Orometry, Intrinsic Dimensionality and Learning: Novel Insights into Network Data

Dissertation

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt im

Fachbereich Elektrotechnik/Informatik

der

Universität Kassel

von

Maximilian Stubbemann

Eingereicht am 27.04.2023

Verteidigt am 30.10.2023

Declaration in accordance with § 8 of the General Provisions for Doctoral Degrees at the University of Kassel dated 14.07.2021.

1. I herewith give assurance that the submitted dissertation
   Orometry, Intrinsic Dimensionality and Learning:
   Novel Insights into Network Data
   is the product of my work alone.

2. I affirm that I have not used any sources or aids other than those explicitly cited. All passages that are drawn from published or unpublished writings, either word-for-word or in paraphrase, have been clearly identified as such.

3. The dissertation or parts of it

   ☒ has/have not been used in another doctoral or tenure process in Germany or abroad.

   ☐ has/have been submitted in another doctoral or tenure process in Germany or abroad:
   Title: -
   University and year of submission: -
   Type of examination or qualification: -
   Published in: -
   The submission refers to the following part(s) of the dissertation: -

4. Both, the submitted digital version and the submitted written copies are identical.

5. Third parties were not involved in the drafting of the content of this dissertation; most specifically I did not employ the assistance of a commercial dissertation advisor.

6. In case of a cumulative dissertation: I have documented the participation of co-authors by attaching a written statement signed by all of the co-authors outlining which parts of jointly written contributions are attributed to which co-authors.

7. I confirm the correctness of the above mentioned statements.

Date　　　　　　Signature

Erklärung gemäß § 8 der Allgemeinen Bestimmungen für Promotionen der Universität Kassel vom 14.07.2021.

1. Bei der eingereichten Dissertation zu dem Thema
   Orometry, Intrinsic Dimensionality and Learning:
   Novel Insights into Network Data
   handelt es sich um meine eigenständig erbrachte Leistung.

2. Anderer als der von mir angegebenen Quellen und Hilfsmittel habe ich mich nicht bedient. Insbesondere habe ich wörtlich oder sinngemäß aus anderen veröffentlichten oder unveröffentlichten Werken übernommene Inhalte als solche kenntlich gemacht.

3. Die Dissertation oder Teile davon habe ich

   ☒ bislang nicht an einer Hochschule des In- oder Auslands als Bestandteil einer Prüfungs- oder Qualifikationsleistung vorgelegt.

   ☐ wie folgt an einer Hochschule des In- oder Auslands als Bestandteil einer Prüfungs- oder Qualifikationsleistung vorgelegt:
   Titel der Arbeit: -
   Hochschule und Jahr: -
   Art der Prüfungs- oder Qualifikationsleistung: -
   Veröffentlicht in: -
   Es handelt sich dabei um folgenden Teil der Dissertation: -

4. Die abgegebenen digitalen Versionen stimmen mit den abgegebenen schriftlichen Versionen überein.

5. Ich habe mich keiner unzulässigen Hilfe Dritter bedient und insbesondere die Hilfe einer kommerziellen Promotionsberatung nicht in Anspruch genommen.

6. Im Fall einer kumulativen Dissertation: Die Mitwirkung von Koautoren habe ich durch eine von diesen unterschriebene Erklärung dokumentiert. Eine Übersicht, in der die einzelnen Beiträge nach Ko-Autoren und deren Anteil aufgeführt sind, füge ich anbei.

7. Die Richtigkeit der vorstehenden Erklärungen bestätige ich.

Datum Unterschrift

Acknowledgements

First of all, I would like to thank Gerd Stumme for the mentorship throughout this journey. The amount of self-determination, freedom and support I experienced during my time as a PhD student was exceptionally high and contributed to the enjoyment of this very special time. I had the opportunity to work on everything I was interested in and always got the help and guidance needed. A special thank goes out to Tom Hanika, who taught me so many things about how to write a publication and how to do research.

I am grateful for all the people of the Knowledge and Data Engineering Group at the University of Kassel who crossed paths with me. This includes Johannes Hirth, Tobias Hille, Maren Koyda, Maximilian Felde, Bastian Schäfermeier and Andreas Schmidt. I especially appreciate the time I spent with my office partner, Dominik Dürrschnabel, who shared hours of laughs and discussions about life in general and science in particular with me.

As my time as a PhD comes to an end, I can say with confidence that the creation of this work would have taken fundamentally longer without Monika Vopicka and Björn Fries. Monika organized and took care of so many things around the research itself. Björn managed our growing collection of servers and GPUs in a way that allowed me to focus solely on my research and experiments. Whenever I needed something, it was available in a (often ridiculously) short amount of time. Last but not least, I thank my family for always supporting me during this time and always being interested in and proud of the things I do and achieve.

To Lena and Charlotte,
who brought joy into my life to an extent I could not have envisioned before.

Abstract

Today, networks are an integral part of our world. Let it be real-life friendship networks or social connections that are based on social media. In this thesis, we contribute to the understanding of networks by studying networks from three different perspectives. **First, we adapt notions and concepts from orometry to metric data and networks to gain novel insights from a *local perspective*.** Specifically, we study measures of local outstandingness and propose concepts to derive small hierarchies from larger networks. These hierarchies are originally designed for the sake of defining dominance relationships between mountain peaks. Our adaption allows to identify outstanding entities on a local level and small hierarchies between them. **Second, we evaluate networks from a *global perspective* by computing the intrinsic dimensionality of whole networks.** Here, a low intrinsic dimensionality stands for data with highly distinguishable data points, which is crucial for learning. To accomplish this, we develop practical algorithms and speed-up techniques to transfer an axiomatically grounded framework to large-scale graph data. Furthermore, as an application, we present a feature selection method based on the developed method for computing intrinsic dimensions. **Third, we propose two novel deep learning methods for representation learning on networks, leading to *condensed perspectives* on them.** The first method learns embeddings with the help of techniques from formal concept analysis. This approach leads to a novel paradigm for embedding learning for bipartite graphs as it does not incorporate simple neighborhood information but the concept lattice structure of the corresponding formal context. The second method is a combination of a graph neural network and a language model and is tailored for a special network structure and the special task of author verification. This task deals with the verification of links between authors and publications. Our method is designed such that it can process raw texts and also incorporates past co-authorship edges.

In conclusion, this thesis contributes to the understanding and investigation of networks from a local, global, and condensed perspective. This is done by proposing novel measures and structures for them based on orometric concepts and intrinsic dimensionality and by providing novel learning methods for bipartite networks in general and author-publication networks in specific.

Zusammenfassung

Netzwerke sind integraler Bestandteil unseres Lebens, seien es echte Freundschaftsnetzwerke oder Verbindungen, welche auf sozialen Medien beruhen. In dieser Arbeit tragen wir dazu bei, solche Netzwerke besser zu verstehen. Dafür studieren wir Netzwerke aus drei Perspektiven:

Erstens, adaptieren wir orometrische Konzepte auf metrische Daten und Netzwerke um diese Daten aus einer *globalen Perspektive* zu betrachten. Genauer gesagt untersuchen wir Maße für lokale Herausragendheit und stellen Konzepte vor, welche es ermöglichen Hierarchien von kleinen Teilmengen aus größeren Netzwerken abzuleiten. Diese Hierarchien sind ursprünglich dafür gedacht um Dominanzbeziehungen zwischen Berggipfeln darzustellen. Unsere Adaption auf Netzwerke führt zu überschaubaren Hierarchien zwischen herausragenden Akteuren und ist somit ein neuartiges Werkzeug im Rahmen der Netzwerkanalyse. **Zweitens evaluieren wir die intrinsische Dimensionalität eines gesamten Netzwerkes um neue Einblicke aus einer *globalen Perspektive* zu erhalten.** Dafür entwickeln wir Algorithmen und Effizienztechniken um ein axiomatisch fundiertes Framework für große Echtweltgraphen anwendbar zu machen. Zusätzlich erläutern wir eine abgeleitete Methode zur Featureselektion, welche sich als kompetitiv zu repräsentativen Baselines erweist. **Drittens stellen wir zwei neuartige Deep-Learning Methoden für das Repräsentationslernen auf Netzwerken vor, was zu *kondensierten Perspektiven* führt.** Die erste Methode lernt Einbettung mit der Hilfe des Begriffsverbandes eines formalen Kontextes. Dieses Vorgehen begründet ein neuartiges Paradigma zum Lernen von Einbettungen auf bipartiten Graphen, welches nicht auf Nachbarschaften, sondern auf Verbandsstrukturen beruht. Die zweite Methode kombiniert ein Graph Neuronales Netz mit einem Sprachmodell. Diese Methode wurde für die Verifikation von Autorenschaften entwickelt. Sie erlaubt, Textdaten und vergangene Ko-Autorenschaften in den Klassifikationsprozess einzubringen.

Zusammenfassend leistet diese Arbeit einen Beitrag zur Untersuchung und Verarbeitung von Netzwerken durch die Einführung neuer Maße und Strukturen basierend auf orometrischen Konzepten und der intrinsischen Dimensionalität. Außerdem stellen wir neuartige Lernmethoden für spezielle Klassen von Netzwerken vor.

Table of Contents

Part I

Foundations

Chapter 1

New Perspectives on Network Data

This thesis focuses on the investigation of network data. Our overall aim is to provide novel quantities, structures and learning methods for such data. We start this thesis by motivating our goal of studying networks from different perspectives. Furthermore, we discuss our contributions to these perspectives.

1.1 Motivation

Networks are all around us. Let it be co-purchasing networks of products on Amazon or networks in academic environments, such as co-author networks of academic authors and citation networks of publications. The systematic study of network structures is known as *social network analysis*, an established topic with a tremendous amount of ongoing research. The extensive interest in this topic has consequently led to a variety of text books [117, 153, 135] that deal with its fundamentals.

In this thesis, we contribute to the understanding of networks by presenting novel notions that provide insides into them. To achieve this, we study networks from three perspectives.

The first perspective that we study is the *local perspective* to which we contribute with the introduction of novel notions and structures. To be more detailed, we identify crucial connections between nodes that are outstanding on a local level, i.e., with respect to their surrounding. For this, we adapt concepts from *orometry*. Orometry is a sub-discipline of geography and deals with the measurement, evaluation and categorization of mountains. In this realm, a variety of notions were established to identify locally outstanding mountain peaks and hierarchic connections between them. In this thesis, we adapt these mountain measures and structures from orometry to metric and network data.

Recently, more and more research has focused on *machine learning on graphs*, motivated by the fact that classification tasks regularly arise in the domain of networks. Examples are the prediction of subject areas of publications in citation networks or category predictions in co-purchasing networks of products. Furthermore, the important task of *link prediction*, i.e., predicting whether there will be or is an edge between two nodes, can be interpreted as a binary classification task on node pairs. When it comes to learning on real-world data, a phenomenon that is regularly discussed is the *curse of dimensionality*. This term is often used for a collection of obstacles that arise in high-dimensional learning [16, 111, 27]. In Pestov [122], it is defined as the phenomenon of (some) features concentrating at specific regions. Thus, different data points can not be discriminated and the data is considered to be of a high *intrinsic dimensionality (ID)*. While intrinsic dimensionality in general is widely investigated, it is an open problem how to measure intrinsic dimensionality in networks and how this notion would relate to the success of learning in such structures. We tackle this problem with an adaption which leads to new insights about the complexity of different networks. The notion of intrinsic dimensionality allows us to quantify the complexity of a whole network and thus creates novel insights from a *global perspective*, which is the second perspective studied in this work.

Our contribution to network analysis will not only consist of defining novel structures and quantities to investigate them. We also provide novel learning for architectures them. This leads to the incorporation of network structure into real-valued weights of neural networks. This learning of representations for complex structures allows us to study networks from a *condensed* perspective.

Our first architecture will be suited to the class of bipartite graphs which occur in a variety of settings, such as the above mentioned author-publication graphs and purchasing graphs between customers and products. In such circumstances, the graphs correspond to *formal contexts*, one of the main structures of interest in *Formal Concept Analysis* (FCA) [58]. We use this connection to introduce a completely novel learning approach that allows to embed nodes from bipartite networks by using the *concept lattice* of the corresponding formal context. This route has not been followed before.

Afterwards, we consider the specific classification task of *author verification*. Here, the task is to verify links in a bipartite author-publication network. We contribute to this specific direction of research with a novel architecture that combines *graph neural networks* (GNNs) with *language models* (LMs). Our architecture is especially suited for verification tasks where additional edges between authors are available.

1.2 Detailed Contribution

We contribute to the understanding of networks and learning on them by following the routes alluded in Section 1.1. The contribution of this thesis can be formulated as follows.

We provide novel insights into network data from different perspectives by transferring orometric concepts and notions of intrinsic dimensionality to them and by proposing learning architectures for representation learning on specific types of networks.

As mentioned, this results into insights from a global, a local and a condensed perspective. Our detailed contribution to the three perspectives are as follows.

We transfer orometric concepts to metric data and networks for local insights. In Part II, we continue the transfer of *(topographic) prominence* and *isolation* from the realm of orometry. We build on recent work [133] which transferred them to graphs. In this thesis, we establish these measures to bounded metric spaces with the help of graph structures that are derived from pairwise distance information. Lemma 3.12 shows that this is a natural generalization of the prominence term coined by Schmidt and Stumme [133].

Furthermore, we close the bridge between network analysis and orometry by defining the abstract structure of *landscapes*, i.e., graphs with height and distance information. For this structure, we are able to establish additional concepts from the realm of mountain peaks. This leads us to the *line parent hierarchy* of a network, which displays dominance relationships between locally outstanding nodes. This line parent hierarchy is a derived structure which leads to insights into networks that have not been present before.

We study the intrinsic dimensionality of network data for global insights. In Part III, we bridge the gap between learning on networks and the concept of intrinsic dimensionality by defining notions of intrinsic dimensionality for graph datasets and by linking them to the success of classification performances of GNNs. In order to do so, we take the mathematical foundation from Hanika et al. [70] for *geometric datasets* to give an explicit formula for the intrinsic dimensionality of finite data. To establish intrinsic dimensionality as a tool for network analysis, we propose speed-up techniques that allow to exactly compute the intrinsic dimensionality of networks with millions of nodes. Furthermore, by using novel approximation methods, we show that we can approximate the ID of **ogbn-papers100M**, a citation network with over 100 millions of nodes and over a billion of edges with an accuracy of over 99.9%. We additionally develop feature selection methods based on our notion of intrinsic dimensionality. We show that features that lower the ID are crucial for learning success, both for Euclidean and network data.

We develop novel learning procedures for networks for condensed insights. In Part IV, our contribution is two-fold. First, we propose a novel learning approach for bipartite networks. This is done by using the *formal concepts* of the formal context which correspond to the bipartite graph. Our approach adapts word2vec [107, 108], which is originally designed for word embeddings. More specifically, we train a neural network on predicting for an object or attribute the objects or attributes it shares formal concepts with. This leads to the derivation of an embedding procedure which is not based on simple neighborhood aggregation, but on the lattice of concepts.

Second, we investigate a more concrete network type from a specific field of application. This concrete application is *author verification*, i.e., the task of verifying if a specific document is written by a specific author. To tackle this problem, we propose LG4AV, a novel architecture that combines a graph neural network with a language model. LG4AV is especially tailored for the application in scientific domains, i.e., authorship verification on research papers. Here, we work with a specific network structure, a bipartite graph of publications and their authors and, additionally, a co-author structure solely living on the authors. LG4AV processes raw text information via a language model and co-author edges by doing multiple rounds of neighborhood aggregation at the input layer.

To sum up, we contribute to the understanding of networks and learning on them by adapting orometric concepts to them, by computing and approximating intrinsic dimensionality on them, and by proposing novel learning procedures for special forms of networks. The following publications are incorporated into this thesis.

Chapter 3: [140] Stubbemann, M., Hanika, T., and Stumme, G. (2020). Orometric methods in bounded metric data. In Berthold, M. R., Feelders, A., and Krempl, G., editors, *Advances in Intelligent Data Analysis XVIII - 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27-29, 2020, Proceedings*, volume 12080 of *Lecture Notes in Computer Science*, pages 496–508. Springer

Chapter 4: [143] Stubbemann, M. and Stumme, G. (2023). The mont blanc of twitter: Identifying hierarchies of outstanding peaks in social networks. In *Machine Learning and Knowledge Discovery in Databases: Research Track - European Conference, ECML PKDD 2023, Turin, Italy, September 18-22, 2023, Proceedings, Part III*, volume 14171 of *Lecture Notes in Computer Science*, pages 177–192. Springer

Chapter 5: [139] Stubbemann, M., Hanika, T., and Schneider, F. M. (2023a). Intrinsic dimension for large-scale geometric learning. *Transactions on Machine Learning Research*

Chapter 6: [141] Stubbemann, M., Hille, T., and Hanika, T. (2023b). Selecting features by their resilience to the curse of dimensionality. *CoRR*, abs/2304.02455

Chapter 7: [43] Dürrschnabel, D., Hanika, T., and Stubbemann, M. (2022). FCA2VEC: Embedding techniques for formal concept analysis. In Missaoui, R., Kwuida, L., and Abdessalem, T., editors, *Complex Data Analytics with Formal Concept Analysis*, pages 47–74. Springer International Publishing

Chapter 8: [142] Stubbemann, M. and Stumme, G. (2022). LG4AV: Combining language models and graph neural networks for author verification. In Bouadi, T., Fromont, É., and Hüllermeier, E., editors, *Advances in Intelligent Data Analysis XX - 20th International Symposium on Intelligent Data Analysis, IDA 2022, Rennes, France, April 20-22, 2022, Proceedings*, volume 13205 of *Lecture Notes in Computer Science*, pages 315–326. Springer

The paper Dürrschnabel et al. [43] is a joint work with Dominik Dürrschnabel and Tom Hanika. The author of this thesis first-authored all content parts which are contained in this thesis and designed and conducted all experiments which are introduced. Futhermore, the author of this thesis first-authored all other publications above and designed and conducted all experiments within them.

Chapter 2

Methodical Foundations

Our methodical foundations are split in two parts. We start by introducing a conceptualization of networks. Afterwards, we introduce the important notions and concepts that are used throughout this thesis.

2.1 The Network Model

The social phenomena which underlie different kinds of networks are of varying nature and complexity. Network science is therefore connected to a variety of disciplines, such as sociology and philosophy. To study networks from the perspective of mathematics and computer science, a mathematical conceptualization is needed. In Brandes et al. [23], the authors aim to give a conceptualization of the different levels of network analysis. The conceptualization consists of three stages:

1. A general *phenomenon...*

2. from which a *network concept* is derived via *abstraction...*

3. ... from which concrete *network data* can generated via a *representation*.

An example given in Brandes et al. [23] are *friendships*. The general phenomenon friendship is abstracted to *friendship networks*. Concrete network data of a specific group of people, for example of a specific school class, can be represented by a mapping from the pairs of pupils to, for example, $[0,1]$, which represents the intense of friendship.

In Brandes [21], networks are defined as mappings $x\colon \mathscr{D} \to \mathscr{W}$ where $\mathscr{D} \subseteq \mathscr{N} \times \mathscr{A}$ is a set of *dyads* between *nodes* $n \in \mathscr{N}$ and affiliations $a \in \mathscr{A}$ and $\mathscr{W}$ is a set of *values*. If $\mathscr{N} = \mathscr{A}$, $\mathscr{D}$ is called an *interaction domain* and x an *one-node network*. if $\mathscr{N} \cap \mathscr{A} = \emptyset$, then

$\mathscr{D}$ is called an *affiliation domain* and x a *two-mode network*. An example of a one-node network could be given by $\mathscr{N}$ being a set of academic authors and x mapping paris of authors to the Jaccard-index of their set of publications, i.e., the fraction of their common publications and the publications that at minimum one of them has co-authored. An example of a two-mode network could be given by $\mathscr{N}$ being a set of authors, $\mathscr{A}$, a set of publications and $x : \mathscr{D} \to \{0,1\}$ indicating for (n,a) whether n co-authored a.

To sum up, networks map dyads of entities to values. If $\mathscr{D} = \mathscr{N} \times \mathscr{N}$ and $\mathscr{W} = \{0,1\}$, networks correspond to *directed, unweighted graphs*. If $\mathscr{W} \subseteq \mathbb{R}_{\geq 0}$, we can interpret networks as *directed, weighted graphs*. In both cases, we interpret $x(m,n) = 0$ as the absence of an edge between m and n. From this point of view, the above definition contains graphs with edge attributes, i.e., graphs where edges have potential real-valued attribute vectors. Furthermore, if we interpret each node $n \in \mathscr{N}$ with the dyad (n,n), the network definition also allows attribute vectors for nodes of graphs.

In this thesis, we will investigate two structures that are related to the network definition above. The first one are *graph datasets*, i.e., graphs where each node is associated to an additional real-valued attribute vector. The second one are *landscapes*, i.e., graphs where nodes are equipped with a non-negative "height" and where we can measure pairwise distances between nodes via an external metric.

In contrast to the above strict definition of networks, other authors often do not make a clear distinction between networks and graphs. In this perspective, a graph is the mathematical structure to model real-world networks. Close to this, Brandes and Erlebach [22] refers to network as the informal concept of objects and interactions and connections between them while graphs are the abstract mathematical objects to formalize them.

In the area of doing machine learning on such structures, the terms graph and networks are often used interchangeable, dropping the distinction above. For the rest of this thesis, we will follow this convention.

2.2 Graphs

As mentioned above, the mathematical structure to represent networks are graphs, which we define in the following. For a detailed introduction into graph theory, we refer to Diestel [41].

Definition 2.1 (Graphs and Neighbors)
A *graph* is a tuple $G = (V,E)$ with V being a set and $E \subseteq \binom{V}{2}$. We call V the *nodes* or *vertices* of G and E the *edges* of G. For each node $v \in V$, we call $N_G(v) := \{v_1 \in V \mid \{v, v_1\} \in E\}$ the

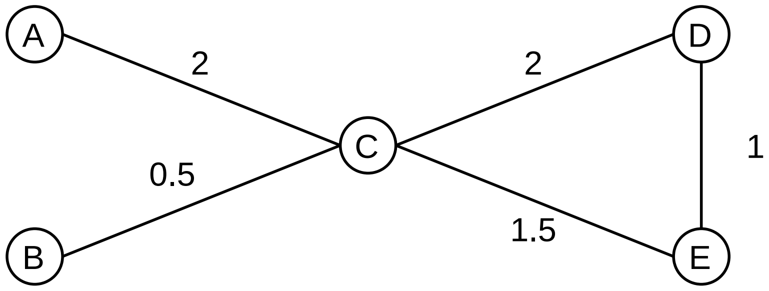


Figure 2.1: Example of a weighted graph. Circles denote nodes, lines edges. Numbers next to edges denote edge weights.

neighbors of v. Furthermore, we call $\deg_G(v) := |N_G(v)|$ the degree of v. If G can be omitted without confusion, we simply write $\deg(v)$ and $N(v)$.

Note that according to our definition, graphs are *undirected* and we do not allow for self-loops as for all $v \in V$ it holds that $\{v,v\} = \{v\} \notin \binom{V}{2}$. In contrast, to get *directed graphs*, we would require that $E \subseteq V \times V$. In real-world networks, the importance of different edges may vary. In co-author graphs for example, some edges may correspond to pairs of authors which cooperate often and thus are strongly connected while other edges may represent rare co-authorships between authors with a low connection. Such information can be incorporated into the graph via edge weights.

Definition 2.2 (Weighted Graphs)
Let $H = (V,E)$ be a graph. We call a function $w : E \to \mathbb{R}_{>0}$ a *weighting function* of H. We call $w(E)$ the *weight* of w. We call the triple $G = (V,E,w)$ a *weighted* graph.

A sketch of a weighted graph is given in Figure 2.1. Often, one considers only subsets of the nodeset with the corresponding edge sets. These smaller graphs are called subgraphs.

Definition 2.3 (Subgraphs and Supergraphs)
Let $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$. If $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$, we call G_1 a *subgraph* of G_2 and G_2 a *supergraph* of G_1. If $E_1 = E_2 \cap \binom{V_1}{2}$, we call G_1 an *induced subgraph* of G_2. In this case, we write $G_2[V_1] := G_1$.

A widely investigated class of networks are connections between (academic) authors and publications. A graph structure arises by the question which author has written which publication. The resulting graph has a special structure as the node set can be separated into two subsets (the authors and the publications) such that there exists no edge within one of these two subsets. Such graphs are regularly considered in the realm of network analysis. They are called bipartite graphs.

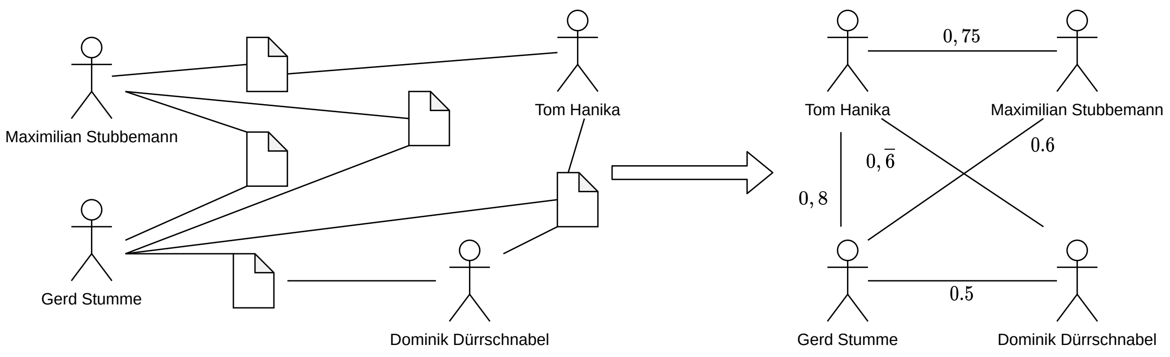


Figure 2.2: A co-author graph as a projection from an author-publicationsgraph. The edges are weighted via Jaccard distances.

Definition 2.4
Let $B = (V,E)$ be a graph with $E \neq \emptyset$. Let $V_1, V_2 \subseteq V$ be two subsets with $V_1 \cap V_1 = \emptyset$ and $V_1 \cup V_2 = V$ such that for all $e \in E$:

$$e \cap V_1 \neq \emptyset \neq e \cap V_2.$$

We then call B *bipartite* and we call $\{V_1, V_2\}$ a *bipartition* of B.

Following the convention to not distinguish between graphs and networks, we will additionally speak of *bipartite networks*. (Weighted) graphs often can be derived from bipartite graphs by considering one set of a bipartition and connect them if they have common neighbors in the original graph.

Definition 2.5
Let $B = (V,E)$ be a bipartite graph with a bipartition $\{V_1, V_2\}$. We call the graph $G = (V_1, E_1)$ with $E_1 := \{\{v_1, v_2\} \mid N_G(v_1) \cap N_G(v_2) \neq \emptyset\}$ the *projection of B on* V_1. The *Jaccard-similarity weighting function* is given via

$$w_{\text{JCS},V_1} : E_1 \to \mathbb{R}_{\geq 0}, \{v_1, v_2\} \mapsto \frac{|N_B(v_1) \cap N_B(v_2)|}{|N_B(v_1) \cup N_B(v_2)|},$$

If $w(u,v) < 1$ for all $u \neq v \in V$, the *Jaccard-distance weighting function* is given via the formula $w_{\text{JCD},V_1} := 1 - w_{\text{JCS},V_1}$.

Examples of graphs that arise as as projections are for example given by co-occurrences of movie actors or co-author graphs which are projections on the authors from the above mentioned bipartite author-publicationgraphs. A small sketch of such a projection with edges weighted via Jaccard-distance weighting is given by Figure 2.2.

In this thesis, especially to study orometric concepts in Part II, we will need graphs as a structure to “traverse” through a node set. Furthermore, we will often need the information, how far away different nodes of a graphs are. For both of these, we will need the concepts of walks and paths in graphs.

Definition 2.6 (Walks and Paths)
Let $G = (V.E)$ be a graph. A sequence $p = (v_i)_{i=0}^{n}$ with $n \in \mathbb{N}_0$ of V is called a *walk* if for all $i \in \{1,\ldots,n\}$ it holds that $\{v_{i-1}, v_i\} \in E$. We call a walk a *path* if for all $i \neq j \in \{1,\ldots,n\}$ it holds that $v_i \neq v_j$ or $\{i, j\} = \{1,n\}$. We call $\operatorname{start}(p) := v_0$ the *starting point* of p and $\operatorname{end}(p) := v_n$ the *end point* of p.

Note, that this definition allows walks of length 0 which consist only of one node. Often, we will assume that our graph is connected, i.e., that we can reach every node from each other node via walks. If not, we will often just work with the largest set of nodes that fulfills this property.

Definition 2.7 (Connectivity and Connected Components)
Let $G = (V,E)$ be a graph. We call G *connected* if for all $v_1, v_2 \in V$ there exist a walk between v_1 and v_2. If $V_1 \subseteq V$ such that $G[V_1]$ is connected and that for all $V_2 \supsetneq V_1$ the graph $G[V_2]$ is not connected, we call $G[V_1]$ a *connected component* of G. We call the connected components with the largest node sets the *biggest connected components* of G.

2.3 Metric Spaces and Relative Neighborhood Graphs

The other data structure we will regularly use in this work is the metric space.

Definition 2.8 (Metric Space and Boundedness)
A tuple (M,d) consisting of a set M and a function $d : M \times M \to \mathbb{R}_{\geq 0}$ is called a *metric space* if

- $\forall x,y \in M : d(x,y) = 0 \Leftrightarrow x = y$, called *positivity*,
- $\forall x,y \in M : d(x,y) = d(y,x)$, called *symmetry* and
- $\forall x,y,z \in M : d(x,z) = d(x,y) + d(y,z)$, called *triangle inequality*.

We call d a *metric* or *distance (function)* on M. If furthermore

$$\sup_{x,y \in M} d(x,y) < \infty,$$

we call M *bounded*. If clear from context, we may omit d and call M a metric space.

Metric data is fundamental to data science, as a large amount of datasets has a form which allows to compute distances between different data points. Often, the data points are elements of $\mathbb{R}^d$. In such cases distances are regularly measured via l^n- metrics, which are given via

$$l^n \colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}, (x,y) \mapsto \sqrt[n]{\sum_{k=1}^{d} |x_k - y_k|^n}$$

For $n = 1$, we speak of the *Manhattan metric* and $n = 2$ gives us the standard *Euclidean metric*.

Definition 2.9 (Shortest Path Distance)
Let $G = (V,E,w)$ be a finite, weighted and connected graph. Let $p = (v_i)_{i=0}^n$ be a path. The *length* of the path p is defined via $l(p) := \sum_{i=1}^n w(\{v_{i-1}, v_i\})$. For $v_1, v_2 \in V$ we define the shortest path distance via

$$d_{\mathrm{SP}}(v_1,v_2) = \begin{cases} 0 & v_1 = v_2 \\ \min\{l(p) \mid p \text{ path from } v_1 \text{ to } v_2\} & v_1 \neq v_2. \end{cases}$$

For unweighted graphs $G = (V,E)$, the shortest path is defined via the shortest path distance of the weighted graph with constant edge weights of 1.

Note, that in the unweighted case, the length of a path is the amount of elements in the path except for the starting point. Using our definition of weighted graphs where we require that $w(e) > 0$ for all $e \in E$, it is common knowledge that the shortest path distance indeed defines a metric on V.

2.3.1 From Metric Spaces to Graphs

The shortest path metric allows us to measure distances in graphs. For some applications we will be interested in the opposite, i.e., in a possibility to generate a graph structure based on metric information. This will give us the ability to traverse from point to point through a metric space. A first approach is to connect each point to points that are close with respect to the given metric.

Definition 2.10 (Nearest Neighbor Graph and Fixed Radius Neighbor Graph)
Let (M,d) be a metric space. If the set M is finite, we define the *nearest neighbor graph*

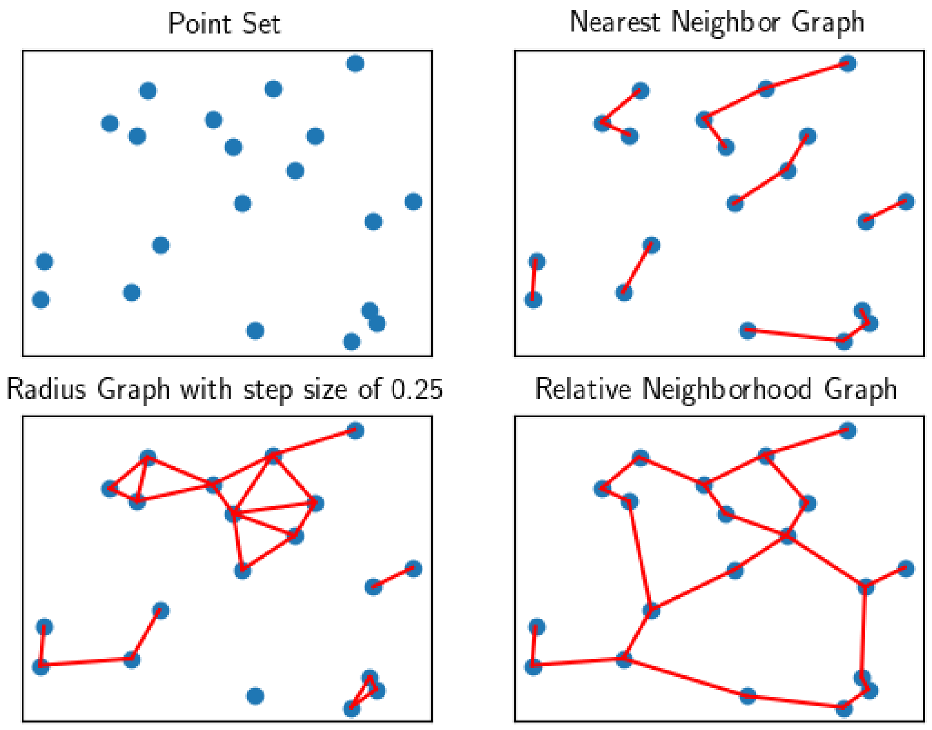

Point Set
Nearest Neighbor Graph
Radius Graph with step size of 0.25
Relative Neighborhood Graph

Figure 2.3: Neighborhood Graphs for randomly distributed points in $\mathbb{R}^2$ equipped with the euclidean metric.

$G_M = (M, E_M)$ via

$$E_M := \{\{m,n\} \in \binom{M}{2} \mid d(m,n) = \min_{n_1 \neq m} d(m,n_1)\}.$$

Let $\delta > 0$. We define the *δ-radius graph* or *δ-step graph*, denoted by $G_{M,\delta}$, as the tuple $(M, E_{M,\delta})$ via

$$E_{M,\delta} := \{\{m,n\} \in \binom{M}{2} \mid d(m,n) \leq \delta\}.$$

While these graphs are intuitive solutions for deriving a graph structure from a metric space, they have the shortcoming that the resulting graph may not be connected. Another approach to get a graph structure is given by connecting two points if there is no third point which is closer to both of them. As we will see, this will indeed result in a connected graph.

Definition 2.11 (Relative Neighborhood Graph)
Let (M,d) be a metric space. Then, we define the *relative neighborhood graph* (RNG) $\mathrm{RNG}(d) = (M, E_{\mathrm{RNG}(d)})$ via

$$\{m_1,m_2\} \in E_{\mathrm{RNG}(d)} \Leftrightarrow \nexists m_3 \in M : \max\{d(m_1,m_3), d(m_2,m_3)\} < d(m_1,m_2).$$

The RNG was first proposed by Toussaint [147] where it was also shown that for finite point sets in $\mathbb{R}^2$ the RNG is a supergraph of the minimum-spanning-tree which implies the connectivity [80]. Because RNGs are commonly only studied for points in $\mathbb{R}^d$ equipped with an l^p metric we could not find a proof for the connectivity of the RNG for arbitrary finite metric spaces. Hence, we prove it here.

Theorem 2.12 *Let (M,d) be a finite metric space. Then* $\mathrm{RNG}(d)$ *is connected.*

Proof. Assume that $\mathrm{RNG}(d)$ is not connected. As M is finite we can choose an unconnected pair $(m_1,m_2) \in M \times M$ such that $d(m_1,m_2)$ is minimal. Since m_1,m_2 are not connected they are especially not adjacent in $\mathrm{RNG}(d)$. Thus, we find $m_3 \in M$ such that $d(m_1,m_3) < d(m_1,m_2)$ and $d(m_2,m_3) < d(m_1,m_2)$. As m_1,m_2 are chosen as the pair with the minimal distance among the unconnected pairs, the pairs (m_1,m_3) and (m_2,m_3) are both connected. Hence, we would have a connection from m_1 to m_2. $\square$

A sketch of deriving neighbor graphs and radius graphs from a metric space is given in Figure 2.3.

2.4 Formal Concept Analysis

Bipartite graphs can be interpreted as a relation between the two distinct partition classes. This leads to a natural connection between network analysis and *Formal Concept Analysis* (FCA) [58] that we will use in Chapter 7 to generate condensed views for bipartite networks. In the following, we recall basic notions from FCA which are needed in this thesis. For a detailed introduction to the whole topic we refer the reader to Ganter and Wille [58]. For our small introduction, we will also need two concepts from order theory, namely *orders* and *lattices*. Their definitions are also taken from Ganter and Wille [58].

Definition 2.13 (Order)
A binary relation $\leq$ on a set M that fulfills for all $x,y,z \in M$ the following conditions:

- $x \leq x$, called *reflexivity*,

- if $x \leq y$ and $y \leq x$, then $x = y$, called *antisymmetry*,
- if $x \leq y$ and $y \leq z$, then $x \leq z$, called *transitivity*,

is called an *order on* M. We then call $(M, \leq)$ an ordered set.

To define lattices, we first need the notions of *infimum* and *supremum*.

Definition 2.14 (Infimum and Supremum)
Let $(M, \leq)$ be an ordered set. If for a subset $A \subseteq M$ there exist a unique largest element $m \in M$ (with respect to the order) such that $m \leq a$ for all $a \in A$, we call m the *infimum of* A. If there is a unique lowest element (with respect to the order) $m \in M$ such that $a \leq m$ for all $a \in A$, we call m the *supremum of* A.

Complete lattices are the orders where every infimum and every supremum exist.

Definition 2.15 (Complete Lattice)
We call an ordered set $(M, \leq)$ a *complete lattice* if the infimum and supremum of every $A \subseteq M$ exists.

We now have the few notions from order theory at hand that will be used in the following. A formal context can be defined as follows.

Definition 2.16 (Formal Context)
A *formal context* is a triple $\mathbb{K} = (G, M, I)$ with $I \subseteq G \times M$. We call G the *objects* of $\mathbb{K}$ and M the *attributes* of $\mathbb{K}$. If $(g, m) \in I$ we say "object g has attribute m". We call a set $A \subseteq G$ an *object set* of $\mathbb{K}$ and a set $B \subseteq M$ an *attribute set* of $\mathbb{K}$.

Formal contexts can be graphically represented by *cross-tables*. An example can be found in Figure 2.4. The key operator in FCA is the *derivation* which maps objects (attributes) to the attributes (objects) which they are in relation with.

Definition 2.17 (Derivation Operators)
Let $\mathbb{K} = (G, M, I)$ be a formal context. The *object derivation* is given by the map

$$\cdot' \colon P(G) \to P(M), A \mapsto A' := \{m \in M \mid \forall g \in A : (g, m) \in I\}$$

and the *attribute derivation* via

$$\cdot' \colon P(M) \to P(G), B \mapsto B' := \{g \in G \mid \forall m \in B : (g, m) \in I\}.$$

Note, that we use the same notation for the object and attribute derivation. For S', whether the object or attribute derivation is meant can be concluded from the fact whether S is a set of objects or attributes. For the sake of simplicity, we use for an object $g \in G$ the notation g' instead of $\{g\}'$ and talk about the derivation of an object. The same holds for attributes $a \in A$. Furthermore, if S is an object set or an attribute set, we call the set S'' the *closure* of S.

The key entities of investigation in FCA are pairs of objects A and attributes B where B is the set of attributes that each object in A has and A is the set of objects to which all attributes of A are incident.

Definition 2.18 (Formal Concepts, Extents and Intents)
A *formal concept* of a formal context $\mathbb{K} = (G, M, I)$ is a pair $C = (A, B) \in P(G) \times P(M)$ such that

$$A' = B \wedge B' = A.$$

We denote by $\mathfrak{B}(\mathbb{K})$ the set of all formal concepts of $\mathbb{K}$. For a formal concept $C = (A, B)$, we call A the *extent* of C and we call B its *intent*.

Definition 2.19 (Superconcepts, Subconcepts and Concept Order)
Let $(A, B), (C, D)$ be formal concepts of a formal context $\mathbb{K} = (G, M, I)$. We define a partial order on the set of all formal concepts of $\mathbb{K}$ via

$$(A, B) \leq (C, D) \Leftrightarrow A \subseteq C. \tag{2.1}$$

If Equation (2.1) holds, we call (A, B) a *subconcept* of (C, D) and (C, D) a *superconcept* of (A, B). We call

$$\underline{\mathfrak{B}}(\mathbb{K}) := (\mathfrak{B}(\mathbb{K}), \leq)$$

the *concept lattice* of $\mathbb{K}$.

The structure $\underline{\mathfrak{B}}(\mathbb{K})$ is, as its name suggests, a lattice.

Theorem 2.20 (Part 1 of the Basic Theorem on Concept Lattices) *[58, Theorem 3, Part 1]*
The ordered set $\underline{\mathfrak{B}}(\mathbb{K})$ is a complete lattice and for a set $\mathscr{C} = \{(A_i, B_i) \mid i \in I\} \subseteq \mathfrak{B}(\mathbb{K})$ it holds that

$$\sup(\mathscr{C}) = \left((\bigcup_{i \in I} A_i)'', \bigcap_{i \in I} B_i \right),$$

$$\inf(\mathscr{C}) = \left(\bigcap_{i \in I} A_i, (\bigcup_{i \in I} B_i)'' \right).$$

	P_1	P_2	P_3	P_4	P_5
Maximilian Stubbemann	×	×		×	
Gerd Stumme		×	×	×	×
Dominik Dürrschnabel			×		×
Tom Hanika	×				×

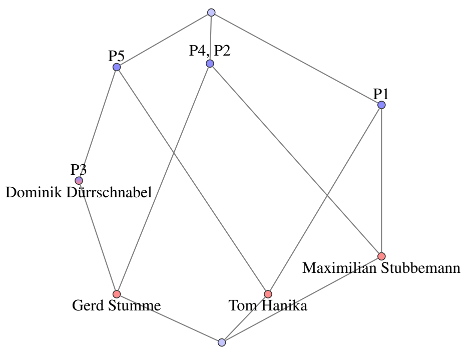


Figure 2.4: Left: The formal context that corresponds to the bipartite graph in Figure 2.2. Right: Its concept lattice. The lattice is can be read as follows: The nodes represent concepts. For each node, the objects that are at nodes below (including the node itself) are contained in the extent and all attributes at nodes above it (including the node itself) are contained in the intent.

An example of a formal context and the corresponding concept lattice can be found in Figure 2.4. The concept lattice is the central structure studied in FCA. The elements of the concept lattice, i.e., the formal concepts as defined in Definition 2.18 will be the key to generate the embeddings in Chapter 7.

