Learning to Aggregate on Structured Data

Clemens Damke

November 18, 2019

PADERBORN UNIVERSITY
The University for the Information Society
Department of Electrical Engineering,
Computer Science and Mathematics
Warburger Straße 100
33098 Paderborn

INTELLIGENT
SYSTEMS
Intelligent Systems Group (ISG)

Master Thesis

Learning to Aggregate on Structured Data

Clemens Damke

1. Reviewer Prof. Dr. Eyke Hüllermeier
Intelligent Systems and Machine Learning Group (ISG)
Paderborn University

2. Reviewer Prof. Dr. Axel-Cyrille Ngonga Ngomo
Data Science Group (DICE)
Paderborn University

November 18, 2019

Clemens Damke
Learning to Aggregate on Structured Data
Master Thesis, November 18, 2019
Reviewers: Prof. Dr. Eyke Hüllermeier and Prof. Dr. Axel-Cyrille Ngonga Ngomo
Supervisor: Vitalik Melnikov

Paderborn University
Intelligent Systems and Machine Learning Group (ISG)
Heinz Nixdorf Institute
Department of Electrical Engineering, Computer Science and Mathematics
Warburger Straße 100
33098 Paderborn

Abstract

Contents

Introduction

1

1.1 Motivation

The field of *machine learning* (ML) on graph-structured data has applications in many domains due to the general expressive power of graphs. Three common types of graph ML problems are

1. **Link prediction:** A graph with an incomplete edge set is given and the missing edges have to be predicted. The generation of friendship suggestions in a social network is a typical example for this.

2. **Vertex classification & regression:** Here a class or a score has to be predicted for each vertex of a graph. In social graphs this corresponds to the prediction of properties of individuals. Another example is the prediction of the amount of traffic at the intersections of a street network.

3. **Graph classification & regression:** In this final problem type a single global class or continuous value has to be predicted for an input graph. The canonical example for this is the prediction of properties of molecule graphs, e.g. the toxicity or solubility of a chemical.

In this thesis we will focus on the last problem type, GC/GR. A ML method for this problem has to accept graphs of varying size and should be permutation invariant wrt. the vertices. Those requirements are not met by the commonly used learners that only accept fixed-size feature vectors as their input, e.g. logistic regression models, support vector machines or multilayer perceptrons.

A GC/GR method has to account for two central aspects of the problem: 1. Local structural analysis and 2. global aggregation. The first aspect is about the extraction of relevant features of substructures of the input graph. The latter is about the way in which the local features are combined into a final class or regression value. The existing GC/GR methods are mostly motivated by local structural graph analysis. The aspect of global aggregation on the other hand is less emphasized by those methods.

There is however a separate branch of research that specifically looks at the problem of learning aggregation functions, called *learning to aggregate* (LTA). Current LTA approaches explicitly learn an aggregation functions for sets which can be interpreted as graphs without edges. The motivation for this thesis is to generalize LTA from sets to arbitrary graphs. The overall goal is to combine the aggregation learning perspective with existing GC/GR methods.

1.2 Goals

To extend LTA to graphs, three goals have to be achieved:

1. **Formalization of LTA:** Before LTA can be extended, its essential characteristics have to be defined. Those characteristics should provide the terminology to formally capture the differences and similarities between LTA and existing GC/GR methods.

2. **Give an LTA interpretation of GC/GR methods:** Using the LTA formalization, representative GC/GR approaches should be restated as LTA instances. Currently there is no comprehensive formulation of the relation between both fields of research; this is addressed by the the second goal.

3. **Define an LTA method for graphs:** Using the LTA perspective on GC/GR, hidden assumptions of the existing approaches should become clear and in which way they share the assumptions of LTA. The last goal is to use those insights to formulate an LTA-GC/GR method that combines ideas from the existing approaches with the LTA assumptions.

1.3 Structure

Chapter 2: Related Work

Chapter 3: Learning to Aggregate on Graphs

Chapter 4: Evaluation

Chapter 5: Conclusion

Related Work 2

Before combining LTA and GC/GR as described in section 1.2, we first give an overview of the state-of-the-art in both fields of research. This is done in three steps:

1. We begin with an overview of the existing LTA methods for set inputs.
2. Before the existing GC/GR methods are described, we look at the graph theoretical fundamentals on which those methods are based.
3. Using the previously introduced fundamentals, the current approaches to tackle the GC/GR problem are then described.

