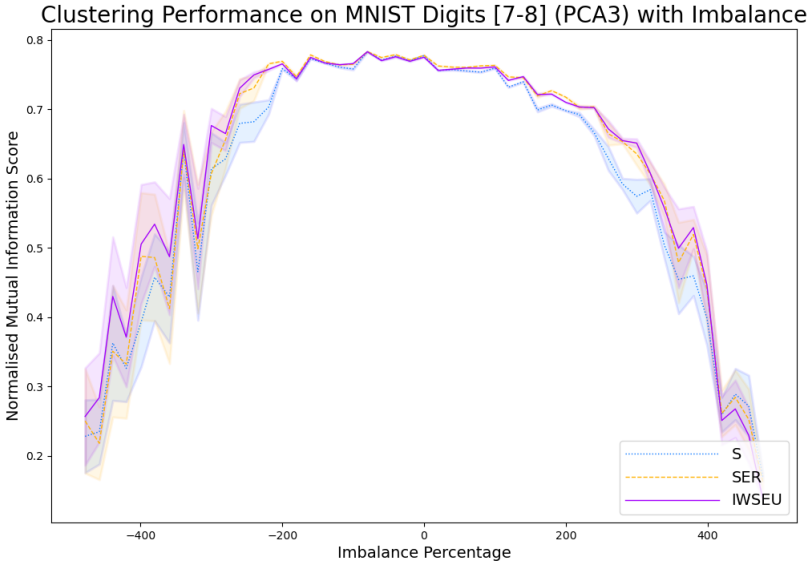


Figure 9.39: NMI score for S, SER, IWSEU on across an imbalance of -480% (more eights) to +480% (more sevens) of either digit, using 3 principal components.

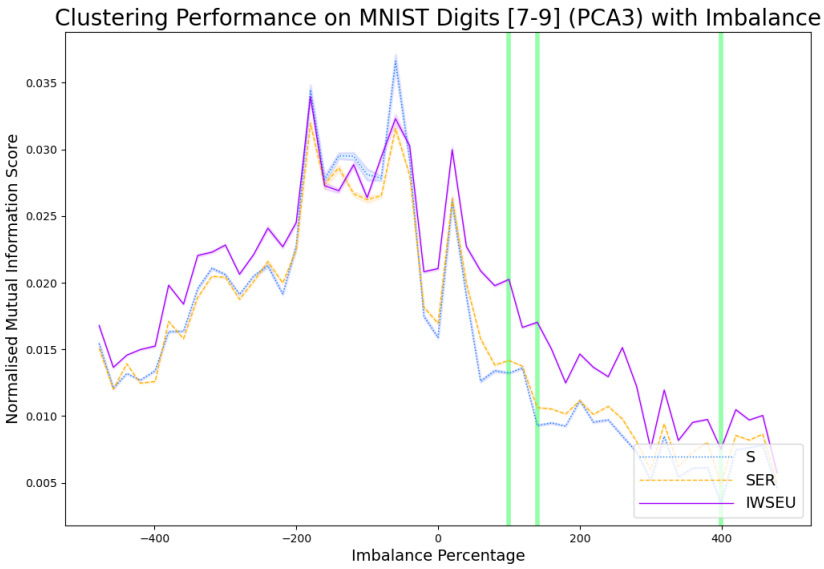


Figure 9.40: NMI score for S, SER, IWSEU on across an imbalance of -480% (more nines) to +480% (more sevens) of either digit, using .

9.10.2 Six principal components

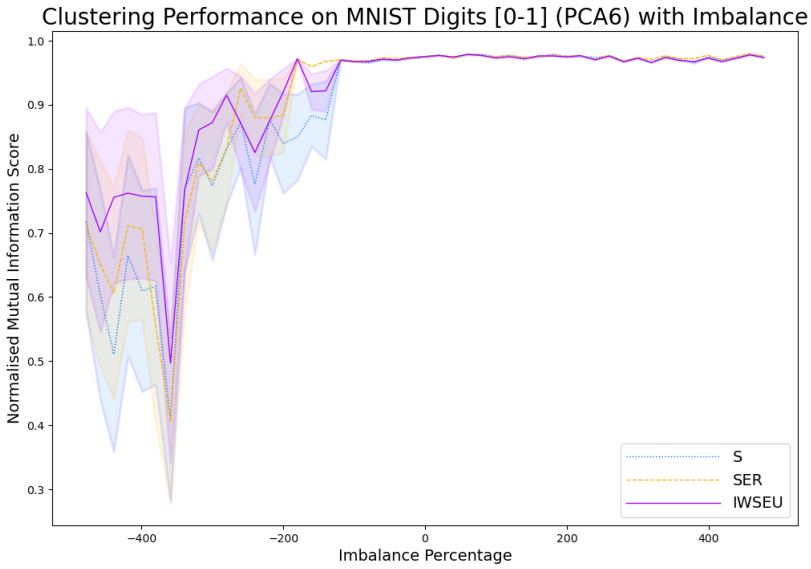


Figure 9.41: NMI score for S, SER, IWSEU on across an imbalance of -480% (more ones) to +480% (more zeros) of either digit, using 6 principal components.

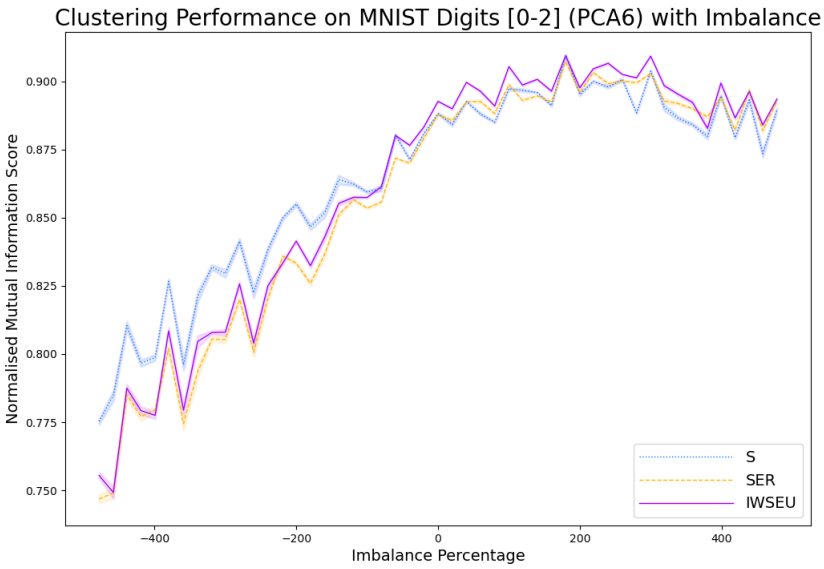


Figure 9.42: NMI score for S, SER, IWSEU on across an imbalance of -480% (more twos) to +480% (more zeros) of either digit, using 6 principal components.

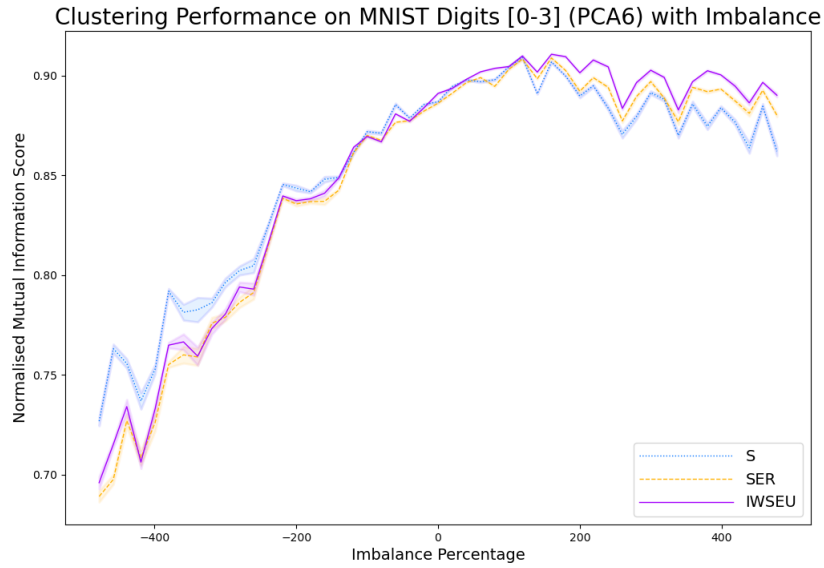


Figure 9.43: NMI score for S, SER, IWSEU on across an imbalance of -480% (more threes) to +480% (more zeros) of either digit, using 6 principal components.

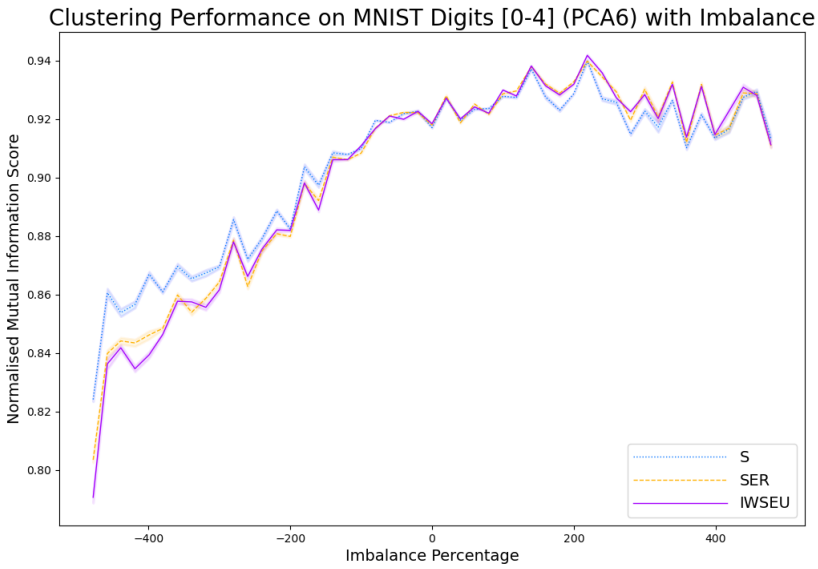


Figure 9.44: NMI score for S, SER, IWSEU on across an imbalance of -480% (more fours) to +480% (more zeros) of either digit, using 6 principal components.

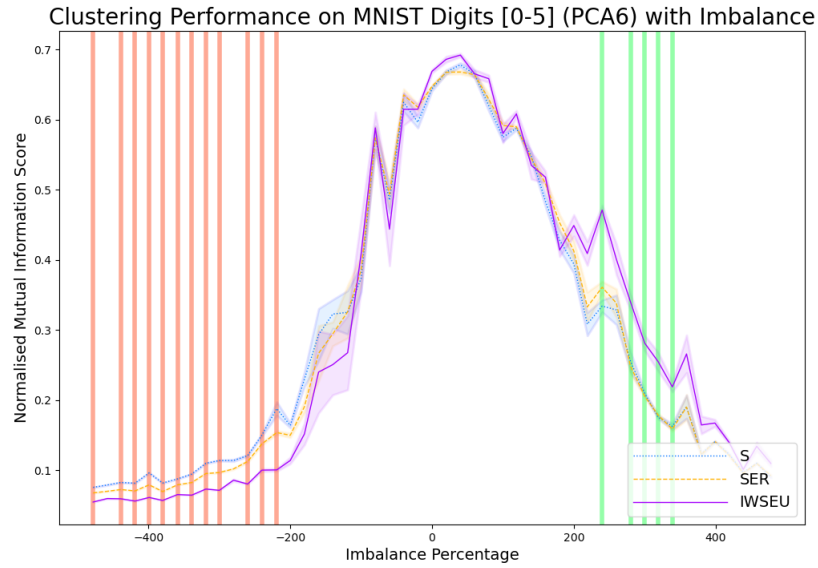


Figure 9.45: NMI score for S, SER, IWSEU on across an imbalance of -480% (more fives) to +480% (more zeros) of either digit, using 6 principal components.