2.5 Learning and Classification

A substantial amount of this work will be connected to the realm of machine learning. In Part III we study its connection to intrinsic dimensionality and Part IV is dedicated to the development of machine learning approaches for specific networks. Machine learning deals with the question of "*[..] how to construct computer programs that automatically improve with experience*", as stated by Mitchell [111, p. xv]. We will use the following formalization.

Definition 2.21 (Learning [111, p.2])
A computer program *learns* from *experience* E with respect to some class of *tasks* T and *performance measure* P, if its performance at tasks in T, as measured by P, improves with experience E.

In this thesis, the experience will be made through *data*, i.e., observations encoded as elements of a set. Here, the data which is used to learn will be called the *training data*. Today, learning procedures are separated by the question whether they learn with the help of additional ground truth information. To be more detailed, learning a function that maps inputs to some kind of output information is called *supervised learning* while learning patterns from input data without the supply of outputs is called *unsupervised learning* [131, p.650]. An example of supervised learning is given by the task of distinguishing pictures of cats and dogs. Identifying friendship communities in social networks can be seen as an unsupervised learning problem. Supervised learning methods are the ones which are more related to this thesis. The most common tasks in this realm are classification tasks. The following formalization is an adaption of the definition of classification from Ester and Sander [49].

Definition 2.22 (Supervised Learning, Classification and Regression)
Given an input domain $\mathbb{X}$, an output domain $\mathbb{Y}$, a set $\mathbb{X}_{\text{train}} \subseteq \mathbb{X}$ of training examples and a given function $\tilde{f}\colon \mathbb{X}_{\text{train}} \to \mathbb{Y}$, the goal of supervised learning is to infer from $\tilde{f}$ a function $f\colon \mathbb{X} \to \mathbb{Y}$ which maps each input $x \in \mathbb{X}$ to its output $f(x) \in \mathbb{Y}$. If $\mathbb{Y} = \mathbb{R}$, we speak of *regression learning* and if $|\mathbb{Y}| < \infty$, we speak of *classification learning*, call f a *classifier* and $f(x)$ the *class* or *label* of $x \in \mathbb{X}$. If $\mathbb{Y} = \{-1, 1\}$, we speak of a *binary classification problem*. We then call $x \in \mathbb{X}_{train}$ a *positive example* if $\tilde{f}(x) = 1$ and a *negative example* if not.

The standard approach is to give the classifier the pairs $(x, \tilde{f}(x))$, where x is a training example, at *training-time* and then predict the labels $f(x)$ for the elements x of a *test set* $\mathbb{X}_{\text{test}} \subseteq \mathbb{X} \setminus \mathbb{X}_{\text{train}}$ at the *testing-time*. If the test set is known at training time (without any label information!), we speak of *transductive learning*. If not, we speak of *inductive learning*.

There exist various notions to quantify the performance of classifiers. Probably the most common one is the *accuracy* which is defined of the ratio of the test examples which are classified correctly. In the case of binary classification, a multiple amount of additional quantities exist. They are built as combinations of the following four quantities:

- True Positives (TP): The amount of test examples which are correctly classified as positive example.
- True Negatives (TN): The amount of test examples which are correctly classified as negative examples.
- False Positives (FP): The amount of test examples which are wrongly classified as positive examples.
- False Negatives: (FN): The amount of test examples which are wrongly classified as negative examples.

From this quantities, the following performance measures are built:

- Precision: $\text{PR} = \frac{\text{TP}}{\text{TP}+\text{FP}}$
- Recall: $\text{RE} = \frac{\text{TP}}{\text{TP}+\text{FN}}$, also called accuracy on the positive examples, denoted with ACC+.

- F1-Score: $\mathrm{F1} = 2\frac{\mathrm{PR}\cdot\mathrm{RE}}{\mathrm{PR}+\mathrm{RE}}$
- Accuracy on the negative examples, $\text{ACC-} = \frac{\mathrm{TN}}{TN+\mathrm{FP}}$
- Geometric mean: $\mathrm{GM} = \sqrt{\text{ACC+}\cdot\text{ACC-}}$

Often, binary classifies do not output definitive decisions, but a probability that an example is positive. In this case, the above notions can be incorporated by using a threshold to assume that an example is positive. However, there are also performance measures, that incorporate the probabilities. One of them is the *AUC score*, also called *ROCAUC score*. This score is built by computing the ratio of true positives on all examples classified as positive (i.e., the precision, in this context also called the *true positive rate*) and the ratio of all wrongly as positive classified examples of all negative examples, called the *false positive rate*. These pairs of true positives and true negatives values are calculated for all possible thresholds $t \in [0,1]$. The area under the curve of the graph with the false positive rates on the x-axis and the true positive rates on the y-axis is then the AUC score. For a more detailed explanation we refer to Fawcett [51].

2.6 Neural Networks

A large extent of modern learning architectures are neural networks. They build the base for the processing of natural language [150, 40, 25] which will be part of the learning architecture in Chapter 8 and for learning from graphs as studied in network analysis [85, 68, 151].

The most basic and fundamental building block of neural networks is the *single-layer perceptron*, or simply *perceptron*, which consists of a sequence of *units* $(u_i)_{i=1}^{d_1}$ where $u_i\colon \mathbb{R}^{d_0} \to \mathbb{R}$ is of the form $(x_1,\dots,x_{d_0})^T \mapsto w_0^i + \sum_{l=1}^{d_0} w_l^i x_l$ where $w^i \in \mathbb{R}^{d_0+1}$ is called the *weight vector* or *parameter vector* of u_i. Let furthermore be $\sigma\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_1}$. The single layer perceptron then has the form

$$f\colon \mathbb{R}^{d_0} \to \mathbb{R}^{d_1}, x \mapsto \sigma\left((u_1(x),\dots,u_{d_1}(x))^T\right)$$

Here, σ is called the *activation function* of f. A plentitude of commonly used activation functions are derived from functions $\mathbb{R} \mapsto \mathbb{R}$ via coordinate-wise application. Commonly used activation functions include the following.

- **ReLU:** $\mathbb{R} \to \mathbb{R}, x \mapsto \begin{cases} x & x >= 0, \\ 0 & x < 0, \end{cases}$

- **Sigmoid:** $\mathbb{R} \to \mathbb{R}, x \mapsto \frac{\exp(x)}{1+\exp(x)}$,
- **Softmax:** $\mathbb{R}^d \to \mathbb{R}^d, x \mapsto \left(\frac{\exp(x_1)}{\sum_{i=1}^{d} \exp(x_i)}, \ldots, \frac{\exp(x_d)}{\sum_{i=1}^{d} \exp(x_i)} \right)^T$.

Note, that single-layer perceptrons can be represented as concatenations of affine-linear functions and activations. Let $W \in \mathbb{R}^{d_1 \times d_0}$ via $W_{i,j} := w^i_j$ for $i \in \{1,\ldots,d_1\}, j \in \{1,\ldots,d_0\}$ and $b \in \mathbb{R}^{d_1}$ via $b_i := w^i_0$. Then, we get

$$f(x) = \sigma(Wx+b).$$

This notation of column vectors being multiplied with matrices from the left is the common notation in linear algebra. However, we will follow a different notation that is more common in the realm of learning with neural networks. Here, it is usual to work with row-vectors that are multiplied from the right with the weight matrix. In this case, we set $W_{i,j} := w^j_i$, set b as above but assuming it to be a row-vector and let single-layer perceptrons map row- vectors x via

$$f(x) = \sigma(xW+b). \tag{2.2}$$

Multi-layer perceptrons are concatenations of multiple single-layer perceptrons. Hence we can define them via concatenations of affine-linear mappings and activations. This elegant view has been brought to the author of this thesis by Kipf [84].

Definition 2.23 (Multi-Layer Perceptron [84, p.9])
Let $\varphi_i \colon \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ be an affine-linear function and let $\sigma_i \colon \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ for all $i \in \{1,\ldots,n\}$. We then call

$$f := \sigma_n \circ \varphi_n \circ \cdots \circ \sigma_1 \circ \varphi_1 \tag{2.3}$$

a *multi-layer perceptron* (MLP) or, more specific, a n-layer perceptron.

The entries of all weight vectors of all layers are called the *weights* or the *parameters* of the network. Note, that there is a common inconsistency on how to count the number of layers. While in the above definition we count the amount of “maps” which are applied on an input, it is also common to identify the “states” an input takes while being inferred. This leads to the unintuitive denomination that a 2-layer network consist of an *input-layer*, one *hidden-layer* and an *output-layer*. This is a result of the view that MLPs propagate inputs by the units through different layers. A sketch of this view is depicted in Figure 2.5.

In standard euclidean settings, the data is represented via a matrix $X \in \mathbb{R}^{m \times d}$, where the rows X_i correspond to so called *feature vectors* or *attribute vectors* of the data points and X is also called the *feature matrix*, *attribute matrix* or *data matrix*. It is common to apply the

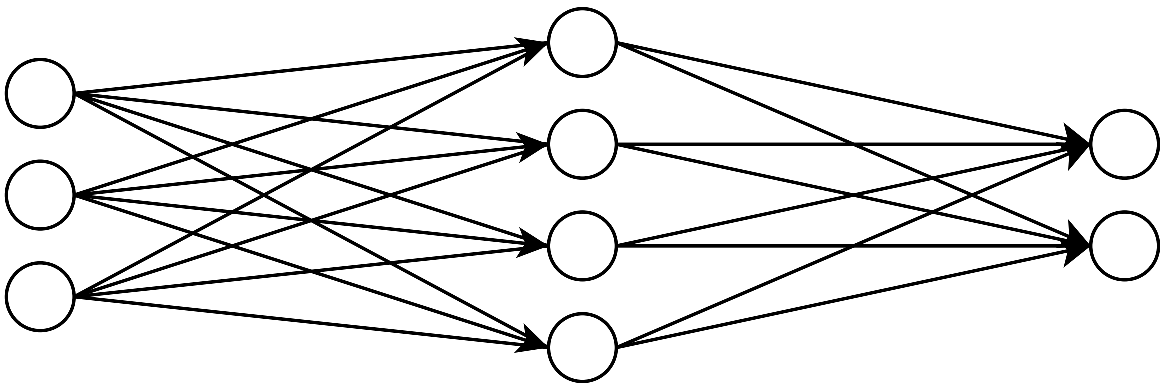


Figure 2.5: A 2-layer perceptron, consisting of an input-layer, one hidden-layer and the output layer. This net expects $3-$dimensional inputs. The first layer has 4 units, the second layer has 2 units.

MLP f to X by applying it to every row of X. If we have a perceptron as in Equation (2.2), we set $B \in \mathbb{R}^{m \times d_1}$ as the matrix which is b in every row and arrive at

$$f(X) = \sigma(XW + B), \tag{2.4}$$

where σ is applied row-wise. This rationale allows us to naturally also lift MLPs of the form Equation (2.3) from $\mathbb{R}^{d_0} \mapsto \mathbb{R}^{d_n}$ to maps $\mathbb{R}^{m \times d_0} \mapsto \mathbb{R}^{m \times d_n}$.

To classify with a neural network, the size of the output layer is set to the amount of classes and softmax is commonly chosen as the activation σ_n of the last layer. The coordinates of the output vector then represent with which probability the example belongs to the corresponding classes. In the binary case with one positive and one negative class, it is also common to have one unit at the last layer and a sigmoid activation as the last activation. The output value is then interpreted as the probability that the input is a positive example.

2.6.1 Training of Neural Networks

We have focused the above explanation on the structure of multi-layer perceptrons and how they map input vectors to output vectors when the weights are chosen. The question remains on how to optimize the weights with the help of the training data of the classification task. This is done with the help of a *loss function* which quantifies for a neural network f and each training example $(x, \tilde{f}(x))$ how far the network output $f(x)$ is from $\tilde{f}(x)$. For this, the training examples are split into small sets of *(mini-)batches*. The training examples of a mini-batches are then fed simultaneously to the network and the parameters of the neural networks are minimized such that the loss on the examples in the batch are minimized. We do not dive deeper into the optimization of neural networks. This topic is part of various fundamental text books, as for example Bishop [16]. In the following chapters, we will often explain how we choose specific parameters connected to the training of neural networks, such as *learning rates*, *optimizers* or *epochs*. These parameters are mentioned for the sake of transparency and reproducibility. The knowledge about the meaning of these terms is **not** crucial to understand this thesis.

2.7 Neural Networks for Texts and Graphs

Until now, we assumed the input of neural networks to be of Euclidean nature, i.e., we assume our data to be of the form $X \in \mathbb{R}^{m \times d}$. However, parts of this work will deal with data of different format. Specifically, we will work with neural networks which are trained on *text data* and with neural nets which are trained on *graph data*.

2.7.1 Neural Networks for Texts

Today, the standard architecture for learning from text data is the transformer architecture from Vaswani et al. [150]. Here, sequences of tokens are fed through the network at once and a so-called *attention mechanism* leads to the tokens "seeing" each other. Thus, the vectors of each token at the hidden and output layer depend on the tokens around it. For more details, we refer to the paper Vaswani et al. [150].

Recent models to process and generate texts are compositions of multiple transformer layers [40, 25, 158] and are pre-trained on large texts. Then, to solve a specific task, these models are further trained on suited training examples. This task-specific training step is called fine-tuning. For our own learning approaches we use BERT [40], which is probably the most known and used of these language models.

2.7.2 Graph Neural Networks

In this realm of graph learning it is common to work with data which consists of a graph structure and additional information, such as external distance information or feature vectors for all nodes. Dealing with the latter is an well-established but still growing research field. The developments have led to a specific class of neural networks, the so called *graph-neural networks* (GCNs). These neural networks work on the following kind of data, which consist of a graph and feature vectors for the nodes.

Definition 2.24 (Graph Dataset)
Let $X \in \mathbb{R}^{n\times d}$ be an attribute matrix. Let furthermore $G = (V,E)$ be a graph with $V = \{v_1,\dots,v_n\}$. We call $D = (G,X)$ a *graph dataset* and we call the row-vector $X_i = (X_{i,1},\dots,X_{i,d})$ the *attribute vector corresponding to v_i*.

Such datasets are commonly investigated in recent machine learning environments. Examples are co-citation networks where each paper is mapped to bag-of-words or embedding vectors [107, 108, 94], such as **ogbn-arxiv** or co-purchasing networks with feature-vectors generated from descriptions, such as **ogbn-products**. Both these datasets are from the *Open Graph Benchmark Node Property Prediction* (OGBN) suite [76].[1]

There is a plentitude of well-known and established architectures for GNNs [85, 68, 151]. Furthermore, a variety of more recent work has focused on scaling GNNs to millions of nodes [129, 144, 161]. One of the most fundamental and basic GNN is the *graph convolutional neural network* (GCN) from Kipf and Welling [85]. The GCN uses the normalized adjacency matrix, which is defined as follows.

Definition 2.25 ((Normalized) Adjacency Matrix)
For a graph $G = (V,E)$ with $V = \{v_1,\dots,v_n\}$, we call $A \in \mathbb{R}^{n\times n}$ with

$$A_{i,j} := \begin{cases} 1 & v_j \in N(v_i) \lor i = j, \\ 0 & \texttt{else} \end{cases}$$

the *adjacency matrix* of G. Let furthermore be $\Lambda \in \mathbb{R}^{n\times n}$ be the diagonal matrix with $\Lambda_{i,i} := \deg(v_i) + 1$. We then call

$$\hat{A} := \Lambda^{-\frac{1}{2}} A \Lambda^{-\frac{1}{2}}$$

the *normalized adjacency matrix* of G. In this context, $\Lambda^{-\frac{1}{2}}$ stands for applying $x \mapsto x^{\frac{1}{2}}$ element-wise to Λ^{-1}.

The GCN model of Kipf and Welling [85] proposes to use the normalized adjacency matrix to aggregate the representations of graph neighbors at every layer. This allows to not only use the hidden representation of a node itself but also of its neighbors for classification. To be more formal, Kipf and Welling [85] replaces layers of the form Equation (2.4) with

$$f(X) = \sigma(\hat{A}XW).$$

[1]https://ogb.stanford.edu/docs/nodeprop/

This model is not sufficiently scalable because of its problem of handling mini-batches. If one, for example, feeds a point through a 2-layer GCN one needs all neighbors of this points and all neighbors of these neighbors to be able to do neighbor aggregation at both layers. Hence, using the GCN will lead to large batch sizes. This problem is often referred to as the *neighbor explosion* or *neighborhood explosion problem* [160, 159, 54]. Kipf and Welling [85] therefore only considered full-batch training, i.e. only having one batch which contains all examples. However, this form of training is not feasible with large graphs with millions of nodes.

To overcome this problem, recent works have proposed to only do neighborhood aggregation at the input layer [129, 144, 161]. Here, multiple rounds of neighborhood aggregation are incorporated by computing $X, \hat{A}X, \dots, \hat{A}^k X$ and then feeding them together into a network without neighborhood aggregation at hidden layers. Since the aggregation is only done at the input layer, this can be done once before training and the network can be trained with usual mini-batching.

In the following, we focus on the SIGN model from Rossi et al. [129]. In detail, this model works as follows. For a fixed $k \in \mathbb{N}$, the aggregations $X, \hat{A}, \dots, \hat{A}^K X$ are computed. The full network then has the form

$$X \mapsto \sigma_2(\sigma_1([XW_0, \hat{A}XW_1, \dots, \hat{A}^k W_k])\hat{W}), \tag{2.5}$$

where $W_0, \dots, W_k, \hat{W}$ are learnable weight matrices, σ_1, σ_2 are activation functions and [.] stands for concatenation. If only a mini-batch should be feed through the net, the input must be restricted to the corresponding rows of $X, \hat{A}X, \dots, \hat{A}^K X$. In Rossi et al. [129] the authors also experimented with inputs of the form $\tilde{A}^k X$ with $\tilde{A}$ not being the adjacency matrix, but for example a matrix representing closed triangles. Furthermore, they tested deeper models with more layers. These cases are not considered in this thesis.

2.8 Representation Learning

Until now, we have focused on supervised learning for classification as in Definition 2.22. However, recent deep learning approaches focus on learning from a large amount of data where there are no additional labels given. Here, a classification task to train a neural network f is derived by the creation of input and output pairs from the data itself. The aim is that the network which is trained on this artificial classification task incorporates relevant knowledge of the underlying data. Then, the network f can be used for different further tasks, either by using the outputs of f as input to another classifier or by training f

further on a specific task, which is called *fine-tuning*. The process of learning such called *representations* or *embeddings* from data is referred to as *representation learning*. In Bengio et al. [12], it is defined as “[...] learning representations of the data that make it easier to extract useful information when building classifiers or other predictors”. Representation learning is common in the realm of learning from text data and graph data. For example, in Mikolov et al. [107, 108] the authors generate vector representations for words by letting a neural network predict for a word the words around it or vice versa. Adaption of this exist for graphs where for a node the nodes next to it in the graphs are predicted [120, 65]. Furthermore, modern large language models are trained on predicting tokens from a large amount of text data [40, 25, 158] as a preparation for further task such as question answering or text generation.

The neural networks that we will propose in Part IV are connected to representation learning as they learn general vector representations from bipartite graphs (Chapter 7) or as they learn to predict links in graph data itself (Chapter 8).

Part II

Landscapes and Orometric Methods

Chapter 3

Prominence and Isolation

In this chapter, we will discuss orometric concepts and their adaption to metric data. To be more specific, we will study the valuation functions *prominence* and *isolation*. They have already been adapted to graphs by Schmidt and Stumme [133]. In this chapter, we extend this work to arbitrary bounded metric spaces. For this, we will derive network structure from metric spaces by using the nearest neighbor graphs defined in Section 2.3.1. We show that our notions is a generalization of the prominence term for networks. The proposed bridge between orometric notions for networks and metric data allows to identify locally outstanding data point and thus provides insights into data from a local perspective.

3.1 Introduction

Metric data is regularly studied in various settings. Let it be cities with their coordinates or feature vectors of inputs for a classification task. In this chapter, we will contribute to the understanding of such metric data and network data by proposing a generalization of measures of local outstandingness from networks to bounded metric spaces. This transfer is indeed a generalization, as we will see in this chapter.

Our approach to identify outstanding elements could for example be used to identify important points in datasets to recommend them to users. For example, such recommendations could provide a set of the most relevant cities in the world with respect to being outstanding in their local surroundings. In some scenarios, the metric data is equipped with an additional valuation function. Such functions, often called *score* or *heigh* functions, are often naturally provided: cities may be ranked by population; the importance of scientific authors by their *h*-index [73]. A naive approach for recommending relevant points in such settings would be: points with higher scores are more relevant. As this method seems reasonable for many applications, some obstacles arise if the “highest” points concentrate into a specific region of the underlying metric space. For example, representing the cities of the world by the twenty most populated ones would include no western European city. [1] Recommending the 100 highest mountains would not lead to knowledge about the mountains outside of Asia.[2]

Our novel approach will overcome this problem: we combine “heights and distances” to provide new valuation functions, called *prominence* and *isolation*. These functions rate points based on their height in relation to the valuations of the surrounding points. This results in valuation functions that reflect the extend to which an item is locally outstanding. The basic idea of the prominence value of a point is given by the minimal descent (with respect to the height function) that is needed to get to another point of at least same height. The isolation, sometimes also called *dominance radius*, values the distance to the next higher point with respect to the metric. These measures are adapted from the field of orometry where isolation and prominence are used in order to identify outstanding mountain peaks. We base our approach on Schmidt and Stumme [133], where the authors proposed prominence and isolation (called dominance in their paper) for networks. We generalize these to the realm of bounded metric space.

We provide insights to the novel valuation functions and demonstrate their ability to identify relevant data points for a given topic in metric knowledge graph applications. The contributions of this chapter are as follows:

- We propose a novel structure, called landscapes, that will be the abstract setting for our orometric notions.
- We propose prominence and isolation for bounded metric spaces. For this, we use the terms from [133] and prominence and isolation terms for landscapes.
- We demonstrate an artificial machine learning task for evaluating our novel valuation functions in metric data.

3.2 Related Work

The novel valuation functions prominence and isolation are inspired by topographic measures, which have their origin in the classification of mountain peaks. The idea of ranking peaks solely by their absolute height was already deprecated in 1978 [56]. The author introduced prominence for geographic mountains, a function still investigated in this realm, e.g., in Torres

[1]https://en.wikipedia.org/wiki/List_of_largest_cities on 2019-06-16

[2]https://en.wikipedia.org/wiki/List_of_highest_mountains_on_Earth on 2019-06-16

et al. [146], where the authors used deep learning methods to identify prominent mountain peaks. Another recent step for this was made in Kirmse and de Ferranti [86], where the authors investigated methods for discovering new ultra-prominent mountains. Isolation and more valuations functions motivated in the orometric realm are collected in Helman [71]. A well-known procedure for identifying peaks and saddles in 3D terrain data is described in Čomić et al. [36]. However, these approaches rely on data that approximates a continuous terrain surface via a regular square grid or a triangulation. Our data cannot fulfill this requirement. Recently the idea of transferring orometric functions to different realms of research gained attention: The authors of Nelson and McKeon [116] used topographic prominence to identify population areas in several U.S. States. In Schmidt and Stumme [133] the authors transferred prominence and isolation to co-author graphs in order to evaluate their potential of identifying ACM Fellows. We build on this for proposing our valuation functions on bounded metric data.

3.3 Landscapes

In the following, we will need *landscapes*, i.e., datasets, where we have a graph structure to traverse between points, a function to measure distances between pairs of points and a function which displays the *height* of a point. The application to bounded metric spaces will arise from the ability to derive landscapes from bounded metric data via neighborhood graphs.

Definition 3.1 (Height function)
Let M be a set. We call a function $h : M \to \mathbb{R}_{\geq 0}$ a *height function* on M.

Definition 3.2 (Landscape)
We call $L = (G, d, h)$ a *(finite) landscape* if $G = (V, E)$ is an undirected and (finite) graph, d is a metric on V such that (V, d) is bounded and h is a height function on V such that h has a unique maximum. We denote this unique highest point by $\max(L)$.

One example of a landscape is given by publication datasets, where co-author information and performance measures, such as the h-index [73], are given under the assumption that their is a unique authors with the highest h-index. Coming from graphs with heights, one can derive a landscape via the shortest path distance.

Definition 3.3 (Induced Network Landscape)
Let $G = (V, E, w)$ be a weighted graph and $h : V \to R_{\geq 0}$ a height function. Let d_{SP} be the

shortest path metric on G. We call

$$L_{\mathrm{SP}}(G,h) := (G, d_{\mathrm{SP}}, h)$$

the *induced network landscape* of G.

It is also possible to derive a landscape structure if we have distance information, but no given graph structure. This can for example be the case if we consider data points given via some sort of coordinates. Then, the distance information can be used to define the graph via the relative neighborhood graph as defined in Definition 2.11.

Definition 3.4 (Induced Metric Landscape)
Let (M,d) be a finite metric space and $h : M \to \mathbb{R}_{\geq 0}$ a height function. Let $\mathrm{RNG}(d) = (M, E_{\mathrm{RNG}(d)})$ be the corresponding RNG. We call the landscape

$$L_{\mathrm{RNG}}(M,d) := (\mathrm{RNG}(d), d, h)$$

the *induced metric landscape*.

For this definition, it is of fundamental interest that RNGs are connected what we have shown in Theorem 2.12.

3.4 Prominence and Isolation

The notions in Schmidt and Stumme [133] where defined for networks with distance and height functions, i.e., the structure which we call landscapes. They are defined as follows.

Definition 3.5 (Isolation)
Let $L = (G,d,h)$ be a landscape. The *isolation of* $v \in V$ is then defined as follows:

- If there is no point of equal height than v, than

  $$\mathrm{iso}(v) := \sup\{d(v,u) \mid u \in V\}$$

  The boundedness of (V,d) guarantees the existence of this supremum.

- If there is at least one other point in V which is higher than v, we define its isolation by

  $$\mathrm{iso}(v) := \inf\{d(v,u) \mid u \in V \colon\, h(u) > h(v)\}.$$

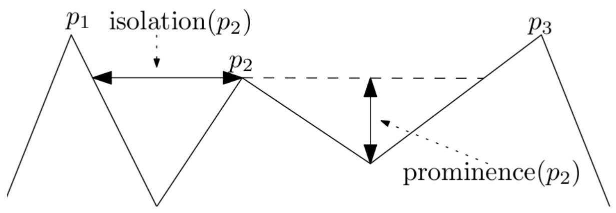


Figure 3.1: Computation of isolation and prominence of a mountain peak. The isolation displays the horizontal distance to a higher point. The prominence depicts how far one has to descend to get to a higher point.

Definition 3.6 (Prominence)
Let Let $L = (G, d, h)$ be a landscape. The *prominence* $\text{prom}_G(v)$ of $v \in V$ is defined by

$$\text{prom}_L(v) := \min\{h(v), \text{mindesc}_G(v)\} \tag{3.1}$$

where

$$\text{mindesc}_L(v) := \inf\{\max\{h(v) - h(u) \mid u \in p\} \mid p \in P_v\}.$$

The set P_v contains of all paths to vertices w with $h(w) > h(v)$, i.e.,

$$P_v := \{\{v_i\}_{i=0}^n \in P \mid v_0 = v \wedge h(v_n) > h(v)\},$$

where P denotes the set of all paths of G.

Informally, $\text{mindesc}_L(v)$ reflects on the minimal descent in order to get to a vertex in G which is “higher” then v. For this the definition makes use of the convention that $\inf(\emptyset) = \infty$. This case results in $\text{prom}_L(v)$ being the height of v. A distinction to the definition in [133] is, that we now consider all paths and not just shortest paths. This change better reflects the calculation of the prominence for mountains.

To sum up, the prominence of a point x reflects how low one has to descend to get to a point y of larger height and the isolation of x reflects the (horizontal) distance to a higher point. A sketch of how to compute the prominence and isolation of a mountain landscape is given via Figure 3.1.

3.5 Prominence in Metric Spaces

Let us now assume that (M, d) is a bounded metric space and that $h : M \to \mathbb{R}_{\geq 0}$ is a height function. As the isolation of points in a landscape only depends on the metric information but not on the edges in the graph, it is straightforward to compute isolation for bounded metric spaces with a height function. However, the question arises how to compute the prominence in this scenario. Here, we introduce two approaches:

1. We compute the prominence for different *fixed radius neighbor graphs* (e.g., graphs where two nodes are connected if their distance is under a specific radius) and then derive the prominence of the bounded metric spaces from this.

2. In the case that the metric space is finite, we also use the prominence of the induced metric landscape from Definition 3.4.

One approach is to compute the prominence of a bounded metric space with the help of decreasing radii $\delta > 0$. Here, the prominence of the metric space can then be defined via a limit process.

Definition 3.7 (δ-Radius landscape)
Let (M,d) be a metric space, h a height function on M and $\delta > 0$. Let $G_{M,\delta}$ be the δ-radius graph from Definition 2.10. We call $L_{M,\delta,h} := (G_{M,\delta}, d, h)$ the *δ-radius landscape*.

This definition allows us to compute the prominence of a bounded metric space for a fixed radius $\delta > 0$.

Definition 3.8 (δ-Prominence)
Let (M,d) be a bounded metric space and $h : M \to \mathbb{R}_{\geq 0}$ be a height function. We define the δ-prominence $\mathrm{prom}_\delta(m)$ of $m \in M$ as $\mathrm{prom}_{L_{M,\delta}}(v)$, i.e., the prominence of m in $L_{M,\delta,h}$ from Definition 3.7.

We now have a prominence term for all metric spaces that depends on a parameter δ. Hence, we want to provide in the following a natural choice for δ. We consider only those values for δ such that corresponding $G_{M,\delta}$ does not exhibit noise, i.e., there is no element without a neighbor.

Definition 3.9 (Minimal Radius)
For a bounded metric space (M,d) with $|M| > 1$, we call

$$\delta_M := \sup\{\inf\{d(m,n) \mid n \in M \setminus \{m\}\} \mid m \in M\}.$$

the *minimal threshold* of M.

Based on this definition a natural notion of prominence for metric spaces (equipped with a height function) emerges via a limit process.

Lemma 3.10 *Let M be a bounded metric space and δ_M as in Definition 3.9. For $m \in M$ the following descending limit exists:*

$$\lim_{\delta \searrow \delta_M} \operatorname{prom}_{M,\delta}(m).$$

Proof. Fix any $\hat{\delta} > \delta_M$ and consider on the open interval from δ_M to $\hat{\delta}$ the function that maps δ to $\operatorname{prom}_\delta(m)$, i.e.,

$$\operatorname{prom}_{(.)}(m) :]\delta_M, \hat{\delta}[\to \mathbb{R}, \delta \mapsto \operatorname{prom}_{M,\delta}(m).$$

It is known that it is sufficient to show that $\operatorname{prom}_{(.)}(m)$ is monotone decreasing and bounded from above. Since we have for any δ that $\operatorname{prom}_\delta(m) \leq h(m)$ holds, we need to show the monotony. Let δ_1, δ_2 be in $]\delta_M, \hat{\delta}[$ with $\delta_1 \leq \delta_2$. If we consider the corresponding graphs (M, E_{M,δ_1}) and (M, E_{M,δ_2}), it easy to see $E_{M,\delta_1} \subseteq E_{M,\delta_2}$. Hence, we have to consider more paths in Equation (3.1) for E_{M,δ_2}, resulting in a not larger value for the infimum. We thus obtain $\operatorname{prom}_{M,\delta_1}(m) \geq \operatorname{prom}_{M,\delta_2}(m)$. $\square$

Definition 3.11 (Limit Prominence in Metric Spaces)
If M is a bounded metric space with $|M| > 1$ and a height function h, we call

$$\operatorname{prom}_M(m) := \lim_{\delta \searrow \delta_M} \operatorname{prom}_{M,\delta}(m)$$

the *limit prominence* of m.

Note, if we want to compute prominence on a real world finite metric dataset, it is possible to directly compute the prominence values: in that case the supremum in Definition 3.9 can be replaced by a maximum and the infimum by a minimum, which leads to $\operatorname{prom}_M(m)$ being equal to $\operatorname{prom}_{M,\delta_M}(m)$. There are results for efficiently creating radius graphs [13]. However, for our needs in this chapter, in particular in the experiment section, a quadratic brute force approach for generating all edges is sufficient. We want to show that our prominence definition for bounded metric spaces is a natural generalization of Definition 3.6.

Lemma 3.12 *Let $G = (V, E)$ be a finite, connected graph with $|V| \geq 2$. Consider the network induced landscape $L_{\mathrm{SP}}(G, h) = (G, d_{\mathrm{SP}}, h)$. Then the prominence $\operatorname{prom}_{L_{\mathrm{SP}}}(\cdot)$ from Definition 3.6 and $\operatorname{prom}_M(\cdot)$ Definition 3.11 coincide.*

Proof. Let $M := V$ be equipped with the shortest path metric d on G. As G is connected and has more than one node, we have $\delta_M = 1$. Hence, (M, E_{δ_M}) from Definition 3.7 and G are equal. Therefore, the prominence terms coincide. $\square$

In the finite case, a second approach to compute prominences on (M,d) is by using the induced metric landscape.

Definition 3.13 (RNG Prominence)
Let (M,d) be a finite metric space. The *rng prominence* of (M,d) is given via the prominence of the induced network landscape from Definition 3.4, i.e.,

$$\mathrm{prom}_{\mathrm{RNG}(d)}(\cdot) := \mathrm{prom}_{L_{\mathrm{RNG}(M,d)}}(\cdot).$$

3.6 Experiments

Now assume that we have given a bounded metric space M which represents the dataset and a given height h. The following questions shall evaluate if our functions isolation and prominence provide useful information about the relevance of given points in the metric space. If (M,d,h) is a metric space equipped with an additional height function, we aim to train a classifier that classifies the points in the dataset as relevant (1) or not (-1). To be more concrete, we aim to classify municipalities in Germany and France as relevant or irrelevant. To have a ground truth, we make the assumption, that a municipality is relevant if it has an university. We admit that the underlying classification is not meaningful in itself. It treats a real geographic case while our model could also handle more abstract scenarios. However, since this setup is essentially a benchmark framework (in which we assume cities with universities to be more relevant) we refrain from employing a more meaningful classification task in favor of a controllable classification scenario. Our research questions are now:

1. **Are prominence and isolation as individual features characteristically for relevance?** We use isolation and/or prominence for a given set of data points as features. To which extend do these features improve learning a classification function for relevance?

2. **Do prominence and isolation provide additional information, not catered by the absolute height?** Do prominence and isolation improve the prediction performance of relevance compared to just using the height? Does a classifier that uses prominence and isolation as additional features produce better results than a classifier that just uses the height?

Table 3.1: Basic statistics of the country datasets extracted from Wikidata.

	Municipalities	University Locations
France	2064	89
Germany	2986	160

We extract information about municipalities in the countries of Germany and France from Wikidata. Wikidata is a structure that stores knowledge via *statements*, linking *entities* via *properties* to *values*. A detailed description can be found in Vrandecic and Krötzsch [152], while Hanika et al. [69] gives an explicit mathematical structure to the Wikidata graph and shows how to use the graph for extracting implicational knowledge from Wikidata subsets. We investigate if prominence and isolation of a given municipality can be used as features to predict university locations in a classification setup. We use the query service of Wikidata[3] to extract points in the country maps from Germany and France and to extract all their universities. All necessary SPARQL queries employed on can be found on the GitHub repository of the paper from which this chapter was derived.[4] In the following, we provide details to our queries.

- Wikidata provides different relations for extracting items that are instances of the notion city. The obvious choice is to employ the *instance of* (P31) property for the item *city* (Q515). Using this, including *subclass of* (P279), we find insufficient results. More specific, we find only 102 French cities and 2215 German cities.[5] For Germany, there exists a more commonly used item *urban municipality of Germany* (Q42744322) for extracting all cities, while to the best of our knowledge, a counterpart for France is not provided.
- The preliminary investigation leads us to use *municipality* (Q15284), again including the *subclass of* (P279) property, with more than 5000 inhabitants.
- Since there are multiple french municipalities that are not located in the mainland of France, we encounter problems for constructing the metric space. To cope with that we draw a basic approximating square around the mainland of France and consider only those municipalities inside.

[3]https://query.wikidata.org/
[4]https://github.com/mstubbemann/Orometric-Methods-in-Bounded-Metric-Data
[5]queried on 2019-08-07

- We find the class of every municipality, i.e, university location or non-university location as follows. We use the properties *located in the administrative territorial entity* (P131) and *headquarters location* (P159) on the set of all universities and checked if these are set in Germany or France. Using only P131 was not sufficient. An example of university that has not set P131 is *TU Dortmund* (Q685557).[6]
- We match the municipalities with the university properties. This is necessary because some universities are not related to municipalities through P131, e.g., *Hochschule Niederrhein* (Q1318081) is located in the administrative location *North Rhine-Westphalia* (Q1198)[8], which is a federal state containing multiple municipalities. For these cases we check the university locations manually. This results in 2064 municipalities (89 university loc.) in France and 2986 municipalities (160 university loc.) in Germany.

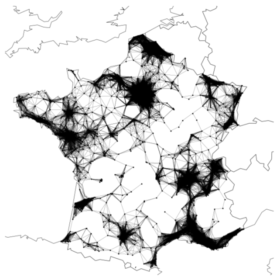
Statistics of the resulting datasets can be found in Table 3.1.

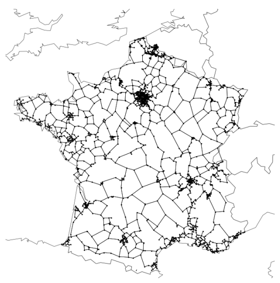
3.6.1 Binary Classification Task

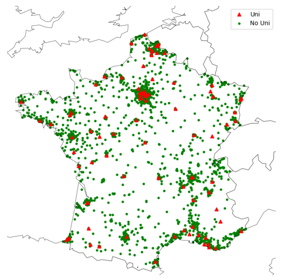
Setup. We consider the question whether isolation and prominence are helpful for the identification of academic importance as a binary classification task, where we use use the “height” and the derived isolation and prominence values as features. Here, the height function is given via the amount of inhabitants. The class labels are given by the binary information whether a city has a university. The following steps are needed for this.

1. **Identify a Metric Space:** To get a metric space, we use the geographic coordinates in longitude and latitude and approximate the great circle distances between the municipalities.
2. **Identify height function:** Since we want to compute the prominence and isolation of the data points, we also have to identify a height function. In the following, we use the population of the municipalities.
3. **Compute isolations:** Based on the steps before we are now able to compute the isolation for all German and French municipalities.
4. **Compute the minimal radii:** For computing the limit prominence values for all data points, we need to compute the minimal radii for the German and French datasets.

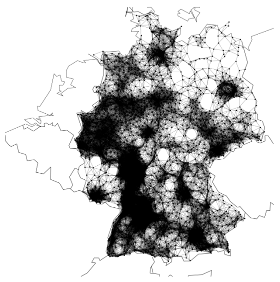
[6]last checked on 2019-10-26

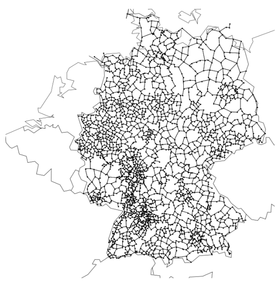

(a) Minimal Radius Graph of France

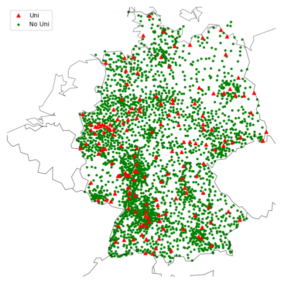

(b) RNG of France



(c) University Locations of France


(d) Minimal Radius Graph of Germany


(e) RNG of Germany



(f) University Locations of Germany

Figure 3.2: Minimal radius graphs, RNGs an university locations of France and Germany.

5. **Compute limit prominences:** Equipped with the minimal radius, we are now able to compute the limit prominence values for all municipalities using Definition 3.11. Note, that for finite datasets the limit can be omitted, i.e.,

$$\mathrm{prom}_M = \mathrm{prom}_{\delta_M}.$$

6. **Compute the RNG:** To compute the RNG prominence with Definition 3.13 we need to compute the induced metric landscape from Definition 3.4. Hence, we compute the RNG with respect to the metric discussed in Item 1.

7. **Compute the RNG prominence:** Finally, the RNG prominence can be computed via Definition 3.13.

The resulting Minimal Radius graphs, RNGs and university locations are depicted in Figure 3.2. We use absolute height, limit prominence, rng prominence and isolation for all data points and normalize them. Since our dataset is highly imbalanced, common classifiers tend to simply predict the majority class. To overcome the imbalance, we use inverse penalty weights with respect to the class distribution. We want to emphasize again that the goal for the to be introduced classification task is not to identify the best classifier. Rather we want to produce evidence for the applicability of employing isolation and prominence as features for learning a classification function. We decide to use logistic regression with L^2 regularization and Support Vector Machines [37] with a radial kernel. For our experiment we use scikit-Learn [119]. As penalty factor for the `SVC` we set $C = 1$. For γ we stick to previous work by [2] and set it to one. For all combinations of population, isolation, limit prominence and rng prominence, we use 100 iterations of 5-fold stratified cross-validation. From the performance measures mentioned in Section 2.5, we report the geometric mean and its components, i.e., ACC+ and ACC-.

Results. The results of the computations are depicted in Table 3.2. We summarize our findings in the following.

- *Isolation is a good indicator for structural relevance.* For both countries and classifiers isolation outperforms population.
- *Combining absolute height with our valuation functions leads to better results.* For all datasets, combining the height with our valuation functions outperforms the solely use of the height. Furthermore, the best results are reached, when the height is combined with local valuation functions in 3 of 4 cases: The SVM has the highest geometric means when isolation, rng prominence and population are combined. For the French dataset, the logistic regression works the best when isolation and rng prominence are combined. For the German dataset, the logistic regression has the highest performance when population and isolation are combined.
- *RNG Prominence is more useful then limit prominence.* We draw from our result that limit prominence, in contrast to RNG prominence, is not a useful as an individual indicator. The g-means for using only limit prominence can not cope with the absolute height, isolation or RNG prominence. Additionally, adding limit prominence as additional feature to height, isolation and RNG prominence does not lead to a performance boost. Note, that both prominence terms are computed on a graph and that all nodes that have at least one graph neighbor of at least equal height have prominence 0. This makes limit prominence a very strict valuation function: recall that we constructed a radius graphs by using radius margins as indicators for edges, leading to a dense graph structure in more dense parts of the metric space. Hence, a point in a more dense part has many neighbors and thus many potential paths that may lead to a very low

or vanishing prominence value. The minimal radius as in Definition 3.9 is about 34 kilometers for Germany and 54 kilometers for France, leading to graphs with 61636 and 103309 edges, respectively. Thus, a municipality has a not vanishing prominence if it is the most populated point in a radius of over 34 kilometers, respectively 54 km. Only 75 municipalities of France have non zero prominence, with 40 of them being university locations. Germany has 104 municipalities with positive prominence with 72 of them being university locations. Thus, limit prominence alone as a feature is insufficient for the prediction of university locations. In contrast, the RNGs are fundamentally sparser. The have 6942 edges for Germany and 4589 edges for France. This leads, in the case of Germany, to 877 points with a positive RNG prominence with 81 of them being university locations. For the case of France, we have 877 municipalities with a positive RNG prominence with 137 of them being university locations. Hence, RNG prominence is not as strict as limit prominence which could be one explanation why it is a more useful feature.

