Clustering, classifying and matching patterns with ensemble techniques

Vom Fachbereich 18
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von
Sergey Sukhanov, M.Sc.

Referent:	Prof. Dr.-Ing. Abdelhak M. Zoubir
Korreferent:	Dr.-Ing. Michael Muma

Darmstadt, 2021

Sergey Sukhanov – Clustering, classifying and matching patterns with ensemble techniques
Darmstadt, Technische Universität Darmstadt
Jahr der Veröffentlichung der Dissertation auf TUprints: 2021
URN: urn:nbn:de:tuda-tuprints-198976
Tag der mündlichen Prüfung: 31. Mai 2021

Acknowledgments

I would like to express my gratitude to all the people who supported and inspired me during my doctoral research.

First of all, I would like to thank my adviser Prof. Dr.-Ing. Abdelhak M. Zoubir for his constant support and delicate mentoring. I received a perfect mixture of wisdom guidance and sufficient freedom in order to stay focused, always move forward, and accomplish this thesis.

I would like to thank my co-advisor Dr.-Ing. Michael Muma for his valuable feedback and assistance that I have received.

I sincerely thank Christian Debes for his continuous help, countless highly valuable advices throughout all these years, enormous trust and reliability. It was really inspiring to working with you, to learn from you, and grow as a professional.

I would like to thank all the Signal Processing Group current and former members who I met during these years: Michael Fauss, Freweyni Teklehaymanot, Mark Ryan Leonard, Tim Schäck, Adrian Šošić, Wenjun Zeng, Patricia Binder, Lala Khadidja Hamaidi, Sahar Khawatmi, Wassim Suleiman, Simon Rosenkranz, Christian Weiss, Nevine Demitri, Michael Muma, Mahmoud El-Hindi, Martin Gölz, Huiping Huang, Di Jin, Jasin Machkour, Afief Dias Pambudi, Dominik Reinhard, Ann-Kathrin Seifert, Aylin Tastan, Amare Kassaw, Toufik Mouchini, Jack Dagdagan, Ziliang Qiao, Stefano Fortunati and Roy Howard. It was a wonderful time being part of the group, enjoying the unique atmosphere, sharing knowledge, and receiving support. I would like to deeply acknowledge Renate Koschella and Hauke Fath for their assistance. You were always reliable and supportive.

My special appreciation goes to all current and former members of the Data Analytics team of AGT R&D GmbH in Darmstadt, especially, to Ivan Tankoyeu, Roel Heremans, Jérôme Louradour, Dmitrii Budylskii, Nikolaos Frangiadakis, Andreas Merentitis, Darya Trofimova, Agoston Torok, Jürgen Hahn, Grigory Alexandrovich, Michael Leigsnering and many others. It was a truly exciting time working on various research challenges and data-driven projects, learning together, having fun while discussing diverse topics, writing papers, participating in data contests, and hackathons. You are all exceptional people.

I would like to also acknowledge colleagues from other departments of AGT, namely, Joachim Schaper for his assistance and help, Thomas Bader for supporting my research

in such a dynamic environment full of deadlines and deliveries, Johannes Wowra, Sergei Liebich, Ilgar Mashayev, Rainer Müller, Robert Biehl, Jason Rambach, Konstantinos Stergakis, Alexandros Lefas and many others.

My special appreciation goes to my students and interns I was working and interacting with during these years: Chenglei Chen, Maximilian Stiefel, Vishal Gupta, Renzhi Wu, Michele Piccolini, Gizem Ekinci, Eike Mentzendorff and Ali Shaban.

Also, I would like to thank the IEEE GRSS Image Analysis and Data Fusion Technical Committee for serving annual contests that helped me to learn, compete and contribute: Naoto Yokoya, Pedram Ghamisi, Junshi Xia, Benjamin Bechtel, Bertrand Le Saux, Gabriele Moser and Devis Tuia.

I would also like to thank many friends who I met through all these years.

I would like to deeply thank my parents Andrey and Elena for their love, trust, and support throughout all my life.

Last but not least my warmest appreciation goes to my beautiful wife Alisa and my wonderful daughters Katja and Tonja for constantly being close, sharing love and smiles, taking care, and believing in me.

Kurzfassung

Diese Arbeit befasst sich mit drei wichtigen allgemeinen Problemen des maschinellen Lernens und der Signalverarbeitung: Clustering, Klassifizierung und Mustervergleich, die bei vielen wissenschaftlichen und praktischen Herausforderungen auftreten. Trotz vieler Lösungen, die im letzten Jahrzehnt vorgeschlagen wurden, sind diese Probleme immer noch mit besonderen Schwierigkeiten verbunden: Viele Ansätze scheitern, wenn es um die Mehrdimensionalität von Signalen geht; Einige Methoden können aufgrund ihrer intrinsischen Komplexitätsbeschränkungen nur mit einer moderaten Datenmenge arbeiten. Bei den meisten Frameworks müssen schwierige Entscheidungen mit Annahmen getroffen werden, die in der Realität möglicherweise nicht zutreffen. Durch die Nutzung des Gruppenlernens oder der Weisheit der Massenkonzepte wird in dieser Arbeit ein Ensemble-Lernparadigma eingeführt, um diese grundlegenden Herausforderungen zu lösen.

Der erste Teil der Dissertation befasst sich mit dem Problem der Identifizierung ähnlicher Objektgruppen, die auch als Clustering bezeichnet werden. Obwohl Clustering in vielen Bereichen weit verbreitet ist, birgt es einige grundlegende intrinsische Herausforderungen (Subjektivität, großer Parametersatz, eigene Annahmen zu resultierenden Clustern usw.), die häufig zufriedenstellende Ergebnisse behindern. Um diesen Herausforderungen zu begegnen, wird ein neuartiges Rahmenwerk für Konsenscluster vorgeschlagen. Es arbeitet mit mehreren Clustering-Ergebnissen und bietet zwei skalierbare Möglichkeiten, um das Problem anzugehen. Erstens bietet der vorgeschlagene Ansatz unter Berücksichtigung der Nachteile der Hamming-Distanz bei auf Koexistenz basierenden Konsensclustering-Methoden die Konstruktion eines aussagekräftigen Distanzmaßes, das mit Datenstrukturen arbeitet, die als Datenfragmente bezeichnet werden. Infolgedessen wird eine neuartige Konsensfunktion um diese Maßnahme herum aufgebaut, die auf einer hierarchischen Clustering-Methode basiert, die stabile und genaue Ergebnisse zeigt. Zweitens ermöglicht es die Formulierung eines Konsens-Clustering-Problems als ein Faktorisierungsproblem der binären Matrix, es effizient mittels einer rekursiven binären Matrix des ersten Ranges zu lösen. Dies bringt eine deskriptive Interpretation der Ergebnisse mit sich, die für große Datensätze und eine große Anzahl von Ensemblemitgliedern geeignet ist.

Der zweite Teil der Dissertation befasst sich mit der Klassifizierungsaufgabe, bei der es darum geht, eine von mehreren vordefinierten Kategorien zu bestimmen, zu denen ein Objekt gehört. Wir lösen hochdimensionale Fernerkundungsdatenfusionsprobleme, indem wir sie als Klassifizierungsaufgabe formulieren und einen dynamischen Klassifikator und ein Ensemble-Auswahlrahmen vorschlagen. Das vorgeschlagene Framework

stützt sich auf das Konzept mehrerer Klassifikatorsysteme und wählt kompetente Klassifikatoren aus einem etablierten Ensemble aus und kombiniert sie, um eine zuverlässige und genaue Klassifizierung zu gewährleisten. Um dies zu ermöglichen, wird eine Kompetenzschätzungs- und Auswahlmethode entwickelt.

Im dritten Teil der Dissertation befassen wir uns mit dem Problem der Ähnlichkeitssuche in Datenströmen, bei dem es darum geht, ähnliche Objekte (oder Ereignisse) in einem Echtzeitdatenstrom zu finden. Aufgrund von Ausreißern, Rauschen und möglichen Verzerrungen in Amplituden- und Zeitdimensionen ist es oft schwierig, die erforderlichen Muster korrekt aus dem Stream abzurufen. Um dies zu ermöglichen, schlagen wir einen dynamischen Normalisierungsmechanismus vor, der es ermöglicht, Streaming-Signal-Teilsequenzen auf die Skala der Abfragevorlage zu bringen. Darüber hinaus erweitern wir es für den Fall, dass mehrere Beispiele für eine Abfragevorlage verfügbar sind, mit denen Sie die Weisheit der Crowd-Konzepte in den Einstellungen für den Mustervergleich nutzen können. Dies verbessert die Musterabruffähigkeiten erheblich, insbesondere wenn Stichprobenvarianz oder Zeitverzerrungen vorliegen.

Die vorgeschlagenen Beiträge zur Clusterbildung, Klassifizierung und Mustererkennung werden an künstlich erzeugten Datensätzen sowie anhand realer Messdaten untersucht und validiert, die aus offenen Quellen stammen oder im Labor der AGT Group (R&D) GmbH, Darmstadt, Deutschland, aufgezeichnet wurden. Es werden mehrere Experimente durchgeführt, um die Leistungskonsistenz der vorgeschlagenen Methoden zu bestätigen und zu verifizieren sowie teilweise in reale Lösungen integriert.

Abstract

This thesis addresses three important general machine learning and signal processing problems: clustering, classification, and pattern matching which arise in many scientific and practical challenges. Despite many solutions proposed throughout the last decade, these problems are still imposing particular difficulties when addressing them: many approaches fail when it comes to the multidimensional nature of signals; some methods are able to operate only with a moderate amount of data due to their intrinsic complexity limitations; the majority of frameworks require hard decisions to be provided with assumptions that might not hold in reality. By leveraging group learning or wisdom of the crowds concepts, this thesis brings in an ensemble learning paradigm in order to solve these fundamental challenges.

The first part of the dissertation addresses the problem of identifying similar groups of objects also known as clustering. While being widely used in many domains, clustering carries several fundamental intrinsic challenges (subjectivity, large parameter set, own assumptions on resulting clusters, etc.) that often hinder satisfactory results. To address these challenges, a novel consensus clustering framework is proposed. Operating on multiple clustering outcomes it provides two scalable ways of approaching the problem. First, by accounting for the drawbacks of the Hamming distance in co-occurrence-based consensus clustering methods the proposed approach offers construction of an expressive distance measure operating with data structures called data fragments. As the result, a novel consensus function is built around this measure based on a hierarchical clustering method demonstrating stable and accurate results. Second, by formulating a consensus clustering problem as a binary matrix factorization problem it allows to efficiently solve it by means of a recursive rank-one binary matrix approximation. This brings descriptive results interpretation suiting large-scale datasets and a high amount of ensemble members.

The second part of the dissertation deals with the classification task that is about deciding for one out of several predefined categories that an object belongs to. We solve high dimensional remote sensing data fusion problems by formulating them as a classification task and proposing a dynamic classifier and ensemble selection framework. Relying on multiple classifier systems concept the proposed framework selects and combines competent classifiers from an established ensemble in order to provide reliable and accurate classification. To enable that, a competence estimation and selection methodology is developed.

In the third part of the dissertation, we address the problem of similarity search in data streams that is about finding similar objects (or events) in a real-time stream of

data. Due to outliers, noise, and potential distortions in amplitude and time dimensions, it is often challenging to correctly retrieve required patterns from the stream in presence of distortions and outliers. To enable this, we propose a dynamic normalization mechanism that allows bringing streaming signal subsequences to the scale of the query template. Additionally, we extend it for the case when multiple examples of a query template are available allowing for leveraging the wisdom of the crowds concepts in pattern matching settings. This significantly improves pattern retrieval capabilities, especially when sampling variance or time distortions are present.

The proposed contributions for clustering, classification, and pattern matching are studied and validated on artificially generated datasets as well as on the real-world measurement data obtained from open sources or recorded in the laboratory of AGT Group (R&D) GmbH, Darmstadt, Germany. Multiple experiments are conducted to confirm and verify the performance consistency of the proposed methods as well as partly integrated into real-world solutions.

Contents

Chapter 1

Introduction

This chapter provides an introduction to the domain of the thesis, highlights current challenges in the field, and provides motivation for contributions presented. Further, it states research aims and objectives, outlines the original contributions, lists all the publications published throughout the time of this doctoral research, and presents an overview of the thesis structure.

1.1 Motivation

Clustering, classification, and pattern matching problems are, probably, one of the main challenges in machine learning and signal processing that are regularly faced by both research and industrial communities. Despite the fact that these problems seem to come from different domains they aim at solving similar tasks: identify similar groups of objects, classify objects into several categories, find similar objects (or events) in a real-time stream of data. Additionally, these problems often share similar technical concepts and frameworks when being addressed.

During the last several decades, many algorithms and methods were proposed in order to solve the above-mentioned challenges [Has19; Fuk13; Anz12; MAY]. Many of them rely on ideas motivated by advances in physics, chemistry, biology, or other natural sciences, inspired by the way humans (or animals) operate or are organized, follow common organizational principles or philosophical aspects.

The wisdom of the crowds concept originating from the group learning methodology has recently brought a significant impact to machine learning and signal processing communities. Based on a simple assumption that, generally, many opinions are superior to a single one, it allowed to boost the performance of existing machine learning and signal processing models and gave birth to many new ones. The idea that a group of individuals is collectively smarter than single experts has been already known from Aristotle's theory of collective judgment appeared in his work Politics [Mil17]. After that, throughout all these years, it was utilized in decision making and problem-solving in many practical areas. In the XXI century, the concept of Wisdom of Crowds was popularized by James Surowiecki in his book [Sur05] in which he demonstrated the

impact of the concept to current behavioral economics, art, psychology, pop culture, and other areas. According to Surowiecki, there are several components required to create a wise crowd:

1. Diversity of opinion
2. Independence
3. Decentralization
4. Aggregation
5. Trust

Diversity of opinion implies a private person's belief even if this belief is based on a specific interpretation of the known facts. Independence expects that people's opinions are not affected by others' opinions. Decentralization assumes that people are using local knowledge to draw conclusions. Aggregation suggests the existence of strategies to transform private judgments into a joint decision. Trust suppose that every individual trusts the crowd to be fair.

Four years later, Oinas-Kukkonen in his book [Oin08] refined and further developed these components. Specifically, he described the possibility to explain how a group of people think as a whole. He pointed out that groups of individuals might be exceptionally intelligent and are often smarter than the smartest participants of such a group. However, to obtain superior results there is a strong need in a way information from experts gets aggregated. Additionally, this aggregation function has to timely deliver the right information to the right parties avoiding much communication as it might bring down the intelligence of the whole group. Finally, he highlighted that the best decisions are obtained by contest and disagreement.

The contributions of Surowiecki and Oinas-Kukkonen allowed us to bring structure into an understanding of the wisdom of crowds paradigm and refine existing methods based on it. Indeed, there were many approaches from different areas developed and inspired by the wisdom of the crowds concept. Developed in 1968 (and still in use nowadays) The Delphi Method [DH63] forecasts the impact of technology on warfare by relying on the assumption of the superiority of group judgments compared to an individual once. Additionally, it utilizes the fact that structured group judgments are superior to unstructured ones. Decision theory in economics, philosophy, and psychology often rely on disconnecting problems from solutions and decision makers [Edw54].

Swarm intelligence, multi-armed bandits, error-correcting codes are all examples of such concept [CK17].

Ensemble learning or model averaging has appeared in order to solve complex machine learning challenges by leveraging the wisdom of the crowds concept. It gave birth to such concepts as voting, averaging, binning, bagging, boosting, and stacking [SR18]. All relying on the wisdom of crowd components, these concepts revolutionized many domains of machine learning and signal processing. The most intuitive and common ensemble aggregation technique applied regularly in machine learning is the majority voting combination scheme. For a binary classification task it is known that when combining T independent classifiers it results in ensemble error rate that can be calculated by:

$$P(\text{Error}) = \sum_{\text{ceil}(t=\frac{T}{2})}^{T} \binom{T}{t} p_{\text{error}}^{t} (1 - p_{\text{error}})^{T-t}$$

where p_{error} is error rate of an individual classifier (also called base classifier). According to that error rate, equation Figure 1.1 shows the dependence of an ensemble error rate over the error rate of a base classifier.

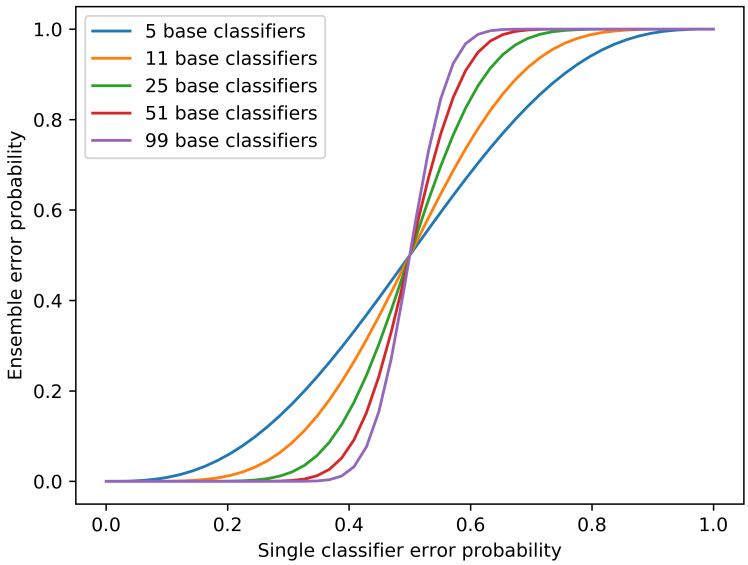
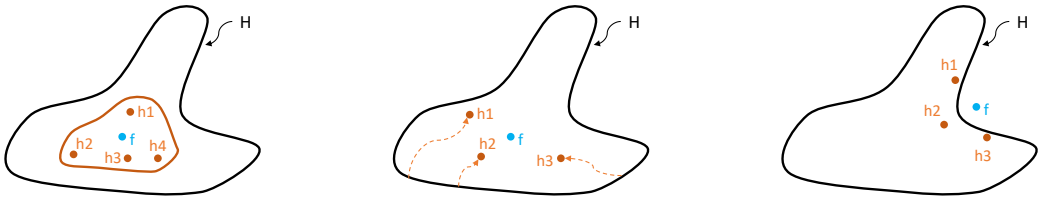


Figure 1.1: Ensemble error probability for binary classification problem and majority voting aggregation scheme

Note should be taken that the ensemble error rate differs depending on the number of base classifiers present in the ensemble approaching its minimum as the number of



(a) Statistical (b) Computational (c) Representational

Figure 1.2: Fundamental reasons for the superiority of an ensemble over a single classifier. H is the hypotheses space of the problem

base classifiers goes to infinity. Another important note is that all base classifiers are to be independent to achieve the performance depicted in Figure 1.1. Unfortunately, this is a requirement that is difficult to achieve in practice. On the other hand, it was shown that even limited base classifier diversity is able to provide plausible combination results [Don+20].

Consider three main problems why ensembles outperform a single model: computational, representational, and statistical [Die+02]. The statistical problem (Figure 1.2(a)) is a common problem for many applications with few data and large hypotheses space. In such cases, there are many hypotheses that provide comparable performance, however, the algorithm has to decide on a single one. The computational problem (Figure 1.2(b)) arises in cases when a learning algorithm is not able to guarantee convergence towards the best hypothesis. Both statistical and computational problems lead to the existence of the variance component of the model error. The representational problem (Figure 1.2(c)) relates to cases when the hypotheses space does not include any plausible approximation of the target classes. This leads to the bias component of the model error. For all these reasons, the performance of the learning algorithm might be sub-optimal or even unacceptable.

Despite the fact that the wisdom of crowds concept is already successfully used in classification tasks, it is still an exotic concept for clustering and pattern matching. Yet for classification scenarios, there are still many open questions including combining ensemble learning techniques with methods that combat class imbalance problems. For clustering ensembles, the scalability is considered to be the biggest challenge since often required clusterings association problem typically exhibits high computational complexity. At the same time, pattern matching tasks were hardly addressed with the help of the wisdom of crowds concept. For these reasons, the dissertation addresses clustering, classification, and pattern matching problems under the view of the wisdom of crowds concept.

1.2 Aims and Research Objectives

This doctoral thesis aims at developing approaches and methods based on the wisdom of crowds concept to solve clustering, classification, and pattern matching problems. The major research objectives concern highly relevant for signal processing and machine learning communities aspects include

- development of scalable algorithms to address clustering, classification and pattern matching problems
- assessment of the developed approaches in simulated and real-world scenarios
- interpretation of the results and demonstration of applicability of the proposed methods to various industrial domains

1.3 Summary of the Contributions

The original contributions of the doctoral thesis covering three main domains of signal processing and machine learning, namely, clustering, classification, and pattern matching are as follows.

1.3.1 Consensus Clustering

Following the wisdom of crowds concept, we develop two consensus clustering frameworks to perform accurate and stable object clustering in high dimensional feature space. First, by utilizing a stable group of objects concept called data fragment we enable a scalable consensus clustering framework. We address the drawbacks of the Hamming distance in co-occurrence-based consensus clustering methods proposing an expressive distance measure on data fragments and building a consensus function around this measure. We extensively evaluate the proposed framework on data collected from real problems as well as perform thorough simulations in order to assess its properties and theoretical performance in different regimes.

Second, we propose a novel consensus clustering approach to efficiently and rapidly combine multiple clustering solutions obtaining interpretable results. Based on the binary matrix factorization idea, the proposed consensus clustering approach is solved by

means of a recursive rank-one binary matrix approximation heuristic. Additionally, we introduce an effective initialization strategy for the heuristic. The proposed framework is evaluated against several synthetic and real data sets to provide a comprehensive view of it.

1.3.2 Dynamic Classifier Selection for Data Fusion

To address the classification problem from the wisdom of crowds point of view, we propose a dynamic selection framework to select and combine competent classifiers. We employ a multiple classifier system in order to solve a data fusion problem proposing a mechanism to establish a competence estimation and selection procedure. The main focus of the framework is to address data fusion problems from heterogeneous sources that exhibit high-class imbalance. We validate our proposed framework on two real data fusion remote sensing problems and identify that it is able to provide accurate classification results even with high-class imbalance. Additionally, we perform synthetic experiments to expand the exploration of the characteristics of the proposed method.

1.3.3 Real-time Pattern Matching

To find similar objects in an online stream of data or perform pattern matching tasks employing all available example templates we propose a real-time pattern matching approach. It allows for finding predefined patterns from dynamic and possibly distorted data streams. For that, we introduce a dynamic z-normalization scheme showing its equivalence to the traditional z-normalization with fixed optimal window size. We demonstrate that the proposed pattern matching approach provides high operational performance on both synthetically generated and real-world use cases outperforming the other state-of-the-art pattern matching methods, especially, when severe time distortions and sampling rate variance present.

1.4 Publications

During the period of doctoral candidacy, the following publications have been produced.

1.4.1 Candidacy-related Publications

The following publications concerning the candidacy matter have been released during the period of doctoral candidacy.

Internationally Refereed Journal Articles

S. Sukhanov, R. Wu, C. Debes, and A. M. Zoubir. "Dynamic pattern matching with multiple queries on large scale data streams". In: *Signal Processing* 171 (2020), p. 107402. URL: http://www.sciencedirect.com/science/article/pii/S0165168419304542.

N. Yokoya, P. Ghamisi, J. Xia, S. Sukhanov, R. Heremans, I. Tankoyeu, B. Bechtel, B. L. Saux, G. Moser, and D. Tuia. "Open Data for Global Multimodal Land Use Classification: Outcome of the 2017 IEEE GRSS Data Fusion Contest". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2018), pp. 1–15.

Internationally Refereed Conference and Workshop Papers

S. Sukhanov, C. Debes, and A. M. Zoubir. "Dynamic selection of classifiers for fusing imbalanced heterogeneous data". In: *Proc. 44th IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP).* 2019.

S. Sukhanov, D. Budylskii, I. Tankoyeu, R. Heremans, and C. Debes. "Fusion of Lidar, Hyperspectral and RGB Data for Urban Land Use and Land Cover Classification". In: *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium.* July 2018, pp. 3864–3867.

S. Sukhanov, C. Debes, and A. M. Zoubir. "Interpretable clustering ensembles using binary matrix factorization". In: *Proc. 43th IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP).* 2018.

S. Sukhanov, V. Gupta, C. Debes, and A. M. Zoubir. "Consensus clustering on data fragments". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Mar. 2017, pp. 4631–4635.

S. Sukhanov, I. Tankoyeu, J. Louradour, R. Heremans, D. Trofimova, and C. Debes. "Multilevel ensembling for local climate zones classification". In: *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS).* July 2017, pp. 1201–1204.

1.4.2 Other Publications

Several publications that have been produced during the period of doctoral candidacy on other matters.

Internationally Refereed Journal Articles

C. Debes, A. Merentitis, S. Sukhanov, M. Niessen, N. Frangiadakis, and A. Bauer. "Monitoring Activities of Daily Living in Smart Homes: Understanding human behavior". In: *IEEE Signal Processing Magazine* 33.2 (Mar. 2016), pp. 81–94.

Internationally Refereed Conference and Workshop Papers

S. Sukhanov, A. Merentitis, C. Debes, J. Hahn, and A. M. Zoubir. "Combining SVMs for Classification on Class Imbalanced Data". In: *2018 IEEE Statistical Signal Processing Workshop (SSP)*. June 2018, pp. 90–94.

1.5 Organization of the Thesis

The rest of this dissertation is structured with respect to its contributions as follows:

Chapter 2 introduces and reviews the main clustering methods, describes their main characteristics, highlighting limitations and disadvantages. Further, state-of-the-art consensus clustering methods are reviewed and the motivation for a need for a new approach is provided. According to the motivation, two novel consensus clustering approaches are proposed and evaluated on a set of synthetic and real-world datasets. An extensive study is conducted to explore the effect of parameters on clustering results. The chapter is closed with a discussion on the applicability of the proposed methods, their advantages, and their downsides.

Chapter 3 focuses on the classification-based data fusion problem. Motivated by two important remote sensing challenges, namely, local climate zones and land use and land

cover classification, we review the state-of-the-art data fusion techniques and multiple classifier systems. Next, a dynamic selection framework is proposed to address data fusion problems with heterogeneous data sources. By starting with multiple classifier systems fundamentals, we introduce a dynamic selection paradigm and discuss its advantages and limitations. Then, for this paradigm, a competence estimation and selection method to fuse competent classifiers is presented. We evaluate the method with synthetic and real datasets, demonstrating the applicability of the proposed framework to remote sensing applications. The chapter is concluded with discussions of results as well as potential applicability of the contribution.

Chapter 4 deals with similarity search in data streams. Starting with a state-of-the-art review in similarity search and time series pattern matching we then formulate the pattern matching problem, extending it to account for multiple template examples, and review the time-series averaging concept as well as the traditional fixed-length window z-normalization method. Further, we propose a dynamic normalization approach that allows bringing streaming signal subsequences to the scale of the query template. We further develop a pattern matching approach utilizing the proposed normalization mechanism and extend it for the case when multiple examples of a query template are available. After that, we evaluate the proposed pattern discovery method on multiple synthetic and real-world datasets and report on its operational performance.

Finally, in **Chapter 5**, we conclude the thesis with a review of the presented contributions as well as provide an outlook for future work.

Chapter 2

Clustering of multi-dimensional data with consensus algorithms

Over the last few decades, data clustering remains an important challenge in many fields of signal processing and data analysis. Being an unsupervised learning technique clustering is usually employed to identify the underlining structure of data distributions by grouping together similar objects into structures (or clusters). The objects are usually represented by multidimensional vectors where each dimension characterize some property or feature of the object. Thus, the main task of clustering is the partitioning of a set of objects into clusters, so that objects of the same cluster share similar properties.

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ be a set of N objects, where $\mathbf{x}_i$ is a vector in a d-dimensional feature space $\mathbb{R}^d$: $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$. Given $\mathbf{X}$, the objective of clustering is to find a partition (clustering) $C = \{C_1, C_2, ..., C_K\}$ of K clusters, such that similar vectors are assigned to the same cluster and dissimilar to different ones.

Due to the fact that similarity between objects might be expressed by various measures and strongly depends on application, the notion of a cluster is, generally, subjective. Especially, when clustering high dimensional data, such similarity might be ambiguous due to data scarcity and curse of dimensionality [Bel03].

The subjectivity of clustering results exist since in many practical problems, hidden clusters might have different sizes, shapes, level of separation, etc. Identifying clusters with these assumptions is considered to be a challenging task even nowadays. The main reason for that is the fact that most of the existing clustering methods usually optimize one objective with its particular assumptions on cluster properties and data.

Figure 2.1 demonstrates four data objects labeled as A, B, C and D with their three attributes each: the shape, the color and the sign inside. Obviously, there are multiple ways to cluster these objects and there is no any definite preference to do so. Depending on the assumptions imposed and the relative importance of attributes one could co-cluster objects A and B, D and C or A and D.

Generally, if one opts to calculate exhaustively the amount of possible ways to cluster N

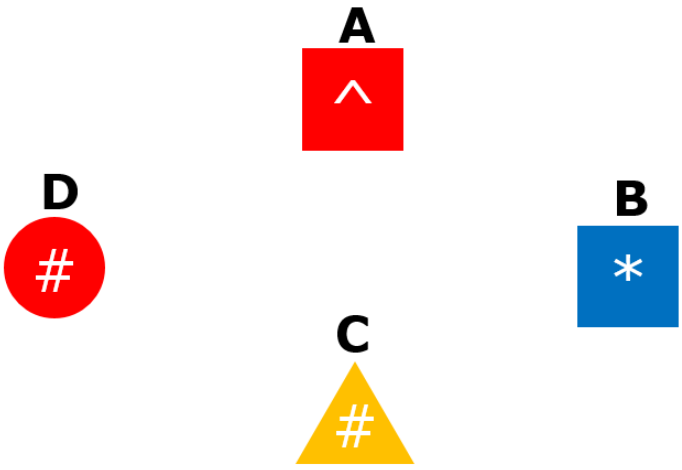


Figure 2.1: Example of four data objects labeled as A, B, C and D with their attributes: the shape, the color and the sign inside the object

objects into K clusters it ends up with the Stirling numbers of the second kind [BH91]:

$$S(N,K) = \frac{1}{K!}\sum_{j=0}^{K}(-1)^{K-j}\binom{K}{j}j^n \approx \frac{K^N}{K!}$$

that, obviously, would require enormous computational effort when the number of objects N increases even to just hundreds. That is why a fast and accurate technique is crucial in order to perform reliable clustering of multidimensional data vectors.

2.1 Clustering techniques overview

Over the past years, there have been many algorithms proposed to perform data clustering task [JMF99]. In the sequel, we review the main fundamental clustering algorithms that, in turn, became basis for many other advanced and scalable clustering techniques.

2.1.1 Partitional clustering

Iterative partitional algorithms iteratively assign initial cluster membership according to a specific notion of a cluster. One of the oldest though most successful and famous algorithms from this group is K-means [Mac67] that by utilizing Euclidean distance between data objects and cluster centroids iteratively reassigns cluster memberships

and updates the location of centroids in Euclidean space. K-means is also known as a prototype-based clustering method as it utilizes a concept of centroids to be a prototype of a cluster. The procedure of grouping data points into K clusters (where K is a positive integer that is specified beforehand) is summarized in Table 2.1.

K-means algorithm
1: Initialize K centroids to be K random data points
2: **repeat**
3: Assign points to their closest centroids
4: According to assignment of points to centroids refine centroids position
5: **until** centroids are not changing or particular number of iterations exceeded.

Table 2.1: Steps of K-means clustering algorithm with random initialization

The algorithm in Table 2.1 is formulated according to an optimization task that is defined to minimize the sum of square errors (SSE) represented by the distance between data points and their corresponding cluster centroids. Thus, the objective of k-means with K clusters is formally defined as to minimize:

$$J = SSE = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mathbf{c}_i||^2 \tag{2.1}$$

where $||\mathbf{x} - \mathbf{c}_i||^2$ is the squared Euclidean distance between data point $\mathbf{x}$ and cluster center $\mathbf{c}_i$. Mean operator over all points assigned to a cluster defines a centroid minimizing SSE:

$$\mathbf{c}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in C_i} x \tag{2.2}$$

where m_i is the number of objects in the i^{th} cluster.

K-means is computationally attractive for large scale problems as its is usually solved by efficient heuristic algorithms that exhibit time complexity $\mathcal{O}((n+K)d)$ and converge quickly to a local minimum. However, K-means has three substantial drawbacks: the number of clusters K is to be provided beforehand; the convergence is sensitive to initialization resulting in different results for different runs; clusters have globular shape due to the notion of centriods that make it practically useless for non-globular clusters (see Figure 2.2) [JMF99].

(a) Ground truth clustering (b) results of k-means with K=7

Figure 2.2: Non-globular cluster shapes limit k-means from identifying them

2.1.2 Hierarchical clustering

Representing data with the help of a dendrogram or a tree diagram lead to the emergence of hierarchical clustering algorithms [RM05]. The main idea behind hierarchical clustering algorithms is that each data point is considered as a separate cluster while a set of nested clusters is organized in a tree structure. In this structure, every node is the union of its children whereas the root contains all the data points. Thus, a node might be a subcluster or a singleton cluster i.e. one data point cluster. Figure 2.3 provides an example where nodes $p2$ and $p3$ are a subcluster of a cluster $\{p2, p3, p4\}$ while $p1$ is a singleton cluster.

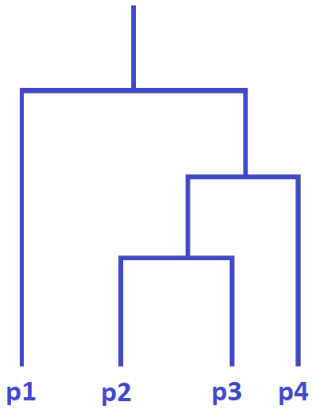
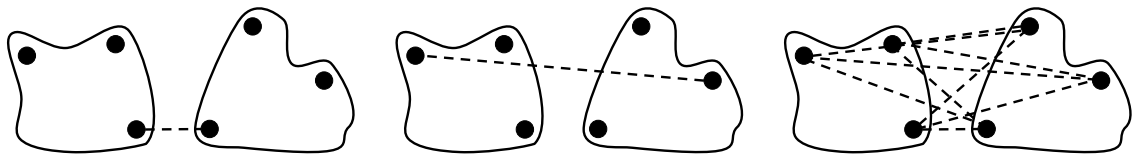


Figure 2.3: Dendrogam

Hierarchical clustering can be considered as a sequence of partitional clusterings that is obtained by cutting the dendrogram at a particular level. There are two ways in which relationship between a cluster and its subclusters and merging/splitting are defined in dendrogram: bottom-up or agglomerative clustering and top-down or divisive clustering.



(a) Single Linkage (b) Complete Linkage (c) Average Linkage

Figure 2.4: Different types of linkages

Agglomerative hierarchical clustering creates a hierarchy by linking similar objects and clusters on different levels of that hierarchy. By beginning with a single object in a separate cluster agglomerative clustering algorithms merge clusters together at each step according to a predefined linkage function that is used: single linkage, complete linkage, average linkage and Ward linkage (Figure 2.4).

Two closest points in different clusters define single linkage (Figure 2.4(a)) that is often used for non-elliptical cluster shapes. However, such linkage is sensitive to outliers and noise. According to single linkage, the distance $D(A, B)$ between clusters A and B is defined as:

$$D(A, B) = \min_{\mathbf{x} \in A, \mathbf{y} \in B} d(\mathbf{x}, \mathbf{y}). \tag{2.3}$$

Two furthest points in different clusters define complete linkage (Figure 2.4(b)) that is less prone to noise and outliers than single linkage, though favors globular shapes and might disconnect large clusters. According to complete linkage, the distance $D(A, B)$ between clusters A and B is defined as:

$$D(A, B) = \max_{\mathbf{x} \in A, \mathbf{y} \in B} d(\mathbf{x}, \mathbf{y}). \tag{2.4}$$

Average linkage is defined as the average distance between all pairs of points coming from different clusters (Figure 2.4(c)). According to complete linkage, the distance $D(A, B)$ between clusters A and B is defined as:

$$D(A, B) = \frac{1}{\big|A\big| \cdot \big|B\big|} \sum_{\mathbf{x} \in A} \sum_{\mathbf{y} \in B} d(\mathbf{x}, \mathbf{y}). \tag{2.5}$$

For Ward linkage clusters are represented by their centroids and establish distance between them according to the increase in the SSE by merging a pair of clusters. Ward linkage suffers from inversion problem in which two merging clusters might be more similar than another two on the previous step. For other linkages the distance between a pair of clusters increases monotonically. For Ward linkage the initial cluster distances

between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ are defined as:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \left|\left|\mathbf{x}_i - \mathbf{x}_j\right|\right|^2. \tag{2.6}$$

According to selected linkage type agglomerative clustering constructs a distance matrix that is further used for dendrogram creation. The steps of agglomerative clustering are summarized in Table 2.2:

Agglomerative hierarchical clustering algorithm
1: Construct distance matrix, if necessary
2: **repeat**
3: Merge two closest clusters
4: Update the distance matrix to update the distances between the new cluster resulting from merging and the rest
5: **until** single cluster remains.