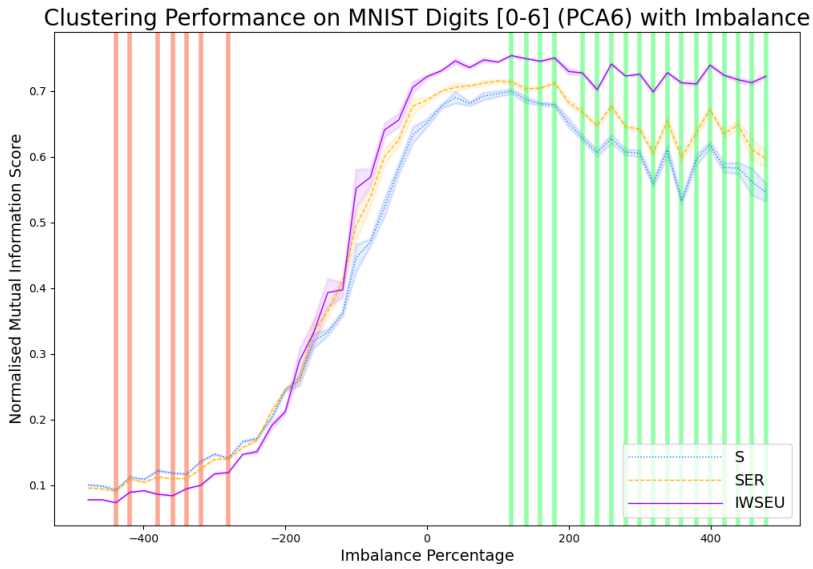


Figure 9.46: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sixes) to +480% (more zeros) of either digit, using 6 principal components.

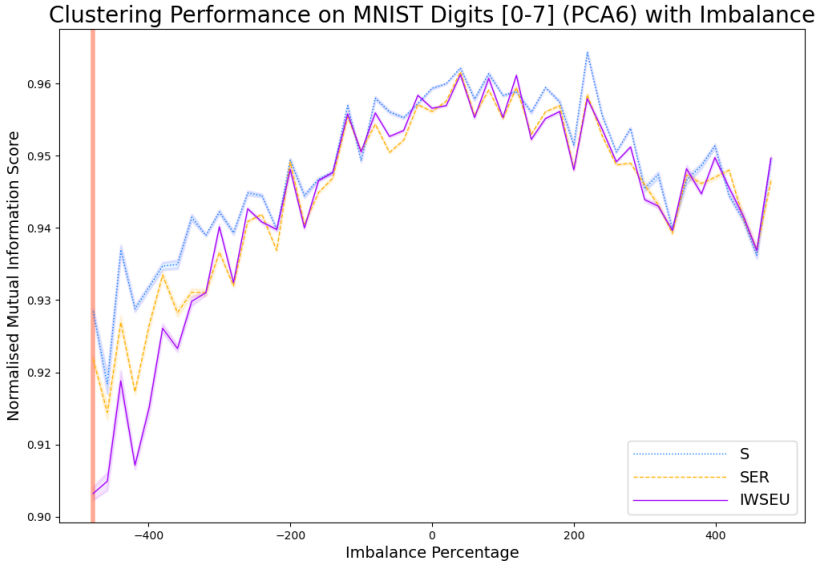


Figure 9.47: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sevens) to +480% (more zeros) of either digit, using 6 principal components.

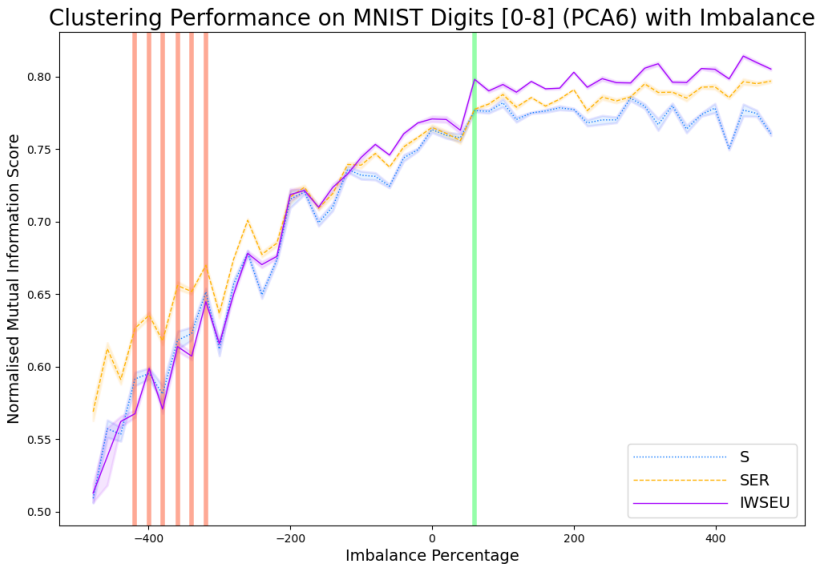


Figure 9.48: NMI score for S, SER, IWSEU on across an imbalance of -480% (more eights) to +480% (more zeros) of either digit, using 6 principal components.

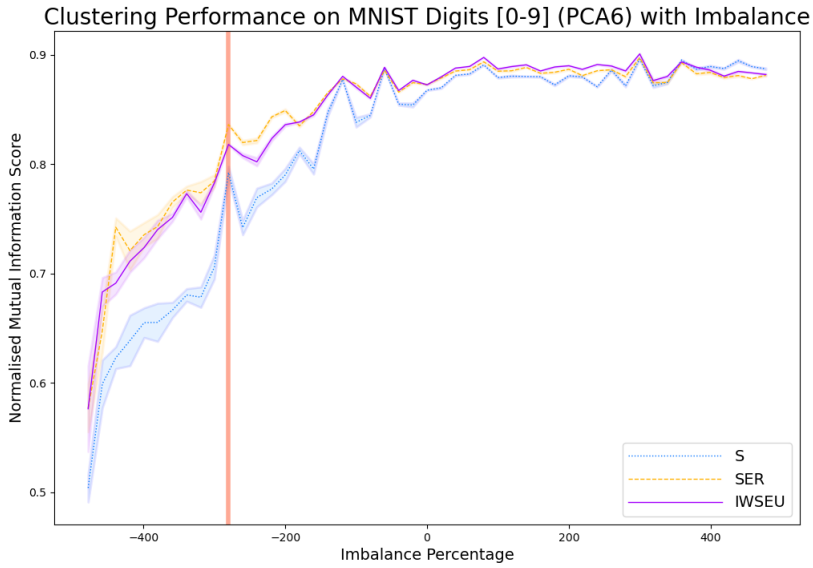


Figure 9.49: NMI score for S, SER, IWSEU on across an imbalance of -480% (more nines) to +480% (more zeros) of either digit, using 6 principal components.

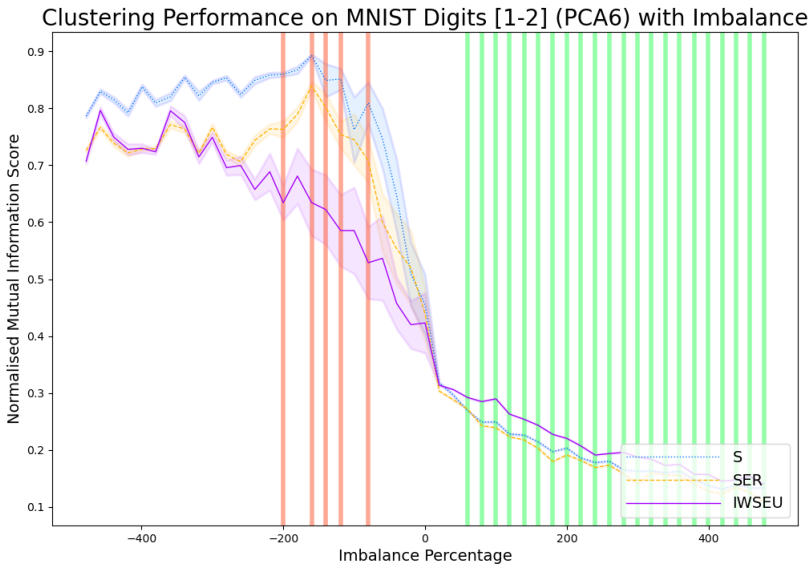


Figure 9.50: NMI score for S, SER, IWSEU on across an imbalance of -480% (more twos) to +480% (more ones) of either digit, using 6 principal components.

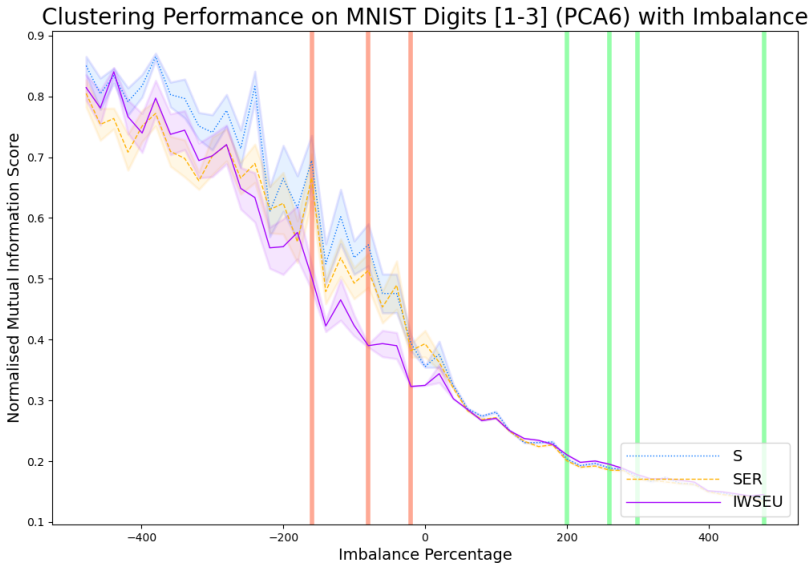


Figure 9.51: NMI score for S, SER, IWSEU on across an imbalance of -480% (more threes) to +480% (more ones) of either digit, using 6 principal components.

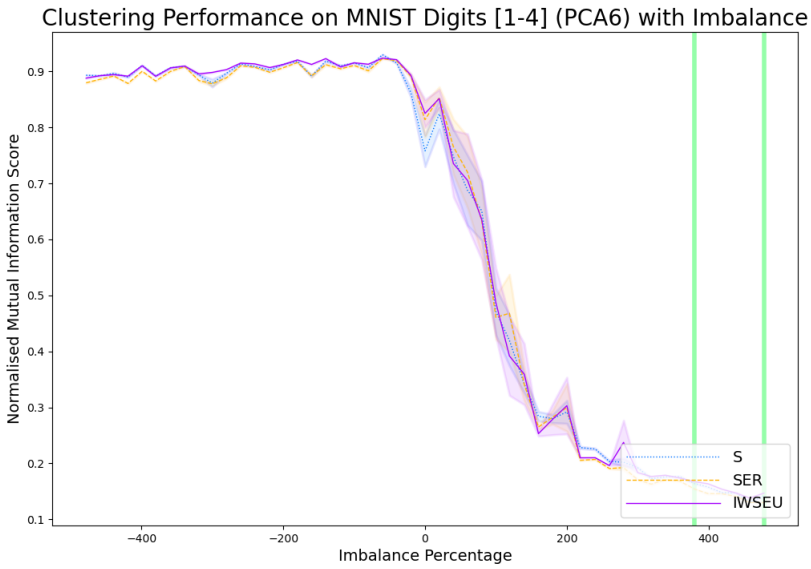


Figure 9.52: NMI score for S, SER, IWSEU on across an imbalance of -480% (more fours) to +480% (more ones) of either digit, using 6 principal components.

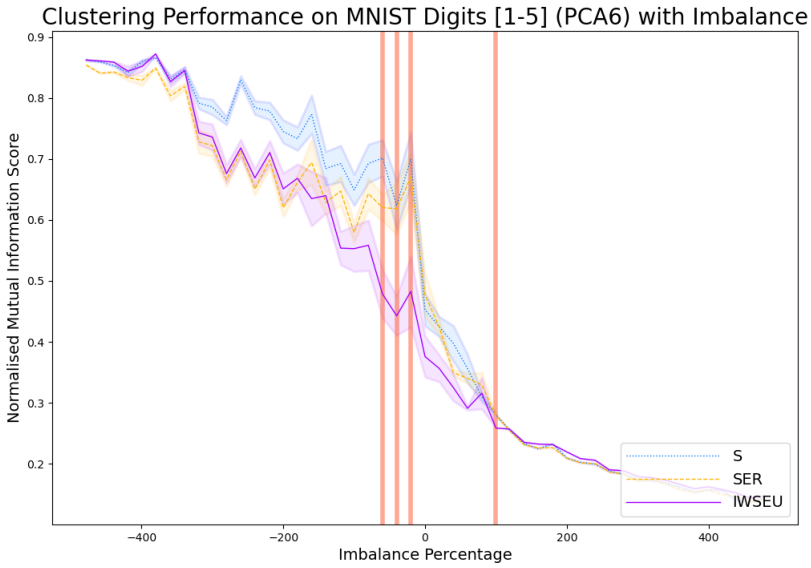


Figure 9.53: NMI score for S, SER, IWSEU on across an imbalance of -480% (more fives) to +480% (more ones) of either digit, using 6 principal components.