- *Support vector machine and logistic regression lead to similar results, except for the RNG prominence.* To the question, whether our valuation functions improve the classification compared with the population feature, SVMs and logistic regressions provide similar answers: isolation always outperforms population, a combination of population and isolation is always better then using just the plain population feature, the same holds for combining the rng prominence with population compared to just use the population. When it comes to the question, whether RNG prominence is a useful feature when added to height and isolation as a third feature, both classifier lead to different results: For SVM classifiers, adding this feature enhances the performance, for logistic regression, it does not.

3.7 Conclusion and Outlook

In this chapter, we presented a novel approach to identify outstanding elements in datasets. For this, we employed orometric valuation functions, namely prominence and isolation and transferred them to the realm of bounded metric spaces. In particular, we generalized previously known results from the field of finite networks.

The theoretical work was motivated by the observation that real-world data is often of metric nature. Furthermore, such datasets are often equipped with some kind of height function in a natural way. Based on this we proposed in this chapter the groundwork for the identification of relevant data points by the means of local outstandingness.

To evaluate the capabilities for identifying locally outstanding data points we selected an artificial classification task. We identified all French and German municipalities from Wikidata and evaluated if a classifier can learn a meaningful connection between our valuation functions and the relevance of a municipality. To gain a binary classification task and to have a benchmark, we assumed that universities are primarily located at relevant municipalities. In consequence, we evaluated if a classifier can use prominence and isolation as features to predict university locations. Our results showed that isolation and prominence are indeed helpful for identifying locally outstanding points and are therefore indeed useful to study data from a local perspective.

Future work could focus on the development of an outstandingness-based item recommender system and the evaluations of its practical usability in an empirical user study. Furthermore, we are interested in the transferability of other orometry-based valuation functions to both metric and network data.

Table 3.2: Results. We report results for all possible feature combinations from population(pop), isolation (iso), rng prominence (rpr) and limit prominence (lpr).

	Country	France				Germany			
	Classifier	SVM		LR		SVM		LR	
	Score	mean	std	mean	std	mean	std	mean	std
iso	ACC+	0.5748	0.0086	0.6265	0.0064	0.5706	0.0040	0.6277	0.0056
	ACC-	0.9597	0.0008	0.9477	0.0009	0.9763	0.0003	0.9599	0.0009
	G-Mean	0.7427	0.0055	0.7705	0.0039	0.7464	0.0026	0.7762	0.0035
lpr	ACC+	0.2361	0.0061	0.4075	0.0067	0.1602	0.0054	0.3299	0.0052
	ACC-	1.0000	0.0000	0.9971	0.0005	1.0000	0.0000	0.9988	0.0002
	G-Mean	0.4858	0.0063	0.6374	0.0053	0.4003	0.0069	0.5740	0.0045
rpr	ACC+	0.4362	0.0067	0.5753	0.0120	0.3185	0.0025	0.5028	0.0082
	ACC-	0.9969	0.0003	0.9909	0.0009	0.9996	0.0001	0.9954	0.0003
	G-Mean	0.6594	0.0051	0.7550	0.0079	0.5642	0.0022	0.7074	0.0058
pop	ACC+	0.4847	0.0039	0.5880	0.0136	0.3589	0.0045	0.5131	0.0071
	ACC-	0.9931	0.0004	0.9837	0.0007	0.9972	0.0001	0.9904	0.0005
	G-Mean	0.6938	0.0028	0.7605	0.0089	0.5982	0.0038	0.7128	0.0049
iso+lpr	ACC+	0.5590	0.0095	0.6135	0.0107	0.5482	0.0054	0.6028	0.0062
	ACC-	0.9625	0.0011	0.9512	0.0011	0.9774	0.0004	0.9660	0.0009
	G-Mean	0.7335	0.0062	0.7639	0.0066	0.7320	0.0036	0.7631	0.0039
iso+rpr	ACC+	0.6031	0.0103	0.6380	0.0047	0.6194	0.0047	0.6551	0.0046
	ACC-	0.9694	0.0012	0.9600	0.0008	0.9806	0.0007	0.9728	0.0004
	G-Mean	0.7646	0.0065	**0.7826**	0.0029	0.7793	0.0029	0.7983	0.0028
iso+pop	ACC+	0.6045	0.0108	0.6336	0.0057	0.6450	0.0073	0.6676	0.0061
	ACC-	0.9703	0.0013	0.9619	0.0008	0.9807	0.0007	0.9752	0.0006
	G-Mean	0.7658	0.0069	0.7807	0.0035	0.7953	0.0045	**0.8069**	0.0037
lpr+rpr	ACC+	0.4244	0.0048	0.5549	0.0076	0.3357	0.0036	0.4939	0.0077
	ACC-	0.9972	0.0004	0.9936	0.0007	0.9996	0.0001	0.9961	0.0003
	G-Mean	0.6505	0.0036	0.7425	0.0052	0.5793	0.0031	0.7014	0.0055
lpr+pop	ACC+	0.4943	0.0059	0.5675	0.0072	0.3766	0.0061	0.5099	0.0082
	ACC-	0.9959	0.0004	0.9902	0.0007	0.9975	0.0003	0.9930	0.0004
	G-Mean	0.7016	0.0042	0.7496	0.0048	0.6129	0.0050	0.7115	0.0057
rpr+pop	ACC+	0.5404	0.0077	0.6184	0.0097	0.4587	0.0049	0.5622	0.0088
	ACC-	0.9914	0.0003	0.9862	0.0008	0.9958	0.0003	0.9923	0.0003
	G-Mean	0.7320	0.0052	0.7809	0.0060	0.6758	0.0036	0.7469	0.0059
iso+lpr+rpr	ACC+	0.5990	0.0099	0.6336	0.0062	0.6159	0.0040	0.6466	0.0054
	ACC-	0.9699	0.0012	0.9621	0.0009	0.9815	0.0005	0.9741	0.0006
	G-Mean	0.7622	0.0063	0.7807	0.0038	0.7775	0.0025	0.7936	0.0033
iso+lpr+pop	ACC+	0.6024	0.0102	0.6279	0.0070	0.6412	0.0071	0.6547	0.0065
	ACC-	0.9709	0.0012	0.9639	0.0009	0.9812	0.0006	0.9774	0.0008
	G-Mean	0.7647	0.0064	0.7779	0.0043	0.7932	0.0044	0.7999	0.0039
iso+rpr+pop	ACC+	0.6301	0.0086	0.6208	0.0104	0.6542	0.0039	0.6584	0.0045
	ACC-	0.9739	0.0013	0.9677	0.0009	0.9826	0.0007	0.9796	0.0007
	G-Mean	**0.7834**	0.0052	0.7750	0.0065	**0.8018**	0.0024	0.8031	0.0027
lpr+rpr+pop	ACC+	0.5406	0.0089	0.6103	0.0096	0.4538	0.0059	0.5599	0.0079
	ACC-	0.9937	0.0006	0.9893	0.0007	0.9962	0.0004	0.9924	0.0003
	G-Mean	0.7329	0.0061	0.7770	0.0061	0.6724	0.0044	0.7454	0.0052
all	ACC+	0.6296	0.0090	0.6134	0.0072	0.6534	0.0038	0.6531	0.0049
	ACC-	0.9740	0.0012	0.9690	0.0007	0.9831	0.0006	0.9810	0.0007
	G-Mean	0.7831	0.0055	0.7709	0.0045	0.8015	0.0023	0.8004	0.0030

Chapter 4

Mountain Graphs and Line Parent Trees

While the last chapter dealt with the computation of prominence and isolation itself, we now study the underlying structures that determine them. More specifically, we will investigate for a node which higher point is reached to determine its prominence and which point is the lowest point on this travel. Hence, after this chapter, we will not only be able to identify locally outstanding points, but also to derive meaningful hierarchies of dominance between them. This further helps to understand networks from local perspectives.

4.1 Introduction

To understand which the locally important nodes of a graph are and how they relate to each other is crucial for many applications where the study of complete large-scale networks is not feasible. One such problem is for example graph visualization. There exist a variety of established approaches for visualization [48, 55, 90]. However, they are often not suited to large real-world networks. Another problem is that the importance of edges in such networks often differs. This is especially possible in networks that are projections from other graphs. Examples of such projections are networks of co-group memberships of Youtube users or co-Follower networks. Such networks exhibit a large amount of low-weighted edges where the set of shared neighbors in the original graph was small. Hence, it is crucial to derive compact representations of structurally important relationships.

To derive more compact representations, many works generate smaller graphs from the original network. This is done by random sampling of substructures [127], by considering only dense parts of the network [137] or by merging vertices and edges [130]. Another approach is to identify and study hierarchic connections [102, 66].

Often, the importance of individual vertices can be measured by a given height function as defined in Definition 3.1. For example, Twitter users can be evaluated by the amount of followers. For such networks, we propose structures that identify hierarchies between locally outstanding nodes. These structures are based on (topographic) prominence. Recall, from the last chapter, that the prominence of a mountain quantifies its local outstandingness by computing the minimal vertical descent that is needed to reach a higher mountain peak. These paths with minimal descent to a higher peak deliver two important reference points for each mountain. First, the lowest point of this path (called the *key col*). Secondly, the first higher peak that is reached after the key col, called the *line parent*. Adopting these notions to networks allows to derive a relatively small tree structure which displays how outstanding nodes are dominated by each other. This simplifies the study of networks because trees can be visualized satisfactory and navigating through them is possible for larger node sets.

When deriving such structures the question arises on how to traverse through the network. Here, it is natural to use the given edges of the graph. However, as mentioned above, some edges in the graph may represent weak connections. Hence, it can be beneficial to remove them as a preprocessing step.

To this point, we propose a method for parameter-free edge-reducing based on the RNG [147], as defined in Definition 2.11. We will show that our edge-reduction technique preserves connectivity of the graph. This is not guaranteed by other approaches which discard edges via a weight threshold or only keep the k most important edges for each node. Our method works as follows.

- First, if desired, we discard edges that represent weak connections. This is done without the need to choose any parameters and in such a way that the connectivity is preserved. This preprocessing step is described in Section 4.5.
- We then identify locally outstanding node “peaks” and a structure that displays how peaks are connected through lower points. We call this structure the *mountain graph*. It is defined in Section 4.3.
- Finally, we compute a hierarchy of the peaks of the graph which uses the line parent relation. We will call the resulting structure the *line parent tree* or the *line parent hierarchy*. It is introduced in Section 4.4.

The key point of our approach are mountain graphs and line parent trees. Discarding unimportant edges is an optional preprocessing step.

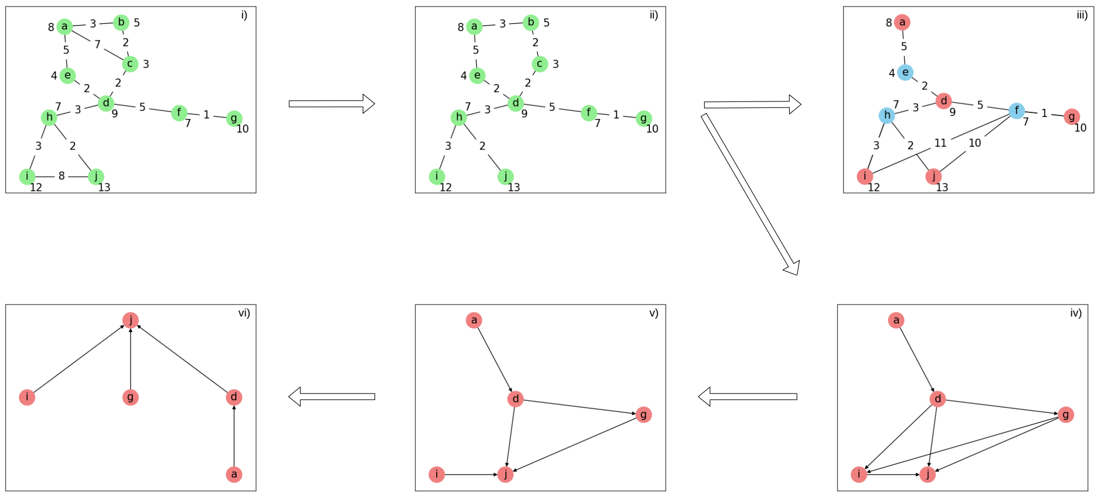


Figure 4.1: Generating the mountain graph and the line parent tree. In i), a graph is displayed with the heights next to the nodes and with the edge weight put in the middle of each edge. The RNG in ii) is derived by discarding the edge between i and j, because h is closer to both i and j than they are to each other and by discarding the edge between a and c because node b is closer to both of them than they are to each other. From this, the mountain graph is derived in iii), where we display the shortest path distances between the cols and peaks. Additionally, we derive from ii) the peak graph which is displayed in iv). According to Definition 4.9, we then discard the edge from g to i because j is closer to the key col f than i and we discard the edge between d and i because j is closer to the key col h to arrive at v). From this, the line parent tree vi) is derived by discarding the edge between d and g because the key col h over which j is reached is closer to g than the key col f over which g is reached.

4.2 Related Work

Deriving compact structures that display important relations in the original network is often done via sampling vertices or edges [127, 91, 95, 100, 101]. In contrast, other works focus on the aggregation of vertices and edges such that the original network can be reconstructed [145, 99, 130]. All these methods have in common that they return a proxy of the original network. Thus, they are not able to identify hierarchies and connections of outstanding vertices that are not approximations or explicit subgraphs of the original graph.

The study of hierarchic structures has gained recent interest. Lu et al. [102] derive acyclic graphs by removing cycles. Other works use likelihoods to derive suitable hierarchies [32, 106] or provide a quantification on how “hierarchical” a graph is [66]. In contrast to our approach, these methods are solely based on the graph structure and are not able to incorporate the “height” of the nodes.

As discussed in the related work part of the last chapter, there is a variety of works that deals with adapting orometric methods to different areas [116, 82, 133, 118]. However, all these works have in common, that they focus on the computation of prominence. In the present chapter, we go a step further and study the underlying structure, i.e., the connections to key cols and line parents which determine prominence values.

4.3 Mountain Graphs

Section 3.5 dealt with using mountain measures for feature engineering, i.e,. to generate quantitative facts over points in landscapes. Now, we present an approach to derive small hierarchies between peaks from larger networks. We first explain how one can derive mountain graphs and line parents from landscapes. Afterwards, we present the already mentioned optional preprocessing step based on RNGs to remove a significant amount of edges while preserving the connectivity of the original network. A complete example of the end-to-end procedure which we develop in the following is given by Figure 4.1.

In the following, we will always assume to have given a **finite** landscape $L = (G, d, h)$ where G is **connected**.[1] As mentioned earlier, our aim is to display hierarchies of peaks and connections between peaks and cols. For this, we first have to clarify how peaks and cols are defined. We will then argue how prominence of peaks in landscapes as in Definition 3.6 is determined by connections between them.

Definition 4.1 (Peaks, Mountain paths, Cols[2])
We call $v \in V$ a *peak* of L if $h(v) > h(u)$ for all $u \in N(v)$ and denote by $P(L)$ the set of peaks of L. A path p of G is a *mountain path* if $\text{start}(p), \text{end}(p) \in P(L)$. We denote by $M(L)$ the set of all mountain paths. For each $p \in M(L)$ we call

$$c(p) := \text{argmin}_{v \in p} h(p)$$

the *col* of p. If this argmin is not unique, we choose the point in the path which is visited first.

To compute the prominence of a peak $v \in P(L)$, we have to identify the cols which connect it with higher peaks.

[1]This is only assumed for simplicity. One could study each connected component for itself if the graph is not connected.

[2]To simplify notations, our definition of cols allow only one col per path which differs from the definition in geography,

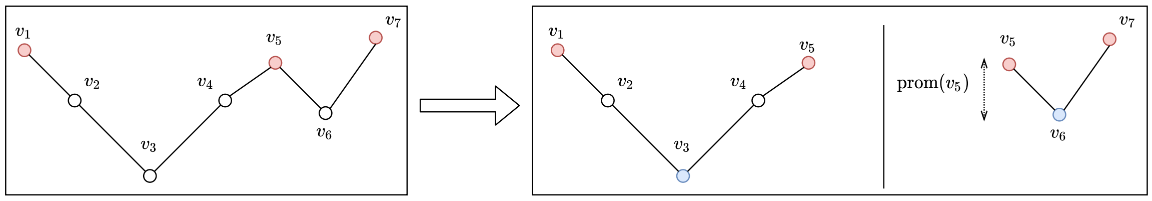


Figure 4.2: Computing the prominence via key cols and line parents: Here, the vertical positioning displays the height of the different points. To compute the prominence of v_5, we first identify the paths that lead to higher peaks. Then, we determine the cols of these paths and compute the height difference of v_5 to these cols. The lower difference yield the prominence of v_5.

Definition 4.2 (Cols of Peaks)
For each $v \in P(L)$ we call the set

$$\begin{aligned}\uparrow_L(v) := \{p \in M(L) \mid\ & \mathrm{start}(p) = v, h(\mathrm{end}(p)) > h(v),\\ & \nexists u \in p \setminus \{\mathrm{end}(p), v\} :\\ & u \in P(L) \wedge h(u) > h(v)\}\end{aligned}$$

the *ascending paths* of v and denote by

$$C_L(v) := \{c(p) \mid p \in \uparrow_L(v)\}$$

the *cols of* v.

We omit L in the index if clear from the context. As prominence as in Definition 3.6 is the *minimal* descent needed to go to higher points, we are just interested in the *highest* cols.

Definition 4.3 (Key Cols)
For each peak $v \in P(L) \setminus \{\max(L)\}$ we call the elements of

$$K_L(v) := \{u \in C_L(v) \mid h(u) = \max_{\tilde{u} \in C_L(v)} h(\tilde{u})\}$$

the *key cols* of v.

Note, that the key cols are the highest possible lowest points of paths ending at higher peaks. Thus, they determine the prominence values of peaks.

Corollary 4.4 *Let $v \in P(L) \setminus \{\max(L)\}$ be a peak of L. Let $u \in K_L(v)$. It then holds that*

$$\mathrm{prom}_L(v) = h(v) - h(u).$$

For $v \in P(L) \setminus \{\max(L)\}$ the prominence is the minimal height difference to a col, i.e.,

$$\mathrm{prom}_L(v) = \min_{u \in C(v)} h(v) - h(u).$$

Again, we write $\mathrm{prom}(v)$ and $K(v)$ if the choice of the landscape is clear. An illustration of determining prominence values via the identification of key cols is given by Figure 4.2.

We are not only interested in the key cols, but also in the higher peaks which are reached after the key cols.

Definition 4.5 (Dominators)
For each peak v, we call the set

$$D_L(v) := \{\mathrm{end}(p) \mid p \in \uparrow_L (v) \wedge c(p) \in K(v)\}$$

the *dominators* of v.

Definition 4.6 (Mountain Graph)
For a given finite landscape $L = ((V,E),d,h)$, let $K(L) := \cup_{v \in P(L)} K_L(v)$ be the set of key cols of L and let $V_{\mathrm{MG}}(L) := P(L) \cup K(L)$ be the *critical points*. Let for $v \in P(L)$ be

$$\uparrow_L^K (v) := \{p \in \uparrow_L (v) \mid c(p) \in K_L(v)\}$$

the ascending paths of v with key cols as cols. Let then

$$E_{\mathrm{MG}}(L) := \bigcup_{v \in P(L)} \left(\bigcup_{p \in \uparrow_L^K(v)} \{\{v, c(p)\}, \{c(p), \mathrm{end}(p)\}\} \right).$$

The graph

$$\mathrm{MG} = (V_{\mathrm{MG}}(L), E_{\mathrm{MG}}(L))$$

is called the *mountain graph* of L. We call the corresponding landscape $L_{\mathrm{MG}} := (\mathrm{MG}_L, d|_{\mathrm{MG}}, h|_{\mathrm{MG}})$ the *mountain landscape of L.*

To sum up, if a peak v_1 is connected via a key col u to a higher peak v_2, we add edges between v_1 and u and between u and v_2 to the mountain graph. Thus, the mountain graph displays which peaks are connected through which key cols to each other. If clear from the context, we omit L and simply write $\mathrm{MG} = (V_{\mathrm{MG}}, E_{\mathrm{MG}})$. The mountain graph contains all relevant information for the computation of prominences as the following theorem shows.

Theorem 4.7 *The following statements hold:*

1. MG *is connected.*
2. $P(L) = P(L_{\mathrm{MG}})$
3. *Consider for each peak* $v \in P(L) \setminus \{\max(L)\}$ *of* L *the set* $N'_{\mathrm{MG}}(v) := \{u \in N_{\mathrm{MG}}(v) \mid \exists v' \in N_{\mathrm{MG}}(u) : h(v') > h(v)\}$. *Then:*

$$u \in N'_{\mathrm{MG}}(v) \Rightarrow \exists u' \in C_L(v) : h(u') \geq h(u).$$

4. *It holds for* $v \in P(L) \setminus \{\max(L)\}$ *that:*

$$\mathrm{prom}_L(v) = \min_{u \in N'_{\mathrm{MG}}(v)} (h(v) - h(u)).$$

Proof. **1)** Let $u \in V_{\mathrm{MG}} \setminus \{\max(L)\}$. We show that u is connected to $\max(L)$. As every $u' \in K(L)$ is connected to some $\hat{u} \in P(L)$ in G_{MG}, it is sufficient to consider the case that $u \in P(L)$. For each dominator $u_1 \in D_L(u)$ of u it holds that $h(u_1) > h(u)$ and u_1 is connected to u in G_{MG}. Analogously, if $u_1 \neq \max(L)$, there exists u_2 with $h(u_2) > h(u_1) > h(u)$ such that u_2 is connected to u_1 and thus to u. Repeating this argument as long as the last element has a dominator leads to chains $(u, u_1, \ldots, u_m)$ where all elements are connected and $h(u_{i-1}) < h(u_i)$. As V_{MG} is finite, there must be an $m \in N$ such that $u_m = \max(L)$ holds.

2) "$\subseteq$:" Let $v \in P(L)$ and $u \in N_{\mathrm{MG}}(v)$. Then there must exist a mountain path $p = (v_0, \ldots, u, \ldots, v_m)$ in L such that $v_0 = v$ or $v_m = v$ and such that $c(p) = u$. If $v_0 = v$ we have

$$h(v) \underbrace{>}_{v \in P(L)} h(v_1) \geq h(c(p)) = h(u).$$

Analogously, if $v = v_m$, we have

$$h(v) \underbrace{>}_{v \in P(L)} h(v_{m-1}) \geq h(c(p)) = h(u).$$

Since $u \in N_{\mathrm{MG}}(v)$ was chosen arbitrary, it follows that $v \in P(L_{\mathrm{MG}})$.

"$\supseteq$:" Let $v \notin P(L)$. We show $v \notin P(L_{\mathrm{MG}})$. If $v \notin V_{\mathrm{MG}}$, we are finished. Otherwise $v \in K(L)$ and hence it must exist $u \in P(L)$ such that $v \in K(u)$ and thus $\{u, v\} \in E_{\mathrm{MG}}$. With an analog argumentation as above, we can see that $h(u) > h(v)$ and thus $v \notin P(L_{\mathrm{MG}})$.

3) Let $u \in N'_{\mathrm{MG}}(v)$. It either holds that $u \in K(v)$ or that there exists $\tilde{v} \in P(L)$ such that $u \in K(\tilde{v})$ and there is an ascending path $p = (\tilde{v}, \ldots, u, \ldots, v) \in \uparrow_L (v)$ with $c(p) = u$.

In the first case, we are finished by choosing $u' = u$. Hence, we assume the existence of $\tilde{p} = (\tilde{v} = v_0, \dots, u = v_i, \dots, v = v_m) \in\uparrow_L(\tilde{v})$. Let now $v' \in N_{\text{MG}}(u)$ with $h(v') > h(v)$. Hence, there exists $v'' \in P(L)$ and a path $p'' = (u_0 = v'', \dots, u_j = u, \dots, u_l = v') \in \text{MG}$ with $c(p'') = u$ such that

$$p'' \in\uparrow_L(v'') \ \texttt{or} \ (u_l, \dots, u_0) \in\uparrow_L(v').$$

Let $p''' := (v = v_m, \dots, v_i = u = u_j, u_{j+1}, \dots, u_l = v')$. If p''' is not a path, there can only be pairs of doubled vertices of the form (v_k, u_s) as p' and p'' are paths. Let s be maximal such that this is the case. Then, $\hat{p} := (v = v_m, v_{m-1}, \dots, v_k, u_{s+1}, \dots, u_l = v')$ is a path. If $\hat{p} \notin\uparrow_L(v)$, there must exist $n \in \{s+1, \dots l-1\}$ with $u_n \in P(L)$ and $h(u_n) > h(v)$. Choose n minimal that this is the case and set $\hat{v} = u_n$. Then set $p := (v = v_m, \dots, v_k, u_{s+1}, \dots, u_n = \hat{v})$. If $\hat{p} \in\uparrow_L(v)$ instead simply set $p := \hat{p}$. In both cases, we get $p \in\uparrow_L(v)$. Let $u' := c(p)$. As either $u' \in p'$ or $u' \in p''$ and $c(p'') = c(p') = u$, it follows $h(u') \geq h(u)$.

4) "$\geq$": Let $u \in K(v)$. It holds that

$$\text{prom}_L(v) = h(v) - h(u) \underbrace{\geq}_{u \in K(v) \subseteq N'_{\text{MG}}} \min_{u' \in N'_{\text{MG}}} (h(v) - h(u'))$$

"$\leq$:" Let $u \in N'_{\text{MG}}(v)$. Choose $u' \in C(v)$ with $h(u') \geq h(u)$. Hence:

$$\text{prom}_L(v) = \min_{\tilde{u} \in C(v)} h(v) - h(\tilde{u}) \leq h(v) - h(u') \leq h(v) - h(u).$$

Since u was chosen arbitrary from $N'_{\text{MG}}(v)$ it directly follows that

$$\text{prom}_L(v) \leq \min_{u \in N'_{\text{MG}}} (h(v) - h(u)).$$

□

Theorem 4.7 shows that to study relations between cols and peaks that determine prominence values, it is sufficient to check the cols to which a peak is connected in the mountain graph. Note, that the key cols and the paths between peaks passing through them have to be determined to derive the mountain graph. Hence, Theorem 4.7 does not allow for a faster computation of prominence values. Instead, it provides a representation that can be used to observe important connections between peaks and cols.

4.4 Line Parent Trees

As the prominence of mountain peaks is computed by descending to cols and then ascending to higher peaks, a hierarchy between peaks arises by the question to which higher peak one can traverse from a given peak to compute the prominence.

Definition 4.8 (Peak Graph)
Let

$$E_P(L) := \bigcup_{v \in P(L)} \{\{\text{start}(p), \text{end}(p)\} \mid p \in \uparrow_L^K(v)\}.$$

We call $\text{PG}(L) := (P(L), E_P(L))$ the *peak graph* of L and we call

$$T_{\text{PG}(L)} := \bigcup_{v \in P(L)} \{(\text{start}(p), c(p), \text{end}(p)) \mid p \in \uparrow_L^K(v)\}$$

the *defining triples* of $\text{PG}(L)$.

Peaks may be connected to different key cols and different higher peaks. To define a meaningful hierarchy on the peaks, we use the metric d to determine a unique line parent for all peaks.

Definition 4.9 (Line Parents)
Let

$$\begin{aligned} T'_{\text{PG}}(L) := \{(v,u,\tilde{v}) \in T_{\text{PG}} \mid \nexists v' : (v,u,v') \in T_{\text{PG}} \wedge \\ (d(u,v') < d(u,\tilde{v}) \vee (d(u,v') = d(u,\tilde{v}) \wedge h(v') > h(\tilde{v})))\}. \end{aligned} \tag{4.1}$$

Let

$$E_{\text{LP}}(L) := \{\{\{v,\tilde{v}\} \mid \exists u : (v,u,\tilde{v}) \in T'_{\text{PG}} \wedge \nexists u', v' : (v,u',v') \in T'_{\text{PG}} \wedge d(v,u') < d(v,u)\}. \tag{4.2}$$

We call $\text{LP}(L) := (P(L), E_{\text{LP}}(L))$ the *line parent graph* of L. If if we have $\{u,v\} \in E_{\text{LP}}(L)$ with $h(v) > h(u)$, v is a *line parent* of u.

Again, we omit L when possible without confusion. In Equation (4.1), we first remove edges to higher peaks that are further away from the corresponding key col. If there are multiple higher peaks with the exact same distance to the key col, we keep the highest peak. Then we remove in Equation (4.2) edges where the key col is further away.

Theorem 4.10 *If for each peak $v \in P(L) \setminus \{\max(L)\}$ the line parent is unique, then it holds that* $\mathrm{LP}(L)$ *is a tree.*

Proof. Let $n := |V_{\mathrm{MG}}|$. As shown for example in Deo [39], it is sufficient to show that $\mathrm{LP}(L)$ is connected and has $n-1$ edges.
Connectivity: We show, that each $u \in P(L)$ is connected to $\max(L)$. Let

$$\mathrm{LP} : P(L) \setminus \{\max(L)\} \to P(L)$$

map every other peak to its line parent. Since for each $u \in P(L)$ it holds that $h(\mathrm{LP}(u)) > h(u)$ and since $P(L)$ is finite, there is $m \in \mathbb{N}$ with $\mathrm{LP}^m(u) = \max(L)$. Let $v_0 := u$ and for $i \in \{1, \dots, n\}$ let $v_i := \mathrm{LP}^i(u)$. Then

$$(u = v_0, v_1, \dots, v_m = \max(L))$$

is a path from u to v.
Edges: It holds that $E_{\mathrm{LP}} = \{\{u, \mathrm{LP}(u)\} \mid u \in P(L) \setminus \{\max(L)\}\}$. Also,

$$\{u, \mathrm{LP}(u)\} = \{w, \mathrm{LP}(w)\} \Rightarrow u = w$$

since $u < \mathrm{LP}(u)$ and $w < \mathrm{LP}(w)$. Thus,

$$|E_{\mathrm{LP}}| = |\{\{u, \mathrm{LP}(u)\} \mid u \in P(L) \setminus \{\max(L)\}\}| = |\{u \mid u \in P(L) \setminus \{\max(L)\}\}| = n-1.$$

□

The uniqueness of the line parent is only violated in two cases.

- There are multiple peaks being reached after the same key col with the exactly same distance to the key and the same height. In such a case, we can enforce the uniqueness by sampling one of the peaks.
- There are key cols $c_1, \dots, c_n$ with corresponding higher peaks $p_1 \dots p_n$ with the exact same distance to the point. In such a case, we choose p_i such that $d(c_i, p_i)$ is minimal. If these minimum is reached multiple times, we enforce uniqueness by sampling one of the higher peaks with minimal distance to the corresponding key col.

The simple edge structure of trees enables a satisfactory visualization even for medium sized node sets. The line parent tree can also be used to study dominance relationships with a non peak as a starting point. In this case, we suggest to navigate through the line parent tree starting with the closest peak with respect to the given metric.

Algorithm 4.1: g2rng: Computing RNGs from Graphs

```
1 Input:  Graph $G = (V, E, w)$ with shortest path metric $d_{\mathrm{SP}}$.
  Output: $\mathrm{RNG}(G) = (V, E_{\mathrm{RNG}(d_{\mathrm{SP}})})$.

1  $\mathrm{RNG}(G) \leftarrow G$
2  FOR $\{v_1, v_2\}$ IN $E_{\mathrm{RNG}(d_{\mathrm{SP}})}$ :
3      $d \leftarrow d_{\mathrm{SP}}(v_1, v_2)$
4      FOR $v_3$ IN $V$ :
5          IF $d_{\mathrm{SP}}(v_1, v_3) < d$ AND $d_{\mathrm{SP}}(v_2, v_3) < d$ :
6              $E_{\mathrm{RNG}(d_{\mathrm{SP}})} \leftarrow E_{\mathrm{RNG}(d_{\mathrm{SP}})} \setminus \{\{v_1, v_2\}\}$
7              BREAK
8  RETURN $\mathrm{RNG}(G)$
```

4.5 Discarding Edges via Relative Neighborhood Graphs

Let $G = (V, E, w)$ be a weighted graph where smaller weights stand for smaller connections and let $h\colon V \to \mathbb{R}_{\geq 0}$ be a height function. One way to extend this to a landscape is given via the shortest path distance as in Definition 3.3. In practical applications, the amount of peaks will often be very low. One reason for this is the huge amount of connections one may have in social networks. Let us for example assume to have a weighted co-follower graph (for example weighted with Jaccard-distance) where the height function is given by the amount of followers. Here, all pairs of users with just one common follower would be connected and thus nearly all users would have a "higher" neighbor and thus will not be peaks. Hence, it is of interest to only keep edges which stand for a strong connection, i.e., edges between users with a large amount of common followers.

A straight-forward way to remove edges would be by either choosing a $k \in \mathbb{N}$ and keep for all vertices only the k edges with the smallest weights or to choose a $t \in (0, 1)$ and remove all edges with weights higher than t. However, besides the disadvantage that in both cases a parameter has to be chosen, this procedure can lead to disconnected graphs. Restricting to the biggest connected component of the resulting graph would then lead to the discarding of whole regions of the graph. To this end, we develop in the following a parameter free, deterministic edge sampling approach which guarantees to preserve connectivity. This approach is based on the relative neighborhood graph (RNG) [147] as defined in Definition 2.11.

In Theorem 2.12, we have already shown that the RNG is always connected. What still has to be proven is that deriving RNGs from the shortest-path metric is indeed an edge-reduction technique, i.e., that edges are just removed and no new edges are added.

Theorem 4.11 (RNG as Edge-Reduction) *Let $G = (V,E,w)$ be a connected, undirected and weighted graph and $d_{\mathrm{SP}} : V \times V \to \mathbb{R}_{\geq 0}$ be the shortest path metric on G. Then it holds that*

$$E_{\mathrm{RNG}(d_{\mathrm{SP}})} \subseteq E.$$

Proof. Let $u,v \in V$ such that $\{u,v\} \notin E$. We will show that $\{u,v\} \notin E_{\mathrm{RNG}(d_{\mathrm{SP}})}$. As u,v are not adjacent there must exist a shortest path $p = (u = v_0,\dots,v_n = v)$ with $n \geq 2$. As p is a shortest path we have $d(v_{i-1},v_i) = w(v_{i-1},v_i)$ for $i \in \{1,\dots n\}$. We thus have

$$d(u,v) = \sum_{i=1}^{n} d(v_{i-1},v_i) > d(v_0,v_1) = d(u,v_1).$$

As $(v_1,\dots,v_n = v)$ is a shortest path between v_1 and v we also have

$$d(v_1,v) = \sum_{i=2}^{n} d(v_{i-1},v_i) < \sum_{i=1}^{n} d(v_{i-1},v_i) = d(u,v).$$

Thus, we have shown that $\{u,v\} \notin E_{\mathrm{RNG}(d_{\mathrm{SP}})}$. □

In the following, we use for a given graph G the term *relative neighborhood graph of G*, denoted by RNG(G), which will always refer to the RNG with respect to the shortest-path-metric. A sketch of an edge-reduction on a graph is given as part of Figure 4.1.

For a graph $G = (V,E,w)$ with a height function h, our standard procedure is to

1. compute the shortest path metric d_{SP},
2. compute RNG(G),
3. derive from this the following landscape.

Definition 4.12 (Essential Landscape)
Let $h : V \to R_{\geq 0}$ be a height function on a weighted graph $G = (V,E,w)$. Let d_{SP} be the shortest path metric on G. We call

$$L(G,h) := (\mathrm{RNG}(G), d_{\mathrm{SP}}, h)$$

the *(essential) landscape* of G and $\mathrm{MG}(G,h) := \mathrm{MG}(L(G,h))$ the *(essential) mountain graph* of G. We call $\mathrm{LP}(G,h) := \mathrm{LP}(L(G,h))$ the *(essential) line parent tree* of G.

If clear from the context, we omit h and write MG(G) and LP(G).

Table 4.1: Network statistics. We display from left to right: 1.) the number of vertices of the network, 2.) its density 3.) the density of the RNG 4.) the number of vertices of the mountain graph, 5.) the density of the mountain graph, 6.) the number of vertices in the line parent tree, 7.) its maximum width and 8.) its depth. Then we show the node sizes and degrees of the sampled graphs serving a) as a comparison for discarding edges via the RNG procedure and b) for serving as a comparison for the mountain graph which is computed from the RNG. For the latter, we apply the sampling baselines on the RNG, not on the original network itself.

									RNG Baselines				MG Baselines			
									ES		CNARW[101]		RPN[95]		RCMH[100]	
	$\|V\|$	D_G	$D_{\mathrm{RNG}(G)}$	$\|V_{\mathrm{MG}}\|$	D_{MG}	$\|V_{\mathrm{LP}}\|$	W_{LP}	DP_{LP}	$\|V\|$	D_G	$\|V\|$	D_G	$\|V\|$	D_G	$\|V\|$	D_G
Twitter>10K	6635	.9958	.0005	1171	.1089	652	88	20	5192.8	.0008	5174	.0007	271.1	.0079	1171	.0021
Twitter>100K	430	1.0000	.0064	146	.1084	84	14	13	374.8	.0084	325.5	.0112	35.9	.0652	146	.0168
ECML/PKDD	742	.0123	.0052	190	.1957	98	21	10	650.7	.0067	560	.0092	77.1	.0304	190	.0153
KDD	1674	.0100	.0036	219	.2236	115	30	8	1575.5	.0041	1407.5	.0041	67.2	.0390	219	.0138
PAKDD	889	.0124	.0054	132	.2155	67	27	5	814.7	.0064	704.1	.0087	34.3	.0812	132	.0222

Computing the RNG of a Graph. The naive approach to compute the RNG for a finite metric space (M,d) would be to check for all pairs $m_1 \neq m_2 \in M$ whether there exists m_3 which is closer to both of them. This results in an algorithm with runtime $\mathcal{O}(|M|^3)$ [147]. For $\mathbb{R}^d$ with a l_p metric, there are algorithms with better runtime [147, 1, 79]. However, these results can not be applied to shortest path metrics. To compute $\mathrm{RNG}(G)$ for a graph $G = (V,E)$, we can use Theorem 4.11 to speed up the computation as we only have to check the elements of E and not all node pairs. The resulting algorithm *g2rng* is displayed in Algorithm 4.1 and runs in $\mathcal{O}(|E||V|)$.

4.6 Mountain Graphs and Line Parents of Real-World Networks

We experiment with networks built from a Twitter follower network [93, 17, 18] which we found at SNAP [96] and with networks that display co-author relations.

Follower Networks. From the Twitter dataset, we derive two weighted co-follower networks. In these networks two users have an edge if they have a common follower. The edges are weighted via Jaccard distance. We derive a version containing users with at least 10,000 followers (**Twitter>10K**) and a network containing users with at least 100,000 followers (**Twitter>100K**). Here, the height of a user is given via the amount of followers.

Co-Author Networks. These networks are extracted from the *Semantic Scholar Open Research Corpus* [4]. The co-author networks are derived by considering communities of authors that regularly publish at a specific conference. We derive datasets for the *European*

(a) Twitter>100K **(b)** Twitter>10K

Figure 4.3: Examples of line parent trees.

Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (**ECML/PKDD**), the *SIGKDD Conference on Knowledge Discovery and Data Mining* (**KDD**) and the *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (**PAKDD**). The heights of the authors are given via h-indices.

To weaken the influence of old publications and social connections that not exist anymore, we only use publication with a “year” attribute between 2010 and 2020. To identify the authors for a specific venue we identify authors with publications at the specific venue. To discard authors which are only weakly connected to the community, we only choose authors which are co-author of at least 2 publications at the specific venue. We generate an edge between two authors if they have at least one joint publication. Here, the edges between a_1, a_2 are weighted via Jaccard-distance using the set of publications authored between 2000 and 2020. As the height function we choose the h-index. For this, we again only consider publications and citations which are dated between the years 2000 and 2020.

Connectivity and Vanishing Distances. When generating the co-author and co-follower networks, two problems occur:

- **Connectivity.** The networks may not be connected. To solve this issue, we restrict them to the biggest connected component with respect to the amount of vertices.

- **Vanishing Edges.** It may happen that an edge has weight 0 which results in the shortest path distance not being a metric. This happens for example if two Twitter users are followed by exactly the same persons. This problem can be solved by replacing zero weights by minimal weights: We first choose the minimal positive edge weight $w_0 > 0$. Then, if an edge has weight zero, we replace its weight with $\frac{w_0}{2}$. In this way, the zero edges are still the edges with the minimal weight but the shortest path distance is a metric.

At the end of this chapter, we display the labeled trees for the co-author networks. In the brackets, we display the height and prominence of each author. For all graphs, we use well-established measures to get further insights into our notions. To be more detailed, we display node sizes and densities of the networks themselves, the RNGs and the mountain graphs derived from the RNGs. Additionally, we display node sizes, maximum widths and depths of the line parent trees derived from the RNGs. The results can be found in Table 4.1. At the end of this chapter, one can found the line parent trees of all three co-author networks.

4.6.1 Comparison with Sampling Approaches

To further understand the steps of our approach, we compare them with commonly used sampling approaches [101, 95, 100]. To be more specific, we sample edges from the original network to get graphs which have an equal amount of edges as the RNG. Then we take the biggest connected component of these graphs. We call these methods the **RNG Baselines**. We use two sampling approaches: First, we sample edges with the probability of an edge e to be chosen being proportional to $1 - w(e)$, where $w(e)$ is the weight of the edge[3]. We call the resulting baseline the *Edge Sampling* (ES) approach. As a second comparison, we use a weighted version of *CNARW* [101], a modern random walk approach.

Additionally, we use sampling approaches to sample from the RNG in such a way, that we have an equal amount of nodes as in the mountain graph and take the biggest component of the resulting network. We call these methods the **MG Baseline**. First, we sample nodes by their PageRank value via *RPN* [95]. Again, we also use a modern random walk based approach, namely *RCMH* [100]. The CNARW method used above relies on common neighbors. Since triangles in the RNG are very uncommon (for these, 2 of the 3 corresponding edges in the original graph need to have the same distance weight), we use RCMH instead.

Note, that we use the comparison with other methods to contextualize our novel structures. As our structures have a different purpose, namely displaying important connections that are derived from the original network, they are not directly comparable to regular sampling approaches. These approaches derive small graphs that behave similar to the original graph with respect to specific measures. This makes it unreasonable to interpret the comparison to our baselines as a competition where higher/lower node sizes or densities are, in some way, better. As our comparison methods include random sources, we repeat them 10 times and report means. Statistics, including sizes of the derived RNGs, mountain graphs and line parent trees, can be found in Table 4.1. Additionally, we include node sizes and densities for all comparison approaches.

[3]We use $1 - w(e)$ instead of $w(e)$ because we assume edge weights to be distances, not similarities.