2.1 Learning to Aggregate

The class of LTA problems was first described by Melnikov and Hüllermeier [MH16]. There an input instance is understood as a composition $\boldsymbol{c}$ of so-called constituents $c_i \in \boldsymbol{c}$, i.e. as a variable-size multiset with $n = |\boldsymbol{c}|$. The assumption in LTA problems is that for all constituents c_i a local score $y_i \in \mathcal{Y}$ is either given or computable. The set of those local scores should be indicative of the overall score $y \in \mathcal{Y}$ of the composition $\boldsymbol{c}$. LTA problems typically require two subproblems to be solved:

1. **Aggregation:** A variadic aggregation function $\mathcal{A} : \mathcal{Y}^* \to \mathcal{Y}$ that estimates composite scores has to be learned, i.e. $y_i \approx \hat{y} = \mathcal{A}(y_1, \dots, y_n)$. Typically the aggregation function $\mathcal{A}$ should be associative and commutative to fit with the multiset-structure of compositions.
2. **Disaggregation:** In case the constituent scores y_i are not given, they have to be derived from a constituent representation, e.g. a vector $x_i \in \mathcal{X}$. To learn this derivation function $f : \mathcal{X} \to \mathcal{Y}$, only the constituent vectors $\{x_i\}_{i=1}^n$ and the composite score y is given. Thus the constituent scores y_i need to be *disaggregated* from y in order to learn f.

Overall LTA can be understood as the joint problem of learning the aggregation function $\mathcal{A}$ and the local score derivation function f. Two main approaches to represent the aggregation function in LTA problems have been explored.

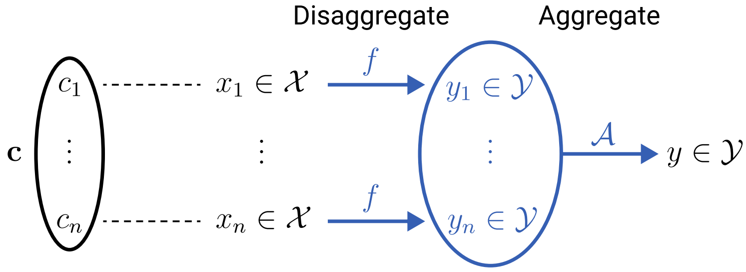

Figure 2.1. Overview of the structure of LTA for multiset compositions.

2.1.1 Uninorm-Aggregation

The first approach uses *uninorms* [MH16] to do so. There the basic idea is to express composite scores as fuzzy truth assignments $y \in [0,1]$. Such a composite assignment y is modeled as the result of a parameterized logical expression of constituent assignments $y_i \in [0,1]$. As the logical expression that thus effectively aggregates the constituents, a uninorm U_λ is used. Depending on the parameter $\lambda \in [0,1]$, U_λ combines a t-norm T and a t-conorm S which are continuous generalizations of logical conjunction and disjunction respectively. One popular choice of norms are the so-called Łukasiewicz norms:

$$\text{t-norm } T(a,b) := \max\{0, a+b-1\}, \quad \text{t-conorm } S(a,b) := \min\{a+b, 1\},$$

$$\text{uninorm } U_\lambda(a,b) := \begin{cases} \lambda T\left(\frac{a}{\lambda}, \frac{b}{\lambda}\right) & \text{if } a,b \in [0,\lambda] \\ \lambda + (1-\lambda) S\left(\frac{a-\lambda}{1-\lambda}, \frac{b-\lambda}{1-\lambda}\right) & \text{if } a,b \in [\lambda,1] \\ \lambda \min\{a,b\} & \text{else} \end{cases} \tag{2.1}$$

At the extreme points $(0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$, T and S coincide with the Boolean operators $\wedge$ and $\vee$; the values at all other points are interpolated as shown in fig. 2.2. The uninorm U_λ uses the conjunctive t-norm T for values below the threshold λ and the disjunctive t-conorm S for values above the threshold. U_λ therefore smoothly interpolates between a conjunctive and disjunctive operator with the extreme points $U_1 = T$ and $U_0 = S$.