Table 2.2: Steps of agglomerative clustering algorithm

The main limitation of agglomerative hierarchical clustering algorithms is its high computational complexity limiting their practical applicability to large data sets. The time complexity of computing distance matrix is $\mathcal{O}(N^2)$ while the clustering procedure (provided storage of the distances between clusters in a sorted list) results in $\mathcal{O}(N^2 \log N)$. Moreover, these algorithms are prone to noise and outliers since the decisions on every hierarchy are fixed and there is no mechanism to refine resulting clusters. To address complexity limitations Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [ZRL96] and CURE [GRS98] clustering approaches were later proposed.

Divisive hierarchical clustering creates a tree diagram by iteratively dividing objects into clusters until the desired number of clusters is achieved. Many methods utilizing this concept were proposed including Principal Direction Divisive Partitioning (PDDP) [Bol98] that is computationally efficient and allows to deal with large data bases.

Generally, hierarchical clustering suffers from several limitations: the lack of globally optimized objective function; inability to handle clusters of different sizes; fixed merging rules that prevent a local optimization criteria from becoming a global one.

2.1.3 Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)

To overcome scalaibility issue and be able to reverse fixed merging decisions BIRCH [ZRL96] algorithm was introduced. The main idea of BIRCH is to minimize number of data scans and as the result running time of clustering procedure. There are several special features in BIRCH: clustering decisions are performed without scanning the entire data set; it removes outliers by exploiting the non uniformity of the data; it creates clustering feature tree - a dendrogram-based compact representation of data to calculate the intra-cluster distances. The idea behind the clustering feature tree is to represent a group of objects by a three-tuple (N, Linear Sum (LS), Square Sum (SS)), where N is the number of data points in a cluster, LS is defined as:

$$\mathrm{LS} = \sum_{i=1}^{N} \mathbf{x}_i \tag{2.7}$$

and SS is defined as:

$$\mathrm{SS} = \sum_{i=1}^{N} \mathbf{x}_i^2 \tag{2.8}$$

By operation with these three-tuple objects BIRCH clustering algorithm steps are summarized in Table 2.3. The major advantage of BIRCH is its scalability comparing

BIRCH clustering algorithm
1: Scan the entire data set and construct an initial clustering feature tree
2: Construct a smaller clustering feature tree by shrinking to a desirable length
3: Perform global clustering by means of a clustering algorithm e.g. k-means or hierarchical clustering on clustering feature entries
4: Refine resulting clusters to fix cases where equally valued data points got assigned to different leaf entries in clustering feature trees

Table 2.3: Steps of BIRCH clustering algorithm

to hierarchical clustering methods and ability to handle outliers.

2.1.4 Density-based clustering methods

Density based methods are based on density search algorithms [Eve+11] that find populated areas in a metric space. Density Based Spatial Clustering of Applications with Noise (DBSCAN) [Est+96] is the most popular algorithm from this group of

methods. By employing Euclidean distance as a metric space, DBSCAN estimates the density in d-dimensional space (where d is the number of features that every objects has). The main assumption that the algorithm has is that all the objects that belong to the same cluster must be mutually density-connected. That might often overestimate the number of resulting clusters, especially considering not-trivial parameter setting of this algorithm [PK06]. DBSCAN algorithm is summarized in Table 2.4.

DBSCAN algorithm
1: Label entire data set as core, border or noise points
2: Exclude noise points
3: Set an edge between all core points that are within a predefined distance of each other
4: Place each group of connected core points into a separate cluster
5: Assign each border point to one of the cluster of its associated core points

Table 2.4: Steps of DBSCAN clustering algorithm

One of the advantages of DBSCAN is that it does not require number of clusters to be known in advance but determines it automatically. Another advantage is that it is able to handle clusters of arbitrary shapes and sizes, however, typical density estimation challenges are applied for this method in high dimensional feature space. The time complexity of the density based clustering is $\mathcal{O}(N \times T_{\text{neigh}})$, where T_{neigh} is the time to perform search in predefined neighborhood. The worst case complexity to find neighbors is $\mathcal{O}(N^2)$.

2.1.5 Spectral clustering

Spectral clustering [NJW02] utilizes a graph partitioning methodology in order to perform clustering of a dataset. Its main advantage is that it makes no direct assumptions on cluster shapes and sizes being suitable for clustering complex data patterns. However, similarity measure utilized in spectral clustering might impact those assumptions. In spectral clustering a weighted graph $G = (V, W)$ is constructed where the data points are the nodes V and connections between them are the weighted edges W_{ij}. The weights are computed according to a local symmetric and non-negative similarity measure that is usually represented by a Gaussian kernel that is parameterized by width σ:

$$W_{i,j} = \exp\Big(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\Big) \tag{2.9}$$

where $||\cdot||$ denotes Euclidean distance.

There are several possible methods available to construct a graph that captures relationship between data points:

- Fully connected graph: vertices which have any non-zero similarities are connected;
- r-neighborhood graph: a vertex is connected to all other if they are located within some radius r, where r is a real number parameter characterizing local structure of the dataset;
- k-nearest neighbor graph: a vertex is connected to its k-nearest neighboring vertices, where k is an integer parameter characterizing local structure of the dataset.

With the help of one of the objective functions (e.g. MinCut, RatioCut, NCut or MinMaxCutConstructed) [ZP05; YS03; DGK04] the resulting graph is then partitioned into two disjoint sets of connected vertices A and B such that:

$$V = A \cup B \tag{2.10}$$

and

$$\phi = A \cap B. \tag{2.11}$$

Degree d_i is then defined for every vertex i as:

$$d_i = \sum_{j=1}^{n} W_{ij} \tag{2.12}$$

forming a degree matrix D which is a diagonal matrix with zero off-diagonal elements. Next, the Laplacian matrix L is formed as:

$$L = D - W. \tag{2.13}$$

After that eigenvectors of L are computed and clustered with i.e. k-means algorithm in order to get final solution.

2.2 Clustering Challenges

Practically, the majority of clustering techniques are not able to provide satisfactory results due to their algorithmic limitations and various issues with data distributions.

The main intrinsic challenges of clustering techniques are, typically, large parameter set and their own assumptions on resulting clusters [FVH06; JMF99]. Different sets of parameters of the same clustering algorithm might result into completely different output: from the one that make sense to meaninglessly identified clusters [FM07]. Moreover, the outcome of multiple runs of the same algorithm, i.e. k-means, might not be consistent due to strong dependence on the initialization (by applying several of the considered clustering algorithms to the same dataset one obtains, typically, different clustering results) [JMF99]. Various algorithmic assumptions might lead to cluster size and shape variations, inability to detect a cluster or improper objects grouping. The reason for that is primarily intrinsic optimization criteria that is different for various clustering algorithms. Evaluating clustering results and finding out which one (from many) is correct is considered to be an extremely challenging task since, generally, there is no ground truth available while cluster validity indices are often ambiguous [SRS08; JMF99; JL05]. Thus, without having any preknowledge about eventual results it is typically assumed that all the solutions are equally plausible.

Issues with data distributions include possible presence of outliers that might make results inaccurate; curse of dimensionality [Bel03] that often cause clusters to exist in different subspaces; density variability that might hinder discovery of low-densed clusters and other. All these data distribution issues are typically leading to uncertainty in clustering solutions and, as the result, wrong clustering output.

The ambiguity when choosing one out of several clustering results lead to the concept of combination that is considered further in this Chapter.

2.3 Consensus Clustering Fundamentals

Consensus clustering has emerged to be a powerful tool for solving practical data clustering problems [SG03] proving to achieve more stable, quality and accurate clustering results than by applying a single clustering algorithm [GF08]. Consensus clustering often allows obtaining novel solutions [Chr11; Top+04] that are not feasible by any single clustering methods.

Motivated by the success of supervised ensemble learning techniques [Bre96] consensus clustering receives multiple clustering solutions (also called an ensemble) and combines them to obtain a single final one. As a result, consensus clustering usually provides a more accurate and stable output [GF08]. Moreover, consensus clustering also allows obtaining novel solutions that are not achievable by any single clustering method as shown empirically [SG03; Chr11] and theoretically [Top+04] in the past.

In various works on consensus clustering, several properties were defined for a consensus clustering algorithm to be successful. The main of them are:

- Novelty: it finds solutions unreachable by any single clustering algorithm;
- Consistency: its results are similar to single clustering algorithms' results that are used for combination;
- Effectiveness: its result provides better average performance than any single clustering algorithm;
- Stability: its result is robust to noise and outliers.

Despite the fact that these properties are expected from every consensus clustering algorithm it is usually challenging to achieve all of them simultaneously [PS11]. Therefore, most of the existing algorithms possess a subset of desired properties.

Given a set of objects, consensus clustering methods consist of two main steps: 1) Generation, in which a set of diverse clusterings is produced and 2) Consensus, where generated clusterings (or some of them) are combined. This combination is usually done without any access to original data on which these clusterings were generated. Finding a proper consensus function that accurately combines given partitions is usually considered to be the biggest challenge in consensus clustering research [TJP05; Iam+11] and is also one of the contributions of this thesis.

On Figure 2.5 a flow chart of consensus clustering process is depicted. Typically, at first, a set of diverse and quality clustering solutions are generated. Second, these clusterings are combined according to a predefined consensus function that outputs a single clustering solution.

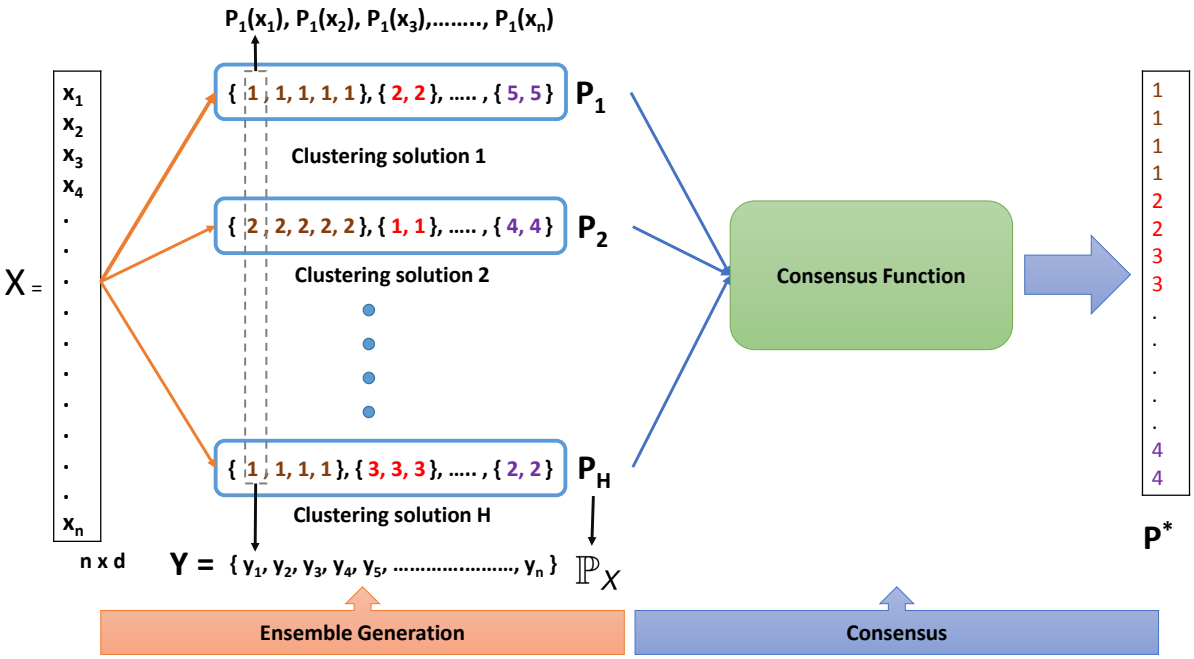


Figure 2.5: Consensus clustering flow chart. Multiple clustering solutions are getting generated and then getting combined according to a predefined consensus function resulting into a single solution

2.4 Problem formulation

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ be a set of N objects, where $\mathbf{x}_i$ is a vector in a d-dimensional feature space $\mathbb{R}^d$: $\mathbf{x}_1, \ldots, \mathbf{x}_N \in R^d$. $\mathbb{P} = \{P_1, P_2, \ldots, P_H\}$ is a set of H partitions (or clusterings) of $\mathbf{X}$, where each $P_h = \{C_1^h, C_2^h, \ldots, C_{K_h}^h\}$ is a single partition of $\mathbf{X}$ with K_h clusters satisfying

- $C_j^h \neq \emptyset, \forall\ j = 1, \ldots, K_h$
- $C_j^h \cap C_p^h = \emptyset, \forall\ j \neq p$
- $\cup_{k=1}^{K_h} C_k^h = \mathbf{X}$.

For each $\mathbf{x}_i$, we define a H-dimensional label vector $\mathbf{y}_i$:

$$\mathbf{y}_i = [P_1(\mathbf{x}_i), P_2(\mathbf{x}_i), \ldots, P_H(\mathbf{x}_i)] \tag{2.14}$$

where $P_h(\mathbf{x}_i)$ is the cluster label of $\mathbf{x}_i$ in partition P_h. Since every clustering P_h assigns symbolic labels to objects $\mathbf{x}_i$ the vectors $\mathbf{y}_i$ consist of categorical values. In addition,

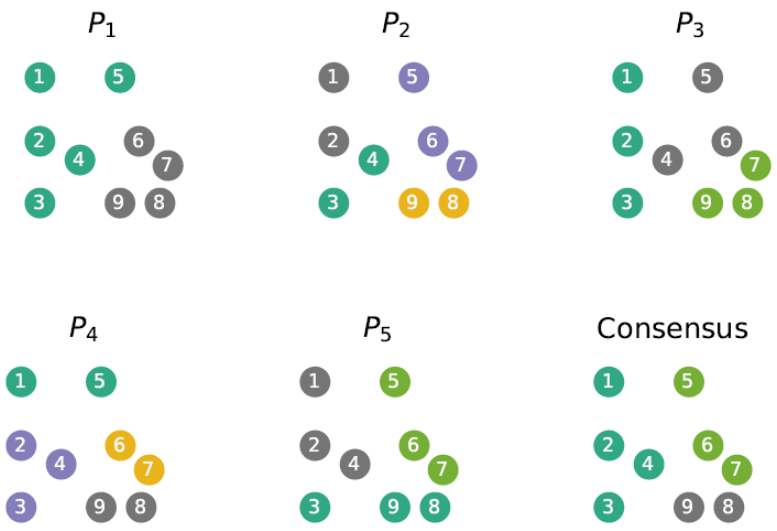


Figure 2.6: Example of five distinct partitions of a toy dataset and a consensus solution

all vectors $\mathbf{y}_i$ are forming a matrix

$$\mathbf{Y} = [\mathbf{y}_1^\top, \mathbf{y}_2^\top, \ldots, \mathbf{y}_N^\top]^\top \tag{2.15}$$

that represents data objects in the ensemble label space. Figure 2.6 depicts an example of a clustering ensemble over $N = 9$ objects clustered differently $H = 5$ times (color encodes cluster label). The consensus solution is depicted on the same Figure 2.6 and has $K = 3$ resulting clusters. For the object co-occurrence-based methods [SG03] the consensus function CF maps $\mathbb{P}$ to a consensus solution as:

$$\text{CF} : \{P_h | h \in \{1, \ldots, H\}\} \rightarrow P^* \tag{2.16}$$

The consensus solution for median partition-based problems provides the maximum cumulative similarity with respect to all clusterings in the ensemble:

$$P^* = \underset{P \in \mathcal{P}_x}{\text{argmax}} \sum_{h=1}^{H} S(P, P_h) \tag{2.17}$$

where S is a similarity measure between partitions, $\mathcal{P}_x$ is a search space with all possible clusterings of $\mathbf{Y}$. The median partition-based problem was proven to be NP-hard for the Mirkin distance (symmetric difference distance) [PS11] while the object co-occurrence-based formulation is not enough rigorous. Moreover, both problems are formulated in a way that does not make use of dominant objects (or centroids) in the ensemble label space and most of the consensus functions provide final labels only.

2.4.1 Partition Generation

Partition generation [Hua+17] is the step where partitions to combine are being created. During this step diverse and quality partitions of data are produced in order to be then combined at the aggregation step. To ensure the quality and diversity of the generated partitions an additional selection phase might be introduced where the most accurate according to some criterion partitions are selected for further consensus.

Random projections [BV05; Wan+15] that is a dimensionality reduction technique is the most popular partition generation approach: random projections of a dataset are generated following by a clustering algorithm [PHL04]. As a result, such partitions are then used as an input of a clustering approach. In the following, we provide details on random projection-based dimensionality reduction often used to create diverse partitions.

Dimensionality reduction is a function that maps data points from high to a low dimensional space, $\mu : \mathbb{R}^d \to \mathbb{R}^{d'}$, where $d < d'$. In case μ is a deterministic mapping, the Euclidean distances between data points are not guaranteed to be preserved that might affect the performance of methods that depend on distances between data points (most of clustering algorithms in such case would suffer from that). To the contrary, randomized embeddings, according to the Johnson-Lindenstrauss (JL) lemma [BV05], provide possibility to embed data points from high into low dimensional space nearly preserving Euclidean distances between them i.e. with low distortion defined as

$$\frac{||\mu(\mathbf{x}_i) - \mu(\mathbf{x}_j)||}{||\mathbf{x}_i - \mathbf{x}_j||} \tag{2.18}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the data points. Specifically, given $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ a set of N objects, where $\mathbf{x}_i$ is a vector in a d-dimensional feature space there exist $1+\epsilon$-distortion embedding into $\mathbb{R}^{d'}$ so that:

$$q = (\frac{c \log N}{\epsilon^2}) \tag{2.19}$$

where c is a constant. The embedding is represented by a random $d \times d'$ matrix D with random orthonormal vectors or embeddings that are represented by other matrices [Hua+17]. Random subspace (RS) method is represented by

$$D = \sqrt{\frac{d}{d'}}(r_{ij}) \tag{2.20}$$

where r_{ij} are randomly chosen with entries $\{0, 1\}$ with exactly single "1" per row and at most single "1" per column and formally, does not satisfy JL lemma but practically

shown to provide low distortions for high dimensional data [BV06; EV13]. RS is performed as

$$X' = XD \tag{2.21}$$

According to that, it is possible to obtain low distortion embeddings by performing random projections from $\mathbb{R}^d$ into $\mathbb{R}^{d'}$. According to Equation 2.21 H projected datasets $\mathcal{X} = \{X_1, X_2, \ldots, X_H\}$ are generated to further use them for clustering.

2.5 Consensus Clustering functions

In the state of the art, there are two main groups of consensus functions: median partition [PS11] and object co-occurrence [Lou+13]. Approaches from the first group are searching for a consensus function that is a solution of an optimization problem. Here the objective is to maximize the sum of similarities between the median clustering and all clusterings in the given ensemble. The choice of similarity measure between partitions is the key challenge that is not yet fully addressed [PS11]. For example, for Mirkin-based distance functions it was proven that the solution of the median partition problem is $\mathcal{NP}$-hard while for other distances there was no in depth analysis done in the literature [PS11; AW19a]. For this reason, there is no universal answer on the approach which should be used to accurately solve practical large-scale consensus clustering problems using median partition-based methods. The second set of methods is based on object co-occurrence which operates on pairs of objects and analyzes whether two objects belong to the same cluster in every partition or not [FJ05]. Despite its solid operational performance the main drawback of co-occurrence based methods is their high computational and memory complexity (usually not less than $\mathcal{O}(N^2)$) and the fact that co-association matrices are not expressive enough to accurately perform aggregation on them (especially when there are not too many partitions or they are of limited diversity) [PS11; Wan11]. In the following, we provide an overview of state-of-the-art of consensus clustering algorithms.

2.5.1 Relabeling and voting

Relabeling and voting based methods are considered to be one of the first consensus clustering algorithms proposed in the literature [PS11]. Consisting of two major steps, these methods, at first, solve the labeling correspondence problem and then apply a voting process to obtain final result. The label corresponding process is usually the most challenging step since there is no correspondence established between partitions

of different clustering algorithms. The labels are symbolic class representations and might vary even for multiple runs of the same clustering algorithm. Methods that belong to this group are Plurality Voting [FB03], Voting-Merging [AW19b], Voting for fuzzy clusterings [DWH02], voting Active Clusters [TA08], Cumulative Voting [AK08], method proposed by Ayad and Kamel [AK10], Zhou and Tang [ZT06], Gordon and Vichi [GV01], Dudoit and Fridlyand [DF03]. Due to the difficulty of solving labeling correspondence problem exactly, the solution of the algorithms of this type are able to typically deal with the partitions with the same number of clusters. Moreover, the computational cost is often $\mathcal{O}(N^3)$ since it usually involves Hungarian algorithm [JV86] in order to solve labeling correspondence problem.

2.5.1.1 Connected Triple Based Similarity

Connected Triple Based Similarity [NG10] operates based on the idea that if two data points share a link with another data point then it reflects the similarity between those two data points. According to Figure 2.7, data points $\mathbf{x}_1$ and $\mathbf{x}_2$ are similar since they are co-clustered in P_2 and P_3, though, they are dissimilar with respect to partition P_1. Thus, cluster C_1^1 and C_2^1 are similar as they have two connected-triples in which cluster C_1^2 and C_1^3 are centers of the triples.

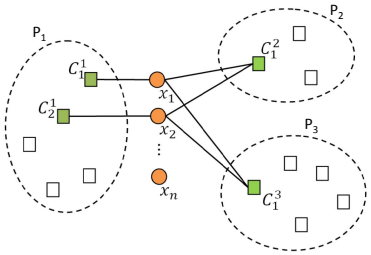


Figure 2.7: Connected Triple Based Similarity. From [NG10]

Besides identifying connected-triples, there is also similarity between clusters in terms of shared data that is taken into account. According to that, the Weighted Connected-triple (WCT) algorithm is established. This algorithm constructs a weighted graph $G = (V, W)$, with V being the set of vertices each representing a cluster in $\mathbb{P}$ and W being a set of weighted edges between clusters. Every edge that connects clusters $C_i, C_j \in V$ receives a weight w_{ij} and is calculated as proportion of the overlapping members:

$$w_{ij} = \frac{|\mathbf{X}_{C_i} \cap \mathbf{X}_{C_j}|}{|\mathbf{X}_{C_i} \cup \mathbf{X}_{C_j}|} \tag{2.22}$$

where $\mathbf{X}_{C_i} \in \mathbf{X}$ denotes objects assigned to cluster C_i. Contrary to counting number of triples, a minimum weight of the two involving edges is taken as triple. Additionally, the number of connected-triples between clusters $C_i, C_j \in V$ whose common neighbor is cluster $C_k \in V$ is considered:

$$\mathrm{WCT}_{ij}^{k} = \min(w_{ik}, w_{jk}) \tag{2.23}$$

The number of $q(1 \leq q < \infty)$ triples between two clusters C_i and C_j is then:

$$\mathrm{WCT}_{ij} = \sum_{k=1}^{q} \mathrm{WCT}_{ij}^{k}. \tag{2.24}$$

The similarity between two clusters C_i and C_j is defined as:

$$\mathrm{Sim}^{\mathrm{WCT}}(i,j) = \frac{\mathrm{WCT}_{ij}}{\mathrm{WCT}_{\max}} \tag{2.25}$$

where, $\mathrm{WCT}_{\max}$ is the maximum WCT_{ij} value of any pair of clusters within $\mathbb{P}$. By employing such cluster-oriented set up in order to enhance the quality of the typical similarity matrix, the similarity between two objects $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$ is then calculated as:

$$S_m(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1, & \text{if} C(\mathbf{x}_i) = C(\mathbf{x}_j) \\ \mathrm{Sim}^{\mathrm{WCT}}(C(\mathbf{x}_i), C(\mathbf{x}_j)) \times \mathrm{DC}, & \text{otherwise} \end{cases} \tag{2.26}$$

where $\mathrm{DC} \in (0, 1]$ is a constant decay factor representing confidence level of accepting a pair of non-identical data points as being similar. By employing similarity between objects $\mathbf{x}_i$ and $\mathbf{x}_j \in \mathbf{X}$, a connected-triple based similarity (CTS) matrix is created, where entries are calculated as

$$\mathrm{CTS}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{M} \sum_{m=1}^{M} S_m(\mathbf{x}_i, \mathbf{x}_j) \tag{2.27}$$

By treating CTS as similarity matrix, an agglomerative clustering is then performed to obtain final solution.

2.5.1.2 SimRank Based Similarity (SRS)

The basic idea of SimRank Based Similarity (SRS) [NG10] is the fact that neighbors are similar if their neighbors are similar as well. In SRC the dataset is represented as a graph with vertices and edges, where edges between vertices indicate similarity. To reveal hidden relations between data points and clusters a bipartite graph representation is used (Figure 2.8).

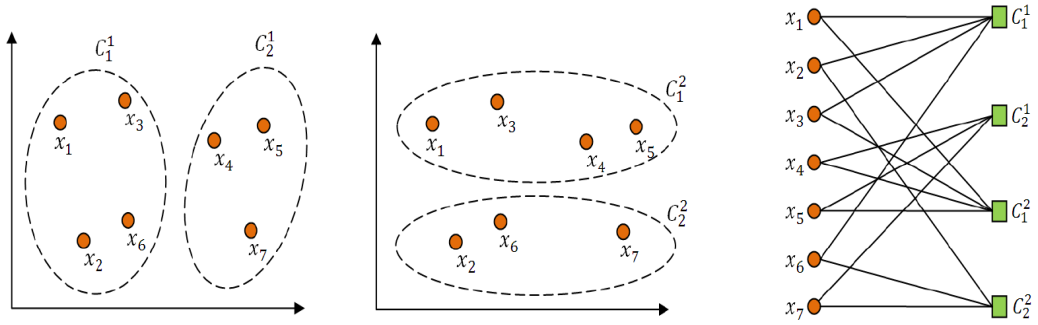


Figure 2.8: SimRank Based Similarity bipartite representation. From [NG10]

To create a graph, a $N \times N$ matrix is built where each entry indicates similarity between a pair of objects or clusters. The similarity between two objects is defined as the average similarity between clusters they belong to while similarity between two clusters is defined as the average similarity between their objects. Then, the similarity measure between a pair of vertices is computed through the iterative refinement process.

2.5.2 Co-association matrix based methods

In co-association matrix-based (an indicator matrix reflecting if a pair of data points are co-clustered) methods the label correspondence problem is avoided. Instead, it has an intermediate step of mapping the partitions into co-association matrix where each cell counts the number of partitions in which a pair of objects $\mathbf{x}_i$ and $\mathbf{x}_j$ is co-clustered. Co-association matrix CA is then defined as:

$$\mathrm{CA}_{i,j} = \frac{1}{H} \sum_{h=1}^{H} \delta(P_t(\mathbf{x}_i), P_t(\mathbf{x}_j)) \tag{2.28}$$

where $P_h(\mathbf{x}_j)$ is the cluster label of $\mathbf{x}_i$ in P_h and

$$\delta(a,b) = \begin{cases} 1, & \text{if} \quad a = b \\ 0, & \text{otherwise} \end{cases} \tag{2.29}$$

Large CA entries indicate frequent co-clustering occurrences of a pair of objects allowing to treat it as similarity matrix between objects. By considering co-association matrix as similarity matrix, a clustering algorithm is applied to obtain final partition. In [Fre01] a threshold is used to produce consensus partition. In [FJ05] co-association

matrix is considered as an adjacency matrix of a graph in which the minimum spanning tree is to be found. Similarly, in Single Link [JMF99] a dendrogram (a tree diagram representing clustering results) is created and then cut according to a threshold. Likewise, a k-cluster lifetime [Fre01] is proposed to obtain K clusters by using a range of thresholds. According to that, other hierarchical clustering algorithms [Li+07] can be used to cut the dendrogram including Complete-Link and Average-Link [JMF99]. In [IBG08] two novel similarity matrix concepts were introduced where the neighborhood of a pair of objects is considered in order to define similarity between these two objects. Another two similar ideas were proposed in [VR09] and in [WYZ09]. The main advantage of co-association matrix-based methods is the ability to perform straightforward implementation of them. This, however, goes with the cost of high computational complexity ($\mathcal{O}(N^2)$) and the choice of hierarchical clustering algorithm to obtain final solution.

2.5.3 Hypergraph partitioning methods

Hypergraph partitioning methods have inspired consensus clustering research to use graph and hypergraph based approaches to find consensus solution. The main idea of those approaches is by having multiple partitions to build a (hyper)graph and then find a best cut. In [SG03], the final consensus partition is defined as the one that shares the most information with other partitions. By employing Normalized Mutual Information (NMI) [Kno+06] the measure of information shared between two clusterings is established leading to three heuristics: Cluster-based Similarity Partitioning Algorithm (CSPA), HyperGraph Partitioning Algorithm (HGPA) and Meta-Clustering Algorithm (MCLA). Introduced in [FB04] Hybrid Bipartite Graph Formulation (HBGF) employs METIS [KK98b] algorithm in order to partition the graph created by modeling objects and clusters together on the same graph. In [AWJ10] a method based on random walker strategy was proposed in order to cut the graph. Generally, graph and hypergraph based approaches are widely used as consensus clustering methods, however, due to the lack of rigorous problem formulation still stay non optimal in many scenarios.

2.5.3.1 Cluster-based Similarity Partitioning Algorithm (CSPA)

The main idea behind CSPA is to create H $N \times N$ similarity matrices, average them and then partition resulting average matrix. The entries of similarity matrices for each clustering have either 0 (if objects are not assigned to the same cluster) and 1

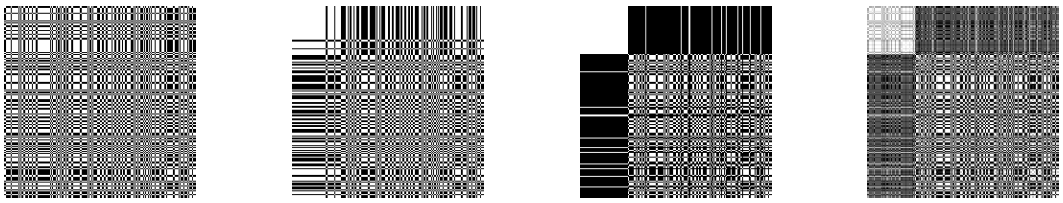
otherwise:

$$S_{P_h}(i,j) = \begin{cases} 1, & if \quad P_h(\mathbf{x}_i) = P_h(\mathbf{x}_j) \\ 0, & \text{otherwise} \end{cases} \tag{2.30}$$

The entry-wise average of H similarity matrices is defined as

$$S = \frac{1}{H} P \cdot P'. \tag{2.31}$$

Similarity matrix in CSPA can be considered as the adjacency matrix of a fully connected graph with nodes being elements of X while edges between a pair of nodes possess weights equal to the number of times respective points are co-clustered. Consensus partition is then obtained by applying graph partitioning algorithm METIS [KK98a] on similarity matrix (Figure 2.9). Though CSPA is straightforward and intuitive ap-



(a) Partition 1 (b) Partition 2 (c) Partition 3 (d) Similarity matrix

Figure 2.9: Co-association matrices of three clusterings and an entry-wise average of those matrices

proach it has space and time complexity limitations. Its worst case time complexity is $\mathcal{O}(N^2KH)$ that limits its applicability to large-scale datasts.

2.5.3.2 Meta-Clustering Algorithm (MCLA)

MCLA is a consensus algorithm that performs clustering of clusters and outputs confidence estimates for cluster membership of each data point. The main principle of MCLA is to form a matrix of similarities between clusters by defining a similarity between a pair of clusters C_i and C_j as the number of points that are grouped together. To compute the ratio of the intersection to the union of the sets of objects corresponding to the two hyperedges Jaccard index [Jac01] is used. The edge weight $w_{a,b}$ between two nodes h_a and h_b is defined according to binary Jaccard measure of their corresponding indicator vector $\mathbf{h}_a$ and $\mathbf{h}_b$ as:

$$w_{a,b} = \frac{\mathbf{h}_a{}^T \mathbf{h}_b}{||\mathbf{h}_a||_2^2 + ||\mathbf{h}_b||_2^2 - \mathbf{h}_a{}^T \mathbf{h}_b}. \tag{2.32}$$

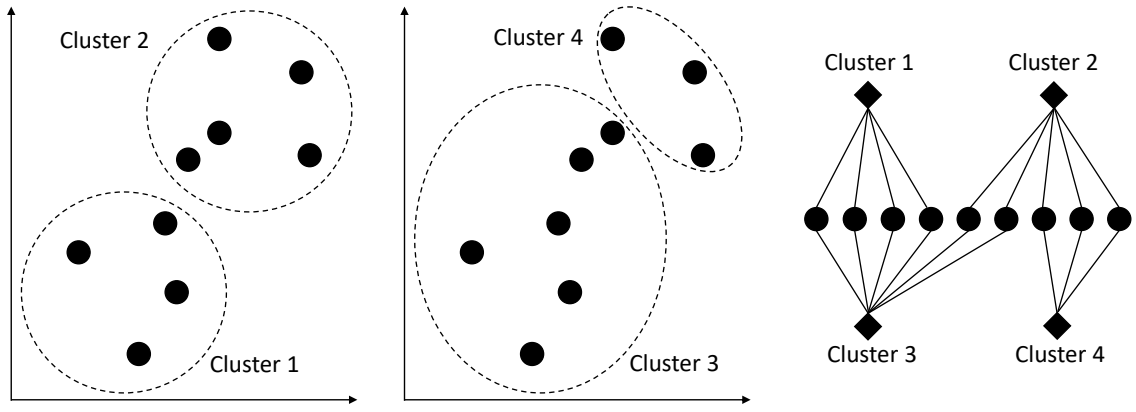
Analogous to CSPA, MCLA employs METIS [KK98a] algorithm in order to obtain meta-clusters that are used to obtain final consensus partitions by counting the number of times each data point falls in a meta-cluster. The biggest advantage of MCLA is that is has low time complexity that is quadratic to number of clusters in set partition $\mathbb{P}$.

2.5.3.3 Hyper-Graph Partitioning Algorithm (HGPA)

The consensus function in HGPA [SG03] is formulated as a hypergraph partitioning algorithm that cuts a minimal number of hyperedges, so that the hypergraph is partitioned into K connected components of approximately same dimension. In such hypergraph, all the vertices and hyperedeges are equally weighted while to obtain comparable sized partitions HMETIS [Kar+99] algorithm is used. The biggest drawback of HGPA is that it suffers from cluster imbalance problem.

2.5.3.4 Hybrid Bipartite Graph Formulation (HBGF)

HBGF combines advantages of both instance- and cluster-based consensus functions. The method constructs graph models for data points and clusters of $\mathbb{P}$ treating them as graph vertices (Figure 2.10). The objective of graph partitioning problem in HBGF is to find K ways of partitioning the graph so that it minimizes the cut while the constraint for the partitioning problem is to have roughly equal amount of vertices in each part. To constructs a graph $G = (V, W)$ the vertices V are represented as $V^P \cup V^I$,



(a) Partition 1 (b) Partition 2 (c) Graph representation

Figure 2.10: HBGF graph representation

where V^P denote clusters and V^I denote data points. The weights W are contracted according to

$$W(i,j) = W(j,i) = \begin{cases} 0, & \text{if } i \text{ and } j \text{ are both clusters or instances} \\ 1, & \text{else if instance } i \text{ belongs to cluster } j \end{cases} \tag{2.33}$$

and,

$$W = \begin{bmatrix} 0 & A^T \\ A^T & 0 \end{bmatrix} \tag{2.34}$$

where A^T is a matrix in which rows refer to instances and columns refer to clusters.