Table 4.2: Mean, median and maximum of the minimal shortest path distance (MSPD) from all non-peaks v to the set P of all peaks (left) and to the set H which contains the $|P|$ highest nodes of the network (right).

	$d(P)_{\text{Mean}}$	$d(P)_{\text{Median}}$	$d(P)_{\text{Max}}$	$d(H)_{\text{Mean}}$	$d(H)_{\text{Median}}$	$d(H)_{\text{Max}}$
Twitter>10K	0.87	0.88	1.00	0.90	0.91	1.00
Twitter>100k	0.86	0.87	0.97	0.92	0.95	0.99
ECML	1.28	1.00	2.98	1.74	1.91	4.91
KDD	1.30	1.00	2.95	1.83	1.95	3.90
PAKDD	1.46	1.00	3.78	1.94	1.96	4.90

We observe that computing the RNG reduces the density by a large margin. It stands out, that this effect is stronger for the dense Twitter networks. When sampling an equal amount of edges, there are nodes which do not belong to the biggest connected component anymore. This results in a higher density of the (biggest component) of the networks created by sampling compared to the RNG.

The resulting line parent trees are much smaller than the original network, reducing the node set by a factor of about 5 to 10 times. Another remarkable point is that the mountain graph is always denser than the RNG from which it is computed and than the graphs which are sampled via the comparison methods. An explanation is that edges from a peak v to a col u in the mountain graph correspond to paths (not edges!) in the RNG. As the amount of paths in a graph is commonly remarkably higher than the amount of edges, this could be one reason for the higher density of the mountain graph.

4.6.2 Distances to Line Parent Trees

To investigate to which extent line parent trees are representative for the structure of the whole network, we study how "dense" the line parent trees lay in the networks by computing the shortest path lengths from all non-peaks to the peaks. To evaluate whether choosing locally outstanding nodes lead to a better representation than choosing nodes solely based on their height, we compare our approach with a "naive" approach of assuming the n highest points to be relevant, where n is the amount of peaks. To be more detailed, we compute for each non-peak v the minimal shortest path distances (MSPD) to all peaks in the original graph G. We do the same using the n highest points instead of the set of peaks. We report means, medians and maximum values over the MSPDs of all non-peaks, the results are in Table 4.2.

Our results show that locally outstanding nodes better reflect the overall network then just choosing the highest nodes, with median and mean values of the MSPDs being fundamentally lower. Furthermore, median MSPDs to peaks are always not higher then 1. In contrast,

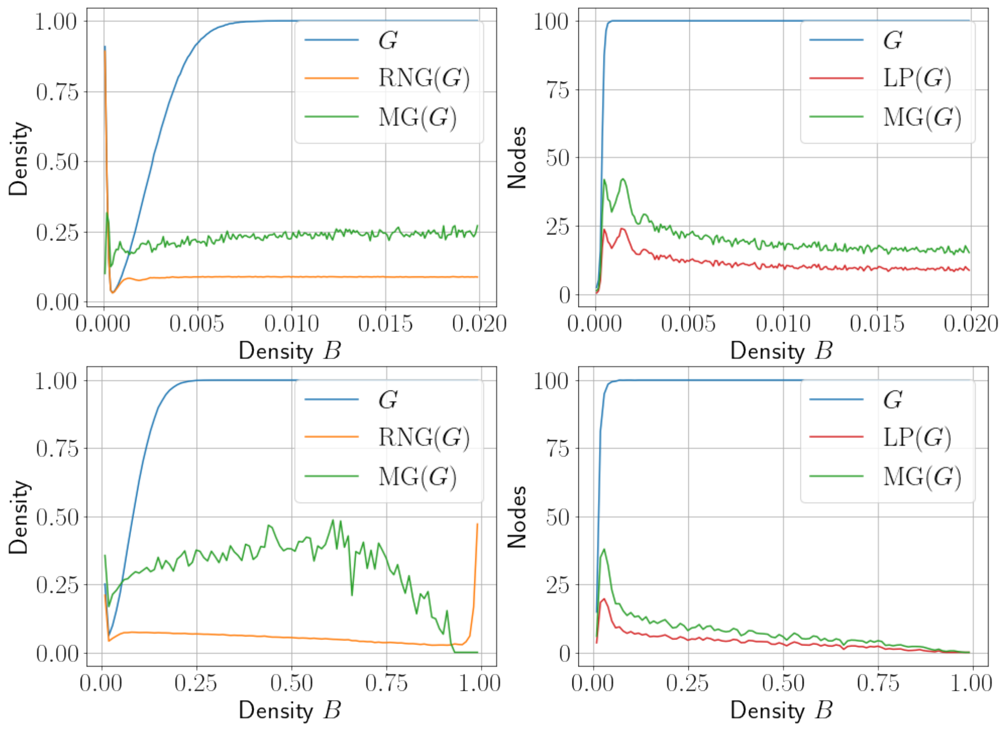


Figure 4.4: Experiments on random data. In both rows, the set M_1 on which is projected has size 100. The other set has size $100,000$ on the first row and size 100 in the second row. The x-axes display the densities of the original bipartite graph B. The left plots display the densities for the resulting network G, which is the biggest connected component of the weighted projection, the line parent tree $\mathrm{RNG}(G)$ and the mountain graph $\mathrm{MG}(G)$. The right pictures plot the node size of $G, \mathrm{MG}(G)$ and $\mathrm{LP}(G)$.

MSPDs to the "highest" nodes have median values of nearly 2 for the sparse co-author networks. This indicates that selecting locally outstanding points indeed lead to a more reasonable representation instead of selecting nodes solely by their hight, ignoring spatial information. Note, that we compute distances in a weighted graph. Hence, shortest path distances (and thus medians and maxima over them) do not have to be integers.

4.7 Experiments on Random Data

To investigate sizes and densities of the RNG, the mountain graph and the line parent tree, we additionally experiment with randomly generated data. Here, we start with a randomly generated bipartite graph with vertex sets M_1, M_2 with $|M_1| = 100$ and $|M_2| \in \{100000, 100\}$. We then project on the vertex set M_1 and weight with the Jaccard distance. The graph G is then given via the biggest connected component of this graph. As height function, we map each vertex to the amount of neighbors in the original bipartite network. This procedure is motivated by the background of often investigated real-world networks. Co-author networks are for example projection from the bipartite author-publication graph. Here, the corresponding height function then would be the amount of papers of an author, where each author is connected to multiple publications but only a small amount of the overall publications. This leads to a small density of the bipartite network.

We generate networks for different densities d. Namely, we iterate d through the set $\{0.0001, 0.0002, \ldots, 0.01999\}$ for $|M_2| = 100,000$ and through $\{0.01, 0.02, \ldots 0.99\}$ for $|M_2| = 100$. The experiments with different sizes of $|M_2|$ allow us to investigate if our methods behave fundamentally different for networks of different kinds. From G we compute the $\text{RNG}(G)$, the mountain graph $\text{MG}(G)$ and the line parent hierarchy $\text{LP}(G))$ of the essential landscape. For each d and $|M_2|$, we repeat this procedure 20 times and display means. The results can be found in Figure 4.4. The following facts stand out.

- The density of both the RNG and the mountain graph of the RNG are growing in a significant smaller pace than the density of G. Considering the case $|M_2| = 100,000$ it is remarkable, that, when G has a density of nearly 1, the density of both other graphs are still under 0.3.

- The characteristic points for describing the resulting mountain landscape build indeed a subset that is remarkably smaller than the vertex size of the biggest component of G.

- Considering the second row, it stands out that for very high densities of nearly 1 of the original bipartite network, the density and thus the amount of edges of the RNG start to rise rapidly. We assume that this is driven by the case, that, if the bipartite graph is nearly complete, there will be a large amount of vertex pairs with the same neighbor set in the bipartite graph. Thus, nearly all shortest path distances are equal and just very few edges will be discarded. In consequence, the mountain graph is built from a nearly complete graph where nearly all edges have similar weights. Thus, there will be only a small amount of peaks and the mountain graph is nearly vanishing.

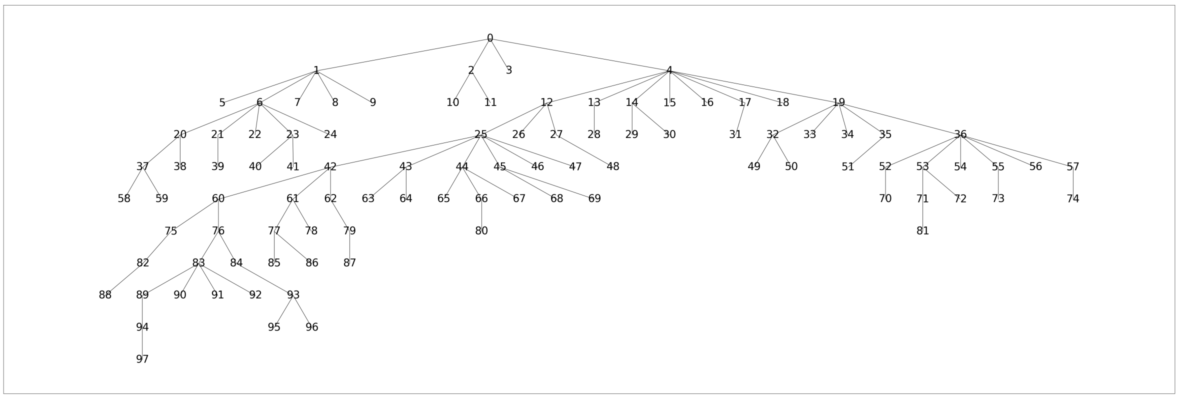
Note, that we use a height function that is directly derived from the graph. Such height functions are indeed reasonable. For example, the amount of followers of Twitter users in co-follower graphs is indeed a useful indicator of their importance.

4.8 Conclusion and Future Work

In this chapter, we showed how the notions of peaks, cols and line parents, which are originally designed to characterize connections and hierarchies between mountains, can be adapted to network analysis. We discussed how these notions can be used to identify meaningful connections and hierarchies between locally outstanding vertices. This leads to novel insights into networks from a local perspective. Our method further benefits from a novel preprocessing procedure that removes unimportant edges without hurting the connectivity of the original network.

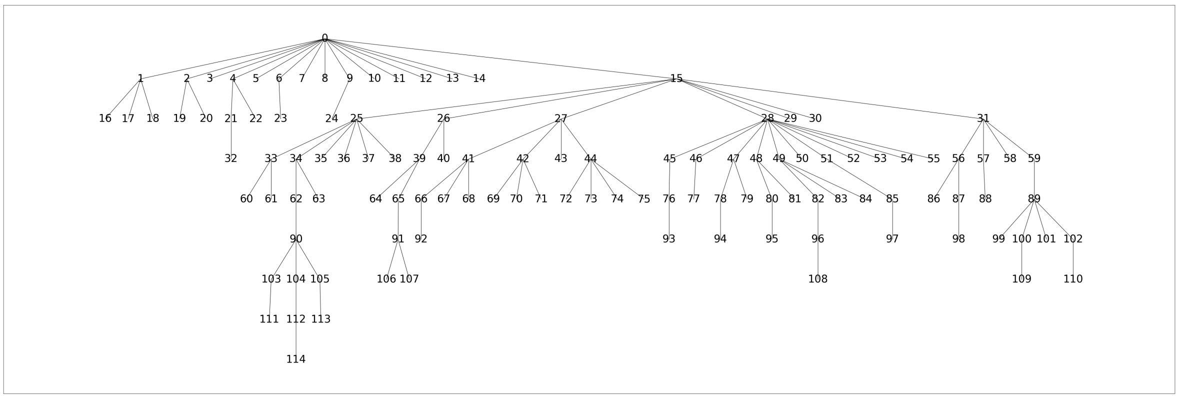
Our experiments indicate that our method finds dependencies and hierarchies from the original network that are remarkably smaller than the original graph and therefore can enhance the comprehension of real-world social networks.

Future work will investigate the application to further kinds of networks such as friendship networks on Facebook. As some of these networks may be unweighted, the question arises on how to use our RNG procedure in this case. On the other hand, it would be interesting to involve temporal aspects in networks, i.e., how the line parent hierarchy of social networks change over time.



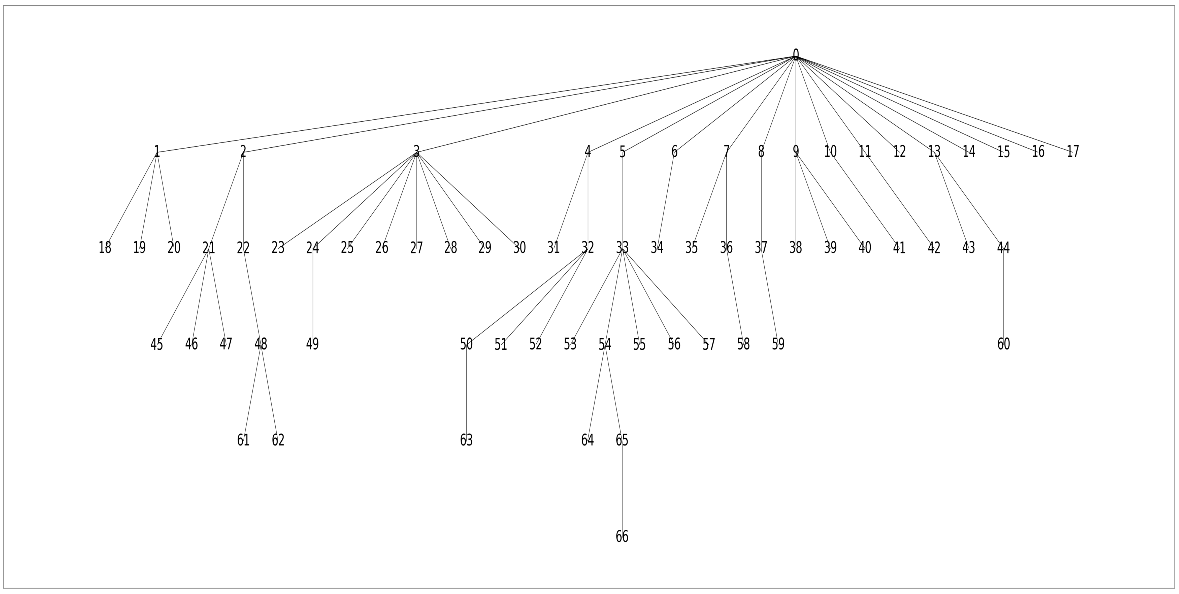
id	name	id	name	id	name	id	name
0	Yoshua Bengio (154, 154)	25	Zhi-Hua Zhou (90, 36)	50	Klemens Böhm (29, 2)	75	Shie Mannor (56, 17)
1	Alexander J. Smola (100, 36)	26	Congfu Xu (14, 2)	51	Kathryn B. Laskey (30, 15)	76	Andreas Hotho (51, 2)
2	Jürgen Schmidhuber (77, 20)	27	S. Muthukrishnan (67, 21)	52	Ahmed AbdelWahab (22, 3)	77	Richard Nock (30, 5)
3	Marc Toussaint (37, 2)	28	Volker Tresp (42, 22)	53	Gennady L. Andrienko (52, 7)	78	Aapo Hyvärinen (49, 30)
4	Jiawei Han (139, 85)	29	Nikunj C. Oza (22, 2)	54	Tomás Pevný (29, 24)	79	Marcos André Gonçalves (44, 19)
5	Christian Wolf (31, 6)	30	Peter J. Stuckey (43, 4)	55	Eyke Hüllermeier (53, 15)	80	Jakub Marecek (13, 9)
6	Zoubin Ghahramani (94, 41)	31	Marko Jereminov (11, 1)	56	Pavlos Protopapas (24, 4)	81	Irina Rish (26, 14)
7	Bernd Bischl (27, 2)	32	Quan Z. Sheng (44, 2)	57	Céline Robardet (24, 2)	82	Bernard De Baets (54, 16)
8	Devis Tuia (47, 19)	33	Eli Upfal (38, 11)	58	Ambuj K. Singh (35, 22)	83	Saso Dzeroski (48, 7)
9	Qian Zhang (79, 23)	34	Iordanis Koutsopoulos (29, 9)	59	Pieter Libin (9, 5)	84	Osmar R. Zaïane (47, 10)
10	Marco Zaffalon (26, 7)	35	Fabio Crestani (33, 1)	60	Steven C. H. Hoi (59, 4)	85	Marc Boullé (17, 4)
11	Ross T. Whitaker (49, 27)	36	Luc De Raedt (55, 8)	61	Masashi Sugiyama (59, 4)	86	Boris Chidlovskii (15, 2)
12	Yong Yu (95, 18)	37	Gianluca Bontempi (49, 20)	62	Chris H. Q. Ding (64, 11)	87	Wagner Meira (40, 15)
13	Matej Oresic (75, 55)	38	Tristan Snowsill (13, 2)	63	Elizabeth S. Burnside (32, 16)	88	Tapio Pahikkala (27, 5)
14	Bhavani M. Thuraisingham (48, 2)	39	Holger Fröning (12, 9)	64	Milind Tambe (71, 37)	89	Louis Wehenkel (39, 1)
15	Jinbo Xu (40, 27)	40	Battista Biggio (35, 6)	65	Sarah Nogueira (4, 2)	90	Fabio Mercorio (15, 1)
16	Rui Chen (48, 16)	41	Jesús Manuel de la Cruz (20, 5)	66	Matthias Weidlich (37, 6)	91	Kaihua Liu (14, 8)
17	Lawrence T. Pileggi (42, 32)	42	Feiping Nie (70, 16)	67	George Tsatsaronis (32, 8)	92	Hongyu Guo (15, 4)
18	Nassir Navab (70, 61)	43	Raymond J. Mooney (72, 23)	68	Sridhar Mahadevan (37, 11)	93	Jerzy Stefanowski (33, 2)
19	Aristides Gionis (56, 9)	44	Ioannis P. Vlahavas (47, 7)	69	Markus Schedl (38, 18)	94	Patrick Gallinari (36, 14)
20	Rémi Munos (59, 23)	45	Samuel Kaski (45, 17)	70	Lena A. Jäger (8, 6)	95	Mark Last (30, 1)
21	Franz Pernkopf (24, 8)	46	Wolfgang Rhode (78, 69)	71	Francesco Calabrese (32, 13)	96	Jesús Cid-Sueiro (15, 1)
22	Kazumi Saito (26, 3)	47	Alex Alves Freitas (54, 15)	72	Elke A. Rundensteiner (43, 20)	97	Nicolas Vayatis (23, 1)
23	C. Lee Giles (70, 33)	48	Barry O'Sullivan (29, 3)	73	Mohamed Nadif (19, 11)		
24	Yuji Matsumoto (42, 19)	49	Sen Wang (35, 10)	74	Henrik Grosskreutz (17, 1)		

Figure 4.5: ECML/PKDD



id	name	id	name	id	name	id	name
0	Jiawei Han (139, 139)	29	Andrew McCallum (89, 6)	58	Ruslan Salakhutdinov (86, 47)	87	Ming Zhou (69, 19)
1	Eamonn Keogh (79, 30)	30	David Lobell (82, 47)	59	Feiping Nie (70, 10)	88	Tianyang Zhang (7, 2)
2	Tat-Seng Chua (80, 6)	31	Fei Wang (90, 7)	60	Stephen P. Boyd (88, 21)	89	Masashi Sugiyama (59, 4)
3	Louise E. Moser (23, 14)	32	Sangram Ganguly (55, 22)	61	Kota Tsubouchi (10, 4)	90	Ravi Kumar (66, 9)
4	Jason I. Hong (60, 8)	33	Wei Zhang (102, 6)	62	George Karypis (71, 11)	91	Achla Marathe (23, 5)
5	Dinggang Shen (85, 14)	34	Eric E. Schadt (102, 20)	63	Stefan Savage (70, 5)	92	Salvatore Scellato (30, 8)
6	Martin Ester (51, 7)	35	Max Welling (64, 7)	64	Benjamin J. Bachman (6, 2)	93	Klaus Ackermann (5, 2)
7	Vitaly Shmatikov (57, 22)	36	Tong Zhang (72, 15)	65	Feng Chen (46, 2)	94	Prantik Bhattacharyya (8, 2)
8	Alex Deng (12, 3)	37	Céline Robardet (24, 4)	66	Cecilia Mascolo (64, 23)	95	Hiroki Sugiura (8, 5)
9	Thomas S. Huang (115, 45)	38	Thore Graepel (44, 13)	67	Virgílio A. F. Almeida (43, 6)	96	Luc De Raedt (55, 6)
10	David L. Small (17, 4)	39	Lawrence A. Donehower (52, 8)	68	Liang-Hao Huang (11, 5)	97	Kunpeng Liu (8, 1)
11	Nick Koudas (55, 10)	40	Sam Shah (12, 3)	69	Ioana Giurgiu (9, 1)	98	Chuan Qin (32, 3)
12	Jie Wang (106, 33)	41	Wei Chen (111, 51)	70	Ming Lei (4, 2)	99	Yasuhiro Fujiwara (39, 7)
13	Cheng-Wei Wu (15, 6)	42	Zheng Liu (88, 9)	71	Yi Li (80, 38)	100	Hiroshi Nakagawa (22, 3)
14	Paul M. Thompson (137, 89)	43	Jayant Kalagnanam (27, 2)	72	Xiaoqiang Zhu (17, 1)	101	Bernhard Pfahringer (45, 5)
15	Michael I. Jordan (134, 59)	44	Yong Yu (95, 18)	73	Kewei Chen (53, 8)	102	Pierre-Alain Fouque (40, 17)
16	Bilson J. L. Campana (9, 2)	45	Yehuda Koren (49, 5)	74	Lawrence Carin (69, 2)	103	Chih-Jen Lin (56, 18)
17	Hiroshi Mamitsuka (28, 5)	46	Robert Chen (66, 3)	75	Ruiyuan Li (20, 9)	104	Duncan Watts (50, 32)
18	Evangelos E. Milios (37, 4)	47	Hans-Peter Kriegel (71, 10)	76	Joe Walsh (9, 3)	105	Vahab S. Mirrokni (50, 7)
19	Guoliang Li (62, 4)	48	Hiroshi Sawada (41, 18)	77	Fanjin Zhang (4, 2)	106	Graham Katz (18, 10)
20	Evangelos Kanoulas (27, 8)	49	Eric Horvitz (80, 8)	78	Souvik Ghosh (12, 4)	107	Jaime Arredondo (13, 4)
21	Dimitrios Gunopulos (58, 6)	50	Matthias Grossglauser (29, 8)	79	Michael Mitzenmacher (66, 7)	108	Anthony J. Bagnall (29, 15)
22	Ed H. Chi (51, 13)	51	João Gama (46, 3)	80	Ryo Asaoka (22, 19)	109	Yukihiro Tagami (7, 3)
23	Yong Ge (34, 6)	52	Prashant J. Shenoy (67, 19)	81	Ken-ichi Kawarabayashi (37, 14)	110	Christine Largouët (9, 7)
24	Murali Ramanathan (41, 2)	53	Diane J. Cook (54, 7)	82	Pedro Domingos (76, 15)	111	Wei Di (19, 4)
25	Bernhard Schölkopf (119, 32)	54	Andrew J. Connolly (69, 25)	83	Yina Tang (6, 2)	112	Yuandong Tian (26, 8)
26	Inderjit Dhillon (71, 26)	55	Jennifer Widom (69, 8)	84	Rodica Pop-Busui (47, 44)	113	Adam D. Smith (47, 5)
27	Anil K. Jain (130, 53)	56	Xiaoming Li (84, 19)	85	Zimu Zhou (34, 7)	114	Estevam R. Hruschka (19, 2)
28	Christos Faloutsos (107, 20)	57	Yunfei Lu (9, 4)	86	Wei Wang (75, 14)		

Figure 4.6: KDD



id	name	id	name	id	name	id	name
0	Jiawei Han (139, 139)	17	Chun Chen (58, 7)	34	Felix Cheung (8, 4)	51	Lior Rokach (52, 15)
1	Christos Faloutsos (107, 52)	18	Ramayya Krishnan (39, 5)	35	Manuel Campos (17, 9)	52	Gang Li (40, 6)
2	Yong Yu (95, 24)	19	Alex Beutel (31, 15)	36	Sadok Ben Yahia (25, 3)	53	Svetha Venkatesh (46, 11)
3	Zhi-Hua Zhou (90, 35)	20	Nan-Chen Hsieh (8, 1)	37	Niloy Ganguly (33, 3)	54	Manabu Okumura (27, 13)
4	Wei Zhang (102, 35)	21	Xiaoming Li (84, 6)	38	Shen Furao (18, 7)	55	Houkuan Huang (20, 2)
5	Eamonn J. Keogh (79, 30)	22	Qian Zhang (79, 6)	39	Elaheh ShafieiBavani (8, 1)	56	Si Quang Le (12, 3)
6	Minghua Zhang (19, 7)	23	Meng Chen (48, 13)	40	Xin Zuo (5, 3)	57	Tharshan Vaithianathan (13, 4)
7	Francesco Bonchi (47, 7)	24	Yusuke Suzuki (37, 26)	41	Samir Mustapha (13, 4)	58	Maguelonne Teisseire (24, 2)
8	Cyrus Shahabi (57, 11)	25	Hao Chen (34, 5)	42	Chris H. Q. Ding (64, 4)	59	Tanya Goyal (12, 3)
9	Chen Wang (71, 6)	26	Hiroyuki Toda (29, 10)	43	Gianluigi Greco (26, 10)	60	Wayne Wobcke (14, 5)
10	Boualem Benatallah (54, 8)	27	Mineichi Kudo (17, 8)	44	Nada Lavrac (41, 8)	61	Md. Rafiul Hassan (14, 5)
11	George Karypis (71, 11)	28	Alexandros Kalousis (26, 13)	45	Chunmei Dong (7, 2)	62	Viktor K. Prasanna (53, 5)
12	Osmar R. Zaïane (47, 10)	29	Ulf Johansson (18, 6)	46	Claudia Eckert (29, 17)	63	Yun Xiong (15, 1)
13	Irwin King (58, 2)	30	Tony F. Chan (65, 47)	47	Hsin-Min Wang (29, 3)	64	Sanparith Marukatat (13, 5)
14	Xueping Zhang (5, 1)	31	Peng Wang (99, 17)	48	Albert Y. Zomaya (66, 18)	65	Kazumi Saito (26, 11)
15	Luc De Raedt (55, 25)	32	Yiyu Yao (63, 25)	49	Kenichi Takahashi (11, 1)	66	Junichiro Mizusaki (15, 6)
16	Tamás D. Gedeon (32, 8)	33	Geoffrey I. Webb (48, 1)	50	Yaqin Wang (25, 11)		

Figure 4.7: PAKDD

Part III

Intrinsic Dimensionality

Chapter 5

Intrinsic Dimensionality for Large-Scale Geometric Learning

In this chapter, we study networks from a *global perspective* by investigating a quantity that evaluates the complexity of a whole network. This quantity aims to display to which extent a dataset is effected by the *curse of dimensionality*. We will discuss this notion in the abstract setting of *geometric datasets* and will later discuss how this abstract setting can be applied to real-world network data.

5.1 Introduction

Real-world networks are often large in size and comprised of complex structures, which distinguishes them from Euclidean data. To consider these properties appropriately is a challenging task for procedures that analyze or learn from said data. Moreover, with increasing complexity of real-world datasets, the necessity arises to quantify to which extent this data suffers from the curse of dimensionality. The common approach for estimating the dimension curse of a particular dataset is through the notion of *intrinsic dimension* (ID) [7, 62, 122]. There exists a variety of work on how to estimate the ID of datasets [50, 97, 38, 61, 7]. Most approaches to quantify the ID are based on distances between data points, assuming the data to be Euclidean. A multitude of works base their modeling on the manifold hypothesis [33, 61], which assumes that the observed data can be embedded in a manifold of low dimension (compared to the number of data attributes). Based on this, the ID is an approximation of the dimension of this manifold. Pestov [121] proposed a different concept of intrinsic dimension by linking it to the mathematical concentration of measure phenomenon. His modeling is based on a thorough axiomatic approach [122–124] which resulted in a novel class of intrinsic dimension functions. In contrast to the manifold hypothesis, Pestov's ID functions measure to which extent a dataset is affected by the curse of dimensionality, i.e., to which extent the complexity of the dataset hinders the discrimination of data points. Yet, to compute said ID functions is an intractable computational endeavor. This limitation was overcome in principle by an adaptation to geometric datasets [70]. However, two problems persisted: First, the computational effort was found to remain quadratic in the number of data points, which is insufficient for datasets of contemporary size; second, it is unclear how to account for complex structure, such as in graph data.

With this in mind, we propose in the present chapter a default approach for computing the intrinsic dimension of geometric data, such as graph data. To do this, we revisit the computation of the ID based on distance functions [70] and overcome, in particular, the inherent computational limitations in the works by Pestov [122] and Hanika et al. [70]. In detail, we derive a novel approximation formula and present an algorithm for its computation. This allows us to compute ID bounds for datasets that are magnitudes larger than in earlier works. That equipped with, we establish a natural approach to compute the ID of networks.

We apply our method to seven real-world datasets and relate the obtained results to performances of classification procedures. Thus, we demonstrate the practical computability of our approach. In addition, we study the extent to which the intrinsic dimension reveals insights into the performance of particularly classes of graph neural networks. With that, we build a bridge between network complexity from a global view and graph learning.

5.2 Related Work

In numerous works, the intrinsic dimension is estimated using the pairwise distances between data points [31, 63]. More sophisticated approaches use distances to nearest neighbors [50, 97, 38, 61]. All these works have in common, that they assume the data to be Euclidean and that they favor local properties.

Recent work has drawn different connections between intrinsic dimension (ID) and modern learning theory. For example, Cloninger and Klock [33] show that functions of the form $f(x) = g(\phi(x))$, where ϕ maps into a manifold of lower dimension, can be approximated by neural networks. On the other hand, Wojtowytsch et al. [156] prove that modern artificial neural networks suffer from the curse of dimensionality in the sense that gradient training on high dimensional data may converge insufficiently. Additional to these theoretical results, there is an increasing interest of empirically estimating the ID of contemporary learning architectures. Li et al. [98] study the ID of neural networks by replacing high dimensional parameter vectors with lower dimensional ones. Their approach results in a non-deterministic

ID. More recent works studied ID in the realm of geometric data and their standard architectures. Ansuini et al. [5] investigate the ID for convolutional neural networks. In detail, they are interested to which extent the ID changes at different hidden layers and how this relates to the overall classification performance. Another work [125] associates an ID to popular benchmark image datasets. These two works on ID estimators do solely rely on the metric information of the data and do not consider any geometric structure of image data.

Our approach allows to incorporate such underlying geometric structures while incorporating the mathematical phenomenon of measure concentration [64, 109, 110]. Linking this phenomenon to the occurrence of the dimension curse was done by Pestov [121–124]. He based his considerations on a thorough axiomatic approach using techniques from metric-measure spaces. The resulting ID functions unfortunately turn out to be practically incomputable. In contrast, Bac and Zinovyev [7] investigate computationally feasible ID estimators that are related to the concentration phenomenon. Yet, their results elude a comparable axiomatic foundation. Our modeling for the ID of large and geometric data is based on Hanika et al. [70]. We build on their axiomatization and derive a computationally feasible method for the intrinsic dimension of large-scale geometric datasets.

5.3 Intrinsic Dimensionality

Since our work is based on the formalization from Hanika et al. [70], we shortly revisit their modeling and recapitulate the most important structures. Based on this, we derive and prove an explicit formula to compute the ID for the special case of finite geometric datasets. This first result is essential for Section 5.4.

Let $\mathscr{D} = (\mathscr{X}, F, \mu)$, where $\mathscr{X}$ is a set and $F \subseteq \mathbb{R}^{\mathscr{X}}$ is a set of functions from $\mathscr{X}$ to $\mathbb{R}$, in the following called *feature functions*. This definition should not be confused with the definition of feature matrices and feature vectors as defined before Equation (2.4). We require that $\sup_{x,y\in\mathscr{X}} d_F(x,y) < \infty$, where $d_F(x,y) := \sup_{f\in F} |f(x) - f(y)|$. If $(\mathscr{X}, d_F)$ constitutes a complete and separable metric space such that μ is a Borel probability measure on $(\mathscr{X}, d_F)$, we call $\mathscr{D}$ a *geometric dataset* (GD). The aforementioned properties are satisfied when it holds that $0 < |\mathscr{X}|, |F| < \infty$ and that F can distinguish all data points, i.e., $d_F(x,y) > 0$ for all $x, y \in \mathscr{X}$ with $x \neq y$.

Two geometric datasets $\mathscr{D}_1 := (\mathscr{X}_1, F_1, \mu_1), \mathscr{D}_2 := (\mathscr{X}_2, F_2, \mu_2)$ are isomorphic if there exists a bijection $\phi : \mathscr{X}_1 \to \mathscr{X}_2$ such that $\overline{F_2} \circ \phi = \overline{F_1}$ and $\phi_*(\mu_1) = \mu_2$, where $\phi_*(\mu_1)(B) := \mu_1(\phi^{-1}(B))$ is the *push-forward measure* and the closures are taken with respect to pointwise convergence. From this point on we identify a geometric dataset with its isomorphism class. The triple $(\{\emptyset\}, \mathbb{R}, \nu_{\{\emptyset\}})$ represents the *trivial geometric dataset*.

Pestov [122] defines the curse of dimensionality as *"[...] a name given to the situation where all or some of the important features of a dataset sharply concentrate near their median (or mean) values and thus become non-discriminating. In such cases, X is perceived as intrinsically high-dimensional."* Thus, the ID estimator aims to compute to which extent the features allow to discriminate different data points. For a specific feature $f \in F$, Hanika et al. [70] therefore defines the *partial diameter* of f with regard to a specific $\alpha \in (0,1)$ such that it displays to which extent f can discriminate subsets with minimal measure $1-\alpha$, i.e., via

$$\mathrm{PartDiam}(f_*(\nu), 1-\alpha) = \inf\{\mathrm{diam}(B) \mid B \subseteq \mathbb{R} \text{ Borel}, \nu(f^{-1}(B)) \geq 1-\alpha\},$$

where $\mathrm{diam}(B) := \sup_{a,b\in B} |a-b|$. The *observable diameter* with respect to α then defines to which extent F can discriminate points with minimum measure $1-\alpha$. It is defined as the supremum of the partial diameter of all $f \in F$, i.e,

$$\mathrm{ObsDiam}(\mathscr{D}, -\alpha) := \sup_{f\in F} \mathrm{PartDiam}(f_*(\mu), 1-\alpha).$$

To observe the discriminability for different minimal measures α, the *discriminability* $\Delta(\mathscr{D})$ of $\mathscr{D}$ and the *intrinsic dimension* $\partial(\mathscr{D})$ are defined via

$$\partial(\mathscr{D}) := \frac{1}{\Delta(\mathscr{D})^2}, \quad \text{where} \quad \Delta(\mathscr{D}) := \int_0^1 \mathrm{ObsDiam}(\mathscr{D}, -\alpha)\, d\alpha. \tag{5.1}$$

In other words, lower values of intrinsic dimensionality correspond to geometric datasets with points that can be better discriminated by the given set of feature functions. This intrinsic dimension function is, in principle, applicable to a broad variety of geometric data, such as metric data, graphs or images. This applicability arises from the possibility to choose suitable feature functions which reflect the underlying data structure. The appropriate choice of feature functions is part of Section 5.5. Furthermore, the ID

$$\partial(\mathscr{D}) = \frac{1}{\Delta(\mathscr{D})^2}$$

respects the formal axiomatization [70] for ID functions, informally:

- **Axiom of concentration:** A sequence of geometric datasets converges against the constant dataset (meaning having no chance to separate data points!), if and only if their IDs diverge against infinity.

- **Axiom of feature antitonicity:** If dataset $\mathscr{D}_1$ has more feature functions then $\mathscr{D}_0$ (i.e. having potentially more information to separate data points), it should have a lower intrinsic dimension.
- **Axiom of continuity:** If a sequence of geometric datasets converge against a specific geometric dataset, the sequence of the IDs should converge against the ID of the limit geometric dataset.
- **Axiom of geometric order of divergence:** If a sequence of geometric datasets converges against the constant dataset, its IDs should diverge against infinity with the same order as $\frac{1}{\Delta(\mathscr{D})^2}$ does.

5.3.1 Intrinsic Dimension of Finite Data

We want to apply Equation (5.1) to real-world network data. In the following, let $\mathscr{D} = (\mathscr{X}, F, \nu)$ such that $0 < |\mathscr{X}| < \infty$ and $0 < |F| < \infty$ and let ν be the normalized counting measure on $\mathscr{X}$, i.e., $\nu(M) := \frac{|M|}{|\mathscr{X}|}$ for $M \subseteq \mathscr{X}$. In this case, it is possible to compute the partial diameter and Equation (5.1), as we show in the following. Let $\alpha \in (0,1)$ and let $c_\alpha := \lceil |\mathscr{X}| \cdot (1-\alpha) \rceil$. The following arguments were already hinted in previous work [70], yet not formally discussed or proven.

Lemma 5.1 *For $f \in F$ it holds that*

$$\operatorname{PartDiam}(f_*(\nu), 1-\alpha) = \min_{|M|=c_\alpha} \max_{x,y \in M} |f(x) - f(y)|.$$

Proof. It holds that

$$\begin{aligned}\operatorname{PartDiam}(f_*(\nu), 1-\alpha) &= \inf\{\operatorname{diam}(B) \mid B \subseteq \mathbb{R} \text{ Borel}, \nu(f^{-1}(B)) \geq 1-\alpha\} \\ &= \inf\{\operatorname{diam}(B) \mid B \subseteq \mathbb{R} \text{ Borel}, |\{x \in \mathscr{X} \mid f(x) \in B\}| \geq c_\alpha\}.\end{aligned}$$

We have to show that

$$\begin{aligned}&\inf\{\operatorname{diam}(B) \mid B \subseteq \mathbb{R} \text{ Borel}, |\{x \in \mathscr{X} \mid f(x) \in B\}| \geq c_\alpha\} = \\ &\qquad \min\{\max_{x,y \in M} |f(x) - f(y)| \mid M \subseteq \mathscr{X}, |M| \geq c_\alpha\}.\end{aligned}$$

"$\leq$:" We show that

$$\{\operatorname{diam}(B) \mid B \subseteq \mathbb{R} \text{ Borel}, |\{x \in \mathscr{X} \mid f(x) \in B\}| \geq c_\alpha\} \supseteq \\ \{\max_{x,y\in M} |f(x) - f(y)| \mid M \subseteq \mathscr{X}, |M| \geq c_\alpha\}.$$

Let $z := \max_{x,y\in M} |f(x) - f(y)|$ such that $M \subseteq \mathscr{X}$ with $|M| \geq c_\alpha$. Without loss of generality we assume that

$$\forall x \in \mathscr{X} : (\exists m_1, m_2 \in M : f(m_1) \leq f(x) \leq f(m_2) \implies x \in M).$$

Let $b := \max_{x\in M} f(x)$, $a := \min_{x\in M} f(x)$, then $M = \{x \in \mathscr{X} \mid f(x) \in [a,b]\}$. Hence,

$$z = b - a \in \{\operatorname{diam}(B) | B \subseteq \mathbb{R} \text{ Borel}, |\{x \in \mathscr{X} | f(x) \in B\}| \geq c_\alpha\}.$$

"$\geq$:" Let $B \subseteq \mathbb{R}$ be Borel with $|\{x \in \mathscr{X} \mid f(x) \in B\}| \geq c_\alpha$. Furthermore, we additionally set $M := \{x \in \mathscr{X} \mid f(x) \in B\}$. It holds that

$$\operatorname{diam}(B) = \sup_{x,y\in B} |x - y| \geq \max_{x,y\in M} |f(x) - f(y)|$$

because of the choice of M. As B was chosen arbitrarily, it follows "$\geq$".

Finally, we need that

$$\min\{\max_{x,y\in M} |f(x) - f(y)| \mid |M| \geq c_\alpha\} = \min\{\max_{x,y\in M} |f(x) - f(y)| \mid |M| = c_\alpha\}.$$

"$\leq$" follows directly from the fact that

$$\{\max_{x,y\in M} |f(x) - f(y)| \mid |M| \geq c_\alpha\} \supseteq \{\max_{x,y\in M} |f(x) - f(y)| \mid |M| = c_\alpha\}.$$

"$\geq$" follows from the fact that for every $|M| \geq c_\alpha$ and for every $N \subseteq M$ with $|N| = c_\alpha$ the following equation holds: $\sup_{x,y\in M} |f(x) - f(y)| \geq \sup_{x,y\in N} |f(x) - f(y)|$. □

This lemma allows for a more tractable formula for the computation of the partial diameter of a finite GD. That in turn enables the following theorem.

Theorem 5.2 *It holds that*

$$\Delta(\mathscr{D}) = \frac{1}{|\mathscr{X}|} \sum_{k=2}^{|\mathscr{X}|} \max_{f\in F} \min_{\substack{M\subseteq\mathscr{X} \\ |M|=k}} \max_{x,y\in M} |f(x) - f(y)|. \tag{5.2}$$

Proof. Consider the function

$$g : (0,1) \to \mathbb{R}, \alpha \mapsto \max_{f \in F} \min_{M \subseteq \mathscr{X}, |M| = c_\alpha} \max_{x,y \in M} |f(x) - f(y)|.$$

Because of Lemma 5.1 we know that $\Delta(\mathscr{D}) = \int_0^1 g(\alpha)\, d\alpha$. The function g is a step function which can be expressed for each $\alpha \in (0,1)$ via

$$g(\alpha) = \sum_{k=1}^{|\mathscr{X}|} \mathbb{I}_{\left(\frac{|\mathscr{X}|-k}{|\mathscr{X}|}, \frac{|\mathscr{X}|+1-k}{|\mathscr{X}|}\right)}(\alpha) \max_{f \in F} \min_{\substack{M \subseteq \mathscr{X} \\ |M| = k}} \max_{x,y \in M} |f(x) - f(y)|$$

almost everywhere, where $\mathbb{I}$ is the indicator function. Hence, Equation (5.2) is a consequence of the definition of the Lebesgue-Integral with the fact that

$$\min_{M \subseteq \mathscr{X}, |M| = 1} \max_{x,y \in M} |f(x) - f(y)| = 0.$$

□

In general, the addition of features should lower the ID since we have additional information that helps to discriminate the data. However, there are certain features that are not helping to further discriminate data points. These are for example:

1. Constant features. This is due to the fact that for a constant feature f it always holds for all $M \subseteq \mathscr{X}$ that $\max_{x,y \in M} |f(x) - f(y)| = 0$.
2. Permutations of already existing features. Let $\tilde{f} : \mathscr{X} \to \mathbb{R}$ have the form $\tilde{f} = f \circ \pi$ with $\pi : \mathscr{X} \to \mathscr{X}$ being a permutation on $\mathscr{X}$. Then there exist for all $M \subseteq \mathscr{X}$ with $|M| = k$ a set $N \subseteq \mathscr{X}$ with $|N| = k$ and $\max_{x,y \in M} |f(x) - f(y)| = \max_{x,y \in N} |\tilde{f}(x) - \tilde{f}(y)|$ and vice versa.