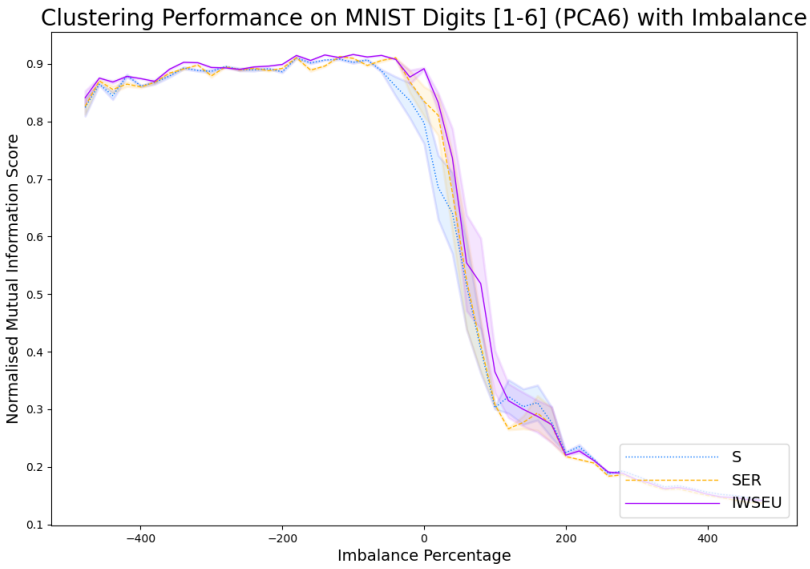


Figure 9.54: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sixes) to +480% (more ones) of either digit, using 6 principal components.

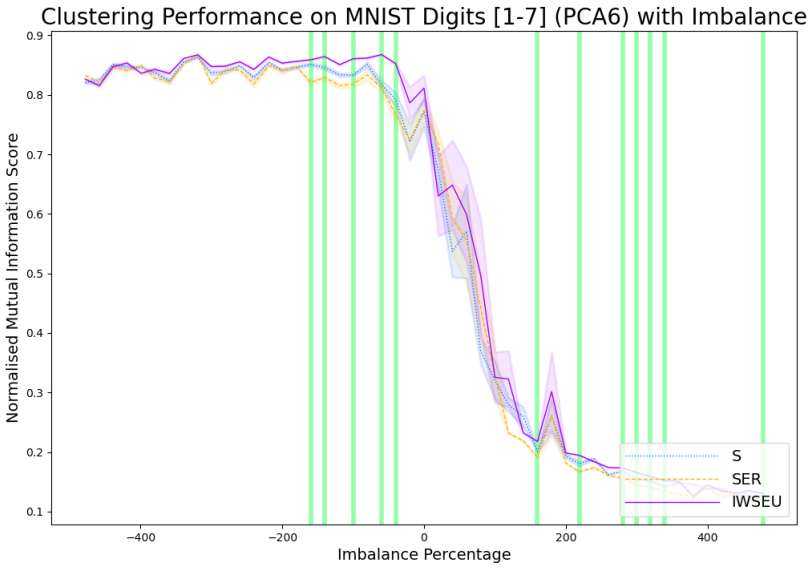


Figure 9.55: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sevens) to +480% (more ones) of either digit, using 6 principal components.

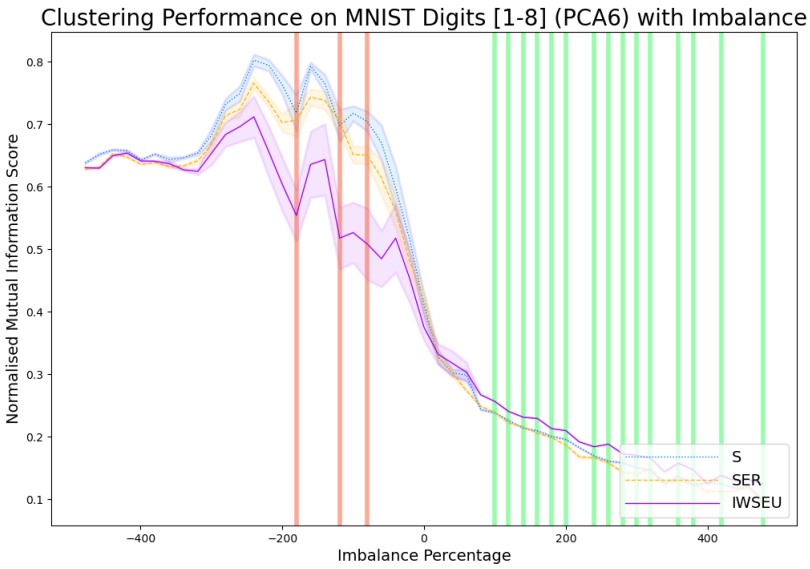


Figure 9.56: NMI score for S, SER, IWSEU on across an imbalance of -480% (more eights) to +480% (more ones) of either digit, using 6 principal components.

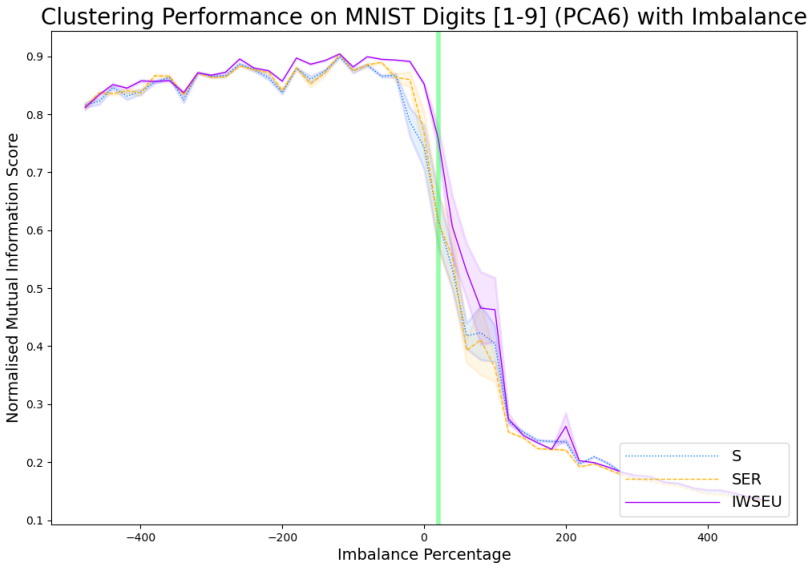


Figure 9.57: NMI score for S, SER, IWSEU on across an imbalance of -480% (more nines) to +480% (more ones) of either digit, using 6 principal components.

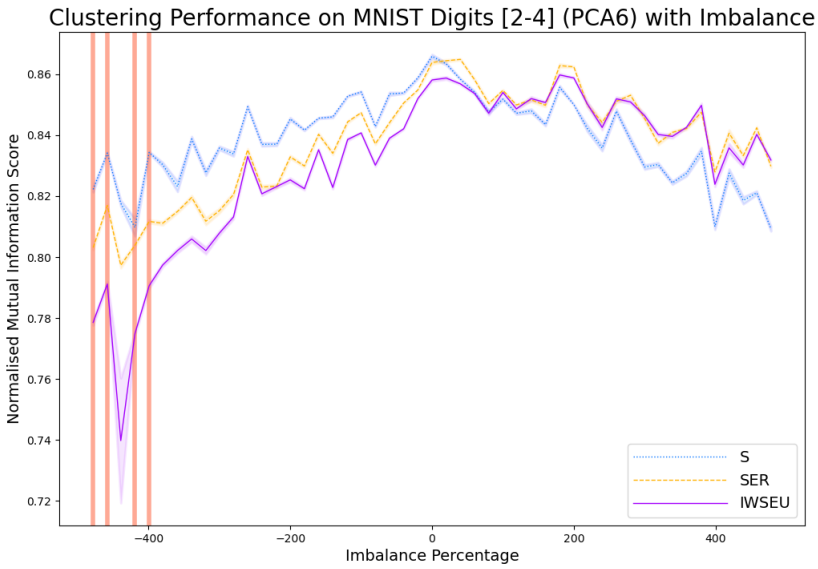


Figure 9.58: NMI score for S, SER, IWSEU on across an imbalance of -480% (more fours) to +480% (more twos) of either digit, using 6 principal components.

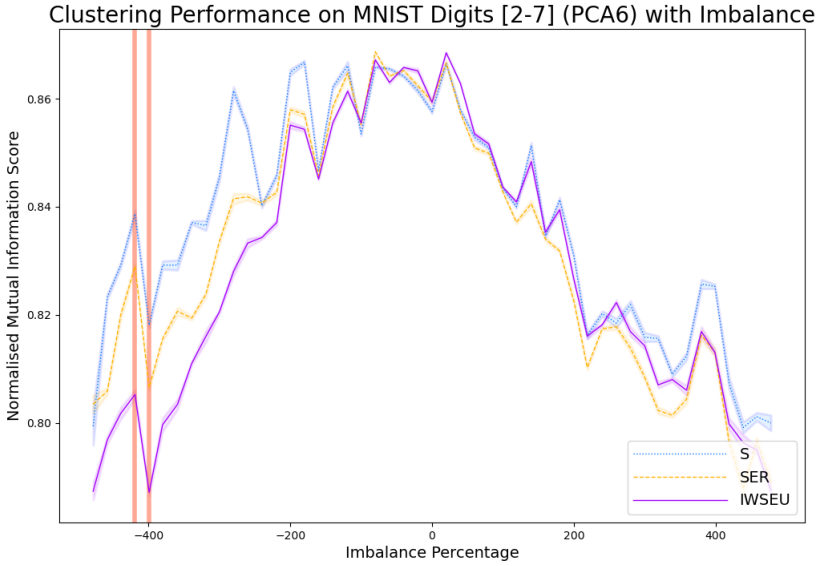


Figure 9.59: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sevens) to +480% (more twos) of either digit, using 6 principal components.

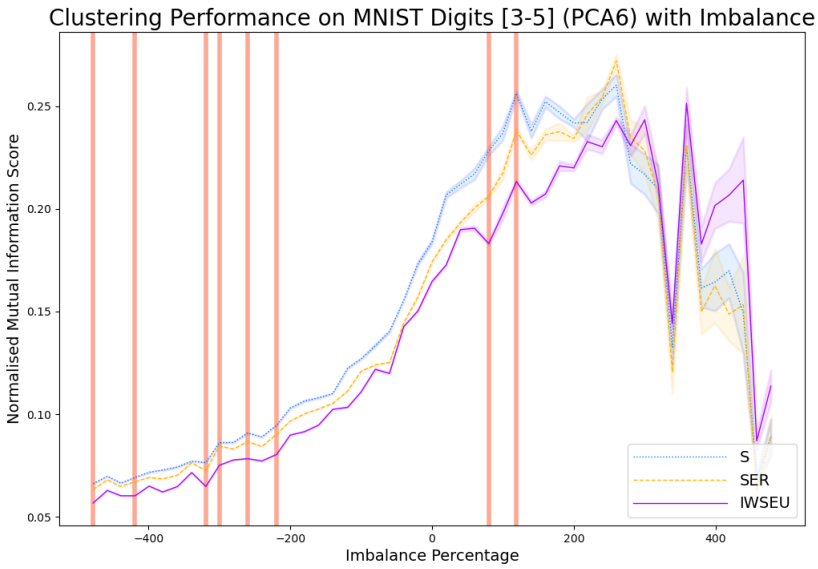


Figure 9.60: NMI score for S, SER, IWSEU on across an imbalance of -480% (more fives) to +480% (more threes) of either digit, using 6 principal components.