2.5.4 Knowledge based cluster ensemble (KB)

Knowledge based consensus clustering [ZW06] consist of four approaches. In all of them, base clusterings are generated by applying k-means with different initializations. The variation between these four approaches arises if objects are weighted and if partition selection is active. The first step in these set of methods is to align clusters in each input partitions. In order to do that, the number of overlapped objects is calculated forming similarity measure between partitions. For the method Voting, final label of i^{th} object is obtained based on plurality voting on label vector. Consider λ be the label vector corresponding to i^{th} object, where $\lambda = \{1,1,1,1,2,2,1,2,1,1\}$. According to plurality voting, i^{th} object will receive label 1. For the method Weighted-voting, in order to compute the weight for each partition mutual information is employed [SG03]. Those weights are then used for plurality voting of label vectors. For every clustering in $\mathbb{P}$, the average mutual information is computed as:

$$\beta_m = \frac{1}{H-1} \sum_{h=1, h \neq m}^{H} \Phi(\lambda^{(m)}, \lambda^{(h)}) \tag{2.35}$$

where Φ is calculated as in [SG03]. For the method Selective voting, partition weights are calculated as:

$$W_m = \frac{1}{\beta_m Z} \tag{2.36}$$

and then are used to filter input partitions. Partition whose mutual information weight is smaller than a predefined threshold are discarded. Such filtering is able to bring substantial improvement in the consensus clustering results [FL08]. The forth Selective weighted-voting approach combines partition filtering mechanism with weighted-voting being more accurate than other three approaches [ZW06].

2.5.5 Mirkin distance based methods

Mirkin distance based methods employ the symmetric difference between two partitions that is defined as the number of disagreements between them. Such a distance is then used in median partition formulation in order to find final solution. In [FS03] three heuristics were proposed to solve median partition problem applying Mirkin distance: Best-of-k (BOK), Simulated Annealing, One-element Move (SAOM) and Best One-element Move (BOM). Later, another four heuristics were proposed in [GMT07]. One of them, called Balls algorithm, creates a graph based on the partitions where every edge gets its weight according to the distance between objects. Agglomerative and Furthest algorithms are the variation of a typical agglomerative clustering algorithm with average and maximum linkages, respectively. In LocalSearch algorithm the cost of moving an object from one cluster to another is defined in order to iteratively form the final clustering. The main drawback of the listed heuristics is the computational complexity ($\mathcal{O}(N^2)$) which limits their application to big data setting.

2.5.6 Consensus clustering limitations

Generally, most existing consensus functions offer a trade-off between accuracy and scalability [Wan11; PS11]. In addition, while being complex and sometimes lacking a clear objective formulation, these methods do not offer a straightforward interpretation of the solution, thus providing no guarantee on its quality. Moreover, there are usually many parameters that should be carefully optimized for every particular task, including the target number of clusters that in practice is often an unknown value.

Recently, due to large data volumes and a demand for higher scalability, the data fragment (DF) concept was introduced in several works on consensus clustering. Employing DF allows pruning the search space for median partition-based consensus clustering methods [VA15] as well as to decrease both memory and time complexity for co-occurrence based approaches. In [Wan11] a data fragment-based consensus method called CA-Tree is introduced where both the dendrogram and co-association matrix are used to obtain a consensus solution. The main drawback of the method is high sensitivity to a partition that is taken as the first layer of the dendrogram making final results unstable. In [Wu+12] three methods adopted from data objects to DFs are presented: a bottom-up agglomerative algorithm F-agglomerative, a top-down approach F-Furtherst and a median partition based local-search heuristic F-LocalSearch. These three methods are all adoptions of corresponding object-based approaches and inherit drawbacks of the original methods: they treat all partitions equally even if

some of them are nonsense. In addition, they employ the Hamming distance [Ham50] as a distance measure which results in non-expressive and quantized representations of distances often leading to a non-optimal consensus solution.

2.5.7 Data Fragments

With the help of the previously introduced label matrix $\mathbf{Y}$ we define a data fragment as follows.

Definition. Data fragment (DF) $\mathbf{F}_l, l = 1, \ldots, L$ is a submatrix of $\mathbf{Y}$ in which all rows are equal to each other, i.e. $\mathbf{y}_i = \mathbf{y}_j \ \forall \mathbf{y}_i, \mathbf{y}_j \in F_l$.

One DF can be considered as a stable group of objects within the ensemble of clusterings where all included points are co-clustered. In fact, a complete cluster can be considered as a DF if all clusterings are agreed on it. We will refer to each fragment $\mathbf{F}_l$ by its unique row vector $\mathbf{f}_l \in \mathbf{F}_l$. The amount of DFs in an ensemble is usually much less than the number of original data points which allows handling large datasets [VA15; Wu+12]. Moreover, with the help of DFs it is possible to naturally extend object co-occurrence- and median partition-based formulations [Wu+12] and effectively use them as elements of a consensus function for ensemble aggregation [NG10]. The fact that every DF represents a set of stable clustered objects a proper (dis)similarity measure between DFs should be established in order to perform a reasonable consensus among partitions. In addition, since $\mathbf{f}_l$ is a categorical data vector, the (dis)similarity has to be defined over this type of data which in general is a challenging task [BCK08].

If we consider matrix $\mathbf{F}_l$ as a set of equal vectors (further denoted as F_l) its cardinality would correspond to the amount of data points that are co-clustered by all ensemble members. From the DF definition it is also clear that $\sum_{l=1}^{L} |F_l| = N$. Intuitively, DFs that have large $|F_l|$ are likely to form stable clusters or substantial parts of them. DFs with small $|F_l|$ correspond to objects on which the consensus is weak (outliers and noisy samples). At the same time, it is important to account for the frequency of each label within each clustering since the distribution of them is in general different, can be imbalanced and depends on the ensemble generation scheme and underlying data structure. In the next section, we propose a dissimilarity measure between DFs that addresses this peculiarity.

2.6 Data Fragments-based consensus

At first, we propose a consensus clustering framework that addresses the drawbacks of the Hamming distance in co-occurance based method and reduces computational and memory complexity. We employ a DF concept to assure scalability of the consensus clustering function while proposing an expressive distance measure on DFs that leads to a significant improvement in the final solution compared to approaches based on Hamming distance. We further build a consensus function around this measure based on a hierarchical clustering approach [Suk+17a].

2.6.1 Dissimilarity measure over data fragments

Lemma 1. *The number of data fragments in an ensemble with H clusterings is bounded by* $\max_{h=1,\dots,H} K_h \leq L \leq N$.

Proof. From Equation 2.14 $y_i^h \in \{1,\dots,K_h\}$. From the properties of clustering $y^h \in [1,\dots,K_h]^T$, so at least there are K_h points: $y_i^h \neq y_j^h \quad \forall i,j$. From the DF definition there exist at least K_h DFs since there are at least K_h non equal elements y_i^h. Since DFs are defined over all partitions $h \in \{1,\dots,H\}$ then the minimum number of DFs is defined by $\max_{h=1,\dots,H} K_h$. Consider $\mathbf{Y} : \boldsymbol{y_i} \neq \boldsymbol{y_j}$. From 2.5.7 every $\mathbf{y}_i$ will form a DF F_i and as the result there will be N distinct DFs. ■

In fact, if there are N DFs it means that there is no any pair of data points $\mathbf{x}_i, \mathbf{x}_j \in X$ on which all clustering solutions are agreed by co-clustering both of them.

Lemma 2. *If $|F| = K_h$ and $K_h = K_j \forall h \neq j, h, j \in \{1,\dots,H\}$ then all clusterings within the ensemble are equal (with regards to co-clustered data points) i.e. $P_h \equiv P_j$.*

Proof. The proof is trivial. ■

This lemma leads to the fact that in such cases the optimal solution for consensus clustering problem is a trivial solution and equals to any clustering within the ensemble.

In order to be able to operate on DFs and establish a dissimilarity measure between them to find consensus partition we first define an error function on categorical vectors

$\mathbf{y}_i$ which we want to minimize. For any partition P with K clusters and a dissimilarity measure between two categorical vectors d_c an error function

$$E(P) = \sum_{k=1}^{K} \frac{1}{|C_k| \cdot (|C_k| - 1)} \sum_{\substack{\mathbf{y}_i, \mathbf{y}_j \in C_k, \\ \mathbf{y}_i \neq \mathbf{y}_j}} d_c(\mathbf{y}_i, \mathbf{y}_j) \tag{2.37}$$

measures the aggregated average dissimilarity between points of every cluster and closely related to optimization criteria of k-means and agglomerative clustering algorithms [SJ13]. The choice of an appropriate dissimilarity measure d_c is critical since it can seriously affect the behavior of the error function. The optimal partition P^* can be then defined as follows:

$$P^* = \underset{P \in \mathcal{P}_x}{\operatorname{argmin}} E(P) \tag{2.38}$$

where $\mathcal{P}_x$ is a search space with all possible clusterings of $\mathbf{Y}$. In previous works on consensus clustering in which the notion of distance between two categorical vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ was defined [GMT07] mainly the Hamming distance was considered.

$$d(\mathbf{y}_i, \mathbf{y}_j) = \sum_{h=1}^{H} \delta(y_i^h, y_j^h) \tag{2.39}$$

where

$$\delta(y_i^h, y_j^h) = \begin{cases} 1, & y_i^h \neq y_j^h \\ 0, & \text{otherwise} \end{cases} \tag{2.40}$$

In fact, the Hamming distance (sometimes also called as overlap measure when introduced as a similarity measure [BCK08]) provides an easy and understandable way to compare two categorical vectors. However, it suffers from a substantial drawback since it assigns equal significance to dissimilarities for all vectors attributes. For many problems (including consensus clustering) the assumption of equal significance of attributes errors is not valid as the partitions in ensemble could be very diverse (every partition has its own number of clusters K_h and may be generated using different distance metrics and clustering methods). Generally, measuring dissimilarities between categorical vectors is not a straightforward task since the content of dissimilarity is highly application specific and categories are often ambiguous or even arbitrary. As an alternative to the Hamming distance there are several data-driven dissimilarity functions that take into account the frequency distribution of values of every attribute. In [BCK08] the authors systematically studied 14 measures for categorical values concluding that the choice of them strongly depends on the assumptions that are imposed on the data. In the sequel, we define a dissimilarity measure on DFs and use DFs further in Equation 2.37 instead of data point labels.

A general distance measure between two categorical vectors $\mathbf{f}_i$ and $\mathbf{f}_j$ representing DF can be defined as:

$$d(\mathbf{f}_i, \mathbf{f}_j) = \sum_{h=1}^{H} w_h \cdot d_h(f_i^h, f_j^h) \tag{2.41}$$

where w_h is the weight for every hth attribute and $d_h(f_i^h, f_j^h)$ defines the dissimilarity between values of this attribute. Since $\mathbf{f}_i$ and $\mathbf{f}_j$ are vectors representing their respective sets we propose accounting for unequal distributions of attribute values of every attribute. For every DF F_i and partition P_h we define a significance value S_i^h according to:

$$S_i^h = \frac{|F_i|}{|C_{f_i^h}| \cdot K_h} \tag{2.42}$$

where $|\cdot|$ is the cardinality of a set and $C_{f_i^h}$ is the cluster with label f_i^h in partition h. A significance value S_i^h shows the relative amount of co-clustered data points in the DF i of the partition h with respect to the number of objects with the same label that are clustered differently assuming that every cluster has equal importance. The dissimilarity d_h between an attribute of two DFs is then defined as:

$$d_h(f_i^h, f_j^h) = \begin{cases} 0, & f_i^h = f_j^h \\ 1-(\frac{2}{K_h}-S_i^h-S_j^h), & \text{otherwise} \end{cases} \tag{2.43}$$

which compares the significance of two DFs with doubled significance of a case when a DF occupies whole cluster. In general, the more data points a DF shares with a cluster, the higher the certainty that this DF is a substantial subset of a cluster. As a result, it ends up in a higher distance with other DFs.

In addition, the proposed dissimilarity considers equal importance of every cluster within a partition independently of the amount of objects assigned to it effectively allowing comparing DFs with different amount of objects. We note that $d_h \in [0, 1]$ and is symmetric

$$d_h(f_i^h, f_j^h) = d_h(f_j^h, f_i^h) \tag{2.44}$$

In Equation 2.41 the weights w_h are assigned to each attribute h to signify its relative importance. Since the partitions in the ensemble can have different quality level (local minima partitions or outliers) we employ w_h as the degree of agreement of a particular partition P_h with all the partitions in the ensemble $\mathbb{P}$. For that we define a distance measure between two partitions of the given ensemble using their respective connectivity matrices [LDJ07], however, formulated on DFs instead of object labels:

$$d(P_1, P_2) = \sum_{i,j}^{L} |\mathbf{M}_{ij}(P_1) - \mathbf{M}_{ij}(P_2)| \times |F_i| \times |F_j| \tag{2.45}$$

where $\mathbf{M}_{ij}(P_i)$ is the connectivity matrix on DFs that is defined as:

$$\mathbf{M}_{ij}(P_h) = \begin{cases} 1, \exists C_k^h \in P_h \mid f_i^h \in C_k^h \text{ and } f_j^h \in C_k^h \\ 0, \quad \text{otherwise} \end{cases} \tag{2.46}$$

Finally, using Equation 2.45 for every attribute $h' \in (1, \ldots, H)$ we define its weight $w_{h'}$ as follows:

$$w_{h'} = \frac{\sum_{h=1}^{H} w_h - w'_{h'}}{\sum_{h=1}^{H} w_h} \tag{2.47}$$

where $w'_{h'}$ is calculated as:

$$w'_{h'} = \sum_{h=1}^{H} d(P_{h'}, P_h) \tag{2.48}$$

and $\sum_{h=1}^{H} w_h = 1$ holds.

As a result, we obtained a dissimilarity measure over DFs (Equation 2.41) that provides expressive distance between categorical vectors. The proposed dissimilarity measure is symmetric and non-negative and can be used to construct co-association matrices commonly used in clustering ensembles and create a (dis)similarity matrix that summarizes richer information than the original one.

To establish a consensus function using dissimilarity measure proposed and solve Problem 2.38 on DFs we employ agglomerative clustering with between-group average linkage [SJ13] on the dissimilarity matrix obtained by applying the proposed dissimilarity measure to all DFs within the ensemble. We call this consensus function DF-based Expressive Consensus (DFEC) [Suk+17a]. Generally, the proposed dissimilarity measure can be used with various clustering methods (e.g. EM-based clustering algorithms [PS11]), in order to obtain consensus solution.

2.6.2 Example

In order to better understand the proposed DFEC method we provide an example on all its building blocks and components. Consider a toy dataset with N=10 that was clustered $H = 3$ times by different clustering algorithms. Table 2.5 provides possible label distribution for this scenario. Table 2.6 demonstrates an example of intermediate structures obtained from Table 2.5. All three clusterings performed with their distinct number of clusters: $K_1 = 3, K_2 = 4, K_3 = 2$. Data fragments table provides an overview on data fragments discovered as well as their quantity. There are $L = 6$

Table 2.5: Three different clustering on N=10 data points

	Clustering 1	**Clustering 2**	**Clustering 3**
1	1	1	1
2	1	2	1
3	2	3	1
4	3	4	2
5	1	1	1
6	1	2	1
7	2	3	2
8	3	4	2
9	2	4	2
10	3	4	2

Table 2.6: An example of resulting data fragments obtained from three base clusterings as well as their corresponding terms $\left|C_{f_i^h}\right|$

Data Fragments				
	P_1	P_2	P_3	$\lvert F_i\rvert$
f_1	1	1	1	2
f_2	1	2	1	2
f_3	2	3	1	1
f_4	2	3	2	1
f_5	2	4	2	1
f_6	3	4	2	3
K_h	3	4	2	

Term $\left\lvert C_{f_i^h}\right\rvert$			
f_i^h	P_1	P_2	P_3
1	4	2	5
2	3	2	5
3	3	2	0
4	0	4	0

data fragments discovered in the dataset. In the table with term $\left|C_{f_i^h}\right|$, the term is provided for every cluster of each clustering. Note, that for clustering P_3 there is only two non-zero $\left|C_{f_i^h}\right|$ terms as there are only two available clusters. Table 2.7 provides significance values defined in Equation 2.42 for every data fragment and clustering. Finally, Table 2.8 shows two connectivity matrices for partitions P_1 and P_2.

2.6.3 Experimental Results

To study the effectiveness of the consensus function based on our proposed dissimilarity measure and compare its performance with the state-of-the-art consensus clustering methods we conduct numerical simulations using synthetic (artificially generated) data and experiments using real-world (measurement data) datasets. All datasets are provided with the ground truth (class labels). For numerical simulations and experiments in order to generate diverse input partitions we use multiple clustering algorithms [SJ13]

Table 2.7: An example of resulting significance values obtained from Table 2.6 according to Equation 2.42

Significance Matrix			
	P_1	P_2	P_3
f_1	0.167	0.250	0.2
f_2	0.167	0.250	0.2
f_3	0.111	0.125	0.1
f_4	0.111	0.125	0.1
f_5	0.111	0.063	0.1
f_6	0.333	0.188	0.3

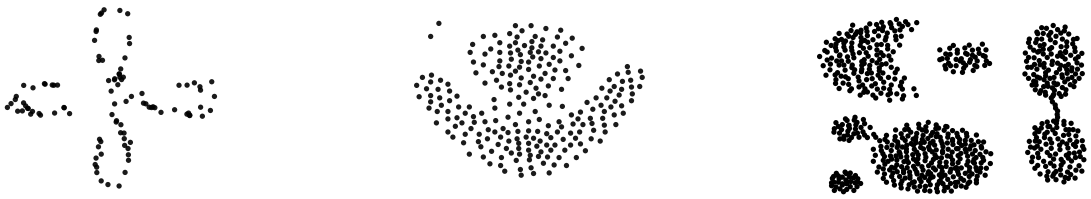
Table 2.8: An example of connectivity matrices for partitions P_1 and P_2 obtained according to Equation 2.46

Connectivity matrix M(P_1)						
	f_1^1	f_2^1	f_3^1	f_4^1	f_5^1	f_6^1
f_1^1	1	1	0	0	0	0
f_2^1	1	1	0	0	0	0
f_3^1	0	0	1	1	1	0
f_4^1	0	0	1	1	1	0
f_5^1	0	0	1	1	1	0
f_6^1	0	0	0	0	0	1

Connectivity matrix M(P_2)						
	f_1^2	f_2^2	f_3^2	f_4^2	f_5^2	f_6^2
f_1^2	1	0	0	0	0	0
f_2^2	0	1	0	0	0	0
f_3^2	0	0	1	1	0	0
f_4^2	0	0	1	1	0	0
f_5^2	0	0	0	0	1	1
f_6^2	0	0	0	0	1	1

also varying their parameters: k-means (with random initialization), hierarchical clustering (with random linkage and number of neighbors), affinity propagation [FD07] (with random damping factor and iterations), BIRCH (with random threshold), DBSCAN (with random eps factor) and mean shift (with random bandwidth). For k-means and hierarchical clustering the number of clusters provided was chosen randomly on the uniform interval [2, true cluster count +2]. As a result, every ensemble consists of ten partitions which are assured to be distinct and different from the ground truth. Such diversity in input partitions does not allow particular clustering results to dominate and thus helps to evaluate the stability of a consensus clustering approach and see whether it is capable of providing a novel solution with respect to the input partition. We chose the cutting threshold for DFEC as well as resulting number of clusters for other consensus clustering methods according to the true number of clusters.

Table 2.9 provides datatsets descriptions while Figure 2.11 shows the patterns of the synthetic datasets utilized.



(a) Petals (b) Flame (c) Aggregation

Figure 2.11: Original patterns of synthetic dataset

Table 2.9: Experimental Data Sets

Dataset	Instances	Dimensions	Classes
petals	100	2	4
flame	240	2	2
two half-rings	373	2	2
aggregation	788	2	7
iris	150	4	3
breast cancer	698	9	2
agaricus lepiota	8123	21	2
magic	19019	10	2

2.6.3.1 Numerical simulations

In this numerical simulation, we compare the performance of the proposed DFEC [Suk+17a] with other DF-based consensus clustering methods: CA-Tree, F-Agglomerative, F-Furthest and F-LocalSearch using four synthetic datasets: petals [Kun16], aggregation [GMT07], flame [FM07] and dim32 [FVH06]. In Table 2.11 we report the Adjusted Rand Index (ARI) that is widely used for clustering evaluation and related to the accuracy measure while operating on pairs of elements and adjusted for chance [HA85]. ARI is defined between two partitions P_a and P_b as:

$$ARI(P_a, P_b) = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}} \tag{2.49}$$

where n_{ij}, a_i, b_j are the values from the contingency table defined in 2.10 where each entry n_{ij} denotes the number of objects in common between C_i^a and C_j^b: $n_{ij} = |C_i^a \cap C_i^b|$.

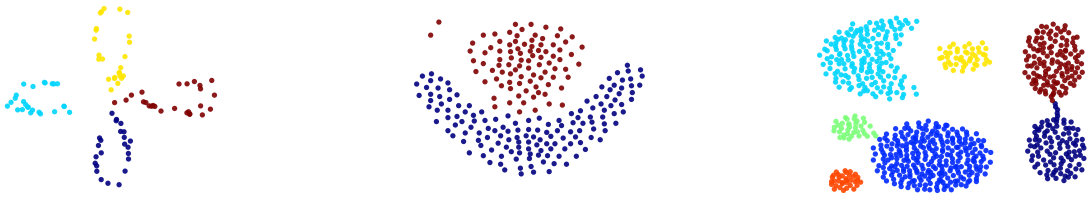
Table 2.10: Contingency Table

P_a/P_b	C_1^b	C_2^b	$\cdots$	$C_{K_b}^b$	Sums
C_1^a	n_{11}	n_{12}	$\cdots$	n_{1K_b}	a_1
C_2^a	n_{21}	n_{23}	$\cdots$	n_{2K_b}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$C_{K_a}^a$	n_{K_a1}	n_{K_a2}	$\cdots$	$n_{K_aK_b}$	a_{K_a}
Sums	b_1	b_2	$\cdots$	b_{K_b}	

Table 2.11: Adjusted Rand Index on synthetic datasets

	Petals	**Flame**	**Aggregation**	**Dim32**
CA-Tree	0.689	0.319	0.649	0.603
F-Agglomerative	0.468	0.078	0.318	0.399
F-Furthest	0.429	0.458	0.615	0.149
F-Local Search	0.091	0.029	0.040	0.000
DFEC	**0.973**	**0.876**	**0.876**	**0.925**

We also provide graphical results for our proposed method DFEC in Figure 2.12 to demonstrate its ability to establish consensus (note that a figure for dataset dim32 is not provided since the dimension of the original data is 32).



(a) Petals (b) Flame (c) Aggregation

Figure 2.12: Consensus solution using proposed method DFEC

2.6.3.2 Real-world data experiments

In this set of experiments, we use a number of real-world datasets taken from the UCI repository [Lic13] that are widely used in consensus clustering research, namely breast cancer, thyroid, wine, wdbc and seeds. In addition to the methods evaluated in the numerical simulation, we evaluate CSPA, HGPA, MCLA [SG03], CTS, SRS, ASRS [NG10], HBGF [FB04] and knowledge based (KB) [ZW06] methods. To evaluate

the quality of the final consensus solution besides ARI we employ two other commonly used external validity indexes: Impurity Index (IMP) which reflects the amount of differently labeled points in clusters [GMT07] and Average Entropy (AE) which is defined similarly to the entropy used in traditional decision tree building [ZZ09]. Note that for the quality solutions, ARI should be large while IMP and AE should be as low as possible. We summarize the results of this experiment in Table 2.

Table 2: Evaluation results using real-world data sets (N - original number of data points, L - number of DFs)

	breast cancer ($N = 698, L = 81$)			**thyroid** ($N = 214, L = 16$)			**wine** ($N = 177, L = 21$)			**wdbc** ($N = 568, L = 131$)			**seeds** ($N = 210, L = 22$)		
	ARI	IMP	AE	ARI	IMP	AE	ARI	IMP	AE	ARI	IMP	AE	ARI	IMP	AE
CSPA	0.017	0.967	0.083	0.155	0.453	0.902	0.231	0.588	0.647	0.232	0.599	0.414	0.457	0.426	0.504
HGPA	0.017	0.960	0.094	0.097	0.439	0.885	0.304	0.492	0.676	-0.001	0.746	0.693	0.262	0.536	0.711
MCLA	0.000	0.000	0.000	0.546	0.164	0.522	0.226	0.542	0.560	0.224	0.537	0.295	0.496	0.281	0.388
CA-Tree	0.477	0.388	0.206	0.281	**0.112**	0.325	**0.389**	0.282	0.619	0.490	0.146	0.260	0.545	0.150	0.321
CTS	0.088	0.868	0.064	0.442	0.206	0.523	0.285	0.463	0.557	0.512	0.264	0.294	0.633	0.220	0.357
SRS	0.096	0.853	0.105	0.442	0.206	0.523	0.302	0.475	0.668	0.438	0.405	0.515	0.623	0.239	0.398
ASRS	0.127	0.776	0.056	0.578	0.140	0.394	0.310	0.463	0.534	0.519	0.202	0.142	0.611	0.172	0.289
HBGF	0.042	0.907	0.075	0.386	0.243	0.669	0.261	0.554	0.577	0.338	0.484	0.295	0.520	0.347	0.421
KB	-0.006	0.293	0.051	0.517	0.196	0.498	0.128	0.305	0.590	0.109	0.194	0.457	0.191	0.307	0.499
F-Agg.	0.191	0.637	0.087	0.273	0.458	0.606	0.169	0.684	0.312	0.264	0.539	0.150	0.316	0.567	0.261
F-Fur.	0.729	0.165	0.175	0.256	0.336	0.616	0.294	0.418	0.375	0.322	0.431	0.392	0.485	0.277	0.354
F-Loc.	-0.002	**0.013**	**0.016**	0.196	0.117	**0.279**	-0.008	0.311	**0.256**	0.241	0.141	**0.112**	0.107	**0.124**	**0.207**
DFEC	**0.845**	0.040	0.166	**0.601**	0.131	0.371	0.367	**0.291**	0.412	**0.594**	**0.113**	0.254	**0.665**	**0.124**	0.346

2.6.3.3 Experiments discussion

Analyzing the evaluation results (Table 2.11 and 2) it can be seen that in most cases proposed DFEC outperforms other methods in terms of ARI. For some datasets IMP and AE are not the lowest for DFEC (however comparable with the winning ones). The main reason for such behavior is the fact that these measures are biased by different aspects of clustering: IMP considers majority-class points in each cluster, AE focuses on distribution of all labels in each cluster while ARI is related to classical accuracy measure. The opposite effect can be observed for F-Furthest, however, the reason for low ARI for this method is the strong initialization dependence. The superior performance of the proposed DFEC is achieved thanks to the introduced distance measure that takes into account the quality of input partitions as well as significance value for each attribute of DF (Equation 2.42). The results on synthetic datasets also demonstrate that novel solutions can be found by DFEC. Finally, Table 2 also confirms that the DF concept allows to significantly decrease the amount of points on which aggregation is performed (L is much lower than N) allowing for larger datasets.

2.7 Non-negative Matrix Factorization-based consensus

To address consensus clustering interpretation problem as well as to allow it to scale to large data sets, we propose a clustering ensemble framework that allows us to accurately combine multiple input clusterings while providing descriptive results interpretation. We formulate a clustering ensemble problem as a Binary Matrix Factorization (BMF) [Zha+07] and efficiently solve it by means of a recursive rank-one binary matrix approximation based on the Alternating Iterative Heuristics algorithm [KG03] and introducing an effective initialization strategy. Besides providing an accurate and stable consensus solution, the proposed framework requires no information about the target number or size of clusters that offers intuitive result treatment and is suited for large-scale datasets and high amount of ensemble members.

2.8 Consensus clustering as binary matrix factorization

To account for the drawbacks of the object co-occurrence- and median partition-based problem formulation we formulate the clustering ensemble problem as a Non-negative Matrix Factorization (NMF) [LS99] problem where the label matrix $\mathbf{Y}$ is factorized into a membership matrix $\mathbf{M}$ and a pattern matrix $\mathbf{Q}$ as

$$\mathbf{Y} \approx \mathbf{M}\mathbf{Q}^\top, \qquad \mathbf{Y}, \mathbf{M}, \mathbf{Q} > 0 \tag{2.50}$$

to minimize the approximation error that is the squared Frobenius norm [Li05] of the residual:

$$\underset{\mathbf{M},\mathbf{Q}}{\operatorname{argmin}} \left\| \mathbf{Y} - \mathbf{M}\mathbf{Q}^\top \right\|_F^2 \tag{2.51}$$

The main issue with NMF on the $\mathbf{Y}$ matrix is that $\mathbf{Y}$ is formed from categorical data vectors since every partition within the ensemble represents a symbolic assignment of a point to a cluster.

To account for that, we transform $\mathbf{Y}$ to a matrix of indicator variables (also known as one-hot or dummy encoding) as:

$$\delta_{i \in K_h} = \begin{cases} 1, & i \in K_h \\ 0, & \text{otherwise} \end{cases} \tag{2.52}$$

and, as a result, obtaining $\mathbf{Y}_b \in \{0,1\}^{N \times \sum_{h=1}^{H} K_h}$ that represents every partition P_h as a binary matrix of size $N \times K_h$. For further convenience we define $T = \sum_{h=1}^{H} K_h$.

Due to the nature of $\mathbf{Y}_b$ the problem transforms to the Binary Matrix Factorization (BMF) [Zha+07] problem where $\mathbf{Y}_b$ is decomposed to a consensus membership matrix $\mathbf{M}$ and a matrix of consensus representations $\mathbf{Q}$ that both have an additional constraint to be binary. The constraint comes from the fact that the consensus clustering solution has to be crisp (i.e. a non-overlapping solution) and provide interpretable results to be able to evaluate the consensus quality. According to the described transformations, Problem 2.51 is now reformulated to a BMF as

$$\underset{\mathbf{M}_b, \mathbf{Q}_b}{\operatorname{argmin}} \left\| \mathbf{Y}_b - \mathbf{M}_b \mathbf{Q}_b^\top \right\|_F^2 \tag{2.53}$$

where $\mathbf{M}_b \in \{0,1\}^{N \times K}$ and $\mathbf{Q}_b \in \{0,1\}^{T \times K}$ are restricted to be binary. Matrix $\mathbf{M}_b$ consists of presence vectors specifying the consensus clustering membership of every object $\mathbf{x}_i$ while $\mathbf{Q}_b$ contains dominant binary patterns of $\mathbf{Y}_b$ that can be interpreted as centroids in the ensemble label space. An additional property of BMF that we would like to achieve is based on the fact that the target number of clusters is an unknown value and we have to induce it based on the ensemble structure. For this we impose a constraint on matrix $\mathbf{Q}_b$ of being able to reconstruct $\mathbf{Y}_b$ with a desired error ε providing a minimum number of centroids K:

$$\forall \mathbf{y}_i \in \mathbf{Y} \; \exists \mathbf{q}_k : ||\mathbf{y}_i - \mathbf{q}_k||_2^2 \leq \varepsilon, k = \{1, \ldots, K\} \tag{2.54}$$

The error ε implicitly controls the target number of clusters K. Note that the error ε and the resulting number of clusters K are directly linked to ensemble diversity.

To preserve the discrete properties of the data, an efficient and elegant solution for BMF can be found by solving a rank-one binary matrix approximation [SJY09] that searches for two binary vectors $\mathbf{m}_b$ and $\mathbf{q}_b$ whose outer product provides the minimum distance (that is the Hamming distance for binary vectors) from the matrix to factorize:

$$\min_{\mathbf{m}_b, \mathbf{q}_b} \left\| \mathbf{Y}_b - \mathbf{m}_b \mathbf{q}_b^\top \right\|_F^2 = \min_{\mathbf{m}_b, \mathbf{q}_b} \sum_{n,t=1}^{N,T} |(\mathbf{Y}_b - \mathbf{m}_b \mathbf{q}_b^\top)_{n,t}| \tag{2.55}$$

Since the rank-one binary matrix approximation minimizes the number of nonzero elements in the residual matrix it provides a useful framework to implicitly assess the quality of the consensus solution. When having such a setting there is the possibility to solve BMF iteratively (for $K = 1$) without specifying the number of clusters but relying on the aggregation of those solutions providing an error ε that would determine the optimal number of centroids. All the data objects then would be centered around their

respective centroid that provides the minimum distance with each of them. Unfortunately, it was shown that Problem 2.55 is NP-hard [SJY09] and that only approximate solutions might be reasonably found. For that several algorithm were proposed [SWS14; Li05], some of them with guaranteed approximation error bound [SJY09; Lu+]. However, many of them require high computational resources and deliver difficulties for high N that is common for current practical clustering tasks. Moreover, since we allow any number of ensemble members with an arbitrary number of clusters the number of columns of the matrix $\mathbf{Y}_b$ can impose computational challenge as well.

To overcome this limitation, we consider the alternating iterative heuristic algorithm [KG03] that performs a non-orthogonal binary matrix factorization. The idea of alternating iterative heuristic is to recursively grow a tree by employing a rank-one binary matrix approximation that splits the matrix in each node into two sub-matrices $\mathbf{Y}_b^1$ and $\mathbf{Y}_b^0$ based on their distance to a dominant pattern as

$$\mathbf{Y}_b(i) = \begin{cases} \mathbf{Y}_b^1, & \text{if } \mathbf{m}_b(i) = 1 \\ \mathbf{Y}_b^0, & \text{otherwise} \end{cases} \tag{2.56}$$

for $1 \leq i \leq n$ and where $\mathbf{Y}_b(i)$ denotes the i^{th} row of matrix $\mathbf{Y}_b$. On the following iteration a rank-one approximation is found for $\mathbf{Y}_b^0$. At the same time, matrix $\mathbf{Y}_b^1$ and its respective pattern vector $\mathbf{q}_b$ are controlled to meet the stopping criteria, otherwise the recursive procedure is continued similarly to $\mathbf{Y}_b^0$. The splitting is stopped when the distance becomes less than the prescribed bound ε.

Alternating iterative heuristic is able to handle large N and H in nearly linear time making it attractive for large scale applications [KG03].

The downside of the alternating iterative heuristic is the fact that it finds only local patterns thus being able to provide a locally optimal solution only. Additionally, to improve local approximation, an efficient initialization mechanism is to be employed. For that we propose to choose an initialization based on the maximum count of repetitive vectors in $\mathbf{Y}$. Such vectors known as data fragments [Suk+17a] constitute stable groups of objects in the label space across all clusterings and serve as centroid candidates. The motivation behind choosing data fragments with the largest cardinality as initialization vectors comes from Lemma 2. Based on that, it is expected that prominent stable groups of points appear to be resulting pattern vectors (or their variants).

Figure 2.13 provides an example of decomposition of a binary matrix using alternating iterative heuristic implementation Proximus [KG03]. According to that, a rank-one approximation is computed iteratively for the original matrix and further for submatrices.