Thus, we have the following Lemma.

Lemma 5.3 *Let $\mathscr{D} = (\mathscr{X}, F, \mu)$ be a finite geometric dataset. Furthermore, let $\hat{F}$ be a set of constant functions $\mathscr{X} \to \mathbb{R}$ and let $\tilde{F}$ be a set of functions $\mathscr{X} \to \mathbb{R}$ such that there exist for each $\tilde{f} \in \tilde{F}$ an $f \in F$ and a permutation $\pi : \mathscr{X} \to \mathscr{X}$ with $\tilde{f} = f \circ \pi$. Let*

$$\mathscr{E} := (\mathscr{X}, F \cup \hat{F} \cup \tilde{F}, \mu).$$

Then it holds that

$$\partial(D) = \partial(E).$$

5.3.2 Computing the Intrinsic Dimension of Finite Data

In this section we will propose an algorithm for computing the ID based on Equation (5.2). For this, given a finite geometric dataset $\mathscr{D}$, we use the shortened notations

$$\begin{aligned}\phi_{k,f}(\mathscr{D}) &:= \min_{M\subseteq\mathscr{X},|M|=k}\max_{x,y\in M}|f(x)-f(y)|,\\ \phi_k(\mathscr{D}) &:= \max_{f\in F}\phi_{k,f}(\mathscr{D}).\end{aligned}$$

Then, Equation (5.2) can be written as

$$\Delta(\mathscr{D}) = \frac{1}{|\mathscr{X}|}\sum_{k=2}^{|\mathscr{X}|}\phi_k(\mathscr{D}) = \frac{1}{|\mathscr{X}|}\sum_{k=2}^{|\mathscr{X}|}\max_{f\in F}\phi_{k,f}(\mathscr{D}). \tag{5.3}$$

The straightforward computation of Equation (5.3) is hindered by the task to iterate through all subsets $M\subseteq\mathscr{X}$ of size k. This yields an exponential complexity with respect to $|\mathscr{X}|$ for computing $\Delta(\mathscr{D})$. We can overcome this towards a quadratic computational complexity in $|\mathscr{X}|$ using the following concept.

Definition 5.4 (Feature Sequence)
For a feature $f\in F$ let $l_{f,\mathscr{D}}\in\mathbb{R}^{|\mathscr{X}|}$ be the increasing sequence of all values $f(x)$ for $x\in\mathscr{X}$. We call $l_{f,\mathscr{D}} = (l_1^{f,\mathscr{D}},\dots,l_{|\mathscr{X}|}^{f,\mathscr{D}})$ the *feature sequence* of f.

Using these sequences, the following lemma allows us to efficiently compute $\phi_{k,f}(\mathscr{D})$.

Lemma 5.5 *For $k\in\{2,\dots,|\mathscr{X}|\}, f\in F$ and $l_{f,\mathscr{D}}$, it holds that*

$$\phi_{k,f}(\mathscr{D}) = \min(\{l_{k+j}^{f,\mathscr{D}} - l_{1+j}^{f,\mathscr{D}} \mid j\in\{0,\dots,|\mathscr{X}|-k\}\}).$$

Proof. For all $j\in\{0,\dots,|\mathscr{X}|-k\}$ there exist $M\subseteq\mathscr{X}$ with $|M|=k$ and

$$l_{k+j}^{f,\mathscr{D}} - l_{1+j}^{f,\mathscr{D}} = \max_{x,y\in M}|f(x)-f(y)|.$$

Thus, it is sufficient to show

$$\phi_{k,f}(\mathscr{D}) \in \{l_{k+j}^{f,\mathscr{D}} - l_{1+j}^{f,\mathscr{D}} \mid j\in\{0,\dots,|\mathscr{X}|-k\}\}.$$

Choose $M\subseteq\mathscr{X}$ with $|M|=k$ such that $\phi_{k,f}(\mathscr{D}) = \max_{x,y\in M}|f(x)-f(y)|$ holds. Furthermore, let $l^M := (l_1^M,\dots,l_k^M)$ be the increasing sequence of values $f(m)$ for $m\in M$ and let $j\in\{0,\dots,|\mathscr{X}|-k\}$ such that $l_1^M = l_{1+j}^{f,\mathscr{D}}$. Since l^M is an ordered sequence of which each element

Algorithm 5.1: The pseudocode to compute $\Delta(\mathscr{D})$ for a finite geometric dataset $\mathscr{D} = (\mathscr{X}, F, \mu)$.

```
Input  : Finite geometric dataset $\mathscr{D} = (\mathscr{X}, F, \mu)$.
Output : $\Delta(\mathscr{D})$
1 forall $f$ in $F$ do
2     Compute feature sequence $l_{f,\mathscr{D}}$.
3 $\Delta(\mathscr{D}) = 0$
4 forall $k$ in $\{2,\ldots,|\mathscr{X}|\}$ do
5     forall $f$ in $F$ do
6         $\phi_{k,f}(\mathscr{D}) = \min_{j\in\{0,\ldots,|\mathscr{X}|-k\}} l^{f,\mathscr{D}}_{k+j} - l^{f,\mathscr{D}}_{1+j}$.
7     $\Delta(\mathscr{D}) += \max_{f\in F} \phi_{k,f}(\mathscr{D})$
8 $\Delta(\mathscr{D}) = \frac{1}{|\mathscr{X}|}\Delta(\mathscr{D})$
9 return $\Delta(\mathscr{D})$
```

is also an element of the ordered sequence $l_{f,\mathscr{D}}$, it holds that $l^M_k \geq l^{f,\mathscr{D}}_{k+j}$ and thus

$$l^M_k - l^M_1 \geq l^{f,\mathscr{D}}_{j+k} - l^{f,\mathscr{D}}_{j+1}.$$

There is an $N \subseteq \mathscr{X}$ with size k such that $\max_{x,y\in N} |f(x) - f(y)| = l^{f,\mathscr{D}}_{k+j} - l^{f,\mathscr{D}}_{k+1}$. Since $M \subseteq \mathscr{X}$ is of size k such that $\max_{x,y\in M} |f(x) - f(y)|$ is minimal, it follows

$$l^M_k - l^M_1 = \max_{x,y\in M} |f(x) - f(y)| \leq \max_{x,y\in N} |f(x) - f(y)| = l^{f,\mathscr{D}}_{k+j} - l^{f,\mathscr{D}}_{k+1}$$

and we thus have

$$\phi_{k,f}(\mathscr{D}) = \max_{x,y\in M} |f(x) - f(y)| = l^M_k - l^M_1 = l^{f,\mathscr{D}}_{k+j} - l^{f,\mathscr{D}}_{k+1}.$$

$\square$

To sum up, Lemma 5.5 enables the efficient computation of $\phi_{k,f}(\mathscr{D})$ via a sliding window, i.e., by using only pairs of elements $(l^{f,\mathscr{D}}_{1+j}, l^{f,\mathscr{D}}_{k+j})$. The algorithm based on this is shown in Algorithm 5.1. We want to provide a brief description of the most relevant steps. In Line 4 we iterate through $k \in \{2,\ldots,|\mathscr{X}|\}$ in order to compute $\phi_k(\mathscr{D})$ in Lines 6 and 7. For this we also need to iterate over all $f \in F$ (Line 5) to compute the necessary values of $\phi_{k,f}(\mathscr{D})$ in Line 6. For a given $f \in F, k \in \{1,\ldots,|\mathscr{X}|\}$, Line 6 consumes $|\mathscr{X}| - k + 1$ subtraction operations. Assuming that computing feature values can be done in constant time, the runtime for computing $\Delta(\mathscr{D})$ from the feature sequences is

$$\mathscr{O}(|F|\sum_{k=2}^{|\mathscr{X}|}|\mathscr{X}|-k+1)=\mathscr{O}(|F|\sum_{k=1}^{|\mathscr{X}|-1}k)=\mathscr{O}(|F|\,|\mathscr{X}|^2).$$

The creation of all feature sequences requires $\mathscr{O}(|F||\mathscr{X}|\log(|\mathscr{X}|))$ computations which is negligible compared to the aforementioned complexity. Thus, Algorithm 5.1 has quadratic complexity with respect to $|\mathscr{X}|$. Therefore, Algorithm 5.1 is a straightforward and easy to implement solution for the computation of the ID. However, its quadratic runtime is obstructive for the application in large-scale data problems, which raises the necessity for a modification. We will present such a modification in the following section.

5.4 Intrinsic Dimension for Large-Scale Data

In order to speed up the computation of the ID we modify Algorithm 5.1 we show that often a large amount of the computations steps conducted in Algorithm 5.1 can be skipped. Furthermore, we provide an efficiently computable approximation of the ID. This approximation consists of the following steps.

1. We approximate the ID by replacing $\{2,\ldots,|\mathscr{X}|\}$ in Line 4 of Algorithm 5.1 with a smaller subset $S\subseteq\{2,\ldots,|\mathscr{X}|\}$, which we represent by $S:=\{s_1,\ldots,s_l\}$. For all $k\notin S$, we will use $\{\phi_{s_1},\ldots,\phi_{s_l}\}$ to estimate ϕ_k. This will eventually lead to two approximations of the ID, an underestimation and an overestimation.

2. We compare the upper and lower approximation to provide an error bound of these approximations with respect to the exact ID. This error bound can be computed without knowing the exact ID.

3. We argue how the computation of the exact ID can be sped up with the help of knowledge about $\phi_{s_i}(\mathscr{D})$ for all $s_i\in S$. For this, we will in particular show that we can replace for all $k\in\{2,\ldots,|\mathscr{X}|\}\setminus M$ the set F with a subset $\hat{F}$ in Line 5-6 of Algorithm 5.1.

4. We derive a formula which estimates the amount of computation cost which is saved by using only subsets of F for the computation of the ID. This information can be used to estimate and decide whether the exact computation of the ID is computational feasible for a specific dataset.

The ensuing algorithm is shown in Algorithm 5.2. The underlying theory that justifies it is presented in the following. This theory will be based on the monotonicity of $i\mapsto\phi_{i,f}(\mathscr{D})$.

Algorithm 5.2: The pseudocode to compute $\Delta_{s,-}(\mathscr{D}), \Delta_{s,+}(\mathscr{D}), \Delta(\mathscr{D})$ for a finite GD $\mathscr{D} = (\mathscr{X}, F, \mu)$.

```
Input  : Finite GD $\mathscr{D} = (\mathscr{X}, F, \mu)$, support sequence $s = (2 = s_1, \ldots, s_l = |\mathscr{X}|)$,
         exact (Boolean)
Output : $\Delta_{s,-}(\mathscr{D}), \Delta_{s,+}(\mathscr{D}), \Delta(\mathscr{D})$
1  forall $f$ in $F$ do
2      Compute feature sequence $l_{f,\mathscr{D}}$.
3  $\Delta(\mathscr{D}) = 0$
4  $s_0 = 1$
5  $\phi_{s_0}(\mathscr{D}) = 0$
6  forall $i$ in $\{1, \ldots, l\}$ do          // Iterate over support sequence indices
7      forall $f$ in $F$ do
8          $\phi_{s_i,f}(\mathscr{D}) = \min_{j \in \{0,\ldots,|\mathscr{X}|-s_i\}} l^{f,\mathscr{D}}_{s_i+j} - l^{f,\mathscr{D}}_{1+j}$          // Use Lemma 5.5
9      $\phi_{s_i}(\mathscr{D}) = \max_{f \in F} \phi_{s_i,f}(\mathscr{D})$
10     $F_{s_{i-1}} = \{f \in F \mid \phi_{s_i,f}(\mathscr{D}) > \phi_{s_{i-1}}(\mathscr{D})\}$     // Compute $F_{s_{i-1}}$ for Lemma 5.12
11     $\Delta(\mathscr{D}) += \phi_{s_i}(\mathscr{D})$
12 $\Delta_{s,-}(\mathscr{D}) = \frac{1}{|\mathscr{X}|}(\Delta(\mathscr{D}) + \sum_{i=1}^{l-1} \sum_{s_i<j<s_{i+1}} \phi_{s_i}(\mathscr{D}))$          // Compute lower ID
     from Definition 5.7
13 $\Delta_{s,+}(\mathscr{D}) = \frac{1}{|\mathscr{X}|}(\Delta(\mathscr{D}) + \sum_{i=1}^{l-1} \sum_{s_i<j<s_{i+1}} \phi_{s_{i+1}}(\mathscr{D}))$          // Compute upper ID
     from Definition 5.7

   // Approximation finished.  Continue with exact computation, if
      desired.
14 if exact then
15     forall $i$ in $\{1, \ldots l\}$ do
16         forall $s_i < j < s_{i+1}$ do          // Iterate between support elements
17             forall $f$ in $F_{s_i}$ do          // Only use $F_{s_i}$ because of Lemma 5.12
18                 $\phi_{s_i,f}(\mathscr{D}) = \min_{j \in \{0,\ldots,|\mathscr{X}|-s_i\}} l^{f,\mathscr{D}}_{s_i+j} - l^{f,\mathscr{D}}_{1+j}$.
19             $\phi_j(\mathscr{D}) = \max(\{\phi_{j,f}(\mathscr{D}) \mid f \in F_i\} \cup \{\phi_{s_i}\})$          // Use Lemma 5.12
20             $\Delta(\mathscr{D}) += \phi_j(\mathscr{D})$
21     $\Delta(\mathscr{D}) = \frac{1}{|\mathscr{X}|}\Delta(\mathscr{D})$
22     return $\Delta_{s,-}(\mathscr{D}), \Delta_{s,+}(\mathscr{D}), \Delta(\mathscr{D})$
23 return $\Delta_{s,-}(\mathscr{D}), \Delta_{s,+}(\mathscr{D})$
```

Theorem 5.6 *For $m > n \geq 2$ and $f \in F$ the following statements hold.*

1. $\phi_{m,f}(\mathscr{D}) \geq \phi_{n,f}(\mathscr{D})$,
2. $\phi_m(\mathscr{D}) \geq \phi_n(\mathscr{D})$.

Proof. The second inequality directly follows from the first one. Since we know per definition that $\phi_{m,f}(\mathscr{D}) = \min_{M\subseteq\mathscr{X},|M|=m} \max_{x,y\in M} |f(x)-f(y)|$ and furthermore that $\phi_{n,f}(\mathscr{D}) = \min_{N\subseteq\mathscr{X},|N|=n} \max_{x,y\in N} |f(x)-f(y)|$, we need to show that for each $M \subseteq \mathscr{X}$ with $|M| = m$ there exist $N \subseteq \mathscr{X}$ with $|N| = n$ and $\max_{x,y\in M} |f(x)-f(y)| \geq \max_{x,y\in N} |f(x)-f(y)|$. It is sufficient to show that for $n = m-1$. Choose $x_1, x_2 \in M$ such that $\max_{x,y\in M} |f(x)-f(y)| = |f(x_1)-f(x_2)|$. As $|M| > 2$ we find $x_3 \in M \setminus \{x_1, x_2\}$. Let $N := M \setminus \{x_3\}$. It holds that

$$\max_{x,y\in M} |f(x)-f(y)| = |f(x_1)-f(x_2)| = \max_{x,y\in N} |f(x)-f(y)|.$$

□

5.4.1 Computing Intrinsic Dimension via Support Sequences

Equipped with Theorem 5.6, we can bound $\Delta(\mathscr{D})$ and thus the intrinsic dimension through computing ϕ_{s_i} for a few $2 = s_1 < s_2 \cdots < s_l = |\mathscr{X}|$.

Definition 5.7 (Support Sequences and Upper / Lower ID)
Let $s = (2 = s_1, \ldots, s_{l-1}, s_l = |\mathscr{X}|)$ be a strictly increasing and finite sequence of natural numbers. We call s a *support sequence* of $\mathscr{D}$. We additionally define

$$\begin{aligned}
\Delta_{s,-}(\mathscr{D}) &:= \frac{1}{|\mathscr{X}|}\left(\sum_{i=1}^{l} \phi_{s_i}(\mathscr{D}) + \sum_{i=1}^{l-1} \sum_{s_i<j<s_{i+1}} \phi_{s_i}(\mathscr{D})\right), \\
\Delta_{s,+}(\mathscr{D}) &:= \frac{1}{|\mathscr{X}|}\left(\sum_{i=1}^{l} \phi_{s_i}(\mathscr{D}) + \sum_{i=1}^{l-1} \sum_{s_i<j<s_{i+1}} \phi_{s_{i+1}}(\mathscr{D})\right)
\end{aligned} \tag{5.4}$$

and call accordingly

$$\partial_{s,-}(\mathscr{D}) := \frac{1}{\Delta_{s,+}(\mathscr{D})^2}$$

the *lower intrinsic dimension* of $\mathscr{D}$ and

$$\partial_{s,+}(\mathscr{D}) := \frac{1}{\Delta_{s,-}(\mathscr{D})^2}$$

the *upper intrinsic dimension* of D.

The governing idea is for $i \in \{1, \ldots, l\}$ and j with $s_i < j < s_{i+1}$ to substitute $\phi_j(\mathscr{D})$ with $\phi_{s_i}(\mathscr{D})$ or $\phi_{s_{i+1}}(\mathscr{D})$. With Theorem 5.6 this results in lower and upper bounds for $\Delta(\mathscr{D})$ and thus for the intrinsic ID. By comparing upper and lower bounds, we can approximate the ID and estimate the approximation error.

Corollary 5.8 *For all support sequences s holds*

$$\Delta_{s,-}(\mathscr{D}) \leq \Delta(\mathscr{D}) \leq \Delta_{s,+}(\mathscr{D}),$$
$$\partial_{s,-}(\mathscr{D}) \leq \partial(\mathscr{D}) \leq \partial_{s,+}(\mathscr{D}).$$

Definition 5.9 (Approximation Error)
For a support sequence s, we call

$$\mathrm{E}(s,\mathscr{D}) := \frac{\partial_{s,+}(\mathscr{D}) - \partial_{s,-}(\mathscr{D})}{\partial_{s,-}(\mathscr{D})}$$

the *(relative) approximation error* of $\partial(\mathscr{D})$ with respect to s.

With the computation of the upper and lower ID it is possible to bound the error with respect to the ID $\partial(\mathscr{D})$. The following corollary can be deduced from Corollary 5.8 and Definition 5.9.

Corollary 5.10 *For a support sequence s the following statements hold.*

1. $\max\{\frac{\partial_{s,+}(\mathscr{D})-\partial(\mathscr{D})}{\partial(\mathscr{D})}, \frac{\partial(\mathscr{D})-\partial_{s,-}(\mathscr{D})}{\partial_{s,-}(\mathscr{D})}\} \leq \mathrm{E}(s,\mathscr{D})$,
2. $\max\{|\partial_{s,+}(\mathscr{D}) - \partial(\mathscr{D})|, |\partial(\mathscr{D}) - \partial_{s,-}(\mathscr{D})|\} \leq |\partial_{s,+}(\mathscr{D}) - \partial_{s,-}(\mathscr{D})|$.

If the error of the approximation of a specific support sequence is not sufficient, further elements can be added to the support sequence. Directly from Equation (5.4) follows the following corollary.

Corollary 5.11 *Let $s = (2 = s_1, \ldots, s_l = |\mathscr{X}|)$ be a support sequence and let additionally be $\hat{s} = (s_1, \ldots, s_i, p, s_{i+1}, \ldots, s_l)$ with a support sequence with an additional element p. Then it holds that*

$$\Delta(\mathscr{D})_{\hat{s},-} = \Delta(\mathscr{D})_{s,-} + \frac{\sum_{p \leq j < s_{i+1}} ((\phi_p(\mathscr{D})) - \phi_{s_i}(\mathscr{D}))}{|\mathscr{X}|},$$
$$\Delta(\mathscr{D})_{\hat{s},+} = \Delta(\mathscr{D})_{s,+} + \frac{\sum_{s_i < j \leq p} \left((\phi_p(\mathscr{D})) - \phi_{s_{i+1}}(\mathscr{D})\right)}{|\mathscr{X}|}.$$

For a given support sequence s, Corollary 5.10 gives us an upper bound for the error when $\partial_{s,+}(\mathscr{D})$ or $\partial_{s,-}(\mathscr{D})$ are used to approximate $\partial(\mathscr{D})$ without knowing $\partial(\mathscr{D})$. Hence, we can compute (a lower bound) for the accuracy when approximating the ID with Definition 5.9. As we can see in Section 5.5.1, Section 5.5.4 and Section 5.6, comparable small support sequences lead to sufficient approximations. Support sequences can also be used to shorten the computation of the exact intrinsic dimension as the following lemma shows.

Lemma 5.12 *Let $s = (2 = s_1, \dots, s_l = |\mathscr{X}|)$ be a support sequence. Furthermore, let $i \in \{1, \dots, l-1\}$ and let $j \in \mathbb{N}$ with $s_i < j < s_{i+1}$. Let*

$$F_{s_i} := \{f \in F \mid \phi_{s_{i+1},f}(\mathscr{D}) > \phi_{s_i}(\mathscr{D})\}.$$

Then it holds that

$$\phi_j(\mathscr{D}) = \max(\{\phi_{j,f}(\mathscr{D}) \mid f \in F_{s_i}\} \cup \{\phi_{s_i}(\mathscr{D})\}).$$

Proof. "$\geq$" follows from Theorem 5.6 and the definition of $\phi_j(\mathscr{D})$. "$\leq$" holds because for $f \in F \setminus F_{s_i}$ it holds that $\phi_{j,f}(\mathscr{D}) \leq \phi_{s_{i+1},f}(\mathscr{D})$, due to Theorem 5.6, and $\phi_{s_{i+1},f}(\mathscr{D}) \leq \phi_{s_i}(\mathscr{D})$, due to the construction of F_{s_i}. □

Hence, given a specific j, it is possible to compute $\phi_j(\mathscr{D})$ using a subset of F. Based on the particular GD $\mathscr{D}$, this fact can considerably speed up the computation of the ID of $\mathscr{D}$, as we will see in Section 5.5.

An algorithm to approximate and compute the ID through support sequences is depicted in Algorithm 5.2. This algorithm takes as input a GD $\mathscr{D}$ and a chosen support sequence s. A reasonable choice for support sequences is discussed in Section 5.5.1. The output is $\Delta_{s,-}(\mathscr{D}), \Delta_{s,+}(\mathscr{D})$, and $\Delta(\mathscr{D})$, if desired (Line 14). In Line 2, all feature sequences are computed. In Line 6 to Line 11, $\phi_{s_i}(\mathscr{D})$ and $F_{s_{i-1}}$, as defined in Lemma 5.12, are computed. From Line 1 to Line 13, the feature sequences and the lower and upper ID are computed. If desired, the exact computation is done in Line 15 to Line 21. Here, we iterate for all support elements (Line 15) through all "gaps" between them (Line 16) and compute $\phi_j(\mathscr{D})$ using Lemma 5.12 (Line 17 to Line 19).

5.4.2 Estimating Computational Costs

Let $s = (s_1, \dots, s_l)$ be a support sequence. After the computation of $F_{s_1}, \dots, F_{s_{l-1}}$, we can estimate how much computation steps we can avoid in order to compute $\partial(\mathscr{D})$ with Algorithm 5.2 compared to Algorithm 5.1. Together with the error function $E(s)$, this estimation can help us to decide if it is desirable to compute the exact value $\partial(\mathscr{D})$ or leave it at $\partial_{s,-}(\mathscr{D})$ and $\partial_{s,+}(\mathscr{D})$. This is done in the following manner. For a specific $f \in F$, Lemma 5.5 shows that the computation of $\phi_{k,f}(\mathscr{D})$ requires $|\mathscr{X}| - k + 1$ different subtractions and to keep the minimum value. Hence, the cost for computing $\Delta(\mathscr{D})$ and therefore $\partial(\mathscr{D})$ via Algorithm 5.1 can be estimated via $\mathscr{O}(|F| \sum_{k=2}^{|\mathscr{X}|}(|\mathscr{X}| - k + 1)) = \mathscr{O}(|F| \sum_{k=1}^{|\mathscr{X}|-1} k) = \mathscr{O}(|F| (\frac{|\mathscr{X}|^2 - |\mathscr{X}|}{2}))$. However, if we use Algorithm 5.2, our cost estimation for all values j with $s_i < j < s_{i+1}$ is

Table 5.1: Statistics of all datasets used in this chapter.

	Nodes	Edges	Attributes
PubMed	19,717	88,648	500
Cora	2,708	10,556	1,433
CiteSeer	3,327	9,104	3,703
ogbn-arxiv	169,343	1,166,243	128
ogbn-products	2,449,029	61,859,140	100
ogbn-mag	1,939,743	21,111,007	128
ogbn-papers100M	111,059,956	1,615,685,872	128

$|F_{s_i}|(|\mathscr{X}|-j+1)$. Hence, for a given support sequence $s=(s_1,\dots,s_l)$, we can estimate how many computations are saved using the following notions.

We address the *naive computation costs* for computing the ID of a GD with

$$\mathrm{C}(\mathscr{D}) := |F|(\frac{|\mathscr{X}|^2-|\mathscr{X}|}{2}).$$

In contrast, for a support sequence $s=(s_1,\dots,s_l)$ of $\mathscr{D}$, the *computation costs* are

$$\mathrm{C}_s(\mathscr{D}) := (|F|\sum_{k=1}^{l}|\mathscr{X}|-s_k+1)+\sum_{k=1}^{l-1}|F_{s_k}|\sum_{s_k<j<s_{k+1}}|\mathscr{X}|-j+1. \tag{5.5}$$

Hence, the *saved costs* of s are

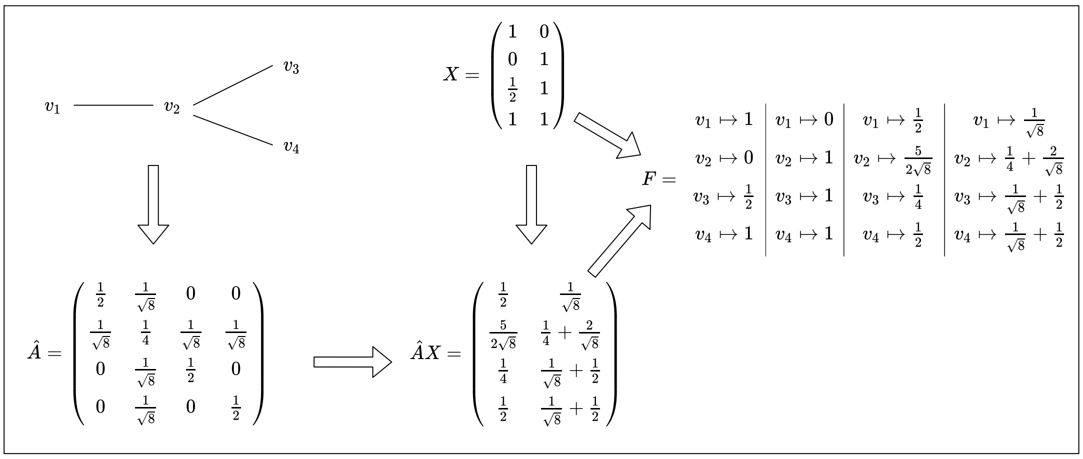
$$\mathrm{SC}_s(\mathscr{D}) := 1-\frac{\mathrm{C}_s(\mathscr{D})}{\mathrm{C}(\mathscr{D})}.$$

Once we have computed $\phi_{s_i}(\mathscr{D})$ and F_i, depending on the saved costs, we can decide to discard the support sequence or to continue further computations with it. Furthermore, using the error estimation, we can decide to compute the exact ID or to settle with the approximation.

5.5 Intrinsic Dimension of Real-World Graph Data

Graph data is of major interest in the realm of geometric learning and beyond. In the following, we assume to have a graph dataset $D=(G,X)$ as defined in Definition 2.24 with $G=(V,E)$ where $V=\{v_1,\dots,v_n\}$ and $X\in\mathbb{R}^{n\times d}$.

Learning from such data is often done, as already discussed in Section 2.7.2, via graph neural networks. In the following, we will focus on networks using the SIGN architec-



v_1
v_2
v_3
v_4
$X = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \frac{1}{2} & 1 \\ 1 & 1 \end{pmatrix}$
$F =$
$v_1 \mapsto 1$, $v_2 \mapsto 0$, $v_3 \mapsto \frac{1}{2}$, $v_4 \mapsto 1$
$v_1 \mapsto 0$, $v_2 \mapsto 1$, $v_3 \mapsto 1$, $v_4 \mapsto 1$
$v_1 \mapsto \frac{1}{2}$, $v_2 \mapsto \frac{5}{2\sqrt{8}}$, $v_3 \mapsto \frac{1}{4}$, $v_4 \mapsto \frac{1}{2}$
$v_1 \mapsto \frac{1}{\sqrt{8}}$, $v_2 \mapsto \frac{1}{4} + \frac{2}{\sqrt{8}}$, $v_3 \mapsto \frac{1}{\sqrt{8}} + \frac{1}{2}$, $v_4 \mapsto \frac{1}{\sqrt{8}} + \frac{1}{2}$
$\hat{A} = \begin{pmatrix} \frac{1}{2} & \frac{1}{\sqrt{8}} & 0 & 0 \\ \frac{1}{\sqrt{8}} & \frac{1}{4} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ 0 & \frac{1}{\sqrt{8}} & \frac{1}{2} & 0 \\ 0 & \frac{1}{\sqrt{8}} & 0 & \frac{1}{2} \end{pmatrix}$
$\hat{A}X = \begin{pmatrix} \frac{1}{2} & \frac{1}{\sqrt{8}} \\ \frac{5}{2\sqrt{8}} & \frac{1}{4} + \frac{2}{\sqrt{8}} \\ \frac{1}{4} & \frac{1}{\sqrt{8}} + \frac{1}{2} \\ \frac{1}{2} & \frac{1}{\sqrt{8}} + \frac{1}{2} \end{pmatrix}$

Figure 5.1: Example of an k-hop geometric dataset with $k = 1$. Given are a graph and an attribute matrix X. Then, the normalized adjacency matrix $\hat{A}$ and then $\hat{A}X$ are computed. The feature set F are the coordinate projections of X and $\hat{A}X$. In the figure, every column after "$F =$" represents one $f \in F$. Note, that in this example the normalization factor $\frac{1}{d_{\max}}$ is 1.

ture [129] as defined in Equation (2.5). Recall, that this means, that the employed networks use inputs of the form

$$(X, \hat{A}X, \hat{A}^2X \dots, \hat{A}^kX), \tag{5.6}$$

where $\hat{A}$ is the normalized adjacency matrix as defined in Definition 2.25. The feature set of the following geometric dataset corresponds to the input in Equation (5.6).

Definition 5.13 (k-hop feature functions and k-hop geometric datasets)
Let $k \in \mathbb{N}$ and $\hat{A}$ be the normalized adjacency matrix of a graph dataset D. Furthermore, let $d_{\max} := \max_{j \in \{1,\dots,d\}} \max_{i,k \in \{1,\dots,n\}} |X_{i,j} - X_{k,j}|$. We call the set

$$F_{D,k} := \{v_i \mapsto \frac{1}{d_{\max}}(\hat{A}^m X)_{i,j} \mid m \in \{0,\dots,k\}, j \in \{1,\dots,d\}\}$$

the *k-hop feature functions* of D. Let ν be the normalized counting measure on V. If there exist for each $v_i, v_k \in V$ with $v_i \neq v_j$ elements $m \in \{0,\dots,k\}, j \in \{1,\dots,d\}$ such that $\frac{1}{d_{\max}}(\hat{A}^m X)_{i,j} \neq \frac{1}{d_{\max}}(\hat{A}^m X)_{k,j}$, then $\mathscr{D}_k = (V, F_{D,k}, \nu)$ is a GD. We call it the *k-hop geometric dataset* of D.

Basic statistics of all seven graph datasets considered in the following sections are depicted in Table 5.1. The statistics for **Cora**, **PubMed** and **CiteSeer** were taken from

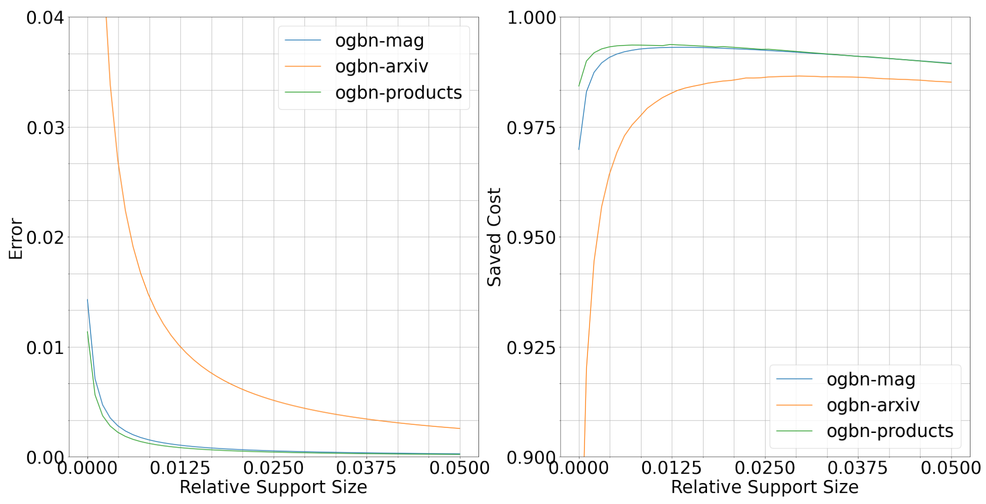


Figure 5.2: Errors and saved costs for approximating and computing the intrinsic dimensionality for $2-hop$ geometric datasets with different lengths of the support sequence.

PyTorch Geometric [1]. The statistics of the OGB datasets were taken from the Open Graph Benchmark. [2] An example of a k-hop geometric dataset is depicted in Figure 5.1. It is well-known that the normalized adjacency matrix $\hat{A} \in \mathbb{R}^{n\times n}$ of a graph has a spectral radius of 1. As $\hat{A}$ is symmetric, this yields $\|\hat{A}x-\hat{A}y\| \leq \|x-y\|$ for $x,y \in \mathbb{R}^n$. The significance of this property is for the respective computations, however, limited, since it primarily leads to insights of the behavior of the columns of X under multiplication with powers of $\hat{A}$. In contrast, the attribute vectors of the vertices are represented via the rows. Moreover, we may point out that we are not considering the Euclidean distances between attribute vectors, but differences between coordinate values. Thus, the spectral radius of $\hat{A}$ does not provide direct insights into $F_{D,k}$.

5.5.1 Choosing Support Sequences

Algorithm 5.2 relies on a proper choice for a support sequence s. To choose s, two properties have to be considered. Namely, the length of the support sequence and the spacing of the elements. Regarding the second point, we decided to use log-scale spacing. To get such a support sequence for a geometric dataset $\mathscr{D} = (\mathscr{X}, F, \mu)$, we first choose a geometric

[1]https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html#torch_geometric.datasets.Planetoid
[2]https://ogb.stanford.edu/docs/nodeprop

sequence $\hat{s} = (s_1, \ldots s_l)$ of length l from $|\mathscr{X}|$ to 2. We derive the final support sequence s from $s' = (\lfloor |\mathscr{X}| + 2 - s_1 \rfloor, \ldots, \lfloor |\mathscr{X}| + 2 - s_l \rfloor)$ by removing duplicated elements.

In the following, we study the error and the saved costs for different lengths l of the support sequence. Here, for a geometric dataset, we investigate how $\mathrm{E}(s, \mathscr{D})$ and $l\,\mathrm{SC}_s(\mathscr{D})$ vary for s chosen with $l \in \{\lfloor 0.001 \cdot |\mathscr{X}| \rfloor, \lfloor 0.002 \cdot |\mathscr{X}| \rfloor, \ldots, \lfloor 0.05 \cdot |\mathscr{X}| \rfloor\}$. Here, if $l = \lfloor r \cdot |\mathscr{X}| \rfloor$, we call $r \in \mathbb{R}$ the *relative support size* of the resulting support sequence s. We experiment with common benchmark datasets, namely **ogbn-arxiv**, **ogbn-mag** and **ogbn-products** from the Open Graph Benchmark [76, 75]. Since for **ogbn-mag** only a subset of vertices is equipped with attribute vectors, we generate the missing vectors via metapath2vec [42]. For all datasets, we consider the 2-hop geometric dataset. The results are depicted in Figure 5.2.

Results

For all datasets, low errors and high saved costs can be reached with a remarkably short support sequence. With relative support sizes of under 0.015 all datasets are approximated with an accuracy of over 99%. Furthermore, the saved costs for sequences with comparable relative support sizes is over 0.98. It stands out, that for the larger datasets **ogbn-mag** and **ogbn-products**, shorter sequences (relative to the size of the dataset) lead to lower errors and higher saved costs then for **ogbn-arxiv**. Our results further indicate, that a relative support size between 0.01 and 0.02 is a reasonable range for maximizing the saved costs. For longer support sequences, the saved cost decrease while the error does not change dramatically, at least for the 2- hop geometric datasets of **ogbn-mag** and **ogbn-products**. Note, that longer support sequence do not always lead to a higher amount of saved costs. For longer support sequences s the costs of computing $\phi_k(\mathscr{D})$ for $k \notin s$ decreases. However, the costs of computing $\phi_{s_i}(\mathscr{D})$ for all elements $s_i \in s$ increase.

5.5.2 Neighborhood Aggregation and Intrinsic Dimension

We study how the choice of k affects the intrinsic dimension value of the k-hop geometric dataset. For this, we compute the intrinsic dimension for $k \in \{0, 1, \ldots, 5\}$ for six datasets: the three datasets mentioned above and **PubMed**, **Cora** and **CiteSeer** [157], which we retrieved from PyTorch Geometric [53]. Furthermore, we train GNNs which use the feature functions of k-hop geometric datasets as information for training and inference. This allows us to discover connections between the ID of specific datasets with respect to the considered feature functions and the performance of classifiers, which rely on these feature functions. For this, we train SIGN models [129] for $k \in \{0, \ldots, 5\}$.

Table 5.2: Intrinsic dimension and performances on classification tasks. In the upper table, we display IDs for all k-hop geometric datasets for $k \in \{0,\dots,5\}$. In the middle table, we display the ID estimated by the MLE baseline. In the lower table we display mean and standard derivations for test accuracy of a standard SIGN model on the classification tasks which belongs to the dataset.

Dataset \ k-hop	0	1	2	3	4	5
PubMed	2542.3425	2336.6611	2077.5821	2077.0953	2077.0886	2077.0848
Cora	6.2523	3.8324	3.6689	3.6627	3.6624	3.6623
CiteSeer	22.3337	11.3166	10.2347	9.8134	9.5491	9.3795
ogbn-arxiv	83.9160	31.4731	31.4731	31.4730	30.7370	30.3767
ogbn-products	$1,169,323.2496$	$1,169,044.4736$	$1,169,044.2216$	$1,169,044.2216$	$1,169,044.2216$	$1,169,044.2216$
ogbn-mag	$2,311.3509$	$2,284.0290$	$2,284.0290$	$2,284.0290$	$2,284.0290$	$2,284.0290$

Dataset \ k-hop	0	1	2	3	4	5
PubMed	24.4623	24.7303	23.3924	22.2779	21.3495	20.5642
Cora	30.6049	28.1785	19.9316	10.8186	9.2970	8.6155
CiteSeer	58.9593	26.5031	16.5556	12.0495	9.3171	7.9572
ogbn-arxiv	16.2948	19.8571	18.9068	18.2265	17.4905	16.9325
ogbn-products	2.8694	4.7542	4.7950	4.7659	4.6943	4.6687
ogbn-mag	30.7024	33.2848	31.5140	30.4844	29.9080	29.5956

Dataset \ k-hop	0	1	2	3	4	5
PubMed	$.6850 \pm .0145$	$.7191 \pm .0123$	$.7378 \pm .0362$	$.7565 \pm .0165$	$.7615 \pm .0160$	$.7571 \pm .0234$
Cora	$.5329 \pm .0120$	$.7223 \pm .0117$	$.7766 \pm .0045$	$.7870 \pm .0076$	$.7917 \pm .0084$	$.7951 \pm .0047$
CiteSeer	$.4975 \pm .0075$	$.6165 \pm .0160$	$.6530 \pm .0101$	$.6677 \pm .0074$	$.6695 \pm .0085$	$.6734 \pm .0080$
ogbn-arxiv	$.5341 \pm .0090$	$.6572 \pm .0052$	$.6903 \pm .0056$	$.6917 \pm .0074$	$.6901 \pm .0083$	$.6890 \pm .0051$
ogbn-products	$.5969 \pm .0016$	$.7204 \pm .0017$	$.7590 \pm .0017$	$.7660 \pm .0014$	$.7678 \pm .0022$	$.7687 \pm .0019$
ogbn-mag	$.2712 \pm .0020$	$.3635 \pm .0029$	$.3879 \pm .0030$	$.3959 \pm .0029$	$.3983 \pm .0050$	$.4012 \pm .0040$

Setup of SIGN Classifiers

For **PubMed**, **Cora** and **CiteSeer**, we train on the classification task provided by Pytorch Geometric [53] which was earlier studied by Yang et al [157]. All Open Graph Benchmark datasets are trained and tested on the official *node property prediction* task.[3] Our goal is not to find optimal classifiers but to discover connections between the choice of k, the ID and classifier performance. Thus, we omit excessive parameter tuning and stick to reasonable standard parameters. For all tasks, we use a simple SIGN model with one hidden inception layer and one classification layer. For **PubMed**, **CiteSeer** and **Cora**, we use batch sizes of 256, hidden layer size of 64 and dropout at the input and hidden layer with 0.5. The learning rate is set to 0.01. All these parameters were taken from Kipf and Welling [85]. For **ogbn-arxiv**, **ogbn-mag** and **ogbn-products**, we stick to the parameters from the SIGN implementations on the OGB leaderbord. For **ogbn-arxiv**, we use a hidden dimension of

[3]https://ogb.stanford.edu/docs/nodeprop/

512, dropout at the input with 0.1 and with 0.5 at the hidden layer. For **ogbn-mag**, we use a hidden dimension of 512, do not dropout at the input and use dropout with 0.5 at the hidden layer. For **ogbn-products**, we use a hidden dimension of 512, input dropout of 0.3 and hidden layer dropout of 0.4. For all ogbn tasks, the learning rate is 0.001 and the batch-size 50000. For all experiments, we train for a maximum of 1000 epochs with early stopping on the validation accuracy. Here, we use a patience of 15. These are the standard parameters of Pytorch Lightning.[4] For all models, we use an Adam optimizer with weight decay of 0.0001. We report mean test accuracies over 10 runs. The intrinsic dimensions and the test accuracy are shown in Table 5.2.

Baseline Estimator

To investigate to which extent our ID function surpasses established ID estimators with respect to estimating the discriminability of a dataset, we also compute all ID values with the maximum likelihood estimator (MLE) [97]. This estimator is commonly used in the realm of deep learning [125, 103, 104]. For our experiments, we use the corrected version proposed by MacKay and Ghahramani [105]. Note, that the MLE is only applicable to datasets $\mathscr{X} \in \mathbb{R}^{n\times d}$ and is thus not able to respect the neighborhood aggregated feature functions. Hence, we incorporate the neighborhood information of a k-hop dataset by concatenating feature vectors with the neighborhood aggregated feature vectors.