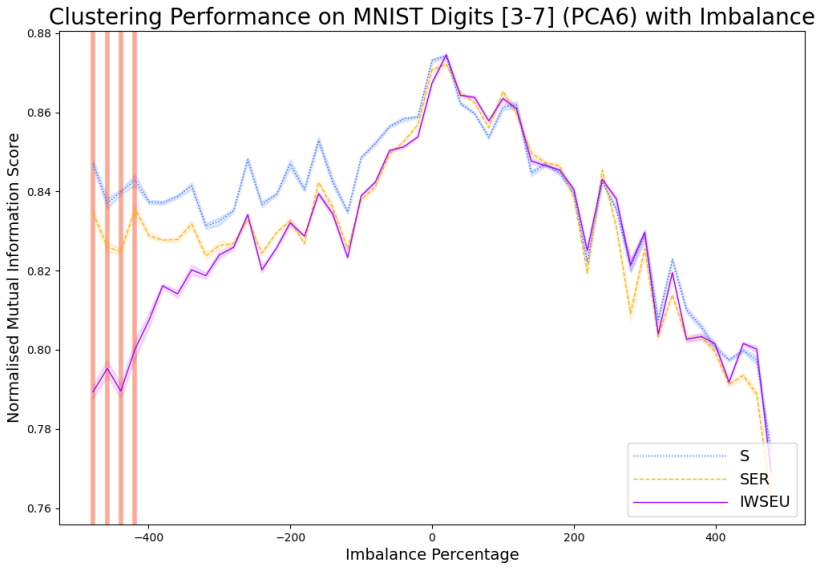


Figure 9.61: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sevens) to +480% (more threes) of either digit, using 6 principal components.

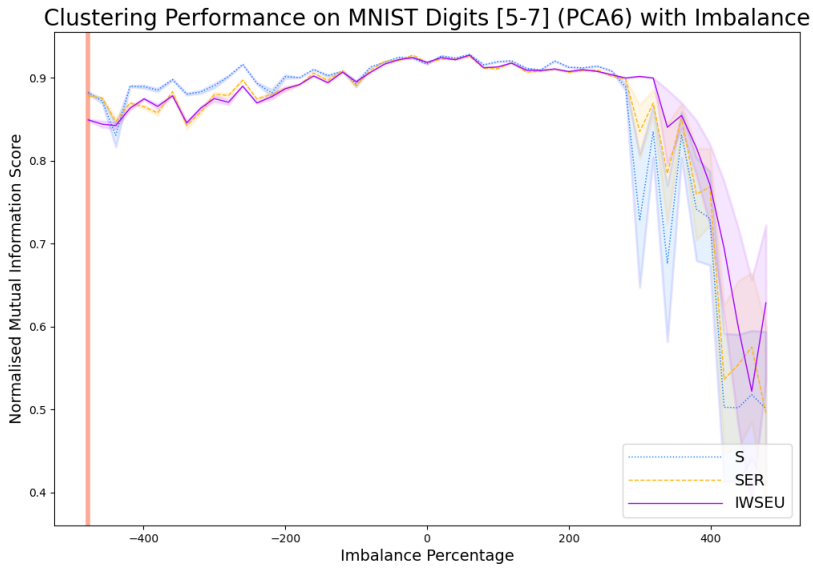


Figure 9.62: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sevens) to +480% (more fives) of either digit, using 6 principal components.

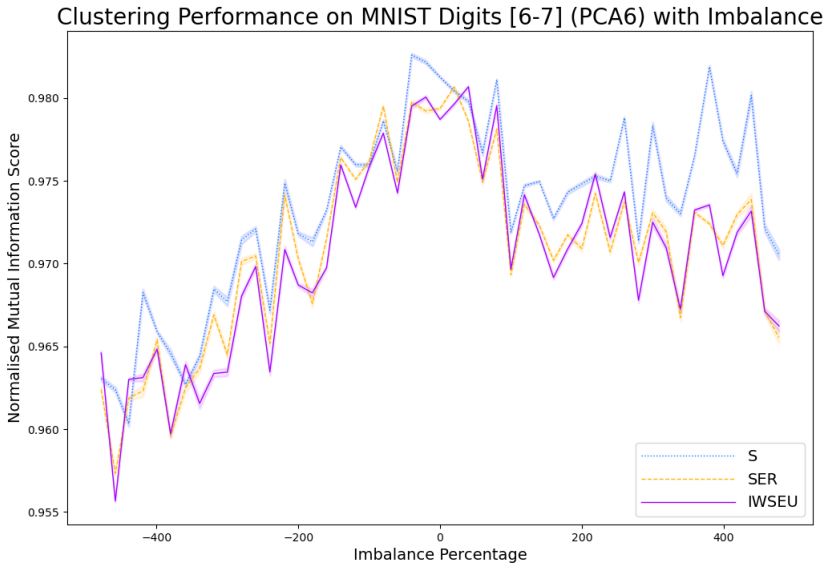


Figure 9.63: NMI score for S, SER, IWSEU on across an imbalance of -480% (more sevens) to +480% (more sixes) of either digit, using 6 principal components.

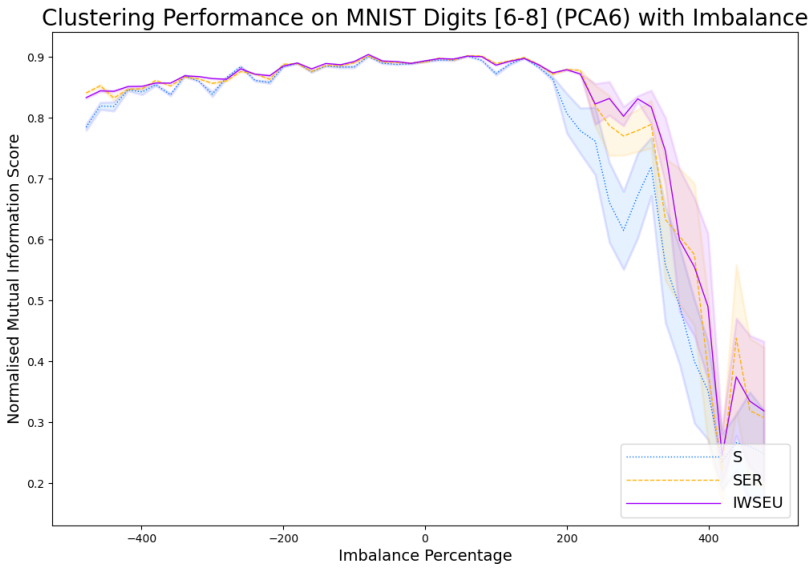


Figure 9.64: NMI score for S, SER, IWSEU on across an imbalance of -480% (more eights) to +480% (more sixes) of either digit, using 6 principal components.

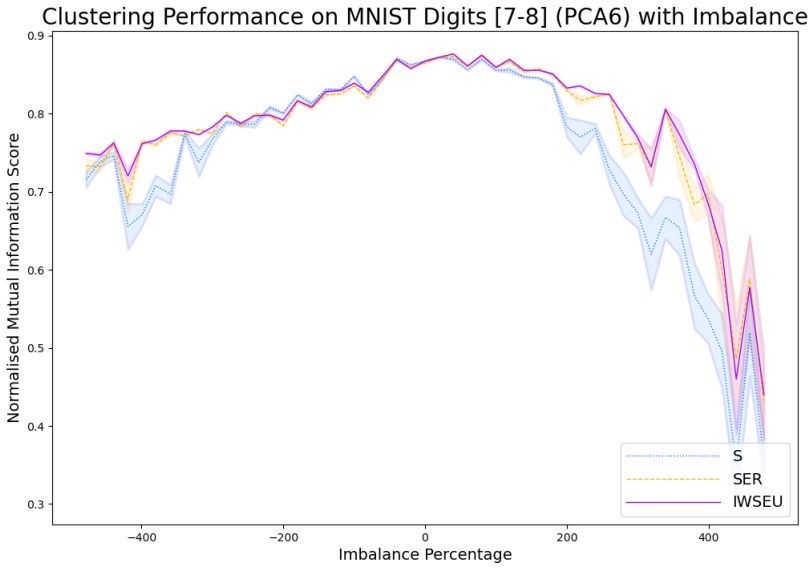


Figure 9.65: NMI score for S, SER, IWSEU on across an imbalance of -480% (more eights) to +480% (more sevens) of either digit, using 6 principal components.

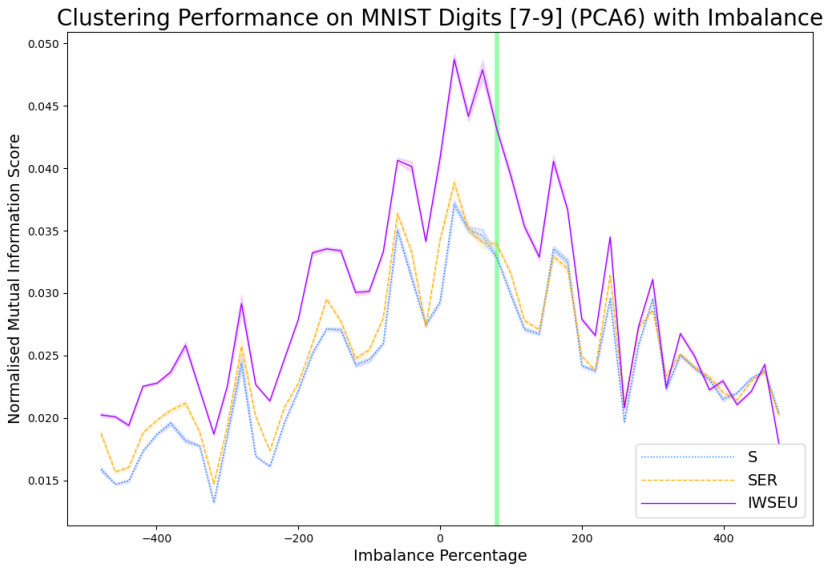


Figure 9.66: NMI score for S, SER, IWSEU on across an imbalance of -480% (more nines) to +480% (more sevens) of either digit, using 6 principal components.

9.11 Clustering Tool

A small graphical application was created using Java with only core libraries. This application was for visualising the clustering problems and instance weighting solutions. The list of features includes:

- Loading datasets, ARFF, CSV
- Importing images as datasets.
- Generating datasets from Gaussian distributions and uniform randomness.
- Instance weighting method based on k-nearest neighbours.
- Instance weighting method based on range-nearest neighbours.
- Instance weighting method based frequency count from equal-width bins (N-d histogram).
- K-means clustering (animated)

- Visualising datasets as a 2-d scatter plot.
- Visualising datasets as a N-d parallel coordinates plot.
- Visualising instance weighting as a graduation colour.
- Calculating intrinsic metrics: Silhouette Coe., B-Cubed Precision

Figures 9.67 to 9.74 show some screenshots of the application.

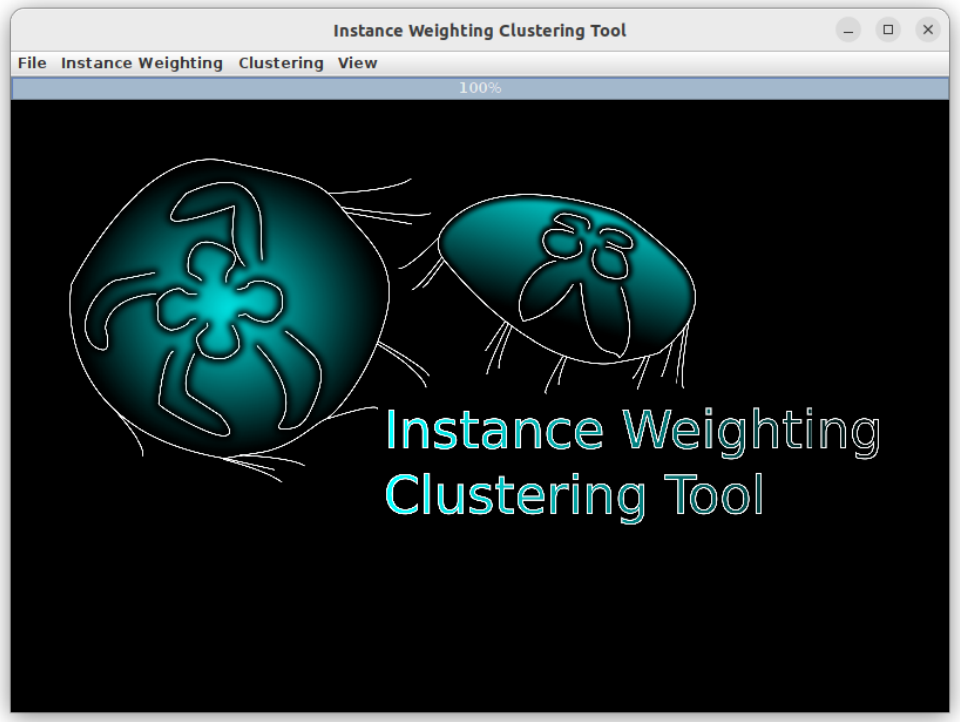


Figure 9.67: Start screen.