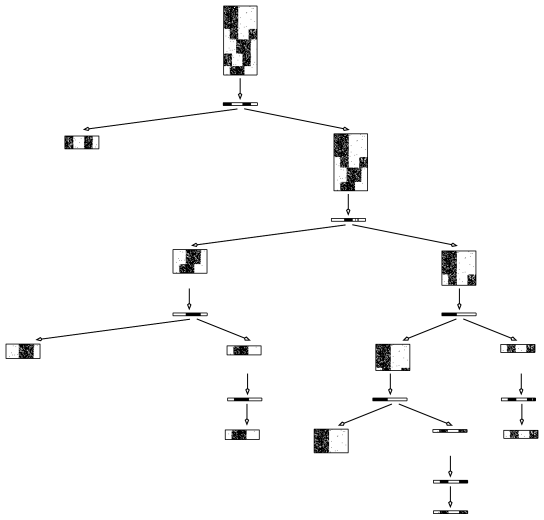


Figure 2.13: An example of decomposition of a binary matrix using alternating iterative heuristic implementation Proximus [KG03]

Based on Figure 2.6 the BMF on matrix $\mathbf{Y}$ solved by rank-one binary matrix approximation provides the following membership and centroid matrices $\mathbf{M}_b$ and $\mathbf{Q}_b$, respectively (for convenience we converted $\mathbf{Q}_b$ back to label representations using inverse one-hot encoding, and defined it as $\mathbf{Q}_{cat}$). Additionally, we colored elements of matrix $\mathbf{M}_b$ according to colors of objects in Figure 2.6.

$$\mathbf{Y} = \begin{bmatrix} 1&1&1&1&1&2&2&2&2 \\ 4&4&1&1&2&2&2&3&3 \\ 1&1&1&3&3&3&2&2&2 \\ 1&2&2&2&1&3&3&4&4 \\ 3&3&1&3&2&2&2&1&1 \end{bmatrix}^\top \qquad \mathbf{M}_b = \begin{bmatrix} 1&1&1&1&0&0&0&0&0 \\ 0&0&0&0&1&1&1&0&0 \\ 0&0&0&0&0&0&0&1&1 \end{bmatrix}^\top$$

$$\mathbf{Q}_{cat} = \begin{bmatrix} 1&4&1&2&3 \\ 2&2&3&3&2 \\ 2&3&2&4&1 \end{bmatrix}$$

Three clusters with their respective centroids are identified. Using matrix $\mathbf{M}_b$ it is possible to find out which object belongs to which centroid and perform further quality analysis. According to the proposed initialization strategy, vector $\mathbf{y}_8$ or $\mathbf{y}_9$ would be chosen to start the decomposition since their respective data fragment shows the largest cardinality.

2.8.1 Experimental Results

In this Section, we evaluate the properties and the performance of the proposed BMF-based consensus function that employs Alternating Iterative Heuristic [KG03] (we call it BMFC [SDZ18] in the sequel) and compare it with other state-of-the-art consensus functions. In all numerical simulations and experiments we generate ensembles that consist of $H = 12$ partitions. The way these partitions were generated is described in every subsection individually. The synthetic (artificially generated) *cone torus, checker board, halfring, boat, petals, aggregation* and real-world (measurement data) *ionosphere, thyroid, wine, glass, wisconsin* datasets that we use in the numerical simulations and the experiments are obtained from [Kun16] and UCI repository [Lic13], respectively and are commonly used in clustering research. In addition, we use a recent dataset *tiselac* provided by the ECML-PKDD 2017 TiSeLaC challenge [Ien+17]. For every dataset the ground truth labels are available and the number of clusters K_t is known. We analyze experimental results for all numerical simulations and experiments in Section 2.8.1.4.

2.8.1.1 Effect of error ε on number of clusters

In the first numerical simulation, we study the effect of the error ε on the number of clusters of the consensus solution. For that we apply BMFC on synthetic datasets while varying the normalized error ε_n in the interval $[0, 1]$ and report the number of clusters on the solution on Figure 3.3. The normalized error ε_n is defined as $\lceil \frac{\varepsilon}{T} \rceil$. To generate ensemble partitions we standardize the data and run four instances of k-means, BIRCH [ZRL96] and mini-batch k-means [Scu10] each. For every clustering instance the target number of clusters is drawn uniformly from the interval $[\max(2, K_t - 2), K_t + 2]$, for BIRCH the values for the branching factor and the subcluster threshold are drawn uniformly from the interval $[40, 60]$ and $[0.35, 0.65]$ correspondingly. For mini-batch k-means the batch size is $\lceil N \times 10^{-3} + 1 \rceil$.

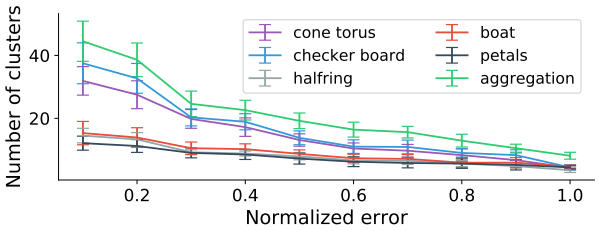


Figure 2.14: Number of clusters K with its standard deviation for BMFC for various values of normalized error ε over clustered partitions

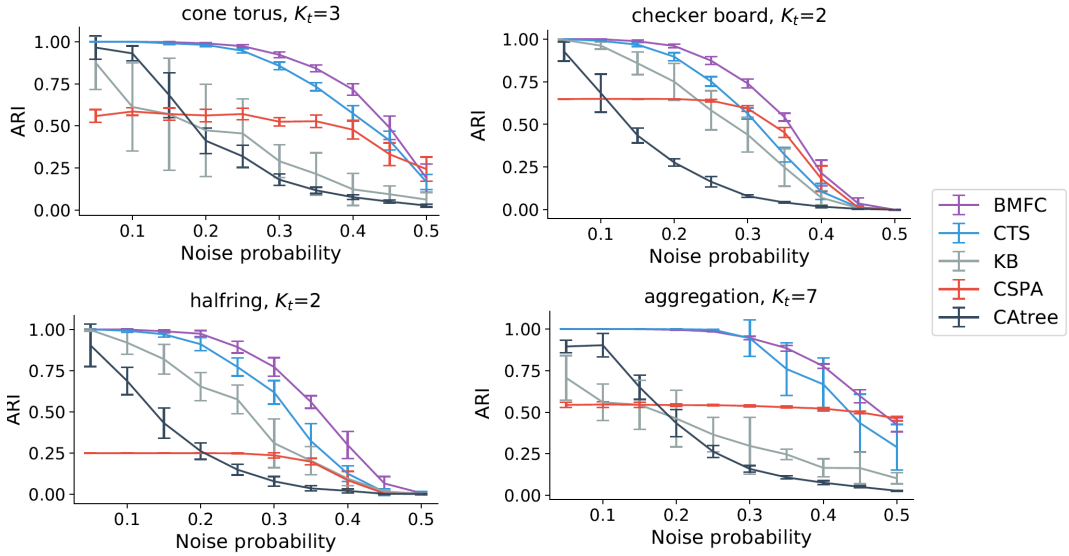


Figure 2.15: ARI with its standard deviation on four synthetic datasets for several consensus functions over noisy partitions

2.8.1.2 Effect of ensemble quality on the performance of BMFC

In this numerical simulation, to understand how ensemble quality affects the performance of the proposed BMFC [SDZ18] we apply random noise that follows a Bernoulli distribution with probability p. The noise is applied to each object label so that the label is flipped to a random cluster label with equal probability $q = \frac{p}{K_t - 1}$. In addition, we perform random permutations of resulting labels with uniform probability. We vary the probability of noise p in the interval $[0.05, 0.5]$ with step size 0.05 and report the average over 100 Monte Carlo iterations Adjusted Rand Index (ARI) [HA85] (Figure 2.15) for BMFC and four other state-of-the-art consensus functions CTS [NG10], Knowledege Based (KB) [ZW06], CSPA [SG03], CAtree [Wan11] (we evaluate here only four methods in order not to overload the plots). ARI shows the similarity between resulting clustering and the ground truth labels providing 1.0 when they are identical independently on the cluster symbolic labels. For BMFC we set $\varepsilon = 0.9$, for the other methods we set the target cluster number $K = K_t$.

2.8.1.3 Performance on synthetic and real datasets

In this experiment, we evaluate the performance of the BMFC [SDZ18] and the other state-of-the-art methods both on synthetic and real-world datasets. In addition to methods evaluated in the previous numerical simulation, we evaluate HBGF [FB04],

HGPA, MCLA [SG03], SRC and ASRS [NG10]. The parameters for BMFC and other methods are set as in the previous numerical simulation. The ensemble partitions are generated as in the first numerical simulation. The results for BMFC, HBGF, CAtree, HGPA and MCLA averaged over 100 Monte Carlo iterations provide ARI with their standard deviation in Table 2.13.

Table 2.13: ARI with its standard deviation on real-world and synthetic data sets for BMFC, HBGF, CAtree, HGPA and MCLA

	BMFC	HBGF	CAtree	HGPA	MCLA
ionosphere	$\mathbf{0.21 \pm 0.03}$	0.18 ± 0.05	0.16 ± 0.05	0.02 ± 0.02	0.17 ± 0.02
thyroid	$\mathbf{0.24 \pm 0.04}$	0.14 ± 0.13	0.11 ± 0.04	0.05 ± 0.03	0.13 ± 0.10
wine	0.80 ± 0.07	$\mathbf{0.81 \pm 0.04}$	0.74 ± 0.17	0.40 ± 0.24	0.79 ± 0.04
glass	$\mathbf{0.26 \pm 0.03}$	0.19 ± 0.03	0.15 ± 0.03	0.14 ± 0.04	0.18 ± 0.04
wisconsin	$\mathbf{0.89 \pm 0.04}$	0.87 ± 0.03	0.84 ± 0.03	0.01 ± 0.01	0.87 ± 0.01
boat	0.44 ± 0.05	0.41 ± 0.01	0.35 ± 0.06	0.33 ± 0.16	0.43 ± 0.05
petals	0.95 ± 0.11	0.91 ± 0.13	0.91 ± 0.10	0.85 ± 0.20	0.96 ± 0.03
aggregation	$\mathbf{0.83 \pm 0.06}$	0.74 ± 0.04	0.65 ± 0.06	0.51 ± 0.13	0.72 ± 0.04
cone torus	$\mathbf{0.37 \pm 0.03}$	0.35 ± 0.02	0.35 ± 0.04	0.18 ± 0.10	0.35 ± 0.04
checker board	0.12 ± 0.08	0.14 ± 0.09	0.06 ± 0.04	0.13 ± 0.10	0.09 ± 0.04
halfring	$\mathbf{0.58 \pm 0.06}$	0.54 ± 0.02	0.34 ± 0.12	0.04 ± 0.01	0.47 ± 0.10

Similarly, the results for CSPA, KB, CTS, SRS and ASRS are summarized in Table 2.14.

Table 2.14: ARI with its standard deviation on real-world and synthetic data sets for CSPA, KB, CTS, SRS and ASRS

	CSPA	KB	CTS	SRS	ASRS
ionosphere	0.17 ± 0.02	0.03 ± 0.06	0.18 ± 0.08	0.18 ± 0.01	0.17 ± 0.02
thyroid	0.13 ± 0.09	0.12 ± 0.05	0.23 ± 0.01	0.23 ± 0.03	0.12 ± 0.02
wine	0.79 ± 0.02	0.17 ± 0.19	$\mathbf{0.81 \pm 0.04}$	0.79 ± 0.04	0.72 ± 0.10
glass	0.15 ± 0.03	0.08 ± 0.05	0.24 ± 0.03	0.24 ± 0.02	0.24 ± 0.02
wisconsin	0.47 ± 0.01	0.12 ± 0.10	0.87 ± 0.03	0.87 ± 0.03	0.85 ± 0.02
boat	$\mathbf{0.45 \pm 0.11}$	0.14 ± 0.13	0.41 ± 0.04	0.40 ± 0.03	0.43 ± 0.06
petals	$\mathbf{0.97 \pm 0.08}$	0.21 ± 0.20	0.89 ± 0.13	0.95 ± 0.07	0.92 ± 0.10
aggregation	0.55 ± 0.01	0.29 ± 0.19	0.80 ± 0.06	0.79 ± 0.03	0.81 ± 0.08
cone torus	$\mathbf{0.37 \pm 0.03}$	0.15 ± 0.13	0.37 ± 0.04	0.36 ± 0.06	0.37 ± 0.06
checker board	0.12 ± 0.10	0.08 ± 0.05	0.14 ± 0.07	0.13 ± 0.06	$\mathbf{0.15 \pm 0.09}$
halfring	0.25 ± 0.01	0.24 ± 0.22	0.56 ± 0.03	$\mathbf{0.58 \pm 0.07}$	0.56 ± 0.03

In Table 2.15 we report ARI and Impurity Index (IMP) [GMT07] for dataset *tiselac* for BMFC, HBGF, CAtree, HGPA, MCLA. IMP indicates the number of differently

labeled objects in clusters and equals 0.0 for pure clusters in the resulting partition. Because of the large size of *tiselac* dataset ($N = 81715$) several methods that rely on the object co-occurrence matrix failed during execution due to the lack of memory (for evaluation we used a working station with 48GB RAM). Additionally, for this dataset we report average execution time.

Table 2.15: Evaluation results on tiselac dataset

	BMFC	HBGF	CAtree	HGPA	MCLA
ARI	$\mathbf{0.35 \pm 0.02}$	0.32 ± 0.01	0.31 ± 0.02	0.13 ± 0.09	0.29 ± 0.03
IMP	$\mathbf{0.43 \pm 0.03}$	0.48 ± 0.02	0.49 ± 0.03	0.46 ± 0.31	0.50 ± 0.06
Time, s	**2.1**	24.8	23.2	35.7	24.1

2.8.1.4 Experiments discussion

By analyzing the evaluation results we observe that the proposed BMFC demonstrates high operational characteristics. Figure 2.14 confirms the expected behavior of BMFC with different allowable error bounds showing that for large errors the number of discovered clusters is decreasing. This provides a useful mechanism to affect the number of clusters of the final solution when it is required. Figure 2.15 indicates that BMFC as well as the other methods are sensitive to the ensemble quality, however, for moderate noise level, BMFC demonstrates resistance to noise and provides acceptable results. From the all three experiments we observe that the proposed BMFC along with providing high operational performance yields low variance. This property indicates the proper choice of the proposed initialization technique that is able to bring the algorithm to the representative objects as it starts. An interesting observation on BMFC can be also done from Tables 2.14 and 2.15. While showing good results in terms of ARI on commonly used real-world and synthetic datasets, on large datasets *tiselac* BMFC clearly outperforms other methods both with respect to solution quality and execution time. This shows the potential of the proposed BMFC to be used on large-scale consensus clustering problems without performance degradation.

2.9 Summary

In this part of the thesis, we proposed two novel consenus clustering approaches that allow to efficiently combine multiple clustering solutions to obtain a single one. As the result, the consensus solution is more accurate and stable than any of the given

clustering solutions. By employing data fragment concept we developed a dissimilarity measure between these structures as well as proposed a distance measure between clustering solutions. This resulted into a novel conensus clustering framework for large scale problems. Additionally, we developed a binary matrix factorization-based consensus clustering method that by factorizing the label matrix allows to obtain interpretable consensus results. Be employing a recursive rank-one binary matrix factorization algorithm we adopted the framework to large datasets as well.

Simulation results for both methods demonstrated applicability of both proposed methods to wide range of clustering problems demonstrating more accurate final clusterings than other state-of-the-art consensus clustering methods. Particularly, the proposed approaches can be used to combine multiple weak clustering of a high dimensional large scale dataset.

Chapter 3

Data fusion from heterogeneous sources with ensembles of classifiers

Data fusion from multiple heterogeneous sources is a typical task for many multisensor applications including remote sensing classification problems. These challenges gained considerable importance due to the ability to collect and process large data volumes from a wide range of sensor types. Remote sensing the earth based on a multitude of different sensor technologies provides the means of measuring the earth in all its different facets going from visual over spectral and temporal characteristics to topological information like height [Suk+17b]. Through the past several years, the spatial resolution of all these sensors keeps improving dramatically providing an opportunity for precise land use and land cover classification of urban areas, detection of objects on Earth, identification of anomalies such as oil spillage, and others [Yok+18; MDH14; Xu+19]. Despite these sensor-level improvements, multisensor-based classification is still a very challenging task in which data fusion approaches take a prominent role [Tui+17]. The huge volume and high diversity of sensors that are used to acquire data impose severe limitations on building reliable classification systems: classification models (even the very complex ones) are often not able to embrace all ambiguous aspects of heterogeneous data [Kha+13; Jos+16]. Particularly, in remote sensing multisensor classification scenarios the data is usually acquired from satellite or airborne sensors providing a vast amount of highly dimensional dense data ranging from RGB to hyperspectral or LiDAR images [Deb+14; Suk+18a]. One of the biggest challenges arising in such applications is the way to fuse all the heterogeneous inputs in order to obtain the best possible classification results [Tui+17]. Unfortunately, besides the multidimensional nature of such heterogeneous data it is usually prone to have many outliers due to sensor failures or imperfections, heavy noise resulting from e.g. atmospheric effects, and often severe class imbalance due to high costs of the data labeling procedure or different class priors [Yok+18].

Recently, traditional data fusion and classification approaches such as SVM [CV95], Random Forests [Bre01] and different ensemble methods [MDH14] are being replaced by more sophisticated architectures of Neural Networks allowing to consider various aspects of multimodal data [Qin+19; Xu+19]. This, however, often imposes limitations on the amount of data required to train such systems while providing the significant potential of increasing classification performance.

One of the major remote sensing data fusion problems that are of interest nowadays is Local Climate Zones (LCZs) classification [Suk+17b; Yok+18]. LCZs classification task has been recently emerged as an important task for the remote sensing community and brought a variety of challenges in order to automate this process to be applied to urban sites all over the world. Continuous climate change and rapid population growth require an accurate, reliable, and generic automatic LCZ classification system in order to facilitate climate and landscape ecology studies, land use planning, weather forecasting, agricultural budgeting, and other activities relying on the precise land use information. LCZs were established as a standardized way to classify land use of rural and urban areas encompassing 17 classes that are meant to cover all possible landscapes on the globe. Formally, LCZ is defined as "a region of uniform surface cover, structure, material, and human activity that spans hundreds of meters to several kilometers on a horizontal scale [SO12]. Through the past few years, numerous studies on urban climatology acknowledged LCZ as a standardized means of describing the surface structure and land cover and proposed systems to create LCZ maps of various regions and cites [WZX15].

Another important data fusion challenge being addressed by the remote sensing community is land-use and land cover classification [Xu+19; Suk+18a; SDZ19]. The goal of this challenge is to automatically establish class labels that describe physical properties of land type or the way land area is used (e.g., water, earth, roads, residential area, etc.) [Zha+20; Den+19]. Formally, land cover data represents the amount of area that is covered by particular surfaces including different vegetation types and wetlands while land-use data provides information on how these areas are utilized [Chu20]. Such information is crucial for urbanization purposes to assess area development decisions and understand the possible effects of changes planned to implement [Xu+19]. It can significantly assist with the assessment of urban growth, help to understand human impact on nature as well as the influence of natural phenomena like floods or storms [Ton+20].

In the work on hyperspectral image classification [MDH14], Multiple classifier systems (MCS) were addressed from the bias-variance decomposition point of view and considered to be a crucial part of the general remote sensing classification framework that significantly affects its performance. Traditionally, the success of MCS is determined by two main aspects: the level of ensemble diversity and ensemble combination strategy [Kun02; Tul+08]. Classifier ensemble diversity has been extensively studied in the past giving the birth of powerful classifiers such as Random Forests [Bre01], Gradient Boosting Machines [Fri00] and others [WGC14]. Ensemble combination strategies that define the way the classifiers' outputs get aggregated received minor attention and has not been frequently considered in remote sensing classification scenarios. In particular, for LCZs classification problems, only the functions that exploit voting or score

averaging methodology (majority voting or averaging of confidence scores) were utilized [Kun02] providing decent average performance, though, being not able to capture various aspects of the problem such as class imbalances or issues with individual classifiers such as low recognition rate in a particular region of the feature space [WGC14].

In this thesis, we consider a dynamic selection (DS) framework to select and fuse competent classifiers of MCS. For this, we propose a competence estimation and selection method to improve the performance of the data fusion system, especially under class imbalance. We evaluate the method with synthetic and real datasets, demonstrating the applicability of the proposed framework.

3.1 Data fusion techniques overview

Data fusion techniques have been widely utilized in multisensor environments in order to fuse and aggregate data coming from different sources [MDH14]. It is often employed to provide more consistent and accurate output information than that provided by any single source. Data fusion techniques can be categorized into three groups: raw data level fusion that combines several sources of raw data, a feature-level fusion that joins meaningful features extracted from raw data, and decision level fusion that aggregates decision of individual systems [Cas13; Fau09]. The most common and intuitive way to solve the data fusion problems of all levels is to employ a classifier [Suk+17b; MDH14]. By concatenating multiple sensor data (or their related features) into a single data set or combining decisions of single source-related classifiers into finals solutions it possible to accurately solve data fusion tasks.

MCS [WGC14] have received significant attention also due to their ability to be a suitable framework for data fusion problems and proved to be more robust to data imperfections as well as to model hyperparameters, often being more accurate than a single classifier [Bre96]. MCS-based data fusion methods [MDH14] provide a natural way of solving heterogeneous multisensor classification problems by training a set of classifiers, each on their own data source (capturing various aspects of it) and then fusing the outputs of resulting classifiers. Solving data fusion tasks on the decision level has multiple advantages: it allows to build and then combine several simple models each responsible for its own data source rather than building a single complex model that tries to embrace all the input dimensions and that might be very difficult to train (many parameters to optimize with insufficient data); it provides the ability to change one of the sources (e.g. due to sensor replacement) and retraining only the model that is responsible for that data source [Fau09; MDH14].

For remote sensing classification problems, it is typical to have a large number of data samples as well as high a dimensional feature space. Hyperspectral, multispectral, LiDAR, and other data types (e.g. crowd-sourced auxiliary site information) used in remote sensing problems usually exhibit a significant amount of dimensions (often correlated), contain noise and outliers [SO12]. All these factors bring substantial difficulties for building classification systems including ones that utilize MCS paradigm [Chu20].

The typical MCS-based approaches that are being used currently in remote sensing classification problems is to separately train a set of heterogeneous classifiers and then combine them using one of the non-trainable fusion functions.

In this chapter, we introduce a novel ensemble combination framework based on a dynamic classifier selection strategy. We propose a combination function for improving the overall performance of MCS, especially, in the presence of class imbalance problem [Suk+18b] that is typical for remote sensing applications. We extensively study the most widely used combiners, pointing at their limitations and constraints. The important outcome of our study is that the choice of a proper ensemble combination method is able to provide a significant increase in the overall performance for the remote sensing problems, particularly for LCZs and land cover classification tasks. This, in turn, can be extended to other general remote sensing classification scenarios.

3.2 Multiple classifier systems

In this section, we introduce the concept of MCS considering its two main aspects: ensemble generation and combination. Then we focus on the combination functions discussing their applicability to remote sensing classification scenarios.

Consider a classification problem where $\{(\mathbf{x}_i, y_i)\}$, $i = 1, ..., N$ is the set of N objects, $\mathbf{x}_i \in \mathbb{R}^d$ is a feature vector produced by concatenation of vectors of lower dimensions $[\mathbf{x}_i^1, \mathbf{x}_i^2, \ldots, \mathbf{x}_i^T]$ each representing one out of T heterogeneous data sources ($d = \sum_{t=1}^{T} d_t$) and $y_i \in \Omega = \{\omega_1, \omega_2 \ldots, \omega_L\}$ is a corresponding label (or class). In remote sensing applications such as land use classification, the objects are usually the multidimensional points on a 2d grid while feature vectors might be raw reflectance values over hyperspectral/multispectral/visible bands, multispectral LiDAR data or higher level features. Let ψ be a classifier

$$\psi : \mathbf{x} \to \Omega \tag{3.1}$$

that produces a vector of discriminant functions $\mathbf{s}(\mathbf{x}) = [s_1(\mathbf{x}), s_2(\mathbf{x}), \ldots, s_L(\mathbf{x})]$. The value of $s_l(\mathbf{x}), l \in \{1, \ldots, L\}$ is a support given by the classifier for the fact of $\mathbf{x}$

belonging to class ω_l known also as certainty score or for some classifiers an estimate of posterior probability $P(\omega_l|\mathbf{x})$. Without loss of generality we assume $s_l(\mathbf{x}) \geq 0$ and $\sum_{l=1}^{L} s_l(\mathbf{x}) = 1$. Classification is then performed according to the max rule

$$\psi(\mathbf{x}) = \omega_l = \underset{l \in \{1,\ldots,L\}}{\operatorname{argmax}}\, s_l(\mathbf{x}). \tag{3.2}$$

Let a pool (ensemble) of T classifiers be given as $\mathcal{A} = \{a_1, \ldots, a_T\}$ where every classifier is trained on its own feature subspace $\mathbb{R}^{d_t}$. The fusion of classifiers' decisions in the pool $\mathcal{A}$ is given by a combination function $\mathcal{F}$ as:

$$f(\mathbf{x}) = \mathcal{F}(\mathbf{s}_1(\mathbf{x}), \mathbf{s}_2(\mathbf{x}), \ldots, \mathbf{s}_T(\mathbf{x})) \tag{3.3}$$

where $\mathbf{s}_t, t \in \{1, \ldots, T\}$ is the vector of discriminant functions produced by the classifier t. For the majority voting, which is the most frequently used combination function, it has been shown to appear as Bayes optimal (i.e. providing zero error rate for infinite ensemble cardinality) when all ensemble members make independent errors and have above random classification rate [Kun02]. It is known that for a binary classification problem under the normal distribution of confidence scores the classification error of an ensemble with averaging of confidence scores as combination function is [Kun02]:

$$P^{a}_{\text{error}} = \Phi\left(\frac{\sqrt{T}(0.5 - p)}{\sigma}\right) \tag{3.4}$$

with mean p (that is the probability of correct classification) and variance σ^2, where Φ is the standard normal cumulative [Zou+18].

3.2.1 Ensemble generation

Ensemble generation plays a crucial role in the overall performance of MCS. The major requirements for ensemble members in a MCS are diversity and accuracy [Hud+13]. While intuition behind accuracy is apparent, the motivation for diversity comes from the requirement for classifiers to make different errors (otherwise in case of the same errors there is no gain possible while combination) [Cho+15; RR20]. Generally, diversity can be achieved in several ways: various training samples or feature sets (bagging and subspace methods, correspondingly) [Suk+18b; Suk+15], various classifier models or parameters of the same models (including architectures and initialization) and many others [Cho+16]. To guide the ensemble generation mechanisms there are several measures of diversity including entropy measure, Kohavi-Wolpert Variance, Measure of difficulty, etc. [Hud+13; Cho+16]. In the remote sensing classification research, traditionally, diversity is usually achieved by utilizing a set of diverse learning models [Ngu+20]. Normally, this option is dictated by the lack of labeled training data and

extremely high dimensionality of feature space making bagging and subspace methods unpractical. To be aligned with the current practice, in this thesis, we consider diverse learning models as an ensemble generation method.

3.2.2 Ensemble selection and combination

The function used to select and combine ensemble members is another extremely important component of MCS. Combination functions (also known as aggregation functions) fuse the outputs obtained by ensemble members according to a predefined operator that is performed on the class labels (e.g. majority voting scheme) or scores provided by every classifier for each class. In this thesis, we consider combination functions operation directly on scores instead of the class labels to encompass generic mathematical treatment of classifier combination problem. In general, class labels can be considered as binarized scores.

Selection of classifiers might be done independently from a combination or be a part of the fusion function. The goal of selection is to either select the most diverse and accurate classifiers and then perform aggregation only on them or perform dynamic selection based on the input object $\mathbf{x}_i$. The wide two classes of combination functions are non-trainable and trainable combiners.

3.2.2.1 Non-trainable

Non-trainable combiners are the most popular combiners due to their simplicity spanning several of combination techniques. In this thesis, we consider generalized mean as a general non-trainable combination scheme that is formulated as:

$$\mu_l(\mathbf{x}, \alpha) = \left(\frac{1}{T} \sum_{t=1}^{T} s_{t,l}(\mathbf{x})^{\alpha} \right)^{\frac{1}{\alpha}} \tag{3.5}$$

where $s_{t,l}$ is the support given for class ω_l by classifier t and $\mu_l(\mathbf{x}, \alpha)$ is the overall support of class ω_l. The choice of α results into different combination rules. For $\alpha \to \inf$ we obtain the maximum rule, for $\alpha \to -\inf$ we obtain minimum rule and for $\alpha \to 1$ we obtain the mean rule. Mean rule is considered to be the most common combination technique in MCS including remote sensing scenarios. In this strategy, the support for each class is obtained as the average of all classifiers j^{th} output. Note should be taken that the mean rule is equivalent to the sum rule only differencing by a normalization factor $\frac{1}{T}$.

The reason for wide utilization of the mean rule is the simplicity of this fusion function that yields low chance of overfitting [Kun02]. On the other hand, it has several substantial drawbacks: in this scheme all base classifiers (including poor and correlated ones) are considered and treated equally that might result in the poor overall performance. Moreover, for binary classification problem the overall ensemble error (under the Gaussianity assumption of scores distribution) is

$$P_e^{m,n} = \Phi\left(\frac{\sqrt{T}(0.5-p)}{\sigma}\right) \tag{3.6}$$

that is different to a single classifier error by the factor of $\sqrt{T}$ in the error function. Obviously, the more the number of classifier are in the pool, the lower the overall error (that theoretically might go even to zero). On the other hand, practically, it is hard to obtain independent classifiers since all of them are usually utilizing the same training dataset that limits diversity.

3.2.2.2 Oracle

Oracle [Kun02] is an important abstract fusion model that picks the classifier (if it exists) that outputs the correct class. In the literature, it is usually regarded as the possible upper bound for the performance of a pool of classifiers [CSC18b]. For Oracle the classification error for normal distribution of scores for two-class problem is:

$$P_e^{o,n} = \Phi\left(\frac{0.5-p}{\sigma}\right)^T \tag{3.7}$$

that is significantly lower comparing to any other fusion techniques. Obviously, this setting seems unrealistic since there is no knowledge in advance which ensemble member makes correct classification on which data point. On the other hand, Oracle provides useful concept to obtain an upper limit performance for a set of classifiers. The Oracle performance is usually regarded as a possible upper-bound for MCS and represents the perfect DS scheme.

3.2.2.3 Dynamic classifier and ensemble selection

One of the most crucial aspects when solving data fusion problems by MCS is the combination function that defines the way the classifiers' outputs get aggregated [Kun02; Tul+08]. Functions that exploit voting or score averaging methodology usually provide decent average performance though are not able to capture various aspects of the

problem such as class imbalances [Suk+18b] or issues with individual classifiers such as low recognition rate in a particular region of the feature space.

Dynamic Selection (DS) [CSC18a] techniques appeared in MCS as a way to address the problem of varying classifier performance with regards to particular regions of feature space. DS is considered to be one of the most promising MCS selection and combination techniques. Motivated by the Oracle, dynamic classifier selection methods pick a classifier (or multiple classifiers from the pool in case of dynamic ensemble selection) based on the region of competence for every object $\mathbf{x}_i$ to classify. For that, the competence of every classifier within the ensemble is estimated for a set of predefined regions. The classifiers that are competent in the region to which the object to classify belongs to are selected to be further combined. Classifier competence estimation in DS techniques plays a crucial role and in the case of poor estimation might result in the adverse selection of classifiers leading to non-optimal performance of the overall MCS [Gal+12]. One of the reasons for that issue is the fact that in many DS methods the classifier performance in local regions is estimated disregarding the class distributions. This aspect becomes critical when the data is highly imbalanced (the distribution of classes is heavily skewed towards some of them [Gal+12]) that is typical especially in many data fusion remote sensing scenarios [Deb+14]. Figure 3.1 shows a typical data fusion flow in the generalization phase. According to Figure 3.1, the dynamic selection mechanism

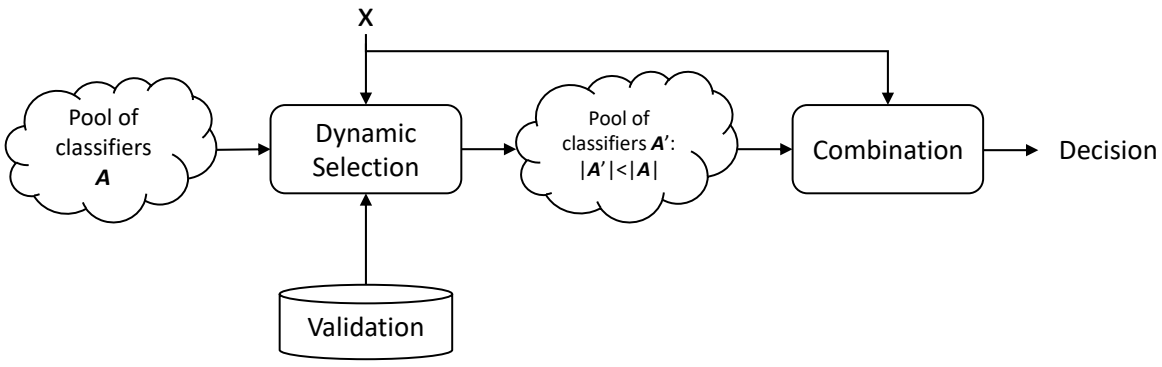


Figure 3.1: Dynamic selection flow

picks classifiers from the pool $\mathcal{A}$ according to their local performance that is typically estimated during the validation process. After that, selected classifiers are combined according to a particular rule outputting the final classification decision.

DS involves three major steps:

- region of competence definition - the way of defining the local region around the object $\mathbf{x}_i$ to classify. Formally, the feature space $\mathbb{R}^d$ is divided into K non-overlapping regions $\mathcal{R} = \{R_1, R_2, \ldots, R_K\}$. Figure 3.2 depicts an example of such regions in a 2d feature space where the space is divided into 7 regions;
- local competence estimation - a procedure of estimating the level of competence of each classifier in all regions $\mathcal{R}$;
- selection mechanism - the method that selects classifiers based on their local competence level and combines them together.

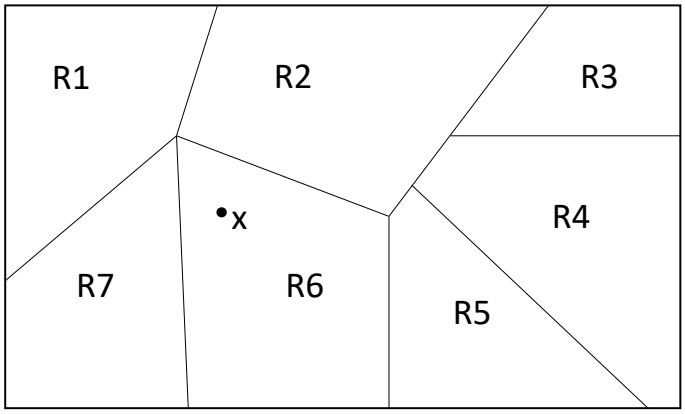


Figure 3.2: Local regions in a feature space for dynamic selection

Depending on the DS mechanism the error of a DS-based combiner P_e^{DS} might vary significantly achieving the error of Oracle P_e^o in case of the perfectly established region of competence and precisely estimated performance of each ensemble member in these regions. However, due to the extremely high dimensional nature of remote sensing data establishing a meaningful region of competence is a non-trivial task. Additionally, due to severe class imbalance, presence of noise, and outliers it is often a challenge to accurately estimate the competence of each classifier in a pool for every region that the feature space is split into.

3.3 Dynamic ensemble selection scheme for Data Fusion

In this section, we introduce our proposed DS-based MCS fusion method that consists of a classifier competence level estimation and a selection technique. Based on the

competence level of each region the classifier selection and combination processes are performed. In the following, we detail the steps of the proposed method.

Consider the optimal Bayes classifier and let $P(\mathbf{x}|\omega_l)$ be the conditional pdf of $\mathbf{x}$ belonging to class ω_l that has the prior probability $P(\omega_l)$. For the optimal Bayes classifier it is known that the probability of correct classification is

$$P_{\text{correct}} = \sum_{l=1}^{L} \int_{R_l} P(\mathbf{x}|\omega_l) P(\omega_l) d\mathbf{x} \tag{3.8}$$

and reaching its maximum when the decision regions R_l are established to maximize the integrands. In real applications the estimation of class posteriors is a challenging task often leading to classifiers with non-Bayesian decision regions. Given an ensemble of classifiers $\mathcal{A}$, let $\mathcal{R}$ be a set of regions spanning the original feature space $\mathbb{R}^d$ and let Σ map each object $\mathbf{x}$ to its corresponding region R_k:

$$\Sigma : \mathbf{x} \rightarrow R_k. \tag{3.9}$$

Let the classifier a_t be competent for the region R_k. The classifier decision is then considered in the combination function $\mathcal{F}$ if $\Sigma(\mathbf{x}) = R_k$. Denote $P(a_t|R_k)$ as the probability of correct classification by a_t in region R_k and $a_{t,k}$ the classifier responsible (possessing enough competence to be used for classification) for region R_k. Then the overall probability of correct classification is:

$$P_{\text{correct}} = \sum_{k=1}^{K} P(R_k) P(a_{t,k}|R_k) \tag{3.10}$$

where $P(R_k)$ is the probability that an object $\mathbf{x}$ appears in region R_k. In order to maximize P_{correct} the classifier a_t has to be assigned to be competent in region R_k so that

$$P(a_{t,k}|R_k) \geq P(a_{t'}|R_k), \forall t' = 1, \ldots, T \tag{3.11}$$

and ties are broken randomly. In the end, we get:

$$P_{\text{correct}} \geq P(a^*) = \sum_{k=1}^{K} P(R_k) P(a^*|R_k) \tag{3.12}$$

where a^* is the classifier with the highest average accuracy among all classifiers in the ensemble $\mathcal{A}$ over the feature space $\mathbb{R}^d$.