To use the MLE ID, we have to convert the k-hop geometric dataset $(V, F_{D,k}, \nu)$ of graph data $D = (X, G)$, where $X \in \mathbb{R}^{n\times d}$, into a real-valued feature matrix $\hat{X}$. This done by concating the rows of $\mathscr{X}$ with the rows of $\hat{A}X, \dots \hat{A}^k X$, i.e., $\hat{X} \in \mathbb{R}^{n\times(k+1)d}$ with

$$\hat{X}_{i,j} := \begin{cases} X_{i,j} & j \in \{1,\dots,d\}, \\ (A^n X)_{i,\hat{j}} & j = nd + \hat{j} \texttt{ for } n \in \{1,\dots,k\}, \hat{j} \in \{1,\dots d\}. \end{cases}$$

The MLE is given via

$$\mathrm{MLE}(\hat{X}) := \frac{1}{n(k-1)} \sum_{i=1}^{n} \sum_{j=1}^{l-1} \log\left(\frac{d(\hat{X}_i, N_l(\hat{X}_i))}{d(\hat{X}_i, N_j(\hat{X}_i))}\right), \tag{5.7}$$

where d is the euclidean metric and $N_j(\hat{X}_i)$ is the j-th nearest neighbor of $\hat{X}_i$ with respect to the Euclidean metric. Thus, the MLE depends on a parameter l, which we set to 5.

We implement the MLE by using the *NearestNeighbors* class of scikit-learn [119] and then building the mean of all $\log(\frac{d(X_i, N_5(X_i))}{d(X_i, N_j(X_i))})$ with $i \in \{1,\dots,n\}$ and $j \in \{1,\dots,5\}$. Here,

[4]https://www.pytorchlightning.ai/

we skip all elements where $d(\hat{X}_i, N_j(\hat{X}_i)) = 0$. This can happen, when $\hat{X}$ has duplicated rows, representing data points with equal attribute vectors.

For **ogbn-mag** and **ogbn-products**, computing Equation (5.7) is not possible due to performance reasons. Here, we sample $169,343$ indices $I \subseteq \{1,\dots,n\}$ and only compute

$$\mathrm{MLE}(\hat{X}) := \frac{1}{n(k-1)} \sum_{i \in I} \sum_{j=1}^{l-1} \log\left(\frac{d(X_i, N_l(X_i))}{d(X_i, N_j(X_i))}\right).$$

We choose such a specific size for I because this is the number of data points in **ogbn-arxiv**, the largest dataset for which the full computation was possible.

Results

We find that one iteration of neighborhood aggregation always leads to a huge drop of the ID when using our ID function. However, consecutive iterations only lead to a small decrease. For the datasets from OGB, some iterations lead to no drop of the ID dimension at all. For **ogbn-mag**, only the first iteration significantly decreases the ID, for **ogbn-products**, only the first two iterations are relevant for decreasing the ID. It stands out, that for **ogbn-arxiv**, the second and third iteration lead to no significant decrease, but the fourth and fifth do. The results for **PubMed** stand out. Here, the second iteration of neighborhood aggregation leads to a comparable decrease as the first one.

Considering the classification performances, the first iteration is again the key factor, leading to a significant increase in accuracy. As for the ID, the **PubMed** dataset behaves differently than the other datasets: the second iteration of neighborhood aggregation leads to a comparable increase in accuracy as the first.

The MLE ID behaves different. Here, no pattern of the first iteration of aggregation being the key for decreasing the data complexity is observed. For some datasets, the first rounds of feature aggregation may even increase the intrinsic dimension. To sum up, our results indicate that our ID is a better indicator for classification performance then the MLE ID.

5.5.3 Synthetic Data

To get further insights into the behavior of our ID notion, we now consider $k-$hop geometric dataset for one-hot encoded graph data, i.e., $D = (X,(V,E))$, with $X = \mathbb{I}_{|V|}$ where $\mathbb{I}_{|V|}$ is the $|V|$-dimensional identity matrix. We consider the case of $V = \{1,\dots,100\}$ and determine the ID for $k \in \{0,\dots,5\}$ for increasing edge sizes. To do so, we place the 4950 possible edges in a random order and add them step by step and compute the ID notions for the $k-$hop geometric dataset. The results can be found in Figure 5.3.

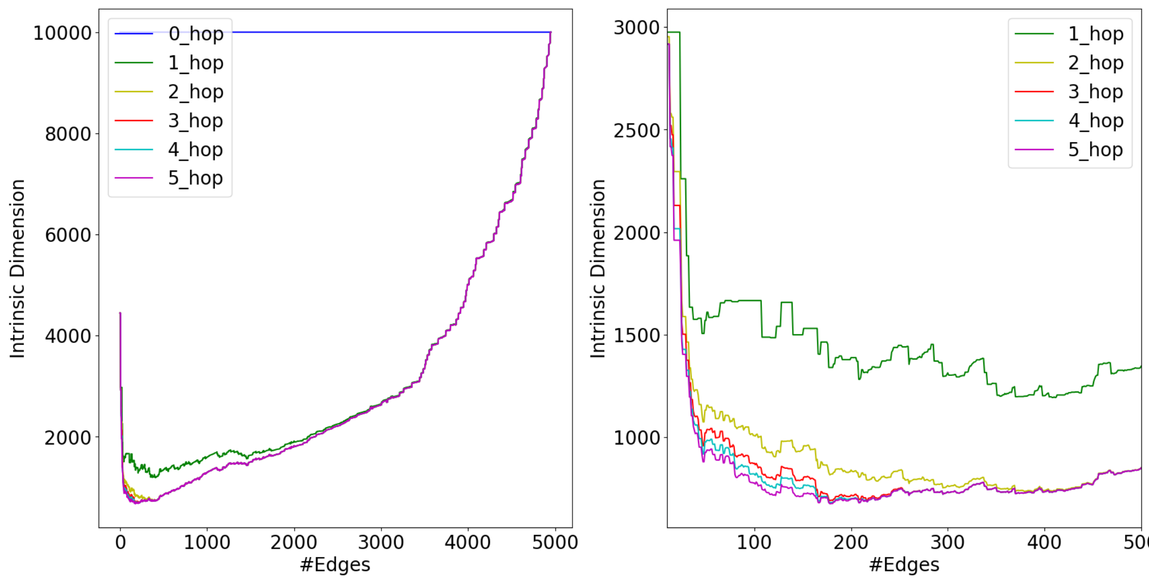


Figure 5.3: Intrinsic dimension of one-hot encoded features. The right plot is a zoom of the left one which hides the 0−hop geometric dataset.

Table 5.3: Approximation of intrinsic dimension for ogbn-papers100M.

k	0	1	2	3	4	5
$\partial_{s,-}(\mathscr{D})$	282.2380	171.7385	148.3323	137.7662	128.2751	125.3418
$\partial_{s,+}(\mathscr{D})$	282.3387	171.7997	148.3852	137.8153	128.3208	125.3864
$\mathrm{E}(s,\mathscr{D})$	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004

Results

The 0−hop geometric dataset has an ID which does not depend on the amount of edges. This is not surprising since it does not incorporate any graph information. For all other k values, the ID first sharply decreases and then increases. This indicates that the addition of neighborhood aggregation is particularly useful for graphs of moderate density. Here, the addition of additional rounds of aggregation beyond the first one can further lower the ID. For higher edge sizes, the ID difference between different k values vanishes.

5.5.4 Approximation of Intrinsic Dimension on Large-Scale Data

To demonstrate the feasibility of our approach, we use it to approximate the ID of the well known, large-scale **ogbn-mag-papers100M** dataset. For this, we construct the support sequence as in Section 5.5.1 with $l = 100.000$. The results are depicted in Table 5.3. On our *Xeon Gold System* with 16 cores, approximating the ID of a k-hop geometric dataset

Table 5.4: Error on randomly generated data.

n	d	$E(s,\mathscr{D})$
10^6	10	$2.55*10^{-4} \pm 1.13*10^{-8}$
10^6	50	$2.55*10^{-4} \pm 5.36*10^{-9}$
10^6	250	$2.55*10^{-4} \pm 2.31*10^{-9}$
10^7	10	$3.08*10^{-4} \pm 4.78*10^{-10}$
10^7	50	$3.08*10^{-4} \pm 6.57*10^{-10}$
10^7	250	$3.08*10^{-4} \pm 6.16*10^{-10}$
10^8	10	$3.55*10^{-4} \pm 3.67*10^{-11}$
10^8	50	$3.55*10^{-4} \pm 1.34*10^{-11}$
10^8	250	$3.55*10^{-4} \pm 2.69*10^{-11}$

build from **ogbn-mag-papers100M** is possible within a few hours. While the ID drops for every iteration of neighborhood aggregation, the decrease becomes smaller. The ID of the different k-hops can be differentiated by the approximation, i.e., $\partial_{s,-}(\mathscr{D}_i) > \partial_{s,+}(\mathscr{D}_{i+1})$ for $i \in \{0,\dots,4\}$. It stands out, that even for such a short support sequence (compared to the size of the dataset), the observed error is remarkably low. In detail, we can approximate the ID with an accuracy of over 99.95%. It is further remarkable, that the error does not change significantly for different k. We observed this effect also for the other datasets. Our results on **ogbn-papers100M** indicate, that with short support sequences, we can sufficiently approximate the ID of large-scale graph data.

5.6 Errors of Random Data

To further understand how our approximation procedure behaves we conducted experiments on random data. We considered different data sizes and different amount of attributes. For this, we experimented with real-valued datasets, i.e. datasets represented by an attribute matrix $X \in \mathbb{R}^{n\times d}$. Here, the feature functions are given by the data columns. To be more detailed, the considered geometric dataset is $\mathscr{D} = (\{X_i \mid i \in \{1,\dots,n\}\}, \{X_i \mapsto X_{i,j} \mid j \in \{1,\dots,d\}\}, \nu)$. Here, ν is again the normalized counting measure and X_i is he $i-th$ row vector of X. We iterate n through $\{10^6, 10^7, 10^8\}$ and d through $\{10, 50, 250\}$. We repeat all experiments 3 times. For all datasets, we build a support sequence as described in Section 5.5.1 with $l = 100,000$. The results can be found in Table 5.4.

For all datasets, the errors are small and the accuracy is over 99.9% for all considered data sizes. The difference in the error for different values of d is negligible. Furthermore, we have small standard deviations. All this indicates that $l = 100,000$ is a reasonable default choice that leads to sufficient approximations in a large range of data and attribute sizes.

5.7 Conclusion and Future Work

We presented a principle way to efficiently compute the intrinsic dimension (ID) of geometric datasets. Our approach is based on an axiomatic foundation and accounts for underlying structures and is therefore especially tailored to the field of geometric learning. We proposed a novel speed up technique for an algorithm which has quadratic complexity with respect to the amount of data points. This enabled us to compute the ID of several real-world graphs with up to millions of nodes. Equipped with this ability, we shed light on connections of classification performances of graph neural networks and the observed intrinsic dimension for common benchmark datasets. Finally, using a novel approximation technique, we were able to show that our method scales to graphs with over 100 million nodes and billions of edges. We illustrated this by using the well-known **ogbn-papers100M** dataset. To sum up, we generated novel global insights into real-world networks by providing a measure of complexity of whole networks that is applicable to modern large-scale graphs.

Future work includes the identification of suitable feature functions for other domains, such as learning on text or image data. Incorporating the structure of such datasets into the computation of intrinsic dimensionality is an open research problem. Another promising research direction is to investigate how the ID of datasets could be manipulated. Since our investigations suggest connections between a low ID and high classification performances, this has the potential to enhance learning procedures.

Chapter 6

Selecting Features by their Resilience to the Curse of Dimensionality

In this chapter, we present an application of the intrinsic dimensionality notion presented in the last section. We investigate which features of a specific dataset are useful to lower the intrinsic dimensionality. This results in a methods for feature selection that is competitive with well-established baselines. We therefore find empirical evidence that features that tame the curse of dimensionality are meaningful for the success of learning.

6.1 Introduction

Information derived by machine learning procedures is rarely comprehensible or interpretable. This includes even procedures originally categorized as explainable, such as decision trees. Addressing this problem, a variety of works aims to simplify data through methods that reduce size or, in particular, the dimensionality. An important class of such methods has the emblematic name *feature selection* (FS). By simply selecting to be discarded feature dimensions, they preserve the explainability of the remaining original dimensions, in contrast to approaches such as principal component analysis. Although there are supervised and unsupervised types of FS, only unsupervised procedures are suited to improve the understanding of data regardless of a specific learning task. Moreover, they do not require costly labels.

In common feature selection methods, the curse of dimensionality, as introduced in the last chapter, is rarely explicitly accounted for, even though it does a) demonstrably influences learning success in high-dimensional real-world scenarios, and b) potentially prevent common FS methods to choose the best features to learn from. Thus, it is crucial to incorporate the intrinsic dimension into the process of selecting features.

We meet this challenge by proposing an unsupervised feature selection method that picks features based on their measurable ability to discriminate different data points. While other methods are based on feature relevance measures, such as variances, or select features by discarding strongly correlated ones, our method ranks features by their resilience against the curse of dimensionality in the sense of Pestov. By adapting corresponding ideas for computing the intrinsic dimensionality from the last chapter, we derive an algorithm that allows to find those features that are able to tame the influence of the curse of dimensionality. Furthermore, adapting the speed-up techniques proposed in Chapter 5 will allow feature selection for large-scale datasets comprised of millions of data points. While Hanika et al. [70] and Chapter 5 are focused on computing the intrinsic dimensionality of datasets, we will rank and select individual features by their ability to discriminate data points. Thus, we identify features that are harmed by the curse of dimensionality by a comparable low extent.

We experimentally show on real-world datasets that features selected by our proposed method are meaningful to learn from. To be more specific, we again experiment on the *OpenML-CC18 Curated Classification benchmark* (Open18) [15] and witness that our method is competitive or even outperforms established feature selection methods with respect to feature selection for classification tasks. Furthermore, we again use OGB [76] to show that our method is capable of feature selection in the realm of social network analysis. To sum up, our experiments indicate that features that are able to discriminate data and thus weaken the curse of dimensionality are highly relevant for learning tasks. The code for this chapter is publicly available in the repository of the paper on which this chapter builds.[1]

6.2 Related Work

We focus on unsupervised feature selection, which is an established research topic [138, 26, 3]. Established methods are often focused on the identification of outstanding features by identifying the relevant features for clustering [47, 24, 46]. Another directions focuses on selecting a subset of important and dissimilar features [162, 52, 112, 69]. Here, feature importance can for example be measured via variances and similarities via correlation coefficients. While applying such methods may result in a decreasing intrinsic dimensionality, they do not explicitly account for the curse of dimensionality in the selection process.

A small amount of research has incorporated intrinsic dimension estimators into feature selection [60, 59, 114, 81]. However, these method either only use the ID to determine the amount of features to select [60] or they are based on recomputing intrinsic dimensionalities after discarding features which limits scalability [59, 114]. Even if applicable to larger

[1]https://github.com/mstubbemann/FSCOD

datasets, these methods build on notions of intrinsic dimensionality that do not aim to quantify the curse of dimensionality [81]. Instead, these notions work under the assumption that the data lies on a manifold of lower Euclidean dimension. The goal of the ID notion is then to approximate the dimension of this manifold.

In contrast, we build our feature selection procedure on a notion of ID which quantifies the influence of the curse of dimensionality which occurs when data points can not be discriminated. Note, that Hanika et al. [70] and the last chapter were solely focused on the intrinsic dimensionality of datasets and thus of the question how all features together can discriminate data points. In this chapter, we present a novel approach to rank and select individual features by their ability to discriminate data points.

6.3 Feature Selection via Discriminability

We again work with geometric datasets $\mathscr{D} = (\mathscr{X}, F, \mu)$ [70]. Recall, that $\mathscr{X}$ is a set of data points and $F \subseteq \mathbb{R}^{\mathscr{X}}$ is a set of feature functions from $\mathscr{X}$ to $\mathbb{R}$. We require $\sup_{x,y\in\mathscr{X}} d_F(x,y) < \infty$, where $d_F(x,y) := \sup_{f\in F}|f(x) - f(y)|$. Furthermore, $(\mathscr{X}, d_F)$ has to be a complete and separable metric space with μ being a Borel probability measure on $(\mathscr{X}, d_F)$.

From this point on, we again consider the special case of *finite geometric datasets*, i.e., $0 < |\mathscr{X}|, |F| < \infty$, with μ being the normalized counting measure. We build our method on the discriminability $\Delta(\mathscr{D})$ of a geometric dataset $\mathscr{D}$. The discriminability has in the case of finite geometric datasets the form

$$\Delta(\mathscr{D}) = \frac{1}{|\mathscr{X}|} \sum_{k=2}^{|\mathscr{X}|} \max_{f\in F} \min_{\substack{M\subseteq\mathscr{X} \\ |M|=k}} \max_{x,y\in M} |f(x) - f(y)|, \tag{6.1}$$

as shown in Theorem 5.2. We again use the notations $\phi_{k,f} := \min_{M\subseteq\mathscr{X},|M|=k} \max_{x,y\in M}|f(x) - f(y)|$, and $\phi_k := \max_{f\in F} \phi_{k,f}$ so the discriminability $\Delta(\mathscr{D})$ can be rewritten as

$$\Delta(\mathscr{D}) = \frac{1}{|\mathscr{X}|} \sum_{k=2}^{|\mathscr{X}|} \max_{f\in F} \phi_{k,f} = \frac{1}{|\mathscr{X}|} \sum_{k=2}^{|\mathscr{X}|} \phi_k, \tag{6.2}$$

as we already noted in Equation (5.3). In the following, we consider feature selection with a fixed feature budget in an unsupervised setting, i.e., without any known labels of a specific classification task at selecting time.

Problem Statement. Given a finite geometric dataset $\mathscr{D} = (\mathscr{X}, F, \mu)$ and a natural number $n_F \ll |F|$, select the n_F most important features of F.

6.3.1 Selection of Discriminating Features

The definition of discriminability as in Equation (6.1) and Equation (6.2) quantifies to which extent the set of all features can discriminate data subsets of different cardinality. The main idea of our feature selection algorithm is to rank features by their ability to discriminate data subsets of different cardinality by solely using this feature.

Definition 6.1 (Discriminability)
The *discriminability of $\mathscr{D}$ with respect to feature $f \in F$* is defined as

$$\Delta(\mathscr{D})_f^* := \frac{1}{|\mathscr{X}|} \sum_{k=2}^{|\mathscr{X}|} \min_{\substack{M \subseteq \mathscr{X} \\ |M|=k}} \max_{x,y \in M} |f(x) - f(y)| = \frac{1}{|\mathscr{X}|} \sum_{k=2}^{|\mathscr{X}|} \phi_{k,f}. \tag{6.3}$$

Enhancing Robustness against Outliers Note, that one data point with an outstanding value $f(x)$ can have a strong influence on $\Delta(\mathscr{D})_f{}^*$ via drastically increasing $\phi_{|\mathscr{X}|,f}$. To weaken this phenomenon, we propose to weight $\phi_{k,f}$ higher for smaller values of k. This leads to the following definition.

Definition 6.2 (Normalized Discriminability and Normalized Intrinsic Dimensionality)
The *normalized discriminability of $\mathscr{D}$ with respect to f* which we define as

$$\Delta(\mathscr{D})_f := \frac{1}{|\mathscr{X}|} \sum_{k=2}^{|\mathscr{X}|} \frac{1}{k} \phi_{k,f}.$$

The *normalized intrinsic dimensionality of $\mathscr{D}$ with respect to f* is then given via

$$\partial(\mathscr{D})_f := \frac{1}{\Delta(\mathscr{D})_f^2}. \tag{6.4}$$

We then can rank features by their discriminability and select the features with the highest discriminability/ lowest intrinsic dimensionality. We call the resulting method *features selection via discriminability* (FSD). It is depicted in Algorithm 6.1.

6.3.2 Discarding Highly Correlated Features

Our methods select features by their ability to separate data points. However, it does not consider connections between individual features, as for example correlations. Thus, if for example the two features that separate the dataset are nearly identical our method may select both as though selecting one of these features would be sufficient as the second one gives no

Algorithm 6.1: Feature Selection via Discriminability (FSD)

```
Input  : Finite geometric dataset $\mathscr{D} = (\mathscr{X}, F, \mu)$. Natural number $k << |F|$ of
         features to select from $F$.
Output : Selected features
1 forall $f$ in $F$ do
2     $\Delta(\mathscr{D})_f := \frac{1}{|\mathscr{X}|}\sum_{k=2}^{|\mathscr{X}|}\frac{1}{k}\phi_{k,f}$
3     $\partial(\mathscr{D})_f = \frac{1}{\Delta(\mathscr{D})_f^2}$
4 Set $l_F$ as the list of all $f \in F$ ordered by ascending $\partial(\mathscr{D})_f$.
5 return $l_F[: k]$
```

extra information. To prevent such cases, we propose to incorporate correlation coefficients into our feature selection process, if desired. In order to do so, we define an additional number n_c of features to remove via correlation coefficients. We then iteratively discard one of the two features f_1, f_2 with the maximal pearson correlation coefficient. The version of our algorithm that incorporates this preprocessing is called *Feature Selection via Discriminability and Correlation* (FSDC).

6.4 Feature Selection for Large-Scale Data

Since computing $\phi_{k,f}$ is in $\mathscr{O}(|\mathscr{X}| - k)$ for fixed $f \in F$ as discussed in Chapter 5, computing $\Delta(\mathscr{D})_f$ is in $\mathscr{O}(\sum_{k=2}^{|\mathscr{X}|}|\mathscr{X}| - k) = \mathscr{O}(|\mathscr{X}|^2)$. Hence, we have a worst case runtime which scales quadratic with $|\mathscr{X}|$. Thus, FSD(C) is applicable to medium-sized datasets with thousands of data points. However, it is not tailored to large-scale data with millions of data points.

We again make use of support sequences as defined in Definition 5.7, i.e., strictly increasing finite sequences of the form $s = (2 = s_1, \ldots, s_l = |\mathscr{X}|)$ to approximate $\Delta(\mathscr{D})$ and $\partial(\mathscr{D})$ by only computing $\phi_{s_i,f}$ for $s_i \in s$.

We approximate the discriminability and intrinsic dimensionality with respect to a feature $f \in F$. The approximation is based on the result, that the map $k \mapsto \phi_{k,f}$ is monotonically increasing as shown in Theorem 5.6. Using this result, we replace for $s_i < j < s_{i+1}$ the value $\phi_{j,f}$ by $\phi_{s_i,f}$ or $\phi_{s_{i+1},f}$. More formally, we get the following definition.

Definition 6.3 (Upper and Lower Normalized Discriminability)
For a feature $f \in F$ and a support sequence s we call

$$\Delta(\mathscr{D})_{s,f}^{+} := \frac{1}{|\mathscr{X}|}\left(\sum_{i=1}^{l}\frac{1}{s_i}\phi_{s_i,f} + \sum_{i=1}^{l-1}\sum_{s_i<j<s_{i+1}}\frac{1}{j}\phi_{s_{i+1},f}\right)$$

the *upper normalized discriminability with respect to f and s* and

$$\Delta(\mathscr{D})^{-}_{s,f} := \frac{1}{|\mathscr{X}|}\left(\sum_{i=1}^{l}\frac{1}{s_i}\phi_{s_i,f} + \sum_{i=1}^{l-1}\sum_{s_i<j<s_{i+1}}\frac{1}{j}\phi_{s_i,f}\right)$$

he *lower normalized discriminability with respect to f and s.*

Definition 6.4 (Upper, Lower and Approximated Normalized Intrinsic Dimensionality) We define the *upper normalized intrinsic dimensionality with respect to f and s* via

$$\partial(\mathscr{D})^{+}_{s,f} := \frac{1}{\left(\Delta(\mathscr{D})^{-}_{s,g}\right)^2}$$

and the *lower normalized intrinsic dimensionality with respect to f and s* via

$$\partial(\mathscr{D})^{-}_{s,f} := \frac{1}{\left(\Delta(\mathscr{D})^{+}_{s,f}\right)^2}.$$

Furthermore we call

$$\partial(\mathscr{D})_{s,f} := \frac{\partial(\mathscr{D})^{+}_{s,f} + \partial(\mathscr{D})^{-}_{s,f}}{2}. \tag{6.5}$$

the *approximated normalized intrinsic dimensionality with respect to f and s.*

Large-Scale Feature Selection via Discriminability The resulting algorithm for large-scale datasets is depicted in Algorithm 6.2. We rank features via ascending $\partial(\mathscr{D})_{s,f}$. If we want to select k features, we choose the first k features of the resulting order. We denominate this method with *large-scale feature selection via discriminability (LSFSD).* If we use the preprocessing via correlations as explained in Section 6.3.2 we instead call the method *large-scale feature selection via discriminability and correlation (LSFSDC).*

6.4.1 Error Ratios of Approximations

Note, that $\partial(\mathscr{D})_{s,f}$ as defined in Equation (6.5) only gives an approximation of $\partial(\mathscr{D})_f$ as defined in Equation (6.4). Thus ranking the features by $\partial(\mathscr{D})_{s,f}$ instead of $\partial(\mathscr{D})_f$ may lead to a different ordering. We are interested in the amount of such changes. Let $l_s := (f_{i_1}, \ldots, f_{i_{|F|}})$ be the list of the features in F ordered by ascending $\partial(\mathscr{D})_{s,f}$.

Algorithm 6.2: Large-Scale Feature Selection via Discriminability (LSFSD)

```
Input  : Finite geometric dataset $\mathscr{D} = (\mathscr{X}, F, \mu)$. Natural number $k << |F|$ of
         features to select from $F$. Support sequence $s = (2 = s_1, \dots, s_l = |\mathscr{X}|)$.
Output : Selected features
1 forall $f$ in $F$ do
2     $\Delta(\mathscr{D})^+_{s,f} := \frac{1}{|\mathscr{X}|}\left(\sum_{i=1}^{l} \frac{1}{s_i}\phi_{s_i,f} + \sum_{i=1}^{l-1}\sum_{s_i<j<s_{i+1}} \frac{1}{j}\phi_{s_{i+1},f}\right)$
3     $\Delta(\mathscr{D})^-_{s,f} := \frac{1}{|\mathscr{X}|}\left(\sum_{i=1}^{l} \frac{1}{s_i}\phi_{s_i,f} + \sum_{i=1}^{l-1}\sum_{s_i<j<s_{i+1}} \frac{1}{j}\phi_{s_i,f}\right)$
4     $\partial(\mathscr{D})^+_{s,f} := \frac{1}{\left(\Delta(\mathscr{D})^-_{s,g}\right)^2}$
5     $\partial(\mathscr{D})^-_{s,f} := \frac{1}{\left(\Delta(\mathscr{D})^+_{s,g}\right)^2}$
6     $\partial(\mathscr{D})_{s,f} := \frac{\partial(\mathscr{D})^+_{s,f} + \partial(\mathscr{D})^-_{s,f}}{2}$
7 Set $l_F$ as the list of all $f \in F$ ordered by ascending $\partial(\mathscr{D})_{s,f}$.
8 return $l_F[: k]$
```

Definition 6.5 (Error Ratio)
The *error ratio of s* is then given by

$$\mathrm{E}(s)^* := \frac{2|\{(k,l) \in \{1,\dots,|F|\} \mid k < l \wedge \partial(\mathscr{D})_{f_{i_k}} > \partial(\mathscr{D})_{f_{i_l}}\}|}{|F|(|F|-1)}.$$

Computing $\partial(\mathscr{D})_{s,f}$ for all $f \in F$ is especially of interest when computing $\partial(\mathscr{D})_f$ is not feasible. In such circumstances, it is also not possible to compute $\mathrm{E}(s)^*$. Hence, we need an approximation or upper bound of $\mathrm{E}(s)^*$ which can be computed without needing $\partial(\mathscr{D})_f$.

Definition 6.6 (Maximal Error Ratio)
We call for a support sequence s

$$\mathrm{E}(s) := \frac{2|\{(k,l) \in \{1,\dots,|F|\} \mid k < l \wedge \partial(\mathscr{D})^+_{s,f_{i_k}} > \partial(\mathscr{D})^-_{s,f_{i_l}}\}|}{|F|(|F|-1)}$$

the *maximal error ratio* of s.

The maximal error ratio $E(s)$ can be computed without knowing $\partial(\mathscr{D})_f$ for all $f \in F$. Per definition it holds that

$$\partial(\mathscr{D})^-_{s,f} \leq \partial(\mathscr{D})_{s,f} \leq \partial(\mathscr{D})^+_{s,f}.$$

Thus, for given features $f,g \in F$ with $\partial(\mathscr{D})^{+}_{s,f} \leq \partial(\mathscr{D})^{-}_{s,g}$ we can conclude

$$\partial(\mathscr{D})_f \leq \partial(\mathscr{D})^{+}_{s,f} \leq \partial(\mathscr{D})^{-}_{s,g} \leq \partial(\mathscr{D})_g.$$

Thus, we get the following corollary.

Corollary 6.7 *For each support sequence s it holds that*

$$\mathrm{E}(s)^* \leq \mathrm{E}(s). \tag{6.6}$$

To sum up, we now have an approximation algorithm which allow us to rank the features by their intrinsic dimensionality and we can efficiently bound the amount of errors this approximation produces.

6.5 Experiments

In the following we empirically examine the following hypothesis.

Discriminative features are meaningful features with respect to learning performances, especially in classification tasks.

The aim is **not** to develop a new feature selection procedure that surpasses all established methods. Thus, we do not compare with all state-of unsupervised feature selection procedures. Instead we use a small set of representative baselines that consists of established methods and common-sense baselines based on correlation and variances. Furthermore, we compare our approach to random feature selection to evaluate if our features are indeed meaningful with respect to classification.

In all our experiments, we consider the features F to be the coordinate projections, i.e., the feature functions are given via the data columns. We evaluate our feature selection method regarding to classification performances. We experiment with a logistic regression classifier on the *OpenML-CC18 Curated Classification benchmark* (Open18)[2] [15]. To further evaluate our selection procedure on modern large-scale networks we also select features for classification with graph neural networks on the subset of the *Open Graph Benchmark* [76] which we already used in Chapter 5. In both experiments, we evaluate how to classify on a small subset of the features. To be more detailed, we only want to keep 10% of all features. We evaluate our feature selection with and without the preprocessing procedure of Section 6.3.2. In the case with the preprocessing step, we use it do discard 10%

[2]https://www.openml.org/search?type=study&study_type=task&id=99&sort=tasks_included

of the features. In both experiments, we will also report results for classification on the full feature set. We evaluate against the following baselines.

- **Random.** Randomly selecting the 10% of features.
- **Correlation Based.** Only use the correlation based feature selection as proposed in Section 6.3.2 until only 10% of the features are left.
- **Variance Based.** Select the 10% of the features with the highest variance. We choose this baseline because it is a straight forward approach to for selecting features by their individual importance.
- **SPEC [162]**. SPEC first builds a complete graph between the individual data points with edge weights indicating similarities of the points and then uses spectral graph theory for estimating feature relevance. In our experiments, we use an RBF Kernel for similarity and evaluate feature relevance via ϕ_2. Note, that we here refer to the notation ϕ_2 from Zhao and Liu [162], not ϕ_2 as used in the rest of this chapter. For details, we refer to Zhao and Liu [162]. We choose this baseline because it was the univariate filter method with the highest classification accuracy in a recent survey on unsupervised feature selection [138].
- **RRFS [52]**. RRFS combines a function @sim which measures similarity of feature pairs and a relevance measure @rel that evaluates the importance of features. It iteratively chooses the next most relevant feature that has a similarity to the last chosen feature which below a specific threshold t. For comparison, we choose t to be the correlation coefficient value of the last pair of features that was used to discard a feature via Section 6.3.2. We use the Pearson correlation coefficient for similarity evaluation and evaluate the relevance of features via their variance. For more details, we refer to Ferreira and Figueiredo [52]. We use this baseline because it was the multivariate filter method with the highest classification accuracy in the survey mentioned above [138].

6.5.1 Feature Selection for Logistic Regression

We select features as explained above and then use a logistic regression classifier. We use the data splits that are provided by Open18 and report mean test accuracies over all splits. We report means over 10 runs of this experiment. Note, that the logistic regression of scikit-learn [119] using the default solver leads to deterministic results. As only the random

Table 6.1: Results on **Open18** experiment. We report the accuracies for our method with (FSDC) and without preprocessing (FSD) via correlation and for all baselines

Task ID	Dataset	$\|F\|$	Full	FSDC	FSD	Random	Corr	Vari	RRFS	SPEC
14	mfeat-fourier	76	.813	.694	**.753**	.433±.032	.457	.751	.751	.630
16	mfeat-karh.	64	.958	.695	**.822**	.466±.028	.288	.810	.810	.680
45	splice	60	.918	.531	.611	.580±.026	.534	.725	.647	**.806**
9910	Bioresponse	1776	.754	.745	**.751**	.704±.008	.739	.747	.707	.550
9977	nomao	118	.947	**.907**	**.907**	.835±.012	.752	.793	.793	.759
9981	cnae-9	856	.944	.884	.884	.391±.032	.404	**.888**	.144	.128
9985	1st-order	51	.479	**.434**	**.434**	.420±.002	.416	.419	.419	.418
167125	Internet-Advertisment	1558	.971	.962	**.963**	.936±.003	.899	**.963**	**.963**	.877

selection method is non-deterministic, this is the only baseline where results of different repetitions vary.

We only experiment on a subset of **Open18**. To be more detailed, we discard all datasets with NaN values or only binary features. Furthermore, we discard all datasets where the logistic regression classifier of scikit-learn was not always able to converge on the full feature set. We use default parameters with the exception that we changed the maximal iterations from 100 to 1000 to increase the chance of convergence. Finally, we arrive at 8 out of 72 datasets of **Open18**. The results are depicted in Table 6.1.

Results and Discussion.

For all datasets, FSD outcompetes the random baseline and for 7 out of 8 datasets FSDC outperforms random selection. Hence, selecting features by their discriminability a reasonable aproach to identify relevant features for learning. Furthermore, for 6 out of 8 baselines, FSD surpasses all baselines. This indicates that our selection approach is competetive with established feature selection methods.

Adding the correlation-based feature selection to FSD does not increase performances and feature selection solely based on correlation coefficients leads to comparable low accuracies. Furthermore, RRFS, which, in our configuration, combines feature variances with dropping strongly correlated features does not surpass the selection of features solely based on variance. All this indicates that the incorporation of correlation between features is not useful in this scenario. It also stands out, that FSDC sometimes lead to the same accuracy than FSD. In these cases it stands to reason that FSD does not select any features that are dropped by FSDC as preprocessing.

Table 6.2: Results of our experiments on **OGBN**. We report, the accuracies for our method with (LSFSDC) and without (LSFSD) and for all baselines except of SPEC.

Dataset	$\lvert F\rvert$	Full	LSFSDC	LSFSD	Random	Corr	Vari	RRFS
arxiv	128	$.691 \pm .006$	$.536 \pm .005$	$\mathbf{.543} \pm .005$	$.500 \pm .021$	$.461 \pm .006$	$.543 \pm .008$	$.534 \pm .003$
products	100	$.748 \pm .006$	$.529 \pm .007$	$\mathbf{.545} \pm .005$	$.413 \pm .038$	$.395 \pm .006$	$.485 \pm .006$	$.519 \pm .008$
mag	128	$.387 \pm .004$	$\mathbf{.292} \pm .005$	$.283 \pm .005$	$.274 \pm .006$	$.269 \pm .004$	$.292 \pm .002$	$.282 \pm .004$

Overall, selecting only 10% often lead to a high drop compared to the original accuracy, indicating that such a small feature budget often not allows for sufficient feature selection. Thus, our experiment mainly gives insights in cases where feature selection is done because only using a small subset of features is computational feasible or to get new insights into the data and classification behavior. It is primarily not designed for feature selection to enhance classification accuracy. Here, the amount of selected features should be set higher. However, the competitiveness of our approach supports our hypothesis that discriminability of features is connected to their relevance for learning.

6.5.2 Feature Selection for Graph Neural Networks

We now evaluate to which extent our feature selection methods helps to select features for training graph neural networks. For this, we use *ogbn-arxiv*, *ogbn-mag* and *ogbn-products*[3] from Open Graph Benchmark [76]. These datasets are fundamentally larger then the ones from **Open18**, with a nodeset size of $169,343$ (**ogbn-arxiv**), $2,449,029$ (**ogbn-products**) and $1,939,743$ **(ogbn-mag)**.

To use LSFSD(C) we need to choose a support sequence. As in Chapter 5, we use log scale spacing. For this, we first choose a geometric sequence $\hat{s} = (s_1, \dots s_l)$ of length $l = 10{,}000$ with $s_1 = |\mathscr{X}|$ and $s_l = 2$ and use the support sequence s which results from $s' = (\lfloor|\mathscr{X}| + 2 - s_1\rfloor, \dots, \lfloor|\mathscr{X}| + 2 - s_l\rfloor)$ via discarding duplicated elements.

For training and classification, we again use a plain SIGN model [129] with one hidden-layer of dimension 512 and do $2-$hop neighborhood aggregation. We train with an Adam optimizer with learning rate of 0.001 and weight decay of 0.0001. We train for a maximum of 1000 epochs with a patience of 15 epochs with respect to validation accuracy. We use a batch size of 256. We dropout at the input layer with a probability of $0.1, 0.3, 0.5$ and at the hidden layer with probability of $0.5, 0.4, 0.5$ for **ogbn-arxiv**, **ogbn-products** and **ogbn-mag**, respectively.

[3]https://ogb.stanford.edu/docs/nodeprop/

We report test accuracies, displaying means and standard deviations over 10 rounds. We use all methods and baselines reported above. For SPEC, computation of the RBF kernel was not possible on our Server with 125 GB RAM due to memory costs. This was true for using scikit-learn as well with plain numpy. The results are depicted in Table 6.2.

Results and Discussion.

In this experiment, preprocessing via Section 6.3.2 indeed can help as LSFSDC lead to higher accuracies then using LSFSD for **ogbn-mag**. LSFSDC also surpasses feature selection based solely on correlations. Note, that RRFS, the other method that combines similarities between features and selection by some measure of feature relevance, do lead to fundamentally worse results. Thus, one can not follow that combining similarity and relevance is the general key to successful feature selection. This is supported by the fact, that feature selection solely based on feature variance leads to high accuracies. For **ogbn-products** and **ogbn-arxiv**, LFSD lead to the best performance, for **ogbn-mag**, the highest accuracy is reached with LSFSDC. In this experiment, our selection method is again competitive with the baselines and surpasses random selection. This supports the hypothesis that selecting features via their discriminability is meaningful.

6.6 Parameter Study on Maximal Errors

We now study the influence of the lengths of support sequences on the maximal error $E(s)$. For this, we generate the support sequence as in Section 6.5.2 with varying value l. We iterate l through $\{\lfloor 0.01 \cdot n \rfloor, \lfloor 0.02 \cdot n \rfloor, \dots, \lfloor 0.2 \cdot n \rfloor\}$, where n is the amount of points in the respective datasets. Here, if $l = \lfloor r \cdot n \rfloor$, we call again r the length of l. Note, that the final support sequence may have a lower amount of elements in l as we discard doubled elements. For all three **ogbn** datasets mentioned above, we display the maximal errors $\mathrm{E}(s)$ with the procedure mentioned in Section 6.3.2 and without. Since for **ogbn-arxiv** the exact computation of $\Delta(\mathscr{D})_f$ is possible, we also compute the "real" error $\mathrm{E}(s)^*$ for this dataset.

6.6.1 Results and Discussion

The results are depicted in Figure 6.1. We first note that the behavior of the maximal errors is not fundamentally effected by the decision for or against discarding strongly correlated features. In both settings, the curves are similar. Only the absolute values tend to be lower without discarding. Hence, the following observation and arguments hold for both cases.

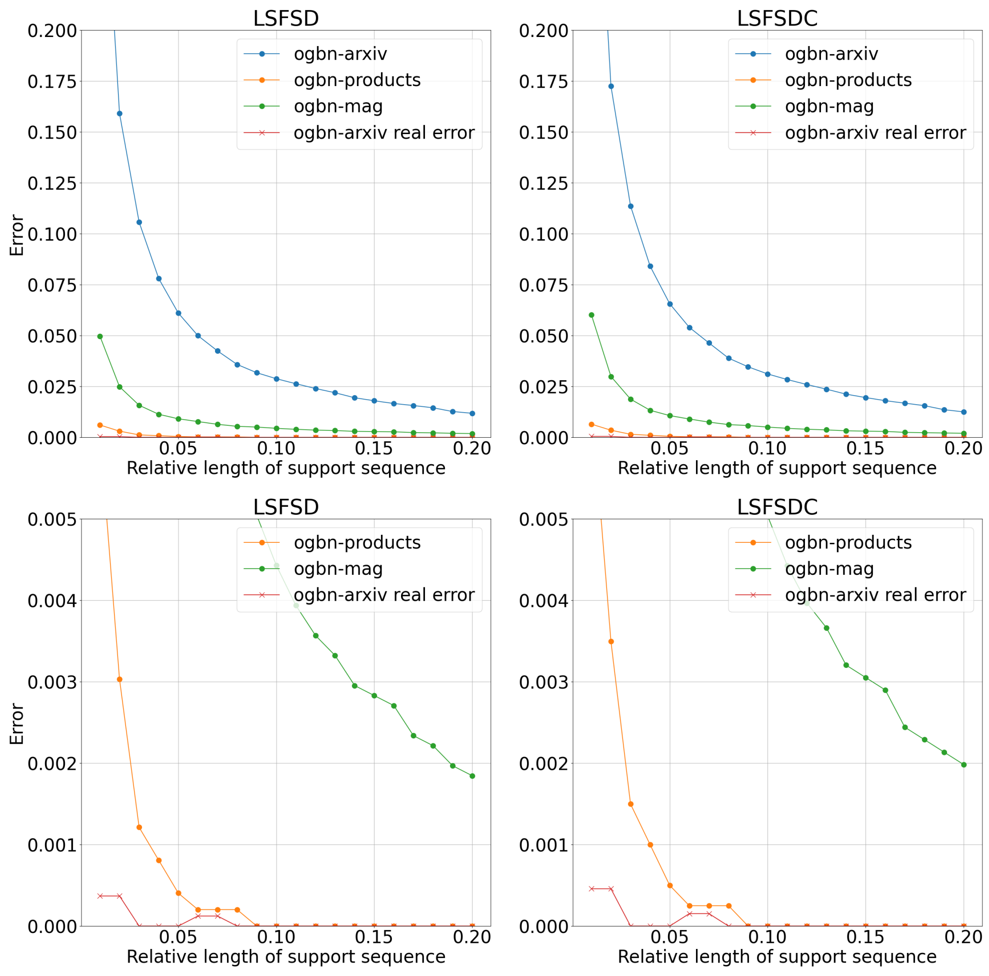


Figure 6.1: The maximal errors $\mathrm{E}(s)$ of all **ogbn** datasets. On the left, we display the results without discarding highly correlated features. On the right, we display the results with. For **ogbn-arxiv**, we also display the “real error” $\mathrm{E}(s)*$. The plots in the second row are zooms into the plots in the first row.