Ultimately, development was abandoned the in favour of the using the rapidly maturing Python data science libraries at the time of writing. This decision was made as implementing everything from scratch and adequately testing it was unnecessarily time-consuming and risk-prone.

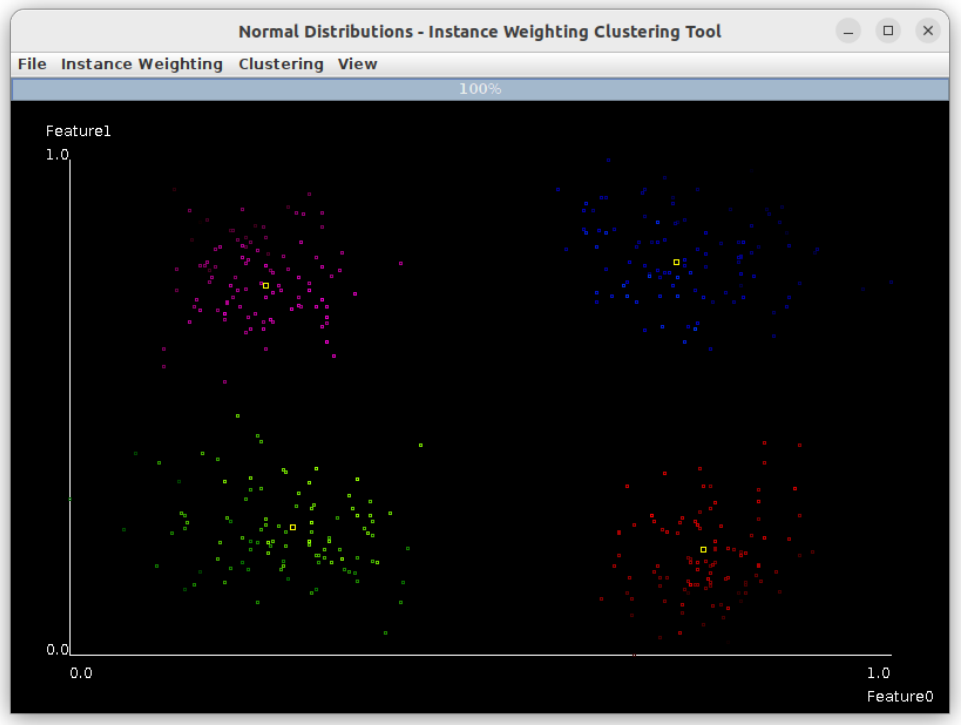


Figure 9.68: A dataset generated from four normal distributions. Instance weighting (knn with $k*$=30) and k-means ($k = 4$) has been applied.

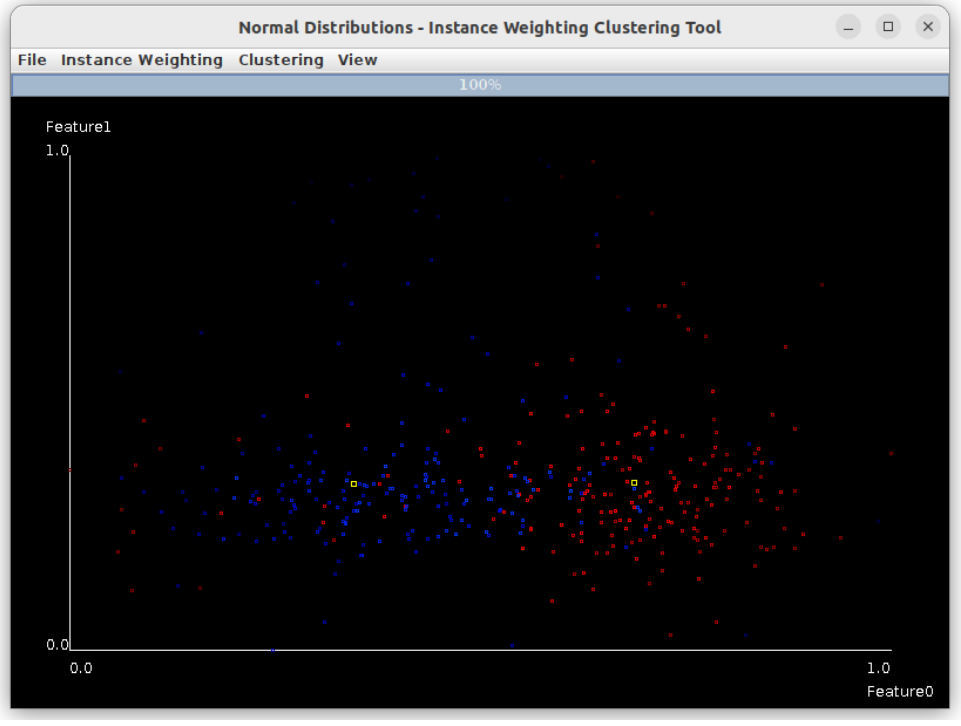


Figure 9.69: A dataset generated from two skewed normal distributions. Instance weighting (knn with $k*$=10) and k-means ($k = 2$) has been applied.

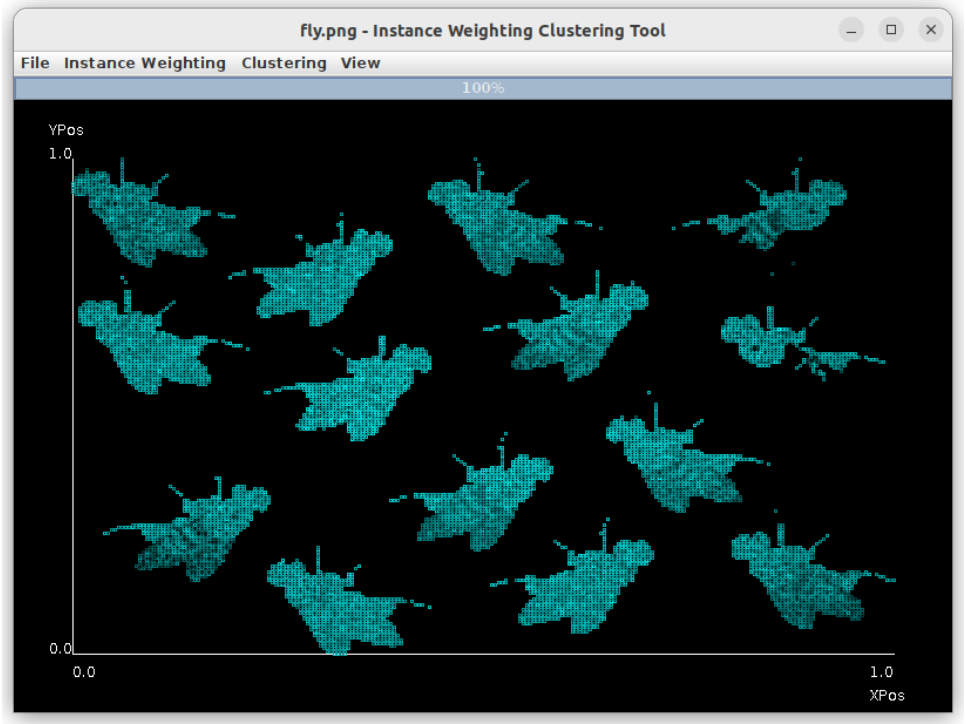


Figure 9.70: A dataset created by importing an image. Instance weighting (knn with $k*$=5) has been applied.

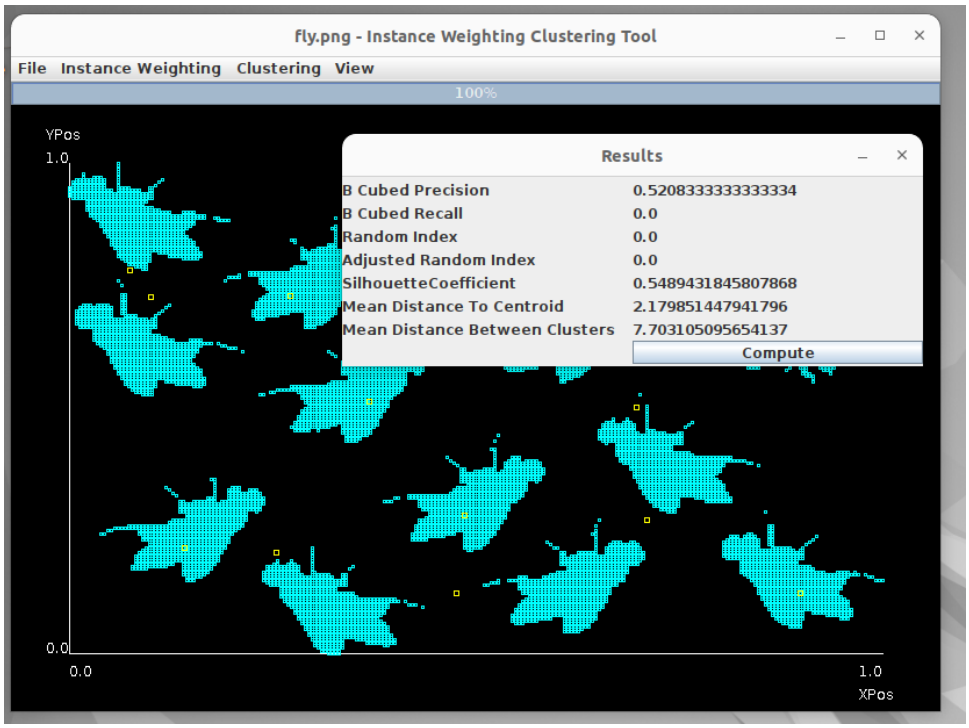


Figure 9.71: A dataset created by importing an image. k-means ($k = 14$) has been applied and the results dialogue is shown (partially implemented metrics).

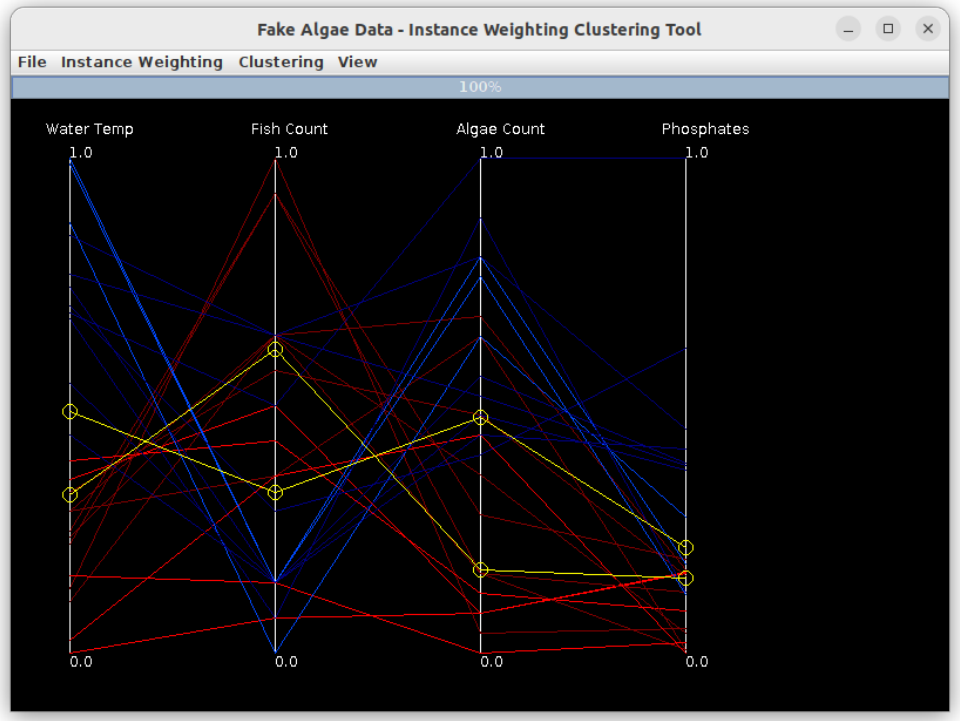


Figure 9.72: A fictitious dataset about algae visualised using the parallel coordinates plot. Instance weighting (histogram-based with bins=5) and k-means ($k = 2$) has been applied.