By imposing the assumption for decision level data fusion that every ensemble member is established by training a classifier on a feature subset $\mathbb{R}^{d_t}$ a classifier a_t is then responsible for its individual feature subspace $\mathbb{R}^{d_t}$. The condition (Eq. (3.12)) to maximize P_{correct} remains the same except the region of competence R_k that becomes R_{t_k}.

To achieve optimal classification performance according to Eq. (3.12) it is necessary to know the probability of correct classification of every classifier in its region. Since this is typically not available, the only possibility to approximate behavior in Eq. (3.12) and maximize average accuracy for the ensemble is by accurately estimating competence of every classifier a_t in its feature space $\mathbb{R}^{d_t}$. In the sequel, we introduce a classifier competence estimation and selection scheme.

The concept of combining classifiers each trained on its subspace $\mathbb{R}^{d_t}$ is similar to Random Forests [Bre01] though the main difference of our approach is that we combine classifiers using DS scheme while Random Forests employs majority voting that considers all ensemble members equally.

3.3.1 Classifier competence estimation and selection

Let the feature space R^{d_t} be partitioned into t_K regions $\{R_{t_1}, R_{t_2}, \ldots, R_{t_K}\}$. In every region the competence of a classifier a_t for the object $\mathbf{x} \in R_{t_k}$, $k \in \{1, 2, \ldots, t_K\}$ is to be estimated. For that, we propose to calculate the weights for every class $l \in L$ according to their class distributions:

$$w_l = \left(N_l \sum_{j=1}^{L} \frac{1}{N_j} \right)^{-1} \tag{3.13}$$

where N_l is the number of samples belonging to class l in the dataset. According to Eq. (3.13) classes that have larger sampling support get smaller weight.

To estimate the competence level of classifiers, the class-specific accuracy, i.e. the accuracy of the class that classifier a_t is predicting correctly for in R_{t_k} is estimated according to

$$\hat{P}_{\omega_l}(a_t|R_{t_k}) = \frac{\sum_{\mathbf{x}_i \in \omega_l} P(\omega_l|\mathbf{x}_i, a_t)}{\sum_{i=1}^{M} P(\omega_l|\mathbf{x}_i, a_t)} \tag{3.14}$$

where M is the number of data points belonging to class ω_l in the region R_{t_k}.

On the prediction stage the activation indicator I_t^l for classifier a_t for each class ω_l is defined as:

$$I_t^l = \begin{cases} 1, & \text{if} \quad \hat{P}_{\omega_l}(a_t|R_{t_k}) - 0.5 w_l > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.15}$$

where 0.5 is the probability of correct classification of a binary random classifier. It was shown earlier that the performance of a random classifier (the one that draws a

class label from uniform distribution) provides a natural selection criteria for choosing the competent classifier for combination [Wol+12]. However, to account for a chance of rare class occurrence, we multiply the performance of the random classifier by the class weight w_l lowering the competence threshold for rare classes in case of their successful local discovery. Note, that the I_t^l guarantees participation of classifier a_t in combination only if it has sufficient competence for region R_{t_k}. In case, there are no competent classifier selected then all ensemble members are taken to submit a single vote each.

As the last step, the ensemble of classifiers is combined according to weighted averaging voting scheme as follows:

$$\mu_l(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} I_t^l s_{t,l}(\mathbf{x}) \tag{3.16}$$

where μ_l is the total support of the ensemble given to class ω_l. The final class label and classification result is defined according to the max rule:

$$\psi(\mathbf{x}) = \omega_l = \underset{l \in \{1,\ldots,L\}}{\operatorname{argmax}} \mu_l(\mathbf{x}). \tag{3.17}$$

To establish the region of competence multiple techniques are utilized including clustering [Mac67], K-Nearest Neighbors [CSC18b], potential function model [BSO14] and decision space [CSC18b; BSO14]. The common assumption for these methods is that classes maintain a certain continuity in the feature space.

3.4 Experimental Results

To evaluate the performance of the proposed method we perform numerical simulations as well as experiments with real-world data comparing the proposed methods (further denoted as DES-B) to two widely used DS techniques LCA [WKB97] and DESP [Wol+12] as well as combination method that averages the scores of all ensemble members (AVG). Note, that LCA and DESP had to be adapted to suit data fusion problems since both of them originally consider unified feature spaces to estimate classifier competence. In all experiments, the dataset was split into training, validation and testing subsets (60/20/20) and to establish the region of competence a k-NN with $k = 7$ was used on validation subset.

3.4.1 Numerical simulation 1

In this simulation, we generate seven $L = 4$-class classification datasets each with $N = 10^5$ points by creating clusters (two per class) of data points following a normal distribution about vertices of an $d = 500$-dimensional hypercube with sides of length 1. For $N_e = 2 \cdot 10^3$ samples the classes are randomly exchanged. Additionally, $d_{\text{rnd}} = \lceil 0.25d \rceil$ random features are generated. For every dataset i we generate $L + i$ classes and then randomly combine i of them in order to get $L = 4$-class imbalance dataset with particular class imbalance level that we define as the ratio between the average of class cardinalities excluding the biggest class to the cardinality of the biggest class. As the base classifier, we use a Random Forests classifier with 100 trees and measure accuracy and F1 measure [PUG14] over $c = 10$ Monte Carlo iterations reporting also the standard deviation of metrics. Additionally, we report the performance of the classifier that was trained on the whole feature space $\mathbb{R}^d$ (denoted as CLFRd) in Figure 3.3.

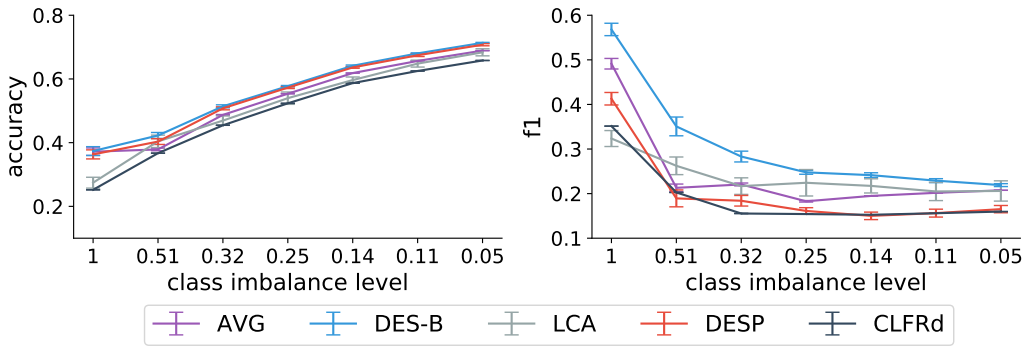


Figure 3.3: Accuracy and F1 measures with standard deviation with respect to class imbalance level on seven artificially generated datasets over 10 Monte Carlo iterations

3.4.2 Numerical simulation 2

In this simulation, we generate four balanced datasets following the same procedure as in the previous simulations except of classes merging (to keep datasets balanced). We set particular parameters of the generator (number or data points, ensemble size, number of classes, number of informative dimensions) in order to cover multiple possible scenarios. For this simulation, we report average accuracy over $c = 10$ Monte Carlo iterations (Table 3.1).

	$N = 10^4, T = 10$ $L = 2, d = 300$	$N = 10^5, T = 3$ $L = 5, d = 150$	$N = 4 \cdot 10^4, T = 2$ $L = 10, d = 40$	$N = 10^4, T = 5$ $L = 7, d = 25$
CLFRd	0.783 ± 0.021	0.608 ± 0.019	0.434 ± 0.023	0.208 ± 0.014
AVG	0.822 ± 0.015	0.612 ± 0.012	0.430 ± 0.012	$\mathbf{0.238 \pm 0.009}$
LCA	0.767 ± 0.011	0.581 ± 0.015	0.391 ± 0.011	0.200 ± 0.013
DESP	0.780 ± 0.019	0.631 ± 0.011	0.418 ± 0.011	0.187 ± 0.015
DES-B	$\mathbf{0.851 \pm 0.017}$	$\mathbf{0.657 \pm 0.014}$	$\mathbf{0.451 \pm 0.008}$	0.233 ± 0.010

Table 3.1: Accuracy measure with standard deviation on four balanced artificially generated datasets over 10 Monte Carlo iterations

3.4.3 IEEE GRSS 2017 Data Fusion dataset experiment

In this experiment, we consider the original dataset provided within the context of the 2017 IEEE GRSS Data Fusion Contest [Tui+17; Yok+18] that comprises several city sites from different parts on the globe. The train and the test parts of the dataset are partitioned on a geographical basis: five city sites are provided with LCZ labels to be able to use them for training while the data for the other four city sites are aimed for testing purposes. Each city site is covered by a set of images from the Landsat-8 and Sentinel-2 satellites with up to 9 multispectral bands and a pixel resolution of 100x100m^2. For the training city sites, annotated LCZ labels are provided on a pixel by pixel basis and represent sparsely distributed segments. In addition to that, the dataset contains a few layers of crowd-sourced data extracted from OpenStreetMap[1] (OSM). OSM layers provide detailed information, encoded by integer values, for water, building and land-use areas.

We enriched the original dataset with auxiliary information gathered from publicly available sources [Suk+17b]. The EarthExplorer[2] system was used for the extraction of additional Landsat-8 images that aimed to provide temporal diversity. Here approximately six to seven additional images have been manually selected from 2015 and 2016. In order to be more flexible on a feature engineering step we also re-extracted OSM information and superposed it to the original images. One of the major issues in crowd sourced data is a scarcity of information for some specific regions of the world.

In order to leverage geoinformation from OpenStreetMap a dedicated feature extraction pipeline is employed. The pipeline starts from the extraction of an image projection coordinate from metadata of the Sentinel-2 images. It allows us to superpose an image of the satellite with geographical coordinates. Based on these coordinates the grid

[1] http://www.openstreetmap.org
[2] https://earthexplorer.usgs.gov

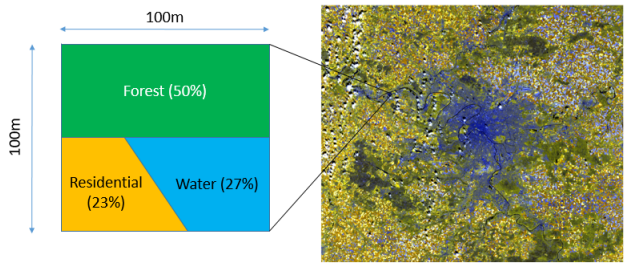


Figure 3.4: A satellite image of Paris metropolitan area is presented on the right part of the figure while the left part of the figures shows an example of a pixel polygon of the image overlapped with OSM layers. The pixel polygon contains relative coverage by a land use layers where 100% is equal to 10000 m^2.

of pixel-related polygons is created for each metropolitan area provided in the data set. Each pixel polygon (100x100m^2) is overlapped with the OSM layers in order to calculate a land use coverage in relation to the pixel area, as illustrated in Figure 3.4. Additionally, the following features are extracted for each pixel polygon: number of buildings located in the polygon, average and maximum floor number of the buildings, average and maximum height of the buildings.

The following set of features were extracted from the multispectral images:

- NDWI: Normalized Difference Water Index[3]
- NDVI: Normalized Difference Vegetation Index[3]
- NDMI: Normalized Difference Moisture Index[3]
- BSI: Bare Soil Index[3]
- AVI: Advanced Vegetation Index[3]
- SI: Shadow Index[3]
- SAM: Spectral Angle Mapper [Osh+13]
- MNF: Minimum Noise Fraction [LWB90]
- OSM: Open Street Map

Each feature represents a 2 dimensional matrix except of the last three features which represent a 3 dimensional matrix where the third dimension equals to the number of

[3]www.spaceanalyzer.com/index.php/advanced-vegetation-index-avi

bands for MNF, the number of class labels for the SAM and the number of unique ids for the OSM (one-hot encoding). The SAM calculates the spectral angle between the image spectra and a known spectra or endmember. It is insensitive to illumination since it uses the vector direction rather than the vector length. The Minimum Noise Fraction transform computes the normalized linear combination of the original bands which maximizes the ratio of the signal to noise. For each class a SAM feature has been calculated. The reference signature is calculated as the mean spectrum over all the pixels belonging to the respective class.

Since only one Sentinel-2 image was provided per site, but two or more Landsat-8 images, we decided to stack the Sentinel-2 image behind each available Landsat-8 image. In this way an extended 3D data cube was constructed with a fixed spatial dimension per site and a spectral band dimension equal to the number of spectral bands from the Landsat-8 plus the number of spectral bands from the Sentinel-2 image. In the end, the spatial dimension is unfolded into a single dimension reflecting the amount of training/testing samples. We denote described set of features as high level features.

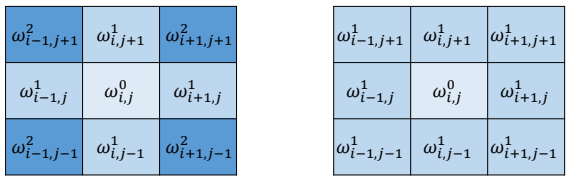
The pool of classifiers consists of three models that are described in the following.

As one of the based models Convolutional Neural Networks (CNN) [Lec+98] were applied, with a small amount of labeled training data.

The set of input features for CNN are the raw Landsat-8 images (9 channels) and the raw Sentinel-2 images (10 channels), normalized so as to have zero mean and unit variance on the training dataset on each channel independently.

Note that several Landsat-8 images are available for each city: the ones provided by the original dataset and the ones downloaded from EarthExplorer. Each combination of a given Landsat-8 image and the Sentinel-2 image of a city is an input to a CNN.

In order to tackle overfitting, we used a rather shallow architectures and a new type of convolutional layer with more parameter sharing. The first layer is a 1x1 convolution with batch normalization and a *tanh* or ReLU [NH10] non-linearity. The second layer is a convolutional layer with tied weights so that the kernel is symmetric with respect to the origin, in order to increase invariance to rotation and share more parameters [Le+10]. We used 3x3 kernels with 2 or 3 trainable weights for each combination (i, j) of input and output channels, distributed as depicted on Figure 3.5. The output of the convolutional layer is connected directly to the softmax layer to produce posterior probability estimates for all the classes to detect.



(a) Kernel with 2 weights

(b) Kernel with 3 weights

Figure 3.5: 3×3 kernels used in the convolutional layer of our CNN

The categorical cross-entropy loss function can be evaluated on locations where the zone is labeled in the training dataset. The average loss was minimized using Stochastic Gradient Descent [Lec+98] with minibatches of snippets of arbitrary size 100x75 randomly cropped from original images. The learning rate was tuned to achieve the best performance with fast convergence. To improve generalization, we used dropout [Sri+14a] which acts like a data-driven regularization method.

RF [Bre01] was used as another base classification model. RF is an ensemble-based classifier that creates and then averages over multiple decision trees created using bagging and random feature selection techniques. The diversity between weak learners allows to prevent overfitting while aggregation improves predictive performance. We employed RF classifier with all high level features described above along with hyper-parameter tuning technique to find the optimal model parameters. Due to the high class imbalance we assigned custom weights w_l to every class l inversely proportional to class frequencies in the training data:

$$w_l = \frac{N}{L \cdot N_l}, \forall l = 1, \ldots, L \tag{3.18}$$

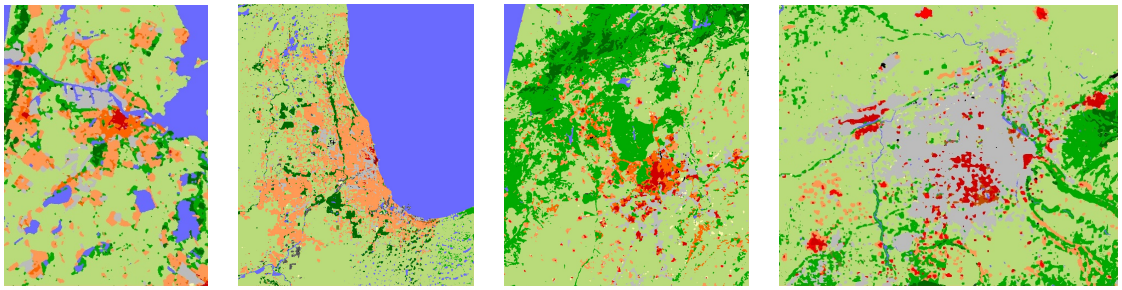
where N is the number of training samples, L is the number of classes and N_l is the number of samples of class i. These weights are then used to weight the Gini criterion for finding splits and also in terminal nodes of each tree to perform weighted majority voting. In that setting, underrepresented classes get higher significance while training and can then be better discovered during prediction.

Gradient Boosting Machines (GBM) [Fri00] were used as the third base model. GBM is the other classification method from the family of tree-based algorithms. Similarly to RF it employs the idea of weak learners ensembling. However, in contrast to RF that can grow classification trees in parallel, GBM builds an ensemble of trees iteratively by minimizing a custom cost function using a gradient decent optimization algorithm. All trees on every iteration are added to the ensemble and participate in the classification.

	CNN	RF	GBM	AVG	LCA	DESP	DES-B
Accuracy	0.82	0.81	0.80	0.79	0.82	0.83	**0.87**
F1	0.81	0.80	0.80	0.80	0.82	0.81	**0.85**

Table 3.2: IEEE GRSS 2017 Data Fusion Contest dataset results

In Table 3.2 provides the overall validation accuracy and F1 measures for the proposed DES-B, CNNs, RF, GBM, LCA, DESP and AVG methods. Figure 3.6 demonstrates final classification results of the proposed DES-B.



(a) Amsterdam (b) Chicago (c) Madrid (d) Xi'an

Figure 3.6: Resulting LCZ maps for the test cities after post-processing

3.4.4 IEEE GRSS 2018 Data Fusion dataset experiment

In this experiment, we consider the data set provided by IEEE GRSS 2018 Data Fusion Contest technical committee [DFC] that contains RGB, Multispectral LiDAR and Hyperspectral imagery (HSI) covering the University of Houston campus and its surrounding areas. The training set consists of labeled segments distributed over a raster of 4768×1202 pixels spanning $M = 20$ unique urban land use and land cover classes (e.g. residential buildings, roads, artificial turfs, cars, etc.) that have strongly unequal distribution. The test set covers a region of 8344×2404 pixels having $341,729$ of them labeled.

To establish the pool of classifiers we trained three Convolutional Neural Networks (CNNs) separately on its own data source: $\mathrm{CNN}_{\mathrm{LiDAR}}, \mathrm{CNN}_{\mathrm{HSI}}, \mathrm{CNN}_{\mathrm{RGB}}$. For every data source, the CNN input was represented by a $S \times S$ multidimensional patch where $S = (2C + 1)R$, C is the context constant and R is a data source scalar ($R_{\mathrm{HSI}} = R_{\mathrm{LiDAR}} = 1, R_{\mathrm{RGB}} = 10$). The branch architecture for $\mathrm{CNN}_{\mathrm{LiDAR}}, \mathrm{CNN}_{\mathrm{HSI}}$ includes (Figure 3.7):

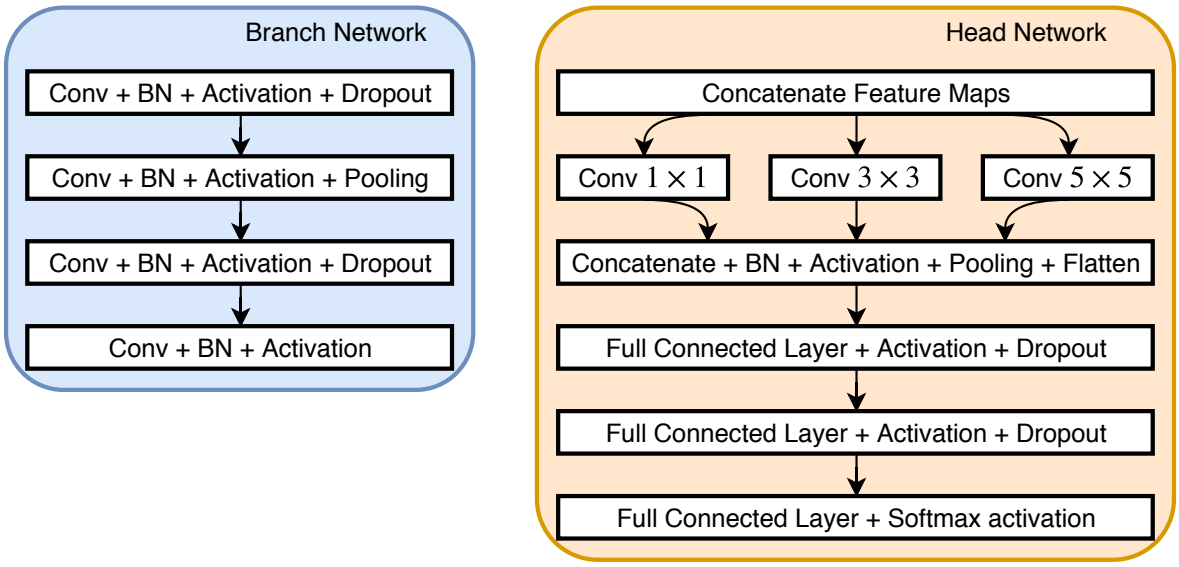


Figure 3.7: Architecture of Branch Network (left) and Head Network (right)

- 64 convolution filters (CF) with kernel size $i \times i$ followed by Batch Normalization (BN) [IS15], activation f_{act} and Dropout [Sri+14b] with probability;
- 64 CF $(i \times i) \Rightarrow$ BN $\Rightarrow f_{act} \Rightarrow$ Max Pooling;
- 128 CF $(i \times i) \Rightarrow$ BN $\Rightarrow f_{act} \Rightarrow$ Dropout;
- 128 CF $(i \times i) \Rightarrow$ BN $\Rightarrow f_{act}$.

Branch networks were created by varying filter size $i \in \{1, 3, 5, 7\}$ producing feature maps representing different local receptive fields (Figure 3.8). A Head Network concatenates created feature maps and performs additional convolutional and dense transformations and outputs class probabilities. Categorical cross-entropy loss function was used to train the CNNs. $\text{CNN}_{\text{LiDAR}}$ Branch Networks were created for each i and a single Head Network resulting into two CNNs with $C \in \{7, 11\}$ and an Exponential Linear Unit (ELU) as f_{act}. The outputs of these two models were combined by averaging posterior probabilities. For CNN_{HSI} two sets of Branch Networks were used: for $C = 3$ and for $C = 7$ followed by single Head Network having $f_{act}(x) = ReLU(x) = max(0, x)$ resulting in three CNNs that were later combined by averaging of posterior probabilities. For CNN_{RGB} the Xception architecture [Cho17] was trained on patches with $C = 7$. For all networks $p_{\text{dropout}} = 0.2$. There was no postprocessing done on the final output in order to provide fair comparison of the fusion functions [Suk+18a].

In Table 3.3 the overall accuracy and F1 measures on the test set are provided for the proposed DES-B, individual CNNs, LCA, DESP and AVG.

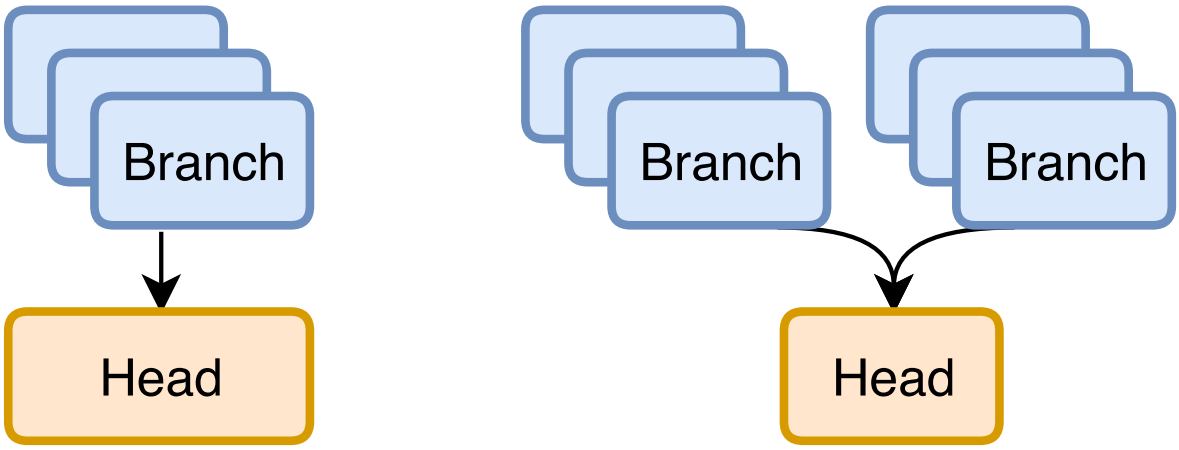


Figure 3.8: (left) Lidar Network architecture draft, (right) HSI Network architecture draft

	CNN_{LiDAR}	CNN_{HSI}	CNN_{RGB}	AVG	LCA	DESP	DES-B
Accuracy	0.69	0.47	0.63	0.65	0.69	0.69	**0.71**
F1	0.67	0.52	0.64	0.67	0.67	0.68	**0.70**

Table 3.3: IEEE GRSS 2018 Data Fusion Contest dataset results

3.4.5 Discussion

By analyzing the evaluation results on all experiments we observe that method based on ensemble paradigm achieve, in general, higher classification accuracy than methods relying on a single classifier. Moreover, the proposed DS technique achieves on average higher performance than other state-of-the-art methods, especially in case of class imbalance. This gain is due to the strategy of estimating classifier competence as well as the way the classifiers' decisions are combined: the adaptive classifier competence threshold allows to keep minority class discovery even when high class imbalance exists. In the first synthetic experiment, it becomes clear that the performance of all except the proposed methods drops in case of severe class imbalance since none of the classifiers in the ensemble is able to discover minority classes. The experiments with real-world remote sensing datasets prove that the proposed DES-B is able to solve data fusion problems discovering small classes while keeping the overall performance on a high level.

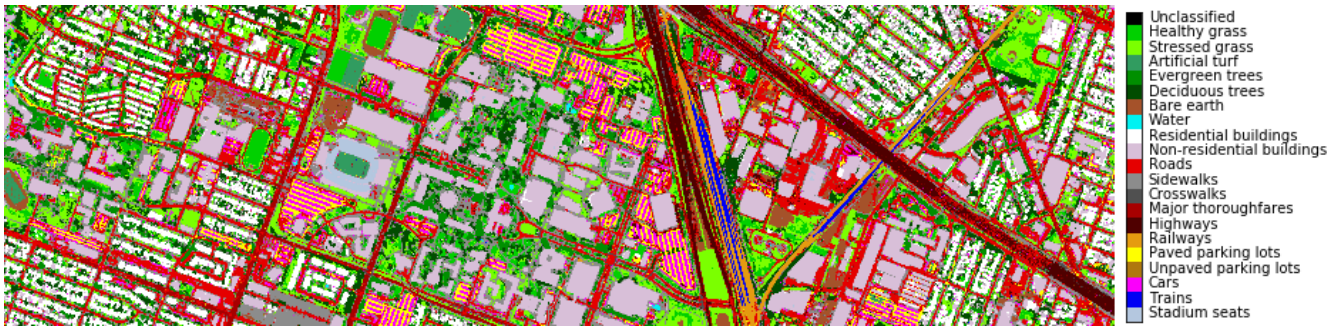


Figure 3.9: Prediction map on the test set of IEEE GRSS 2018 data fusion contest

3.5 Summary

In this part of the thesis, we proposed a novel dynamic ensemble selection scheme that allows us to efficiently combine multiple heterogeneous data sources providing accurate classification results even with high class imbalance. Simulation results demonstrated the applicability of the proposed dynamic selection approach to various class imbalance degrees while experimental results on several remote sensing problems demonstrated the superiority of the proposed method with respect to other state-of-the-art dynamic selection approaches. Particularly, we showed that the proposed approach can be used to combine data sources of different nature, spatial resolution, and scarcity levels providing powerful framework to solve local climate zones classification as well as land use and land cover classification problems.

Chapter 4

Dynamic Pattern Matching with Multiple Queries on Data Streams

Similarity search in data streams is an important but challenging task in many practical areas where real-time pattern retrieval is required. Dynamic and fast updating data streams are often subject to outliers, noise, and potential distortions in amplitude and time dimensions. Such conditions typically lead to a failure of existing pattern matching algorithms and to the inability to retrieve required patterns from the stream. The main reason for such failures is the limitation of data normalization utilized in the majority of methods. Another reason is the lack of means to consider multiple examples of the same template to account for possible variations of the query signal. In this chapter, we propose a dynamic normalization approach that allows bringing streaming signal subsequences to the scale of the query template. This significantly improves pattern retrieval capabilities, especially when sampling variance or time distortions are present. We further develop a pattern matching approach utilizing the proposed normalization mechanism and extend it for the case when multiple examples of a query template are available. Multiple synthetic and real data experiments demonstrate that this allows to considerably improve the pattern matching rate for distorted data streams, providing real-time performance.

4.1 Introduction

The problem of pattern matching in real-time data streams (also known as similarity search) has recently emerged as an important task for signal processing and data mining, finding its applications in various domains that deal with streaming data processing. Reliable real-time subsequence discovery from dynamic data sources is an essential challenge for finance, industry, health care, networks, and other fields [PAM12; Fu11; Rak+12; Suk+20]. It allows finding advantageous market state, preventing damages of equipment or infrastructure, timely recognizing development of severe health complications or avoiding traffic collisions [Dau+18; Rak+12]. In such applications, the data sources dynamically generate possibly high throughput data stream signals that often suffer from the presence of noise and outliers, potential variance in sampling rate, or varying nature of the underlying process leading to time distortions and change in amplitude scale [KK03; GFS18]. The reason for that might be potential clock drift

or jitter that happen in various sensor devices used for Internet-of-Things applications including smart homes, Industry 4.0, and others. Performing subsequence matching under such severe conditions is known to be a challenge [Rak+12].

Given a query (template) sequence and a data stream, the general objective of a pattern matching problem is to reveal and timely report subsequences in the given data stream that are similar to the query. In practical scenarios, it often happens that multiple examples that relate to the same query template are available for solving the pattern matching problem. This provides the possibility to account for template variations and as a result, be less susceptible to potential disturbances in the data stream. The availability of multiple query examples has driven the development of a number of time series averaging methods [Mar18; Mor+18; SCG15; PKG11; AM17]. Averaging multiple query templates has enabled traditional pattern matching approaches to be applied on the averaged template signal demonstrating better performance than when applied to each template instance separately [Mar18; PKG11; AM17].

Currently, deep neural networks have become one of the most powerful tools for solving pattern matching problems when a significant number of query examples [WYO16; CCC16] is available. LSTM-based architectures have recently shown superior performance across multiple domains and signal natures when formulating pattern matching as a time series classification problem [Zha+17; Ism+19]. However, such methods, due to their complex architectures and resulting in large parameter space, require a significant amount of query examples to provide a stable generalization performance and consequently a high pattern matching rate [LMT16]. In many practical applications, collecting these examples might be extremely costly or even unfeasible.

For the case of a single query template example, several approaches were proposed in the literature in the last decade, all accounting for possible time and amplitude distortions by utilizing Dynamic Time Warping (DTW) [Fu11; Ore+17]. In the past, DTW was successfully applied in various domains proving to be a powerful technique for sequence comparison and retrieval [KL83; BC94; RM03; Mül07]. Formally, DTW is a dissimilarity measure that allows finding the best alignment between two sequences reporting their degree of mismatch. Despite the fact that DTW is not a distance metric, it is nevertheless used by many algorithms in order to align and compare a pair of time-series signals, potentially of different lengths. Currently, pattern discovery has become one of the domains where DTW is extensively and successfully utilized [Rak+12; Ore+17; Fu11].

One of the earliest pattern discovery approaches utilizing DTW is the SPRING algorithm [SFY07] that is built on two main ideas: 1) star-padding which reduces the

time and space complexity to linear with respect to the data stream size, guaranteeing that the minimum distance is obtained; 2) subsequence time-warping matrix (STWM) which is an extension of a warping matrix natively utilized by DTW. Later, several modifications of SPRING were proposed in order to improve the matching performance [Pen+08; NWR10]. Later, Gong et al. [Gon+14; GFS18] introduced Normalization supported SPRING (NSPRING) which performs z-normalization of streaming data by considering mean and standard deviation of a predefined number of subsequent data samples. Giao et al. [GA16] proposed ISPRING that integrates min-max normalization into SPRING by considering the minimum and maximum values on a fixed-length monitoring window.

Another DTW-based pattern matching approach is UCR-DTW which was proposed in [Rak+12] to achieve subsequence discovery on large-scale time-series data by efficient pruning techniques. Similar to NSPRING and ISPRING, UCR-DTW employs a fixed-length sliding window in order to normalize the streaming data and as the result discover hidden patterns in the stream.

When multiple query examples are available an averaging procedure is applied to extend the above-mentioned pattern discovery methods. This is considered to obtain an average template query and utilize it further as the query for a pattern discovery algorithm [AM17]. In such cases, the possible presence of outliers might cause structural distortions of the resulting average template and might limit the performance of the subsequent pattern matching approach [AM17; SCG15].

One of the other limitations of existing pattern discovery methods is that the predominant practice for normalizing data streams is to employ normalization approaches such as min-max [Oga+10] or z-normalization (also known as normalization to zero mean and unit variance or simply standardization) [TSK14] with a fixed-length sliding window [Gon+14; GA16; Rak+12]. Fixed length sliding window-based normalization methods often cause unwanted structural disturbances, compromising the subsequent DTW by damaging its natural robustness to time distortion. The main reason for that is the signal length variation that is an intrinsic advantageous assumption of DTW [AC01; AE03; She+18] as well as the uniform scaling that is a global stretching or shrinking of time series in the time axis [Keo+04; Fu+08; RNR11; She+17].

To jointly address the above-mentioned limitations of the time series averaging and the fixed window length-based normalization methods in multiple template-based pattern discovery problem, in this chapter, we introduce a dynamic z-normalization mechanism that progressively scales incoming samples of the dynamically evolving data stream. By

utilizing dynamic z-normalization and adopting the STWM concept [SFY07], we propose a real-time DTW-based pattern matching approach. It is able to accurately report discovered subsequences that contain possible amplitude distortions by utilizing multiple query examples. Thanks to the additive property of the dynamic z-normalization, the proposed approach is able to discover hidden patterns under relatively large time distortions without the need for any predefined scaling parameters and is faster than a state-of-the-art lower bounding technique that is extended to supporting normalization.

In the following, we formulate the pattern matching problem, extending it to account for multiple template examples, and review the time-series averaging concept as well as the traditional fixed-length window z-normalization method. Then, we introduce a dynamic z-normalization mechanism, and based on that, we propose a DTW-based pattern matching approach that provides accurate and instant results for stream monitoring tasks. Finally, we evaluate the proposed pattern discovery method on artificially generated (synthetic) and real-world datasets (measurement data) and report on its operational performance.

4.2 Problem formulation

Consider a real-time data stream S to be a semi-infinite ordered set $\{s_0, s_1, s_2, \ldots, s_t\}$, where s_t is the most recent value. Let $S[t_b{:}t_e] = \{s_{t_b}, s_{t_b+1}, \ldots, s_{t_e}\}$ be a subsequence of S, starting with s_{t_b} and ending with s_{t_e}. The objective of a pattern matching problem with a single query is formulated as follows: given a query (template) sequence $Q = \{q_0, q_1, \ldots, q_{m-1}\}$ with a fixed length m, find the subsequences of S that are similar to Q. To formalize the formulation, similarity is usually replaced by dissimilarity between Q and $S[t_b{:}t_e]$ that becomes smaller in case of their match. We denote the dissimilarity between Q and $S[t_b{:}t_e]$ as $D(S[t_b{:}t_e], Q)$. Given a data stream S that possibly contains noise and various distortions (stretching, shrinking or shifting) along time or/and amplitude axis, we consider the three following problems.