For the larger datasets, **ogbn-products** and **ogbn-mag**, the maximal error $\mathrm{E}(s)$ is negligible for comparable small relative support sizes. For relative support sizes of 0.1, the maximal mistakes are under 0.01. For **ogbn-arxiv** we have an maximal error of around 0.025 for this relative support size and overall higher maximal errors for all relative support lengths.

Comparing $\mathrm{E}(s)^*$ and $\mathrm{E}(s)$ for **ogbn-arxiv** it stands out, that the maximal error is a strong overestimation of the error $\mathrm{E}(s)^*$. As computing $\mathrm{E}(s)^*$ for both other datasets is not feasible, we can not verify whether this is also the case for them. However, we know because of Equation (6.6), that $\mathrm{E}(s)^*$ is bound by the maximal error which is already negligible for **ogbn-products** and **ogbn-mag**.

6.7 Conclusion and Future Work

We built on the developments from Chapter 5 to derive a novel unsupervised feature selection method that accounts for the intrinsic dimensionality of datasets. Our approach identifies those features that are able to discriminate data points and are thus responsible for taming the curse of dimensionality. Our experiments provide evidence that intrinsic dimension-based selection of features is competitive with well established procedures, occasionally outperforming them. This holds both on learning from real-valued and network data. Furthermore, we demonstrated that sampling techniques can be used to scale our feature selection method to more than millions of data points.

We identify as a natural next step for future work scenarios where the features are not given by coordinate projection, i.e., go beyond columns in tabular data. The generality of our modeling of features allows to encode, e.g., edge information for graphs. Thus, our method can be used for edge-sampling procedures that are important for training graph neural networks to learn from social networks. However, the question on how to encode edge information via feature functions is open.

While the present work emphasizes on understanding *data* by selecting features that break the curse of dimensionality, future work has to tackle as well the problem of understanding the behavior of specific *model classes*. This helps to further strengthen our understanding on the interplay between the success of learning and the presence of the curse of dimensionality.

Part IV

Machine Learning

Chapter 7

FCA2VEC: Representation Learning via Formal Concept Analysis

In this chapter, we present *FCA2VEC*, an approach that learns representations for partition classes of bipartite graphs with the help of formal concept analysis. For this, we assume that the bipartite network is represented by a formal context as defined in Definition 2.16. As this approach compresses complex network structure into simple vector representations, it allows to study networks from a condensed perspective.

7.1 Introduction

Learning real-valued vector representations from complex data structures has shown to be successful for a variety of tasks, such as link prediction, clustering, and information retrieval. Hence, today it is widely applied for a variety of data, such as text data [107, 108] or graphs [120, 65]. In this chapter, we contribute to this line of research by presenting FCA2VEC, an embedding approach that connects representation learning with *formal concept analysis* (FCA). For a given bipartite graph, FCA2VEC generates generates condensed representations with the formal concepts of the corresponding formal context. Our development is guided by two questions: First, how can vector space embeddings be exploited for coping more efficiently with problems from FCA? Second, to what extent can conceptual structures from FCA contribute to the embedding of formal context like data structures, e.g., bipartite graphs?

Equipped with this problem setting we show how the word2vec approach [107, 108] can be applied to generate vector embeddings for bipartite networks. To do so, we consider the corresponding formal context. We discuss how the formal concepts can be used to generate input data for training a neural network on learning representations. Our experiments show, that the generated embeddings can successfully be applied to cluster attributes by their implicational dependencies.

7.2 Related Work

There is an plethora of principle investigations for embedding *finite ordinal data* in real vector spaces, first of all *measurement structures* [134] (which we found via [155]) and *ordinal formal contexts* [154, 57, 45, 44]. A commonly used approach for the reduction of formal contexts is binary matrix decomposition [11, 10]. However, we are only aware of one FCA-based learning model that computes real valued vector representations of objects and attributes: in [34], the authors transfer latent semantic analysis (LSA) to the realm of FCA. While all methods mentioned above are indeed useful to generate real-valued representations, they are not tackling our problem of adapting modern neural network-based embedding approaches to the realm of FCA.

Our learning method is an adaption of word2vec [107, 108] and derived works such as node2vec [65]. Multiple approaches that aim to establish ties between FCA and neural networks were proposed in the past. For example, Kuznetsov et al. [92] uses concept lattices to derive neural network architectures for classification tasks while Caro-Contreras and Mendez-Vazquez [29] uses a neural network to compute concept lattices. However, to the best of our knowledge, there are no previous works of adapting the embedding approach of word2vec to formal contexts.

7.3 Word2Vec

We adapt the word2vec approach [107, 108] that generates vector embeddings for words from large text corpora. The model gets as input a list of sentences. It is then trained using one of two different approaches: predicting for a target word the context words around it (the *Skip-gram* model, called SG); predicting from a set of context words a target word (the *Continuous Bag of Words* model, called CBOW). In detail, word2vec works as follows.

Let $V = \{v_1, \dots, v_n\}$ be the vocabulary. We identify V as a subset of the vector space $\mathbb{R}^n$ via mapping the words to the standard basis of $\mathbb{R}^n$, i.e. via

$$\phi : V \to \mathbb{R}^n, v_i \mapsto e^i := (0,0,\underbrace{\dots,0,1,0,\dots}_{i-\text{th position}},0).$$

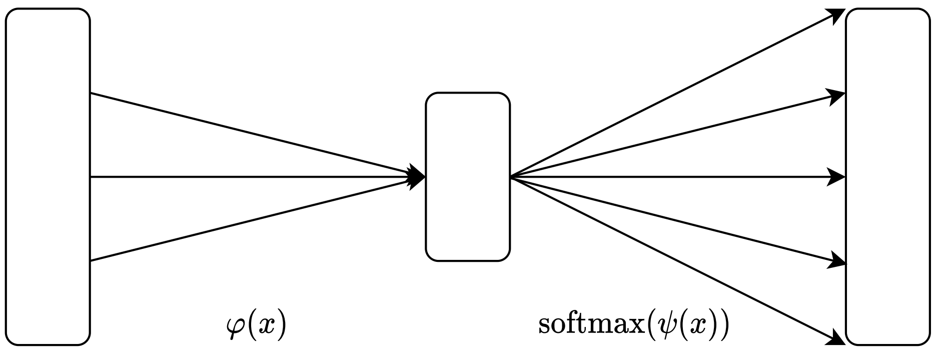


Figure 7.1: The structure of the word2vec architecture. The neural network consists of an input and output layer of size n and a hidden layer of size d, where $d \ll n$. The weights at the hidden layers are used as the embeddings.

This identification is commonly known under the term *one-hot encoding*. The learning task then is the following: Find for a given $d \in \mathbb{N}$ with $d \ll n$ a linear map $\varphi : \mathbb{R}^n \to \mathbb{R}^d$, i.e., in the case of working with row vectors, a a matrix $W \in \mathbb{R}^{n \times d}$ which obeys the goal: words that appear in similar contexts shall be mapped closely by φ. The final embedding vectors of the words of the vocabulary are given by the map

$$\Upsilon : V \to \mathbb{R}^d, v \mapsto \varphi(\phi(v)). \tag{7.1}$$

Here, we call Υ the *encoder* of V. Note, that it is well-established for the sake of simplicity to not distinguish between $v \in V$ and its one-hot encoding $\phi(v)$ and therefore also to not distinguish between Υ and φ. To obtain the embeddings, word2vec uses a neural network approach. This network consists of two linear maps and a softmax activation function, cf. Figure 7.1. The first linear function, the encoder, maps the input from $\mathbb{R}^n$ to $\mathbb{R}^d$. The second one, called the *decoder*, maps from $\mathbb{R}^d$ back to $\mathbb{R}^n$. In detail, the neural net function has the structure

$$f : \mathbb{R}^n \to \mathbb{R}^n, x \mapsto \mathrm{softmax}(\psi(\varphi(x))).$$

In the notion of Section 2.6, we have a 2-layer network with no biases, no activation at the hidden layer, and a softmax activation at the output layer. In this notation the hidden layer is used to determine the embeddings. We refer the reader again to Figure 7.1. In the realm of word2vec, Mikolov et. al. [107] proposed two different approaches to obtain the parameters of ϕ, ψ from input data. Those are called the Skip-gram and the Continuous Bag of Words architecture. We recollect them in the following.

7.3.1 The Skip-Gram and the Continuous Bag of Words Architecture

The SG architecture trains the network to predict for a given *target word* the *context words* around it. Each training example consists of a target word and a finite sequence of context words. We formalize these as tuples

$$(t,(c_i)_{i=0}^{l}) \in V \times V^{<\mathbb{N}},$$

where $V^{<\mathbb{N}}$ is the set of finite sequences of elements of V. The SG architecture generates the input-output pairs

$$(\phi(t),\phi(c_0)),\dots,(\phi(t),\phi(c_l))$$

as training examples, where ϕ is the one-hot encoding function, as introduced above. In the CBOW model, in contrast to SG, the training pairs are generated differently from $(t,(c_i)_{i=0}^{l})$. We take the middle point of the vectors $\phi(c_0),\dots,\phi(c_l)$ and try to predict the target word $\phi(t)$, hence the generated input-output training pair is

$$(\frac{1}{l}\sum_{i=0}^{l}\phi(c_i),\phi(t)).$$

Both architectures employ the same kind of loss function to learn the weights of W and U. The error term is computed through cross-entropy loss. A detailed explanation can be found in Rong [128].

In word2vec, the pairs of target word and context words are generated from text data sequences, i.e., lists of sentences. The word2vec approach has a window size $m \in \mathbb{N}$ as a parameter, i.e., for a given sentence $s=(w_i)_{i=0}^{l} \in V^{<\mathbb{N}}$ pairs of target word and context word sequences are defined in the following manner. For every $i \in \{0,\dots,l\}$ a reduced window size $m_i \in \{0,\dots,m\}$ is chosen randomly and the pair $(w_i,(w_{i+k})_{k=-m_i,k\neq 0}^{m_i})$ is used as pair of target word and context word sequence.

Note, in the case of word embeddings, the size of the vocabulary often reaches a level where computing the softmax is computationally infeasible. Hence, the softmax layer is often approximated/replaced by one of the two following approaches. The *hierarchical softmax* [113] stores the elements of the vocabulary in a binary Huffman tree and then only uses the values to the path to an element to compute its probability. Another approach presented in [108] is *negative sampling*, which uses the sigmoid function. In the experimental part of this work we deal with formal contexts of a size where applying the softmax layer is possible. In the following, we will restrict ourself to the SG architecture and will discuss how it can be transferred to the realm of formal concept analysis.

7.4 Object2Vec and Attribute2Vec

We adapt word2vec to the domain of formal concept analysis and study its aptitude in a different area. Namely, we introduce an approach to compute embeddings of objects and attributes that incorporates proximity. Here, nearness of two objects or attributes refers to the amount of concepts that include both.

The idea of adapting word2vec to non-text mining problems is a common approach these days. Particular examples for that are node2vec [65] and deepwalk [120]. In the realm of networks, it was shown that SG based architectures for node embeddings can beat former approaches that use classic graph measures. They significantly enhanced node classification and link prediction [120, 65] tasks. To do so, they interpret nodes as words for their vocabulary, use random walks through the graphs to generate "sentences", and then employ word2vec. Since it is common to focus solely on the SG approach in the realm of node embedding [120, 65], we will do the same from this point on.

In the following we present an approach to use the concepts of a given formal context to generate embeddings of the object set or attribute set. Referring to its origin, we name our novel methods *object2vec* and *attribute2vec*, respectively. The family of both of them is what we call *FCA2VEC*. Both methods work in the same manner with the only difference that object2vec generates embeddings for objects and attribute2vec generates embeddings for attributes. We therefore focus on object2vec. For attribute2vec, only the roles of the objects and attributes in the following explanations have to be switched. The basic idea of object2vec is to interpret two objects to be more close to each other, if they are included in more concept extents together. Hence, the set of extents of a formal context is used to generate a low dimensional embedding of the object set G.

In the following we explain how to adapt the SG architecture to the realm of formal concept analysis. We show how to generate multisets of training examples from a given formal context. As an analogy for target word and context words we introduce target object and context object sets. From this we can draw pairs as already done in SG.

7.4.1 Skip Gram in the Realm of Object2Vec

Let $\mathbb{K} := (G,M,I)$ be a (finite) formal context. The vocabulary is given by $G = \{g_1, \dots g_n\}$. Furthermore, let be $\phi : G \to \mathbb{R}^n, g_i \mapsto e^i$ the one-hot encoding of our vocabulary (objects). We derive our *training examples* from the set

$$\mathrm{T}(\mathbb{K}) := \{(a, A \setminus \{a\}) \mid a \in A, |G| > |A| > 1, \exists B \subseteq M : (A,B) \in \mathfrak{B}(\mathbb{K})\},$$

	1	2	3	4	5	6	7	8	9
a				×	×	×			×
b	×	×	×			×			
c	×	×		×		×		×	
d	×	×	×	×		×			
e	×		×			×			
f				×	×	×	×		
g			×	×	×	×	×		
h			×		×	×	×		

Figure 7.2: Formal context of the classical *Living Beings and Water* example from [58].

where every element is a pair of a *target object* and some extent in which a is element of. More specifically, we remove a from this extent. We interpret then $A \setminus \{a\}$ as the *object context set*. The word "context" here refers to the word2vec approach and is not be confused with "formal context". Note that we do not generate any training examples from the concept (G, G') since the extent G does not provide any information about the formal context.

In the SG model, the input and output training pairs are generated from target object and an object context set pairs. From such a pair $(t, C) \in \mathrm{T}(\mathbb{K})$. the generated training examples are given by

$$\mathrm{T}_{\mathrm{SG}}(t, C) := \{(\phi(t), \phi(c)) \mid c \in C\}.$$

Using this it is possible for some pairs $(t_1, C_1), (t_2, C_2) \in \mathrm{T}(\mathbb{K})$ where we have $(t_1, C_1) \neq (t_2, C_2)$ that

$$\mathrm{T}_{\mathrm{SG}}(t_1, C_1) \cap \mathrm{T}_{\mathrm{SG}}(t_2, C_2) \neq \emptyset.$$

Hence, samples can be generated multiple times in this setup. To give an impression of our modeling we furnish the following example.

Consider a classical formal context from [58], called "Living Beings and Water", which we depicted in Figure 7.2. We map the objects with the one-hot encoding: $\phi : G \mapsto \mathbb{R}^8$, with $\phi(a) = e^1, \phi(b) = e^2, \ldots, \phi(h) = e^8$. Using this we easily find a training sample from T_{SG} which is generated two times. Consider the concepts $(\{a, f, g\}, \{4, 5, 6\})$ and $(\{f, g, h\}, \{5, 6, 7\})$. The first concept generates the pairs of target object and object context sets $(a, \{f, g\})$ as well as $(f, \{a, g\})$ and $(g, \{a, f\})$. The second formal concept generates the pairs $(f, \{g, h\}), (g, \{f, h\})$ and $(h, \{f, g\})$. If we train in the SG architecture we derive from the pair $(f, \{a, g\})$ the training examples (e^6, e^1) and (e^6, e^7). Also, from the pair $(f, \{g, h\})$ we derive the examples $(e^6, e^7), (e^7, e^8)$. Hence, the training example (e^6, e^7) is shown to the neural network at least twice per epoch.

Algorithm 7.1: The pseudocode of generating training examples for object2vec. The algorithm takes a formal context as input. It returns a list of pairs to train the neural network.

```
Input  : a formal context $(G,M,I)$
Output : A list $L$ of training examples.
1 $L \leftarrow [\,]$
2 $L_{\text{ext}} <-$ list of extents of $(G,M,I)$ (excluding $G$).
3 forall $A$ in $L_{\text{ext}}$ do
4     forall $o$ in $A$ do
5         forall $\check{o}$ in $A$: do
6             if $o \neq \check{o}$ then
7                 add $((\phi(o), \phi(\check{o})))$ to $L$
8 return $L$
```

7.5 Clustering Attributes with FCA2VEC

In FCA, implications on the set of attributes of a formal context are of major interest. While computing the canonical base, i.e., the minimal base of the implicational theory of a formal context, is often infeasible, one could be interested in implications of smaller attribute subsets. This leads naturally to the question of how to identify attribute subsets that cover a large part of the canonical base. In detail, the resulting task is as follows: Let (G,M,I) be a formal context and $\mathscr{L}$ the canonical base (G,M,I). Using a simple clustering procedure (in our case k-means), find for a given $k \in \mathbb{N}_0$ a partitioning of M in k clusters such that the ratio of implications completely contained in one cluster is as high as possible. We investigate to which extent our proposed approach attribute2vec maps attributes closely that are meaningful for the aforementioned task. We conduct this research by computing an embedding via attribute2vec and run the k-means clustering algorithm on top of it. For this experiment, we use the **wiki44k** dataset taken from Ho et al. [74] and then adapted by Hanika et al. [69]. It consists of relational data extracted from Wikidata in December 2014. Even though it is constructed to be a dense part of the Wikidata knowledge graph, it is relatively sparse for a formal context. It consists of **45021** objects, **101** attributes, has a density of 0.04 and **21923** formal concepts. Our experimental pipeline looks as follows.

Applying attribute2vec on wiki44k We start by computing vector embeddings of the **wiki44k** attributes using attribute2vec with SG.

K-means clustering We use the computed embedding to cluster our attributes with the k-means algorithm. As implementation we rely on the scikit-learn software package.

For the initial clustering, we use the so called *k-means++* technique by Arthur and Vassilvitskii [6]. The method from scikit-learn runs internally for ten times with different seeds and returns the best result encountered with respect to the costs of the clusters.

Computation of the intra-cluster implications For clusterings $\mathscr{C}$, we compute the ratio of *intra-cluster* implications. An implication drawn from the canonical base, i.e., $A \to B \in \mathscr{L}$, is called intra-cluster if there is some $C \in \mathscr{C}$ such that $A \cup B \subseteq C$. The canonical base of **wiki44k** has the size **7040**.

Random clusterings To evaluate the ratios computed in the last step we use the following baseline approaches. As a first baseline we make use of a random procedure. This results in a random clustering of the attribute set. Using an arbitrary random clustering with respect to cluster sizes is unreasonable for comparison. Hence, for each k-means clustering obtained above we generate 10 random clusterings of the same size and the same cluster size distribution. For those we also compute the intra-cluster implication ratio.

Naive k-means clustering As a second baseline we envision a more sophisticated procedure. We call this the "naive" clustering approach. In this setting we encode an attribute m through a binary vector representation using the objects from $\{m\}'$. We then run ten rounds of k-means and compare the results with the attribute2vec approach.

Setting. We repeat all experiments for 10 times and report means and standard derivations. We do mini-batching with a batch-size of 512 over 200 epochs and report results for $k \in \{2,3,5,10,20\}$. We use an embedding dimension of $d = 128$, which is a common choice in the realm of node embeddings based on word2vec [65, 120]. We use a learning rate scheduler with linear decay and optimize the learning rate from γ from the possibilities $\{0.0025, 0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5\}$. The task is easier, if the cluster size distribution is imbalanced, i.e., if nearly all attributes are in one cluster. We therefore decided to use the cluster size distribution as our optimization criterion. To be more specific, we use the parameters where the size of the largest cluster is minimized. Note, that this yields an unsupervised optimization criterion that does not depend on the implication data we use for testing. The results can be found in Table 7.1.

Results and Discussion. Attribute2vec outperforms both the random and the naive baselines. This indicates, that the generated embeddings indeed capture implicational knowledge

Table 7.1: Results of clustering objects with attribute2vec.

k	2	3	5	10	20
Attribute2vec	**.3178** ± .0037	**.3158** ± .0000	**.2043** ± .0600	**..0494** ± .0658	**.0062** ± .0026
Naive	.0158 ± .0000	.0155 ± .0001	.0050 ± .0037	.0035 ± .0005	.0010 ± .0005
Random	.0225 ± .0140	.0071 ± .0034	.0034 ± .0020	.0006 ± .0003	.0004 ± .0004

of the formal context. As application for our attribute2vec approach we envision the computation of parts of the canonical base of a formal context. Taking the average maximum cluster sizes into account we claim that this application is reasonable. We do not consider the then necessary computation of all formal concepts as a disadvantage. The computation of the canonical base is in general far more complex than computing the set of all concepts and our embedding. Note, that the naive clustering baseline performs fundamentally worse then attribute2vec. We presume that *k*-means clustering applied to the binary representation vectors is not a particularly useful approach to capture implicational knowledge.

7.6 Conclusion and Future Work

In this chapter we presented FCA2VEC, a first approach for bringing data and conceptual structures from FCA to the realm of embedding techniques based on word2vec. We specified ourself to the well-known kip-gram architecture and showed how it can be used to generate training examples that reflect which objects or attributes are contained in the same formal concepts. In our investigation we have found convincing experimental evidence that FCA-based methods can profit from word2vec-like embedding procedures as they can be used to cluster for example attributes into partitions such that a large amount of implications is intra-cluster. Thus, these representations are a reasonable tool to investigate networks from a condensed perspective. This is in particular true fo the identification of implicational knowledge, as studied in FCA.

The ideas for object2vec and attribute2vec have the limitation that they do require the computation of the concept lattice. In future work we will investigate if this obligation can be weakened through approximations or computing embeddings only with a subset of the concept lattice or by computing only a sample of object or attribute derivations.

Another direction would be to step away from static real-valued embeddings and to neural network architectures that incorporate the concept lattice structure in different layers. Here, the distant vision would be to have a family of *concept neural networks* as an analogy to graph neural networks.

Chapter 8

LG4AV: Combining Language Models and Graph Neural Networks for Author Verification

In this chapter we discuss possibilities to learn representations for a specific real-world scenario. More specifically, we consider the case where documents with available text information are connected to (potential) authors. Additionally, we will incorporate known edges between different authors. The resulting neural network has the aim to represent the graph structure such that potential edges between authors and unknown documents can be predicted. Since we here represent a complex network structure via real-valued weights of neural network, this approach falls into the category of condensed perspectives on networks. Our approach is motivated by the task of verifying authorships of academic publications and therefore has its application in the realm of bibliometrics.

8.1 Introduction

Evaluation of research strongly depends on bibliometric databases. Today, they are used for the assessment of productivity and impact of researchers, conferences and affiliations. Because of their rising relevance for the evaluation of the scientific output of individual authors, it is crucial that the information which is stored in such databases is correct. However, with the rapid growth of publication output [19], automatic inspections and corrections of information in bibliometric data is needed. A major challenge in this area is *authorship verification* (AV), which aims to verify if a document is written by a specific author. In general, AV is widely investigated [88, 149, 67], with a majority of existing work handling author verification by capturing writing styles [30, 67], assuming that they are unique among different authors. This assumption does not hold in environments where the available texts are short and contain uniform language patterns. An example of this is given by author verification tasks for scientific documents. In such settings, the availability of full texts is rare because bibliometric datasets often contain only abstracts and titles. In such scenarios the variety of writing styles and linguistic usage is rather limited.

Additionally, the focus in AV research is on documents with one author, while verification of multi-author documents is seldom done. Here, the information about known multi-authorships can enhance the verification process because it provides a meaningful graph structure. For example, scientific authors are more likely to write papers that would also fit to their co-authors and Twitter users are expected to post about the same topics as the persons they follow. The incorporation of such graph structures is rarely investigated.

We fill this gap with LG4AV. Our novel architecture combines language models and graph neural networks to verify whether a document belongs to a potential author. This is done without the explicit recap of the known documents of this author at decision time which can be a computational bottleneck. This is especially true for authors with a large amount of known documents. Additionally, LG4AV does not rely on any handcrafted stylometric features.

By incorporating a graph neural network structure into our architecture, we use known relations between potential authors. In this way, we are able to account for the fact that authors are more likely to turn to topics that are present in their social neighborhood. We experimentally evaluate the ability of our model to make verification decisions in bibliometric environments and we review the influence of the individual components on the quality of the verification decisions.

8.2 Related Work

Authorship verification is a commonly studied problem. PAN@CLEF[1] provides regular competitions in this realm. However, their past author verification challenges were based on a setting where either small samples of up to ten known documents for each unknown document were provided (2013-2015) or pairs of documents were given where the task was to decide whether they were written by the same person (2020, 2021). Both scenarios are not applicable for bibliometric environments, where the amount of known document can reach up to hundreds.

[1]https://pan.webis.de/shared-tasks.html

Many well-established methods for author verification develop specific hand-crafted features that capture stylometric and syntactic patterns of documents. For example, Hürlimann et al. [77] uses features such as sentence-lengths, punctuation marks and frequencies of n-grams to make verification decisions. While the use of n-grams was already studied in earlier work [83], there are still recent methods that build upon them [126]. Another well established approach is given by Koppel et al. [89] where the authors successively remove features and observe how this reduces the distinction between two works. This approach is still known to be the gold standard [126, 14]. Despite its advantages, it is known to perform worse on short texts. Therefore, Bevendorff et al. [14] proposes a modification that is also applicable to shorter texts. However, this work experiments with documents of 4,000 words per document, which is still much longer then abstracts of scientific publications.

Recently, methods based on neural network architectures, such as RNNs and transformer models, emerged [8, 9]. Note, that both of these approaches need to train head layers for each individual author. This makes them impractical for AV in bibliometric data where thousands of authors has to be considered.

One of the few works that experiments with bibliometric data is [67], which uses full text of a small subset of authors and ignores co-authorship relations. Most other works that deal with bibliometric data tackle the closely related problem of *authorship attribution* (AA), i.e., with questions of the kind "who is the author of *d*" instead of "is *a* author of *d*" [72, 20, 28].

One of the few works in the realm of AV that takes into account that research papers are multi-author documents is Sarwar et al. [132]. In their work, the authors derive a similarity based graph structure of text fragments for authorship attribution for multi-author documents. In contrast, our aim is to incorporate past co-author relations to verify potential authorships.

8.3 Problem

In the following, we formulate the problem we consider, introduce our architecture and discuss the individual components of LG4AV.

Let t be a fixed time point and let $G = (A, E)$ be a graph with $A = \{a_1, \dots, a_n\}$ being a set of authors and $E \subseteq \binom{A}{2}$ a set of undirected and unweighted edges that represent connections until t. Additionally, let D be a set of documents. Let, for all authors $a \in A$, be $D(a) \subseteq D$ the set of their known documents until t. Let U be a set of documents created after t with unknown authorships. The goal is to verify for a set $P \subseteq A \times U$ of potential author-document pairs, whether for each $(a, u) \in P$ a is an author of the unknown document u. More formally, the aim is to find for each tuple a verification score $f(a, u) \in [0, 1]$.

Hence, we aim to infer from the data present at *training time* information about authors to verify potential documents of them that occur at *testing time*. Our formulation differs from the usual setting where the problem is either broken down to sequences of pairs $(D_i, d_i)_{i=1}^{l}$ where the task is to determine for each $i \in \{1, \ldots, l\}$ if the *unknown document* d_i is from the same author as the set of *known documents* D_i. These settings are closely connected in the sense that each author a can be interpreted as the set of his known documents $D(a)$ at training time. However, approaches adopted to this setting often assume to have already pairs of known sets and unknown document [77] at training time or they explicitly use the set of known documents for verification [126, 78, 136]. In contrast, we train a neural network on incorporating information of known documents of an author to make it possible to make verification decisions at testing time without explicitly recapping these documents for each unknown document. This is especially useful in settings where the amount of unknown documents can be large for some authors. Here, a need to explicitly use all these documents for every single verification of an unknown document can be a computational bottleneck.

8.4 Combining Language Models and Graph Neural Networks for Author Verification

We develop an end-to-end model to tackle the author verification problem. For this, we additionally assume to have for each author $a \in A$ a vector representation $x_a \in \mathbb{R}^s$. At training time, our model gets as input pairs $(a, d) \in A \times D$ and is trained on predicting whether a is author of D, i.e., $d \in D(a)$. For inference at testing time, the model gets as input pairs $(a, u) \in P$ and decides whether a is author of u by computing a verification score $f(a, u)$. For both training and testing, author-document pairs (a, d) are forwarded through the network in the following manner. We add a special token to represent the current author to the beginning of d. The resulting document is then guided through a language model. Additionally, we incorporate a graph neural network structure by

1. computing vector representations of $a \in A$ that depend on its graph neighbors,

2. combining these vector representations with the output of a language model

3. and forwarding through a fully connected layer to get a verification score.

In the following, we discuss the individual components and give a detailed explanation of the network inference. A sketch of LG4AV is given by Figure 8.1.

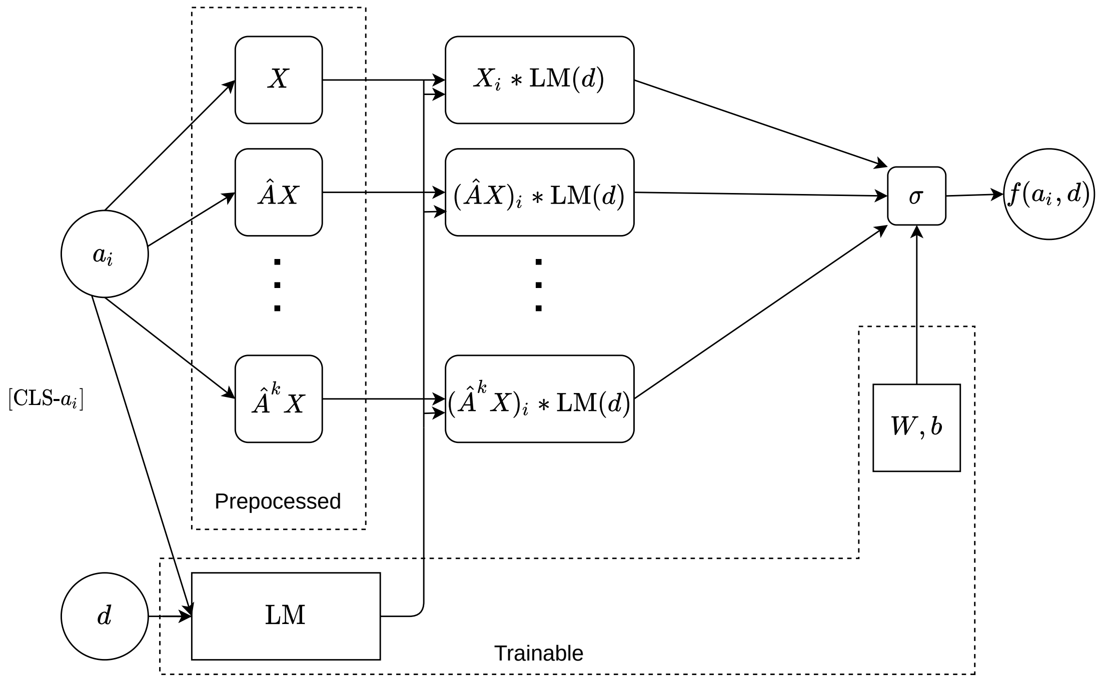


Figure 8.1: Forward step. LG4AV gets as input an author a_i and a document d. The author specific cls-token is added to the front of d which is then feed through the language model (which is BERT in our case). The i-th rows of $X, \hat{A}X, \ldots, \hat{A}^kX$ are individually component-wise multiplied with the output of LM. The resulting vectors are concatenated and fed through a fully connected layer with a sigmoid activation.

8.4.1 The Language Model

We choose a neural-network based language model build upon the transformer architecture [150]. More specifically, we choose the standard BERT [40] model which we assume the reader to be familiar with. While this model is originally intended for sentences, it is possible to feed arbitrary sequences of specific maximum length (512 tokens for regular BERT models) through the network. Hence, we feed the full text of the document at once through BERT and extract the output of the first token (a specific classification token, denoted by [CLS-a]). This practice is established and has already been applied to abstracts of scientific documents [35] as well as social media data [163]. To sum up, our language model can be interpreted as a map

$$\mathrm{LM} : D \cup U \to \mathbb{R}^m.$$

To combine the output of the language model with the neighborhood aggregated author vector representation, we need to ensure that the output of the language model has the same size as the author features, i.e., $m = s$. Hence,we extend LM by a linear layer on top of the BERT model, if needed.

8.4.2 Author Tokens

To give the language model information about the current author $a \in A$, we replace the regular [CLS] token by an author-dependent classification token [CLS-a]. Hence, the information of the current author is encoded into the input of the language model. Roughly speaking, if a pair (a,d) is fed through the whole network at training time, the author information is not only incorporated into the LM layers via backpropagation. Instead, LM also sees a in the forward step. To nourish from the optimization of the [CLS] token that was done in the pre-training procedure of BERT, we initialize for each author $a \in A$ the token-embedding of [CLS-a] with the token-embedding of [CLS].

8.4.3 Choice of the Graph Neural Network

To avoid the neighborhood exploding problem, we adapt the idea of SIGN [129]. As explained in Section 2.7.2, SIGN uses input layers with different neighborhood aggregations of the form

$$X \mapsto \hat{A}^k X.$$

We combine all these inputs with the language model outputs for our model. Our complete architecture looks as follows.

8.4.4 LG4AV

The exact network inference is done in the following manner. Let $X \in \mathbb{R}^{n \times s}$ be the feature matrix of all authors where the $i-th$ row represents the feature vector of the i-th author, i.e. $X_i = X_{a_i}$. Let $k \in \mathbb{N}$. Let $\hat{A}$ be the normalized adjacency matrix from Definition 2.25 of the co-author graph. For a given pair (a_i,d) of an author and a document the network inference is done in the following manner. For all $l \in \{0,\dots,k\}$, we concatenate the vectors

$$v_l(a_i,d) := (\hat{A}^l X)_i * \mathrm{LM}(d).$$

Here, $\mathrm{LM}(d))$ is also treated as a row-vector and $*$ denotes the element-wise product. To derive a verification score from this concatenated vector, we feed $v(a_i,d)$ into a fully connected layer with weight matrix $W \in \mathbb{R}^{s(k+1)\times 1}$, a bias $b \in \mathbb{R}$ and sigmoid activation. Hence, the full network inference of LG4AV for one training example is given by the equation

$$f(a_i,d) := \sigma(v(a_i,d))W + b).$$

Training of LG4AV. For training we use all pairs $(a,d) \in A \times D$ with $d \in D(a)$ as positive examples. For each $a \in A$ we sample $|D(a)|$ documents $d \in D \setminus D(a)$ to generate negative examples. We use binary cross entropy as loss function.

8.5 Experiments

We use a dataset which contains publication information of the German and international AI research community [87]. It contains titles and abstracts from Semantic Scholar [4] which are needed for LG4AV but are not included in DBLP[2]. The relations between authors and papers are based on DBLP which is, in our experience, comparably accurate with good name disambiguation. This is crucial to prevent wrong authorship information in the data itself.

We use the dataset of the German AI researchers as our first dataset. As a second dataset, we extract all authors with publications at the KDD conference and all their publications from the dataset of the international AI researchers. We refer to the first as the **GAI** and to the second as the **KDD** data. Basic statistics of the resulting data can be found in Table 8.1. We generate training, validation and testing data for both datasets in the following manner.

- We discard all publications without an English abstract.
- For each publication, we use the title and the abstract as input for the language model and concatenate them via a new line char to generate the text representation.
- We build the co-author graph of all authors until 2015. From this co-author graph, we discard all author nodes that do not belong to the biggest connected component. Let A be the set of authors that are nodes in this graph. We use this graph for the neighborhood aggregation. We denote the set of publications until 2015 of these authors with D_{train}.
- We generate for all authors and each of their publications a positive training example. For all authors, we then sample papers from D_{train} which they are not an author of. We sample in such a way that we have for each author an equal amount of positive and negative examples.
- We use data from 2016 for validation. More specifically, we use for all authors $a \in A$ all publications that they have (co-) authored in 2016 as positive validation examples. Let D_{val} be the set of these publications. We sample for all authors papers from D_{val} that they are not author of as negative validation examples. Again, we sample in such

[2]https://dblp.org/

Table 8.1: Basic statistics of the datasets. We display from left to right: 1.) The number of authors in A, 2.) the number of edges in the co-author graph of these authors at training time, 3.) the number of training examples, 4.) the number of validation examples, 5.) the number of test examples.

	# Authors	# Edges	# Train	# Validation	# Test
GAI	1669	4315	175118	14314	41558
KDD	3056	9592	254096	19976	61804

a way that we have for each author an equal amount of positive and negative validation examples.

- We use publications from 2017 and newer to analogously generate test data.

Baselines

N-Gram Baseline. This baseline is strongly inspired by the baseline script of the AV challenge of PAN@CLEF 2020. For all authors we generate a "superdocument" by concatenating all their documents that are available for training. For all pairs of authors a and documents d at validation and testing time, we measure the similarity between d and the superdocument of a. If the similarity is above a given threshold t, we classify the pair as a positive example. To measure similarities, we build a character-based n-gram TF-IDF vectorizer upon all papers available at training time, using only the 3000 most frequent n-grams across the documents that the vectorizer is built on.

We tune n via grid-search on $\{1 \dots 10\}$ and choose the value that corresponds to the highest AUC on the validation set. We tune the threshold on the set $\{\frac{1}{999}i - \frac{1}{999} \mid i \in \{1, \dots, 1000\}\}$. Note, that this means to sample 1000 evenly spaced points in $[0,1]$. We also test the median of the distances of the validation examples as threshold. Since the AUC is independent of this threshold, we tune on the validation F1 score after the best n is chosen.

GLAD [77]. This method is intended for pairs of the form (D,d) where D is a set of documents and d is a single document. Note, that GLAD needs such pairs already for training. Since our data consists of pairs (a,d) with a an author and d a document, we build training examples for GLAD in the following manner. Let P_{train} be the set of all author-document pairs available at training time and let, for all (a,d), be $l_{(a,d)} \in \{-1,1\}$ the label of that pair. For each author a we collect the set $D_{a,+} := \{d \mid (a,d) \in P_{\text{train}}, l_{(a,d)} = 1\} = \{d_{a,+,0}, \dots, d_{a,+,m}\}$ of positive and the set $D_{a,-} := \{d \mid (a,d) \in P_{\text{train}}, l_{(a,d)} = -1\} = \{d_{a,-,0}, \dots, d_{a,-,m}\}$ of negative training examples. For each $i \in \{0, \dots, m\}$, we train GLAD with $(D_{a,+} \setminus \{d_{a,+,i}\}, d_{a,+,i})$ with a positive label and $(D_{a,+} \setminus \{d_{a,+,i}\}, d_{a,-,i})$ with a negative label. For validation and testing, we replace pairs (a,d) by $(D_{a,+}, d)$.

GLAD works as follows. For each pair (D,d) a vector representation is computed that consists of features that are solely build on D or d, such as for example average sentence length and joint features, which are build on D and d, such as entropy of concatenations of documents of D and d. This vector representations are then fed into a support-vector machine. While the authors of [77] uses a linear support-vector machine with default parameter setting of scikit-learn [119], we enhance GLAD by tuning the c-parameter of the support-vector machine and additionally experiment with radial kernels where we tune the γ-parameter. For both parameters, we grid search over $\{10^{-3}, \dots, 10^{3}\}$ on the validation AUC.

RBI [126]. The ranking-based impostors method verifies a pair (D,d) with the help of a set D_e of external documents. To use exactly the information available for training and thus have a fair comparison, we use $D_{a,+}$ as the known documents and $D_{a,-}$ as the external documents for each pair (a,d).

By studying for each $d_i \in D$ how many documents of D_e are closer to d than d_i is to d, the impostors method computes a verification score where pairs with higher scores are more likely to be positive examples. To compute vector representations for documents, we stick to the procedure in Potha and Stamatatos [126]. We choose the following parameters for RBI. We grid search $k \in \{100, 200, 300, 400\}$, choose cosine similarity as the similarity function and select the aggregation function between mean, minimum and maximum function. For the meaning of the parameters, we refer to Potha and Stamatatos [126]. We use the AUC score on the validation data to choose the best parameters. To derive binary predictions from the verification scores, we use the median of all verification scores from the validation data.

Siamese BERT (S-BERT) [148] In this method, pairs of documents are fed through BERT and the network is trained to put document pairs close together which are from the same author. To derive examples from a training pair (a,d), we sample from $D_{a,+} \setminus \{d\}$ 3 documents d_1, d_2, d_3 to generate the training examples $(d_1,d), (d_2,d), (d_3,d)$. If (a,d) is a validation or test example we set the distance of (a,d) to the mean of the distances of $(d_1,d), (d_2,d), (d_3,d)$.

This model is trained with a linear decaying learning rate starting at $5 \cdot 10^{-5}$ and weight decay of 0.01. We use a contrastive loss function with a margin of 0.1 and cosine distance because this led to the highest AUC score in Tyo et al. [148]. To choose the threshold t which separates positive and negative examples at validation and testing time, we grid search over $\{0, 0.02, \dots, 1.998, 2\}$ on the validation F1 score and classify all pairs as positive which

Table 8.2: Results. For S-Bert, we report means over runs with an AUC over 0.51.

	GAI			**KDD**		
	AUC	ACC	F1	AUC	ACC	F1
N-Gram	.8624	.7874	.7817	.7592	.6887	.7008
GLAD	.8329	.7510	.7167	.7323	.6698	.6203
RBI	.8251	.7478	.7391	.7452	.6823	.6647
S-BERT	.9196	.8492	.8516	.8207	.7373	.7602
LG4AV	**.9247**	**.8541**	**.8569**	**.8522**	**.7675**	**.7808**

have a distance not higher then t. We use a batch-size of 2 and accumulate 4 batches for an effective batch size of 8. As S-BERT tends to lead to unstable results, which is commonly observed for BERT models [115, 161], we do 10 runs for both datasets and report mean values of the runs which lead to a reasonable solution, i.e., an AUC over 0.51.

Configuration of LG4AV and Implementation Details

We work with a LG4AV model with $k = 2$ which is a common choice in the realm of GNNs [85, 68]. To derive classification decisions for computing F1 and accuracy scores, we grid search a threshold t over $\{0, 0.002, \dots, 0.998, 1\}$ on the validation F1 score.

We use the "regular" BERT base uncased model and train for 3 epochs. After the element-wise multiplication of the BERT output and the text features, we dropout with probability of 0.1. We use ADAM with weight decay of 0.01 and a learning rate of $2 \cdot 10^{-5}$ with linear decay and a batch size of 4. We do gradient accumulation of 4 for an effective batch size of 16. To generate features for each author, we feed their known documents through the not fine-tuned BERT model and build the mean point vector of the vector representations of the [CLS] tokens. Because of the instability of BERT fine-tuning [115, 161], we do 10 runs of LG4AV and report mean scores.

Results and Discussion

The results can be found in Table 8.2. LG4AV outperforms all baselines. We especially point out that LG4AV also leads to slight improvement over S-BERT, which also uses language models and is, because of the sampling procedure that increases the amount of training examples, more time-expensive. As we replace one training example with 3 new samples for S-BERT, an epoch lasts about a factor 3 longer for S-BERT.