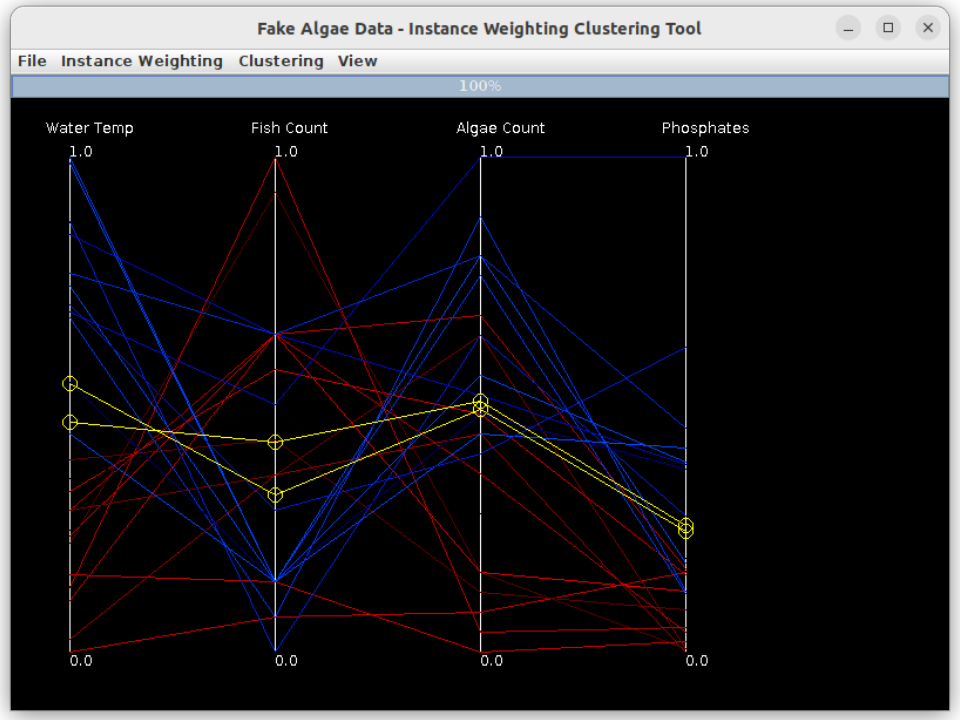


Figure 9.73: A fictitious dataset about algae visualised using the parallel coordinates plot. Instance weighting (knn with k^*=5) and k-means ($k = 2$) has been applied.

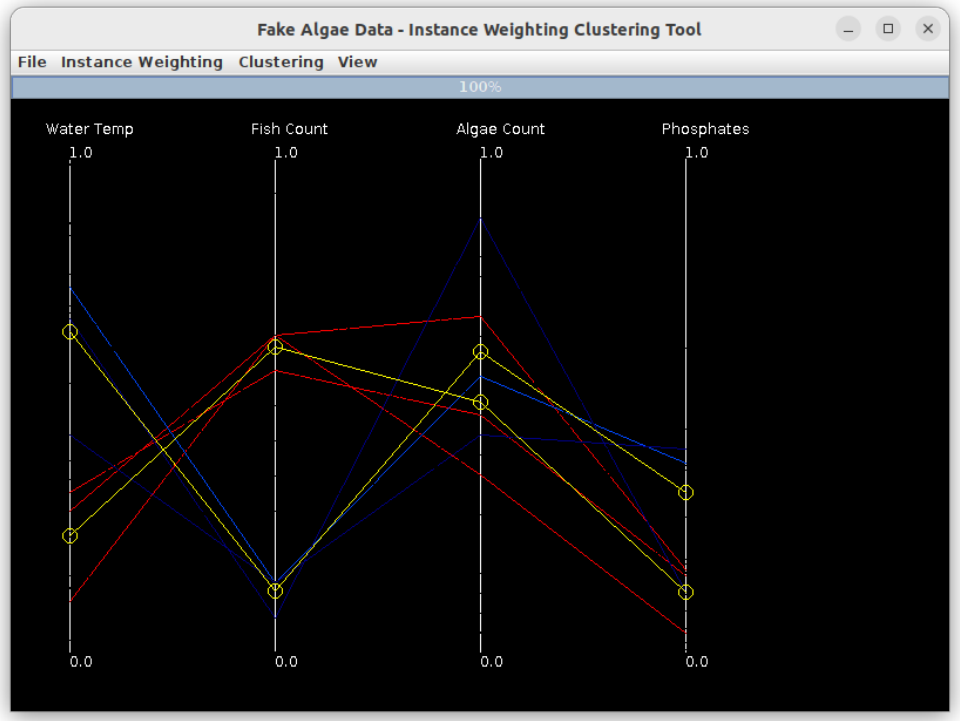


Figure 9.74: A fictitious dataset about algae visualised using the parallel coordinates plot. Instance weighting (range nearest neighbours with ϵ=0.5) and k-means ($k = 2$) has been applied.

9.12 Software Implementations of Algorithms

9.12.1 KMeans

```
import numpy as np
import pandas as pd

name = "K-Means"
version = "1.0.0.1"

class Cluster:
    def __init__(self, cycle, centroid, cols):
        self.cy = cycle
        self.c = centroid
        self.i = pd.DataFrame(columns=cols)

def p_root(value, root):
    root_value = 1 / float(root)
    return float(value) ** float(root_value)

def calculate_distance(a, b, p):
        if (len(a) != len(b)) :
            raise Exception(
                "Array item counts are different. A:" + str(a) +
                  " B:" + str(b))
        return ( p_root( sum( pow( abs(i-j), p) for i, j in zip(a, b)), p))

def clusters_to_labels(clusters):
    labels = pd.DataFrame({'label':[]})
    for cluster_index in range(len(clusters)):
        for instance_id in clusters[cluster_index].i.index:
            labels.at[instance_id, 'label'] = cluster_index
    labels.sort_index(inplace=True)
    labels['label'] = labels['label'].astype(int)
    return labels

class Kmeans:
    def __init__(self, k, p, max_cycles, verbose):
        self.k = k
        self.p = p
        self.max_cycles = max_cycles
```

```
        self.verbose = verbose

        # Resultant labels for querying after fit
        self.labels_ = []
        self.centroids_ = []

    def fit(self, df):
        #### INITIALISING KMEANS ####

        # Basic sanity checks
        if((df is None) or (self.k is None) or (self.p is None) or
           (self.max_cycles is None) or (self.verbose is None)):
            print("Missing parameters.")
            return
        if(self.k < 2):
            print("The k value must be 2 or higher.")
            return []
        if(len(df) < self.k):
            print("There is too few data points.")
            return []
        if(self.max_cycles < 1):
            print("Max cycles (the limit on the number of iterations) must be positive.")
            return []

        # Copy data set
        data = df.copy()

        # Create clusters position
        clusters = [];
        num_features = len(data.values[0])

        # Initially randomly choose instances to place cluster centroids at
        centroid_instances = data.sample(frac=1).head(self.k)
        for centroid_index in centroid_instances.index:
            clusters.append(Cluster(0,
                            centroid_instances.loc[centroid_index].values, data.columns))

        # Assign instances to their nearest centroid
        for instance_index, instance in data.iterrows():
            nearest_cluster_distance = None
            nearest_cluster = None
```

```
    # Find cluster with nearest centroid
    for cluster in clusters:
        # Work out the distance
        distance = calculate_distance(instance, cluster.c, self.p)

        # Is this centroid near the nearest to this instance
        if  (nearest_cluster_distance == None) or (distance <
             nearest_cluster_distance):
            nearest_cluster_distance = distance
            nearest_cluster = cluster

    # Add instance to cluster it is nearest to
    nearest_cluster.i.loc[instance_index] = instance


# Quantisation error indicates fast the clustering
# converges and can indicate if there are issues
# for each iteration, the instances report how
# far they are from their assigned cluster.
if self.verbose:
    quant_error = []

# Run algorithm, until no more instances are reassigned to clusters or max cycles is
     exceeded
cycles = 0;
reassignment = True;
while(reassignment and cycles < self.max_cycles):

    # Print the intrim states
    #print('#', end='')
    if self.verbose:
        print("Cycle " + str(cycles))
        for cluster_index in range(len(clusters)):
            print("Cluster: " + str(cluster_index))
            print("Position: " + str(clusters[cluster_index].c))
            print("Instances: " + str(len(clusters[cluster_index].i)))

    # Calculate the quantisation error
    if self.verbose:
        total_distance = 0
```

```
    total_count_distances = 0
    for cluster_index in range(len(clusters)):
        for instance_index, instance in clusters[cluster_index].i.iterrows():
            # measure the distance between cluster and instance
            total_distance = total_distance + calculate_distance(instance,
                 clusters[cluster_index].c, self.p)
            total_count_distances = total_count_distances + 1
    quant_error.append(total_distance / total_count_distances)

# Used later to decide to whether to loop again
reassignment = False;

# Create clusters position
new_clusters = []

# Clusters compute their new centroids
for cluster_index in range(len(clusters)):
    # Average this clusters instances to find new centroid location
    if len(clusters[cluster_index].i) > 0:
        new_centroid = []
        # Sum up values for each feature
        for feature in range(num_features):
            # Work out average
            average = clusters[cluster_index].i.iloc[:, feature].sum() /
                 len(clusters[cluster_index].i)
            # Build up new centroid position
            new_centroid.append(average)
    else:
        # No instances to calculate new position
        new_centroid = clusters[cluster_index].c
    # Create new cluster at new position
    new_clusters.append(Cluster(cycles, new_centroid[:], data.columns))

# Assigning points from the old clusters positions to the new cluster positions
for cluster_index in range(len(clusters)):
    # For each instance in the old clusters...
    for instance_index, instance in clusters[cluster_index].i.iterrows():
        # Which of the new clusters is nearest?
        nearest_cluster_distance = None
        nearest_cluster_index = None
        nearest_cluster = None
```

```
                # Find cluster with the nearest centroid, try each cluster
                for new_cluster_index in range(len(new_clusters)):
                    # Work out the distance to the new cluster
                    distance = calculate_distance(instance,
                         new_clusters[new_cluster_index].c, self.p)
                    # Is this centroid nearest to this instance
                    if  (nearest_cluster_distance == None) or (distance <
                         nearest_cluster_distance):
                        nearest_cluster_distance = distance
                        nearest_cluster_index = new_cluster_index
                        nearest_cluster = new_clusters[new_cluster_index]
                # Is this cluster the cluster the instance was previously assigned to?
                if cluster_index != nearest_cluster_index :
                    # Yes the instances are still moving around we will loop again
                    reassignment = True
                # Add instance to cluster it is nearest to
                nearest_cluster.i.loc[instance_index] = instance

            # Over write old clusters
            clusters = new_clusters[:]
            # Increment cycles
            cycles = cycles + 1

        self.labels_ = clusters_to_labels(clusters)['label'].to_numpy()
        self.centroids_ = [cluster.c for cluster in clusters]
```