Problem 1: real-time monitoring. Report the subsequence $S[t_b{:}t_e]$ by the time tick t_e+1 if $D(S[t_b{:}t_e], Q) \leq \epsilon$; where ϵ is a predefined constant. This problem arises in many streaming applications where the pattern is to be discovered in real time and is typically further extended to another two problems depending on the task at hand [SFY07].

Problem 2: disjoint query. Find all subsequences $S[t_b{:}t_e]_k$, $k \geq 1$ that satisfy:

(a) $D(S[t_b{:}t_e]_k, Q) \leq \epsilon$, where $k \geq 1$

(b) $D(S[t_b{:}t_e]_k, Q)$ is the minimum among all $D(S[t'_b{:}t'_e]_k, Q)$, where $S[t'_b{:}t'_e]_k$ is any subsequence that overlaps with $S[t_b{:}t_e]_k$.

Problem 3: top k query. Find $k \geq 1$ disjoint subsequences of S satisfying condition (b) of Problem 2 providing the smallest distances to the query sequence Q.

Since all the three problems are tightly coupled and are all considered as general pattern matching problems, we address them simultaneously. Additionally, when multiple examples of the same query template are available, the formulation of the above-mentioned problems remains the same, except the fact that there are L template examples available and the dissimilarity between the template and stream subsequence is defined as $D(S[t_b{:}t_e], \bar{Q})$, where $\bar{Q}$ is the average template sequence.

4.2.1 Time series alignment and averaging

The concept of an average of a set of objects has been successfully utilized in many domains of signal processing, machine learning and data mining, demonstrating its effectiveness on many problems, including data clustering and classification [AM17; SCG15]. Time series averaging allows to define a prototype that captures the central tendency of a set of time series, reducing the impact of unwanted effects such as noise and outliers. Despite its effectiveness, averaging of time series is considered to be a non trivial task due to the possibly varying length of sequences and requires their simultaneous alignment. To address this problem, time series averaging is formulated as an optimization task:

$$\bar{Q}^* = \underset{\bar{Q} \in E}{\operatorname{argmin}} \sum_{l=1}^{L} D_{\mathrm{DTW}}(\bar{Q}, Q_l) \tag{4.1}$$

where D_{DTW} is the DTW distance between an average sequence candidate $\bar{Q}$ and one of L sequences Q_l; E is a space of all possible time series induced by DTW and $\bar{Q}^*$ is the average sequence. Obviously, the space E might be considerably large limiting the solution of time series averaging to be performed in reasonable time. Formally, time series averaging under the DTW assumption is proven to be an NP-complete task, practically utilized by introducing several effective heuristics [PKG11; KMM19]. Figure 4.1 demonstrates an example of a time series signal averaging process based on the data of the Gun Point dataset [Dau+18]. Practically, none of the state-of-the-art pattern discovery methods incorporates a template averaging process and is only able to deal with the averaged sequence as the query template.

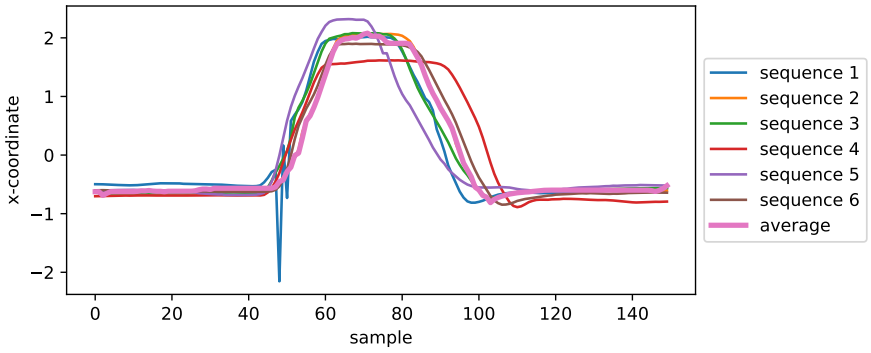


Figure 4.1: An example of time series averaging method using DTW Barycenter Averaging [PKG11] on Gun Point dataset (six sequence of class 2 are randomly taken to be averaged) [Dau+18]

4.2.2 Limitations of traditional normalization approaches

Data normalization is an essential step preceding pattern matching as it reveals structural differences between query and streaming signals by bringing both signals to similar amplitude ranges. The typically used z-normalization of a data sample s_k is defined on a sequence $S[t_b{:}t_e] = \{s_{t_b}, \ldots, s_{t_e}\}$ containing the value s_k as:

$$s'_{(t_b,t_e),k} = \frac{s_k - \mu_{(t_b,t_e)}}{\sigma_{(t_b,t_e)}}, \quad k = t_b, \ldots, t_e \tag{4.2}$$

where μ_{t_b,t_e} and σ_{t_b,t_e} are the estimates of mean and standard deviation of $S[t_b{:}t_e]$, respectively. Performing z-normalization ensures the transformation of the input signal to the one with nearly zero mean and unit variance [Zou+18]. However, defining the time range $S[t_b{:}t_e]$ in streaming scenarios is usually a challenging task, imposing limitations and harmful assumptions [Rak+12]. Failure to select proper scaling subsequence might often result in significant distortions of the structure of the signal being scaled. The predominant practice to perform a z-normalization is by using a fixed length sliding window. However, in cases where a pair of signals is compared via DTW, too large or too short normalization window might result in snapping of unwanted signal parts or signal fragmentation, correspondingly. That usually results into erroneous normalization and typically yields larger DTW distances between both signals, keeping time-distorted subsequences undiscovered. An example of such scenarios is demonstrated in Figure 4.2 (Cases 1 and 2): an improper normalization window length leads to an erroneously scaled signal. Case 1 shows a scenario when the pattern in the stream has much smaller duration than the query while Case 2 demonstrates an example when the pattern has much higher duration than the query. Finding an intrinsic pattern duration normalization window in pattern matching that employs DTW

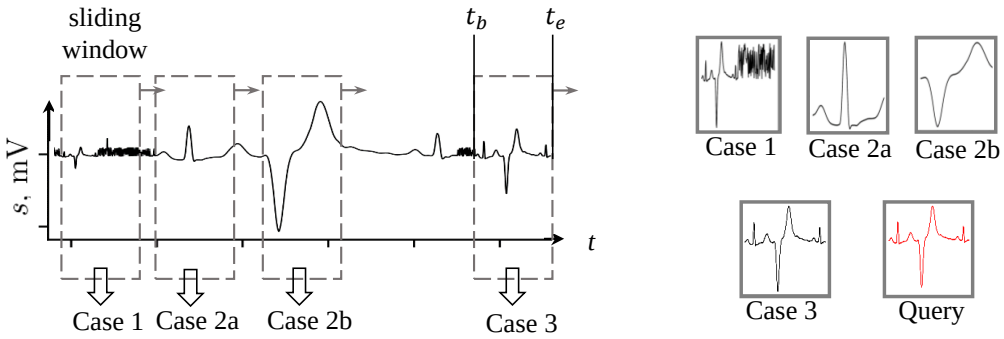


Figure 4.2: Cases 1 and 2 demonstrate z-normalization with improper window length; normalization window for case 3 equals to intrinsic length of the pattern leading to proper scaling

is practically ineffective as DTW assumes the possibility of time distortions between matching signals.

4.3 Proposed pattern matching method

In the sequel, we introduce a concept of dynamic z-normalization that in contrast to the fixed window length z-normalization does not require any assumptions or pre-knowledge about the duration of the pattern. We motivate this by proposing a prefix normalization and further prove the invariance properties of such a normalization. We then propose a pattern matching approach based on DTW for a single query example relying on that concept. We extend the framework to naturally support multiple examples of a query template in order to leverage the template diversity.

4.3.1 Dynamic Time Warping (DTW)

DTW [Fu11] is a dissimilarity measure defined between two arbitrary length time series $X = \{x_0, \ldots, x_{m-1}\}$ and $Y = \{y_0, \ldots .y_{n-1}\}$ as:

$$\begin{aligned} DTW(X,Y) =& D(m-1, n-1) \\ D(i,j) = ||x_i - y_j|| &+ \min \begin{cases} D(i, j-1) \\ D(i-1, j) \\ D(i-1, j-1) \end{cases} \\ D(-1,-1) = 0,\ & D(i,-1) = D(-1,i) = +\infty \\ i = 0, ..., m-1;\ & j = 0, ..., n-1 \end{aligned} \tag{4.3}$$

creating a time warping matrix (TWM) of $m \times n$ by a bottom-up dynamic programming process. The TWM is progressively constructed by appending a column that contains the distance between a point of the signal Y and all points of the signal X. Accordingly, the TWM contains elements $D(i, j)$ that keep accumulated distances between subsequences $X[0{:}i]$ and $Y[0{:}j]$ aligning them in a way that minimizes this distance. The alignment defines a mapping between the elements of X and Y and is described by a set of contiguous matrix indices. These indices form a continuous warping path that starts in $(0, 0)$ progressing to (i, j) indicating the alignment that yields minimum accumulated distance.

4.3.2 Dynamic z-normalization for DTW

Let the pattern $S[t_b{:}t_e]$ that is hidden in a data stream S has its intrinsic duration T_{int} that might be different from the duration of the query signal Q. Consider a real time data stream S containing the pattern of interest. This pattern has its starting time tick t_b and forms a substream $S[t_b{:}]$.

4.3.2.1 Prefix normalization

We define prefix normalization of a point s_k as:

$$s'_{t_b,k} = \frac{s_k - \mu_{t_b,k}}{\sigma_{t_b,k}}, \quad k = t_b, \ldots, t_e \tag{4.4}$$

where $\mu_{t_b,k}$ is

$$\mu_{t_b,k} = \frac{1}{k+1-t_b} \sum_{l=t_b}^{k} s_l \tag{4.5}$$

and $\sigma_{t_b,k}$ is

$$\sigma_{t_b,k} = \sqrt{\left(\frac{1}{k+1-t_b} \sum_{l=t_b}^{k} s_l^2 - \mu_{t_b,k}^2 \right)} \tag{4.6}$$

providing normalization parameters for a data point s_k considering its preceding signal. Obviously, such a normalization has a weakness: for small k there are only few values considered in order to obtain $\sigma_{t_b,k}$ and $\mu_{t_b,k}$ leading practically to $\sigma_{t_b,k} < \sigma_{t_b,t_e}$ and $\mu_{t_b,k} \neq \mu_{t_b,t_e}$. That brings an amplification and an amplitude shift of the prefix normalized signal, as compared to traditional z-normalization with scaling parameters obtained on a fixed-length time series. In order to mitigate that, we introduce the amplification factor η_k

$$\eta_k = \frac{\sigma_{t_b,t_e}}{\sigma_{t_b,k}}, \quad k = t_b, \ldots, t_e \tag{4.7}$$

and the shift factor δ_k

$$\delta_k = \frac{\mu_{t_b,k} - \mu_{t_b,t_e}}{\sigma_{t_b,t_e}}, \quad k = t_b, \ldots, t_e \tag{4.8}$$

to compensate for possible amplification and amplitude shift, respectively. Note should be taken that with the increase of k, η_k and δ_k converge to 1 and 0, respectively. When η_k and δ_k are known, it is possible to compensate for them:

$$s''_k = \frac{s'_{t_b,k}}{\eta_k} + \delta_k \tag{4.9}$$

obtaining a dynamically normalized value of s_k. Figure 4.3 demonstrates an example of the dynamic z-normalization on an ECG signal [Rak+12].

Further, we propose to incorporate the concept of dynamic normalization into DTW by using the amplification and the shift factors of the query sequence only. To allow that, we first demonstrate their invariance properties by defining them on continuous functions via replacing the summations in Equation (4.5) with integration.

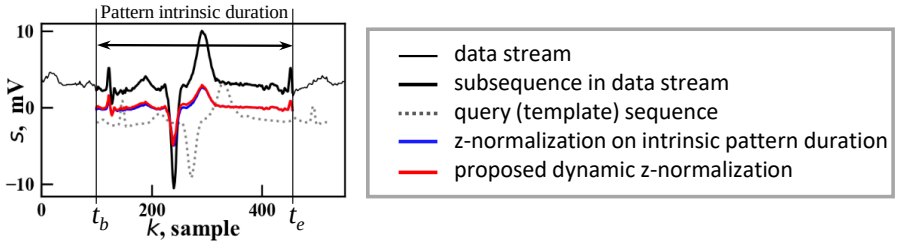


Figure 4.3: A comparison of proposed dynamic z-normalization and traditional z-normalization on an ECG signal sequence: original signal is z-normalized by applying a window of the length of pattern intrinsic duration and by the proposed dynamic z-normalization. Since the hidden subsequence matches the template, both normalization mechanisms provide very similar results. In practice, pattern length is unknown that limits performance of traditional z-normalization

4.3.2.2 Invariance properties

Consider a continuous function $f(x)$ defined on an interval $x \in [x_l, x_u]$ and a continuous function $g(x)$:

$$g(x) = C_2 f(C_1(x + C_0)) + C_3 \tag{4.10}$$

where C_0, C_1, C_2 and C_3 are constants. The domain of definition for $g(x)$ is then $[x'_l, x'_u] = [\frac{x_l}{C_1} - C_0, \frac{x_u}{C_1} - C_0]$. Since $g(x)$ is transformed from $f(x)$ by performing horizontal and vertical stretching and shifting, it is similar to $f(x)$ by its shape or structure. For the introduced earlier amplification and shift factors under the condition $\frac{x'-x'_l}{x'_u-x'_l} = \frac{x-x_l}{x_u-x_l}$, the following holds:

$$\eta'_{x'} = \eta_x, \quad \delta'_{x'} = \delta_x \tag{4.11}$$

where η_x and $\eta'_{x'}$ are the amplification factors for the prefix normalized $f(x)$ and $g(x)$ at x and x', respectively; δ_x and $\delta'_{x'}$ are the corresponding shift factors. The derivation is obtained by plugging Equation (4.10) into Equation (4.7) and is detailed in Section A.3. For the discrete case, with the subsequence $S[t_b{:}t_e]$ being scaled and shifted version of Q, the invariants in Equation (4.11) become:

$$\eta'_{k'} = \eta_k, \quad \delta'_{k'} = \delta_k \tag{4.12}$$

under the condition that $\frac{k'-t_b}{t_e-t_b} = \frac{k}{m-1}$; where η_k and δ_k are the amplification and shift factors for Q, respectively and η'_k and δ'_k are those of $S[t_b{:}t_e]$.

4.3.2.3 DTW embedding

Obviously, the amplification and shift factors can be easily obtained for the query (or multiple queries in case they are available). However, when searching for a subsequence

in a real-time streaming sequence, its exact starting and ending time ticks t_b and t_e are unknown until the subsequence is retrieved. At the same time, it is required to perform a proper normalization and as the result to find this subsequence. When the pattern in the stream and the query sequence are similar, exploiting Equation (4.12) allows to obtain a dynamic normalization of the pattern by using the amplification and the shift factors of the query sequence. For that, it is required to align the samples of the pattern and the query sequence establishing the mapping between k and k' in Equation (4.12). Consequently, this brings us to the point of establishing a normalization procedure directly in the DTW process. For that, we define the normalized DTW distance D_{norm} between the query Q and sub-subsequence $S[t_b{:}t]$ of $S[t_b{:}t_e]$ as:

$$\begin{aligned} D_{\text{norm}}(S[t_b{:}t], Q) =& D(t, m-1) \\ D(k', k) = d(k', k) + \min & \begin{cases} D(k', k-1) \\ D(k'-1, k) \\ D(k'-1, k-1) \end{cases} \\ D(-1,-1) = 0,\ D(k',-1) &= D(-1,k) = +\infty \\ k' = t_b, ..., t; \quad k &= 0, ..., m-1 \end{aligned} \tag{4.13}$$

where $d(k', k)$ is the distance between the dynamically normalized signal and is obtained by:

$$d(k', k) = ||(\frac{s'_{t_b,k'}}{\eta'_{k'}} + \delta'_{k'}) - (\frac{q'_{0,k}}{\eta_k} + \delta_k)|| \tag{4.14}$$

where $q'_{0,k}$ is the prefix normalized q_k on $Q[0{:}k]$ and $s'_{t_b,k'}$ is the prefix normalized $s_{k'}$ on $S[t_b{:}k']$; η_k and δ_k are the amplification and shift factors of $q'_{0,k}$ on the query Q; $\eta'_{k'}$ and $\delta'_{k'}$ are the amplification and shift factors of $s'_{t_b,k'}$ on the hidden pattern sequence $S[t_b{:}t_e]$.

With Equation (4.12), Equation (4.14) can be simplified to:

$$d(k', k) = ||\frac{s'_{t_b,k'} - q'_{0,k}}{\eta_k}||. \tag{4.15}$$

Analog to the original DTW, D_{norm} is obtained by constructing a TWM that satisfies the additive property of DTW: the consecutive value s_{t+1} arriving at time $t+1$ does not affect previously scaled values. The alignment procedure defines the normalization parameters of $s_{k'}$ according to its mapped query value q_k and results in a proper scaling when truly similar $s_{k'}$ and q_k are mapped. According to that, $D_{\text{norm}}(S[t_b{:}t], Q)$ reaches its minimum when t increases to t_e if $S[t_b{:}t_e]$ and Q are truly similar. As $S[t_b{:}t_e]$ gets more similar to Q, $D_{\text{norm}}(S[t_b{:}t], Q)$ becomes smaller since $d(k', k)$ gets smaller for each aligned pair of k' and k.

Figure 4.4 illustrates the process of the proposed dynamic z-normalization when obtaining $D_{\text{norm}}(S[t_b{:}t_e], Q)$.

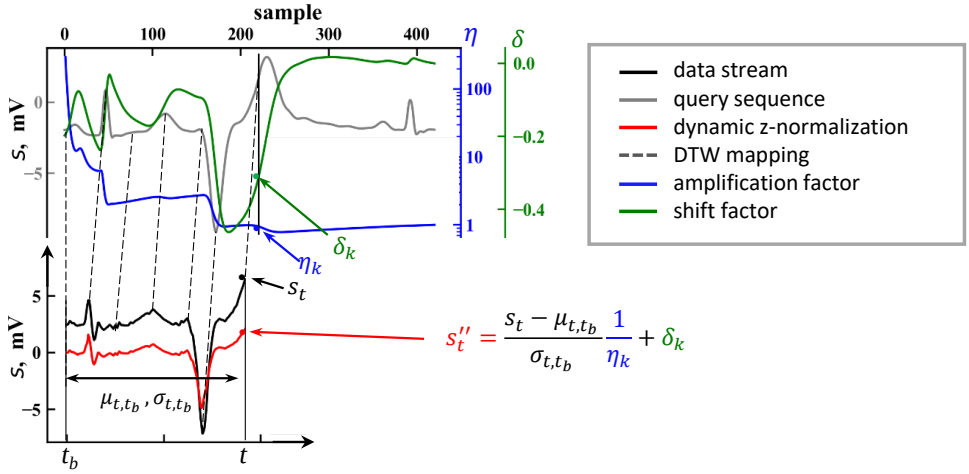


Figure 4.4: The proposed dynamic z-normalization. By integrating a normalization procedure into the DTW mapping, the value s_t is dynamically z-normalized according to its mapped query value. Index k used for η and σ refers to the length of the considered signal at time point t.

4.3.3 Dynamic Normalization based Real-time Pattern Matching (DNRTPM)

In this section, we provide the details on our proposed Dynamic Normalization based Real-time Pattern Matching (DNRTPM) [Suk+20] algorithm that is based on the introduced dynamic z-normalization principle and the STWM concept [SFY07]. After that we extend it to Multiple DNRTPM (MDNRTPM) that makes use of multiple query templates in order to solve the pattern matching problem.

4.3.3.1 DNRTPM algorithm

At the initial step the query sequence $Q = \{q_0, q_1, \ldots, q_{m-1}\}$ is prefix-normalized according to Equation (4.4) resulting into $Q' = \{q'_0, q'_1, \ldots, q'_{m-1}\}$. Additionally, the corresponding amplification factors $\eta = \{\eta_0, \eta_1, \ldots, \eta_{m-1}\}$ are calculated according to Equation (4.7).

For the given query and the incoming data stream S DNRTPM constructs an STWM in order to align the query with the streaming subsequence and return the DTW distance between them. Being updated for every incoming streaming value a cell of STWM stores two scalars: $B(t,k)$ and $D(t,k)$. $B(t,k)$ denotes the index of the starting point of the current candidate pattern and allows to obtain the warping path. $D(t,k)$ is

the accumulative distance of the warping path of the current candidate pattern and is obtained by:

$$\begin{aligned} D(t,k) =& \min \begin{cases} D'(t,k-1) \\ D'(t-1,k) \\ D'(t-1,k-1) \end{cases} \\ D'(i,j) =& || \frac{s'_{B(i,j),t} - q'_{0,k}}{\eta_k} || + D(i,j); \quad i = t-1,t;\ j = k-1,k. \\ D(t,-1) =& 0,\ D(-1,k) = +\infty; t = 0,\ldots,n;\ k = 0,\ldots,m-1. \end{aligned} \tag{4.16}$$

where $q'_{0,k}$ is the kth prefix-normalized value of the query sequence; $s'_{B(i,j),t}$ is the prefix-normalized s_t on the incoming streaming subsequence $S[B(i,j){:}t]$ by Equation (4.4). When a new value s_{t+1} arrives at the time tick $t+1$ a column is appended to the right side of the STWM for further alignment and the distance calculation process. $B(t,k)$ is obtained as follows:

$$\begin{aligned} B(t,k) =& \begin{cases} B(t-1,k), & \text{if } D(t,k) = D'(t-1,k) \\ B(t,k-1), & \text{if } D(t,k) = D'(t,k-1) \\ B(t-1,k-1), & \text{if } D(t,k) = D'(t-1,k-1) \end{cases} \\ B(t,0) =& t;\ t = 0,\ldots,n;\ k = 0,\ldots,m-1. \end{aligned} \tag{4.17}$$

The main operational principle of DNRTPM is illustrated in Figure 4.5 where a data stream S, its dynamically normalized substreams, a query Q and STWM are depicted. The main difference between Equation (4.13) and Equation (4.16) is that for the former the result of the min operator is decoupled from $d(k',k)$ since t_b (the index of the subsequence start) is fixed and does not change in time (Equation (4.14)). When having a dynamic stream, the starting index B differs for all three options at the right side of the min operator, requiring therefore to consider the difference before applying the min operator in Equation (4.16).

For the efficient calculation of $s'_{B(i,j),t}$ two lists that keep the prefix summation PS and the prefix summations of squares PSS are maintained as

$$\begin{aligned} \text{PS} =& \left\{ \text{ps}_{(B(t-1,\min)-1)}, \ldots, \text{ps}_t \right\} \\ \text{PSS} =& \left\{ \text{pss}_{(B(t-1,\min)-1)}, \ldots, \text{pss}_t \right\} \\ B(t-1,\min) =& \min_{0 \le i < m} B(t-1,i) \end{aligned} \tag{4.18}$$

where ps_i and pss_i are the prefix summation and the prefix sum of squares values obtained up to the time tick i and are defined as:

$$\begin{aligned} \text{ps}_i =& \text{ps}_{i-1} + s_i; \\ \text{pss}_i =& \text{pss}_{i-1} + s_i^2; \\ i =& 0,\ldots,t; \\ \text{ps}_{-1} =& \text{pss}_{-1} = 0. \end{aligned} \tag{4.19}$$

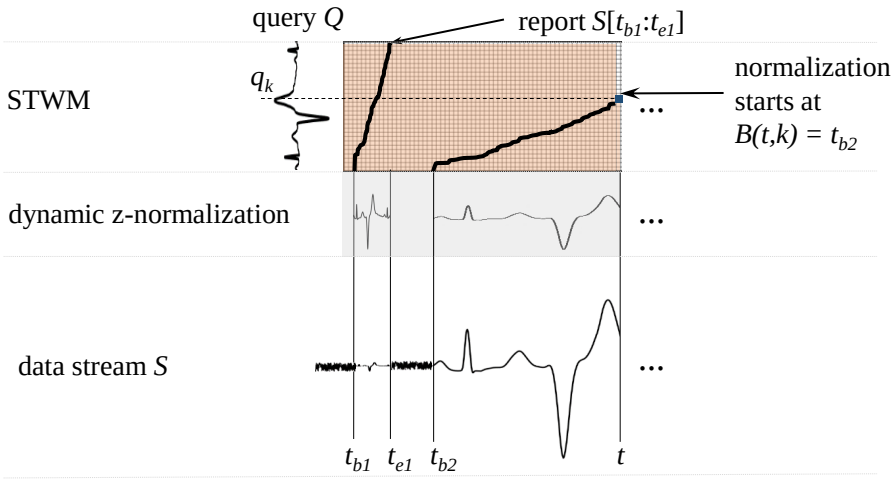


Figure 4.5: An example of subsequence matching procedure employed by DNRTPM: STWM colored in orange allows to perform alignment of the query and candidate subsequence by calculating distance between every element of query Q and dynamically normalized element of the data stream S. After that, the warping path (depicted as black curve on top of STWM) is established according to Equation (4.16).

Having PS and PSS, it is possible to efficiently obtain $s'_{B(i,j),t}$ by:

$$\begin{aligned} s'_{B(i,j),t} &= \frac{s_t - \mu_{B(i,j),t}}{\sigma_{B(i,j),t}} \\ \mu_{B(i,j),t} &= \frac{\mathrm{ps}_t - \mathrm{ps}_{(B(i,j)-1)}}{t - B(i,j) + 1} \\ \sigma_{B(i,j),t} &= \sqrt{\frac{\mathrm{pss}_t - \mathrm{pss}_{(B(i,j)-1)}}{t - B(i,j) + 1} - \mu^2_{B(i,j),t}}. \end{aligned} \tag{4.20}$$

Both PS and PSS are implemented by circular buffer dequeue, achieving constant $O(1)$ time of appending and removing elements at their both ends. The space required to store both PS and PSS is comparable to space that the query signal requires. An update of PS and PSS for the value s_{t+1} is performed by removing the first element from the left sides of both dequeues and appending ps_{t+1} and pss_{t+1} to their right sides, respectively.

For real time monitoring problems that require rapid retrieval, the subsequence $S[B(t, m-1){:}t]$ is reported at time tick t if $D(t, m-1)$ is smaller than a predefined threshold ϵ. For the case of disjoint query tasks (that report non-overlapping subsequences), the subsequence $S[B(t, m-1){:}t]$ is only reported when it is confirmed that all the upcoming subsequences that overlap with it provide larger distances. This is achieved by keeping the current minimum distance $D_{\min}$ and the corresponding sub-

sequence S_{opt} (starts at t_s and ends at t_e) reporting $S[B(t, m-1){:}t]$ only when:

$$\forall_k, D(t,k) \geq D_{\min} \vee B(t,k) > t_e. \tag{4.21}$$

$D_{\min}$ and S_{opt} are updated as the current distance $D(t, m-1)$ and subsequence $S[B(t, m-1){:}t]$ when $D(t, m-1) < D_{\min}$.

The pseudo-code of the DNRTPM algorithm for disjoint query problems is summarized in Algorithm 1 (Section A.4). The adoption for real-time monitoring is done by reporting ($D(t, m-1)$, $B(t, m-1)$, t) as soon as the condition $D(t, m-1) < \epsilon$ of Algorithm 1 is satisfied. Additionally, the adoption for the top k query is done by keeping the best k discovered subsequences reported from the output D_{min}, t_s, t_e of Algorithm 1 and maintaining ϵ as the smallest distance for the k subsequences. When comparing MDNRTPM and DNRTPM in terms of their computational and space complexities a note should be taken that MDNRTPM brings additional increase only by a constant factor that is the number of query templates L.

4.3.3.2 Multiple DNRTPM (MDNRTPM)

In the case of multiple query examples, we extend our proposed DNRTPM approach to MDNRTPM. MDNRTPM by performs dynamic z-normalization for all examples simultaneously considering an average distance between the stream subsequence and the templates.

At first, in MDNRTPM, all the template examples $Q_l, (l = 1 \dots L)$ are aligned so that the correspondence between every sample of all template examples is established. After that, as in DNRTPM, every query sequence $Q_l = \left\{q_0^l, q_1^l, \dots, q_{m_l-1}^l\right\}$ is prefix-normalized according to Equation (4.4), resulting into $Q^{l'} = \left\{q_0^{l'}, q_1^{l'}, \dots, q_{m_l-1}^{l'}\right\}$. The amplification factors $\eta^l = \{\eta_0^l, \eta_1,^l \dots, \eta_{m_l-1}^l\}$ and shift factors $\delta^l = \{\delta_0^l, \delta_1,^l \dots, \delta_{m_l-1}^l\}$ are calculated according to Equation (4.7). Since the alignment is established, as in the DNRTPM, the STWM is then simultaneously constructed for every template example Q_l. The main difference of MDNRTPM with respect to DNRTPM is the way the distance between the samples of streaming subsequence and samples of every template is calculated. In MDNRTPM, to calculate the distance, at first, the dynamically normalized values of every template according to Equation (4.9) are obtained and for each of them the distance is calculated according to Equation (4.14). Then, all the resulting distances are averaged over all available templates. Thus, the pseudo code in Section A.4 is changed to account for multiple template examples as well as to calculate the averaged distance.

4.3.3.3 Space and time complexity

According to Equation (4.16) and Equation (4.17), as well as to Algorithm 1, only the values D and B corresponding to the columns of STWM of the current and previous time tick are needed to be kept in memory. Additionally, two dequeues PS and PSS whose length is comparable to m are required leading to $O(m)$ space complexity (in the worst case $m = t$). As result, the space complexity of the proposed DNRTPM and MDNRTPM method is $O(m)$. The time complexity to fill each cell of the warping matrix is $O(1)$, leading to $O(m)$ per time tick or $O(mt)$ for the whole process.

4.4 Experiments

In this section, we evaluate our proposed DNRTPM and MDNRTPM algorithms and compare their performance with other state-of-the-art pattern matching methods, namely, UCR-DTW [Rak+12], ISPRING [GA16] and NSPRING [Gon+14] based on several artificially generated and real-world (measurement data) datasets.

The numerical simulations and experiments are primarily conducted by solving a top k query task since the requirement of setting a predefined threshold for real-time monitoring and disjoint query tasks strongly depends on the dataset and the domain it is coming from. Practically, the threshold is to be tuned in order to make real-time monitoring or disjoint query tasks reporting only k non-overlapping subsequences, providing a similar result as top k query provides. For time series alignment and averaging for all approaches, we apply DBA [PKG11]. All numerical simulations and experiments are performed using a PC with Intel(R) Xeon(R) W-2133 CPU 3.60GHz $\times$ 12 and 8GB 2666MHz $\times$ 8 RAM.

4.4.1 Numerical simulation: Dynamic z-normalization

In this numerical simulation, we compare the performance of the proposed dynamic z-normalization and the traditional z-normalization with fixed length window. For that, we generate three synthetic (artificial) time series signals: cat, spoon and stairs, each of 200 samples (see Figure 4.7) used as query sequences. The shapes of the three synthetic signals were chosen to cover close-to-reality cases where signals have abrupt transitions (cat), smooth transitions (spoon) and step-wise decreasing pattern (stairs). The data stream is created by concatenating white noise sequences of 180 samples with

the three query signals after adding to them distortion by scaling in both amplitude and time directions by 200% and 75%, respectively, as well as shifting in amplitude direction by 5. To reveal the effect of window size on the result of z-normalization,

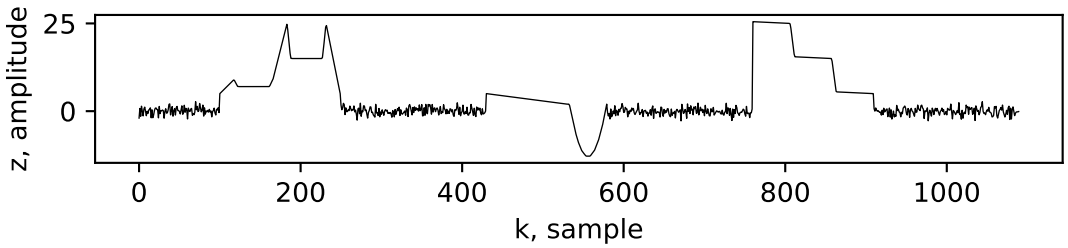


Figure 4.6: A data stream generated by concatenation of white noise with distorted queries

we run a sliding window z-normalization on the resulting data stream with different window sizes: the intrinsic pattern duration window ($T = T_{\text{int}}$), a 50% greater window ($T > T_{\text{int}}$) and a 50% smaller window ($T < T_{\text{int}}$). We run DNRTPM with best query (top 1 query) on the resulting data stream obtaining the normalized values according to Equation (4.9).

Table 4.1 summarizes DTW distances between the normalized pattern subsequences in data stream and their corresponding query signals. The results are obtained using 100 Monte Carlo iterations. On Figure 4.6 the normalization result on one of the Monte

Normalization type	signal 'cat'	signal 'spoon'	signal 'stairs'
z-normalization $T < T_{\text{int}}$	77.25	36.57	36.96
z-normalization $T > T_{\text{int}}$	84.91	24.22	30.26
z-normalization $T = T_{\text{int}}$	3.96	**1.84**	**0.67**
dynamic normalization	**3.70**	2.28	1.09

Table 4.1: DTW distances between the z-normalized query signals and their corresponding distorted subsequences in data streams for the proposed dynamic and traditional z-normalization

Carlo iteration is depicted. According to Table 4.1 note should be taken that the proposed dynamic normalization provides comparable results to z-normalization with the intrinsic pattern duration window proving their practical equivalence. However, in practice, the intrinsic pattern duration is not known that would limit the performance of traditional z-normalization approach. In the current scenario, when the patter duration is known, the DTW distances in Table 4.1 for these two normalization approaches have no significant differences.

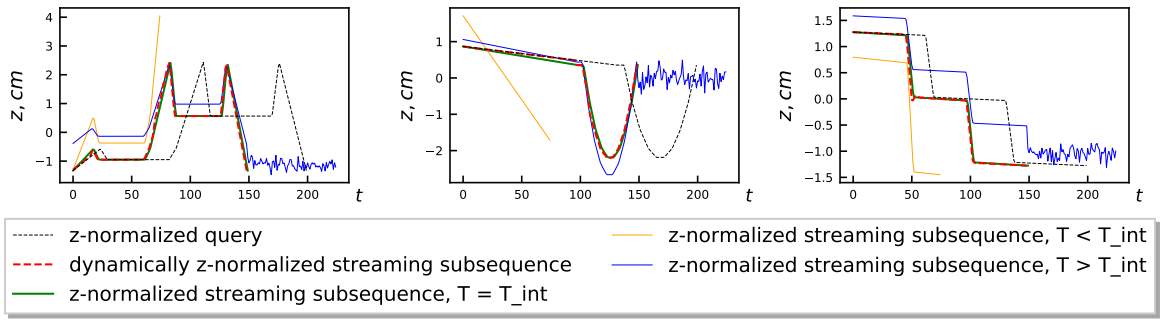


Figure 4.7: Dynamic and sliding window z-normalization with different window lengths on one of the Monte Carlo iterations. An improper window makes z-normalization unable to bring the query and the pattern hidden in the stream to the same scale. Dynamic z-normalization and z-normalization with the intrinsic pattern duration window are nearly identical and both are at the same scale in z direction as the normalized query sequence

4.4.2 Numerical simulation: Pattern matching on synthetic data

In this numerical simulation, we utilize synthetic query signals generated in the previous numerical simulation as well as their reflected over time axis versions to demonstrate DNRTPM's and MDNRTPM's robustness to uniform scaling. The six resulting synthetic query signals establish six different query template classes. To create multiple examples per template class, we resample each original query four times by a random factor drawn from a uniform distribution U[0,2) resulting into five examples per template class.