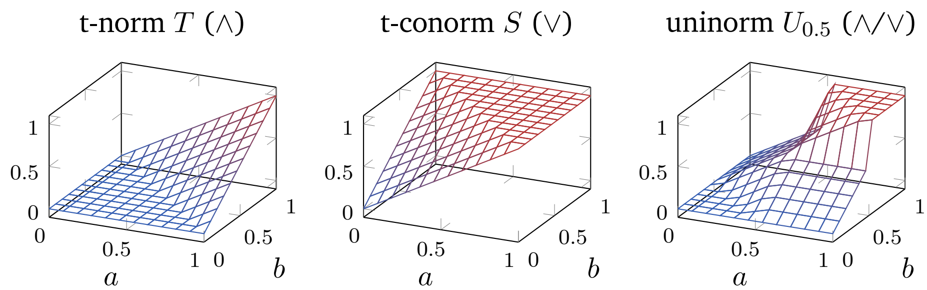

Figure 2.2. The Łukasiewicz norms and the corresponding uninorm for $\lambda = 0.5$.

Since t-norms and t-conorms are commutative and associative they can also be applied to non-empty sets of arbitrary size, i.e. $T(\{y_1, \dots, y_n\}) = T(y_1, T(\{y_2, \dots, y_n\}))$ with fixpoint $T(\{y\}) = y$. Using this extension, a uninorm U_λ can be applied to sets which turns it into a parameterized aggregation function $\mathcal{A}_\lambda : [0,1]^* \to [0,1]$. In this simple model the LTA aggregation problem boils down to the optimization of λ. The LTA disaggregation problem is solved by jointly optimizing a logistic regression model, i.e. the constituent scores $\{y_i \in [0,1]\}_{c_i \in \boldsymbol{c}}$ are described by $y_i = \left(1 + \exp(-\theta^\top x_i)\right)^{-1}$. Overall an LTA model is therefore described by the uninorm parameter λ and the regression coefficients θ.

2.1.2 OWA-Aggregation

Recently Melnikov and Hüllermeier [MH19] have looked at an alternative class of aggregation functions. Instead of using fuzzy logic to describe score aggregation, *ordered weighted average* (OWA) operators were used. OWA aggregators work by sorting the input scores and then weighting them based on their sort position, i.e.

$$\mathcal{A}_\lambda(y_1, \dots, y_n) := \sum_{i=1}^{n} \lambda_i y_{\pi(i)}, \tag{2.2}$$

where $\lambda \in \mathbb{R}^n$ is a weight vector with $\|\lambda\|_1 = 1$ and π is a sorting permutation of the input scores with $y_i < y_j \Rightarrow \pi(i) < \pi(j)$. Depending on the choice of the vector λ, the OWA function $\mathcal{A}_\lambda$ can express common aggregation functions like $\min$ (if $\lambda = (1, 0, \dots, 0)$), $\max$ (if $\lambda = (0, \dots, 0, 1)$) or the arithmetic mean (if $\lambda = \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$).

To deal with varying composition sizes n, the weights $\{\lambda_i\}_{i=1}^n$ can however not be statically assigned. Instead they are interpolated using a so-called *basic unit interval monotone* (BUM) function $q : [0,1] \to [0,1]$. It takes constituent positions that are normalized to the unit interval, i.e. $\frac{i}{n} \in [0,1]$. The BUM function q is then used to interpolate a weight for any normalized sort position via $\lambda_i := q\left(\frac{i}{n}\right) - q\left(\frac{i-1}{n}\right)$. Because q is monotone with $q(0) = 0$ and $q(1) = 1$, it always holds that $\|\lambda\|_1 = q(1) - q(0) = 1$. Using this model, the aggregation problem boils down to optimizing the shape of q.