9.12.2 LOFKMeans

```
import random
import numpy as np
import pandas as pd
from sklearn import preprocessing
from sklearn import neighbors


class Cluster:
    def __init__(self, cycle, centroid, cols):
        self.cy = cycle
        self.c = centroid
        self.i = pd.DataFrame(columns=cols)
```

```
def p_root(value, root):
    root_value = 1 / float(root)
    return float(value) ** float(root_value)


def calculate_distance(a, b, p):
        if (len(a) != len(b)) :
            raise Exception(
                "Array item counts are different. A:" + str(a) +
                  " B:" + str(b))
        return ( p_root( sum( pow( abs(i-j), p) for i, j in zip(a, b)), p))

def clusters_to_labels(clusters):
    labels = pd.DataFrame({'label':[]})
    for cluster_index in range(len(clusters)):
        for instance_id in clusters[cluster_index].i.index:
            labels.at[instance_id, 'label'] = cluster_index
    labels.sort_index(inplace=True)
    labels['label'] = labels['label'].astype(int)
    return labels




class LOFKmeans:
    def __init__(self, k, p, max_cycles, lofk, verbose):
        # Capture parameters
        self.k = k
        self.p = p
        self.max_cycles = max_cycles
        self.lofk = lofk
        self.verbose = verbose

        # Resultant labels for querying after fit
        self.labels_ = []
        self.centroids_ = []

    def fit(self, df):
        # Basic sanity checks
        if((df is None) or (self.k is None) or
            (self.p is None) or (self.max_cycles is None) or
        (self.lofk is None) or (self.verbose is None)):
```

```
        print("Missing parameters!")
        return
    if(len(df.columns) < 2):
        print("Minimum 2 dimmensional data")
        return
    if(self.k < 2):
        print("The k value must be 2 or higher")
        return
    if(len(df) < self.k):
        print("There is too few data points!")
        return
    if(self.max_cycles < 1):
        print("Max cycles (the limit on the number of iterations) must be positive!")
        return
    if(len(df) < self.lofk):
        print("The neighbours for LOF algorithm must be less than number of data points!")
        return

    # Copy data set
    data = df.copy()

    # Create clusters position
    clusters = [];
    num_features = len(data.values[0])

    # Calculate the LOF for instance weights over the whole dataset pre-clustering
    clf = neighbors.LocalOutlierFactor(n_neighbors=self.lofk)
    y_pred = clf.fit_predict(data)

    # Scores for how outlying each instance is
    # LOF -1 for anomalies/outliers and 1 for inliers.
    x_scores = clf.negative_outlier_factor_

    # Normalise weights between 0 and 1
    min_max_scaler = preprocessing.MinMaxScaler()
    x_scores = min_max_scaler.fit_transform(x_scores.reshape(-1, 1))

    # Create the weights column contianing the weights
    data["weight"] = x_scores

    # List the columns, this is for each cluster list of instances
```

```
cols = list(data.columns)

# Choose less outlying instances to initially place the centroids
# never choose the same instance twice clusters need to be in different places
centroid_index = 0
used_indexes = []
while centroid_index < self.k:
    # Randomly pick an instance
    potential_index = random.randint(0,len(data)-1)
    # Have we already used this instance as cluster position
    if potential_index not in used_indexes:
        instance = data.iloc[potential_index].values
        # look at the weight column and draw a random number, if the higher the weight
        # the more likely the random is going to be less and active the statement
        # which set the position of instance as a centroid location
        random_float_a = random.random()
        random_float_b = random.random()
        # Using two random and threshold to further basis towards higher values
        if((random_float_a + random_float_b) > 1):
            random_float = 1
        else:
            random_float = (random_float_a + random_float_b)
        instance_weight = instance[len(instance)-1]
        if(random_float <= instance_weight):
            # This instance becomes gets centroid place on it
            if self.verbose:
                print("Using instance as centroid: " +
                      str(instance[0:(len(instance)-1)]))
            clusters.append(Cluster(0, instance[0:(len(instance)-1)], cols))
            centroid_index = centroid_index + 1
            used_indexes.append(potential_index)

# Assign instances to their nearest centroid
for instance_index, instance in data.iterrows():
    nearest_cluster_distance = None
    nearest_cluster = None

    # Find cluster with nearest centroid
    for cluster in clusters:
        distance = calculate_distance(instance.iloc[0:(len(instance)-1)].values,
             cluster.c, self.p)
```

```
        # Is this centroid near the nearest to this instance
        if  (nearest_cluster_distance == None) or (distance <
             nearest_cluster_distance):
            nearest_cluster_distance = distance
            nearest_cluster = cluster

    # Add instance to cluster to its nearest cluster
    nearest_cluster.i.loc[instance_index] = instance

# Quantisation error indicates fast the clustering
# converges and can indicate if there are issues
# for each iteration, the instances report how
# far they are from their assigned cluster.
if self.verbose:
    quant_error = []



# Run algorithm, until no more instances are reassigned to clusters or max cycles is
     exceeded
cycles = 0;
reassignment = True;
while(reassignment and cycles < self.max_cycles):

    # Print the intrim states
    #print('#', end='')
    if self.verbose:
        print("Cycle " + str(cycles))
        for cluster_index in range(len(clusters)):
            print("Cluster: " + str(cluster_index))
            print("Position: " + str(clusters[cluster_index].c))
            print("Instances: " + str(len(clusters[cluster_index].i)))

    # Calculate the quantisation error
    if self.verbose:
        total_distance = 0
        total_count_distances = 0
        for cluster_index in range(len(clusters)):
            for instance_index, instance in clusters[cluster_index].i.iterrows():
                # measure the distance between cluster and instance
```

```
            total_distance = total_distance +
                calculate_distance(instance.iloc[0:(len(instance)-1)].values,
                clusters[cluster_index].c, self.p)
            total_count_distances = total_count_distances + 1
    quant_error.append(total_distance / total_count_distances)

# Used later to decide to whether to loop again
reassignment = False;

# Create clusters position
new_clusters = []

# Clusters compute their new centroids
for cluster_index in range(len(clusters)):
    # Average this clusters instances to find new centroid location
    new_centroid = []
    # Sum up values for each feature
    for feature in range(num_features):
        # Work out the position of the new centroid,
        # taking the weight into consideration
        average = 0
        total_instance = 0
        total_weight = 0
        for instance_index, instance in clusters[cluster_index].i.iterrows():
            instance_weight = instance.iloc[len(instance)-1]
            total_instance = total_instance + (instance.iloc[feature] *
                instance_weight)
            total_weight = total_weight + instance_weight
        if total_weight > 0:
            average = total_instance / total_weight
            # Build up new centroid position
            new_centroid.append(average)
        else:
            # No weight the centroid doesn't move
            new_centroid.append(clusters[cluster_index].c[feature])
    # Create new cluster at new position
    new_clusters.append(Cluster(cycles, new_centroid[:], data.columns))

# Assigning points from the old clusters positions to the new cluster positions
for cluster_index in range(len(clusters)):
    # For each instance in the old clusters...
```

```
            for instance_index, instance in clusters[cluster_index].i.iterrows():
                # Which of the new clusters is nearest?
                nearest_cluster_distance = None
                nearest_cluster_index = None
                nearest_cluster = None
                # Find cluster with the nearest centroid, try each cluster
                for new_cluster_index in range(len(new_clusters)):
                    distance = calculate_distance(instance.iloc[0:(len(instance)-1)].values,
                               new_clusters[new_cluster_index].c, self.p)
                    # Is this centroid near the nearest to this instance
                    if  (nearest_cluster_distance == None) or (distance <
                         nearest_cluster_distance):
                        nearest_cluster_distance = distance
                        nearest_cluster_index = new_cluster_index
                        nearest_cluster = new_clusters[new_cluster_index]
                # Is this cluster the cluster the instance was previously assigned to?
                if cluster_index != nearest_cluster_index :
                    # Yes the instances are still moving around we will loop again
                    reassignment = True
                # Add instance to cluster it is nearest to
                nearest_cluster.i.loc[instance_index] = instance

            # Over write old clusters
            clusters = new_clusters[:]
            # Increment cycles
            cycles = cycles + 1


        self.labels_ = clusters_to_labels(clusters)['label'].to_numpy()
        self.centroids_ = [cluster.c for cluster in clusters]
```

9.12.3 ILOFKMeans

```
import random
import numpy as np
import pandas as pd
from sklearn import preprocessing
from sklearn import neighbors

class Cluster:
```

```
    def __init__(self, cycle, centroid, cols):
        self.cy = cycle
        self.c = centroid
        self.i = pd.DataFrame(columns=cols)

def p_root(value, root):
    root_value = 1 / float(root)
    return float(value) ** float(root_value)

def calculate_distance(a, b, p):
        if (len(a) != len(b)) :
            raise Exception("Array item counts are different. A:" + str(a) + " B:" + str(b))
        return ( p_root( sum( pow( abs(i-j), p) for i, j in zip(a, b)), p))

def clusters_to_labels(clusters):
    labels = pd.DataFrame({'label':[]})
    for cluster_index in range(len(clusters)):
        for instance_id in clusters[cluster_index].i.index:
            labels.at[instance_id, 'label'] = cluster_index
    labels.sort_index(inplace=True)
    labels['label'] = labels['label'].astype(int)
    return labels


class ILOFKmeans:
    def __init__(self, k, p, max_cycles, lofk, verbose):
        # Capture parameters
        self.k = k
        self.p = p
        self.max_cycles = max_cycles
        self.lofk = lofk
        self.verbose = verbose

        # Resultant labels for querying after fit
        self.labels_ = []
        self.centroids_ = []

    def fit(self, df):
        #### INITIALISING KMEANS ####

        # Basic sanity checks
```

```
if((df is None) or (self.k is None) or
    (self.p is None) or (self.max_cycles is None) or
(self.lofk is None) or (self.verbose is None)):
    print("Missing parameters!")
    return
if(len(df.columns) < 2):
    print("Minimum 2 dimmensional data")
    return
if(self.k < 2):
    print("The k value must be 2 or higher")
    return
if(len(df) < self.k):
    print("There is too few data points!")
    return
if(self.max_cycles < 1):
    print("Max cycles (the limit on the number of iterations) must be positive!")
    return
if(len(df) < self.lofk):
    print("The neighbours for LOF algorithm must be less than number of data points!")
    return

# Copy data set
data = df.copy()

# Create clusters position
clusters = [];
num_features = len(data.values[0])

# Calculate the LOF for instance weights over the whole dataset pre-clustering
clf = neighbors.LocalOutlierFactor(n_neighbors=self.lofk)
y_pred = clf.fit_predict(data)

# Scores for how outlying each instance is
# LOF -1 for anomalies/outliers and 1 for inliers.
x_scores = clf.negative_outlier_factor_

# Normalise weights between 0 and 1
min_max_scaler = preprocessing.MinMaxScaler()
x_scores = min_max_scaler.fit_transform(x_scores.reshape(-1, 1))

# Create the weights column contianing the weights
```

```
data["weight"] = x_scores

# List the columns, this is for each cluster list of instances
cols = list(data.columns)

# Choose less outlying instances to place the initially place the centroids
# never choose the same instance twice clusters need to be in different places
centroid_index = 0
used_indexes = []
while centroid_index < self.k:
    # Randomly pick an instance
    potential_index = random.randint(0,len(data)-1)
    # Have we already used this instance as cluster position
    if potential_index not in used_indexes:
        instance = data.iloc[potential_index].values
        # look at the weight column and draw a random number, if the higher the weight
        # the more likely the random is going to be less and active the statement
        # which set the position of instance as a centroid location
        random_float_a = random.random()
        random_float_b = random.random()
        # Using two random and threshold to further basis towards higher values
        if((random_float_a + random_float_b) > 1):
            random_float = 1
        else:
            random_float = (random_float_a + random_float_b)
        instance_weight = instance[len(instance)-1]
        if(random_float <= instance_weight):
            # This instance becomes gets centroid place on it
            if self.verbose:
                print("Using instance as centroid: " +
                      str(instance[0:(len(instance)-1)]))
            clusters.append(Cluster(0, instance[0:(len(instance)-1)], cols))
            centroid_index = centroid_index + 1
            used_indexes.append(potential_index)

# Assign instances to their nearest centroid
for instance_index, instance in data.iterrows():
    nearest_cluster_distance = None
    nearest_cluster = None

    # Find cluster with nearest centroid
```

```
    for cluster in clusters:
        # Work out the distance
        distance = calculate_distance(instance.iloc[0:(len(instance)-1)].values,
             cluster.c, self.p)

        # Is this centroid near the nearest to this instance
        if  (nearest_cluster_distance == None) or (distance <
             nearest_cluster_distance):
            nearest_cluster_distance = distance
            nearest_cluster = cluster

    # Add instance to cluster to its nearest cluster
    nearest_cluster.i.loc[instance_index] = instance

# Quantisation error indicates fast the clustering
# converges and can indicate if there are issues
# for each iteration, the instances report how
# far they are from their assigned cluster.
if self.verbose:
    quant_error = []


# Run algorithm, until no more instances are reassigned to clusters or max cycles is
     exceeded
cycles = 0;
reassignment = True;
while(reassignment and cycles < self.max_cycles):

    # Print the intrim states
    #print('#', end='')
    if self.verbose:
        print("Cycle " + str(cycles))
        for cluster_index in range(len(clusters)):
            print("Cluster: " + str(cluster_index))
            print("Position: " + str(clusters[cluster_index].c))
            print("Instances: " + str(len(clusters[cluster_index].i)))

    # Calculate the quantisation error
    if self.verbose:
        total_distance = 0
        total_count_distances = 0
```

```
    for cluster_index in range(len(clusters)):
        for instance_index, instance in clusters[cluster_index].i.iterrows():
            # measure the distance between cluster and instance
            total_distance = total_distance +
                 calculate_distance(instance.iloc[0:(len(instance)-1)].values,
                                                              clusters[cluster_index].c,
                                                                   self.p)
            total_count_distances = total_count_distances + 1
    quant_error.append(total_distance / total_count_distances)

# Used later to decide to whether to loop again
reassignment = False;

# Create clusters position
new_clusters = []

# Clusters compute their new centroids
for cluster_index in range(len(clusters)):
    # Average this clusters instances to find new centroid location
    if len(clusters[cluster_index].i) > 0:
        new_centroid = []
        # Sum up values for each feature
        for feature in range(num_features):
            # Work out the position of the new centroid,
            # taking the weight into consideration
            average = 0
            total_instance = 0
            total_weight = 0
            for instance_index, instance in clusters[cluster_index].i.iterrows():
                instance_weight = instance.iloc[len(instance)-1]
                total_instance = total_instance + (instance.iloc[feature] *
                     instance_weight)
                total_weight = total_weight + instance_weight
            if total_weight > 0:
                average = total_instance / total_weight
                # Build up new centroid position
                new_centroid.append(average)
            else:
                # No weight the centroid doesn't move this should never happen
                new_centroid.append(clusters[cluster_index].c[feature])
        # Create new cluster at new position
```

```
            new_clusters.append(Cluster(cycles, new_centroid[:], data.columns))

    # Calculate the weight of each instance in each cluster
    for cluster_index in range(len(clusters)):

        # Check there is enough instance in the cluster
        # calculate the a LOF value
        if(len(clusters[cluster_index].i) > 3):

            # Prediction of which instances are outliers using LOF
            y_pred = clf.fit_predict(clusters[cluster_index].i)

            # Scores for how outlying each instance is
            x_scores = clf.negative_outlier_factor_

            # Normalise weights between 0 and 1
            min_max_scaler = preprocessing.MinMaxScaler()
            x_scores = min_max_scaler.fit_transform(x_scores.reshape(-1, 1))

            clusters[cluster_index].i["weight"] = x_scores
        else:
            clusters[cluster_index].i["weight"] = 1.0

    # Assigning points from the old clusters positions to the new cluster positions
    for cluster_index in range(len(clusters)):
        # For each instance in the old clusters...
        for instance_index, instance in clusters[cluster_index].i.iterrows():
            # Which of the new clusters is nearest?
            nearest_cluster_distance = None
            nearest_cluster_index = None
            nearest_cluster = None
            # Find cluster with the nearest centroid, try each cluster
            for new_cluster_index in range(len(new_clusters)):
                distance = calculate_distance(instance.iloc[0:(len(instance)-1)].values,
                           new_clusters[new_cluster_index].c, self.p)
                # Is this centroid near the nearest to this instance
                if  (nearest_cluster_distance == None) or (distance <
                     nearest_cluster_distance):
                    nearest_cluster_distance = distance
                    nearest_cluster_index = new_cluster_index
                    nearest_cluster = new_clusters[new_cluster_index]
```

```
                # Is this cluster the cluster the instance was previously assigned to?
                if cluster_index != nearest_cluster_index :
                    # Yes the instances are still moving around we will loop again
                    reassignment = True
                # Add instance to cluster it is nearest to
                nearest_cluster.i.loc[instance_index] = instance

            # Over write old clusters
            clusters = new_clusters[:]
            # Increment cycles
            cycles = cycles + 1

        self.labels_ = clusters_to_labels(clusters)['label'].to_numpy()
        self.centroids_ = [cluster.c for cluster in clusters]
```