To generate the stream, for every original query signal, we apply linear interpolation and re-sampling in the time dimension to obtain the length of $\frac{m}{\lambda}$, where m is the length of the original signal and λ is a varying uniform scaling factor. Then, by amplitude scaling and shifting (scaling and shifting factors are randomly drawn from uniform distributions U[0,10) and U[-5,5), respectively), we generate 30 signals from every previously uniformly-scaled template resulting into 180 patterns in total. The data stream is then created by concatenating these 180 patterns with $\frac{2m}{\lambda}$ samples of white noise.

We perform a top 30 disjoint query task on the resulting data stream for each of the six classes. When querying a particular class on the data stream, a corresponding pattern hidden in the stream is considered to be retrieved (or recalled) when the overlapping percentage between the query and any of the 30 reported subsequences is greater than

a predefined percentage (α) that is defined between two subsequences $S[i{:}j]$ and $S[i'{:}j']$ according to [NWR10] as:

$$\alpha = \begin{cases} \frac{\min(j,j')-\max(i,i')+1}{\max(j,j')-\min(i,i')+1}, & \text{if } \min(j,j') \geq \max(i,i') \\ 0, & \text{if } \min(j,j') < \max(i,i') \end{cases} \tag{4.22}$$

The recall rate is defined as the percentage of patterns hidden in the data stream corresponding to the retrieved query. We set $\alpha = 50\%$ and perform the experiment for each uniform scaling factor ranging from 0.25 to 10 over 100 Monte Carlo iterations. The average recall rate over all iterations for the methods considered for different uniform scaling factors is shown in Figure 4.8.

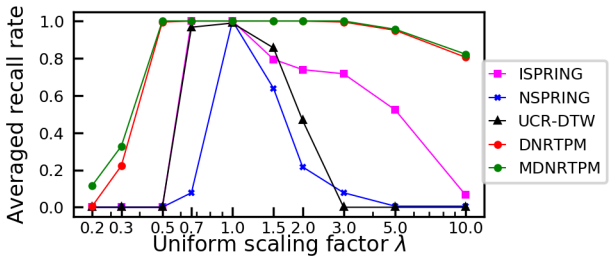


Figure 4.8: Recall rate for several pattern matching methods for varying uniform scaling factors

4.4.3 Pattern matching on UCR archive

In this experiment, we compare all early considered methods by performing the top k query task on the UCR archive dataset [Dau+18], which contains 128 real-world datasets collected from various fields. UCR archive is recognized to be a benchmark in the community for assessing the performance of pattern matching algorithms due to its accurate structure, dataset diversity and convenient data arrangement: each dataset contains labeled pre-segmented z-normalized sequences of time series belonging to a particular class. Moreover, all sequences within the same dataset are of equal length for nearly all datasets in the archive. The data recording process [HCK13; PAM12; CMU; RK04; Yan+09] as well as manual segmentation procedures [HCK13; PAM12; CMU; RK04; Yan+09] were reported to be extremely tedious and are practically unfeasible in real-time pattern matching tasks or can only be done with significant effort. For this reason, in order to simulate realistic streaming scenarios, we adjust each UCR dataset by performing the following operations: we randomly scale and shift the signal by factors drawn from U[0,10) and U[-5,5), respectively. We further uniformly scale in time axis by λ (or $\frac{1}{\lambda}$ by a chance of 50%) drawn from U(1,2). Finally, all training

sequences are randomly shuffled and concatenated into a single data stream, while the testing sequences of the same class are split into triples and used as query templates (we obtain three template examples per class). For every testing triple in every dataset, we perform a top k disjoint query, where k is the number of sequences in the training set that belong to the same class as the ones in the triple.

In this experiment, for all the considered methods, we evaluate the recall rate (AoR), the overlapping percentage (AoD), the DTW distance between the z-normalized query and the z-normalized retrieved pattern, the query time and the reporting delay. Recall (AoR) is defined as the percentage of subsequences in the data stream with the same class label as in the triple that are found by top k query. Query time is the wall clock time of a single query. Reporting delay measures the difference between the time when the whole hidden pattern gets available to query and its reporting time. To measure the reporting delay, firstly, a top k query is performed to get the kth smallest distance and then this distance is set as the threshold for a stream monitoring task. In this experiment, the disjoint query according to Equation (4.21) is not applied since all the discovered subsequences are reported as soon as they have a distance smaller than the threshold. The reporting delay is the difference between the ending time tick of the pattern and the reporting time tick of the first subsequence that overlaps with the pattern (both time ticks are in the data stream clock) divided by the length of the pattern. Table 4.2 summarizes the results averaged over five Monte Carlo iterations.

		ISPRING	NSPRING	UCR-DTW	DNRTPM	MDNRTPM
Recall	averaged value	0.30	0.37	0.33	0.42	**0.43**
	best on x% datasets	8%	15%	21%	23%	**33%**
AoD	averaged value	0.78	0.80	0.78	0.82	**0.85**
	best on x% datasets	8%	16%	20%	22%	**34%**
DTW distance	averaged value	22.13	23.67	27.45	19.55	**18.14**
	best on x% datasets	18%	17%	9%	24%	**32%**
Query time	averaged value (seconds)	5.52	**0.44**	45.34	0.97	1.26

Table 4.2: Evaluation performance summary on UCR datasets

4.4.4 Pattern matching on mouse trajectories data

In this experiment, we evaluate the performance of all the methods on the real-world mouse dynamics dataset. The dataset was obtained by the author by collecting PC

mouse device movements by seven users. The users followed four types of trajectories with the mouse device: 8, &, % and $\star$ performing eight trials for every trajectory type (resulting into 32 recording per user). The users performed random mouse movements in between of every trial. The mouse location coordinates (x and y) were recorded with a sampling frequency of 20Hz. The data stream was generated by concatenating all the trajectories resulting into 224 hidden patterns that were labeled for testing purposes but kept unsegmented. To assess the performance of all earlier considered methods every labeled trajectory as well as 3 other randomly picked trajectories of the same class were used as query templates for a top $k(k = 56)$ query task resulting in 224 top k tasks. The evaluated recall, AoD and DTW distance metrics for all the methods are provided in Table 4.3.

		ISPRING	NSPRING	UCR-DTW	DNRTPM	MDNRTPM
X axis	Recall	0.561	0.554	0.615	0.712	**0.721**
	AoD	0.816	0.839	0.802	**0.848**	0.843
	DTW distance	12.40	13.37	14.11	11.32	**11.12**
Y axis	Recall	0.526	0.497	0.537	0.638	**0.651**
	AoD	0.802	0.810	0.802	0.808	**0.812**
	DTW distance	10.24	10.61	11.90	9.78	**9.43**

Table 4.3: The recall, AoD and DTW distance measures averaged over 224 top $k(k = 56)$ queries on the continuous mouse dynamics data stream

4.4.5 Discussions

The results in Table 4.1 and Figure 4.7 show that an improper normalization window significantly affects the normalization process making possible to miss parts of the hidden pattern or include an unwanted streaming signal. As it is seen from the increased DTW distance, this would further lead to performance degradation of a consequent pattern matching approach. The reason for that is that in case of z-normalization, the non-intrinsic pattern duration window does not allow to bring the signal to the right scale. As a result, sliding window based normalization methods (e.g. z-normalization) require a proper window length to be set. As demonstrated in Figure 4.7, the proposed dynamic z-normalization is nearly identical to z-normalization applied on a window that equals to the pattern length ($T = T_{\text{int}}$).

The evaluated recall rate summarized in Figure 4.8 demonstrates that window-free nature of dynamic z-normalization makes the proposed DNRTPM and MDNRTPM robust to distortions in both time and amplitude axis. According to Figure 4.8, by

severe uniform scaling (λ ranges from 0.5 to 3) DNRTPM (and MDNRTPM) consistently provide a recall rate of 1. The reason for the rapid decrease of the recall rate for the other state-of-the-art methods with the presence of uniform scaling ($\lambda \neq 1$) lies in their normalization mechanism: an improper window length is unable to bring hidden subsequences to the right scale. Moderate domination of ISPRING with respect to NSPRING and UCR-DTW is due to min-max normalization that ISPRING is employing while the stream does not contain any extreme values. For $\lambda < 0.5$ the performance of all considered methods decreases. The reason for that is that all the methods favor shorter sequences, thus performing wrong retrievals. For DNRTPM (and MDNRTPM) according to Equation (4.16) the distance between the query Q of length m and any subsequence in the stream $S[i{:}j]$ is the summation of n_d distances each obtained by Equation (4.14), where $n_d >= \max(j - i + 1, m)$. When λ is much smaller than 1, for a pattern subsequence $S[i'{:}j']$, difference betwenn j' and i' is much greater than m and $n_d >= j' - i'$. Thus, n_d can be smaller and there is a smaller number of distances obtained by Equation (4.14) in the summation, so that the distance is smaller. However, the fixed window length normalization amplifies this effect and decreases the performance of ISPRING, NSPRING and UCR-DTW more rapidly than that of DNRTPM and MDNRTPM.

The results of the experiment on real data (UCR and PC mouse device datasets summarized in Table 4.2 and Table 4.3, respectively) demonstrate that the proposed DNRTPM and MDNRTPM are able to achieve a higher subsequence matching performance than the other methods. Particularly, both proposed methods achieve significantly higher recall as well as provide quality retrieval of subsequences (see AoD and DTW distance metrics). From the execution performance point of view, we observe that the running time of UCR-DTW and ISPRING is significantly higher than the one of DNRTPM and MDNRTPM that might be a limiting factor in many practical applications.

Finally, all numerical simulations and experiments demonstrate the effectiveness of MDNRTPM over the other methods, including DNRTPM. This is achieved since MDNRTPM considers all template examples simultaneously and not only an averaged template. This provides more flexibility while calculating the DTW distance and is less susceptible to outliers in template examples.

4.5 Summary

In this chapter, we addressed the problem of real-time pattern matching from dynamic and possibly distorted data streams. We introduced a dynamic z-normalization scheme

showing its equivalence to the traditional z-normalization with fixed optimal window size. Utilizing the proposed normalization scheme, we further introduced a real-time pattern matching method and its extension to support multiple template examples. We demonstrated that the proposed pattern matching approaches provide high operational performance on both artificially generated and real-world use cases outperforming the other state-of-the-art pattern matching methods, especially, when severe time distortions and sampling rate variance present.

Chapter 5

Conclusions and Future Work

In this thesis, three fundamental machine learning and signal processing challenges, namely, clustering, classification, and pattern matching were addressed from the wisdom of the crowds concept point of view. By leveraging this concept, the dissertation brings in an ensemble learning paradigm principle in order to solve these fundamental challenges. Specifically, scalable methods to address clustering, classification, and pattern matching problems were developed and assessed by conducting experiments in simulated and real-world environments. The results of this study revealed the applicability of the proposed methods to various industrial domains.

5.1 Conclusions

5.1.1 Consensus Clustering

According to the wisdom of the crowd concept, two consensus clustering frameworks were proposed to perform accurate and stable object clustering in a high-dimensional feature space. The first framework employs the data fragments concept allowing for a scalable consensus clustering solution. Additionally, by utilizing a distance measure based on data fragments and addressing the drawbacks of the Hamming distance in co-occurrence-based consensus clustering methods an expressive distance measure is proposed. After that, a consensus function is built around this measure. An extensive evaluation of the proposed framework is conducted indicating its theoretical performance in different situations as well as effectiveness for solving real-world problems.

The second framework is based on a novel consensus clustering approach to efficiently and rapidly combine multiple clustering solutions and obtaining interpretable results. Based on the idea of binary matrix decomposition, the proposed consensus clustering method is solved through the recursive rank one binary matrix approximation heuristic. In addition, an effective initialization strategy for the heuristic algorithm is introduced. The proposed framework is evaluated against several synthetic and real data sets to provide a comprehensive view on its performance. The results show its superiority against other state-of-the-art methods and the ability to scale for large data sets.

5.1.2 Dynamic Classifier Selection for Data Fusion

A data fusion problem from heterogeneous sources is considered and addressed from a classification point of view. For that, a dynamic classifier selection framework to select and combine competent classifiers is proposed. A multiple classifier system is employed in order to solve a data fusion problem proposing a mechanism to establish a competence estimation and selection procedure. The main focus of the framework is to address data fusion problems from heterogeneous sources that exhibit high-class imbalance. The proposed framework is validated on two real data fusion remote sensing problems as well as a series of synthetic experiments. The results prove that the proposed framework is able to provide accurate classification results even with high-class imbalance and can be used to combine data sources of different nature, spatial resolution, and scarcity levels providing powerful framework to solve local climate zones classification as well as land use and land cover classification problems.

5.1.3 Dynamic Pattern Matching with Multiple Queries on Data Streams

An extended for multiple template examples pattern matching problem is formulated to address data stream-based similarity search problem. A dynamic z-normalization mechanism that progressively scales incoming samples of the dynamically evolving data stream is proposed to address the limitations of fix-length z-normalization. By utilizing dynamic z-normalization and multiple template-based pattern matching problem formulation a real-time DTW-based pattern matching approach is proposed enabling to accurately report discovered subsequences that contain possible amplitude distortions. The proposed pattern discovery method is evaluated on synthetic and real-world datasets demonstrating its high operational performance. Particularly, the results of the experiments on UCR data archive and collected PC mouse device datasets indicate superior subsequence matching performance to the other state-of-the-art methods. Additionally, both proposed methods achieve significantly higher recall as well as provide quality retrieval of subsequences.

5.2 Challenges and Future Work

In addition to the above-mentioned contributions, the dissertation brings up several challenges that can be studied in future research. Some of these challenges are outlined as follows.

5.2.1 Consensus Clustering

An important aspect for considering in future work is related to the consensus function used in Section 2.6. While agglomerative clustering [SJ13] is a natural way to be used as the consensus function, it has two main drawbacks: the necessity to provide a target number of clusters and the type of linkage. An important improvement in this area would be a design of a parameter-free consensus function that deduces the target number of clusters from those of partition members. Another important and natural extension for approaches proposed in Section 2.6 and Section 2.7 is the ability for consensus functions to report degree of conensus for every object. Such a measure would allow to assess overall ensemble impurity level as well as to consider every object from this point of view.

5.2.2 Dynamic Classifier Selection for Data Fusion

One of the most important aspects of dynamic classifier selection methods is the establishment of the region of competence for ensemble members. In many dynamic selections methods including the one proposed in Chapter 3 a K-Nearest Neighbors [CSC18b] approach is utilized. Despite the fact, that K-Nearest Neighbors is a well-established approach for such applications, its performance depends a lot on the data type and requires careful parameter selection.

5.2.3 Dynamic Pattern Matching with Multiple Queries on Data Streams

The contribution for pattern matching with multiple queries presented in this thesis deals with the challenge of time-series-based similarity search in two steps: query templates averaging and real-time pattern matching itself. To jointly address these tasks, an interesting future study could go into the direction of a few short learning principle [TLL20; Nar+19]. A key challenge for that would be to design a framework to be able to dynamically monitor real-time data stream and utilize a classifier trained on a few examples that would perform pattern matching task. There were already several attempts to address similar tasks in this way [FD19; IK20], however, the complete framework for dynamic pattern matching with multiple queries on data streams is still missing.

Appendix

A.3 Proof

Proof of Equation (4.11): for a continuous function $f(x)$ defined on a continuous interval $x \in [x_l, x_u]$, the prefix normalization is:

$$f(x)' = \frac{f(x) - \mu_x}{\sigma_x} \tag{A.1}$$

where μ_x is

$$\mu_x = \frac{1}{x - x_l} \int_{x_l}^{x} f(t)dt \tag{A.2}$$

and σ_x is

$$\sigma_x = \sqrt{\frac{1}{x - x_l} \int_{x_l}^{x} f(t)^2 dt - \mu_x^2} \tag{A.3}$$

The amplification and shift factors are:

$$\eta_x = \frac{\sigma_x}{\sigma_{x_u}}, \quad \delta_x = \frac{\mu_x - \mu_{x_u}}{\sigma_{x_u}} \tag{A.4}$$

For function $g(x) = C_2 f(C_1(x + C_0)) + C_3$, its prefix mean is:

$$\begin{aligned}
\mu'_{x'} =& \frac{1}{x' - x'_l} \int_{x'_l}^{x'} g(t)dt \\
=& \frac{1}{x' - \frac{x_l}{C_1} + C_0} \int_{\frac{x_l}{C_1} - C_0}^{x'} C_2 f(C_1(t + C_0)) + C_3 dt \\
=& \frac{C_2}{x' - \frac{x_l}{C_1} + C_0} \int_{\frac{x_l}{C_1} - C_0}^{x'} f(C_1(t + C_0))dt + C_3 \\
& (\text{let } v = C_1(t + C_0)) \\
=& \frac{C_2}{C_1 x' - x_l + C_0 C_1} \int_{x_l}^{(x' + C_0)C_1} f(v)dv + C_3
\end{aligned} \tag{A.5}$$

The prefix standard deviation is:

$$\begin{aligned}
\sigma'_{x'} =& \sqrt{\frac{1}{x'-x'_l}\int_{x'_l}^{x'} g(t)^2 dt - \mu^2_{x'}} \\
=& \sqrt{\frac{1}{x'-x'_l}\int_{x'_l}^{x'} (C_2 f(C_1(t+C_0))+C_3)^2 dt - \mu^2_{x'}} \\
& (\text{let } v = C_1(t+C_0)) \\
=& \sqrt{\frac{1}{C_1 x' - x_l + C_0 C_1}\int_{x_l}^{(x'+C_0)C_1} (C_2 f(v)+C_3)^2 dv - \mu^2_{x'}} \\
=& \sqrt{\frac{\int_{x_l}^{(x'+C_0)C_1}(C_2^2 f(v)^2 + 2C_2C_3 f(v) + C_3^2)dv}{C_1 x' - x_l + C_0 C_1} - \mu^2_{x'}} \\
=& \sqrt{\frac{\int_{x_l}^{(x'+C_0)C_1} C_2^2 f(v)^2 dv - \frac{C_2^2}{C_1 x' - x_l + C_0 C_1}(\int_{x_l}^{(x'+C_0)C_1} f(v)dv)^2}{C_1 x' - x_l + C_0 C_1}}
\end{aligned} \tag{A.6}$$

Under the condition that $\frac{x-x_l}{x_u-x_l} = \frac{x'-x'_l}{x'_u-x'_l}$:

$$x' = \frac{x}{C_1} - C_0 \tag{A.7}$$

Substitute Equation (A.7) into Equation (A.5) and Equation (A.6):

$$\begin{aligned}
\mu'_{x'} =& \frac{C_2}{C_1 x' - x_l + C_0 C_1}\int_{x_l}^{(x'+C_0)C_1} f(v)dv + C_3 \\
=& \frac{C_2}{x - x_l}\int_{x_l}^{x} f(v)dv + C_3 \\
=& C_2\mu_x + C_3 \\
\sigma'_{x'} =& \sqrt{\frac{\int_{x_l}^{(x'+C_0)C_1} C_2^2 f(v)^2 dv - \frac{C_2^2}{C_1 x' - x_l + C_0 C_1}(\int_{x_l}^{(x'+C_0)C_1} f(v)dv)^2}{C_1 x' - x_l + C_0 C_1}} \\
=& C_2\sqrt{\frac{\int_{x_l}^{x} f(v)^2 dv - \frac{1}{x-x_l}(\int_{x_l}^{x} f(v)dv)^2}{x - x_l}} \\
=& C_2\sigma_x
\end{aligned} \tag{A.8}$$

Therefore:

$$\begin{aligned}
\eta'_{x'} =& \frac{\sigma'_{x'}}{\sigma'_{x'_u}} = \frac{C_2\sigma_x}{C_2\sigma_{x_u}} = \eta_x \\
\delta'_{x'} =& \frac{\mu'_{x'} - \mu'_{x'_u}}{\sigma'_{x'}} = \frac{C_2\mu_x + C_3 - C_2\mu_{x_u} - C_3}{C_2\sigma_x} = \quad \delta_x
\end{aligned} \tag{A.9}$$

A.4 DNRTPM pseudocode

Algorithm 1: DNRTPM

Input: A new value s_t at time-tick t
Output: Matched subsequence $S_{i,t}$ if any
Initialization: Initialize PS and PSS as empty deque; obtain prefix normalized query Q and its amplification factor by eq. (4.4) and eq. (4.7)
end initialization
Compute ps_t, pss_t by Equation (4.19) and append to the end of PS, PSS;
for $k \leftarrow 0$ **to** $m-1$ **do**
  Compute $D_{t,k}$ and $B_{t,k}$ by Equation (4.16) and Equation (4.17);
end
Remove $\{\text{ps}_{(B(t-1,\min)-1)}, \ldots, \text{ps}_{(B(t,\min)-2)}\}$ and $\{\text{pss}_{(B(t-1,\min)-1)}, \ldots, \text{pss}_{(B(t,\min)-2)}\}$ from the beginning of PS, PSS;
if $D_{min} < \epsilon$ **then**
  if $\forall_k,\ D(t,k) \geq D_{min} \vee B(t,k) > t_e$ **then**
    output $D_{\min}$, t_s, t_e;
    $D_{\min} = +\infty$;
    for $k \leftarrow 0$ **to** $m-1$ **do**
      if $B(t,k) \leq t_e$ **then**
        $D(t,k) = +\infty$;
      end
    end
  end
end
if $D(t,m-1) < \epsilon$ **then**
  if $D(t,m-1) < D_{min}$ **then**
    $D_{\min} = D(t,m-1)$;
    $t_s = B(t,m-1)$;
    $t_e = t$;
  end
end
for $k \leftarrow 0$ **to** $m-1$ **do**
  $D(t-1,k) = D(t,k)$;
  $B(t-1,k) = B(t,k)$;
end

List of Acronyms

AoD	Accuracy on Retrieval
AoR	Accuracy on Detection
ARI	Adjusted Rand Index
AE	Average Entropy
AMI	Adjusted Mutual Information
AVI	Advanced Vegetation Index
ASRS	Approximate SimRank Based Similarity
BIRCH	Balanced Iterative Reducing and Clustering using Hierarchies
BMF	Binary Matrix Factorization
BMFC	Binary Matrix Factorization-based Consensus
BOK	Best-of-K Consensus Algorithm
BOM	Best One-element Move
BSI	Bare Soil Index
CA	Co-association
CF	Clustering Feature
CNNs	Convolutional Neural Networks
CTS	Connected Triple Based Similarity
CSPA	Cluster-based Similarity Partitioning Algorithm
CURE	Clustering Using Representatives Algorithm
DB	Davies-Bouldin
DBSCAN	Density-based spatial clustering of applications with noise
DESP	Dynamic Ensemble Selection Performance
DF	Data Fragment
DFC	Data Fusion Contest

DFEC	Data Fragment-based Expressive Consensus
DNRTPM	Dynamic Normalization based Real-time Pattern Matching
DS	Dynamic Selection
DTW	Dynamic Time Warping
ELU	Exponential Linear Unit
EM	Expectation Maximization
HBGF	Hybrid Bipartite Graph Formulation
HGPA	Hyper-Graph Partitioning Algorithm
HSI	Hyperspectral imagery
IMP	Impurity Index
KB	Knowledge based cluster ensemble
KDE	Kernel Density Estimation
LCA	Local Classifier Accuracy
LCZ	Local Climate Zone
LS	Linear Sum
MCS	Multiple Classifier Systems
MCLA	Meta-Clustering Algorithm
MDNRTPM	Multiple Dynamic Normalization based Real-time Pattern Matching
MNF	Minimum Noise Fraction
NDMI	Normalized Difference Moisture Index
NDWI	Normalized Difference Water
NDVI	Normalized Difference Vegetation Index
NMF	Non-negative Matrix Factorization
NMI	Normalized Mutual Information

NP	Non-Polynomial time
NP-hard	Nondeterministic Polynomial time hard
OSM	Open Street Map
pdf	Probability density function
RS	Random Subspace
SAM	Spectral Angle Mapper
SI	Shadow Index
SRS	SimRank Based Similarity
SS	Square Sum
SSE	Sum of Squared Error
STWM	Subsequence Time-Warping Matrix
SVD	Singular Vector Decomposition
SVM	Support Vector Machines
TWM	Time Warping Matrix
UCR	University of California
WCT	Weighted Connected Triple

List of Symbols

$\mathbf{1}$	Vector of ones
$\underset{x}{\text{argmax}}\, y$	Returns the value of x that maximizes y
$A_{i,j}$	Element of $\mathbf{A}$
c_f	f^{th} element of vector $\mathbf{c}$
$c_{i,i}$	i^{th} element of the main diagonal of $\mathbf{C}$
$\text{diag}(\cdot)$	Returns a diagonal matrix when the argument is a vector, or returns a vector containing the elements of the main diagonal when the argument is a matrix
e	Base of the natural logarithm, also called Napier's constant
$\text{eigv}(\cdot)$	Returns the unit-norm principal eigenvector of a matrix
$\text{E}\{\cdot\}$	Expectation operator
$\mathbf{I}$	Identity matrix
j	$\sqrt{-1}$
$\min_i x_i$	Returns the minimum x_i for all possible indices i
N	Number of objects
$\text{rank}(\cdot)$	Rank of a matrix
$\text{tr}(\cdot)$	Trace of a matrix
$\mathcal{F}$	Set of all subchannel indices
$\mathcal{N}_k$	Set that contains the indices of users belonging to group k
$\mathcal{O}(\cdot)$	Complexity order of the argument
$\mathbb{C}$	Set of complex numbers
$\mathbb{R}$	Set of real numbers
$\mathbb{Z}$	Set of integer numbers
$(\cdot)^{\text{T}}$	Transpose of a vector or matrix
$(\cdot)^{\text{H}}$	Conjugate transpose of a vector or matrix
$(\cdot)^*$	Conjugate of a scalar, vector, or matrix
$(\cdot)^+$	Pseudoinverse of a vector or matrix
$(\cdot)^{-1}$	Inverse of a square matrix
$\lvert\cdot\rvert$	Absolute value of a scalar
$\lVert\cdot\rVert$	Euclidean norm or 2-norm of a vector
$\lVert\cdot\rVert_1$	1-norm of a vector

Bibliography

[AC01] J. Aach and G. M. Church. "Aligning gene expression time series with time warping algorithms". In: *Bioinformatics* 17.6 (2001), pp. 495–508 (cit. on p. 77).

[AWJ10] D. D. Abdala, P. Wattuya, and X. Jiang. "Ensemble Clustering via Random Walker Consensus Strategy". In: *2010 20th International Conference on Pattern Recognition*. Aug. 2010, pp. 1433–1436 (cit. on p. 29).

[AW19a] T. Alqurashi and W. Wang. "Clustering ensemble method". In: *International Journal of Machine Learning and Cybernetics* (2019), pp. 1227–1246 (cit. on p. 25).

[AW19b] T. Alqurashi and W. Wang. "Clustering ensemble method". In: *International Journal of Machine Learning and Cybernetics* 10.6 (June 2019), pp. 1227–1246. URL: https://doi.org/10.1007/s13042-017-0756-7 (cit. on p. 26).

[Anz12] Y. Anzai. *Pattern recognition and machine learning*. Elsevier, 2012 (cit. on p. 1).

[AE03] T. Argyros and C. Ermopoulos. "Efficient subsequence matching in time series databases under time and amplitude transformations". In: *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE. 2003, pp. 481–484 (cit. on p. 77).

[AM17] A. Arribas-Gil and C. Matias. "A time warping approach to multiple sequence alignment". In: *Statistical applications in genetics and molecular biology* 16 2 (2017), pp. 133–144 (cit. on pp. 76, 77, 79).

[AK08] H. G. Ayad and M. S. Kamel. "Cumulative Voting Consensus Method for Partitions with Variable Number of Clusters". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.1 (Jan. 2008), pp. 160–173 (cit. on p. 26).

[AK10] H. G. Ayad and M. S. Kamel. "On Voting-based Consensus of Cluster Ensembles". In: *Pattern Recogn.* 43.5 (May 2010), pp. 1943–1953. URL: http://dx.doi.org/10.1016/j.patcog.2009.11.012 (cit. on p. 26).

[Bel03] R. Bellman. *Dynamic Programming*. Dover Books on Computer Science Series. Dover Publications, 2003. URL: https://books.google.de/books?id=fyVtp3EMxasC (cit. on pp. 11, 20).

[BC94] D. J. Berndt and J. Clifford. "Using dynamic time warping to find patterns in time series." In: *KDD workshop*. Vol. 10. Seattle, WA. 1994, pp. 359–370 (cit. on p. 76).

[BV05] A. Bertoni and G. Valentini. "Random projections for assessing gene expression cluster stability". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 1. IEEE. 2005, pp. 149–154 (cit. on p. 24).

[BV06] A. Bertoni and G. Valentini. "Randomized maps for assessing the reliability of patients clusters in DNA microarray data analyses". In: *Artificial Intelligence in Medicine* 37.2 (2006), pp. 85–109. URL: http://www.sciencedirect.com/science/article/pii/S0933365706000364 (cit. on p. 25).

[Bol98] D. Boley. "Principal Direction Divisive Partitioning". In: *Data Mining and Knowledge Discovery* 2.4 (Dec. 1998), pp. 325–344. URL: https://doi.org/10.1023/A:1009740529316 (cit. on p. 16).

[BCK08] S. Boriah, V. Chandola, and V. Kumar. "Similarity Measures for Categorical Data: A Comparative Evaluation". In: *Proceedings of the SIAM International Conference on Data Mining, SDM, Atlanta, Georgia, USA*. 2008, pp. 243–254. URL: http://dx.doi.org/10.1137/1.9781611972788.22 (cit. on pp. 34, 36).

[Bre96] L. Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140. URL: http://dx.doi.org/10.1023/A:1018054314350 (cit. on pp. 21, 55).

[Bre01] L. Breiman. "Random Forests". In: *Mach. Learn.* 45.1 (2001) (cit. on pp. 53, 54, 63, 69).

[BSO14] A. S. Britto Jr, R. Sabourin, and L. E. Oliveira. "Dynamic selection of classifiers—a comprehensive review". In: *Pattern recognition* 47.11 (2014), pp. 3665–3680 (cit. on p. 64).

[BH91] P. Butzer and M. Hauss. "On Stirling functions of the second kind". In: *Studies in Applied Mathematics* 84.1 (1991), pp. 71–91 (cit. on p. 12).

[Cas13] F. Castanedo. "A review of data fusion techniques". In: *The Scientific World Journal* 2013 (2013) (cit. on p. 55).

[CK17] A. Chakraborty and A. K. Kar. "Swarm intelligence: A review of algorithms". In: *Nature-Inspired Computing and Optimization*. Springer, 2017, pp. 475–494 (cit. on p. 3).

[Cho+15] J. Choi, D. Kim, K. Plataniotis, and Y. Ro. "Classifier Ensemble Generation and Selection with Multiple Feature Representations for Classification Applications in Computer-Aided Detection and Diagnosis on Mammography". In: *Expert Systems with Applications* 46 (Oct. 2015) (cit. on p. 57).

[Cho+16] J. Y. Choi, D. H. Kim, K. N. Plataniotis, and Y. M. Ro. "Classifier ensemble generation and selection with multiple feature representations for classification applications in computer-aided detection and diagnosis on mammography". In: *Expert Systems with Applications* 46 (2016), pp. 106–121. URL: http://www.sciencedirect.com/science/article/pii/S0957417415007010 (cit. on p. 57).

[Cho17] F. Chollet. "Xception: Deep Learning With Depthwise Separable Convolutions". In: *Proceedings of the IEEE CVPR*. 2017, pp. 1251–1258 (cit. on p. 71).

[Chr11] I. T. Christou. "Coordination of Cluster Ensembles via Exact Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.2 (2011), pp. 279–293 (cit. on pp. 20, 21).

[Chu20] D. Chu. "Land-Cover Classification". In: *Remote Sensing of Land Use and Land Cover in Mountain Region*. Springer, 2020, pp. 181–194 (cit. on pp. 54, 56).

[CMU] CMU. *Graphics Lab Motion Capture Database*. URL: `mocap.cs.cmu.edu` (cit. on p. 93).

[CV95] C. Cortes and V. Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297 (cit. on p. 53).

[CSC18a] R. M. O. Cruz, R. T. Sabourin, and G. D. C. Cavalcanti. "Dynamic classifier selection: Recent advances and perspectives". In: *Information Fusion* 41 (2018), pp. 195–216. URL: `http://www.sciencedirect.com/science/article/pii/S1566253517304074` (cit. on p. 60).

[CSC18b] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti. "Dynamic classifier selection: Recent advances and perspectives". In: *Information Fusion* 41 (2018), pp. 195–216 (cit. on pp. 59, 64, 101).

[CCC16] Z. Cui, W. Chen, and Y. Chen. "Multi-Scale Convolutional Neural Networks for Time Series Classification". In: *CoRR* abs/1603.06995 (2016). arXiv: `1603.06995`. URL: `http://arxiv.org/abs/1603.06995` (cit. on p. 76).

[DH63] N. Dalkey and O. Helmer. "An experimental application of the Delphi method to the use of experts". In: *Management science* 9.3 (1963), pp. 458–467 (cit. on p. 2).

[Dau+18] H. A. Dau et al. *The UCR Time Series Classification Archive*. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`. 2018 (cit. on pp. 75, 79, 80, 93).

[Deb+14] C. Debes et al. "Hyperspectral and LiDAR Data Fusion: Outcome of the 2013 GRSS Data Fusion Contest". In: *IEEE JSTARS* 7.6 (2014), pp. 2405–2418 (cit. on pp. 53, 60).

[Den+19] Z. Deng, X. Zhu, Q. He, and L. Tang. "Land use/land cover classification using time series Landsat 8 images in a heavily urbanized area". In: *Advances in Space Research* 63.7 (2019), pp. 2144–2154 (cit. on p. 54).

[DFC] G. DFC. *2018 IEEE GRSS Data Fusion Contest*. `http://www.grss-ieee.org/community/technical-committees/data-fusion` (cit. on p. 70).

[DGK04] I. S. Dhillon, Y. Guan, and B. Kulis. "Kernel K-Means: Spectral Clustering and Normalized Cuts". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '04. Seattle, WA, USA: Association for Computing Machinery, 2004, pp. 551–556. URL: `https://doi.org/10.1145/1014052.1014118` (cit. on p. 19).

[Die+02] T. G. Dietterich et al. “Ensemble learning”. In: *The handbook of brain theory and neural networks* 2 (2002), pp. 110–125 (cit. on p. 4).

[DWH02] E. Dimitriadou, A. Weingessel, and K. Hornik. “A Combination Scheme for Fuzzy Clustering”. In: *Advances in Soft Computing — AFSS 2002*. Ed. by N. R. Pal and M. Sugeno. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 332–338 (cit. on p. 26).

[Don+20] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma. “A survey on ensemble learning”. In: *Frontiers of Computer Science* (2020), pp. 1–18 (cit. on p. 4).

[DF03] S. Dudoit and J. Fridlyand. “Bagging to improve the accuracy of a clustering procedure”. In: *Bioinformatics* 19.9 (June 2003), pp. 1090–1099. eprint: http://oup.prod.sis.lan/bioinformatics/article-pdf/19/9/1090/801378/btg038.pdf. URL: https://doi.org/10.1093/bioinformatics/btg038 (cit. on p. 26).

[Edw54] W. Edwards. “The theory of decision making.” In: *Psychological bulletin* 51.4 (1954), p. 380 (cit. on p. 2).

[EV13] E. Elhamifar and R. Vidal. “Sparse subspace clustering: Algorithm, theory, and applications”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.11 (2013), pp. 2765–2781 (cit. on p. 25).

[Est+96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. “A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD’96. Portland, Oregon: AAAI Press, 1996, pp. 226–231. URL: http://dl.acm.org/citation.cfm?id=3001460.3001507 (cit. on p. 17).

[Eve+11] B. S. Everitt, S. Landau, M. Leese, and D. Stahl. “Cluster Analysis. –John Wiley & Sons”. In: *Ltd., New York* (2011), p. 330 (cit. on p. 17).

[Fau09] M. Faundez-Zanuy. “Data Fusion at Different Levels”. In: *Multimodal Signals: Cognitive and Algorithmic Issues*. Ed. by A. Esposito, A. Hussain, M. Marinaro, and R. Martone. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 94–103 (cit. on p. 55).