In the OWA approach the BUM function q is modeled as a piecewise linear spline. This spline is described by $m+1$ points $\left\{\left(\frac{j}{m}, a_j\right)\right\}_{j=0}^{m}$, the so-called knots of the spline. The curve of q is obtained by linearly interpolating between neighboring knots as shown in fig. 2.3. If $0 = a_0 \leq a_1 \leq \cdots \leq a_m = 1$, q is a BUM function. The LTA aggregation problem is therefore solved by optimizing $a \in \mathbb{R}^{m+1}$ under this constraint. The disaggregation problem is tackled by adding the scores $y_1, \dots, y_M \in \mathbb{R}$ to the learnable parameters of the model where M is assumed to be the finite number of constituents. Currently the OWA approach requires all possible constituents to be part of the training dataset since it does not consider constituent features $x_i \in \mathcal{X}$ to

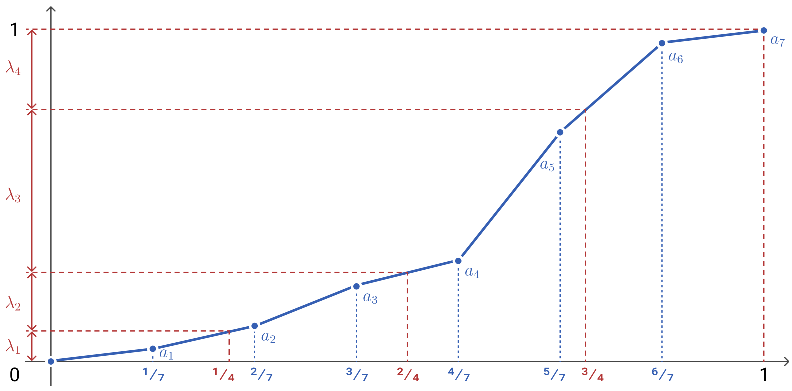

Figure 2.3. Illustration of how a describes q and its relation to λ ($n = 4$, $m = 7$).

predict the scores of previously unseen constituents.

2.2 Graph Theoretical Fundamentals

2.2.1 Graph Isomorphism Checks

2.2.2 Spectral Graph Theory

2.3 Graph Regression and Classification

The existing approaches to tackle the GC/GR problem can be categorized into three main families: 1. Explicit graph embeddings, 2. Graph kernels and 3. Graph neural networks. We will now look at the characteristics of those families and give an overview of specific methods.

2.3.1 Explicit Graph Embeddings

The basic idea of graph embedding approaches is to map a graph G to some finite vector space $\mathcal{X} = \mathbb{R}^d$. By embedding a graph any classification or regression algorithm that works with vectors can then be applied. There are three main types of explicit graph embedding approaches.

Fingerprint embeddings

The first works on graph embeddings were motivated by the study of chemical structures [AB73][WW86]. There a molecule can be interpreted as a labeled graph for which the GC/GR problem corresponds to the prediction of some chemical property, e.g. toxicity or solubility. Fingerprint embeddings try to find a fixed set of subgraphs

$S_1, \ldots, S_d$ in an input graph. The embedding of the input graph G is a binary vector $x \in \{0,1\}^d$ with $x_i = \mathbb{1}[S_i \text{ is subgraph of } G]$, where $\mathbb{1}$ denotes the indicator function. This simple approach requires a careful choice of subgraphs by domain experts but can still be competitive with the other more recent approaches we will look at next. Fingerprint embeddings are for example used in multiple state-of-the-art toxicity prediction tools like RASAR [Lue+18][Tox], ProTox [Drw+14][Ban+18b][Ban+18a] or the Toxicity Estimation Software Tools [Tes].

Unsupervised embeddings

While fingerprint embeddings use a fixed set of subgraphs to characterize a given graph, unsupervised embedding methods determine relevant substructures dynamically. One such method is `graph2vec` [Nar+17] which is based on the `doc2vec` [LM14] document embedding method from natural language processing. `graph2vec` takes a set of graphs $\mathbb{G} = \{G_1, \ldots, G_N\}$ as input and finds an embedding $\varphi : \mathbb{G} \to \mathbb{R}^d$ in which graphs that share isomorphic subgraphs are mapped closer to each other than those that do not.

2.3.2 Implicit Graph Embeddings via Kernels

2.3.3 Graph Neural Networks

Spatial GNNs

Spectral GNNs

Learning to Aggregate on Graphs

3

4 Evaluation

5 Conclusion

5.1 Review

5.2 Future Directions

Appendix A

Bibliography

[AB73] George W. Adamson and Judith A. Bush. „A method for the automatic classification of chemical structures“. In: Information Storage and Retrieval 9.10 (1973), pp. 561–568 (cit. on p. 6).