Note, that LG4AV uses for all authors their papers until 2015 to build the positive training examples and for computing their feature vector. Hence, considering the positive examples,

Table 8.3: Results. We report for all models the AUC-Score, the accuracy and the F1-score. We compare the regular LG4AV with $k = 2$ (LG4AV-2) with a model with $k = 0$ (LG4AV-0) and a model with $k = 2$ with freezed BERT layers (LG4AV-F)

	GAI			**KDD**		
	AUC	ACC	F1	AUC	ACC	F1
LG4AV-F	.8384	.7576	.7767	.7621	.6866	.7110
LG4AV-0	.9207	.8492	.8519	.8465	.7619	.7771
LG4AV-2	**.9247**	**.8541**	**.8569**	**.8522**	**.7675**	**.7808**

the network is trained on verifying author-document pairs where the document is additionally used to build the features of this author. Thus, one could expect that the availability to generalize to unseen documents is limited. However, the results on the test set, which contains only documents not used for the features vectors, show that this is not the case.

The performance gap between the datasets is remarkable. Because the **GAI** data uses data from all domains of AI while the **KDD** data is limited to authors with connections to topics of the KDD, it stands to reason that the documents of the **KDD** dataset are more topically related. Thus, the worse results on the **KDD** data support our hypothesis that AV for bibliometric data is not about distinguishing writing styles, but about identifying relevant topics of authors.

8.5.1 Ablation Study

The novelty of LG4AV lays in the connection of two components, namely a GNN and a fine-tunable language model. Hence, it is crucial to evaluate the individual components with respect to their influence on the verification decisions. In order to do so, we run the experiments in Section 8.5 with two additional LG4AV models.

LG4AV-F This model coincides with the model used in Section 8.5 with the difference, that we freeze all BERT parameters and just train the weight matrix W. This allows us to understand if the process of fine-tuning the BERT parameters is indeed necessary for successful author verification.

LG4AV-0. Here, the parameter k is set to 0. Hence, this model does not use any graph information. It just uses the individual author features and does not include any neighborhood aggregation. We use this model to evaluate to which extent the graph information enhances the verification process.

LG4AV-2 The LG4AV model with $k = 2$ which was also used in Table 8.2. This is our "regular" LG4AV which uses both BERT fine-tuning and neighborhood aggregation.

Procedure

We also run 10 rounds of LG4AV-0 and LG4AV-F and report mean scores. For this runs, we use the same 10 different random seeds for weight initialization and for shuffling the training data which were used for LG4AV-2. We decided for this approach as early experiments indicated that the performances of LG4AV-2 and LG4AV-0 were better for the same random seeds (and therefore same shuffles of training data). This means, that the seeds which lead to the higher/lower values for LG4AV-2 generally also lead to better results for LG4AV-0.

Results and Discussion

The results can be found in Table 8.3. Since the results of LG4AV-2 and LG4AV-0 are very close, we use statistical significance tests. Based on the procedure explained in Section 8.5.1 and the observation that both models have the tendency of having better/worse results for the same random seeds, we decide to use a paired t-test and a Wilcoxon-test over the 10 runs for significance testing. LG4AV-2 always outperforms LG4AV-0 with a significance level of 0.05. On the **KDD** dataset, LG4AV-2 outperforms LG4AV-0 with a significance level of 0.01 on all metrics.

Comparing the performance of LG4AV-F with the results in Table 8.2, the frozen models perform on line with GLAD and RBI, which shows that LG4AV-F is (to some extent) capable of successful author verification. We especially point that out because the freezing of the BERT layers significantly decreases the runtime since only the $1 + k \cdot 768$ parameters of the top layer have to be trained. On a NVIDIA RTX 2060 SUPER, each epoch of LG4AV-F and LG4AV-2 on the **GAI** data last for about 40 minutes and for about 2 hour and 15 minutes, respectively. For the **KDD** data, one epoch needs about 1 hour and about 4 hours and 15 minutes, respectively. With limited resources available it would be a reasonable compromise to, for example, train the whole model for just one epoch and then freeze the layers of the language model.

To sum up, the results indicate that the incorporation of co-author information can lead to additional enhancements. Still, the comparable results for $k = 0$ show that our idea of combining text features with a language model even works without the addition of co-author information. On the other hand, if the BERT layers are frozen, the performance declines considerably. Hence, the fine-tuning of the language model is an integral point for the successful author verification with LG4AV.

8.6 Conclusion and Outlook

In this chapter, we presented LG4AV, a novel architecture for author verification. We worked with a complex network structure that consist of co-author links between authors and links between authors and their publications. By combining a language model with a graph neural network, our model does not depend on any handcrafted features. Instead we incorporate information of co-author edges and raw texts into weights of neural network to study this special network from a condensed perspective. LG4AV surpasses methods that use handcrafted stylometric and n-gram text features when it comes to verification of short and, to some extent, standardized texts. Hence, LG4AV is especially helpful to learn real-valued representations of authorship information in bibliometric datasets, especially when only abstracts and titles are available.

Future work could include applications of LG4AV to different settings, as for example social media posts. Additionally, it is promising to investigate temporal evolution of interests of authors. Here, one could for example study correlations between the temporal distance of the training and test set and the classification performance.

Part V

Conclusion and Outlook

Chapter 9

Conclusion

In this thesis, we contributed to the realm of network analysis by providing different perspectives on the structure of networks and by establishing learning procedures for special classes of networks. The broad goal was to gain new insights by providing novel quantities and structures to analyze networks and to develop approaches for representation learning. Our specific contributions are recapped in the following.

First, we proposed insights from a *local perspective* by studying locally outstanding points and dominance relations between them. For this, we transferred orometric concepts to the realm of metric and network data. We started in Chapter 3 with transferring the notions of prominence and isolation, that have already been defined for networks, to bounded metric data. To do so, we used different graph structures that can be defined on metric datasets. We hence built a bridge between orometric concepts for networks and metric spaces. Our experiments showed, that our novel approach of evaluating the local outstandingness allows to identify relevant data points. More specifically, we identified important locations both in France and in Germany. Our experiments further showed that the relative neighborhood graph is particularly useful to define a graph structure on metric data for the sake of identifying locally outstanding points.

To build further bridges between network analysis and orometry, we transferred in Chapter 4 the notion of line parents to networks. This allowed to infer small hierarchies of important "peaks" from larger graphs. In this way, we provide novel insights into networks and a navigation paradigm that has not been studied before. To discard weak edges as a preprocessing step, we introduced a novel and parameter free method that guarantees to preserve the connectivity of the original graph. For this, we again made use of the relative neighborhood graph.

Our experiments on real-world data showed that we can derive small line parent hierarchies from larger networks. We empirically showed that the derived substructures are representative for the full network in the sense that they lay dense in the original network.

Second, we studied networks from a *global perspective* by investigating intrinsic dimensionality as a measure for network complexity. For this, we built in Chapter 5 on recent work that established intrinsic dimensionality for geometric datasets, a structure that allows to incorporate feature functions into the computation that can go beyond coordinate projections of attribute vectors.

In this thesis we established connections between the above mentioned notion of intrinsic dimensionality, network analysis and learning on graphs by making this notion applicable to real-world network data and by studying connections between intrinsic dimensionality and the success of classification tasks.

By proposing speedup-techniques for an algorithm that is in its original form of quadratic runtime complexity, we scaled notion of intrinsic dimensionality to large-scale networks. Our experiments on real-world graphs showed that our speedup technique allowed us to save over 99% of the potential computations steps, which made it possible to compute the intrinsic dimension of graphs with millions of nodes.

Furthermore, we were able to efficiently approximate the intrinsic dimension of a graph with over 100 millions of nodes. To be more specific, we gave empirical evidence that it is possible to approximate the intrinsic dimension of the well-known **ogbn-papers100M** with an accuracy of over 99,9% in a few hours.

We investigated connections between intrinsic dimensionality and learning in multiple ways. We first studied how the intrinsic dimensionality and the classification performance behave for multiple amounts of neighborhood aggregation. We indeed identified a connection between drops in the intrinsic dimensionality and enhancement in classification accuracy. Here, the outstanding observation was that the first round of neighborhood aggregation is crucial both for lowering the intrinsic dimensionality and the classification performance.

Furthermore, we showed in Chapter 6 that our notions of intrinsic dimensionality can be adapted to feature selection. For this, we ranked and selected features by their ability to discriminate data points. We grounded our novel feature selection procedure on Euclidean and network data. Selecting features that tame the curse of dimensionality fundamentally outperforms random selection and is competitive with established and reasonable baselines. Thus, our experiments show that features that are resilient to the curse of dimensionality are indeed meaningful for learning.

Third, we proposed novel architectures for representation learning to study social networks from a *condensed perspective*.

We introduced two novel procedures for learning on graphs. First, with establishing FCA2VEC, we were able to transfer common word embedding approaches to formal concept analysis. By doing so, we were able to provide a novel embedding procedure for bipartite graphs that is based on the structure of a concept lattice. Our experiments show, that this approach leads to embeddings that are meaningful with respect to implicational knowledge of the network.

Second, we proposed a novel neural network architecture that combines pre-trained language models with a graph neural network. The resulting architecture, called LG4AV, is especially tailored for the complex real-world structure of academic authors that are connected to publications by authorships and at the same time to other authors by co-authorships. We successfully applied this architecture to the task of author verification. Our results indicate that the language model is the key factor for successful author verification and that the incorporation of co-author information via neighborhood aggregation can lead to further enhancements in the classification process.

To sum up, we proposed novel measures and structures to get new insights from networks from both local and global perspectives. Furthermore, we provided learning architectures that incorporate network information into real-valued model weights and therefore allow to study networks from a condensed perspective.

Chapter 10

Outlook

The different parts of this thesis open multiple doors for future research. We examine some of them in the following and discuss arising obstacles.

Our adaption of orometric concepts gave us new insights into networks. To further understand the phenomena which underlie the discovered network structures, it will be fruitful to incorporate temporal aspects. In the case of line parent hierarchies, the question arises how to define a line parent hierarchy for a temporal network. The first question here would be how to model temporal networks. If one decides to model them by sequences of graphs, a first approach could be to model temporal line parents as a sequence of "normal" line parents. In this case, one could investigate the changes of the line parent tree from time-stamp to time-stamp. Future research then has to tackle the question on how to identify and quantify changes and differences between different line parents. This could lead to deeper thoughts on the meaning of the line parent hierarchy for the social phenomenon from which the network concept was abstracted. Here, one would study which changes in the line parent hierarchy correspond to which circumstances on the "real-world" level.

This thesis has studied intrinsic dimensionality as a complexity measure for network data and we have drawn first connections to machine learning. However, there are multiple possible approaches yet to investigate for strengthening this connection. A first possibility would be to explicitly evaluate or manipulate the intrinsic dimensionality at different layers while training a neural network. While this route has been followed in computer vision [5, 98], it is not established in the area of graph learning. Here, it would be a crucial question on which layers of a graph neural network the intrinsic dimensionality should be measured and how it should be incorporated into the training process.

Finally, the introduced learning methods in this thesis, namely FCA2VEC and LG4AV, lead to further possibilities for graph learning. Our aim to connect formal concept analysis with representation learning can be continued by investigating different representation approaches. The current methods generates vector representations which can be used as input for common machine learning tools. This is a consequence of it being an adaption of word2vec. However, another paradigm of incorporating structure into learning, especially in the realm of social networks, has been widely discussed in this work. Namely, the paradigm of neighborhood aggregation via graph neural networks. This leads to the question, how this paradigm can be adapted to the area of formal concept analysis. This could lead to a novel class of architectures for bipartite graphs. Here, the more distant vision would be a class of concept neural networks which use an innovative formal concept aggregation scheme to incorporate the lattice structure into learning.

Future research on LG4AV could include temporal aspects. Here, an interesting question will be whether larger time spans between training and testing makes it probable that authors change their interest which would make it harder to predict potential future papers for them. It will also be crucial to study whether a bigger time span makes the incorporation of co-author information more or less important.

To sum up, we identified multiple potentials of future research which are mainly connected to two directions. The first research direction would be to incorporate temporal aspects into the notions and concepts introduced. The second direction would deal with adapting and using the proposed methods to develop novel learning procedures or to modify already present approaches.

List of Figures

List of Tables

References

[1] Agarwal, P. K. and Matousek, J. (1992). Relative neighborhood graphs in three dimensions. In Frederickson, G. N., editor, *Proceedings of the Third Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 27-29 January 1992, Orlando, Florida, USA*, pages 58–65. ACM/SIAM.

[2] Akbani, R., Kwek, S., and Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. In Boulicaut, J., Esposito, F., Giannotti, F., and Pedreschi, D., editors, *Machine Learning: ECML 2004, 15th European Conference on Machine Learning, Pisa, Italy, September 20-24, 2004, Proceedings*, volume 3201 of *Lecture Notes in Computer Science*, pages 39–50. Springer.

[3] Alelyani, S., Tang, J., and Liu, H. (2013). Feature selection for clustering: A review. In Aggarwal, C. C. and Reddy, C. K., editors, *Data Clustering: Algorithms and Applications*, pages 29–60. CRC Press.

[4] Ammar, W., Groeneveld, D., Bhagavatula, C., Beltagy, I., Crawford, M., Downey, D., Dunkelberger, J., Elgohary, A., Feldman, S., Ha, V., Kinney, R., Kohlmeier, S., Lo, K., Murray, T., Ooi, H., Peters, M. E., Power, J., Skjonsberg, S., Wang, L. L., Wilhelm, C., Yuan, Z., van Zuylen, M., and Etzioni, O. (2018). Construction of the literature graph in semantic scholar. In Bangalore, S., Chu-Carroll, J., and Li, Y., editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 3 (Industry Papers)*, pages 84–91. Association for Computational Linguistics.

[5] Ansuini, A., Laio, A., Macke, J. H., and Zoccolan, D. (2019). Intrinsic dimension of data representations in deep neural networks. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 6109–6119.

[6] Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. In Bansal, N., Pruhs, K., and Stein, C., editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 1027–1035. SIAM.

[7] Bac, J. and Zinovyev, A. Y. (2020). Local intrinsic dimensionality estimators based on concentration of measure. In *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*, pages 1–8. IEEE.

[8] Bagnall, D. (2015). Author identification using multi-headed recurrent neural networks. In Cappellato, L., Ferro, N., Jones, G. J. F., and SanJuan, E., editors, *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015*, volume 1391 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[9] Barlas, G. and Stamatatos, E. (2020). Cross-domain authorship attribution using pre-trained language models. In Maglogiannis, I., Iliadis, L., and Pimenidis, E., editors, *Artificial Intelligence Applications and Innovations - 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, June 5-7, 2020, Proceedings, Part I*, volume 583 of *IFIP Advances in Information and Communication Technology*, pages 255–266. Springer.

[10] Belohlávek, R. and Trnecka, M. (2015). From-below approximations in boolean matrix factorization: Geometry and new algorithm. *J. Comput. Syst. Sci.*, 81(8):1678–1697.

[11] Belohlávek, R. and Vychodil, V. (2010). Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J. Comput. Syst. Sci.*, 76(1):3–20.

[12] Bengio, Y., Courville, A. C., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828.

[13] Bentley, J. L. (1975). A survey of techniques for fixed radius near neighbor searching. Technical report, SLAC, SCIDOC, Stanford, CA, USA. SLAC-R-0186, SLAC-0186.

[14] Bevendorff, J., Stein, B., Hagen, M., and Potthast, M. (2019). Generalizing unmasking for short texts. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 654–659. Association for Computational Linguistics.

[15] Bischl, B., Casalicchio, G., Feurer, M., Gijsbers, P., Hutter, F., Lang, M., Mantovani, R. G., van Rijn, J. N., and Vanschoren, J. (2021). Openml benchmarking suites. In Vanschoren, J. and Yeung, S., editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual.*

[16] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer Science+ Business Media.

[17] Boldi, P., Rosa, M., Santini, M., and Vigna, S. (2011). Layered label propagation: a multiresolution coordinate-free ordering for compressing social networks. In Srinivasan, S., Ramamritham, K., Kumar, A., Ravindra, M. P., Bertino, E., and Kumar, R., editors, *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, pages 587–596. ACM.

[18] Boldi, P. and Vigna, S. (2004). The webgraph framework I: compression techniques. In Feldman, S. I., Uretsky, M., Najork, M., and Wills, C. E., editors, *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004*, pages 595–602. ACM.

[19] Bornmann, L. and Mutz, R. (2015). Growth rates of modern science: A bibliometric analysis based on the number of publications and cited references. *J. Assoc. Inf. Sci. Technol.*, 66(11):2215–2222.

[20] Bradley, J. K., Kelley, P. G., and Roth, A. (2008). Author identification from citations. *Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep.*

[21] Brandes, U. (2016). Network positions. *Methodological Innovations*, 9:2059799116630650.

[22] Brandes, U. and Erlebach, T. (2005). *Fundamentals*, pages 7–15. Springer Berlin Heidelberg, Berlin, Heidelberg.

[23] Brandes, U., Robins, G., McCranie, A., and Wasserman, S. (2013). What is network science? *Network science*, 1(1):1–15.

[24] Breaban, M. and Luchian, H. (2011). A unifying criterion for unsupervised clustering and feature selection. *Pattern Recognit.*, 44(4):854–865.

[25] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

[26] Cai, J., Luo, J., Wang, S., and Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79.

[27] Callan, R. (2003). *Artificial intelligence*. Red Globe Press.

[28] Caragea, C., Uban, A. S., and Dinu, L. P. (2019). The myth of double-blind review revisited: ACL vs. EMNLP. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2317–2327. Association for Computational Linguistics.

[29] Caro-Contreras, D. E. and Mendez-Vazquez, A. (2013). Computing the concept lattice using dendritical neural networks. In Ojeda-Aciego, M. and Outrata, J., editors, *Proceedings of the Tenth International Conference on Concept Lattices and Their Applications, La Rochelle, France, October 15-18, 2013*, volume 1062 of *CEUR Workshop Proceedings*, pages 141–152. CEUR-WS.org.

[30] Castro-Castro, D., Arcia, Y. A., Brioso, M. P., and Guillena, R. M. (2015). Authorship verification, average similarity analysis. In Angelova, G., Bontcheva, K., and Mitkov, R., editors, *Recent Advances in Natural Language Processing, RANLP 2015, 7-9 September, 2015, Hissar, Bulgaria*, pages 84–90. RANLP 2015 Organising Committee / ACL.

[31] Chávez, E., Navarro, G., Baeza-Yates, R. A., and Marroquín, J. L. (2001). Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321.

[32] Clauset, A., Moore, C., and Newman, M. E. (2008). Hierarchical structure and the prediction of missing links in networks. *Nature*.

[33] Cloninger, A. and Klock, T. (2021). A deep network construction that adapts to intrinsic dimensionality beyond the domain. *Neural Networks*, 141:404–419.

[34] Codocedo, V., Taramasco, C., and Astudillo, H. (2011). Cheating to achieve formal concept analysis over a large formal context. In Napoli, A. and Vychodil, V., editors, *Proceedings of The Eighth International Conference on Concept Lattices and Their Applications, Nancy, France, October 17-20, 2011*, volume 959 of *CEUR Workshop Proceedings*, pages 349–362. CEUR-WS.org.

[35] Cohan, A., Feldman, S., Beltagy, I., Downey, D., and Weld, D. S. (2020). SPECTER: document-level representation learning using citation-informed transformers. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. R., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2270–2282. Association for Computational Linguistics.

[36] Čomić, L., De Floriani, L., and Papaleo, L. (2005). Morse-smale decompositions for modeling terrain knowledge. In Cohn, A. G. and Mark, D. M., editors, *Spatial Information Theory*, pages 426–444, Berlin, Heidelberg. Springer Berlin Heidelberg.

[37] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20(3):273–297.

[38] Costa, J., Girotra, A., and Hero, A. (2005). Estimating local intrinsic dimension with k-nearest neighbor graphs. In *IEEE/SP 13th Workshop on Statistical Signal Processing, 2005*, pages 417–422.

[39] Deo, N. (1975). Graph theory with applications to engineering and computer science. *Networks*, 5(3):299–300.

[40] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

[41] Diestel, R. (2005). Graph theory 3rd ed. *Graduate texts in mathematics*, 173(33):12.

[42] Dong, Y., Chawla, N. V., and Swami, A. (2017). metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 135–144. ACM.

[43] Dürrschnabel, D., Hanika, T., and Stubbemann, M. (2022). FCA2VEC: Embedding techniques for formal concept analysis. In Missaoui, R., Kwuida, L., and Abdessalem, T., editors, *Complex Data Analytics with Formal Concept Analysis*, pages 47–74. Springer International Publishing.

[44] Dürrschnabel, D. and Stumme, G. (2023a). Greedy discovery of ordinal factors. *CoRR*, abs/2302.11554.

[45] Dürrschnabel, D. and Stumme, G. (2023b). Maximal ordinal two-factorizations. *CoRR*, abs/2304.03338.

[46] Dutta, D., Dutta, P., and Sil, J. (2014). Simultaneous feature selection and clustering with mixed features by multi objective genetic algorithm. *Int. J. Hybrid Intell. Syst.*, 11(1):41–54.

[47] Dy, J. G. and Brodley, C. E. (2004). Feature selection for unsupervised learning. *J. Mach. Learn. Res.*, 5:845–889.

[48] Eades, P. (1984). A heuristic for graph drawing. *Congressus numerantium.*

[49] Ester, M. and Sander, J. (2000). *Knowledge Discovery in Databases - Techniken und Anwendungen.* Springer.

[50] Facco, E., d'Errico, M., Rodriguez, A., and Laio, A. (2017). Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):1–8.

[51] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognit. Lett.*, 27(8):861–874.

[52] Ferreira, A. J. and Figueiredo, M. A. T. (2012). An unsupervised approach to feature discretization and selection. *Pattern Recognit.*, 45(9):3048–3060.

[53] Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with pytorch geometric. *CoRR*, abs/1903.02428.

[54] Fey, M., Lenssen, J. E., Weichert, F., and Leskovec, J. (2021). Gnnautoscale: Scalable and expressive graph neural networks via historical embeddings. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 3294–3304. PMLR.

[55] Fruchterman, T. M. J. and Reingold, E. M. (1991). Graph drawing by force-directed placement. *Softw. Pract. Exp.*

[56] Fry, S. (1987). Defining and sizing-up mountains. *Summit, Jan.–Feb*, pages 16–21.

[57] Ganter, B. and Glodeanu, C. V. (2012). Ordinal factor analysis. In Domenach, F., Ignatov, D. I., and Poelmans, J., editors, *Formal Concept Analysis - 10th International Conference, ICFCA 2012, Leuven, Belgium, May 7-10, 2012. Proceedings*, volume 7278 of *Lecture Notes in Computer Science*, pages 128–139. Springer.

[58] Ganter, B. and Wille, R. (1999). *Formal Concept Analysis: Mathematical Foundations.* Springer-Verlag, Berlin.

[59] Golay, J. and Kanevski, M. F. (2017). Unsupervised feature selection based on the morisita estimator of intrinsic dimension. *Knowl. Based Syst.*, 135:125–134.

[60] Gómez, W., Leija, L., and Díaz-Pérez, A. (2010). Mutual information and intrinsic dimensionality for feature selection. In *Proceedings of the 7th International Conference on Electrical Engineering, Computing Science and Automatic Control, CCE 2010 (Formerly known as ICEEE), September 8-10, 2010, Tuxtla Gutierrez, Mexico*, pages 339–344. IEEE.

[61] Gomtsyan, M., Mokrov, N., Panov, M., and Yanovich, Y. (2019). Geometry-aware maximum likelihood estimation of intrinsic dimension. In Lee, W. S. and Suzuki, T., editors, *Proceedings of The 11th Asian Conference on Machine Learning, ACML 2019, 17-19 November 2019, Nagoya, Japan*, volume 101 of *Proceedings of Machine Learning Research*, pages 1126–1141. PMLR.

[62] Granata, D. and Carnevale, V. (2016). Accurate estimation of the intrinsic dimension using graph distances: Unraveling the geometric complexity of datasets. *Scientific reports*, 6(1):1–12.

[63] Grassberger, P. and Procaccia, I. (2004). Measuring the strangeness of strange attractors. In *The theory of chaotic attractors*, pages 170–189. Springer.

[64] Gromov, M. and Milman, V. D. (1983). A topological application of the isoperimetric inequality. *American Journal of Mathematics*, 105(4):843–854.

[65] Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In Krishnapuram, B., Shah, M., Smola, A. J., Aggarwal, C. C., Shen, D., and Rastogi, R., editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864. ACM.

[66] Gupte, M., Shankar, P., Li, J., Muthukrishnan, S., and Iftode, L. (2011). Finding hierarchy in directed online social networks. In Srinivasan, S., Ramamritham, K., Kumar, A., Ravindra, M. P., Bertino, E., and Kumar, R., editors, *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, pages 557–566. ACM.

[67] Halvani, O., Winter, C., and Graner, L. (2019). Assessing the applicability of authorship verification methods. In *Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES 2019, Canterbury, UK, August 26-29, 2019*, pages 38:1–38:10. ACM.

[68] Hamilton, W. L., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034.

[69] Hanika, T., Marx, M., and Stumme, G. (2019). Discovering implicational knowledge in wikidata. In Cristea, D., Ber, F. L., and Sertkaya, B., editors, *Formal Concept Analysis - 15th International Conference, ICFCA 2019, Frankfurt, Germany, June 25-28, 2019, Proceedings*, volume 11511 of *Lecture Notes in Computer Science*, pages 315–323. Springer.

[70] Hanika, T., Schneider, F. M., and Stumme, G. (2022). Intrinsic dimension of geometric data sets. *Tohoku Mathematical Journal*, 74(1):23 – 52.

[71] Helman, A. (2005). *The Finest Peaks-Prominence and Other Mountain Measures*. Trafford, Victoria, British Columbia.

[72] Hill, S. and Provost, F. J. (2003). The myth of the double-blind review?: author identification using only citations. *SIGKDD Explor.*, 5(2):179–184.

[73] Hirsch, J. E. (2005). An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences*, 102(46):16569–16572.

[74] Ho, V. T., Stepanova, D., Gad-Elrab, M. H., Kharlamov, E., and Weikum, G. (2018). Rule learning from knowledge graphs guided by embedding models. In Vrandecic, D., Bontcheva, K., Suárez-Figueroa, M. C., Presutti, V., Celino, I., Sabou, M., Kaffee, L., and Simperl, E., editors, *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, volume 11136 of *Lecture Notes in Computer Science*, pages 72–90. Springer.

[75] Hu, W., Fey, M., Ren, H., Nakata, M., Dong, Y., and Leskovec, J. (2021). OGB-LSC: A large-scale challenge for machine learning on graphs. In Vanschoren, J. and Yeung, S., editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual.*

[76] Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

[77] Hürlimann, M., Weck, B., van den Berg, E., Suster, S., and Nissim, M. (2015). GLAD: groningen lightweight authorship detection. In Cappellato, L., Ferro, N., Jones, G. J. F., and SanJuan, E., editors, *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015*, volume 1391 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[78] Jankowska, M., Milios, E. E., and Keselj, V. (2014). Author verification using common n-gram profiles of text documents. In Hajic, J. and Tsujii, J., editors, *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 387–397. ACL.

[79] Jaromczyk, J. W. and Kowaluk, M. (1987). A note on relative neighborhood graphs. In Soule, D., editor, *Proceedings of the Third Annual Symposium on Computational Geometry, Waterloo, Ontario, Canada, June 8-10, 1987*, pages 233–241. ACM.

[80] Jaromczyk, J. W. and Toussaint, G. T. (1992). Relative neighborhood graphs and their relatives. *Proc. IEEE*, 80(9):1502–1517.

[81] Jr., C. T., Traina, A. J. M., Wu, L., and Faloutsos, C. (2010). Fast feature selection using fractal dimension. *J. Inf. Data Manag.*, 1(1):3–16.

[82] Karatzoglou, A. (2020). Applying topographic features for identifying speed patterns using the example of critical driving. In Berres, A. and Kurte, K. R., editors, *IWCTS@SIGSPATIAL 2020: Proceedings of the 13th ACM SIGSPATIAL International Workshop on Computational Transportation Science, Seattle, Washington, 3 November, 2020*, pages 6:1–6:4. ACM.

[83] Kešelj, V., Peng, F., Cercone, N., and Thomas, C. (2003). N-gram-based author profiles for authorship attribution. In *Proceedings of the conference pacific association for computational linguistics*, volume 3, pages 255–264.

[84] Kipf, T. N. (2020). *Deep learning with graph-structured representations*. PhD thesis, Universiteit van Amsterdam.

[85] Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[86] Kirmse, A. and de Ferranti, J. (2017). Calculating the prominence and isolation of every mountain in the world. *Progress in Physical Geography: Earth and Environment*, 41(6):788–802.

[87] Koopmann, T., Stubbemann, M., Kapa, M., Paris, M., Buenstorf, G., Hanika, T., Hotho, A., Jäschke, R., and Stumme, G. (2021). Proximity dimensions and the emergence of collaboration: a hyptrails study on german AI research. *Scientometrics*, 126(12):9847–9868.

[88] Koppel, M. and Schler, J. (2004). Authorship verification as a one-class classification problem. In Brodley, C. E., editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM.

[89] Koppel, M., Schler, J., and Bonchek-Dokow, E. (2007). Measuring differentiability: Unmasking pseudonymous authors. *J. Mach. Learn. Res.*, 8:1261–1276.

[90] Koren, Y. (2005). Drawing graphs by eigenvectors: theory and practice. *Computers & Mathematics with Applications*, 49(11):1867–1888.

[91] Krishnamurthy, V., Sun, J., Faloutsos, M., and Tauro, S. L. (2003). Sampling internet topologies: How small can we go? In Arabnia, H. R. and Mun, Y., editors, *Proceedings of the International Conference on Internet Computing, IC '03, Las Vegas, Nevada, USA, June 23-26, 2003, Volume 2*, pages 577–580. CSREA Press.

[92] Kuznetsov, S. O., Makhazhanov, N., and Ushakov, M. (2017). On neural network architecture based on concept lattices. In Kryszkiewicz, M., Appice, A., Slezak, D., Rybinski, H., Skowron, A., and Ras, Z. W., editors, *Foundations of Intelligent Systems - 23rd International Symposium, ISMIS 2017, Warsaw, Poland, June 26-29, 2017, Proceedings*, volume 10352 of *Lecture Notes in Computer Science*, pages 653–663. Springer.

[93] Kwak, H., Lee, C., Park, H., and Moon, S. B. (2010). What is twitter, a social network or a news media? In Rappa, M., Jones, P., Freire, J., and Chakrabarti, S., editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 591–600. ACM.

[94] Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1188–1196. JMLR.org.

[95] Leskovec, J. and Faloutsos, C. (2006). Sampling from large graphs. In Eliassi-Rad, T., Ungar, L. H., Craven, M., and Gunopulos, D., editors, *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 631–636. ACM.

[96] Leskovec, J. and Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data.

[97] Levina, E. and Bickel, P. J. (2004). Maximum likelihood estimation of intrinsic dimension. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pages 777–784.

[98] Li, C., Farkhoor, H., Liu, R., and Yosinski, J. (2018). Measuring the intrinsic dimension of objective landscapes. In *ICLR (Poster)*. OpenReview.net.

[99] Li, F., Zou, Z., Li, J., and Li, Y. (2019a). Graph compression with stars. In Yang, Q., Zhou, Z., Gong, Z., Zhang, M., and Huang, S., editors, *Advances in Knowledge Discovery and Data Mining - 23rd Pacific-Asia Conference, PAKDD 2019, Macau, China, April 14-17, 2019, Proceedings, Part II*, volume 11440 of *Lecture Notes in Computer Science*, pages 449–461. Springer.

[100] Li, R., Yu, J. X., Qin, L., Mao, R., and Jin, T. (2015). On random walk based graph sampling. In Gehrke, J., Lehner, W., Shim, K., Cha, S. K., and Lohman, G. M., editors, *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 927–938. IEEE Computer Society.

[101] Li, Y., Wu, Z., Lin, S., Xie, H., Lv, M., Xu, Y., and Lui, J. C. S. (2019b). Walking with perception: Efficient random walk sampling via common neighbor awareness. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*, pages 962–973. IEEE.

[102] Lu, C., Yu, J. X., Li, R., and Wei, H. (2016). Exploring hierarchies in online social networks. *IEEE Trans. Knowl. Data Eng.*, 28(8):2086–2100.

[103] Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S. N. R., Schoenebeck, G., Song, D., Houle, M. E., and Bailey, J. (2018a). Characterizing adversarial subspaces using local intrinsic dimensionality. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[104] Ma, X., Wang, Y., Houle, M. E., Zhou, S., Erfani, S. M., Xia, S., Wijewickrema, S. N. R., and Bailey, J. (2018b). Dimensionality-driven learning with noisy labels. In Dy, J. G. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3361–3370. PMLR.

[105] MacKay, D. J. and Ghahramani, Z. (2005). Comments on maximum likelihood estimation of intrinsic dimension by e. levina and p. bickel (2005). *The Inference Group Website, Cavendish Laboratory, Cambridge University.*

[106] Maiya, A. S. and Berger-Wolf, T. Y. (2009). Inferring the maximum likelihood hierarchy in social networks. In *Proceedings of the 12th IEEE International Conference on Computational Science and Engineering, CSE 2009, Vancouver, BC, Canada, August 29-31, 2009*, pages 245–250. IEEE Computer Society.

[107] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y., editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

[108] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

[109] Milman, V. (1988). The heritage of p. lévy in geometrical functional analysis. *Astérisque*, 157(158):273–301.

[110] Milman, V. (2010). *Topics in Asymptotic Geometric Analysis*, pages 792–815. Birkhäuser Basel, Basel.

[111] Mitchell, T. M. (1997). *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill.

[112] Mitra, P., Murthy, C. A., and Pal, S. K. (2002). Unsupervised feature selection using feature similarity. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):301–312.

[113] Mnih, A. and Hinton, G. E. (2008). A scalable hierarchical distributed language model. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1081–1088. Curran Associates, Inc.

[114] Mo, D. and Huang, S. H. (2012). Fractal-based intrinsic dimension estimation and its application in dimensionality reduction. *IEEE Trans. Knowl. Data Eng.*, 24(1):59–71.

[115] Mosbach, M., Andriushchenko, M., and Klakow, D. (2021). On the stability of fine-tuning BERT: misconceptions, explanations, and strong baselines. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[116] Nelson, G. D. and McKeon, R. (2019). Peaks of people: Using topographic prominence as a method for determining the ranked significance of population centers. *The Professional Geographer*, 71(2):342–354.

[117] Newman, M. (2018). *Networks*. Oxford university press.

[118] Pavlík, J. (2015). Topographic spaces over ordered monoids. *Math. Appl.*

[119] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830.

[120] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: online learning of social representations. In Macskassy, S. A., Perlich, C., Leskovec, J., Wang, W., and Ghani, R., editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 701–710. ACM.

[121] Pestov, V. (2000). On the geometry of similarity search: Dimensionality curse and concentration of measure. *Inf. Process. Lett.*, 73(1-2):47–51.

[122] Pestov, V. (2007). Intrinsic dimension of a dataset: what properties does one expect? In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2007, Celebrating 20 years of neural networks, Orlando, Florida, USA, August 12-17, 2007*, pages 2959–2964. IEEE.

[123] Pestov, V. (2008). An axiomatic approach to intrinsic dimension of a dataset. *Neural Networks*, 21(2-3):204–213.

[124] Pestov, V. (2010). Intrinsic dimensionality. *ACM SIGSPATIAL Special*, 2(2):8–11.

[125] Pope, P., Zhu, C., Abdelkader, A., Goldblum, M., and Goldstein, T. (2021). The intrinsic dimension of images and its impact on learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[126] Potha, N. and Stamatatos, E. (2020). Improved algorithms for extrinsic author verification. *Knowl. Inf. Syst.*, 62(5):1903–1921.

[127] Rafiei, D. and Curial, S. (2005). Effectively visualizing large networks through sampling. In *16th IEEE Visualization Conference, IEEE Vis 2005, Minneapolis, MN, USA, October 23-28, 2005, Proceedings*, pages 375–382. IEEE Computer Society.

[128] Rong, X. (2014). word2vec parameter learning explained. *CoRR*, abs/1411.2738.

[129] Rossi, E., Frasca, F., Chamberlain, B., Eynard, D., Bronstein, M. M., and Monti, F. (2020). SIGN: scalable inception graph neural networks. *CoRR*, abs/2004.11198.

[130] Royer, L., Reimann, M., Andreopoulos, B., and Schroeder, M. (2008). Unraveling protein networks with power graph analysis. *PLoS Comput. Biol.*, 4(7).

[131] Russell, S. and Norvig, P. (2003). *Artificial intelligence - a modern approach, 2nd Edition*. Prentice Hall series in artificial intelligence. Prentice Hall.

[132] Sarwar, R., Urailertprasert, N., Vannaboot, N., Yu, C., Rakthanmanon, T., Chuangsuwanich, E., and Nutanong, S. (2020). CAG: stylometric authorship attribution of multi-author documents using a co-authorship graph. *IEEE Access*, 8:18374–18393.

[133] Schmidt, A. and Stumme, G. (2018). Prominence and dominance in networks. In Faron-Zucker, C., Ghidini, C., Napoli, A., and Toussaint, Y., editors, *Knowledge Engineering and Knowledge Management - 21st International Conference, EKAW 2018, Nancy, France, November 12-16, 2018, Proceedings*, volume 11313 of *Lecture Notes in Computer Science*, pages 370–385. Springer.

[134] Scott, D. (1964). Measurement structures and linear inequalities. *Journal of Mathematical Psychology*, 1(2):233 – 247.

[135] Scott, J. (2012). *What is social network analysis?* Bloomsbury Academic.

[136] Seidman, S. (2013). Authorship verification using the impostors method. In Forner, P., Navigli, R., Tufis, D., and Ferro, N., editors, *Working Notes for CLEF 2013 Conference , Valencia, Spain, September 23-26, 2013*, volume 1179 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[137] Seidman, S. B. (1983). Network structure and minimum degree. *Social Networks*, 5(3):269–287.

[138] Solorio-Fernández, S., Carrasco-Ochoa, J. A., and Martínez-Trinidad, J. F. (2020). A review of unsupervised feature selection methods. *Artif. Intell. Rev.*, 53(2):907–948.

[139] Stubbemann, M., Hanika, T., and Schneider, F. M. (2023a). Intrinsic dimension for large-scale geometric learning. *Transactions on Machine Learning Research.*

[140] Stubbemann, M., Hanika, T., and Stumme, G. (2020). Orometric methods in bounded metric data. In Berthold, M. R., Feelders, A., and Krempl, G., editors, *Advances in Intelligent Data Analysis XVIII - 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27-29, 2020, Proceedings*, volume 12080 of *Lecture Notes in Computer Science*, pages 496–508. Springer.

[141] Stubbemann, M., Hille, T., and Hanika, T. (2023b). Selecting features by their resilience to the curse of dimensionality. *CoRR*, abs/2304.02455.

[142] Stubbemann, M. and Stumme, G. (2022). LG4AV: Combining language models and graph neural networks for author verification. In Bouadi, T., Fromont, É., and Hüllermeier, E., editors, *Advances in Intelligent Data Analysis XX - 20th International Symposium on Intelligent Data Analysis, IDA 2022, Rennes, France, April 20-22, 2022, Proceedings*, volume 13205 of *Lecture Notes in Computer Science*, pages 315–326. Springer.

[143] Stubbemann, M. and Stumme, G. (2023). The mont blanc of twitter: Identifying hierarchies of outstanding peaks in social networks. In *Machine Learning and Knowledge Discovery in Databases: Research Track - European Conference, ECML PKDD 2023, Turin, Italy, September 18-22, 2023, Proceedings, Part III*, volume 14171 of *Lecture Notes in Computer Science*, pages 177–192. Springer.

[144] Sun, C. and Wu, G. (2021). Scalable and adaptive graph neural networks with self-label-enhanced training. *CoRR*, abs/2104.09376.

[145] Toivonen, H., Zhou, F., Hartikainen, A., and Hinkka, A. (2011). Compression of weighted graphs. In Apté, C., Ghosh, J., and Smyth, P., editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 965–973. ACM.

[146] Torres, R. N., Fraternali, P., Milani, F., and Frajberg, D. (2018). A deep learning model for identifying mountain summits in digital elevation model data. In *First IEEE International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2018, Laguna Hills, CA, USA, September 26-28, 2018*, pages 212–217. IEEE Computer Society.

[147] Toussaint, G. T. (1980). The relative neighbourhood graph of a finite planar set. *Pattern Recognit.*, 12(4):261–268.

[148] Tyo, J., Dhingra, B., and Lipton, Z. C. (2021). Siamese bert for authorship verification. In Faggioli, G., Ferro, N., Joly, A., Maistro, M., and Piroi, F., editors, *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021*, volume 2936 of *CEUR Workshop Proceedings*, pages 2169–2177. CEUR-WS.org.

[149] van Halteren, H. (2004). Linguistic profiling for authorship recognition and verification. In Scott, D., Daelemans, W., and Walker, M. A., editors, *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain*, pages 199–206. ACL.

[150] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

[151] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[152] Vrandecic, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

[153] Wasserman, S. and Faust, K. (2007). *Social network analysis - methods and applications*, volume 8 of *Structural analysis in the social sciences*. Cambridge University Press.

[154] Wille, U. (1996). Representation of finite ordinal data in real vector spaces. In Bock, H.-H. and Polasek, W., editors, *Data Analysis and Information Systems*, pages 228–240, Berlin, Heidelberg. Springer Berlin Heidelberg.

[155] Wille, U. (1997). The role of synthetic geometry in representational measurement theory. *journal of mathematical psychology*, 41(1):71–78.

[156] Wojtowytsch, S. and E, W. (2020). Can shallow neural networks beat the curse of dimensionality? A mean field training perspective. *IEEE Trans. Artif. Intell.*, 1(2):121–129.

[157] Yang, Z., Cohen, W. W., and Salakhutdinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 40–48. JMLR.org.

[158] Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

[159] Zeng, H., Zhang, M., Xia, Y., Srivastava, A., Malevich, A., Kannan, R., Prasanna, V. K., Jin, L., and Chen, R. (2021). Decoupling the depth and scope of graph neural networks. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 19665–19679.

[160] Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. K. (2020). Graphsaint: Graph sampling based inductive learning method. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[161] Zhang, W., Yin, Z., Sheng, Z., Li, Y., Ouyang, W., Li, X., Tao, Y., Yang, Z., and Cui, B. (2022). Graph attention multi-layer perceptron. In Zhang, A. and Rangwala, H., editors, *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 4560–4570. ACM.

[162] Zhao, Z. and Liu, H. (2007). Spectral feature selection for supervised and unsupervised learning. In Ghahramani, Z., editor, *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pages 1151–1157. ACM.

[163] Zhu, J., Tian, Z., and Kübler, S. (2019). Um-iu@ling at semeval-2019 task 6: Identifying offensive tweets using BERT and svms. In May, J., Shutova, E., Herbelot, A., Zhu, X., Apidianaki, M., and Mohammad, S. M., editors, *Proceedings of the 13th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2019, Minneapolis, MN, USA, June 6-7, 2019*, pages 788–795. Association for Computational Linguistics.

Template

This thesis has been built with the help of the CUED PhD thesis template. This LaTeX template is at the time of writing this thesis available at https://github.com/kks32/phd-thesis-template. The template (not this thesis itself!) is published under the following MIT license.