[FD19] S. Feng and M. F. Duarte. “Few-shot learning-based human activity recognition”. In: *Expert Systems with Applications* 138 (2019), p. 112782. URL: http://www.sciencedirect.com/science/article/pii/S0957417419304786 (cit. on p. 101).

[FB04] X. Z. Fern and C. E. Brodley. “Solving Cluster Ensemble Problems by Bipartite Graph Partitioning”. In: *Proceedings of the International Conference on Machine Learning*. 2004 (cit. on pp. 29, 42, 49).

[FL08] X. Z. Fern and W. Lin. “Cluster Ensemble Selection”. In: *Statistical Analysis and Data Mining* 1.3 (2008), pp. 128–141. URL: http://dx.doi.org/10.1002/sam.10008 (cit. on p. 32).

[FS03] V. Filkov and S. Skiena. "Integrating microarray data by consensus clustering". In: *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*. Nov. 2003, pp. 418–426 (cit. on p. 33).

[FB03] B. Fischer and J. M. Buhmann. "Bagging for path-based clustering". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.11 (Nov. 2003), pp. 1411–1415 (cit. on p. 26).

[FVH06] P. Franti, O. Virmajoki, and V. Hautamaki. "Fast Agglomerative Clustering Using a k-Nearest Neighbor Graph". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 28. 11. 2006, pp. 1875–1881. URL: http://dx.doi.org/10.1109/TPAMI.2006.227 (cit. on pp. 20, 41).

[FJ05] A. L. N. Fred and A. K. Jain. "Combining Multiple Clusterings Using Evidence Accumulation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 27.6 (2005), pp. 835–850. URL: http://dx.doi.org/10.1109/TPAMI.2005.113 (cit. on pp. 25, 28).

[Fre01] A. Fred. "Finding Consistent Clusters in Data Partitions". In: *Multiple Classifier Systems*. Ed. by J. Kittler and F. Roli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 309–318 (cit. on pp. 28, 29).

[FD07] B. J. Frey and D. Dueck. "Clustering by passing messages between data points". In: *Science* 315 (2007), p. 2007 (cit. on p. 40).

[Fri00] J. H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *Annals of Statistics* 29 (2000), pp. 1189–1232 (cit. on pp. 54, 69).

[Fu+08] A. W. Fu, E. Keogh, L. Y. Lau, C. A. Ratanamahatana, and R. C. Wong. "Scaling and time warping in time series querying". In: *The International Journal on Very Large Data Bases* 17.4 (2008), pp. 899–921 (cit. on p. 77).

[FM07] L. Fu and E. Medico. "FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data". In: *BMC Bioinformatics* 8.1 (2007), pp. 1–15. URL: http://dx.doi.org/10.1186/1471-2105-8-3 (cit. on pp. 20, 41).

[Fu11] T. Fu. "A review on time series data mining". In: *Engineering Applications of Artificial Intelligence* 24.1 (2011), pp. 164–181 (cit. on pp. 75, 76, 82).

[Fuk13] K. Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013 (cit. on p. 1).

[Gal+12] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.4 (July 2012), pp. 463–484 (cit. on p. 60).

[GA16] B. C. Giao and D. T. Anh. "Improving spring method in similarity search over time-series streams by data normalization". In: *International Conference on Nature of Computation and Communication*. Springer. 2016, pp. 189–202 (cit. on pp. 77, 90).

[GMT07] A. Gionis, H. Mannila, and P. Tsaparas. "Clustering Aggregation". In: *ACM Trans. Knowl. Discov. Data* 1.1 (2007). URL: http://doi.acm.org/10.1145/1217299.1217303 (cit. on pp. 33, 36, 41, 43, 50).

[GF08] A. Goder and V. Filkov. "Consensus clustering algorithms: Comparison and refinement". In: *Proceedings of the Meeting on Algorithm Engineering & Expermiments*. Society for Industrial and Applied Mathematics. 2008, pp. 109–117 (cit. on pp. 20, 21).

[GFS18] X. Gong, S. Fong, and Y. Si. "Fast multi-subsequence monitoring on streaming time-series based on Forward-propagation". In: *Inf. Sci.* 450 (2018), pp. 73–88 (cit. on pp. 75, 77).

[Gon+14] X. Gong, Y. Si, S. Fong, and S. Mohammed. "NSPRING: Normalization-supported SPRING for subsequence matching on time series streams". In: *2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI)*. 2014 (cit. on pp. 77, 90).

[GV01] A. D. Gordon and M. Vichi. "Fuzzy partition models for fitting a set of partitions". In: *Psychometrika* 66.2 (June 2001), pp. 229–247. URL: https://doi.org/10.1007/BF02294837 (cit. on p. 26).

[GRS98] S. Guha, R. Rastogi, and K. Shim. "CURE: An Efficient Clustering Algorithm for Large Databases". In: *SIGMOD Rec.* 27.2 (June 1998), pp. 73–84. URL: http://doi.acm.org/10.1145/276305.276312 (cit. on p. 16).

[Ham50] R. W. Hamming. "Error-detecting and error-correcting codes". In: *Bell System Technical Journal*. Vol. 29(2). 1950, pp. 147–160 (cit. on p. 34).

[Has19] M. Hassani. *Overview of Efficient Clustering Methods for High-Dimensional Big Data Streams*. Ed. by O. Nasraoui and C.-E. Ben N'Cir. Cham: Springer International Publishing, 2019, pp. 25–42. URL: https://doi.org/10.1007/978-3-319-97864-2_2 (cit. on p. 1).

[HCK13] B. Hu, Y. Chen, and E. Keogh. "Time series classification under more realistic assumptions". In: *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM. 2013, pp. 578–586 (cit. on p. 93).

[Hua+17] D. Huang, C.-D. Wang, J.-H. Lai, and C.-K. Kwoh. "Toward Multi-Diversified Ensemble Clustering of High-Dimensional Data". In: *arXiv preprint arXiv:1710.03113* (2017) (cit. on p. 24).

[HA85] L. Hubert and P. Arabie. "Comparing partitions". In: *Journal of Classification* 2.1 (1985), pp. 193–218. URL: http://dx.doi.org/10.1007/BF01908075 (cit. on pp. 41, 49).

[Hud+13] D. Hudson, A. G. Marshall, Y. Yin, O. Alves, and H. H. Hendon. "Improving intraseasonal prediction with a new ensemble generation strategy". In: *Monthly Weather Review* 141.12 (2013), pp. 4429–4449 (cit. on p. 57).

[Iam+11] N. Iam-On, T. Boongoen, S. Garrett, and C. Price. "A Link-Based Approach to the Cluster Ensemble Problem". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.12 (2011), pp. 2396–2409 (cit. on p. 21).

[IBG08] N. Iam-on, T. Boongoen, and S. Garrett. "Refining Pairwise Similarity Matrix for Cluster Ensemble Problem with Cluster Relations". In: *Discovery Science*. Ed. by J.-F. Jean-Fran, M. R. Berthold, and T. Horváth. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 222–233 (cit. on p. 29).

[Ien+17] D. Ienco, R. Gaetano, C. Dupaquier, and P. Maurel. "Land Cover Classification via Multitemporal Spatial Data by Deep Recurrent Neural Networks". In: *IEEE Geosci. Remote Sensing Lett.* 14.10 (2017), pp. 1685–1689. URL: https://doi.org/10.1109/LGRS.2017.2728698 (cit. on p. 48).

[IS15] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *ICML*. 2015, pp. 448–456 (cit. on p. 71).

[Ism+19] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. Muller. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* (Mar. 2019). URL: https://doi.org/10.1007/s10618-019-00619-1 (cit. on p. 76).

[IK20] T. Iwata and A. Kumagai. *Few-shot Learning for Time-series Forecasting*. 2020. arXiv: 2009.14379 [stat.ML]. URL: https://arxiv.org/pdf/2009.14379.pdf (cit. on p. 101).

[Jac01] P. Jaccard. "Étude comparative de la distribution florale dans une portion des Alpes et des Jura". In: *Bulletin del la Société Vaudoise des Sciences Naturelles* 37 (1901), pp. 547–579 (cit. on p. 30).

[JL05] A. K. Jain and M. H. C. Law. "Data Clustering: A User's Dilemma". In: *Pattern Recognition and Machine Intelligence: First International Conference, PReMI 2005, Kolkata, India, December 20-22, 2005. Proceedings*. Ed. by S. K. Pal, S. Bandyopadhyay, and S. Biswas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–10 (cit. on p. 20).

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. "Data Clustering: A Review". In: *ACM Comput. Surv.* 31.3 (Sept. 1999), pp. 264–323. URL: http://doi.acm.org/10.1145/331499.331504 (cit. on pp. 12, 13, 20, 29).

[JV86] R. Jonker and T. Volgenant. "Improving the Hungarian assignment algorithm". In: *Operations Research Letters* 5.4 (1986), pp. 171–175. URL: http://www.sciencedirect.com/science/article/pii/0167637786900738 (cit. on p. 26).

[Jos+16] N. Joshi, M. Baumann, A. Ehammer, R. Fensholt, K. Grogan, P. Hostert, M. R. Jepsen, T. Kuemmerle, P. Meyfroidt, E. T. Mitchard, et al. "A review of the application of optical and radar remote sensing data fusion to land use mapping and monitoring". In: *Remote Sensing* 8.1 (2016), p. 70 (cit. on p. 53).

[Kar+99] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. "Multilevel hypergraph partitioning: Application in VLSI domain". In: *IEEE TRANS. VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS*. 1999, pp. 69–529 (cit. on p. 31).

[KK98a] G. Karypis and V. Kumar. "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs". In: *SIAM J. Sci. Comput.* 20.1 (Dec. 1998), pp. 359–392. URL: http://dx.doi.org/10.1137/S1064827595287997 (cit. on pp. 30, 31).

[KK98b] G. Karypis and V. Kumar. "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs". In: *SIAM J. Sci. Comput.* 20.1 (Dec. 1998), pp. 359–392. URL: http://dx.doi.org/10.1137/S1064827595287997 (cit. on p. 29).

[KK03] E. Keogh and S. Kasetty. "On the need for time series data mining benchmarks: a survey and empirical demonstration". In: *Data Mining and knowledge discovery* 7.4 (2003), pp. 349–371 (cit. on p. 75).

[Keo+04] E. Keogh, T. Palpanas, V. Zordan, D. Gunopulos, and M. Cardle. "Indexing large human-motion databases". In: *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment. 2004, pp. 780–791 (cit. on p. 77).

[Kha+13] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi. "Multisensor data fusion: A review of the state-of-the-art". In: *Information fusion* 14.1 (2013), pp. 28–44 (cit. on p. 53).

[KMM19] S. Khorram, M. G. McInnis, and E. Mower Provost. "Trainable Time Warping: Aligning Time-series in the Continuous-time Domain". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2019, pp. 3502–3506 (cit. on p. 79).

[Kno+06] Z. F. Knops, J. A. Maintz, M. A. Viergever, and J. P. Pluim. "Normalized mutual information based registration using k-means clustering and shading correction". In: *Medical image analysis* 10.3 (2006), pp. 432–439 (cit. on p. 29).

[KG03] M. Koyutürk and A. Grama. "PROXIMUS: A Framework for Analyzing Very High Dimensional Discrete-attributed Datasets". In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '03. Washington, D. C.: ACM, 2003, pp. 147–156. URL: http://doi.acm.org/10.1145/956750.956770 (cit. on pp. 44, 46–48).

[KL83] J. Kruskal and M. Liberman. *The Symmetric Time-Warping Problem: From Continuous to Discrete*. 1983 (cit. on p. 76).

[Kun16] L. Kuncheva. *Artificial data sets [online]*. 2016. URL: http://pages.bangor.ac.uk/%5C~mas00a/activities/artificial%5C_data.htm (cit. on pp. 41, 48).

[Kun02] L. I. Kuncheva. “A theoretical study on six classifier fusion strategies”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 2 (2002), pp. 281–286 (cit. on pp. 54, 55, 57, 59).

[Le+10] Q. V. Le, J. Ngiam, Z. Chen, D. Chia, P. W. Koh, and A. Y. Ng. “Tiled convolutional neural networks”. In: *NIPS*. 2010 (cit. on p. 68).

[LMT16] A. Le Guennec, S. Malinowski, and R. Tavenard. “Data augmentation for time series classification using convolutional neural networks”. In: *ECML/PKDD workshop on advanced analytics and learning on temporal data*. 2016 (cit. on p. 76).

[Lec+98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proc. IEEE*. 1998 (cit. on pp. 68, 69).

[LWB90] J. B. Lee, A. S. Woodyatt, and M. Berman. “Enhancement of high spectral resolution remote-sensing data by a noise-adjusted principal components transform”. In: *IEEE Transactions on Geoscience and Remote Sensing* 28 (1990) (cit. on p. 67).

[LS99] D. D. Lee and H. S. Seung. “Learning the parts of objects by nonnegative matrix factorization”. In: *Nature* 401 (1999), pp. 788–791 (cit. on p. 44).

[Li05] T. Li. “A General Model for Clustering Binary Data”. In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. KDD ’05. Chicago, Illinois, USA: ACM, 2005, pp. 188–197. URL: http://doi.acm.org/10.1145/1081870.1081894 (cit. on pp. 44, 46).

[LDJ07] T. Li, C. Ding, and M. I. Jordan. “Solving Consensus and Semi-supervised Clustering Problems Using Nonnegative Matrix Factorization”. In: *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. 2007, pp. 577–582 (cit. on p. 37).

[Li+07] Y. Li, J. Yu, P. Hao, and Z. Li. “Clustering Ensembles Based on Normalized Edges”. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Z.-H. Zhou, H. Li, and Q. Yang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 664–671 (cit. on p. 29).

[Lic13] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: http://archive.ics.uci.edu/ml (cit. on pp. 42, 48).

[Lou+13] A. Lourenço, S. R. Bulò, A. L. N. Fred, and M. Pelillo. “Consensus Clustering with Robust Evidence Accumulation”. In: *Energy Minimization Methods in Computer Vision and Pattern Recognition - 9th International Conference, EMMCVPR 2013, Lund, Sweden, August 19-21, 2013. Proceedings*. 2013, pp. 307–320. URL: https://doi.org/10.1007/978-3-642-40395-8_23 (cit. on p. 25).

[Lu+] H. Lu, J. Vaidya, V. Atluri, H. Shin, and L. Jiang. "Weighted Rank-One Binary Matrix Factorization". In: *Proceedings of the 2011 SIAM International Conference on Data Mining*, pp. 283–294. eprint: http://epubs.siam.org/doi/pdf/10.1137/1.9781611972818.25. URL: http://epubs.siam.org/doi/abs/10.1137/1.9781611972818.25 (cit. on p. 46).

[Mac67] J. MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297. URL: http://projecteuclid.org/euclid.bsmsp/1200512992 (cit. on pp. 12, 64).

[Mar18] P. Marteau. "Times series averaging and denoising from a probabilistic perspective on time-elastic kernels". In: *International Journal of Applied Mathematics and Computer Science* (2018). URL: https://hal.archives-ouvertes.fr/hal-01401072 (cit. on p. 76).

[MDH14] A. Merentitis, C. Debes, and R. Heremans. "Ensemble Learning in Hyperspectral Image Classification: Toward Selecting a Favorable Bias-Variance Tradeoff". In: *IEEE JSTARS* 7.4 (2014) (cit. on pp. 53–55).

[Mil17] F. Miller. "Aristotle's Political Theory". In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Winter 2017. Metaphysics Research Lab, Stanford University, 2017 (cit. on p. 1).

[MAY] R. Millham, I. E. Agbehadji, and H. Yang. "Pattern Mining Algorithms". In: *Bio-inspired Algorithms for Data Streaming and Visualization, Big Data Management, and Fog Computing*. Springer, pp. 67–80 (cit. on p. 1).

[Mor+18] M. Morel, C. Achard, R. Kulpa, and S. Dubuisson. "Time-series averaging using constrained dynamic time warping with tolerance". In: *Pattern Recognition* 74 (Feb. 2018), pp. 77–89. URL: https://hal.sorbonne-universite.fr/hal-01630288 (cit. on p. 76).

[Mül07] M. Müller. "Dynamic time warping". In: *Information retrieval for music and motion* (2007), pp. 69–84 (cit. on p. 76).

[NH10] V. Nair and G. E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *ICML*. Omnipress, 2010, pp. 807–814. URL: http://www.icml2010.org/papers/432.pdf (cit. on p. 68).

[Nar+19] J. Narwariya, P. Malhotra, L. Vig, G. Shroff, and V. Tv. *Meta-Learning for Few-Shot Time Series Classification*. 2019. arXiv: 1909.07155 [cs.LG]. URL: https://arxiv.org/pdf/1909.07155.pdf (cit. on p. 101).

[NG10] I. Natthakan and S. Garrett. "LinkCluE: A MATLAB Package for Link-Based Cluster Ensembles". In: *Journal of Statistical Software* 36.1 (2010), pp. 1–36. URL: https://www.jstatsoft.org/index.php/jss/article/view/v036i09 (cit. on pp. 26–28, 34, 42, 49, 50).

[NJW02] A. Y. Ng, M. I. Jordan, and Y. Weiss. "On spectral clustering: Analysis and an algorithm". In: *Advances in neural information processing systems*. 2002, pp. 849–856 (cit. on p. 18).

[Ngu+20] T. T. Nguyen, A. V. Luong, M. T. Dang, A. W.-C. Liew, and J. McCall. "Ensemble Selection based on Classifier Prediction Confidence". In: *Pattern Recognition* 100 (2020), p. 107104 (cit. on p. 57).

[NWR10] V. Niennattrakul, D. Wanichsan, and C. A. Ratanamahatana. "Accurate Subsequence Matching on Data Stream under Time Warping Distance". In: *New Frontiers in Applied Data Mining Lecture Notes in Computer Science* (2010), pp. 156–167 (cit. on pp. 77, 93).

[Oga+10] E. Ogasawara, L. C. Martinez, D. Oliveira, G. Zimbrão, G. L. Pappa, and M. Mattoso. "Adaptive normalization: A novel data normalization approach for non-stationary time series". In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2010, pp. 1–8 (cit. on p. 77).

[Oin08] H. Oinas-Kukkonen. "Network analysis and crowds of people as sources of new organisational knowledge". In: *Knowledge Management: Theoretical Foundation* (2008), pp. 173–189 (cit. on p. 2).

[Ore+17] I. Oregi, A. Pérez, J. Del Ser, and J. A. Lozano. "On-line dynamic time warping for streaming time series". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2017, pp. 591–605 (cit. on p. 76).

[Osh+13] S. Oshigami, Y. Yamaguchi, T. Uezato, A. Momose, Y. Arvelyna, Y. Kawakami, T. Yajima, S. Miyatake, and A. Nguno. "Mineralogical mapping of southern Namibia by application of continuum-removal MSAM method to the HyMap data". In: *Int. J. Remote Sens.* 34.15 (2013) (cit. on p. 67).

[PK06] M. S. P. Tan and V. Kumar. "Introduction to Data Mining". In: Addison-Wesley Companion, 2006. Chap. Cluster Analysis: Basic Concepts and Algorithm, pp. 487–568 (cit. on p. 18).

[PAM12] PAMAP. *Physical Activity Monitoring for Aging People*. 2012. URL: `www.pamap.org/demo.html` (cit. on pp. 75, 93).

[PUG14] S. P. Parambath, N. Usunier, and Y. Grandvalet. "Optimizing F-measures by cost-sensitive classification". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2123–2131 (cit. on p. 65).

[PHL04] L. Parsons, E. Haque, and H. Liu. "Subspace clustering for high dimensional data: a review". In: *Acm Sigkdd Explorations Newsletter* 6.1 (2004), pp. 90–105 (cit. on p. 24).

[Pen+08] Z. Peng, S. Liang, J. Yan, W. Han, and S. Yang. "Fast similarity matching on data stream with noise". In: *2008 IEEE 24th International Conference on Data Engineering Workshop*. 2008 (cit. on p. 77).

[PKG11] F. Petitjean, A. Ketterlin, and P. Gançarski. "A Global Averaging Method for Dynamic Time Warping, with Applications to Clustering". In: *Pattern Recogn.* 44.3 (Mar. 2011), pp. 678–693. URL: `http://dx.doi.org/10.1016/j.patcog.2010.09.013` (cit. on pp. 76, 79, 80, 90).

[PS11] S. V. Pons and J. R. Shulcloper. "A Survey of Clustering Ensemble Algorithms". In: *International Journal of Pattern Recognition and Artifical Intelligence* 25.3 (2011), pp. 337–372. URL: http://dx.doi.org/10.1142/S0218001411008683 (cit. on pp. 21, 23, 25, 33, 38).

[Qin+19] R. Qin, X. Huang, W. Liu, and C. Xiao. "Semantic 3D Reconstruction Using Multi-View High-Resolution Satellite Images Based on U-Net and Image-Guided Depth Fusion". In: *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*. July 2019, pp. 5057–5060 (cit. on p. 53).

[Rak+12] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. "Searching and mining trillions of time series subsequences under dynamic time warping". In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 12*. 2012 (cit. on pp. 75–77, 80, 83, 90).

[RK04] C. A. Ratanamahatana and E. Keogh. "Making time-series classification more accurate using learned constraints". In: *Proceedings of the 2004 SIAM International Conference on Data Mining*. SIAM. 2004, pp. 11–22 (cit. on p. 93).

[RM03] T. M. Rath and R. Manmatha. "Word image matching using dynamic time warping". In: *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. Vol. 2. IEEE. 2003, pp. II–II (cit. on p. 76).

[RR20] V. H. A. Ribeiro and G. Reynoso-Meza. "Ensemble learning by means of a multi-objective optimization design approach for dealing with imbalanced data sets". In: *Expert Systems with Applications* (2020), p. 113232 (cit. on p. 57).

[RNR11] S. Rodpongpun, V. Niennattrakul, and C. A. Ratanamahatana. "Efficient subsequence search on streaming data based on time warping distance". In: *ECTI Transactions on Computer and Information Technology (ECTI-CIT)* 5.1 (2011), pp. 2–8 (cit. on p. 77).

[RM05] L. Rokach and O. Maimon. "Clustering Methods". In: *Data Mining and Knowledge Discovery Handbook*. Ed. by O. Maimon and L. Rokach. Boston, MA: Springer US, 2005, pp. 321–352. URL: http://dx.doi.org/10.1007/0-387-25465-X_15 (cit. on p. 14).

[SR18] O. Sagi and L. Rokach. "Ensemble learning: A survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4 (2018), e1249 (cit. on p. 3).

[SRS08] S. Saitta, B. Raphael, and I. F. Smith. "A comprehensive validity index for clustering". In: *Intelligent Data Analysis* 12.6 (2008), pp. 529–548 (cit. on p. 20).

[SFY07] Y. Sakurai, C. Faloutsos, and M. Yamamuro. "Stream Monitoring under the Time Warping Distance". In: *2007 IEEE 23rd International Conference on Data Engineering*. 2007 (cit. on pp. 76, 78, 86).

[Scu10] D. Sculley. "Web-scale K-means Clustering". In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: ACM, 2010, pp. 1177–1178. URL: http://doi.acm.org/10.1145/1772690.1772862 (cit. on p. 48).

[SJ13] C. Shah and A. Jivani. "Comparison of data mining clustering algorithms". In: *2013 Nirma University International Conference on Engineering (NUiCONE)*. 2013, pp. 1–4 (cit. on pp. 36, 38, 39, 101).

[SJY09] B.-H. Shen, S. Ji, and J. Ye. "Mining Discrete Patterns via Binary Matrix Factorization". In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '09. Paris, France: ACM, 2009, pp. 757–766. URL: http://doi.acm.org/10.1145/1557019.1557103 (cit. on pp. 45, 46).

[She+17] Y. Shen, Y. Chen, E. Keogh, and H. Jin. "Searching time series with invariance to large amounts of uniform scaling". In: *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE. 2017, pp. 111–114 (cit. on p. 77).

[She+18] Y. Shen, Y. Chen, E. Keogh, and H. Jin. "Accelerating Time Series Searching with Large Uniform Scaling". In: *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM. 2018, pp. 234–242 (cit. on p. 77).

[SWS14] Z. Shi, L. Wang, and L. Shi. "Approximation method to rank-one binary matrix factorization". In: *2014 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE. 2014, pp. 800–805 (cit. on p. 46).

[SCG15] S. Soheily-Khah, A. D. Chouakria, and É. Gaussier. "Progressive and Iterative Approaches for Time Series Averaging". In: *AALTD@PKDD/ECML*. 2015 (cit. on pp. 76, 77, 79).

[Sri+14a] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014) (cit. on p. 69).

[Sri+14b] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A simple way to prevent neural networks from overfitting". In: *JMLR* 15.1 (2014), pp. 1929–1958 (cit. on p. 71).

[SO12] I. D. Stewart and T. R. Oke. "Local Climate Zones for Urban Temperature Studies". In: *Bulletin of the American Meteorological Society* 93.12 (2012) (cit. on pp. 54, 56).

[SG03] A. Strehl and J. Ghosh. "Cluster Ensembles — a Knowledge Reuse Framework for Combining Multiple Partitions". In: *J. Mach. Learn. Res.* 3 (2003), pp. 583–617. URL: http://dx.doi.org/10.1162/153244303321897735 (cit. on pp. 20, 21, 23, 29, 31, 32, 42, 49, 50).

[Suk+18a] S. Sukhanov, D. Budylskii, I. Tankoyeu, R. Heremans, and C. Debes. "Fusion of Lidar, Hyperspectral and RGB Data for Urban Land Use and Land Cover Classification". In: *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*. July 2018, pp. 3864–3867 (cit. on pp. 53, 54, 71).

[SDZ18] S. Sukhanov, C. Debes, and A. M. Zoubir. "Interpretable clustering ensembles using binary matrix factorization". In: *Proc. 43th IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*. 2018 (cit. on pp. 48, 49).

[SDZ19] S. Sukhanov, C. Debes, and A. M. Zoubir. "Dynamic selection of classifiers for fusing imbalanced heterogeneous data". In: *Proc. 44th IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*. 2019 (cit. on p. 54).

[Suk+17a] S. Sukhanov, V. Gupta, C. Debes, and A. M. Zoubir. "Consensus clustering on data fragments". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2017, pp. 4631–4635 (cit. on pp. 35, 38, 41, 46).

[Suk+15] S. Sukhanov, A. Merentitis, C. Debes, J. Hahn, and A. M. Zoubir. "Bootstrap-based SVM aggregation for class imbalance problems". In: *Signal Processing Conference (EUSIPCO), 2015 23rd European*. Aug. 2015, pp. 165–169 (cit. on p. 57).

[Suk+18b] S. Sukhanov, A. Merentitis, C. Debes, J. Hahn, and A. M. Zoubir. "Combining SVMs for Classification on Class Imbalanced Data". In: *2018 IEEE Statistical Signal Processing Workshop (SSP)*. June 2018, pp. 90–94 (cit. on pp. 56, 57, 60).

[Suk+17b] S. Sukhanov, I. Tankoyeu, J. Louradour, R. Heremans, D. Trofimova, and C. Debes. "Multilevel ensembling for local climate zones classification". In: *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. July 2017, pp. 1201–1204 (cit. on pp. 53–55, 66).

[Suk+20] S. Sukhanov, R. Wu, C. Debes, and A. M. Zoubir. "Dynamic pattern matching with multiple queries on large scale data streams". In: *Signal Processing* 171 (2020), p. 107402. URL: http://www.sciencedirect.com/science/article/pii/S0165168419304542 (cit. on pp. 75, 86).

[Sur05] J. Surowiecki. *The wisdom of crowds*. Anchor, 2005 (cit. on p. 1).

[TSK14] P. N. Tan, M. Steinbach, and V. Kumar. *Introduction to data mining*. Pearson Addison Wesley, 2014 (cit. on p. 77).

[TLL20] W. Tang, L. Liu, and G. Long. *Interpretable Time-series Classification on Few-shot Samples*. 2020. arXiv: 2006.02031 [cs.LG]. URL: https://arxiv.org/pdf/2006.02031.pdf (cit. on p. 101).

[Ton+20] X.-Y. Tong, G.-S. Xia, Q. Lu, H. Shen, S. Li, S. You, and L. Zhang. "Land-cover classification with high-resolution remote sensing images using transferable deep models". In: *Remote Sensing of Environment* 237 (2020), p. 111322 (cit. on p. 54).

[TJP05] A. Topchy, A. K. Jain, and W. Punch. "Clustering ensembles: models of consensus and weak partitions". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.12 (2005), pp. 1866–1881 (cit. on p. 21).

[Top+04] A. P. Topchy, M. H. C. Law, A. K. Jain, and A. L. Fred. "Analysis of Consensus Partition in Cluster Ensemble". In: *Proceedings of the Fourth IEEE International Conference on Data Mining*. ICDM '04. 2004, pp. 225–232. URL: http://dl.acm.org/citation.cfm?id=1032649.1033458 (cit. on pp. 20, 21).

[Tui+17] D. Tuia, G. Moser, B. L. Saux, B. Bechtel, and L. See. "2017 IEEE GRSS Data Fusion Contest: open data for global multimodal land use classification". In: *IEEE Geosci. Remote Sens. Mag.,* 5 (2017) (cit. on pp. 53, 66).

[Tul+08] S. Tulyakov, S. Jaeger, V. Govindaraju, and D. Doermann. "Review of classifier combination methods". In: *Machine learning in document analysis and recognition*. Springer, 2008, pp. 361–386 (cit. on pp. 54, 59).

[TA08] K. Tumer and A. K. Agogino. "Ensemble Clustering with Voting Active Clusters". In: *Pattern Recogn. Lett.* 29.14 (Oct. 2008), pp. 1947–1953. URL: http://dx.doi.org/10.1016/j.patrec.2008.06.011 (cit. on p. 26).

[VA15] S. Vega-Pons and P. Avesani. "On pruning the search space for clustering ensemble problems". In: *Neurocomputing* 150, Part B (2015), pp. 481–489. URL: http://www.sciencedirect.com/science/article/pii/S0925231214012272 (cit. on pp. 33, 34).

[VR09] S. Vega-Pons and J. Ruiz-Shulcloper. "Clustering Ensemble Method for Heterogeneous Partitions". In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Ed. by E. Bayro-Corrochano and J.-O. Eklundh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 481–488 (cit. on p. 29).

[WZX15] J. Wang, Q. Zhan, and Y. Xiao. "Hierarchical Climate Zone as a tool for spatial planning – Case study of Wuhan, China". In: *Proc. Int. Conf. on Computers in Urban Planning and Urban Management*. 2015 (cit. on p. 54).

[Wan11] T. Wang. "CA-Tree: A Hierarchical Structure for Efficient and Scalable Coassociation-Based Cluster Ensembles". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 41.3 (2011), pp. 686–698 (cit. on pp. 25, 33, 49).

[WYZ09] X. Wang, C. Yang, and J. Zhou. "Clustering Aggregation by Probability Accumulation". In: *Pattern Recogn.* 42.5 (May 2009), pp. 668–675. URL: http://dx.doi.org/10.1016/j.patcog.2008.09.013 (cit. on p. 29).

[Wan+15] Y. Wang, X. Lin, L. Wu, W. Zhang, Q. Zhang, and X. Huang. "Robust subspace clustering for multi-view data by exploiting correlation consensus". In: *IEEE Transactions on Image Processing* 24.11 (2015), pp. 3939–3949 (cit. on p. 24).

[WYO16] Z. Wang, W. Yan, and T. Oates. "Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline". In: *CoRR* abs/1611.06455 (2016). arXiv: 1611.06455. URL: http://arxiv.org/abs/1611.06455 (cit. on p. 76).

[Wol+12] T. Woloszynski, M. Kurzynski, P. Podsiadlo, and G. W. Stachowiak. "A measure of competence based on random classification for dynamic ensemble selection". In: *Information Fusion* 13.3 (2012), pp. 207–213 (cit. on p. 64).

[WKB97] K. Woods, W. P. Kegelmeyer, and K. Bowyer. "Combination of multiple classifiers using local accuracy estimates". In: *IEEE transactions on pattern analysis and machine intelligence* 19.4 (1997), pp. 405–410 (cit. on p. 64).

[WGC14] M. Woźniak, M. Graña, and E. Corchado. "A survey of multiple classifier systems as hybrid systems". In: *Information Fusion* 16 (2014), pp. 3–17 (cit. on pp. 54, 55).

[Wu+12] O. Wu, W. Hu, S. J. Maybank, M. Zhu, and B. Li. "Efficient Clustering Aggregation Based on Data Fragments." In: *IEEE Trans. Systems, Man, and Cybernetics, Part B* 42.3 (2012), pp. 913–926. URL: http://dblp.uni-trier.de/db/journals/tsmc/tsmcb42.html (cit. on pp. 33, 34).

[Xu+19] Y. Xu, B. Du, L. Zhang, D. Cerra, M. Pato, E. Carmona, S. Prasad, N. Yokoya, R. Hänsch, and B. Le Saux. "Advanced Multi-Sensor Optical Remote Sensing for Urban Land Use and Land Cover Classification: Outcome of the 2018 IEEE GRSS Data Fusion Contest". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.6 (June 2019), pp. 1709–1724 (cit. on pp. 53, 54).

[Yan+09] A. Yang, A. Giani, R. Giannatonio, K. Gilani, et al. "Distributed human action recognition via wearable motion sensor networks". In: *Journal of Ambient Intelligence and Smart Environments* 1.2 (2009), pp. 103–115 (cit. on p. 93).

[Yok+18] N. Yokoya, P. Ghamisi, J. Xia, S. Sukhanov, R. Heremans, I. Tankoyeu, B. Bechtel, B. L. Saux, G. Moser, and D. Tuia. "Open Data for Global Multimodal Land Use Classification: Outcome of the 2017 IEEE GRSS Data Fusion Contest". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2018), pp. 1–15 (cit. on pp. 53, 54, 66).

[YS03] S. X. Yu and J. Shi. "Multiclass spectral clustering". In: *Proceedings Ninth IEEE International Conference on Computer Vision*. Oct. 2003, 313–319 vol.1 (cit. on p. 19).

[ZP05] L. Zelnik-Manor and P. Perona. "Self-tuning spectral clustering". In: *Advances in neural information processing systems*. 2005, pp. 1601–1608 (cit. on p. 19).

[Zha+20] C. Zhang, P. A. Harrison, X. Pan, H. Li, I. Sargent, and P. M. Atkinson. "Scale Sequence Joint Deep Learning (SS-JDL) for land use and land cover classification". In: *Remote Sensing of Environment* 237 (2020), p. 111593 (cit. on p. 54).

[ZZ09] M. Zhang and Z. Zhou. "Multi-instance clustering with applications to multi-instance prediction". In: *Applied Intelligence* 31.1 (2009), pp. 47–68. URL: http://dx.doi.org/10.1007/s10489-007-0111-x (cit. on p. 43).

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: An Efficient Data Clustering Method for Very Large Databases". In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*. SIGMOD '96. Montreal, Quebec, Canada: ACM, 1996, pp. 103–114. URL: http://doi.acm.org/10.1145/233269.233324 (cit. on pp. 16, 17, 48).

[Zha+07] Z. Zhang, T. Li, C. Ding, and X. Zhang. "Binary Matrix Factorization with Applications". In: *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*. ICDM '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 391–400. URL: https://doi.org/10.1109/ICDM.2007.99 (cit. on pp. 44, 45).

[Zha+17] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu. "Convolutional neural networks for time series classification". In: *Journal of Systems Engineering and Electronics* 28 (Feb. 2017), pp. 162–169 (cit. on p. 76).

[ZW06] Z. Zhi-Hua and T. Wei. "Clusterer ensemble". In: *Knowl.-Based Syst.* 19.1 (2006), pp. 77–83. URL: http://dx.doi.org/10.1016/j.knosys.2005.11.003 (cit. on pp. 32, 42, 49).

[ZT06] Z.-H. Zhou and W. Tang. "Clusterer ensemble". In: *Knowledge-Based Systems* 19.1 (2006), pp. 77–83 (cit. on p. 26).

[Zou+18] A. M. Zoubir, V. Koivunen, E. Ollila, and M. Muma. *Robust Statistics for Signal Processing*. Cambridge University Press, 2018 (cit. on pp. 57, 80).

Erklärungen laut Promotionsordnung

§ 8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§ 8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde.

§ 9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§ 9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 15.03.2021