[Ban+18b] Priyanka Banerjee, Andreas O. Eckert, Anna K. Schrey, and Robert Preissner. „ProTox-II: a webserver for the prediction of toxicity of chemicals“. In: Nucleic Acids Research 46.W1 (2018), W257–W263 (cit. on p. 7).

[Drw+14] Malgorzata N. Drwal, Priyanka Banerjee, Mathias Dunkel, Martin R. Wettig, and Robert Preissner. „ProTox: a web server for the in silico prediction of rodent oral toxicity“. In: Nucleic Acids Research 42.W1 (2014), W53–W58 (cit. on p. 7).

[LM14] Quoc Le and Tomas Mikolov. „Distributed Representations of Sentences and Documents“. In: Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32. ICML'14. Beijing, China: JMLR.org, 2014, pp. II–1188–II–1196. arXiv: `1405.4053v2 [cs.CL]` (cit. on p. 7).

[Lue+18] Thomas Luechtefeld, Dan Marsh, Craig Rowlands, and Thomas Hartung. „Machine Learning of Toxicological Big Data Enables Read-Across Structure Activity Relationships (RASAR) Outperforming Animal Test Reproducibility“. In: Toxicological Sciences 165.1 (2018), pp. 198–212 (cit. on p. 7).

[MH16] Vitalik Melnikov and Eyke Hüllermeier. „Learning to Aggregate Using Uninorms“. In: Machine Learning and Knowledge Discovery in Databases. Springer International Publishing, 2016, pp. 756–771 (cit. on pp. 3, 4).

[MH19] Vitalik Melnikov and Eyke Hüllermeier. „Learning to Aggregate: Tackling the Aggregation/Disaggregation Problem for OWA“. In: Proceedings of The Eleventh Asian Conference on Machine Learning. Ed. by Wee Sun Lee and Taiji Suzuki. Vol. 101. Proceedings of Machine Learning Research. Nagoya, Japan: PMLR, 2019, pp. 1110–1125 (cit. on p. 5).

[Nar+17] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, et al. „graph2vec: Learning Distributed Representations of Graphs“. In: (July 17, 2017). arXiv: `1707.05005v1 [cs.AI]` (cit. on p. 7).

[WW86] Peter Willett and Vivienne Winterman. „A Comparison of Some Measures for the Determination of Inter-Molecular Structural Similarity Measures of Inter-Molecular Structural Similarity“. In: Quantitative Structure-Activity Relationships 5.1 (1986), pp. 18–25 (cit. on p. 6).

Websites

[Ban+18a] Priyanka Banerjee, Robert Preissner, Andreas Eckert, and Anna K. Schrey. ProTox-II - Prediction Of Toxicity Of Chemicals. Charité – Universitätsmedizin Berlin. 2018. URL: http://tox.charite.de/protox_II/ (visited on Nov. 18, 2019) (cit. on p. 7).

[Tes] Toxicity Estimation Software Tool (TEST). EPA. URL: https://www.epa.gov/chemical-research/toxicity-estimation-software-tool-test (visited on Nov. 18, 2019) (cit. on p. 7).

[Tox] ToxTrack - Cheminformatics Modeling. ToxTrack Inc. URL: https://toxtrack.com/ (visited on Nov. 18, 2019) (cit. on p. 7).

List of Figures

Erklärung zur Masterarbeit

Ich, Clemens Damke (Matrikel-Nr. 7011488), versichere, dass ich die Masterarbeit mit dem Thema *Learning to Aggregate on Structured Data* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen der Arbeit, die ich anderen Werken dem Wortlaut oder dem Sinn nach entnommen habe, wurden in jedem Fall unter Angabe der Quellen der Entlehnung kenntlich gemacht. Das Gleiche gilt auch für Tabellen, Skizzen, Zeichnungen, bildliche Darstellungen usw. Die Masterarbeit habe ich nicht, auch nicht auszugsweise, für eine andere abgeschlossene Prüfung angefertigt. Auf § 63 Abs. 5 HZG wird hingewiesen.

Paderborn, 18. November 2019

Clemens Damke