9.12.4 IWSE Clustering Algorithm

```
import os
import warnings
import random
from joblib import Parallel, delayed
import pandas as pd
import numpy as np
from scipy import spatial
from sklearn.neighbors import KernelDensity
from sklearn.cluster import SpectralClustering


warnings.simplefilter("ignore") # to ignore spectral clustering not fully connected graph
     warning.

name = "Instance Weighted Spectral Ensemble"
version = "1.0.0.2"

def execute_base_cluster(index, df_copy, k, nn, mn, mx, weighting_method):
    # To avoid instances never being sampled, job index 0 always performs 1 full clustering
         of the dataset.
    if index == 0:
        mn = 100
        mx = 100
```

```
# Matrix for the results of this set of bags base clusterers, this is co assoication
    matrix
coassoc = np.array([[0] * len(df_copy) for _ in range(len(df_copy))], dtype=np.byte) #
    byte allows for soft max of 128 bags.

df_temp = df_copy.copy()

# SRSWOR (replace = False) - sampling data using the instance weights
if weighting_method == 'U':
    # flip a coin to decide, approximtely 50% of bags executed will each type in this mode
    if bool(random.getrandbits(1)):
        weighting_method = 'H'
    else:
        weighting_method = 'L'

# Randomly choose a sample size between mn and mx
sample = random.randint(mn, mx) / 100

if weighting_method == 'H':
    # sample favouring points far from neighbors
    df_sample = df_temp.sample(n=int(len(df_temp)*sample), weights='_w', replace=False)
elif weighting_method == 'L':
    # sample favouring points near to neighbors
    df_sample = df_temp.sample(n=int(len(df_temp)*sample), weights='_iw', replace=False)
elif weighting_method == 'R':
    # uniformly random sub-sample, (no weights)
    df_sample = df_temp.sample(n=int(len(df_temp)*sample), replace=False)
else:
    raise ValueError("Invalid weighting option")


# We can now remove weight columns
df_sample = df_sample.loc[:, df_sample.columns != '_w']
df_sample = df_sample.loc[:, df_sample.columns != '_iw']

# Run the spectral clustering
spectral = SpectralClustering(n_clusters=k, affinity='nearest_neighbors', n_neighbors=nn,
    n_jobs=-1)
spectral.fit(df_sample)
df_sample["label"] = spectral.labels_
```

```
    # Update affinity matrix
    index = df_sample.index.to_numpy()
    labels = df_sample["label"].to_numpy()
    label_look_up = dict(zip(index, range(len(index)))) # necessary becuase index 5 is not
        necessarily in position 5 anymore, this remembers where 5 is in the order of index
        and labels
    for y in index:
        for x in index:
            if x >= y:
                if labels[label_look_up[x]] == labels[label_look_up[y]]:
                    coassoc[y][x] = coassoc[y][x] + 1
    coassoc_flipped = np.rot90(np.fliplr(np.copy(coassoc)))
    np.fill_diagonal(coassoc_flipped, 0)
    return coassoc + coassoc_flipped

def add_exp_weights(df, bandwidth):
    # Calcualate density and inverted density
    kde = KernelDensity(kernel='exponential', bandwidth=bandwidth).fit(df)
    exp_scores = kde.score_samples(df)

    # 0.01 to avoid 0 values
    exp_scores_norm = (exp_scores - min(exp_scores)) / (max(exp_scores) - min(exp_scores)) +
        0.01
    exp_scores_norm_inverted = abs(exp_scores_norm + -1.0) + 0.01

    df["_w"] = exp_scores_norm
    df["_iw"] = exp_scores_norm_inverted

    return df

class IWSE:
    def __init__(self, k, nn, mn, mx, bags, weighting_method):
        # Capture parameters
        self.k = k
        self.nn = nn
        self.mn = mn
        self.mx = mx
        self.bags = bags
        self.weighting_method = weighting_method
```

```
    # Initialise variables for querying
    self.labels_ = []

def fit(self, df):
    # check reserves column names are not used
    if self.weighting_method != "R" and ("_w" not in df.columns or "_iw" not in
         df.columns):
        raise ValueError("Dataframe is missing weight column _w! Use IWSE.add_mnn_weights
             or IWSE.add_exp_weights functions to generate _w and _iw columns")

    # check data frame size
    if len(df) <= self.k or len(df) <= self.nn:
        raise ValueError("Dataframe is too small for specified k and nn values.")

    # Take a copy of the dataset, which we will work with
    df_copy = df.copy()
    df_copy.sort_index(inplace=True)

    # Parallel process the bags
    results = Parallel(n_jobs=os.cpu_count())(delayed(execute_base_cluster)(job_index,
         df_copy, self.k, self.nn, self.mn, self.mx, self.weighting_method) for job_index
         in range(self.bags))

    # Sum coassoc matrices together from the sets of bags
    coassoc = np.array([[0] * len(df) for _ in range(len(df))])
    for i in range(self.bags):
        coassoc = coassoc + results[i]

    # Run final spectral clustering, using the affinity matrix calculated by the above
         runs
    final_spectral = SpectralClustering(n_clusters=self.k, affinity='precomputed',
         n_neighbors=self.nn)
    final_spectral.fit(coassoc) # use the combined co-association matrix as the affinty
         matrix for spectral.

    self.labels_ = final_spectral.labels_
```

9.13 Research Publications

The research reported in this thesis resulted a two publications:

Moggridge, P, Helian, N, Sun, Y & Lilley, M 2023, On Instance Weighted Clustering Ensembles, Paper presented at The 31th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium, 4/10/23 - 6/10/23

Abstract: Ensemble clustering is a technique which combines multiple clustering results, and instance weighting is a technique which highlights important instances in a dataset. Both techniques are known to enhance clustering performance and robustness. In this research, ensembles and instance weighting are integrated with the spectral clustering algorithm. We believe this is the first attempt at creating diversity in the generative mechanism using density based instance weighting for a spectral ensemble. The proposed approach is empirically validated using synthetic datasets comparing against spectral and a spectral ensemble with random instance weighting. Results show that using the instance weighted sub-sampling approach as the generative mechanism for an ensemble of spectral clustering leads to improved clustering performance on datasets with imbalanced clusters.

Moggridge, P, Helian, N, Sun, Y, Lilley, M & Veneziano, V 2020, Instance Weighted Clustering: Local Outlier Factor and K-Means. in L Iliadis, PP Angelov, C Jayne & E Pimenidis (eds), Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference: Proceedings of the EANN 2020. Proceedings of the In-

ternational Neural Networks Society, Springer Nature, pp. 435-446. https://doi.org/10.1007/978-3-030-48791-1_34

Abstract: Clustering is an established unsupervised learning method. Substantial research has been carried out in the area of feature weighting, as well instance selection for clustering. Some work has paid attention to instance weighted clustering algorithms using various instance weighting metrics based on distance information, geometric information and entropy information. However, little research has made use of instance density information to weight instances. In this paper we use density to define instance weights. We propose two novel instance weighted clustering algorithms based on Local Outlier Factor and compare them against plain k-means and traditional instance selection.

There was also some research made in an adjacent field prior to a change to the current topic:

Moggridge, P, Helian, N, Sun, Y, Lilley, M, Veneziano, V & Eaves, M 2019, Improving the MXFT Scheduling Algorithm for a Cloud Computing Context, International Journal of Grid and Utility Computing (IJGUC), vol. 10, no. 6, pp. 618 – 638. https://doi.org/10.1504/IJGUC.2019.102711

Abstract: In this paper, the Max-Min Fast Track (MXFT) scheduling algorithm is improved and compared against a selection of popular algorithms. The improved versions of MXFT are called Min-Min Max-Min Fast Track (MM-MXFT) and Clustering Min-Min Max-Min Fast Track (CMMMXFT). The key difference is using Min-Min for the fast track. Experimentation revealed that

despite Min-Min's characteristic of prioritising small tasks at the expense of overall makespan, the overall makespan was not adversely affected and the benefits of prioritising small tasks were identified in MMMXFT. Experiments were conducted by using a simulator with the exception of one real-world experiment. The real-world experiment identified challenges faced by algorithms which rely on accurate execution time prediction.

Moggridge, P, Helian, N, Sun, Y, Lilley, M, Veneziano, V & Eaves, M 2017, Revising Max-min for Scheduling in a Cloud Computing Context. in 2017 IEEE 26th International Conference on Enabling Technologies: : Infrastructure for Collaborative Enterprises (WETICE). IEE, The 26th IEEE International Cnference on Enable Technologies: Infrastructure for Collaborative Enerprises, Poznan , Poland, 21/06/17. https://doi.org/10.1109/WETICE.2017.58

Abstract: Adoption of Cloud Computing is on the rise and many datacenter operators adhere to strict energy efficiency guidelines. In this paper a novel approach to scheduling in a Cloud Computing context is proposed. The algorithm Maxmin Fast Track (MXFT) revises the Max-min algorithm to better support smaller tasks with stricter Service Level Agreements (SLAs), which makes it more relevant to Cloud Computing. MXFT is inspired by queuing in supermarkets, where there is a fast lane for customers with a smaller number of items. The algorithm outperforms Max-min in task execution times and outperforms Min-min in overall makespan. A by-product of investigating this algorithm was the development of simulator called "ScheduleSim" which makes it simpler to prove a scheduling algorithm before committing to a specific scheduling problem in Cloud Computing and therefore might be a useful

precursor to experiments using the established simulator CloudSim.